

**UNIVERSIDADE SALVADOR – UNIFACS**

**Tecnologia da Informação**

**Relatório Avaliação 3**

**Loja de Ferramentas**

Alisson Rayan Santos de Souza – RA: 1272314418

Davi dos Reis da Fonseca Ramos – RA: 1272316509

George Gebers Brizolla – RA: 12724113504

Júlio César de Jesus Barreto Souza – RA: 12723120855

Wesley Cristian Carvalho Dantas - RA: 1272311443

**Salvador - BA**

**2024**

# **UNIVERSIDADE SALVADOR – UNIFACS**

## **Tecnologia da Informação**

### **Relatório Avaliação 3**

#### **Loja de Ferramentas**

Trabalho apresentado ao curso de Ciências da Computação da UNIFACS, como requisito parcial para a aprovação na disciplina de Sistemas Distribuídos e Mobile, ministrada pelo professor Adailton Cerqueira.

Alisson Rayan Santos de Souza – RA: 1272314418

Davi dos Reis da Fonseca Ramos – RA: 1272316509

George Gebers Brizolla – RA: 12724113504

Júlio César de Jesus Barreto Souza – RA: 12723120855

Wesley Cristian Carvalho Dantas - RA: 1272311443

**Salvador - BA**

**2024**

## SUMÁRIO

Introdução .....	4
Fundamentação Teórica.....	5
> React .....	5
> Express .....	5
> Docker .....	5
> Banco de Dados.....	5
Projeto de Implementação .....	6
> Estrutura do Sistema .....	6
> Containerização com Docker .....	7
Considerações Finais.....	8
Bibliografia .....	9

## Introdução

Nos últimos anos, o desenvolvimento de aplicações web requer métodos eficientes e eficazes para atender à tarefas diferentes e solicitações de manutenção. Para isso, surgiram tecnologias como React, Express e Docker, fornecendo certa facilidade ao construir sistemas novos e bem projetados. Neste projeto, essas tecnologias são combinadas para criar um aplicativo que inclui componentes dinâmicos de front-end, back-end e relatórios, todos executados em contêineres separados via Docker.

Essas ferramentas foram escolhidas pela eficiência e compatibilidade com os sistemas mais utilizados pelos devs. O React foi selecionado como front-end devido à sua capacidade de criar uma interface dinâmica e responsiva, essencial para uma experiência de usuário perfeita. Nos bastidores, o Express oferece flexibilidade para gerenciar processos e aplicativos de maneira fácil e eficiente. SQLite é usado como banco de dados devido à sua simplicidade e funcionalidade. Ideal para projetos que requerem armazenamento de curto prazo e fácil in. O Docker desempenha um papel importante na simplificação do desenvolvimento e implantação de ambientes, eliminando problemas de integração e garantindo que os aplicativos funcionem em diferentes sistemas.

O projeto possui três containers: um para o cliente (front-end), um para o servidor (back-end) e um terceiro dedicado à seção de referência. Este departamento é responsável pela administração do sistema, desenvolvimento operacional e manutenção. Além disso, os contêineres são mais convenientes e flexíveis e podem ser usados em vários designs.

O objetivo principal é mostrar como novas tecnologias podem ser combinadas para criar sistemas bem integrados e desenvolvidos sistematicamente. Durante o desenvolvimento, encontramos muitos problemas, como implementação inicial de contêineres, integração entre componentes, etc., mas esses problemas foram resolvidos imediatamente por meio de alterações de configuração e testes. Por fim, este projeto mostra como utilizar as ferramentas disponíveis no mercado para criar uma solução poderosa e eficaz.

## Fundamentação Teórica

### > React

O React é uma biblioteca JavaScript amplamente utilizada para a criação de interfaces de usuário. Ele permite o desenvolvimento de componentes reutilizáveis, que facilitam a manutenção e a escalabilidade do sistema. A escolha do React para o frontend deve-se à sua eficiência no gerenciamento de estados e à ampla comunidade que oferece suporte e bibliotecas complementares.

### > Express

O Express é um framework minimalista para Node.js, projetado para construir APIs robustas e simples. Foi utilizado tanto no backend principal quanto no módulo de relatórios, oferecendo flexibilidade na criação de rotas e middleware. Sua popularidade e compatibilidade com bibliotecas de terceiros foram fatores decisivos na escolha.

### > Docker

O Docker é uma plataforma de virtualização leve baseada em contêineres. Ele foi utilizado para criar três serviços isolados: cliente, servidor e módulo de relatório. A imagem base escolhida foi `node:alpine3.19`, devido ao seu tamanho reduzido e desempenho eficiente. A definição dos serviços no `docker-compose.yml` facilitou o gerenciamento e a comunicação entre os contêineres.

### > Banco de Dados

Um banco de dados SQLite foi integrado ao servidor, utilizando um volume Docker para garantir a persistência dos dados entre reinicializações. A escolha pelo SQLite foi motivada por sua simplicidade e pela ausência de necessidade de um servidor de banco de dados separado, tornando-o ideal para projetos pequenos e médios.

## Projeto de Implementação

### > Estrutura do Sistema

#### 1. Cliente (React):

Construído com React, o cliente é responsável pela interface com o usuário. Ele consome APIs do servidor para exibir e manipular dados.

<http://localhost:3000>

#### 2. Servidor (Express):

Gerencia o banco de dados e fornece endpoints RESTful para o cliente. Inclui rotas para criar, ler, atualizar e excluir (CRUD) dados no SQLite.

<http://localhost:5000>

[] Os endpoints do servidor são:

- /api/produtos
- /api/clientes
- /api/vendedores
- /api/vendedores/{id do vendedor}/produtos
- /api/pedidos

#### 3. Relatório (Express):

Um serviço independente que acessa o banco de dados compartilhado para gerar relatórios com base nos dados existentes.

<http://localhost:4000>

[] Os endpoints do relatório são:

- /produtos-mais-vendidos
- /produtos-por-pedido
- /gasto-medio-por-cliente
- /baixo-estoque

## > Containerização com Docker

- O `docker-compose.yml` define três serviços:
  - **Cliente:** exposto na porta 3000.
  - **Servidor:** exposto na porta 5000.
  - **Relatório:** exposto na porta 4000, compartilhando o mesmo banco de dados com o servidor.

## > Passos para Inicialização

1. Clonar o repositório

```
git clone https://github.com/deveju/A3-SDM-2024.2-DOCKER.git
```

2. Ir até o repositório clonado

```
cd A3-SDM-2024.2-DOCKER/A3_2024.2-Docker
```

3. Subir os serviços:

```
docker-compose up
```

4. Autenticação front-end:

User: `admin` | Senha: `admin`

Os contêineres serão inicializados automaticamente, com a comunicação entre cliente, servidor e relatório configurada via rede interna do Docker.

## **Considerações Finais**

O projeto apresentou resultados satisfatórios, destacando a integração eficaz entre os componentes. A aplicação foi capaz de atender aos requisitos funcionais e técnicos, aproveitando ao máximo a modularidade oferecida pelo Docker.

Desafios enfrentados incluíram a configuração de volumes para persistência de dados e a coordenação entre os serviços no Docker. Ambos foram superados com ajustes no `docker-compose.yml` e na documentação do projeto.



## Bibliografia

1. **Posgraduando** - “Como Escrever um Artigo Científico.”:  
<https://posgraduando.com/como-escrever-um-artigo-cientifico/>
2. **ReactJS** - Documentação: <https://react.dev/>
3. **ExpressJS** - Documentação: <https://expressjs.com/>
4. **Docker** - Documentação: <https://docs.docker.com/>
5. **Stack Overflow**: <https://stackoverflow.com>
6. **DevOps Directive**: <https://www.youtube.com/@DevOpsDirective>