Minerando Repositórios de Software relacionados com COVID-19 no GitHub

Rodrigo Meira Neves[1], **Crescencio Lima (orientador)**[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)
45078-300 – Vitória da Conquista – BA – Brasil

rmeiraneves@gmail.com, crescencio@gmail.com

Abstract. *Several projects have been developed by programmers worldwide during the SARS-CoV-2 (COVID-19) pandemic to help combat the disease. These include web systems and applications designed to inform the public about the risks of the disease, the number of cases and deaths during the pandemic, and the main ways to prevent contagion and the spread of the virus. Additionally, supercomputer-powered systems have been used for genetic analysis of the new coronavirus, contributing to the development of vaccines and potential treatments to fight the disease. This study aims to analyze software repositories related to COVID-19 on the GitHub source code hosting platform. Currently, there are more than 130,000 repositories on COVID-19 created by developers and companies worldwide. In this research, we will use the PyDriller library to mine these repositories to identify important information about the projects. Furthermore, at least five projects will be selected for in-depth analysis, providing details on the technologies, programming languages, libraries, and development practices used. This information will be presented through a simple, user-friendly, and educational interface, accessible online to any user with an internet connection. In this way, any individual or programmer interested in reproducing the analysis can rely on and be inspired by the thousands of software repositories already developed, making it easier to read, analyze, and select projects, especially those related to health and personal care.*

.

Resumo. *Vários projetos foram desenvolvidos por programadores de todo o mundo durante o período da pandemia do vírus SARS-CoV-2 (COVID-19) para auxiliar no combate da doença. Entre eles podemos mencionar sistemas web e aplicativos propostos a informar à população sobre os riscos da doença, os números de casos e mortes da pandemia e as principais formas para evitar o contágio e proliferação do vírus, como também sistemas utilizados por super computadores para análise genética do novo coronavírus e suas contribuições no desenvolvimento de vacinas e possíveis fármacos para combate à doença. Logo, este trabalho tem como objetivo analisar repositórios de software relacionados com a COVID-19 na plataforma de hospedagem de código-fonte e arquivos GitHub. Atualmente são mais de cento e trinta mil repositórios sobre a COVID-19 feitos por desenvolvedores e empresas de todo o mundo. Neste trabalho, utilizamos a biblioteca PyDriller capaz de minerar esses repositórios de softwares, com objetivo de identificar informações importantes sobre os projetos. Cinco projetos foram selecionados para análise mais aprofundada, informando tecnologias, linguagens de programação, bibliotecas e práticas de*

programação que foram utilizadas. Tais informações foram apresentadas por meio de uma interface simples, amigável e didática, que poderá ser acessada virtualmente por qualquer usuário conectado a internet. Assim, qualquer pessoa ou programador que se interessar em reproduzir a análise, poderá se basear e se inspirar nos milhares de repositórios de softwares já desenvolvidos e que estarão de fácil acesso para leitura, análise e escolha, principalmente se o projeto tiver relação à saúde e cuidados pessoais.

1. Introdução

O desenvolvimento exponencial das tecnologias de software vem promovendo grandes mudanças na sociedade em geral. Entre elas está a capacidade de resolução de problemas antes difíceis de serem solucionados, principalmente em um curto período de tempo. Este fenômeno tem se tornado cada vez mais evidente e vem sendo desenvolvido por diversos estudiosos de todo o globo, muitos deles de forma colaborativa.

O vírus SARS-CoV-2, mais conhecido como COVID-19, provocou situações fora do comum comparado a outras situações que já ocorreram na humanidade, nunca presenciado por grande parte da população mundial. Esse evento pode ser comparado, no último século, apenas aos registros da época de guerras mundiais e da ocorrência da gripe espanhola.

Nos últimos anos, a codificação social tornou-se uma importante abordagem colaborativa para o desenvolvimento de software de código aberto. A codificação social é uma abordagem de desenvolvimento de software que enfatiza a colaboração formal e informal, permitindo uma experiência diferente de desenvolvimento de software, onde as atividades e os interesses de um desenvolvedor são facilmente visualizados por outros desenvolvedores (Thung, 2013).

As empresas, instituições e poderes públicos que conseguiram manter suas atividades em meio ao isolamento social tiveram como instrumento principal a tecnologia. Entre as possibilidades de ferramentas tecnológicas visando auxiliar o combate à COVID-19, podemos citar a plataforma GitHub. A rede social GitHub contém cerca de 24 milhões de usuários cadastrados e hospeda mais de 83 milhões de repositórios (GitHubArchive7: Ano 2019). Para a codificação social, projetos de software também são chamados de repositórios ou, simplesmente, projetos. O repositório pode conter um conjunto de artefatos principais (por exemplo, documentos de texto, código-fonte, imagens), além de outras fontes de dados associadas a eles, como as características sobre o seu usuário proprietário.

Diante desse contexto, este artigo propôs analisar um protótipo de mineração de repositórios de software relacionados à COVID-19 na plataforma GitHub, utilizando a biblioteca PyDriller integrada à plataforma Google Colab. O sistema foi criado para lidar com grandes volumes de dados, auxiliando pesquisadores e desenvolvedores na análise e seleção de repositórios para futuras implementações ou contribuições. Para isso, foram definidos objetivos específicos, incluindo a análise do processo de descoberta de repositórios, a extração de informações relevantes sobre os projetos e o desenvolvimento de um protótipo capaz de examinar ao menos cinco repositórios, identificando tecnologias utilizadas, linguagens de programação e práticas adotadas, entre outros dados.

O restante deste artigo está estruturado da seguinte forma: Seção 2 apresenta o

referencial teórico da pesquisa. Seção 3 descreve a infraestrutura e as tecnologias utilizadas para o desenvolvimento do projeto. Seção 4 detalha a proposta desenvolvida e as metodologias adotadas . Seção 5 apresenta os resultados esperados . Seção 6 discute as conclusões da pesquisa e sugere trabalhos futuros.

2. Fundamentação Teórica

2.1. A Pandemia da COVID-19

Com a popularização da Internet e o surgimento de outras inovações tecnológicas, os níveis de produção industrial no mundo cresceram, visando atender uma demanda com um nível de exigência cada vez maior em um ambiente cada vez mais competitivo (Santos, Alberto, Lima e Charrua-Santos, 2018).

A pandemia do COVID-19 nasceu digital, pois surgiu na Província de Wuhan e logo chegou a Shenzhen, dois pólos de tecnologia da China. Em seguida, o vírus se alastrou para a Coreia do Sul e Japão, norte da Itália, Califórnia, onde fica o *Silicon Valley*, e o estado norte-americano de Washington, sede da Microsoft e da Amazon. Todas essas regiões estão fortemente interligadas pela tecnologia, com intensa movimentação de pessoas, informação, serviços e produtos entre elas (www. fundacaofhc.org.br, 2020).

Nas epidemias passadas, podemos encontrar uma diferença significativa em relação à pandemia da COVID-19, que é o fato de naquela época não se ter tecnologias avançadas ou mecanismos de desenvolvimento de sistemas que pudessem ajudar no combate do vírus. Recentemente, contamos com diversas plataformas de desenvolvimento, programadores e empresas voltadas para a criação de soluções através da tecnologia, entre elas, sistemas e aplicativos capazes de auxiliar no combate à pandemia. Tudo graças à evolução dos processos de desenvolvimento de software.

2.2. Processos de Desenvolvimento de Software

Um processo de software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software (Fuggetta, 2000). Um processo de software bem definido deve indicar as atividades a serem executadas, os recursos requeridos, os artefatos consumidos e produzidos e os procedimentos a serem adotados (métodos, técnicas, modelos de documentos, entre outros).

A área de processos de software, como uma disciplina autônoma, é relativamente nova, tendo surgido em meados da década de 80. Ao longo desses pouco mais de 20 anos, vários esforços vêm sendo realizados, com destaque para a criação de padrões de qualidade de processo (Fuggetta, 2000), tais como a norma ISO/IEC 12207 (ISO, 1995) e modelos de maturidade como MPS.BR e CMMI (Chrissis et al., 2003).

Com a evolução dos processos de software e dos ambientes de desenvolvimento, tivemos vários avanços recentes em várias áreas tecnológicas que possibilitaram um crescimento explosivo na capacidade de gerar, coletar, armazenar e transmitir dados digitais. Na primeira década do século 21 já temos a possibilidade de armazenar vários gigabytes em dispositivos portáteis e alguns terabytes em computadores pessoais a um custo acessível. Uma quantidade quase incomensurável de informações de diversos tipos, origens, formatos e finalidades estão disponíveis na Internet, podendo ser acessadas a partir destes dispositivos comuns.

O baixo custo dos dispositivos e do acesso a redes de computadores fez também com que o número de usuários destes sistemas aumentasse consideravelmente. Novas ferramentas permitem que estes usuários criem conteúdo digital de forma relativamente simples e barata, o que só faz aumentar a quantidade de informações disponíveis para outros usuários.

É importante então ter ferramentas que possibilitem a procura de informação entre esta avalanche de dados. A distinção entre dado e informação é sutil mas importante: dados podem ser coletados de forma rápida, simples e automática, e armazenados em grande volume a baixo custo; informações são de nível semântico mais alto. De forma simplista podemos considerar texto como sendo dados, e o conteúdo deste texto como sendo informações.

2.3. Mineração de Dados

Mineração de dados é o nome dado ao conjunto de técnicas que possibilita o aprendizado prático de padrões a partir de dados, possibilitando explicações sobre a natureza destes dados e previsões a partir dos padrões encontrados (Ian, 2005). De acordo com Jackson (1986), existem duas categorias principais de mineração de dados: preditiva, que envolve o uso de atributos do conjunto de dados para prever valores desconhecidos ou futuros para um conjunto de dados relacionado; e descritiva, que foca na descoberta de padrões que descrevem os dados e que podem ser interpretados por humanos. Ambas categorias podem envolver a criação de um modelo que descreve os dados e podem ser usadas para produzir mais informações sobre os dados analisados.

Informações podem ser obtidas a partir de dados através de técnicas de interpretação, anotação, classificação, agrupamento, sumarização, etc. destes dados ou de técnicas que permitam a associação e correlação de outras informações (possivelmente de outras fontes). Várias destas técnicas fazem parte do conjunto de técnicas, ferramentas, procedimentos e algoritmos conhecidos comumente como Mineração de Dados (Data Mining), que por sua vez faz parte de um processo conhecido como Descoberta de Conhecimento em Bancos de Dados (KDD, *Knowledge Discovery in Databases*).

De acordo com (Usama, 1996), podemos enumerar os passos do processo de descoberta de conhecimentos com a lista a seguir. Os passos da lista correspondentes às etapas do processo mostrado na Figura 2.1.

1. Compreensão do domínio da aplicação, do conhecimento prévio relevante e dos objetivos do usuário final do processo;
2. Criação de um conjunto de dados para uso no processo de descoberta através da seleção dos dados e/ou atributos relevantes (seleção);
3. Limpeza e pré-processamento dos dados, remoção de ruído e desvios (se possível e apropriado), decisão de como proceder com dados com atributos incompletos, normalização ou indexação de sequências temporais, extração de atributos numéricos de documentos e logs, etc. (pré-processamento);
4. Redução e reprojeção dos dados em outro conjunto de coordenadas, se necessário. Isto pode ser feito através da seleção de atributos úteis ou relevantes para representar adequadamente os dados sem perda de precisão, sempre dependendo do objetivo a ser alcançado. Se possível e desejável, mudar a representação dos dados para que a mesma seja invariante a aspectos que são relevantes (ex. escala, orientação) (transformação);

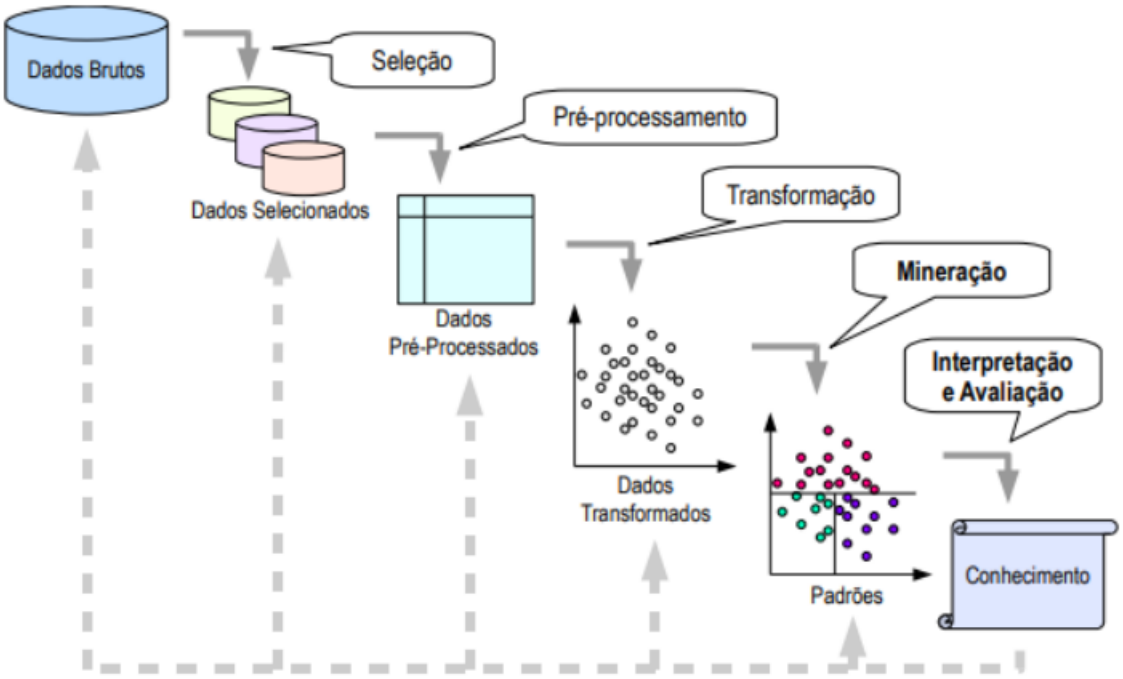


Figura 1. Processo de Descoberta de Conhecimento em Bancos de Dados (Usama, 1996).

5. Escolha da tarefa de mineração de dados, considerando o objetivo genérico do processo (classificação, regressão, agrupamento, etc.);
6. Escolha do(s) algoritmo(s) de mineração de dados, baseado no objetivo geral e na consequente estrutura imposta aos dados. A decisão do(s) algoritmo(s) envolve a escolha de modelos, parâmetros, formas de execução, etc;
7. Mineração dos dados em si: busca de padrões de interesse usando algoritmos e dados selecionados; (mineração)
8. Interpretação dos resultados da mineração de dados, inclusive avaliação dos padrões, regras, etc. encontrados pelo processo de mineração; (interpretação e avaliação)
9. Consolidação e avaliação dos conhecimentos obtidos, documentação e elaboração de relatórios, resolução de conflitos com conhecimentos previamente existentes.

Os passos do processo ilustrado pela Figura 1 não precisam necessariamente ser seguidos na ordem descrita: a descoberta de conhecimentos em bancos de dados é um processo iterativo e exploratório; portanto alguns de seus passos podem ser executados novamente dependendo do resultado de passos posteriores. É importante ressaltar também o papel da visualização no processo: técnicas de visualização podem ser usadas em vários passos do processo para a tomada de decisão sobre atributos, dados e algoritmos a ser usados.

O passo do processo que nos interessa é justamente o da mineração de dados, embora seja imperativo dominar os passos intermediários pois estes influenciam diretamente no resultado do processo de mineração, em particular no caso de dados da Web.

2.4. Python

Existem diversas tecnologias que auxiliam no processo de mineração e exploração desses dados, entre elas, a linguagem de programação Python. Python é uma linguagem de programação muito utilizada em atividades de análise de dados, sendo também uma linguagem de propósito geral, ou seja, não é exclusiva para atividades de análise de dados. Esta característica tem contribuído de forma significativa para aumentar o seu uso dentro das empresas, uma vez que muitas equipes de desenvolvimento de software já possuem um certo conhecimento com Python, o que facilita a implantação em produção de modelos desenvolvido na mesma plataforma (Lopes, 2019).

Para o desenvolvimento de aplicações científicas e que não requer configuração para ser usada, podemos citar a ferramenta desenvolvida pelo Google chamada Colaboratory. Para usar este ambiente, é necessário apenas ter uma conta gmail, pois todo notebook ficará armazenado no Drive. O código é executado em uma máquina virtual dedicada à sua conta. As máquinas virtuais são recicladas após um determinado tempo ocioso, ou caso a janela seja fechada.

As características mais relevantes do Python são a sua capacidade de integração com outras linguagens e seu sistema de bibliotecas bem maduro. As bibliotecas apresentadas a seguir são fortemente analíticas e oferecerão uma completa caixa de ferramentas de ciência de dados composta de funções altamente otimizadas para trabalhar, configuração ideal de memória, pronta para realizar operações de script com desempenho ideal (Boschetti e Massaron 2015).

Para que aconteça a exploração de dados por meio do Google Colaboratory, através da linguagem de programação Python, mais especificamente pela biblioteca PyDriller, que possui funções dedicadas a análise de projetos desenvolvidos através da ferramenta de controle de versão Git e que faz mineração em repositórios de software, sendo possível extrair informações e exportar rapidamente para arquivos CSV, é necessário que haja uma plataforma que hospeda esses software, como por exemplo o GitHub.

2.5. GitHub

O GitHub é uma plataforma de enormes proporções quando se trata do número de projetos hospedados, se aproximando de 38 milhões no começo de agosto de 2021, e igualmente enorme número de usuários, 63 milhões (GitHub. 2021). Por causa dessa magnitude de projetos existentes na plataforma, costuma ser necessário definir algum tipo de filtro a ponto de viabilizar essa mineração. Um filtro comum costuma ser o número de ”estrelas” que um projeto possui (Xu, 2020), métrica geralmente vista como popularidade de um repositório na plataforma. Entretanto, alguns trabalhos chamam atenção para como essa métrica pode ser mal interpretada e enviesar os dados caso mal utilizada (Borges, 2018).

No contexto do GitHub, um *commit* representa uma alteração registrada no código-fonte de um projeto, funcionando como um ponto no histórico do repositório e permitindo que mudanças sejam acompanhadas e revertidas se necessário. Já uma *branch* (ramificação) é uma linha independente de desenvolvimento, possibilitando que diferentes funcionalidades ou correções sejam implementadas sem afetar a versão principal do código. Quando uma *branch* atinge um estágio estável, ela pode ser mesclada (*merge*) à ramificação principal, garantindo que as alterações sejam incorporadas ao projeto. Por outro lado, um *fork* ocorre quando um desenvolvedor cria uma cópia independente de um

repositório existente, permitindo modificações sem alterar o projeto original. Esse mecanismo é amplamente utilizado para contribuições em projetos de código aberto, pois possibilita que desenvolvedores experimentem novas ideias e, posteriormente, enviem sugestões de melhorias por meio de *pull requests*.

3. Infraestrutura e Tecnologias

Para a realização deste estudo, foram utilizadas tecnologias bem estabelecidas que permitem a mineração de dados de repositórios de software hospedados no GitHub, relacionados à COVID-19. Abaixo detalhamos os principais componentes tecnológicos e sua relevância no desenvolvimento deste trabalho.

3.1. Google Colab

O Google Colab foi escolhido como ambiente de desenvolvimento devido à sua capacidade de fornecer infraestrutura de computação gratuita e acessível diretamente via navegador. Essa plataforma suporta Python nativamente, o que facilita o uso de bibliotecas específicas para análise e mineração de dados. Além disso, o Colab oferece suporte integrado ao Google Drive, permitindo salvar resultados de maneira persistente e compartilhar notebooks entre diferentes colaboradores do projeto.

Outra vantagem do Colab é a ausência de necessidade de configuração complexa. O ambiente já possui pacotes essenciais pré-instalados e possibilita a instalação de bibliotecas adicionais de forma simples por meio de comandos como `!pip install`. Isso permitiu que as bibliotecas PyDriller e PyGithub fossem rapidamente configuradas, reduzindo o tempo necessário para a preparação do ambiente.

3.2. Linguagem de Programação Python

Python foi selecionado como a principal linguagem de programação devido à sua versatilidade, facilidade de aprendizado e vasta comunidade de desenvolvedores. Python é amplamente utilizado para tarefas relacionadas à análise de dados, mineração de software e integração com *Application Programming Interface* (API). Além disso, sua sintaxe legível permitiu que o código fosse mantido claro e modular, facilitando futuras melhorias e reuso do código em diferentes contextos.

A capacidade de Python de lidar com formatos de dados como JSON e suas bibliotecas nativas para manipulação de datas e strings foram fundamentais para processar os dados obtidos da API do GitHub e apresentá-los de forma legível.

3.3. Biblioteca PyDriller

A PyDriller desempenhou um papel central neste trabalho, ao oferecer uma interface amigável para mineração de dados em repositórios Git. Essa biblioteca simplifica o acesso a informações como:

- Histórico de *commits*;
- Informações detalhadas sobre *branches* e contribuidores;
- Alterações realizadas em arquivos específicos.

A PyDriller foi utilizada para extrair dados históricos de repositórios com grande volume de *commits*, permitindo identificar tendências e avaliar a evolução

dos projetos. Por exemplo, foi possível coletar métricas detalhadas como o número de *commits* realizados ao longo do tempo, o que reflete a atividade de desenvolvimento de cada repositório. Confira a documentação oficial da biblioteca Pydriller: https://pydriller.readthedocs.io/en/latest/.

3.4. Biblioteca PyGithub

A PyGithub foi utilizada para realizar consultas avançadas diretamente à API do GitHub. Com ela, foi possível implementar filtros de busca sofisticados, como o termo ”COVID”aplicado a nomes e descrições de repositórios, e classificar os resultados com base no número de estrelas. A biblioteca também permitiu a extração de informações como:

- Linguagens de programação utilizadas nos projetos;
- Tópicos associados a cada repositório (tags);
- Métricas de popularidade (número de estrelas, *forks* e *watchers*);
- Data da última atualização.

Adicionalmente, a PyGithub possibilitou acessar a lista de colaboradores de cada projeto, permitindo a análise da dimensão colaborativa dos repositórios relacionados à COVID-19. Essa funcionalidade foi essencial para entender o grau de engajamento da comunidade em projetos abertos.

3.5. API do GitHub

A API oficial do GitHub serviu como *backend* principal para a coleta de dados. Utilizando um token de autenticação, o projeto garantiu acesso seguro e confiável às informações dos repositórios. A API oferece um conjunto amplo de *endpoints* que permitiram obter não apenas os metadados dos projetos, mas também realizar buscas refinadas com alta performance.

Por exemplo, a busca foi configurada para retornar os repositórios mais relevantes com base no número de estrelas, priorizando projetos que têm maior impacto e reconhecimento dentro da comunidade de desenvolvedores. Além disso, os dados brutos obtidos pela API foram formatados e processados em Python para atender às necessidades específicas da pesquisa.

3.6. Infraestrutura de Suporte

A integração entre Google Colab, Python e as bibliotecas mencionadas foi um diferencial estratégico. Com a capacidade do Colab de lidar com grande volume de dados, foi possível minerar informações de repositórios populares de forma eficiente.

Uma preocupação importante ao definir a infraestrutura e tecnologias foi garantir que os resultados fossem precisos e facilmente reprodutíveis. Por isso, o uso de bibliotecas bem estabelecidas como PyDriller e PyGithub assegurou a confiabilidade dos dados extraídos. Além disso, o Google Colab forneceu um ambiente padronizado, minimizando problemas de compatibilidade ou dependências ausentes.

4. Proposta

Este trabalho propõe um protótipo de mineração de repositórios de softwares relacionados à COVID-19 na plataforma GitHub, utilizando a biblioteca PyDriller integrada à

plataforma Google Colab. A proposta focou na criação de um sistema capaz de lidar com grandes volumes de dados, auxiliando pesquisadores e desenvolvedores a analisar e selecionar repositórios para futuras implementações ou contribuições.

Para alcançar o objetivo geral, os seguintes objetivos específicos foram definidos:

- Analisar o panorama dos processos de descoberta de repositórios de software, detalhando as fases que os compõem.
- Minerar repositórios no GitHub por meio da biblioteca PyDriller, extraindo informações gerais sobre os sistemas, seus objetivos e características importantes.
- Desenvolver um protótipo de sistema que ofereça uma análise detalhada de pelo menos cinco repositórios, incluindo dados sobre tecnologias, linguagens de programação e práticas adotadas.

4.1. Metodologia

A metodologia adotada para este projeto seguiu o processo de um estudo observacional, utilizando a estrutura de um experimento controlado, que abrangeu desde a concepção da ideia até a apresentação dos resultados. O processo teve início com a definição da Ideia do Experimento, que consistiu em minerar dados de repositórios no GitHub, focando especialmente em projetos relacionados à pandemia de COVID-19. A partir dessa ideia, foi determinado que o objetivo seria analisar as tecnologias, linguagens de programação e contribuições de desenvolvedores para esses projetos. Essa fase inicial envolveu entender a necessidade de dados para explorar o comportamento dos projetos e as tecnologias mais usadas.

A Definição do Experimento veio a seguir, onde foram estabelecidos os parâmetros para a coleta de dados. Optou-se por analisar repositórios ativos sobre a COVID-19 para entender melhor o panorama das tecnologias utilizadas na área. A ferramenta PyDriller foi escolhida para minerar os dados necessários, e a biblioteca PyGithub foi utilizada para interação com o GitHub.

Com o experimento definido, passamos para a fase de Planejamento do Experimento, onde foi decidido o uso de PyDriller e PyGithub no ambiente Google Colab. Aqui, a configuração das bibliotecas foi realizada e os repositórios foram selecionados. Também foram escolhidos os critérios para analisar a atividade dos repositórios, como o número de *commits* e a frequência das atualizações, além de definir como coletaríamos as informações necessárias.

A Operação do Experimento foi a fase em que a coleta de dados foi realizada. Usamos o PyDriller para minerar os repositórios, extraindo informações sobre o histórico de *commits* e a contribuição dos desenvolvedores. Durante essa fase, foi necessário garantir que todos os dados extraídos estivessem alinhados aos objetivos do experimento, para que as análises subsequentes fossem precisas e representativas do comportamento dos projetos ao longo do tempo.

Com os dados coletados, a Análise e Interpretação das informações começou. Nessa etapa, foi possível identificar padrões e *insights*, como a predominância de certas linguagens de programação e o comportamento das contribuições nos repositórios. A análise também envolveu o exame das tendências nas atualizações dos projetos, o que permitiu uma visão mais clara sobre o impacto dessas iniciativas no contexto da pandemia.

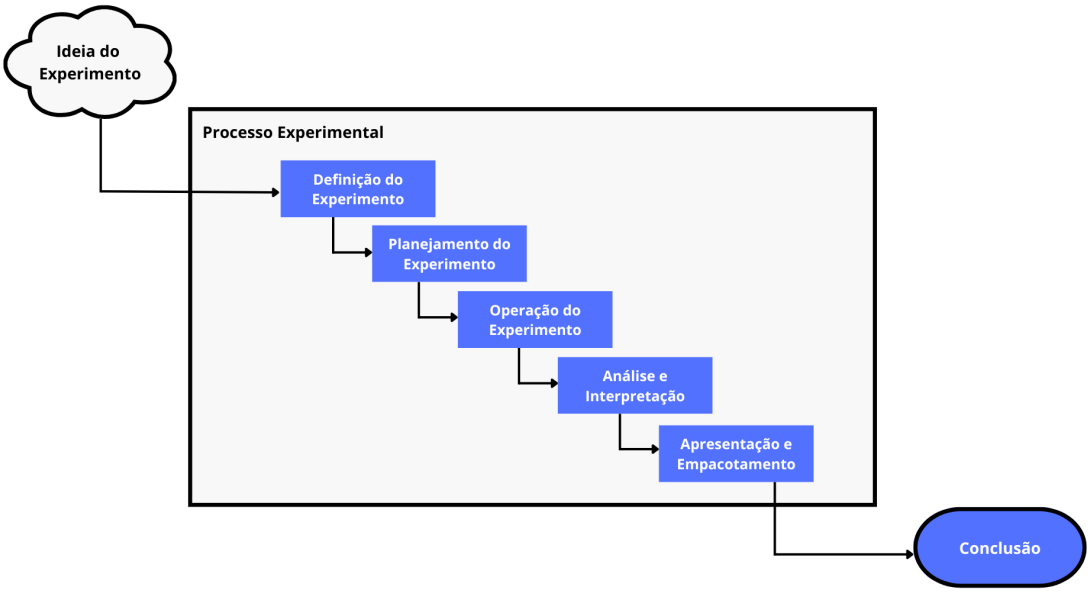


Figura 2. Processo de Experimentação e Suas Fases. Wohlin https://link.springer.com/book/10.1007/978-3-662-69306-3.

Por fim, os resultados foram organizados e apresentados na fase de Apresentação e Empacotamento. Aqui, os dados foram transformados em uma apresentação visual, facilitando a interpretação das informações extraídas. O formato final permitiu que os resultados fossem acessíveis e compreensíveis, não apenas para os pesquisadores envolvidos, mas também para outros que pudessem se interessar pelos achados. Esse empacotamento visou garantir que os resultados pudessem ser facilmente acessados e utilizados para futuras pesquisas ou aplicações.

O processo culminou nas Conclusões, nas quais foram destacados os principais resultados da análise, como as tecnologias mais utilizadas nos repositórios e o perfil das contribuições dos desenvolvedores. A análise proporcionou uma visão sobre como as soluções tecnológicas estavam sendo desenvolvidas durante a pandemia e abriu portas para futuras investigações, incluindo a inclusão de mais repositórios e métricas que possam aprofundar ainda mais os *insights* gerados.

5. Resultados

A análise dos repositórios relacionados à COVID-19 no GitHub revelou informações valiosas sobre cinco projetos classificados com mais estrelas. Através da mineração realizada com a biblioteca PyDriller, foi possível identificar as principais tendências tecnológicas, práticas de desenvolvimento e o nível de colaboração em projetos selecionados. Os resultados obtidos estão descritos a seguir:

5.1. Repositórios Minerados

1. **COVID-19 (Johns Hopkins University):** Este repositório fornece dados globais detalhados sobre a pandemia, sendo amplamente utilizado para análises científicas e governamentais. Com atualizações constantes (7691 *commits*), destaca-se como

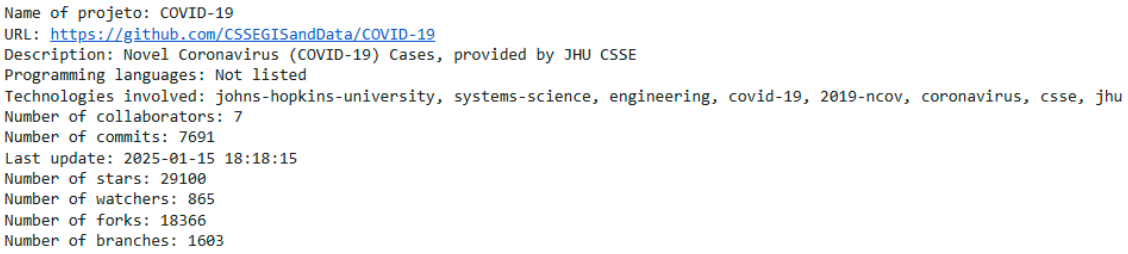


Figura 3. A Figura Apresenta a Mineração do Projeto COVID-19 (Johns Hopkins University).

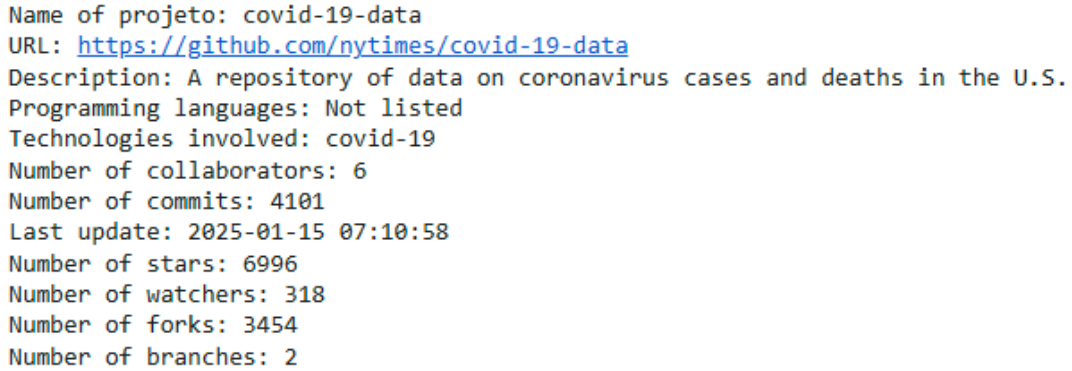


Figura 4. A Figura Apresenta a Mineração do Projeto COVID-19 Data (New York Times).

um dos projetos mais acessados e confiáveis globalmente, possuindo 120 *branches* ativos que possibilitam experimentação e melhorias contínuas. Acesso ao repositório no GitHub: https://github.com/CSSEGISandData/COVID-19. Observe os resultados na Figura 3.

2. **COVID-19 Data (New York Times):** Focado em dados regionais dos Estados Unidos, este repositório oferece informações detalhadas sobre casos, hospitalizações e tendências. É uma fonte confiável para análises específicas no país, com atualizações regulares e dados acessíveis via API. Sua organização é enxuta, com apenas 2 *branches*, refletindo um modelo linear de desenvolvimento. Acesso ao repositório no GitHub: https://github.com/nytimes/covid-19-data. Veja os resultados na Figura 4.
3. **COVID-19 India React:** Combinando dados em tempo real com uma interface intuitiva, este repositório se destaca pela eficiência na análise de dados regionais na Índia. Desenvolvido com React e SCSS (Sassy CSS), prioriza a experiência do usuário e é reconhecido pela acessibilidade e eficiência em apresentar dados complexos de forma visualmente agradável. Acesso ao repositório no GitHub: https://github.com/covid19india/covid19india-react. Observe a Figura 5.
4. **COVID-19 (Tokyo Metropolitan Government):** Este projeto colaborativo fornece dados e visualizações relacionados à pandemia em Tóquio, com forte engajamento da comunidade local, demonstrado pela participação ativa de 317 colaboradores. Construído com Vue.js e TypeScript, apresenta interfaces ricas e interativas que refletem a relevância do projeto para a região. Acesso ao repositório no GitHub: https://github.com/Tokyo-Metro-Gov/covid19. Confira os resultados na

```
Name of projeto: covid19india-react
URL: https://github.com/covid19india/covid19india-react
Description: Tracking the impact of COVID-19 in India
Programming languages: JavaScript, SCSS, HTML, Shell
Technologies involved: covid19, covid19-india, analytics, visualization, tracker, covid-19, coronavirus
Number of collaborators: 108
Number of commits: 1038
Last update: 2025-01-15 11:17:16
Number of stars: 6851
Number of watchers: 280
Number of forks: 3406
Number of branches: 3
```

Figura 5. A Figura Apresenta a Mineração do Projeto COVID-19 India React.

```
Name of projeto: covid19
URL: https://github.com/Tokyo-Metro-Gov/covid19
Description: 東京都 新型コロナウイルス感染症対策サイト / Tokyo COVID-19 Task Force website
Programming languages: Vue, TypeScript, Python, Ruby, SCSS, Shell, JavaScript, Dockerfile, HTML
Technologies involved: covid-19
Number of collaborators: 317
Number of commits: 22764
Last update: 2025-01-09 07:22:15
Number of stars: 6254
Number of watchers: 178
Number of forks: 1964
Number of branches: 3
```

Figura 6. A Figura Apresenta a Mineração do Projeto COVID-19 (Tokyo Metropolitan Government).

Figura 6.

5. **COVID-19 Data (Our World in Data):** Uma das principais referências globais para análise e visualização de dados da pandemia, este repositório é amplamente utilizado em publicações científicas. O código modular facilita o uso por outros desenvolvedores e pesquisadores acadêmicos, reforçando sua relevância. Acesso ao repositório no GitHub: https://github.com/owid/covid-19-data. Observe os resultados na Figura 7.

5.2. Análise dos Resultados

Os resultados obtidos por meio do PyDriller revelaram informações detalhadas sobre os repositórios analisados, destacando padrões e características relevantes dos projetos relacionados à COVID-19 no GitHub. A ferramenta permitiu extrair dados sobre a frequência e a distribuição de *commits*, identificando períodos de maior atividade de desenvolvimento. Foi possível observar que a maioria dos repositórios apresentou um pico inicial de

```
Name of projeto: covid-19-data
URL: https://github.com/owid/covid-19-data
Description: Data on COVID-19 (coronavirus) cases, deaths, hospitalizations, tests • All countries • Updated daily by Our World in Data
Programming languages: Python, HTML, Jupyter Notebook, R, Shell, Batchfile, Makefile
Technologies involved: covid-19, covid, coronavirus, sars-cov-2
Number of collaborators: 64
Number of commits: 31320
Last update: 2025-01-15 15:40:53
Number of stars: 5656
Number of watchers: 149
Number of forks: 3638
Number of branches: 16
```

Figura 7. A Figura Apresenta a Mineração do Projeto COVID-19 Data (Our World in Data).

```
!pip install pydriller PyGithub

from pydriller import Repository
from github import Github

GITHUB_TOKEN = "" # TODO: Substitua as aspas pelo token gerado em sua conta do GitHub
g = Github(GITHUB_TOKEN)
```

Figura 8. Configurações Iniciais do Projeto.

```
def fetch_detached_repositories(termo='covid', limit=10):
    query = f"{termo} in:name,description"
    repos = g.search_repositories(query=query, sort="stars", order="desc")
    return repos[:limit]
```

Figura 9. Método Responsável Pela Busca e Filtro dos Resultados.

contribuição, muitas vezes associado ao início da pandemia, seguido de uma diminuição gradual ao longo do tempo.

Além disso, o PyDriller forneceu informações sobre os desenvolvedores envolvidos, como a quantidade de contribuidores por projeto. Isso evidenciou uma concentração de contribuições em poucos desenvolvedores em muitos dos repositórios, enquanto outros apresentaram um caráter mais colaborativo, com uma distribuição mais homogênea entre os participantes.

Os dados extraídos também destacaram o uso predominante de certas linguagens de programação, como Python e JavaScript, frequentemente empregadas para desenvolver soluções de análise de dados e ferramentas web relacionadas à COVID-19. Adicionalmente, a análise do conteúdo dos *commits* permitiu identificar melhorias de código, correções de bugs e novas funcionalidades implementadas nos projetos.

Em relação às mensagens dos *commits*, a pesquisa evidenciou a variabilidade na qualidade e detalhamento das descrições fornecidas pelos desenvolvedores. Muitos *commits* careciam de informações claras, enquanto outros apresentavam um nível de detalhamento que facilitava a compreensão das alterações realizadas. Esses resultados oferecem uma visão abrangente sobre o panorama dos repositórios analisados e fornecem insumos valiosos para aprimorar a seleção e o desenvolvimento de projetos futuros.

5.3. Código Explicado

Para iniciar o desenvolvimento na plataforma Google Colab foi necessário realizar algumas configurações iniciais, como instalar as bibliotecas Pydriller e PyGithub e importar suas classes. Em seguida, definir um Token de autenticação do GitHub e inicializar um objeto Github para interagir com a API, permitindo analisar repositórios e acessar informações. Observe a implementação na figura 8.

Após as configurações, o método fetch_detached_repositories(termo='covid', limit=10) é responsável pela busca no GitHub dos repositórios cujo nome ou descrição contenham o termo especificado, ordenando por número de estrelas e retornando os primeiros resultados. Veja a figura 9.

Em seguida, o método mining_info_repository(repo) é responsável por exibir detalhes de um repositório, incluindo nome, URL, descrição, linguagens de programação,

```python
def mining_info_repository(repo):
    print(f"Name of projeto: {repo.name}")
    print(f"URL: {repo.html_url}")
    print(f"Description: {repo.description}")

    linguagens = repo.get_languages()
    print(f"Programming languages: {', '.join(linguagens.keys()) if linguagens else 'Not listed'}")

    technologies = repo.get_topics()
    print(f"Technologies involved: {', '.join(technologies) if technologies else 'Not listed'}")

    collaborators = repo.get_contributors().totalCount
    print(f"Number of collaborators: {collaborators}")

    commits = repo.get_commits().totalCount
    print(f"Number of commits: {commits}")

    last_update = repo.updated_at
    print(f"Last update: {last_update.strftime('%Y-%m-%d %H:%M:%S')}")

    print(f"Number of stars: {repo.stargazers_count}")
    print(f"Number of watchers: {repo.subscribers_count}")
    print(f"Number of forks: {repo.forks_count}")
    print(f"Number of branches: {repo.get_branches().totalCount}")

    print("-" * 80)
```

Figura 10. Método Responsável por Exibir Informações dos Projetos.

```python
highlighted_covid_repositories = fetch_detached_repositories()

for repo in highlighted_covid_repositories:
    mining_info_repository(repo)
```

Figura 11. Código Responsável por Obter e Imprimir os Resultados.

tecnologias usadas, número de colaboradores, *commits*, última atualização, estrelas, *watchers*, *forks* e *branches*. Observe a figura 10.

Por fim, o código executa a função fetch_detached_repositories() para obter os 10 repositórios mais populares relacionados a ”covid”e imprime suas informações usando mining_info_repository(repo). Observe na figura 11.

Para acessar o projeto Minerando Repositórios de Softwares Relacionado com COVID-19 no GitHub acesse o link: https://github.com/rodrigomeiraneves/pydriller-covid-minering.

5.4. Ameaças a validade

No desenvolvimento deste trabalho, algumas ameaças à validade foram identificadas, como a biblioteca PyDriller, que embora seja uma ferramenta poderosa para mineração de repositórios, suas funcionalidades dependem da estrutura e da disponibilidade de metadados nos repositórios analisados. Caso os repositórios não sigam boas práticas na documentação ou organização de *commits*, a extração de dados pode ser limitada ou imprecisa.

A seleção dos repositórios relacionados à COVID-19 foi baseada em critérios específicos que podem não contemplar todos os projetos relevantes, excluindo, potencialmente, repositórios que poderiam trazer *insights* valiosos. O desempenho da análise depende diretamente da disponibilidade e estabilidade da API do GitHub, que pode estar

sujeita a limitações de requisição, impactando a coleta de dados. Também, a metodologia adotada pode influenciar os dados coletados e as conclusões obtidas. Fatores como a escolha de métricas e critérios para classificação de repositórios podem introduzir viéses.

Apesar dessas ameaças, o trabalho buscou mitigar possíveis limitações por meio da adoção de critérios claros para a seleção e análise dos repositórios, além de explorar ferramentas amplamente reconhecidas no campo da mineração de software. Contudo, reconhece-se que há espaço para aprimoramento e ampliação dos métodos empregados em trabalhos futuros.

6. Conclusão

O protótipo desenvolvido no Google Colab permitiu extrair e organizar informações relevantes de grandes repositórios de código. Entre as contribuições, destacam-se a facilidade de acesso, permitindo que pesquisadores analisem rapidamente características como número de *commits*, linguagens utilizadas e estrutura de *branches*, e a organização detalhada, que forneceu *insights* sobre práticas de desenvolvimento, tecnologias emergentes e padrões de colaboração, na qual foram as principais metas do projeto. Além disso, a análise aprofundada dos repositórios orientou futuras implementações e contribuições, demonstrando o potencial da mineração de repositórios como ferramenta para pesquisa e inovação.

Esses resultados destacam a importância da mineração de repositórios como um processo essencial para compreender tendências tecnológicas e colaborar em projetos de impacto global. O protótipo não apenas facilitou a extração e organização de dados, mas também revelou padrões de desenvolvimento e colaboração que podem servir de modelo para futuras iniciativas. A combinação da biblioteca PyDriller com a flexibilidade do Google Colab provou ser uma abordagem eficiente e acessível, mostrando que a análise de grandes volumes de informações pode ser simplificada e potencializada para apoiar tomadas de decisão e estimular avanços na pesquisa científica e tecnológica.

6.1. Trabalhos Futuros

Embora o projeto tenha alcançado muitos de seus objetivos, algumas áreas ficaram de fora devido a limitações de tempo e recursos. Uma das principais restrições foi a impossibilidade de analisar um número maior de repositórios de software, o que poderia oferecer uma visão mais abrangente sobre as contribuições dos desenvolvedores no combate à COVID-19. Além disso, o protótipo desenvolvido, baseado na biblioteca PyDriller, pode ser expandido para incluir métricas adicionais, como a frequência de commits e a evolução do código ao longo do tempo.

Outra possível melhoria seria o armazenamento dos dados minerados em uma base de dados estruturada, permitindo consultas e a geração de relatórios em formatos como CSV (Comma-Separated Values). Além disso, a adaptação do protótipo para oferecer maior interatividade ao usuário, incluindo visualizações detalhadas e uma interface mais intuitiva, tornaria a ferramenta ainda mais acessível e eficaz.

Futuras contribuições podem surgir de outros pesquisadores ou estudantes, que poderiam explorar novos aspectos da mineração de dados em repositórios de software, como a implementação de técnicas avançadas de análise de código ou a integração com

outras bibliotecas especializadas. Também seria interessante aplicar a metodologia proposta em outras áreas da saúde ou até mesmo em diferentes tipos de software *open-source*. Ampliar a análise para incluir repositórios de diversas plataformas, além do GitHub, poderia proporcionar uma visão mais global sobre as contribuições tecnológicas durante a pandemia e expandir a aplicabilidade do protótipo para outros contextos.

Referências

THUNG, Ferdian; BISSYANDE, Tegawende F.; LO, David; JIANG, Lingxiao. Network structure of social coding in GitHub. In: 17th EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, 2013, Eindhoven. Anais... Eindhoven: IEEE, 2013. p. 323-326.

Orii, N. (2012). Collaborative topic modeling for recommending GitHub repositories.

MATHIEU, John; MAYNARD, M. Travis; RAPP, Tammy; GILSON, Lucy. Team effectiveness 1997-2007: A review of recent advancements and a glimpse into the future. Journal of Management, v. 34, n. 3, p. 410-476, 2008.

ORNELL, F.; SCHUCH, J. B.; SORDI, A. O.; KESSLER, F. H. P. Pandemia de medo e Covid-19: impacto na saúde mental e possíveis estratégias. Debates em Psiquiatria, Rio de Janeiro, v. 10, n. 2, p. 12–16, 2020.

NABUCO, G.; PIRES DE OLIVEIRA, M. H. P.; AFONSO, M. P. D. O impacto da pandemia pela COVID-19 na saúde mental: qual é o papel da Atenção Primária à Saúde?. Revista Brasileira de Medicina de Família e Comunidade, Rio de Janeiro, v. 15, n. 42, p. 2532, 2020.

ALMEIDA JUNIOR, S. de; SILVA, M. M. da; POPOLIM, R. S.; GONÇALVES, C. R.; MELO, M. R. S. de; BULGO, D. C. Dissemination of knowledge and scientific production in professionalizing courses: A report of experience. Pubsaúde, v. 2, p. 1-8, 2019.

CERQUEIRA, T. G. de O. Explorando características sociais e de colaboração na recomendação de projetos no GitHub. 2020. 138 f. Tese (Doutorado em Ciência da Computação) – Programa de Pós-Graduação em Ciência da Computação, Centro de Engenharia Elétrica e Informática, Universidade Federal de Campina Grande, Paraíba, Brasil, 2020.

BISSYANDE, Tegawendé F.; THUNG, Ferdian; LO, David; JIANG, Lingxiao; RÉVEILLÈRE, Laurent. Orion: a software project search engine with integrated diverse software artifacts. In: 18th INTERNATIONAL CONFERENCE ON ENGINEERING OF COMPLEX COMPUTER SYSTEMS, 2013, Lyon. Anais... Lyon: IEEE, 2013. p. 242-245.

JIANG, Jyun-Yu; CHENG, Pu-Jen; WANG, Wei. Open source repository recommendation in social coding. In: 40th INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 2017, Tokyo. Anais... Tokyo: ACM, 2017. p. 1173-1176.

LAROSE, D. T. Discovering Knowledge in Data: An Introduction to Data Mining. John Wiley and Sons, Inc, 2005.

DABBISH, Laura; STUART, Colleen; TSAY, Jason; HERBSLEB, Jim. Social coding in GitHub: transparency and collaboration in an open software repository. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK, 2012, Seattle. Anais... Seattle: ACM, 2012. p. 1277-1286.

GHAREHYAZIE, Mohammad; RAY, Baishakhi; FILKOV, Vladimir. Some from here, some from there: Cross-project code reuse in GitHub. In: MINING SOFTWARE REPOSITORIES (MSR), 2017 IEEE/ACM 14th International Conference on, 2017, Buenos Aires. Anais... Buenos Aires: IEEE, 2017. p. 291-301.

VASILESCU, Bogdan; FILKOV, Vladimir; SEREBRENIK, Alexander. Perceptions of diversity on GitHub: A user survey. In: EIGHTH INTERNATIONAL WORKSHOP ON COOPERATIVE AND HUMAN ASPECTS OF SOFTWARE ENGINEERING, 2015, Florence. Anais... Florence: IEEE Press, 2015. p. 50-56.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. Técnicas de pesquisa: planejamento e execução de pesquisas, amostragens e técnicas de pesquisas, elaboração, análise e interpretação de dados. 5ª ed. São Paulo: Atlas, 2002.

SANTAELLA, Lucia. Comunicação e Pesquisa – Projetos para Mestrado e Doutorado. São Paulo: Hacker Editores, 2001.

SANTOS, B. P., ALBERTO, A., Lima, T. D. F. M., & CHARRUA-SANTOS, Indústria 4.0: desafios e oportunidades. Revista Produção e Desenvolvimento, v. 4, n. 1, p. 111-124, 2018.

DE LIMA, Alison Gustavo; PINTO, Giuliano Scombatti. INDÚSTRIA 4.0. Revista Interface Tecnológica, v. 16, n. 2, p. 299-311, 2019. Centro de Informações de Rede de Internet da China. Relatório de pesquisa de desenvolvimento da internet da China, 2018. Página inicial. Disponível em: ¡ http://www.cnnic.net/¿. Acesso em: 04/05/2020.

Fundação Fernando Henrique Cardoso: Pandemia e transformação digital: as mudanças nas vidas das empresas e das pessoas, 2020. Página inicial. Disponível em: ¡ https://fundacaofhc.org.br/¿ Acesso em: 04/05/2020.

Instituto de Pesquisa Econômica Aplicada: Pesquisa analisa desafios para o avanço da medicina de precisão no país, 2020. Página inicial. Disponível em: ¡ https://www.ipea.gov.br/¿ Acesso em: 06/05/2020.

FUGGETTA, Antonio. Software Process: A Roadmap. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), 2000, Irlanda. Anais... Irlanda: IEEE, 2000.

ISO/IEC 12207 (1995), Information Technology - Software life cycle processes. Amendment 1 (2002), Amendment 2 (2004).

CHRISSIS, Mary Beth; KONRAD, Mike; SHRUM, Sandra. CMMI: Guidelines for Process Integration and Product Improvement. Boston: Addison Wesley, 2003.

BERGER, Pedro. Instanciação de Processos de Software em Ambientes Configurados na Estação TABA. 2003. Dissertação (Mestrado) — COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2003.

BERTOLLO, Gleidson; SEGRINI, Bruno; FALBO, Ricardo de Almeida.

Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE (SBQS), 5. , 2006, Vila Velha.

HARRISON, William; OSSHER, Harold; TARR, Peter. Software Engineering Tools and Environments: A Roadmap. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), 2000, Limerick. Anais... Limerick: IEEE, 2000. p. 263-277.

FALBO, Ricardo A.; NATALI, André C. C.; MIAN, Paulo G.; BERTOLO, G. Ruy; RUY, F. B. ODE: Ontology-based Software Development Environment. In: IX CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, 2003, La Plata. Memórias... La Plata: ACCI, 2003. p. 1124-1135.

GUARINO, Nicola. Formal Ontology and Information Systems. In: FIRST INTERNATIONAL CONFERENCE ON FORMAL ONTOLOGY IN INFORMATION SYSTEMS, 1998, Trento. Anais... Trento: IOS Press, 1998.

FAYYAD, Usama M.; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic; UTHURUSAMY, Ramasamy (ed.). Advances in Knowledge Discovery and Data Mining. 1. ed. Cambridge: MIT Press, 1996.

WITTEN, Ian H.; FRANK, Eibe. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. 2. ed. San Francisco: Morgan Kaufmann Publishers, 2005.

JACKSON, Peter. Introduction to Expert Systems. Reading: Addison-Wesley, 1986.

BOSCHETTI, Alberto; MASSARON, Luca. Python Data Science Essentials. Birmingham: Packt Publishing Ltd, 2015.

KALLIAMVAKOU, Eirini; GOUSIOS, Georgios; BLINCOE, Kelly; SINGER, Leif; GERMAN, Daniel M.; DAMIAN, Daniela. The promises and perils of mining GitHub. In: 11th WORKING CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), 2014, New York. Anais... New York: Association for Computing Machinery, 2014. p. 92-101. ISBN 9781450328630. DOI: 10.1145/2597073.2597074. Disponível em: https://doi.org/10.1145/2597073.2597074.

XU, Yisen; WU, Fan; JIA, Xiangyang; LI, Lingbo; XUAN, Jifeng. Mining the use of higher-order functions: An exploratory study on Scala programs. Empirical Software Engineering, v. 25, p. 1-38, 2020. DOI: 10.1007/s10664-020-09842-7.

GitHub, Inc. Github public repos and users. https://github.com/search, August 2021. 2.2.2.

BORGES, Hudson; VALENTE, Marco Tulio. What's in a GitHub star? Understanding repository starring practices in a social coding platform. Journal of Systems and Software, v. 146, p. 112-129, dez. 2018. ISSN 0164-1212. DOI: 10.1016/j.jss.2018.09.016. Disponível em: http://dx.doi.org/10.1016/j.jss.2018.09.016.

GitHub, Inc. Github api. https://docs.github.com/en/rest, 2021. 2.2.2, 3.1.2.

MOMBACH, Thaís; VALENTE, Marco Tulio. GitHub REST API vs GHTorrent

vs GitHub Archive: A comparative study. 2018.