**Centro Universitário Salesiano**

Sistemas de Informação

**Guilherme Henrique da Silva**

**Sistema WEB de Alta Disponibilidade com Docker, Nginx e MongoDB**

VITÓRIA

2024

# Introdução

Foi criado um sistema WEB de alta disponibilidade que realiza operações de CRUD (Create, Read, Update, Delete) em um banco de dados MongoDB. Foi utilizado algumas tecnologias para garantir a escalabilidade de gerenciamento do sistema:  
Docker: Para separar em conteiners;  
Docker Compose: Para orquestração dos contêiners;  
Nginx: Para balanceamento de carga;  
Node.js (v16.13.0): Para implementação do serviço web que realiza o CRUD;  
MongoDB: Para armazenamento de dados;  
Swagger: Para documentação das APIs;

# Desenvolvimento

O projeto está organizado na seguinte estrutura: Uma pasta principal nomeada de “Projeto CRUD” contendo o Docker-compose, nginx.conf e uma pasta nomeada “web”.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

O arquivo Docker-compose.yml na construção do projeto cria e gerencia múltiplos serviços que compõem a aplicação, garantindo que todos os componentes necessários sejam executados de maneira integrada. O Docker composse permite que todos esses serviços sejam iniciados e gerenciados com um único comando (Docker-compose up -d). Segue a imagem do arquivo abaixo:

Texto

Descrição gerada automaticamente

O arquivo nginx.conf é responsável pela configuração do Nginx como um balanceador de carga, distribuindo uniformemente o tráfego entre as instâncias do serviço web, garantindo alta disponibilidade da aplicação.Texto

Descrição gerada automaticamente  
O bloco “usptream” define um grupo de servidores chamado web\_service. A lista contém as três instâncias do serviço web (web1, web2 e web3), todas acessando a porta 3000 dentro de seus respectivos contêiners.  
Já o bloco server utiliza a porta padrão para HTTP: 80. Redireciona todas as solicitações recebidas para o grupo de servidores (upstream) através do proxy\_pass. Isso permite que o Nginx distribua o tráfego entre as instâncias. Já os proxy\_headers guardam informações sobre a solicitação original, como endereço de ip do cliente, para manter a integridade das solicitaçoes.

O serviço web foi implementado em Node.js na pasta web. O arquivo index.js configura o servidor Express e as rotas CRUD, além de integrar o Swagger para documentação das APIs:

Texto

Descrição gerada automaticamente  
Foi usado “express.json()” para analisar requisições JSON, Swagger UI para definir a rota “/api-docs” e servir a documentação da API, swaggerDocument para carregar o swagger.json que contém a especificação OpenAPI e mongoose.connect que conecta ao serviço MongoDB. As rotas CRUD estão definidas em GET, POST, PUT e DELETE/items e o srvidor na porta 3000.

O Dockerfile define como construir a imagem Docker para o serviço web. Ele especifica o ambiente necessário, as dependências, e os passos para configurar e iniciar o servidor Node.js:

Tela de computador com letras brancas em fundo preto

Descrição gerada automaticamente  
Esse dockerfile garante que o ambiente de execução seja independente da máquina onde o contêiner será executado.

O arquivo swagger.json foi construído a partir do Swagger Editor (editor.swagger.io) e documenta as rotas da API. Ele é usado para descrever a estrutura da API, incluindo os endpoints disponíveis, os métodos HTTP, os parâmetros de entrada/saída e as respostas esperadas:Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

# Conclusão

Foi desenvolvido um sistema web de alta disponibilidade utilizando as tecnologias mencionadas incluindo o balanceamento de carga com Nginx e a documentação com Swagger. É possível clonar o repositório a partir do github: <https://github.com/Guil-Henrique/Projeto-CRUD>. Para rodar a aplicação é necessário atender alguns requisitos:  
Docker instalado;  
Node.js (Foi utilizado v16.13.0);  
Instalar as dependências do serviço web: cd web / npm install  
Certificar-se que as portas não estão em uso por outra aplicação  
Iniciar os contêners: Docker-compose up -d