CENTRO UNIVERSITÁRIO DE BRASÍLIA – CEUB

Ciência de Dados e Machine Learning

Fundamentos de Big Data

RELATÓRIO TÉCNICO PROJETO FINAL

SOLUÇÃO INTEGRADA COM DOCKER, STREAMLIT E MONGODB

DATASET OLIST

Integrantes:

Gabriel Francisco – 22408417

Guilherme Mendes Carlos – 22408949

Mariana Almeida – 22401569

Pedro Rebello Borges de Barros – 22405550

Brasília – 2025

Sumário

1. Introdução

Este relatório apresenta uma solução prática voltada à construção de um ambiente integrado composto por uma aplicação Streamlit e um banco de dados MongoDB, ambos executados em contêineres Docker. Foi utilizado como base de dados o **Dataset Olist**, um conjunto real de informações de vendas do comércio eletrônico brasileiro. O projeto aplica conceitos fundamentais de estruturação de ambiente, deploy automatizado, ingestão de dados e visualização interativa, utilizando ferramentas amplamente empregadas em projetos modernos de dados e aplicações web.

2. Propósito do Projeto

O objetivo do projeto foi construir um ambiente totalmente reprodutível, capaz de:

- Carregar automaticamente dados de um dataset real (Olist) no MongoDB.
- Disponibilizar uma aplicação Streamlit para visualização interativa.
- Organizar todos os componentes em contêineres Docker, garantindo fácil execução em qualquer máquina.
- Criar um fluxo automatizado com scripts Bash para inicialização e gerenciamento dos serviços.

3. Arquitetura Adotada

Foi utilizada uma arquitetura baseada em serviços desacoplados, cada qual em seu contêiner, conectados através de uma rede Docker interna.

Componentes da arquitetura:

- Aplicação Streamlit (frontend interativo)
- MongoDB (armazenamento dos dados)
- Mongo Express (ferramenta web de administração do banco)
- Rede Docker personalizada (mybridge) para comunicação estável
- Scripts Bash para automação de execução

A arquitetura garante modularidade, isolamento e fácil manutenção.

4. Componentes Utilizados

- Python 3.9-slim (imagem base da aplicação)
- Streamlit (interface gráfica web)
- Plotly (visualizações interativas)
- Pandas (leitura e transformação do dataset)
- pymongo (conexão com o MongoDB)
- MongoDB (banco NoSQL orientado a documentos)
- Mongo Express (gerenciamento do banco via navegador)
- Docker e Docker Compose (orquestração de serviços)
- wait-for-it.sh (controle de dependências entre serviços)

5. Desenvolvimento e Implementação

5.1 Estruturação do Diretório

O repositório foi dividido logicamente para facilitar manutenção:

/mongodb (configuração do banco e rede)

/streamlit (aplicação, Dockerfile e dataset)

/scripts start.sh, stop.sh e utilidades

docker-compose.yml

5.2 Auto-Seeding dos Dados (ETL)

Para eliminar a necessidade de carga manual, foi implementado no **app.py** um processo automático que:

1. Conecta-se ao MongoDB usando pymongo.
2. Verifica se a coleção está vazia.
3. Caso positivo, lê o arquivo dataset_final_simple.csv via Pandas, converte para dicionários e insere os dados com insert_many.

Esse procedimento garante que o ambiente esteja pronto na primeira execução.

5.3 Configuração de Redes e Contêineres

Foi criada a rede personalizada: mybridge

Essa rede permite que os contêineres se comuniquem usando nomes estáveis (ex.: mongodb:27017).
O Dockerfile da aplicação foi otimizado usando python:3.9-slim, reduzindo o tamanho da imagem.

5.4 Automação de Execução

O script **start.sh** realiza:

- Limpeza de arquivos antigos
- Cópia de versões atualizadas
- Gerenciamento de permissões
- Iniciação controlada via: docker-compose up -d

Também foi integrado o utilitário wait-for-it.sh, que aguarda o MongoDB estar disponível antes de inicializar o Streamlit, prevenindo erros por dependências não prontas.

6. Como Executar o Ambiente em Contêiner

O projeto foi estruturado para permitir a execução completa do ambiente com um único comando, por meio do script automatizado start.sh. Este script cria a rede necessária, inicializa o MongoDB e o Mongo Express, aguarda a disponibilidade do banco de dados e, por fim, inicia a aplicação Streamlit. Embora a execução automática seja a forma recomendada de utilização, a execução manual também está descrita abaixo

6.1 Guia de Uso Automático

A execução automática é a forma mais simples de iniciar o projeto. Este método já realiza todas as etapas necessárias: clona os repositórios, ajusta permissões, acessa os diretórios corretos e inicia o script responsável por subir todos os serviços (MongoDB, Mongo Express e Streamlit).

Etapas explicadas:

1. Clonar o repositório base contendo a estrutura utilizada como referência.
2. Ajustar permissões para garantir que o diretório possa ser manipulado.
3. Acessar a pasta base onde os projetos serão organizados.
4. Clonar o repositório do projeto final, que contém o dashboard e configurações.
5. Entrar no diretório do projeto para ter acesso ao script de inicialização.
6. Executar o script start.sh, que prepara o ambiente e inicia todos os contêineres.

Comandos:

```
git clone https://github.com/klaytoncastro/ceub-bigdata.git
chown -R labihc ceub-bigdata
cd ceub-bigdata
git clone https://github.com/0GabrielF0/Trabalho-Final-BD.git
cd Trabalho-Final-BD
bash start.sh
echo ""
```

6.2 Guia de Uso Manual

A execução manual detalha o processo, permitindo visualizar como o ambiente é preparado passo a passo.

Etapas explicadas:

1. Preparar o ambiente inicial, clonando o repositório base e acessando o diretório correto.
2. Assumir privilégios administrativos, caso seja necessário executar comandos de manutenção.
3. Ajustar permissões para permitir manipulação adequada do repositório.
4. Acessar o diretório base onde as pastas serão organizadas.
5. Clonar o repositório do projeto final, que contém os scripts e a aplicação.
6. Entrar no diretório do projeto, onde estão os arquivos essenciais.
7. Executar o script start.sh, iniciando todos os serviços e carregando o dataset automaticamente.

Comandos

```
git clone https://github.com/klaytoncastro/ceub-bigdata.git
sudo su -
chown -R labihc ceub-bigdata
cd ceub-bigdata
git clone https://github.com/0GabrielF0/Trabalho-Final-BD.git
cd Trabalho-Final-BD
bash start.sh
```

6.3 Estrutura do Repositório

A organização do repositório segue a seguinte estrutura:

- start.sh: script de automação responsável pela criação da rede, inicialização dos serviços e sincronização de dependências.
- stop.sh: script utilizado para encerrar todos os serviços de forma ordenada.
- mongodb/: contém o arquivo docker-compose.yml que define o MongoDB e o Mongo Express, além do utilitário wait-for-it.sh, responsável por aguardar a disponibilidade do banco.
- streamlit/: contém o arquivo app.py (dashboard) e a pasta data_processed com o arquivo dataset_final_simple.csv utilizado no carregamento inicial.

6.4 Acesso aos Serviços

Após a execução do start.sh, os serviços estarão acessíveis pelos seguintes endereços:

- Dashboard Streamlit: http://localhost:8501
- Mongo Express: http://localhost:8081 (credenciais: User: admin / Pass: pass)

6.5 Parar o Ambiente

Para encerrar todos os contêineres de forma segura, utilize:

bash stop.sh

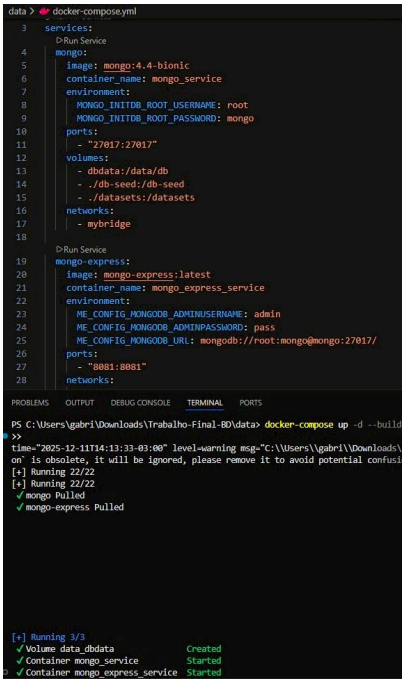
Esse comando encerra o banco de dados, o Mongo Express e a aplicação.

7. Evidências de Funcionamento

Esta seção apresenta as evidências visuais de que o ambiente foi executado corretamente, demonstrando o funcionamento dos serviços em contêiner, o carregamento da aplicação e o acesso à interface gráfica do dashboard.

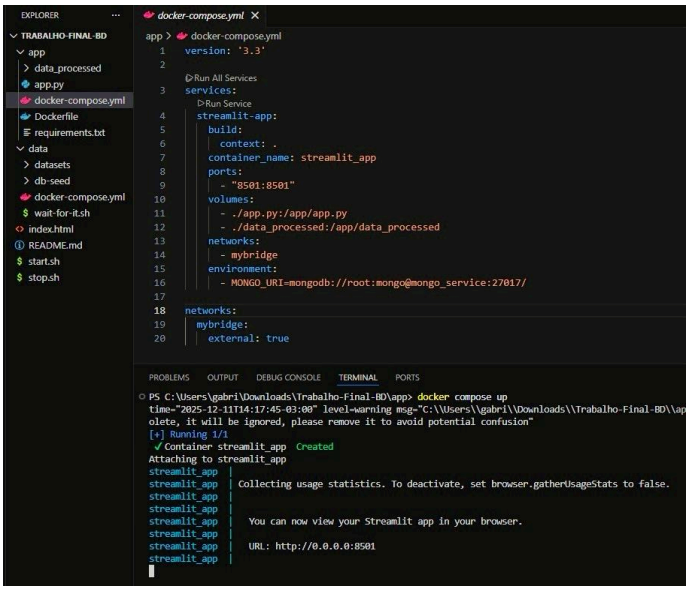
7.1 Evidência do MongoDB e Mongo Express em Execução

A primeira imagem demonstra o funcionamento do serviço de banco de dados. No terminal, é possível observar que os contêineres *mongo_service* e *mongo_express_service* foram criados e iniciados corretamente. A configuração do arquivo *docker-compose.yml* também confirma que o banco foi inicializado com sucesso, bem como a porta 8081 destinada ao Mongo Express.



7.2 Evidência do Streamlit em Execução

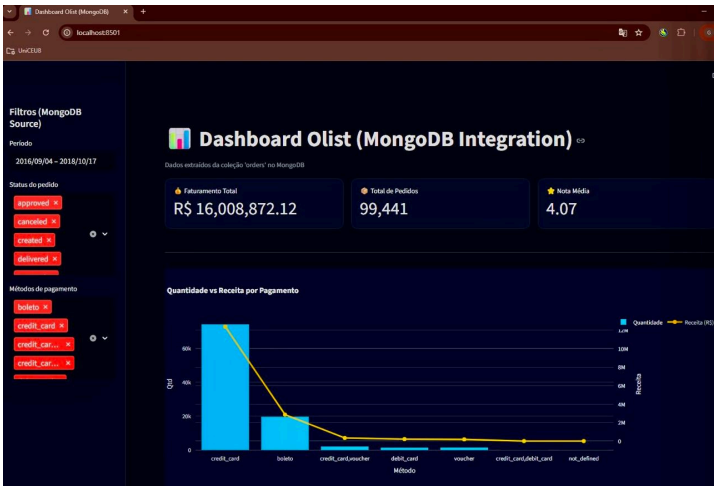
A segunda imagem mostra a inicialização da aplicação Streamlit. O terminal exibe claramente que o contêiner *streamlit_app* foi criado e iniciado, indicando a porta padrão de execução (8501). A mensagem “You can now view your Streamlit app in your browser” confirma que o serviço está ativo e disponível para acesso.



7.3 Evidência da Interface do Dashboard Funcionando

A terceira imagem comprova o funcionamento do dashboard acessado por meio do endereço http://localhost:8501. A tela apresenta os indicadores, filtros e gráficos carregados corretamente, evidenciando que:

- A aplicação Streamlit foi iniciada com sucesso;
- Os dados foram carregados a partir do MongoDB;
- A comunicação entre os contêineres está funcionando plenamente.



8. Conclusão

A solução desenvolvida cumpre integralmente os requisitos propostos, permitindo a execução integrada de uma aplicação Streamlit com um banco MongoDB dentro de contêineres Docker. O ambiente é reprodutível, modular, automatizado e contém mecanismos internos de carga de dados. Os desafios encontrados, relacionados a portas, rede e dependências, foram solucionados com ajustes técnicos adequados, permitindo que todo o sistema funcione de forma estável e acessível.