UNICESUMAR – UNIVERSIDADE CESUMAR

CENTRO DE CIÊNCIAS EXATAS TECNOLÓGICAS E AGRÁRIAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

PERFORMANCE DE SISTEMAS GERENCIADORES DE BANCO DE DADOS: COMPARATIVO ENTRE POSTGRESQL E MONGODB

THIAGO TAVARES DAMACENO

MARINGÁ/PR

2021

THIAGO TAVARES DAMACENO

PERFORMANCE DE SISTEMAS GERENCIADORES DE BANCO DE DADOS: COMPARATIVO ENTRE POSTGRESQL E MONGODB

Artigo apresentado ao Curso de Graduação em Engenharia de Software da Universidade Cesumar – UNICESUMAR como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software, sob a orientação do Prof. Mestre Aparecido Vilela Junior

MARINGÁ/PR

2021

THIAGO TAVARES DAMACENO

PERFORMANCE DE SISTEMAS GERENCIADORES DE BANCO DE DADOS: COMPARATIVO ENTRE POSTGRESQL E MONGODB

Artigo apresentado ao Curso de Graduação em Engenharia de Software da Universidade Cesumar – UNICESUMAR como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software, sob a orientação do Prof. Mestre Aparecido Vilela Junior

Aprovado em: ____ de _____ de _____.

BANCA EXAMINADORA

_____

Prof. Mestre Arthur Cattaneo Zavadski – Unicesumar

_____

Prof. Mestre Marcello Erick Bonfim – Unicesumar

PERFORMANCE DE SISTEMAS GERENCIADORES DE BANCO DE DADOS: COMPARATIVO ENTRE POSTGRESQL E MONGODB

Thiago Tavares Damaceno

RESUMO

Este artigo tem como objetivo comparar a performance dos sistemas gerenciadores de banco de dados (SGBDs) PostgreSQL e MongoDB, analisando as principais operações de persistência de dados. A fim de conhecer as principais abordagens utilizadas pelos SGBDs deste artigo, foram revistos alguns conceitos referentes a SGBDs relacionais e orientados a documentos. Para realizar as análises de performance, um ambiente de testes foi construído com a linguagem de programação Typescript, visando automatizar as análises e gerar os resultados. Através dos testes realizados, pôde-se notar que o PostgreSQL é mais performático na consulta de dados relacionados e em inserções, já o MongoDB se sobressai em operações que envolvam alguma condição ou filtro de busca, sejam elas consultas, alterações ou exclusões.

Palavras-chave: Performance. SGBD

PERFORMANCE OF DATABASES MANAGMENT SYSTEMS: COMPARATIVE BETWEEN POSTGRESQL AND MONGODB

ABSTRACT

This article aims to compare the performance of the Databases management systems (DBMSs) PostgreSQL and MongoDB, analysing the main data persistence operations. Aiming at knowing the main approaches used by the relational and document-oriented DBMSs of this article, was revised some related concepts of relational and document-oriented DBMSs. To make the performance analyses, a test enviromment was built using the Typescript programming language, with the objective of automate the analyzes and generate de results. Throught the executed tests, it could be noted that the PostgreSQL is more performative in data relational queries and insertions, already the MongoDB stands out in oparations that contain some condition or search filter, wheter in queries, updates or deletes.

Keywords: DBMS. Performance

1 INTRODUÇÃO

Com o crescimento exponencial do volume de dados armazenados e consumidos, começam a surgir as dúvidas sobre qual sistema gerenciador de banco de dados utilizar. Serviços de *streaming*, redes sociais, jogos e *e-commerces* em dias de promoção exigem um tempo de resposta extremamente pequeno para cada operação. No setor de *e-commerce*, houve um aumento de 154,6% no número de usuários entre os anos de 2011 e 2021 (CRUZ, 2021).

Originalmente, os SGBDs foram criados para armazenar e permitir a manipulação dos dados, garantindo a sua segurança e integridade. Com o aumento da diversificação de aplicações, vários tipos de SGBDs foram criados, alguns preocupando-se com a organização e redução de redundância dos dados, outros priorizando a performance.

Os SGBDs relacionais são eficientes em organizar os dados, pois os separam em várias tabelas, o que evita redundâncias e facilita a manutenção (NIELD, 2016), como o PostgreSQL, contudo, com o aumento do volume de dados armazenados, a complexidade nas consultas também se eleva.

Com o avanço em *Big Data* e com a necessidade de se armazenar mais informações, os SGBDs não relacionais surgem como uma alternativa para auxiliar na manipulação dessa grande quantidade de dados. Ao utilizar uma estrutura mais simples e sem relacionamentos, como os documentos JSON no MongoDB, a performance em operações de consulta e escrita tende a ser superior em relação aos SGBDs relacionais (HOWS, MEMBREY e PLUGGE, 2019).

Cada tipo de SGBD possui os seus propósitos, porém, comparando um SGBD relacional a um orientado a documentos, qual é o real impacto na performance em diferentes tipos de manipulações de dados?

2 OBJETIVOS

2.1 OBJETIVO GERAL

Comparar a performance dos SGBDs PostgreSQL e MongoDB em diferentes manipulações de dados.

2.2 OBJETIVOS ESPECÍFICOS

- Compreender os SGBDs PostrgreSQL e MongoDB;
- Definir o ambiente de testes, contendo a infraestrutura, tecnologias utilizadas, bases de dados e os tipos de testes a serem feitos;
- Desenvolver o ambiente de testes;
- Executar os testes e obter dados de performance dos SGBDs analisados;
- Analisar os impactos na performance para cada cenário.

3 METODOLOGIA

Foi utilizado o método de pesquisa quantitativa, comparando a performance do PostgreSQL (relacional) com o MongoDB (orientado a documentos) em diferentes cenários.

Através de fontes secundárias, foi feita uma pesquisa com fundamentação teórica para compreender e definir o conceito de bancos de dados e SGBD, além da aplicação e estruturação utilizada pelo PostgreSQL e MongoDB.

Para o ambiente de testes, foi utilizado o Docker (criador e organizador de contêineres) para a criação e execução dos SGBDs. Tanto a aplicação, quanto o PostgreSQL e MongoDB foram executados localmente.

A plataforma responsável por criar as estruturas, realizar a manipulação dos dados e testes de performance foi criada para o ambiente de desenvolvimento NodeJs, utilizando a linguagem de programação Typescript.

Através de dados públicos do IBGE (2020), duas bases de dados foram criadas. A primeira contendo os vinte e sete estados do Brasil e a outra com cinco mil quinhentos e setenta munícipios, ambas contendo nome e código UF.

Após definir as bases de dados, foram feitos testes, visando mensurar o tempo gasto para a sua execução (em milissegundos), tais como:

- Inserir todos os municípios;
- Consultar todos os municípios;
- Consultar um município pelo nome;
- Alterar o nome de um município;
- Excluir um município;
- Consultar o estado de um município.

Após a execução dos testes e obtenção dos dados de performance na forma de gráficos, os resultados foram apresentados de acordo com o conjunto de dados utilizado e suas respectivas manipulações.

4 DESENVOLVIMENTO

4.1 REFERENCIAL TEÓRICO

4.1.1 OPERAÇÕES DE CRUD

De acordo com a documentação para desenvolvedores da Mozilla, CRUD (Create, Read, Update e Delete) é um acrônimo em inglês para criar, ler, atualizar e excluir, as quatro operações básicas de persistência de dados.

4.1.2 BANCO DE DADOS E SGBDs

Segundo NIELD (2016), “um banco de dados é qualquer coisa que colete e organize dados”, uma planilha, um arquivo de texto contendo configurações de um programa, um arquivo binário de objetos serializados ou até mesmo uma estrutura que possa armazenar e permitir a coleta de dados.

Em *softwares* modernos, é comum a utilização de um SGBD (aplicação para armazenar e controlar os dados), ou seja:

> Um sistema gerenciador de banco de dados (SGBD) é uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é, portanto, um sistema de software de propósito geral que facilita os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações. A definição de um banco de dados implica especificar os tipos de dados, as estruturas e as restrições para os dados a serem armazenados em um banco de dados (ELMASRI e SHAMKANT, 2005, p. 10).

Em outras palavras, o SGBD é o responsável por armazenar e garantir a integridade e segurança dos dados, provendo interfaces para que o usuário possa manipulá-lo.

4.1.3 POSTGRESQL

O PostgreSQL é um SGBD relacional, no qual tabelas (estruturas que armazenam determinados conjuntos de dados) possuem relacionamentos entre si, através de chaves identificadoras. Para NIELD (2016), a separação dos dados em diferentes tabelas elimina a redundância, facilitando a manutenção. Por exemplo, dadas as duas tabelas a seguir:

Tabela 1 – Exemplo 1 de cidades com seus respectivos códigos UF

Cidade	Código UF
Maringá	41
Londrina	41
Recife	26

Fonte: IBGE (2020)

Tabela 2 – Exemplo 2 de estados com seus respectivos códigos UF

Estado	Código UF
São Paulo	35
Pernambuco	26
Paraná	41

Fonte: IBGE (2020)

A Tabela 1, contendo o nome de algumas cidades, está relacionada com a Tabela 2 através do campo Código UF. Desta maneira, caso seja necessário alterar o nome de algum estado, somente os dados referentes aos estados serão modificados.

4.1.4 MONGODB

O MongoDB é um SGBD não relacional orientado a documentos, que armazena os dados em estruturas no formato JSON (Objeto de notação Javascript, descrito na RFC 4627).

Segundo HOWS, MEMBREY E PLUGGE (2019), o MongoDB utiliza-se do formato JSON por ser uma boa estrutura para descrever o conteúdo de um documento, não necessitando de uma estruturação prévia.

Analisando os seguintes dados:

Tabela 3 - Exemplo 2 de cidades com seus respectivos códigos UF

Cidade	Código UF
Ribeirão Preto	35
São Paulo	35
Campo Mourão	41

Fonte: IBGE (2020)

Tabela 4 - Exemplo 2 de estados com seus respectivos códigos UF

Estado	Código UF
São Paulo	35
Paraná	41

Fonte: IBGE (2020)

Os dados referentes às cidades e estados, no formato JSON, podem ser organizados da seguinte maneira: [{ cidade: Ribeirão Preto, estado: São Paulo, codigo_uf: 35 }, { cidade: São Paulo, estado: São Paulo, codigo_uf: 35 }, { cidade: Campo Mourão, estado: Paraná, codigo_uf: 41 }].

Organizando os dados em apenas uma estrutura, a consulta será mais simples e performática, no entanto, produz dados com redundância, o que aumenta a complexidade de manutenção. Também é possível organizar esses dados em duas estruturas separadas, semelhante ao PostgreSQL, relacionando-as através de alguma propriedade em comum em futuras consultas, embora esta prática não seja recomendada para o MongoDB.

4.2 *HARDWARE* E SISTEMA OPERACIONAL UTILIZADO

O sistema operacional escolhido para a criação e execução do ambiente de testes foi o Ubuntu, versão 20.04 LTS. Em relação ao *hardware*, a seguinte configuração (todos os componentes com os *clocks* padrão de fábrica) foi utilizada:

- CPU: i5 10400f 6/12 (núcleos/*threads*);
- RAM: 16GB DDR4 2666 MHz;
- SSD: NVME 2000 MB/s Leitura 1750 MB/s escrita;
- GPU: GTX 1660 6GB GDDR5.

4.3 DADOS UTILIZADOS

Através de uma API (interface de aplicações) pública, disponibilizada em 2020 pelo IBGE (Instituto Brasileiro de Geografia e Estatística), duas bases de dados no formato JSON foram criadas. A primeira contendo os vinte e sete estados do Brasil, e a segunda com cinco mil quinhentos e setenta munícipios, ambas com nome e código UF (utilizado para relacionar estado com municípios).

Para ambos os SGBDs (PostgreSQL e MongoDB), tais dados foram organizados em duas estruturas, evitando redundâncias no armazenamento.

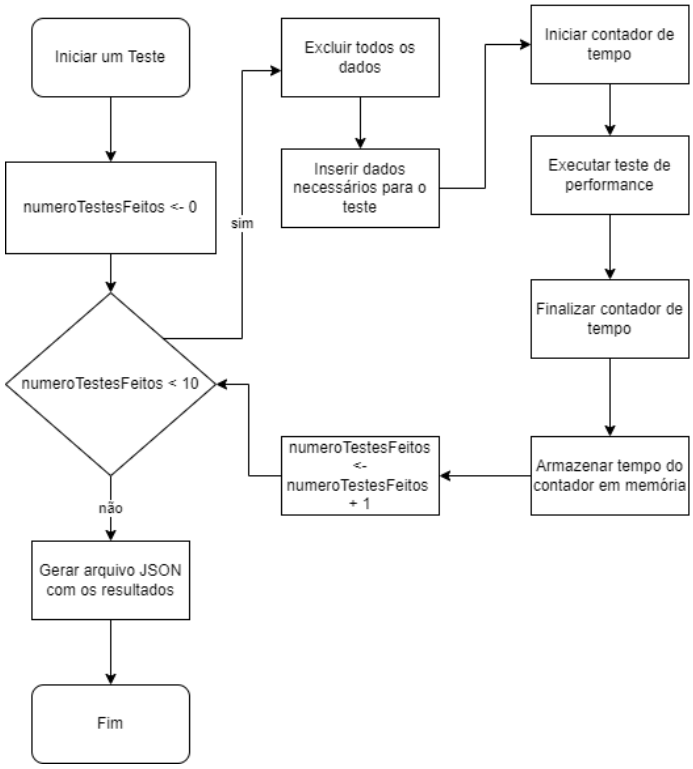
4.4 TESTES DE PERFORMANCE

Utilizando dos dados de municípios e estados do Brasil, visando contemplar todas as operações de CRUD, bem como o relacionamento entre as duas bases de dados, foram criados os seguintes testes:

- Inserir todos os municípios;
- Buscar todos os municípios;
- Buscar um município pelo nome;
- Alterar o nome de um município, informando o nome antigo e o novo;
- Excluir um município informando o nome;
- Buscar o estado informando o nome do município.

Para a execução de cada teste, seguiu-se este fluxo:

Figura 1 – Fluxo para a execução de um teste



Fonte: O autor

Para cada um dos seis testes em um total de dez vezes, todas as tabelas e estruturas do SGBD são limpas, os dados necessários para a realização das análises são inseridos, um contador de tempo é iniciado e, ao final da análise, é obtido o tempo total gasto em milissegundos, em seguida é armazenado esse valor internamente. Após as dez execuções, um arquivo no formato JSON é gerado contendo a quantidade de execuções realizadas para cada teste (dez por padrão), o tempo de cada execução e a média de tempo gasto.

4.5 INSTALAÇÃO E INICIALIZAÇÃO DOS SGBDs

Tanto o PostgreSQL quanto o MongoDB foram criados e executados com o Docker. Através da ferramenta Docker Compose (um facilitador para a criação e controle de contêineres com o Docker), ambos os SGBDs são criados, configurados e inicializados.

O Docker e o Docker Compose foram escolhidos pela facilidade de criação e remoção dos SGBDs utilizados de forma limpa e rápida. Com o comando *docker-compose up*, ambos os

contêineres são inicializados com suas respectivas configurações e imagens dos SGBDs e, com o comando *docker-compose down*, todos os contêineres são excluídos.

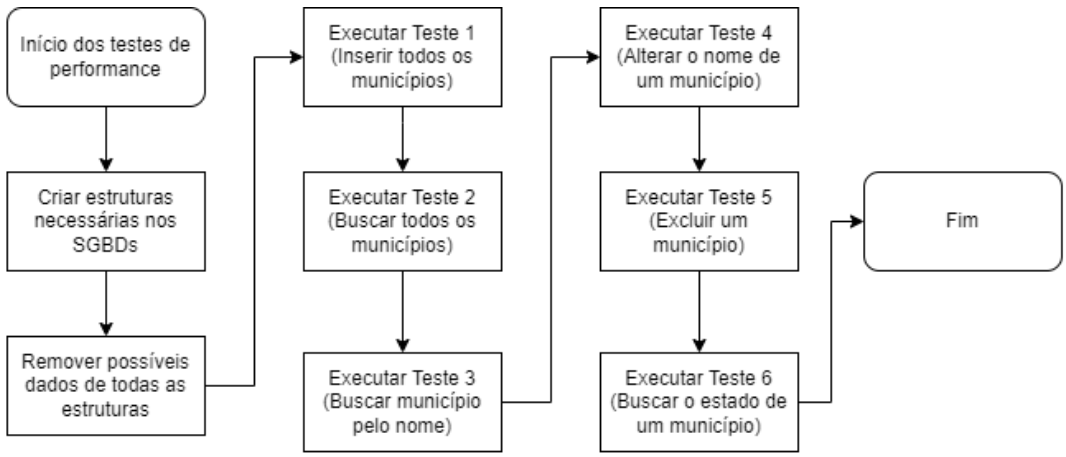
4.6 DESENVOLVIMENTO COM NODEJS E TYPESCRIPT

Todo o desenvolvimento do ambiente de testes foi feito com a linguagem de programação Typescript, utilizando-se do interpretador javascript NodeJS.

Para que não ocorressem impactos na performance durante a comunicação com os SGBDs, foram utilizados os *drivers* nativos *pg* para o PostgreSQL e *mongodb* para o MongoDB.

Ao executar o ambiente em Typescript, através do comando *yarn start* ou *npm start*, todas as estruturas necessárias são criadas, os dados dos SGBDs são excluídos e, por fim, os seis testes são executados seguindo o fluxograma:

Figura 2 – Fluxo de execução de todo o ambiente de testes



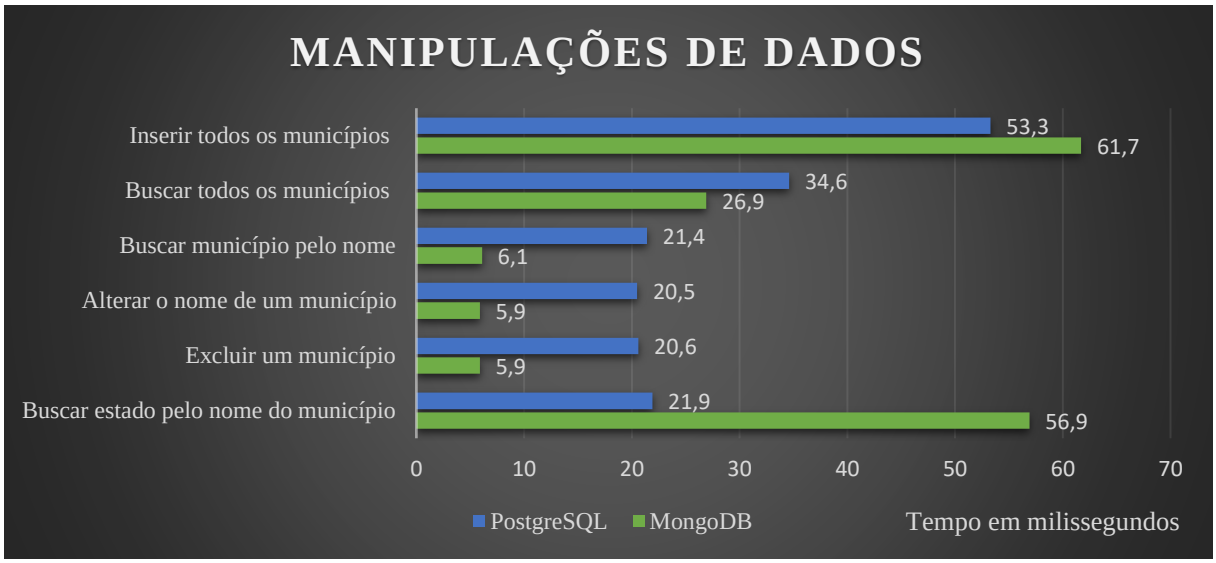
Fonte: O autor

Após a execução dos testes, dentro do diretório “resultados”, um arquivo para cada teste será criado no formato JSON, contendo a quantidade de vezes em que cada teste foi executado (por padrão, dez vezes), uma descrição do tipo do teste, os resultados de cada execução e a média de todas as execuções (em milissegundos).

Todo o código, bem como as instruções para a execução dos testes podem ser encontrados em <https://github.com/ThiagoDamaceno/TCC>.

5 RESULTADOS E DISCUSSÃO

Tabela 1 – Resultados das análises de performance



Fonte: O autor

Inseridos os cinco mil quinhentos e setenta municípios do Brasil, no primeiro teste, o PostgreSQL gastou um total de 53,3 milissegundos, ficando um pouco melhor do que o MongoDB, que levou 61,7 milissegundos. Para este volume de dados, o PostgreSQL possuiu uma leve vantagem, mas sem diferenças tão impactantes.

Consultando os cinco mil quinhentos e setenta municípios, o MongoDB, com 26,9 milissegundos, teve uma leve vantagem ao PostgreSQL, com 34,6 milissegundos. O MongoDB levou vantagem na busca sem filtros dos dados, embora em aplicações reais não seja recomendada a consulta de todos os dados de uma única vez.

Ao envolver uma condição de comparação no comando de busca, o MongoDB demorou 6,1 milissegundos para completar a operação, já o PostgreSQL demorou 21,4 milissegundos. Envolvendo um filtro na busca, no caso o nome do município, o MongoDB é mais performático do que o PostgreSQL.

Na alteração do nome de um município, também envolvendo uma condição de filtro, o MongoDB demorou 5,9 milissegundos e o PostgreSQL, 20,5 milissegundos. Seguindo na mesma linha do teste anterior, tratando-se de buscas com condições na mesma estrutura, ou seja, sem relacionamentos, o MongoDB é superior.

Para excluir um município, também com um filtro, o MongoDB gastou, igualmente ao teste anterior, 5,9 milissegundos; e o PostgreSQL, 20,6 milissegundos. Reforçando novamente os dois testes anteriores, o MongoDB foi mais eficiente nestes cenários.

No último teste, buscando um estado através do nome do município, ou seja, envolvendo um relacionamento entre as duas estruturas, o MongoDB levou 56,9 milissegundos e o PostgreSQL, 21,9 milissegundos. Agora o cenário é invertido, tratando-se de relacionamento entre diferentes estruturas, o PostgreSQL é extremamente performático, ao contrário do MongoDB.

6 CONCLUSÃO

Após os resultados obtidos nas análises de performance, concluo que o PostgreSQL, como o esperado de um SGBD relacional, é mais performático para realizar consultas com dados relacionados, já o MongoDB se sobressai em operações que necessitam aplicar algum filtro nos dados.

A escolha de qual SGBD utilizar será de acordo com as especificidades de cada aplicação. Na maioria dos cenários, envolvendo uma necessidade baixa ou média de performance, o PostgreSQL é mais versátil, tendo em vista que a probabilidade de se armazenar dados relacionados é alta. Aplicações envolvendo *big data* e ciências de dados necessitam de uma alta performance para buscar e filtrar grandes volumes de dados, cenários em que o MongoDB é mais aconselhado.

Uma terceira abordagem é de utilizar ambos. Necessitando de dados com relacionamentos e uma performance elevada, a utilização do MongoDB e do PostgreSQL juntos pode ser a ideal, no entanto, este cenário considera apenas performance, o emprego de mais de um SGBD em uma aplicação tende a aumentar a complexidade e custos com infraestrutura.

REFERÊNCIAS

CRUD. Disponível em <https://developer.mozilla.org/pt-BR/docs/Glossary/CRUD>. Acesso em: 08 ou 2021.

DE SOUZA, Alexandre Morais; PRADO, Edmir P. V.; SUN, Violeta; FANTINATO, Marcelo. **Critérios para Seleção de SGBD NoSQL**: o Ponto de Vista de Especialistas com base na Literatura. *In*: SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO (SBSI), 10, 2014, Londrina. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2014. p. 149-160. DOI: <https://doi.org/10.5753/sbsi.2014.6109>. Acesso em 02 set, 2021.

ELMASRI Ramez, NAVATHE Shamkant. **Sistemas de Banco de Dados.** São Paulo: Pearson Addison Wesley, 2005.

HOWS, David; MEMBREY, Peter; PLUGGE, Eelco. **Introdução ao MongoDB**. São Paulo: Novatec Editora Ltda. 2019.

IBGE - API de Localidades: Municípios. Disponível em <https://servicodados.ibge.gov.br/api/docs/localidades#api-Municipios-municipiosGet>. Acesso em: 10 set 2021.

IBGE - API de Localidades: UFs. Disponível em <https://servicodados.ibge.gov.br/api/docs/localidades#api-UFs-estadosGet>. Acesso em: 10 set 2021.

JOSE, B; ABRAHAM, S. **Performance analysis of NoSQL and Relational databases with MongoDB and MySQL**. Materials today: PROCEEDINGS, 2020, pp 2036-2043. DOI: <https://doi.org/10.1016/j.matpr.2020.03.634>. Acesso em: 24 set. 2021.

MongoDB Documentation. Disponível em < https://docs.mongodb.com/>. Acesso em: 01 set 2021.

NIELD, Thomas. **Introdução a linguagem SQL: Abordagem prática para iniciantes**. São Paulo: Novatec Editora Ltda, 2016.

Overview of Docker Compose. Disponível em < https://docs.docker.com/compose/>. Acesso em 25 ago 2021.

PostgreSQL Documentation. Disponível em < https://www.postgresql.org/docs/>. Acesso em: 04 set 2021.

RFC 4627: The application/json Media Type for Javascript Object Notation (JSON). Disponível em <https://www.rfc-editor.org/info/rfc4627>. Acesso em: 08 ou 2021.

ROCKENBACH, Dinei André et al. Estudo Comparativo de Bancos de Dados NoSQL. **Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação**, [S.l.], v. 1, n. 8, abr. 2018. ISSN 2446-7634. Acesso em: 02 set. 2021. DOI: <https://doi.org/10.5281/zenodo.1228503>.

Typescript Documentation. Disponível em <https://www.typescriptlang.org/docs/>. Acesso em 25 ago 2021.

Y. Li and S. Manoharan. **A performance comparison of SQL and NoSQL databases**. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), 2013, pp. 15-19. DOI: <https://doi.org/10.1109/PACRIM.2013.6625441>. Acesso em: 24 set. 2021.