

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Arquitetura de Software Distribuído

Raphael Silveira

LOGÍSTICA BASEADA EM MICROSERVIÇOS

Belo Horizonte

2023

Raphael Silveira

LOGÍSTICA BASEADA EM MICROSERVIÇOS

Trabalho de Conclusão de Curso de Especialização
em Arquitetura de Software Distribuído como
requisito parcial à obtenção do título de especialista.

Orientador(a): Luiz Alberto Ferreira Gomes

Belo Horizonte

2023

RESUMO

Este projeto apresenta uma proposta arquitetural baseada em cloud e orientada a microsserviços para o segmento logístico. A empresa Boa entrega necessita atualizar suas aplicações visando expansão de parcerias e negócios em todo o território nacional. O novo sistema não deve impactar nas atividades atuais da empresa e deve manter uma integração com as aplicações já implantadas e em funcionamento na companhia. A partir do levantamento de requisitos a arquitetura será modelada, e uma prova de conceito funcional de uma parte da aplicação será desenvolvida com a finalidade de avaliar o atendimento dos requisitos estipulados.

Palavras-chave: cloud, microsserviços, arquitetura de software

SUMÁRIO

1. Objetivos do trabalho.....	5
2. Descrição geral da solução	6
2.1. Apresentação do problema.....	6
2.2. Descrição geral do software (Escopo)	7
3. Definição conceitual da solução	9
3.1. Requisitos Funcionais	9
3.2 Requisitos Não-Funcionais	10
3.3. Restrições Arquiteturais	11
3.4. Mecanismos Arquiteturais	11
4. Modelagem e projeto arquitetural.....	12
4.1. Modelo de componentes	13
4.2. Modelo de implantação	17
5. Prova de Conceito (POC) / protótipo arquitetural	19
5.1. Implementação e Implantação	19
5.2. Código	26
6. Avaliação da Arquitetura.....	32
6.1. Análise das abordagens arquiteturais.....	32
6.2. Cenários	32
6.3. Avaliação.....	33
6.4. Resultado.....	49
7. Conclusão.....	51
REFERÊNCIAS.....	52
APÊNDICES.....	53

1. Objetivos do trabalho

A empresa Boa Entrega é uma transportadora de grande porte que atua no segmento logístico em todo o território nacional atendendo empresas do varejo. No atual contexto de pandemia as pessoas não podendo sair de suas casas tem se servido de sites de e-commerce para realizar as compras cotidianas. A alta demanda por este tipo de serviço incentivou muitas empresas a se especializarem em entregar essas mercadorias, disputando espaço nesse competitivo mercado.

O objetivo geral deste projeto é apresentar uma proposta arquitetural baseada em microsserviços para a Boa Entrega com o intuito de modernizar sua plataforma de aplicações, buscando maior segurança e robustez, proporcionando assim a expansão de suas parcerias de negócios.

Os objetivos específicos são:

- Empregar uma arquitetura orientada a serviços (SOA) permitindo o reuso de sistemas legados;
- Fazer uso de tecnologias baseadas em Cloud Computign possibilitando o acesso às tecnologias mais avançadas disponíveis no mercado;
- Utilizar o um modelo de banco de dados por microsserviço, permitindo o dimensionamento do seu próprio armazenamento conforme necessário e o isolamento a falhas dos outros serviços;
- Aplicar o conceito de orquestração para gerenciar a consistência de dados entre microsserviços no cenário de transações distribuídas;
- Criar um sistema modular e multiplataforma, que possibilite o acesso via web browser em computadores pessoais e por dispositivos móveis;
- Possibilitar a gestão logística através de controles de estoque, monitoramento de pedidos, acompanhamento de entregas e disponibilidade de cargas.

2. Descrição geral da solução

Esta seção se destina a descrever a solução arquitetural definida para a aplicação “Gestão de Serviços de Logística”.

2.1. Apresentação do problema

A Boa Entrega possui um catálogo tecnológico composto por soluções já implantadas, das mais variadas áreas de aplicação e em diferentes plataformas computacionais. Algumas destas soluções foram desenvolvidas pela TI da própria transportadora, outras foram desenvolvidas por empresas contratadas. A seguir, apresentam dados detalhados dos sistemas existentes:

- Sistema Administrativo-Financeiro (SAF): aplicação desenvolvida há alguns anos por empresa contratada, consistindo em uma solução de gestão administrativa e financeira completa e escalável. É um sistema legado baseado em tecnologias .Net Core e banco de dados SQL Server. Não atende todas as necessidades de negócio atuais por ser de difícil integração com outras aplicações e não oferece suporte à web e a dispositivos móveis;
- Sistema de Gestão de Entregas (SGE): solução web desenvolvida pela TI da própria Boa Entrega, suporta a gestão completa de todo o processo das entregas, desde a programação e roteamento até a gestão de estoques. Foi construído em módulos independentes, fazendo uso de alguns microsserviços para acesso às bases de dados locais. No desenvolvimento do back-end foi empregado PHP, já no front-end foram utilizadas as tecnologias HTML, JavaScript e Flutter. Possui uma integração de alto acoplamento com o SAF, considerando que este não oferece facilidades com integração;
- Sistema de Faturamento e Cobrança (SFC): software também desenvolvido pela equipe própria de TI, permitindo gerir todo o processo de faturamento e acompanhar a cobrança dos valores devidos junto aos clientes;
- Portal Corporativo: funcionalidades desenvolvidas na forma de componentes, acessíveis por meio de páginas web desenvolvidas em Javascript, HTML e CSS. Acessado por colaboradores e clientes mediante autenticação com usuário e senha, permite visualização seletiva dos principais recursos que necessitam, tais como: visualizar, editar e es-colher rotas

pré-existent, planejar e acompanhar entregas e verificar eventuais problemas com veículos em rota de entrega. Essas funcionalidades são utilizadas como serviços, muitos fazem parte do SGE, e podem ser acessados também por meio deste, nas interfaces web e móvel.

Para atender as metas e aos objetivos operacionais traçados, a Boa Entrega deve manter todos os registros de operações e realizações de entregas, bem como uma boa gestão financeira que se inicia na emissão do Conhecimento de Transporte Rodoviário, passando pelo faturamento e pela gestão de estoque dos armazéns onde as mercadorias ficam guardadas.

A Boa entrega visando aprimorar seus processos de gestão elaborou um plano de Metas e Diretrizes para a área de TI, definindo as seguintes prioridades para o próximo triênio:

- Automatizar todos os processos de entrega visando aprimorar a apuração, conferência e faturamento mantendo um nível de remuneração adequado;
- Implementar integrações entre seus sistemas e de suas parceiras propiciando que as entregas possam ser realizadas em parceria nas várias etapas do processo. Isto demanda que os sistemas atuais sejam adaptados e novos componentes incorporados baseado na arquitetura orientada a serviços;
- Utilizar geotecnologias em todos os processos que envolvam localização facilitando a identificação e atualização de informações relativas às entregas agendadas e realizadas;
- Tornar viável o uso de todas as tecnologias da informação e softwares necessários para atender às demandas dos clientes, fornecedores e parceiros.

2.2. Descrição geral do software (Escopo)

O sistema deve ser constituído de quatro módulos e contemplar as funcionalidades e recursos que atendam o plano de Metas e Diretrizes para a área de TI. Aplicação será denominada Gestão de Serviços de Logística (GSL) e seus módulos devem incorporar os recursos dos sistemas SAF, SGE e SFC através de:

- Um módulo de cadastro para informações necessárias ao negócio da empresa que serão armazenadas em Enterprise Information Systems (EIS);
- Um módulo de serviços ao cliente fazendo o uso de Business Process Management (BPM);
- Um módulo para realizar a gestão estratégica de todas as atividades da empresa fazendo uso de uma ferramenta de gestão corporativa adquirida no mercado;
- Um módulo de ciência de dados para apoio às tomadas de decisão empregando recursos de Big Data e Data Warehouse.

Este sistema será acessado pelo usuário denominado Remetente, que geralmente possui um e-commerce, e faz uso de self storages para armazenar os produtos que comercializa. Ele precisa ter um controle sobre os estoques e seus respectivos produtos armazenados, para poder ter a disponibilidade de atender os pedidos recebidos.

É importante ressaltar, que o sistema legado permanecerá em funcionamento, e integrado a nova plataforma, reaproveitando assim toda a inteligência já desenvolvida pela Boa Entrega que faz uso de algoritmos de otimização de traçado para geração de rotas de entrega. Neste mecanismo de roteirização 3 fatores são os principais influenciadores na geração de um traçado: a distância entre os endereços considerando as rotas possíveis, o custo de combustível para percorrer a rota e o tempo da rota considerando o horário previsto de entrega. Os dados de remetentes, destinatários e estoques, que estão disponibilizando os produtos para o atendimento de um pedido serão alimentados pelo próprio cliente remente através da nova plataforma e enviados via integração para o sistema legado afim de definir a melhor rota para uma determinada entrega.

Também será possível, através da nova plataforma, o remetente acompanhar o processo de entrega do produto que está enviando ao seu cliente, pois através do processo de integração, o sistema legado irá transmitir para a plataforma todos os dados de entrega e as evoluções nas etapas de atendimento, permitindo o cliente remetente acompanhar o envio da mercadoria até o destinatário, seu cliente final.

3. Definição conceitual da solução

Esta seção descreve os requisitos funcionais e não funcionais contemplados neste projeto arquitetural. Os requisitos funcionais se referem as especificações envolvidas nas funcionalidades da aplicação, enquanto os requisitos não funcionais definem propriedades e restrições do sistema.

3.1. Requisitos Funcionais

Acesso

- RF01 – O sistema deve permitir o registro de um novo usuário como remetente na plataforma, informando email, senha e os dados da empresa;
- RF02 – O sistema deve permitir acesso às funcionalidades da plataforma somente mediante processo de login prévio e checagem das credenciais.

Módulo de Cadastro

- RF03 – Cadastro de Depósitos (Privado): o sistema deve permitir obter e manter informações de depósitos;
- RF04 – Cadastro de Produtos (Privado): o sistema deve permitir obter e manter informações de produtos dos segmentos em que atua;
- RF05 – Cadastro de Destinatários: o sistema deve permitir obter e manter informações de destinatários e compartilhá-los com os demais remetentes da plataforma;
- RF06 – Cadastro de Fornecedores: o sistema deve permitir obter e manter informações de fornecedores e compartilhá-los com os demais remetentes da plataforma;
- RF07 – Associação de produtos à estoques (Privado): o sistema deve permitir a associação de um produto previamente cadastrado a um também previamente cadastrado no sistema afim de sinalizar que naquele local aquele produto está disponível.

Módulo de Serviços ao Cliente

- RF08 – Processo de atendimento (Privado): o sistema deve permitir ao remetente definir, acompanhar e analisar o fluxo de atendimento ao seu cliente.
- RF09 – Monitoramento de Pedidos (Privado): o sistema deve permitir ao remetente acompanhar os novos pedidos recebidos e com base na sua disponibilidade em estoques, definir de onde deve partir a coleta do produto;
- RF10 – Monitoramento de Entregas (Privado): o sistema deve permitir ao remetente acompanhar a evolução dos processos de entrega do produto que está sendo enviado até o seu destino.

Módulo de Gestão Estratégica

- RF11 – Indicadores de Entregas (Privado) – o sistema deve fornecer dados relacionados à eficiência e efetividade das entregas realizadas com base nos dados provenientes dos processos de entrega do produto na forma de cockpit;
- RF12 – Indicadores de Estoques (Privado) – o sistema deve fornecer dados relacionados a produtos armazenados e nível dos estoques com base nos dados provenientes das associações de produtos à estoques na forma de cockpit.

Módulo de Ciência de Dados

- RF13 – Big Data (Privado) – obter informações dos bancos de dados para gerar informações relacionadas às entregas como:
 - Identificar a região do país com maior número de entregas visando a adoção de self stores mais próximas a fim de otimizar as entregas;
 - Identificar o produto com maior demanda afim de manter um nível de estoque saudável para atendimentos as demandas e identificar as tendências dos clientes.

3.2 Requisitos Não-Funcionais

- RNF01 – Segurança – O sistema deve possuir mecanismo e segurança no acesso;

- RNF02 – Interoperabilidade – O sistema deve permitir que os serviços se comuniquem para realizar ações necessárias;
- RNF03 – Desempenho – O sistema deve ser rápido;
- RNF04 – Disponibilidade – O sistema deve se manter ativo mesmo diante de excesso de requisições
- RNF05 – Acessibilidade – O sistema deve ser acessível em ambientes Web para desktop e ambientes móveis nativos;
- RNF06 – Usabilidade – O sistema deve ser simples de usar.

3.3. Restrições Arquiteturais

R1: Os serviços serão desenvolvidos em NodeJS Typescript;

R2: Será empregado o conceito de Application Programming Interfaces (API) na criação dos serviços;

R3: O sistema deve ser modular com arquitetura orientada a serviços.

R4: A integração entre os serviços deve utilizar o protocolo HTTP seguindo o padrão REST;

R5: Cada serviço deve ter seu escopo bem definido e isolado com banco de dados próprio;

R6: A sincronia de dados entre os serviços de back-end deve ocorrer de forma orquestrada para garantir a integridade da aplicação.

3.4. Mecanismos Arquiteturais

Análise	<i>Design</i>	Implementação
Linguagem	Linguagem de Programação	Typescript
Persistência	Banco de dados NoSQL	MongoDB
Persistência	Biblioteca ORM	TypeORM

Front-end	Interface de comunicação com o usuário do sistema	Angular
Back-end	API webservice	NodeJS
Comunicação entre módulos e/ou sistemas	Intercâmbio de dados baseado em notação de objeto JSON	REST
Log do sistema	Biblioteca de Log	Log4Js
Teste Unitário	Biblioteca de Testes Unitários	Jest
Autenticação e Autorização	Verificação de credenciais de acesso	JWT
Versionamento	Versionamento de Código	GIT
Repositório	Repositório de Código	GITHUB
Deploy	Entrega Contínua	AWS CodePipeline
Infraestrutura	Infraestrutura como Código	AWS CloudFormation
Mensageria	Serviço de gerenciamento de streaming	Amazon MSK
Contêiner	Serviço de containerização de software	Docker
Orquestrador	Serviço de gerenciamento de contêineres	AWS ECS
Virtualização	Serviço de computação escalável	AWS EC2

4. Modelagem e projeto arquitetural

Nesta seção apresenta-se a modelagem arquitetural da solução proposta, de forma a permitir seu completo entendimento através dos diagramas de componentes e implantação, bem como os modelos de dados dos microsserviços que compõe a arquitetura.

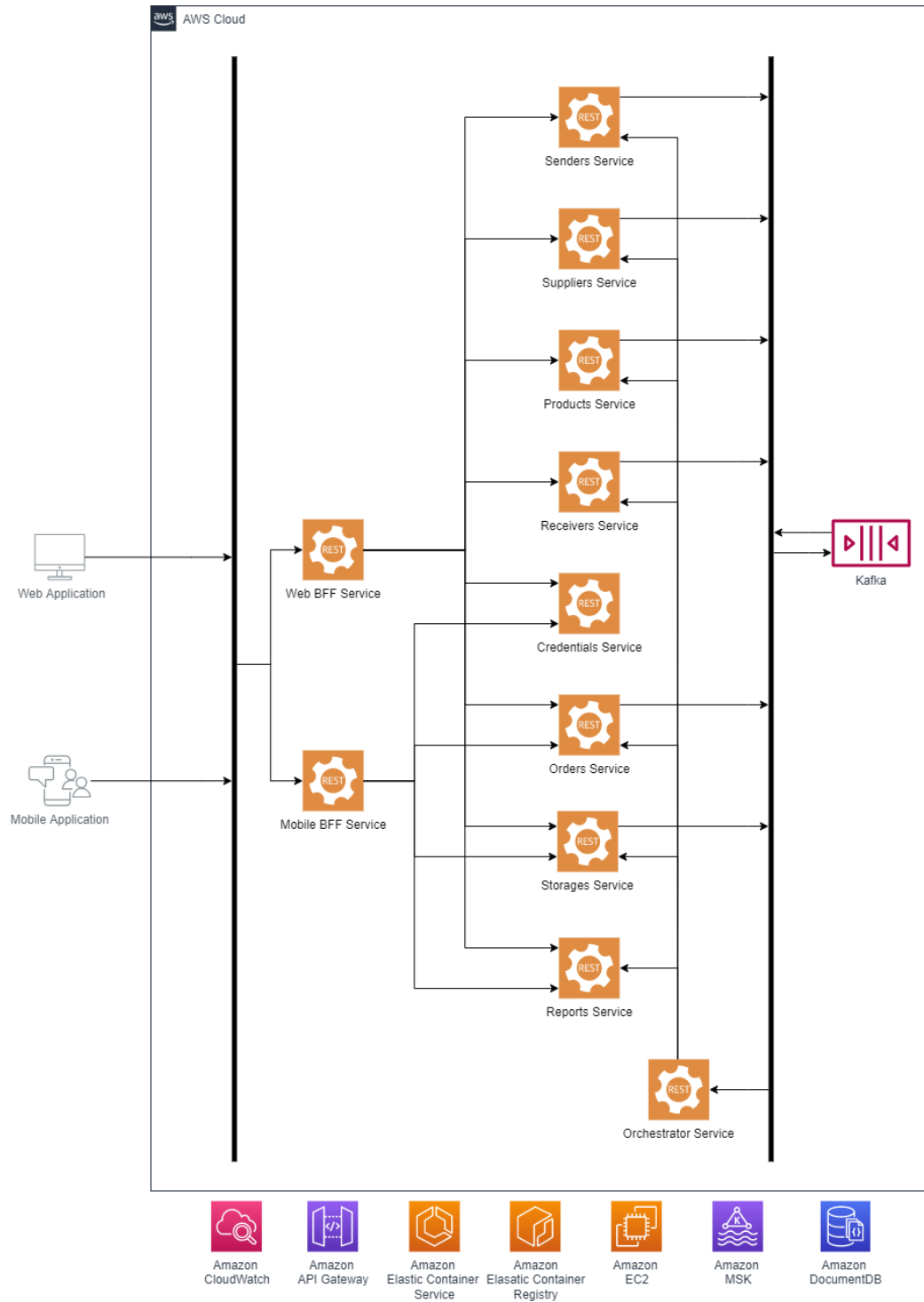


Figura 1 - Visão Geral da Solução.

4.1. Modelo de componentes

O diagrama de componentes da aplicação, que segue um padrão arquitetural orientado a microsserviços, consiste em uma coleção de pequenos serviços independentes e fracamente

acoplados, cada um com suas responsabilidades específicas e controle de dados próprio e um serviço orquestrador para garantir a integridade das informações entre si e a estrutura legada.

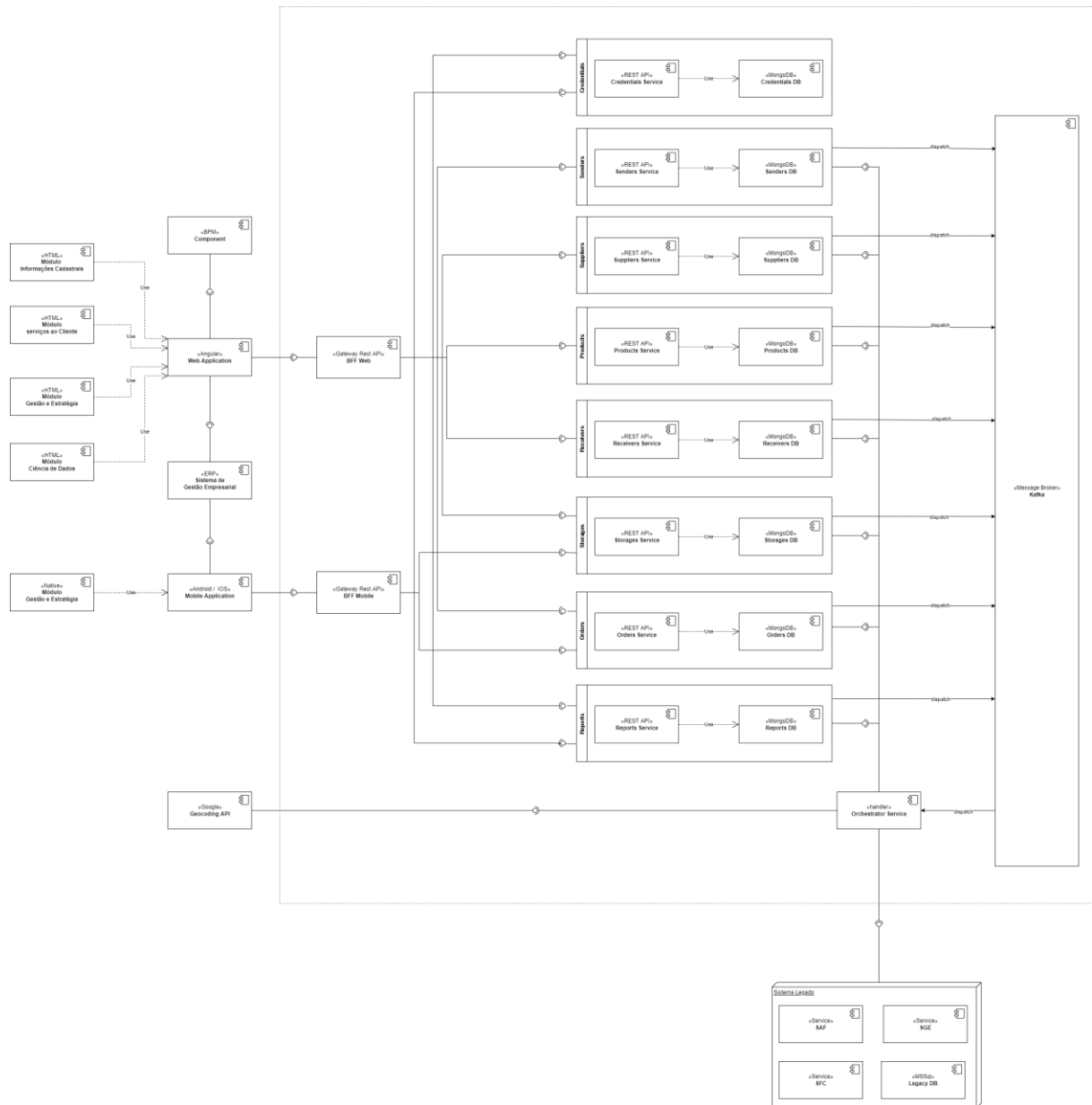


Figura 2 – Diagrama de Componentes.

De acordo com o diagrama apresentado na Figura 2, a solução possui as seguintes entidades:

- **Web Application** (Módulos: Informações Cadastrais, Serviços ao Cliente, Gestão e Estratégia, Ciência de Dados) – Aplicativo desenvolvido em Angular que gera conteúdo em HTML dos módulos do sistema que são renderizados no navegador do usuário, de forma responsiva para desktop.

- *Mobile Application* (Módulo: Gestão e Estratégia) – Aplicativo desenvolvido nativamente em Android e IOS contendo apenas o módulo de Gestão e Estratégia.
- Sistema de Processos – Aplicação para gerenciamento de processos do negócio baseados em fluxos BPM integrada ao módulo Serviços ao Cliente.
- Sistema de Gestão - Aplicação terceira adquirida no mercado para gestão corporativa integrada ao módulo Gestão e Estratégia.
- *Geocoding API* – Serviço do Google que retorna as coordenadas geográficas de uma localidade a partir de um endereço informado.
- *Web BFF* – REST API que serve como ponto de entrada para as requisições dos clientes que utilizam a plataforma web, fazendo o encaminhamento da chamada para os devidos serviços de back-end utilizados pela aplicação.
- *Mobile BFF* – REST API que serve como ponto de entrada para as requisições dos clientes que utilizam a plataforma mobile, fazendo o encaminhamento da chamada para os devidos serviços de back-end utilizados pela aplicação.
- *Message Broker* – Plataforma para transmissão de dados em fluxo contínuo que permite que serviços independentes se comuniquem entre si.
- *Senders Service* – REST API responsável por gerir os dados dos remetentes, empresas clientes da Boa Entrega. Cadastra, fornece e mantém os dados armazenados em banco de dados NoSql MongoDB e faz uso do *message broker* a fim de estimular o *Orchestrator Service* a manter a sincronia de informações com os demais serviços da plataforma.
- *Credentials Service* – REST API responsável por gerenciar as credenciais e as sessões de acesso vinculadas aos remetentes na plataforma. Cadastra, fornece e mantém as credenciais em banco de dados NoSql MongoDB e faz uso de Json Web Token JWT para gerenciar as sessões de acesso.
- *Receivers Service* – REST API responsável por gerir os dados dos destinatários das mercadorias. Os dados gerenciados por este serviço são compartilhados entre todos os remetentes que fazem uso da plataforma. Cadastra, fornece e mantém os dados

armazenados em banco de dados NoSql MongoDB e faz uso do *message broker* a fim de estimular o *Orchestrator Service* a manter a sincronia de informações com os demais serviços da plataforma.

- *Suppliers Service* – REST API responsável por gerir os dados de fornecedores. Os dados gerenciados por este serviço são compartilhados entre todos os remetentes que fazem uso da plataforma. Cadastra, fornece e mantém os dados armazenados em banco de dados NoSql MongoDB e faz uso do *message broker* a fim de estimular o *Orchestrator Service* a manter a sincronia de informações com os demais serviços da plataforma.

- *Products Service* - REST API responsável por gerir os dados de mercadorias. Os dados gerenciados por este serviço são restritos a cada remetente, permitido o armazenamento de informações de acordo com seu segmento. Cadastrar, fornece e mantém os dados armazenados em banco de dados NoSql MongoDB e faz uso do *message broker* a fim de estimular o *Orchestrator Service* a manter a sincronia de informações com os demais serviços da plataforma.

- *Storages Service* - REST API responsável por gerir os dados de depósitos e mercadorias armazenadas. Os dados gerenciados por este serviço são restritos a cada remetente, permitindo o armazenamento de informações de acordo com seu segmento e mercadorias previamente cadastradas. Cadastra, fornece e mantém os dados armazenados em banco de dados NoSql MongoDB e faz uso do *message broker* a fim de estimular o *Orchestrator Service* a manter a sincronia de informações com os demais serviços da plataforma.

- *Orders Service* - REST API responsável por gerir os dados de pedidos registrados na plataforma. Os dados gerenciados por este serviço são restritos a cada remetente, permitindo o acesso exclusivo a pedidos registrados com suas mercadorias. Cadastra, fornece e mantém os dados armazenados em banco de dados NoSql MongoDB e faz uso do *message broker* a fim de estimular o *Orchestrator Service* a manter a sincronia de informações com os demais serviços da plataforma.

- *Reports Service* – REST API responsável por gerir dados para relatórios a serem interpretados através de visões analíticas. Recebe estímulos a partir do *Orchestrator Service* para registro de dados consolidados de depósitos e pedidos dos remetentes. Os dados

gerenciados por este serviço são restritos a cada remetente. Cadastra, fornece e mantém os dados armazenados em banco de dados NoSql MongoDB.

- *Orchestrator Service* – Serviço de integração responsável organizar as transações que abrangem os serviços dispostos na arquitetura. Como cada serviço possui seu próprio banco de dados, o *Orchestrator Service* tem a proposta de manter a consistência de dados entre os serviços distribuídos da plataforma e manter a integração com o sistema legado. Recebe estímulos provenientes do consumo de tópicos específicos do *message broker*, manipula e trabalha os dados recebidos consumindo e propagando as informações por meio de requisições REST entre os demais.

4.2. Modelo de implantação

A seguir é apresentado o modelo de implantação que representa a relação entre os componentes do sistema e como devem ser implantados em suas respectivas infraestruturas de sustentação. No modelo a seguir não está explícita a arquitetura de cluster, mas deve-se considerar um sistema usufruindo de todo o serviço de gerenciamento que a computação em nuvem pode proporcionar, portanto trata-se de uma arquitetura de alta performance e resiliente a falhas.

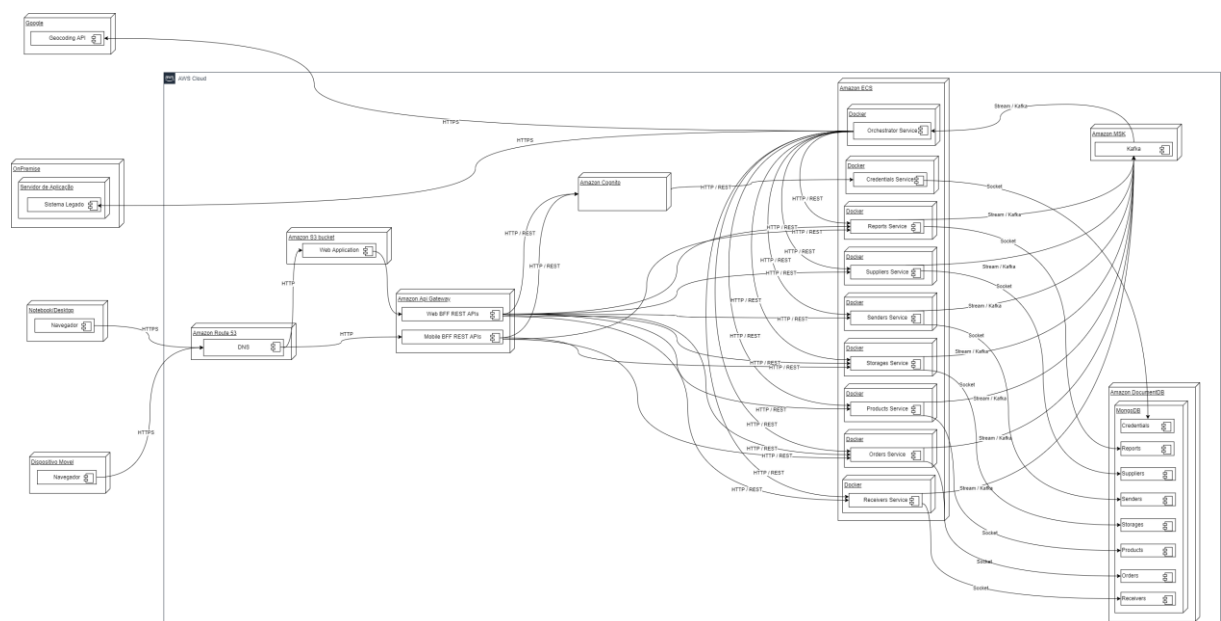


Figura 3 – Diagrama de Implantação.

A arquitetura apresentada na figura 3 será alocada na plataforma de computação em nuvem AWS visando a utilização de seus serviços gerenciáveis, evitando assim trabalhos extras com provisionamento e gerenciamento de servidores e adotando a prática da infraestrutura como código. Para garantir a alta performance e a disponibilidade do serviço em horário integral, 24 horas por dia, 7 dias por semana serão utilizados os seguintes artefatos computacionais:

- Docker – conjunto de ferramentas de plataforma como serviço utilizado para empacotar as aplicações Orchestrator Service, Credentials Service, Reports Service, Suppliers Service, Senders Service, Storages Services, *Products Service* e Receivers Service dentro de contêineres isolados e portáteis para implantação.
- Amazon Cognito – serviço que permite cadastramento, login e controle de acesso de usuários à aplicação. Oferece suporte à autenticação multifatorial e à criptografia de dados em repouso e em trânsito. Será utilizado em conjunto com o *Credentials Service* para vincular as credenciais de acesso ao negócio.
- Amazon ECS – serviço totalmente gerenciado pela cloud que oferece suporte ao Docker para orquestração de contêineres possibilitando a implantação, gerenciamento e escalabilidade das aplicações facilmente.
- Amazon Document DB – serviço de banco de dados JSON altamente disponível e escalável, gerenciado pela cloud e compatível com o MongoDB. Será utilizado para provisionar os bancos de dos microsserviços desenvolvidos.
- Amazon MKS – serviço gerenciado pela nuvem que facilita a ingestão e o processamento de dados de streaming em tempo real com o Apache Kafka. Nele será criado um cluster do Kafka, que servirá como *message broker* para que as apis estimulem o serviço orquestrador.
- Amazon Api Gateway – serviço gerenciado que administra todas as tarefas envolvidas no recebimento e processamento de centenas de milhares de chamadas de API simultâneas com gerenciamento de tráfego, suporte de CORS e controle de autorização e acesso. Nesta arquitetura ele faz o encapsulamento dos serviços de back-end expondo apenas as rotas usadas pelos módulos do sistema e integração com o sistema legado.

- Amazon S3 bucket – serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e performance. Nele são hospedados os sites estáticos individuais que correspondem aos módulos do sistema.
- Amazon Route 53 – web service de DNS altamente escalável e disponível que transforma um endereço como `www.exemplo.com.br` para endereços de IP numéricos. Nesta arquitetura o Route 53 conecta efetivamente as requisições do usuário à infraestrutura em execução na AWS.

5. Prova de Conceito (POC) / protótipo arquitetural

Nesta seção será apresentada a prova de conceito (POC) desenvolvida com o intuito de atender aos requisitos especificados neste relatório através do desenvolvimento de 3 Casos de Uso (UC), com a descrição das tecnologias aplicadas na implementação.

5.1. Implementação e Implantação

Para a realização da prova de conceito deste projeto foram elaborados 3 casos de uso, com foco no controle de estoques. As funções desenvolvidas permitem o cadastro de depósitos, a associação de produtos a depósitos e a visualização dos depósitos e seus respectivos indicadores de armazenamento. Por se tratar de uma arquitetura baseada em microsserviços, onde cada serviço tem seu banco de dados específico, o desafio é manter a integridade da informação e sincronia no processo de execução das tarefas envolvidas sem perder performance. O objetivo é identificar se a arquitetura definida atende os requisitos não-funcionais elencados, durante a execução dos processos propostos nas histórias dos casos de uso.

Foram necessárias a configuração do Kafka e a implementação de 7 serviços da arquitetura proposta para a realização desta POC, sendo eles:

- *Web-BFF*;
- *Web-Front*;
- *Credentials-Service* e o provisionamento de seu respectivo banco de dados;

- *Senders-Service* e o provisionamento de seu respectivo banco de dados;
- *Products-Service* e o provisionamento de seu respectivo banco de dados;
- *Storages-Service* e o provisionamento de seu respectivo banco de dados;
- *Reports-Service* e o provisionamento de seu respectivo banco de dados;
- *Orchestrator-Service*.

Os casos de uso, suas respectivas descrições resumidas no formato estória, fluxos de execução principal e diagramas de sequência apresentados nesta sessão servem para assimilar toda a cadeia de processos envolvida na efetivação das ações que serão avaliadas.

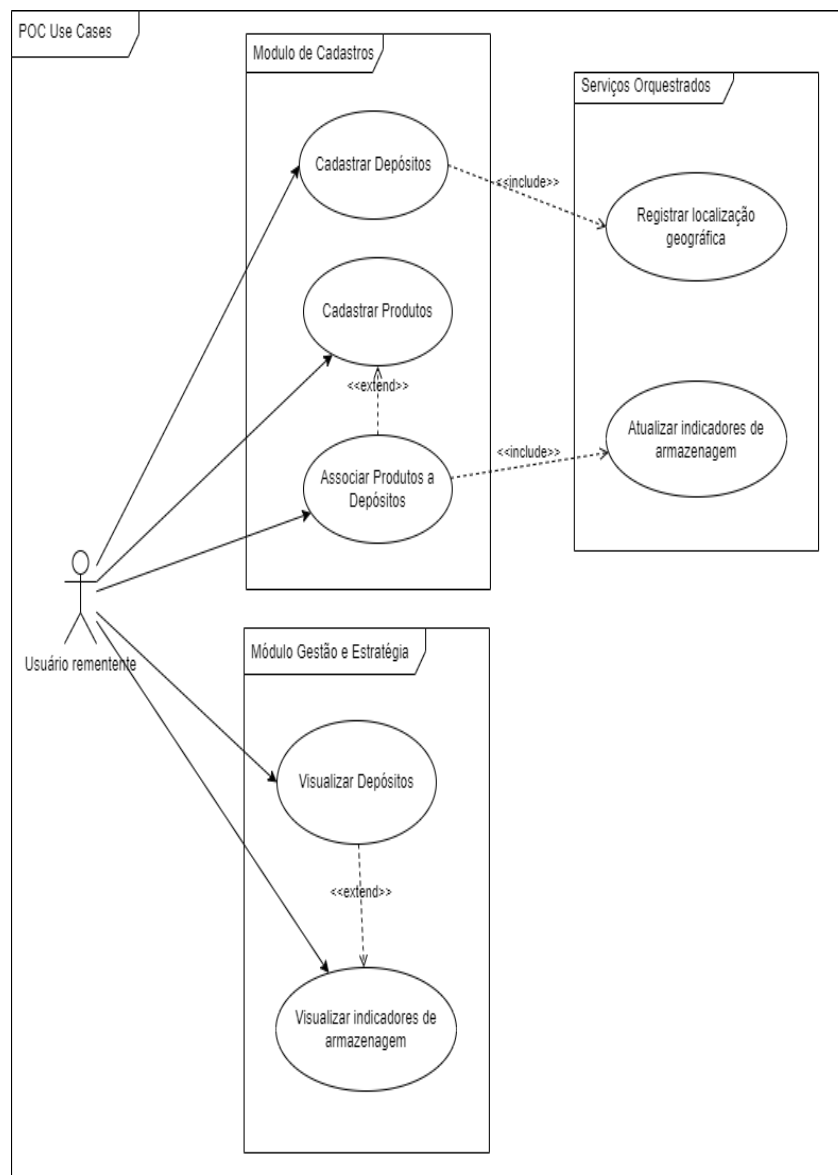


Figura 4 – Casos de Uso desenvolvidos para POC.

ID	UC01
Estória do Usuário	Cadastrar Depósitos
Criador	Como remetente das entregas
Descrição	<p>Eu quero cadastrar dados dos depósitos contratados para armazenamento dos produtos através da plataforma Web, realizando autenticação prévia, selecionando o fornecedor do depósito e preenchendo os dados necessários para cadastro como:</p> <ul style="list-style-type: none"> • nome; • telefone; • email; • volume contratado; • endereço. <p>Se o processo ocorrer de forma satisfatória deverá apresentar mensagem de sucesso na tela. Assincronamente, o sistema deve:</p> <ul style="list-style-type: none"> • identificar as coordenadas geográficas do depósito a partir do endereço registrado e atualizar a informação no cadastro; • criar o registro de indicadores com a capacidade contratada.
Valor do negócio	Para identificar e localizar depósitos contratados.
Prioridade	Alta
Estimativa	50 horas

ID	UC02
Estória do Usuário	Associar produto ao depósito
Criador	Como remetente das entregas
Descrição	<p>Eu quero associar produtos, previamente cadastrados, aos estoques através da plataforma web, de forma a refletir a real utilização do armazenamento contratado, realizando autenticação prévia, selecionando:</p> <ul style="list-style-type: none"> • depósito; • produto; • quantidade armazenada. <p>Se o processo ocorrer de forma satisfatória deverá apresentar mensagem de sucesso na tela. Assincronamente, o sistema deve:</p> <ul style="list-style-type: none"> • contabilizar os produtos armazenados no estoque, calculando volume total, valor total, quantia total, e a porcentagem de ocupação; • atualizar o registro de indicadores com a capacidade contratada, e a ocupação atualizada do armazenamento.
Valor do negócio	Para refletir o uso real dos depósitos contratados, averiguar a disponibilidade do produto quando receber pedidos e ter ciência do valor de mercadorias armazenadas.
Prioridade	Alta
Estimativa	60 horas

ID	UC03
Estória do Usuário	Visualizar indicadores dos estoques

Criador	Como remetente das entregas
Descrição	<p>Eu quero consultar os dados atualizados dos depósitos contratados possibilitando visualizar através de indicadores:</p> <ul style="list-style-type: none"> • capacidade contratada do depósito; • total de volume de produtos armazenado no estoque; • percentual de utilização do estoque; • total de produtos armazenados; • valor total dos estoques de produtos. <p>O acesso se dará através da plataforma web, com autenticação prévia, e se a busca de informação ocorrer de forma satisfatória, os indicadores serão apresentados em dashboard no formato cockpit.</p>
Valor do negócio	Gerir os níveis de armazenamento dos depósitos possibilitando a melhor distribuição dos produtos e controlar disponibilidade. Visualizar o valor dos produtos estocados para fins de contabilidade.
Prioridade	Alta
Estimativa	40 horas

Esta POC tem o objetivo de validar os seguintes requisitos não funcionais:

- RNF01 Segurança – O sistema deve possuir mecanismo e segurança no acesso: escolhido devido a importância de se manter os dados sigilosos. Os critérios de aceite são:
 - Os serviços de back-end devem ser encapsulados e o sistema deve permitir acesso somente mediante BFF;
 - O sistema deve impedir o acesso a páginas privadas com requisições não autenticadas;

- O sistema deve barrar a tentativa de um determinado usuário remetente requisitar os dados sigilosos de outro usuário remetente no sistema.
- RNF02 Interoperabilidade – O sistema deve permitir que os serviços se comuniquem para realizar ações necessárias: escolhido pois se trata de um item essencial em uma arquitetura de software distribuído. Os critérios de aceite são:
 - O Serviço Orquestrador deve se comunicar com as apis isoladas e serviços externos a fim de validar e manter a integridade dos dados na propagação das informações;
 - O BFF deve se comunicar com as apis isoladas para validar e manter a integridade dos dados nas consultas e cadastros;
 - A comunicação entre os serviços deve ocorrer mediante requisições REST e por streaming através de serviço de mensageria.
- RNF03 Desempenho – O sistema deve ser rápido para confirmar o sucesso de um cadastro, para realizar consultas de dados e propagar informações entre os demais serviços: escolhido com o intuito de garantir uma boa performance da aplicação. Os critérios de aceite são:
 - O registro de dados realizados pelas apis isoladas não devem demorar mais de 2 segundos;
 - As consultas internas realizadas pelas apis isoladas não devem demorar mais de 1 segundo para retornar dados;
 - A comunicação entre os serviços e a propagação das informações entre eles não devem demorar mais de 3 segundos.

Para o desenvolvimento desta POC foram utilizadas ferramentas específicas para o back-end e para o front-end.

As seguintes ferramentas foram utilizadas no desenvolvimento back-end:

- Linguagem de programação: TypeScript com NodeJS;

- Gerenciador de Pacotes: Yarn;
- Bibliotecas: typeorm para acesso ao banco de dados, axios para comunicação http, express para criação da estrutura de websservices, celebrate para validação dos campos nas requisições, tsringe para aplicar o conceito de injeção e dependências, jsonwebtoken e bcryptjs para autenticação e credenciais de acesso.
- Banco de dados: os bancos de dados utilizados pelos serviços desenvolvidos são MongoDB e para a implementação desta POC fez-se uso da plataforma Atlas MongoDB, que fornece clusterers gerenciáveis pela cloud e gratuitos para desenvolvimento e estudos.
- Mensageria: fez-se uso do serviço de mensageria Kafka monitorado pelo Zookeeper.

As seguintes ferramentas foram utilizadas no desenvolvimento front-end:

- Linguagem de programação: TypeScript com AngularJS, CSS e HTML.
- Gerenciador de Pacotes: NPM;
- Bibliotecas: ng-bootstrap para estilização do front-end com CSS, ngx-toaster para emissão de notificações ao usuário, @angular/google-maps componente Angular para uso do Google Maps, ng2-charts para criação de gráficos responsivos e reativos para angular, baseados no componente Chart.js, ngx-pagination para controle de paginação de listas obtidas do back-end e ngx-mask para aplicar formatação na apresentação de conteúdo.

O provisionamento desta POC se deu na Amazon Web Services utilizando as seguintes ferramentas:

- Docker Desktop para construção das imagens das aplicações;
- AWS CLI para se conectar ao ambiente da Amazon Web Services através de linha de comando a partir da máquina local;

- Cloud Formation para especificar a infraestrutura e definir os serviços utilizados na POC e criar o ambiente cloud através do AWS CLI;

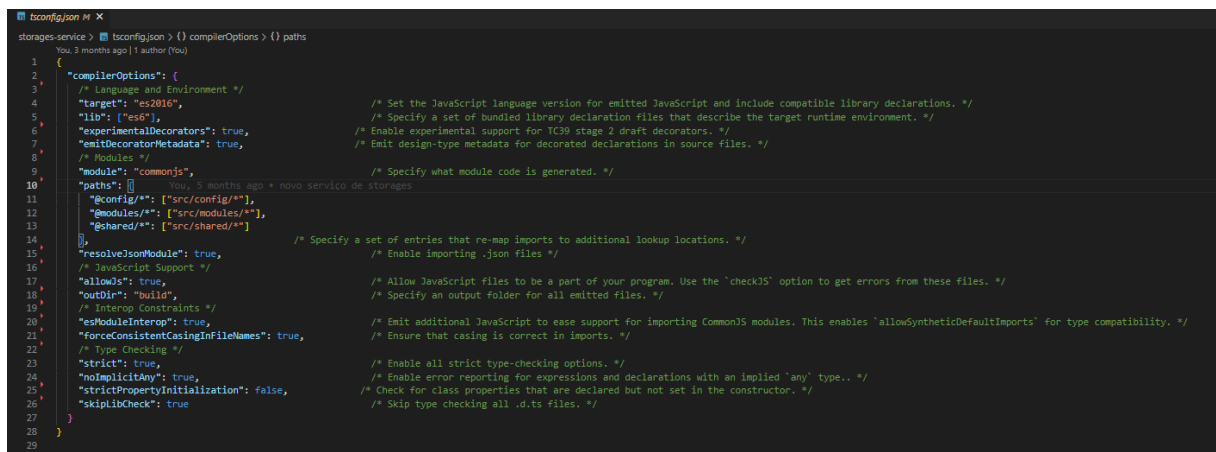
A POC se encontra disponível para acesso em <http://logis-loadb-ed0xydtihm3e-885ab48e11a65558.elb.us-east-1.amazonaws.com:8081> e para utilizar o sistema deve-se autenticar na aplicação com as seguintes credenciais:

- Usuário: bitols@gmail.com;
- Senha: *pacatatucotiano*.

5.2. Código

Para o desenvolvimento das aplicações de back-end, após a inicialização do projeto com o comando `yarn init -y` foram necessários preparar previamente o ambiente de desenvolvimento para a utilização do TypeScript. Para isto foi necessário a utilização das bibliotecas:

- typescript - linguagem para JavaScript que adiciona tipos adicionais que suportam ferramentas para aplicativos em alta escala para qualquer navegador, para qualquer host em qualquer sistema operacional;
- ts-node-dev – ferramenta utilizada durante o desenvolvimento que reinicia o processo do nó de destino quando qualquer arquivo fonte for alterado.



```

1 {
2   "compilerOptions": {
3     /* Language and Environment */
4     "target": "es2016", /* Set the JavaScript language version for emitted JavaScript and include compatible library declarations. */
5     "lib": ["es6"], /* Specify a set of bundled library declaration files that describe the target runtime environment. */
6     "experimentalDecorators": true, /* Enable experimental support for TC39 stage 2 draft decorators. */
7     "emitDecoratorMetadata": true, /* Emit design-type metadata for decorated declarations in source files. */
8     /* Modules */
9     "module": "commonjs", /* Specify what module code is generated. */
10    "paths": [
11      "@config/*": ["src/config/*"],
12      "@modules/*": ["src/modules/*"],
13      "@shared/*": ["src/shared/*"]
14    ], /* Specify a set of entries that re-map imports to additional lookup locations. */
15    "resolveJsonModule": true, /* Enable importing .json files */
16    /* JavaScript Support */
17    "allowJs": true, /* Allow JavaScript files to be a part of your program. Use the 'checkJS' option to get errors from these files. */
18    "outDir": "build", /* Specify an output folder for all emitted files. */
19    /* Interop Constraints */
20    "esModuleInterop": true, /* Emit additional JavaScript to ease support for importing CommonJS modules. This enables 'allowSyntheticDefaultImports' for type compatibility. */
21    "forceConsistentCasingInFileNames": true, /* Ensure that casing is correct in imports. */
22    /* Type Checking */
23    "strict": true, /* Enable all strict type-checking options. */
24    "noImplicitAny": true, /* Enable error reporting for expressions and declarations with an implied 'any' type.. */
25    "strictPropertyInitialization": false, /* Check for class properties that are declared but not set in the constructor. */
26    "skipLibCheck": true /* Skip type checking all .d.ts files. */
27  }
28 }
29

```

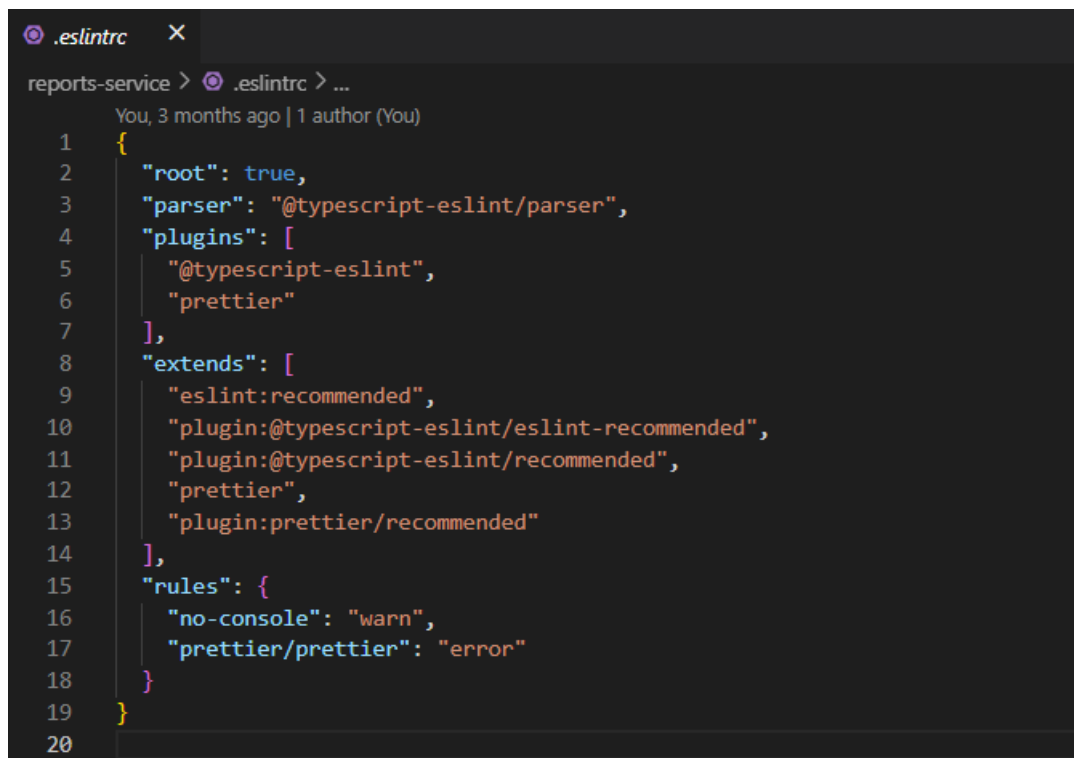
Figura 5 – Arquivo tsconfig.json com a configuração do TypeScript no projeto storages-service.

O desenvolvimento dos serviços envolvidos na arquitetura pautou-se pela empregabilidade de conceitos de boas práticas e qualidade no código, introduzindo recursos como princípios SOLID e Domain-drive Design (DDD).

Foram utilizadas para auxiliar no emprego de padrões de projeto as bibliotecas:

- eslint - uma ferramenta para identificar e relatar padrões encontrados no código;
- prettier - um formatador de código opinativo que impõe um estilo consistente analisando o código;
- eslint-config-prettier – desativa todas as regras do eslint que tem potencial de interferir nas regras do prettier;
- eslint-plugin-prettier – transforma regras do prettier e eslint.

Em conjunto, o eslint define as convenções de código e o prettier realiza a formatação automática com base nas regras do eslint.



```
.eslintrc
reports-service > .eslintrc > ...
You, 3 months ago | 1 author (You)
1  {
2    "root": true,
3    "parser": "@typescript-eslint/parser",
4    "plugins": [
5      "@typescript-eslint",
6      "prettier"
7    ],
8    "extends": [
9      "eslint:recommended",
10     "plugin:@typescript-eslint/eslint-recommended",
11     "plugin:@typescript-eslint/recommended",
12     "prettier",
13     "plugin:prettier/recommended"
14   ],
15   "rules": {
16     "no-console": "warn",
17     "prettier/prettier": "error"
18   }
19 }
20
```

Figura 6 – Arquivo .eslintrc com a configuração do eslint no projeto *reports-service*.



```
{  
  "semi": true,  
  "trailingComma": "all",  
  "singleQuote": true,  
  "printWidth": 80,  
  "arrowParens": "avoid"  
}
```

Figura 7 – Arquivo `.prettierrc` com a configuração do prettier no projeto *reports-service*.

Os projetos foram estruturados de forma a separar as responsabilidades da aplicação, tendo como principais diretórios:

- `config` – reservada para as bibliotecas externas utilizadas nos projetos;
- `modules` – utilizada para armazenar tudo o que abrange as áreas de conhecimento nos projetos e diretamente ligadas à área de negócios;
- `shared` – para fontes de uso geral e compartilhado com mais de um módulo dentro dos projetos;
- `useCases` – dentro de cada módulo do projeto e são responsáveis por aplicar todas as regras que o serviço precisa atender;
- `infra` – dentro de cada módulo utilizado para tudo aquilo que dá sustentação ao negócio, como repositórios de banco de dados, controllers e rotas do módulo. Dentro de `shared` para referenciar as rotas dos módulos e compartilhá-las com os demais módulos do projeto;
- `domain` – dentro de cada módulo para as interfaces e tipos que definem o negócio.

No back-end as APIs CRUD possuem negócio específico e são totalmente isoladas, com banco de dados próprio e sem acoplamento entre si, portanto estas possuem apenas um módulo.



Figura 8 – Estrutura de diretórios dos projetos de apis do back-end.

Já as aplicações *web-bff* e *orchestrator-service*, que fazem o papel de interagir entre os serviços, a estrutura possui mais de um módulo em seus projetos.

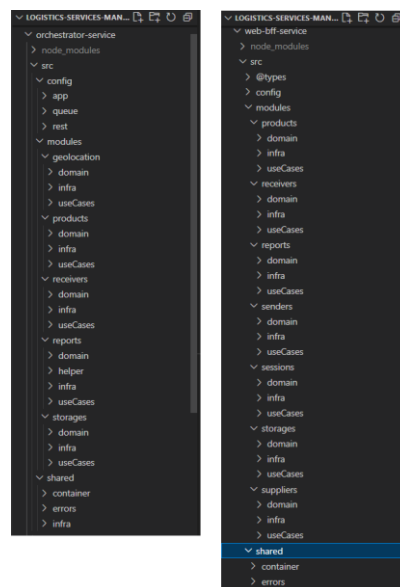


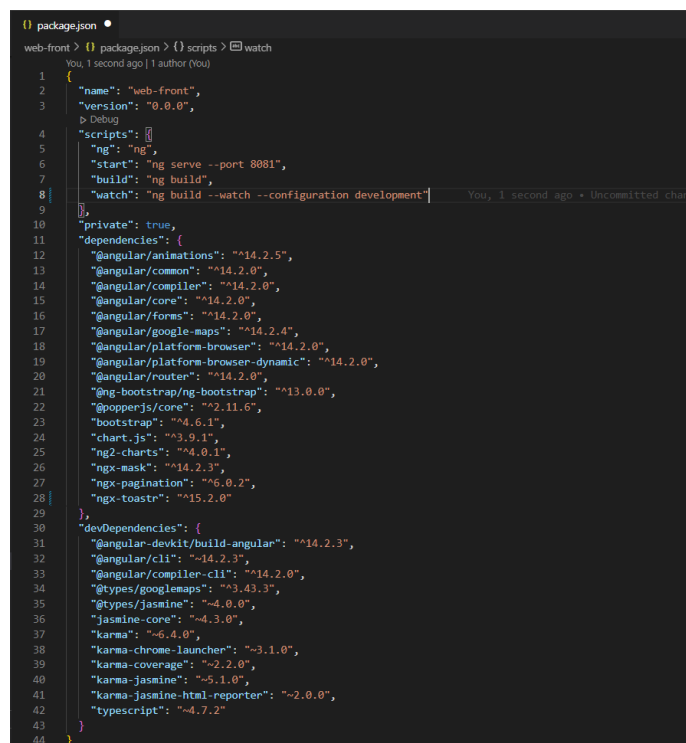
Figura 9 – Estrutura de diretórios dos projetos *web-bff* e *orchestrator-service* respectivamente.

Para o desenvolvimento do front-end foi realizado o setup do projeto através do Angular Cli com o comando `ng new`, que já cria um padrão estrutural para os diretórios do projeto. A criação das classes, componentes e serviços também podem ser realizadas através de comandos Angular Cli, o que facilita a criação e organização. A seguir alguns exemplos de comandos utilizados para estas finalidades:

- `ng g class models/storages --type=model` – cria um arquivo denominado `storages.model.ts` dentro do diretório `models` para a classe modelo `Storages`;

- `ng g c components/add-storages` – cria dentro do diretório *components/add-storages* um grupo de arquivos dedicados ao componente de tela, sendo eles: *add-storages.components.ts* para desenvolvimento da lógica de programação do componente, *add-storages.components.html* para construção da parte visual do componente com linguagem de marcação HTML e *storages.component.css* para estilização visual do componente com CSS;
- `ng g s services/storages` – cria dentro do diretório *services* um arquivo denominado *storages.services.ts* para desenvolvimento da lógica de programação para consumo, no caso deste trabalho caso, de dados dos estoques provenientes da api do bff.

Para a instalação de pacotes no projeto foi utilizado o *NPM*, e assim como nos projetos de back-end as dependências ficam salvas no arquivo *package.json*.



```

1  {
2    "name": "web-front",
3    "version": "0.0.0",
4    "scripts": {
5      "ng": "ng",
6      "start": "ng serve --port 8081",
7      "build": "ng build",
8      "watch": "ng build --watch --configuration development"
9    },
10   "private": true,
11   "dependencies": {
12     "@angular/animations": "^14.2.5",
13     "@angular/common": "^14.2.0",
14     "@angular/compiler": "^14.2.0",
15     "@angular/core": "^14.2.0",
16     "@angular/forms": "^14.2.0",
17     "@angular/google-maps": "^14.2.4",
18     "@angular/platform-browser": "^14.2.0",
19     "@angular/platform-browser-dynamic": "^14.2.0",
20     "@angular/router": "^14.2.0",
21     "@ng-bootstrap/ng-bootstrap": "^13.0.0",
22     "@popperjs/core": "^2.11.6",
23     "bootstrap": "^4.6.1",
24     "chart.js": "^3.9.1",
25     "ng2-charts": "^4.0.1",
26     "ngx-mask": "^14.2.3",
27     "ngx-pagination": "^6.0.2",
28     "ngx-toastr": "^15.2.0"
29   },
30   "devDependencies": {
31     "@angular-devkit/build-angular": "^14.2.3",
32     "@angular/cli": "~14.2.3",
33     "@angular/compiler-cli": "^14.2.0",
34     "@types/googlemaps": "^3.43.3",
35     "@types/jasmine": "~4.0.0",
36     "jasmine-core": "~4.3.0",
37     "karma": "~6.4.0",
38     "karma-chrome-launcher": "~3.1.0",
39     "karma-coverage": "~2.2.0",
40     "karma-jasmine": "~5.1.0",
41     "karma-jasmine-html-reporter": "~2.0.0",
42     "typescript": "~4.7.2"
43   }
44 }

```

Figura 10 – Arquivo *package.json* do front-end.

Para que os pacotes instalados sejam reconhecidos pelo *Angular* seus respectivos módulos devem ser carregados no arquivo *app.module.ts*

```

TS app.modules.ts
web-front > src > app > TS app.modules.ts > ...
29
You, 2 days ago | 1 author (You)
30 @NgModule({
31   declarations: [
32     AppComponent,
33     AddProductsComponent,
34     ProductsDetailsComponent,
35     ProductsListComponent,
36     LoginComponent,
37     HomeComponent,
38     AddStoragesComponent,
39     StoragesDetailsComponent,
40     StoragesListComponent,
41     StoragesManagerComponent,
42     AddStoragesProductsComponent,
43     RmvStoragesProductsComponent,
44     StoragesReportComponent
45   ],
46   imports: [
47     BrowserModule,
48     AppRoutingModule,
49     FormsModule,
50     HttpClientModule,
51     NgxPaginationModule,
52     NgxMaskModule.forRoot(),
53     NgbModule,
54     BrowserAnimationsModule,
55     ToastrModule.forRoot({
56       preventDuplicates: true,
57       progressBar: true,
58       countDuplicates: true,
59       extendedTimeout: 2000,
60       positionClass: 'toast-bottom-right',
61     })
62     ,
63     GoogleMapsModule,
64     NgChartsModule,
65   ],
66   providers: [httpInterceptorProviders],
67   bootstrap: [AppComponent]
68 })
69 export class AppModule { }

```

Figura 11 – Arquivo *app.modules.ts* do front-end.

Os diretórios do projeto são organizados automaticamente a partir dos comandos Angular Cli, mantendo um padrão estrutural de fácil entendimento.

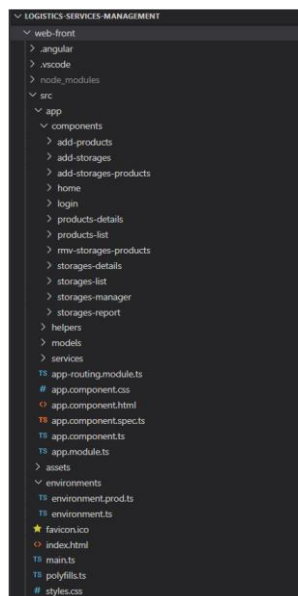


Figura 12 – Estrutura de diretórios do projeto *Web-Front*.

O código fonte da POC está disponível em: <https://github.com/bitols/logistics-services-management>.

6. Avaliação da Arquitetura

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção, visando todos os cenários levantados a seguir.

6.1. Análise das abordagens arquiteturais

A arquitetura adotada é baseada em microsserviços e aborda os princípios de alta coesão e baixo acoplamento. Ela possui APIs de serviços CRUD com bancos de dados próprios, regras de negócios isoladas e com responsabilidades únicas. Fica a cargo dos serviços de BFF e Orquestração a responsabilidade de interação e comunicação entre as diversas APIs.

Baseando-se nas restrições da arquitetura e nos requisitos não-funcionais, o maior desafio em ter uma estrutura granular é manter a segurança de informações sigilosas, a transparência na comunicação entre as peças envolvidas e o alto desempenho na execução de fluxos de processos e integridade nos dados armazenados.

6.2. Cenários

1. Cenário 1: Toda requisição de página privada do sistema deve validar as credenciais de acesso do usuário. Quando o sistema identificar a falta de autenticação, ou expiração da sessão, deve redirecionar o usuário para a página de login;
2. Cenário 2: Após realizar o cadastro de um novo depósito o sistema deve:
 - 2.1. De forma assíncrona e orquestrada, identificar as coordenadas geográficas do local a partir do endereço informado e atualizar o cadastro;
 - 2.2. De forma assíncrona e orquestrada, criar o registro no Relatório de Depósitos com sua capacidade de armazenamento;
3. Cenário 3: Após realizar a associação uma determinada quantidade de produtos ao depósito o sistema deve:

- 3.1. De forma assíncrona e orquestrada calcular e atualizar os indicadores de armazenamento no Relatório de depósitos;
4. Cenário 4: Ao visualizar os indicadores dos depósitos, o sistema deve prover dados atualizados;
5. Cenário 5: Ao executar a visualização de depósitos e seus indicadores, o sistema deve ter um tempo de resposta aceitável abaixo de 3 segundos.

Na priorização foi utilizado o método de Árvore de Utilidade reduzida e com prioridades. Por esse procedimento a categorização ocorre de acordo os atributos de qualidade a que estão relacionados e então qualificados por importância e complexidade, considerando as características do requisito.

Atributo de Qualidade	Cenário	Importância	Complexidade
Segurança	Cenário 1	Alta	Baixa
Interoperabilidade	Cenário 2	Alta	Média
Interoperabilidade	Cenário 3	Alta	Média
Interoperabilidade	Cenário 4	Alta	Média
Desempenho	Cenário 5	Alta	Alta

6.3. Avaliação

O processo de avaliação dos cenários levantados na sessão anterior é realizado aplicando o padrão estímulo-resposta com o intuito de identificar os pontos de risco, sensibilidade e tradeoffs.

Cenário 1:

Atributo de Qualidade	Segurança
Requisito de Qualidade	O sistema deve impedir acesso sem autenticação.
Preocupação	Não permitir acessos sem as devidas credenciais.
Estímulo	Acessar página privada sem possuir uma sessão válida.
Mecanismo	<i>Web-Front, Web-BFF e Credentials-Service.</i>
Medida de Resposta	O sistema deve redirecionar para a tela de login quando não for

	requisição autenticada.
Riscos	Falha de autenticação pode causar acesso indevido a informações sensíveis por indivíduos mal-intencionados.
Pontos de sensibilidade	N/A
Tradeoffs	N/A

Evidências:

- Ao acessar o sistema pela primeira vez, é apresentada a tela inicial com a barra de navegação apresentando apenas o nome da Boa entrega no lado esquerdo e o menu para *Login* no lado direito:



Figura 13 – Página inicial sem *Login*.

- Ao informar credenciais incorretas o processo de login irá barrar a entrada no sistema:

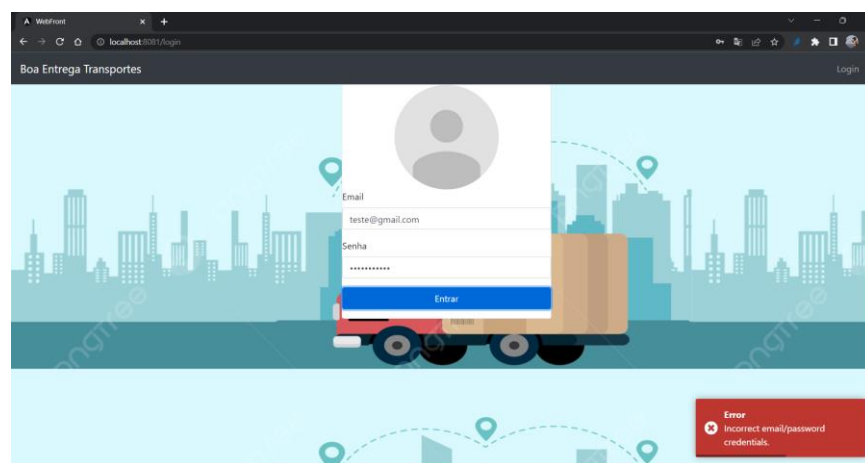


Figura 14 – Tentativa de *Login* com credenciais incorretas.

- Ao acessar o sistema após processo de *Login*, é apresentada a tela inicial com a barra de navegação apresentando o nome da Boa entrega e os menus “Cadastros”, “Processos” e “Gestão” no lado esquerdo e o nome da empresa do usuário no lado direito:



Figura 15 – Página inicial da aplicação após realizar processo de *Login*

- Já *logado* no sistema, o usuário acessa normalmente as páginas privadas da aplicação, como por exemplo o “Cadastro de Depósitos”:

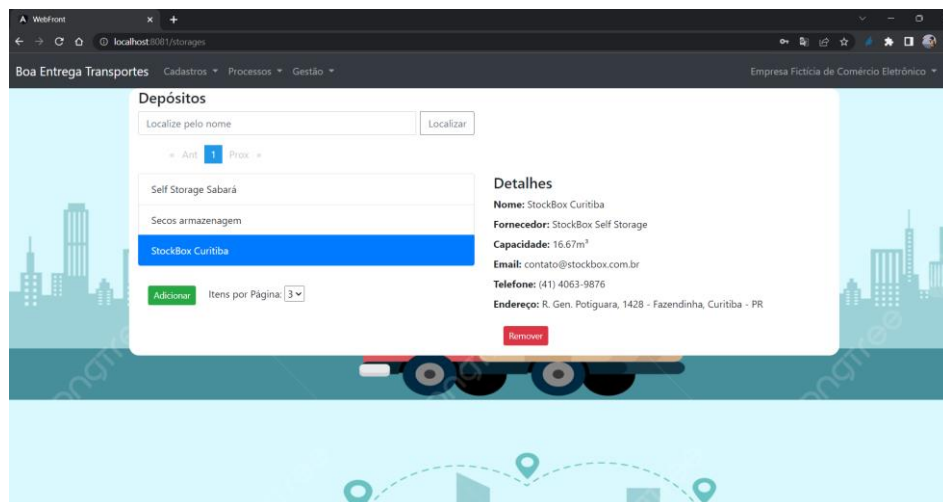


Figura 16 – Página “Cadastro de Depósitos”

- Ao tentar acessar a página “Cadastro de Depósitos” sem estar devidamente *logado* no sistema, ocorre o redirecionamento para a página de *Login* com uma notificação informando problemas ao requisitar depósitos:

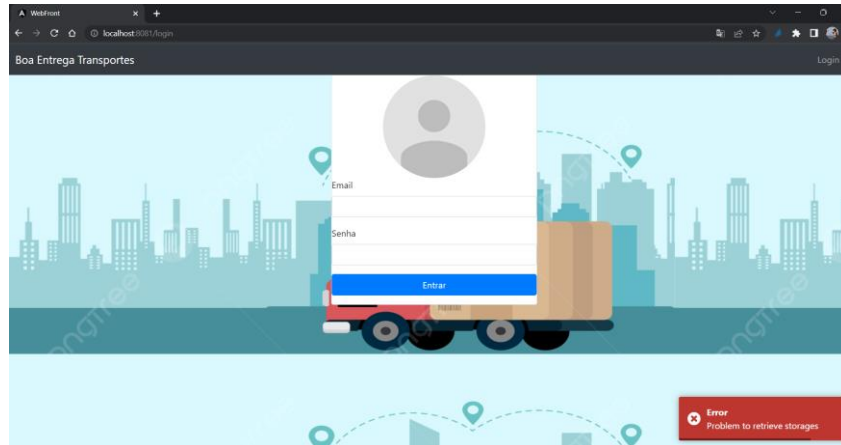


Figura 17 – Redirecionamento para o *Login* ao tentar acessar uma página privada

Este controle de acesso é possível graças a dois mecanismos:

- No lado do back-end, ao *middleware* desenvolvido no *Web-BFF* que intercepta todas as requisições recebidas e encaminha o token ao *Credentials-Service* para validação. Quando o token é negado, um *status code* 401 é retornado pelo bff;

```

TS isAuthenticated.ts X
web-bff-service > src > shared > infra > middlewares > TS isAuthenticated.ts > [e] isAuthenticated > [e] credential
You, 2 months ago | 1 author (You)
1 import AppError from '@shared/errors/AppErrors';
2 import { NextFunction, Request, Response } from 'express';
3 import { container } from 'tsyringe';
4 import { ValidateSessionsUseCase } from '@modules/sessions/useCases/ValidateSessionsUseCase';
5 export const isAuthenticated = async (
6   request: Request,
7   response: Response,
8   next: NextFunction,
9 ): Promise<void> => {
10   const authHeader = request.headers.authorization;
11   if (!authHeader) {
12     throw new AppError('JWT is missing.', 401);
13   }
14
15   const validateSession = container.resolve(ValidateSessionsUseCase);
16   const credential = await validateSession.execute({ token: authHeader });
17
18   request.credential = credential;
19   return next();
20 };

```

Figura 18 – *Middleware* no *Web-bff* que intercepta requisições realizadas.

```

TS ValidateSessionsUseCase.ts X
web-bff-service > src > modules > sessions > useCases > TS ValidateSessionsUseCase.ts > ValidateSessionsUseCase > execute
You, 2 months ago | 1 author (You)
1 import AppErrors from '@shared/errors/AppErrors';
2 import { inject, injectable } from 'tsyringe';
3 import { IValidationSessions } from '../domain/models/requests/IValidationSessions';
4 import { ICredentials } from '../domain/models/responses/ICredentials';
5 import { ISessionsRepository } from '../domain/repositories/ISessionsRepository';
6
7 @injectable()
8 export class ValidateSessionsUseCase {
9   constructor(
10     @inject('SessionsRepository')
11     private credentialsRepository: ISessionsRepository,
12   ) {}
13
14   public async execute(data: IValidationSessions): Promise<ICredentials> {
15     const credential = await this.credentialsRepository.validateSession(data);
16
17     if (!credential) {
18       throw new AppErrors('JWT expired.', 401);
19     }
20
21     return credential;
22   }
23 }
24

```

Figura 19 – Use case usado pelo *middleware* para encaminhar o token ao *Credentials-Service*

- No lado do front-end, ao *handler* desenvolvido para interceptar uma requisição feita ao back-end e adicionar, quando houver, o token ao seu cabeçalho. Então, aguardando o retorno da chamada, que ao identificar um *status code* 401 promove a limpeza da sessão e consequentemente a saída do sistema.

```

TS http.interceptor.ts X
web-front > src > app > handler > TS http.interceptor.ts > ...
6
7 You, 4 weeks ago | 1 author (You)
8 @Injectable()
9 export class HttpRequestInterceptor implements HttpInterceptor {
10   constructor(private sessionsService: SessionsService, private router: Router) {}
11
12   intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
13     const token = this.sessionsService.getToken();
14
15     req = req.clone({
16       setHeaders: {
17         Authorization: `Bearer ${token}`
18       }
19     });
20
21     return next.handle(req).pipe(
22       catchError((error) => {
23         if (
24           error instanceof HttpErrorResponse &&
25           error.status === 401
26         ) {
27           return this.handle401Error(req, next);
28         }
29         return throwError(() => error);
30       })
31     );
32
33     private handle401Error(request: HttpRequest<any>, next: HttpHandler) {
34       if (this.sessionsService.isLoggedIn()) {
35         this.sessionsService.clean();
36       }
37       this.router.navigate(['login']);
38       return next.handle(request);
39     }
40   }
41
42
43
44   export const httpInterceptorProviders = [
45     { provide: HTTP_INTERCEPTORS, useClass: HttpRequestInterceptor, multi: true },
46   ];
47

```

Figura 20 – *Handler* no *web-front* que valida o retorno do *web-bff*

Cenário 2:

Atributo de Qualidade	Interoperabilidade
Requisito de Qualidade	O sistema deve permitir a comunicação entre os serviços da arquitetura.
Preocupação	Utilizar padrão na troca de informações entre os serviços da arquitetura, e iniciar assincronamente dois processos orquestrados para geração de informação.
Estímulo	Cadastrar um novo depósito.
Mecanismo	<i>Web-Front</i> , <i>Web-BFF</i> , <i>Storages-Service</i> , <i>Suppliers-Service</i> , <i>Reports-Service</i> , e <i>Orchestrator-Service</i> .
Medida de Resposta	Após cadastrar um novo depósito, o sistema deve identificar a localização geográfica a partir do endereço informado e atualizar o registro do novo cadastro. O sistema também deve criar um registro no relatório de indicadores com sua capacidade inicial contratada.
Riscos	A Api do Google, consultada no processo, pode não encontrar uma localização a partir dos dados informados. A api de relatórios <i>Reports-Service</i> pode estar indisponível durante o registro.
Pontos de sensibilidade	Dados de localização do depósito e indicadores inconsistentes.
Tradeoffs	N/A

Evidências:

- Com a tela de cadastros de depósitos aberta, o usuário clica em “Adicionar”;

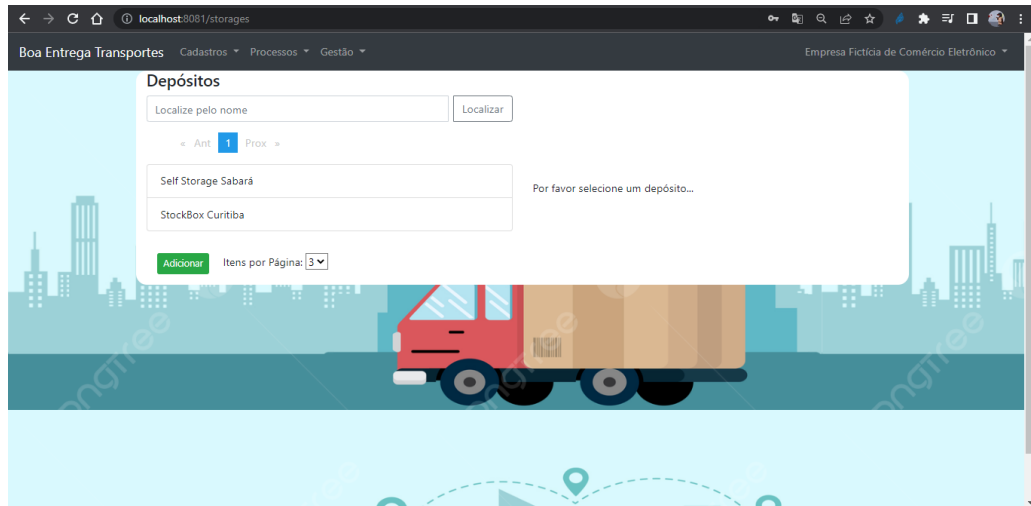


Figura 21 – Web-Front, página Cadastro de Depósitos

- Então o *Web-Front* apresenta um modal com formulário para preenchimento dos dados do novo depósito para registro;
- Quando o usuário clicar em “Enviar” o *Web-Front* faz uma requisição de cadastro de novo depósito ao *Web-BFF*, passando os dados informados no formulário;

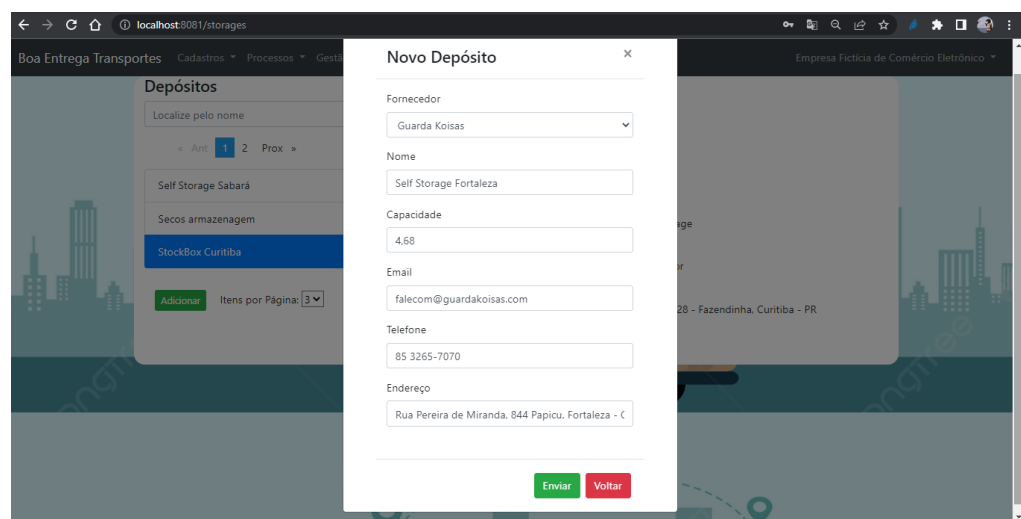


Figura 22 – Web-Front, modal Novo Depósito

- O *Web-BFF* recebe a requisição, e encaminha os dados de cadastro de depósito ao *Storages-Service*, que por sua vez registra a informação em banco de dados e envia mensagens aos tópicos do Kafka para identificação de localização a partir de endereço e registro de relatório de capacidade do novo depósito respectivamente;

```
[INFO] 12/10/2022, 4:17:21 PM Request POST /storages/ received
[INFO] [StoragesController][create] storage: Self Storage Fortaleza
[INFO] [GetSuppliersUseCase][execute] Request {"id":"6278ed499abb0b3b8d94456c"} from service: 68.159ms
[INFO] [CreateStoragesUseCase]
[execute] Request register {"name":"Self Storage Fortaleza","capacity":4.68,"email":"falecom@guardakoissas.com","phone":"8532657070","address":"Rua P
eira de Miranda, 844 Papicu, Fortaleza - CE","supplierId":"6278ed499abb0b3b8d94456c","senderId":"62901dd61bf53edbc1baalea"} to service: 250.696ms
[INFO] [StoragesController][create] Mount response: 0.842ms
[INFO] [StoragesController][create] Total execution: 322.126ms
```

Figura 23 – Web-BFF, recebendo e direcionando requisição ao Storages-Service

```
[INFO] 12/10/2022, 4:17:21 PM Request POST /storages/ received
[INFO] [StoragesController][create] name:Self Storage Fortaleza
[INFO] [CreateStoragesUseCase]
[execute] Register {"name":"Self Storage Fortaleza","capacity":4.68,"email":"falecom@guardakoissas.com","phone":"8532657070","address":"Rua Pereira d
e Miranda, 844 Papicu, Fortaleza - CE","supplierId":"6278ed499abb0b3b8d94456c","senderId":"62901dd61bf53edbc1baalea"} to data base: 96.997ms
[INFO] [CreateStoragesUseCase]
[execute] Produce message {"id":"6394b1115b821adb1cedabe8","address":"Rua Pereira de Miranda, 844 Papicu, Fortaleza - CE"} to topic storage-
location: 48.94ms
[INFO] [CreateStoragesUseCase]
[execute] Produce message {"storageId":"6394b1115b821adb1cedabe8","senderId":"62901dd61bf53edbc1baalea","capacity":4.68} to topic storage-
capacity: 6.23ms
[INFO] [StoragesController][create] Mount response: 0.625ms
[INFO] [StoragesController][create] Total execution: 154.219ms
```

Figura 24 – Storages-Service, registro em banco de dados e disparo de mensagens para o Kafka

- O *Orchestrator-service* fica consumindo os tópicos do Kafka e recebe as mensagens em seus devidos *handlers*;
- Ao receber a mensagem no tópico *storage-location* o *Orchestrator-Service* identifica a posição geográfica do depósito a partir do endereço consultando a api do Google Geo-code e envia os dados para o *Storages-Service* atualizar o registro;
- Ao receber a mensagem no tópico *storage-capacity* o *Orchestrator-Service* calcula a capacidade inicial do depósito e envia os dados para o *Reports-Service* atualizar o relatório.

```
[INFO] [storageLocation.handler] message received:{"id":"6394b1115b821adb1cedabe8","address":"Rua Pereira de Miranda, 844 Papicu, Fortaleza - CE"}
[INFO] [storageLocation.handler] Request location for address: Rua Pereira de Miranda, 844 Papicu, Fortaleza - CE to Google's Geocode Api: 349.084ms
[INFO] [storageLocation.handler] Update storageId: 6394b1115b821adb1cedabe8 location: {"lat":-3.7409289,"lng":-38.4791607}: 120.709ms
[INFO] [storageLocation.handler] Total execution: 473.502ms
```

Figura 25 – Orchestrator-Service, busca de localização e requisição ao Storages-Service

```
[INFO] [storageCapacityControl.handler] message received:
{"storageId":"6394b1115b821adb1cedabe8","senderId":"62901dd61bf53edbc1baalea","capacity":4.68}
[INFO]
[storageCapacityControl.handler] Update report storagesCapacityControl: {"storageId":"6394b1115b821adb1cedabe8","senderId":"62901dd61bf53edbc1baalea",
,"capacity":4.68}: 195.063ms
[INFO] [storageCapacityControl.handler] Total execution: 196.032ms
```

Figura 26 – Orchestrator-Service, atualização de capacidade do depósito e requisição ao Reports-Service

```
[INFO] 12/10/2022, 4:17:22 PM Request PATCH /storages/6394b1115b821adb1cedabe8/location received
[INFO] [StoragesController][updateLocation] storageId:6394b1115b821adb1cedabe8 location: {"lat":-3.7409289,"lng":-38.4791607}
[INFO] [StoragesController][updateLocation] Mount response: 0.657ms
[INFO] [StoragesController][updateLocation] Total execution: 114.089ms
```

Figura 27 – Storages-Service, atualização de localização do depósito em banco de dados


```
[INFO] 12/10/2022, 4:17:21 PM Request POST /reports/storages received
[INFO] [ReportsController][registerStoragesReport] {"storageId":"6394b1115b821adb1cedabe8","capacity":4.68,"stored":0,"usage":0,"products":
[],"value":0,"senderId":"62901dd61bf53edbc1baalea","items":0}
[INFO] [RegisterStoragesReportUseCase][execute] Register {"storageId":"6394b1115b821adb1cedabe8","capacity":4.68,"stored":0,"usage":0,"products":
[],"value":0,"senderId":"62901dd61bf53edbc1baalea","items":0} to data base: 31.813ms
[INFO] [ReportsController][registerStoragesReport] Mount response: 0.39ms
[INFO] [ReportsController][registerStoragesReport] Total execution: 80.048ms
```

Figura 28 – *Reports-Service*, atualização de relatório de capacidade do depósito em banco de dados

- Ao acessar “Processos”, “Controlar Depósitos”, e seleccionar o depósito cadastrado, será possível visualizar no lado direito da página, o mapa com a marcação da localização de acordo com o endereço informado no cadastro.

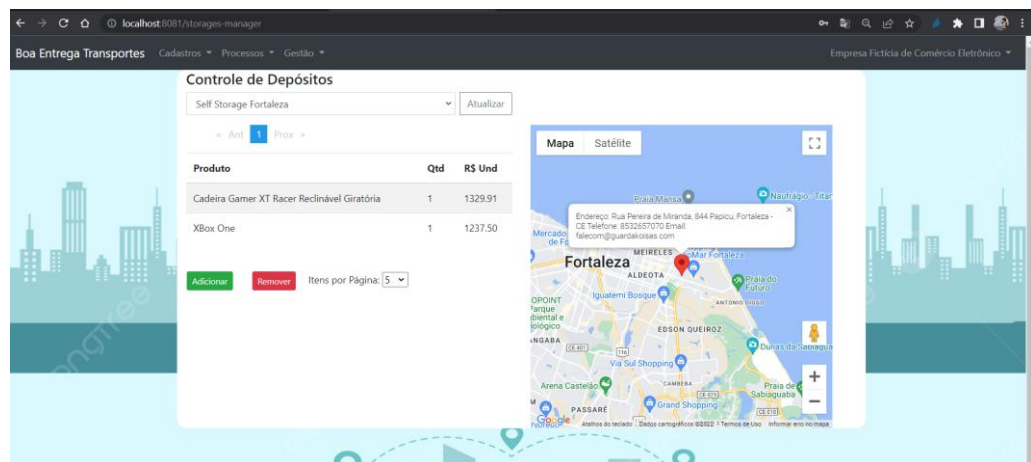


Figura 29 – *Web-Front*, apresentação da localização do depósito no mapa

Cenário 3:

Atributo de Qualidade	Interoperabilidade
Requisito de Qualidade	O sistema deve permitir a comunicação entre os serviços da arquitetura.
Preocupação	Utilizar padrão na troca de informações entre os serviços da arquitetura.
Estímulo	Associar um produto ao depósito
Mecanismo	<i>Web-Front</i> , <i>Web-BFF</i> , <i>Products-Service</i> , <i>Storages-Service</i> , <i>Reports-Service</i> e <i>Orchestrator-Service</i>
Medida de Resposta	O sistema deve atualizar os indicadores do depósito na API de Relatórios de acordo com as atualizações ocorridas na API de Depósitos.
Riscos	O <i>Orchestrator-Service</i> sobrecarregar, devido ao alto número de requisições e cálculos, e o processo ser interrompido. A api de

	relatórios <i>Reports-Service</i> pode estar indisponível durante o registro.
Pontos de sensibilidade	Dados de indicadores do depósito inconsistente.
Tradeoffs	N/A

Evidências:

- Após o usuário acessar “Processos”, “Controlar Depósitos” e selecionar o depósito cadastrado, é possível visualizar do lado esquerdo da página uma tabela de produtos em estoque, suas respectivas quantidades armazenadas e seus valores unitários. Abaixo desta tabela, está o botão “Adicionar”;
- Ao clicar em “Adicionar”, o *Web-Front* apresenta um modal com formulário para preenchimento dos dados de associação de produtos ao estoque;

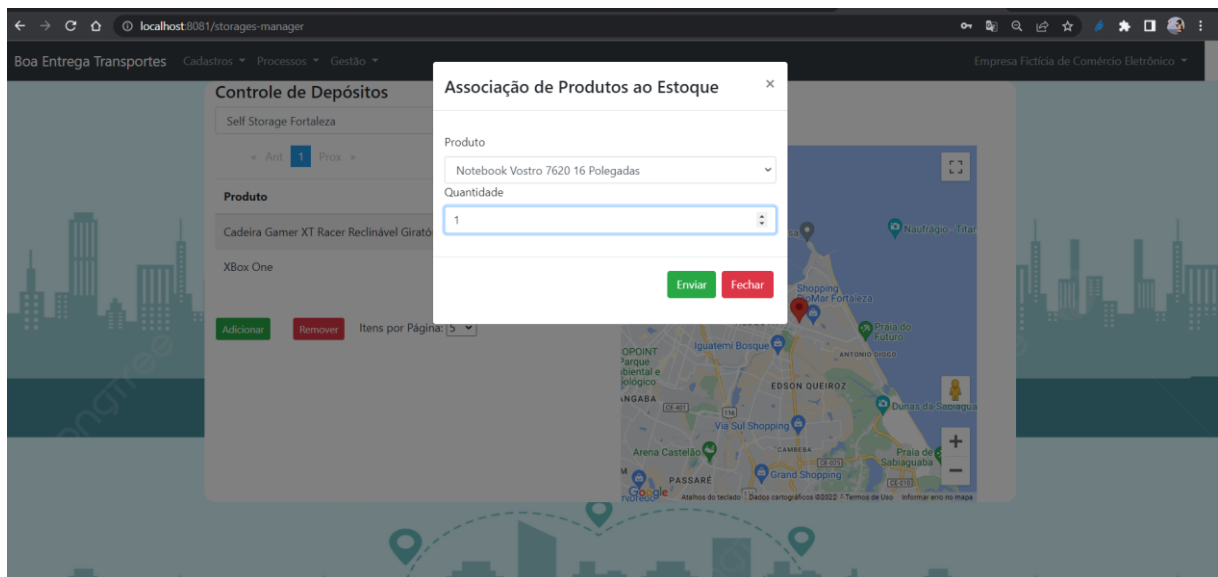


Figura 30 – *Web-BFF*, recebendo e direcionando requisição ao *Storages-Service*

- Quando o usuário clicar em “Enviar” o *Web-Front* faz uma requisição de associação de uma certa quantidade de produtos ao depósito ao *Web-BFF*, passando os dados informados no formulário;
- O *Web-BFF* recebe a requisição, e encaminha os dados de cadastro de produtos no depósito ao *Storages-Service*, que por sua vez registra a informação em banco de dados e envia mensagem ao tópico do Kafka para atualização e registro de relatório de capacidade do depósito;

```
[INFO] 12/13/2022, 11:53:48 AM Request POST /storages/6394b1115b821adb1cedabe8/products received
[INFO][StoragesController][addStoredProducts] storageId: 6394b1115b821adb1cedabe8, productId: 633b516d61f2bd4fb26b5d02, quantity: 1
[INFO][GetStoragesUseCase][execute] Request {"id":"6394b1115b821adb1cedabe8"} from service: 35.518ms
[INFO][AddStoragesProductsUseCase]
[execute] Request register {"storageId":"6394b1115b821adb1cedabe8","productId":"633b516d61f2bd4fb26b5d02","quantity":1} to service: 174.953ms
[INFO][StoragesController][addStoredProducts] Total execution: 211.587ms
```

Figura 31 – *Web-BFF*, recebimento da requisição de cadastro e encaminhamento ao *Storages-Service*

```
[INFO] 12/13/2022, 11:53:48 AM Request POST /storages/products received
[INFO][StoragesController][addStoreProduct] storageId:6394b1115b821adb1cedabe8 add product: Notebook Vostro 7620 16 Polegadas
[INFO][CreateStoragesProductsUseCase][execute] Produce message {"increase":true,"storedProduct":
{"id":"633b516d61f2bd4fb26b5d02","name":"Notebook Vostro 7620 16 Polegadas","height":0.09,"width":0.34,"length":0.48,"value":9499,"storageId":"6394b1115b821adb1cedabe8","productId":"633b516d61f2bd4fb26b5d02","quantity":1}} to topic storage-capacity: 103.214ms
[INFO][CreateStoragesProductsUseCase]
[execute] Register {"name":"Notebook Vostro 7620 16 Polegadas","height":0.09,"width":0.34,"length":0.48,"value":9499,"productId":"633b516d61f2bd4fb26b5d02","storageId":"6394b1115b821adb1cedabe8","quantity":1} to data base: 104.985ms
[INFO][StoragesController][addStoreProduct] Mount response: 0.307ms
[INFO][StoragesController][addStoreProduct] Total execution: 106.253ms
```

Figura 32 – *Storages-Service*, registro em banco de dados e disparo de mensagens para o Kafka

- O *Orchestrator-service* fica consumindo os tópicos do Kafka e recebe as mensagens em seus devidos *handlers*;
- Ao receber a mensagem no tópico *storage-products* o *Orchestrator-Service* busca as informações do relatório do depósito, soma a quantidade de produtos recebidas pelo tópico, recalcula os índices do depósito e envia os dados para o *Reports-Service* atualizar o relatório.

```
[INFO][storageProductControl.handler] message received:{"increase":true,"storedProduct":
{"id":"633b516d61f2bd4fb26b5d02","name":"Notebook Vostro 7620 16 Polegadas","height":0.09,"width":0.34,"length":0.48,"value":9499,"storageId":"6394b1115b821adb1cedabe8","productId":"633b516d61f2bd4fb26b5d02","quantity":1}}
[INFO][storageProductControl.handler] Update report storageProductControl: {"increase":true,"storedProduct":
{"id":"633b516d61f2bd4fb26b5d02","name":"Notebook Vostro 7620 16 Polegadas","height":0.09,"width":0.34,"length":0.48,"value":9499,"storageId":"6394b1115b821adb1cedabe8","productId":"633b516d61f2bd4fb26b5d02","quantity":1}}: 315.89ms
[INFO][storageProductControl.handler] Total execution: 316.556ms
```

Figura 33– *Orchestrator-Service*, atualização de quantidade de produtos e capacidade do depósito, e requisição ao *Reports-Service*

```
[INFO] 12/13/2022, 11:53:48 AM Request POST /reports/storages received
[INFO][ReportsController]
[registerStoragesReport] {"storageId":"6394b1115b821adb1cedabe8","capacity":4.68,"stored":0.22021400000000002,"usage":4.705427350427351,"products":
[{"id":"63161acd28d01b5804dee398","name":"Cadeira Gamer XT Racer Reclinável Giratória","items":1,"value":1329.91,"stored":0.18447000000000002,"usage":3.9416666666666673},
{"id":"63161b5c28d01b5804dee39a","name":"XBox One","items":1,"value":1237.5,"stored":0.021056000000000002,"usage":0.44991452991453},
{"id":"633b516d61f2bd4fb26b5d02","name":"Notebook Vostro 7620 16 Polegadas","items":1,"value":9499,"stored":0.014688,"usage":0.31384615384615383}],{"value":12066.41,"senderId":"62901dd61bf53edbc1baa1ea","items":3}
[INFO][RegisterStoragesReportUseCase]
[execute] Register {"storageId":"6394b1115b821adb1cedabe8","capacity":4.68,"stored":0.22021400000000002,"usage":4.705427350427351,"products":
[{"id":"63161acd28d01b5804dee398","name":"Cadeira Gamer XT Racer Reclinável Giratória","items":1,"value":1329.91,"stored":0.18447000000000002,"usage":3.9416666666666673},
{"id":"63161b5c28d01b5804dee39a","name":"XBox One","items":1,"value":1237.5,"stored":0.021056000000000002,"usage":0.44991452991453},
{"id":"633b516d61f2bd4fb26b5d02","name":"Notebook Vostro 7620 16 Polegadas","items":1,"value":9499,"stored":0.014688,"usage":0.31384615384615383}],{"value":12066.41,"senderId":"62901dd61bf53edbc1baa1ea","items":3} to data base: 28.119ms
[INFO][ReportsController][registerStoragesReport] Mount response: 0.612ms
[INFO][ReportsController][registerStoragesReport] Total execution: 120.457ms
```

Figura 34 – *Reports-Service*, atualização de relatório de capacidade do depósito e quantidade de produtos em banco de dados

- Ao final do processo de associação de produtos, o *Web-Front* recebe do *Web-BFF* o retorno de sucesso no cadastro, e assim o produto recém adicionado passa a ser visível na tela de controle do depósito.

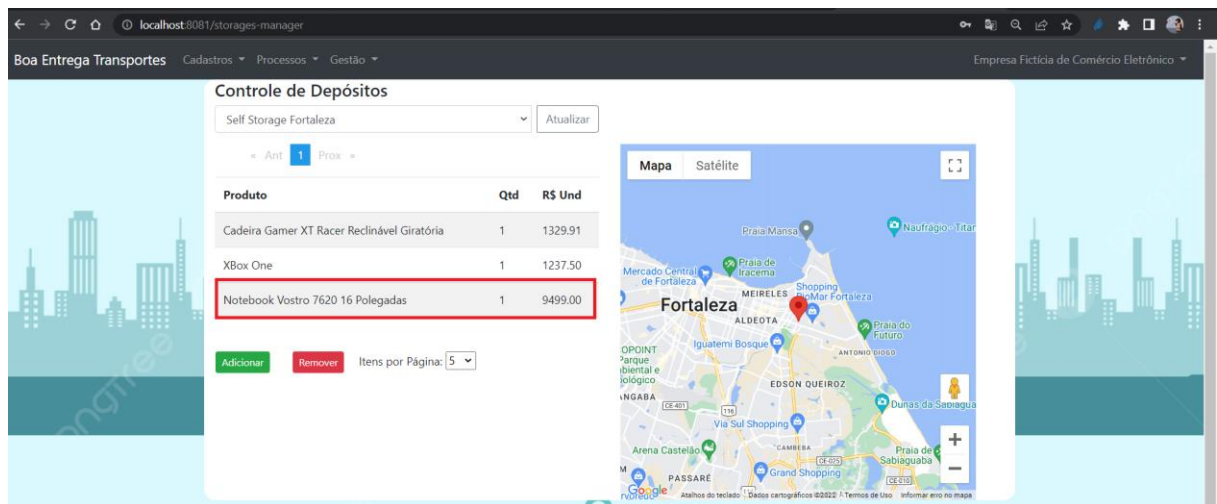


Figura 35 – *Web-Front*, apresentação do produto na tela de controle de depósito

Cenário 4:

Atributo de Qualidade	Interoperabilidade
Requisito de Qualidade	O sistema deve permitir a comunicação entre os serviços da arquitetura
Preocupação	Utilizar padrão na troca de informações entre os serviços da arquitetura
Estímulo	Consultar dados de depósitos e seus indicadores
Mecanismo	<i>Web-Front</i> , <i>Web-BFF</i> , <i>Suppliers-Service</i> , <i>Storages-Service</i> e <i>Reports-Service</i>
Medida de Resposta	O sistema deve buscar informações de depósitos relacionados ao remetente na API de Depósitos, buscar os indicadores dos depósitos na API de Relatórios e realizar a composição dos dados para apresentação.
Riscos	A API de Depósitos ou a de Relatórios estarem indisponíveis.
Pontos de sensibilidade	Falha na requisição
Tradeoffs	N/A

Evidências:

- Ao acessar o menu “Gestão”, “Indicadores de Depósitos”, o *Web-Front* requisita o *Web-BFF* a lista de depósitos cadastradas pelo remetente no sistema;



Figura 36 – *Web-Front*, página inicial, menu Gestão -> Indicadores de Depósitos.

- Quando o *Web-BFF* recebe a requisição, ele encaminha requisições ao *Suppliers-Service* e ao *Storages-Service* que respectivamente buscam os dados necessários nos seus bancos de dados e retornam a ao *Web-BFF*, que por sua vez monta a informação e retorna ao *Web-Front*;

```
[INFO] 10/30/2022, 6:26:56 PM Request GET /senders/62901dd61bf53edbc1baalea/storages received
[INFO] [SendersController][getStorages] senderId: 62901dd61bf53edbc1baalea
[INFO] [GetAllSuppliersUseCase][execute] Request all from service: 82.171ms
[INFO] [GetStoragesBySenderUseCase][execute] Request all {"senderId":"62901dd61bf53edbc1baalea"} from service: 77.789ms
[INFO] [SendersController][getStorages] Mount response: 2.683ms
[INFO] [SendersController][getStorages] Total execution: 164.509ms
```

Figura 37 – *Web-BFF* recebendo e processando a requisição de depósitos do remetente.

```
[INFO] 10/30/2022, 6:26:56 PM Request GET /suppliers/ received
[INFO] [SuppliersController][getAll]
[INFO] [GetAllSuppliersUseCase][execute] Request all from data base: 44.038ms
[INFO] [SuppliersController][getAll] Mount response: 4.285ms
[INFO] [SuppliersController][getAll] Total execution: 51.184ms
```

Figura 38 – *Suppliers-Service* recebendo e processando a requisição de fornecedores.

```
[INFO] 10/30/2022, 6:26:56 PM Request GET /storages/senders/62901dd61bf53edbc1baalea received
[INFO] [StoragesController][getAllBySenderId] senderId:62901dd61bf53edbc1baalea
[INFO] [GetAllStoragesBySenderIdUseCase][execute] Request all {"senderId":"62901dd61bf53edbc1baalea"} from data base: 41.075ms
[INFO] [StoragesController][getAllBySenderId] Mount response: 3.781ms
[INFO] [StoragesController][getAllBySenderId] Total execution: 47.337ms
```

Figura 39 – *Storages-Service* recebendo e processando a requisição de depósitos do remetente.

- Quando o *Web-Front* recebe as informações de depósitos cadastrados do remente, é apresentada a tela para selecionar um depósito específico para visualizar os indicadores. Ao selecionar um depósito, o *Web-Front* encaminha ao *Web-BFF* a requisição para obter os dados de relatórios;

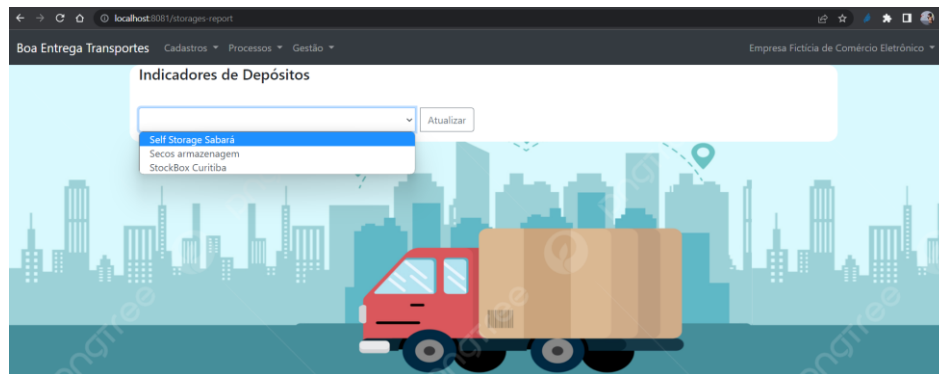


Figura 40 – *Web-Front*, Indicadores de Depósitos, seletor de depósito.

- Quando o *Web-BFF* recebe a requisição, ele encaminha requisições ao *Storages-Service* e ao *Reports-Service*, que respectivamente buscam os dados necessários nos seus bancos de dados e retornam a ao *Web-BFF*, que por sua vez monta a informação e retorna ao *Web-Front*;

```
[INFO] 10/30/2022, 6:27:06 PM Request GET /storages/634b62827a54f78f32878610/reports received
[INFO] [StoragesController][getStoragesReport] storageId:634b62827a54f78f32878610
[INFO] [GetStoragesUseCase][execute] Request {"id":"634b62827a54f78f32878610"} from service: 0.053ms
[INFO] [GetStoragesReportUseCase][execute] Request {"storageId":"634b62827a54f78f32878610"} from service: 78.934ms
[INFO] [StoragesController][getStoragesReport] Mount response: 0.541ms
[INFO] [StoragesController][getStoragesReport] Total execution: 125.11ms
```

Figura 41 – *Web-BFF* recebendo e processando a requisição de relatório de depósito.

```
[INFO] 10/30/2022, 6:27:06 PM Request GET /storages/634b62827a54f78f32878610 received
[INFO] [StoragesController][getById] id:634b62827a54f78f32878610
[INFO] [GetStoragesUseCase][execute] Request {"id":"634b62827a54f78f32878610"} from data base: 39.374ms
[INFO] [StoragesController][getById] Mount response: 0.624ms
[INFO] [StoragesController][getById] Total execution: 40.791ms
```

Figura 42 – *Storages-Service* recebendo e processando a requisição de informação do depósito.

```
[INFO] 10/30/2022, 6:27:06 PM Request GET /reports/storages/634b62827a54f78f32878610 received
[INFO] [ReportsController][getStoragesReport] storageId:634b62827a54f78f32878610
[INFO] [GetStoragesReportUseCase][execute] Request {"storageId":"634b62827a54f78f32878610"} from data base: 44.254ms
[INFO] [ReportsController][getStoragesReport] Mount response: 4.035ms
[INFO] [ReportsController][getStoragesReport] Total execution: 50.352ms
```

Figura 43 – *Reports-Service* recebendo e processando a requisição de relatório de depósito.

- Ao receber o resultado da pesquisa, o *Web-Front* monta na tela um dashboard no formato cockpit para o usuário, auxiliando nas tomadas de decisão.

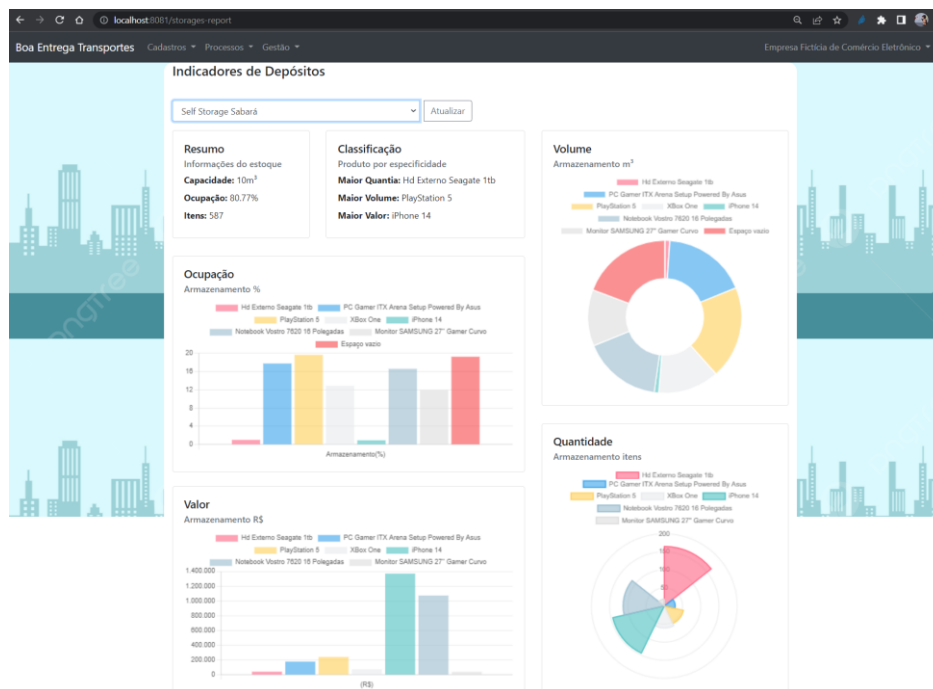


Figura 44 – *Web-Front*, Indicadores de Depósitos, Dashboard.

Cenário 5:

Atributo de Qualidade	Desempenho
Requisito de Qualidade	O sistema deve possuir bom desempenho
Preocupação	O sistema deve ter um tempo de resposta ao usuário abaixo de 3 segundos
Estímulo	Consultar dados de depósitos e seus indicadores
Mecanismo	<i>Web-Front, Web-BFF, Suppliers-Service Storages-Service e Reports-Service</i>
Medida de Respsota	O sistema deve ter rapidez na busca e composição das informações
Riscos	Com a alta demanda os tempos de resposta na comunicação podem aumentar
Pontos de sensibilidade	Latência de rede
Tradeoffs	N/A

Evidências:

- No processo executado no cenário 4, por se tratar da mesma execução de processo, é possível efetuar a análise de dados para o cenário 5, porém dando ênfase aos tempos de execução;
- Na requisição que o *Web-BFF* processou a busca de Depósitos é possível verificar o tempo total de execução: 164,509ms, atendendo os requisitos não funcionais;

```
[INFO] 10/30/2022, 6:26:56 PM Request GET /senders/62901dd61bf53edbc1baalea/storages received
[INFO] [SendersController][getStorages] senderId: 62901dd61bf53edbc1baalea
[INFO] [GetAllSuppliersUseCase][execute] Request all from service: 82.171ms
[INFO] [GetStoragesBySenderUseCase][execute] Request all {"senderId":"62901dd61bf53edbc1baalea"} from service: 77.789ms
[INFO] [SendersController][getStorages] Mount response: 2.683ms
[INFO] [SendersController][getStorages] Total execution: 164.509ms
```

Figura 45 – Tempos de execução do *Web-BFF* para consulta de depósitos.

- Na requisição que o *Suppliers-Service* processou a busca de fornecedores é possível analisar o tempo total de execução: 51,18ms, atendendo os requisitos não funcionais;

```
[INFO] 10/30/2022, 6:26:56 PM Request GET /suppliers/ received
[INFO] [SuppliersController][getAll]
[INFO] [GetAllSuppliersUseCase][execute] Request all from data base: 44.038ms
[INFO] [SuppliersController][getAll] Mount response: 4.285ms
[INFO] [SuppliersController][getAll] Total execution: 51.184ms
```

Figura 46 – Tempos de execução do *Suppliers-Service* para consulta de fornecedores.

- Na requisição que o *Storages-Service* processou a busca de depósitos do remetente é possível analisar o tempo total de execução: 47,337ms, atendendo os requisitos não funcionais;

```
[INFO] 10/30/2022, 6:26:56 PM Request GET /storages/senders/62901dd61bf53edbc1baalea received
[INFO] [StoragesController][getAllBySenderId] senderId:62901dd61bf53edbc1baalea
[INFO] [GetAllStoragesBySenderIdUseCase][execute] Request all {"senderId":"62901dd61bf53edbc1baalea"} from data base: 41.075ms
[INFO] [StoragesController][getAllBySenderId] Mount response: 3.781ms
[INFO] [StoragesController][getAllBySenderId] Total execution: 47.337ms
```

Figura 47 – Tempos de execução do *Storages-Service* para consulta de depósitos.

- Na requisição que o *Web-BFF* processou a busca de Relatório de Depósito é possível verificar o tempo total de execução: 125,11ms, atendendo os requisitos não funcionais;


```
[INFO] 10/30/2022, 6:27:06 PM Request GET /storages/634b62827a54f78f32878610/reports received
[INFO] [StoragesController][getStoragesReport] storageId:634b62827a54f78f32878610
[INFO] [GetStoragesUseCase][execute] Request {"id":"634b62827a54f78f32878610"} from service: 0.053ms
[INFO] [GetStoragesReportUseCase][execute] Request {"storageId":"634b62827a54f78f32878610"} from service: 78.934ms
[INFO] [StoragesController][getStoragesReport] Mount response: 0.541ms
[INFO] [StoragesController][getStoragesReport] Total execution: 125.11ms
```

Figura 48 – Tempos de execução do *Web-BFF* para consulta de relatórios de depósito.

- Na requisição que o *Storages-Service* processou a busca de informação do depósito é possível analisar o tempo total de execução: 40,791ms, atendendo os requisitos não funcionais;

```
[INFO] 10/30/2022, 6:27:06 PM Request GET /storages/634b62827a54f78f32878610 received
[INFO] [StoragesController][getById] id:634b62827a54f78f32878610
[INFO] [GetStoragesUseCase][execute] Request {"id":"634b62827a54f78f32878610"} from data base: 39.374ms
[INFO] [StoragesController][getById] Mount response: 0.624ms
[INFO] [StoragesController][getById] Total execution: 40.791ms
```

Figura 49 – Tempos de execução do *Storages-Service* para consulta de relatórios de depósito.

- Na requisição que o *Reports-Service* processou a busca de relatório de indicadores do depósito é possível analisar o tempo total de execução: 50,352ms, atendendo os requisitos não funcionais.

```
[INFO] 10/30/2022, 6:27:06 PM Request GET /reports/storages/634b62827a54f78f32878610 received
[INFO] [ReportsController][getStoragesReport] storageId:634b62827a54f78f32878610
[INFO] [GetStoragesReportUseCase][execute] Request {"storageId":"634b62827a54f78f32878610"} from data base: 44.254ms
[INFO] [ReportsController][getStoragesReport] Mount response: 4.035ms
[INFO] [ReportsController][getStoragesReport] Total execution: 50.352ms
```

Figura 50 – Tempos de execução do *Reports-Service* para consulta de relatórios de depósito.

6.4. Resultado

Considerando os atributos de qualidade escolhidos, os testes realizados e as evidências coletadas, a avaliação permitiu comprovar que a arquitetura proposta atende as necessidades do projeto. Nas evidências são possíveis constatar que os requisitos não funcionais avaliados são atendidos. Nesta avaliação foram considerados os requisitos apresentados no quadro abaixo.

Requisitos Não Funcionais	Testado	Homologado
RNF1 Segurança: O sistema deve possuir mecanismo e segurança no acesso	SIM	SIM
RNF02 Interoperabilidade: O sistema deve permitir que os serviços se comuniquem para realizar ações necessárias	SIM	SIM
RNF03 Desempenho: O sistema deve ser rápido	SIM	SIM

Ao avaliar a segurança, fica evidente o bloqueio ao acesso, e até mesmo a ausência de meios de acesso às páginas privadas do sistema. O processo de autorização que ocorre a cada requisição que passa pelo *Web-BFF*, ao validar a assinatura do token consultando o *Credentials-Service*, é transparente ao usuário e desempenha o papel de barreira quando identifica a ausência do token ou sua expiração.

A interoperabilidade foi avaliada em três cenários distintos, e se comprova a capacidade dos distintos serviços envolvidos na arquitetura se comunicarem de forma harmônica. A arquitetura não está atrelada apenas a um modelo de comunicação entre os microsserviços, pois:

- o modelo *API Composition* é utilizado quando o *Web-BFF* busca dados dos distintos serviços e faz uma junção em memória para retornar informação requisitada;
- o padrão SAGA de orquestração é utilizado no *Orchestrator-Service*, como o próprio nome do serviço já sugere, em que se direciona eventos aos serviços que participam de um processo para executarem transações locais em sequência.

Durante o desenvolvimento da POC e da análise dos requisitos, foram identificados alguns pontos limitantes na arquitetura proposta que são inerentes a uma aplicação orientada a serviços distribuídos que possuem seus próprios bancos de dados. Esta situação é descrita pelo teorema CAP, que afirma que um sistema de armazenamento distribuído não pode garantir simultaneamente consistência, disponibilidade e partição tolerante a falhas.

Partindo deste pressuposto e considerando que nenhum sistema distribuído está protegido contra falhas de rede, e por isso a partição deve ser tolerada, o arquiteto deve optar entre disponibilidade ou consistência no tratamento dos dados. Quando opta por consistência renuncia-se à disponibilidade, pois deve haver consenso entre os bancos de dados envolvidos, gerando bloqueios/ interrupções para manter a integridade das informações em qualquer sinal

de falha, até que se garanta a devida propagação da informação. Ao se optar por disponibilidade, o processo desconsidera a consistência da informação nos bancos de dados distribuídos, sempre mantendo o sistema ativo, intermitente e respondendo às demandas do cliente mesmo que ocorram falhas na propagação de dados entre os serviços.

Neste projeto foi optado por disponibilidade sobre consistência, com o intuito de manter o sistema rápido e capaz de processar grandes volumes de informações em um curto espaço de tempo sem gerar interrupções ao usuário. Porém, diante da importância da consistência da informação para a área de negócio que a aplicação se propõe atender, a evolução da arquitetura abordando fluxos paralelos para tratamento de exceções e consolidação da informação entre os bancos de dados distribuídos se faz necessária.

7. Conclusão

Neste projeto foi apresentado um protótipo arquitetural de um sistema logístico orientado a serviços distribuídos. Constatou-se que os objetivos do projeto foram alcançados quando os requisitos avaliados foram devidamente atendidos, porém foram identificadas e apresentadas algumas limitações que não impactam a aceitação da proposta atual, visto que podem ser consideradas uma evolução natural da arquitetura. Como sugestão para uma próxima versão da aplicação, fica o desafio de projetar e implementar um fluxo independente para tratamento de exceções e consolidação de dados entre os nós envolvidos.

REFERÊNCIAS

GOULART, Verci Douglas Garcia; DE CAMPOS, Alexandre. **Logística de Transporte – Gestão Estratégica no Transporte de Cargas**. São Paulo: Érica, 2018.

IORDACHE, Anca. **Docker Compose: From Local to Amazon ECS**. <<https://www.docker.com/blog/docker-compose-from-local-to-amazon-ecs/>>, 2021.

KRUISZ, Manuel. **API Gateway vs Backend For Frontend**. <<https://www.manuelkruisz.com/blog/posts/api-gateway-vs-bff>>, 2019.

LAM, Lauro. **Boom do e-commerce: vendas online das PMEs movimentaram R\$2,3 bilhões em 2021**. <<https://olhardigital.com.br/2022/01/14/pro/vendas-online-das-pmes-movimentaram-r23-bilhoes-em-2021/>>, 2022.

LEWIS, James. FOWLER, Martin. **Microservices**. <<https://martinfowler.com/articles/microservices.html>>, 2014.

IORDACHE, Anca. **Docker Compose: From Local to Amazon ECS**. <>

RABELO, Eduardo. **Arquitetura de Software: O Teorema CAP**. <<https://oieduardorabelo.medium.com/arquitetura-de-software-o-teorema-cap-73b3dc45dd34>>, 2019.

SOUSA REIS, Marco Antônio de. **Uma Arquitetura de Big Data as a Service Baseada no Modelo de Nuvem Privada**. <https://repositorio.unb.br/bitstream/10482/33759/1/2018_MarcoAnt%C3%B4niodeSousaReis.pdf>, Brasília, 2018.

APÊNDICES

URL de acesso ao sistema:

<http://logis-loadb-ed0xydtihm3e-885ab48e11a65558.elb.us-east-1.amazonaws.com:8081>

URL do projeto no GitHub:

<https://github.com/bitols/logistics-services-management>

URL da apresentação da POC no Youtube:

https://www.youtube.com/watch?v=d0_09ulmPl8