**UNIVERSIDADE FEDERAL DO CEARÁ**

**CAMPUS DE QUIXADÁ**

**CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**GUILHERME SEIXAS DE OLIVEIRA SANTOS**

# ANÁLISE DE *FRAMEWORKS BACK-END* ESCRITOS EM JAVA: UMA PERSPECTIVA EM CRITÉRIOS

**QUIXADÁ**

**2023**

GUILHERME SEIXAS DE OLIVEIRA SANTOS

# ANÁLISE DE *FRAMEWORKS BACK-END* ESCRITOS EM JAVA: UMA PERSPECTIVA EM CRITÉRIOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Livia Almada Cruz.

QUIXADÁ

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

S235a Santos, Guilherme Seixas de Oliveira.
Análise de frameworks back-end escritos em Java : uma perspectiva em critérios / Guilherme Seixas de Oliveira Santos. – 2023.
73 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2023.
Orientação: Profa. Dra. Livia Almada Cruz.

1. Java (Linguagem de programação de computador). 2. Framework (Arquivo de computador). 3. Back-end. 4. Análise comparativa. I. Título.

CDD 004

---

GUILHERME SEIXAS DE OLIVEIRA SANTOS

# ANÁLISE DE *FRAMEWORKS BACK-END* ESCRITOS EM JAVA: UMA PERSPECTIVA EM CRITÉRIOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: ___/___/___.

BANCA EXAMINADORA

---

Profa. Dra. Livia Almada Cruz (Orientadora)
Universidade Federal do Ceará (UFC)

---

Prof. Dra. Antonia Diana Braga Nogueira
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Jefferson de Carvalho Silva
Universidade Federal do Ceará (UFC)

A Deus pela dádiva da fortaleza. A minha querida família e amigos pelo apoio incondicional. A todos que, de alguma maneira, colherão frutos deste trabalho.

## AGRADECIMENTOS

Inicialmente, expresso minha profunda gratidão a Deus pela força concedida ao longo da minha jornada acadêmica e, especialmente, nesta fase crucial da graduação. Sua luz e amor foram fundamentais para me manter firme e perseverante.

À minha família, dedico meu sincero agradecimento pelo amor incondicional e apoio inabalável ao longo destes longos anos. À minha mãe, Valdenia, sou imensamente grato pelo incentivo nos meus estudos desde a infância, orientando-me a ser sempre melhor, focado e determinado. Você não apenas me ensinou a trilhar esse caminho, mas também intercede por mim diariamente em suas orações.

Ao meu pai, Jozilei, agradeço por toda a força e disponibilidade constante ao longo da minha vida e especialmente durante esse período desafiador.

Às minhas irmãs, Franciele e Camilla, expresso minha gratidão pelo carinho contínuo e por acreditarem em mim.

A Quixadá, expresso minha sincera gratidão pela recepção que encontrei, pelo acolhimento, pelas valiosas lições que aprendi nesta cidade encantadora e pelo privilégio de estudar no campus mais lindo que tive a oportunidade de conhecer. Além disso, agradeço às pessoas que conheci aqui e que transformaram profundamente minha vida.

À minha orientadora, Dra. Livia Almada Cruz, quero expressar minha gratidão por todo apoio e orientação. Mesmo não sendo da mesma área, você desempenhou um papel fundamental, contribuindo significativamente para o sucesso deste trabalho por meio de suas orientações e reuniões dedicadas. Este trabalho só se tornou possível graças à sua orientação precisa e apoio constante.

Agradeço também aos membros da minha banca, Prof. Dra. Antonia Diana Braga Nogueira e Prof. Dr. Jefferson de Carvalho Silva, pela avaliação cuidadosa e contribuições valiosas.

Expresso minha profunda gratidão à Universidade Federal do Ceará, em especial ao Campus Quixadá, estendendo meu agradecimento a todos os servidores, professores e demais indivíduos que compartilharam este percurso ao longo da minha jornada acadêmica.

"... I said remember this feeling, I passed the pictures around, of all the years that we stood there on the sidelines wishing for right now" (Taylor Swift, 2010.)

## RESUMO

Os *frameworks*, que surgem como abstrações essenciais para a redução de esforço no desenvolvimento de software, desempenham papel crucial na implementação de regras de negócios, mecanismos de segurança e configuração de gerenciamento de dados nas aplicações do lado do servidor, ou mais conhecidas como aplicações *back-end*. O presente trabalho realiza a análise de três *frameworks* em Java: Spring Boot, Vert.x e Dropwizard, considerando critérios específicos definidos para avaliação. A escolha desses *frameworks* baseou-se em dados extraídos da plataforma GitHub, priorizando o número de estrelas em seus repositórios oficiais. Os critérios de avaliação foram divididos em duas categorias: os de contexto, que abrangem características e ambiente em torno do *framework*, como popularidade, atividade da comunidade, qualidade da documentação, artigos publicados e suporte à inicialização de projetos, enquanto os de aplicação, que fundamentam o desenvolvimento prático, incluem mapeamento objeto-relacional de dados e suporte à arquitetura de software *Representational State Transfer* (REST). A análise revela nuances e particularidades significativas dos *frameworks* em questão, oferecendo diferentes visões para o desenvolvimento de software *web back-end* em Java.



**Palavras-chave**: análise comparativa; framework; java (linguagem de programação).

**ABSTRACT**

Frameworks, which emerge as essential abstractions to reduce effort in software development, play a crucial role in implementing business rules, security mechanisms, and data management configuration in server-side applications, commonly referred to as *back-end* applications. This essay conducts an analysis of three Java frameworks: Spring Boot, Vert.x, and Dropwizard, considering specific criteria defined for evaluation. The selection of these frameworks was based on data extracted from the GitHub platform, prioritizing the number of stars in their official repositories. Evaluation criteria were divided into two categories: contextual criteria, covering characteristics and the environment around the framework, such as popularity, community activity, documentation quality, published articles, and support for project initialization; and application criteria, grounding practical development, including object-relational data mapping and support for REST software architecture. The analysis reveals significant nuances and peculiarities of the frameworks in question, providing different perspectives for *web back-end* software development in Java.
Keywords: comparative analysis; framework; java (programming language).

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| APIs | Interfaces de Programação de Aplicação |
| CRUD | *Create, Read, Update and Delete* |
| HTML | *HyperText Markup Language* |
| HTTP | *Hypertext Transfer Protocol* |
| IOC | Inversão de Controle |
| IOT | Internet das Coisas |
| JVM | *Java Virtual Machine* |
| REST | *Representational State Transfer* |
| URL | *Uniform Resource Locator* |
| WWW | *World Wide Web* |

**SUMÁRIO**

# 1 INTRODUÇÃO

Com o rápido desenvolvimento da tecnologia da informação, ocorreu a digitalização abrangente de quase todos os aspectos da sociedade moderna. Nesse contexto, o uso do software se tornou uma atividade comum e essencial para todos (WICAKSONO *et al.*, 2016).

Em vista desse contexto, torna-se evidente que o uso de software desempenha um papel importante para o progresso na atualidade. Conforme destacado por Sommerville (2011), "O mundo moderno não poderia existir sem o software. Infraestruturas e serviços nacionais são controlados por sistemas computacionais, e a maioria dos produtos elétricos inclui um computador e um software que o controla". O autor ressalta que a engenharia de software, além de projetar, desenvolver e manter projetos complexos, também exerce influência significativa em diversos setores.

O surgimento da *World Wide Web* (WWW) revolucionou o cenário tecnológico e impulsionou o desenvolvimento de software baseado na *web*. Conforme destacado por Berners-Lee (1990), o criador da WWW, o objetivo principal era criar um sistema que permitisse o compartilhamento de documentos e recursos entre entidades. Dessa forma, o desenvolvimento de software baseado na *web* permitiu a criação de aplicações distribuídas, onde a lógica e o processamento ocorrem tanto no lado do cliente quanto no lado do servidor.

Com isso, as exigências de mercado supracitadas são percebidas também na área do desenvolvimento de softwares para a *web*, especialmente no que se refere ao desenvolvimento de *back-end* (OLIVEIRA, 2019). Isso se deve à sua importância ao introduzir as regras de negócio, segurança de dados, gerenciamento de banco de dados e outros aspectos essenciais para o desenvolvimento de aplicações do lado do servidor.

A evolução das aplicações levou ao surgimento dos *frameworks* de software, que desempenham um papel fundamental na melhoria da qualidade e na redução do esforço de desenvolvimento (ALMEIDA, 2018). Segundo Ghimire (2020), os *frameworks* de software são compostos por um conjunto de bibliotecas e ferramentas padronizadas para uma linguagem de programação específica. Seu principal objetivo é proporcionar uma abordagem concisa e eficiente na construção de aplicações de software.

A utilização de *frameworks* de software no desenvolvimento de aplicações *web*, especialmente no contexto de *back-end*, torna evidente a necessidade de uma análise embasada em critérios entre os diferentes *frameworks* disponíveis no mercado. Tendo em vista que esses *frameworks* oferecem benefícios significativos, como a reutilização de código e a eficiência,

permitindo que os programadores iniciem rapidamente o desenvolvimento de uma aplicação e se concentrem na implementação da lógica de negócios (GHIMIRE, 2020).

Este trabalho analisa *frameworks web back-end* mais populares em Java, considerando critérios mensuráveis e suas operacionalidades distintas. O objetivo principal é orientar desenvolvedores na escolha dessas tecnologias, elucidando características e padrões para o desenvolvimento de projetos *back-end web*. Além de fornecer uma orientação teórica e prática, a pesquisa busca preencher uma lacuna acadêmica ao concentrar-se na análise crítica desses *frameworks* no contexto *back-end*, contribuindo para uma melhor compreensão dessas ferramentas fundamentais no cenário de desenvolvimento.

A estrutura deste trabalho é organizada da seguinte forma: no Capítulo 2, são apresentados os fundamentos teóricos que embasam os conceitos-chave importantes para este estudo. O Capítulo 3 aborda os trabalhos relacionados, fornecendo um panorama do contexto em que este estudo se insere. No Capítulo 4, são detalhados os procedimentos metodológicos adotados. No Capítulo 5, são apresentados os resultados obtidos a partir das análises realizadas. Por fim, no Capítulo 6 é apresentada a conclusão do trabalho. Essa estrutura visa proporcionar uma visão abrangente e organizada do trabalho, permitindo uma compreensão clara e sequencial do desenvolvimento do estudo.

## 1.1 Objetivos

Nesta seção, são expostos o objetivo geral e os objetivos específicos do trabalho atual.

### *1.1.1 Objetivo geral*

O objetivo geral deste trabalho é apresentar uma análise entre diferentes *frameworks* de software *back-end* escritos em Java baseada em critérios definidos.

### *1.1.2 Objetivos específicos*

1. Identificar *frameworks* de software *back-end* escritos em Java que tenham maior popularidade.

2. Analisar em cada *framework* identificado uma série de critérios identificados como: popularidade, atividade da comunidade, qualidade da documentação, avaliação de artigos

publicados, suporte à inicialização de projeto, suporte a mapeamento objeto-relacional e suporte à arquitetura de software REST.

3. Diferenciar os *frameworks* de acordo com os critérios definidos.

# 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são abordados os fundamentos essenciais deste trabalho, bem como seu impacto nas atividades realizadas. Explora-se a compreensão dos conceitos de desenvolvimento *web*, arquitetura de aplicações *web* e *framework* de software *web back-end*.

## 2.1 Desenvolvimento *web*

A criação da WWW por Tim Berners-Lee marcou significativamente o mundo da tecnologia e da comunicação. No início da década de 1990, enquanto trabalhava, Berners-Lee desenvolveu um sistema inovador que permitiria o compartilhamento e acesso a informações de forma dinâmica e eficiente mediante uma rede de computadores (BERNERS-LEE, 1990). Esse sistema inovador deu origem à *web*, mais especificamente à *web* 1.0, inaugurando uma era de interconexão global e abrindo as portas para a disseminação de conteúdo online.

A *web* 1.0 foi caracterizada pela predominância de hiperlinks e conteúdo estático, assemelhando-se a páginas impressas (HUBSPOT, 2017). Nessa fase, a ênfase estava centrada na disponibilização de informações aos usuários, com uma comunicação unidirecional. Os internautas tinham um papel passivo, limitando-se ao consumo de conteúdo, enquanto a interação e criação de conteúdo eram restritas. Na *web* 1.0, era raro encontrar vídeos e a participação ativa dos usuários era principalmente direcionada ao envio de e-mails por meio de formulários de cadastro (HUBSPOT, 2017).

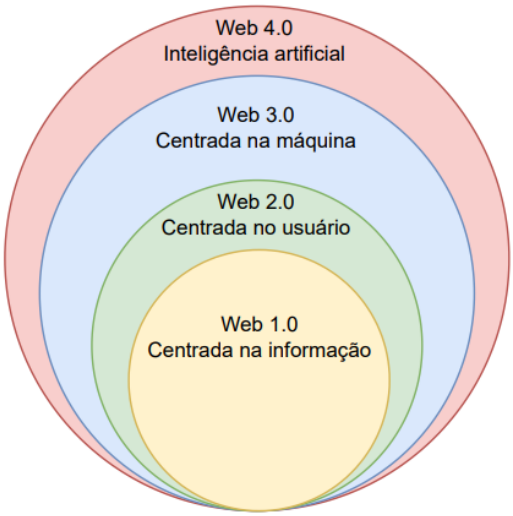
Com a evolução da internet, surgiram novas vertentes da *web*, como a *web* 2.0, na qual as páginas passaram a apresentar conteúdo dinâmico e interativo. Nessa nova plataforma, a premissa fundamental consistia em aprimorar continuamente o software com base na usabilidade dos usuários, aproveitando a inteligência coletiva (O'REILLY, 2005).

Além disso, a transformação digital impulsionou o surgimento da *web* 3.0, que visa integrar dados e informações de forma mais inteligente, proporcionando uma experiência personalizada e semântica aos usuários (HUBSPOT, 2017). Essa evolução da *web* representa um salto significativo na interação com a tecnologia, pois é centrada na máquina, abrindo caminho para um ambiente digital mais inteligente e adaptável.

Por fim, a *web* 4.0 desponta como uma promissora visão de futuro, fundamentada no aproveitamento de tecnologias como inteligência artificial, Internet das Coisas (IOT) e realidade virtual, com o propósito de proporcionar uma experiência ainda mais imersiva e conectada aos

usuários (HUBSPOT, 2017). A Figura 1 ilustra a evolução da *web* destacando suas principais características, como apresentadas nos parágrafos anteriores.

Figura 1 – Demonstração da evolução da *web*.



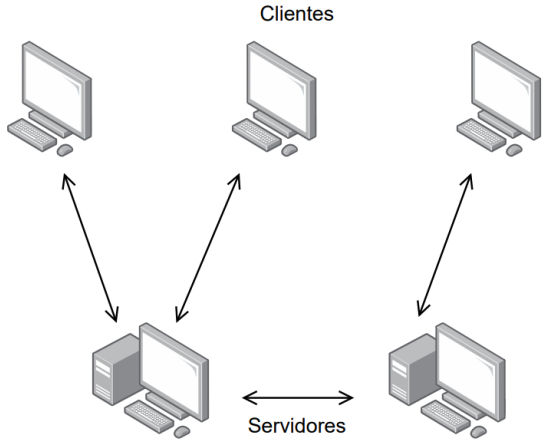

Fonte: Adaptado de (LIMA, 2018).

A contribuição inicial de Berners-Lee não se limitou apenas à criação da *web* em si, mas também abrangeu o desenvolvimento de tecnologias e padrões fundamentais que possibilitaram aspectos evolutivos nessa inovação. Ele desenvolveu protocolos para o gerenciamento de transferência de dados, como o HTTP, além de criar artefatos para estruturação e formatação de conteúdo na *web*, como a linguagem *HyperText Markup Language* (HTML) (BERNERS-LEE, 1990).

Além disso, Berners-Lee propôs o modelo cliente-servidor para sistemas de hipertexto distribuído (BERNERS-LEE, 1990), o qual definiu a arquitetura inicial da *web*. Nesse modelo, os servidores de hipertexto estão interconectados, permitindo a comunicação entre eles, enquanto os clientes realizam requisições aos servidores que após processarem os dados recebidos enviam respostas de volta a eles.

A Figura 2 apresenta o modelo cliente-servidor proposto por Berners-Lee (1990). Nesta representação, podemos observar dois servidores interconectados por meio de linhas que simbolizam o mecanismo de comunicação entre eles. Os clientes, que atuam de maneira independente, possuem linhas de comunicação que representam as requisições enviadas pelos usuários, assim como representam as respostas que foram processadas e enviadas pelos servidores. Essas conexões são bidirecionais, pois os servidores não apenas se comunicam entre si trocando informações e dados, mas também recebem requisições dos usuários que são processadas e enviadas como respostas.

Figura 2 – Demonstração do modelo de cliente e servidor proposto por Berners-Lee.




Fonte: Adptado de (BERNERS-LEE, 1990).

Desde então, a *web* tem passado por constantes avanços impulsionados por diferentes atores no ramo da tecnologia. Esses avanços têm possibilitado o desenvolvimento de novas aplicações e aprimoramentos na plataforma *web*. O desenvolvimento aplicações *web* é julgado de extrema importância, pois permite acompanhar as demandas crescentes dos usuários e oferecer aplicações com máximo de rapidez e qualidade (MOREIRA, 2009).

Dado esse contexto, é notório o impacto da evolução contínua da *web* na vida das pessoas, impulsionada pela sua criação por Berners-Lee e pelas contribuições constantes de diversos protagonistas na área tecnológica.

## 2.2 Arquitetura de aplicações *web*

Há alguns anos, as páginas presentes na plataforma *web* eram estáticas e usadas apenas para exibição de conteúdo aos usuários, funcionando apenas a partir de cliques em links para carregar novas páginas (MCFARLAND, 2014). Ao longo do tempo, novas linguagens surgiram para atender a diversas outras demandas da *web*. Por exemplo, o CSS é utilizado para estilização, enquanto o JavaScript é responsável pela interatividade, que conforme mencionado por (EIS, 2015), são considerados os fundamentos do *front-end* moderno.

Uma aplicação *web* é composta por diferentes camadas, cada uma desempenhando um papel específico no funcionamento do sistema. As principais camadas relevantes para este trabalho são o *back-end* e o *front-end*. O *front-end* é a camada inicial que os usuários encontram ao acessar um site, uma intranet ou um sistema *web* (CITRUS7, 2017).

O *front-end* é a camada da aplicação *web* por meio da qual os usuários interagem, inserindo dados e entradas que serão processados. É no *front-end* que as páginas *web* ganham

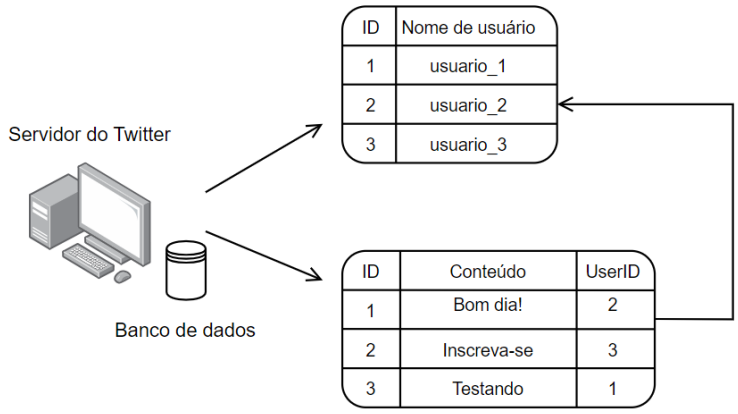
vida, proporcionando uma experiência do usuário mais aprimorada por meio de uma interface bem desenvolvida (ALMEIDA, 2018). Nessa camada, os usuários podem interagir com as aplicações *web*, inserindo dados e informações que serão processados pelo *back-end*, a arquitetura de aplicação *web* do lado do servidor.

O *back-end* é o termo utilizado para denominar a camada *web* responsável pelo gerenciamento de negócios. É papel dessa camada aplicar as regras de negócio e lidar com persistência dos dados das aplicações através do gerenciamento de banco de dados (SOMMERVILLE, 2011). A estrutura do *back-end* desempenha um papel fundamental na arquitetura de um sistema, garantindo o correto funcionamento do lado do servidor e armazenamento dos dados.

Para ilustrar essa estrutura, a Figura 3 apresenta uma representação simplificada do *back-end* da rede social Twitter. O *back-end*, também conhecido como servidor, desempenha o papel de fornecer respostas aos dispositivos solicitantes ou clientes (FLUFFY, 2019).

O servidor armazena geralmente os dados em um banco de dados. A maioria dos bancos de dados utiliza um modelo relacional, o qual simplifica o armazenamento dos dados em tabelas (FLUFFY, 2019). Essas tabelas possuem características e atributos que se referenciam entre si quando possuem o mesmo valor, como pode ser observado na Figura 3.

Figura 3 – Simples demonstração da arquitetura *back-end* na rede social Twitter.




Fonte: Adaptado de (FLUFFY, 2019).

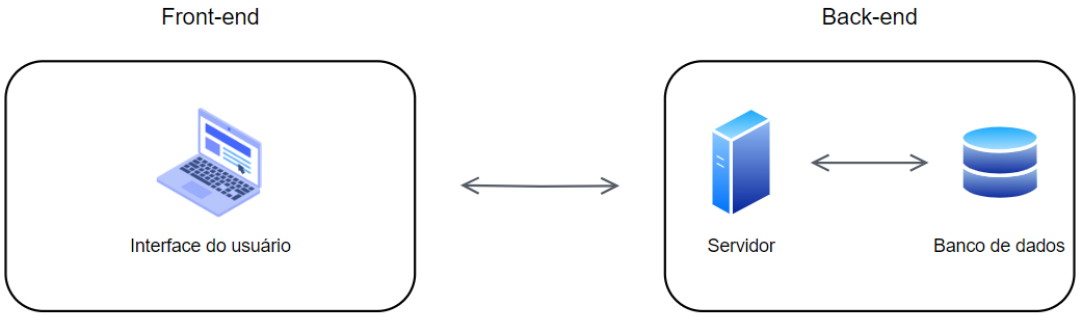
Em uma aplicação *web*, a interação entre o *front-end* e o *back-end* desempenha um papel essencial. O *front-end* assume a responsabilidade de criar uma interface amigável para os usuários, permitindo a interação e a visualização dos dados. Por sua vez, o *back-end* é responsável pela lógica de negócios e pela manipulação eficiente dos dados (CITRUS7, 2017). Essa colaboração entre as duas camadas é fundamental para o funcionamento e o sucesso da aplicação.

É importante destacar a integração harmoniosa dessas camadas na arquitetura de uma aplicação *web*, permitindo aos usuários interagir com a interface do *front-end* enquanto o *back-end* gerencia os processos e a lógica de negócios por trás (CITRUS7, 2017).

A Figura 4 apresenta uma representação visual que ilustra a interação entre o *front-end* e o *back-end*, proporcionando uma compreensão mais clara dessa dinâmica.

O *front-end* consiste na interface pela qual o usuário interage e envia dados para o *back-end*. O *back-end*, por sua vez, é composto pelo servidor e pelo banco de dados, que estão interconectados. O *back-end* processa os dados recebidos e retorna uma resposta ao *front-end*, completando assim o ciclo de interação entre o usuário, através da interface do *front-end* e o sistema, através do *back-end*.

Figura 4 – Demonstração das arquiteturas *front-end* e *back-end* para aplicações *web*.




Fonte: Elaborado pelo autor.

## 2.3 *Framework* de software *web back-end*

Conforme (BRANAS, 2014), um *framework* é uma aplicação pré-escrita que disponibiliza um conjunto de bibliotecas e componentes, proporcionando maior produtividade, flexibilidade, manutenibilidade e testabilidade no desenvolvimento de aplicações *web*. Essas características fazem com que os *frameworks* sejam amplamente utilizados no desenvolvimento de aplicações *back-end*, desempenhando um papel essencial nesse contexto.

Conforme a definição de (MATTSON, 1996), um *framework* é descrito como uma arquitetura generativa projetada para alcançar reutilização máxima. Essa arquitetura é representada por um conjunto de classes abstratas e concretas, com comportamento encapsulado que pode ser especializado por subclasses.

Schmidt *et al.* (2013) definem um *framework* como um software projetado para ser instanciado, parcialmente completo. O *framework* estabelece uma arquitetura para uma

família de subsistemas e fornece os construtores básicos para que eles possam ser criados ou instanciados.

Consequentemente, os *frameworks* de software são importantes justamente por permitem que funcionalidades já implementadas possam ser reutilizadas pelos desenvolvedores na construção das aplicações, garantindo eficiência e redução de esforço e gastos (ALMEIDA, 2018).

Ao utilizar um *framework*, como o Spring Boot, os desenvolvedores se beneficiam da vasta gama de funcionalidades já implementadas, como o tratamento de solicitações HTTP, o uso de anotações personalizadas, o mapeamento de objetos para bancos de dados e a autenticação de usuários. Isso não apenas acelera o desenvolvimento, mas também aumenta a confiabilidade e a segurança das aplicações *back-end*.

A Figura 5 demonstra uma abstração que o *framework web back-end* Spring Boot fornece para mapear solicitações da *web*. Para a classe demonstrada abaixo, o uso da anotação *Controller*, indica que a classe serve ao papel de um controlador, ou seja, é responsável por lidar com solicitações HTTP e pode ser acessada pela rota exibida na anotação *RequestMapping*.

Figura 5 – Exemplo do uso de anotações no *framework* Spring Boot, um recurso de abstração útil no desenvolvimento de software.

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/api/users")
public class UsuarioController { }
```

Fonte: Elaborado pelo autor.

No desenvolvimento de aplicações de software, os *frameworks* desempenham um papel imprescindível, especialmente no contexto das aplicações *back-end*. Eles oferecem abstrações e eficiência que permitem aos desenvolvedores agilizar o processo de construção e implementação dessas aplicações (GHIMIRE, 2020). Além disso, os *frameworks* fornecem uma estrutura sólida e padronizada, que facilita a organização e a manutenção do código-fonte.

# 3 TRABALHOS RELACIONADOS

Nesta seção são apresentados alguns trabalhos relacionados com o presente trabalho.

## 3.1 Uma Comparação entre Micro *Frameworks Web* para o Desenvolvimento de Aplicações *Back-end* em Java

Em (OLIVEIRA, 2019) é apresentado um comparativo entre micro *frameworks* para desenvolvimento de aplicações *web back-end* em Java, mais populares em número de estrelas nos seus repositórios na plataforma do Github, sendo eles: Spark, Jooby e Pippo.

Ao contrário dos *frameworks* convencionais, os micro *frameworks* surgem como uma alternativa para aqueles que buscam a estrutura e agilidade de desenvolvimento proporcionadas por um *framework*, porém com menos componentes e menor sobrecarga de um *framework* abrangente (SILVA, 2016).

Para realizar a análise comparativa dos micro *frameworks*, foram utilizados os seguintes critérios de comparação: documentação, curva de aprendizado, popularidade, suporte a Inversão de Controle (IOC), suporte a arquitetura REST, validação da entrada, artigos publicados e tamanho da comunidade.

A fim de comparar os micro *frameworks* com os critérios de comparação selecionados, foi implementada a aplicação do PetClinic, uma aplicação disponível gratuita desenvolvida no *framework* de software Spring Boot. De modo que, para cada *framework* foi implementada a aplicação com algumas modificações para melhor encaixe do contexto do trabalho para avaliar cada critério selecionado. [1]

A partir dos resultados obtidos da comparação dentre os micro *frameworks web back-end* selecionados foi constatado que o Spark Framework foi o melhor para a implementação da aplicação de teste selecionada, o PetClinic.

O trabalho de (OLIVEIRA, 2019) se assemelha ao presente trabalho por ser uma comparação entre artefatos de desenvolvimento *web back-end* em Java utilizando critérios, contudo diferentemente do trabalho de (OLIVEIRA, 2019) que compara micro *frameworks*, neste trabalho é feita a comparação entre *frameworks*.

[1] O PetClinic é uma aplicação de uma clínica veterinária, em que os seus usuários são os próprios funcionários da clínica que, no decorrer de seu trabalho, precisam visualizar e gerenciar informações sobre os veterinários, os clientes e seus animais de estimação (OLIVEIRA, 2019).

## 3.2 *Comparative study on Python web frameworks: Flask and Django*

No trabalho de (GHIMIRE, 2020), é feito um estudo sobre as características, vantagens e limitações de dois *frameworks* de desenvolvimento *web* para a linguagem de programação Python: Flask e Django.

Para desenvolver o estudo comparativo, foi construída uma aplicação do tipo rede social e eCommerce para Flask e Django, respectivamente. A comparação foi iniciada pelo desenvolvimento do aplicativo de rede primeiro com Flask e finalizado com o aplicativo de e-commerce usando Django.

Os critérios de comparação utilizados para comparar os dois *frameworks* foram padrões de design, requisições e roteamento, infraestrutura e configurações, tratamento de erros, cache, flexibilidade e suporte ao desenvolvimento.

Após a comparação de cada critério para ambos *frameworks*, verificou-se que as vantagens mais significativas do Flask eram que ele fornece simplicidade, flexibilidade, controle refinado e mais rápido e fácil de aprender. Por outro lado, o Django era mais fácil de trabalhar devido a seu suporte para bibliotecas.

O trabalho de (GHIMIRE, 2020) se assemelha com o atual trabalho por apresentar uma análise comparativa criteriosa de *frameworks web*. Todavia, o trabalho atual apresenta divergências por ser uma comparação entre *frameworks web back-end* escritos em Java.

## 3.3 Análise Comparativa entre *Frameworks Frontend* Baseados em Javascript para Aplicações *Web*

No trabalho de (FERREIRA; ZUCHI, 2021) é apresentado um comparativo entre *frameworks* de software baseados em Javascript, sendo eles: Angular, Vue e React. Para selecionar os *frameworks*, foi utilizado o site Hot*Frameworks*, que cria rankings de *frameworks* de diversas linguagens baseando-se na popularidade dos projetos desenvolvidos.

Para a realizar a análise comparativa, foi desenvolvida uma aplicação pequena para cada *framework* em que foram analisados critérios como: arquitetura, documentação e comunidade, compatibilidade com navegadores, tamanho do pacote de arquivos e tempo de renderização da página.

Como conclusão, por se tratar de *frameworks* de mesma linguagem base, foi possível observar grandes similaridades no código de cada um e, também, suas particularidades, o que é

acarreta vantagens e desvantagens em índices pontuais.

O trabalho de Ferreira e Zuchi (2021), se assemelha ao presente trabalho por também ser uma comparação entre *frameworks web* baseada em critérios definidos pelos autores. Contudo, o trabalho atual se difere por ser uma análise comparativa entre *frameworks web* de arquitetura *back-end* escritos em Java.

## 3.4 Comparação entre os trabalhos

Esse trabalho propõe comparar *frameworks* de software *web back-end* escritos na linguagem de programação Java. No Quadro 1 a seguir, foram listados todos os trabalhos mencionados nesta seção, juntamente com as diferenças e semelhanças em relação aos parâmetros de: tipo de *framework*, linguagem, critérios e aplicação.

O primeiro critério avalia se os *frameworks* estudados são de arquitetura *back-end*, *front-end* ou ambos. O segundo critério analisa a linguagem de programação predominante em que o *framework* está desenvolvido. O terceiro critério avalia se os critérios utilizados para comparar os *frameworks* do trabalho são de contexto (não requerem execução ou aplicação de código) ou de aplicação (requerem execução ou aplicação de código como *benchmarks* ou CRUD). Por fim, o último critério analisa qual o tipo de aplicação desenvolvida para realizar a análise dos critérios comparativos presentes ou mensuráveis nos *frameworks*. [2] [3]

Quadro 1 – Comparação entre os trabalhos.

| **Trabalho** | **Tipo de *framework*** | **Linguagem** | **Critérios** | **Aplicação** |
|---|---|---|---|---|
| (OLIVEIRA, 2019) | *Back-end* | Java | Contexto e aplicação | Aplicação Pet-Clinic |
| (GHIMIRE, 2020) | *Back-end* e *front-end* | Python | Contexto e aplicação | Rede social |
| (FERREIRA; ZUCHI, 2021) | *Front-end* | JavaScript | Contexto e aplicação | CRUD |
| Presente trabalho | *Back-end* | Java | Contexto e aplicação | CRUD |

Fonte: Elaborado pelo autor.

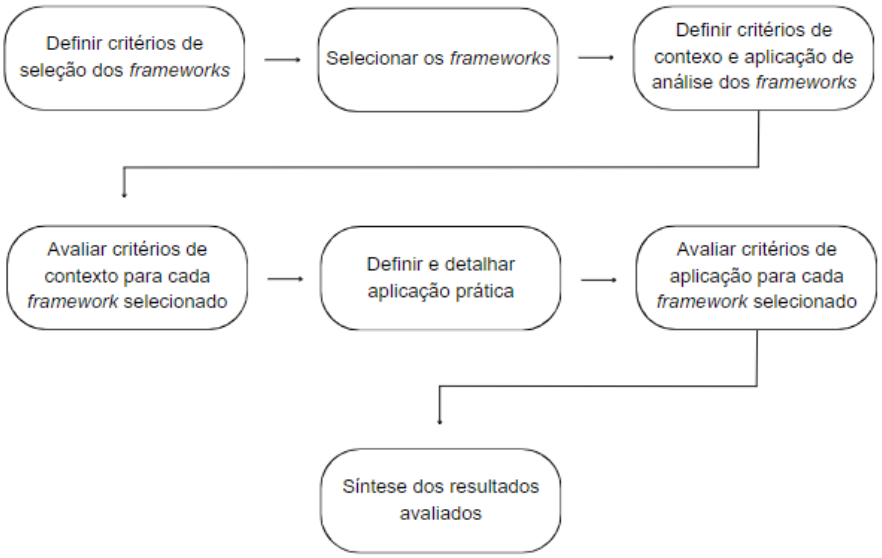
[2] *Benchmark* é a prática de medir e avaliar o desempenho computacional, protocolos de rede, dispositivos e outros componentes. Seu propósito principal é possibilitar uma comparação justa entre diferentes soluções ou entre as evoluções subsequentes de um sistema de teste (BOUCKAERT *et al.*, 2010).

[3] CRUD é um acrônimo que abrange quatro operações fundamentais de gerenciamento de dados em sistemas: *Create* (Criar), *Read* (Ler), *Update* (Atualizar) e *Delete* (Excluir), que correspondem à criação de novos recursos, recuperação de informações, atualização de dados existentes e exclusão de registros, respectivamente. É amplamente empregado para avaliar o tempo de execução dessas operações em sistemas.

# 4 METODOLOGIA

Neste capítulo, são descritos os procedimentos metodológicos adotados para a realização deste trabalho sendo eles: definir critérios de seleção dos *frameworks*, selecionar os *frameworks*, definir critérios de contexto e aplicação para avaliação dos *frameworks*, avaliar critérios de contexto para cada *framework* selecionado, definir e detalhar aplicação prática, avaliar critérios de aplicação para cada *framework* selecionado e síntese dos resultados avaliados. As etapas são ilustradas na Figura 6 abaixo e detalhadas nas seções subsequentes.

Figura 6 – Etapas metodológicas deste trabalho.




Fonte: Elaborado pelo autor.

## 4.1 Definir critérios de seleção dos *frameworks*

A etapa inicial deste estudo envolve a definição dos critérios para selecionar os *frameworks* a serem estudados. Esses critérios foram estabelecidos com base em uma revisão bibliográfica de trabalhos relacionados ao domínio deste estudo, a fim de identificar aspectos relevantes para a comparação dos *frameworks* de aplicação *web back-end*.

Verificou-se que a plataforma GitHub é amplamente reconhecida por sua importância no desenvolvimento de estudos na área de Engenharia de Software (XAVIER *et al.*, 2018). Portanto, optou-se por utilizar essa plataforma como principal fonte de dados para definir os critérios de seleção dos *frameworks*.

## 4.2 Selecionar os *frameworks*

Nesta etapa, é realiza a seleção dos *frameworks* dentro dos limites estabelecidos na etapa anterior.

Para isso, foi definido o uso do website *HotFrameworks* como ponto de partida na definição dos *frameworks*. O *HotFrameworks* cria rankings de *frameworks* em várias linguagens, levando em consideração a popularidade dos projetos desenvolvidos utilizando esses *frameworks* no GitHub, bem como o número de questões relacionadas a eles no Stack Overflow (FERREIRA; ZUCHI, 2021).

## 4.3 Definir critérios de contexto e aplicação de análise dos *frameworks*

Esta etapa define os critérios considerados relevantes para realizar a análise dos *frameworks* selecionados na etapa anterior.

Para alcançar esse objetivo, foi realizada uma revisão bibliográfica em trabalhos relacionados que abordam o mesmo escopo deste estudo. O objetivo principal dessa revisão, foi para identificar critérios significativos, os quais podem ser classificados como de contexto ou de aplicação. Os critérios de contexto são aqueles que podem ser avaliados sem a necessidade de testes ou execução direta. Esses critérios envolvem aspectos como documentação, tamanho da comunidade, disponibilidade de recursos e facilidade de uso. Por outro lado, os critérios de aplicação são aqueles que requerem testes ou execução de código para a avaliação. Esses critérios incluem métricas de desempenho, escalabilidade e eficiência.

Ao considerar esses critérios, busca-se obter uma visão abrangente e embasada na análise dos *frameworks*.

## 4.4 Avaliar critérios de contexto para cada *framework* selecionado

Nesta etapa, o objetivo é realizar uma avaliação dos critérios de contexto selecionados na etapa anterior, os quais não requerem testes práticos ou execução direta, para cada *framework*.

A avaliação dos critérios de contexto é fundamental em um *framework*, pois permite uma análise de aspectos importantes que não são diretamente observáveis por meio da execução de código, como atividade da comunidade em torno do *framework* e qualidade de sua documentação.

Realizando a avaliação desses critérios para cada *framework* selecionado, é possível

obter uma visão mais abrangente das suas próprias características e do seu contexto de uso e atuação na tecnologia da informação.

## 4.5 Definir e detalhar aplicação prática

Esta etapa define e traz informações sobre a aplicação prática desenvolvida neste trabalho, para cada um dos *frameworks* selecionados, a fim de avaliar os critérios de aplicação a serem definidos.

## 4.6 Avaliar critérios de aplicação para cada *framework* selecionado

Nesta etapa, o objetivo é realizar a avaliação dos critérios de aplicação, que requerem testes práticos ou execução direta de código, para cada *framework* selecionado.

Para isso, uma aplicação previamente definida para cada *framework* foi desenvolvida, levando em consideração o contexto e as limitações apresentadas na etapa anterior.

## 4.7 Síntese dos resultados avaliados

Esta etapa visa realizar uma síntese dos resultados obtidos ao analisar os *frameworks* em critérios de contexto e de aplicação, conforme foi descrito nas seções 4.4 e 4.6. Com base nisso, é realizada uma avaliação dos resultados obtidos.

Essa etapa permite identificar e destacar as diferenças e contexto de uso de cada *framework* estudado. Tendo a análise desses resultados como fundamental para a conclusão deste trabalho, fornecendo informações diversas sobre cada *framework* abordado.

# 5 RESULTADOS

Neste capítulo, são apresentados os resultados obtidos provenientes da execução dos procedimentos metodológicos definidos neste trabalho.

## 5.1 Critérios de seleção dos *frameworks*

Após a revisão bibliográfica para os trabalhos relacionados elencados, observa-se que, na maioria dos estudos, a seleção de *frameworks* é baseada em critérios de popularidade. Esses critérios podem envolver pesquisas de mercado, análise de fóruns ou sites relacionados, e até mesmo o número de estrelas nos repositórios dos *frameworks* ou a quantidade de questões relacionadas no Stack Overflow.

Portanto, neste trabalho, a popularidade dos *frameworks* é considerada um critério definido e relevante. Esse critério também foi abordado em outros estudos da literatura, como os de (FRANCO, 2011) e (ALMEIDA, 2018).

No entanto, para esta pesquisa, foi adotada uma abordagem mais específica para avaliar a popularidade dos *frameworks*, utilizando o número de estrelas na plataforma do GitHub como métrica. Essa abordagem também foi empregada em estudos anteriores, conforme demonstrado por (OLIVEIRA, 2019).

Além disso, o presente autor estabeleceu algumas limitações para esta pesquisa. Essas limitações incluem a seleção de *frameworks* com código aberto disponível na plataforma GitHub, que sejam escritos na linguagem de programação Java e que pertençam à arquitetura de aplicação *web back-end*.

O Quadro 2 apresenta uma comparação dos trabalhos citados anteriormente nesta seção, destacando as diferenças ou semelhanças em relação à abordagem do critério de popularidade. Esse quadro permite uma melhor visualização das diferentes abordagens adotadas nos estudos anteriores em relação à avaliação da popularidade dos *frameworks*.

Quadro 2 – Trabalhos e seus respectivos critérios de popularidade utilizados.

| **Trabalho** | **Critério de popularidade** |
|---|---|
| (FRANCO, 2011) | Pesquisa de mercado na época |
| (ALMEIDA, 2018) | GitHub e Stack Overflow |
| (OLIVEIRA, 2019) | Estrelas no GitHub |
| Presente trabalho | Estrelas no GitHub |

Fonte: Elaborado pelo autor.

## 5.2 Seleção dos *frameworks*

Conforme definido na metodologia desta etapa, o website *HotFrameworks* foi utilizado como ponto de partida para avaliar o critério de popularidade. Para realizar a pesquisa, foi buscado no website por *frameworks* com o filtro de linguagem de programação Java, e então foi retornado todos os *frameworks* escritos em Java e suas respectivas pontuações, baseada na popularidade dos projetos desenvolvidos no GitHub e no número de questões referenciando os no Stack Overflow (FERREIRA; ZUCHI, 2021). De modo que para este trabalho foi considerado apenas os dez mais populares *frameworks* conforme os resultados apresentados na Tabela 1 abaixo.

Tabela 1 – *Frameworks* e suas respectivas pontuações obtidas no site HotFrameworks.

| ***Framework*** | ***Pontuação*** |
|---|---|
| Spring Boot | 90 |
| JSF | 81 |
| Google Web Toolkit | 77 |
| JHipster | 72 |
| Blade | 68 |
| Vert.x | 67 |
| Dropwizard | 65 |
| Wicket | 63 |
| Vaadin | 61 |
| Struts | 58 |

Fonte: Adaptado de (HOTFRAMEWORKS, 2023).

Em seguida, foi realizada uma análise dos dez *frameworks* mencionados anteriormente para verificar se possuíam código aberto disponível no repositório do GitHub. Para cada *framework*, foi buscado seu repositório de origem no GitHub. Verificou-se que os *frameworks* JSF, Google Web Toolkit, JHipster e Wicket não apresentavam código aberto na plataforma do GitHub, sendo assim, foram excluídos deste estudo. Por outro lado, os *frameworks* Spring Boot, Blade, Vert.x, Dropwizard, Vaadin e Struts apresentavam código aberto disponível no GitHub e também eram de arquitetura *web back-end*, portanto, foram selecionados como candidatos para este trabalho. Seguidamente, foi realizada uma análise no número de estrelas presentes nos repositórios de cada um dos *frameworks* candidatos no GitHub, visando selecionar os *frameworks* em Java com maior número de estrelas.

A Tabela 2 a seguir mostra os *frameworks* candidatos e suas respectivas quantidades de estrelas em seus repositórios de código aberto no GitHub.

Para este trabalho, foi definido a escolha de três *frameworks* com o maior número de

Tabela 2 – *Frameworks* e seus números de estrelas na plataforma GitHub.

| Framework | Número de estrelas |
|---|---|
| Spring Boot | 67.600 |
| Vert.x | 13.600 |
| Dropwizard | 8.400 |
| Vaadin | 1.800 |
| Struts | 1.200 |

Fonte: Elaborado pelo autor.

estrelas. Portanto, os *frameworks* selecionados como objetos de estudo para este trabalho são: **Spring Boot**, **Vert.x** e **Dropwizard**.

### *5.2.1 Descrição geral de cada framework selecionado*

Como demonstrado na seção acima, três *frameworks* foram selecionados para este estudo. Nessa seção são apresentadas as descrições de cada *framework* selecionado.

Além disso, se faz de grande importância estabelecer uma compreensão do contexto de uso de cada *framework*. Para isso, um simples projeto *Hello World* foi elaborado em cada *framework*, estrutura amplamente reconhecida como o primeiro passo para testar os fundamentos de uma linguagem de programação e compreender seu contexto de uso, abrangendo variáveis, tipos de dados, entre outros aspectos.

#### *5.2.1.1 Spring Boot*

O Spring Boot é um *framework* que simplifica e acelera o desenvolvimento de aplicativos *web* e microsserviços. Ele permite a criação de aplicativos independentes baseados na estrutura Spring para produção, que podem ser executados com facilidade. A maioria dos aplicativos Spring Boot exige configurações mínimas (SPRING, 2023a).

Por exemplo, o Spring Boot padrão incorpora o Apache Tomcat, um servidor *web* Java amplamente utilizado. Isso significa que é possível iniciar um aplicativo rapidamente, sem a necessidade de configurações adicionais. Ao fazer uso das ferramentas fornecidas pela estrutura Spring, você se beneficia de soluções prontas para uso, economizando tempo e esforço, em vez de escrever uma quantidade significativa de código adicional (SPRING, 2023a).

O Spring Boot oferece suporte às linguagens de programação Java, Kotlin e Groovy, permitindo o desenvolvimento de uma variedade de aplicações, incluindo microsserviços, aplicações *web* responsivas, sistemas reativos e muito mais.

#### *5.2.1.2 Hello World em Spring Boot*

Para realizar esse simples projeto, inicialmente é necessário criar um projeto base com Spring Boot. Para isso, o *Spring Initializr* foi utilizado e apenas uma dependência foi selecionada, a de *web* e a linguagem de programação base foi o Java.

Inicialmente na classe criada *AppApplication*, três anotações foram utilizadas: *RestController*, *RequestMapping* e *SpringBootAplication*, a primeira delas mapeia a classe como um controlador Spring, que são estruturas responsáveis por lidar com solicitações HTTP e retornar respostas adequadas, a segunda mapeia uma URL base para todas as requisições feitas na rota *"/api"* e a última anotação mapeia a classe como uma classe de configuração e de entrada padrão do Spring Boot.

Em seguida, foi definido um método público que retorna uma String chamado *helloWorld*, que foi anotado com *GetMapping*, que significa que uma rota HTTP com o verbo *GET* foi configurada na rota *"/helloWorld"*.

Sendo assim, o projeto foi configurado e após executar o aplicativo, o Spring Boot identifica a classe controladora e mapeia a rota *"/api/helloWorld"* no ambiente e na porta local, responsável por retornar a *string "Hello World!"*. A Figura 7 demonstra a configuração do projeto *Hello World*.

Figura 7 – *Hello World* construído com o *framework* Spring Boot em Java.

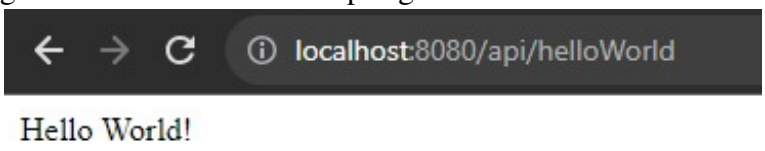
```java
@RestController
@RequestMapping("/api")
@SpringBootApplication
public class AppApplication {

  @GetMapping("/helloWorld")
  public String helloWorld() {
    return "Hello World!";
  }
```

Fonte: Elaborado pelo autor.

A Figura 8 a seguir demonstra o retorno obtido ao acessar a rota mapeada localmente no Spring Boot.

Figura 8 – *Hello World* em Spring Boot via URL.




Fonte: Elaborado pelo autor.

#### *5.2.1.3 Vert.x*

Vert.x é um *framework* que funciona mais como um conjunto de ferramentas, sendo utilizado principalmente para a criação de aplicativos reativos na *Java Virtual Machine* (JVM). [1]

Além disso, o Vert.x é simultâneo, assíncrono, adaptável e dimensionável de acordo com suas necessidades, oferecendo uma pilha de aplicativos modernos de diferentes tipos, e se algo não for encontrado, há uma grande chance de que alguém tenha feito isso no ecossistema de código aberto Vert.x mais amplo (VERT.X, 2023a).

O Vert.x é compatível com duas linguagens de programação: Java e Kotlin.

#### *5.2.1.4 Hello World em Vert.x*

Para realizar o simples projeto *Hello World* em Vert.x, é necessário definir um projeto base usando o *framework*. Para isso, o Vert.x *Starter* foi utilizado, possibilitando a criação do projeto base com uma única dependência de *web* e com a linguagem de programação Java.

A classe inicial é definida como uma subclasse da classe *AbstractVerticle*, sendo definido assim para realizar a configuração de seu próprio *verticle* [2]. Em vez de implementar o *verticle* diretamente, muitas vezes é mais simples apenas estender esta classe, e no caso mais simples, basta substituir o método *start(Promise)* para iniciar (VERT.X, 2023b).

Seguidamente, um servidor HTTP é criado utilizando o Vert.x juntamente com um manipulador de requisições para lidar com solicitações HTTP. Neste caso, quando uma solicitação é recebida, o cabeçalho padrão dela é definido como texto e a resposta retornada é a *string Hello World!*. De modo que o servidor também é configurado para executar na porta 8080 e com duas manipulações de sucesso ou falha, tratadas por meio do parâmetro da função. A Figura 9 demonstra a configuração do projeto *Hello World*.

[1] A JVM é a máquina virtual do Java, sendo uma ferramenta importante para interpretar o código Java e rodá-lo em qualquer plataforma (COODESH, 2023b).

[2] Verticles são pedaços de código que são implantados e executados pelo Vert.x. Os *verticles* podem ser escritos em qualquer uma das linguagens suportadas pelo Vert.x e um único aplicativo pode incluir *verticles* escritos em múltiplas linguagens.

Figura 9 – *Hello World* construído com o *framework* Vert.x em Java.

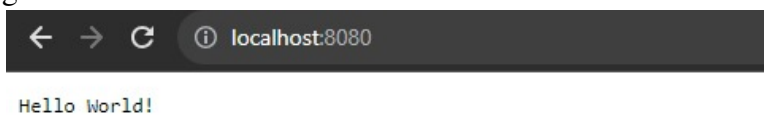
```java
public class MainVerticle extends AbstractVerticle {

  @Override
  public void start(Promise<Void> startPromise) throws Exception {
    vertx.createHttpServer().requestHandler(req -> {
      req.response()
        .putHeader( s: "content-type", s1: "text/plain")
        .end( s: "Hello World!");
    }).listen( port: 8080, http -> {
      if (http.succeeded()) {
        startPromise.complete();
        System.out.println("HTTP server started on port 8080");
      } else {
        startPromise.fail(http.cause());
      }
    });
  }
}
```

Fonte: Elaborado pelo autor.

A Figura 10 a seguir demonstra o retorno obtido ao acessar a rota mapeada localmente pelo Vert.x.

Figura 10 – *Hello World* em Vert.x via URL.




Fonte: Elaborado pelo autor.

#### *5.2.1.5 Dropwizard*

Dropwizard é um *framework* exclusivamente Java para o desenvolvimento de serviços *web* REST de alto desempenho e fáceis de operar.

O Dropwizard unifica bibliotecas estáveis e maduras do ecossistema Java em um pacote simples e leve para a construção de sistemas. Essas funcionalidades são extraídas para bibliotecas reutilizáveis, sendo assim o seu serviço é enxuto e focado, reduzindo o tempo de lançamento no mercado e encargos de manutenção de software (DROPWIZARD, 2011)

#### *5.2.1.6 Hello World em Dropwizard*

Para realizar o simples projeto *Hello World* em Dropwizard, é necessário definir um projeto base usando o *framework*. Para isso, o arquétipo do Dropwizard foi utilizado, possibilitando a criação do projeto base na linguagem de programação Java.

Foi definida uma classe *HelloResource*, que dentro do Dropwizard funciona como

uma classe de recurso para a manipulação REST. Nessa classe, um método HTTP *GET* de retorno *String* foi definido e anotado para ter seu mapeamento na rota *"/helloWorld"*. A Figura 11 a seguir demonstra a definição do recurso para o *Hello World*.

Figura 11 – *Hello World* construído com o *framework* Dropwizard em Java.

```
@Path("/helloWorld")
public class HelloResource {

    1 usage
    public HelloResource() { }

    @GET
    public String get() {
        return "Hello World!";
    }
```

Fonte: Elaborado pelo autor.

Para ocorrer o funcionamento correto do recurso e da rota *GET*, ela precisa ser necessariamente registrada pelo Jersey [3], utilizado por padrão no Dropwizard e no método *run* da classe de aplicação padrão que estende de outra classe de configuração padrão. A Figura 12 demonstra o registro do recurso definido anteriormente no método específico.

Figura 12 – Registro de um recurso para o Jersey via função no Dropwizard.

```
public class HelloWorldServiceApplication extends Application<HelloWorldServiceConfiguration> {

    public static void main(final String[] args) throws Exception {...}

    @Override
    public String getName() { return "Hello World!"; }

    @Override
    public void initialize(final Bootstrap<HelloWorldServiceConfiguration> bootstrap) {}

    @Override
    public void run(final HelloWorldServiceConfiguration configuration,
                    final Environment environment) {
        environment.jersey().register(new HelloResource());
    }
```

Fonte: Elaborado pelo autor.

E por fim, a Figura 13 demonstra o retorno ao acessar a rota localmente definida pelo Dropwizard.

[3] Jersey é um *framework* para aplicações REST. Ele permite que seja possível escrever classes limpas e testáveis que mapeiam solicitações HTTP para objetos Java simples (DROPWIZARD, 2023b).

Figura 13 – *Hello World* em Dropwizard via URL.


Fonte: Elaborado pelo autor.

## 5.3 Critérios de contexto e aplicação para análise dos *frameworks*

Após a revisão bibliográfica, verificou-se que vários critérios são considerados na avaliação de *frameworks*, abrangendo desde critérios de desempenho e escalabilidade até critérios mais teóricos, como tamanho da documentação ou presença de uma comunidade ativa. O Quadro 3 apresenta alguns trabalhos utilizados como referência, juntamente com alguns dos critérios selecionados por eles para analisar os *frameworks*.

Quadro 3 – Trabalhos e alguns dos critérios selecionados para analisar seus *frameworks*.

| **Trabalho** | **Critérios considerados** |
|---|---|
| (FRANCO, 2011) | Popularidade, vagas de emprego e fóruns |
| (ALMEIDA, 2018) | Desempenho, documentação, curva de aprendizagem e configuração |
| (OLIVEIRA, 2019) | Documentação, popularidade, tamanho da comunidade e suporte a REST |

Fonte: Elaborado pelo autor.

Com o auxílio da revisão bibliográfica e da análise desses trabalhos, este estudo considera diversos critérios significativos para a análise dos três *frameworks* selecionados na etapa anterior, apresentando critérios já vistos nos estudos apresentados e novos critérios identificados pelo presente autor. Os critérios selecionados abrangem critérios de contexto (de 1 a 5) e critérios de aplicação (6 e 7), sendo eles:

1. Popularidade.
2. Atividade da comunidade.
3. Qualidade da documentação.
4. Avaliação de artigos publicados.
5. Suporte à inicialização de projeto.
6. Suporte a mapeamento objeto-relacional.
7. Suporte à arquitetura de software REST.

A seguir, é descrito como cada critério é avaliado para cada *framework* selecionado.

***Popularidade***

Neste critério, através da ferramenta de busca gratuita do Google, o Google Trends, é usado o nome de cada *framework* como termo de pesquisa em formato de uma *string* e aplicando filtros específicos disponibilizados pela própria ferramenta, visando mensurar a popularidade relativa dos *frameworks* ao longo do tempo.

***Atividade da comunidade***

Neste critério, é avaliado a atividade da comunidade a partir de fontes relevantes no contexto da tecnologia. Para essa análise, as fontes escolhidas foram o StackOverflow e o GitHub, duas plataformas populares que desempenham um papel fundamental como fóruns de discussão para questões relacionadas a programação, incluindo *frameworks*.

***Qualidade da documentação***

Neste critério, é avaliado o suporte a recursos de aprendizado, considerando a disponibilidade de funcionalidades ou tutoriais objetivos e a extensão da documentação, diretamente nas próprias páginas dos *frameworks*.

***Avaliação de artigos publicados***

Neste critério, é avaliado a relevância dos *frameworks*, buscando dados a partir de trabalhos acadêmicos ou técnicos, que abordam ou fazem menção aos referidos *frameworks*. Para isso, o software gratuito *Publish or Perish* é utilizado, responsável por trazer dados relevantes sobre trabalhos acadêmicos de diferentes bases de dados a partir de filtros específicos.

***Suporte à inicialização de projeto***

Neste critério, é avaliado a simplicidade e praticidade para iniciar um novo projeto base utilizando cada *framework*, considerando a facilidade de incorporar recursos e dependências essenciais.

***Suporte a mapeamento objeto-relacional***

Neste critério, é avaliado se cada *framework* analisado oferece suporte a tecnologia de mapeamento objeto-relacional e como essa integração é realizada. O mapeamento objeto-relacional refere-se a estruturas desenvolvidas para transformar os dados da estrutura lógica de um banco de dados relacional em objetos relacionais. O objetivo é simplificar o acesso de maneira rápida e eficiente.

***Suporte à arquitetura de software REST***

Neste critério, é avaliado se cada *framework* em questão dá suporte a arquitetura de software REST e como é feito isso, que implica no uso de elementos como o protocolo HTTP para lidar com requisições e respostas e formatos como o JSON para a manipulação de dados e recursos.

## 5.4 Avaliação dos critérios de contexto para cada *framework*

Na etapa anterior, foram definidos todos os critérios selecionados neste trabalho e a configuração de sua avaliação para os *frameworks*. No entanto, nesta etapa, apenas os critérios considerados de contexto são avaliados.

### *5.4.1 Popularidade*

A popularidade dos *frameworks* é medida através da ferramenta Google Trends. O Google Trends é uma ferramenta do Google que permite a visualização de tópicos que ocorrem praticamente em tempo real, tendo importância para apresentar dados de tendências (GOOGLE, 2023).

Para isso, o Google Trends oferece uma série de filtros que podem ser aplicados à pesquisa, permitindo uma análise mais precisa. Esses filtros incluem a seleção da região geográfica em que a pesquisa é realizada, a definição do intervalo de tempo desejado, que pode ser tanto um período pré-definido quanto um intervalo manual, a escolha entre diversas categorias nas quais a pesquisa se encaixa, e a opção de especificar o tipo de pesquisa, como *web*, notícias, *YouTube*, entre outros.

Os parâmetros escolhidos para os filtros da ferramenta, utilizados para cada *fra-*

*mework*, são os seguintes: a região geográfica é definida como "Mundo", o intervalo de tempo é estabelecido em cinco anos, a categoria selecionada é "Java"e o tipo de pesquisa é restrito à categoria *web*. Essas configurações foram escolhidas visando obter resultados relevantes e abrangentes em todo o mundo em um intervalo de tempo significativo, para a análise de popularidade dos *frameworks*. A Figura 14 a seguir demonstra os filtros aplicados na ferramenta do Google Trends para a avaliação do critério de popularidade.

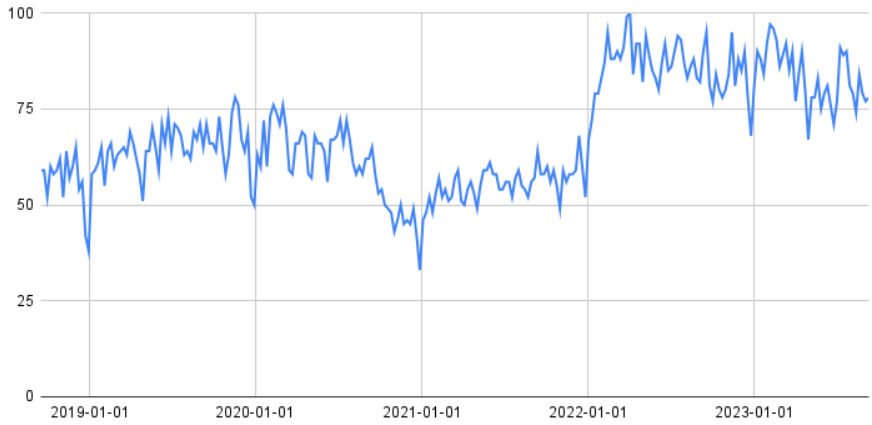
Figura 14 – Filtros específicos na ferramenta Google Trends.


Fonte: (GOOGLE, 2023).

#### *5.4.1.1 Spring Boot*

Para avaliar a popularidade do *framework* Spring Boot na ferramenta, o termo de pesquisa "Spring Boot"foi empregado em conjunto com o filtro definido na descrição da seção anterior. Os resultados obtidos estão apresentados na Figura 15.

Figura 15 – Pesquisa no Google Trends para o *framework* Spring Boot.




Fonte: Adaptado de (GOOGLE, 2023).

É perceptível na Figura 15 que, ao longo de cinco anos de análise, o *framework* Spring Boot demonstrou uma notável estabilidade e relevância no que tange à popularidade.

Inicialmente, configurado como o final de 2018, o *framework* apresentou um nível de interesse significativo, superando o valor de 50 no gráfico [4]. Durante esse ano, foram registradas

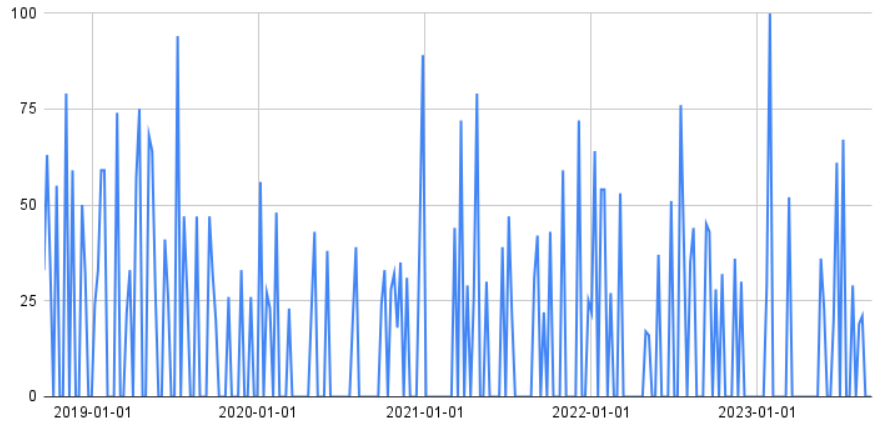
[4] Os números representam o interesse de pesquisa relativo ao ponto mais alto no gráfico de uma determinada região em um dado período. Um valor de 100 representa o pico de popularidade de um termo. Um valor de 50 significa que o termo teve metade da popularidade. Uma pontuação de 0 significa que não havia dados suficientes sobre o termo (GOOGLE, 2023).

quedas no interesse, embora não tão acentuadas a ponto de cair abaixo do limiar de 50, até que ocorreu um declínio mais pronunciado no final de 2018. No início de 2019, o Spring Boot voltou a apresentar um crescimento e embora tenha havido períodos breves de declínio nesse interesse ao longo do tempo, o *framework* demonstrou uma notável capacidade de ressurgir como tendência, mantendo-se em alta até o final de 2020, quando ocorreu uma breve queda. No entanto, esse declínio foi prontamente revertido em 2021, e o *framework* permaneceu em alta e experimentou um crescimento constante durante os anos de 2022 e 2023, com quedas não tanto significativas e apresentando níveis de interesse entre 70 e 100.

#### *5.4.1.2 Vert.x*

A mesma abordagem foi utilizada para avaliar a popularidade do *framework* Vert.x, com o termo de pesquisa "Vert.x"junto ao filtro especificado anteriormente, aplicados na ferramenta. Os resultados da pesquisa estão dispostos na Figura 16.

Figura 16 – Pesquisa no Google Trends para o *framework* Vert.x.




Fonte: Adaptado de (GOOGLE, 2023).

Ao analisar a Figura 16, é evidente que o *framework* Vert.x passou por distintas fases ao longo do tempo, que incluem períodos de interesse mínimo, moderado e elevado, alternando entre eles constantemente.

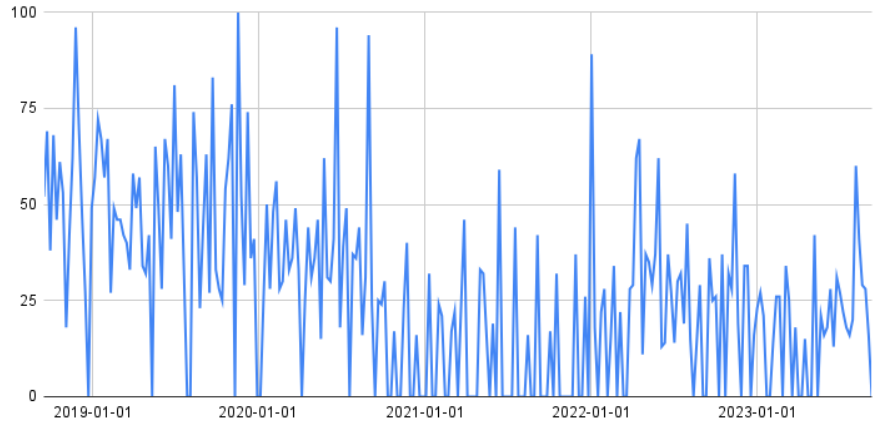
No início, configurado como o final de 2018, o *framework* teve um interesse moderado, mas logo experimentou quedas acentuadas nos níveis de interesse, chegando a patamares mínimos. No ano seguinte, em 2019, o *framework* viu um ressurgimento em seu interesse, embora de forma intermitente, com destaque para um pico notável de aproximadamente 90. No entanto, em 2020, o interesse pelo *framework* diminuiu em relação ao ano anterior, mantendo

uma alternância entre níveis mínimos, moderados e elevados de interesse, sendo o último um valor de aproximadamente 85. Esses padrões de alternância de níveis de interesse se mantiveram até o ano atual, 2023, em que se registra o pico mais alto de interesse pelo *framework*, de 100.

#### *5.4.1.3 Dropwizard*

O termo "Dropwizard"junto ao filtro de especificação da ferramenta do Google Trends, foram empregados para avaliar a popularidade do *framework*. Os resultados estão expostos na Figura 17.

Figura 17 – Pesquisa no Google Trends para o *framework* Dropwizard.



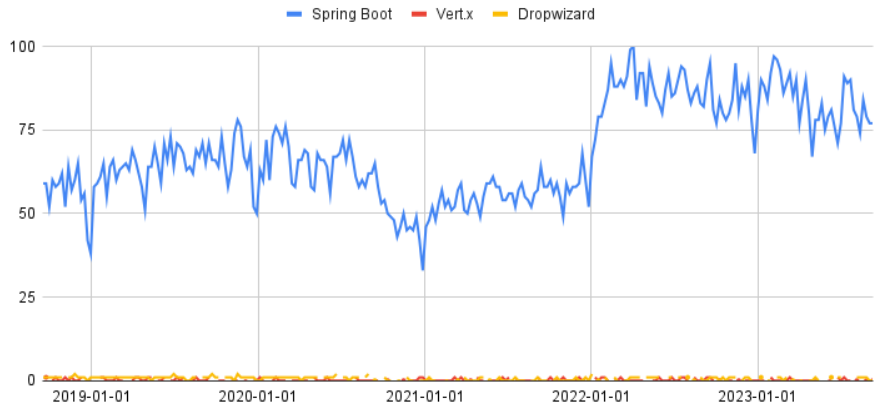

Fonte: Adaptado de (GOOGLE, 2023).

Analisando a Figura 17 é possível notar que o *framework* Dropwizard apresentou alternados picos de interesse ao longo do intervalo de cinco anos em que a pesquisa foi empregada. Isso inclui períodos de interesse mínimos, moderados e elevados.

Inicialmente, configurado como o final de 2018, o *framework* Dropwizard despertou um interesse considerável, atingindo níveis notáveis de popularidade. Ao longo do ano, manteve uma tendência positiva, com flutuações que indicavam um interesse sustentado. Em 2019, o Dropwizard continuou a atrair atenção, embora de forma intermitente, alcançando um pico notável de cerca de 100 durante o final do ano. Entretanto, em 2020, o interesse pelo *framework* diminuiu em comparação ao ano anterior, mantendo uma alternância entre níveis mínimos, moderados e elevados de interesse, de modo que estes padrões persistiram até o ano atual, 2023, que iniciou com o índice de interesse de 25, aproximadamente, e continuou alternando entre interesse mínimo e moderado, até que o cenário recebeu um destaque nas últimas fases com um pico de interesse próximo de 60.

Além disso, outra pesquisa no Google Trends foi conduzida, utilizando os três termos mencionados anteriormente de forma simultânea, com o propósito de proporcionar uma visão mais abrangente do interesse ao longo dos anos em relação aos *frameworks* Spring Boot, Vert.x e Dropwizard. A Figura 18 mostra o resultado da pesquisa.

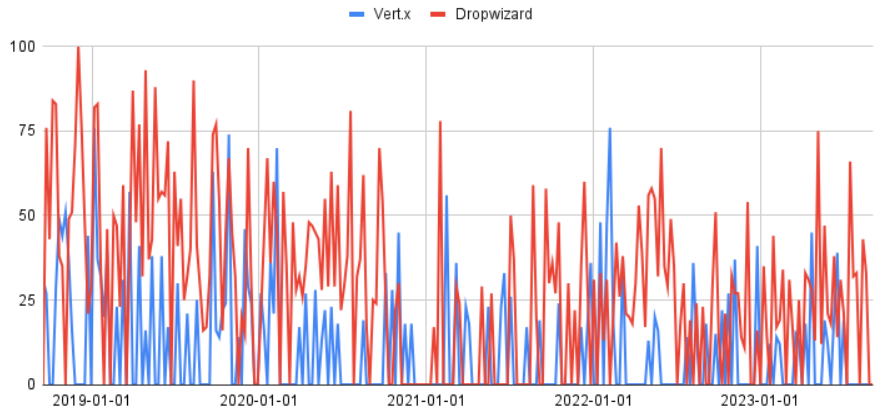
Figura 18 – Pesquisa simultânea no Google Trends para os *frameworks* Spring Boot, Vert.x e Dropwizard.




Fonte: Adaptado de (GOOGLE, 2023).

Como é possível perceber, o *framework* Spring Boot esteve em alta durante todo o intervalo de tempo, apresentando variações não tanto expressivas. Enquanto os *frameworks* Vert.x e Dropwizard apresentaram interesse mínimo e a constância de baixo interesse durante todo o período. Portanto, para uma melhor visualização e apuração dados acerca da popularidade ao longo do tempo dos *frameworks*, a mesma pesquisa é aplicada apenas para os *frameworks* Vert.x e Dropwizard, cujos resultados estão dispostos na Figura 19.

Figura 19 – Pesquisa simultânea no Google Trends para os *frameworks* Vert.x e Dropwizard.




Fonte: Adaptado de (GOOGLE, 2023).

Analisando a Figura 19, é possível perceber a predominância do interesse ao longo

do período pelo *framework* Dropwizard sob o *framework* Vert.x. O Vert.x iniciou o período com um interesse baixo, permanecendo em patamares baixos ou moderados com índices de até 50 até o final de 2018. No entanto, a partir de 2019, houve um aumento gradual, com picos intermitentes, mas ainda se mantendo relativamente baixo em relação aos níveis do Dropwizard, que experimentou uma trajetória diferente, iniciando com interesse nulo em 2018 e apresentando algumas variações nesse mesmo ano que destaca o maior pico de interesse do *framework*, de 100.

O cenário se repete durante todo o período, até o ano atual de 2023, o *framework* Dropwizard apresenta níveis de interesse acima dos níveis de interesse do *framework* Vert.x, sendo o maior nível de interesse do último, 75, no início de 2022.

### *5.4.2 Atividade da comunidade*

Para avaliar a atividade das comunidades associadas a cada um dos *frameworks* abordados neste estudo, como definido na etapa metodológica, o presente autor escolheu duas fontes de dados relevantes no contexto da tecnologia da informação: o StackOverflow e o GitHub.

Ambas plataformas oferecem informações sobre tecnologias, projetos de código aberto e questões relacionadas a *frameworks* e programação, tornando-as recursos valiosos para esta análise.

Para o StackOverflow, é avaliado o número de perguntas relevantes, filtradas posteriormente por tags específicas e organizadas conforme atividades mais recentes para cada um dos *frameworks*.

Para o GitHub, é considerado o número de *issues* como fator de análise. *Issues* são recursos usados para os usuários rastrearem problemas ou tarefas, podendo levantar melhorias, bugs, discussões e qualquer outro tipo de colaboração em um repositório de código ou em outros tipos de projetos na plataforma (Github, Inc., 2023).

#### *5.4.2.1 Spring Boot*

Para realizar a análise no StackOverflow, a tag “Spring Boot” foi empregada para pesquisar perguntas relevantes, resultando em um total de cento e quarenta e três mil cento e sessenta e nove questões pertinentes. Além disso, as perguntas foram classificadas por ordem de atividade mais recente, destacando um número alto de questões modificadas nos últimos minutos.

A pesquisa no GitHub foi conduzida na seção de *issues*, usando o termo “Spring Boot”. O resultado dessa pesquisa revelou um total de cento e sessenta e três mil registros, ordenados pelos mais recentes. Foi revelado um alto número de *issues* abertas nas últimas horas.

Os resultados para as duas análises estão apresentados na Tabela 3.

#### *5.4.2.2 Vert.x*

No fórum StackOverflow, a tag “Vert.x” foi aplicada na seção de busca para encontrar o total de perguntas disponíveis, resultando em um total de duas mil e seiscentos e quarenta e quatro, sendo perguntas e questões relacionadas ao uso e *features* do *framework*, tendo como mais recente as questões abertas nos últimos dias.

Na seção de *issues* do GitHub, foi utilizado o termo de busca “Vert.x”. Como resultado, foi obtido um retorno de onze mil registros, os quais foram também ordenados por data, apresentando os mais recentes no topo. Nesse processo, foi possível identificar a presença de *issues* que foram abertas nos últimos dias como as mais recentes.

Os resultados para ambas análises estão apresentados na Tabela 3.

#### *5.4.2.3 Dropwizard*

No fórum StackOverflow, a tag “Dropwizard” foi usada para buscar pelo número de perguntas levantadas no site relacionadas a essa tag. Como resultado, foram retornadas duas mil e onze questões relacionadas ao *framework*, com questões abertas nos últimos dias.

No GitHub, na seção de *issues*, a pesquisa foi realizada usando o termo de busca “Dropwizard”. Foram obtidos um total de oito mil e seiscentos resultados, os quais foram classificados com base na atividade mais recente, exibindo as *issues* mais recentemente abertas, que foram nas últimas horas. Na Tabela 3 abaixo os resultados são exibidos.

Tabela 3 – Total de *issues* e perguntas para os *frameworks* Spring Boot, Vert.x e Dropwizard.

| **Métricas** | **Spring Boot** | **Vert.x** | **Dropwizard** |
|---|---|---|---|
| Total de perguntas | 143.169 | 2.644 | 2.011 |
| Total de *issues* | 163.000 | 11.000 | 8.600 |

Fonte: Elaborado pelo autor.

### *5.4.3 Qualidade da documentação*

Para avaliar a qualidade da documentação dos *frameworks* analisados neste estudo, as próprias páginas oficiais de documentação de cada *framework* são utilizadas. A análise tem embasamento no suporte a recursos de aprendizado, incluindo a presença de tutoriais objetivos e a disponibilização de funcionalidades, bem como a extensão da documentação fornecida.

#### *5.4.3.1 Spring Boot*

No site do Spring Boot, existe uma seção dedicada ao aprendizado, oferecendo uma variedade de recursos para auxiliar no desenvolvimento com o *framework*. Essa seção está organizada em diferentes abas, cada uma com seu propósito específico.

Na aba visão geral, são apresentados os principais tópicos relacionados ao Spring Boot. Ela destaca os guias mais populares, que funcionam como tutoriais práticos para implementar funcionalidades específicas com o *framework*. Esses guias incluem estimativas de tempo de implementação, informações sobre as ferramentas e tecnologias necessárias, como a versão do Java e do Maven, e fornecem um passo a passo detalhado para criar a funcionalidade desejada. Além disso, nessa aba, é possível encontrar um canal no YouTube que disponibiliza vídeos recentes sobre o *framework* Spring Boot, bem como uma aba de podcasts relacionados.

A aba início rápido oferece um processo simplificado para criar uma aplicação *web* básica que pode ser executada em um navegador. Esse processo é dividido em três etapas simples: inicialização do projeto usando o *initializr* do Spring Boot, adição de código à classe principal e realização de teste prático. Apesar da simplicidade, esse guia é altamente funcional e útil para iniciar rapidamente uma aplicação com Spring Boot.

A aba de guia é a parte mais extensa e valiosa do Spring Boot. As guias são divididas em três tipos: guias de início, guias de tópicos e tutoriais. Os guias disponibilizados no site foram cuidadosamente projetados para promover a produtividade imediata, permitindo que seja possível aproveitar os lançamentos e técnicas mais recentes do Spring, conforme recomendado pela própria equipe, independente do que esteja sendo construído (SPRING, 2023a).

As guias de início oferecem uma introdução rápida à estrutura e às convenções do Spring Boot, com um tempo de conclusão estimado de quinze a vinte minutos. Os guias de tópicos exploram temas mais amplos e complexos com um tempo de conclusão de no máximo sessenta minutos, enquanto os tutoriais aprofundam ainda mais o conhecimento, apresentando

projetos práticos que requerem de duas a três horas para serem concluídos e fornecem informações contextuais e aprofundadas, particularmente úteis em cenários do mundo real.

Por fim, a aba blog traz uma coleção de artigos, publicações, anúncios de lançamentos, notícias e informações sobre eventos relacionados à comunidade do Spring Boot. Essa guia é uma fonte valiosa para se manter atualizado com as últimas novidades e tendências no mundo do Spring Boot.

#### *5.4.3.2 Vert.x*

O *framework* Vert.x apresenta em sua página inicial guias cuidadosamente organizados, fornecendo informações de acordo com sua nomenclatura específica.

A guia de início oferece uma visão geral do Vert.x, abordando seu funcionamento interno e processamento. Adicionalmente, inclui uma seção de início rápido, contendo um tutorial simples e um iniciador de projetos que simplifica a criação de aplicações baseadas em Vert.x.

A documentação do Vert.x constitui outra seção notável. Ela abrange uma ampla gama de tópicos, incluindo os conceitos fundamentais da *web* no contexto do Vert.x e a manipulação de respostas no framework, entre outros. Cada tópico na documentação é acompanhado de uma descrição abrangente e exemplos funcionais relevantes. Além disso, é possível buscar informações sobre tópicos específicos de interesse para explorar o uso do *framework*.

Outra seção enriquecedora no site do Vert.x é a seção de recursos. Nesta área, é disponibilizada uma lista de perguntas frequentes feitas por outros desenvolvedores, bem como canais de comunicação, como o Discord, fóruns de programação e grupos de usuários que apresentam questões relevantes do *framework*. A seção como fazer, é particularmente valiosa, pois orienta na implementação de funcionalidades específicas com o Vert.x. Ela detalha o que é realizado, as tecnologias e ferramentas necessárias, uma explicação detalhada do contexto e motivo de uso no *framework* Vert.x, seguido da criação do projeto base, implementação de funcionalidades, e, por fim, inclui uma seção dedicada aos testes. Além disso, a seção fornece links externos para guias avançados e conteúdos ricos em exemplos práticos do uso do *framework* em diferentes contextos.

Finalmente, o site do Vert.x apresenta duas seções adicionais: o blog e a comunidade, complementando o conjunto abrangente de recursos e suporte oferecidos aos desenvolvedores, que funcionam como um suporte extra para os usuários.

#### *5.4.3.3 Dropwizard*

No site oficial do framework *Dropwizard*, uma variedade de guias destacam sua documentação. Esses guias abrangem diversas áreas, incluindo informações gerais que respondem às dúvidas frequentes relacionadas ao contexto do *framework*, bem como detalhes sobre sua descrição e notas de lançamento.

Uma guia particularmente interessante é a de iniciando com o Dropwizard, cujo principal objetivo é orientar o usuário pelo processo de criação de um projeto simples usando o Dropwizard. O início deste guia apresenta as bibliotecas e tecnologias que o Dropwizard utiliza e explora os motivos por trás de suas escolhas. Em seguida, ele oferece tutoriais práticos que detalham a criação do projeto base com o arquétipo do Dropwizard, gerenciamento de dependências e a criação de classes de configuração, aplicação, representação e recursos. Esses tutoriais são acompanhados por explicações detalhadas e códigos funcionais e práticos.

Adicionalmente, o Dropwizard disponibiliza uma abrangente seção de guia do usuário. Nesta seção, os usuários podem encontrar tópicos relevantes no contexto do *framework*, incluindo informações sobre o cliente do Dropwizard, injeção de dependências, autenticação, entre outros. Cada um desses guias de usuário oferece uma breve descrição de como o Dropwizard fornece cada recurso, complementada por códigos práticos que facilitam o entendimento e a aplicação desses recursos dentro do contexto do *framework*.

### *5.4.4 Avaliação de artigos publicados*

Para realizar a avaliação dos artigos publicados, é utilizada a ferramenta de software gratuita *Publish or Perish*, conforme mencionado anteriormente. O *Publish or Perish* é um software especializado que coleta e analisa citações acadêmicas de diversas fontes de dados. Ele extrai as citações brutas, realiza análises detalhadas e fornece uma variedade de métricas de citação, incluindo o número de artigos e o total de citações (HARZING, 2023).

O software mencionado anteriormente oferece uma variedade de ferramentas e bases de dados para a pesquisa de artigos e citações acadêmicas, com a possibilidade de aplicar filtros específicos. No entanto, para este trabalho, somente o Google Scholar é utilizado como base de dados. Além disso, foi utilizado o número máximo de resultados disponibilizados pelo software, que é mil.

Para avaliar os artigos publicados em cada um dos *frameworks*, são consideradas

as métricas de citação que abrangem trabalhos, citações, anos de citação e anos de publicação. Essas métricas incluem o número total de resultados encontrados, a soma das contagens de citações em todos os resultados encontrados, o número de anos desde o primeiro ano encontrado nos resultados selecionados até o ano da pesquisa em relação às citações, e os anos de publicação mais antigos e mais recentes encontrados nos resultados, respectivamente.

#### *5.4.4.1 Spring Boot*

Para avaliar os artigos publicados no *framework* Spring Boot, foi realizada a pesquisa com filtros de título e palavras-chave, especificamente "Spring Boot"e "Java *Framework*", para recuperar todos os artigos acadêmicos relacionados a esse contexto. Os resultados estão dispostos na Tabela 4.

#### *5.4.4.2 Vert.x*

A mesma abordagem foi utilizada para avaliar os artigos publicados para o *framework* Vert.x. A pesquisa com os filtros de título e palavras-chave, "Vert.x"e "Java *Framework*", respectivamente, foram utilizados para recuperar todos os dados relacionados a esse domínio. Os resultados estão dispostos na Tabela 4.

#### *5.4.4.3 Dropwizard*

De maneira semelhante, para o *framework* Dropwizard foi realizada a pesquisa com os filtros de título e palavras-chave: "Dropwizard"e "Java *Framework*".

Sintetizando os resultados da análise dos artigos publicados sobre os *frameworks*, é possível observar que o Spring Boot se destaca com uma expressiva quantidade de trabalhos publicados, totalizando 241. Além disso, apresenta o maior número de citações, somando 734 em trabalhos acadêmicos. Destaca-se também o amplo intervalo de citação e publicação, estendendo-se até o ano de 2023, indicando o contínuo uso e relevância do *framework* nesse contexto, além de possuir um intervalo mais abrangente em comparação aos outros dois *frameworks*.

O Vert.x, por sua vez, possui um número significativamente menor de trabalhos publicados, apenas 6, e 20 citações em trabalhos acadêmicos. Apesar de compartilhar o mesmo intervalo de publicação com o Spring Boot, indício de que possui trabalhos publicados relacionados recentemente, observa-se que as citações estão limitadas até o ano de 2020.

Já o Dropwizard apresenta uma presença mais modesta, com apenas um trabalho publicado e 10 citações, todas datadas no ano de 2014. O intervalo de publicação do Dropwizard abrange de 2014 a 2023, indicando uma menor presença na literatura acadêmica em comparação aos outros *frameworks* analisados. Os resultados sintetizados estão apresentados na Tabela 4 abaixo.

Tabela 4 – Avaliação de artigos publicados para os *frameworks* Spring Boot, Vert.x e Dropwizard pela ferramenta *Publish or Perish*.

| **Métricas** | **Spring Boot** | **Vert.x** | **Dropwizard** |
|---|---|---|---|
| Trabalhos | 241 | 6 | 1 |
| Citações | 734 | 20 | 10 |
| Anos de citação | 2013-2023 | 2013-2020 | 2014-2014 |
| Anos de publicação | 2013-2023 | 2013-2023 | 2014-2023 |

Fonte: Elaborado pelo autor.

### *5.4.5 Suporte à inicialização de projeto*

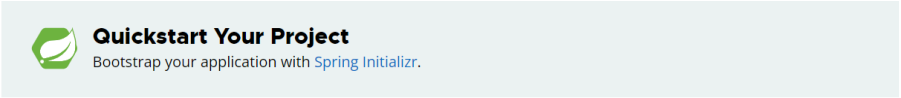
Para avaliar o critério de suporte à inicialização de projeto, é investigado se cada *framework* tem uma ferramenta de inicialização que permite que um projeto base seja gerado com configurações e dependências de desenvolvimento selecionadas manualmente, e como ela é apresentada para uso.

#### *5.4.5.1 Spring Boot*

O *framework* Spring Boot oferece a ferramenta em sua página principal para a rápida inicialização de projetos utilizando o próprio *framework*. Ao clicar neste serviço, a página é redirecionada para a ferramenta de inicialização oficial do Spring Boot, conhecida como Spring *Initializr*.

A Figura 20, apresenta o primeiro contato com a ferramenta de inicialização do Spring Boot em sua página inicial que permite que um projeto base seja configurado com características pré-definidas.

Figura 20 – Apresentação da ferramenta de inicialização de projeto do Spring Boot.

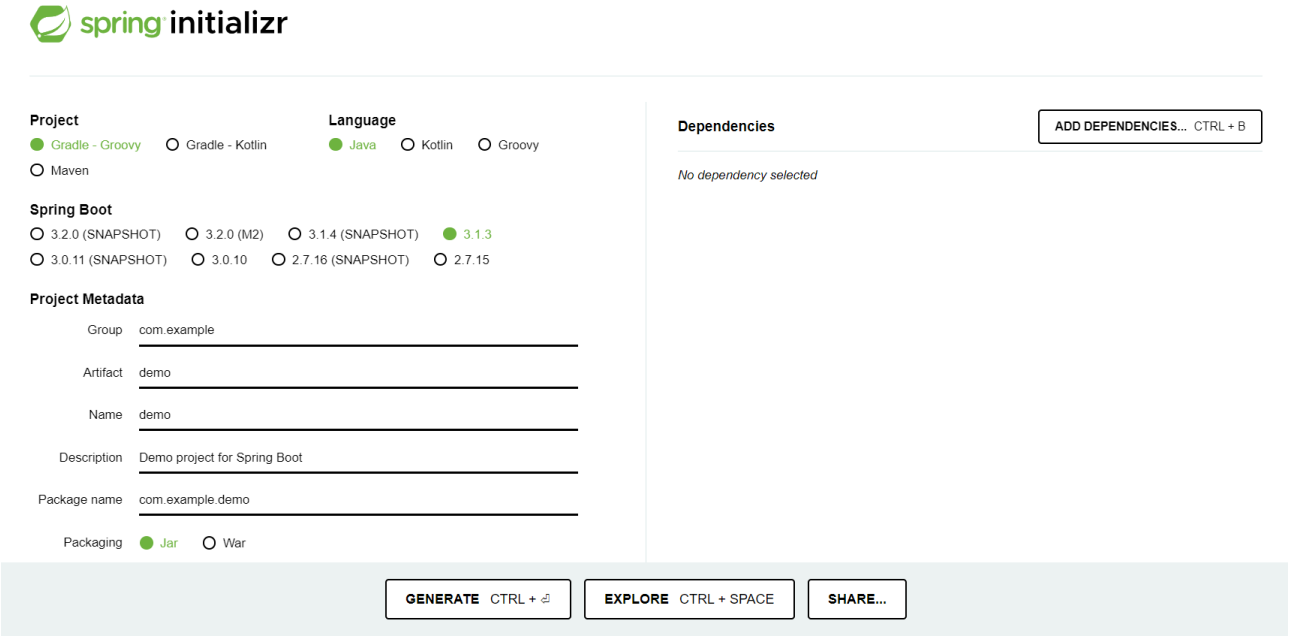



Fonte: (SPRING, 2023a).

O Spring *Initializr* é uma plataforma que disponibiliza uma API extensível para a

geração de projetos. Essa ferramenta oferece implementações para uma variedade de conceitos comuns (INITIALIZR, 2023). A interface do Spring Initializr é amigável e intuitiva e simplifica o processo de configuração de um projeto. A Figura 21 demonstra sua interface.

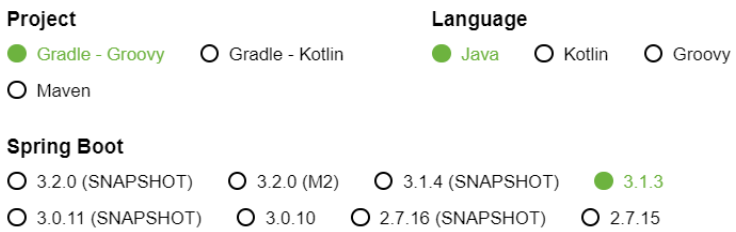
Figura 21 – Ferramenta Spring *Initializr*.




Fonte: (SPRING, 2023b).

A ferramenta permite que seja facilmente selecionado o tipo de sistema de construção, com opções como Maven, Gradle (Groovy) ou Gradle (Kotlin). Além disso, é possível especificar a linguagem de programação base para o projeto, com escolhas entre Java, Kotlin e Groovy e a versão do Spring Boot a ser utilizada no projeto. As especificações das características listadas acima são demonstradas na Figura 22.

Figura 22 – Especificação de dados na ferramenta Spring *Initializr*.

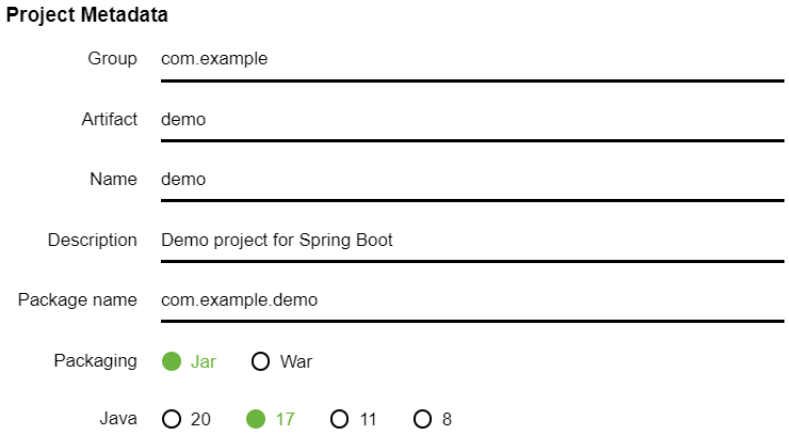



Fonte: (SPRING, 2023b).

Ademais, o Spring *Initializr* permite a definição de metadados essenciais para o projeto, como grupo, artefato, nome, descrição, tipo de empacotamento e a versão do Java desejada. Assim como, é possível escolher entre uma lista de dependências disponíveis para

adicionar ao projeto. Isso inclui dependências relacionadas a ferramentas de desenvolvimento, desenvolvimento *web*, segurança, configuração de banco de dados, mensageria e vários outros aspectos essenciais para o desenvolvimento de aplicativos *back-end* de forma abrangente e eficaz. A representação dessas informações presentes na ferramenta estão dispostas nas Figuras 23 e 24.

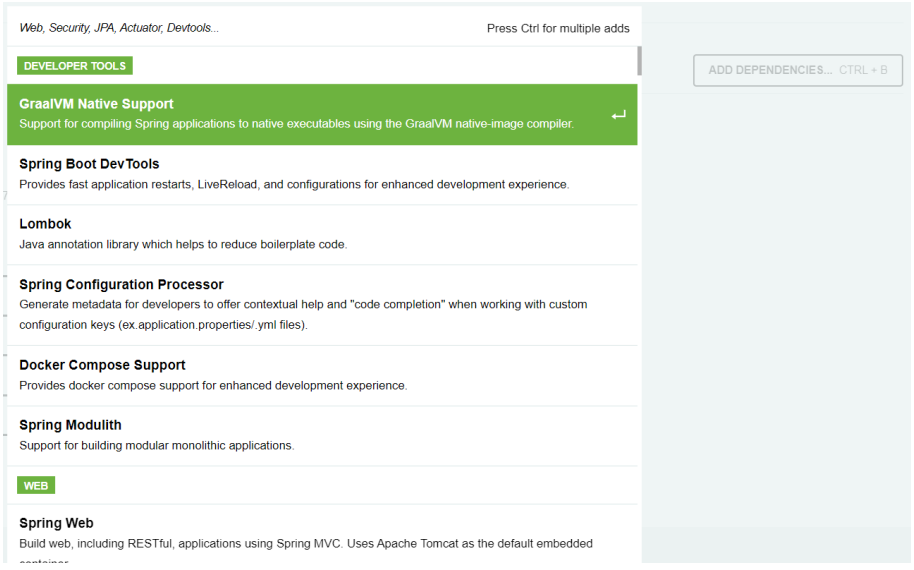
Figura 23 – Metadados do projeto na ferramenta Spring *Initializr*.




Fonte: (SPRING, 2023b).

A lista de dependências que podem ser injetadas em um projeto base utilizando Spring Boot são apresentadas em forma de modal. A Figura 24 demonstra isso.

Figura 24 – Dependências na ferramenta Spring *Initializr*.




Fonte: (SPRING, 2023b).

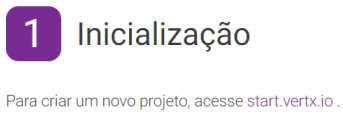
Após preencher os detalhes do projeto no inicializador do Spring Boot, é possível gerar o projeto base e abri-lo em um ambiente de desenvolvimento. O projeto é configurado automaticamente conforme a escolha do sistema de construção, seja Maven ou Gradle.

Dependendo das dependências adicionadas ao projeto base, é possível inicializá-lo imediatamente sem a necessidade de configurar propriedades adicionais, como a dependência de *web*, que fornece suporte para o desenvolvimento de uma aplicação *web*. No entanto, se ocorreu a inclusão de dependências que exigem configuração específica, como a configuração de dados como o *Spring Data JPA*, é necessário realizar algumas configurações básicas para garantir o funcionamento adequado do projeto e evitar erros inesperados.

#### *5.4.5.2 Vert.x*

O *framework* Vert.x apresenta em seu site uma seção de inicialização que contém um redirecionamento para o inicializador do próprio *framework*, o Vert.x *starter*. A Figura 25 demonstra a seção descrita acima.

Figura 25 – Apresentação da ferramenta de inicialização de projeto do Vert.x.

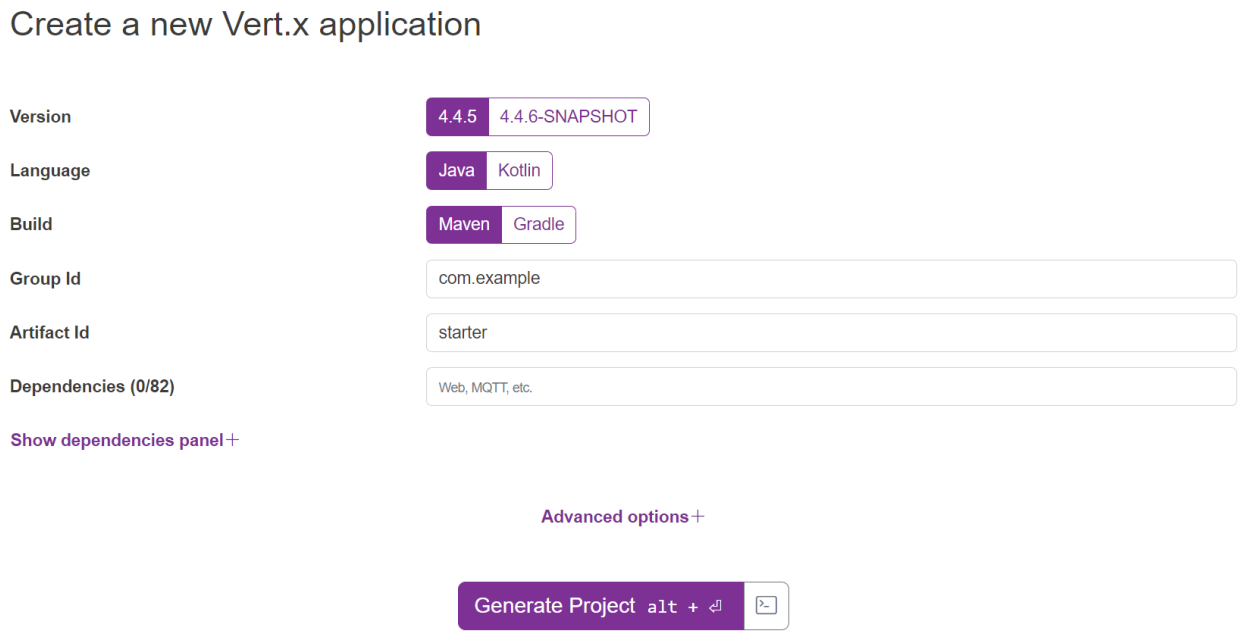



Fonte: (COMMUNITY, 2023).

Ao utilizar a ferramenta fornecida pelo próprio *framework*, para iniciar um novo projeto através dela, é possível escolher a versão do *framework*, a linguagem de programação incluindo Java e Kotlin e o método de construção, com opções como Maven e Gradle. Além disso, a ferramenta possibilita a personalização de características do projeto e apresenta um painel de dependências, permitindo a fácil inclusão de recursos para tarefas diversas, como desenvolvimento *web*, acesso a banco de dados, microserviços, testes, práticas de DevOps, entre outros. A Figura 26 apresenta a ferramenta de inicialização do Vert.x com suas funcionalidades.

Após definir as funcionalidades desejadas na ferramenta de inicialização do Vert.x, um novo projeto base é gerado de forma automática e configurado de acordo com as opções e dependências selecionadas. Este projeto é pronto para uso imediato, dependendo das dependências escolhidas. As dependências que não exigem configuração adicional permitem que o projeto possa ser aberto em um editor de código e seja facilmente executado. Após configurar a classe principal de acordo com as simples instruções disponibilizadas na própria documentação de inicialização do *framework* e por meio dos métodos de construção, sendo Maven ou Gradle.

Figura 26 – Interface da ferramenta de inicialização do Vert.x.




Fonte: (VERT.X, 2023c).

#### *5.4.5.3 Dropwizard*

O *framework* Dropwizard, em sua página inicial, oferece uma seção de introdução para iniciar o uso do *framework*.

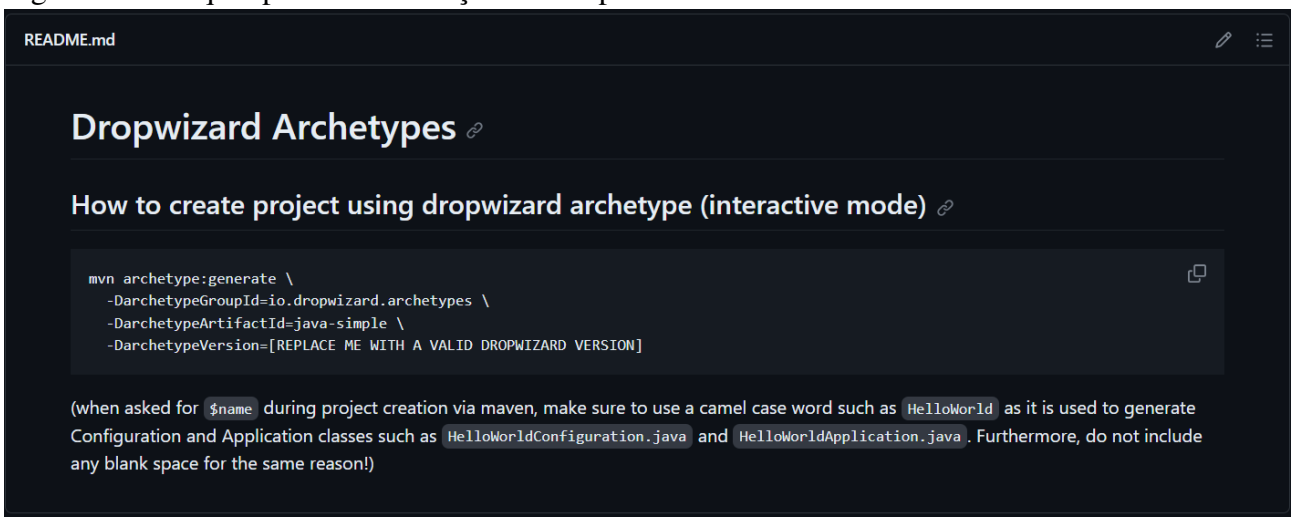
Embora não disponha de uma ferramenta interativa dinâmica, como nos casos do Spring Boot e Vert.x, o Dropwizard fornece informações para começar a utilizar o *framework* de maneira eficaz.

A documentação do *framework* recomenda o uso do arquétipo do Dropwizard, que direciona para uma página no GitHub. Este arquétipo permite a criação de um projeto base com o *framework* através de um comando disponibilizado em que é necessário adicionar a versão do Dropwizard desejada.

Durante o processo de criação, é aconselhável utilizar o gerenciador de dependências Maven, conforme recomendado pelo próprio *framework* em seu site. Além disso, ao definir o nome do projeto, é importante seguir a convenção *CamelCase*, por exemplo, "HelloWorld". Essa convenção é essencial, pois o Dropwizard gera classes de configuração e aplicação com base no nome do projeto. É necessário também não incluir espaços em branco no nome. A Figura 27 a seguir demonstra o arquétipo do Dropwizard em forma de um comando executável, bem como a apresentação de algumas das restrições explicadas acima.

Após a execução do comando Maven para o arquétipo, substituindo o campo obri-

Figura 27 – Arquétipo de inicialização do Dropwizard.




Fonte: (DROPWIZARD, 2023a).

gatório por uma versão válida do Dropwizard, é solicitado via terminal o fornecimento de metadados essenciais para o projeto base. Esses metadados incluem informações como *groupId*, *artifactId*, versão, pacote, descrição e outros detalhes importantes para configurar o projeto. A Figura 28 a seguir demonstra os metadados do projeto configurados após ser construído.

Figura 28 – Processo de construção com sucesso de um projeto base utilizando o *framework* Dropwizard.

```
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: java-simple:2.0.26
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: com.test
[INFO] Parameter: artifactId, Value: TestDropwizard
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: test
[INFO] Parameter: packageInPathFormat, Value: test
[INFO] Parameter: package, Value: test
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: name, Value: TestDropwizard
[INFO] Parameter: groupId, Value: com.test
[INFO] Parameter: description, Value: null
[INFO] Parameter: shaded, Value: true
[INFO] Parameter: artifactId, Value: TestDropwizard
[INFO] Project created from Archetype in dir: C:\Users\Suporte\repositorios-pessoal\TestDropwizard
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  33.949 s
[INFO] Finished at: 2023-09-25T13:39:36-03:00
[INFO] ------------------------------------------------------------------------
```

Fonte: Elaborado pelo autor.

Após a criação do projeto, é necessário navegar para o diretório do projeto recém-criado e então abri-lo em um ambiente de execução de código e seguir as instruções de execução via terminal gerado pelo próprio sistema.

## 5.5 Definição e detalhamento da aplicação prática

Para avaliar os critérios de aplicação que requerem implementação de código, é necessário realizar uma abordagem prática. Essa abordagem requer uma modelagem de negócios, bem como uma definição de requisitos e seus relacionamentos.

Com esse propósito em mente, se faz necessário desenvolver uma aplicação de teste em cada um dos *frameworks* abordados neste trabalho. O presente autor definiu que a aplicação utilizada para avaliar as particularidades práticas de cada *framework* é um simples sistema CRUD, modelo de sistema já utilizado em outros trabalhos acadêmicos de mesmo escopo, como os de (ALMEIDA, 2018) e (FERREIRA; ZUCHI, 2021).

O sistema CRUD foi planejado pelo presente autor como um catálogo de filmes, com o seu principal propósito voltado para o gerenciamento de filmes. Além de possibilitar operações de CRUD em registros de filmes, o sistema também estende essa funcionalidade para outras entidades relacionadas. Isso significa que os usuários podem criar, ler, atualizar e excluir informações não apenas sobre filmes, mas também sobre todas as entidades inter-relacionadas.

É relevante destacar que a aplicação é construída utilizando banco de dados relacional para cada um dos *frameworks*, para que os dados sejam modelados de uma forma que eles sejam percebidos pelo usuário como tabelas, ou mais formalmente relações.

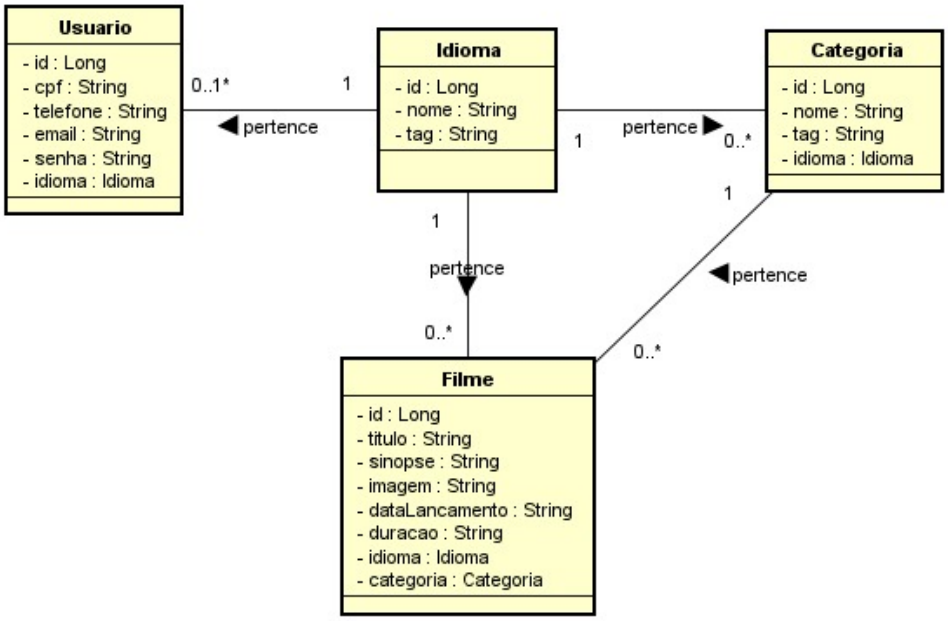
Para representar as entidades e seus relacionamentos é utilizado o software Astah, uma ferramenta eficaz e produtiva para o design e estrutura de software.

A Figura 29 apresenta as entidades da aplicação e seus respectivos relacionamentos. No contexto mostrado, um usuário possui atributos específicos e um idioma associado, estabelecendo uma relação onde zero ou vários usuários podem compartilhar o mesmo idioma. Essa mesma relação é aplicada entre a categoria e o idioma, assim como entre o filme e o idioma e categoria.

As aplicações desenvolvidas estão disponíveis em repositórios de acesso público [5].

[5]
- Spring Boot: https://github.com/guiseixas/crud-springboot-tcc
- Vert.x: https://github.com/guiseixas/crud-vertx-tcc
- Dropwizard: https://github.com/guiseixas/crud-dropwizard-tcc

Figura 29 – Diagrama de classes para a aplicação deste trabalho elaborado no software Astah.




Fonte: Elaborado pelo autor.

## 5.6 Avaliação dos critérios de aplicação para cada *framework*

Nesta etapa, são avaliados especificamente os critérios de aplicação para os *frameworks* abordados neste estudo: Spring Boot, Vert.x e Dropwizard.

### *5.6.1 Suporte a mapeamento objeto-relacional*

A aplicação de bancos de dados relacionais no desenvolvimento de aplicações *web back-end* é uma prática amplamente adotada. Essa escolha se fundamenta na estrutura relacional desses bancos, que oferecem eficiência no mapeamento de relacionamentos, assim como no armazenamento e recuperação de dados.

Para viabilizar essa integração, surgem abstrações que buscam simplificar a manipulação desses bancos, representando uma técnica que visa aproximar o paradigma de desenvolvimento de aplicações orientadas a objetos ao paradigma do banco de dados relacional. A aplicação da técnica de mapeamento objeto-relacional ocorre por meio de um mapeador objeto-relacional, geralmente implementado como uma biblioteca ou *framework*, facilitando o mapeamento e a utilização do banco de dados (TREINAWEB, 2020).

Neste critério, é analisado se o *framework* oferece suporte à utilização de uma ferramenta de mapeamento objeto-relacional e como essa integração é realizada. Esta avaliação é

especialmente relevante, uma vez que as aplicações desenvolvidas neste trabalho estão fundamentadas em uma base de dados relacional, modelo de base de dados mais utilizado na maioria das aplicações comerciais conforme o ranking de popularidade sobre sistemas de banco de dados estabelecido no DB-Engines (2023).

#### *5.6.1.1 Spring Boot*

Conforme destacado na documentação do *framework* Spring Boot, é possível empregar diversas ferramentas de integração objeto-relacional para estratégias de transação, gerenciamento de recursos, entre outros aspectos. Destaca-se que o Hibernate [6] é a escolha mais comum e amplamente utilizada nas aplicações. Sendo o *framework* escolhido para a abstração de banco de dados na aplicação desenvolvida com *Spring Boot*.

Após a inclusão de dependências necessárias para integrar o Hibernate: *web*, *JPA*, que descreve uma interface para o Hibernate e a dependência de banco de dados relacional, descritas na documentação do *framework*, o projeto já está configurado para utilizar o Hibernate.

Concluída essa fase, torna-se necessário configurar as entidades da aplicação. Estas devem possuir anotações e recursos específicos, possibilitando o mapeamento preciso dessas entidades como tabelas no banco de dados, inclusive seus atributos representados como colunas.

#### *5.6.1.2 Vert.x*

Na documentação do Vert.x, é destacada uma ferramenta de integração objeto-relacional, o Hibernate Reactive. O Hibernate Reactive funciona como uma versão estendida do Hibernate comum, ele destina-se ao uso em um ambiente de programação reativo, como o Vert.x, em que as operações de persistência são orquestradas via um fluxo reativo, em vez da invocação direta de funções síncronas em código Java processual (HIBERNATE, 2023). O Hibernate Reactive foi utilizado na aplicação desenvolvida com Vert.x.

Como já definido antes, o Vert.x trabalha com *verticles*, que são estruturas próprias definidas para a execução da aplicação Vert.x, portanto, para utilizar o Hibernate Reactive, além de adicionar dependências necessárias conforme a documentação, é necessário configurá-lo no *verticle* principal do projeto que herda recursos de estrutura do Vert.x e apresenta o método

[6] O Hibernate é um *framework* de mapeamento objeto-relacional praticado por grande parte dos desenvolvedores *back-end*. De modo que a manipulação de banco de dados, que antes era manual, ganhou mais agilidade com os *frameworks* como o Hibernate (COODESH, 2023a).

*asyncStart*, para realizar operações de inicialização assíncronas (não bloqueantes para outros processos) quando o *verticle* é implantado.

Durante essa configuração, são realizadas operações simultâneas, incluindo a preparação do banco de dados relacional, o estabelecimento de rotas e a especificação da localização do banco. Todo esse processo é encapsulado em uma operação assíncrona, culminando na execução do Hibernate Reactive, conforme demonstra a Figura 30 a seguir, que ilustra o processo de configuração do Hibernate Reactive no contexto assíncrono do *verticle* principal do Vert.x.

Figura 30 – Configuração do Hibernate Reactive no Vert.x.


Fonte: Elaborado pelo autor.

Após isso, o Hibernate Reactive já está configurado corretamente, agora, as entidades precisam ser configuradas com recursos específicos para poderem ser reconhecidas como tabelas no banco de dados e seus atributos como colunas dessas tabelas.

#### *5.6.1.3 Dropwizard*

O Dropwizard provê em sua documentação, uma seção única que destaca a compatibilidade com Hibernate. O módulo Dropwizard Hibernate fornece acesso gerenciado ao Hibernate, um mapeador de relação de objetos relacional padrão da indústria (DROPWIZARD, 2023b). Portanto, o Hibernate foi utilizado como mapeamento na aplicação desenvolvida com Dropwizard.

É necessário, inicialmente, configurar as dependências do projeto Dropwizard e com os módulos fornecidos na documentação, além de configurar recursos extras em um arquivo de extensão *yml*, para que o projeto esteja configurado para utilizar o Hibernate, consoante a documentação do *framework*.

Posteriormente, é crucial realizar a configuração das classes que representam as entidades do projeto. Essa etapa visa permitir o mapeamento dessas classes como tabelas e a representação de seus atributos como colunas no banco de dados.

#### *5.6.1.4 Mapeamento objeto-relacional para uma entidade*

É essencial configurar todas as entidades do projeto com anotações específicas, para que as classes Java sejam devidamente mapeadas para tabelas no banco de dados a partir do uso da anotação *Entity*. Para que assim, o Hibernate possa realizar operações de manipulação de dados por meio das operações CRUD. Os três *frameworks* utilizados neste trabalho empregam o Hibernate, sendo que o Vert.x, embora utilize uma versão estendida do Hibernate para programação reativa em que o *framework* está fundamentado, compartilha um mapeamento de tabelas idêntico ao utilizado pelos outros *frameworks* envolvidos, como o Spring Boot e o Dropwizard.

Além disso, é preciso definir a chave primária da classe e sua estratégia de geração. Isso assegura a presença de uma chave primária no banco de dados. Seguidamente, é necessário especificar os atributos que funcionarão como colunas da tabela da classe, podendo adicionar configurações adicionais, como restrições de nulidade, limites, entre outras, conforme necessário. Na Figura 31 a seguir, é apresentado o padrão de mapeamento necessário para o mapeamento de uma entidade no Hibernate, da entidade Filme neste caso, em que a classe é anotada com a configuração de entidade para definir que é uma tabela, o atributo identificador é definido como chave primária única com estratégia de incremento automático e os atributos são definidos como colunas. Desse modo, assim que a aplicação for executada, no banco de dados uma tabela chamada "filme"é criada, com colunas definidas pelos atributos da classe Java "Filme", por conta do Hibernate.

O mesmo processo foi realizado para definir as tabelas das entidades restantes da aplicação desenvolvida nos *frameworks*.

É crucial destacar que o Hibernate também oferece abstração para a execução de consultas SQL realizadas no banco de dados utilizando sua própria linguagem HQL, que simplifica o processo utilizando os nomes diretos das entidades para realização de consultas, sendo totalmente orientado a objetos e proporcionando maior simplificação e eficiência na realização de consultas específicas.

A Figura 32, de caráter exemplificativo no contexto do *framework* Spring Boot, delineia uma consulta personalizada para a entidade "filme"do projeto. Essa consulta utiliza a linguagem HQL, definida pelo Hibernate, para realizar uma seleção com filtro baseado no título do filme. O título é comparado com o parâmetro passado à função, retornando os filmes correspondentes cujo título contenha a *string* enviada.

Figura 31 – Mapeamento da classe Filme utilizando o Hibernate integrado aos *frameworks* Spring Boot, Vert.x e Dropwizard.

```java
@Entity
public class Filme {

    @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;

    //Atributos e suas configurações
    @Column(nullable = false)
    @NotBlank
    @NotEmpty
    private String titulo;

    @Column(nullable = false)
```

Fonte: Elaborado pelo autor.

Figura 32 – Exemplo de consulta HQL para buscar uma lista de filmes com base em uma *string* título no Spring Boot.

```java
1 usage  guilherme seixas
@Query("SELECT f FROM Filme f WHERE LOWER(f.titulo) LIKE LOWER(CONCAT('%',:tituloFilme,'%')) ")
List<Filme> searchName(String tituloFilme);
```

Fonte: Elaborado pelo autor.

### *5.6.2 Suporte à arquitetura de software REST*

A arquitetura de software REST configura-se como uma das mais importantes para o desenvolvimento de software para *web*. REST é o acrônimo de *Representational State Transfer* (Transferência de Estado Representativo) e pode ser definido como um conjunto de boas práticas de uso do protocolo HTTP para expor *web services*, ou seja, serviços responsáveis por executar determinada lógica ou regra de negócio, geralmente utilizados para integração entre aplicações.

Neste critério, é avaliado como cada *framework* em questão oferece suporte à arquitetura de software REST, o que envolve a utilização de protocolos como o HTTP para o gerenciamento de requisições e respostas, bem como formatos como o JSON para a manipulação de recursos.

#### *5.6.2.1 Spring Boot*

Conforme destacado na documentação do Spring Boot, REST rapidamente emergiu como o padrão de fato para o desenvolvimento de serviços *web*, devido à sua facilidade de construção e consumo (SPRING, 2023a).

No Spring Boot, a criação de um projeto com arquitetura REST é simplificada por meio de convenções, anotações e padrões fornecidos pelo próprio *framework*, a partir de bibliotecas de serialização e desserialização, a exemplo o Jackson, que funcionam internamente para manipular formatos como JSON e disponibilizando anotações que simplificam a manipulação das solicitações e respostas HTTP.

Por exemplo, ao definir um controlador no Spring Boot, é possível manipular solicitações e respostas usando protocolo HTTP e dados em formato JSON. No contexto da aplicação de catálogo de filmes desenvolvida, o controlador de filmes, demonstrado na Figura 33 e mapeado na rota "/filme", possibilita o envio de requisições por meio dos diferentes verbos HTTP para realizar operações CRUD.

Figura 33 – Controlador de filme na aplicação de catálogo de filmes no Spring Boot.

```
@Controller
@RequestMapping(value = "/filme")
public class FilmeController {
```

Fonte: Elaborado pelo autor.

Na Figura 34 abaixo, é apresentado um método configurado no controlador de filmes, com a responsabilidade de receber uma requisição HTTP para buscar um registro de filme na base de dados com base em seu *id* e retornar um JSON do objeto, caso ele exista.

Figura 34 – Método *GET* responsável por realizar a operação CRUD de encontrar um filme por *id* no Spring Boot.

```
guilherme seixas
@GetMapping("/filmeById/{id}")
public ResponseEntity<?> getFilmeById(@PathVariable Long id) {
    try{
        Optional<Filme> filme = filmeService.findById(id);
        if(filme.isEmpty()){
            return new ResponseEntity<>( body: "Não existe filme com esse id.", HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<>(filme, HttpStatus.OK);
    }catch (Exception e){
        return new ResponseEntity<>(e.getMessage(), HttpStatus.BAD_REQUEST);
    }
}
```

Fonte: Elaborado pelo autor.

#### *5.6.2.2 Vert.x*

O framework Vert.x oferece uma dependência injetável para o desenvolvimento de aplicativos *web* eficientes e escaláveis, conhecida como *Vert.x Web*. Com essa estrutura, o Vert.x disponibiliza um conjunto de funcionalidades de baixo nível para lidar com protocolo HTTP e manipuladores que recebem e retornam objetos em formato JSON, também utilizando o Jackson para serialização e desserialização de classes e objetos (VERT.X, 2023a).

Conforme descrito pelo próprio framework, a dependência *Vert.x Web* simplifica consideravelmente a implementação de Interfaces de Programação de Aplicação (APIs) no padrão REST, uma vez que, essencialmente, encaminha as URLs para os manipuladores apropriados (VERT.X, 2023a).

Na Figura 35 a seguir, tem-se o mapeamento de uma rota para buscar um registro de filme na base de dados com base em seu *id* e retornar um JSON do objeto caso ele exista, definida na classe principal do *verticle* do Vert.x. Isso acontece mediante uma instância de roteamento, que funciona como um despachador essencial no encaminhamento de requisições HTTP para os manipuladores, que processam as solicitações, neste caso, a função *getFilmeById*.

Figura 35 – Mapeamento de uma rota de verbo HTTP *GET* para buscar um filme por seu *id* no Vert.x.

```
router.get("/filmeById/:id").respond(this::getFilmeById);
```

Fonte: Elaborado pelo autor.

A função encarregada de buscar um filme pelo seu *id* recebe como parâmetro um objeto *RoutingContext*, que representa o contexto de uma solicitação HTTP. Na situação mencionada, ao utilizar o método *GET* com base no identificador (*id*), o parâmetro enviado na URL deve ser o número correspondente ao identificador do filme. Isso permite que a aplicação encontre a entidade filme correspondente ao identificador enviado na requisição e retorne as informações no formato de um objeto JSON, a partir de um processo interno de desserialização. A função exemplificada acima está demonstrada na Figura 36.

#### *5.6.2.3 Dropwizard*

Com base na documentação oficial do framework Dropwizard, o Dropwizard é uma estrutura eficiente e de fácil utilização projetada para o desenvolvimento *web* com uma arquitetura de software voltada para REST de alto desempenho.

Figura 36 – Função para buscar um filme por seu *id* utilizando o padrão REST no Vert.x.

```
private Uni<Filme> getFilmeById(RoutingContext ctx) {
  long id = Long.parseLong(ctx.pathParam( name: "id"));
  return emf.withSession(session -> session
      .find(Filme.class, id))
    .onItem().ifNull().continueWith(Filme::new);
}
```

Fonte: Elaborado pelo autor.

Para isso, é necessário configurar os recursos do *framework* interno Jersey, que desempenham um papel fundamental na definição da estrutura fundamental de uma aplicação REST no Dropwizard. Esses recursos contêm implementações que especificam como a aplicação deve produzir e consumir dados no formato JSON e como deve lidar com solicitações HTTP. Segundo Medium (2019), esses recursos são a essência de um aplicativo Dropwizard, em que cada classe de recurso está vinculada a um modelo de URL específico. A Figura 37 a seguir especifica uma classe de recurso descrita acima para a entidade de filme da aplicação desenvolvida neste trabalho, mapeada na rota "/filme", e produz e consome dados em formato JSON.

Figura 37 – Classe de recurso para a entidade filme mapeada no Dropwizard.

```
@Path("/filme")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public class FilmeResource {
```

Fonte: Elaborado pelo autor.

Na classe ilustrada na Figura 37 acima, estão definidas funções que correspondem às operações CRUD, sendo cada uma delas anotada com o verbo HTTP apropriado e a rota URL pela qual são acessadas. Por exemplo, a função responsável por buscar um filme com base no seu identificador é anotada com o verbo HTTP *GET* e espera receber um identificador único correspondente ao filme na requisição. Na função, uma instância de filme é buscada com base no seu identificador e, se existir, os dados são retornados na resposta da requisição. A Figura 38 a seguir, apresenta a função descrita.

#### *5.6.2.4 Retorno JSON*

Ao fazer uma requisição HTTP do tipo *GET* na rota especificada, *"/filmeById/id"*, em cada um dos *frameworks* deste trabalho, um objeto JSON correspondente a entidade filme

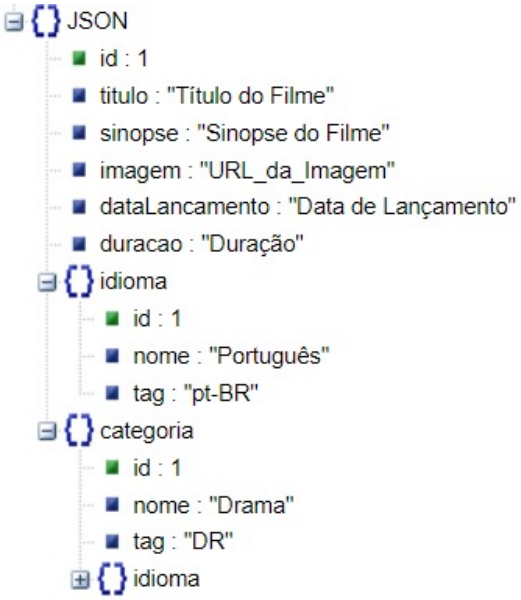
Figura 38 – Função para buscar um filme por seu *id* utilizando o padrão REST no Dropwizard.

```
@GET
@Path("/filmeById/{id}")
public Response getFilmeById(@PathParam("id") Long id) {
    Filme filme = filmeDAO.getFilmeById(id);
    if(filme == null) {
        throw new NotFoundException("Filme nao encontrado");
    }
    return Response.ok(filme).build();
}
```

Fonte: Elaborado pelo autor.

é retornado, neste caso, um filme cujo identificador é 1 é retornado, conforme ilustrado na Figura 39 e considerando que este mesmo objeto filme foi persistido na base igualmente nos três *frameworks*.

Figura 39 – Objeto JSON do tipo filme retornando da aplicação desenvolvida neste trabalho através do método encontrar um filme por *id* nos *frameworks* Spring Boot, Vert.x e Dropwizard.




Fonte: Elaborado pelo autor.

## 5.7 Síntese dos resultados avaliados

Nesta seção, são apresentados de forma sintetizada os resultados relativos aos critérios de contexto e de aplicação avaliados para os *frameworks* abordados neste trabalho. A síntese dos resultados desta análise está disposta de maneira compacta no Quadro 4 a seguir, que destaca algumas conclusões essenciais.

Quadro 4 – Resultados sintetizados para os critérios de contexto e de aplicação avaliados nos *frameworks* deste trabalho: Spring Boot, Vert.x e Dropwizard.

| Critérios | Spring Boot | Vert.x | Dropwizard |
|---|---|---|---|
| Popularidade | Alto interesse | Médio interesse | Baixo interesse |
| Atividade da comunidade | Alta atividade recente | Média atividade recente | Média atividade recente |
| Qualidade da documentação | Tutoriais e guias | Tutoriais e guias | Tutoriais e guias |
| Avaliação de artigos publicados | Médio número de publicações | Baixo número de publicações | Baixo número de publicações |
| Suporte à inicialização de projeto | Ferramenta via *web* | Ferramenta via *web* | Arquétipo via terminal |
| Suporte a mapeamento objeto-relacional | Hibernate | Hibernate Reactive | Hibernate |
| Suporte à arquitetura de software REST | Anotações | Roteamento e *handlers* | Anotações |

Fonte: Elaborado pelo autor.

Com base nos resultados apresentados no Quadro 4 e nas análises dos *frameworks web back-end*, destacam-se algumas conclusões essenciais.

No quesito de popularidade, o Spring Boot evidencia um notável interesse, possuindo picos significativos com estabilidade média acima de 60 e mantendo valores individuais altos. Por outro lado, o Vert.x demonstra um interesse moderado, por apresentar média próxima de 50 e valores individuais moderados de forma geral. O Dropwizard, apresenta um menor interesse, com uma média registrada de aproximadamente 30 e apresentando picos individuais mais baixos que os demais. Esses números revelam uma maior popularidade do Spring Boot, uma posição intermediária para o Vert.x e uma presença menor no caso do Dropwizard.

Na atividade da comunidade, o Spring Boot se destaca com uma alta atividade recente, evidenciada por questionamentos e interações nos fóruns de programação nos últimos minutos. Por outro lado, o Vert.x e o Dropwizard mostram uma atividade média recente, caracterizada pela disponibilidade de questionamentos e interações apenas nos últimos dias, com uma exceção de últimas horas para as *issues* do Dropwizard. Essa dinâmica sugere que o Vert.x e o Dropwizard são *frameworks* emergentes, enquanto o Spring Boot mantém uma posição consolidada, com interações constantes da comunidade.

Quanto à qualidade da documentação, todos os *frameworks* se destacam, oferecendo tutoriais, guias e suporte em diferentes níveis de complexidade. Essa abordagem facilita o aprendizado e a implementação prática, contribuindo para a adoção eficaz dos *frameworks*.

A análise dos artigos publicados ressalta o Spring Boot em termos de quantidade total de artigos e de citações, evidenciando sua média presença na literatura acadêmica. Isso é notável quando se considera que o número máximo de resultados na análise de dados foi mil. Por outro lado, o Vert.x e o Dropwizard demonstram uma menor presença na literatura acadêmica, apresentando números de resultados muito inferiores ao limite máximo estabelecido na análise. Essa distribuição reflete uma maior relevância acadêmica atribuída ao Spring Boot em relação ao Vert.x e o Dropwizard.

No quesito suporte à inicialização de projeto, o Spring Boot e o Vert.x se destacam ao oferecerem ferramentas de inicialização via *web* com interfaces amigáveis e usabilidade intuitiva. Em contrapartida, o Dropwizard utiliza um arquétipo via terminal menos prático no ponto de vista de usabilidade.

O suporte ao mapeamento objeto-relacional é fornecido pelo Hibernate em todos os *frameworks*, sendo uma escolha comum e robusta, conforme suas próprias documentações. Vale destacar que, no caso específico do Vert.x, a implementação do Hibernate utilizada é a do Hibernate Reactive, sendo uma implementação própria para oferecer suporte a operações assíncronas e reativas.

Quanto ao suporte à arquitetura de software REST, todos os *frameworks* oferecem suporte. O Spring Boot e o Dropwizard utilizam anotações para o mapeamento de controladores de requisições e definição de métodos *HTTP*, enquanto o Vert.x adota configuração via roteamento e *handlers*.

Com a síntese elaborada, destaca-se o Spring Boot como um *framework* consolidado, enquanto o Vert.x e o Dropwizard surgem como opções emergentes no cenário atual. É crucial ressaltar que todos apresentam utilidade, cada um com suas particulares contribuições para o mercado e para o desenvolvimento de software.

# 6 CONCLUSÃO

Neste trabalho, o principal objetivo consistiu na avaliação dos *frameworks web back-end* Spring Boot, Vert.x e Dropwizard, escritos em Java e reconhecidos como populares no cenário atual da programação. A seleção destas tecnologias foi conduzida de forma criteriosa, pautada em uma análise de trabalhos acadêmicos que exploram o escopo de avaliação de *frameworks*. Durante esse processo, identificou-se que a métrica de popularidade, era amplamente utilizada na literatura acadêmica. Assim, além de adotar essa métrica, especificou-se que o número de estrelas presentes nos repositórios oficiais dos *frameworks* no GitHub, seria o critério significativo para a escolha destas tecnologias, proporcionando uma base relevante e quantificável para a seleção.

Para a avaliação dos *frameworks* selecionados, adotou-se uma abordagem abrangente, incorporando critérios identificados por meio de uma revisão bibliográfica de trabalhos correlatos, que serviram de auxílio e exploraram tanto aspectos teóricos quanto práticos na análise de *frameworks*. Neste trabalho em específico, esses critérios foram categorizados como critérios de contexto e critérios de aplicação. Os critérios de contexto fundamentaram-se no ambiente em que o *framework* está inserido e em suas características de forma geral no contexto do desenvolvimento *web*, englobando popularidade, atividade da comunidade, qualidade da documentação, artigos publicados e suporte à inicialização de projeto. Enquanto os critérios práticos buscaram identificar padrões e analisar como ocorre o suporte prático oferecido por cada *framework*, sendo eles, mapeamento objeto-relacional de dados e suporte à arquitetura de software REST. Essa abordagem combinada proporcionou uma avaliação abrangente, considerando tanto a adequação ao contexto quanto a eficácia prática dos *frameworks* em questão.

A expectativa desta avaliação de *frameworks*, pautada em diversos critérios predefinidos, é proporcionar aos desenvolvedores de software uma compreensão mais aprofundada sobre os *frameworks* abordados, abrangendo tanto o aspecto teórico quanto a praticidade, que considera padrões e convenções de código no desenvolvimento de software. Além disso, visa possibilitar uma compreensão das particularidades distintas de cada *framework*. Este trabalho pretende, assim, servir como um guia útil, oferecendo suporte e potencialmente induzindo ao uso de uma dessas tecnologias no desenvolvimento de software *web back-end* com Java.

Os caminhos para pesquisas futuras, derivados das conclusões deste estudo, incluem:

1. Ampliar a análise para outros *frameworks* em Java, considerando fontes de dados além do código aberto no GitHub, como repositórios privados e dados de uso em projetos

comerciais.

2. Estender a investigação para além do escopo atual, analisando *frameworks* disponíveis em outras linguagens de programação como PHP e Python, proporcionando uma visão mais abrangente e comparativa do cenário tecnológico.
3. Investigar novos critérios de avaliação pertinentes na literatura, que vão além dos aspectos já considerados, buscando enriquecer a compreensão das nuances específicas de cada *framework*. Essa exploração engloba dimensões como desempenho, escalabilidade e o suporte fornecido para testes unitários ou automatizados, visando uma análise mais abrangente e aprofundada das capacidades e características de cada ferramenta.

# REFERÊNCIAS

ALMEIDA, F. E. V. d. **Um Comparativo entre Frameworks JavaScript para o Desenvolvimento de Aplicações Front-End**. 2018. 42 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Software) - Universidade Federal do Ceará, Campus de Quixadá, Quixadá, 2018. Disponível em: https://repositorio.ufc.br/bitstream/riufc/39459/1/2018_tcc_fevalmeida.pdf. Acesso em: 10 maio 2023.

BERNERS-LEE, T. Information management: A proposal. **CERN**, I, p. 1–21, May 1990.

BOUCKAERT, S.; GERWEN, J.; MOERMAN, I.; PHILLIPS, S. C.; WILANDER, J. **Benchmarking Computers and Computer Networks**. [*S. l.*], 2010.

BRANAS, R. **AngularJS Essentials**: Design and Construct Reusable, Maintainable, and Modular Web Applications with AngularJS. Birmingham, United Kingdom: Packt Publishing, 2014.

CITRUS7. **O Que é Front-End e Back-End?** 2017. Disponível em: https://citrus7.com.br/artigo/o-que-e-front-end-e-back-end/. Acesso em: 16 maio 2023.

COMMUNITY, E. V. **Vert.x Get Started Guide**. 2023. Disponível em: https://vertx.io/get-started/. Acesso em: 18 set. 2023.

COODESH. **O que é Hibernate?** 2023. Disponível em: https://coodesh.com/blog/dicionario/o-que-e-hibernate/. Acesso em: 19 nov. 2023.

COODESH. **O que é JVM**: Máquina Virtual do Java. 2023. Disponível em: https://coodesh.com/blog/dicionario/o-que-e-jvm/#:~:text=JVM%3A%20m%C3%A1quina%20virtual%20do%20Java,aplicativos%20Java%20em%20seu%20computador. Acesso em: 03 nov. 2023.

DB-ENGINES. **Db-Engines**. 2023. Disponível em: https://db-engines.com/en/ranking. Acesso em: 28 set. 2023.

DROPWIZARD. **Dropwizard Getting Started**. 2011. Disponível em: https://dropwizard.github.io/dropwizard/0.6.2/getting-started.html. Acesso em: 03 set. 2023.

DROPWIZARD. **Dropwizard Archetypes**. 2023. Disponível em: https://github.com/dropwizard/dropwizard/tree/master/dropwizard-archetypes. Acesso em: 25 set. 2023.

DROPWIZARD. **Dropwizard Getting Started**. 2023. Disponível em: https://www.dropwizard.io/en/release-4.0.x/getting-started.html. Acesso em: 03 nov. 2023.

EIS, D. **Guia Front-End - O Caminho das Pedras para ser um dev Front-End**. [S.l.]: Casa do Código, 2015. I.

FERREIRA, H. K.; ZUCHI. Análise Comparativa entre Frameworks Frontend Baseados em JavaScript para Aplicações Web. 2021. 118-134 p. **Interfacetecnológica**: Revista Eletrônica do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo. Disponível em: https://revista.fatectq.edu.br/interfacetecnologica/article/view/502/302. Acesso em: 16 maio 2023.

FLUFFY. **Introduction to client-server architecture (frontend backend)**. 2019. Disponível em: https://fluffy.es/introduction-to-client-server/. Acesso em: 20 maio 2023.

FRANCO, R. S. T. **Estudo comparativo entre frameworks java para desenvolvimento de aplicações web**: JSF 2.0, Grails e Spring web MVC. 90 p. Dissertação (Monografia de especialização) – Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná, Curitiba, 2011.

GHIMIRE, D. **Comparative study on Python web frameworks**: Flask and Django. 2020. Bachelor's Thesis. Disponível em: https://www.theseus.fi/bitstream/handle/10024/339796/Ghimire_Devndra.pdf?sequence=2&isAllowed=y. Acesso em: 16 maio 2023.

Github, Inc. **GitHub Documentation**. 2023. Disponível em: https://docs.github.com/en. Acesso em: 03 set. 2023.

GOOGLE. **Google Trends**. 2023. Disponível em: https://trends.google.com.br/trends/explore. Acesso em: 11 set. 2023.

HARZING, A.-W. **Publish or Perish**. 2023. Disponível em: https://harzing.com/resources/publish-or-perish. Acesso em: 01 set. 2023.

HIBERNATE. **Hibernate Reactive**. 2023. Disponível em: https://hibernate.org/reactive/. Acesso em: 19 nov. 2023.

HUBSPOT. **O Que é Front-End e Back-End?** 2017. Disponível em: https://br.hubspot.com/blog/marketing/evolucao-web#:~:text=Proposta%20pelo%20f%C3%ADsico%20e%20cientista,V%C3%ADdeos%20eram%20raros. Acesso em: 16 jun. 2023.

INITIALIZR, S. **Spring Initializr**. 2023. Disponível em: https://github.com/spring-io/initializr/. Acesso em: 10 set. 2023.

LIMA, G. d. **Representação, recuperação e acesso da informação**: a evolução da Biblioteca 1.0 à Biblioteca 3.0. 2018. Documento não publicado. Disponível em: https://repositorio.ufmg.br/bitstream/1843/41357/2/Representa%C3%A7%C3%A3o%20recupera%C3%A7%C3%A3o%20e%20acesso%20da%20informa%C3%A7%C3%A3o_a%20evolu%C3%A7%C3%A3o%20da%20Biblioteca%201_0%20%C3%A0%20Biblioteca%203_0.%C2%A0.pdf. Acesso em: 10 maio 2023.

MATTSON, M. **Object-Oriented Frameworks**: A Survey of Methodological Issues. 128 p. Tese (Doutorado) – Universidade de Lund, Sweden, 1996.

MCFARLAND, D. S. **JavaScript & jQuery**: The Missing Manual. 3rd. ed. Sebastopol, United States of America: O'Reilly, 2014.

MEDIUM. **How to Design RESTful Web Services with Dropwizard**. 2019. Disponível em: https://medium.com/swlh/how-to-design-restful-web-services-with-dropwizard-d5681a127cba. Acesso em: 09 nov. 2023.

MOREIRA, D. D. R. **Um estudo da tecnologia Web 2.0**. 1-101 p. Monografia, 2009.

OLIVEIRA, C. I. **Uma Comparação Entre Micro Frameworks Web Para O Desenvolvimento De Aplicações Back-end Em Java**. 2019. 66 f. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Universidade Federal do Ceará, Campus de Quixadá, Quixadá, 2019. Disponível em: https://repositorio.ufc.br/bitstream/riufc/49763/1/2019_tcc_ideocosta.pdf. Acesso em: 10 maio 2023.

O'REILLY, T. **What Is Web 2.0**: Design Patterns and Business Models for the Next Generation of Software. 2005. Disponível em: https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html. Acesso em: 16 jun. 2023.

SCHMIDT, D. C.; STAL, M.; ROHNERT, H.; BUSCHMANN, F. **Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects**. [S.l.]: John Wiley & Sons, 2013. v. 2.

SILVA, M. F. **Utilização dos modelos full-stack framework e micro-framework para o desenvolvimento de aplicações web escaláveis em linguagem PHP**. Caratinga: [*S. n.*], 2016. 1–77 p. Monografia de Graduação, Faculdade de Ciência da Computação das Faculdades Integradas de Caratinga.

SOMMERVILLE, I. **Engenharia de Software**. 9° ed.. ed. [S.l.]: Pearson, 2011. I.

SPRING. **Spring Boot**. 2023. Disponível em: https://spring.io/projects/spring-boot. Acesso em: 25 set. 2023.

SPRING. **Spring Initializer**. 2023. Disponível em: https://start.spring.io/. Acesso em: 25 set. 2023.

TREINAWEB. **O que é ORM?** 2020. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-orm. Acesso em: 19 nov. 2023.

VERT.X. **Vert.x**. 2023. Disponível em: https://vertx.io/. Acesso em: 03 nov. 2023.

VERT.X. **Vert.x Docs**. 2023. Disponível em: https://vertx.io/docs/apidocs/. Acesso em: 03 nov. 2023.

VERT.X. **Vert.x Starter**. 2023. Disponível em: https://start.vertx.io/. Acesso em: 18 set. 2023.

WICAKSONO, A. *et al.* Design and implementation of information means sma 3 madiun based on codeigniter framework. In: IEEE. **Proceedings of the 2016 International Conference on Science in Information Technology (ICSITech)**. [*S. l.*], 2016.

XAVIER, L.; COELHO, J.; S., L. Um estudo empírico sobre critérios de seleção de repositórios github. In: **Conferência Brasileira de Software (CBSOFT) - VI Workshop de Visualização, Manutenção e Evolução de Software (VEM)**. São Carlos: [s.n.], 2018. p. 8.