O projeto foi estruturado em três grandes partes: **bancos de dados**, **APIs** e **interface front-end**. Todo o ambiente foi configurado utilizando **Docker**, o que garantiu consistência no desenvolvimento e facilidade de implantação.

Para os bancos de dados, foram utilizados **dois containers separados**, um com **MongoDB** e outro com **MySQL**, ambos criados por meio de um arquivo **docker-compose.yml.** O container do **MongoDB** foi configurado com a imagem oficial **mongo:latest**, exposto na porta **27018**, com usuário e senha definidos via variáveis de ambiente. Já o container do **MySQL** utilizou a imagem **mysql:latest**, com a porta interna **3306** mapeada para a porta **3307** do host, e também teve um banco de dados inicial criado automaticamente, com credenciais personalizadas. Para ambos os bancos, volumes Docker foram criados para garantir a **persistência dos dados**.

As **APIs foram desenvolvidas em Java utilizando Spring Boot e Lombok**, com uma API para cada banco. A primeira API foi conectada ao **MongoDB**, utilizando Spring Data Mongo. A entidade Pessoa, comum às duas APIs, possui os campos id, nome e telefone. Essa API realiza as operações básicas de CRUD e está devidamente configurada para se comunicar com o Mongo através do application.properties. A segunda API foi conectada ao **MySQL**, utilizando Spring Data JPA e o driver JDBC correspondente. Ambas as APIs foram empacotadas utilizando **Dockerfiles com múltiplos estágios**, com a intenção de criar imagens a partir do código-fonte. No entanto, **durante a criação das imagens e containers das APIs em Java, erros ocorreram na fase de inicialização**, e infelizmente não foi possível corrigir o problema a tempo. Por isso, as duas APIs foram **executadas diretamente dentro do IntelliJ IDEA**, o que permitiu que o desenvolvimento seguisse normalmente.

O **front-end foi desenvolvido em React**, utilizando Node.js como ambiente de execução. As dependências foram instaladas com npm install, e a aplicação foi iniciada localmente com npm start, rodando na porta **3000**. A comunicação com as APIs foi feita utilizando a biblioteca **axios**, permitindo o envio e a leitura de dados tanto da API do Mongo quanto da API do MySQL. A estrutura foi mantida clara, consumindo ambas as fontes de dados com facilidade.  
  
Ao final, foi criado um **docker-compose para subir simultaneamente os containers dos dois bancos de dados e das APIs Java**, a fim de consolidar todo o ambiente em um único comando. As **imagens foram adicionadas corretamente ao compose**, porém, **apenas o banco do Mongo funcionou como esperado**. O restante dos serviços, incluindo as APIs e o MySQL, **não executavam corretamente**, e apesar das tentativas de ajuste, **o problema não foi resolvido a tempo**.