Centro Federal de Educação Tecnológica de Minas Gerais
Departamento de Computação
Engenharia de Computação

Lorena Gomes de Oliveira Cabral

Uma abordagem para comparação de bibliotecas de software em Java

Belo Horizonte, 2023

Lorena Gomes de Oliveira Cabral

Uma abordagem para comparação de bibliotecas de software em Java

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais como atividade avaliativa da disciplina de Trabalho de Conclusão de Curso I.

Centro Federal de Educação Tecnológica de Minas Gerais

Orientador: Prof. Dr. Eduardo Cunha Campos

Belo Horizonte

2023

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Bibliotecas, frameworks e APIs são ferramentas muito úteis no cenário de desenvolvimento de softwares. São compostas por métodos definidos que ajudam na comunicação com sistemas de software e obtenção de respostas de forma que um computador pode entender (LIBRARY..., s.d.). São também úteis para facilitar as tarefas de desenvolvimento, permitindo a reutilização de código escrito por terceiros (MORA; NADI, 2018). Praticamente, todos os grandes projetos conhecidos no mercado são implementados utilizando diversas bibliotecas como suporte para potencializar e reduzir o tempo gasto no progresso no desenvolvimento.

Contudo, escolher uma biblioteca adequada nem sempre é uma tarefa fácil para um programador. A escolha errada pode gerar impactos negativos em termos de custo, tempo e esforço (LARIOS VARGAS et al., 2020). Idealmente, uma biblioteca eficiente deve haver algumas características, como boa documentação e estabilidade. Mas existem uma extensa variedade de métricas que podem, juntas, compor aspectos que podem ou não fazer sentido para determinado problema enfrentado por um desenvolvedor.

Portanto, considerar todas as possíveis variáveis que envolvem bibliotecas de desenvolvimento de software e optar por uma dentre tantas opções pode demandar muito tempo. Na rotina de um time de desenvolvimento, dentre tantos prazos e metas, os principais fatores que determinam a decisão de um programador acabam sendo subjetivos (LARIOS VARGAS et al., 2020).

Dada a despadronização de fatores na escolhas de bibliotecas para projetos de desenvolvimento de software, o objetivo desse trabalho é criar uma referência de comparação, baseado em um conjunto de métricas, para ajudar programadores a fazerem escolhas coerentes conforme suas necessidades.

O trabalho está organizado da seguinte forma: o Capítulo 2 apresenta conceitos fundamentais para o entendimento do trabalho; o Capítulo 3 apresenta e compara trabalhos relacionados; o Capítulo 4 indica a metodologia utilizada no desenvolvimento deste trabalho; o Capítulo 5 apresenta o cronogramas e atividades desenvolvidas até o momento; o Capítulo 6 apresenta as conclusões preliminares.

2 Fundamentação Teórica

2.1 Linguagem de Programação Java

Java não é somente uma linguagem de programação, como também uma plataforma de computação (ORACLE, 2023b). Foi criada em 1995 pela Sun Microsystems e é considerada uma das ferramentas mais populares para programação, de acordo com o Tiobe Index (SOFTWARE, 2023). Foi projetada para ser simples, familiar, orientado a objetos, robusta, segura e independente de qualquer plataforma. Isso porquê, a **linguagem de programação** Java possui uma máquina virtual específica para cada ambiente que permite executar o arquivo bytecode, um código intermediário compilado a partir do arquivo fonte.

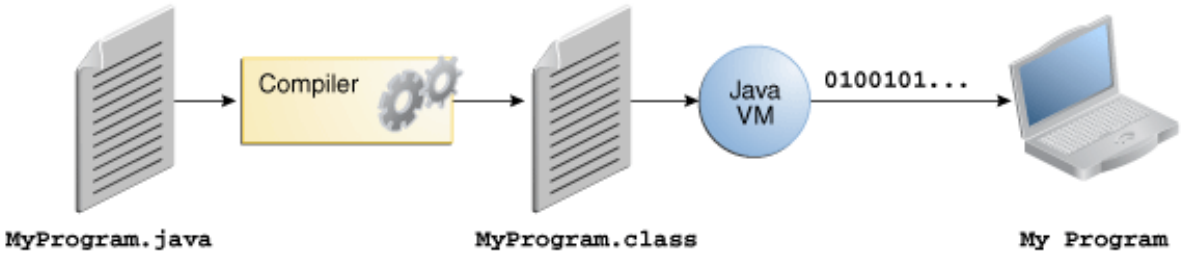

Figura 1 – Fluxo da linguagem de programação Java (ORACLE, 2023a)

Já a **plataforma** Java, composta pela interface de programação de aplicativos Java (API) e máquina virtul Java (JVM), diferente de algumas plataformas comuns como Microsoft Windows, Linux, Solaris OS e Mac OS, é baseada somente em software e executada sobre outras plataformas baseadas em hardware (ORACLE, 2023a).

Por suas características marcantes e grande repercussão entre programadores, Java possui também uma gama de bibliotecas, padrão e também de terceiros, que abrange desde manipulação simples de strings e operações de entrada/saída até recursos avançados, como acesso a banco de dados, programação de rede e criptografia.

A disponibilidade de uma ampla variedade de bibliotecas é uma das grandes vantagens do ecossistema Java. No entanto, essa diversidade também pode tornar a escolha da biblioteca ideal uma tarefa desafiadora. Nesse processo de seleção, é crucial considerar fatores como a qualidade e estabilidade da biblioteca, o nível de suporte fornecido pela comunidade de desenvolvedores, a disponibilidade de documentação detalhada e atualizada, além da reputação, quantidade e qualidade de contribuições. Esses aspectos podem ser indicativos da saúde e confiabilidade da biblioteca.

2.2 Apache Maven

Apache Maven é uma ferramenta de gerenciamento e compreensão de projetos de software criada pela Apache Software Foundation (THE APACHE SOFTWARE FOUNDATION, Accessed: 2023[a]). Sua principal intenção é facilitar o processo de criação de projetos baseados em Java, gerar um sistema de construção uniforme, fornecer informações de projetos e incentivar boas práticas de desenvolvimento. Dessa forma, os desenvolvedores podem automatizar tarefas comuns de compilação, teste, empacotamento e distribuição de software que consumiriam muito tempo e configurações.

A utilização dessa ferramenta é fundamentada em determinar dependências, plugins e definir estruturas importantes no Project Object Model (`POM.xml`). Esse arquivo de configuração contém valores padrões para a maioria dos projetos. Ao executar uma tarefa, o Maven procura o POM no diretório atual, realiza a leitura e obtém as informações necessárias para execução (THE APACHE SOFTWARE FOUNDATION, Accessed: 2023[c]).

Uma das principais vantagens do Apache Maven é o grande e crescente repositório de bibliotecas e metadados prontos para uso em projetos Java, o **Repositório Central Maven**.

2.2.1 Repositório Central Maven

O Maven, desde o início, trabalha com o conceito de armazenamento em repositórios, tanto de dependências baixadas quanto de artefatos construídos. Existem dois tipos de repositórios. O **repositório local** é um diretório no computador onde o Maven é executado. Ele armazena em cache os downloads remotos e contém artefatos de compilação temporários que ainda não foram lançados. Os **repositórios remotos** referem-se a qualquer outro tipo de repositório, acessado por protocolos (THE APACHE SOFTWARE FOUNDATION, Accessed: 2023[b]).

O repositório central Maven é um repositório remoto de software, seguro e confiável, gerenciado pelo Apache Maven. Ao especificar as dependências do projeto no arquivo de configuração `pom.xml` (Project Object Model), o Maven consulta o repositório central para baixar as versões específicas das bibliotecas para a construção correta do projeto.

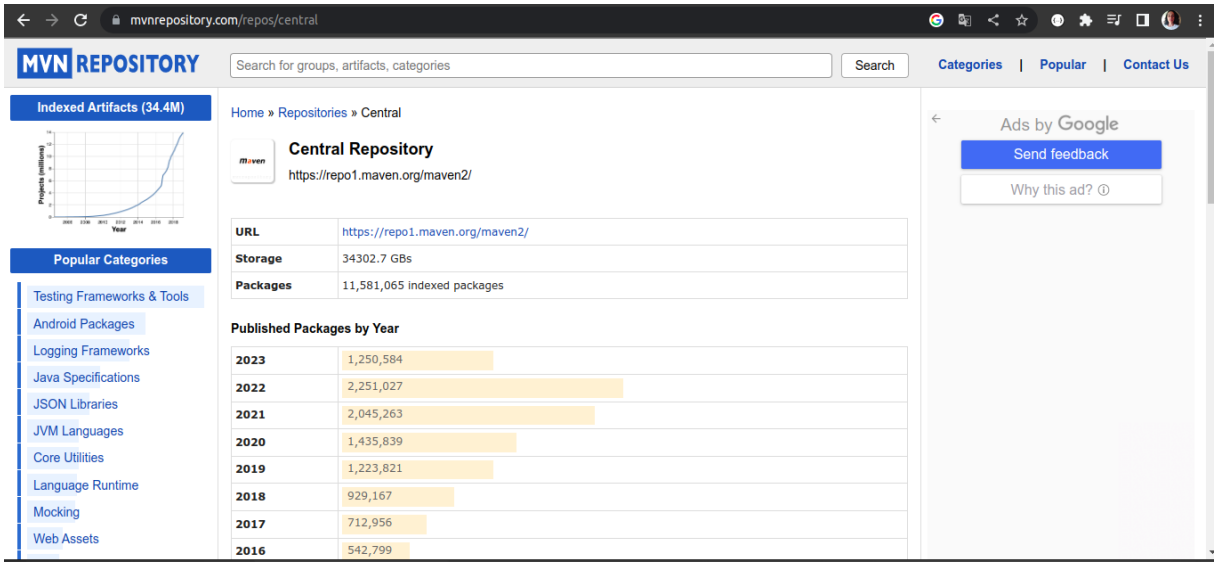

Figura 2 – Interface do Repositório Central Maven (MAVEN..., Accessed: 2023)

As buscas podem ser realizadas através de requisições à API do repositório. Exemplo de uma busca ao Repositório Central Maven:

```
https://search.maven.org/solrsearch/select?q=a:junit&wt=json
```

O request acima retorna a quantidade de resultados encontrados (1), a quantidade de versões disponíveis (2), a versão mais atual (3), a data do último release (4) e etc. Esses elementos podem ser úteis para avaliar a popularidade e o uso ativo de uma biblioteca, o que pode ser um indicador da confiabilidade pela comunidade.

Figura 3 – Resposta da requisição à API do Maven Central Repository

2.3 GitHub

GitHub é uma plataforma de hospedagem de código para controle de versão e colaboração. Permite que múltiplas pessoas trabalhem em conjunto em projetos de qualquer lugar (GITHUB, Accessed: 2023[b]). O funcionamento é baseado na hospedagem de repositórios do Git e fornecimento de ferramenta para envio de código aprimorado por meios de linha de comando, discussões encadeadas, pull requests e revisão de código (GITHUB, Accessed: 2023[a]). Essas funcionalidades também podem ser utilizadas através de solicitações HTTP à API pública da plataforma, que é baseada no protocolo REST e retorna as respostas em formado JSON. Esse suporte torna possível a criação de scripts, aplicativos ou integrações personalizadas para automatizar tarefas, extrair informações de repositórios, realizar análises, integrar sistemas, entre outros.

Exemplo de uma busca pelos repositórios públicos do GitHub utilizando a API através de uma requisição GET:

```
https://api.github.com/search/repositories?q=junit&language=java
```

Nessa busca, procura-se coletar informações dos repositórios que possuem projetos Java e que utilizam da biblioteca Junit.

A resposta contém alguns elementos interessantes como total de repositórios encontrados (1), estrelas (2), usuários que acompanham as atividades do repositório (3), quantidade de cópias dos repositórios (4) e etc.


```
{
    "total_count": 52848,(1)
    "incomplete_results": false,
    "items": [
        {
            "id": 106310,
            "node_id": "MDEwOlJlcG9zaXRvcnkxMDYzMTA=",
            "name": "junit4",
            "full_name": "junit-team/junit4",
            "private": false,
            "owner": {
                "login": "junit-team",
                "id": 874086,
                "node_id": "MDEyOk9yZ2FuaXphdGlvbjg3NDA4Ng==",
```


Figura 4 – Total de repositórios encontrados que utilizam Junit

Figura 5 – Estrelas, whatchers e forks de um dos repositórios que utilizam da biblioteca Junit

Além disso, o GitHub é amplamente reconhecido como uma plataforma essencial para a colaboração e a descoberta de projetos de código aberto. Sua comunidade diversificada e engajada de desenvolvedores contribui para um ecossistema rico e dinâmico. Com inúmeros projetos e bibliotecas hospedados no GitHub, essa plataforma se tornou uma fonte valiosa de informações e indicadores que podem auxiliar na avaliação da qualidade de uma biblioteca de software. Através de recursos como estrelas, forks, issues abertas, solicitações de pull e discussões ativas, é possível obter insights sobre o nível de adoção, envolvimento da comunidade, manutenção contínua e qualidade percebida de uma biblioteca.

3 Trabalhos Relacionados

Nesta seção, são apresentados os trabalhos relacionados que serviram como base e referência para o desenvolvimento deste trabalho de conclusão de curso.

3.1 Which library should I use? A metric-based comparison of software libraries

O artigo "Which library should I use? A metric-based comparison of software libraries", publicado por Fernando López de la Mora e Sarah Nadi (MORA; NADI, 2018) aborda a dificuldade que desenvolvedores podem enfrentar para escolher a biblioteca apropriada a usar, já que muitas vezes vantagens e desvantagens de cada biblioteca são desconhecidas e as diferentes características em diferentes situações podem preocupar.

Como solução, é proposto métricas de software para ajudar os desenvolvedores a escolher as bibliotecas mais adequadas às suas necessidades com base em várias métricas extraídas de várias fontes, como repositórios de software, sistemas de rastreamento de problemas e sites de perguntas e respostas.

São apresentadas os processos de extração e as ferramentas utilizadas para a obtenção da:

- Popularidade;
- Frequência de lançamentos;
- Tempos de resposta e fechamento de problemas;
- Última atualização;
- Compatibilidade com versões anteriores;
- Facilidade de migração;
- Propensão à falhas;
- Performance e segurança.

3.2 What are the characteristics of popular APIs? A large-scale study on Java, Android, and 165 libraries

O artigo "What are the characteristics of popular APIs? A large-scale study on Java, Android, and 165 libraries", publicado por Caroline Lima e Andre Hora (LIMA; HORA, 2019), discute as características que diferenciam uma APIS populares de APIs com menores frequências de uso pela comunidade de desenvolvedores.

Para responder ao questionamento levantado, os autores propõem avaliar tamanho, legibilidade, documentação, estabilidade e adoção pelos clientes. Foram analisadas 1491 APIs Java e Android e 165 bibliotecas.

Detectou-se que, algumas APIs grandes e de grande popularidade, em algumas ocasiões, continham mais comentários e instabilidade do que APIs comuns. Isso provou que, métricas isoladas não são bons parâmetros para analisar a confiabilidade de uma biblioteca. O ecossistema desempenha um papel importante. É necessário aplicar um conjunto de indicadores para gerar uma análise mais contextualizada do cenário em que se está inserido.

4 Metodologia

A metodologia deste trabalho consiste de 6 etapas, apresentadas na Figura 6:

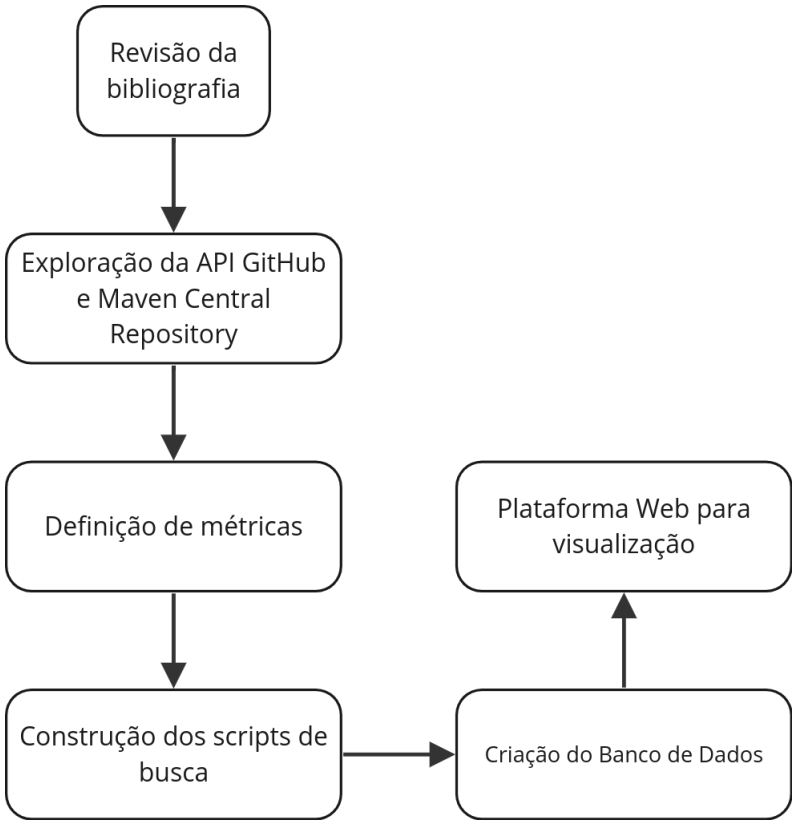

Figura 6 – Fluxograma da metodologia do trabalho

As estapas consistem em:

1. Revisão da bibliografia dos trabalhos relacionados às principais métricas utilizadas para escolhas de bibliotecas de software e a importância e impacto delas na comunidade.
2. Exploração da base de dados disponibilizada no GitHub e repositório Maven.
3. Definição de um filtro e categorização para os dados relevantes ao trabalho.
4. Elaboração do projeto inicial com os requisitos e abordagem para o novo conjunto de métricas.
5. Construção de scripts para tratamento e indexação dos dados.
6. Desenvolvimento de uma plataforma web para comparação de bibliotecas.

5 Cronograma

A seguir, na Tabela 1 é apresentado o cronograma com as atividades realizadas até o momento, de acordo com a legenda disponível na Tabela 2.

Atividades	Meses (Jan/2023 - Ago/2023)							
	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago
Revisão da bibliografia								
Explorar API GitHub								
Explorar API Maven								
Entrega TCC1								
Definição métricas								
Scripts de busca								
Banco de dados								
Aplicação web								
Entrega TCC2								

Tabela 1 – Cronograma de atividades

Cores	Legenda
	Atividade finalizada
	Atividade em andamento
	Atividade não iniciada

Tabela 2 – Legenda de cores da Tabela 1

5.1 Explorar API GitHub

O objetivo dessa atividade foi ler a documentação para consumir a API do GitHub. Foram exploradas formas de obter informações relevantes para a pesquisa e realizados testes práticos.

5.2 Explorar API Maven

O objetivo dessa atividade foi ler a documentação para consumir a API do Maven Central Repository. Foram exploradas formas de obter informações relevantes para a pesquisa e realizados testes práticos.

5.3 Definição das métricas e scrip de busca

Essas etapas estão sendo realizada em conjunto. A definição das métricas estão sendo elaboradas conforme os scripts vão sendo executados com sucesso. Alguns indicadores, teoricamente definidas nas fontes bibliográficas, ao tentar serem buscadas na prática apresentam impedimentos como tempo para execução do script ou até mesmo falta de informação de como adquirir a métrica através das APIs disponíveis.

6 Conclusões Preliminares

Com esse trabalho, na esfera da disciplina de TCC 1, foi possível estudar na literatura os trabalhos relacionados e assim, definir as contrariedades e disposições deste trabalho. Portanto, foi possível construir uma bagagem suficiente de informação para o desenvolvimento de um produto final para a disciplina de TCC 2.

Referências

GITHUB. *GitHub - Sobre o Git*. Accessed: 2023. https://docs.github.com/pt/get-started/using-git/about-git. Citado 1 vez na página 12.

GITHUB. *GitHub Hello World Guide*. Accessed: 2023. https://docs.github.com/pt/get-started/quickstart/hello-world. Citado 1 vez na página 12.

LARIOS VARGAS, Enrique et al. Selecting third-party libraries: The Practitioners' Perspective. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020. DOI: 10.1145/3368089.3409711. Citado 2 vezes na página 7.

LIBRARY home. Disponível em: https://open.library.ubc.ca/docs. Citado 1 vez na página 7.

LIMA, Caroline; HORA, Andre. What are the characteristics of popular apis? A large-scale study on Java, Android, and 165 libraries. *Software Quality Journal*, v. 28, n. 2, p. 425–458, 2019. DOI: 10.1007/s11219-019-09476-z. Citado 1 vez na página 16.

MAVEN Central Repository. Accessed: 2023. https://mvnrepository.com/repos/central. Citado 0 vez na página 11.

MORA, Fernando López de la; NADI, Sarah. Which library should I use? *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, 2018. DOI: 10.1145/3183399.3183418. Citado 2 vezes nas páginas 7, 15.

ORACLE. *Java Tutorials: Getting Started*. Oracle Corporation. 2023. Disponível em: https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html. Citado 1 vez na página 9.

ORACLE. *O que é o Java?* Oracle Corporation. 2023. Disponível em: https://www.java.com/pt-BR/download/help/whatis_java.html. Citado 1 vez na página 9.

SOFTWARE, TIOBE. *TIOBE Index*. 2023. Disponível em: https://www.tiobe.com/tiobe-index/. Citado 1 vez na página 9.

THE APACHE SOFTWARE FOUNDATION. *Apache Maven Documentation*. Accessed: 2023. https://maven.apache.org/. Citado 1 vez na página 10.

THE APACHE SOFTWARE FOUNDATION. *Introduction to Repositories*. Accessed: 2023. https://maven.apache.org/guides/introduction/introduction-to-repositories.html. Citado 1 vez na página 10.

THE APACHE SOFTWARE FOUNDATION. *Introduction to the POM*. Accessed: 2023. https://maven.apache.org/guides/introduction/introduction-to-the-pom.html. Citado 1 vez na página 10.