WRITEME: uma Ferramenta de Auxílio à Escrita de READMEs Baseada em Dados Abertos

Hilton Pintor Bezerra Leite
Voxar Labs - Universidade Federal de Pernambuco
hpbl@cin.ufpe.br

Lucas Figueiredo
Voxar Labs - Universidade Federal de Pernambuco
lsf@cin.ufpe.br

Veronica Teichrieb
Voxar Labs Universidade Federal de Pernambuco
vt@cin.ufpe.br

João Marcelo Teixeira
Voxar Labs - Universidade Federal de Pernambuco
jmxnt@cin.ufpe.br

***Resumo*—Repositórios de código são acompanhados de um arquivo chamado README (leia-me, em português) que deve ser lido antes do uso do software que acompanham. Pesquisas recentes identificaram que tipos de conteúdos estão presentes nesses arquivos, porém faltam ferramentas que divulguem e apliquem esses conhecimentos como forma de suporte a desenvolvedores no processo de escrita dos documentos. Neste trabalho apresentamos uma ferramenta de auxílio à escrita de READMEs, que usa dados abertos dos repositórios mais populares do GitHub para gerar recomendações de seções. Os repositórios usados como exemplo são os que mais receberam estrelas, e são de projetos da linguagem de programação escolhida pelo usuário. A ferramenta consiste de um back-end que adquire os READMEs pela API do GitHub, e classifica suas seções de acordo com o tipo de conteúdo presente. No front-end é apresentada uma interface web onde o usuário pode escolher as seções que se aplicam ao seu projeto, e fazer download do esqueleto do seu README.**

***Palavras-chave*—Repositórios de código, documentação de projetos, arquivos LEIAME**

***Abstract*—Software repositories are accompanied by a README file that should be read before using the software. Recent research has identified what types of content are present on those files, but we still lack tools that spread and apply this knowledge to support developers while writing READMEs. In this work, we present a README assist tool that uses data from popular open source repositories on GitHub to generate recommendations of sections. The repositories used as example are the ones with most stars from the programming language chosen by the user. The tool consists of a back-end that acquires READMEs through the GitHub API, and classifies their sections according to their contents. At the front-end a web interface is presented so users can choose which sections apply to their project, and download the generated README.**

***Keywords*—Code repositories, project documentation, README files**

I. Introdução

Projetos de software são tradicionalmente acompanhados de um arquivo README (leia-me, em português), cuja função é instruir usuários e colaboradores sobre o uso, desenvolvimento, manutenção e quaisquer outras informações que o autor considere relevante de serem documentadas para leitura prévia ao uso do mesmo.

Em portais de hospedagem e compartilhamento de código como o GitHub[1], o README ganha destaque ao ter seu conteúdo convertido para HyperText Markup Language (HTML), e apresentado na página inicial do projeto, ditando assim muitas vezes a primeira impressão que um desenvolvedor, ou usuário em potencial, tem do projeto.

Com mais de 100 milhões de repositórios, sendo mais de 28 milhões deles de código aberto, o GitHub se tornou um alvo de pesquisa de diversos trabalhos acadêmicos, que procuram extrair informações dos códigos e dos READMEs de sua base de dados. Tarefa que se torna viável devido à facilidade de acesso aos repositórios abertos do GitHub através de sua Application Programming Interface (API) pública.

Considerando a significativa importância dos READMEs, pesquisas recentes se propuseram a investigar os tipos de conteúdos que formam estes documentos, com o intuito de servir de referência para desenvolvedores [1]. Apesar da divulgação dos conhecimentos desenvolvidos nessa área avançar no contexto científico, o acesso e uso deste conhecimento é restrito ao meio acadêmico.

Buscando facilitar o processo de escrita de READMEs, Terrasa et al. [2] propuseram uma ferramenta que utiliza técnicas de Processamento de Linguagem Natural (PLN) para prover sugestões de estrutura, exemplos de código, e abstrações. No entanto, para atingir tal objetivo se faz necessário minerar o código do projeto e considerar suas especificidades, requerendo implementações específicas para cada linguagem de programação, motivos que neste caso levaram a solução a se limitar à linguagem de programação Nit[2].

Com intuito de auxiliar desenvolvedores na escrita dos seus READMEs, divulgando os conhecimentos adquiridos sobre a constituição desses arquivos no GitHub, e levando em conta as especificidades dos projetos escritos em cada linguagem de programação, neste trabalho foi projetada uma ferramenta de auxílio à escrita de READMEs, baseada em dados de repositórios de código aberto do GitHub.

A arquitetura da ferramenta tem caráter genérico, recebendo como entrada do usuário a linguagem de programação que utilizou no seu projeto. Porém, para realização do estudo de caso, essa funcionalidade não foi implementada, restringindo o uso da ferramenta às linguagens Python e Swift.

Neste artigo relatamos o funcionamento da ferramenta e os resultados obtidos através do seu uso por 7 desenvolvedores.

[1]https://github.com

[2]https://nitlanguage.org/

Na Seção 2 introduzimos o contexto onde o nosso trabalho está inserido, e quais são os trabalhos relacionados. Na Seção 3 apresentamos a arquitetura da ferramenta, quais são seus principais módulos, e como se relacionam. A Seção 4 relata a metodologia utilizada no estudo de caso, em conjunto com os resultados obtidos. Por fim, na Seção 5 apresentamos as conclusões retiradas da construção e teste da ferramenta, além dos trabalhos futuros.

II. Contexto

Antes de descrever a organização e funcionamento da ferramenta desenvolvida, é relevante explicar o contexto em que ela se insere, os artefatos, e os trabalhos com os quais se relaciona e utiliza como embasamento.

A. Github

No contexto do desenvolvimento de software moderno, o uso de ferramentas de controle de versão, como o Git[3], faz parte da rotina dos desenvolvedores tanto em projetos individuais quanto no desenvolvimento distribuído em equipes.

Plataformas de hospedagem e distribuição de código, como GitHub, GitLab[4], e BitBucket[5], proveem uma interface onde os repositórios de código podem ser acessados e gerenciados por todos os membros da equipe de desenvolvimento através da web.

Além do compartilhamento entre os envolvidos no desenvolvimento do software, essas plataformas possibilitam compartilhamento do repositório com terceiros que podem vir a integrar a equipe, ou a utilizar o software tanto como produto final, ou como ferramenta para construção de outros produtos, como no caso de bibliotecas e frameworks.

Tanto no contexto de empresas e laboratórios de pesquisa que produzem código proprietário quanto no de código aberto, repositórios bem documentados facilitam a utilização, reuso, extensão e compartilhamento dos softwares. Principalmente neste último contexto, projetos que servem o mesmo propósito podem ter numa boa documentação um diferencial competitivo, tornando-se mais atrativos para potenciais usuários.

O GitHub atualmente se destaca como a mais popular destas plataformas, tendo um alcance de mais de 31 milhões de desenvolvedores cadastrados, além de possibilitar acesso a desenvolvedores sem contas também.

Quando um usuário do GitHub acha um repositório interessante, ou quer parabenizar os desenvolvedores do projeto pelos seus esforços, é comum que o usuário dê uma estrela (star) a esse projeto. Dessa forma, o repositório ganha mais destaque, se tornando então esta, uma métrica que pode ser utilizada para indicar a popularidade dos repositórios.

[3]https://git-scm.com/
[4]https://gitlab.com
[5]https://bitbucket.org

B. READMEs

Arquivos README acompanham softwares com o intuito de serem lidos antes do uso dos mesmos, por conter instruções que os desenvolvedores consideram necessárias de serem obtidas para utilizar corretamente o software. Esses arquivos podem ter diferentes extensões, mas no contexto de desenvolvimento de software, geralmente são escritos na linguagem de marcação Markdown[6] (Figura 1).

O uso dessa linguagem possibilita uma organização hierárquica do documento em seções que variam do nível 1 (o mais importante) ao nível 6 (o menos importante), analogamente às tags de heading do HTML. Além de texto padrão, negrito, itálico, e sublinhado, ela também permite a incorporação de outros conteúdos como imagens, blocos de códigos, e links externos.

Ao criar um repositório Git através da interface do GitHub, juntamente com informações como o nome do mesmo, é perguntado se o usuário deseja que o repositório seja inicializado com um README, o que demonstra a importância que a plataforma atribui à existência do documento nos seus repositórios.

Outro fator que indica a relevância atribuída a READMEs pelo GitHub é o fato destes arquivos serem renderizados (convertidos do Markdown para HTML), e apresentados na página inicial do repositório, sendo então responsáveis por passar a primeira impressão do projeto de software, facilitando o cumprimento da função indicada pelo seu nome.

Segundo Lethbridge et al. [3], desenvolvedores têm dificuldades em escrever documentação, priorizando documentações simples, mas poderosas, em comparação a mais complexas. Consideramos que READMEs têm o potencial de se enquadrar nestas documentações priorizadas, por serem mais simples que wikis ou documentos mais tradicionais, e ao mesmo tempo poderosos, por possibilitar o uso do software sem instruções adicionais.

É possível encontrar diversos guias em blogs e sites que tentam guiar desenvolvedores a escrever bons READMEs. De forma geral, estes guias são descritos com base em experiências prévias de seu autor, e não necessariamente refletem as práticas mais utilizadas pela comunidade.

Além disso, os guias que se propõem a serem genéricos perdem o poder de auxiliar na escolha de conteúdos específicos para a linguagem de programação do projeto que está sendo documentado. Motivados pela falta de conhecimento científico sobre os tipos de conteúdos que constituem os READMEs, Prana et al. [1] propuseram um conjunto de categorias de conteúdos que caracterizam a população destes documentos no GitHub, além de um classificador capaz de fazer a análise das seções dos READMEs, para saber à que categoria se encaixam de forma automática. A ferramenta proposta neste projeto utiliza destes conhecimentos obtidos através de dados para instruir desenvolvedores sobre que tipo de conteúdo seus READMEs devem ter; e do classificador

[6]https://daringfireball.net/projects/markdown/

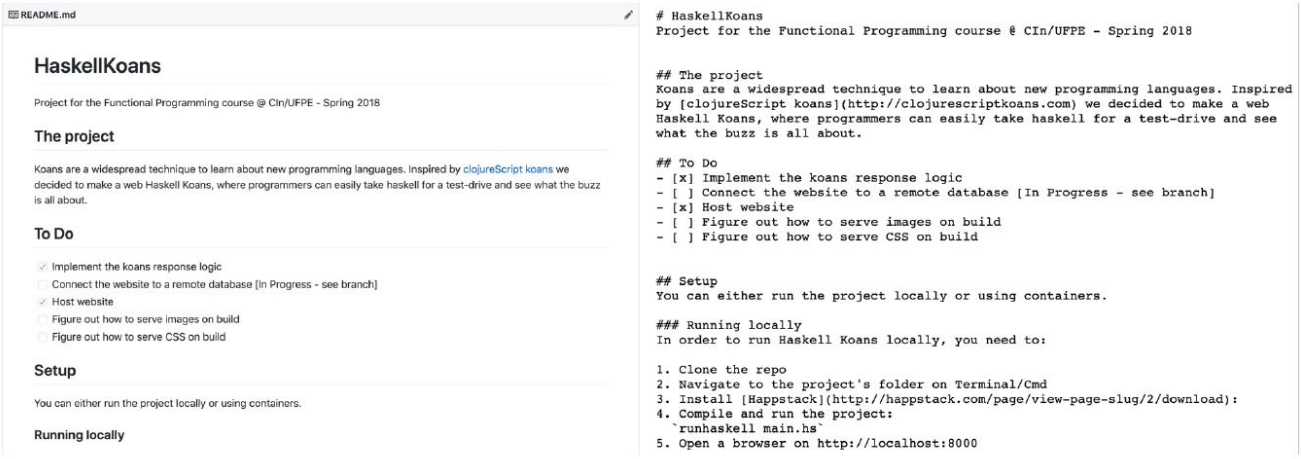

Figura 1. À esquerda um README como apresentado no GitHub, e na direita sua versão crua em Markdown.

disponibilizado para indicar quais títulos de seções geralmente são utilizados em cada categoria de conteúdo.

C. Mineração de Repositórios e READMEs

O grande número de repositórios de código aberto (mais de 28 milhões) no GitHub, combinado com a facilidade de acesso a eles, proporcionada pela API pública da plataforma, torna o ecossistema de desenvolvimento de software na mesma um objeto de estudo de crescente interesse.

Através de requisições na API é possível se formar uma base de dados para mineração e estudo, na perspectiva de Engenharia de Software, com as características desejadas, como por exemplo: repositórios populares de uma linguagem de programação específica. A seguir listaremos alguns dos trabalhos de mineração de repositórios e READMEs que inspiraram ou se relacionam com a nossa ferramenta.

Alguns desses projetos se dedicaram a tentar encontrar similaridades entre repositórios para diferentes propósitos. Zhang et al. [4] propuseram um sistema de recomendação (RepoPal) com o propósito de identificar repositórios similares no GitHub, utilizando três métricas: a similaridade do conteúdo dos READMEs, se foram favoritados por usuários de perfis similares, e se foram favoritados por um mesmo usuário num curto período de tempo. Koskela et al. [5] projetaram um método de recomendação de software de código aberto no GitHub, utilizando informações sociais do usuário que está fazendo a pesquisa, em conjunto com a linguagem de programação desejada, tópicos, e informações extraídas de um README utilizado como base.

Rahman et al. [6] realizaram dois estudos de caso sobre o efeito de diferentes medidas de similaridade em documentos relacionados ao desenvolvimento de software, como READMEs e arquivos de código fonte, com o propósito de medir suas eficácias nas tarefas de descobrir recomendações de projetos e localização de bugs.

Portugal et al. [7] propuseram um processo de mineração de READMEs e issues de repositórios do GitHub para achar possíveis requisitos que possam inspirar o processo de elicitação para aplicações de um mesmo domínio. Prana et al. [1] realizaram a primeira pesquisa com intuito de descobrir qual o conteúdo típico dos arquivos README no GitHub. Através da análise manual de uma base de repositórios obtidos aleatoriamente, foi constatado que os conteúdos presentes nas seções desses arquivos costumam ser relativos a: o quê, porque, como, quando, referências, e contribuição.

Também foi um dos objetivos da pesquisa gerar um classificador que pudesse fazer a classificação das seções de qualquer README (escrito em inglês) de forma automática. O classificador projetado obteve score F1 de 0,746, foi disponibilizado, e é utilizado na nossa ferramenta para agrupar as seções dos repositórios sendo minerados, de forma que o usuário possa entender que tipo de conteúdo deve conter nas seções.

Uma vez que foi mapeado o conteúdo típico dos READMEs do GitHub, Terrasa et al. [2] propuseram uma ferramenta para auxiliar desenvolvedores a escrever os seus READMEs através de sugestões durante o processo de escrita do documento. O protótipo reportado tem a limitação de ser específico para a linguagem de programação Nit, pois suas sugestões vem também da mineração do código do projeto sendo documentado. Também não foram apresentados resultados sobre a eficiência percebida do protótipo por usuários.

A ferramenta proposta neste trabalho se assimila à de Terrasa et al. [2] por ser uma ferramenta de auxílio à escrita de READMEs, porém toma abordagem diferente para realizar sua função, prezando por uma abordagem genérica, para funcionar com dados dinâmicos de qualquer linguagem de programação reconhecida pelo GitHub.

III. A Ferramenta

Dada a importância dos arquivos README para o desenvolvimento de software moderno, foi idealizada uma ferramenta de auxílio à escrita desses documentos, baseada

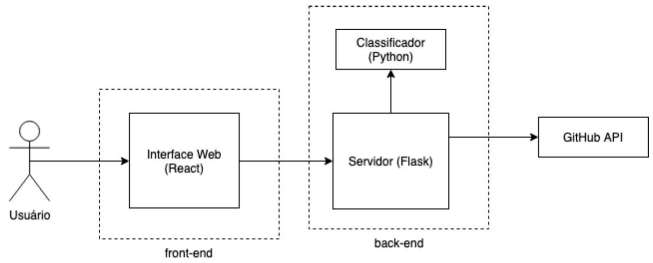

Figura 2. Arquitetura da ferramenta.

em dados abertos. Pela orientação a dados, é conferida à ferramenta um caráter dinâmico, fazendo com que ela seja mais resiliente ao desenvolvimento e evolução das linguagens de programação e seus ecossistemas. Esta característica também a protege de perpetuar um viés pessoal dos seus autores, como é comum nos guias que se propõem a ajudar na escrita de READMEs.

Projetamos a ferramenta numa arquitetura com módulos independentes (Figura 2). Escrevemos o back-end em Python, com auxílio do micro-framework Flask[7]. Ele foi adicionado num contêiner Docker[8], com o intuito de facilitar o processo de implementação e implantação em qualquer máquina, pois para executar o projeto não são necessárias dependências instaladas localmente, apenas do suporte a Docker. Escrevemos o front-end em HTML, Cascading Style Sheets (CSS), e JavaScript, por meio do framework React[9].

A. Aquisição dos READMEs

Cada linguagem de programação tem suas especificidades e um ecossistema de ferramentas complementares, o que reflete em seções específicas nos READMEs, que não fazem sentido para outras linguagens. Um exemplo são seções relacionadas a gerenciadores de dependências, onde para um projeto em Swift é comum que estejam presentes seções com o título dos gerenciadores de pacote compatíveis com a linguagem: Cocoapods[10], Carthage[11], Swift Package Manager[12].

Apresentar seções com estes títulos para projetos escritos em linguagens que não são suportadas pelos mesmos não seria útil para o usuário. Para considerar as especificidades das linguagens, mostrando apenas sugestões de seções relevantes para a utilizada no projeto sendo documentado, a linguagem de programação foi considerada como um dos critérios de filtro para saber se um repositório deve fazer parte da base dados.

Porém este critério não é suficiente, pois com ele podemos acabar adquirindo uma base de dados onde os repositórios são da linguagem desejada, mas cujos READMEs não são bem escritos, o que faz com que não sejam aptos a serem utilizados como base para sugestões de construção de um bom README.

[7]http://flask.pocoo.org/
[8]https://www.docker.com/
[9]https://pt-br.reactjs.org/
[10]https://cocoapods.org/
[11]https://github.com/Carthage/Carthage
[12]https://swift.org/package-manager/

```
[
  {
    file_id: '1',
    heading_level: 4,
    readme_file_name: 'Quick.Quick.md',
    section_codes: [
      1,
      3,
    ],
    section_id: '1',
    title: 'Nimble',
  },
  // ...
]
```

Figura 3. Recorte do array de seções classificadas.

Com o intuito de tentar minimizar a ocorrência destes repositórios indesejados na base de dados, usamos a popularidade dos mesmos, representada pela quantidade de estrelas que recebeu no GitHub, como critério de ordenação. Com esses dois critérios conseguimos consultar a API do GitHub e obter uma resposta com informações dos 100 repositórios mais famosos escritos na linguagem desejada.

Das informações recebidas, coletamos o nome dos repositórios, e para cada um deles fazemos uma nova requisição à API para obter a URL de download dos arquivos. Por fim é feita uma última requisição utilizando a URL de download, onde o conteúdo dos arquivos são recebidos como resposta e salvos localmente.

O processo de aquisição dos READMEs relatado é realizado de forma automática pelo back-end, tendo como entrada apenas a linguagem de programação desejada.

B. Classificação dos READMEs

Após a aquisição dos READMEs, é feito o processo de classificação dos arquivos, para que possamos saber que tipo de conteúdo está presente em cada seção, servindo eventualmente para agrupamento das seções de READMEs diferentes.

O classificador utilizado é de código aberto e pode ser encontrado em https://github.com/gprana/READMEClassifier/. Juntamente com o classificador, o autor disponibilizou scripts Python responsáveis pelo treinamento do classificador, pré-processamento de novos markdowns, e classificação dos arquivos processados, como consta em Prana et al. [1].

O treinamento do classificador, com o conjunto de dados disponibilizado, foi realizado, e em seguida os modelos gerados, juntamente com os scripts de pré-processamento e classificação, foram inseridos como um módulo ao back-end.

Construímos então um JSON com um array (Figura 3) cujos elementos são objetos que representam as seções classificadas. Em cada objeto consta o nome do arquivo README a quem a seção pertence, o nível da seção, seu título, e quais classificações obteve.

A fim de passar o conhecimento sobre como é disposto o aninhamento das seções para o front-end, é gerado um segundo JSON com uma representação de árvore (Figura 4) de cada README. Cada nó da árvore corresponde a uma seção, e seus filhos são as sub-seções, ou seja, seções de um nível menor que vem após uma de nível maior no README. Para geração da árvore de cada README foi utilizada a biblioteca Markdown, juntamente com a extensão toc, que é a responsável por adicionar a funcionalidade à biblioteca.

Também foi utilizada uma extensão chamada fenced_code, que faz com que no parsing do README, blocos de código que usam o carácter não sejam confundidos com uma nova seção, que em markdown é sinalizada pelo uso do mesmo caractere.

Com os dois JSONs que foram gerados, é possível que o front-end apresente os dados e permita as interações desejadas, sem ser responsável por fazer a aquisição, processamento e classificação dos dados. Embora cada módulo funcione independentemente, na versão atual da ferramenta a integração entre eles é feita de forma manual (Ver Seção 5 para detalhamentos sobre trabalhos futuros).

Para fins de validação da ferramenta, foram executadas as rotinas de geração dos dados necessários para o front-end duas vezes. Uma com a linguagem Swift como entrada, e outra com a linguagem Python. Na Seção 4 descreveremos o estudo de caso que utilizou os dados gerados.

C. Interface Web

Para que os usuários possam visualizar quais são as seções mais frequentes, e o tipo de conteúdo que é esperado nelas, foi projetada uma interface web onde o usuário pode ver as recomendações de seções e selecionar as que ele deseja incluir em seu novo README. Ao fim do uso, o usuário pode fazer o download de um arquivo README.md com as seções escolhidas e exemplos de uso das mesmas.

Antes da implementação realizamos duas etapas de prototipação. A primeira consistiu de desenhar protótipos de baixa fidelidade da interface no papel, para que fosse possível visualizar as diferentes formas gráficas que as funcionalidades desejadas poderiam assumir (Figura 5).

Após apresentar as alternativas para outros desenvolvedores e designers, foi selecionada a que indicava melhor usabilidade, e em sequência transformada em um protótipo de alta fidelidade para guiar a implementação do front-end.

O front-end foi implementado utilizando HTML, CSS, e JavaScript, por meio do framework React. Para facilitar a criação e configuração do projeto inicial, foi utilizada a ferramenta create-react-app. Foi feito uso do linter eslint[13] para manter um estilo de código coeso, reforçado com um pre-commit hook que só aceita os commits se os arquivos modificados estiverem sem nenhuma falha de linting[14].


```
{
    "DeclarativeHub.Bond.md": [
        {
          "children": [
            {
              "children": [],
              "id": "requirements",
              "level": 2,
              "name": "Requirements"
            },
            {
              "children": [
                {
                  "children": [],
                  "id": "carthage",
                  "level": 3,
                  "name": "Carthage"
                },
                {
                  "children": [],
                  "id": "accio",
                  "level": 3,
                  "name": "Accio"
                },
                {
                  "children": [],
                  "id": "cocoapods",
                  "level": 3,
                  "name": "CocoaPods"
                }
              ],
              "id": "installation",
              "level": 2,
              "name": "Installation"
            },
          ],
          "id": "bond-swift-bond",
          "level": 1,
          "name": "Bond, Swift Bond"
        },
        // ...
    ],
    // ...
}
```


Figura 4. Recorte do JSON de seções no formato de árvore.

Ao acessar a ferramenta, o usuário é recebido com um texto introdutório que explica a motivação e propósito da ferramenta. Em seguida são explicados os tipos de conteúdos encontrados nos READMEs [1], para que ele saiba o que seu README deve conter (Figura 6). Por fim, é feita a escolha da linguagem de programação.

Uma vez escolhida a linguagem de programação, o usuário é redirecionado para a tela de composição do seu README (Figura 7). Do lado esquerdo são dispostas as sugestões,

[13]https://eslint.org/

[14]Linting is the process of running a program that will analyse code for potential errors.

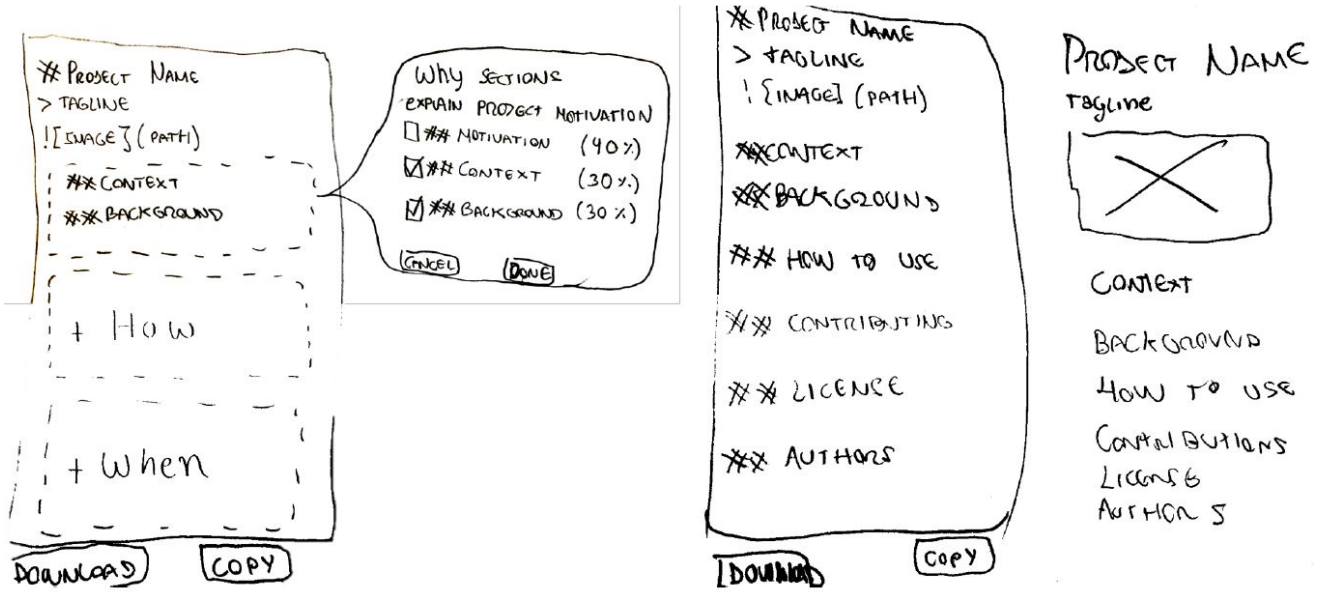

Figura 5. Protótipos de papel da interface web.

que consistem das seções utilizadas pelos repositórios mais famosos da linguagem, agrupadas pela classificação que obtiveram, e ordenadas pela frequência em que ocorrem na base de dados.

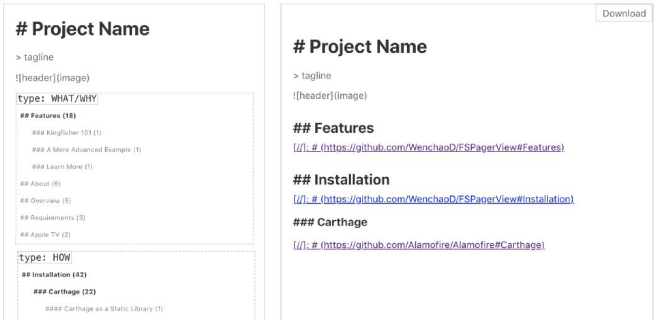

Figura 7. Recorte da tela de composição de um README em Swift. Ao lado esquerdo as sugestões de seções. A selecionadas (negrito) são adicionadas à seção de visualização, com exemplos de uso, no lado direito.

Ao selecionar uma sugestão de seção, ela é adicionada ao lado direito, onde o usuário pode visualizar as escolhas que fez até o momento, e tem a possibilidade de apertar no botão de download para salvar localmente o arquivo README.md gerado. Para cada seção utilizada, é provido um exemplo de uso da mesma, no formato de um hiperlink para algum dos READMEs que possua tal seção na sua estrutura. O hiperlink serve como referência para o usuário, sem atrapalhar a renderização do seu arquivo gerado, pois são adicionados como comentários, e portanto ignorados no processo de renderização.

Caso uma seção selecionada apresente subseções, estas são mostradas de maneira similar com sua frequência, podendo também serem escolhidas para fazer parte do README gerado. As subseções podem ter elas próprias outras subseções que são mostradas e podem ser selecionadas da mesma forma até que não haja mais subseções.

Por fim, é esperado que uma vez escolhida quais seções vão fazer parte do README, o usuário faça download do arquivo gerado, acrescente o conteúdo das seções, e faça as modificações que achar necessária para documentar o seu projeto, encerrando o processo de co-autoria fora da ferramenta.

IV. Estudo de Caso

Foi realizado um estudo de caso, com intuito de validar a ferramenta na perspectiva da percepção de utilidade pelos usuários. Neste capítulo descreveremos a metodologia utilizada no experimento, e os resultados que obtivemos.

A. Experimento

Como a ferramenta se propõe a dar auxílio à escrita de READMEs, projetamos um experimento no qual os usuários compararam um README que escreveram com apoio da ferramenta a um que havia sido previamente escrito pelos próprios. O objetivo do experimento consistiu em descobrir se existe uma utilidade perceptível na ferramenta, facilitando o processo de escrita de READMEs de qualidade, na perspectiva dos usuários.

O experimento consistiu de 4 etapas. Na primeira etapa pedimos que o usuário escolhesse qual linguagem de programação iria utilizar (Swift, ou Python), e em seguida submetesse um README que tivesse escrito para um projeto na linguagem selecionada.

A segunda etapa teve como objetivo entender quais fontes foram utilizadas como referência na composição do documento, se o usuário considera como sendo um bom

Tabela I
RESPOSTAS À PERGUNTA: QUAIS FONTES VOCÊ UTILIZOU PARA SABER QUE TIPO DE CONTEÚDO DEVERIA CONSTAR NESTE README?

Respostas	**Porcentagem**
Projetos famosos no GitHub (com a mesma linguagem)	71,4
Projetos famosos no GitHub (de outra linguagem)	42,9
Projetos próprios anteriores	28,6
Blogs e guias online	28,6
Não utilizei nenhuma fonte	14,3
Artigos científicos	0

Tabela II
RESPOSTAS À PERGUNTA: VOCÊ CONSIDERA ESSE COMO SENDO UM BOM README?

Linguagem	**Sim**	**Não**
Swift	1	3
Python	2	1
Total	3	4

README, e qual importância ele atribui à ter um bom README para o sucesso de um projeto.

Na etapa seguinte, pedimos ao usuário que utilizasse a nossa ferramenta para construção de um novo README para o projeto que escolheu. O instruímos a considerar apenas as sugestões que julgassem fazer sentido para o projeto sendo documentado. Ao fim da composição, o novo documento deveria ser submetido.

Por fim, a última etapa consistiu em descobrir se ocorreu um processo de co-autoria com a ferramenta, e em medir a percepção dos usuários no que diz respeito à capacidade da ferramenta de auxiliar o processo de escrita de um bom README.

As 4 partes do experimento foram compiladas em um formulário contendo 10 perguntas, que foi enviado para que os usuários respondessem online. O acesso à ferramenta foi feita através do domínio do repositório do projeto no GitHub Pages: https://hpbl.github.io/WRITEME.

B. Resultados

No total 7, usuários participaram do experimento respondendo ao questionário, sendo 3 deles em Python, e 4 em Swift. Os resultados obtidos e as análises que derivamos deles são relatadas nesta seção.

Observamos que a maioria dos usuários utiliza repositórios famosos no GitHub como inspiração (Tabela I). Nossa ferramenta utiliza da mesma fonte para dar recomendações, compilando as informações contidas nos 100 repositórios mais populares da linguagem escolhida.

Dois dos três usuários que realizaram o experimento em Python consideram que seu README é bom, enquanto três dos quatro que fizeram em Swift, não consideram seus READMEs como bons (Tabela II). Todos os usuários consideram que um bom README é de grande ou extrema importância para o sucesso de um projeto (Tabela III).

Após o uso da ferramenta para a construção de um novo README, 57,1% dos usuários relataram que todas as seções do novo README vieram de sugestões da ferramenta, e o restante combinou seções de sua autoria com sugestões da ferramenta (Figura 8).

Tabela III
RESPOSTAS À PERGUNTA: O QUÃO IMPORTANTE VOCÊ CONSIDERA QUE É TER UM BOM README PARA O SUCESSO DE UM PROJETO?

Resposta	**Porcentagem**
5 (Extrema importância)	71,4
4	28,6
3	0
2	0
1 (Nenhuma importância)	0

Figura 8. Respostas à pergunta: Neste novo README, de onde vieram as seções utilizadas?

Repetimos a pergunta feita anteriormente sobre a qualidade do README, dessa vez referente ao novo arquivo gerado com auxílio da ferramenta (Figura 9). Percebemos que agora 3 dos 4 usuários, que consideravam seu README como não sendo bom, mudaram de opinião. O usuário que achou que ambos os seus READMEs não são bons, foi também o único que não achou que o documento foi melhorado após o uso da ferramenta (Figura 10).

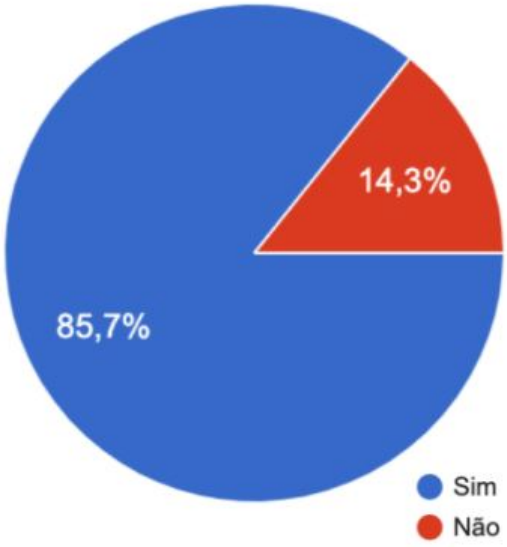

Figura 9. Respostas à pergunta: Você considera esse novo README como sendo um bom README?

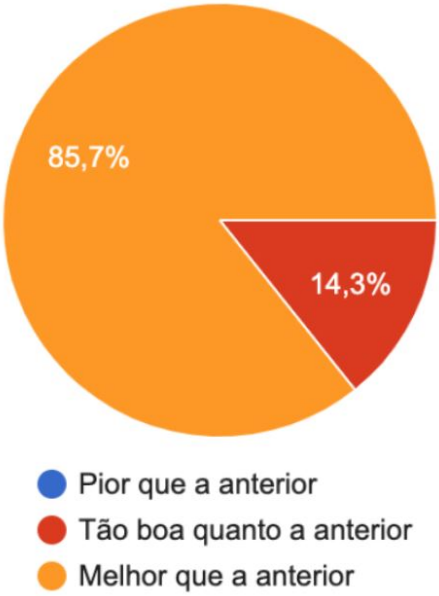

Figura 10. Respostas à pergunta: Em comparação com o README anterior, essa nova versão é:

Por fim perguntamos se os usuários teriam interesse em utilizar a ferramenta para ajudar a construir o README de um novo projeto. Todos os 7 entrevistados responderam que sim.

V. Conclusões e Trabalhos Futuros

Apresentamos neste relatório o projeto de uma ferramenta de auxílio à escrita de READMEs baseada em dados. Detalhamos sua arquitetura e funcionamento. Também foi realizado um estudo de caso com 7 desenvolvedores, a fim de avaliar o uso da ferramenta.

Devido ao baixo número de entrevistados no estudo de caso, não podemos atribuir valor estatístico aos resultados do experimento, porém temos um indicativo de que a ferramenta ajuda os desenvolvedores a escrever READMEs que eles consideram de boa qualidade, visto que os usuários fizeram uso das recomendações dadas pela ferramenta, e perceberam melhoria na qualidade dos seus READMEs.

Também o consenso que gostariam de usar a ferramenta novamente, entre os entrevistados, indica uma aceitação e percepção de valor na ferramenta. A arquitetura que propomos deve funcionar para qualquer linguagem de programação suportada pelo GitHub, atingindo assim o caráter genérico que era um dos objetivos deste projeto.

Com essa visão em mente, esse será o primeiro trabalho futuro: integrar os módulos para que a ferramenta possa funcionar, de forma autônoma, para qualquer linguagem que o usuário deseje.

Após finalização da ferramenta, novos testes com usuários devem ser realizados para medir a utilidade da mesma de forma estatisticamente significativa. Também é desejável desenvolver análise dos READMEs gerados com auxílio da mesma, para avaliar a qualidade dos arquivos independentemente da percepção dos usuários.

No uso da ferramenta, é perceptível que existem entre as recomendações seções que são específicas aos projetos que a utilizaram, e não devem servir para o usuário. Aumentar a quantidade de repositórios utilizados como referência, atualmente 100, e aumentar o número mínimo de ocorrências, atualmente 2, fará com que a ferramenta mostre menos casos específicos, e dê mais destaque às seções que se repetem com maior frequência na base de dados.

Referências

[1] G. A. A. Prana, C. Treude, F. Thung, T. Atapattu, and D. Lo, "Categorizing the content of github readme files," *Empirical Software Engineering*, vol. 24, no. 3, pp. 1296–1327, 2019.

[2] A. Terrasa, J. Privat, and G. Tremblay, "Using natural language processing for documentation assist," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[3] T. C. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: The state of the practice," *IEEE software*, vol. 20, no. 6, pp. 35–39, 2003.

[4] Y. Zhang, D. Lo, P. S. Kochhar, X. Xia, Q. Li, and J. Sun, "Detecting similar repositories on github," in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2017, pp. 13–23.

[5] M. Koskela, I. Simola, and K. Stefanidis, "Open source software recommendations using github," in *International Conference on Theory and Practice of Digital Libraries*. Springer, 2018, pp. 279–285.

[6] M. M. Rahman, S. Chakraborty, G. Kaiser, and B. Ray, "A case study on the impact of similarity measure on information retrieval based software engineering tasks," *arXiv preprint arXiv:1808.02911*, 2018.

[7] R. L. Q. Portugal, M. A. Casanova, T. Li, and J. C. S. do Prado Leite, "Gh4re: Repository recommendation on github for requirements elicitation reuse." in *CAiSE-Forum-DC*, 2017, pp. 113–120.

Data Driven README

Software is meant to be used, and the first step towards making your project usable is having a good README. If you're here that probably means you already know that, but what you might not know is what should be included in your README?

There are several opinionated guides about how to write a README. However, with an ever-growing number of programming languages, tools, and processes, the idea of having one definitive generic guide is bound to become outdated, as it does not take into consideration the specific details and nuances of projects built on different languages, for different systems.

This tool will not write a README for you, but it will give you suggestions of what content should be included in your document, according to which programming language you're using. All suggestions are derived from live data from the most popular GitHub repositories and scientific research.

What should be in your README?

A study conducted in 2018[1] answered this question by querying GitHub repos and analyzing the content of each section from their READMEs. They found that the content of those sections can be classified in the following categories:

1. **WHAT:** An introduction on what your project does.
2. **WHY:**The motivation behind your project, it's advantages.
3. **HOW:** Instructions on how to use the project.
4. **WHEN**The status of the project, it's versions and roadmap.
5. **WHO:** The people responsible for the project, license information, code of conduct.
6. **REFERENCES:** External documentation, support, and related projects.
7. **CONTRIBUTION:** Instructions on how to contribute to the project (sometimes a stand-alone file).
8. **OTHER:** Any type of content that does not fit any of the above categories.

Figura 6. Tela inicial com explicação da motivação da ferramenta e dos tipos de conteúdo presente nos READMEs.