INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAJAZEIRAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

STACK DEVELOPMENT EDUCATION - UMA EXTENSÃO PARA O EDITOR DE CÓDIGOS *VS CODE* QUE ARMAZENA DADOS DA BASE DE CONHECIMENTO STACK OVERFLOW

DANIEL DANTAS CATARINA

Cajazeiras
2022

DANIEL DANTAS CATARINA

STACK DEVELOPMENT EDUCATION - UMA EXTENSÃO PARA O EDITOR DE CÓDIGOS *VS CODE* QUE ARMAZENA DADOS DA BASE DE CONHECIMENTO STACK OVERFLOW

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Me. Francisco Paulo de Freitas Neto.

Cajazeiras
2022

IFPB / Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Suellen Conceição Ribeiro CRB-2218

C357s Catarina, Daniel Dantas

Stack development education: uma extensão para o editor de códigos vs *code* que armazena dados da base de conhecimento *stack overflow* / Daniel Dantas Catarina. – Cajazeiras/PB: IFPB, 2022.

46f.:il.

Trabalho de Conclusão de Curso (Tecnólogo Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba-IFPB, Campus Cajazeiras. Cajazeiras, 2022. Orientador(a): Prof. Me. Francisco Lopes Lavor Neto.

1. Tecnologia da Informação. 2. Códigos. 3. Dados. 4. *Stack overflow*. 5. JavaScript.

I. Catarina, Daniel Dantas. II. Titulo

CDU: 004 C357s

SERVIÇO PÚBLICO FEDERAL
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
COORDENAÇÃO DO CURSO SUPERIOR EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAMPUS CAJAZEIRAS

ATA 31/2022 - CADS/UNINFO/DDE/DG/CZ/REITORIA/IFPB

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO (TCC)
CURSO: ANÁLISE E DESENVOLVIMENTO DE SISTEMAS (ADS)

Às 09h00 do dia 05 do mês de MAIO do ano de 2022, o(a) aluno(a) **DANIEL DANTAS CATARINA**, matrícula **201812010027**, apresentou, como parte dos requisitos para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, seu trabalho de conclusão de curso, tendo como título **"STACK DEVELOPMENT EDUCATION - UMA EXTENSÃO PARA O EDITOR DE CÓDIGOS VS CODE QUE ARMAZENA DADOS DA BASE DE CONHECIMENTO STACK OVERVFLOW**". Constituíram a banca examinadora os professores **Francisco Paulo de Freitas Neto** (orientador), **Paulo Ewerton Gomes Fragoso** (examinador) e **Fábio Abrantes Diniz** (examinador).

Após a apresentação e as observações dos membros da Banca Examinadora, ficou definido que o trabalho foi considerado **APROVADO** com nota **95**, com a condição de que o (a) aluno (a) entregue, no prazo máximo de 30 dias, a versão final do trabalho com as correções sugeridas pelos membros da banca examinadora. Eu, FÁBIO ABRANTES DINIZ, Coordenador do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, lavrei a presente ata, que segue assinada digitalmente por mim e pelos membros da banca examinadora.

Cajazeiras, 11 de maio de 2022.

Documento assinado eletronicamente por:

- Francisco Paulo de Freitas Neto, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 11/05/2022 17:46:54.
- Fabio Abrantes Diniz, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 11/05/2022 23:02:41.
- Paulo Ewerton Gomes Fragoso, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 12/05/2022 09:51:12.
- Daniel Dantas Catarina, ALUNO (201812010027) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS, em 12/05/2022 14:40:58.

Este documento foi emitido pelo SUAP em 11/05/2022. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 293459
Código de Autenticação: d9aab40bd7

Dedico esse trabalho a minha família, amigos e principalmente a minha esposa que sempre esteve presente me dando apoio, e me inspirando com sua força.

”A educação é a arma mais poderosa que você pode usar para mudar o mundo.”

Nelson Mandela

RESUMO

A Tecnologia da Informação é algo que cresce cada vez mais, e de certa forma tomou conta da vida de todos, pois, está presente na educação, saúde, politica, e até vida pessoal. Com isso, cresce o interesse das pessoas em se tornar desenvolvedores e entrar no mercado. Porém, o mundo da programação é vasto, e existem várias linguagens, e cada linguagem com suas respectivas bibliotecas e frameworks. Proporcionando, aos desenvolvedores, dificuldades no aprendizado, pois, encontram documentação mal explicada, exemplos de código que nem sempre deixam claro o que está sendo feito, e abordam de forma breve alguma funcionalidade. Portanto, este Trabalho de Conclusão de Curso se propõe a construção de uma extensão para o editor de códigos *VSCode*, que surge como um auxílio para os desenvolvedores javascript fazerem pesquisas de erros de códigos ter e acesso fácil a códigos e exemplos de bibliotecas e *frameworks JavaScript* na plataforma *Stack overflow*. Além de solucionar algumas limitações da busca *multi-tag* do *stackoverflow*.

Palavras-chave: *Stack overflow*. JavaScript. VSCode. Extensão. Aprendizado.

ABSTRACT

Information Technology is something that grows more and more, and in a way it has taken over everyone's lives, as it is present in education, health, politics, and even personal life. As a result, people's interest in becoming developers and entering the market grows. However, the world of programming is vast, and there are several languages, and each language with its respective libraries and frameworks. Providing developers with learning difficulties, as they find poorly explained documentation, code examples that do not always make it clear what is being done, and briefly address some functionality. Therefore, this Course Conclusion Work proposes the construction of an extension for the *VSCode* code editor, which appears as an aid for javascript developers to do code error research and have easy access to codes and examples of libraries and *JavaScript frameworks* on the *Stack overflow* platform. In addition to solving some limitations of the *multi-tag* search of *stackoverflow*.

Keywords: *Stack overlow*. JavaScript. VSCode. Extension. Apprenticeship.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ADS	Análise e Desenvolvimento de Sistemas
API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
HTML	*Hyper Text Markup Language*
IDE	*Integrated Development Environment*
IFPB	Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
MEC	Ministério da Educação
NPM	*Node Package Manager*
TCC	Trabalho de Conclusão do Curso
TI	Tecnologia da Informação
UI	*User Interface*
VSCode	*Visual Studio Code*

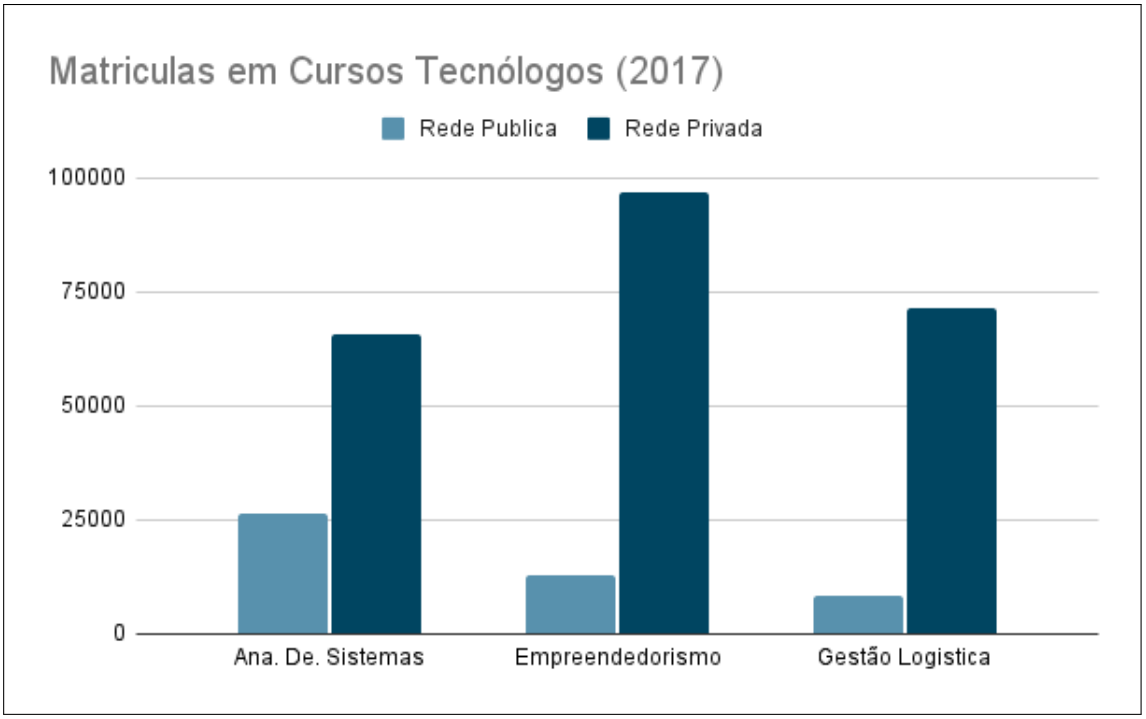
SUMÁRIO

1 INTRODUÇÃO

A área de Tecnologia da informação (TI) vem crescendo bastante durante os últimos anos, e cada vez mais, existem pessoas ingressando nessa área. Com a alta demanda, várias tecnologias se despontam, surgindo uma enorme variedade de caminhos no qual o programador pode seguir, consequentemente pressionando a comunidade de desenvolvedores a está sempre em constante aprendizado, e se adequando ao que o mercado propõe.

Como pode ser visto na Figura 1 adaptada do último censo do Ministério da Educação (MEC, 2017), o curso de análise e desenvolvimento de sistemas está entre os cursos tecnólogos com maior número de matrículas. Mostrando que existe um interesse das pessoas em entrar na área de tecnologia, e seguir uma carreira como programador.

Figura 1 – Matriculas em Redes Públicas e Privadas



Fonte: Adaptado de (MEC, 2017)

Porém, a curva de aprendizado para dominar a programação pode ser um pouco grande, tendo em vista que podem surgir dificuldades, já que existe um grande número de linguagens e *frameworks* a serem dominados. Essas dificuldades podem surgir por alguns fatores frequentes, tais como: a documentação abordar uma linguagem

rebuscada, que pode acabar assustando desenvolvedores inciantes que acabaram de ingressar no aprendizado da devida tecnologia a qual a documentação se refere, como também pode haver exemplos de código que acabam não sendo claros no que proponham explicar, ou até explicações resumidas de mais, na qual deixa dúvidas no desenvolvedor, e venha a dificultar ainda mais o aprendizado.

Um recurso bastante utilizado pelos desenvolvedores, é o reuso de *software*. Como formas de reuso de *software* os desenvolvedores procuram em fóruns e sites de perguntas e respostas, códigos explicativos feitos pela comunidade, que podem contribuir na parte base da construção de uma *feature* de projeto. "O reuso de software tem sido indicado durante os processos de desenvolvimento e manutenção de *software*. Pois, oferece benefícios tais como: redução do tempo e custos no desenvolvimento; produtos de melhor qualidade e confiabilidade; uso eficaz de especialistas;, etc."(SOMMERVILLE, 2019).

Uma plataforma muito utilizado pela comunidade é o *Stack Overflow*[1], que é um site de perguntas e respostas (*posts*) relacionadas ao desenvolvimento de software contendo serviços que auxilie principalmente os desenvolvedores iniciantes. De acordo com (TREUDE et al., 2012 apud ROCHA; MAIA, 2016), o *Stack Overflow* possui mais de 12 milhões de visitantes e 135 milhões de visualizações de páginas todo mês. Tornando essa plataforma, uma fonte de base de conhecimento importante para que os desenvolvedores possam utilizar no processo de desenvolvimento ou manutenção dos seus softwares.

Porém, o *Stack Overflow* é um fórum em que existem muitos usuários, proporcionando uma grande quantidade de informações, consequentemente contribui para a existência de respostas que acabam não suprindo as dúvidas do usuário. O *Stack Overflow* oferece mecanismos de busca, tais como: barra de pesquisa; filtragem por *tags*, usuários, entre outros (ROCHA; MAIA, 2016). Estes mecanismos auxiliam os desenvolvedores a encontrarem suas respostas, no entanto, retornam muito conteúdo. E no caso da barra de pesquisa, dependendo da formulação da pesquisa do usuário, esta pode excluir *posts* que poderiam ajudar no entendimento.

1.1 PROBLEMÁTICA

A demanda de pessoas com interesse em programação é grande. Porém, existem muitas dificuldades que são encontradas para que uma pessoa consiga evoluir na programação, assim como foram introduzidos no Capítulo 1, podem ser das mais diversas. Tendo em vista esse problema, esse trabalho tem o intuito de desenvolver uma

[1] https://pt.stackoverflow.com/

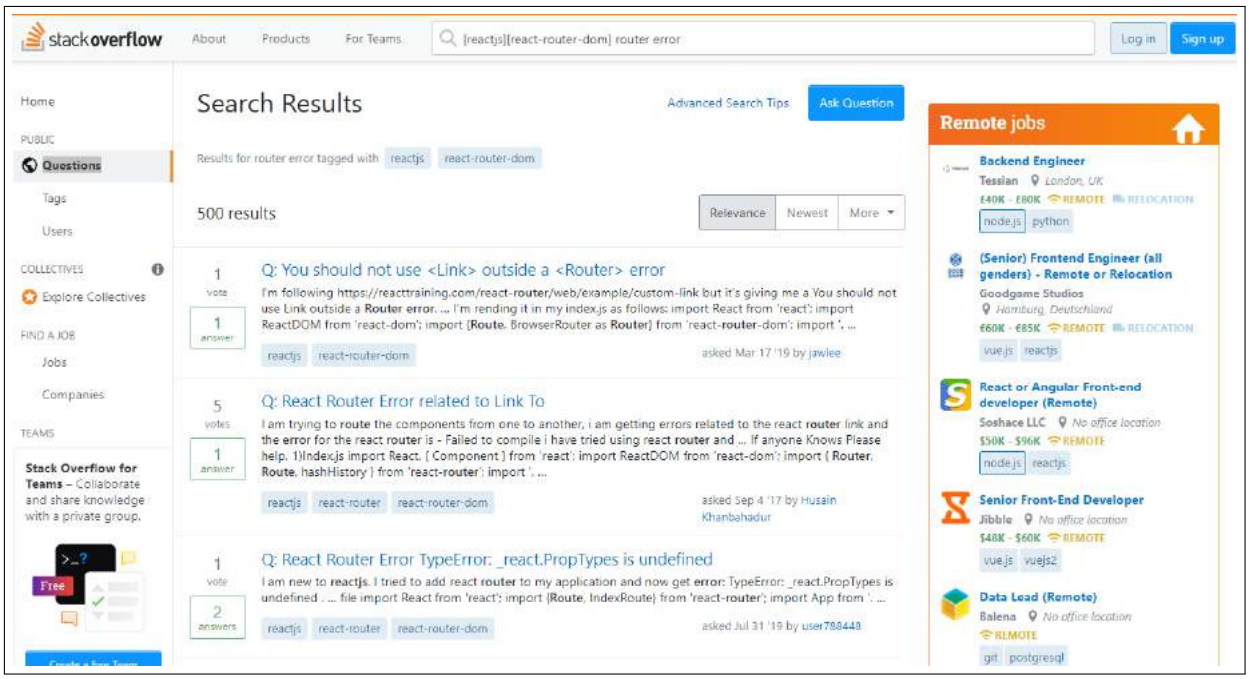
extensão para o editor código *Visual Studio Code* (VSCode), adicionando no mesmo um mecanismo de busca que auxilia os desenvolvedores javascript com o suporte a buscas de erros e dúvidas usando como base para a pesquisa, as dependências que o projeto javascript possui em seu arquivo *package.json*, arquivo esse sendo usado para armazenar os dados das dependências presentes no projeto javascript.

Este trabalho propõe atender programadores de todos os níveis que programam na linguagem *javascript*, e utilizam o (VSCode) como editor de códigos principal, sendo assim uma ferramenta de ensino e autodidatismo. O intuito é oferecer uma extensão que conta com um motor de busca de fácil acesso na ferramenta, permitindo que desenvolvedores façam pesquisas na base de dados do *Stack Overflow*, utilizando das dependências já instaladas no arquivo package.json presente nos projetos dos desenvolvedores, como parâmetro de pesquisa. O acesso aos dados vai ser feito pela *API* aberta do *Stack Overflow*.

A API do *Stack Overflow*, oferece em seu mecanismo de busca a possibilidade de realizar buscas filtrando por *tags*. Segundo o próprio *Stack Overflow*, *tags* são palavras chaves ou rótulos, sendo usados para relacionar e categorizar os posts e perguntas. Sendo assim, cada *post* tem consigo um agrupamento de *tags* para auxiliar sua identificação.

O *Stack Overflow* já possui de um motor de busca em seu site, que já funciona bem, no mesmo é permitido adicionar na busca de texto convencional a filtragem por tags, que permitem que todas as respostas venham sobre aquela determinada tag, o que melhora bastante o resultado final das respostas, tendo em vista que filtra melhor e deixa mais próximo do que seria a resposta necessária para o usuário que está realizando a pesquisa. A Figura 2 apresenta um exemplo de busca utilizando tags.

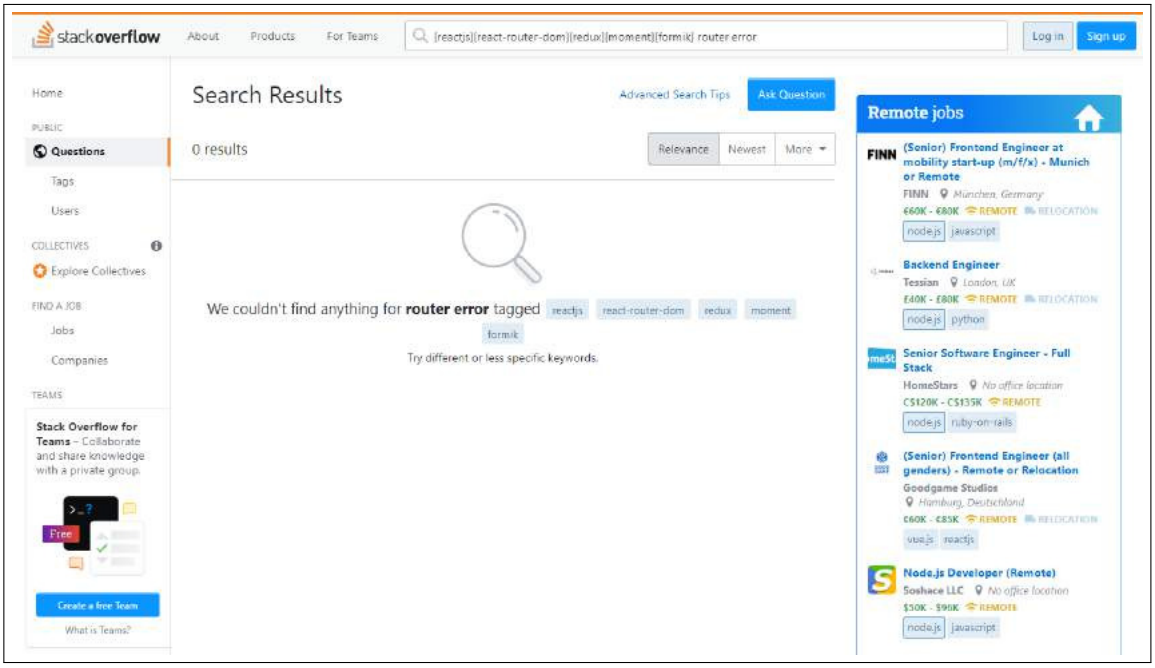
Figura 2 – Busca por *Tag*



Fonte: Elaborado pelo autor

Durante o desenvolvimento da extensão foi encontrado um problema relacionado à busca do *stack overflow*, tanto em seu motor de busca, quanto em sua API. O problema ocorria quando eram feitas buscas com muitas *tags*, ao longo que o número de *tags* ia sendo inserido, o motor de busca e a API começavam a excluir muitos resultados. Isso ocorre porque existe um problema relacionado a busca multi *tag* que causa exclusão de resultados, que seriam importantes. Esse problema é perceptivo quando é usado um número muito grande *tags*, pois o *Stack Overflow* retorna esse *post* ou resposta apenas se o *post* possuir todas as *tags* que foram mandadas como pode ser visto na Figura 3.

Figura 3 – Busca por muitas *Tags*



Fonte: Feita Pelo Autor

Como a ideia da extensão a ser desenvolvida parte de utilizar de dependências do projeto, isso seria um problema tendo e vista que a extensão vai sempre utilizar com base todas as dependências para pesquisa. Então, para resolver o problema, nesse trabalho também vai ser feita uma API que contará com um banco de dados, onde vai ser indexado todos os dados conforme as pesquisas vão sendo realizadas pela extensão. O banco de dados escolhido foi o *Elastic Search*, banco muito utilizado para indexação de dados e busca textual, que de acordo com o próprio site do *Elastic Search*[2] é utilizado por grandes nomes como: Netflix, Uber, Slack e Microsoft.

Existem extensões que também utilizam da *api* aberta do *stack overflow*, sendo as extensões *Stack Overflow View* [3], e *Stack Overflow Search* [4], ambas são extensões que permitem realizar buscas na base de dados do *Stack Overflow*. Porém, a extensão que foi desenvolvida tem como diferencial o uso das bibliotecas já instaladas no projeto para auxiliar na busca, e tem seus dados indexados em uma base de dados, onde a extensão acessa se comunicando com um serviço que provê os dados.

[2] https://www.elastic.co/pt/nossos-clientes-dão-vida-à-busca
[3] https://github.com/IsaacSomething/stackoverflow-view-vscode
[4] https://github.com/gcrev93/code-stackoverflow

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Esse trabalho tem como objetivo desenvolver uma extensão para o editor de códigos VSCode, com o intuito de servir de auxílio para os desenvolvedores javascript fazerem pesquisas de erros de códigos e ter acesso fácil a códigos e exemplos de bibliotecas e frameworks javascript na plataforma Stack overflow.

1.2.2 Objetivos Específicos

- Desenvolver o campo de pesquisa para realizar a busca dentro do VSCode.
- Realizar uma leitura das dependências instaladas no projeto, e as usar como parâmetro para a pesquisa.
- Relacionar as dependências instaladas no projeto com tags que o stackoverflow usa para relacionar os posts.
- Renderizar um view com todo o resultado e detalhamento dos posts obtidos na pesquisa, dentro do VSCode.

1.3 METODOLOGIA

As seguintes atividades tiveram que ser realizadas para que o trabalho fosse concluído com exito:

- **Atividade 1 - Realizar busca de artigos e ferramentas semelhantes:** fase na qual foi feito uma busca de artigos e ferramentas já existentes que abordam o tema geral do trabalho que está sendo escrito.
- **Atividade 2 - Definição do escopo principal da ideia:** fase na qual foram realizadas reuniões e discussões para decidir que tipo de ferramenta de fato seria desenvolvida no final desse trabalho.
- **Atividade 3 - Coleta de dados:** fase na qual foi aberto um formulário para coletar dados de desenvolvedores, sobre algumas questões que seriam importantes para o desenrolar do trabalho.
- **Atividade 4 - Desenvolver um protótipo da ferramenta:** nessa etapa foi desenvolvido um protótipo de como seria o funcionamento da ferramenta a ser desenvolvida.

- **Atividade 5 - Desenvolver uma versão inicial da ferramenta:** nessa etapa foi desenvolvido uma versão inicial do sistema que tem o intuito de mostrar o primeiro passo da ferramenta.
- **Atividade 6 - Desenvolver melhor a filtragem dos resultados do mecanismo de busca da ferramenta:** nessa etapa serão feitos filtros que melhoram o resultado das buscas que o desenvolvedor faz na base de dados do *Stack Overflow*, retornando assim apenas *posts* com scores mais altos, e que usam *tags* semelhantes às dependências instaladas no projeto do desenvolvedor que faz a busca.
- **Atividade 7 - Desenvolver uma solução para a exclusão de resultados da busca *multi tag*:** essa etapa terá como intuito criar alguma solução para o problema de exclusão de resultados, quando é feito uma requisição com várias *tags* diferentes para a base de dados do *Stack Overflow*.
- **Atividade 8 - Conclusão do documento:** nessa etapa foi feito a conclusão da versão final do trabalho de conclusão de curso.

1.4 ORGANIZAÇÃO DO DOCUMENTO

O restante deste documento foi dividido em mais três capítulos. O capítulo 2 apresenta toda a fundamentação teórica da ideia proposta pelo documento, descrevendo sobre o *Visual Studio Code* e suas extensões, boas práticas de programação e organização dos arquivos, *Stack Overflow* e sua *API*, e os trabalhos relacionados.

O capítulo 3 destrincha a solução a ser desenvolvida, coletando dados que foram levantados a partir de uma pequena pesquisa de preferências de editores de códigos, feito com alguns alunos do IFPB Campus Cajazeiras, além de alguns *user stories*, protótipos, e arquitetura da solução a ser desenvolvida.

E por fim no capítulo 4 são apresentados os resultados que foram alcançados, e a conclusão do documento.

2 REFERENCIAL TEÓRICO

Neste capítulo será apresentada a fundamentação teórica, detalhando sobre os principais temas abordados neste trabalho, tais como: *Visual Studio Code* e sua extensibilidade, boas práticas de programação e organização dos arquivos, *Stack Overflow* e sua *API* e, por fim, destaca os trabalhos relacionados.

2.1 VISUAL STUDIO CODE E SUA EXTENSIBILIDADE

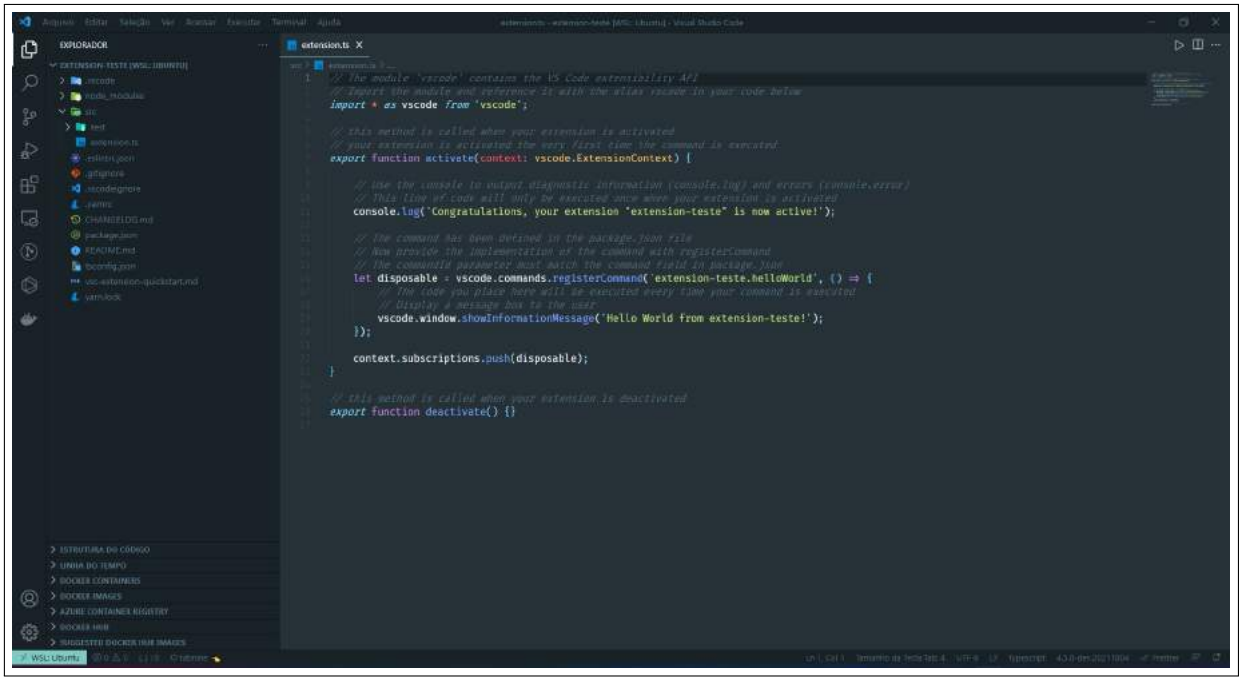
O VSCode é um editor de código criado pela *Microsoft*, que segundo a mesma, teve o intuito inicial de apresentar de forma mais simples e prático o ambiente de desenvolvimento semelhante a *Integrated Development Environment* (IDE) *Visual Studio*, que tem o intuito de ser um ambiente de desenvolvimento para aplicações C#, e .NET. O foco do *VSCode* é o desenvolvimento de aplicações *web* usando *javascript*, *typescript* e *Node.js*. Além disso, possui uma vasta biblioteca de extensões criadas pela comunidade, que permite que o mesmo seja usado para desenvolvimento de aplicações com as mais diversas linguagens. (MICROSOFT, 2022) (RASK et al., 2021)

O VSCode tem seu núcleo desenvolvido em *Node.js* e sua UI desenvolvida em *HTML* e *CSS*, e utiliza o *Framework Electron*, para fazer *build* do seu executável para as plataformas *Windows*, Linux e *Mac*. Além disso, possui o código fonte aberto, com um mecanismo de integração de código-fonte externo via extensões, que usam um ecossistema *Node.js*, sendo a mesma tecnologia presente em seu núcleo. (MICROSOFT, 2022)(RASK et al., 2021)

Segundo a (MICROSOFT, 2022), a empresa incentiva a comunidade a desenvolver extensões para a ferramenta, com o objetivo de que possa ser maleável a maioria das tecnologias. As extensões podem ser criadas por meio da junção de duas bibliotecas que o *dev* precisa instalar globalmente via *Node Package Manager* (NPM), sendo a famosa biblioteca de geração de código Yeoman[1], e o gerador de código da *Microsoft* que funciona por meio do *Yeoman*, por isso se faz necessário a instalação do mesmo. Exemplo de código gerado pelo gerador de código do VSCode é apresentado na Figura 4.

[1] https://yeoman.io/learning/index.html

Figura 4 – Estrutura de código *VSCode Extension*



Fonte: Feito Pelo Autor

Após o desenvolvimento das extensões, as mesmas podem ser publicadas no *Extension Visual Studio Marketplace* [2], em que todas as extensões podem ser cadastradas e disponibilizadas para ser efetuado o download diretamente pelo gerenciador de extensões do *VSCode*. Existem os mais diversos tipos de extensões disponiveis no gerenciador de extensões, que vão desde extensões que dão suporte a desenvolvimento a linguagens diferentes, como extensões que servem para ajudar o desenvolvedor em alguma finalidade, como o *ESLint* [3] *Prettier*[4] que são extensões com foco em padronizar código para seguir as recomendações e boas práticas no desenvolvimento de código coletivo (MICROSOFT, 2022).

2.2 *STACK OVERFLOW* E API

O *Stack Overflow* é uma plataforma de perguntas e respostas, em forma de *posts*, em que os desenvolvedores relatam problemas e dúvidas obtidas durante o desenvolvimento de software com alguma linguagem de programação ou *framework*. Os *posts* criados podem ser acessado e respondidos pela comunidade cadastrada no ambiente Stack OverFlow. Também existem no *Stack Overflow posts* tutoriais, que

[2] https://marketplace.visualstudio.com/VSCode
[3] https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint
[4] https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode

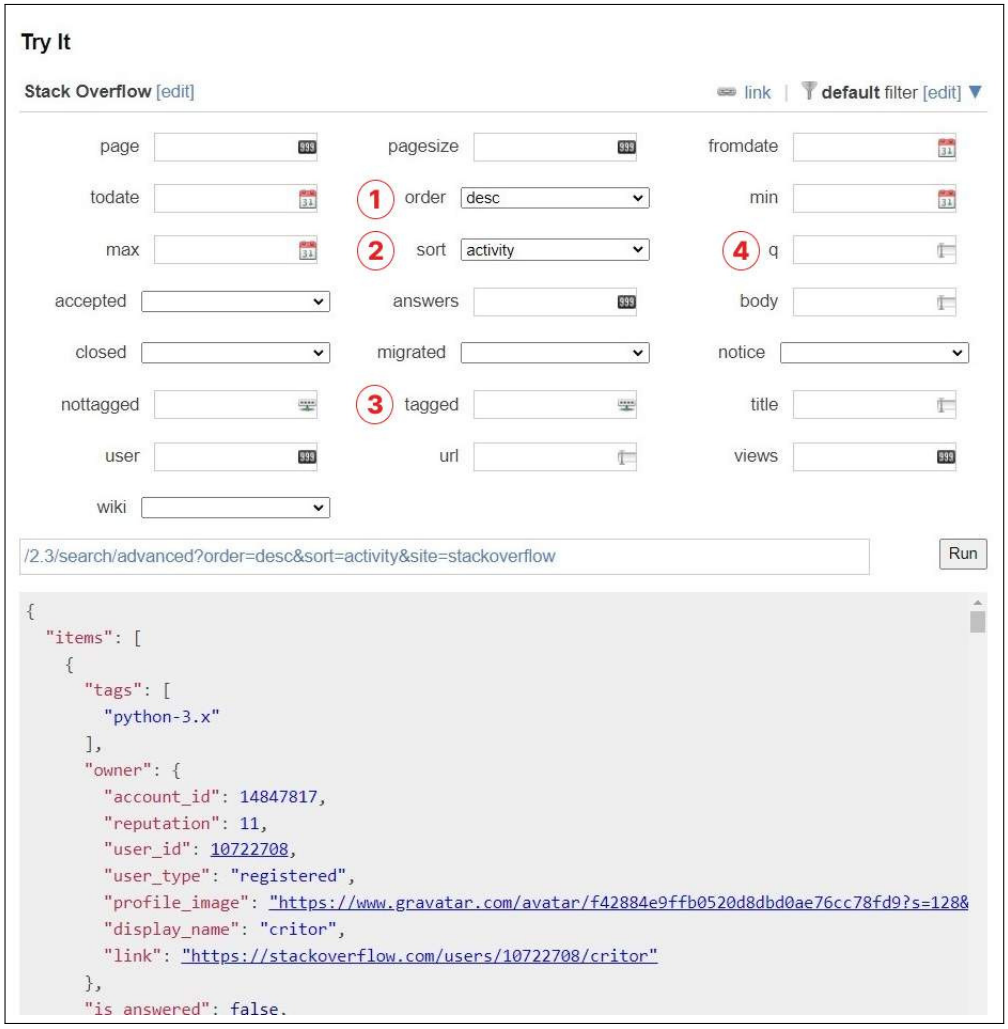
são *posts* auto explicativos, que tem o objetivo de mostrar a implementação de uma certa funcionalidade de uma forma mais simples e intuitiva que as documentações convencionais. (STACKOVERFLOW, 2022)

Como já foi relatado por (TREUDE et al., 2012 apud ROCHA; MAIA, 2016) o *Stack overflow* contém uma base dados de bastante referência e confiança para os desenvolvedores de todas as áreas de desenvolvimento e linguagens. O *Stack Overflow* possui em média 12 milhões de visitantes e 135 milhões de visualizações a cada mês, com isso o mesmo ganhou bastante confiança por parte dos desenvolvedores. O mesmo possui de diversas formas de validações das respostas dos *posts*, ou dá veracidade dos *posts* autoexplicativos, que podem ser desde validação de usuários com contas profissionais ou com *score* alto, sendo feito com base no número de respostas, e aprovação por parte dos desenvolvedores que visualizam os *posts*.

O *Stack Overflow* também provê seus dados por meio de uma *API REST* [5] *(Representational State Transfer)*, em que o mesmo oferece algumas rotas que entregam dados como *posts*, respostas de *posts*, dados de usuários, etc. O *Stack Overflow* também possui uma rota que serve para fazer pesquisas mais avançadas, adicionando alguns filtros para entregar resultados mais próximos do que o requisitante das informações precisa, conforme apresentado na Figura 5. Na mesma figura pode ser visto que a requisição recebe inúmeros parâmetros, onde pode ressaltar 4 em especiais, que estão enumerados na imagem. O parâmetro numero 1 é usado para ordenar os resultados retornados pela requisição em crescente ou decrescente, o parâmetro de número 2 é para dizer qual o critério de ordenação para pesquisa, seja ela por atividade ou por *score*, o parâmetro de número 3 é utilizado para passar as tags que o requisitante quer usar na pesquisa feita a API, e o parâmetro numero 4 é o texto propriamente dito que será usado para ser feita a pesquisa.

[5] https://api.stackexchange.com/docs

Figura 5 – Rota de pesquisa avançada na API



Fonte: (STACKOVERFLOW, 2022)

O *Stack Overflow* oferece um número limitado de 10000 requisições por dia, para cada endereço de IP de origem da requisição, mas caso o projeto abordado seja de grande escala, o desenvolvedor pode registrar sua aplicação, no que o *Stack Overflow* chama de *Stack Apps*. O Stack Apps, é um registro de aplicações que o *Stack Overflow* possui para projetos que são avaliados pelo *Stack Overflow*, e ganham o selo de oficial. O desenvolvedor faz o registro do seu produto, passando algumas informações como nome, descrição do produto e domínio da aplicação, com isso o *Stack Overflow* registra em sua base dados a aplicação e provê uma chave para ser utilizada na requisição, assim aumentando o limite diário de requisições. (STACKOVERFLOW, 2022)

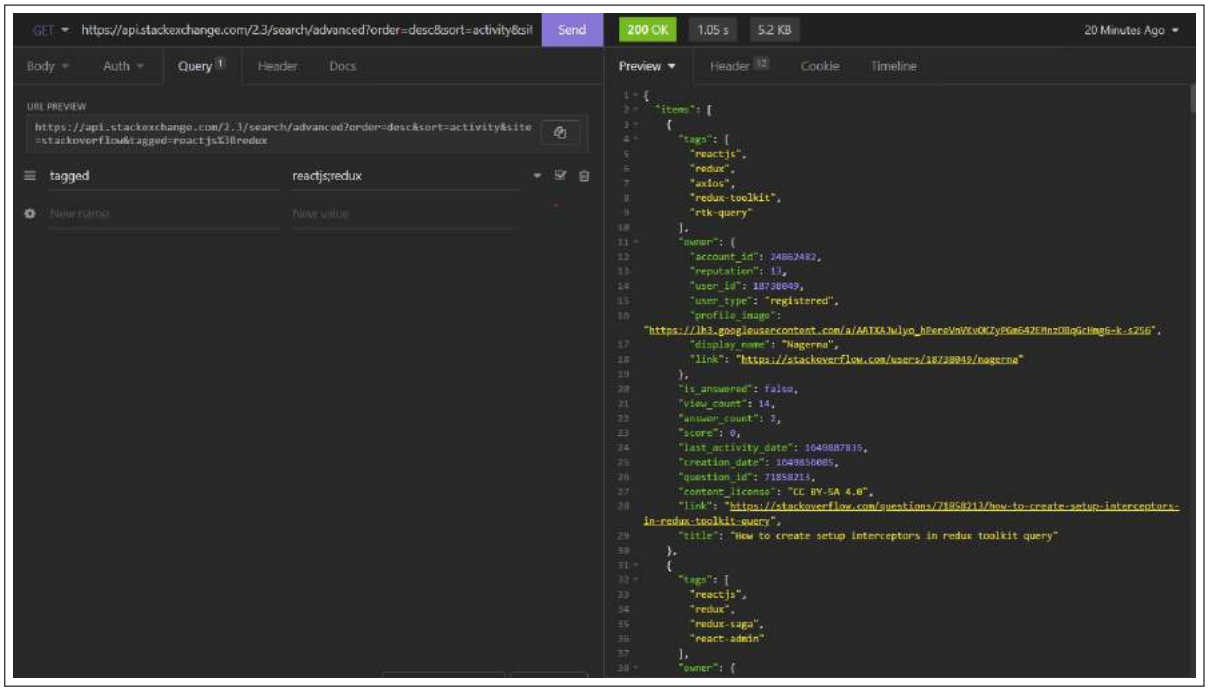
A API provê para o desenvolvedor acesso a diversas rotas para obtenção de dados através de requisições HTTP. O *Stack Oveflow* disponibiliza acesso às informações como respostas de posts, perguntas, proprio detalhamento do post, artigos,

entre outros. Além de também fazer possuir umas rotas específicas para pesquisa em seu motor de busca. (STACKOVERFLOW, 2022)

2.2.1 Busca com tags

Em seu modelo de busca avançada a *Stack API* permite que a pesquisa seja feita de diversas formas, e uma das formas que mais se destaca é a busca feita por tags. A API permite que seja passado por meio de *Query Param* o parametro tagged, onde nele podemos colocar tags separadas por ";", isso facilita um pouco a busca, já que assim pode ser feito um direcionamento do assunto em específico que está sendo pesquisado. Com isso se for passado a tag "node"no parametro *tagged*, a *Stack API* retornaria posts em que os autores colocaram node como tag, e assim para as demais tags. A *Stack API* permite que seja passado mais de uma tag para pesquisa, porém só será retornado posts que possuem todas as tags pesquisadas, sendo assim quanto maior o número de tags menos posts serão retornados pela *Stack API*. Um exemplo pode ser visto na Figura 6, em que a mesma exemplifica uma requisição com uma busca por tags, usando o simulador de requisição HTTP Insomnia [6].

Figura 6 – Rota de busca por *tag*



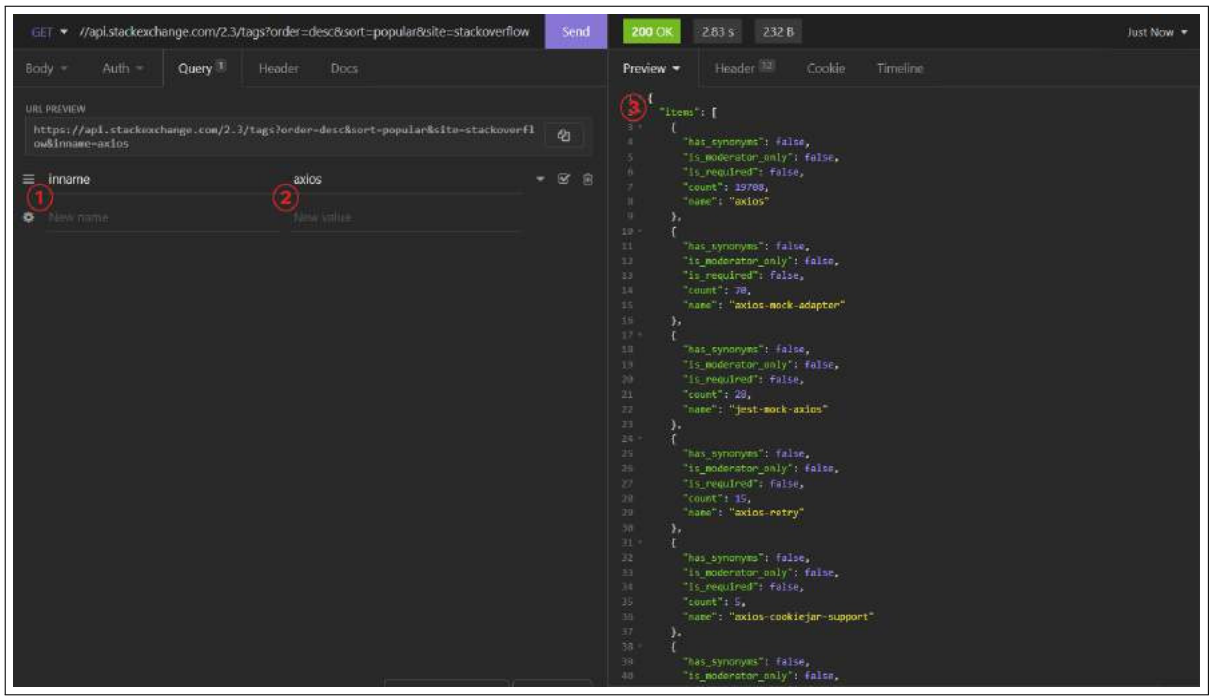
Fonte: Feita pelo autor

As tags podem ser encontradas através de uma outra rota HTTP, que a *Stack API* possui. Nessa rota pode-se ser feita uma requisição com uma palavra, e a *Stack*

[6] https://insomnia.rest/

API retorna todas as tags relacionadas a palavra pesquisada. Um exemplo da requisição pode ser visto na Figura 7, em que é feito uma busca de todas as *tags* para a palavra-chave "axios". A mesma figura apresenta uma numeração de 3 itens para fácil identificação do processo, no item 1 temos o tipo de pesquisa de tag, onde é usada a tipo *inname*, onde vai ser pesquisado qualquer *tag* que seja composta pelo texto pesquisado no item 2, que é a *tag* a ser pesquisada, e no item 3 é mostrado os resultados daquela busca de *tag*.

Figura 7 – Rota de busca de *tags*



Fonte: Feita pelo autor

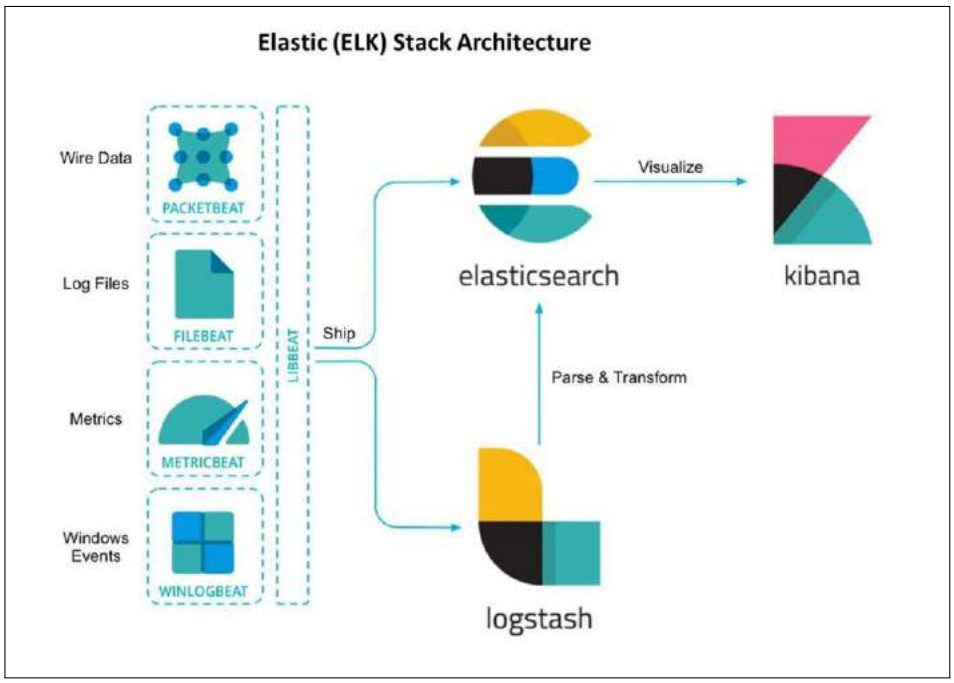
2.3 *ELASTIC SEARCH*

Segundo o site oficial do *Elasticsearch* (ES) (ELASTICSEARCH, 2022) o mesmo é um mecanismo de busca e análise de dados distribuídos, gratuito e aberto para os principais tipos de dados disponíveis, tais como dados textuais, numéricos, geoespaciais, estruturados e não estruturados. O ES é desenvolvido sobre o Apache Lucene e foi lançado pela primeira vez em 2010 pela Elasticsearch N.V. Conhecido por suas REST APIs simples, natureza distribuída, velocidade e escalabilidade. O ES é o componente central do Elastic Stack, que se trata de um agrupamento de ferramentas gratuitas que o mesmo possui, e também são abertas a ingestão, armazenamento e análise de dados.

Comumente chamado de ELK *Stack* (pelas iniciais de ES, Logstash e Kibana), o *Elastic Stack* inclui uma rica coleção de agentes *lightweight* conhecidos como *Beats* para enviar dados ao ES. E o ES utiliza o Kibana em sua implementação que é uma ferramenta de visualização e gerenciamento de dados para o *Elasticsearch* que fornece histogramas em tempo real, gráficos de linhas, gráficos de pizza e mapas. O Kibana também inclui aplicações avançadas como o Canvas, que permite aos usuários criar infográficos dinâmicos personalizados com base em seus dados, e o *Elastic Maps* para visualizar dados geoespaciais.(ELASTICSEARCH, 2022)

O *Elastic Search* é bastante utilizado em motores de busca, devido ao seu kit bastante completo para buscas e indexação. Motores de busca são ferramentas utilizadas para buscarem determinados conjuntos de informação e são indispensáveis nas pesquisas de informações na web. E possui como finalidade fornecer resultados relevantes para aqueles que o usam, de modo que se satisfaçam com os resultados obtidos. E uma das formas para alcançar esse é utilizando da indexação, que permite guardar informação útil. (QUINTANA, 2012). Segue na Figura 8 uma representação da arquitetura do *Elastic Search*, na mesma pode-se observar o processo de como o *Elastic Search* funciona, onde temos todos os serviços de logs de arquivos e metricas presentes no *Elastic Search*, que usa do *logstash* para transformação dos dados e o *kibana* para a visualização do mesmo. O Logstash se trata de uma pipeline que realiza processamento de dados, e tem seu código open source, e funciona do lado do servidor, o que permite realizar a ingestão dos mais diversos tipos de dados, o intuito é transformar esses dados antes do processo indexação no Elasticsearch. (ELASTICSEARCH, 2022)

Figura 8 – Arquitetura *Elastic Search*



Fonte: (ELASTICSEARCH, 2022)

Segundo o site oficial, o *ElasticSearch* possui uma arquitetura de pesquisa textual bastante rápida, tornando a latência do momento em que um documento é indexado até que ele se torne buscável muito pequena, que o torna adaptável a casos de uso sensíveis ao tempo como análise de dados de segurança e monitoramento de infraestrutura. O ES divide e armazena seus dados em índices, sendo conjuntos de documentos que se relacionam, esses documentos possuem o formato *JavaScript Object Notation* (JSON), sendo assim cada documento se correlaciona com um conjunto de chaves. (ELASTICSEARCH, 2022)

O ES utiliza de uma estrutura de dados, que se chama indice invertido, que foi prejada para permitir buscas textuais de forma mais rápida. Um índice invertido lista cada palavra exclusiva que esteja inserida em qualquer documento e identifica os documentos em que cada palavra aparece.(ELASTICSEARCH, 2022), Além disso, é permitido usar o ES para consultas de texto em documentos, denominada *Full-Text Search*. Com esse tipo de consulta é possível utilizar modelos de similaridade, que pode ser usado para associação de documentos indexados, com base em uma pesquisa feita. (SOARES, 2020)

O ES foi utilizado nesse projeto, com o intuito de indexar e armazenar dados do *Stack Overflow* para poderem ser recuperados posteriormente usando de técnicas de busca textual, técnicas essas sendo disponibilizadas como ferramenta pelo ES, e

com isso, pode-se obter um bom desempenho na busca, seja relacionado a tempo ou propriamente a qualidade da busca, tendo mais documentos ligados ao texto que foi informado na pesquisa. A relação da pesquisa com os documentos é feita por meio das buscas textuais nos atributos título e tags, sendo informações que os posts armazenados possuem.

2.4 BANCOS DE DADOS *NOSQL*

Cada vez mais o número de aplicações que lidam com diversos tipos de dados, vem crescendo, assim como o caso de sistemas que coletam dados de sensores ou de redes sociais. Assim se faz necessário desenvolver e utilizar de tecnologias que gerenciam e dão suporte a toda essa quantidade de dados heterogêneos, e ainda apresentem um bom desempenho. Para que esses objetivos sejam atingidos, vem sendo propostos sistemas de gerenciamento de banco de dados (SGBDs), conhecidos como NoSQL Multi-modelo, que cada vez mais vem sendo utilizados. Ainda não é claro quais os SGBDs que se adaptam melhor aos projetos, pois ainda são tecnologias consideravelmente recentes. (AQUINO; MELLO, 2021).

Se faz necessário alta disponibilidade e velocidade para as aplicações, devido ao alto tráfico de dados nos sistemas modernos, esses sistemas influenciaram o desenvolvimento desses bancos de dados, para suprir toda essa quantidade de dados, e ainda oferecer um bom desempenho para essas aplicações. Essas tecnologias vão além dos bancos de dados convencionais e já consolidados, os bancos de dados relacionais. Estes bancos vêm impulsionando, um movimento conhecido como movimento NoSQL. Esse movimento tem o intuito de resolver problemas específicos, esses bancos NoSQL são divididos de acordo com suas peculiaridades e otimizações. Por exemplo, a Amazon utiliza de sistemas chave-valor (key-value) para gerênciar sua lista de produtos mais vendidos, preferências do consumidor, gerenciamento de dados de produtos, entre outras aplicações que a mesma possui, em seus ecossistemas.(ROCKENBACH et al., 2018).

Um banco de dados NOSQL presente no projeto que foi desenvolvido, foi o banco de dados Redis, que se trata de um banco de dados de armazenamento de dados em memória, de código aberto que é geralmente usado como base de dados ou cache de sistemas. É fornecido pelo redis uma estrutura com os mais variados tipos de dados, sendo eles strings, hashes, listas, conjuntos e índices geoespaciais. O Redis também possui diversos tipos de persistência e armazenamento, e fornece alta disponibilidade de dados por meio do *Redis Sentinel*. Para obter o melhor desempenho, o Redis trabalha com um conjunto de dados na memória. Dependendo do seu caso de

uso, o Redis pode persistir seus dados despejando periodicamente o conjunto de dados no disco ou anexando cada comando a um log baseado em disco.(REDIS, 2022).

2.5 TRABALHOS RELACIONADOS

Devido à importância que o *Stack Overflow* possui para a comunidade de desenvolvedores, surgiram vários trabalhos que possuem o intuito de estudar, e aproveitar dessa base de dados. Alguns trabalhos foram considerados de bastante importância para esse que está sendo desenvolvido, e valem a pena ser citados.

Um trabalho que é de grande importância para a solução a ser desenvolvida por esse documento é o trabalho de (ROCHA; MAIA, 2016), que trata o desenvolvimento de uma *API* que gera documentação com base em *posts*, usando vários tipos de metodologia de pesquisa. No final os autores selecionaram quatro metodologias diferentes de pesquisa e filtros para geração de documentação para uma dada API.

Outro trabalho bastante interessante que usa a *API* do *Stack Overflow* é o de (AHASANUZZAMAN et al., 2018), que faz o uso de dados de *posts* do *Stack Overflow*, para ajudar desenvolvedores no designer de novas *APIS*. Em direção a com o objetivo de classificar as postagens relacionadas ao problema da API, o autor desenvolve uma abordagem de aprendizagem supervisionada usando um CRF que pode classificar frases relacionadas com o problema.

Um trabalho que deu um bom caminho sobre indexação, que ajudou a estabelecer conceitos utilizados nesse trabalho que está sendo apresentado nesse TCC, é o trabalho de (SOARES, 2020), trabalho esse que propõe o uso de abordagens de processamento textual que se aplicam no problema de indexação e busca de impressões digitais. Nesse trabalho são implementadas duas abordagens que utilizam de dados textuais para o seu desenvolvimento: o método 1, usa de tabelas hash e locality sensitive hashing; Já o método 2, utiliza de índices invertidos do Elastic e Locality sensitive hashing.

Vale a pena também fazer uma menção ao artigo escrito por (PONZANELLI et al., 2013), que se trata de um documento que trás a ideia do desenvolvimento de um plugin integrado a IDE Eclipse, que integra uma webview no Eclipse com o motor de busca do Stack Overflow. O resultado foi uma seção no eclipse que permitia realizar buscas no Stack Overflow, e permitir que desenvolvedores que usavam o eclipse, pudessem ter resultados de problemas de forma mais acessível dentro da IDE.

A ferramenta proposta por esse documento possui uma grande espiração nos

trabalhos citados nesse capítulo, porém com o intuito de fornecer uma solução similar a ambas, de forma totalmente integrada ao editor de códigos, e utilizando de indexação de dados e leitura das dependências abertas no projeto para fornecer melhores resultados.

3 SOLUÇÃO DESENVOLVIDA

Como foi apresentado na sessão 1.1, esse trabalho propôs o desenvolvimento de uma Extensão [1] e uma Api [2] que serviram como um mencanismo para a pesquisa de problemas e erros de códigos, integrado ao ambiente de desenvolvimento via extensibilidade. O ambiente de desenvolvimento escolhido para desenvolvimento da extensão foi o editor de código *VSCode*. O presente capítulo apresenta elementos de análise, projeto e implementação da extensão proposta.

3.1 *USER STORIES*

User Story (US), ou histórias de usuário, são descrições dos requisitos sob o ponto de vista desse usuário. A *User Story* busca descrever de uma forma simples e coerente as necessidades do cliente (RAHARJANA et al., 2021).

As histórias de usuários estão ganhando cada vez mais espaço no processo de desenvolvimento de software, principalmente no desenvolvimento ágil de software. As histórias de usuários são o artefato mais utilizado no desenvolvimento ágil de software, que expressam requisitos do ponto de vista do usuário. Uma história de usuário é uma especificação semiestruturada de requisitos escrita em linguagem natural. Um modelo de história de usuário pode ter o seguinte formato: como [OMS],EU quer/quer/precisa/-pode/gostaria de [O QUE], para que [POR QUÊ]. Ele contém elementos importantes de requisitos: QUEM deseja, O QUE se espera do sistema e, opcionalmente, PORQUE é importante. (RAHARJANA et al., 2021)

Foram identificadas as seguintes histórias de usuário para a extensão proposta:

- ***User Story* 01 - Pesquisa:** Como desenvolvedor, gostaria que a extensão tivesse algum mecanismo de pesquisa, no qual poderá realizar pesquisas no editor de código. Para assim poder realizar pesquisas de problemas no próprio editor de código.
- ***User Story* 02 - Exibição:** Como desenvolvedor, gostaria que o resultado da pesquisa fosse exibido no próprio editor de código. para assim visualizar o resultado da pesquisa no próprio editor de código.

[1] https://github.com/daniel-dantas/stack-development-education-extension
[2] https://github.com/daniel-dantas/stack-development-education-api

- ***User Story* 03 - Filtragem:** Como desenvolvedor, gostaria que houvesse um filtro na pesquisa feita, com base nas dependências instaladas no projeto javascript. Para assim poder ter uma pesquisa focada no conteúdo aberto no workspace do projeto.
- ***User Story* 04 - Acesso ao detalhemento do post no próprio editor**: Como desenvolvedor, gostaria de ver o detalhamento do post dentro do proprio VSCode, junto com os exemplos de código e melhores respostas. Para assim obter todo o detalhamento do post pesquisa no próprio editor de código.

3.2 PROJETO DA SOLUÇÃO PROPOSTA

3.2.1 Diagrama de atividade

Os Diagramas de Atividades permitem que fluxos de execução sejam representados e conduzidos por processamentos, mostrando todo o fluxo de execução existente de uma atividade a outra. São diagramas importantes, pois permitem a modelagem de aspectos dinâmicos e intuitivos de um sistema computacional, podendo ser usados para descrever mais detalhamento o fluxo de execução ao nível de algoritmo. (COSTA et al., 2019)

Para facilitar a compreensão do funcionamento da extensão, uma representação desse fluxo pode ser visto na Figura 9. O funcionamento da extensão se dá da seguinte forma: a extensão disponibiliza para o *VSCode* uma barra de pesquisa na qual o desenvolvedor irá colocar a descrição do problema que o mesmo está tendo, em seguida, esse texto entra no Core da extensão e passa para o *Command Search*, sendo o controlador responsável por coletar a informação e processar. Para isso, o *Command Search* também utiliza de outra funcionalidade chamada *Command Dependency*, que é um controlador responsável por se comunicar com uma *feature* do *VSCode Core* chamada *Workspace Provider*, que é uma funcionalidade de acesso a *workspace* do projeto que está aberto no *VSCode*.

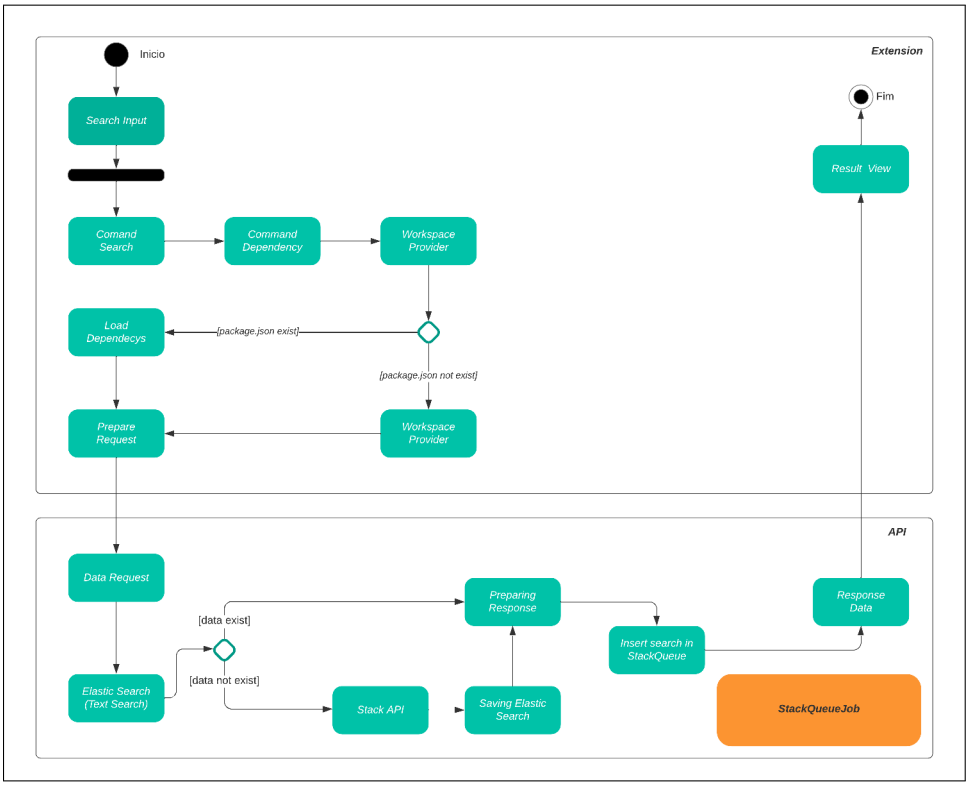
O *Command Dependency* acessa o *workspace*, e procura o arquivo *package.json* do projeto, arquivo no qual tem todas as informações de dependências do projeto, e assim, carrega as dependências e coloca em um *array* que é entregue para o *Command Search* novamente. Já com as informações de dependências e o problema que o desenvolvedor quer pesquisar, é preparado a estrutura para uma requisição HTTP, sendo feita para a *API* que foi desenvolvida para processar a pesquisa.

Após receber os dados da pesquisa, a *API* faz o processo de realizar a montagem de uma busca no *Elastic Search*, onde os dados são montados, e é feito uma

validação se existe dados relevantes para aquela pesquisa. Caso exista, é feito um processo de preparação da resposta para a extensão, e em seguida a pesquisa é inserida em uma fila para reprocessamento, e em caso de não existir dados no *elastic search*, é feito uma requisição HTTP para a *Stack API*, e em seguida os dados são indexados no *Elastic Search*, para serem usados para fácil acesso em uma próxima pesquisa.

Em seguida, a API insere os dados em uma fila que futuramente vai ser reprocessada por uma automação de código chamada *StackQueueJob*, e após inserir na fila a API devolve a resposta para a extensão montar a página de visualização da pesquisa.

Figura 9 – Diagrama de atividade da extensão e API

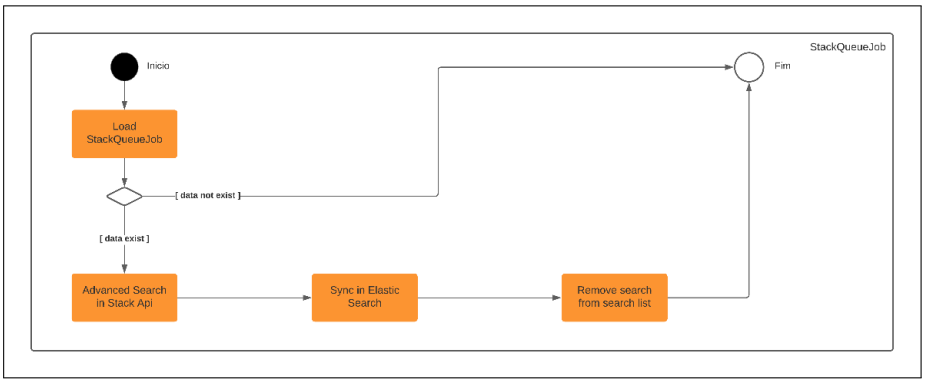


Fonte: Feito pelo autor

A *StackQueueJob* é uma automação de código que roda a cada minuto, o seu intuito é refazer uma busca na *Stack API* e sincronizar os dados obtidos com o *Elastic Search*, a fim de inserir novos dados que não estavam presentes no *ES*. Para isso ela carrega as pesquisas que estão salvas na *StackQueue*, verifica se existe dados, caso exista é realizado uma pesquisa avançada na *Stack API* e em seguida é feito o processo de sincronizar os dados no *Elastic Search*, após isso a pesquisa é removida

da *StackQueue* e encerra sua execução. Uma representação desse fluxo, pode ser visualizado na Figura 10

Figura 10 – Diagrama de atividade da *StackQueueJOB*



Fonte: Feito pelo autor

3.2.2 Arquitetura da extensão

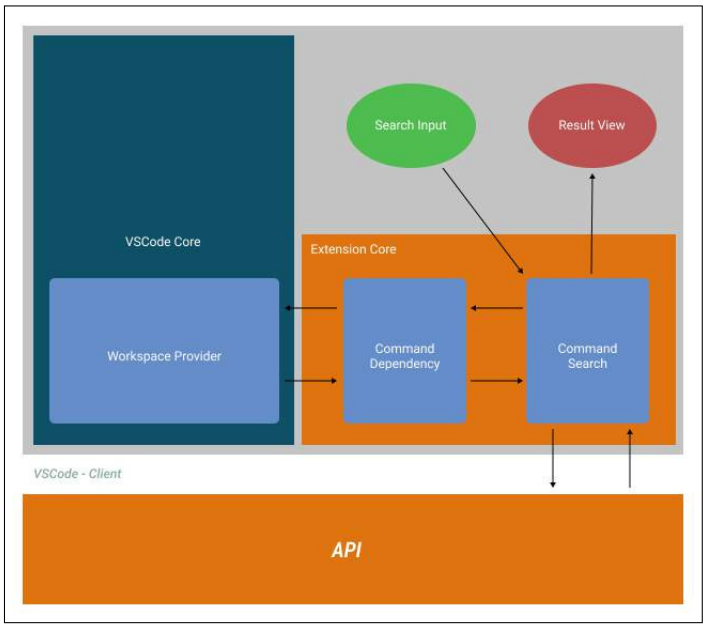
No capítulo anterior foi apresentado um diagrama de atividade, no qual teve o intuito de apresentar o passo de execução da extensão. Assim como o capítulo anterior, esse tem o intuito de voltar no fluxo de execução da ferramenta, só que mostrando ao nível mais técnico a execução de todos os passos da ferramenta.

Conforme apresentado na seção 3.2.1, a extensão provê para o usuário um campo de texto, que é renderizado e apresentado ao usuário, para digitar a informação de pesquisa. Após isso, esse campo de texto entrega a string digitada no campo para o modulo *Command Search* realizar todo o processamento da pesquisa, para isso o *Command Search* realiza uma chamada para outro módulo, sendo o módulo *Command Dependecy*. Nesse módulo é realizado o processo de carregamento dos dados de dependências presentes no projeto, para isso, ele acessa um módulo presente no núcleo do próprio *VSCode*, sendo o módulo *Workspace Provider*, módulo esse sendo nativo do *VSCode*, sua função é acessar objetos de *Workspace* que estão abertos no editor de código.

Com acesso a esses objetos do *Workspace* do projeto, o módulo *Command Dependecy* localiza o arquivo *package.json*, e realiza um processo de leitura do arquivo, pegando apenas o que está no atributo *dependencies*, que no caso seria as dependências do projeto. Após ler essas dependências, é devolvido essas dependências em forma de um *array*, para o módulo *Command Search*, ele realiza uma requisição HTTP para a *API*, que realiza todo o processamento (Mais a frente será explicado o processo que a API faz para processar esses dados), após processada as informações, a API devolve para o módulo *Command Search*, um array de posts do Stack OVerflow,

em seguida os posts são organizados, e inseridos em uma tela que é renderizada e entregue para o usuário. Uma representação da arquitetura de funcionamento da extensão, pode ser vista na Figura 11.

Figura 11 – Arquitetura da Extensão



Fonte: Feito pelo autor

Em seguida, temos o serviço backEnd, ilustrada na Figura 12 que foi construído para processar as informações e passar para a extensão. A API recebe as informações passadas pela extensão, através de um controler, sendo chamado de *Search Controller*. Após a requisição chegar ao controller, é feito um processo de busca textual no banco de dados Elastic Search usando os dados de pesquisa fornecidos pela extensão, e também usando das dependências passadas. Já com o resultado da pesquisa em mãos, o *controller* faz um processo de validação dos dados, onde é verificado se existe dados no *Elastic Search*, que coincidem com as informações a serem pesquisadas, caso exista é feito um processo de preparação da resposta.

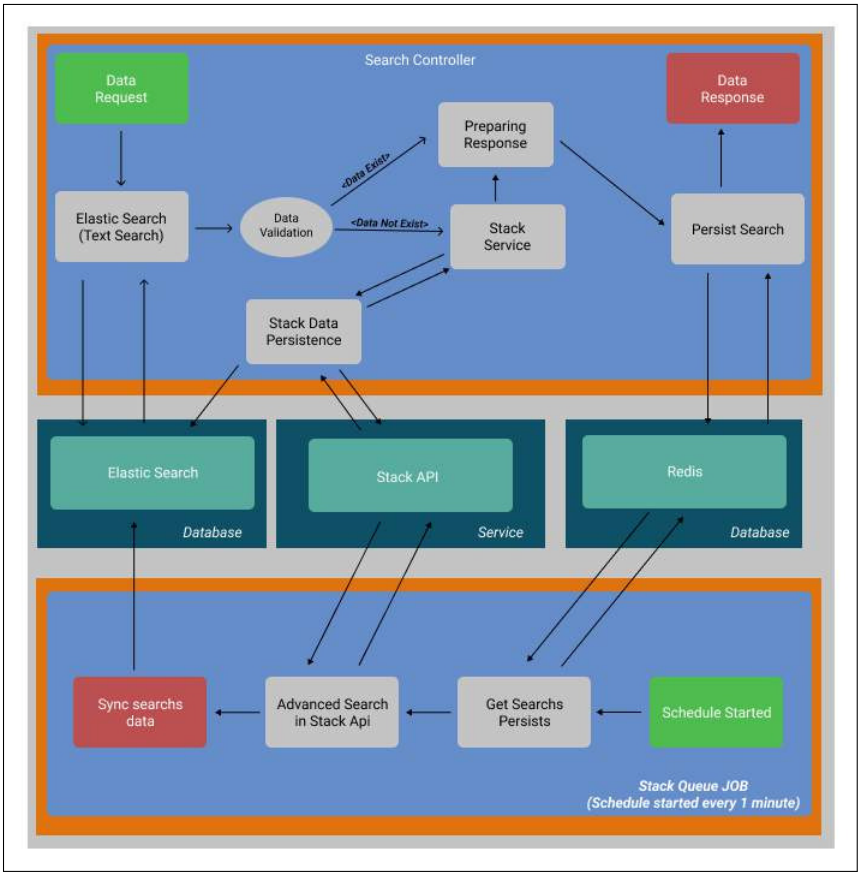
Caso não exista dados armazenados no *Elastic Search*, é feita uma pesquisa avançada por meio de um serviço chamado *Stack Service*, esse serviço faz uma busca na *Stack API*, e em seguida persiste os dados no banco de dados *Elastic Search*. Após persistir os dados, o *service* entrega esses dados para o *controller* novamente, onde também é feito o processo de preparação da resposta, após a preparação da resposta feita, é feito o processo de armazenamento da pesquisa feita, no banco de dados *Redis*, para futuramente uma *JOB*, que funciona como automação de código, vai pegar esses dados e refazer a pesquisa. Após os dados serem salvos no *Redis*, o *Search Controller*

entrega a resposta para a requisição feita.

Além do controller, a API também conta com a execução de *JOBs*, sendo automações de código que rodam em determinado horário, com o intuito de fazer fluxos de execução agendados. A API possui uma JOB, sendo chamada de *Stack Queue JOB* essa JOB tem a finalidade de refazer a pesquisas que já foram feitas, com o intuito de atualizar os dados dos posts no *Elastic Search*, sendo uma automação programada para ser feita a cada um minuto, e só para quando todas as pesquisas são refeitas.

Após o início do seu ciclo de execução, a mesma carrega todos os dados salvos no banco de dados *Redis*, dados esses que são salvos no fluxo de execução do *Search Controller*. Após carregar os dados, é feito o processo de *Advanced Search* na *Stack API*, processo esse que usa os dados da pesquisa feita, junto com as dependências para agrupar alguns posts, em seguida é feito o processo de sincronização dos dados recebidos pela consulta feita na *Stack API*, onde é persistido todos os novos posts que ainda não tinham sido salvos no banco de dados do *Elastic Search*. Após concluir o salvamento dos dados a JOB finaliza seu fluxo de execução. Uma representação de todo o serviço Back End, pode ser encontrado na Figura 12

Figura 12 – Arquitetura do Serviço *Back End* por completo



Fonte: Feito pelo autor

3.3 IMPLEMENTAÇÃO

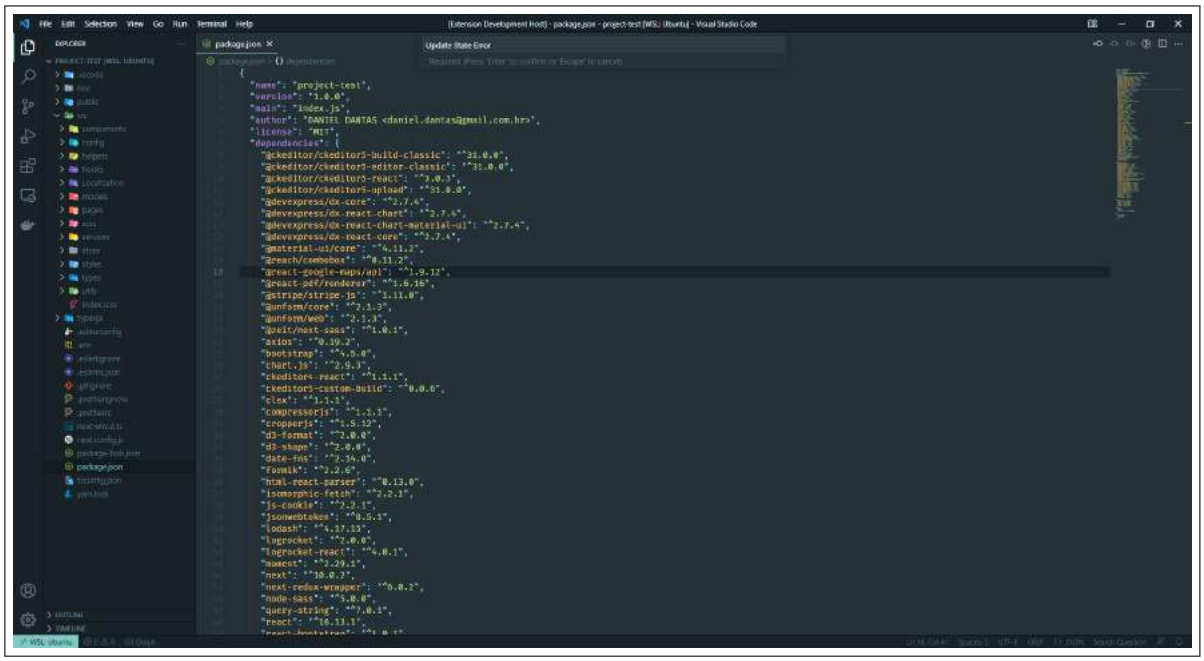
3.3.1 Funcionamento da Extensão

Pensado na facilidade de integração com o *VSCode*, e em manter o desempenho e praticidade que o mesmo editor passa, a extensão foi desenvolvida com um fluxo execução bem simples e telas bem menos carregadas de informações. Foram desenvolvidas 2 telas e um campo de pesquisa, o campo de pesquisa presente na Figura 13 possui o design padrão do *VSCode*, e as telas são views carregadas integradas ao *VSCode*, sendo as telas mostradas na Figura 14 e na Figura 15, ambas utilizando de um design simples, para que não jogasse informações de mais na tela, tendo em vista que o desenvolvedor teria aquela área como um ambiente de trabalho, e um número de informação muito alto poderia prejudicar o foco do desenvolvedor, o que iria contra o objetivo do projeto que era oferecer praticidade aliado a produtividade.

Após a instalação da extensão, a mesma fica disponível pelo comando: CTRL + ALT + A, ou pelo botão com nome *Search Question* que fica na barra inferior do

VSCode. Ao clicar ou iniciar pelo atalho, é apresentado um campo de texto, a frente da área principal de edição de código, onde o usuário pode preencher com a informação que deseja fazer a busca, esse componente e processo pode ser visto na Figura 13.

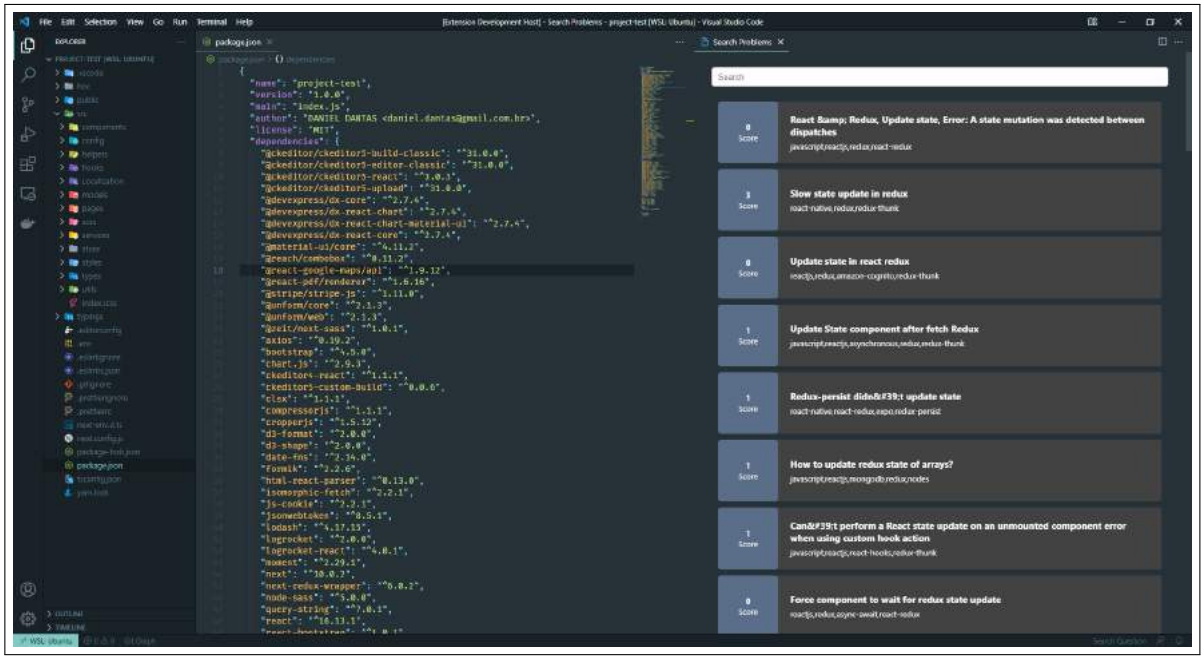
Figura 13 – Pesquisa na Extensão



Fonte: Feito pelo autor

Logo em seguida após obter a informação que foi digitado no campo de texto a extensão vai acessar o workspace do projeto e carregar o arquivo *package.json*, com o arquivo carregado a extensão vai ler as dependências presentes nesse arquivo, e vai enviar essas informações para a *API*, a *API* vai processar a pesquisa, verificar se possui respostas para essa pesquisa no *Elastic Search*, caso não exista, vai fazer uma requisição direta para a *API* do Stack Overflow, já com os dados, a *API* volta com uma lista de posts para serem processados pela extensão, a extensão carrega esses dados, e em seguida insere-os em uma view que vai listar os posts ao lado da workapace principal do projeto aberto, essa view pode ser vista na Figura 14.

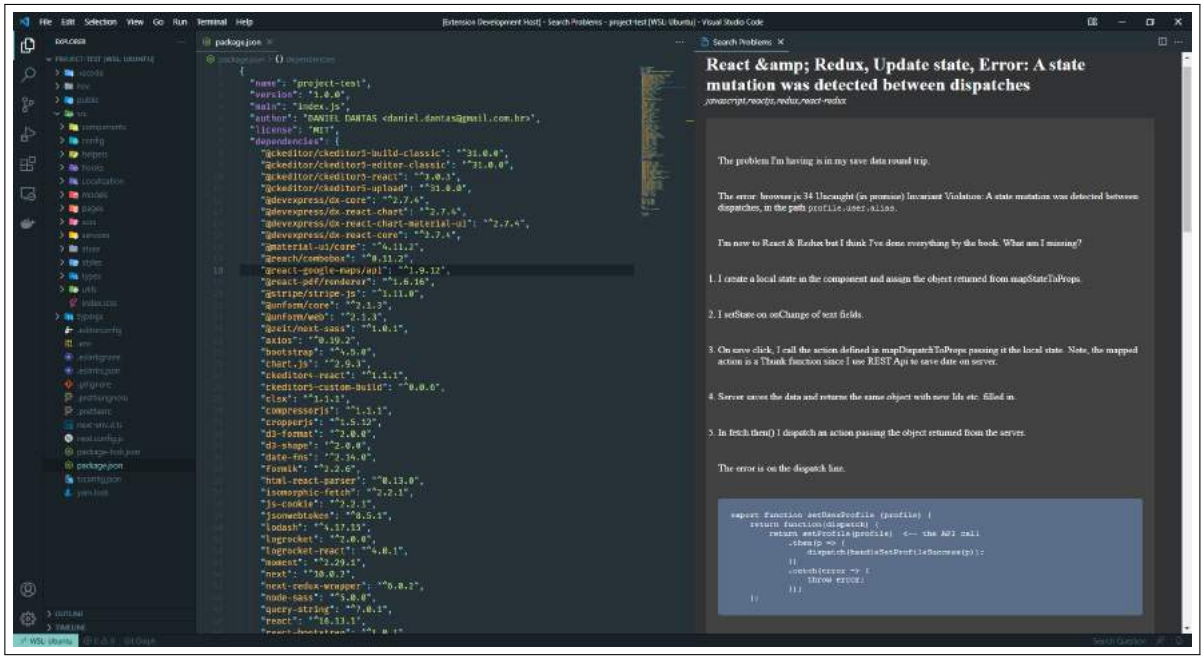
Figura 14 – Resultados obtidos na pesquisa



Fonte: Feito pelo autor

Na view que é exibida, a extensão lista todos os Posts obtidos, em uma ordenação de maior coerência com a pesquisa feita, e com informações das tags que esses posts possuem, além de também conter o Score que o post possui. Após a ação de click em uma das opções, a view de listagem é fechada e em seguida é aberta uma view apenas com o detalhamento do post, nesse detalhamento será possível observar as informações principais do posts, exemplos de código, e abaixo de tudo terá uma lista com as melhores respostas e comentários que aquele post recebeu, essa pagina de detalhamento pode ser vista na Figura 15.

Figura 15 – Detalhamento do *Post*



Fonte: Feito pelo autor

3.3.2 Tecnologias utilizadas

Relembrando que o desenvolvimento da extensão foi dividido em duas partes, uma parte que seria a extensão em si, que vai ser instalada no *VSCode*, e a outra parte é o desenvolvimento da *API*, acessada pela extensão, que vai cuidar de se comunicar com stack overflow e armazenar os posts.

Para o desenvolvimento da extensão, foi usado o projeto de desenvolvimento de extensão provido pela *Microsoft*, citado no Capítulo 2.1. O mesmo oferece vários serviços de acessos aos núcleos de desenvolvimento do *VSCode*, podendo haver manipulação de alguns componentes, ou leitura de algumas informações que podem ser extraídas do próprio editor de código. Foram utilizados 2 recursos do núcleo do *VSCode*. O primeiro recurso foi o de acesso ao Workspace do projeto, sendo o *TreeDataProvider*, esse recurso provê acesso à árvore de arquivos do projeto aberto. Esse recurso foi utilizado para identificar e mapear o arquivo *package.json* do projeto. O segundo recurso do *VSCode* abordado é o de acesso a elementos de interface gráfica, sendo utilizado para adicionar o botão com nome *"Stack Question"* na barra inferior do *VSCode*, e também esse recurso foi utilizado para a criação da barra de pesquisa e as *Views* que exibem os resultados e detalhamento dos posts. Além de recursos do núcleo do *VSCode*, foi utilizado também da biblioteca *axios* para realizar as requisições da extensão para a *API*, e o *handlebars* para elaboração de interfaces junto com o núcleo de interface gráfica do *VSCode*.

Para o desenvolvimento da *API*, foi feito um projeto *NodeJS* [3] com a linguagem *Typescript* [4], nesse projeto foi utilizado os pacotes de conexão com os bancos: *Elastic Search*[5] e o *Redis*[6]. Também foi explorado a biblioteca *axios*, que dessa vez seria usada para realizar requisições para a *API* do *Stack Overflow*, além do axios também foi instalada a biblioteca *node-cron*, biblioteca utilizada para criar *crons* para *jobs* ou automações de código que a *API* possui.

Como citado no parágrafo a cima, foram utilizados dois bancos de dados. Primeiramente foi utilizado o *Elastic Search*, que é um banco utilizado para indexar e armazenar dados, com foco em pesquisa, assim como foi citado no Capítulo 2.3. O outro banco utilizado também, foi o redis, esse banco de dados funciona com armazenamento em memória, e foi utilizado para armazenar pesquisas para serem refeitas por uma *Job* que cuidava de reprocessar a pesquisa e indexar os dados no *Elastic Search*.

3.4 RESULTADOS OBTIDOS NO SITE VS RESULTADOS OBTIDOS NA EXTENSÃO

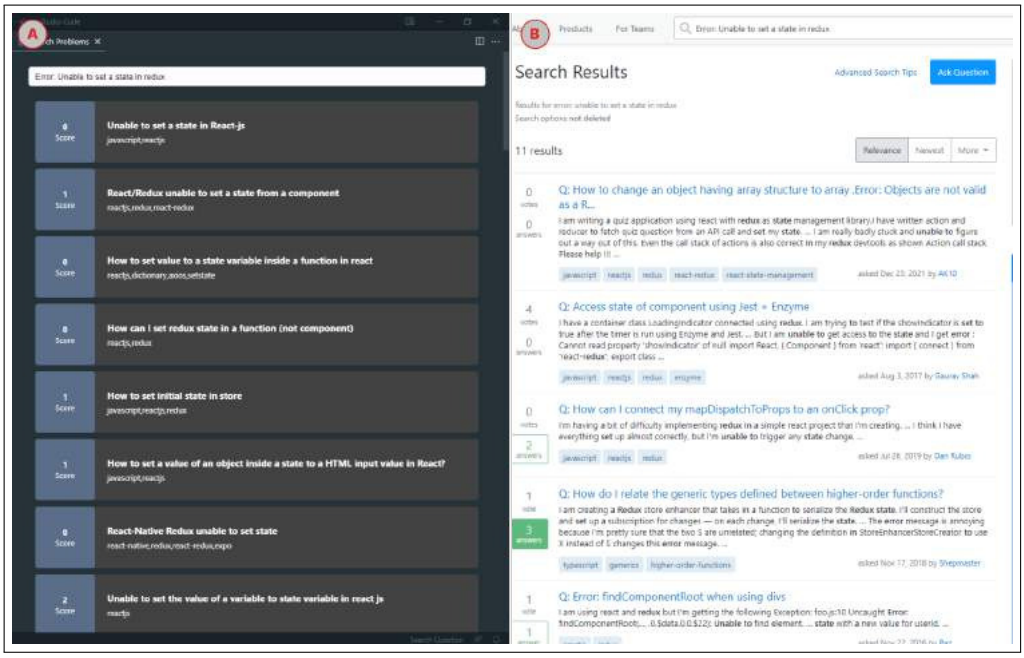
Assim como foi mencionado no Capítulo 1.1, a ferramenta que foi desenvolvida tem o intuito de fazer pesquisas na base de dados do *Stack Overflow*, utilizando como parâmetros para a pesquisas, as dependências obtidas no arquivo *package.json*, para assim tentar obter um nível de assertividade melhor nos resultados. O Stack Overflow possui um recurso que poderia ajudar a alcançar um maior resultado, que é sua busca com tags integradas, porém como também foi mencionado no Capítulo 1.1, o Stack Overflow exclui resultados importantes em sua pesquisa ao longo que vai sendo incrementado mais tags. A solução então desenvolvida, foi usar de indexação junto com um banco de dados, para indexar os posts e auxiliar na busca.

É importante relembrar que a extensão desenvolvida funciona de forma totalmente integrada ao editor de códigos, fazendo com que o desenvolvedor permaneça ali focado em seu ambiente de trabalho, enquanto realiza a pesquisa de algum problema. Já a pesquisa feita diretamente pelo Stack Overflow, precisaria ter um deslocar de ambiente para o navegador, e entrar na pagina do Stack Overflow. A interface da extensão é simples e limpa, tendo um design simplista, enquanto o site possui todo o design padrão, que pode ser bem mais completa para acompanhar a pesquisa, já a extensão tem seu foco na simplicidade pois, é necessário, para um melhor funcionamento acoplado ao editor de códigos.

[3] https://nodejs.org/docs/latest-v14.x/api/
[4] https://www.typescriptlang.org/docs/
[5] https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
[6] https://redis.io/docs/

Como a extensão usa do arquivo package.json do projeto da aplicação de um desenvolvedor, a extensão armazena essas dependências do projeto para realizar a busca, como conseguência a mesma acaba sendo mais centralizada e focada ao conteúdo do projeto, do que a busca convencional pelo Stack Overflow. Além de termos dados indexados em uma API que faz o processo de intermédio entre extensão e Stack Overflow, temos um maior numero de resultados coerentes ao conteúdo que temos no projeto aberto. Um exemplo onde pode-se ver isso é na Figura 16, onde se faz a pesquisa de um erro ao inserir dados em estado no redux, na extensão e no site do Stack Overflow. A pesquisa da extensão teve seus resultados bem mais focados ao que tem no pacote de dependência presentes no arquivo package.json do projeto do desenvolvedor, o que tornou a pesquisa mais direcionada, enquanto o site do Stack Overflow tiveram resultados que seriam coerentes com o redux, mas deixou a pesquisa um pouco mais aberta e geral.

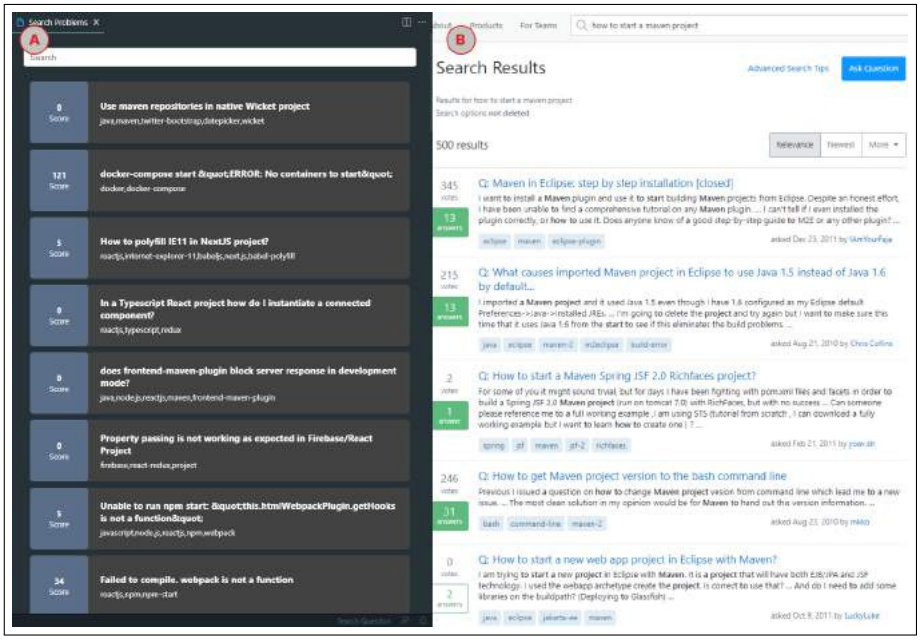
Figura 16 – Comparativo de pesquisa direcionada. A) Ferramenta proposta. B) site do stack overflow



Fonte: Feito pelo autor

Por outro lado, a extensão só vai possuir um melhor filtro de resultados apenas em condições em que a pesquisa tenha haver com o projeto aberto na extensão. Se pegar um caso em que a pesquisa seja diferente do conteúdo do projeto aberto, o site do Stack Overflow irá entregar melhores resultados, por conta de sua pesquisa mais aberta e geral, isso pode ser visto também na Figura 17.

Figura 17 – Comparativo de pesquisa aberta. A) Ferramenta proposta. B) site do stack overflow



Fonte: Feito pelo autor

Deste modo, percebe-se que a extensão pode vir a ser utilizado para sanar eventuais dúvidas ou problemas durante o trabalho de desenvolvimento de software, bem como durante a aprendizagem de determinadas tecnologias.

4 CONCLUSÃO

Como abordado no documento, a tecnologia vem crescendo cada vez mais, estando cada vez mais presente no nosso meio. Desde tarefas mais simples até tarefas mais complexas fazemos o uso de alguma tecnologia. E consequentemente, o mercado de tecnologia também cresce e junto com ele a demanda por profissionais capacitados a trabalhar nesse meio. Assim, o interesse de pessoas a querer entrar nesse meio profissional aumenta.

Esse documento se propôs a ideia de desenvolver uma extensão para auxiliar desenvolvedores de todos os níveis sejam eles iniciantes ou experientes, que usam frequentemente a linguagem de JavaScript nos projetos, oferecendo aos mesmos uma ferramenta totalmente integrada ao seu ambiente de desenvolvimento. O intuito é oferecer uma ferramenta de pesquisa de problemas e erros de código de forma prática, usando de dados já presentes ali no projeto aberto para ajudar a direcionar o caminho da pesquisa.

Todos os objetivos propostos neste trabalho, foram desenvolvidos e implementados junto a ferramenta desenvolvida. A extensão junto com a sua *API* desenvolvida, conseguiram contemplar e cumprir desde o objetivo geral estabelecido, até os objetivos específicos que também foram contemplados.

Foram encontradas algumas dificuldades no desenvolvimento da extensão. A primeira dificuldade foi problema na *API* do Stack Overflow para o desenvolvimento, o mesmo excluia muitos dados de posts importantes, quando era usado a busca multi tag, isso foi contornado com o desenvolvimento da *API* e o uso de indexação para poder agrupar corretamente os posts por tags.

Outro ponto para destacar, foi a *API* do *Stack Overflow* a qual possui um controle de requisições, que provê um máximo de 10000 requisições HTTP por dia. Pensando em um cenário pequeno isso não seria um problema, mas quando se pensa que a *API*, sendo desenvolvida para a extensão, tiver acessos acima do limiar de controle do stack Overflow, pode acontecer de ter um estouro de requisições e *API* do Stack Overflow negar essas requisições. Uma solução para isso, seria o registro da aplicação junto ao *Stack Overflow*.

Uma terceira dificuldade encontrada, foi um problema de infraestrutura para abrigar a *API*, o Banco de Dados *Elastic Search* e o *Redis*. Durante o desenvolvimento,

não se conseguiu pensar uma solução para ser feito o deploy da *API* em algum servidor, tendo em vista que os custos para fazer *deploy* de toda a estrutura, era maior que o recurso monetário presente para investimento na ferramenta, o que fez que todos os exemplos fossem rodados localmente, via *container* de aplicação. Caso o IPFB, campus Cajazeiras, disponibilizasse um servidor de aplicação, a extensão poderia ser disponibilizada a público, sendo possível obter *feedbacks* de uso em cenários reais.

Para trabalhos futuros, pode-se ser aumentado o suporte da extensão a diferentes linguagens, que assim como o JavaScript, também são ricos em bibliotecas e frameworks. Bons exemplos de linguagens que se pensa em dar suporte futuramente são as linguagens: *Python*, *C* e *Java*, tendo em vista que essas linguagens possuem uma estrutura parecida com o *JavaScript*. A ideia seria ler seus arquivos de dependência assim como é feito com o *package.json* e carregar para a *API*, logo a *API* vai fazer o mesmo processamento que já é feito para as dependências do *JavaScript*. Para trabalhos futuros, também é pensado em fazer o deploy da API e publicar a extensão na loja de extensões do *VSCode*, tendo em vista que isso foi inviabilizado devido a recursos monetários escassos, para arcar com a infraestrutura necessária para rodar a *API* e os dois bancos de dados utilizados pela *API*.

REFERÊNCIAS

AHASANUZZAMAN, M.; ASADUZZAMAN, M.; ROY, C. K.; SCHNEIDER, K. A. Classifying stack overflow posts on api issues. In: **2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)**. [S.l.: s.n.], 2018. p. 244–254.

AQUINO, A. C. d. V. de; MELLO, R. dos S. Um levantamento sobre sistemas de gerenciamento de bancos de dados nosql multimodelo. In: SBC. **Anais do XVI Escola Regional de Banco de Dados**. [S.l.], 2021. p. 31–40.

COSTA, Í.; SANTOS, R.; VIANA, D.; RIVERO, L. Activities in space: Design e avaliação de um jogo sério para o ensino de modelagem de diagrama de atividades. **SBC-Proceedings of SBGames**, 2019.

ELASTICSEARCH. **What Is ElasticSearch**. 2022. Disponível em: <https://www.elastic.co/pt/what-is/elasticsearch>.

MEC. **Censo de educação superior 2017 - Divulgação dos principais resultados**. 2017. Disponível em: <http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=97041-apresentac-a-o-censo-superior-u-ltimo&Itemid=30192>.

MICROSOFT. **Visual Studio Code Documentation**. 2022. Disponível em: <https://code.visualstudio.com/docs>.

PONZANELLI, L.; BACCHELLI, A.; LANZA, M. Seahawk: Stack overflow in the ide. In: **Proceedings of the 2013 International Conference on Software Engineering**. [S.l.]: IEEE Press, 2013. (ICSE '13), p. 1295–1298. ISBN 9781467330763.

QUINTANA, A. D. C. **Avaliação das técnicas de otimização para motores de busca**. Tese (Doutorado), 2012.

RAHARJANA, I. K.; SIAHAAN, D.; FATICHAH, C. User stories and natural language processing: A systematic literature review. **IEEE Access**, IEEE, v. 9, p. 53811–53826, 2021.

RASK, J. K.; MADSEN, F. P.; BATTLE, N.; MACEDO, H. D.; LARSEN, P. G. Visual studio code vdm support. **John Fitzgerald, Tomohiro Oda, and Hugo Daniel Macedo (Editors)**, p. 35, 2021.

REDIS. **Redis Documentation**. 2022. Disponível em: <https://redis.io/docs/>.

ROCHA, A. M.; MAIA, M. A. Automated api documentation with tutorials generated from stack overflow. In: . New York, NY, USA: Association for Computing Machinery, 2016. (SBES '16), p. 33–42. ISBN 9781450342018. Disponível em: <https://doi.org/10.1145/2973839.2973847>.

ROCKENBACH, D. A.; ANDERLE, N.; GRIEBLER, D.; SOUZA, S. Estudo comparativo de bancos de dados nosql. **Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação (REABTIC)**, v. 1, n. 8, 2018.

SOARES, J. M. S. Indexação de impressões digitais utilizando uma abordagem textual. 2020.

SOMMERVILLE, I. **Engenharia de software**. [S.l.]: PEARSON BRASIL, 2019. ISBN 9788543024974.

STACKOVERFLOW. **About StackOverflow**. 2022. Disponível em: <https://stackoverflow.com/company>.

TREUDE, C.; FILHO, F. F.; CLEARY, B.; STOREY, M.-A. Programming in a socially networked world: the evolution of the social programmer. **The Future of Collaborative Software Development**, p. 1–3, 2012.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Cajazeiras
Rua José Antônio da Silva, 300, Jardim Oásis, CEP 58.900-000, Cajazeiras (PB)
CNPJ: 10.783.898/0005-07 - Telefone: (83) 3532-4100

Documento Digitalizado Ostensivo (Público)

TCC - Daniel Dantas Catarina

Assunto: TCC - Daniel Dantas Catarina
Assinado por: Daniel Catarina
Tipo do Documento: Anexo
Situação: Finalizado
Nível de Acesso: Ostensivo (Público)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- Daniel Dantas Catarina, ALUNO (201812010027) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS, em 03/06/2022 13:39:23.

Este documento foi armazenado no SUAP em 03/06/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 537405
Código de Autenticação: 5d344a728d