Instituto Federal de Educação, Ciência e Tecnologia da Paraíba Campus João Pessoa

Programa de Pós-Graduação em Tecnologia da Informação

Nível Mestrado Profissional

MATIAS SEVERINO RIBEIRO NETO

PLATAFORMA PARA ACOMPANHAMENTO DE MÉTRICAS DE PROCESSO E PROJETO DE SOFTWARE A PARTIR DE DADOS EXTRAÍDOS DO GITHUB

DISSERTAÇÃO DE MESTRADO

JOÃO PESSOA – PB

Dezembro de 2023

Matias Severino Ribeiro Neto

PLATAFORMA PARA ACOMPANHAMENTO DE MÉTRICAS DE PROCESSO E PROJETO DE SOFTWARE A PARTIR DE DADOS EXTRAÍDOS DO GITHUB

Dissertação de Mestrado apresentada como requisito parcial para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós-Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB.

Orientadora: Prof. Dra. Juliana Dantas Ribeiro Viana de Medeiros

Coorientadora: Prof. Dra. Heremita Brasileiro Lira

João Pessoa – PB 2023

Dezembro de 2023

Dados Internacionais de Catalogação na Publicação – CIP
Biblioteca Nilo Peçanha – IFPB, *campus* João Pessoa

R484p Ribeiro Neto, Matias Severino.
Plataforma para acompanhamento de métricas de processo e projeto de *software* a partir de dados extraídos do GITHUB / Matias Severino Ribeiro Neto. – 2024.
76 f. : il.

Dissertação (Mestrado em Tecnologia da Informação) – Instituto Federal da Paraíba – IFPB / Programa de Pós-Graduação em Tecnologia da Informação - PPGTI.

Orientação: Profa. Dra Juliana Dantas Ribeiro V. de Medeiros.
Coorientação : Profa. Dra Heremita Brasileiro Lira

1.Métricas de processo. 2. Projeto de *software*. 3. Apoio a tomada de decisão. 4. Desenvolvimento de *software*. 5. GitHub. I. Título.

CDU 004.4 (043)

Bibliotecária responsável Lucrecia Camilo de Lima – CRB15/132

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU*

MESTRADO PROFISSIONAL EM TECNOLOGIA DA INFORMAÇÃO

MATIAS SEVERINO RIBEIRO NETO

Plataforma para Acompanhamento de Métricas de Processo e Projeto de Software a Partir de Dados Extraídos do GitHub

Dissertação apresentada como requisito para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós- Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB - Campus João Pessoa.

Aprovado em 21 de dezembro de 2023

Membros da Banca Examinadora:

Dra. Juliana Dantas Ribeiro Viana de Medeiros

IFPB - PPGTI

Dra. Heremita Brasileiro Lira

IFPB - PPGTI

Dr. Danyllo Wagner Albuquerque

IFPB

Dr. Katyusco de Farias Santos

IFPB

João Pessoa/2023

Documento assinado eletronicamente por:

- **Juliana Dantas Ribeiro Viana de Medeiros**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 22/12/2023 10:25:55.
- **Heremita Brasileiro Lira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 22/12/2023 14:00:34.
- **Danyllo Wagner Albuquerque**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 22/12/2023 23:06:12.
- **Katyusco de Farias Santos**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 22/01/2024 11:28:55.

Este documento foi emitido pelo SUAP em 11/12/2023. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 507175
Código de Autenticação: 7f2bd7da48

Dedico este trabalho a todas as pessoas que, de forma direta ou indireta, contribuíram significativamente para sua realização. Aos desafios enfrentados, pois foram eles que reforçaram meu compromisso inabalável com a busca constante pelo desenvolvimento.

RESUMO

No âmbito do desenvolvimento de *software*, a tomada de decisões é uma atividade que requer acesso a informações precisas e relevantes. Para profissionais envolvidos nesse processo, contar com ferramentas capazes de sintetizar e apresentar essas informações de maneira clara é crucial. O *GitHub*, uma plataforma amplamente utilizada para controle de versões de projetos de *software* online, oferece uma API (*Application Programming Interface*) que permite a consulta de dados e a obtenção de indicadores essenciais para os projetos. Embora essa funcionalidade contribua para manter os participantes informados e engajados, a apresentação clara dessas informações ainda representa um desafio, exigindo esforços consideráveis para integrar os dados coletados a outros sistemas ou desenvolver soluções específicas. Diante desse cenário, este estudo tem como objetivo principal auxiliar na tomada de decisões em projetos de *software*, através da disponibilização de métricas de processo e de projeto extraídas de repositórios *GitHub*. A pesquisa iniciou-se com uma exploração de trabalhos científicos para identificar métricas de software relevantes. Em seguida, foi concebida a abordagem da Plataforma GMC (Gerenciamento de Métricas) para extrair e processar dados, independentemente da linguagem de programação utilizada. A Plataforma é um sistema de código aberto que busca proporcionar uma abordagem abrangente e adaptável ao diversificado cenário de monitoramento de projetos de desenvolvimento de *software* através da disponibilização de métricas de processo e de projeto. Para avaliação da Plataforma, foi aplicado o processo de extração e tratamento de dados em dois projetos públicos do *GitHub*, além de uma análise das métricas por meio de um questionário direcionado a profissionais do desenvolvimento de *software*. Os resultados da avaliação destacam a utilidade em oferecer informações relevantes para identificar falhas e monitorar a progressão do projeto, facilitando a tomada de decisões estratégicas através de indicadores, tabelas e gráficos. Uma limitação identificada durante a avaliação foi a ausência de opções de filtragem de informações na plataforma GMC, que permita a personalização de consultas dinâmicas e interativas na base de dados. A extração de dados do GitLab foi identificada como uma oportunidade de melhoria para que seja possível aumentar a abrangência de utilização da ferramenta. Os resultados da avaliação demonstram que a Plataforma GMC tem potencial para contribuir com a melhoria contínua dos processos de desenvolvimento de software.

Palavras-chaves: Métricas de Processo e Projeto de *Software*; Apoio à Tomada de Decisão; *GitHub*;

ABSTRACT

In the context of software development, decision-making is an activity that requires access to precise and relevant information. For professionals involved in this process, having tools capable of synthesizing and presenting this information clearly is crucial. GitHub, a widely used platform for online software project version control, offers an API (Application Programming Interface) that allows data querying and obtaining essential project indicators. Although this functionality helps keep participants informed and engaged, the clear presentation of this information still poses a challenge, requiring considerable efforts to integrate collected data into other systems or develop specific solutions. Given this scenario, this study aims to primarily assist in decision-making in software projects by providing process and project metrics extracted from GitHub repositories. The research began with an exploration of scientific works to identify relevant software metrics. Then, the GMC (Metrics Management) Platform approach was conceived to extract and process data, regardless of the programming language used. The Platform is an open-source system that seeks to provide a comprehensive and adaptable approach to the diverse software development project monitoring landscape by providing process and project metrics. To evaluate the Platform, the data extraction and processing process was applied to two public GitHub projects, along with an analysis of the metrics through a questionnaire directed at software development professionals. The evaluation results highlight the usefulness of providing relevant information to identify failures and monitor project progression, facilitating strategic decision-making through indicators, tables, and graphs. A limitation identified during the evaluation was the lack of information filtering options in the GMC platform, allowing for customization of dynamic and interactive queries in the database. GitLab data extraction was identified as an improvement opportunity to increase the tool's utilization scope. The evaluation results demonstrate that the GMC Platform has the potential to contribute to the continuous improvement of software development processes.

Keywords: Process and Software Project Metrics; Decision-Making Support; GitHub.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMP	*Assert Metrics Panel*
AWS	*Amazon Web Service*
APF	Analise de Pontos de Função
API	*Application Programming Interface*
BSD	*Berkeley Software Distribution*
CLI	*Command Line Interface*
DBA	*Database Administrator*
DEVOPS	*Software Development (dev) and Operations (ops)*
FCR	*First Call Resolution*
GMC	Plataforma para Gerenciamento de Métricas
GPL	*General Public License*
GUI	*Graphical User Interface*
HTTP	*HyperText Markup Language*
IFPB	Instituto Federal de Educação Ciência e Tecnologia da Paraíba
JSON	*JavaScript Object Notation*
LGPD	Lei Geral de Proteção de Dados
MB	*Megabyte*
MIT	*Massachusetts Institute of Technology*
NoSQL	*Not Only SQL*
QA	*Quality Assurance*
REST	*Representational State Transfer*

SOLID *Single Responsibility Principle, Open/Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, Dependency Inversion Principle.*

SP *Story Points*

URI *Uniform Resource Identifier*

URL *Uniform Resource Locator*

VCS *Version Control System*

WEB *World Wide Web*

YAML *Ain't a Markup Language*

LISTA DE SÍMBOLOS

#	Cerquilha
+	Adição
-	Subtração
|	Módulo
∑	Somatório

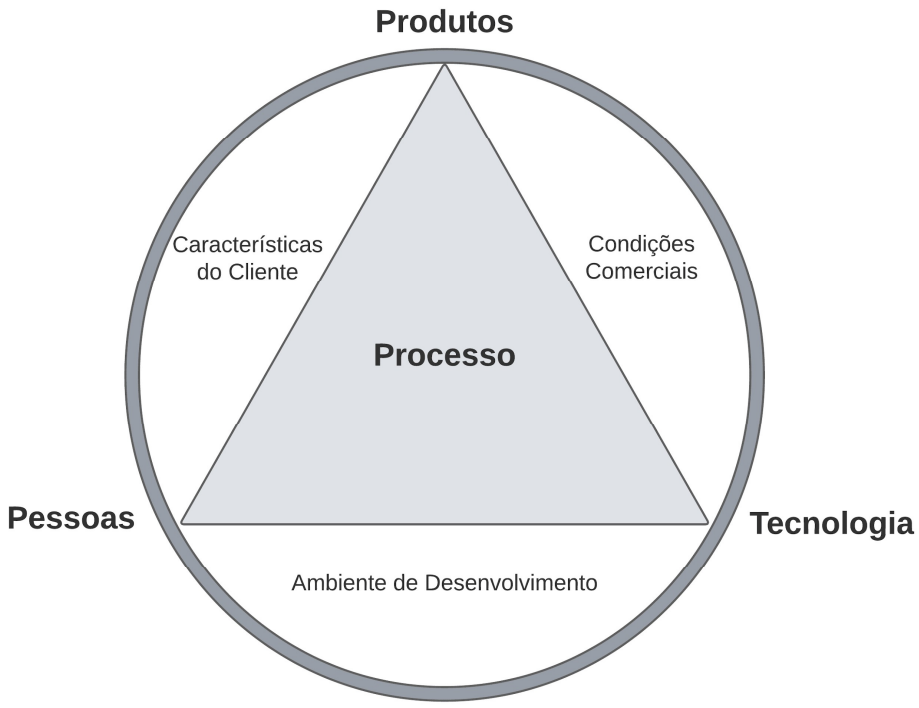
SUMÁRIO

1. INTRODUÇÃO

O desenvolvimento de *software* é um processo composto por atividades e eventos interdependentes, que frequentemente ocorrem em ambientes incertos. Sommerville (2018) afirma que essas atividades englobam gerenciamento, planejamento, modelagem, análise, especificação, projeto, implementação, teste e manutenção de *software*. De acordo com Sommerville (2018), o processo de desenvolvimento de *software* é concebido com o propósito de possibilitar uma compreensão profunda e uma gestão controlada de cada atividade envolvida. Essa abordagem visa a minimização das contingências e incertezas que possam surgir durante as fases de construção e manutenção do *software*.

Na figura 1, Pressman et al., (2021) resume o processo de criação do *software* no centro da estrutura, destacando três pontos fundamentais: pessoas, produtos e tecnologia. O processo assume uma posição central, formando um triângulo que interliga esses elementos cruciais, exercendo influência profunda não apenas na qualidade do *software*, mas também no desempenho organizacional. A complexidade dos produtos pode impactar substancialmente a qualidade e o desempenho da equipe. Além disso, a tecnologia empregada no processo também desempenha um papel significativo. Importante notar que o triângulo do processo está inserido em um contexto mais amplo, circundado por condições ambientais que englobam o ambiente de desenvolvimento (e.g. ferramentas de *software* integradas), as condições de negócios tais como e.g., prazos e regras de negócios e as características do cliente como e.g., facilidade de comunicação e colaboração.

Figura 1 - Fatores determinantes para qualidade do software e eficácia organizacional



Fonte: Adaptação de PRESSMAN et al., (2021)

No contexto da gestão de um projeto de *software*, visando alcançar uma eficácia ampliada, é essencial estabelecer bases sólidas em métricas tangíveis. Como é possível identificar a não realização de uma tarefa sem a presença de uma métrica para embasar tal conclusão? Como se pode determinar que a qualidade é insatisfatória? De uma maneira ou outra, o gestor de projeto inevitavelmente se apoia na prática de mensuração de *software*, que engloba a aplicação de uma ou mais métricas (WAZLAWICK, 2019). De acordo com Wazlawick (2019), é de extrema relevância evitar qualquer ambiguidade entre os seguintes termos:

- Métrica: que corresponde à escala na qual os valores de uma medida são coletados;
- Medição: o processo de obtenção de medidas;
- Medida: o valor adquirido para alguma dimensão específica do software.

No desenvolvimento de *software*, a medição pode ser considerada como um dado ou efeito para relatar sobre o andamento das atividades ou produção dos envolvidos no projeto. Já a métrica representa uma recorrência que traz consigo valor informativo, motivacional, investigativo ou algo que pode ocorrer em um momento futuro. A métrica ajuda no entendimento de riscos atrelados a perda de resultados esperados, na mudança do processo ou nas práticas consolidadas que alcançaram um melhor desempenho (NICOLETTE, 2015).

Fenton (2014) descreve três contextos básicos para entender e controlar um projeto de desenvolvimento de *software* baseados nas informações de gerentes e desenvolvedores. Primeiro, a medição pode mostrar com mais detalhes o que está acontecendo durante o desenvolvimento e a manutenção do projeto de *software*, neste sentido o gerente de projeto pode avaliar os artefatos para as próximas iterações e compreender melhor onde cada problema afeta o projeto. Em segundo lugar, a medição permite controlar o que está acontecendo no projeto, podendo muitas vezes prever o que acontecerá e fazer modificações em processos e produtos para atingir os objetivos finais. Em terceiro lugar, a medição nos encoraja a melhorar nossos processos e produtos. A qualidade do que está sendo construído pode ser diretamente relacionada às expectativas daqueles que tomam decisões baseadas em medições durante o andamento do projeto ou de medições de históricos dos dados coletados.

Métricas de *software* também proporcionam informações necessárias para criar toda a estrutura do projeto de *software* de forma simples, computável, persuasiva, consistente e objetiva (PRESSMAN et al., 2021). Neste contexto, esta pesquisa explora métricas de processo e projeto de *software* visando desenvolver uma Plataforma para Gerenciamento de Métricas, denominada GMC[1], que permita a visualização de métricas de *software* extraídas do *GitHub*.

[1] G – Gerenciamento, MC - Métricas

1.1. Motivação e Definição do Problema

Todos os anos, a empresa *Octoverse Report* (2023) analisa dados de desenvolvedores e repositórios de código do *GitHub* para compartilhar tendências entre hábitos de trabalho, produtividade e satisfação com a carreira. No início de 2023, mais de 100 milhões de contas estavam cadastradas no *GitHub*, um crescimento que estava previsto somente para 2025 e no Brasil mais de 3 milhões de desenvolvedores utilizam o *GitHub* para controle e manutenção de projetos de *softwares*.

No estudo conduzido por De Bassi et al. (2018), demonstra-se a viabilidade de avaliar as contribuições individuais dos desenvolvedores por meio da aplicação de métricas de qualidade. Além disso, a pesquisa evidencia que é possível monitorar em tempo real a evolução dessas métricas. A variação ao longo do tempo de cada métrica pode ter um impacto considerável em atributos de qualidade de *software* tais como: manutenibilidade, testes, reutilização e compreensão do código-fonte. Portanto, as métricas de qualidade desempenham um papel fundamental ao fornecer informações sobre o impacto de cada contribuição no projeto de *software*.

A pesquisa conduzida por Kozik et al. (2018) se concentra na análise de dados com o objetivo de auxiliar as empresas na avaliação da qualidade de seu *software*. O estudo demonstra que indicadores estratégicos podem desempenhar um papel fundamental na tomada de decisões organizacionais e que esses indicadores são construídos com base em métricas de qualidade de *software*.

Para consultar e exibir métricas de processo e projeto de *software* existem ferramentas ou sistemas comerciais, como por exemplo, a *API GitHub* [2](2023), o *Elastic Stack* [3](2023) e o *Sonarqube* [4](2023) que abordam diversas funcionalidades, incluindo métricas relacionadas ao produto e aos recursos computacionais do servidor, como processamento e extração de dados relacionados à infraestrutura, bem como ao código fonte do projeto.

Aprimorar a apresentação de métricas de forma sintetizada em comparação com a API do GitHub é um dos desafios centrais abordados por esta pesquisa, visando facilitar a compreensão e aplicação prática para os colaboradores envolvidos no projeto. Buscando melhorar a visualização das métricas de processo e projeto de *software*, propondo uma abordagem alternativa a existente como a API de métricas do GitHub, e que omitisse a necessidade de codificação para exibir essas métricas. Dessa forma, almejamos não apenas simplificar o acesso às informações das métricas, mas também tornar o processo mais intuitivo e eficiente para os envolvidos no projeto, promovendo uma compreensão mais abrangente e facilitando a tomada de decisões.

Através da extração de dados dos projetos disponíveis do *GitHub* é possível transformá-los em informações ou métricas. O escopo desta pesquisa se concentra na exploração de métricas relacionadas a processos e projetos de *software*, bem como no desenvolvimento de uma plataforma dedicada à sua

[2] API *GitHub*: https://docs.github.com/pt/rest?apiVersion=2022-11-28
[3] *Elastic Stack*: https://www.elastic.co/pt/elastic-stack
[4] Sonarqube: https://www.sonarsource.com/products/sonarqube/

apresentação. A plataforma denominada GMC (Plataforma para Gerenciamento de Métricas) foi concebida como uma solução de código aberto para exibição de métricas, eliminando a necessidade de conhecimentos das ferramentas de visualização de métricas ou da criação de código-fonte para comunicação com APIs. Isso permite uma contribuição substancial para a melhoria do processo de tomada de decisões por parte da equipe de projetos de *software*.

Oliveira (2017) realizou uma pesquisa com o propósito de identificar os fatores mais influentes na produtividade dos desenvolvedores em organizações de *software*. Uma característica notável do referido estudo é o direcionamento primordial aos gestores de equipes de desenvolvimento de *software*. Entretanto, a proposta da pesquisa realizada no âmbito do desenvolvimento da Plataforma GMC, é ampliar o enfoque, considerando também todos os envolvidos no projeto. Nesse contexto, busca-se não apenas atender às necessidades dos gestores, mas também compreender e abordar as expectativas e interesses dos demais envolvidos no processo de desenvolvimento de *software*. Essa abordagem abrangente visa enriquecer a compreensão das métricas e promover uma gestão holística e eficaz no ambiente de desenvolvimento de *software*. A análise estatística computacional através de métricas pode fornecer aos gerentes e engenheiros de *software insights* significativos e capacitar suas equipes a tomar melhores decisões. É essencial que esses *insights* sejam oportunos e acionáveis, para que possam ser utilizados de forma eficaz pela equipe de desenvolvimento (PRESSMAN et al., 2021). Como contribuição, esta pesquisa também proporciona aos envolvidos no projeto a capacidade de utilizar a Plataforma GMC como uma fonte acessível para adquirir métricas, as quais detêm o potencial de atuar na melhoria da qualidade do projeto de *software*.

1.2. Objetivos

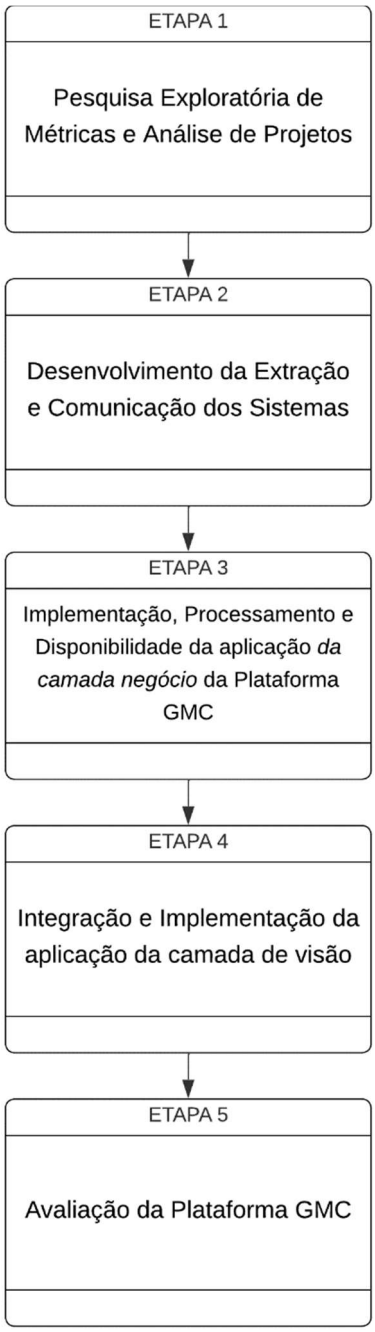
O objetivo geral desta pesquisa é auxiliar os tomadores de decisão envolvidos em projetos de *software* através da disponibilização de métricas de processo e projeto de *software* extraídos dos repositórios *GitHub*. Para atingir o objetivo geral, o trabalho prevê os seguintes objetivos específicos:

- Realizar pesquisa exploratória na literatura para identificar métricas de processo e projeto de *software.*
- Analisar projetos de desenvolvimento de *software Open Source* disponíveis no *GitHub*.
- Definir um conjunto de métricas de processo e projeto de *software* a serem extraídas do *GitHub.*
- Projetar e implementar uma plataforma para extrair dados do *GitHub*, processá-los e gerar métricas de processo e projeto de *software.*
- Avaliar as métricas exibidas na plataforma simulando um ambiente real produtivo e também por meio de um questionário aplicado aos profissionais de desenvolvimento de *software*.

1.3. Metodologia

Com o propósito de atingir os objetivos deste trabalho, foi necessário realizar uma investigação sobre métricas relacionadas a processo e projeto de *software*. Para isso, foi conduzida uma pesquisa exploratória por meio de um levantamento e uma investigação bibliográfica de trabalhos técnicos. O objetivo dessa investigação preliminar foi identificar quais métricas estão sendo utilizadas no processo de desenvolvimento de *software*. Na metodologia foram definidas cinco etapas ilustradas na Figura 2 a seguir.

Figura 2 - Etapas da pesquisa



Fonte: Própria

- **Etapa 1:** Pesquisa Exploratória de Métricas e Análise de Projetos. A etapa inicial do

procedimento envolveu a condução de uma pesquisa bibliográfica na literatura, juntamente com uma análise detalhada de projetos de código aberto hospedados no repositório *GitHub*. O objetivo principal desta fase foi realizar uma pesquisa abrangente para identificar as métricas exploradas em trabalhos científicos relevantes. Além disso, buscou-se analisar os dados que poderiam ser extraídos dos projetos armazenados no repositório *GitHub*, destacando as soluções implementadas para os desafios enfrentados, ou seja, os problemas que os projetos conseguiram solucionar e avaliar a amplitude das informações disponíveis. Essa abordagem visa proporcionar uma base sólida para a subsequente automatização do processamento de dados obtidos.

- **Etapa 2:** Desenvolvimento da Extração e Comunicação dos Sistemas. O objetivo primordial nesta etapa é a implementação de uma API que possibilite a interação entre a ferramenta GDDownloader e os sistemas da camada de visão e de negócio foram desenvolvidos como parte integrante desta pesquisa. Esses sistemas foram construídos utilizando diversas linguagens de programação com o propósito de compor a plataforma denominada GMC. O desenvolvimento desta API foi essencial para a automatização na comunicação entre a ferramenta de extração de dados e a plataforma GMC não seria viável. Dessa forma, o GDDownloader é aprimorado para extrair os dados previamente definidos dos projetos já analisados no repositório *GitHub*. Isso é feito com o intuito de obter a estrutura do conjunto de dados e estabelecer novas rotinas que atendam aos requisitos de comunicação estabelecidos pela plataforma de métricas GMC.

- **Etapa 3:** Implementação, Processamento e Disponibilidade da aplicação de negócio da Plataforma GMC. Nesta etapa, realizamos a implementação de uma aplicação de visão encarregada de efetuar consultas na base de dados *NoSQL* onde os dados foram extraídos. Além disso, nessa etapa ocorro todo processamento da transformação de dados em métricas, também implementamos algoritmos baseados nas descrições das métricas suportadas pela Plataforma (vide Tabela 4) e adicionamos o código-fonte necessário para disponibilizar uma API que fornece as informações processadas que podem ser consumidos por meio da camada de visão da Plataforma GMC.

- **Etapa 4:** Integração e Implementação da camada de visão. Esta etapa teve por objetivo implementar uma aplicação para a camada de visão da plataforma de métricas GMC. Essa aplicação estabelece a comunicação com as demais aplicações implementadas, mais especificamente, entre a ferramenta de extração e o sistema da camada de negócio, e permite que o usuário tenha acesso às métricas.

- **Etapa 5:** Avaliação da Plataforma. Esta avaliação foi conduzida em duas fases. A primeira fase abrangeu as etapas de extração, integração e visualização das métricas relacionadas aos

processos e projetos de *software*. Na segunda fase, envolveu a aplicação de um questionário direcionado a profissionais de desenvolvimento de *software*, fundamentado nas métricas previamente investigadas durante a pesquisa.

Após seguir as etapas delineadas, levando em consideração o contexto e os objetivos previamente estabelecidos, este estudo progrediu significativamente na disponibilização de métricas associadas aos processos e projetos de *software*, utilizando dados extraídos do *GitHub*. Essa abordagem visa proporcionar à equipe de desenvolvimento de *software* maior controle das atividades, contribuindo para uma gestão mais eficiente do trabalho realizado.

1.4. Aplicabilidade

A Plataforma GMC desenvolvida no âmbito desta pesquisa se propõe a apoiar na tomada de decisão de gestores e equipe de desenvolvimento a partir da disponibilização de métricas de processo e projeto de *software*. A plataforma GMC é *Open Source* e agrega dados sobre métricas de processo e projeto de *software* armazenadas no repositório *GitHub*. Esses dados são extraídos, processados e armazenados em um banco de dados, permitindo o consumo dessas informações por meio de uma aplicação *web*. Os resultados são apresentados na forma de indicadores, tabelas e gráficos, tornando as métricas acessíveis aos envolvidos no projeto de forma mais clara e conveniente tendo como base ferramentas exploradas nesta pesquisa e a API do *GitHub*.

Os envolvidos no projeto podem obter informações processadas de forma simples e instantânea sobre projetos em andamento ou até mesmo projetos de *software* concluídos que ficaram armazenados no repositório *GitHub*. Essas métricas podem ajudar no monitoramento da qualidade, produtividade, desempenho da equipe, eficiência e no processo de *software* utilizado.

Empresas tanto do setor público quanto do setor privado que empregam a plataforma *GitHub* têm a possibilidade de incorporar o sistema resultante desta pesquisa com o intuito de apresentar métricas correlacionadas aos domínios de processo e projeto de desenvolvimento de *software*. Esta integração é de alcance abrangente, independente da escala do projeto em questão, permitindo que qualquer membro da equipe aproveite plenamente os recursos fornecidos pela Plataforma GMC.

Apesar de sua semelhança com um painel web que integra gráficos e indicadores de métricas de processos e projetos de *software*, a estrutura desta pesquisa está centrada na análise dessas métricas. A aplicação de diversos algoritmos, explorados ao longo do estudo, visa aprimorar os métodos de processamento dessas métricas. O foco central desta pesquisa é aprimorar a compreensão das pessoas em relação a projetos de *software*, alcançando esse objetivo por meio da extração e apresentação de métricas obtidas do *GitHub*.

1.5. Estrutura do Documento

Os capítulos seguintes estão organizados da seguinte forma: O segundo capítulo deste trabalho aborda a apresentação da base teórica essencial para uma compreensão aprofundada da pesquisa. Nesse contexto, realizamos uma análise das métricas de *software*, exploramos as ferramentas pertinentes, com ênfase no repositório GitHub. Além disso, oferecemos uma explicação detalhada dos cálculos métricos empregados para transformar dados brutos em informações processadas, ao mesmo tempo em que discutimos o processo de extração dos dados do repositório GitHub. No terceiro capítulo, apresentamos os trabalhos relacionados diretamente à pesquisa, destacando suas distinções e examinando de que maneira essas contribuições influenciaram o progresso do estudo. No quarto capítulo, delineia-se a abordagem GMC, contextualizando a concepção e a busca em resolver desafios relacionados a métricas de *software*. A abordagem GMC parte da identificação de lacunas nas práticas atuais, chegando à apresentação de uma solução estratégica para ajudar na tomada de decisão para projetos de *software*. No quinto capítulo, é apresentada em detalhes a proposta da plataforma GMC. Isso inclui a abordagem da arquitetura das aplicações integradas, a investigação e análise das métricas, o processo de extração de dados do GitHub, bem como a implementação detalhada das camadas de visão e negócio. O sexto capítulo dedica-se à avaliação da plataforma, estruturada em duas fases distintas. Na primeira etapa, foram conduzidos testes abrangentes, avaliando desde a extração até a apresentação dos dados na camada de visão. Na segunda fase, um questionário foi administrado para avaliar as métricas investigadas durante esta pesquisa, bem como para coletar informações junto aos profissionais que atuam, ou atuaram, no processo de desenvolvimento de projetos de *software*. No sétimo capítulo são exibidas as limitações do trabalho e seus impactos. E, por fim, as considerações e apontamentos para futuras pesquisas.

1. FUNDAMENTOS

Este capítulo tem por finalidade apresentar conceitos fundamentais relacionados a métricas de processo e projeto de software no contexto do desenvolvimento de sistemas. Além disso, são explorados conceitos relacionados a repositórios de código, com foco na utilização do *GitHub*. Por último, são apresentados os trabalhos relacionados a esta pesquisa.

1.1. Métricas de Processo e Projeto de *Software*

Conforme destacado por Sommerville (2018), as métricas podem ser classificadas em dois tipos principais: métricas de controle e métricas de previsão. As métricas de controle, também identificadas como métricas de processo, desempenham um papel crucial na gestão de processos, enquanto as métricas de previsão, categorizadas como métricas de projeto, contribuem para a projeção das características do *software*.

Uma das abordagens é a mensuração do tempo necessário para a conclusão de um processo específico que pode ser abordada por meio de diversas estratégias, incluindo a medição do tempo total alocado ao processo, a consideração do tempo no calendário ou a análise do tempo investido por analistas ou engenheiros designados. Adicionalmente, os recursos necessários para a execução desse processo podem variar, abrangendo o esforço total em pessoa/dias, custos de viagem ou recursos computacionais (SOMMERVILLE, 2018).

Pressman et al. (2021) destacam que as métricas de processo de *software* podem ser utilizadas para fins estratégicos e no decorrer de longos períodos, podem ser utilizadas para o aperfeiçoamento do processo de *software*. Já as métricas de projeto permitem avaliar o estado de um projeto em andamento, rastrear os riscos em potencial, descobrir áreas problemáticas antes que se tornem críticas, ajustar o fluxo de tarefas, avaliar a habilidade da equipe de projeto para controlar a qualidade dos artefatos (PRESSMAN et al., 2021).

De acordo com Pressman et al. (2021), as métricas de projeto e de processo de *software* são medidas quantitativas que permitem verificar a eficácia do processo de *software* e ajudam a entender melhor o projeto e a construção do *software* produzido. Baseados em artefatos de projetos em andamento ou já realizados, é possível construir métricas para guiar novas atividades. Essas métricas permitem avaliar a condição de um projeto em andamento, rastrear riscos em potencial, descobrir áreas problemáticas antes de tornarem-se críticas, ajustar o fluxo de trabalho, avaliar a capacidade da equipe do projeto.

Várias pesquisas sobre métricas de *software* abordam os profissionais da área de computação para relatar suas experiências. Ram et al. (2020) realizaram um estudo de caso com quatro empresas de *software* que utilizam soluções com métricas em trabalhos diários. Um questionário foi elaborado para documentar as opiniões dos profissionais sobre as métricas. Entre essas métricas pesquisadas estão

complexidade de código, densidade de alterações, *bugs* e testes de código.

Uma das principais percepções na pesquisa de Ram *et al.* (2020) é a confiança em métricas para os membros da equipe. Métricas práticas como número de alterações críticas de um projeto contribuem para a qualidade do *software* e consequentemente para o modelo de processo utilizado. Embora que muitas métricas de código sejam subjetivas e dependem de um profissional para interpretá-la, algumas métricas simples podem ser acionáveis e de fácil manipulação, como por exemplo a quantidade de defeitos em um período de tempo.

Para enfatizar as métricas de processo de *software*, Mladenova (2020) considera as métricas mais usadas para testes e examina as métricas que podem ser aplicadas no processo de desenvolvimento. Na Figura 3 são classificadas as duas principais categorias de métricas: processo e projeto de software.

Figura 3 - Classificação das métricas de qualidade de software

Métricas de Processo de Software	Métricas de Projeto de Software
Tempo de desenvolvimento	Riscos
Custo de desenvolvimento	Plano de desenvolvimento de software
Eficiência da equipe	
Defeitos detectados durante o desenvolvimento	

Fonte: Adaptação de Mladenova (2020)

As métricas de projeto podem medir riscos potenciais no início do projeto dentro do plano de trabalho, ou seja, essas métricas ajudam a monitorar o custo do projeto com base na experiência e nos dados coletados de projetos anteriores. Nas métricas de processo, as informações coletadas e analisadas podem prever problemas futuros e medir a eficiência da equipe ao longo do projeto e as métricas de produto trazem valores como tamanho do *software*, complexidade, performance (MLADENOVA, 2020). Os resultados da pesquisa de Mladenova (2020) mostram que a avaliação de qualidade é um extenso processo que deve ser feito em todas as fases de desenvolvimento do sistema juntamente com as métricas de *software*, e que as ferramentas ou plataformas não cobriam o plano de verificação de qualidade do *software*.

Hernández *et al.* (2019) conduziram uma revisão sistemática que identificou 21 métricas de produtividade de equipes, incluindo métricas relacionadas ao tempo, problemas resolvidos e não concluídos durante as atividades da Sprint. As métricas delineadas no estudo estão voltadas para avaliar características do software e a produtividade da equipe por meio de cálculos específicos conforme lista definida na Tabela 1.

Tabela 1 - Métricas identificadas

ID	Métrica	Descrição
M1	Tempo de ciclo	Tempo que leva para concluir uma tarefa específica de início ao fim
M2	Tempo de espera	Tempo que ocorre entre a solicitação de um cliente e a liberação do produto
M3	Tempo de estado	Tempo médio que uma tarefa permanece em um estado (A fazer, em andamento, bloqueado, progresso interrompido, em revisão de código)
M4	Preparação para lançamento	Tempo que a equipe precisa para finalizar a Sprint e implantar o produto
M5	Defeitos externos	Número de defeitos reportados pelo cliente, usuário ou pessoal externo a equipe, durante um período de tempo
M6	Desempenho dos testes	Razão entre os pontos de testes e o tempo total gasto no desenvolvimento médio em horas/pessoa
M7	Densidade de defeitos dos testes	Densidade de defeitos encontrados em testes unitários
M8	Qualidade do código	Impacto das alterações no código e na qualidade do código fonte
M9	Velocidade	Proporção entre pontos de história de usuários concluídos e a iteração. Somatório dos pontos de história concluídos em uma iteração vezes o número médio de horas por pontos de história no projeto
M10	Burndown	Quantidade de trabalho a ser feito em uma Sprint
M11	Retorno do investimento	Medida da relação custo-benefício. Razão entre valor médio em pontos
M12	Porcentagem de trabalhos aceitos	Horas de trabalho gastas em histórias de usuários concluídas durante uma Sprint, divididas pelo total de horas de trabalho gastas na Sprint
M13	Esforço de desenvolvimento e testes	Esforço necessário de desenvolver e testar um produto de software
M14	Capacidade de inovação de uma equipe	Número de ideias-chave das partes interessadas, número de ideias geradas pela equipe, número de ideias geradas por terceiros
M15	Eficiência de fluxo	Proporção entre a quantidade de tempo gasto de trabalho e a quantidade de tempo de espera
M16	Entregas	Quantidade de trabalho realizado por um membro da equipe
M17	Defeitos cobertos	Número de defeitos que um membro da equipe deixa passar em um produto lançado em uma Sprint
M18	Desempenho da implantação	Razão entre o número de lançamentos de produtos pelo número de horas por pessoa
M19	Capacidade de trabalho	Horas de trabalho gastas durante uma Sprint para histórias de usuários concluídas
M20	Fator de foco	Proporção entre horas de trabalho gastas e velocidade em horas
M21	Aumento de valor	Relação entre pontos de história de iteração concluídos e os pontos de história médios de toda as interações concluídas

Fonte: Adaptação de Hernández *et al.* (2019)

A revisão realizada por Hernández *et al.* (2019) destaca uma tendência crescente na medição

do valor adicionado às tarefas ou atividades de desenvolvimento de software, principalmente sob a perspectiva da metodologia ágil Scrum. No entanto, a pesquisa também apontou uma lacuna significativa na definição de métricas que abordem diretamente a motivação da equipe e a satisfação do cliente.

Em relação a métricas de produtividade, a pesquisa de Wolter (2022) define uma métrica que mede a produtividade dos engenheiros de *software* utilizando como base a literatura de trabalhos técnicos científicos já realizados. Essa abordagem faz uso de metadados dos repositórios acessíveis publicamente, coletados por meio do *GitHub*, e pode ser aplicada tanto a uma abordagem tradicional de engenharia de *software* quanto a uma abordagem de desenvolvimento adaptada na empresa. Além disso, a funcionalidade dessa abordagem foi demonstrada por meio da análise de um projeto de engenharia de *software*.

No trabalho de Wolter (2022), foram pesquisados e priorizados componentes métricos com o objetivo de encontrar a melhor adequação para a medição de produtividade. Esses componentes incluem:

1. SLOC (*Source Lines of Code*): Que descreve o número de linhas de código que um desenvolvedor adicionou ao *software*.
2. *Commits*: Representando as alterações no código-fonte registradas pelos desenvolvedores.
3. *Bug fixing*: Relacionado à correção de defeitos ou problemas no *software*.
4. *Community Activity*: Englobando a atividade e interação da comunidade de desenvolvedores em torno do projeto.

Os valores de metadados individuais são determinados dividindo o compartilhamento total do componente métrico em diferentes intervalos de valores. Para isso, um valor máximo é definido e as faixas de valores são divididas em partes iguais com base nesse valor (Wolter, 2022). A avaliação do componente métrico é dividida em diferentes faixas de valor, com base em um valor máximo definido pelo usuário. Cada faixa de valor está associada a um resultado de avaliação, calculado dividindo a participação geral do componente métrico pelo mesmo número de faixas de valor. O resultado final para cada componente de metadados não pode exceder a avaliação do componente métrico. Essa fórmula é uma representação geral, e a implementação exata dependerá da métrica específica e das categorias de metadados relevantes para a avaliação da produtividade.

$$\text{Produtividade} = \sum_{n=1}^{componente\ métrico} \quad Avaliação\ do\ Componente\ Métrico_n$$

$$Avaliação\ do\ Componente\ Métrico = \frac{\sum_{n=1}^{\#Componente\ metadados} \quad Componente\ de\ metadados_n}{\#Componentes\ de\ metadados}$$

- n é o número de desenvolvedores avaliados.

- Avaliação do Componente Métrico é determinada com base no desempenho do desenvolvedor nas categorias metadados relevantes.

Para definir métricas de software, muitas empresas optam por aderir às diretrizes da norma ISO/IEC 29110[5]. Essa norma, reconhecida internacionalmente, define um conjunto estruturado de tarefas para a gestão e desenvolvimento de software em pequenas e médias organizações. Seu foco principal é fornecer um modelo adaptável e escalável que pode ser ajustado conforme o tamanho e a complexidade da organização. A norma ISO/IEC 29110 destaca a importância de métricas de software como uma ferramenta essencial para medir e monitorar a qualidade do processo de desenvolvimento. Essa norma propõe algumas métricas que podem ser alinhadas com diretrizes da empresa como: Prazos de entrega; taxas de defeitos; produtividade da equipe; acompanhamento do progresso; e taxa de resolução de problemas.

Jiang et al. (2020) apresenta um estudo empírico sobre métricas de processo relacionadas à prevenção de defeitos ou *bugs* de projetos em evolução. Cinco métricas de processos foram abordados: Números de *committers* distintos; número de revisões; grau de modificação do código; número de linhas modificadas; e o número médio de linhas modificadas. As métricas foram retiradas de 37 versões de 12 projetos de *softwares*.

No estudo realizado por Lima (2015), foi desenvolvido um conjunto de métricas focadas em contribuições, baseadas em evidências empíricas obtidas diretamente de gestores de projetos e equipes. Durante doze semanas, cinco dessas métricas foram coletadas de 48 profissionais de desenvolvimento distribuídos em quatro equipes. Após a coleta, os dados foram apresentados aos gestores de projeto e equipe para determinar a relevância de cada métrica no contexto de avaliação de contribuições. De acordo com Lima (2015), as métricas relacionadas à contribuição de código e à média de complexidade por método receberam *feedbacks* mais positivos durante as avaliações, em contraste com as métricas associadas a *bugs*, que foram recebidas de forma mais cautelosa. O âmbito das métricas analisadas incluíram a contribuição de código, a complexidade média por método, a incidência de *bugs* e o engajamento na correção dessas ocorrências.

Uma das pesquisas de Oliveira (2017) foi conduzida com líderes de equipe e gestores de *software* para explorar, definir, identificar e compreender como eles empregam suas percepções em relação à eficiência de seus desenvolvedores de *software*. Essa investigação foi realizada por meio de entrevistas semiestruturadas, que abordaram os seguintes tópicos: 1. O que é produtividade do desenvolvedor? 2. Como é identificada a produtividade do desenvolvedor? 3. Qual a utilidade das percepções sobre a produtividade do desenvolvedor para a organização de *software*?

[5] Norma ISO/ICE 2910-1:2016: https://www.iso.org/obp/ui/#iso:std:iso-iec:tr:29110:-1:ed-2:v1:en

Os resultados obtidos evidenciaram que a produtividade dos desenvolvedores está intrinsecamente ligada tanto a suas competências técnicas quanto comportamentais. Essa produtividade é medida pela capacidade de concluir tarefas de forma tempestiva, mantendo um padrão de qualidade adequado e atendendo às expectativas das partes interessadas, tudo isso sem a necessidade de retrabalho. Essas percepções, fornecidas pelos gestores de *software*, desempenham um papel crucial na administração eficaz dos projetos de *software* e na tomada de decisões, auxiliando na alocação de recursos de maneira mais eficiente e na identificação de áreas que necessitam de melhoria.

A relação entre o cumprimento de prazos nas atividades e a produtividade emergiu como uma das conclusões mais recorrentes dentre as respostas dos gerentes de *software* que participaram das entrevistas. A finalização de tarefas sem a necessidade de retrabalho foi destacada como uma prática valorizada por toda a equipe. Essa dinâmica mais eficiente possibilita uma alocação mais estratégica dos desenvolvedores, direcionando-os para tarefas críticas e projetos estratégicos. Nesta pesquisa conduzida por Oliveira (2017), foram também investigadas métricas de produtividade, juntamente com a perspectiva dos desenvolvedores em relação à percepção das organizações de *software*. A Tabela 2 apresenta um conjunto de métricas que auxiliam equipes de projetos a tomarem decisões na elaboração de projetos de *software*, com foco na produtividade.

Tabela 2 - Métricas de produtividade do desenvolvedor

Grupo	Métrica	Descrição
Métricas baseadas em Código	*CodeOwned*/*Time*	Número de arquivos de código fonte de propriedade do desenvolvedor pelo tempo de projeto
	SLOC/Tempo	Quantidade de linhas de código fonte de propriedade do desenvolvedor pelo tempo de projeto
	HalsteadEff/Time	Esforço de Halstead [6]dos arquivos de código fonte de propriedade do desenvolvedor pelo tempo de projeto
Métricas baseadas em *Commits*	*Commits*/Tempo	Número de *commits* realizados pelo desenvolvedor pelo tempo de projeto
	CommitsLM/Tempo	Quantidade de linhas modificadas por *commit* do

[6] O esforço de Halstead refere-se a uma medida de complexidade do software proposta por Maurice H. Halstead.

		desenvolvedor pelo tempo de projeto
	CommitsED/Tempo	Distância de edição de Levenshtein [7]dos *commits* realizados pelo desenvolvedor pelo tempo de projeto

Fonte: Oliveira (2017)

Ainda no estudo conduzido por Oliveira (2017), uma busca na literatura foi realizada com o propósito de identificar os fatores mais influentes na produtividade dos desenvolvedores em organizações de *software*. Dentre os fatores técnicos, humanos e organizacionais analisados, a métrica de produtividade mais frequentemente empregada para avaliar os desenvolvedores é a relação entre a quantidade de linhas de código e o tempo (LOC/Tempo).

As métricas mais utilizadas incluem a relação entre a quantidade de tarefas executadas e o tempo (Tarefas/Tempo) e o esforço de Halstead (Esforço de Halstead dos arquivos de código fonte de propriedade do desenvolvedor pelo tempo de projeto) em relação ao tempo (Halstead/Tempo). Essas métricas são utilizadas para avaliar a eficiência e a eficácia dos desenvolvedores na produção de *software* em um período determinado. A Tabela 3 apresenta algumas das métricas exploradas na pesquisa de Oliveira (2017) que foram utilizadas no desenvolvimento da Plataforma GMC.

Tabela 3 - Métricas e fórmulas exploradas

Métrica	Fórmula
Taxa de produção - TP	$TP = \frac{quantidade\ de\ solicitações\ atendidas}{tempo\ de\ resolução}$
Taxa de código - TC	$TC = \frac{quantidade\ de\ arquivos\ alterados}{tempo\ de\ resolução}$
Taxa de linhas alteradas - TL	$TL = \frac{quantidade\ de\ linhas\ alteradas}{tempo\ de\ projeto}$
Taxa de *commits* realizados - TCR	$TCR = \frac{número\ de\ commits\ realizados}{tempo\ de\ projeto}$

Fonte: Adaptação de Oliveira (2017)

1.2. Extração de Métricas do GitHub

Várias ferramentas ajudam a controlar o histórico de dados do projeto de *software* para que seja possível extrair métricas. Uma delas é o *Git*, uma ferramenta de controle de versão que registra alterações em um ou vários arquivos ao longo do tempo, podendo ser recuperados a qualquer momento (CHACON, 2022). Além disso, o *GitHub,* serviço de hospedagem em nuvem que fornece um sistema de controle de versão (VCS - *Version Control System*) baseado no *Git*. O *GitHub* permite que os

[7] Distância de Levenshtein: É uma medida que avalia a diferença ou similaridade entre dois conjuntos de código-fonte, como, por exemplo, duas versões de um arquivo.

desenvolvedores trabalhem juntos em projetos compartilhados e façam alterações que possibilitam a edição simultânea de arquivos por diversas pessoas de forma pública ou privada. Isso encoraja a colaboração em tempo real, permitindo que equipes trabalhem juntas para criar e editar conteúdo de um *software* (GITHUB, 2023).

Werner *et al.* (2022) define *GitHub* como uma plataforma de hospedagem de código que utiliza o sistema de controle de versão *Git* para gerenciar alterações em projetos colaborativos. Essa plataforma possibilita a criação de repositórios públicos ou privados e oferece uma interface *web* para que desenvolvedores possam compartilhar o código e trabalhar em conjunto de maneira segura, simplificando assim a colaboração ao permitir que os desenvolvedores revisem e discutam as alterações, revisem o histórico de alterações e problemas e tarefas do projeto.

Além de armazenamento de arquivos, o *GitHub* permite através de comandos interagir com os arquivos armazenados, verificando versões, quem alterou, o que foi alterado, onde cada texto ou código foi modificado. Toda configuração e funcionamento é documentada na página *web* do GITHUB (2023), cada ramificação em um projeto, conhecida como "*branch*", proporciona a oportunidade para cada colaborador direcionar seus esforços de desenvolvimento de maneira independente. Posteriormente, é possível realizar a fusão, também denominada "*merge*", para unificar os arquivos ou códigos desenvolvidos por diferentes pessoas.

Embora o *GitHub* seja uma plataforma colaborativa *open source*, aberta a contribuições de qualquer indivíduo, é altamente recomendável e possível ativar um *token* de autenticação em substituição ao acesso por meio de *login* e senha. Esta medida visa resguardar informações pessoais e proteger alterações no código disponibilizado. Na documentação mais recente do GITHUB (2023), encontra-se um guia detalhado para a obtenção do primeiro acesso e a configuração de um *token* de segurança para identificação no repositório.

O uso de ferramentas visuais para tomada de decisão é comum no processo de desenvolvimento de *software*. Na pesquisa de Venigalla *et al.* (2021) foi construído um *plugin* de navegador web para exibir informações essenciais de forma visual sobre um projeto *GitHub*, sendo dividido em duas categorias; métricas de código fonte e métricas de manutenção do projeto. Este *plugin* foi avaliado por 2 pesquisadores, 3 estudantes de graduação e 10 desenvolvedores de *software* profissionais. A partir da pesquisa, descobriram que a maioria dos avaliadores concordam que a ferramenta aumenta a produtividade e recomendam o uso.

A Figura 4 apresenta algumas métricas derivadas do repositório GitHub-ReactiveX/RxJava6[8]. Estas englobam indicadores de manutenção como: taxa de autores ativos, proporção de *bugs* reportados em relação aos resolvidos, e a magnitude do impacto dos *commits* em termos de arquivos e linhas de código (VENIGALLA et al.,2021). Tais informações reforçam a importância de métricas de processo

[8] Reactive Extensions for the JVM: https://github.com/ReactiveX/RxJava

e projeto de software e assim fornecem *insights* sobre a participação contínua dos colaboradores, a eficácia na resolução de problemas e a profundidade das contribuições em *commits*.

O *GitQ* construído na pesquisa de Venigalla et al. (2021) retrata algumas das informações complexas do projeto, como um *feed* visual para ajudar os desenvolvedores a investigar melhor os projetos para contribuição (VENIGALLA et al.,2021). Durante o período desta pesquisa, identificou-se a indisponibilidade de acesso a alguns servidores essenciais para o funcionamento do *GitQ*. Em virtude dessa limitação, a realização de testes em outros repositórios tornou-se inviável, impossibilitando, assim, a apresentação abrangente dos resultados obtidos.

Figura 4 - Relatório de avaliação gerado pelo *GitQ*

Fonte: Venigalla et al. (2021)

Uma alternativa para criar gráficos e visualizações de dados do *GitHub* é utilizando o motor de busca Pharo com a API (*Application Programming Interface*) Roassal. Bergel (2022) descreve como é possível criar visualizações de informações dos repositórios do *GitHub* através do *GitHub Actions*. Gráficos são gerados por meio de *scripts* que utilizam o mecanismo Roassal, o qual funciona na linguagem de programação Pharo. Roassal proporciona uma API e mapeia objetos específicos de domínio arbitrários para elementos visuais.

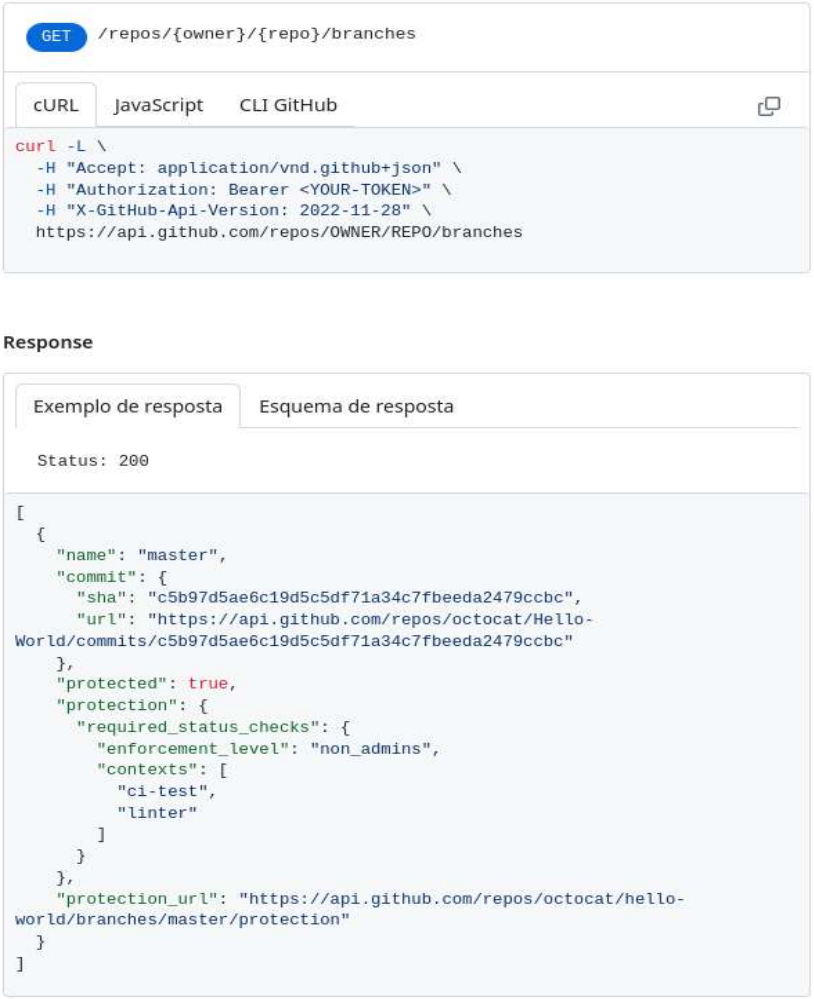
Utilizando o *GitHub Actions* [9], que é uma ferramenta que o *GitHub* oferece para automatizar uma variedade de atividades, como por exemplo analisar, testar código, implantar aplicativos, entre outras. Esses fluxos de trabalho são definidos em arquivos YAML dentro do próprio repositório e são acionados por eventos como *push* de código, criação de *pull requests*, criação de releases, entre outros. Branco (2022) Analisa a influência por meio da mineração dos dados dos repositórios mais populares e da execução da ferramenta de coleta de métricas de qualidade de código.

O *GitHub* disponibiliza também uma API com recursos que podem ser acessados para cada projeto. De acordo com De Brito *et al.* (2021), sistemas podem ser elaborados a partir dessa API e disponibilizados para gerentes de projeto, tendo sido uma ferramenta com base no *GitHub* para prover e melhorar o ensino no processo de aprendizado nas disciplinas de programação, além de disponibilizar métricas relacionadas às contribuições dos alunos.

O *GitHub* oferece duas interfaces programáticas distintas: a API REST e a API GraphQL. Estas APIs podem ser integradas por meio de ferramentas como o *GitHub* CLI, utilitários de linha de comando como o *Curl*[10], bibliotecas oficiais denominadas Octokit e também bibliotecas de conjunto de código desenvolvidos por outros colaboradores que não fazem parte do projeto. A Figura 5 exibe um exemplo de código para listar *branches* através da API do GitHub. É importante destacar que certos recursos podem estar disponíveis em uma API e ausentes na outra. A decisão de qual API utilizar deve ser fundamentada nas demandas específicas do projeto e na habilidade do desenvolvedor. Adicionalmente, as IDs de nó (Identificadores únicos atribuídos a cada nó no grafo de dados) fornecidas permitem uma transição fluida entre a API REST e a API *GraphQL* (GITHUB REST API, 2023).

[9] Documentação do GitHub Actions: https://docs.github.com/pt/actions
[10] Linha de commandos Curl: https://curl.se/

Figura 5 - Exemplo de código para listar *branches*



Fonte: GITHUB REST API (2023)

Os comandos exibidos na Figura 5 fazem uma chamada para a API do GitHub para recuperar informações sobre as *branches* de um repositório específico. Para obter uma compreensão mais detalhada, a lista a seguir descreve as finalidades específicas de cada comando conforme documentação do Curl:

- *curl*: É um comando utilizado para transferir dados com URLs.
- -L: Instrui o *curl* a seguir redirecionamentos se a resposta da API retornar um código de status 3xx.
- -H: Especifica cabeçalhos HTTP adicionais para incluir na solicitação.
- "*Accept: application/vnd.github+json*": Indica que o cliente aceita uma resposta no formato JSON da API do *GitHub*.
- "*Authorization: Bearer* <YOUR-TOKEN>": Usa um *token* de acesso para autenticar a solicitação. Esse *token* geralmente é associado a um usuário e fornece as permissões necessárias

para acessar o repositório.

- "*X-GitHub-Api-Version*: 2022-11-28": Especifica a versão da API do *GitHub* que deve ser utilizada. Neste caso, está definida como "2022-11-28".

- https://api.github.com/repos/OWNER/REPO/branches: A URL da API que está sendo acessada. O "OWNER" é nome do proprietário do repositório e "REPO" é nome do repositório em questão.

No contexto desta pesquisa, a API REST do GITHUB (2023) foi integrada por meio do *software* GDDownloader[11]. Todos os procedimentos para invocação de serviços da API foram encapsulados nessa ferramenta. Assim, o GDDownloader gerenciou as solicitações e a segurança à API REST do *GitHub*, abrangendo entidades como *commits*, comentários vinculados aos *commits*, *issues*, comentários associados às *issues* e respectivos eventos das *issues*.

[11] GDDownloader: https://github.com/thdiaman/GDDownloader

2. TRABALHOS RELACIONADOS

Este capítulo desempenha um papel fundamental ao fornecer o embasamento teórico essencial para o desenvolvimento deste trabalho. Abrangendo investigações anteriores sobre a extração de dados e sua transformação em métricas no contexto do desenvolvimento de *software*, busca-se uma compreensão abrangente das abordagens teóricas existentes. Não apenas situando a pesquisa atual em um panorama mais amplo, mas também destacando a evolução do conhecimento no campo, este capítulo estabelece um fundamento sólido para a presente investigação, destacando tendências, lacunas e contribuições únicas.

O trabalho de Lima Ramos (2022) concentrou-se em uma revisão sistemática da literatura, explorando as estratégias empregadas na extração e disponibilização de métricas de qualidade e produtividade em projetos ágeis de *software*. Posteriormente, o estudo desenvolveu uma arquitetura de *software* focalizada na extração de dados das ferramentas *Open Project* e *TestLink*, utilizadas no âmbito do Laboratório Assert para a gestão de projetos. Essa arquitetura foi projetada para simplificar a análise precisa dos dados, resultando na obtenção de métricas de qualidade e produtividade específicas utilizando dados de *bugs* e testes de *software*.

A pesquisa também formalizou um fluxo de trabalho que orienta a extração e consolidação das métricas desejadas, permitindo a progressão da proposta arquitetural para incorporar outras fontes de dados, como Jira, *GitLab* e *GitHub*. Em seguida, uma Plataforma de Métricas foi implementada como prova de conceito, utilizando o ecossistema do Laboratório Assert como ambiente de experimentação. Na última fase do estudo, a Plataforma *Assert Metrics Panel* (AMP) foi implantada, validando a solução em um projeto real de PD&I desenvolvido no Laboratório Assert.

A abordagem de métricas na Plataforma AMP, na pesquisa De Lima Ramos (2022), difere da abordagem proposta nesta pesquisa. A Plataforma AMP se concentrou na extração de dados do *OpenProject* e do *TestLink* e exibe métricas relacionadas a *bugs*, testes e tempo gasto para executá-los. Enquanto a Plataforma GMC, proposta neste trabalho, extrai dados de processo e projeto de *software* armazenados no repositório *GitHub*. Além disso, a extração feita na Plataforma GMC tem como objetivo capturar e transformar em métricas, alguns eventos que ocorrem na linha do tempo do projeto de *software*, como por exemplo, *issues e commits*.

A Diamantopoulos *et al.* (2020) construíram uma plataforma que extraído do *GitHub*, dados relacionados a contribuições dos projetos e transforma em métricas de qualidade que podem ser usadas no processo de desenvolvimento de *software*. A pesquisa obteve dados de 3.000 (três mil) projetos publicados na plataforma *GitHub* de forma livre, com diferentes tamanho e complexidade. Entretanto, contempla apenas projetos desenvolvidos na linguagem de programação Java. Ainda na pesquisa de Diamantopoulos *et al.* (2020), a análise das métricas foi possível devido a criação de um programa de computador na linguagem Python para extração de informações de vários projetos de *software* e a

construção de novos conjuntos de dados. Este programa de extração e mineração de dados de Diamantopoulos *et al.* (2020) contribuiu para a construção da base de dados da Plataforma GMC, desenvolvida no escopo desta dissertação.

Como resultado da pesquisa, Diamantopoulos et al. (2020) obtiveram 25 métricas que quantificam as contribuições da equipe de projeto de *software*, desempenho de projeto e relacionada a código fonte. As métricas abordadas descrevem o perfil de cada colaborador, tanto em termos de desenvolvimento como de atividades realizadas nos projetos. Os conjuntos de dados processados nessa extração estão disponíveis na Internet possibilitando uma análise prática e objetiva para novas implementações.

O processo de extração de dados do *GitHub* da Plataforma GMC utiliza como base, o *software* desenvolvido pelos pesquisadores em Diamantopoulos et al. (2020). Contudo, foram introduzidas modificações no código-fonte (Disponível no repositório adaptacao_gddownloader[12]) para viabilizar a integração do extrator com os sistemas da Plataforma GMC, os quais adotam a linguagem de programação Java e o *framework Spring Boot*. Essas implementações adicionais incluíram o desenvolvimento de uma API para facilitar a comunicação entre o GDDownloader e a camada de negócios, proporcionando assim a interoperabilidade entre aplicações com diferentes linguagens de programação.

Além disso, a Plataforma GMC desenvolvida no escopo desta pesquisa tem como propósito construir uma representação visual das métricas e oferecer uma alternativa aos envolvidos no projeto na tomada de decisões em projetos de *software*. O propósito principal consiste na melhoria técnica da integração da ferramenta GDDownloader com outros sistemas, por meio do desenvolvimento de uma API dedicada. Adicionalmente, busca-se a criação de painéis visuais na plataforma GMC que facilitem a visualização de novas métricas relacionadas a projetos e processos de *software*.

Um estudo de caso definido por Zabardast et al. (2022) foi realizado como parte de uma colaboração contínua com um parceiro industrial. Este estudo de caso teve dois principais objetivos. O primeiro deles consistiu em analisar o processo de desenvolvimento de *software* com a finalidade de quantificar as contribuições dos desenvolvedores e identificar potenciais discrepâncias entre a propriedade e a contribuição no âmbito do projeto.

Neste estudo, foi possível desenvolver um modelo de alinhamento entre propriedade e contribuição, denominado OCAM (*Ownership and Contribution Alignment Model*). O objetivo primordial desse modelo consistiu em calcular a proporção da contribuição das equipes em relação aos módulos de *softwares*. Dado que a contribuição pode ser originária de diversas fontes, o OCAM leva em consideração as contribuições relacionadas à produção de código, à abertura de *tickets* e às revisões de código.

[12] Adaptação do GDDownloader: https://github.com/matiasribeiro/adaptacao_gddownloader

O modelo OCAM classifica os contribuintes de forma a proporcionar uma visão abrangente do alinhamento entre propriedade e contribuição no contexto do desenvolvimento de *software*. Por fim, a contribuição é um fenômeno multidimensional que não pode ser representado por métricas individuais. Foi viável avaliar o nível de contribuição aos componentes utilizando métricas já existentes que foram coletadas durante o processo de desenvolvimento, sem a necessidade de impor uma carga adicional às organizações por meio da instalação de ferramentas adicionais.

As métricas incorporadas na construção do modelo OCAM de Zabardast et al. (2022), tais como o número de *commits*, número de tickets abertos, número de solicitações de *pull* e a complexidade das solicitações de *pull*, serão utilizadas neste estudo. O objetivo é introduzir novas abordagens de visualização dessas métricas, proporcionando alternativas que permitam a todos os envolvidos no projeto acompanhar de maneira mais eficiente as diferentes fases do desenvolvimento do *software*.

Este estudo marca uma progressão em relação às pesquisas conduzidas por Zabardast et al. (2022) no domínio da abordagem de métricas de processo e projeto de *software*. A pesquisa proposta neste trabalho concentra-se na apresentação de métricas com o propósito de auxiliar no processo decisório, dando origem à plataforma *web* GMC. Essa plataforma foi concebida para oferecer suporte a todas as pessoas da equipe de *software* envolvidas no projeto. Com a finalidade principal de facilitar a tomada de decisão em projetos, a plataforma proporciona uma representação de novas métricas sugeridas e de métricas propostas na literatura técnica.

Já a pesquisa conduzida por MEJÍA *et al.* (2021) concentra-se em dois grupos gerais de atividades: o Gerenciamento de Projetos (GP) e o Processo de Implantação de *Software* (SI). Dentro do grupo de atividades de GP, são consideradas as seguintes etapas: planejamento, execução do plano do projeto, avaliação e controle de projetos e encerramento. Já no grupo de atividades de SI, as etapas envolvem o início da implementação, análise de requisitos, arquitetura e design detalhado, construção, integração e entrega do produto.

Foram examinados os dados de tempo e custo em cada grupo, priorizando a relação entre as estimativas e os resultados reais das atividades, conforme preconizado pela norma ISO/IEC 29110. As métricas foram desdobradas nos níveis de gerência, liderança, equipe do projeto e no contexto global do projeto, proporcionando uma avaliação minuciosa das distintas dimensões temporais e custos associados às atividades desenvolvidas.

A tabela 4 apresenta as métricas abordadas pelos autores, diretamente relacionadas ao trabalho conduzido nesta pesquisa na plataforma GMC. As métricas destacadas na referida tabela foram reunidas para aplicação em novos contextos de visualização. A plataforma GMC não apenas incorpora métricas já propostas na literatura técnica, mas também introduz métricas sugeridas por este estudo como; métricas de produtividade. Dentro dessa abordagem, a ferramenta GDDownloader é adaptada para aprimorar a visualização dessas métricas, atendendo às demandas específicas de diferentes sistemas e plataformas. Este esforço evidencia a importância da adaptação de ferramentas de *software*

existentes para melhor atender a contextos variados e às necessidades específicas de cada cenário.

Tabela 4 - Relação de métricas abordadas por autores

Autor (ano)	Descrição das Métricas
De Lima Ramos (2022)	Quantidade de *bugs* registrados
	Quantidade de planos de testes
	Total de testes executados
	Tempo gasto em testes
	Colaboradores que mais registraram *bugs* e testes no projeto
	Módulos com mais *bugs* e testes registrados
Diamantopoulos et al. (2020)	Média, tempo e mudança de código do sistema desenvolvido
	Atividades abertas e fechadas
	Média de problemas, alterações e atividades registradas
	total de arquivos modificados
	alterações por período no dia
	Total de linhas de código alteradas
	Períodos de atividades dos colaboradores
MEJÍA et al. (2021)	Tempo gasto na etapa da implementação
	Tarefas repetidas
	Tempo gasto por atividades
	Tempo gasto por colaboradores
	Média de tempo de atividades
	Níveis de métrica de gerente, líder e equipe geral do projeto
Zabardast et al. (2022)	Quantidade de mudança de código
	Complexidade do código
	Número de atividades
	Número de alterações

Este estudo representa um avanço em relação às pesquisas correlatas, ao aprofundar a exploração e consolidação das métricas investigadas, além de introduzir novas perspectivas de visualização como indicadores, tabelas e gráficos para os participantes envolvidos em projetos de *software*. Ao investigar e agrupar métricas relacionadas ao projeto e ao processo de desenvolvimento de *software*, esta pesquisa busca proporcionar auxílio às equipes de desenvolvimento, capacitando-as a conduzir de maneira mais eficiente suas atividades e a gerenciar projetos em tempo real. Isso facilita a interpretação de eventos que possam ocorrer durante o desenvolvimento.

3. PLATAFORMA PARA GERENCIAMENTO DE MÉTRICAS – GMC

Neste capítulo, é delineada a definição da abordagem GMC, oferecendo uma compreensão de como foi concebida e contextualizando o problema em questão. Ao explorar a evolução da abordagem GMC, partimos da identificação de desafios nas práticas atuais de utilização de métricas no desenvolvimento de *software*, apresentando, ao final, uma proposta para aprimorar a gestão dessas métricas.

Essa proposta está elaborada com a criação de uma plataforma voltada ao gerenciamento de métricas de processos e projetos de *software*. Nos capítulos subsequentes, será detalhada a construção da arquitetura GMC, incluindo uma explicação minuciosa de cada etapa realizada para explorar dados e transformá-los em métricas acessíveis à equipe de desenvolvimento de *software*.

3.1. Motivação para Abordagem GMC

Sommerville (2028) exemplifica que diversas empresas acumulam dados sobre seu software, abrangendo desde a quantidade de pedidos de alteração de requisitos até o número de defeitos identificados nos testes. Contudo, permanece incerto se essas métricas são empregadas de maneira sistemática para a comparação entre atributos do software e processos utilizados, ou para avaliar os efeitos de alterações em processos e ferramentas de software. Uma considerável parcela das pesquisas sobre métricas de software concentrou-se em métricas fundamentadas em código e em processos de desenvolvimento orientados a planos. Contudo, nos dias de hoje, observa-se um aumento significativo no desenvolvimento de software por meio de práticas como reuso e configuração de aplicações existentes, assim como a adoção crescente de métodos ágeis.

Embora as pesquisas na literatura dos últimos cinco anos investiguem métricas de software, conforme já abordado neste trabalho, persistem incertezas sobre quais métricas podem oferecer contribuições substanciais ao projeto de desenvolvimento de software. Quais métricas devem ser destacadas? De que maneira as exibir? Ainda é prático adaptar dados para apresentar métricas e assegurar a usabilidade entre a equipe de desenvolvimento? Essas questões continuam a demandar uma abordagem detalhada e esclarecedora.

O GitHub (2023) disponibiliza uma API REST para acessar métricas com base nos dados presentes nos repositórios do GitHub. Cada requisição à API REST inclui um método HTTP e um caminho específico. A depender do ponto de extremidade da API REST, é possível que seja necessário detalhar cabeçalhos de requisição, informações de autenticação, parâmetros de consulta ou parâmetros de corpo. A documentação de referência da API REST fornece detalhes sobre o método HTTP, o caminho e os parâmetros associados a cada ponto de extremidade. Além disso, apresenta exemplos elucidativos de solicitações e respostas para cada ponto específico.

Apesar do GitHub oferecer uma API para acessar métricas, ainda demanda um esforço considerável de desenvolvimento para implementar, configurar e manter um sistema capaz de consumir esses serviços. A plataforma de gerenciamento de métricas GMC surge como uma alternativa às opções nativas do GitHub, permitindo a extração e transformação dos dados do repositório armazenados na plataforma em métricas de processo e projeto de software. A plataforma GMC diferencia-se pela capacidade de realizar essas tarefas sem a necessidade de codificação extensiva ou alocação significativa de tempo para configurar ferramentas de suporte. A plataforma GMC extrai, processa e apresenta métricas de maneira prática e instantânea, proporcionando uma interpretação facilitada e agilizando o processo de tomada de decisão dentro do projeto de software.

Ao explorarmos as métricas de processo e projeto de software, a abordagem GMC visa não apenas identificar as métricas relevantes na literatura, mas também construir um ambiente prático e de fácil utilização. Buscamos agilizar o processo de tomada de decisão para toda a equipe envolvida no projeto, proporcionando uma ferramenta que não apenas compile, mas interprete e apresente de maneira acessível as métricas essenciais. A construção desse ambiente busca não somente reunir dados, mas transformá-los em informações valiosas, facilitando assim uma interpretação mais eficiente e contribuindo para decisões mais informadas e estratégicas no contexto do desenvolvimento de software.

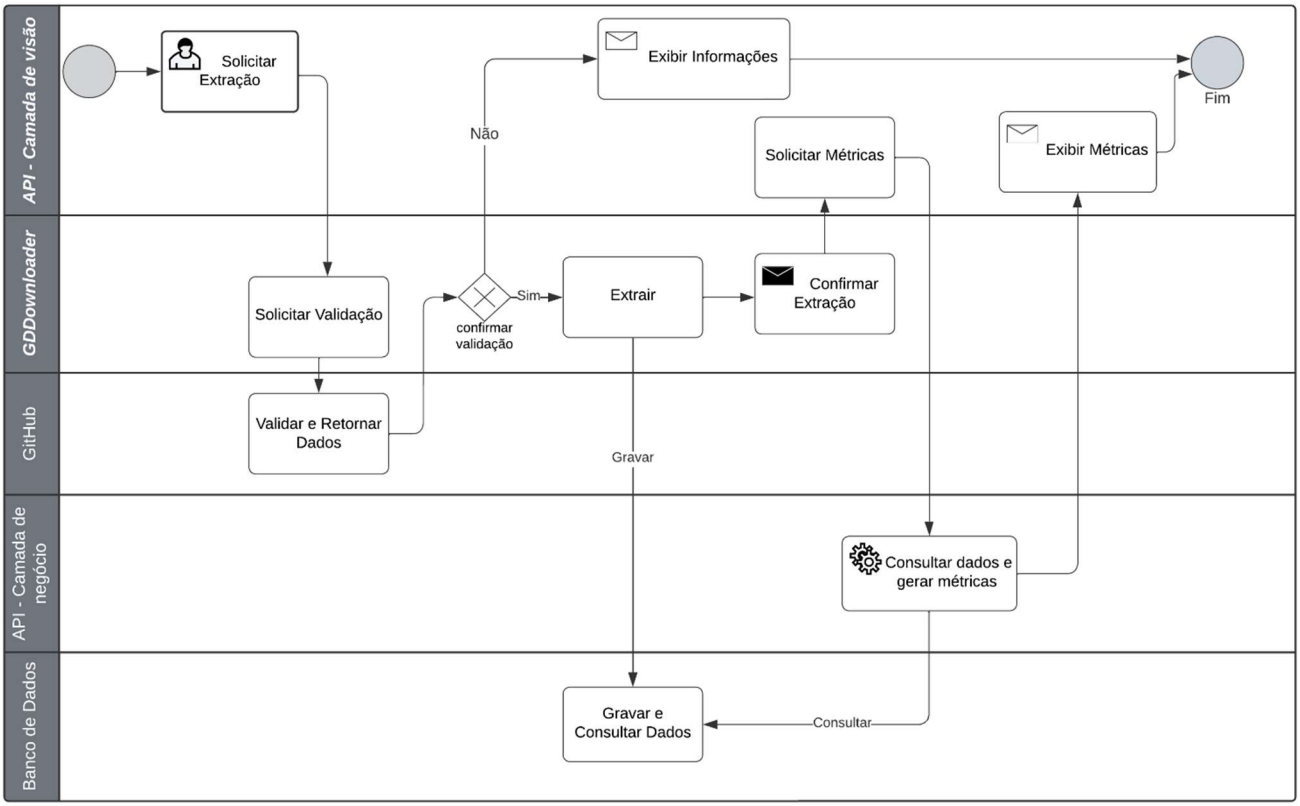
3.2. Arquitetura da Plataforma GMC

A Plataforma GMC visa disponibilizar uma solução alternativa em comparação a API de métricas do GitHub para a disponibilização de métricas exploradas do repositório *GitHub*. A GMC constitui-se como um conjunto integrado de sistemas e ferramentas que operam de forma coordenada, visando a extração de dados, a geração de métricas relacionadas a projetos e processos de *software*, e a apresentação dessas métricas por meio de uma página web integrada à referida plataforma.

Cada módulo do sistema é individualmente estruturado para facilitar a comunicação por meio de APIs, conferindo ao sistema flexibilidade para incorporar novas implementações em diversas camadas, seja na camada de dados, na camada de negócios ou na camada de visualização. Essa abordagem modular e orientada à API contribui para a adaptabilidade do sistema, permitindo a incorporação de atualizações e a expansão das funcionalidades conforme necessário.

No desenvolvimento desta pesquisa, foi essencial estabelecer a comunicação entre os sistemas criados e as ferramentas empregadas, sendo essa interação completamente efetuada por meio de APIs. O fluxo arquitetural, como ilustrado na Figura 6, tem início com a solicitação de extração efetuada pelo usuário no sistema API-Camada de visão. Posteriormente, ocorre a validação por meio do *token* de acesso, seguida pela extração, gravação na base de dados e disponibilização das métricas.

Figura 6 - Fluxograma de solicitação de extração de métricas da plataforma GMC



Fonte: Própria

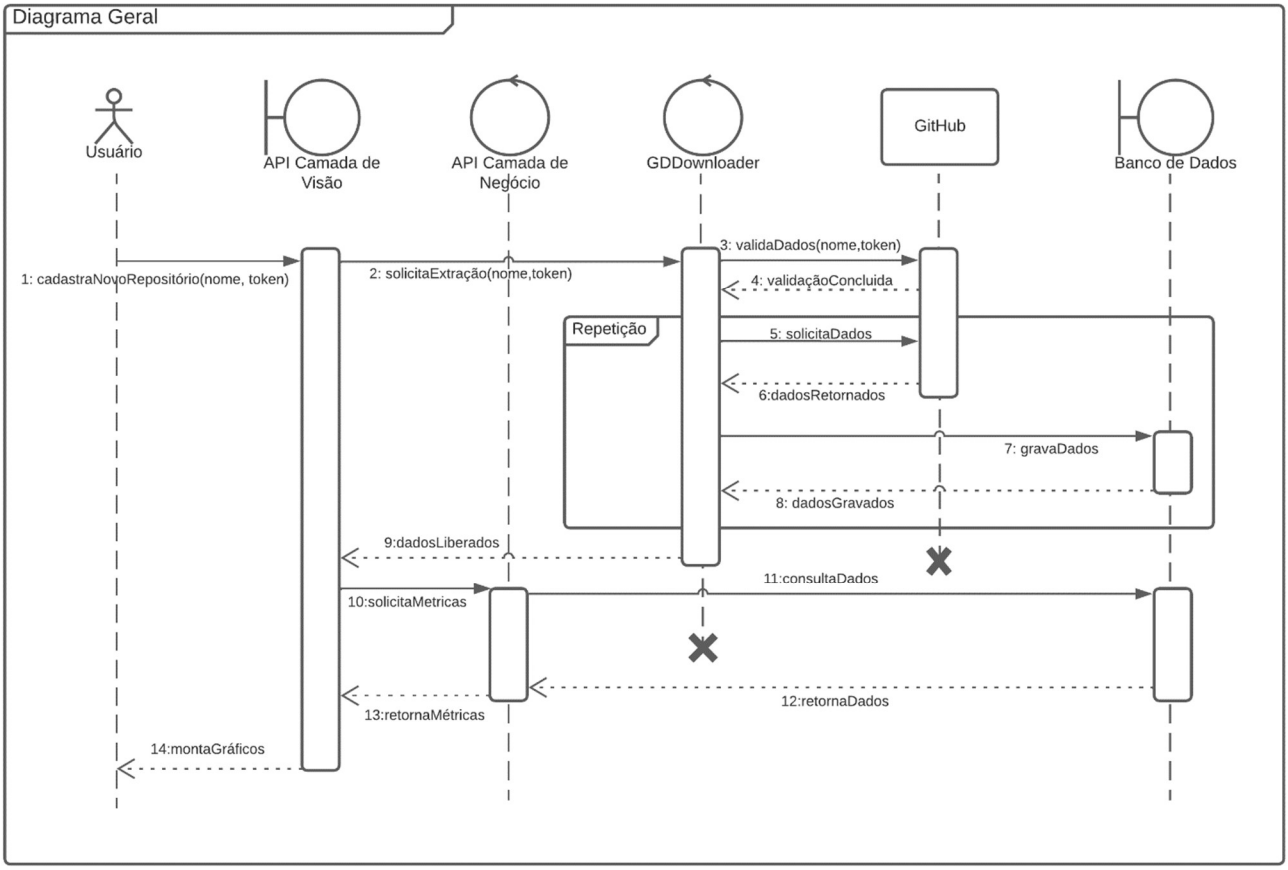
A Figura 7 destaca a divisão modular por meio de faixas no fluxograma. A ferramenta GDDownloader, originada na pesquisa de Diamantopoulos et al. (2020), foi adaptada para facilitar a comunicação entre os sistemas das camadas de visão e de negócio. É relevante notar que a ferramenta GDDownloader, descrita como um componente do sistema, é desprovida de uma interface voltada para o usuário. Sua concepção visa operar como um sistema em segundo plano, ou seja, de maneira não interativa e imperceptível ao usuário final.

Para proporcionar uma compreensão mais detalhada do funcionamento da plataforma, a Figura 9 apresenta o diagrama geral de sequência referente ao comportamento da plataforma GMC quando o usuário adiciona um novo repositório do *GitHub* e solicita a visualização das métricas. O processo tem início na camada de visão, que inicializa a solicitação e instrui ao GDDownloader para extrair os dados. Após a conclusão da validação do *token*, a ferramenta GDDownloader requisita os dados ao repositório remoto do *GitHub*. Este processo continua em um ciclo até que todos os dados autorizados sejam completamente extraídos e armazenados no banco de dados.

Imediatamente após a conclusão da gravação, uma mensagem autorizando o acesso aos dados é enviada a camada de visão. Este, por sua vez, encaminha uma requisição HTTP através da API para a aplicação da camada de negócio. A aplicação da camada de negócio executa as consultas para geração de métricas utilizando os dados recuperados do banco de dados *NoSQL*. Com todas as etapas de processamento de informações concluídas, as métricas são apresentadas na aplicação de visão por meio

de indicadores, gráficos e tabelas. Este processo é executado em segundos, mesmo diante de grandes projetos, garantindo uma resposta praticamente instantânea e tornando a visualização das métricas transparentes para os usuários.

Figura 7 - Diagrama de sequência da plataforma GMC



Fonte: Própria

3.3. Etapa 1: Pesquisa e Análise Exploratória

Como parte desta etapa, inicialmente foi realizada uma pesquisa exploratória com o objetivo de investigar na literatura as métricas utilizadas em projetos de *software*. Utilizando informações sobre métricas de processo e projeto de *software* extraídas do GitHub, foram realizadas investigações diretas em repositórios de trabalhos científicos, conforme Tabela 5. O objetivo era obter uma visão abrangente e compreender como essas métricas são abordadas nos estudos científicos.

Tabela 5 - Fontes de investigações diretas

Descrição
IEEE (http://ieeexplore.ieee.org/Xplore/)
ACM (http://portal.acm.org/)
Google Acadêmico (https://scholar.google.com/)
Capes Periódicos (https://www-periodicos-capes-gov-br.ezl.periodicos.capes.gov.br/)

Fonte: Própria

A pesquisa concentrou-se principalmente em estudos que tratavam das métricas de processo e projeto de *software* no intervalo de 2017 a 2023, especialmente aqueles que extraiam dados do *GitHub* para convertê-los em métricas. Métricas de processo e projeto de *software* que incluem; Informações sobre andamento do projeto, métricas relativas a tempo de investigação, tempo de conclusão, desempenho dos colaboradores, gerenciamento das atividades durante o projeto de *software* e suas interconexões (SOMMERVILLE, 2018).

O propósito era compreender a maneira como os colaboradores desses projetos acompanhavam essas métricas. O critério de exclusão estabelecido visava evitar a inclusão de trabalhos que se concentrassem exclusivamente em métricas de produto de *software* ou métricas relativas a hardware. Buscou-se identificar padrões de utilização e analisar como essas métricas podem contribuir positivamente para melhorar os processos, ajudar os colaboradores e aprimorar do desenvolvimento de *software*.

A partir da análise exploratória, os projetos listados na Tabela 6 foram cuidadosamente selecionados com base em trabalhos científicos que abordam métricas de *software*, incluindo métricas tanto de processo quanto de projeto de *software*. As pesquisas foram revisadas para identificar as métricas de processo e projeto de *software* mais amplamente empregadas, as quais foram detalhadamente relatadas nas investigações científicas. O resumo dos títulos dos trabalhos identificados está apresentado na Tabela 6, organizados por autor, ano e título.

Tabela 6 - Trabalhos científicos para análise de métricas investigadas

(Autor, Ano)	Título
(DE BASSI, 2018)	Medindo a contribuição dos desenvolvedores no código-fonte usando métricas de qualidade
(BRANCO, 2022)	Análise da Influência do *GitHub Actions* sobre a Qualidade do Código
(CHACON, 2022)	Tudo que você precisa saber sobre *Git*
(DE BRITO, 2021)	Uma ferramenta de acompanhamento em tempo real da contribuição individual de alunos de cursos de computação no desenvolvimento de projetos de *software* acadêmicos hospedados no *GitHub*
(DIAMANTOPOULOS, 2021)	Empregando Métricas de Contribuição e Qualidade para Quantificar o Processo de Desenvolvimento de *Software*
(DE LIMA RAMOS, 2022)	Monitoramento de métricas de qualidade e produtividade em projetos ágeis de *software* através da integração de dados extraídos de ferramentas de gestão e testes

(FENTON, 2014)	Métricas de *software*: uma abordagem rigorosa e prática
(JIANG et al., 2020)	Quais métricas de processo são significativamente importantes para a mudança de defeitos em projetos em evolução: um estudo empírico
(KOZIK et al., 2018)	*Framework Q-Rapids* para análise avançada de dados para melhorar o desenvolvimento rápido de *software*
(LIMA, 2015)	Avaliação da contribuição de desenvolvedores com métricas baseadas em mineração de repositório
(HERNÁNDEZ, 2021)	Métricas de produtividade para equipes de trabalho ágeis de desenvolvimento de *software*: uma revisão sistemática
(MEJÍA et al., 2021)	Uma proposta de métricas para desenvolvimento de *software* baseada na ISO/IEC 29110
(MLADENOVA, 2020)	Métricas de Qualidade de *Software* – Pesquisa, Análise e Recomendação. In: Conferência Internacional de Automação e Informática 2020 (ICAI)
(NICOLETTE, 2015)	Métricas de Desenvolvimento de *Software*
(OLIVEIRA, 2017)	Fatores de influência na produtividade de desenvolvedores de organizações de *software*
(VENIGALLA et al., 2021)	*GitQ*- Rumo ao uso de emblemas como dicas visuais para projetos *GitHub*. Laboratório de Pesquisa em *Software* Inteligente e Análise Humana (RISHA)
(WOLTER, 2022)	Medindo e avaliando uma métrica para desempenho de engenharia de *software*
(ZABARDAST et al., 2022)	Propriedade versos contribuição: Investigando o alinhamento entre propriedade e contribuição

Fonte: Própria

No âmbito da engenharia de *software*, o uso de métricas não se limita apenas à mensuração de resultados finais, mas se estende à compreensão aprofundada das atividades em andamento. Essa abordagem, respaldada pela literatura de Pressman et al. (2021) e Sommerville (2018), propicia um ambiente de desenvolvimento mais controlado e adaptável, permitindo ajustes contínuos para atender aos objetivos estratégicos definidos. As métricas de processo e projeto de *software* são quantificadas ao longo das atividades e na avaliação do estado do projeto, visando objetivos estratégicos.

Baseada nas definições de Pressman et al. (2021) e Sommerville (2018), as métricas de processo e projeto de *software* foram classificadas para assim serem apresentadas na plataforma GMC; incluindo a média de defeitos identificados por atividade ou bloco de código, a taxa de esforço dedicado a módulos ou pequenas alterações, produtividade, quantidade de alterações em um determinado período ,quantidade de linhas de código-fonte, o número de módulos do projeto, taxas de entregas dentro do prazo estimado e a quantidade de entregas (*branches*) por período. A integração destas métricas, estabelecendo relações entre processo e projeto, busca promover uma maior qualidade e produtividade no desenvolvimento de *software*. A Tabela 7 lista as métricas elencadas para serem disponibilizadas pela Plataforma GMC.

Tabela 7 - Métricas disponibilizadas na plataforma GMC

	Métrica (Fonte)	**Fórmula**	**Tipo**
1	Tempo de Espera - TE (HERNÁNDEZ et al., 2019)	$TE = \frac{data\ final\ da\ entrega}{data\ da\ solicitação\ inicial}$	Processo e Projeto
2	Taxa de produção - TP (HERNÁNDEZ et al., 2019)	$TP = \frac{quantidade\ de\ solicitações\ atendidas}{tempo\ de\ resolução}$	Processo
3	Taxa de código - TC (OLIVEIRA, 2017)	$TC = \frac{número\ de\ arquivos\ do\ desenvolvedor}{tempo\ de\ resolução}$	Projeto
4	Taxa de linhas alteradas - TL (OLIVEIRA, 2017)	$TL = \frac{quantidade\ de\ linhas\ alteradas}{tempo\ de\ resolução}$	Processo e Projeto
5	Taxa de *commits* realizados - TCR (OLIVEIRA, 2017)	$TCR = \frac{número\ de\ commits\ realizados}{tempo\ de\ resolução}$	Processo e Projeto
6	Taxa de *commits* modificados - TCM (OLIVEIRA, 2017)	$TCM = \frac{\mid linhas\ modificadas\ por\ commit \mid}{tempo\ de\ resolução}$	Projeto
7	Avaliação de componente métrico - ACM WOLTER (2022)	$ACM = \frac{\sum_{n=1}^{\#-Metadata\ components} Metadata\ component_n}{\#Metadata\ components}$	Processo e Projeto

Fonte: Própria

Nesta etapa, conduzimos também uma pesquisa nos principais repositórios de código aberto disponíveis no *GitHub*, os quais foram utilizados como base de dados para testes durante a implementação da plataforma GMC. Esses repositórios foram selecionados ao longo da pesquisa, adotando critérios específicos. Estabelecemos como parâmetros um mínimo de cinquenta colaboradores ou cinco equipes, valores que se alinham com algumas práticas ágeis de desenvolvimento de *software*, como *SCRUM* [13]e *XP*[14], que preconizam equipes com até dez pessoas. Além disso, exigimos uma existência mínima de dois anos, visando filtrar projetos robustos e atualizados.

Adicionalmente, a seleção desses projetos foi orientada por licenças específicas, a saber: MIT, BSD, Apache 2.0 e GPLv3 devido à sua ampla aceitação em projetos de código aberto conforme relatado no GitHub (2023). A escolha foi guiada pela seleção de projetos conforme a Tabela 8, na qual qualquer um desses atenderia aos critérios previamente mencionados neste trabalho. Um dos fatores determinantes para a escolha foi a facilidade na extração de dados e o acesso completo a todas as informações por meio do *token* do GitHub.

Tabela 8 - Lista de projeto de código aberto no repositório do *GitHub*

Nome do Repositório	Link
Linux	https://github.com/torvalds/linux
VSCode	https://github.com/microsoft/vscode
Django	https://github.com/django/django
React	https://github.com/facebook/react
Flutter	https://github.com/flutter/flutter
Electron	https://github.com/electron/electron
Node.js	https://github.com/nodejs/node
Spring Boot	https://github.com/spring-projects/spring-boot
PyTorch	https://github.com/pytorch/pytorch
Cypress	https://github.com/cypress-io/cypress
RxJS	https://github.com/ReactiveX/rxjs
Junit	https://github.com/junit-team/junit5
Spring Initializr	https://github.com/spring-io/initializr
Selenium	https://github.com/SeleniumHQ/selenium
Nginx	https://github.com/nginx/nginx
Jupyter Notebooks	https://github.com/jupyter/notebook
Docker	https://github.com/docker/docker-ce
Go	https://github.com/golang/go
Elasticsearch	https://github.com/elastic/elasticsearch
SonarQube	https://github.com/SonarSource/sonarqube
Vue.js	https://github.com/vuejs/vue
Angular	https://github.com/angular/angular
Express.js	https://github.com/expressjs/express

Ao aprofundarmos a pesquisa sobre os projetos inicialmente identificados, também avaliamos

[13] Guia do *SCRUM*: https://scrumguides.org/

[14] *Framework* Ágil de *Software Extreme Programming*: https://www.agilealliance.org/glossary/xp/

a relevância desses projetos no cotidiano dos profissionais de desenvolvimento e sua constante participação no *GitHub*. Dois projetos em particular, Selenium e Spring Initializr, foram identificados como suficientes para validar o desenvolvimento da Plataforma, conforme detalhado no Capítulo 4.

3.4. Etapa 2: Desenvolvimento da Extração e Comunicação dos Sistemas

Nesta seção apresentamos um detalhamento das atividades realizadas para automatizar a extração de dados do *GitHub*, eliminando a necessidade de intervenção direta do usuário, e também as atividades para a comunicação entre os sistemas distintos, incluindo a ferramenta GDDownloader e o a camada de visão.

Para realizar a extração e a formação do conjunto de dados, foi utilizada a ferramenta GDDownloader da pesquisa de Diamantopoulos et al. (2020), que foi desenvolvida na linguagem Python. A ferramenta GDDownloader utiliza *tokens* para acessar o repositório *GitHub*. Esses *tokens* foram utilizados para autenticar a aplicação e permitir o acesso aos dados necessários para a extração das informações. Dessa forma, foi possível garantir a segurança e a integridade dos dados durante o processo de extração.

Embora seja viável extrair os dados com as permissões padrão do *token* de acesso, é aconselhável optar por todas as permissões disponíveis no *token* do *GitHub* para prevenir potenciais problemas na extração de dados vinculados ao projeto. Essa abordagem assegura a obtenção máxima de informações do repositório. Caso ocorra a recusa de alguma permissão durante a extração, a ferramenta GDDownloader interrompe o processo, reportando o erro ao sistema de visualização, onde o usuário possui acesso.

É relevante notar que o *GitHub* oferece suporte a dois tipos de *personal access tokens*: *fine-grained personal access tokens e personal access tokens* (*classic*). Para os propósitos deste trabalho, optou-se pelo uso do tipo *classic*, dada sua capacidade de controlar com precisão as permissões de acesso. Este enfoque proporciona uma camada adicional de segurança, fortalecendo a integridade do processo de colaboração e garantindo um ambiente mais seguro para a gestão de projetos.

Um dos desafios identificados na comunicação entre a ferramenta de extração e o servidor do *GitHub* está relacionado ao controle de requisições via API. O servidor *GitHub* impõe limites específicos, os quais são determinados pelo tipo de conta do usuário. Para contas gratuitas, o limite de requisições com *token* é de 5.000 (cinco mil) por hora, conforme indicado na documentação do GITHUB (2023).

Essa restrição pode resultar em situações em que o sistema tenta realizar várias requisições para a primeira extração e carga no banco de dados. No entanto, após a conclusão da primeira carga, a ferramenta GDDownloader, conforme descrita por Diamantopoulos et al. (2020), passa a operar realizando apenas atualizações nos novos dados. Essa abordagem reduz significativamente a quantidade de requisições aos servidores do *GitHub*, otimizando a eficiência do sistema e mitigando

os impactos dos limites impostos pela API.

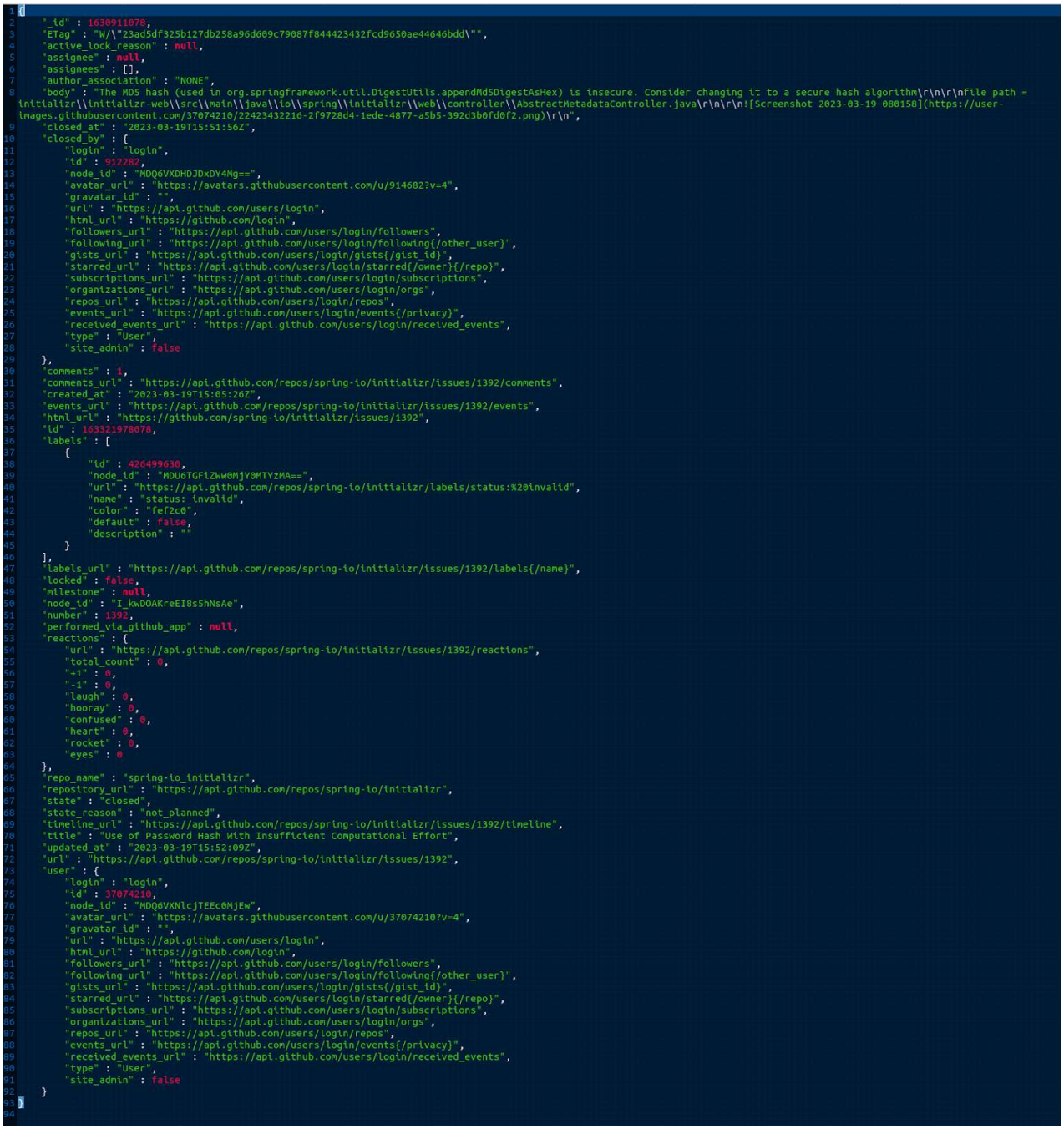
Para solicitações ao *GitHub* sem autenticação, ou seja, sem *token* de acesso, o limite cai para 60 (sessenta) solicitações por usuário/hora. Neste caso as solicitações são controladas pelo IP que chega aos servidores do *GitHub*. Os cabeçalhos de resposta descrevem o *status* do limite da taxa atual. Para a plataforma de métricas desenvolvidas nesta pesquisa o controle de requisições foi implementado para ser realizado a cada solicitação da API REST.

É relevante destacar que uma das vantagens proporcionadas pelo uso da ferramenta GDDownloader é a eficiente gestão das requisições HTTP no servidor *GitHub*. Se um usuário efetuar solicitações diretamente no *GitHub*, é provável que se depare com limitações conforme documentação do GITHUB (2023), dependendo do contexto de uso do sistema. No cenário da plataforma GMC, após a conclusão da extração completa dos dados, não é mais necessário requisitar informações ao servidor *GitHub*. Em caso de atualizações, apenas os dados específicos necessários serão solicitados.

Na adaptação da ferramenta GDDownloader, foi necessário implementar uma comunicação com o sistema da camada de visão, utilizando uma *Application Programming Interface* (API). A API da camada de visão inicia o processo enviando uma requisição HTTP para o GDDownloader, que interpreta a solicitação e executa a validação e extração dos dados no repositório remoto do *GitHub*. Para manter a arquitetura modular e coesa, a ferramenta GDDownloader permanece fiel ao seu propósito original de facilitar a obtenção de dados de maneira eficiente e organizada.

Na condução desta pesquisa, optou-se pelo uso do banco de dados *NoSQL* MongoDB (2023) devido principalmente a facilidade e compatibilidade da ferramenta de extração GDDownloader. Essa escolha foi motivada pelo fato de o GDDownloader ter sido programado para extrair dados em coleções não relacionais. O tamanho do conjunto de dados armazenado após a extração pode variar, dependendo da complexidade e do volume de informações no momento da extração do projeto. Para o gerenciamento e manipulação eficientes desses dados, empregou-se o Robo 3T (2023), uma interface gráfica de usuário (GUI) que simplifica a administração e a interação com bancos de dados MongoDB (2023). A Figura 8 ilustra a estrutura do conjunto de dados "*issues*" armazenado no banco de dados, proporcionando a visão com todos os detalhes do conjunto de dados formado por vários elementos. Este conjunto de dados é composto por diversos elementos que estão organizado em pares de chave e valor. Por exemplo, chave "id" possui o valor 912282. A tecnologia do MongoDB foi escolhida devido à disponibilidade de recursos gratuitos que atendem às exigências de manipulação dos dados gerados na pesquisa.

Figura 8 - Exemplo de um conjunto de dados



```
{
    "_id" : 1630911070,
    "ETag" : "W/\"23ad5df325b127db258a96d609c79087f844423432fcd9650ae44646bdd\"",
    "active_lock_reason" : null,
    "assignee" : null,
    "assignees" : [],
    "author_association" : "NONE",
    "body" : "The MD5 hash (used in org.springframework.util.DigestUtils.appendMd5DigestAsHex) is insecure. Consider changing it to a secure hash algorithm\r\n\r\nfile path = initializr\\initializr-web\\src\\main\\java\\io\\spring\\initializr\\web\\controller\\AbstractMetadataController.java\r\n\r\n![Screenshot 2023-03-19 080158](https://user-images.githubusercontent.com/37074210/224234322l6-2f9728d4-1ede-4877-a5b5-392d3b0fd0f2.png)\r\n",
    "closed_at" : "2023-03-19T15:51:56Z",
    "closed_by" : {
        "login" : "login",
        "id" : 912282,
        "node_id" : "MDQ6VXDHDJDxDY4Mg==",
        "avatar_url" : "https://avatars.githubusercontent.com/u/914682?v=4",
        "gravatar_id" : "",
        "url" : "https://api.github.com/users/login",
        "html_url" : "https://github.com/login",
        "followers_url" : "https://api.github.com/users/login/followers",
        "following_url" : "https://api.github.com/users/login/following{/other_user}",
        "gists_url" : "https://api.github.com/users/login/gists{/gist_id}",
        "starred_url" : "https://api.github.com/users/login/starred{/owner}{/repo}",
        "subscriptions_url" : "https://api.github.com/users/login/subscriptions",
        "organizations_url" : "https://api.github.com/users/login/orgs",
        "repos_url" : "https://api.github.com/users/login/repos",
        "events_url" : "https://api.github.com/users/login/events{/privacy}",
        "received_events_url" : "https://api.github.com/users/login/received_events",
        "type" : "User",
        "site_admin" : false
    },
    "comments" : 1,
    "comments_url" : "https://api.github.com/repos/spring-io/initializr/issues/1392/comments",
    "created_at" : "2023-03-19T15:05:26Z",
    "events_url" : "https://api.github.com/repos/spring-io/initializr/issues/1392/events",
    "html_url" : "https://github.com/spring-io/initializr/issues/1392",
    "id" : 1633219780l8,
    "labels" : [
        {
            "id" : 426499630,
            "node_id" : "MDU6TGFiZWw0MjY0MTYzMA==",
            "url" : "https://api.github.com/repos/spring-io/initializr/labels/status:%20invalid",
            "name" : "status: invalid",
            "color" : "fef2c0",
            "default" : false,
            "description" : ""
        }
    ],
    "labels_url" : "https://api.github.com/repos/spring-io/initializr/issues/1392/labels{/name}",
    "locked" : false,
    "milestone" : null,
    "node_id" : "I_kwDOAKreEI8sShNsAe",
    "number" : 1392,
    "performed_via_github_app" : null,
    "reactions" : {
        "url" : "https://api.github.com/repos/spring-io/initializr/issues/1392/reactions",
        "total_count" : 0,
        "+1" : 0,
        "-1" : 0,
        "laugh" : 0,
        "hooray" : 0,
        "confused" : 0,
        "heart" : 0,
        "rocket" : 0,
        "eyes" : 0
    },
    "repo_name" : "spring-io_initializr",
    "repository_url" : "https://api.github.com/repos/spring-io/initializr",
    "state" : "closed",
    "state_reason" : "not_planned",
    "timeline_url" : "https://api.github.com/repos/spring-io/initializr/issues/1392/timeline",
    "title" : "Use of Password Hash With Insufficient Computational Effort",
    "updated_at" : "2023-03-19T15:52:09Z",
    "url" : "https://api.github.com/repos/spring-io/initializr/issues/1392",
    "user" : {
        "login" : "login",
        "id" : 37074210,
        "node_id" : "MDQ6VXNlcjTEEc0MjEw",
        "avatar_url" : "https://avatars.githubusercontent.com/u/37074210?v=4",
        "gravatar_id" : "",
        "url" : "https://api.github.com/users/login",
        "html_url" : "https://github.com/login",
        "followers_url" : "https://api.github.com/users/login/followers",
        "following_url" : "https://api.github.com/users/login/following{/other_user}",
        "gists_url" : "https://api.github.com/users/login/gists{/gist_id}",
        "starred_url" : "https://api.github.com/users/login/starred{/owner}{/repo}",
        "subscriptions_url" : "https://api.github.com/users/login/subscriptions",
        "organizations_url" : "https://api.github.com/users/login/orgs",
        "repos_url" : "https://api.github.com/users/login/repos",
        "events_url" : "https://api.github.com/users/login/events{/privacy}",
        "received_events_url" : "https://api.github.com/users/login/received_events",
        "type" : "User",
        "site_admin" : false
    }
}
```


Fonte: Própria

3.5. Etapa 3: Implementação, Processamento e Disponibilidade da Aplicação da Camada de Negócio

A aplicação da camada de negócio foi desenvolvida para executar os algoritmos das métricas, realizar consultas no conjunto de dados armazenado no banco de dados MongoDB, e conduzir a análise, processamento e aplicação de filtros nos dados. Isso resulta na organização e disponibilização dos dados por meio de APIs, que são serviços continuamente consumidos pela camada de visão, integrando-se assim à plataforma GMC.

Todos os dados brutos e não relacionais são processados na aplicação da camada de negócio e

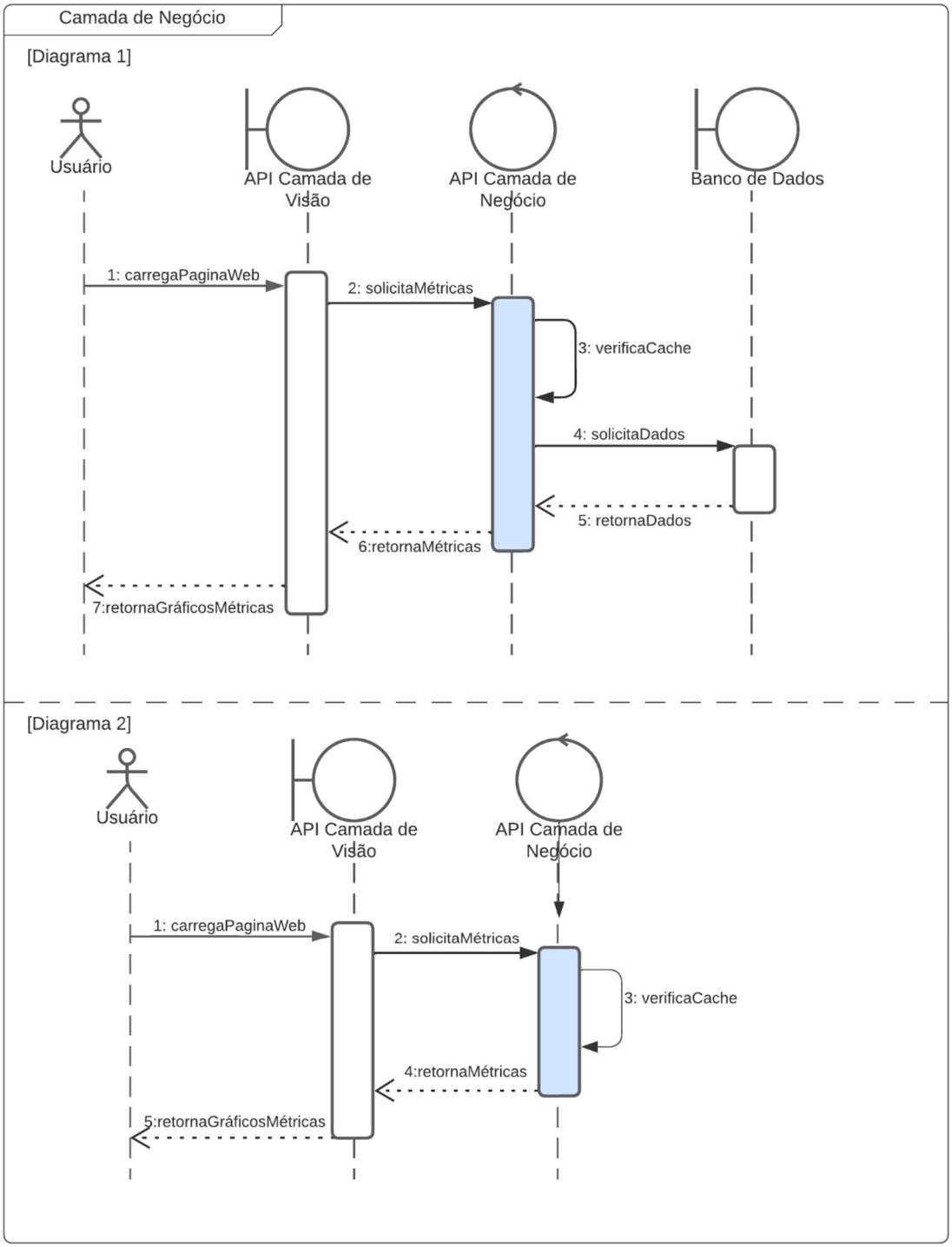
consumidos para serem tratados através de consultas implementadas com o *framework Spring Boot*. Na camada de negócio também foram implementadas as configurações para documentação e testes do sistema com o SWAGGER (2023) onde é possível visualizar a descrição das funcionalidades que a *API* oferece. Para cada funcionalidade do serviço, foram descritos os parâmetros que devem ser fornecidos quando essa funcionalidade é chamada e os valores que serão retornados aos usuários da *API*.

O *Spring Framework* é notavelmente modular, permitindo aos desenvolvedores uma seleção precisa dos módulos necessários para suas aplicações. No núcleo desse *framework*, destacam-se os módulos do *container* principal, que oferecem um sistema de configuração flexível e um sofisticado mecanismo de injeção de dependência. Além disso, o *Spring Framework* proporciona um suporte abrangente e fundamental para diversas arquiteturas de aplicativos, incluindo recursos voltados para mensagens, transações de dados e camadas de persistência. Isso também engloba um conjunto de funcionalidades específicas para aplicativos web (SPRING FRAMEWORK, 2023).

Na implementação da plataforma web GMC, a escolha do *framework Spring* foi fundamentada em sua notável flexibilidade e capacidade de configuração, que se estende à orquestração entre a base de dados e à criação da aplicação. Segundo a documentação Spring Framework (2023), o *Spring* também oferece uma base sólida para o desenvolvimento de aplicativos altamente complexos e eficientes, tornando-se uma escolha de extrema relevância para a comunidade de desenvolvedores que buscam soluções avançadas no campo da engenharia de *software*.

A especificação de uma API é um documento importante que descreve como um serviço deve ser usado e como os dados devem ser trocados entre o cliente e o servidor. Uma especificação bem documentada facilita a integração de clientes e servidores, além de ajudar na documentação da API. O Swagger (2023) permite descrever APIs claras e interativas, também é possível gerar automaticamente bibliotecas de clientes para uma API em vários idiomas. O Swagger faz todo esse trabalho automaticamente solicitando ao serviço que contenha uma API que retorne arquivos YAML ou JSON com uma descrição detalhada. A Figura 9 mostra o comportamento das requisições processadas no sistema da camada de negócio.

Figura 9 - Diagrama de sequência da camada de negócio



Fonte: Própria

A Figura 9 esboça o fluxo de processamento das solicitações na camada de negócio referentes ao carregamento de métricas na plataforma GMC. Conforme ilustrado no Diagrama 1, ao solicitar a camada de negócio as métricas de visualização, a camada de visão passa pelo seguinte processo: a camada de negócio verifica se a consulta já foi previamente realizada e está armazenada em *cache*. Se afirmativo, o sistema obtém os resultados diretamente do *cache*, conforme apresentado no Diagrama 2, otimizando o uso de recursos da aplicação. Caso contrário, a camada de negócio efetua a consulta

na base de dados, constrói o conjunto de métricas da Tabela 2 e disponibiliza a resposta por meio de um serviço na *API*. Vale ressaltar que esse procedimento é executado em questão de milissegundos, sendo imperceptível ao usuário.

3.6. Etapa 4: Integração e Implementação da camada de visão

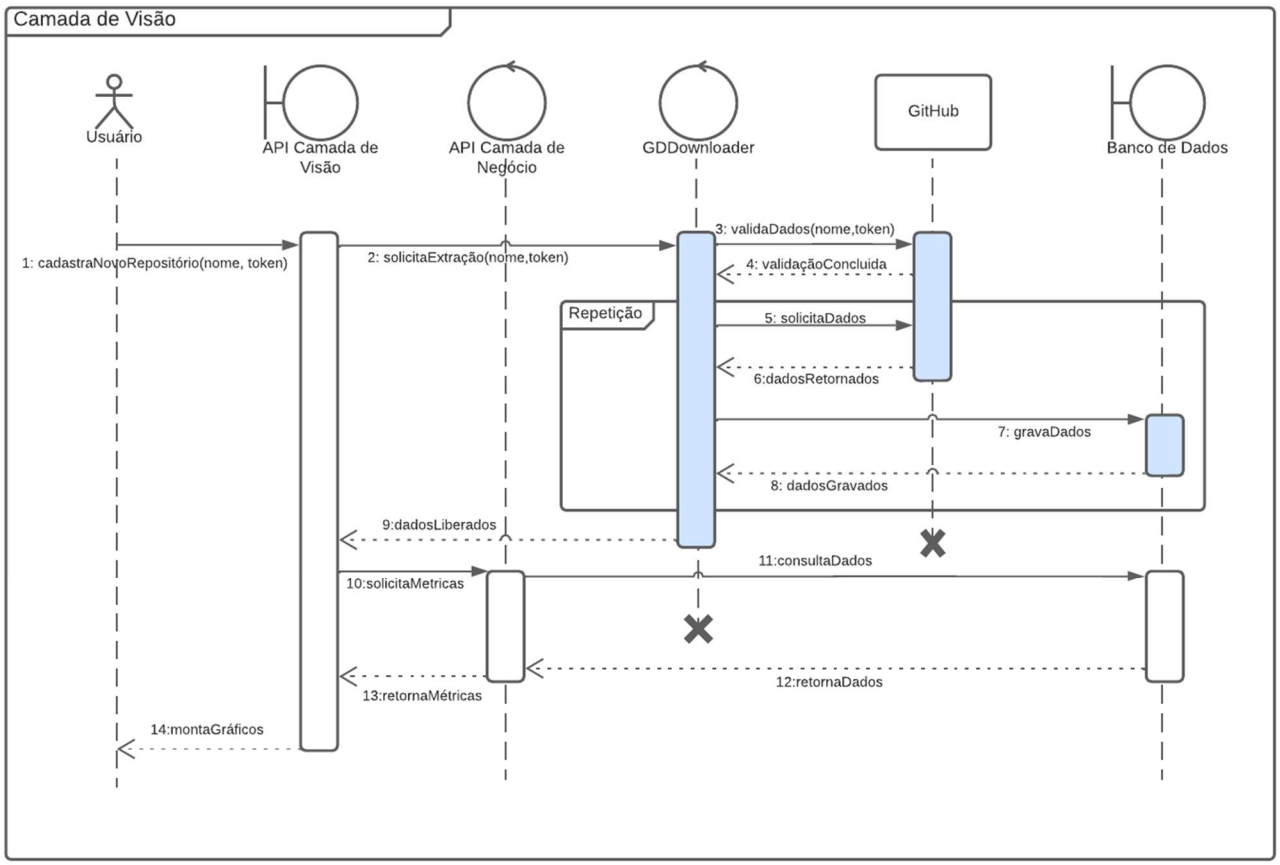
Para disponibilizar as métricas de processo e projeto de *software* baseadas nos conjuntos de dados já extraídos e armazenados no banco de dados foi necessário desenvolver uma aplicação para a camada de visão (*web*) que faz a comunicação com a camada de negócio e exibe as informações através de indicadores, tabelas e gráficos. A camada de visão foi desenvolvida utilizando o *framework* Angular e a biblioteca de gráficos *Chart.js*. A integração entre a aplicação e o servidor é realizada exclusivamente por meio de *API*, sendo que consultas ativas são armazenadas em *cache* para otimizar a performance do sistema.

Para tornar a interação entre os usuários e a plataforma sobre métricas o mais intuitiva possível, a usabilidade é um fator determinante para garantir a adoção e a satisfação desses usuários. O processo de análise, extração, processamento e disponibilização de métricas tem início com a *URL* (*Uniform Resource Locator*) do repositório *GitHub* e o *token* de acesso com as devidas permissões. Com apenas esses parâmetros, o usuário do sistema pode gerar as métricas disponibilizá-las na plataforma.

A camada de visão é o ponto focal de todas as métricas disponíveis aos usuários. É nesta interface que os indicadores, tabelas e gráficos são apresentados de maneira clara e acessível, permitindo uma interpretação rápida de informações processadas. Algumas dessas informações são semelhantes às encontradas diretamente no *GitHub*, mas a plataforma GMC se destaca ao exibir indicadores com visualizações diversas e exclusivas como relações de produtividade.

O diagrama de sequência apresentado na Figura 10 descreve um procedimento semelhante ao representado na Figura 7. No entanto, neste esquema, destaca-se a integração entre o GDDownloader e o *GitHub*. Para o usuário final, a camada de visão assume a responsabilidade de encapsular todo o fluxo, desde o processo de cadastro até o armazenamento e exibição das métricas relacionadas aos projetos de *software*.

Figura 10 - Diagrama da camada de visão



Fonte: Própria

A Plataforma GMC proporciona aos usuários indicadores quantitativos de métricas de processo e projeto de *software* das alterações realizadas durante o projeto ou depois de sua finalização. Oferece tabelas organizadas por colaboradores em períodos de tempo dinamicamente definidos, apresentando uma variedade de gráficos que detalham as alterações ao longo do projeto por colaborador. Estes gráficos incluem uma relação cruzada com a quantidade de repositórios adicionados pelo usuário. Além disso, a plataforma disponibiliza métricas de produtividade e desempenho, apresentadas de forma clara através de tabelas e gráficos, como pode ser observado na Figura 11.

A Figura 11 ilustra a exposição das métricas aos participantes do projeto, apresentando indicadores pontuais de atividades segregados por repositório. Além disso, são exibidos gráficos que dinamicamente revelam a produtividade e desempenho dos desenvolvedores ao longo de vários períodos de tempo, com ênfase nas métricas abordadas nesta pesquisa. A análise das métricas é conduzida de maneira individualizada para cada repositório, proporcionando uma compreensão mais detalhada. Essa abordagem facilita a identificação de padrões e tendências específicos, contribuindo para uma análise mais precisa e informada no contexto do projeto.

Figura 11 - Tela Inicial da plataforma GMC

Fonte: Própria

4. AVALIAÇÕES DA PLATAFORMA

A avaliação da Plataforma GMC corresponde a Etapa 5 da metodologia prevista para execução da presente pesquisa e foi realizada de duas formas distintas. A primeira contemplou a execução das etapas de extração, integração e visualização das métricas de dois projetos disponíveis no *GitHub*. A segunda avaliação envolveu a aplicação de um questionário direcionado a profissionais de desenvolvimento de *software*, com base nas métricas previamente exploradas na pesquisa.

4.1. Análise do Processo de Extração, Integração e Visualização da Plataforma

A primeira avaliação da Plataforma GMC foi realizada em um ambiente que busca simular um ambiente produtivo como uma aplicação em funcionamento na Internet. Para seleção dos projetos a serem avaliados, além da disponibilidade no *GitHub*, foram considerados outros fatores críticos, como a relevância dos projetos para a atividade de pesquisa e os problemas solucionados. Para ter um volume considerável de dados para análise, buscou-se por projetos com mais de dois anos de existência, com um mínimo cinquenta colaboradores e que ainda estivesse em atividade. Nesse contexto, dois projetos foram suficientes para a avaliação devido ao volume de dados disponível para extração, de aproximadamente 87MB.

Um dos projetos selecionados para testar a extração foi o projeto *Spring Initializr* que disponibiliza uma API expansível para a geração de projetos iniciais. Quando o desenvolvedor começa a criação de um projeto, o *Spring Initializr* atua para prover todas as dependências necessárias e as versões da linguagem de programação Java e tecnologias que compõem o projeto. Além disso, oferece um serviço configurável através da Internet que disponibiliza uma interface de usuário para a configuração do projeto a ser gerado e *endpoints* acessíveis via HTTP simples.

Outro projeto utilizado para avaliação da Plataforma foi o *Selenium* que é uma ferramenta de código aberto gratuita que viabiliza a execução de testes automatizados, licenciada sob a Apache 2.0. O *Selenium* é capaz de operar tanto em navegadores web quanto diretamente no código fonte, proporcionando a identificação de falhas na interface do usuário em sistemas web. É crucial observar que os projetos em questão foram criteriosamente escolhidos para os testes desta avaliação, visando reproduzir de maneira fiel um ambiente real.

Com a finalidade de apresentar métricas precisas, é importante que o projeto selecionado seja acompanhado de forma gradual, com iterações que respeitem o tempo dedicado a cada atividade. Contudo, é válido destacar que qualquer projeto escolhido para exibir métricas na plataforma GMC operará de maneira normal e representativa. A Tabela 9 exibe os resultados do teste para a primeira extração do repositório conduzida em um ambiente controlado, apresentando a quantidade de dados extraídos, as especificações do hardware empregado, bem como o tempo e a data da extração para os repositórios do *Spring Initializr* e *Selenium*.

Tabela 9 - Dados de teste de extração no repositório *GitHub*

Nome do Repositório	Colaboradores (und)	*Issue* (und)	*Commits* (und)	*Hardware*	Tempo de Extração (min)	Data da Extração
Spring Initializr	96	1401	2238	8GB de memória RAM DDR4, 2 núcleos de processamento e um link de Internet de 100Mbps compartilhado.	30	15/11/2023
Selenium	379	13.009	30.773	8GB de memória RAM DDR4, 2 núcleos de processamento e um link de Internet de 100Mbps compartilhado.	60	15/11/2023

Fonte: Própria

Na Figura 12, encontra-se o código-fonte desenvolvido para realizar testes e consultas em uma base de dados, previamente criada e destinada à extração. A execução manual do código foi realizada imediatamente após a extração, com os dados registrados na base *NoSQL*. Este procedimento teve como propósito validar a igualdade entre as informações extraídas e aquelas disponíveis nos projetos *Selenium* e *Spring Initializr* mantidos no repositório *GitHub*.

Existem várias opções na Internet para montar esses servidores *web*. No momento desta pesquisa, a *Amazon Web Services* (*AWS*) se destaca como uma plataforma que oferece uma ampla variedade de serviços, abrangendo desde capacidade computacional até armazenamento de dados, gerenciamento de bancos de dados, entre outros. A característica fundamental da *AWS* é a sua abordagem de pagamento por uso, permitindo que os usuários paguem apenas pelos recursos que consomem. Reconhecida por sua confiabilidade, escalabilidade e constante inovação, a AWS é adotada por empresas de todos os portes em todo o mundo. Além disso, a plataforma oferece acesso gratuito a recursos básicos, suficientes para os testes realizados neste estudo, como destacado na Tabela 9.

Figura 12 - Código-fonte para consultas de testes *NoSQL*

```
1 - db.commits.aggregate([])
2 - db.commitComments.aggregate([])
3 - db.contributors.aggregate([])
4 - db.issueComments.aggregate([])
5 - db.issueEvents.aggregate([])
6 - db.issues.aggregate([])
7 - db.projects.aggregate([])
8 - db.stats.aggregate([])

9 - db.commits.find({
    "repo_name": {
        $in: ["spring-io/initializr", "SeleniumHQ_selenium"]
    }
})


10 - db.contributors.aggregate(
  [
  { "$project" : { "repositorio" : "$contributions.repo_name"}}
  ]
)

11 - db.issue.aggregate(
  [
    { "$match": { "repo_name" : "SeleniumHQ_selenium" } },
    { "$group": { "_id": "$repo_name", "quantidade": { "$sum": 1 } } }
  ]
)

12 - db.commits.aggregate(
  [
    { "$project" : { "email" : "$commit.author.email", "ano" : "2023", "nomeRepositorio" : "$repo_name"}},
    { "$group" : { "_id" : { "email" : "$email", "ano" : "$ano", "nomeRepositorio" : "$nomeRepositorio"},
      "total" : { "$sum" : 1}}
    }
  ]
)
```

Fonte: Própria

Na Figura 12, são linhas de código para diversas operações realizadas nos conjuntos de dados, abrangendo: *Commits*, *commitComments*, *contributors*, *issueComments*, *issueEvents*, *issues*, *projects* e *stats*. Estas operações processam os registros exportados e armazenados na base de dados *NoSQL*, englobando dois repositórios específicos. As operações numeradas de 1 a 8 visam recuperar os registros sem aplicar critérios de filtragem. Em contrapartida, as operações 10 e 11 segmentam os dados com base no repositório e consolidam a quantidade total. Por fim, a operação 12 realiza uma filtragem utilizando o e-mail do contribuidor e o ano de 2023, seguida pelo agrupamento dos resultados pelo repositório. Todos esses códigos, e alguns adicionais, estão disponíveis online e podem ser acessados por consultas_testes[15].

Devido à grande quantidade de dados resultantes das consultas, na ordem de milhares, a exposição completa desses dados no âmbito da pesquisa seria impraticável. Assim, a validação dos dados extraídos foi conduzida por meio de amostragem, permitindo a verificação da correspondência com as informações disponíveis no repositório remoto do *GitHub*. Entretanto, uma representação reduzida dos resultados, obtidos ao consultar o projeto na base de dados, é apresentada na Figura 13.

[15] Consultas de testes: https://zenodo.org/records/10574281/files/consultas_testes.js?download=1

Figura 13- Dados relativo ao conjunto de dados do projeto

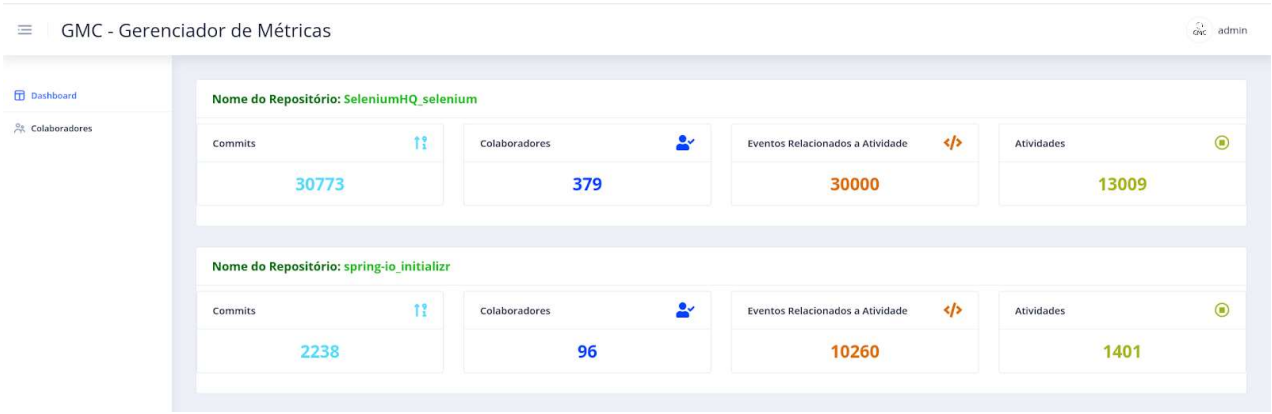

```
{
    "_id" : 7613,
    "issue_comment_url" : "https://api.github.com/repos/SeleniumHQ/selenium/issues/comments{/number}",
    "issue_events_url" : "https://api.github.com/repos/SeleniumHQ/selenium/issues/events{/number}",
    "issues_url" : "https://api.github.com/repos/SeleniumHQ/selenium/issues{/number}",
    "keys_url" : "https://api.github.com/repos/SeleniumHQ/selenium/keys{/key_id}",
    "labels_url" : "https://api.github.com/repos/SeleniumHQ/selenium/labels{/name}",
    "language" : "Java",
    "languages_url" : "https://api.github.com/repos/SeleniumHQ/selenium/languages",
    "license" : {
        "key" : "apache-2.0",
        "name" : "Apache License 2.0",
        "spdx_id" : "Apache-2.0",
        "url" : "https://api.github.com/licenses/apache-2.0",
    },
    "merges_url" : "https://api.github.com/repos/SeleniumHQ/selenium/merges",
    "milestones_url" : "https://api.github.com/repos/SeleniumHQ/selenium/milestones{/number}",
    "mirror_url" : null,
    "name" : "selenium",
    "network_count" : 7904,
    "notifications_url" : "https://api.github.com/repos/SeleniumHQ/selenium/notifications{?since,all,participating}",
    "open_issues" : 253,
    "open_issues_count" : 253,
    "private" : false,
    "pulls_url" : "https://api.github.com/repos/SeleniumHQ/selenium/pulls{/number}",
    "pushed_at" : "2023-11-15T11:37:45Z",
    "releases_url" : "https://api.github.com/repos/SeleniumHQ/selenium/releases{/id}",
    "repo_name" : "SeleniumHQ_selenium",
    "size" : 2121889,
    "ssh_url" : "git@github.com:SeleniumHQ/selenium.git",
    "stargazers_count" : 28200,
    "stargazers_url" : "https://api.github.com/repos/SeleniumHQ/selenium/stargazers",
    "statuses_url" : "https://api.github.com/repos/SeleniumHQ/selenium/statuses/{sha}",
    "subscribers_count" : 1313,
    "subscribers_url" : "https://api.github.com/repos/SeleniumHQ/selenium/subscribers",
    "subscription_url" : "https://api.github.com/repos/SeleniumHQ/selenium/subscription",
    "tags_url" : "https://api.github.com/repos/SeleniumHQ/selenium/tags",
    "teams_url" : "https://api.github.com/repos/SeleniumHQ/selenium/teams",
    "topics" : [
        "dotnet",
        "java",
        "javascript",
        "python",
        "ruby",
        "selenium",
        "webdriver"
    ],
    "trees_url" : "https://api.github.com/repos/SeleniumHQ/selenium/git/trees{/sha}",
    "updated_at" : "2023-11-15T13:01:24Z",
    "url" : "https://api.github.com/repos/SeleniumHQ/selenium",
    "visibility" : "public",
    "watchers" : 28200,
    "watchers_count" : 28200,
    "web_commit_signoff_required" : false
}

{
    "_id" : 1055386,
    "issue_comment_url" : "https://api.github.com/repos/spring-io/initializr/issues/comments{/number}",
    "issue_events_url" : "https://api.github.com/repos/spring-io/initializr/issues/events{/number}",
    "issues_url" : "https://api.github.com/repos/spring-io/initializr/issues{/number}",
    "keys_url" : "https://api.github.com/repos/spring-io/initializr/keys{/key_id}",
    "labels_url" : "https://api.github.com/repos/spring-io/initializr/labels{/name}",
    "language" : "Java",
    "languages_url" : "https://api.github.com/repos/spring-io/initializr/languages",
    "license" : {
        "key" : "apache-2.0",
        "name" : "Apache License 2.0",
        "spdx_id" : "Apache-2.0",
        "url" : "https://api.github.com/licenses/apache-2.0",
    },
    "merges_url" : "https://api.github.com/repos/spring-io/initializr/merges",
    "milestones_url" : "https://api.github.com/repos/spring-io/initializr/milestones{/number}",
    "mirror_url" : null,
    "name" : "initializr",
    "network_count" : 1692,
    "notifications_url" : "https://api.github.com/repos/spring-io/initializr/notifications{?since,all,participating}",
    "open_issues" : 50,
    "open_issues_count" : 50,
    "private" : false,
    "pulls_url" : "https://api.github.com/repos/spring-io/initializr/pulls{/number}",
    "pushed_at" : "2023-11-02T13:20:19Z",
    "releases_url" : "https://api.github.com/repos/spring-io/initializr/releases{/id}",
    "repo_name" : "spring-io_initializr",
    "size" : 10509,
    "ssh_url" : "git@github.com:spring-io/initializr.git",
    "stargazers_count" : 3240,
    "stargazers_url" : "https://api.github.com/repos/spring-io/initializr/stargazers",
    "statuses_url" : "https://api.github.com/repos/spring-io/initializr/statuses/{sha}",
    "subscribers_count" : 151,
    "subscribers_url" : "https://api.github.com/repos/spring-io/initializr/subscribers",
    "subscription_url" : "https://api.github.com/repos/spring-io/initializr/subscription",
    "tags_url" : "https://api.github.com/repos/spring-io/initializr/tags",
    "teams_url" : "https://api.github.com/repos/spring-io/initializr/teams",
    "topics" : [],
    "trees_url" : "https://api.github.com/repos/spring-io/initializr/git/trees{/sha}",
    "updated_at" : "2023-11-14T12:39:58Z",
    "url" : "https://api.github.com/repos/spring-io/initializr",
    "visibility" : "public",
    "watchers" : 3240,
    "watchers_count" : 3240,
    "web_commit_signoff_required" : false
}
```


Fonte: Própria

A Figura 14 apresenta um exemplo conciso desses indicadores relacionados a contribuições, colaboradores, eventos vinculados a contribuição e atividades criadas. Essas métricas também estão disponíveis de maneira alternativa na plataforma GMC, proporcionando aos usuários uma abordagem flexível para acessar e analisar essas informações.

Figura 14 - Indicadores diretos de dados



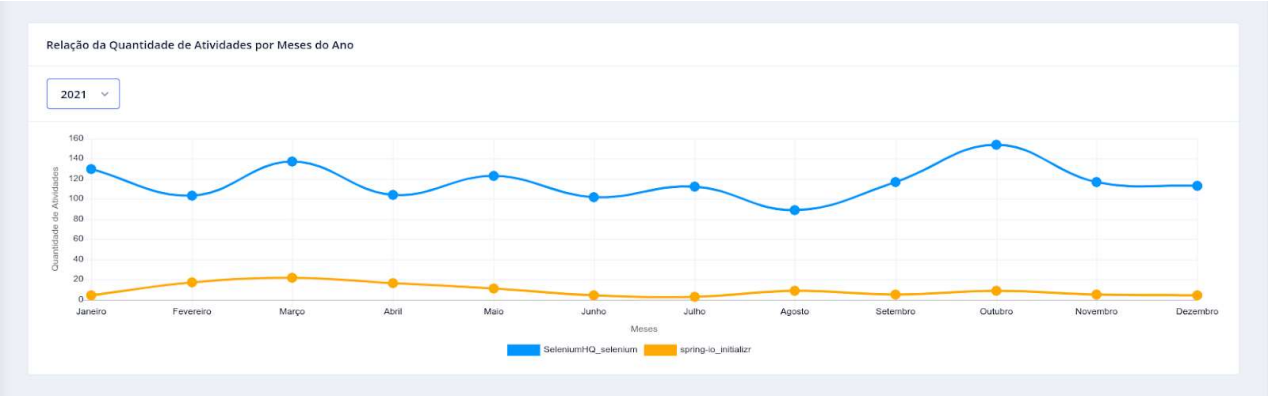
Fonte: Própria

Para criar e visualizar dinamicamente métricas de processo e projeto, a plataforma GMC utiliza gráficos que simplificam a análise e permitem aos usuários comparar indicadores, como a taxa de entregas realizadas por período, o número de modificações e a quantidade de solicitações atendidas ao longo de diversos intervalos temporais. Essa abordagem facilita a compreensão e avaliação das métricas, promovendo uma interpretação mais intuitiva e comparativa dos dados relevantes.

As Figuras 15, 16 e 17 ilustram exemplos de métricas que foram extraídas, processadas e disponibilizadas na aplicação *web* da Plataforma GMC. Os gráficos das figuras exibem a contagem das atividades realizadas por todos os colaboradores nos projetos Selenium e Spring Initializr ao longo dos meses dos anos 2021, 2022, 2023, respectivamente. É importante observar que, dependendo da quantidade de projetos integrados na plataforma GMC, o gráfico exibido estabelece a relação entre todos os repositórios, permitindo a seleção individual de um projeto específico.

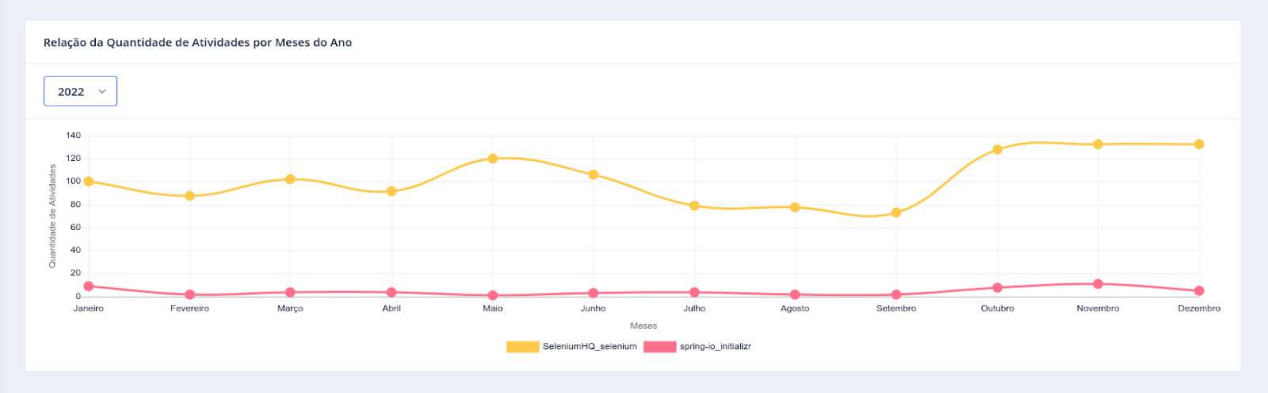
A geração dinâmica do gráfico, em conformidade com o ano escolhido, proporciona uma visualização adaptativa e personalizada. Além disso, destaca-se a funcionalidade de seleção de pontos no gráfico na Figura 15, que viabiliza a visualização direta da quantidade total de atividades. Essa característica interativa amplia a capacidade analítica do usuário, possibilitando uma inspeção mais aprofundada das informações exibidas.

Figura 15 - Relação de quantidade de atividades por meses do ano de 2021



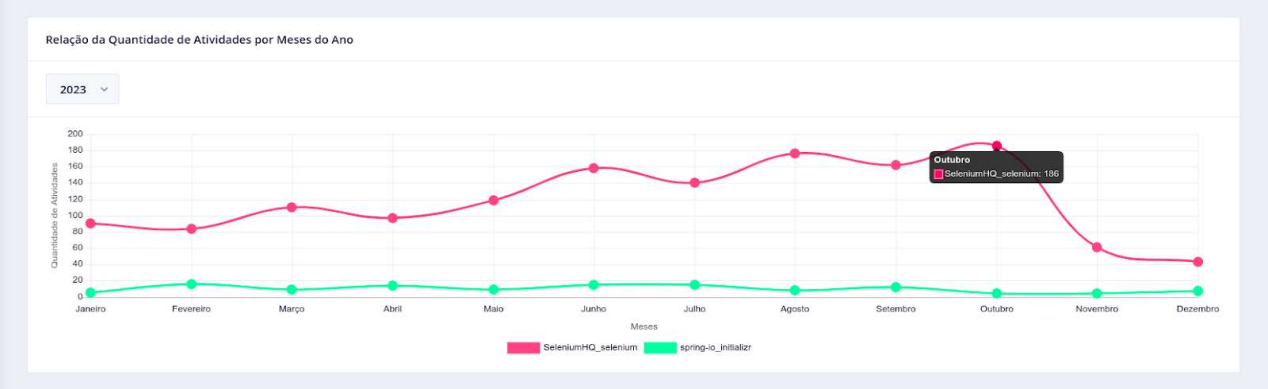
Fonte: Própria

Figura 16 - Relação de quantidade de atividades por meses do ano 2022



Fonte: Própria

Figura 17 - Relação de quantidade de atividades por meses do ano 2023

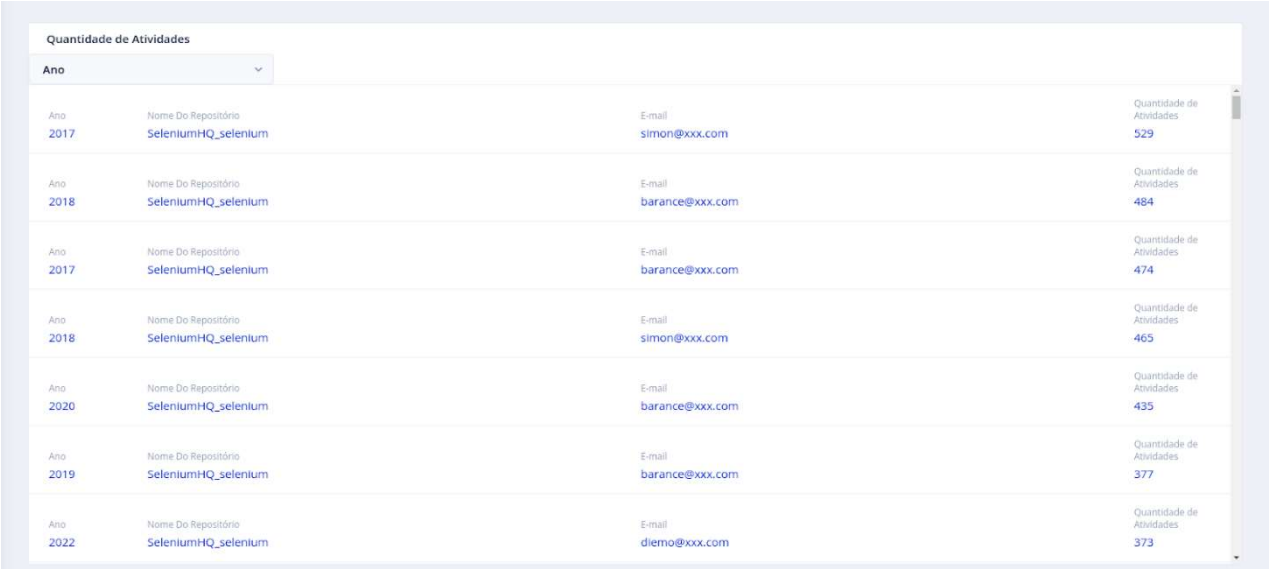


Fonte: Própria

As métricas inter-relacionadas entre tempo, atividades e colaboradores estão visíveis na Figura 18, nesta figura é apresentada a distribuição das atividades realizadas por cada colaborador ao longo de vários anos, a partir de 2017 (neste exemplo foi o ano com a maior quantidade de atividades), organizada em ordem crescente de quantidade de atividades. Os dados na tabela estão ordenados com base na quantidade de atividades, destacando assim os sete principais contribuintes ao projeto *Selenium* em cada ano. Além disso, é possível realizar ordenação por meses do ano, conforme demonstrado na Figura 21, que apresenta os sete primeiros registros da quantidade de atividades realizadas no projeto

Spring Initializr, categorizados por ano/mês. As informações que identificam o colaborador foram ocultadas por motivo de segurança.

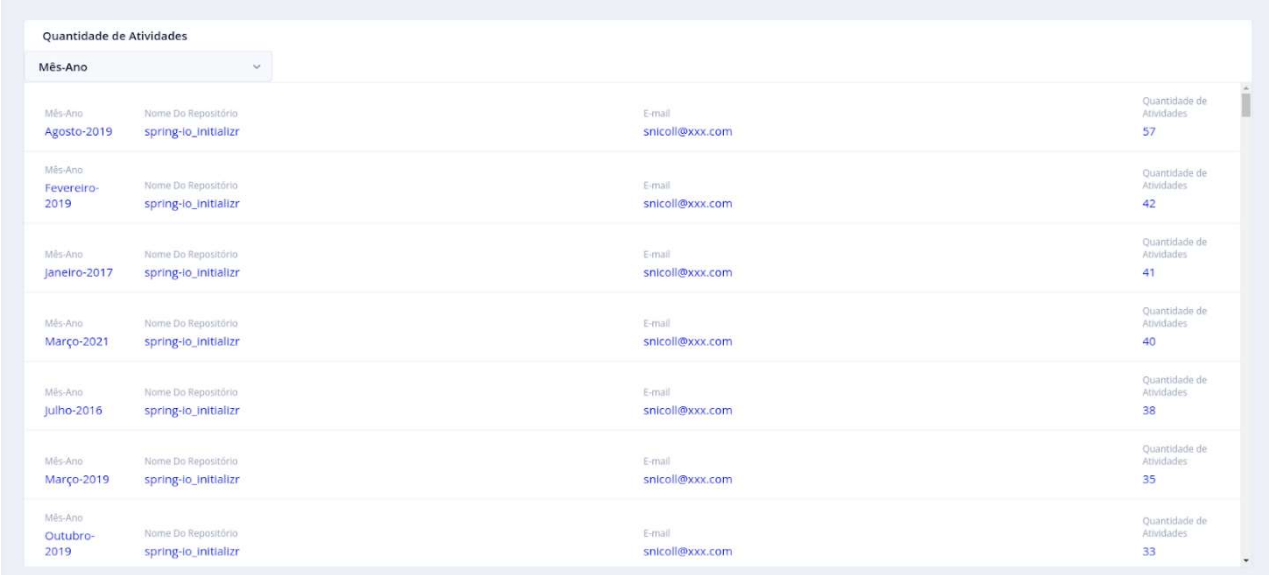
Figura 18 - Quantidade de atividades por colaborador no ano



Fonte: Própria

A Figura 19 apresenta a distribuição da quantidade de atividades por colaborador, filtradas por mês e ano, e ordenadas em ordem decrescente de atividades. Nesta figura, é crucial observar a qualidade das atividades realizadas ao aplicar dois filtros simultaneamente, referentes ao mês e ano, em relação aos dados registrados na base de dados.

Figura 19 - Quantidade de atividades por colaborador no mês/ano



Fonte: Própria

Para visualizar métricas de produtividade com base na pesquisa de WOLTER (2022), é essencial realizar uma análise individual de cada repositório. Isso permitirá a adaptação do componente

métrico escala *Commit* (Resultado da relação da quantidade de *committers* e total de desenvolvedores) e escala SLOC (Resultado da relação das linhas de código fonte removidas e acrescentadas), além de estabelecer níveis e intervalos de valores adequados. As Tabelas 10 e 11 apresentam adaptações específicas realizadas neste estudo, seguindo as diretrizes estabelecidas por Wolter (2022) em relação aos valores de nível e escala. Os cálculos foram feitos levando em consideração a média dos valores mínimos e máximos das escalas de Commits e SLOC. O repositório Selenium é apresentado na Tabela 10 e o repositório *Spring Initializr* na Tabela 11.

Os dados nas tabelas foram obtidos a partir das informações referentes ao ano de 2023, através de uma pesquisa aleatória envolvendo cinco desenvolvedores que interagiram com o *GitHub* em cada repositório, conforme apresentado na Figura 20. É fundamental destacar que a Figura 20 ilustra a produtividade dos desenvolvedores, fundamentada nos dados gerados durante o projeto dos repositórios apresentados. Essa métrica foi adaptada e viabilizada pelo método de cálculo delineado na pesquisa de Wolter (2022), proporcionando uma abordagem robusta e fundamentada na análise dos dados pertinentes.

Tabela 10 - Escala adaptada para o repositório *Selenium*

Escala *Commit*			Escala SLOC	
Nível	Escala		Nível	Escala
1	0 - 100		10	0 – 1500
2	100 - 200		20	1500 – 3000
3	200 - ∞		30	3000 - ∞

Fonte: Própria

Tabela 11 - Escala adaptada o repositório *Spring Initializr*

Escala *Commit*			Escala SLOC	
Nível	Escala		Nível	Escala
1	0 - 10		10	0 - 10
2	10 - 20		20	10 - 20
3	20 - ∞		30	20 - ∞

Fonte: Própria

Na Figura 20, os dados estão organizados por repositório ao longo do eixo vertical, com uma escala máxima de 1, conforme definido na pesquisa de WOLTER (2022). Ao longo do eixo horizontal estão os nomes de cinco dos desenvolvedores associados a cada repositório. Nesta figura, é realizada

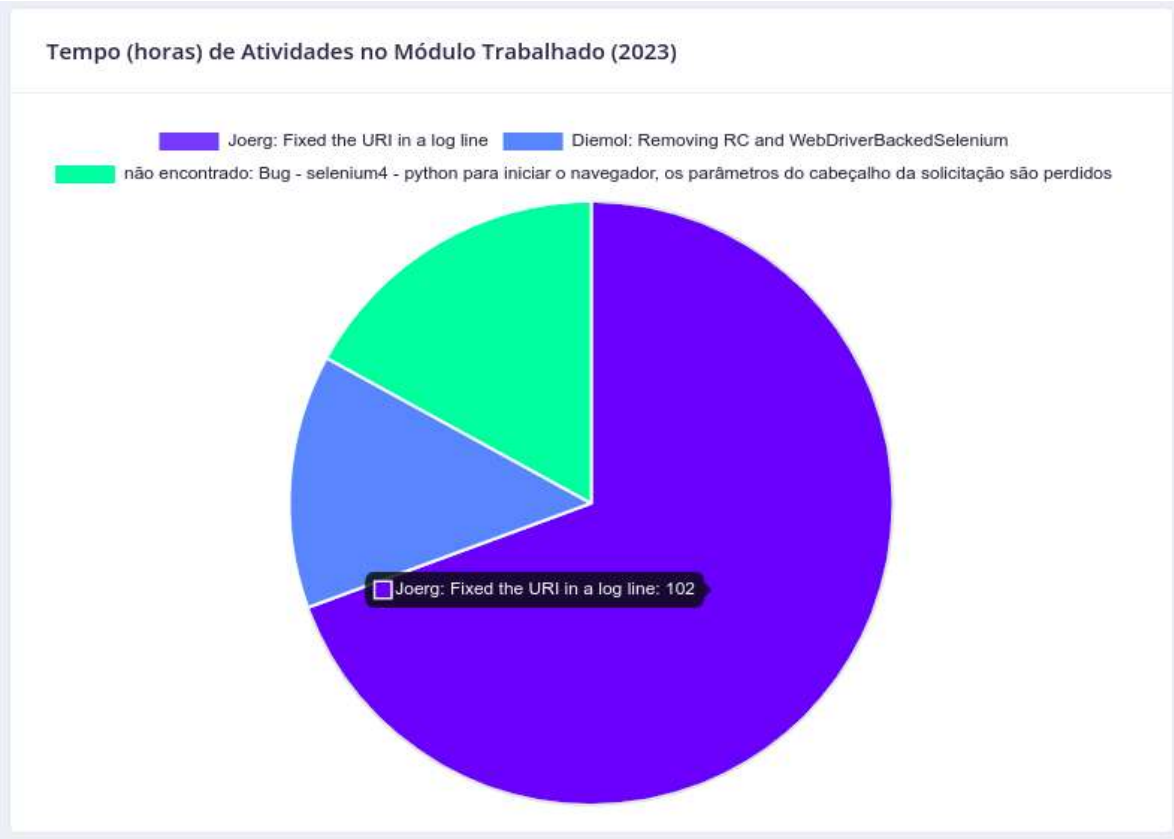
uma comparação visual para destacar a produtividade de diferentes desenvolvedores e sua relação com projetos distintos. No entanto, também é possível conduzir uma análise individual para determinar quais foram os mais produtivos em um mesmo projeto. Na figura, observamos que o desenvolvedor Snicoll, do projeto *Spring Initializr*, foi o mais produtivo globalmente, enquanto o desenvolvedor Boni, do projeto *Selenium*, também se destacou em termos de produtividade.

Figura 20 - Produtividade dos desenvolvedores para o ano de 2023

Fonte: Própria

A Figura 21 apresenta a quantidade de tempo gasto em horas no ano de 2023, de três atividades nos repositórios *Selenium* e *Spring Initializr*, essas atividades foram retornadas da consulta na base de dados e processadas até a camada de visão. A imagem destaca como um recurso pode ser empregado e transformado em métricas na plataforma GMC. Essa métrica foi determinada pela diferença entre a data e a hora em que o desenvolvedor iniciou a atividade, englobando uma ou várias tarefas, até a conclusão e subsequente submissão ao repositório *GitHub*. Especificamente, o desenvolvedor denominado Joerg dedicou 102 horas para corrigir a questão relacionada a "*Fixed the URI in a log line*".

Figura 21 - Quantidade de tempo nas atividades



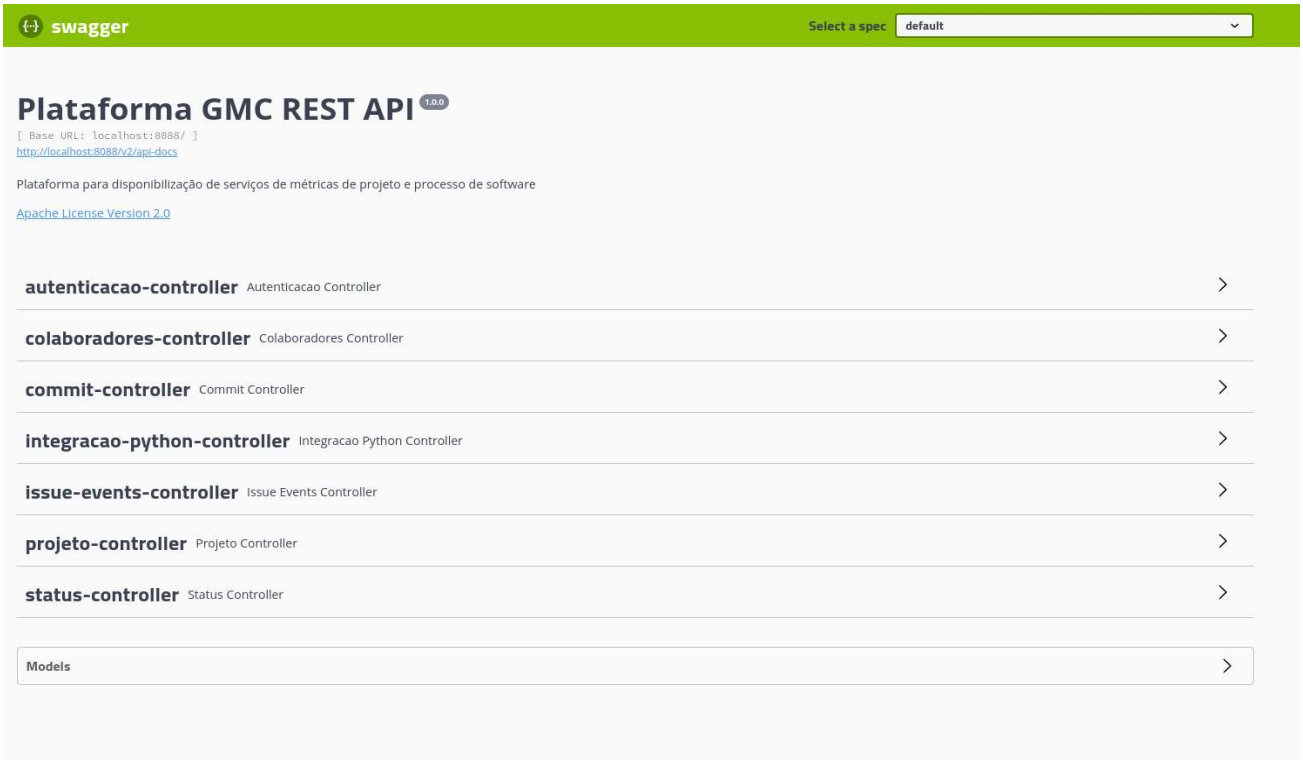
Fonte: Própria

Diversas consultas ou *scripts* podem ser executados no banco de dados *NoSQL* para recuperar os dados destinados ao processamento na camada de negócio, antes de serem apresentados na camada de visão. Um conjunto de códigos-fonte, elaborado em *JavaScript* e associado a essas consultas, é responsável por grande parte do processamento na plataforma GMC, visando a apresentação de indicadores, tabelas ou gráficos. Os exemplos de consulta mencionados estão disponíveis *online* e podem ser acessados em código_consultas[16].

Para compor a documentação técnica da plataforma GMC, a Figura 22 apresenta uma lista dos serviços disponíveis na plataforma GMC. Cada serviço, acessível na camada de visão, está disponibilizado e documentado para testes na interface do Swagger. O *Swagger* desempenha um papel crucial ao documentar esses serviços de maneira prática e organizada, facilitando integrações com outros sistemas. Utilizando um formato JSON, o Swagger disponibiliza uma abordagem padronizada para descrever *APIs REST* da plataforma. A documentação gerada pelo *Swagger* vai além da simples descrição, incluindo uma funcionalidade de testar interativamente os *endpoints* da *API* diretamente na própria interface documentada.

[16] *Scripts* utilizados: https://zenodo.org/records/10574281/files/scripts_consultas_gerais.js?download=1

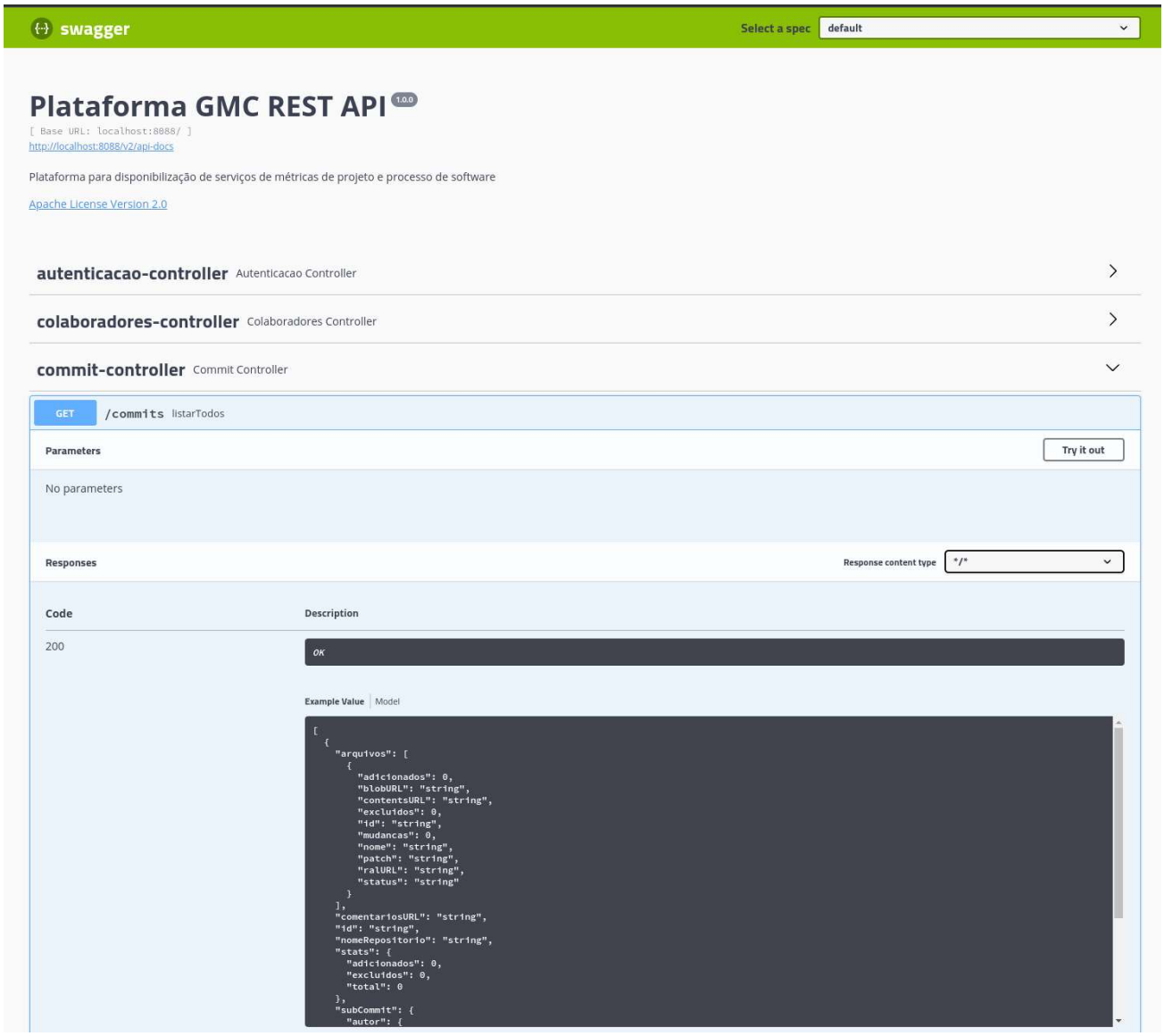
Figura 22 - Lista de serviços por categorias



Fonte: Própria

Além disso, o *Swagger* da plataforma GMC possibilita a definição de esquemas de dados para as entradas e saídas dos *endpoints* da API. Essa capacidade não apenas organiza a estrutura dos serviços, mas também simplifica a validação automática, assegurando que apenas dados válidos sejam transmitidos e recebidos pela *API*. Um exemplo está na Figura 23 onde o método *GET* dentro do serviço commit-controller podemos verificar o JSON de retorno da consulta.

Figura 23 - Detalhes da API de serviços



Fonte: Própria

4.2. Avaliação dos tipos de métricas disponibilizadas pela Plataforma

Neste capítulo, serão apresentados os detalhes da avaliação das métricas investigadas durante a pesquisa, bem como os resultados obtidos a partir desse processo de avaliação, proporcionando uma visão abrangente do impacto e relevância das métricas no contexto de desenvolvimento de *software*.

A Plataforma GMC na sua primeira versão dá suporte a 7 (sete) métricas de *software*, conforme detalhado anteriormente na Tabela 7. Com o objetivo de avaliar se as referidas métricas são efetivamente utilizadas na indústria, assim como, identificar outras métricas utilizadas poderiam ser incorporadas futuramente na Plataforma GMC, foi realizado um *survey* com alguns desenvolvedores de *software*.

Esta avaliação foi realizada por meio de um questionário composto por nove perguntas,

conforme apresentado no perguntas_questionário[17] e disponível *online*. O questionário foi desenvolvido com base nas etapas exploradas no capítulo 1.3 desta pesquisa, estruturando as perguntas no contexto de métricas de processo e projeto de *software*, com foco em dados extraídos do GitHub. Algumas questões foram formuladas de maneira aberta, permitindo que os avaliadores mencionassem outras ferramentas ou métricas de *software* pertinentes, outras questões de múltiplas escolhas, para delimitar o conteúdo abordado. Devido ao prazo limitado de duas semanas para coletar essas informações, o questionário não passou por teste piloto, mas foi realizada a validação pela orientadora Prof. Dra. Juliana Dantas Ribeiro Viana de Medeiros e coorientadora Prof. Dra. Heremita Brasileiro Lira, ambas fazem parte do corpo docente do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba.

O questionário foi aplicado a profissionais com experiência passada ou atual em desenvolvimento de *software*. A disseminação da pesquisa ocorreu de maneira efetiva, utilizando e-mails e mensagens instantâneas de comunicação direta, durante um intervalo de 10 dias no mês de novembro de 2023. Ao analisar as abordagens adotadas por profissionais na avaliação do desempenho de projetos ou processos de desenvolvimento de *software*, tratado na primeira questão (Q1), percebe-se uma variedade de respostas, conforme ilustrado na Figura 24.

Figura 24 - Texto resumido das respostas Q1

Utilização de métricas no desenvolvimento de *software*, utilizando relatórios gerados pelo Jira. Aborda pontos como APF, Story Points, resoluções em 1º nível, *bugs* recorrentes, *bugs* novos por versão, tempo médio de reparo, tempo médio entre falhas, índice de indisponibilidade do sistema e dados das ferramentas atuais. Uso do Git e Jira para medir desempenho em branches, alterações de código, tempo de resolução e produtividade da equipe. Métricas individuais, como a avaliação baseada na quantidade de atividades realizadas, assim como a observação da cobertura de testes e o tempo gasto em branches. Diferentes abordagens de medição, incluindo a preferência por avaliar através da entrega e a utilização de ferramentas do *GitHub* e extensões do Visual Studio Code. A medição de desempenho, tanto em setores públicos quanto privados, muitas vezes é feita de forma ad-hoc, com a gestão baseada em dados sendo frequentemente negligenciada.

Fonte: Própria

A segunda questão (Q2) teve por objetivo avaliar a utilização da API de métricas *GitHub* disponível em: Documentação API-Métricas GitHub. Destaca-se uma informação relevante: 95% dos participantes que responderam ao questionário afirmaram nunca ter utilizado a API de métricas disponibilizada pelo *GitHub*. 5% dos participantes optaram por explorar a API de métricas do *GitHub* de maneira experimental, empregando análise para avaliar as alterações e *commits* efetuados.

Além disso, destaca-se o uso de ferramentas especializadas, como o Sonar, desempenhando um

[17] Perguntas do questionário: https://zenodo.org/records/10574281

papel crucial na avaliação do código, incluindo a identificação e quantificação de erros. No âmbito da gestão de projetos, o *Open Project*[18] é utilizado como uma ferramenta integral para coordenar e administrar as diferentes fases do projeto. Entre as observações relevantes, ressalta-se que algumas respostas enfatizam que a medição de desempenho é primariamente conduzida com foco na entrega e na qualidade do produto final, em detrimento da ênfase no tempo despendido em tarefas específicas. Ademais, é mencionado que, embora ferramentas disponíveis no *GitHub* sejam utilizadas por alguns profissionais, a avaliação de desempenho frequentemente é negligenciada, apesar da presença dessas ferramentas.

A questão (Q3) foi formulada para a avaliação das métricas, conforme relatado neste estudo. As métricas listadas na Tabela 7 estão associadas à questão específica do questionário. Optou-se por empregar uma abordagem mais contextual na descrição das métricas, alinhando o texto de maneira mais próxima da aplicação prática. As opções disponíveis para a avaliação incluíram: "Essencial", indicando uma importância significativa; "Pode ser útil", sugerindo utilidade potencial; "Irrelevante", denotando falta de pertinência; e "Não sei avaliar", para situações em que a avaliação não pôde ser determinada com segurança. A tabela 12 mostra o resumo da avaliação.

Tabela 12 - Resumo da avaliação de métricas do questionário

Descrição	Avaliação Geral
Quantidade de alterações em um determinado tempo	Pode ser útil
Quantidade de arquivos de código alterados em um módulo do *software*	Pode ser útil
Desempenho individual baseado em entregas e duração das atividades	Pode ser útil
Desempenho da equipe baseado no tempo decorrido das entregas das atividades	Essencial
Quantidade de linhas de código alteradas em um grupo de atividades	Irrelevante
Número de alterações realizadas por determinado desenvolvedor	Pode ser útil
Quantidade de pessoas que alteraram determinadas atividades	Pode ser útil
Tecnologias utilizadas nas demandas no projeto	Pode ser útil
Tempo estimado da atividade baseado no projeto	Essencial
Categorias das atividades baseado na descrição	Pode ser útil

Fonte: Própria

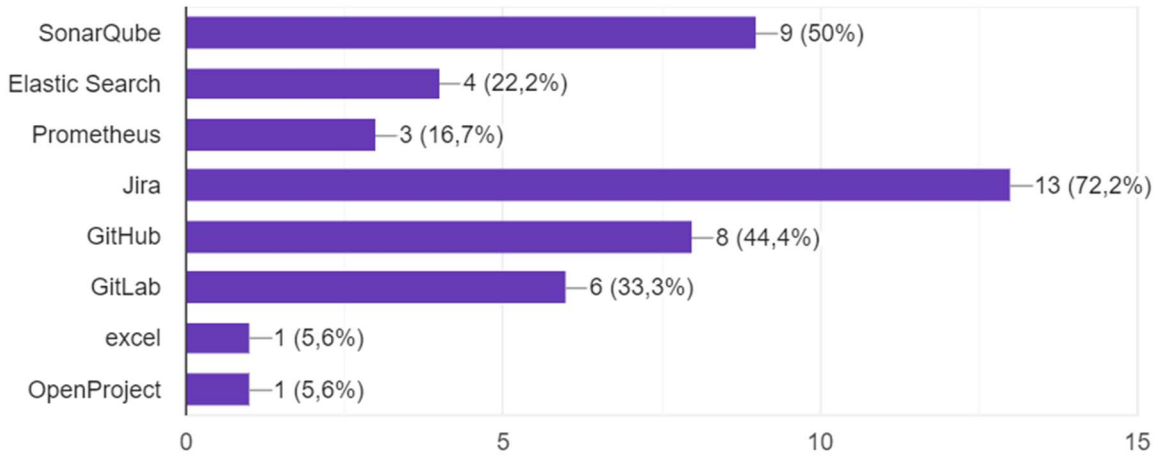
Quanto à adoção de ferramentas para o monitoramento de métricas aborada na questão (Q4), observa-se que a maioria dos participantes, representando 72%, emprega o *Jira Software*. O *SonarQube* [19]é utilizado por 50% da amostra, enquanto o *GitHub* é a ferramenta de escolha para 44%. Em seguida,

[18] Plataforma de Gerenciamento de Projetos de código aberto: https://www.openproject.org/

[19] Plataforma utilizada para avaliar a qualidade do código em projetos de software: https://www.sonarsource.com/products/sonarqube/

33% dos participantes fizeram uso do *GitLab*[20]. Mencionam-se, em menor proporção, outras ferramentas, tais como *Elasticsearch*[21], *Prometheus*[22], *OpenProject* e *Microsoft Excel*[23], contudo, estas apresentam uma taxa de utilização relativamente reduzida. Este panorama revela uma clara preferência pelas ferramentas mais estabelecidas, enquanto também destaca a diversidade de escolhas, mesmo que em menor escala, evidenciando um cenário variado no uso de ferramentas para o acompanhamento de métricas conforme Figura 25.

Figura 25 - Ferramentas utilizadas para acompanhar métricas



Fonte: Própria

Na questão (Q5), em relação à avaliação da eficácia das ferramentas utilizadas para medir métricas de projetos e processos de software, os profissionais de software destacaram aspectos fundamentais que influenciam sua percepção. O *Jira* recebeu avaliações positivas, especialmente quando empregado por equipes ágeis que adotam a prática de quebrar tarefas em pequenas atividades. A ênfase recai na importância de uma cultura de utilização eficiente, onde todos os membros da equipe, inclusive o *Product Owner* [24](P.O.), participam ativamente.

Para entender melhor como é feito o controle mais eficaz e aprimorado da qualidade de *software*, a questão (Q6) obteve respostas que alguns especialistas recorrem a recursos como relatórios do *Jira*, Análise de Pontos de Função (APF), *Story Points (SP)*, *First Call Resolution* (FCR), além de métricas como *bugs* novos e recorrentes, tempo médio de reparo, tempo médio entre falhas e o índice de indisponibilidade do sistema. Essa variedade reflete a adaptabilidade e a contextualização das estratégias de avaliação de desempenho na prática profissional desses especialistas em

[20] Plataforma de gerenciamento de ciclo de desenvolvimento de software: https://about.gitlab.com/
[21] Mecanismo de análise de dados distribuídos e de código aberto: https://www.elastic.co/pt/
[22] Sistema de monitoramento de métricas de sistemas e serviços de código aberto: https://prometheus.io/
[23] Ferramenta para criação de planilhas, realização de cálculos e análise de dados: https://www.microsoft.com/pt-br/microsoft-365/excel
[24] Representa os interesses do cliente e define as funcionalidades do produto no framework ágil Scrum: https://www.scrum.org/

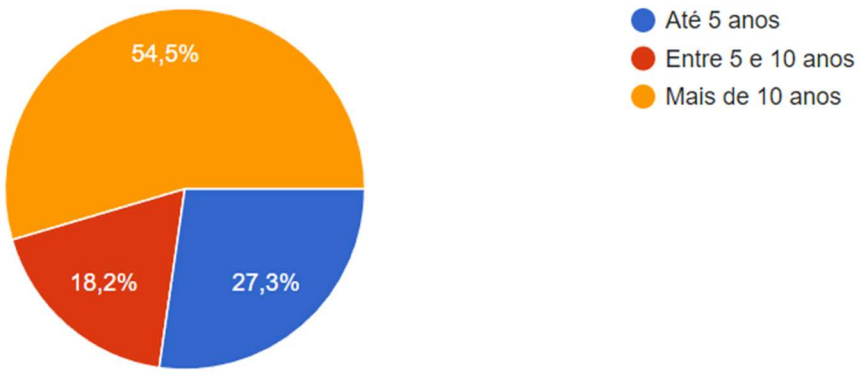
desenvolvimento de *software*.

A questão (Q7) deixa livre para o profissional sugerir métricas para acompanhamento de projeto de *software*, as respostas apresentam uma variedade de sugestões de métricas para avaliar e melhorar o desenvolvimento de *software*. Dentre as propostas, destaca-se o foco no tempo médio por complexidade como uma métrica de eficiência, buscando entender quanto tempo é necessário para concluir atividades de diferentes níveis de complexidade ao longo de uma sprint. Além disso, são discutidas métricas relacionadas à qualidade do *software*, como falhas de implementação, índice de previsibilidade, cobertura de testes e taxa de erros/falhas. A ênfase nas métricas de projeto e processo como previsibilidade de lançamentos futuros, identificação de *bugs* em produção e análise de requisitos imaturos reflete uma abordagem abrangente para melhorar a qualidade do produto final.

Outras métricas gerenciais, como o índice de *turnover* (rotatividade de pessoas na equipe) e o índice de tempo de resposta do cliente, são sugeridas como indicadores importantes para avaliar o impacto humano e a eficácia na comunicação com os clientes. O uso do *code review* como uma prática eficaz para analisar a qualidade do desenvolvimento é enfatizado, destacando a importância da revisão de código como uma oportunidade para compreender as implementações, aprender e promover a troca de conhecimento entre membros da equipe. Essa abordagem é vista como uma forma de garantir não apenas a qualidade técnica, mas também a transferência eficaz de conhecimento dentro da equipe de desenvolvimento.

Quanto à experiência profissional na área de desenvolvimento de *software* abordada na questão (Q8) e exibida na Figura 26, os resultados indicam que 54,5% dos participantes acumulam mais de 10 anos de vivência nesta esfera, evidenciando um contingente significativo de profissionais com vasta experiência. Adicionalmente, 27,3% possuem menos de 5 anos de experiência, enquanto 18,2% situam-se no intervalo de 5 a 10 anos de atuação no campo do desenvolvimento de *software*. Essa distribuição de anos de experiência proporciona uma compreensão abrangente do perfil profissional dos participantes da pesquisa.

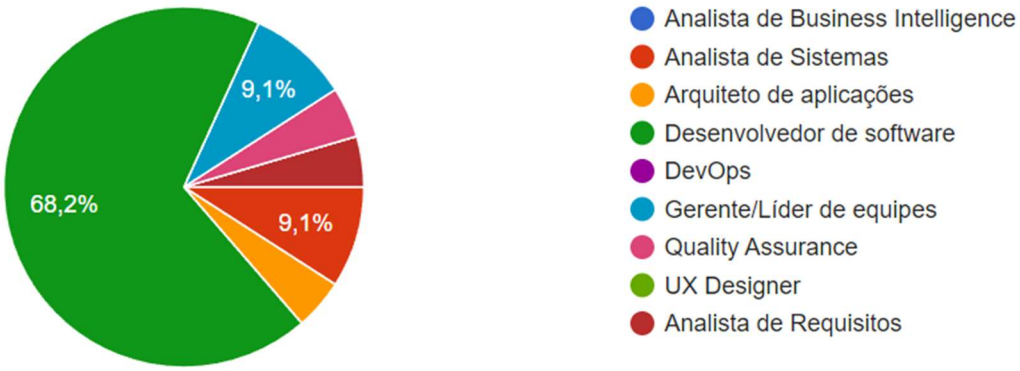
Figura 26 - Experiências dos participantes



Fonte: Própria

A questão (Q9) aborda de maneira direta as responsabilidades dos profissionais no âmbito do desenvolvimento de *software* demonstrada na Figura 27. No referido questionário, 68,2% dos participantes se autodeclararam desenvolvedores de *software*, enquanto 9,1% desempenham funções como Gerentes de Projetos, Líderes de Equipes ou Analistas de Sistemas. Os demais participantes englobam profissionais atuantes em diversas áreas, tais como Garantia de Qualidade (QA), Analista de Requisitos e Arquiteto de Aplicações.

Figura 27 - Atuação dos participantes



Fonte: Própria

Em resumo, as respostas fornecem uma visão abrangente e crítica das métricas, abordando aspectos como eficiência, qualidade técnica, gestão de equipe e comunicação, sugerindo uma abordagem holística para avaliação e melhoria do processo de desenvolvimento de *software*. A flexibilidade e a variedade de métricas oferecidas pelo Jira foram ressaltadas, com particular destaque para a capacidade de avaliar a entrega de histórias e tarefas estimadas. Essa versatilidade é apontada como essencial, visto que cada empresa e equipe possui características únicas no contexto de métricas de desenvolvimento de *software*. O *Elasticsearch* também foi mencionado positivamente, sendo reconhecido por sua utilidade em projetos críticos. Esses profissionais destacaram a capacidade desta ferramenta em fornecer métricas detalhadas e variadas, com foco especial no monitoramento de recursos.

O *GitLab* recebeu reconhecimento pela sua capacidade em atender às necessidades de desenvolvimento, testes e *DevOps*. Esta ferramenta foi apontada como eficaz na entrega do necessário para o desenvolvimento de *software* dentro do escopo da equipe, demonstrando sua relevância em ambientes multifuncionais. A avaliação do *SonarQube* baseia-se na quantidade de erros críticos identificados pela ferramenta, ressaltando a importância da qualidade do código na análise de métricas. Algumas respostas mencionaram a utilidade do Jira para atribuição de tarefas e acompanhamento do progresso, evidenciando sua função central na gestão de projetos.

A questão sobre ações para um controle mais eficaz e aprimoramento da qualidade do *software* recebeu respostas abrangentes e diversificadas dos participantes. Algumas sugestões destacam a

importância de um *Scrum Master*[25] eficiente, capaz de ajustar o processo conforme a realidade da equipe, e a realização de reuniões mensais para acompanhamento e ajustes no processo, demonstrando uma abordagem ágil na gestão. Outras sugestões enfatizam a necessidade de investimento de esforço dos gestores, líderes e equipe na eficácia da comunicação, estabelecendo objetivos claros para cada métrica a ser coletada. A atenção à utilização correta de padrões de projeto, como SOLID e *clean code*[26], também é ressaltada para evitar práticas indesejadas como *OverEngineering*[27].

Houve destaque para a importância da fase inicial de entendimento e elaboração do anteprojeto, envolvendo uma equipe multidisciplinar de negócios e tecnologia. A sugestão de projetar e finalizar cada módulo separadamente, em vez de todos simultaneamente, visando evitar possíveis impactos negativos no projeto como um todo. A padronização e a reavaliação periódica de códigos foram apontadas como práticas essenciais, assim como o foco em planejamento, análise de negócios e análise de requisitos. Algumas respostas destacam a necessidade de implementar práticas como integração contínua, entrega contínua, testes automatizados extensivos e desenvolvimento guiado por comportamento.

Outras sugestões incluem a obrigatoriedade de registrar o tempo gasto em tarefas, o foco na usabilidade e a realização de homologações bem elaboradas com os usuários do sistema. A ideia de ter um segundo desenvolvedor avaliando o código para uma segunda opinião é sugerida como uma prática valiosa. Em geral, as respostas evidenciam a importância do planejamento eficaz, da comunicação transparente, da utilização de padrões de projeto, da implementação de práticas ágeis e da revisão contínua para um controle mais eficaz e aprimoramento da qualidade do *software* com métricas.

A indagação acerca das sugestões de métricas para controle e aprimoramento da qualidade do *software* obteve respostas que abordam uma gama variada de aspectos relacionados ao monitoramento e avaliação. Dentre as sugestões apresentadas, destacam-se métricas-chave como o tempo médio por complexidade, falhas de implementação e índice de previsibilidade, volume de alterações e categorização de atividades, índices gerenciais, cobertura de testes de *software* e taxa de erros/falhas, além da métrica geral de entrega e o *code review* [28]como uma métrica de qualidade. As respostas podem ser acessadas online através do respostas_questionário[29].

---

[25] Desempenha um papel de liderança e facilitação para ajudar a equipe no *framework* Scrum: https://www.scrum.org/

[26] Estilo de programação que enfatiza a clareza, conceito popularmente criado por Robert C. Martin: https://blog.cleancoder.com/

[27] Refere-se à prática de criar uma solução mais complexa ou robusta: https://blog.cleancoder.com/

[28] Prática comum no desenvolvimento de software para revisão de código de outras pessoas da equipe: https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/

[29] Respostas do questionário: https://zenodo.org/records/10574281

5. CONCLUSÃO

Este capítulo relembra as contribuições da pesquisa, e apresenta algumas limitações e perspectivas para trabalhos futuros. Essa reflexão é crucial para garantir uma compreensão abrangente e confiável dos resultados obtidos. Ao reconhecer e analisar essas limitações, é possível fornecer uma contextualização para a interpretação das informações, contribuindo para uma visão mais completa e informada da pesquisa realizada.

5.1. Contribuições da Pesquisa

O objetivo principal deste estudo é auxiliar os tomadores de decisão envolvidos em projetos de *software* através da disponibilização de métricas de processo e projeto de *software* extraídos dos repositórios *GitHub*. destacadas em trabalhos científicos e amplamente adotadas na indústria com o objetivo de auxiliar os tomadores de decisão que atuam em projetos de *software* através da disponibilização de métricas de extraídas de repositórios *GitHub*. A primeira contribuição a ser destacada foi o mapeamento das principais métricas de processo e projeto de software recomendadas na literatura (Tabela 1).

Outra contribuição é a Plataforma GMC que um software para visualizar métricas de processo e projeto de *software* extraídos dos repositórios *GitHub* por meio de indicadores, tabelas e gráficos. O código-fonte da Plataforma está acessível no repositório do *GitHub como* front-end-gmc[30], backend-gmc[31], adaptacao_gddownloader[32]. Qualquer interessado pode fazer o download e realizar modificações de acordo com suas necessidades, seguindo sempre os princípios da licença GNU AGPLv3, sem incorrer em custos.

A arquitetura modular da plataforma GMC foi concebida para simplificar integrações futuras por meio de *APIs*, tornando-se assim uma opção alternativa para consolidar dados e apresentar métricas extraídas de dados do *GitHub* e relacionadas a projetos e processos de *software*. Este trabalho apresenta a arquitetura de forma detalhada, incluindo uma explicação minuciosa de cada etapa realizada para explorar dados e transformá-los em métricas acessíveis à equipe de desenvolvimento de *software*.

Uma das vantagens da plataforma GMC em comparação com a *API* de métricas do *GitHub* reside na capacidade de personalização das métricas e na gestão da quantidade de requisições HTTP efetuadas na plataforma para consultas. Todo o processamento pode ser ajustado de acordo com a infraestrutura local ou na *Internet* do ambiente utilizado, não dependendo mais exclusivamente de servidores remotos. As métricas podem ser facilmente implementadas e visualizadas na aplicação da camada de visão. Foram desenvolvidos módulos para a segurança dos dados da aplicação, adaptáveis às necessidades da empresa. Dessa forma, a plataforma GMC surge como uma alternativa para a

[30] Repositório do *GitHub* para camada de visão: https://github.com/matiasribeiro/front-end-gmc
[31] Repositório do *GitHub* para camada de negócio: https://github.com/matiasribeiro/back-end-gmc
[32] Repositório do *GitHub* para a adaptação do GDDownloader: https://github.com/matiasribeiro/adaptacao_gddownloader

exibição de métricas com base no uso de dados armazenados no repositório *GitHub*. A Plataforma GMC permite que as métricas possam ser destacadas de diversas maneiras, com alternativas para sua visualização. Vale ressaltar que essas métricas não estão disponíveis em todas as ferramentas ou *softwares* utilizados, conforme indicado pelas respostas da avaliação e pela pesquisa exploratória realizada.

Por fim, podemos destacar como contribuição os instrumentos utilizados para avaliação da Plataforma GMC conforme detalhado na seção 4.2. O processo de avaliação foi feito por meio de um questionário respondido por profissionais do setor de desenvolvimento de *software*, provenientes de empresas públicas e privadas no Brasil. Essa avaliação procurou analisar a relação entre as informações exibidas na plataforma GMC e a perspectiva de profissionais.

5.2 Limitações da Pesquisa

Nesta seção, discutimos as limitações encontradas durante a condução da pesquisa, com o objetivo de identificar os fatores que possam influenciar e comprometer os resultados. A análise dessas limitações é crucial para assegurar a integridade e a confiabilidade do estudo, uma vez que possibilita uma reflexão crítica sobre os potenciais impactos nas conclusões obtidas.

Uma limitação identificada neste estudo está associada à pesquisa exploratória. Em decorrência da busca direta nos repositórios de trabalhos científicos, é possível que alguns trabalhos relevantes tenham sido excluídos do escopo da pesquisa que mapeou as métricas a serem contempladas na Plataforma. A definição do conjunto de métricas suportado pela Plataforma poderia ser melhorada e complementada também com a realização de um estudo de caso. Essa abordagem multifacetada poderia aprimorar significativamente a compreensão abrangente e aprofundada das métricas em questão. Entretanto, não foi realizada devido a restrições de tempo.

Outra limitação está nos projetos utilizados como base para extração de dados, apesar de conter dados suficientes para avaliações, acrescentar mais projetos para análise e avaliação poderia reforçar as métricas pesquisadas. Projetos monitorados em tempo real dentro do contexto de desenvolvimento de *software* poderia aprimorar as informações sobre métricas e consequentemente as exibições das informações na plataforma GMC.

A concepção e avaliação do questionário utilizado para mensurar as métricas investigadas neste estudo poderiam ser refinadas. O período alocado para a coleta de dados desempenhou um papel crucial e pode ter resultado em lacunas, possivelmente deixando de contemplar alguma métrica relevante no âmbito de processos ou projetos de *software*. Ademais, observa-se que a quantidade de pessoas que responderam ao questionário poderia ter sido maior, o que poderia contribuir para uma análise mais robusta e representativa das métricas em questão.

O método de coleta de informações pode ser aprimorado por meio de entrevistas ou monitoramento em tempo real de projetos de *software*, visando obter dados mais precisos e

informações abrangentes sobre as métricas de *software*. A plataforma GMC, por exemplo, poderia ser submetida a uma avaliação em tempo real em um projeto piloto de desenvolvimento de *software*. Isso permitiria que os membros envolvidos no projeto avaliassem as métricas apresentadas e trabalhassem em conjunto para refiná-las, promovendo assim uma melhoria contínua.

Mesmo diante das limitações, as contribuições deste trabalho foram significativas no contexto de métricas de *software*. Foi possível obter uma visão abrangente das métricas de processo e projeto de *software*, conforme utilizadas e descritas em trabalhos científicos, apresentando uma visualização simplificada por meio de uma plataforma. A exploração cuidadosa das limitações inerentes a este estudo ressalta nosso compromisso com a integridade científica e a transparência. Ao identificar essas limitações, almejamos contribuir para o desenvolvimento de futuras pesquisas mais robustas e abrangentes sobre métricas de *software*. Convidamos os leitores a refletirem sobre essas considerações e a participarem do contínuo diálogo que impulsiona a evolução do conhecimento na área.

5.3. Trabalhos Futuros

A pesquisa sobre métricas de *software* e a criação da nova plataforma GMC, destinada ao gerenciamento de métricas de processo e projeto de *software*, geram novos desafios e oportunidades que evidenciam a necessidade de dar continuidade a este trabalho. A seguir, delinearemos uma visão abrangente de vários aspectos a serem explorados em futuras pesquisas, com o objetivo de fornecer informações valiosas:

- Integrar outros repositórios de código, como o *GitLab*, e estabelecer relações entre dados para processar métricas de maneira mais precisa e clara é fundamental. Isso se deve ao fato de que cada repositório opera com uma API de serviços distinta. Em outras palavras, este trabalho pode estruturar uma infraestrutura para conectar repositórios de código, visando obter métricas específicas que não são facilmente visualizadas nos repositórios de origem.

- Prosseguir com a pesquisa e investigar outras métricas no ciclo de desenvolvimento de *software*, expandindo o escopo da pesquisa incorporando algoritmos adicionais para o cálculo de métricas.

- Desenvolver opções de filtragem de informações na plataforma GMC, permitindo personalização pelo usuário, possibilitando consultas dinâmicas e interativas na base de dados.

- Modificar o código fonte para integrar diversas APIs, aumentando assim o volume de dados a ser processado. Personalizar o módulo de segurança para controle de acesso conforme lei geral de proteção de dados pessoais (LGPD), também conhecida como Lei nº 13.709/2018.

- Fazer um estudo de validação das métricas exploradas neste trabalho com uma empresa de

software por um período específico para analisar a contribuição efetiva que a equipe de desenvolvimento pode obter.’’

- Conduzir entrevistas com profissionais de desenvolvimento de *software* a respeito de métricas e realizar uma análise qualitativa dos dados resultantes.

Os futuros trabalhos planejados têm como meta a contínua otimização da pesquisa em métricas de *software*, visando fornecer um suporte aos profissionais envolvidos no desenvolvimento de projetos em variados contextos. Essas iniciativas, ao ampliarem a capacidade de análise e personalização das métricas, têm o potencial não apenas de agilizar a tomada de decisões, mas também de proporcionar novos detalhes. Ao adaptar o processo de coleta de dados e promover a integração com diversas fontes, busca-se criar uma base robusta para análises detalhadas, contribuindo assim para a eficácia e eficiência no gerenciamento de projetos de *software*.

REFERÊNCIAS

BERGEL, Alexandre. Generating Visualizations From GitHub. In: Agile Visualization with Pharo. Apress, Berkeley, CA, 2022. p. 233-261.

BRANCO, G. Arthur. Análise da Influência do GitHub Actions sobre a Qualidade do Código. Belo Horizonte, MG. 2022. Disponível em: http://bib.pucminas.br:8080/pergamumweb/vinculos/000011/0000118d.pdf. Acesso em: 04 jan. 2023.

CHACON, Scott; STRAUB, Ben. Pro Git: Everything you need to know about Git. Git, 2022. Disponível em: https://git-scm.com/. Acesso em: 21 nov. 2022.

DE BASSI, Patricia Rücker et al. Measuring developers' contribution in source code using quality metrics. In: 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2018. p. 39-44.

DE BRITO, Lucas Marcel Silva; BERNARDO, João Helis. Classwork: Uma ferramenta de acompanhamento em tempo real da contribuição individual de alunos de cursos de computação no desenvolvimento de projetos de software acadêmicos hospedados no GitHub. In: Anais do XXXII Simpósio Brasileiro de Informática na Educação. SBC, 2021. p. 1128-1139.

DE LIMA RAMOS, RAFAEL ANDERSON. Monitoramento de métricas de qualidade e produtividade em projetos ágeis de software através da integração de dados extraídos de ferramentas de gestão e testes. Orientadora: Juliana Dantas Ribeiro Viana de Medeiros. 2023. Dissertação (Mestrado) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, João Pessoa, PB, 2022.

DIAMANTOPOULOS, Themistoklis et al. Employing Contribution and Quality Metrics for Quantifying the Software Development Process. 17th International Conference on Mining Software Repositories (MSR '20), New York, NY, USA, p. 558-562, 2020. Disponível em: https://dl.acm.org/doi/10.1145/3379597.3387490. Acesso em: 6 dez. 2021.

ELASTIC STACK. ELASTIC. 2023. Por que se chama Elastic Stack? O que é o ELK Stack?. Disponível em: https://www.elastic.co/pt/what-is/elk-stack. Acesso em: 16 jan. 2023.

FENTON, Norman; BIEMAN, James. Software Metrics: A Rigorous and Practical Approach. 3. ed. Broken Sound Parkway NW: CRC press, 2014.

GITHUB et al. Getting started with GitHub documentation: Learn how to start building, shipping, and maintaining software with GitHub. Explore our products, sign up for an account, and connect with the world's largest development community.. [S. l.], 1 fev. 2023. Disponível em: https://docs.github.com/pt/get-started/quickstart/hello-world. Acesso em: 8 fev. 2023.

GITHUB REST API. Quickstart. 2022. Disponível em: https://docs.github.com/pt/rest/quickstart?apiVersion=2022-11-28. Acesso em: 20 mar. 2023.

HERNÁNDEZ, Giovanni et al. Métricas de productividad para equipo de trabajo de desarrollo ágil de software: una revisión sistemática, v. 22, p. 63-68, 2019. Disponível em: https://doi.org/10.22430/22565337.1510. Acesso em: 18 maio 2021.

JIANG, Li; JIANG, SHUJUAN; GONG, Lina; DONG, Yue; YU, QIAO. Which Process Metrics Are Significantly Important to Change of Defects in Evolving Projects: An Empirical Study. IEEE Access, IEEE, v. 8, p. 93705 - 93722, 2020. DOI 10.1109/ACCESS.2020.2994528. Disponível em: https://ieeexplore.ieee.org/abstract/document/9094061. Acesso em: 20 set. 2021.

Jira Software. O que é o Jira Software? Disponível em:

<https://www.atlassian.com/br/software/jira/guides/getting-started/introduction#dig-into-specific-features\>. Acesso em: 12 de set. 2023.

KOZIK, R.; CHORÁS, M.; PUCHALSKI, D.; RENK, R. Q-Rapids framework for advanced data analysis to improve rapid software development. In: J Ambient Intell Hum Comput, 2018.

LIMA, Jalerson et al. Avaliação da contribuição de desenvolvedores com métricas baseadas em mineração de repositório. In: Congresso Internacional de Manutenção e Evolução de Software (ICSME) da IEEE, 2015.p. 536-540.

MEJÍA, Jezreel et al. A proposal of metrics for software development based on the ISO/IEC 29110 standard. In: 2021 10th International Conference On Software Process Improvement (CIMPS). IEEE, 2021.

MLADENOVA, Tsvetelina. Software Quality Metrics–Research, Analysis and Recommendation. In: 2020 International Conference Automatics and Informatics (ICAI). IEEE, 2020. p. 1-5.

NICOLETTE, David. Software Development Metrics. Shelter Island, NY: Manning Publications Co, 2015. ISBN 9781617291357.

OCTOVERSE REPORT (org.). The state of open source software, 2023. Disponível em: https://octoverse.github.com/. Acesso em: 14 fev. 2023.

OLIVEIRA, Edson César Cunha de. Fatores de influência na produtividade de desenvolvedores de organizações de software. 2017. 187 f. Tese (Doutorado em Informática) - Universidade Federal do Amazonas, Manaus, 2017.

PAPAMICHAIL, Michail D. et al. Contribution and Quality Metrics for Quantifying the Software Development Process Dataset. Dataset, Ano da Publicação. Disponível em: https://zenodo.org/record/2556151. Acesso em: 19 jun. 2022.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de software: Uma abordagem Profissional. 9ª ed. rev. Porto Alegre: McGraw Hill Brasil, 2021. p.460-487.

PRESTON WERNER, Tom et al. Documentação do GitHub: Obtenha ajuda, onde quer que você esteja em sua jornada no GitHub. GitHub, 2022. Disponível em: https://docs.github.com/pt. Acesso em: 09 maio 2022.

RAM, Prabhat et al. Actionable Software Metrics: An Industrial Perspective. In: Proceedings of the Evaluation and Assessment in Software Engineering. 2020. p. 240-249.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson Education do Brasil, 2018. ISBN 978-85-430-2497-4.

SONARQUBE. SonarQube Documentation. Disponível em: https://docs.sonarqube.org/latest. Acesso em: 18 jan. 2023.

SPRING FRAMEWORK. Documentação Oficial. Disponível em: https://docs.spring.io/spring-framework/reference/overview.html. Acesso em: 11 jul. 2023.

SWAGGER et al. What Is Swagger?. Swagger is a set of open-source tools built around the OpenAPI Specification that can help you design, build, document and consume REST APIs. 1. fev. 2023. Disponível em: https://swagger.io/docs/specification/about/. Acesso em: 10 fev. 2023.

VENIGALLA, AKHILA SRI MANASA; BOYALAKUNTA, KOWNDINYA; CHIMALAKONDA, SRIDHAR. GitQ- Towards Using Badges as Visual Cues for GitHub Projects. Research in Intelligent Software & Human Analytics (RISHA) Lab, Indian Institute of Technology Tirupati, India, 8 Jul. 2021.

Disponível em: https://arxiv.org/abs/2107.03761. Acesso em: 17 nov. 2021.

WAZLAWICK, Raul. Engenharia de software: conceitos e práticas. Elsevier Editora Ltda. 2ª ed. cap. 9. Gerenciamento de Projeto de Software. 2019.

WOLTER, Thomas. Creating, Measuring and Evaluating a Metric for Software Engineering Performance. 2022.

ZABARDAST, Ehsan; GONZALEZ-HUERTA, Javier; TANVEER, Binish. Ownership vs contribution: Investigating the alignment between ownership and contribution. In: 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C). IEEE, 2022.