Big Data e NoSQL

Projeto Final

Lucas Rocha Dantas | 2311204
Renato Portilho Costa | 24101301
Pedro Rafael Santos Pignata | 24101218
Pedro Pereira Dutra | 24101026

https://github.com/LucasRD3105/projeto-final

Introdução

Este relatório apresenta o desenvolvimento de uma solução completa para gestão de estoque utilizando conceitos e ferramentas aplicadas no contexto de Big Data e aplicações modernas baseadas em containers. O objetivo do projeto é demonstrar a criação de um sistema CRUD (Create, Read, Update, Delete) para gerenciamento de produtos, integrando banco de dados NoSQL, backend em Python e interface de visualização interativa

O ambiente foi estruturado com Docker para padronização e portabilidade, MongoDB como banco NoSQL para armazenar os registros, Flask para disponibilização de endpoints e Streamlit para construção de uma interface visual com dashboards e métricas analíticas. A solução permite visualizar o estoque, analisar indicadores em tempo real e realizar operações de cadastro, edição e remoção de itens

Objetivo do projeto

O objetivo principal é disponibilizar uma plataforma funcional que permita:

- Gerenciar produtos de um acervo (CRUD completo)
- Visualizar métricas do estoque em dashboards interativos
- Usar tecnologias relevantes na área de Big Data, como NoSQL e ambientes containerizados
- Demonstrar integração entre backend, banco de dados e interface visual
- Garantir fácil replicação do ambiente por meio de Docker

Tecnologias Utilizadas

Docker
Utilizado para construir e executar containers, garantindo que o ambiente do projeto se mantenha consistente independentemente da máquina onde é executado. O container inclui o backend e o Streamlit, enquanto o MongoDB também roda em container próprio via docker-compose

MongoDB
Banco de dados NoSQL orientado a documentos, ideal para projetos de Big Data pela sua escalabilidade e flexibilidade de schema. Utilizado para armazenar os registros de produtos, contendo: nome categoria quantidade preço

Python + Flask
Flask foi a base inicial para construção de endpoints e manipulação do banco, fornecendo suporte a operações CRUD. Embora o fluxo principal do CRUD esteja integrado diretamente no Streamlit, o Flask compõe a estrutura do backend do projeto

Streamlit
Framework utilizado para construir a interface interativa (frontend). Permite criação de dashboards com métricas do acervo, exibição de tabelas, gráficos e operações CRUD com experiência simplificada

Plotly
Biblioteca utilizada para gerar gráficos dinâmicos no dashboard, como: gráfico de barras para distribuição de produtos, gráfico de rosca para categorias

Pandas
Empregado para manipulação dos dados recebidos do MongoDB e para exportação de relatórios CSV

Arquitetura da solução

Backend / API
- Gerenciamento das conexões com MongoDB
- Execução de consultas, inserções e atualizações
- Disponibilização de lógica da aplicação

Banco de Dados (MongoDB)
- Armazena documentos JSON correspondentes aos produtos
- Fornece consultas rápidas para operações CRUD
- Permite escalabilidade, característica importante em cenários de Big Data

Interface (Streamlit Dashboard)
- Exibe métricas do sistema: quantidade total de itens, valor acumulado em estoque e total de categorias
- Apresenta gráficos interativos
- Permite operações de cadastro, edição, exclusão e visualização completa dos dados
- Gera relatórios CSV para download

Desenvolvimento do projeto

O desenvolvimento iniciou-se com a definição do escopo do sistema: criar uma aplicação CRUD para gerenciamento de produtos, com integração visual e banco NoSQL escalável. As etapas a seguir descrevem todo o fluxo técnico que compõe o projeto

Preparação do Ambiente

Foi criado um arquivo Dockerfile contendo as configurações do container principal. Esse container inclui:

- Python configurado com as bibliotecas necessárias
- Instalação de dependências a partir do requirements.txt
- Configuração para executar o Streamlit como serviço principal

Além disso, o arquivo docker-compose.yml foi criado para orquestrar:

- o container da aplicação
- o container do banco MongoDB

Isso permite que os serviços se comuniquem corretamente, garantindo escalabilidade e isolamento do ambiente

Implementação do Banco de Dados

O banco MongoDB foi configurado com uma coleção chamada produtos, dentro do banco estoque_db. Cada documento segue um modelo simples e flexível, características típicas de bancos NoSQL, permitindo rápida manipulação e consultas eficientes mesmo com grande volume de dados

Desenvolvimento da Aplicação em Python

A aplicação central foi desenvolvida em Python utilizando:
Pymongo: para manipulação do MongoDB
Pandas: para manipulação dos dados antes de apresentar
Plotly: para geração de gráficos
Streamlit: para construção do dashboard interativo

O arquivo app.py concentra toda a lógica de visualização e CRUD
Ele implementa três áreas principais:
Dashboard Geral

- Total de itens no estoque
- Valor total acumulado
- Categorias registradas
- Gráfico horizontal com os produtos e quantidades
- Gráfico de rosca representando categorias
- Exibe a tabela completa com todos os itens
- Permite exportar relatório em CSV

Cadastro de Itens

- Permite inserir produtos com nome, categoria, quantidade e preço
- Verifica duplicidade para evitar registros repetidos

Edição e Exclusão de Produtos

- Permite selecionar item existente
- Alterar quantidade e preço
- Excluir o item permanentemente

Integração com Docker

Após finalizar a aplicação, os containers foram executados, gerando um ambiente completamente funcional e reprodutível, permitindo que qualquer pessoa execute o sistema sem necessidade de instalar manualmente todas as dependências

Execução do projeto

Para executar o projeto é necessário ter o Docker e o Docker Compose instalados, além do acesso ao diretório do repositório no GitHub. Após realizar o clone do projeto por meio do comando git clone https://github.com/LucasRD3105/projeto-final, basta acessar o diretório e iniciar os serviços com docker-compose up --build, permitindo que os containers da aplicação e do MongoDB sejam construídos e executados automaticamente. Todo o ambiente é configurado de forma padronizada graças ao uso do Docker, eliminando a necessidade de instalação manual das dependências. Uma vez que os containers estejam ativos, a aplicação pode ser acessada diretamente pelo navegador através do endereço http://localhost:8501, onde estarão disponíveis o dashboard geral com métricas do estoque, o sistema CRUD completo integrado ao banco de dados e todas as funcionalidades de visualização, edição, cadastro e exportação de relatórios

Conclusão

O projeto desenvolvido demonstra a aplicação prática de conceitos fundamentais de Big Data utilizando tecnologias modernas e escaláveis. A combinação entre MongoDB, Docker e Streamlit proporciona uma solução completa, replicável e funcional para gerenciamento de estoque, oferecendo tanto uma API estruturada quanto uma interface intuitiva de análise e visualização

A utilização de um banco NoSQL permite flexibilidade e eficiência no armazenamento dos dados, enquanto Docker garante a padronização do ambiente, facilitando implantação e manutenção. Além disso, Streamlit contribui com uma interface de alto nível, permitindo que o usuário visualize métricas, gráficos e relatórios de maneira clara e objetiva

O sistema atinge os objetivos propostos e demonstra uma abordagem consistente, integrando backend, banco de dados e frontend em um único fluxo operacional coeso, sendo uma solução aplicável em diferentes cenários dentro da área de Big Data e gestão de informações