INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO
RIO GRANDE DO SUL
CAMPUS CANOAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RAFAEL DE LUCA

# Catálogo Online e Administração Simplificada: Desenvolvimento de uma Plataforma Web para Divulgação e Gerenciamento de Produtos de Higiene e Limpeza

Canoas, 15 de dezembro de 2023.

RAFAEL DE LUCA

# Catálogo Online e Administração Simplificada: Desenvolvimento de uma Plataforma Web para Divulgação e Gerenciamento de Produtos de Higiene e Limpeza

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Campus Canoas.

Prof. Dr. Dieison Soares Silveira
Orientador

Canoas, 15 de dezembro de 2023.

Ministério da Educação
Secretaria de Educação Profissional, Científica e Tecnológica
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul
*Campus* Canoas

# ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 07 dias do mês de dezembro de 2023, às 10 horas, em sessão pública na sala E7 do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, Campus Canoas, na presença da Banca Examinadora presidida pelo(a) Professor(a): Dr. Dieison Soares Silveira e
composta pelos examinadores:
1. MSc. Tatiane Coreixas de Moraes,
2. Dr. Gustavo Neuberger,
o aluno **Rafael de Luca** apresentou o Trabalho de Conclusão de Curso intitulado: **Catálogo Online e Administração Simplificada: Desenvolvimento de uma Plataforma Web para Divulgação e Gerenciamento de Produtos de Higiene e Limpeza** como requisito curricular indispensável para a integralização do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Após reunião em sessão reservada, a Banca Examinadora deliberou e decidiu pela **APROVAÇÃO** do referido trabalho, divulgando o resultado formalmente ao aluno e demais presentes e eu, na qualidade de Presidente da Banca, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.

Dieison Soares Silveira
Assinado digitalmente por Dieison Soares Silveira
DN: C=BR, OU=IFRS, O=Campus Canoas, CN=Dieison Soares Silveira, E=coord.tads@canoas.ifrs.edu.br
Razão: Eu estou aprovando este documento
Localização: Canoas-RS
Data: 2023-12-07 15:10:04
Foxit Reader Versão: 9.2.0

Presidente da Banca Examinadora

gov.br Documento assinado digitalmente
TATIANE COREIXAS DE MORAES
Data: 08/12/2023 10:15:42-0300
Verifique em https://validar.iti.gov.br

Examinador 01

Gustavo Neuberger
Assinado digitalmente por Gustavo Neuberger
DN: C=BR, OU=IFRS Campus Canoas, O=Gustavo Neuberger, CN=Gustavo Neuberger, E=gustavo.neuberger@canoas.ifrs.edu.br
Razão: Eu sou o autor deste documento
Localização: Canoas (RS)
Data: 2023-12-08 17:24:44
Foxit Reader Versão: 9.7.2

Examinador 02

gov.br Documento assinado digitalmente
RAFAEL DE LUCA
Data: 15/12/2023 20:20:19-0300
Verifique em https://validar.iti.gov.br

Rafael de Luca

Aluno

## RESUMO

A divulgação adequada dos serviços prestados e oferecidos por uma empresa e seus clientes é um fator fundamental para o sucesso de qualquer empresa. O objetivo desse trabalho foi de criar uma aplicação web para uma distribuidora e fabricante de produtos de higiene e limpeza divulgar seus produtos e captar clientes. Além de criar uma interface agradável para o cliente que deseja pesquisar por produtos, também foi objetivo desse trabalho desenvolver uma ferramenta de fácil uso para os administradores do catálogo para inclusão, edição e exclusão dos produtos no banco de dados. A aplicação utilizou uma arquitetura cliente-servidor. No lado do cliente o sistema utilizou a linguagem de marcação HTML e a linguagem de estilização CSS para realizar a interface estática. E utilizou a linguagem de programação JavaScript e framework React para as lógicas de negócio no lado cliente. No lado do servidor foi utilizado Java como linguagem de programação, junto com o framework Spring Boot. Utilizamos um banco de dados em memória H2 no ambiente testes e o banco de dados PostgreSQL no ambiente de produção. Os objetivos dessa aplicação foram validados com usuários finais.



**Palavras-chave:** Divulgação, Catálogo Online, Aplicação web.

## ABSTRACT

Adequate promotion of the services provided and offered by a company and its customers is a fundamental factor in the success of any company. The main goal of this work was to create a web application for a distributor and manufacturer of hygiene and cleaning products to promote its products and attract customers. In addition to creating a pleasant interface for customers who want to search for products, the objective of this work was also to develop an easy-to-use tool for catalog administrators to include, edit and delete products in the database. The application used client-server architecture. On the client side, the system used the HTML markup language and the CSS styling language to realize the static interface. The application used JavaScript programming language and React framework for client-side business logic. On the server side, Java was used as a programming language, along with the Spring Boot framework. We use an H2 in-memory database in the testing environment and the PostgreSQL database in the production environment. The objectives of this application were validated with end users.



**Key-words:** Promotion, Online Catalog, Web Application

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATUAS E SIGLAS

| | |
|---|---|
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| CSS | Cascade Style Sheet |
| CORS | Cross Origin Resource Sharing |
| DTO | Data Transfer Object |
| IOS | Iphone Operating System |
| JPA | Java Persistence API |
| JSON | JavaScript Object Notation |
| JWT | Json Web Token |
| HTML | Hyper Text Markup Language |
| HTTP | Hypertext Transfer Protocol |
| ORM | Object Relational Mapping |
| POM | Project Object Model |
| REST | Representation State Transfer |
| SMS | Short Message Service |
| UML | Unified Modeling Language |
| URL | Uniform Resource Locator |

**SUMÁRIO**

# 1 INTRODUÇÃO

Atualmente três empresas do grupo Melhoramentos Higiene Ltda utilizam um sistema ERP[1] Desktop, no qual não existe um sistema de divulgação dos produtos da empresa. A empresa não deseja divulgar todos os seus produtos em uma loja virtual devido a restrições contratuais com alguns fornecedores. Esses fornecedores têm distribuidores e atacadistas no Brasil inteiro e devido a diferentes cargas tributárias entre vendas interestaduais, os mesmos não desejam que alguns de seus produtos sejam disponibilizados via loja virtual. Outro problema para implantação de uma loja virtual é que clientes compram os mesmos produtos por diferentes valores, dependendo da quantidade e prazo de pagamento. E, por último, a equipe comercial da Empresa Melhoramentos Higiene acredita que não é profissional ficar passando vários sites de fornecedores diferentes para o cliente buscar por produtos; isso deixa a experiência do usuário muito prejudicada. Visto que, muitas vezes, os sites dos fornecedores estão desatualizados e contém produtos que já saíram de linha ou que a Melhoramentos Higiene não deseja trabalhar com os mesmos.

Assim sendo, o objetivo principal deste de trabalho de conclusão de curso é criar uma plataforma web na qual a distribuidora de nome fantasia Melhoramentos Higiene possa realizar a divulgação desses produtos sem divulgação dos preços dos mesmos. O sistema web deverá possibilitar a todos consultar produtos e filtrar por categoria, embalagem e fornecedor. O sistema deverá possibilitar apenas a usuários autenticados, cadastrar novos produtos, editar produtos cadastrados e excluir produtos que já saíram de linha ou que estão indisponíveis no estoque. Essa interface de cadastro de produtos deverá ser amigável e de fácil utilização. Assim os próprios usuários da equipe do comercial interno poderão administrar a adição ou remoção de novos produtos do portfólio de vendas da empresa, sem ter que ficar realizando chamados para a equipe de informática para realizar o mesmo.

Com relação à metodologia científica esta proposta de conclusão de curso será de abordagem qualitativa, natureza de pesquisa aplicada e uma pesquisa exploratória. Utilizaremos uma observação passiva e participante e uma amostra não probabilística para realizar os testes. A estrutura visual do projeto será desenvolvida em diagrama UML (Linguagem de Modelagem unificada). As linguagens utilizadas serão Java, para o *back-end*; JavaScript, HTML e CSS para *front-end*. Na metodologia também estão as descrições das principais tecnologias e ferramentas de programação utilizadas no projeto, entre elas podemos citar o IntelliJ (ambiente de desenvolvimento Integrado) e o Spring Boot. Para testes em unitários e funcionais será utilizada a ferramenta JUnit[2].

---

[1]Sistema Integrado de Gestão empresarial. Um software que integra vários módulos como comercial, fiscal, faturamento, compras, folha de pagamento, financeiro, entre outros.

[2]Disponível em: <https://junit.org/junit5/> . Acesso em: 20 de setembro de 2023.

## 1.1. JUSTIFICATIVA

A empresa de nome fantasia Melhoramentos Higiene, que é uma empresa de distribuição de produtos de higiene e limpeza, como sanizitantes e cosméticos, para o ramo institucional. Ela atua focada principalmente no atendimento de empresas, como: industrias, restaurantes, hospitais, clínicas, escritórios, supermercados, entre outros. Possui unidades em Curitiba, Itajaí e Porto Alegre, sendo a unidade de Porto Alegre atuante no mercado desde o ano de 1998 e atualmente atende a região da grande Porto Alegre, vale dos Sinos e serra gaúcha. A empresa utiliza um software ERP (sistema integrado de gestão empresarial) onde centraliza a automação de todas as suas rotinas, sejam elas comerciais, financeiras, fiscais, recursos humanos, compras, qualidade, entre outras. O software utilizado é CIGAM[3] da empresa que leva o mesmo nome situada em Novo Hamburgo, Rio Grande do Sul. O sistema apesar de ter muitos recursos é um sistema Desktop que funciona no modo cliente-servidor, onde o servidor local disponibiliza acesso apenas aos computadores da rede local. A CIGAM não disponibiliza um sistema web para seus clientes, apenas um módulo mobile (aplicativo apenas para celular) que deve ser adquirido separadamente ao qual já foi feito teste e não atende as necessidades da empresa Melhoramentos Higiene.

Atualmente a equipe comercial tem que usar sites de fornecedores para divulgar os produtos que a distribuidora vende, o site da própria distribuidora está desatualizado com produtos e fornecedores que a distribuidora já não trabalha mais. A venda da distribuidora é uma venda técnica, na qual o consultor comercial deve oferecer diluidores automáticos de produtos químicos, lavadoras industriais, produto adequado para tratamento de piso, detergente e secante de louça para lavadoras industriais, entre outros. Assim sendo, a equipe comercial não deseja uma loja virtual onde o cliente possa realizar um autoatendimento e efetuar o seu pedido. Deseja apenas uma plataforma onde o cliente possa conhecer os produtos os quais a empresa tem em estoque e que o comprador institucional agende uma visita para o consultor comercial lhe indicar os produtos adequados e fazer uma proposta personalizada.

## 1.2. PROPOSTA DE SOLUÇÃO

O tema desta proposta de trabalho de conclusão de curso será o desenvolvimento de um sistema web que permita aos clientes consultar de forma detalhada os produtos disponíveis para compra e assim agendar uma visita com um consultor de vendas. Apenas usuários autenticados da equipe comercial interna poderão adicionar editar e excluir produtos, fornecedores e categorias de produtos. Apenas o usuário administrador poderá cadastrar usuários que tenham as permissões de adição, edição e exclusão. Usuários não autenticados terão a permissão apenas de listagem de produtos, categorias, contatos e demais informações.

A versão inicial do projeto foi desenvolvida em *localhost* (máquina local) e os testes realizados em um banco de dados em memória. O back-end do sistema será

[3]Disponível em: <https://www.cigam.com.br> . Acesso em: 15 de maio de 2023.

hospedado em uma plataforma na nuvem como o heroku[4] por questões de desempenho e segurança. O *front-end* será hospedado na plataforma netlify[5] .

A principal motivação deste trabalho de conclusão de curso é deixar mais dinâmica e agradável à experiência do usuário de consulta de produtos os quais a distribuidora tem em estoque e mais dinâmica o cadastro, edição e exclusão de produtos os quais a empresa tem em seu portfólio. Uma segunda motivação é melhorar a divulgação da marca da distribuidora de produtos de higiene através de divulgação desse portfólio em sites de busca. E, por último, reduzir o custo da empresa com impressão de folders e catálogos de produtos, os quais ficam desatualizados rapidamente.

## 1.3. OBJETIVOS

### 1.3.1. Objetivo Geral

Implementar um sistema de web de divulgação de produtos.

### 1.3.2. Objetivos Específicos

- Definir as linguagens de programação mais adequadas ao projeto
- Fazer a análise de requisitos.
- Definir as tecnologias a serem utilizadas no projeto.
- Fazer a modelagem de Requisitos.
- Contração de infraestrutura de uma plataforma na nuvem para hospedar o software.
- Realização de testes funcionais e unitários na base de dados de homologação.
- Implantar o sistema em base de homologação em l*ocalhos*t e em base de produção na nuvem.

## 1.4. ORGANIZAÇÃO DO TCC

A seguir, as seções que explicitam mais detalhadamente o problema e a solução proposta. São apresentados o capítulo 2 que apresenta a metodologia da pesquisa, o capítulo 3 apresenta a modelagem do sistema, o capítulo 4 apresenta a arquitetura do backend, o capítulo 5 a arquitetura do frontend, o capítulo 6 os resultados e discussões. Por fim, o capítulo 7 apresenta as considerações finais.

[4]O Heroku é uma plataforma de nuvem como serviço que suporta várias linguagens de programação.

[5]A Netlify é uma empresa de computação em nuvem com sede em São Francisco que oferece hospedagem e serviços de back-end sem servidor para aplicativos da web e sites estáticos.

# 2 METODOLOGIA

Com relação à metodologia científica o seguinte trabalho é classificado como sendo uma pesquisa de abordagem qualitativa. Com relação à natureza da pesquisa trata-se de uma obra de pesquisa aplicada, visto que procura usar tecnologias da área de informática para resolver um problema cotidiano de uma empresa. E, por fim, com relação aos objetivos trata-se de uma pesquisa exploratória. Conforme Gil (2002, p.41) pesquisas exploratórias tem como objetivo proporcionar uma maior familiaridade com o problema, com o objetivo de tornar este problema mais explícito.

Foi utilizada uma técnica de observação passiva e participante, visto que se observaram as rotinas de inserção, atualização e exclusão de produtos. A observação participante inscreve-se em uma abordagem no qual o observador participa ativamente nas atividades de coleta de dados, assim sendo requerida a capacidade do investigador de se adaptar a situação, (PAWLOWSKI, ANDERSEN, TROELSEN E SCHIPPERIJN, 2016). E, posteriormente, propor uma solução e automação para eventuais problemas detectados durante a observação participante.

A amostra utilizada será uma amostra não probabilística para testar a aplicação na base de testes. Serão utilizados dois dos consultores comerciais internos da empresa para testar a versão inicial do produto. Depois de testado e implantado na nuvem a aplicação ficará disponível para todos os consultores da empresa. E depois estendido para os consultores da região do Paraná e Santa Catarina.

A elaboração da estrutura do projeto, para visualização do mesmo, será feita através de diagrama UML. Através do diagrama de objetos e diagrama de entidades e relacionamento. Para modelagem de dados e criação de diagramas UML será utilizado à ferramenta Astah ULM versão community[6].

O software será desenvolvido em localhost e posteriormente feito o *deploy* (implementação na nuvem). O *back-end* fica hospedado no Heroku e o *front-end* na plataforma Netlify, conforme já mencionado na introdução. Para o desenvolvimento do *back-end* serão utilizada a linguagem Java, o framework Spring Boot e a IDEA[7] IntelliJ[8]. Java é uma linguagem de programação orientada a objetos desenvolvida pela Sun Microsystems em 1995. É muito utilizada como linguagem de *back-end* tanto para aplicações WEB como aplicações Desktop[9]. O Spring[10] é um ecossistema

[6]Disponível em: <https://astah.net> . Acesso em: 10 de maio de 2023.

[7]IDEA é ambiente de desenvolvimento integrado que possui várias bibliotecas e ferramentas que agilizam no processo de desenvolvimento de software.

[8]Disponível em: < https://www.jetbrains.com/pt-br/idea/> . Acesso em: 10 de maio de 2023.

[9] Disponível em: < https://docs.oracle.com/en/java/>. Acesso em: 10 de maio de 2023.

[10]Disponível em: < https://spring.io/ >. Acesso em: 15 de maio de 2023.

de *frameworks* extremamente maduro, seguro, confiável e robusto muito utilizado no mercado. Entre os sistemas Spring podemos citar Spring Boot, Spring Framework, Spring Data, SringCloud, Spring Session, Spring for Android, Spring Web Flow, entre outros. Além de agilidade no desenvolvimento, oferece estruturação de camadas, injeção de dependências, bibliotecas, acesso ao banco de dados e vários *design patterns* (padrões de desenvolvimento) utilizados no mercado. O Spring Boot é uma versão enxuta do Spring *framework* que tem como principal objetivo fazer a aplicação rodar o mais rápido possível. Ou seja, ao criar o projeto ele já está pronto para rodar, o que acarreta em uma alta produtividade em projetos Java.

Com relação ao *front-end* será utilizado à linguagem de marcação de hipertexto HTML, muito utilizada em projetos de aplicação web. A linguagem de estilização CSS (*Cascading Style Sheets*) e JavaScript para alguns efeitos de animações e transições de imagens, textos e divs (elementos utilizados para definir containers em uma aplicação web).

O sistema terá três ambientes de desenvolvimento. Os ambientes de desenvolvimento, onde será utilizado um Banco de dados em memória como o H2[11] por motivos de agilidade. O ambiente de testes e homologação foi utilizado o Banco de dados Relacional PostgreSQL[12] para gerar um carga inicial, colocar produtos testes, testes de desempenho e treinamento de novos consultores a usar a ferramenta. Por fim, o ambiente de Produção, onde será o ambiente operacional propriamente dito.

Por fim para testes, serão realizados testes unitários, funcionais e integrados com a ferramenta JUnit, que é uma ferramenta de código aberto utilizada para realização de testes automatizados na linguagem Java.

O projeto utilizará o padrão de camadas Web Services, no qual o frontend se comunica com o backend via arquivos no formato JSON[13] e requisições HTTP[14], conforme a mostra a Figura 1. JSON é um formato baseado em texto padrão para representar dados estruturados com base na sintaxe do objeto JavaScript. JSON são muito utilizados para aplicações web do tipo cliente-servidor. O protocolo HTTP é um protocolo da camada de aplicação utilizado para transmissão de documentos hipermídia. É muito utilizado para comunicação entre servidores web e clientes web. Devido a essa clara separação entre frontend e backend, para que aconteça uma comunicação segura entre essas camadas serão utilizados tokens (dispositivos geradores de senhas); mais especificamente o token JWT[15].

---

[11]H2 é um sistema de gerenciamento de banco de dados relacional escrito em Java. Ele pode ser incorporado a aplicativos Java.

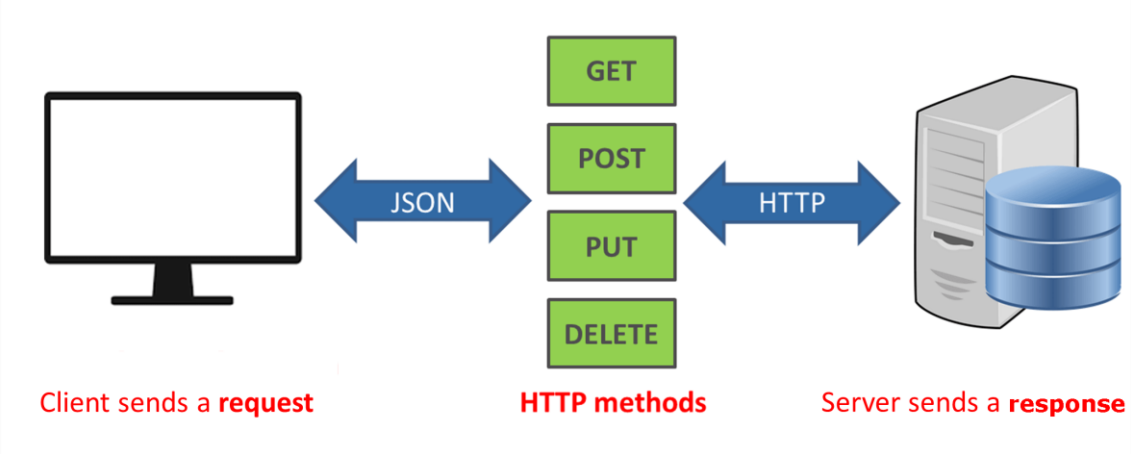
[12]PostgreSQL é um banco de dados relacional de código aberto com reputação de confiabilidade, segurança, robustez de recursos e desempenho.

[13]Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/JSON>. Acesso em: 20 de setembro de 2023.

[14]Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP />. Acesso em: 15 de maio de 2023.

[15]Disponível em: < https://jwt.io>. Acesso em: 20 de setembro de 2023.

Figura 1: Requisições do frontend via arquivo json e resposta do backend




Fonte: Site (2019) [16]

[16] Disponível em: <https://aprendiendoarduino.wordpress.com/2019/10/27/api-rest/>. Acesso em: 17 de julho de 2023.

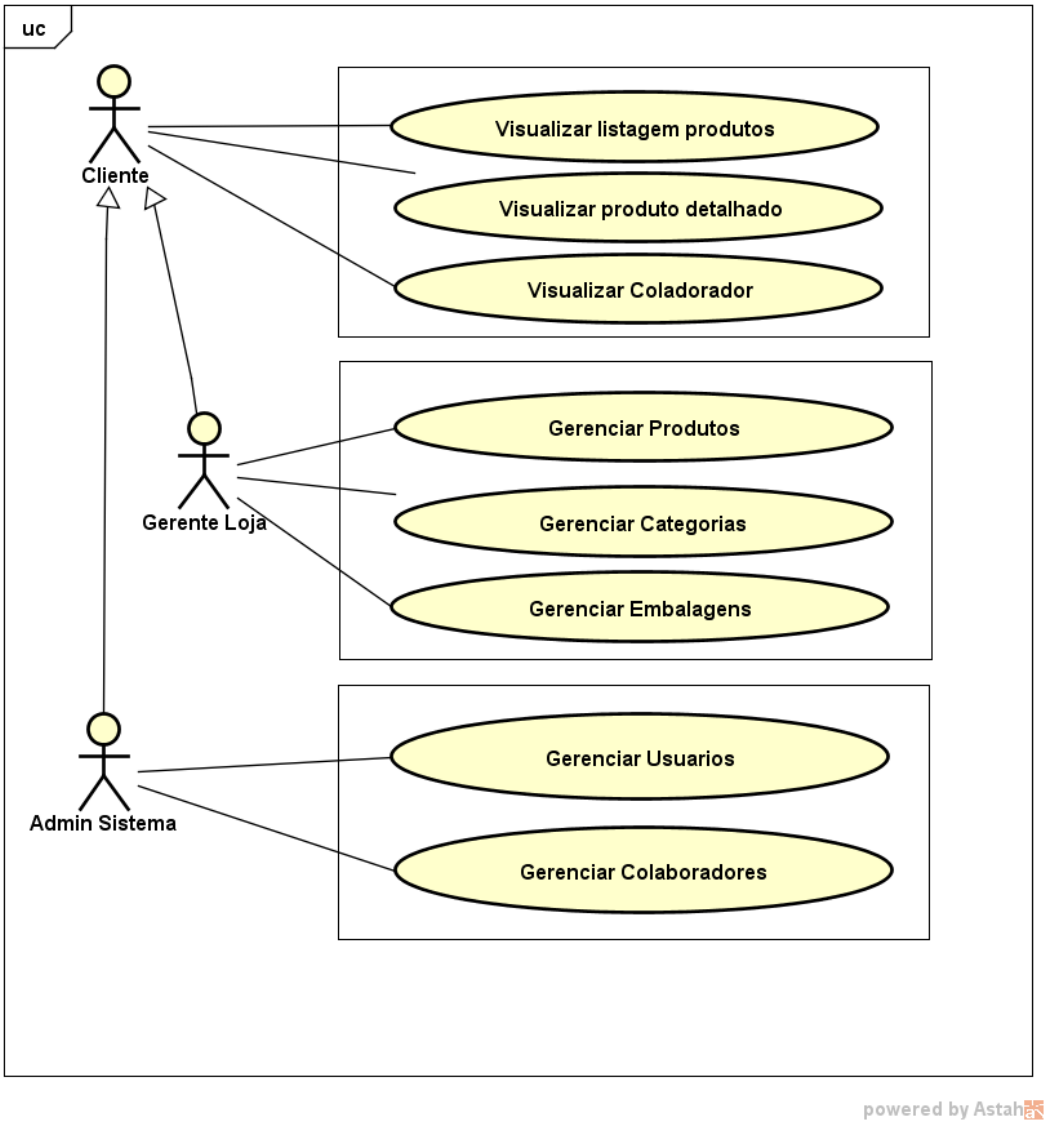
# 3 MODELAGEM DO SISTEMA

## 3.1. DIAGRAMA DE CASOS DE USO

Após a análise dos requisitos da aplicação, foi realizado o diagrama de casos de uso para melhor entendimento das funcionalidades da aplicação. Conforme mostra a Figura 2 a aplicação terá três atores. O usuário cliente que vai consultar os produtos e demais informações do sistema. O usuário gerente de loja que vai ter direito a inserir, editar e excluir produtos, categorias, embalagens e demais entidades da aplicação. E, por último o usuário administrador de sistema que vai ter acesso a consultar e editar os usuários cadastrados na aplicação e também a gerenciar os colaboradores. Por motivos de segurança e lógica de negócio não será permitido excluir usuários da aplicação, apenas atualizações dos seus dados e editar as suas permissões de acesso.

O usuário com permissão de gerente de loja e/ou administrador de sistema consegue fazer as consultas e todas as demais operações que o usuário com a permissão de cliente. Entretanto o usuário cliente não consegue fazer as mesmas operações de gerente de loja e nem de administrador de sistemas.

Figura 2: Diagrama de caso de uso



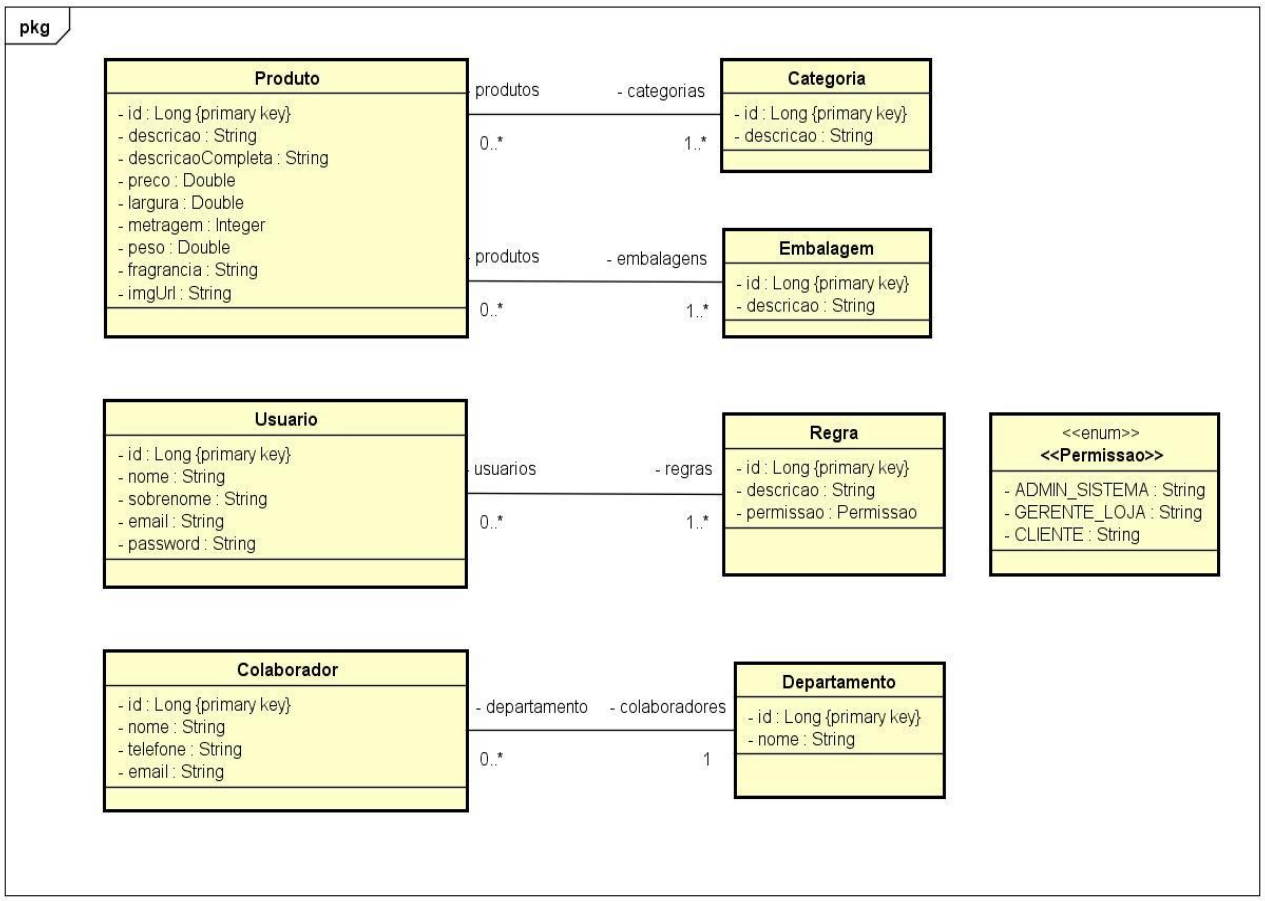

Fonte: Autoria Própria

## 3.2. DIAGRAMA CONCEITUAL

Conforme já mencionado no capítulo anterior a ferramenta escolhida para criação do modelo conceitual (diagrama Entidade-Relacionamento) e diagrama de casos de uso foi o Astah versão community. Conforme exibe a Figura 3, é possível observar que o sistema vai ter sete entidades. Sendo que as duas principais entidades são Produto e a Usuario. As entidades Categoria e Embalagem são relacionadas à entidade produto. A entidade de Regra é relacionada à entidade Usuario. Note que foi criada uma classe do tipo enumerado chamada Permissao, apenas para melhor organização de qual permissão cada usuário vai poder ter no sistema. A entidade Colaborador será uma entidade que se relaciona com a entidade Departamento. Essa entidade Colaborador foi criada apenas para mostrar os contatos diretos dos

colaboradores por setor para o cliente que estiver navegando pela aplicação e necessitar de um atendimento direcionado. O diagrama de entidade relacionamento da Figura 3 também mostra como cada entidade se relacionada. Sendo que um Produto é obrigado a pertencer a pelo menos uma categoria e embalagem, mas pode pertencer a várias categorias ou embalagens. A entidade Usuario deve possuir pelo menos uma Regra; entretanto pode possuir mais de uma Regra. E um Colaborador deve pertencer a um e no máximo um Departamento.

Figura 3: Diagrama Conceitual de Entidade-Relacionamento

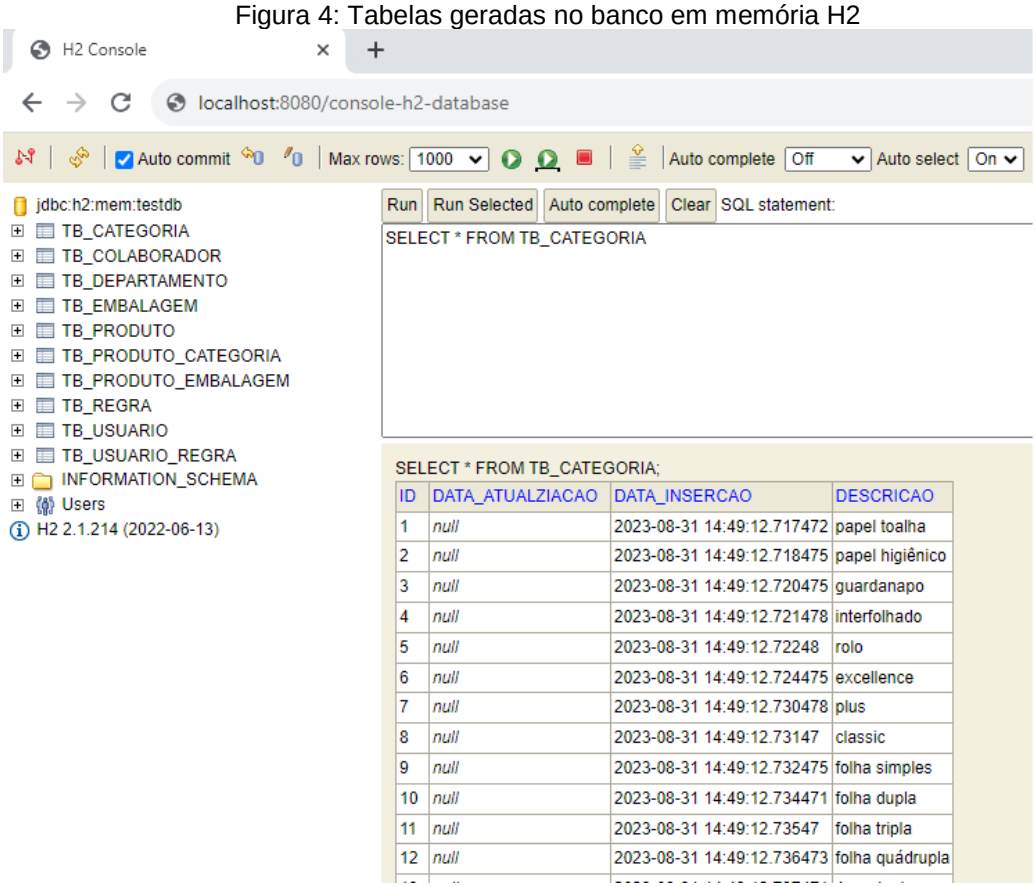



Fonte: Autoria Própria

## 3.3. MODELAGEM DE BANCO DE DADOS

A Figura 4 apresenta o banco de dados H2, pode-se observar nessa figura que o sistema possui dez tabelas. Devido as relação de muitos para muitos entre algumas entidades, será gerada três tabelas de relacionamento (tb_produto_categoria, tb_produto_embalagem e tb_usuario_regra) conforme a Figura 4 abaixo.

Figura 4: Tabelas geradas no banco em memória H2




| ID | DATA_ATUALZIACAO | DATA_INSERCAO | DESCRICAO |
|---|---|---|---|
| 1 | *null* | 2023-08-31 14:49:12.717472 | papel toalha |
| 2 | *null* | 2023-08-31 14:49:12.718475 | papel higiênico |
| 3 | *null* | 2023-08-31 14:49:12.720475 | guardanapo |
| 4 | *null* | 2023-08-31 14:49:12.721478 | interfolhado |
| 5 | *null* | 2023-08-31 14:49:12.72248 | rolo |
| 6 | *null* | 2023-08-31 14:49:12.724475 | excellence |
| 7 | *null* | 2023-08-31 14:49:12.730478 | plus |
| 8 | *null* | 2023-08-31 14:49:12.73147 | classic |
| 9 | *null* | 2023-08-31 14:49:12.732475 | folha simples |
| 10 | *null* | 2023-08-31 14:49:12.734471 | folha dupla |
| 11 | *null* | 2023-08-31 14:49:12.73547 | folha tripla |
| 12 | *null* | 2023-08-31 14:49:12.736473 | folha quádrupla |

Fonte: Autoria Própria

# 4 ARQUITETURA DO BACKEND

Nesta seção serão apresentadas as tecnologias utilizadas na aplicação. As linguagens de programação, os padrões de projeto, os padrões de arquitetura, frameworks, bibliotecas e banco de dados. Um das principais tecnologias utilizados neste trabalho foi o GitHub[17], o qual foi utilizado para versionamento de código tanto do backend como do frontend.

O Backend é onde ficam as funcionalidades da aplicação relacionadas ao servidor e banco de dados. Conforme já mencionado e justificado no capitulo de metodologia a linguagem utilização foi Java e o framework Spring Boot e foi utilizada uma aplicação do tipo cliente-servidor.

## 4.1. APLICAÇÃO REST

Para o padrão de arquitetura de projetos foi utilizado o REST[18]. Uma aplicação para ser considera REST, ela deve obedecer alguns critérios específicos, que visam padronizar o código e a comunicação da aplicação em diferentes plataformas. Entre as principais características de uma aplicação REST pode-se citar o padrão arquitetura cliente-servidor, armazenamento em cache[19], comunicação s*tateless*[20] e utilização de sistemas em camadas.

No sistema em camadas, cada camada do sistema tem uma funcionalidade definida. Essa distribuição de camadas pode ser bem observada de acordo com a Figura 5. Essas camadas são ordenadas hierarquicamente e interagem entre si. Na aplicação foram utilizadas três camadas: controladora, serviço e repositório. A camada controladora recebe as requisições do frontend e exibe envias essas requisições a camada de serviço. No projeto foram utilizados requisições do tipo HTTP e do tipo GET, para realizar buscas; do tipo POST, para realizar inserções de dados; do tipo PUT, para realizar atualizações de dados, e do tipo DELETE, para realizações exclusões do banco de dados. A camada de serviço possui todas as lógicas de negócio, como validação de objetos, conversão de objetos, cálculos, entre outros. E, por último, a camada de repositório que faz acesso ao banco de dados

[17] Disponível em: < https://github.com/>. Acesso em: 05 de maio de 2023.

[18]Disponível em: < https://www.hostinger.com.br/tutoriais/api-restful />. Acesso em: 20 de maio de 2023.

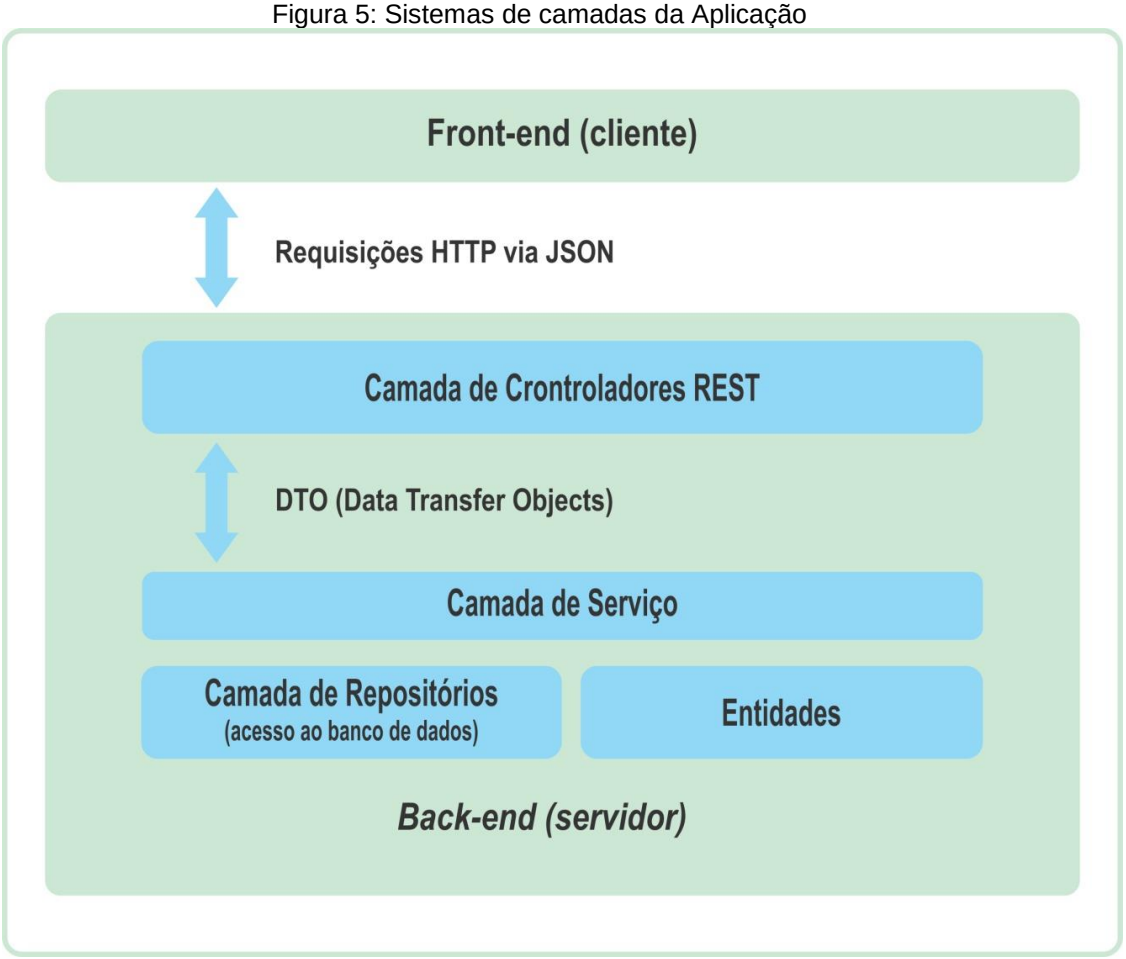
[19] Memória de acesso rápido. Aplicações que fazem uso de memória cache deixam as solicitações entre cliente e servidor mais rápidas.

[20] Cada solicitação contém todos os dados necessários para que seja atendida. Ou seja, uma requisição não depende de uma sessão anterior para que seja atendida corretamente.

propriamente dito; fazendo as operações de seleção, inserção, atualização e exclusão de dados nas tabelas da aplicação. Para não haver um quebra do padrão de camadas REST, cada camada só consegue se comunicar com a camada diretamente acima ou abaixo. Por exemplo, a camada de serviço pode se comunicar com a camada controladora e repositório, mas não é possível instanciar um objeto do tipo repositório direto na camada controladora, isso causaria uma quebra no padrão de camadas.

Como é possível observar na Figura 5 a camada de serviço e a camada de controlador se comunicam através de um objeto denominado DTO[21] (*Data Transfer Object*). DTO é um objeto simples utilizado no padrão de projeto DTO que tem o objetivo de otimizar a comunicação entre cliente e servidor. O DTO agrupa vários parâmetros em uma única chamada, causando assim uma menor sobrecarga na consulta de dados ao servidor. As classes DTO não contêm regras de negócio.

Figura 5: Sistemas de camadas da Aplicação



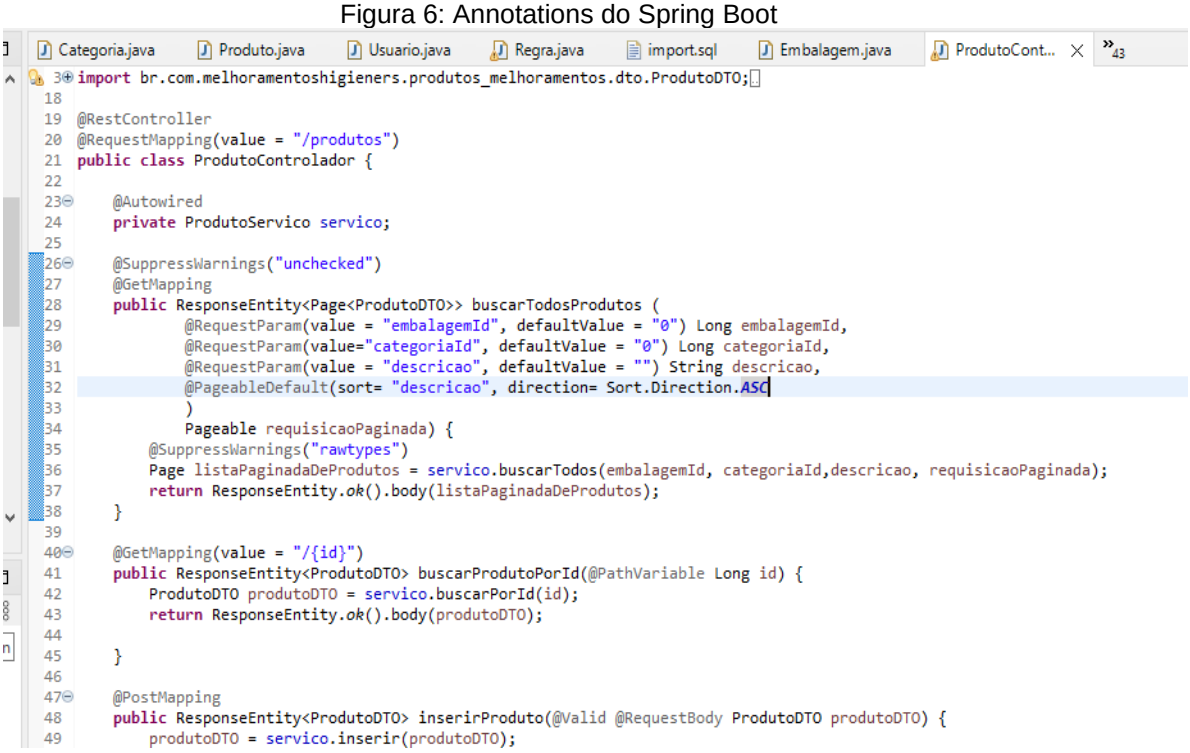

Fonte: Autoria Própria

[21]Disponível em: < https://www.baeldung.com/java-dto-pattern />. Acesso em: 20 de setembro de 2023.

## 4.2. SPRING BOOT

Nesse subcapítulo sobre back-end será explicado um pouco do funcionamento do Spring Boot e das principais bibliotecas do Spring utilizadas no desenvolvimento do sistema. A aplicação Spring Boot faz uso de *annotations*[22] conforme podemos observar na Figura 6. Na Figura 6 é possível verificar a *annotation @RestController*, que identifica que a classe é uma do tipo controladora; a *annotation @GetMapping*, indicando que o método faz um requisição do tipo GET; entre outras *annotations*. Essas *annotations* tornam o código mais enxuto e limpo. As *annotations* usadas em cima das classes permitem identificar se a classe será uma classe do tipo controladora, de serviço, de repositório, uma *interface*[23] (classe abstrata que possuem métodos sem corpo), entre outras. *Annotations* do Spring Boot permitem o programador desenvolver um código de maneira mais modular. *Annotation* em cima de métodos ou parâmetros fazem com que o Spring Boot faça o gerenciamento desses objetos. Entende-se por gerenciamento a instanciação desses objetos e exclusão dos mesmos quando não forem mais necessários.

Figura 6: Annotations do Spring Boot




Fonte: Autoria Própria

Dentro do Spring Boot foram utilizadas algumas bibliotecas do próprio Spring e outras do Java. Como exemplo é possível citar a API para listas[24] do Java, a API

[22]Disponível em: <.https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. Acesso em: 15 de maio de 2023.

[23]Disponível em: <https://www.w3schools.com/java/java_interface.asp />. Acesso em: 20 de setembro de 2023.

[24]Disponível em: <https://docs.oracle.com/javase/8/docs/api/java/util/List.html />. Acesso em: 20 de maio de 2023.

para se trabalhar com datas[25] de hora do Java. Dentro do projeto String Boot é utilizado o gerenciador de dependências Maven[26]. Esse gerenciador é usado para fazer o build[27] do projeto utilizando o conceito de POM (*Project Object Model*), ou seja, ele permite configurar todas as dependências do projeto através de um arquivo único de projeto chamado pow.xml que fica localizado na raiz do projeto conforme podemos observar na Figura 7.

Figura 7: Arquivo pow.xml - dependências e configurações Spring Boot

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.3</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>br.com.melhopramentoshigieners.com.br</groupId>
    <artifactId>produtos_melhoramentos</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>produtos_melhoramentos</name>
    <description>CRUD de produtos para promocao online</description>
    <properties>
        <java.version>17</java.version>
        <spring-cloud.version>2021.0.3</spring-cloud.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
```

Fonte: Autoria Própria

[25]Disponível em: <https://docs.oracle.com/javase/8/docs/api/java/util/Date.html />. Acesso em: 20 de maio de 2023.

[26]Disponível em: <https://maven.apache.org/what-is-maven.html />. Acesso em: 20 de maio de 2023.

[27] A tradução de build seria construir, mas no conceito de tecnologia significa compilar todas as classes java do projeto com a construção de um único arquivo. Assim deixando aplicação pronta para ser instalada em localhost ou na nuvem.

O motivo de utilizar bibliotecas do Spring e do Java é que eles facilitam e agilizam no desenvolvimento, além serem são constantemente testadas e atualizadas pela comunidade.

No projeto foram utilizadas as seguintes dependências gerenciadas pelo maven: spring boot starter data jpa, spring boot starter web, spring boot starter validation, spring, boot starter test, spring boot starter security, spring security test, spring security crypto, spring security oath2 autoconfigure, h2 database, postgresql e junit. Todas as informações sobre essas bibliotecas podem ser acessadas na documentação oficial do spring[28].

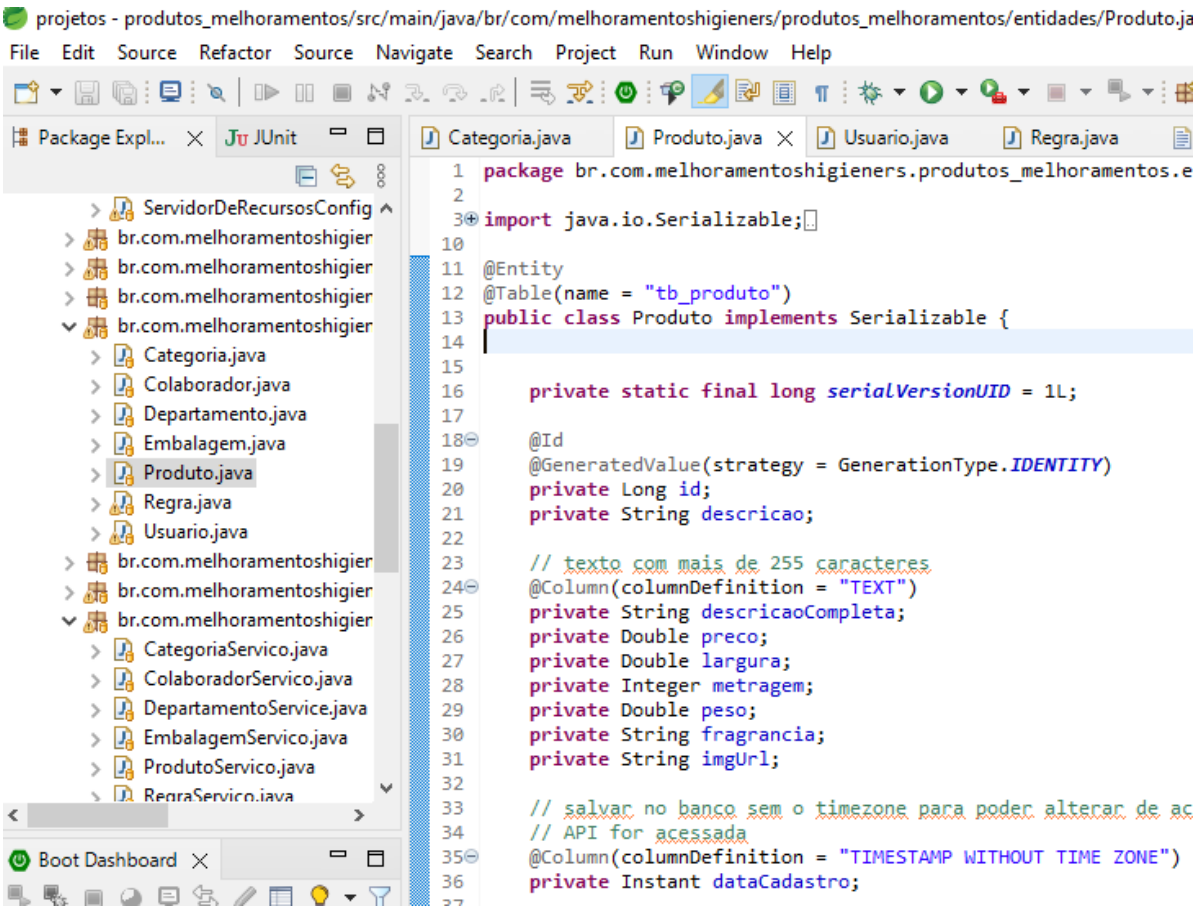
A JPA[29] (Java Persistence API) é a especificação padrão da plataforma java para realizar mapeamento objeto-relacional e a persistência de dados. A JPA do spring facilita muito problema de ter que ficar convertendo em diversos momentos as classes Java (que são orientadas a objetos) para o banco de dados relacional (tabelas), ou seja, fazer a comunicação entre objetos persistentes (classes java instanciadas) e o banco de dados. Ela se utiliza da técnica ORM[30] (*Object Relactional Mapping*), que consiste em fazer o mapeamento das entidades com as tabelas do banco de dados, tornando assim possível a comunicação entre o banco de dados e a linguagem de programação. A Figura 8 mostra o mapeamento da classe Produto com a tabela produtos denominada tb_produtos utilizando a *annotation @Table*.

[28]Disponível em: <https://docs.spring.io/spring-boot/docs/current/maven-plugin/reference/htmlsingle/>. Acesso em: 15 de maio de 2023.

[29]Disponível em: < https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ >. Acesso em: 25 de maio de 2023.

[30]Disponível em: < https://hibernate.org/orm/>. Acesso em: 10 de Abril de 2023.

Figura 8: Exemplo de Mapeamento objeto relacional utilizando JPA




Fonte: Autoria Própria

## 4.3. ENDPOINTS

Entende-se por *endpoint*[31] os terminais de um serviço web que podem ser referenciados e aos quais as mensagens de um servidor web podem ser endereçadas. Também se pode defini-los como a interface de uma aplicação na qual uma aplicação solicita um serviço a outra aplicação. Na prática seriam os endereços URL[32] (*Uniform Resource Locator*) da aplicação. Alguns endpoints serão públicos e o usuário não precisa de nenhuma permissão específica para acessar o mesmo. Para outros endereços da aplicação é necessários estar logado. As seguintes operações são permitidas sem estar logado (não precisa permissão): fazer consulta a lista de produtos, produtos detalhado, embalagens, categorias e colaboradores. Operações de inserção, atualização e exclusão de entidades apenas usuário devidamente autorizados conseguem realizar. Usuário logado como gerente de loja

[31]Disponível em: <https://www.cloudflare.com/pt-br/learning/security/api/what-is-api-endpoint/>. Acesso em: 15 de maio de 2023.

[32] Refere-se ao endereço de rede na qual se encontra algum recurso da aplicação web

consegue manipular, produtos, embalagens e categorias. Para poder listar, atualizar e inserir usuários no sistema o usuário precisa estar logado como administrador do sistema. O único endpoint que permite o usuário a realizar uma operação POST sem estar logado é justamente o endpoint de login. Para poder acessar os endpoint de usuários e regras, o usuário precisa estar logado como administrador de sistema, mesmo que seja apenas para operação de consulta. Os endereços foram hospedados na variável host[33].

A Tabela 1 abaixo mostra todos os endpoints da aplicação, quais métodos [34] HTTP estão sendo utilizados na requisição e qual a permissão necessária para realizar a operação. As requisições GET buscam listagens de objetos ou objetos do banco de dados, requisições POST foram usadas para inserir novos objetos como produtos, categorias, embalagens, colaboradores e usuários. Requisições PUT para atualizar dados e requisições e requisições DELETE para excluir objetos do banco de dados.

Tabela 1: Endpoints do Sistema

| **Endereço URL (endpoints)** | **Método HTPP** | **Permissão necessária** |
|---|---|---|
| host/categorias | GET | nenhuma |
| host/categorias/{id} | GET | nenhuma |
| host/categorias | POST | GERENTE_LOJA |
| host/categorias/{id} | PUT | GERENTE_LOJA |
| host/categorias/{id} | DELETE | GERENTE_LOJA |
| host/colaboradores | GET | nenhuma |
| host/colaboradores/{id} | GET | nenhuma |
| host/colaboradores | POST | ADMIN_SISTEMA |
| host/colaboradores/{id} | PUT | ADMIN_SISTEMA |
| host/colaboradores/{id} | DELETE | ADMIN_SISTEMA |
| host/departamentos | GET | nenhuma |
| host/departamentos/{id} | GET | nenhuma |
| host/departamentos | POST | ADMIN_SISTEMA |
| host/departamentos/{id} | PUT | ADMIN_SISTEMA |
| host/departamentos/{id} | DELETE | ADMIN_SISTEMA |
| host/embalagens | GET | nenhuma |
| host/embalagens/{id} | GET | nenhuma |
| host/embalagens | POST | GERENTE_LOJA |
| host/embalagens/{id} | PUT | GERENTE_LOJA |
| host/embalagens/{id} | DELETE | GERENTE_LOJA |

[33] Refere-se ao endereço de rede para acessar a aplicação. Com o seguinte formato http://socket. Um socket tem o formato url:porta

[34]Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods />. Acesso em: 20 de setembro de 2023.

| | | |
|---|---|---|
| host/produtos | GET | nenhuma |
| host/produtos/{id} | GET | nenhuma |
| host/produtos | POST | GERENTE_LOJA |
| host/produtos/{id} | PUT | GERENTE_LOJA |
| host/protudos/{id} | DELETE | GERENTE_LOJA |
| host/regras | GET | ADMIN_SISTEMA |
| host/oauth/token | POST | nenhuma |
| host/usuarios | GET | ADMIN_SISTEMA |
| host/usuarios/{id} | GET | ADMIN_SISTEMA |
| host/usuarios | POST | ADMIN_SISTEMA |
| host/usuarios/{id} | PUT | ADMIN_SISTEMA |
| host/usuarios/{id} | DELETE | ADMIN_SISTEMA |

Fonte: Autoria Própria

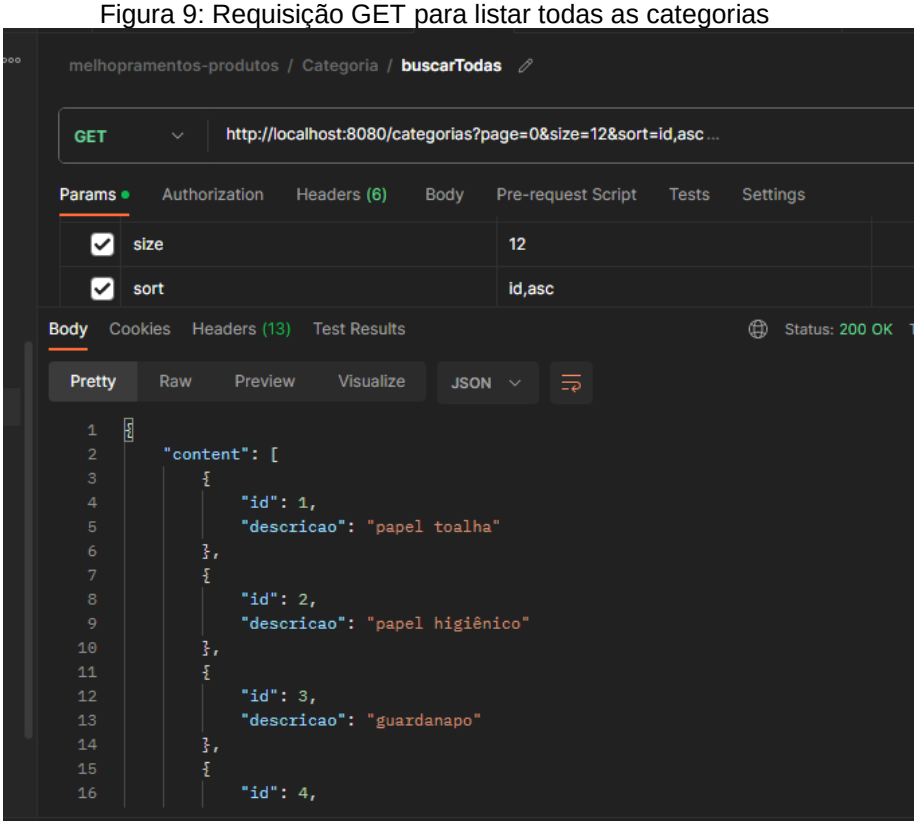
Para realizar essas requisições ao backend foi utilizado o software postman[35], que é um testador de requisições de código aberto. É possível observar a Figura 9 e Figura 10 duas requisições que obtiveram sucesso e retorna na Figura 9 uma lista de categorias; e na Figura 10 um produto novo criado. A Figura 9 retorna o código HTTP 200[36] de sucesso e a Figura 10 o código HTTP 201[37]*created*, de objeto criado com sucesso. Na Figura 9 é uma requisição GET e na Figura 10 é uma requisição POST de inserção de um novo produto no banco de dados, que retorna uma entidade produtoDTO para o cliente. Na próxima seção, veremos algumas requisições que possuem algum tipo de erro e como foram tratados esses erros.

[35]Disponível em: < https://www.postman.com />. Acesso em: 10 de outubro de 2023.

[36]Disponível em: <https://www.rfc-editor.org/rfc/rfc9110.html#name-200-ok>. Acesso em: 10 de setembro de 2023.

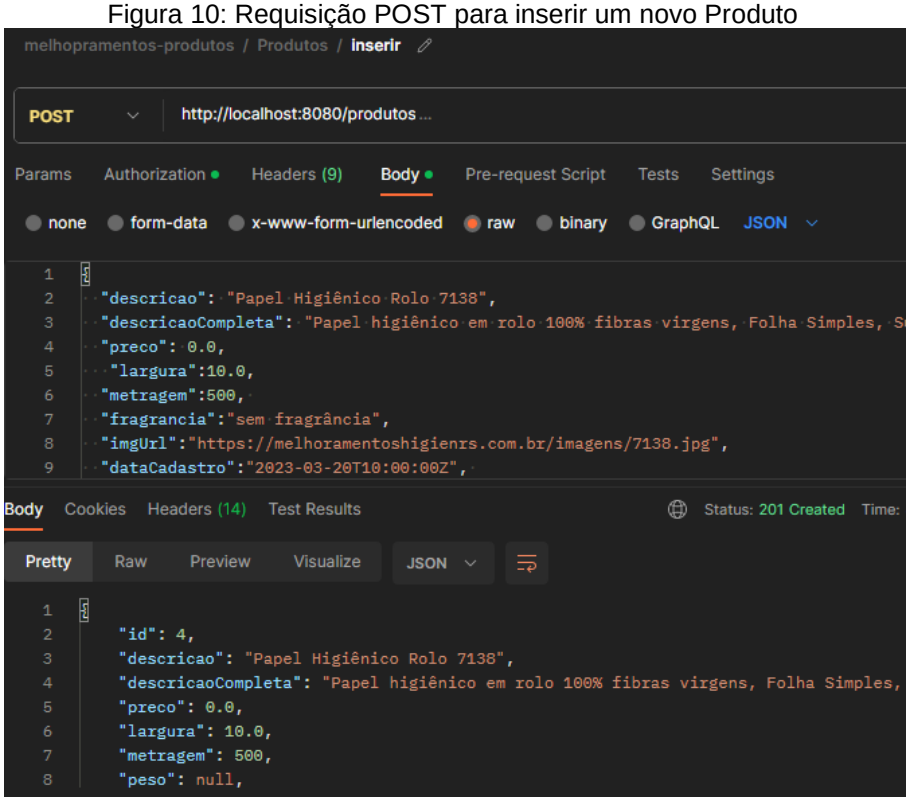
[37]Disponível em: <https://www.rfc-editor.org/rfc/rfc9110.html#name-201-created>. Acesso em: 10 de setembro de 2023.

Figura 9: Requisição GET para listar todas as categorias




Fonte: Autoria Própria

Figura 10: Requisição POST para inserir um novo Produto

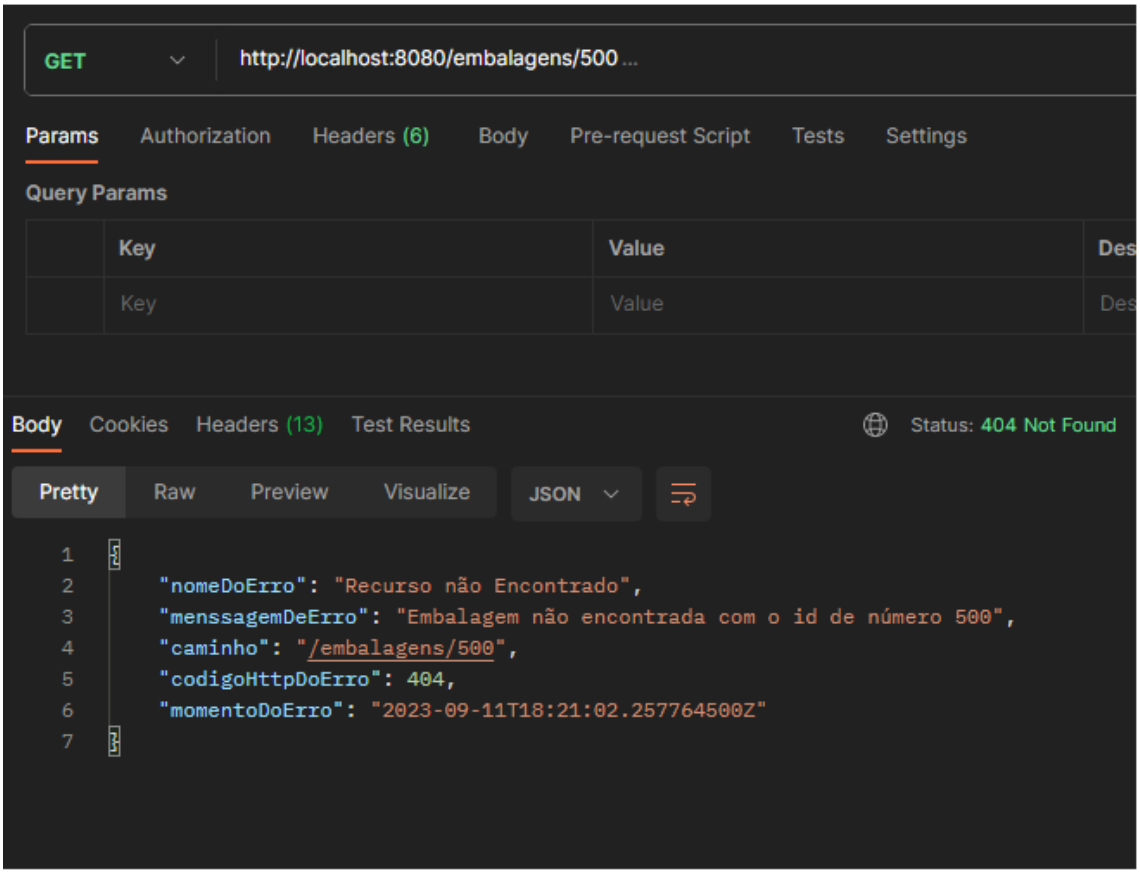



Fonte: Autoria Própria

## 4.4. TRATAMENTO DE EXCEÇÕES

Considerando uma exceção[38] como um erro inesperado do sistema o qual ocasiona uma interrupção imediata do sistema. Os principais erros para os quais foi desenvolvido tratamento de exceções personalizadas foram os erros de entidade não encontrada e erro de integridade de banco de dados. Na Figura 11 é possível observar o erro de produto não encontrado com o id de número 550, código HTTP 404[39] *Notfound*.

Figura 11: Exceção de requisição GET de buscar um produto com id inexistente




Fonte: Autoria Própria

Já na Figura 12 o erro que ocorre é erro de integridade de banco de dados, código HTTP 400[40] *BadRequest*, pois não é possível excluir uma embalagem que já tem um produto cadastrado no banco de dados com esse código de embalagem. A mesma lógica também é valida para tentar excluir uma permissão que já tem um

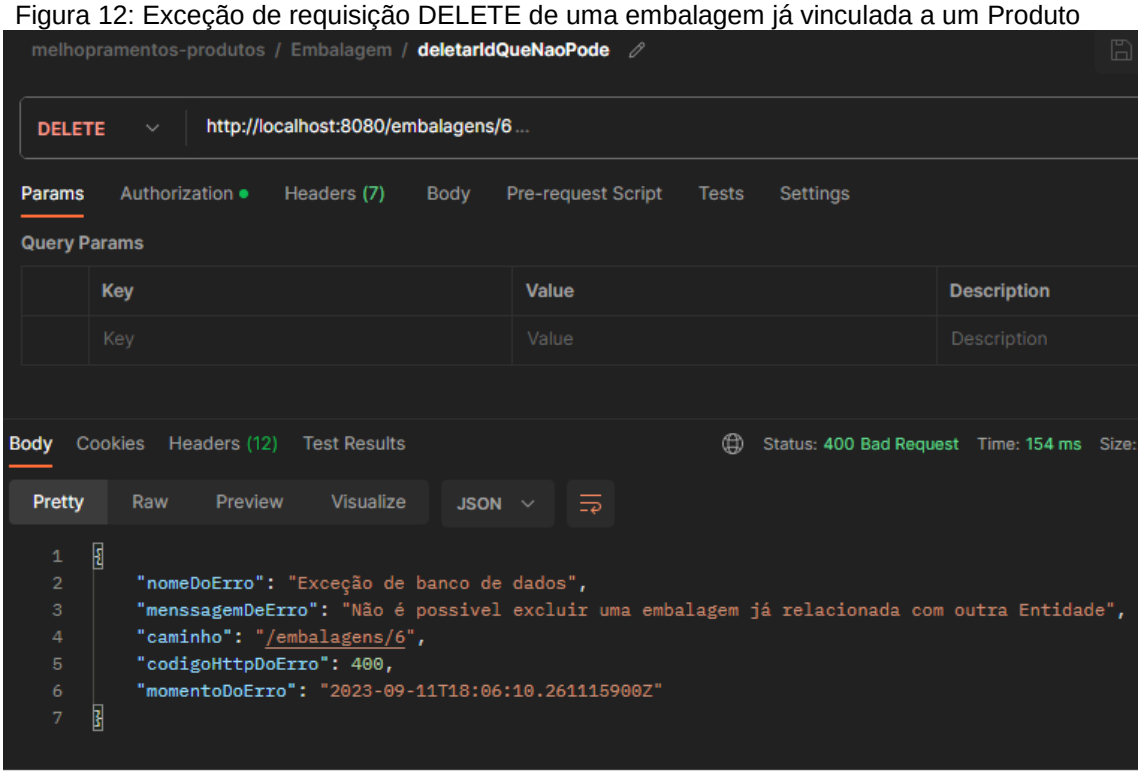
[38]Disponível em: <https://www.w3schools.com/java/java_try_catch.asp />. Acesso em: 05 de setembro de 2023.

[39]Disponível em: <https://www.rfc-editor.org/rfc/rfc9110.html#name-404-not-found >. Acesso em: 10 de setembro de 2023.

[40]Disponível em: <https://www.rfc-editor.org/rfc/rfc9110.html#name-400-bad-request >. Acesso em: 10 de setembro de 2023.

usuário vinculado a ela, ou um departamento que já tem um colaborador vinculado a esse departamento.

Figura 12: Exceção de requisição DELETE de uma embalagem já vinculada a um Produto




Fonte: Autoria Própria

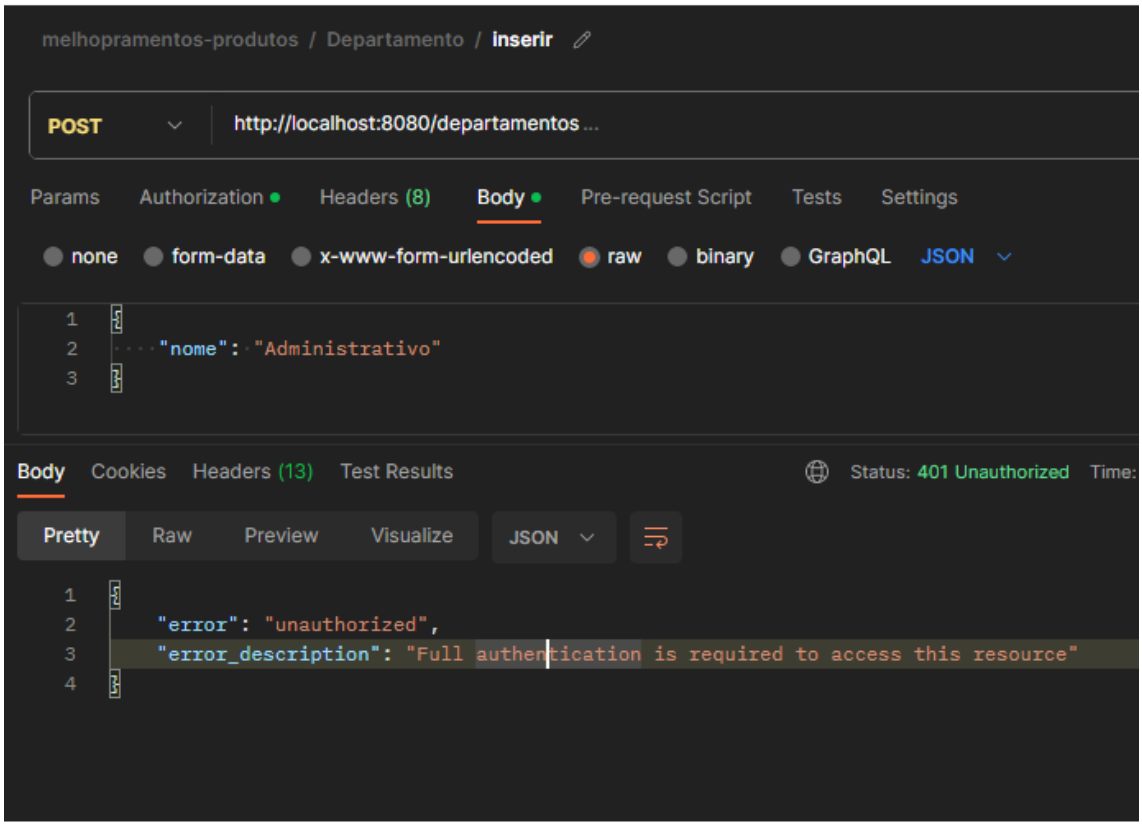
## 4.5. SEGURANÇA

### 4.5.1. Exceções de Usuários

Dois tipos de exceções que se referem à segurança da aplicação também receberam tratamento personalizado. São as exceções do usuário não logado tentar fazer algum tipo de operação que não seja consulta de banco de dados. Também foi tratada a exceção do usuário logado, mas sem a permissão necessária para acessar determinado endpoint ou realizar operações de escrita. A exceção de usuário tentar realizar uma requisição a qual necessita permissão especial e o usuário não está logado, responde com o erro de código HTTP 401[41] *Unauthorized*, conforme podemos verificar pela Figura 13.

[41]Disponível em: < https://www.rfc-editor.org/rfc/rfc9110.html#name-401-unauthorized >. Acesso em: 10 de setembro de 2023.

Figura 13: Exceção de requisição POST para inserir um departamento novo sem estar logado com a permissão necessária

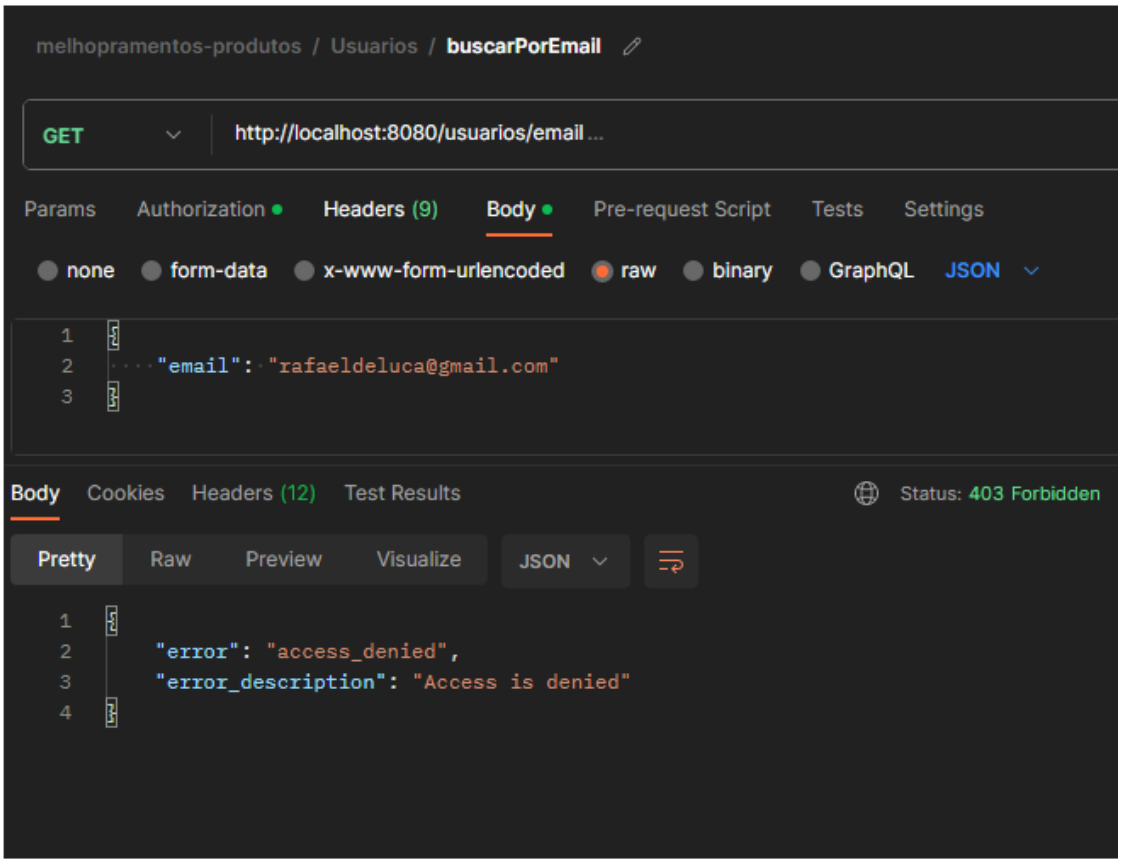



Fonte: Autoria Própria

Já o usuário logado como a permissão de gerente de loja, se tentar realizar alguma operação na entidade de usuário, mesmo que apenas consulta, vai receber o erro 403[42] *Forbidden*, conforme é possível observar pela Figura 14. Apenas usuários com a regra admin de sistema podem ter acesso à entidade Usuario.

[42]Disponível em: < https://www.rfc-editor.org/rfc/rfc9110.html#name-403-forbidden>. Acesso em: 10 de setembro de 2023.

Figura 14: Exceção de requisição GET para consultar usuário mail sem estar logado como admin de sistema



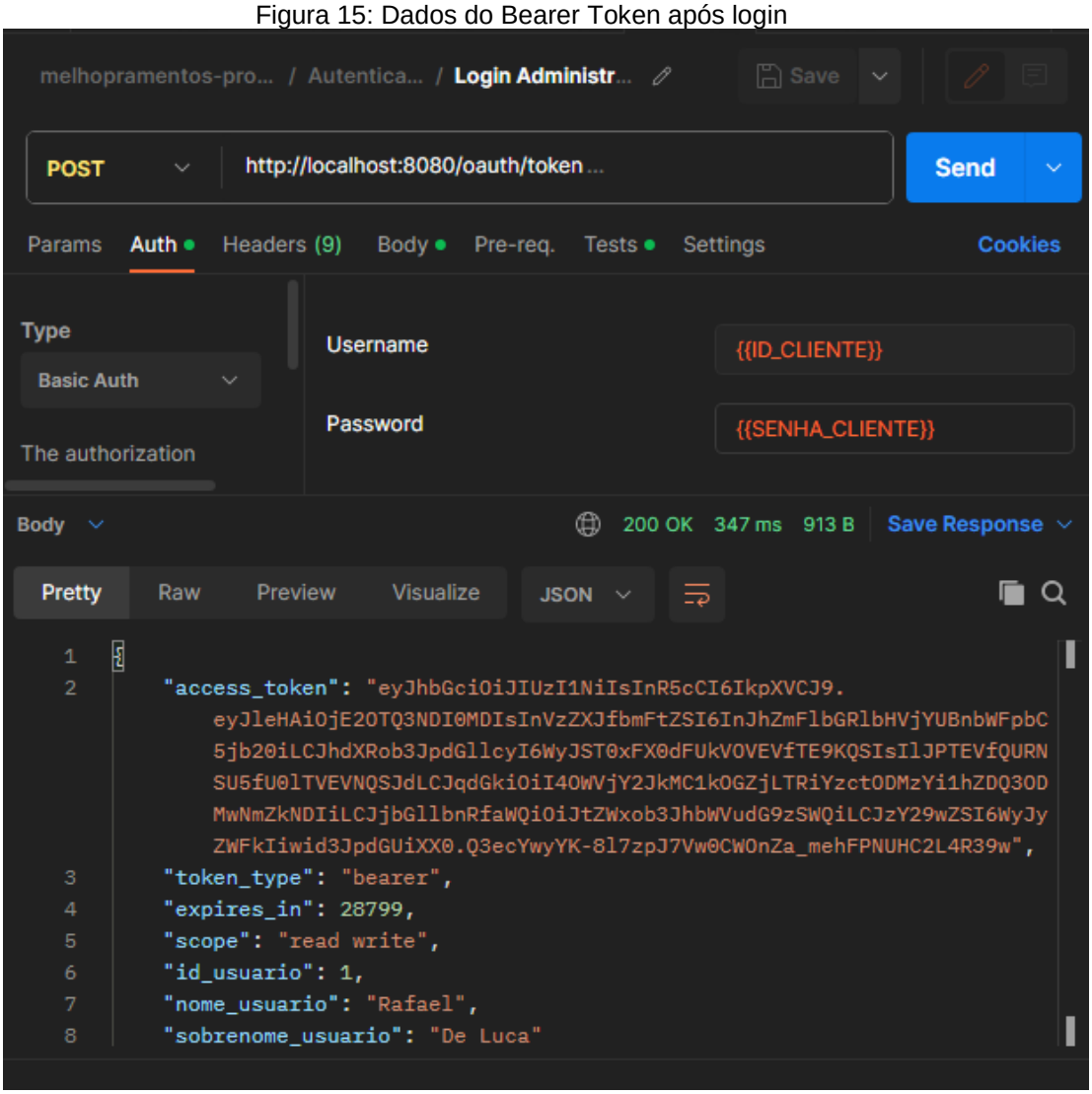

Fonte: Autoria Própria

### 4.5.2. Bearer Token

Conforme já foi exposto no capítulo de metodologia o token utilizado do sistema é o token jwt, mais específico o Bearer Token[43]. Esse bearer token, pode ser traduzido como token ao portador, ou seja, o usuário que portar esse token tem acesso aos endpoints e requisições que necessitem de autenticação. O bearer token é uma string enigmática grande conforme é possível verificar pela Figura 15, geralmente essa string é gerada por um servidor de recursos em resposta a uma reposta de solicitação de login. O bearer token também tem um tempo de expiração (em segundos), após esse tempo o usuário deve realizar nova autenticação para ter acesso a recursos protegidos. O token também possui parâmetros informando se o usuário tem acesso somente leitura dos recursos ou acesso completo.

[43]Disponível em: < https://oauth.net/2/bearer-tokens/>. Acesso em: 20 de maio de 2023.

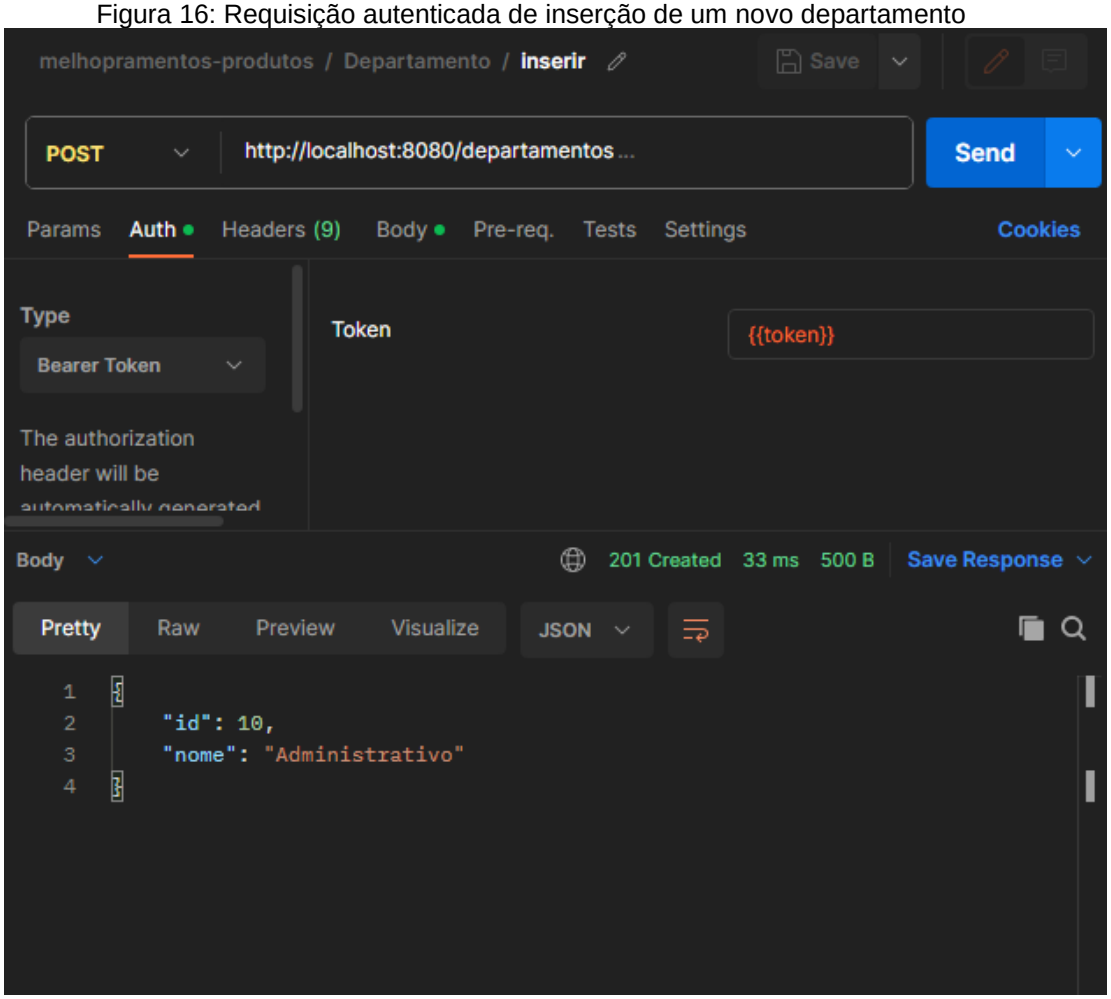
Figura 15: Dados do Bearer Token após login




Fonte: Autoria Própria

O cliente agora logado no sistema vai sempre enviar esse bearer token no cabeçalho dessas requisições para acessar recursos protegidos. É possível verificar agora pela Figura 16 que o cliente devidamente autenticado agora consegue realizar uma requisição com o verto POST de inserir um novo departamento; requisição que foi negada conforme observamos na Figura 13, visto que o usuário não estava logado, e por consequência não enviou o bearer token no cabeçalho da sua requisição.

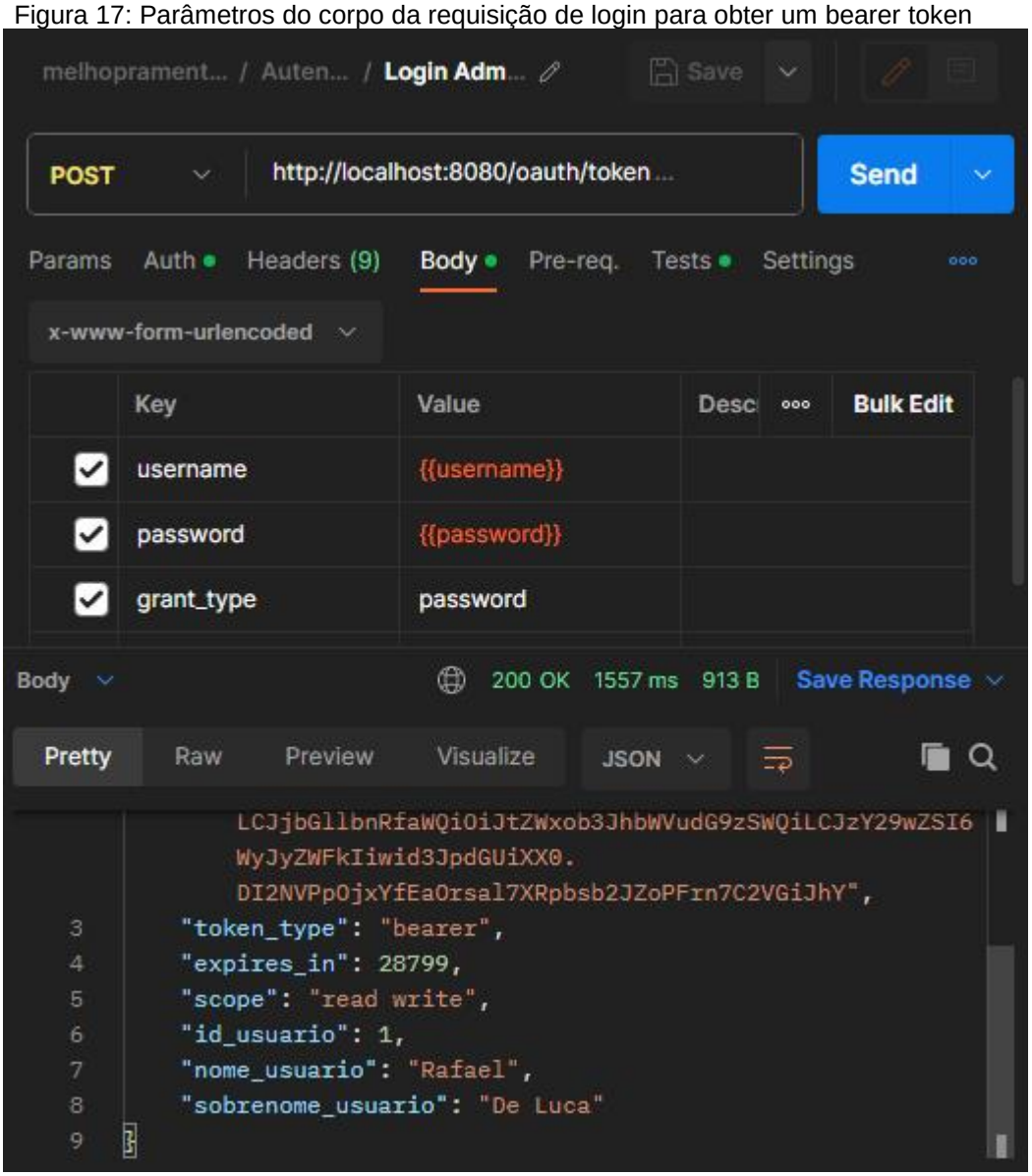
Figura 16: Requisição autenticada de inserção de um novo departamento




Fonte: Autoria Própria

Esse token jwt pode ser considerado relativamente seguro, pois para gerar o mesmo o usuário precisa informar cinco parâmetros: o usuário da aplicação e senha da aplicação, que são representados pelas variáveis id_cliente e senha_cliente, enviados por tipo de autenticação básica conforme é possível observar pela Figura 15 acima. E pelos três parâmetros enviados no corpo da requisição (usuário, senha e grant_type) conforme podemos observar pela Figura 17. Os valores desses parâmetros foram definidos de acordo com variáveis de ambiente do sistema, para que dados sensíveis como usuário e senha não ficarem expostas no código da aplicação.

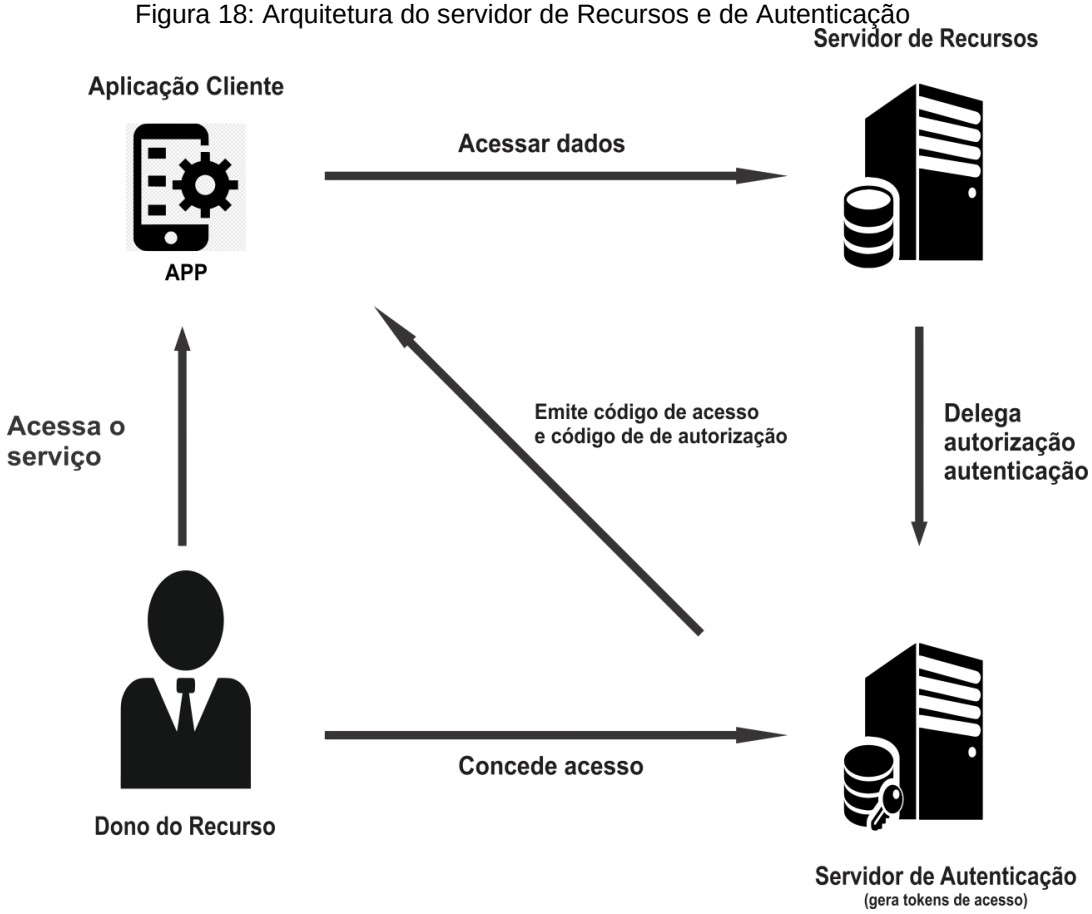
Figura 17: Parâmetros do corpo da requisição de login para obter um bearer token




Fonte: Autoria Própria

Com relação à configuração de arquitetura de obtenção de token o sistema possui 2 servidores: um servidor que fornece os recursos e um servidor de autenticação; ambos esse servidores são representados por 2 classes javas de configuração. A Figura 18 representa esse padrão de arquitetura de divisão de tarefas por de acordo com a função do servidor.

Figura 18: Arquitetura do servidor de Recursos e de Autenticação




Fonte: Autoria Própria

O servidor de autenticação gera o token válido, verifica se o usuário possui um token válido e concede acesso ao mesmo aos recursos protegidos do sistema. Já o servidor de recursos possui as configurações de quais são os endpoint públicos e protegidos e, com relação aos endpoint protegidos, qual a permissão necessária que o usuário tem que possuir para acessar determinado recurso e com qual verbo HTTP. A Figura 19 exemplifica algumas dessas configurações do servidor de recursos.

Figura 19: Configurações de recursos protegidos no servidor de Recursos

```
@Override
public void configure(HttpSecurity segurancaHttp) throws Exception {

    // liberando para acessar a interface do banco de dados em memória h2
    if (Arrays.asList(ambiente.getActiveProfiles()).contains("test")==true) {
        segurancaHttp.headers().frameOptions().disable();
    }

    segurancaHttp.authorizeRequests()
            .antMatchers(ROTA_PUBLICA).permitAll() // publica a rota para fazer autenticacação
            .antMatchers(ROTA_BANCO_H2).permitAll()
            .antMatchers(HttpMethod.GET,ROTA_CONSULTAR_CATALOGO).permitAll()
            .antMatchers(HttpMethod.POST,ROTA_CRUD_ENTIDADES).hasRole(String.valueOf(Permissao.GERENTE_LOJA))
            .antMatchers(HttpMethod.PUT,ROTA_CRUD_ENTIDADES).hasRole(String.valueOf(Permissao.GERENTE_LOJA))
            .antMatchers(HttpMethod.DELETE,ROTA_CRUD_ENTIDADES).hasRole(String.valueOf(Permissao.GERENTE_LOJA))
            .antMatchers(HttpMethod.OPTIONS,ROTA_CRUD_ENTIDADES).hasRole(String.valueOf(Permissao.GERENTE_LOJA))
            .antMatchers(HttpMethod.HEAD,ROTA_CRUD_ENTIDADES).hasRole(String.valueOf(Permissao.GERENTE_LOJA))
            .antMatchers(HttpMethod.POST,ROTA_CRUD_COLABORADORES).hasRole(String.valueOf(Permissao.ADMIN_SISTEMA))
            .antMatchers(HttpMethod.PUT,ROTA_CRUD_COLABORADORES).hasRole(String.valueOf(Permissao.ADMIN_SISTEMA))
            .antMatchers(HttpMethod.DELETE,ROTA_CRUD_COLABORADORES).hasRole(String.valueOf(Permissao.ADMIN_SISTEMA))
            .antMatchers(ROTA_ADMINTRADORES).hasRole(String.valueOf(Permissao.ADMIN_SISTEMA));
}

// configuração de CORS
// Cross-origin resource sharing
// por padão os framework bloqueiam que o frontend hospeda em diferente host do backend faça requisições ao backend
// liberar aqui explicadamente quais endereços podem fazer requisições ao backend
@Bean
public CorsConfigurationSource configuracaoDaFonteDeCors() {

    CorsConfiguration configuracaoDeCors = new CorsConfiguration();

    String [] hostLiberados = origensCors.split(",");
    configuracaoDeCors.setAllowedOriginPatterns(Arrays.asList(HOST_LIBERADOS));
```

Fonte: Autoria Própria

### 4.5.3. Configuração de CORS

O servidor de recursos também possui as configurações de CORS[44]. Este é um mecanismo de segurança, que por padrão do Spring, bloqueia qualquer requisição do navegador que esteja em um domínio ou porta diferente de onde esta sendo executado o backend. Nas configurações de CORS são informados os endereços de frontend que o navegador deve permitir o carregamento dos recursos de da aplicação. Essa configuração é muito importante quando se pretende hospedar o frontend em local diferente do backend e também para não permitir que qualquer domínio fique fazendo requisições ao seu backend.

Uma configuração de segurança importante também utilizada na aplicação não refere-se ao acesso ao recursos na camada web (controlador) e sim na camada de serviço (lógica de negócios da aplicação). Nessa camada todas as requisições que envolviam alterações de dados foram utilizadas a *annotation @Transacional*[45]. Essa anotação faz com que o Spring Boot monitore essa requisição e ela seja realizada de acordo o princípio da atomicidade que diz que o processo é executado por completo ou ele volta a seu estado original.

[44]Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/>. Acesso em: 25 de setembro de 2023.

[45]Disponível em: <https://www.baeldung.com/spring-transactional-propagation-isolation/>. Acesso em: 20 de setembro de 2023.

# 5 ARQUITETURA DO FRONTEND

## 5.1. LINGUAGENS

Nesse capítulo serão abordadas as linguagens utilizadas e as principais bibliotecas utilizadas no desenvolvimento da parte visual e interativa da aplicação. Conforme já visto no capítulo da metodologia foi utilizada a linguagem de marcação HTML[46] (*Hyper Text Markup Language*) e de linguagem de estilização CSS[47](*Cascading Style Sheets*) para criação do layout estático do site.

A parte de lógica de programação para desenvolvimento do frontend foi utilizada a linguagem TypeScript[48] junto com o a biblioteca React.[49] A linguagem TypeScript é considerada um superset (super conjunto) do linguagem JavaScript[50], ou seja, é uma linguagem que contém todas as funcionalidades e recursos do javaScript e mais alguns recursos. Ela diferentemente do JavaScript, é um linguagem do tipo tipada, o que torna um pouco mais complexo de programar; entretanto, evita erros inesperados na hora de execução do código e avisa erros de compilação durante o processo de desenvolvimento. Visto que os navegadores apenas interpretam JavaScript puro, o TypeScript compila (transpila) para JavaScript em tempo de execução. Foi usado editor de código-fonte Visual Studio Code[51] para desenvolvimento do código do frontend.

## 5.2. BIBLIOTECAS

Para instalação, atualização e remoção de bibliotecas do projeto foi utilizado o gerenciador de dependências e pacotes yarn[52.] O gerenciamento dessas bibliotecas pode ser feito através de linhas de comando no console ou através do arquivo package.json localizado na raiz do projeto frontend.

[46]Disponível em: <https://www.w3schools.com/html />. Acesso em: 15 de setembro de 2023.

[47]Disponível em: <https://www.w3schools.com/css />. Acesso em: 15 de setembro de 2023.

[48]Disponível em: <https://www.typescriptlang.org/html />. Acesso em: 20 de setembro de 2023.

[49]Disponível em: <https://react.dev/>. Acesso em: 10 de agosto de 2023.

[50]Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 20 de setembro de 2023.

[51] Disponível em: < https://code.visualstudio.com//>. Acesso em: 20 de setembro de 2023.

[52]Disponível em: < https://yarnpkg.com//>. Acesso em: 20 de setembro de 2023.

A principal biblioteca utilizada no frontend foi o React. O React é uma biblioteca de código aberto da linguagem JavaScript muito utilizada para construir interfaces gráficas para o usuário em aplicações web. Durante o desenvolvimento com React são muito utilizados dois conceitos: o de renderização[53] e de arquitetura baseada em componentes. A definição de renderização aplicado ao React seria converter código JavaScript em elementos visuais no navegador. Com relação à arquitetura baseada em componentes, esta consiste em desenvolver componentes mais simples e utilizar os mesmos para fazer componentes mais complexo, salientando que componentes no React são funções do TypeScript. Por exemplo, criar um componente de cabeçalho, um de Card e um de rodapé e depois usar todos esses seus componentes simples de criar um mais complexo chamado de página inicial. Cabe salientar que esses componentes podem ser reutilizados várias vezes em diferentes locais da aplicação. Segue a Figura 20 que mostra uma parte do código de um componente complexo chamado de Rotas que se utilizada de vários componentes mais simples como Header, Home, Produtos, entre outros para a criação da sua interface visual. Entre as diversas vantagens de biblioteca React é possível citar a sua simplicidade e também sua compatibilidade com dispositivos móveis; a base de código da biblioteca React serve com base de código para aplicações Android e IOS.

[53]Disponível em: < https://pt-br.legacy.reactjs.org/docs/rendering-elements.html/>. Acesso em: 05 de agosto de 2023.

Figura 20: Função React (componente) denominado Rotas

```
import NotFound from "./componentes/NotFound";

function Rotas() {

  return (

    <>
      <Router history={history}>
        <Header></Header>
        <Switch>
          <Route path="/" exact>
            <Home></Home>
          </Route>
          <Route path="/produtos" exact>
            <Produtos></Produtos>
          </Route>
          <Route path="/produtos/:produtoId">
            <ProdutoDetalhado></ProdutoDetalhado>
          </Route>
          <Route path="/contatos">
            <Contatos></Contatos>
          </Route>
```

Fonte: Autoria Própria

Entre os principais recursos utilizados do react podemos destacar as Props, ReactRouter, ReactHooks, useState, useEffect, BootStrap,Loaders e axios. Visto que componentes React nada mais são que funções que retornam itens HTML, as Props seriam os argumentos (parâmetros) dessas funções. As Props[54] são passadas como argumentos de funções TypeScript de maneira muito similar a atributos de componentes HTML. Os componentes devem receber as Props como um objeto TypeScript.

Outro recurso utilizado foi o ReactRouter[55] que é considerado uma biblioteca estrutura do projeto React. É uma biblioteca javaScript pura e por isso é necessário instalar os types da biblioteca para ficar compatível com o supertipo TypeScript. Visto que a biblioteca React não possui funções para manipulação de rotas o ReactRouter acaba sendo muito utilizada pela comunidade de desenvolvedores web que trabalham com React.

[54]Disponível em: <https://pt-br.legacy.reactjs.org/docs/components-and-props.html />. Acesso em: 05 de agosto de 2023.

[55]Disponível em: < https://www.w3schools.com/react/react_router.asp html/>. Acesso em: 10 de agosto de 2023.

Considerando componentes Reacts como componentes visuais que possuem um cliclo de vida com vários eventos específicos. Os ReactHooks[56] são funções especiais às quais se comportam de acordo com o ciclo de vida de um componente React. Os mesmos permitem usar estados e outros recursos em um componente. Os ReactHooks são utilizados para montar componentes visuais de acordo com as respostas que vierem das requisições feitas ao backend. Os Hooks permitem a renderização dos componentes em diferentes momentos do ciclo de vida, por exemplo, no momento que o componente é montado pela primeira vez, no momento que algum estado (dado) é atualizado e no momento que o componente deixa de existir. Dois tipos de ReactHooks muito utilizados no sistema foram o useState[57] e o useEffect[58]. O useState é usado para armazenar estado de um componente. Ele tem a capacidade de monitorar as mudanças no estado de um componente e reagir de acordo com essas mudanças. Por exemplo, no momento que esse dado (estado) que está dentro do componente for alterado, o useState que está monitorando o componente, observa essa alteração e reflete essas mudanças na tela. Já o useEffect é um ReactHook que permite executar uma função (ação) no componentes em dois momentos. No momento da montagem pela primeira vez e no momento em um dado do componente foi alterado. Ou seja, assim como o useState, ele tem a capacidade de observar e reagir a mudanças de estado de um componente. Ele se diferencia do useState por lidar com efeitos, como: chamadas a uma API, modificações do DOM e tarefas assíncronas, enquanto o useState é mais usado para controlar variáveis de estado de um componente funcional. O useEffect necessita de dois parametros, o primeiro é o função a ser executada assim que o componente for montado e, o segundo, uma lista de dependências a serem monitoradas para executar a função novamente, sempre que algum dado for alterado.

A biblioteca Bootstrap[59] foi utilizada para criar o site 100% responsivo, tornando o site adaptado tanto para tela de celular, tablet e desktop. Bootstrap é um framework de código-fonte aberto que auxilia no desenvolvimento de aplicações responsivas. A Figura 21 exibe os breakpoints utilizados no desenvolvimento do site. Definindo breakpoint com sendo as larguras mínimas das telas que a disposição dos elementos são redimensionados para uma melhor experiência do usuário.

[56]Disponível em: < https://legacy.reactjs.org/docs/hooks-intro.html />. Acesso em: 15 de agosto de 2023.

[57]Disponível em: < https://legacy.reactjs.org/docs/hooks-state.html />. Acesso em: 15 de agosto de 2023.

[58]Disponível em: <https://legacy.reactjs.org/docs/hooks-effect.html />. Acesso em: 15 de agosto de 2023.

[59]Disponível em: < https://getbootstrap.com/>. Acesso em: 20 de agosto de 2023.

Figura 21: Larguras da tela dos breakpoint da biblioteca bootstrap

| Breakpoint | Class infix | Dimensions |
|---|---|---|
| X-Small | *None* | <576px |
| Small | sm | ≥576px |
| Medium | md | ≥768px |
| Large | lg | ≥992px |
| Extra large | xl | ≥1200px |
| Extra extra large | xxl | ≥1400px |

Fonte: https://getbootstrap.com/docs/5.0/layout/breakpoints

A biblioteca loaders[60] foi utilizada para tornar a experiência do usuário mais agradável enquanto ele aguarda a resposta de uma requisição ao backend. Ao invés do usuário ficar observando uma tela branca enquanto aguarda a resposta de uma requisição, ele visualiza caixas retangulares que simulam o carregamento de dados.

Para requisições ao backend foi utilizada a biblioteca React chamada de axios[61]. Axios é uma biblioteca de requisições do React baseada em Promises[62]. Promisses representam a conclusão ou falha de uma operação assíncrona e seu valor. Requisições são operações assíncronas. Programação assincrona[63] é uma técnica utilizada na qual o permite que o sistema inicie uma tarefa que pode ser de longa duração e ainda seja capaz de responder a outros eventos enquanto essa tarefa assíncrona é executada. Muito útil na redenrização de componentes React, visto que se o componente depender dos dados da requisição ao backend para ser montando, se ele for montado antes da resposta da requisição, pode ser montando com falha ou incompleto.

Além dos loaders, também foram utilizados toast[64] para uma melhor experiência do usuário. Toast são caixas de notificação elegantes que aparecem no canto superior direito da tela mostrando ao usuário se a sua requisição teve sucesso ou erro. Para utilização de toast foi utilizada a biblioteca React-toatity.[65]

## 5.3. SEGURANÇA

Conforme já descrito uma das características de aplicações React é a componentização. Onde a partir de componentes mais simples construímos componentes mais complexos. Esses componentes mais complexos são utilizados

[60]Disponível em: < https://skeletonreact.com/ >. Acesso em: 20 de agosto de 2023.

[61]Disponível em: < https://axios-http.com / >. Acesso em: 25 de agosto de 2023.

[62] Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/>. Acesso em: 20 de agosto de 2023.

[63]Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Asynchronous / >. Acesso em: 20 de agosto de 2023.

[64]Disponível em: <https://fkhadra.github.io/react-toastify/introduction/ >. Acesso em: 10 de outubro de 2023.

[65]Disponível em: < https://github.com/fkhadra/react-toastify / >. Acesso em: 20 de outubro de 2023.

de páginas e, por fim, essas páginas são exibidas (renderizadas) na tela para o usuário. A Tabela 2 mostra de acordo com cada rota qual componente é renderizado. Nas rotas públicas não é preciso estar logado no sistema para renderizar o componente, já as rotas privadas (restritas) são necessárias permissões de acordo com o perfil do usuário. Caso o usuário que não esteja logado e tentar acessar uma rota privada, ele é direcionado para a página de login. Se estiver logado e com a permissão necessária, o usuário acessa a rota privada. O frontend da aplicação consegue saber se o usuário está logado ou não verificando se o usuário tem ou não um token válido armazenado no LocalStorage do navegador. O funcionamento do LocalStorage será explicado com mais detalhes posteriormente.

Tabela 2: Rotas e Componentes

| **Rota** | **Página (componente)** | **Acesso** |
|---|---|---|
| / | Home | Público |
| /produtos | Produtos | Público |
| /produtos/:produtoId | ProdutoDetalhado | Público |
| /contatos | Contatos | Público |
| /empresa | Empresa | Público |
| /admin | Autenticação | Público |
| /admin/autenticar | Autenticacao | Público |
| /admin/autenticar/login | Autenticacao | Público |
| /admin/sms | Admin/CadastroSms | Privado |
| /admin/categorias | Admin/Categorias/ListagemCategorias | Privado |
| /admin/categorias/inserir | Admin/Categorias/CadastroCategoria | Privado |
| /admin/categorias/:categoriaId | Admin/Categorias/CadastroCategoria | Privado |
| /admin/embalagens | Admin/Embalagens/ListagemEmbalagens | Privado |
| /admin/embalagens/inserir | Admin/Embalagens/CadastroEmbalagem | Privado |
| /admin/embalagens/:embalagemId | Admin/Embalagens/CadastroEmbalagem | Privado |
| /admin/produtos | Admin/Produtos/ListagemProdutos | Privado |
| /admin/produtos/inserir | Admin/Produtos/CadastroProduto | Privado |
| /admin/produtos/:produtoId | Admin/Produtos/CadastroProduto | Privado |
| /admin/usuarios | Admin/Usuarios/ListagemUsarios | Privado |
| /admin/usuarios/inserir | Admin/Usuarios/CadastroUsuario | Privado |
| /admin/usuarios/:usuarioId | Admin/Usuarios/CadastroUsuario | Privado |

Fonte: Autoria Própria

# 6 RESULTADOS E DISCUSSÕES

## 6.1. APLICAÇÃO

A tela inicial da aplicação possui um menu superior, cabeçalho, logo principal, link para a API web do *WhatsApp*[66] e link para o catálogo de produtos conforme é possível observar pela Figura 22.

Figura 22: Página inicial da Aplicação


Fonte: Autoria Própria

Já na Figura 23 podemos observar a responsividade da aplicação para telas de celular, onde aparece o menu hambúrguer a direita ao invés de um menu do tipo lista no cabeçalho superior, e a mudança na disposição do logo e do texto. Todas as páginas da aplicação são responsivas para telas de celular, tablet e desktop de acordo os breakpoints do boostrap.

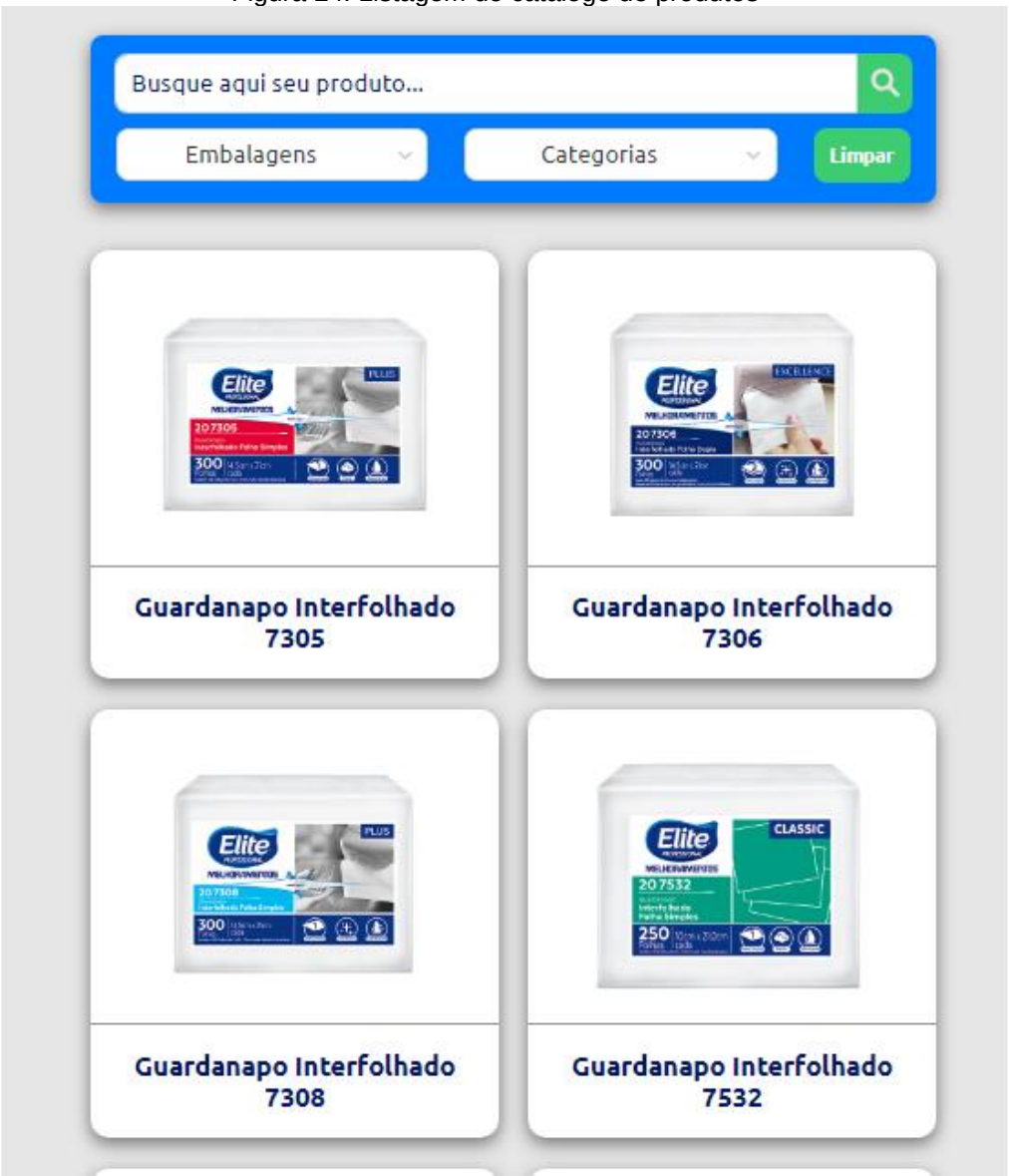
[66] Disponível em: < https://web.whatsapp.com/>. Acesso em: 20 de outubro de 2023.

Figura 23: Responsividade da página inicial para celular


Fonte: Autoria Própria

A rota aplicação que mostra o catálogo de produtos faz uma listagem dos produtos, de doze itens (card de produtos) por página na visualização para desktop. Ela possui os cards com a descrição e imagem dos Produtos. Possui também um componente na parte superior que permite o usuário filtrar produtos por descrição, embalagem ou categoria, conforme é possível observar pela Figura 24.

Figura 24: Listagem do catálogo de produtos



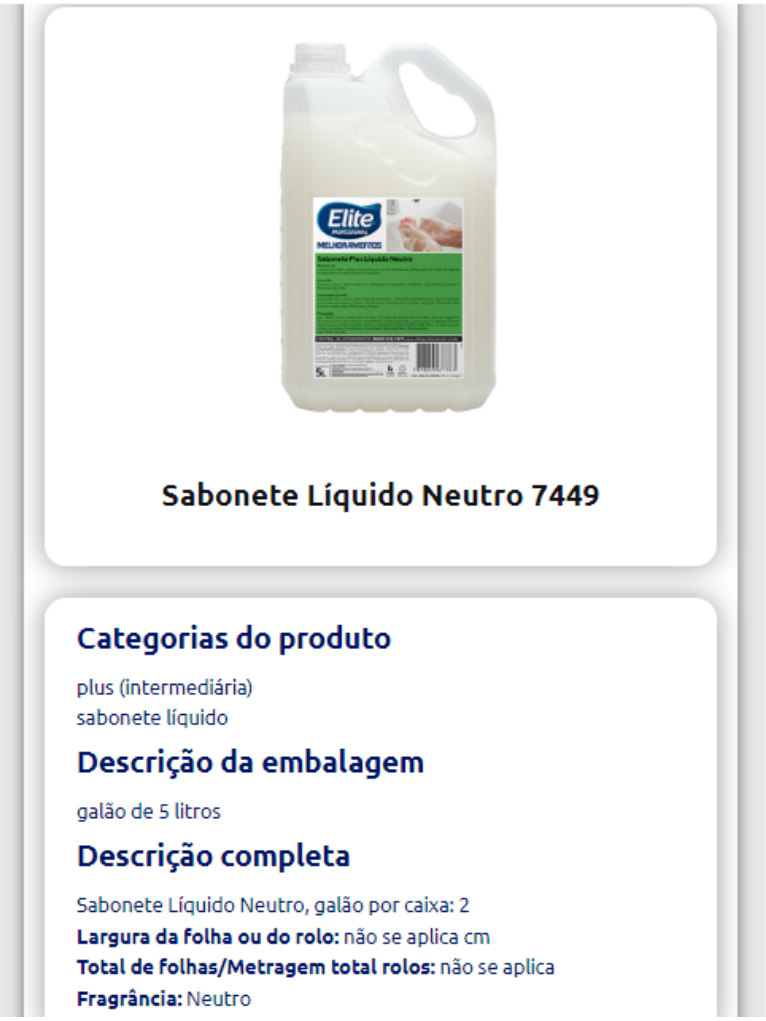

Fonte: Autoria Própria

Como a busca de produtos é paginada, ela possui um componente de paginação na parte inferior que permite o usuário navegar pela listagem de produtos, conforme podemos observar pela Figura 25. A aplicação vai renderizar doze, nove, seis, quatro ou um componente por tela de acordo com o tamanho da tela do dispositivo que está acessando a aplicação. Ao clicar no card de cada produto o usuário é direcionado para uma área de detalhes do produto, onde é possível acessar uma descrição completa do produto com suas categorias e embalagens disponíveis e demais dados do produto, como mostra a Figura 26.

Figura 25: Componente de paginação no rodapé página de listagem de produtos


Fonte: Autoria Própria

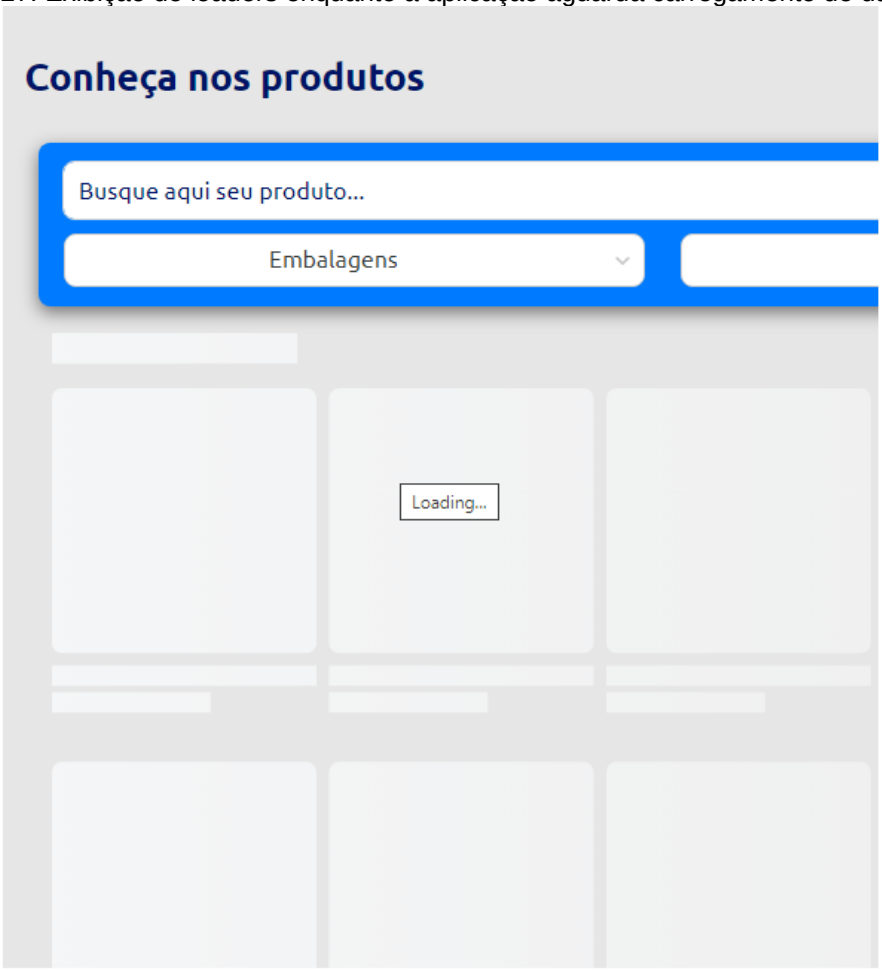
Figura 26: Página com a descrição detalhada do Produto




Fonte: Autoria Própria

Conforme mencionado do capítulo anterior, para melhorar a experiência do usuário e o mesmo não ficar vendo uma tela branca enquanto espera o carregamento dos dados do backend, foram utilizados loaders para algumas páginas da aplicação conforme podemos verificar pela Figura 27.

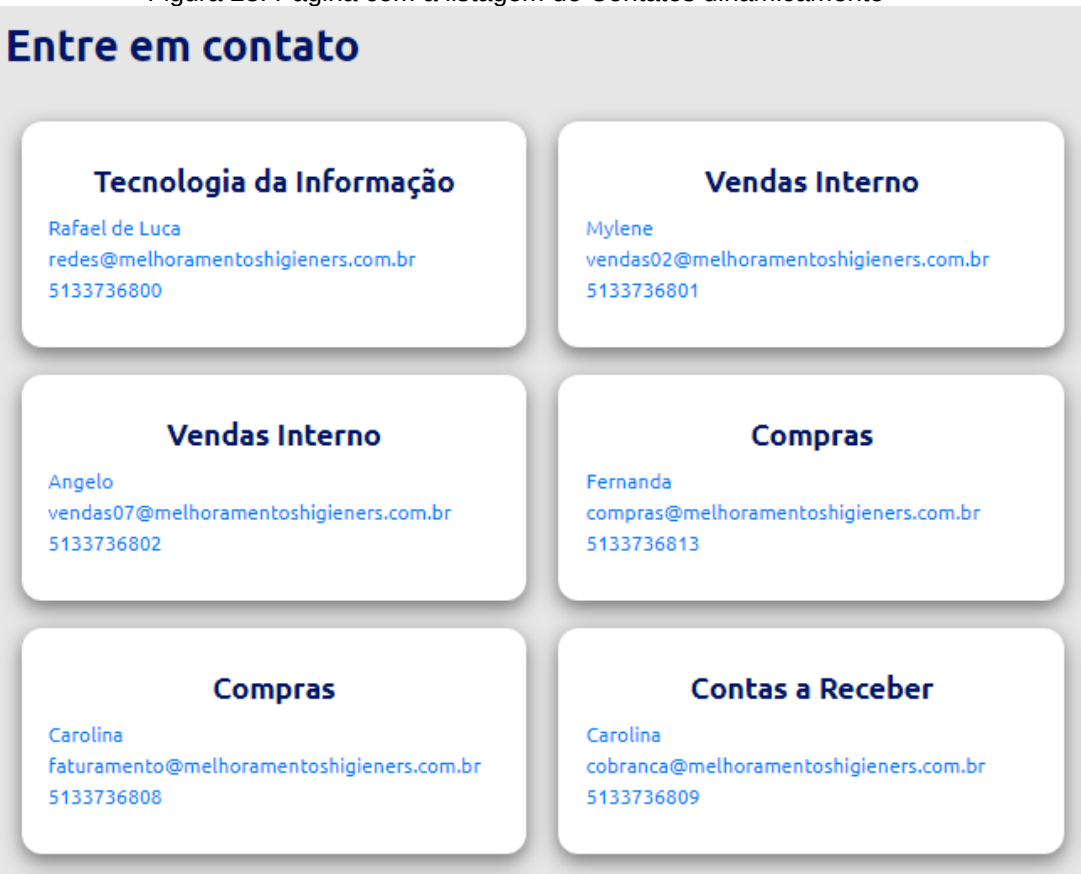
Figura 27: Exibição de loaders enquanto a aplicação aguarda carregamento de dados




Fonte: Autoria Própria

Na rota de contatos são renderizados dinamicamente os principais contatos dos colaboradores da empresa com nome, departamento, email e telefone, permitindo assim realizar um atendimento mais direcionado para o cliente. Segue Figura 28 com a tela de contatos.

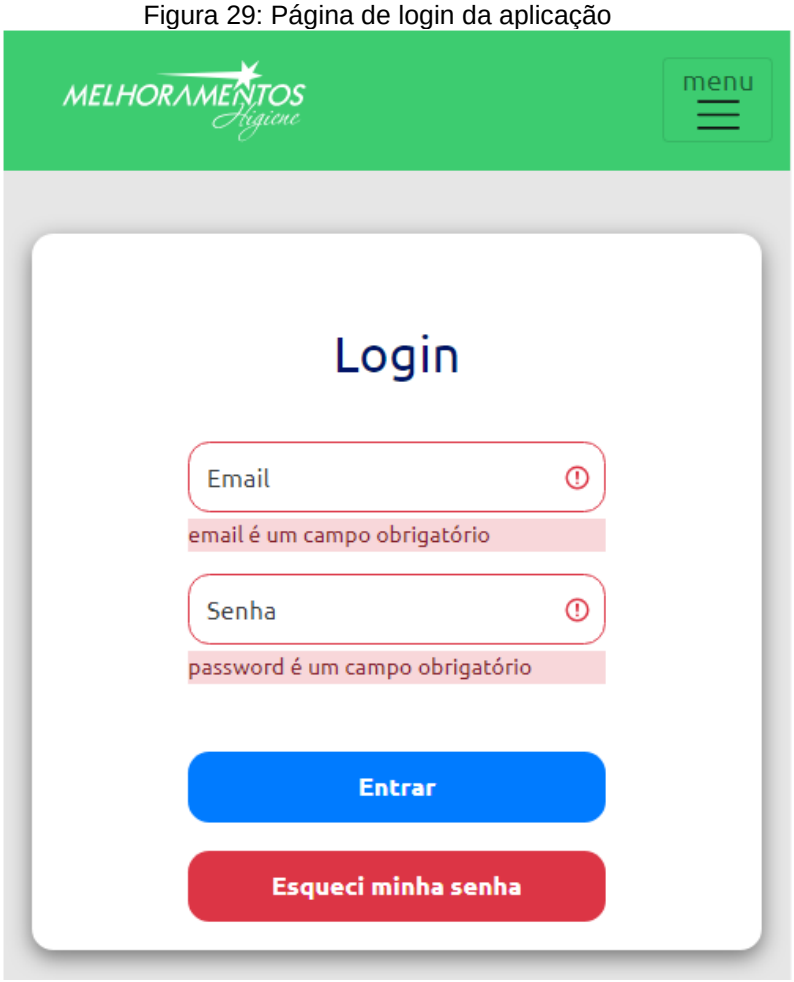
Figura 28: Página com a listagem de Contatos dinamicamente




Fonte: Autoria Própria

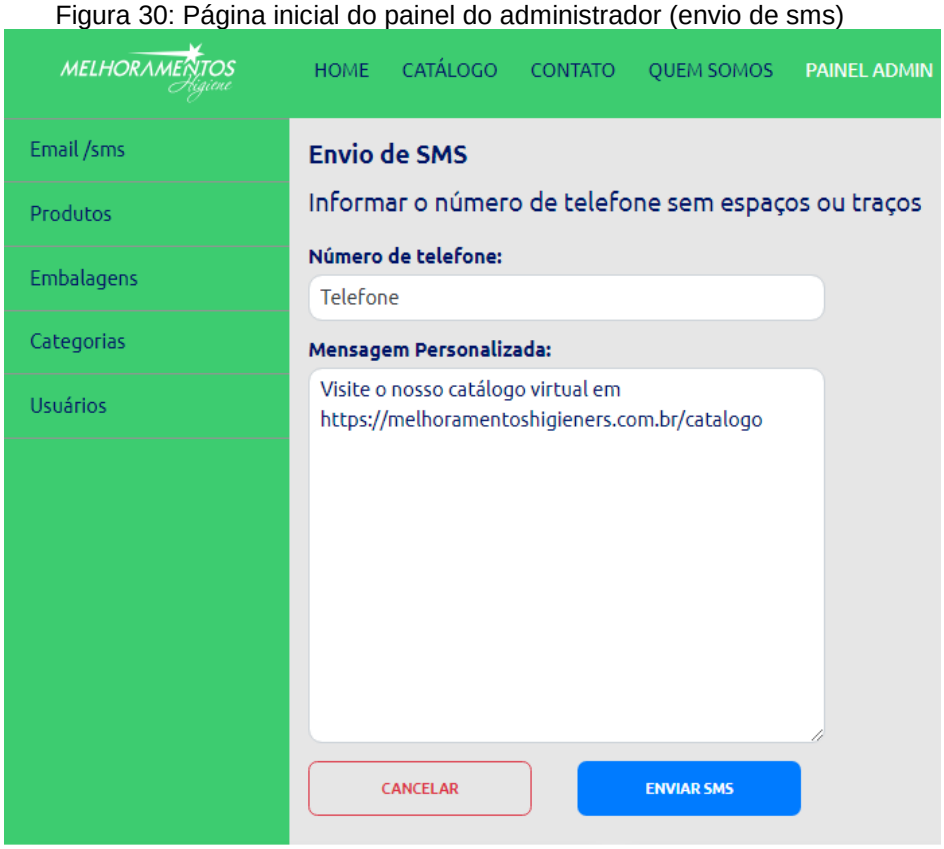
As rotas acima são a parte pública da aplicação que fica disponível para acesso sem necessidade de login. A área de envio de sms, gerenciamento (cadastro, edição e exclusão) de produtos, gerenciamento de categorias, gerenciamento de departamentos e gerenciamento de usuários são áreas restritas que o usuário precisa ter um token válido e a permissão adequada para acesso de cada rota específica. A figura 29 mostra a tela de login antes de login e a figura 30 após o login, onde o usuário é redirecionado para a tela de envio de sms. Na tela de login são feitas as validações se os campos de email e senha foram preenchidos corretamente, antes de enviar a requisição ao backend. São verificados se o campo de email é um email válido e se o campo senha tem no mínimo 6 caracteres. Após a tentativa de login o usuário recebe um toast no canto direito superior da tela, mostrando se o login foi realizado com sucesso ou houve erro na tentativa de login.

Figura 29: Página de login da aplicação




Fonte: Autoria Própria

Figura 30: Página inicial do painel do administrador (envio de sms)




Fonte: Autoria Própria

No canto superior direito (na tela de dimensões de desktop) também é renderizado dinamicamente o componente de login com o email do usuário logado e no lugar do componente (botão) de login é renderizado um novo componente com o texto “sair” para o usuário realizar logoff do sistema. Podemos observar pela Figura 31 a diferença do componente de login antes e após o login do usuário de email mariadasilva@gmail.com.

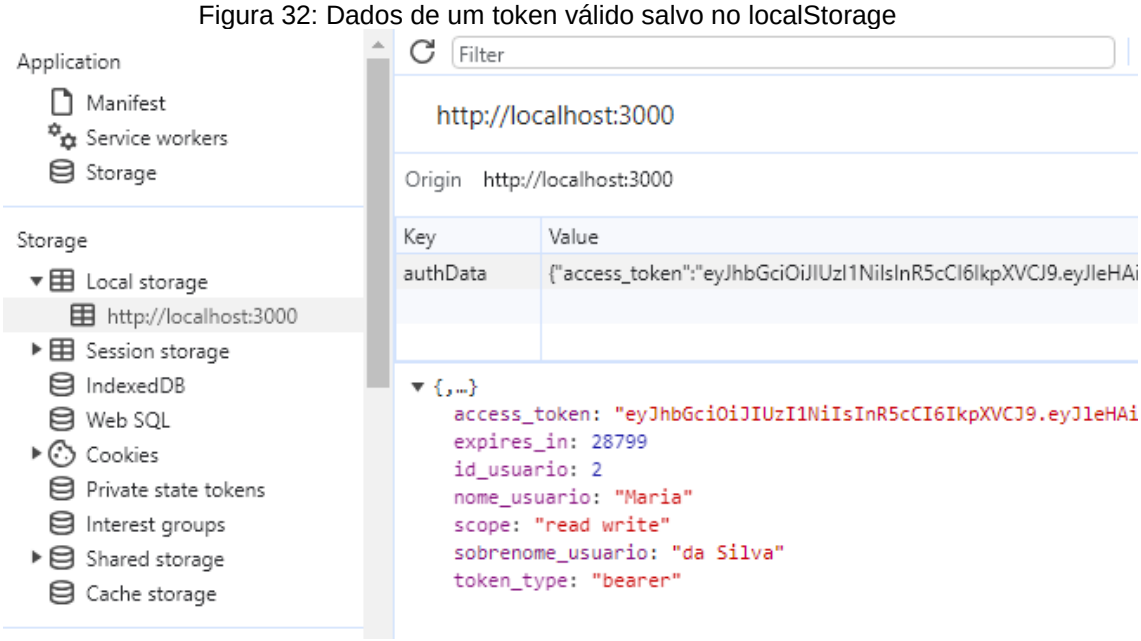
Figura 31: Componente de Login antes e após o usuário logar no sistema


Fonte: Autoria Própria

Para a realização desses componentes dinamicamente, é necessário o frontend do sistema saber se o usuário está logado ou não, e para isso foram usadas duas ferramentas: monitoramento do contexto global e o LocalStorare.[67].O LocalStorage é um recurso dos navegadores que permite a aplicação armazenar dados do usuário. No localStorage os dados não tempo de expiração. No caso da nossa aplicação o token tem válidade de 8 horas (28800 segundos) passado esse tempo o usuário tem que realizar novo login. Foi definido um tempo de 8 para expiração do token, visto que esse foi considerado um tempo razoável para o usuário não ter que ficar se autenticando toda hora e não deixar um tempo muito grande, por questão de segurança, caso o usuário esquecer de sair da aplicação ao finalizar o uso. O LocalStorage utilizada a estrutura de dados do tipo chave e valor e os dados são sempre armazenados no formato de string. A Figura 32 mostra o LocalStorage da aplicação com token válido de um usuário logado de nome Maria da Silva e os dados do bearer token.

[67]Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage/>. Acesso em: 20 de outubro de 2023.

Figura 32: Dados de um token válido salvo no localStorage




Fonte: Autoria Própria

O monitoramento do contexto global é feita através da criação de um contexto com dois parâmetros: o dado a ser monitorado e um função para setar (atribuir) um novo valor para esse dado. No exemplo acima o botão de login fica monitorando se usuário possui um token válido no localStorage, se sim, ele renderizado o botão com o nome do usuário, caso contrário fica apenas o botão de login. Cada vez que é alterado esse dado no LocalStorage (usuário faz logoff ou realiza login com outro usuário) o componente reage a essa alteração e renderiza novamente. Para esse monitoramente de estado global foi utilizada a Context[68] API do React que permite que um componente do tipo UseState possa posso criar um estado global e que qualquer componente da aplicação possa observar e reagir as alterações de algum dado desse componente.

A Tabela 3 mostra quais permissões o usuário necessita para acessar determinada área restrita.

Tabela 3: Rotas  e Permissões de Recursos Protegidos

| **Rota** | **Permissão necessária** | **Acesso** |
|---|---|---|
| /admin/sms | ROLE_CONSULTOR | Privado |
| /admin/categorias | ROLE_GERENTE_LOJA | Privado |
| /admin/categorias/inserir | ROLE_GERENTE_LOJA | Privado |
| /admin/categorias/:categoriaId | ROLE_GERENTE_LOJA | Privado |
| /admin/embalagens | ROLE_GERENTE_LOJA | Privado |
| /admin/embalagens/inserir | ROLE_GERENTE_LOJA | Privado |
| /admin/embalagens/:embalagemId | ROLE_GERENTE_LOJA | Privado |
| /admin/produtos | ROLE_GERENTE_LOJA | Privado |

[68]Disponível em: < https://legacy.reactjs.org/docs/context.html />. Acesso em: 10 de agosto de 2023.

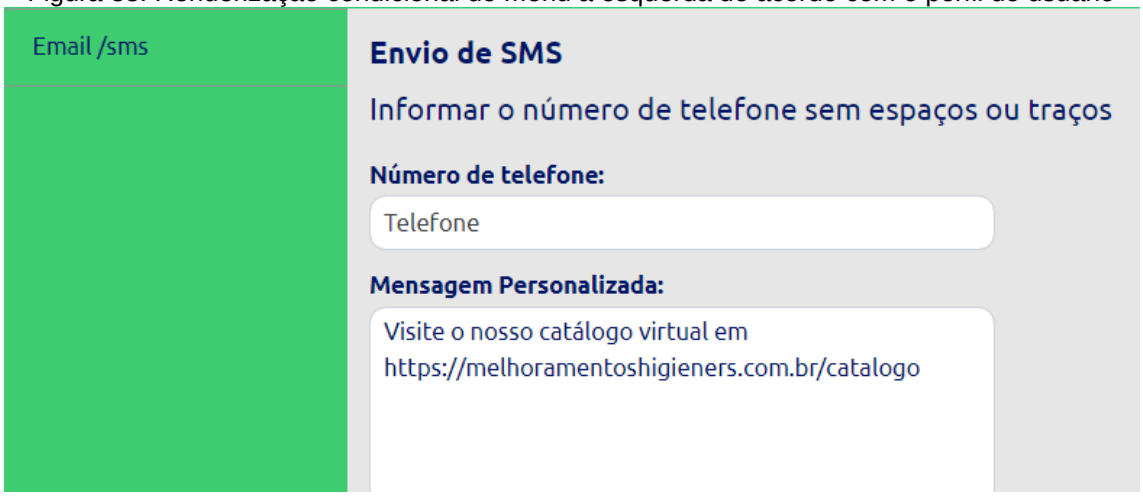
| | | |
|---|---|---|
| /admin/produtos/inserir | ROLE_GERENTE_LOJA | Privado |
| /admin/produtos/:produtoId | ROLE_GERENTE_LOJA | Privado |
| /admin/usuarios | ROLE_ADMIN_SISTEMA | Privado |
| /admin/usuarios/inserir | ROLE_ADMIN_SISTEMA | Privado |
| /admin/usuarios/:usuarioId | ROLE_ADMIN_SISTEMA | Privado |

Fonte: Autoria Própria

Após realizar login, o usuário pode ter acesso há três funcionalidades de acordo com o perfil do usuário. As funcionalidades são de enviar sms ao cliente, cadastrar produtos, categorias e embalagens e gerenciar usuário do sistema. Primeiramente será exposto como fica o menu da área de recursos protegidos de acordo com o perfil do usuário logado e depois a exibição do usuário realizando algumas das funcionalidades disponíveis.

Utilizando como exemplo da base de testes o usuário de email claudiadasilva@gmail.com, que tem apenas permissão de Consultor. Ao logar na área administrativa apenas vai renderizar o componente de envio de sms no menu a esquerda do painel do administrador, conforme é possível observar pela Figura 33.

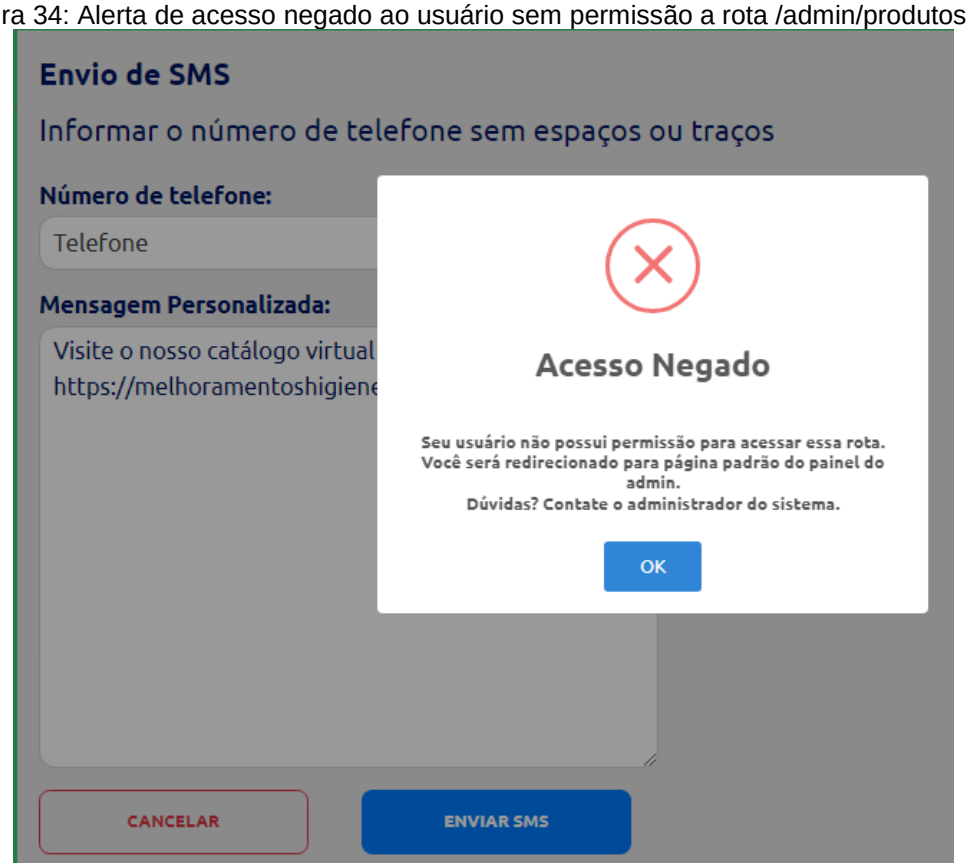
Figura 33: Renderização condicional do menu a esquerda de acordo com o perfil do usuário




Fonte: Autoria Própria

Entretanto, mesmo se o usuário souber o caminho de uma rota protegida a qual ele não tem permissão e tentar digitar o endereço da rota direto no navegador, ele vai receber uma resposta de acesso negado a requisição. A Figura 34 exibe essa resposta do frontend de acesso negado com o alerta na tela. A Figura 34 mostra a usuária Claudia da Silva tentando acessar o endpoint /admin/produtos, o qual seria necessário ter a permissão de Gerente de Loja.

Figura 34: Alerta de acesso negado ao usuário sem permissão a rota /admin/produtos




Fonte: Autoria Própria

Uma das funcionalidades da aplicação é o consultor poder enviar a página do catálogo com uma mensagem personalizada para o celular de um cliente via sms, como é possível observar pela Figura 30. Para envio de sms (Short Message Service) foi utilizada a API do twillio[69], que é uma ferramenta que fornece ferramentas de comunicação de forma programáveis. Apesar de ser uma ferramenta paga o twillio tem um plano grátis (com funcionalidades limitadas) que podem ser usado para teste. Para podermos ver todas as opções do menu área de recursos protegidos renderizadas foi feita o login com o usuário de email rafaeldeluca@gmail.com, que possui todas as permissões (consultor, gerente de loja e administrador de sistemas). A Figura 30 também mostra todos as opções da área de recursos protegidos exibidas no menu a esquerda.

Outra funcionalidade da área de recursos protegidos é o usuário, com permissão de gerente de loja, poder gerenciar produtos, categorias e embalagens. A Figura 35 exibe a listagem de todas as categorias em ordem alfabética e os botões de inserir uma nova categoria, editar ou excluir uma categoria existente.

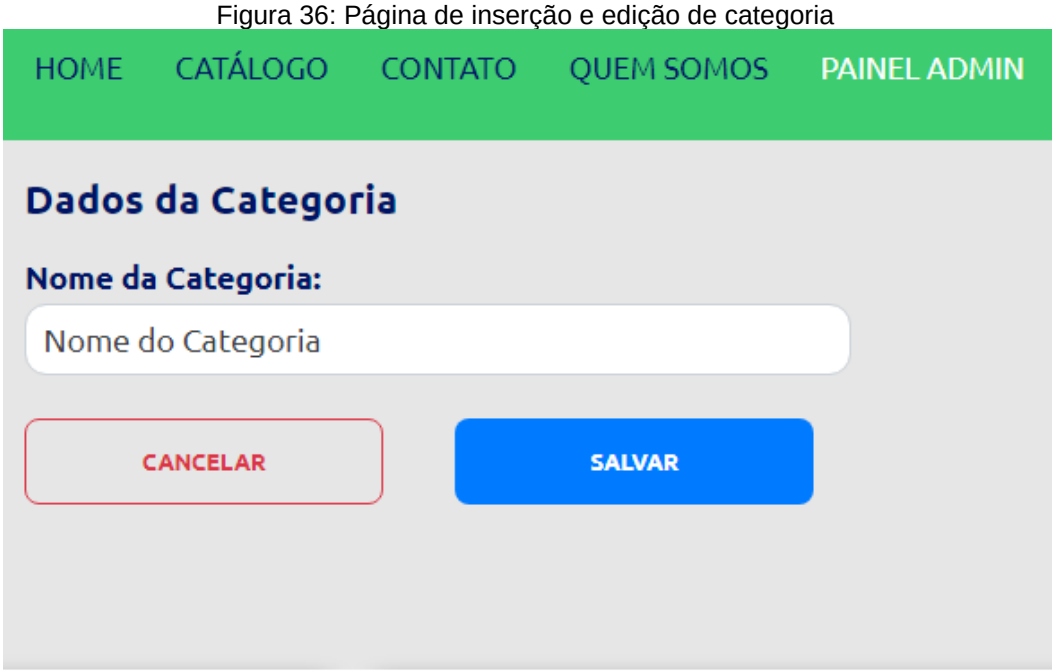
[69]Disponível em: < https://www.twilio.com/pt-br />. Acesso em: 10 de outubro de 2023.

Figura 35: Página de listagem de categorias e botão para inserir nova categoria


Fonte: Autoria Própria

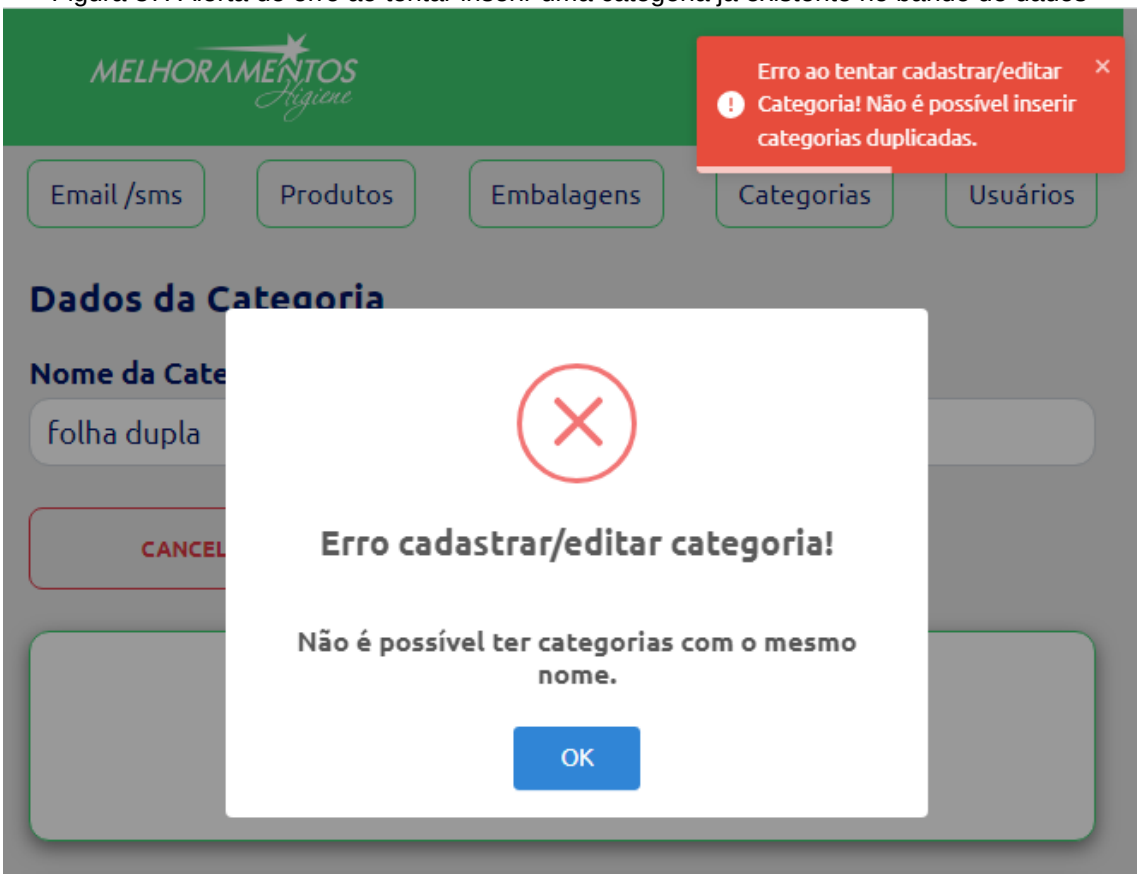
A Figura 36 mostra a área de inserção de cadastro de uma nova categoria, onde são feitas todas as validações dos campos já no frontend; entretanto, também é feita uma validação dos dados no backend antes de inserir os mesmos no banco de dados, conforme já foi mencionado no capítulo anterior. Não é possível inserir uma categoria com a mesma descrição no banco de dados, ao tentar realizar essa operação o usuário recebe um alerta na tela e um toast com a mensagem de erro conforme mostra a Figura 37 e o usuário é redirecionado para a página de listagem de categorias. No exemplo da Figura 37 o usuário tenta inserir uma categoria com a descrição de “folha dupla”, a qual já existe no bando de dados.

Figura 36: Página de inserção e edição de categoria




Fonte: Autoria Própria

Figura 37: Alerta de erro ao tentar inserir uma categoria já existente no bando de dados




Fonte: Autoria Própria

Caso o usuário preencha o campo de descrição de categoria corretamente e não exista uma categoria com essa descrição no banco, o usuário recebe um toast no canto superior direito de categoria inserida com sucesso e a página de listagem de categorias é renderizada novamente. A Figura 38 mostra o usuário inserindo com sucesso a categoria com a descrição de “amaciante” e a mesma já é renderizada na página novamente exibindo as categorias novamente listadas em ordem alfabética.

Figura 38: Inserindo uma categoria com sucesso com o nome Amaciante

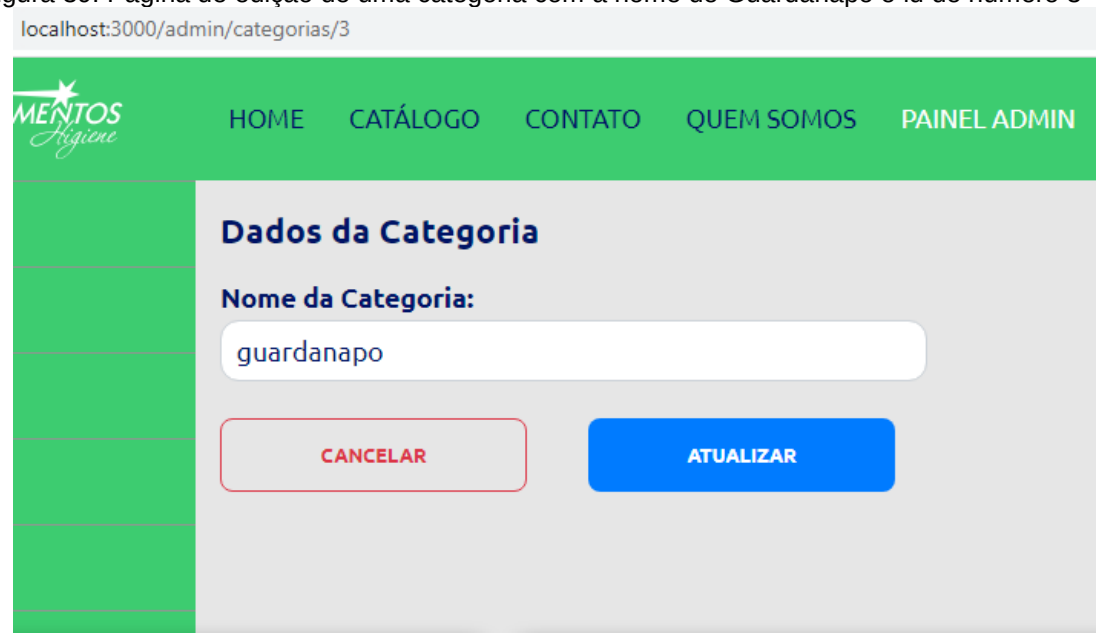

Fonte: Autoria Própria

Para essa renderização da listagem dinamicamente, sem o usuário ter que atualizar a página, foi utilizado o Padrão Observador[70] (Observer Pattern) que consiste em ficar monitorando um objeto e reagir a alguma mudança no estado (dado monitorado) desse objeto. No caso foi criando um componente useSate que fica monitorando a lista de categorias, caso seja inserida, editada ou excluída alguma categorias do banco de dados a página renderiza a lista novamente. Na área edição (atualização) de uma categoria é renderizada a mesma página que na área de inserção da categoria. A aplicação verifica que o usuário clicou no botão de edição e já busca do backend a descrição da categoria e o botão que antes estava com a descrição de “salvar” agora é renderizado com a descrição de “atualizar” conforme exibe a Figura 39. Na Figura 39 podemos observar que o frontend também busca o id único da categoria e informa o mesmo na barra de endereços do

[70]Disponível em: <https://tableless.com.br/design-patterns-em-javascript-observer/>. Acesso em: 10 de outubro de 2023.

navegador. Uma vez atualizada com sucesso a categoria, a listagem de categorias é renderizada novamente.

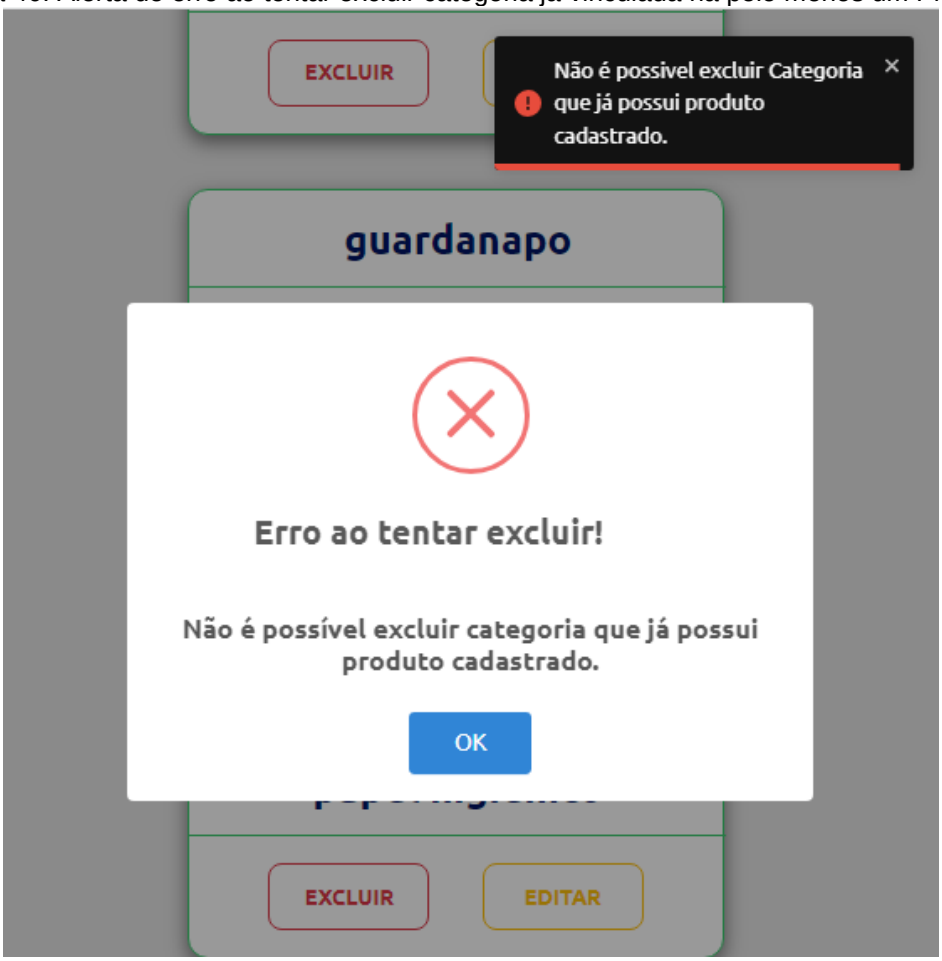
Figura 39: Página de edição de uma categoria com a nome de Guardanapo e id de número 3




Fonte: Autoria Própria

No processo de exclusão de uma categoria é realizada uma validação no banco de dados se já não existe um produto vinculado a essa categoria, e com o objetivo de não acontecer um erro de integridade referencial, não é permitida essa operação. A Figura 40 exibe um usuário tentando excluir uma categoria já vinculada a um produto e recebendo um alerta de operação não realizada. Caso a categoria não tenha nenhum produto vinculado a ela, ela é excluída com sucesso e a lista de categorias é renderizada e reordenada dinamicamente, já sem a categoria excluída.

Figura 40: Alerta de erro ao tentar excluir categoria já vinculada há pelo menos um Produto

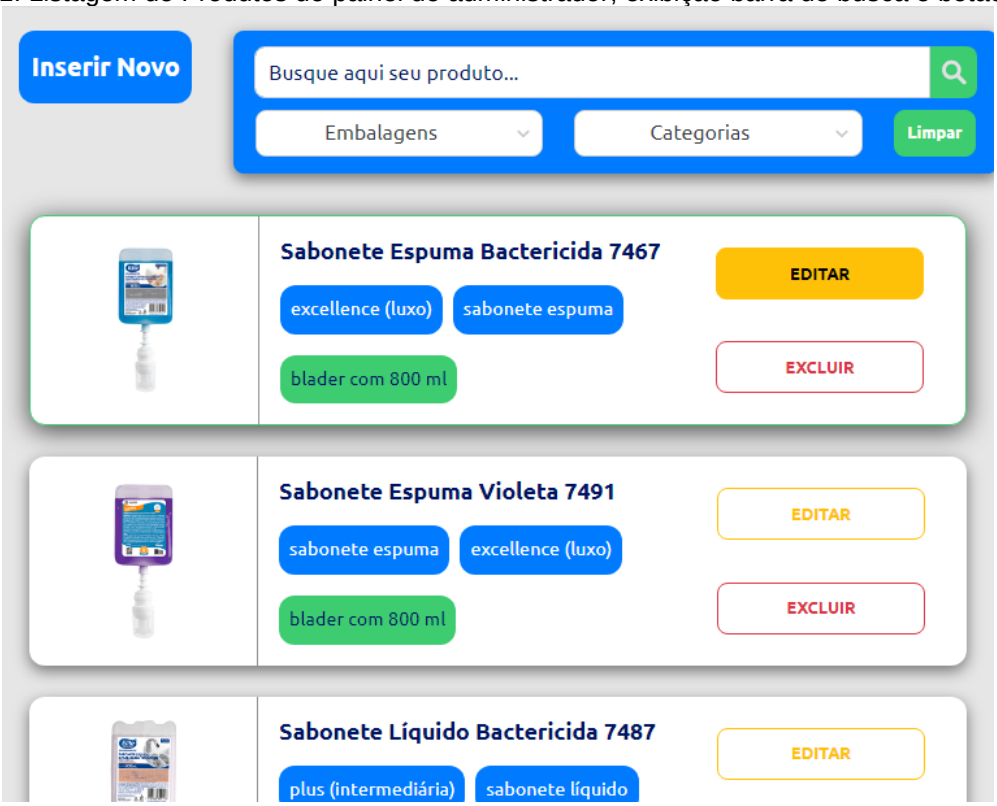



Fonte: Autoria Própria

Não se faz necessário à exibição as telas de gerenciamento de embalagens, visto que nelas são usados os mesmos componentes e validações que nas categorias, apenas consulta a tabela de embalagens agora. Assim sendo, seguem as telas mais complexas da aplicação que foram as telas de gerenciamento de produtos e gerenciamento de usuário.

Já na tela de listagem de produtos (do painel do administrador), não é exibida uma lista simples de produtos e sim uma lista paginada. Nessa lista são monitorados dois componentes: o componente de busca de produtos no topo da página e o componente de paginação no rodapé do card de listagem de produtos. Quando um estado (dado monitorado via componente useState) é alterado é um desses componentes, a listagem de produtos é renderizada dinamicamente, de acordo com o padrão observador conforme já mencionado. A Figura 41 mostra a listagem de produtos da área protegida onde junto com o a imagem e descrição do produto, também são exibidos componentes visuais das categorias e embalagens que esse produto pertence.

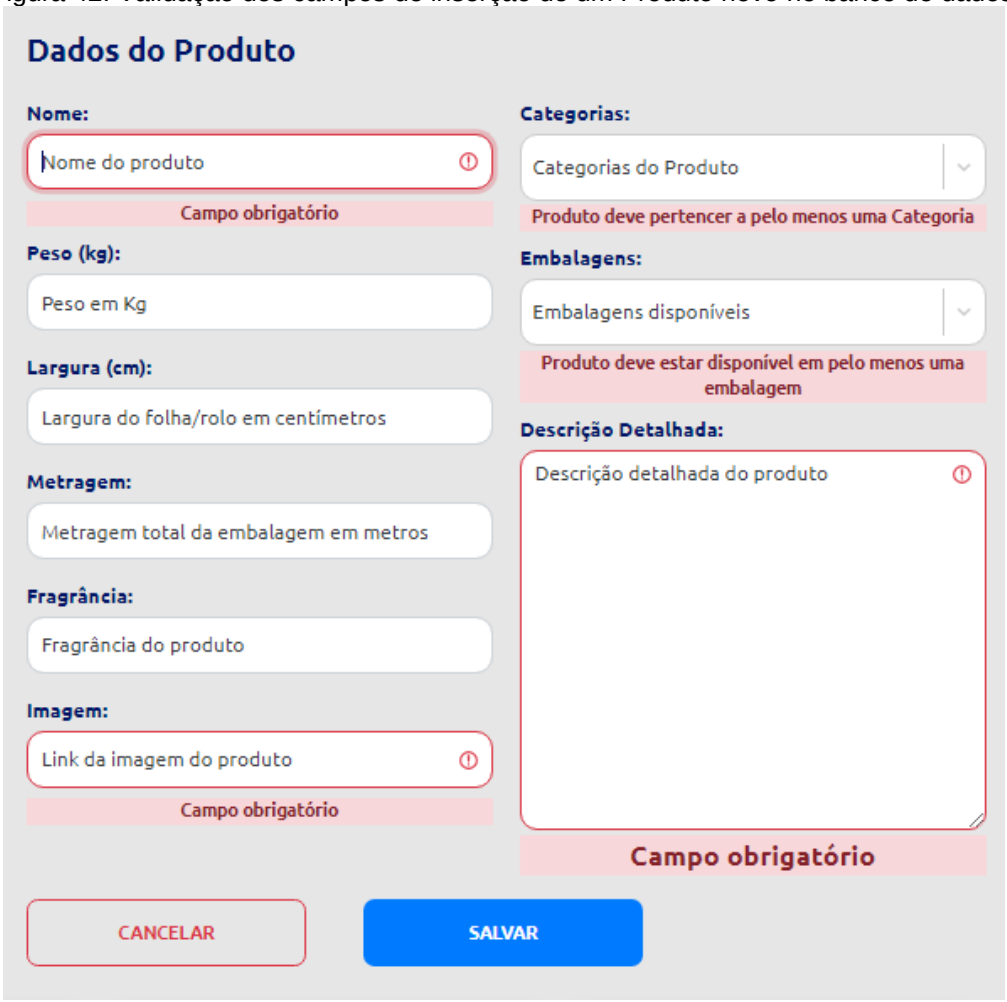
Figura 41: Listagem de Produtos do painel do administrador, exibição barra de busca e botão inserir.




Fonte: Autoria Própria

O usuário pode listar, inserir, editar e excluir produtos da base de dados. Na área de cadastro do produto são feitas validações mais complexas como produto deve estar vinculado à pela menos um categoria e embalagem, quantidade máxima e mínima de caracteres por campo, link da imagem deve ser uma url válida, entre outras; conforme mostra a Figura 42.

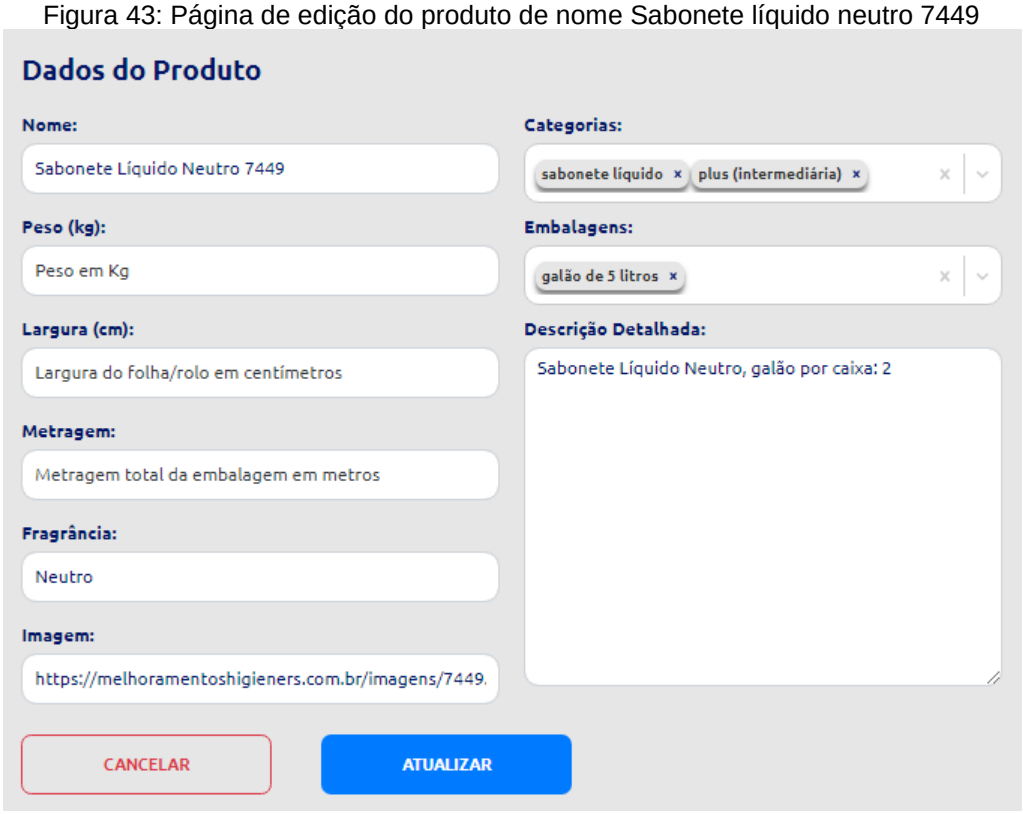
Figura 42: Validação dos campos de inserção de um Produto novo no banco de dados




Fonte: Autoria Própria

Se o usuário inserir todos os campos de maneira correta, após inserir um produto novo ele é direcionado a para a página de listagem de produtos e lista de produtos é renderizada já com o produto novo inserido. A página de edição, renderiza o mesmo componente de inserção de um produto novo, a diferença é que como é uma edição, a página já busca do backend os dados dos produtos e preenche os campos do formulário e a descrição do botão “salvar” exobi agora o texto “atualizar”. A Figura 43 exibe edição de produto com descrição de sabonete líquido neutro 7449. O usuário também tem a opção de excluir um produto do banco de dados.

Figura 43: Página de edição do produto de nome Sabonete líquido neutro 7449



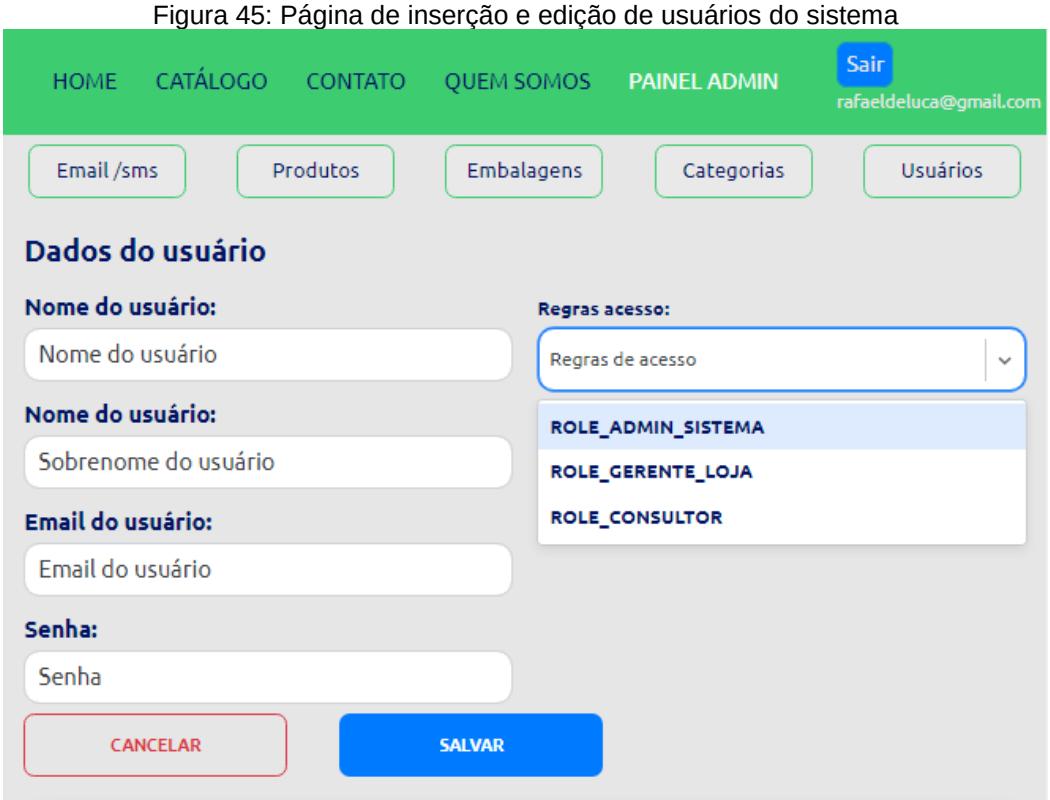

Fonte: Autoria Própria

Por último a tela de gerenciamento de usuários, no qual o usuário tem que ter a permissão de administrador de sistema para ter acesso. Nela são listados todos os usuários do sistema de suas respectivas permissões conforme a mostra a Figura 44. Por questão de regra de negócio, o botão excluir um usuário foi desabilitado conforme é possível verificar pela Figura 44. O usuário também pode inserir e editar um usuário do sistema, conforme exibe a Figura 45.

Figura 44: Página de listagem de usuários e suas permissões


Fonte: Autoria Própria

Figura 45: Página de inserção e edição de usuários do sistema




Fonte: Autoria Própria

## 6.2. TESTES

Conforme já mencionado no capítulo de metodologia para realização de testes foi utilizada a biblioteca JUnit5 do Java. Foram realizados testes na camada de controle para testar requisições, na camada de serviço para testar as lógicas de negócio da aplicação e na camada de repositório para testar os eventos de acesso ao banco de dados propriamente dito. O foco foi realizar testes na Entidade Produto que envolve listagem, inserção, edição e exclusão de produtos. Como para algumas operações de gerenciamento de Produtos é necessário estar autenticado, foi necessário simular autenticação de usuário com token válido.

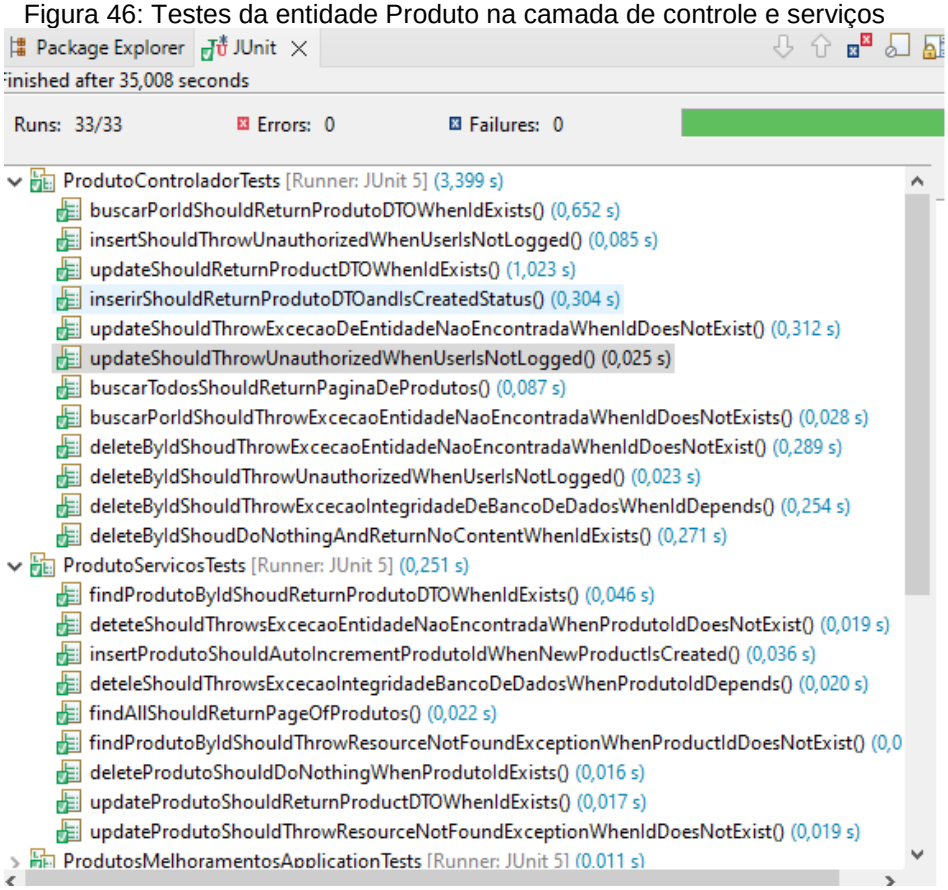
Na camada de repositório foram realizados os testes mais simples na entidades Embalagem e Produto. Testes unitários dos seguintes métodos: listar todas as embalagens, buscar uma embalagem por id, inserir uma nova embalagem, atualizar uma embalagem e excluir uma embalagem. Testes para os mesmos métodos foram realizados para a entidade Produto, simulando as situações de erro de tentar buscar um produto com id inexistente e tentar excluir um Produto já vinculado e uma embalagem ou categoria.

Na camada de serviços foram realizados testes funcionais onde antes de listar, salvar, atualizar e excluir uma entidade do banco de dado foi realizado todas as validações dos parâmetros (sanitização de variáveis) de acordo com as regras de negócio da aplicação. Ou seja, verificar antes de inserir ou atualizar um produto se os campos obrigatórios estavam preenchidos e de acordo com número máximo e mínimo de caracteres, verificar se havia pelo menos uma categoria e embalagem vinculadas à um Produto, verificar se a campo imagem do produto possuía uma url válida, entre outras validações.

Por fim foram realizados os testes mais complexos, os testes de integração da camada de controle. Estes testes foram os mais complexos visto que é necessário mocar (simular dados) da camada de serviço, simular uma autenticação de um usuário de acordo com seu perfil e ainda realizar todas as requisições do tipo HTTP, visto que a camada de controle faz a comunicação entre as requisições do frontend ao backend. Por exemplo, foi testado tentar cadastrar um produto sem estar autenticado, como está operação não é permitida, o erro esperado era um erro HTTP de código 401 (Unauthorized). Testado cadastrar um produto logado com o perfil de consultor, nesse caso era esperado um erro de requisição HTTP de código 403 (Forbidden). Testado também cadastrar ou atualizar um produto logado com o perfil de gerente de loja e neste caso a operação deveria ser realizada com sucesso, devolvendo uma resposta de requisição HTTP de código 201 (Created) para inserção e código 200 (Success) para atualização.

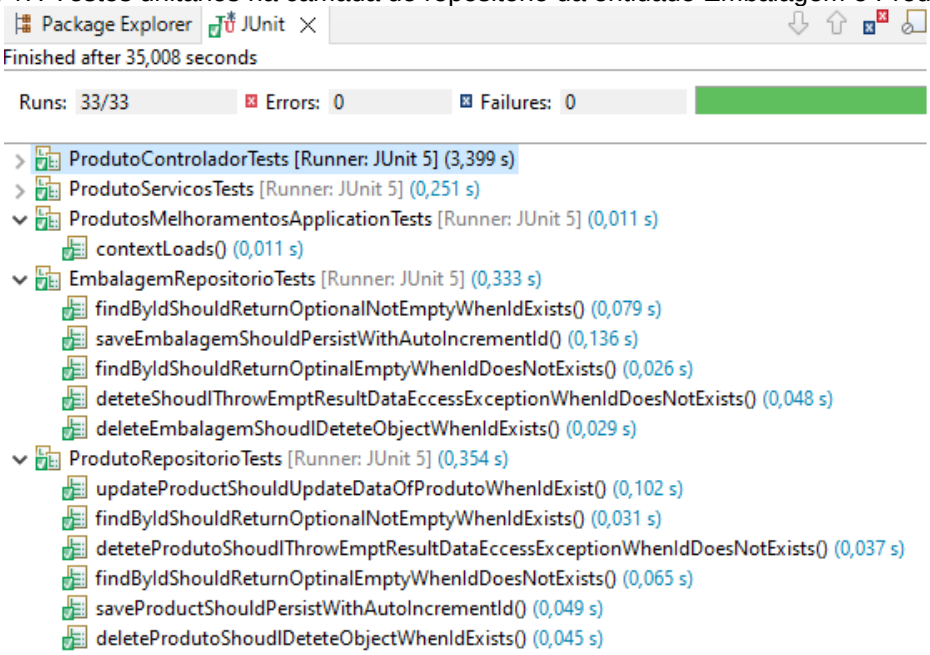
A Figura 46 exibe a descrição dos testes realizados na camada de controle e serviços. A Figura 47 exibe os testes realizados na camada de repositório da entidade Produto e entidade Embalagem. Foram realizados 33 testes com sucesso.

Figura 46: Testes da entidade Produto na camada de controle e serviços




Fonte: Autoria Própria

Figura 47: Testes unitários na camada de repositório da entidade Embalagem e Produto




Fonte: Autoria Própria

# 7 CONSIDERAÇÕES FINAIS

O principal objetivo desse trabalho foi implementar um sistema web de divulgação de produtos. A justificativa para a criação desse sistema foi desenvolver uma aplicação onde a distribuidora e fabricante Melhoramentos Higiene Ltda pudesse divulgar seus produtos de maneira agradável ao usuário final e gerenciar os mesmos de maneira eficiente. A metodologia utilizada foi uma pesquisa de abordagem qualitativa, exploratória e aplicada, utilizando-se de uma técnica de observação passiva e participante.

Com relação às tecnologias utilizadas foram utilizadas no frontend a linguagem de marcação HTML, de estilização CSS e de programação TypeScript. Junto com a biblioteca React e ambiente de desenvolvimento integrado Visual Code. Já com relação ao backend foi utilizada a linguagem de programação Java junto com o framework Spring Boot. No ambiente desenvolvimento integrado Intellij. Com relação aos bancos de dados foi utilizado um banco de dados em memória H2 no ambiente de desenvolvimento e testes e um banco de dados PostgreSQL no ambiente de produção. Por fim, foi utilizada a biblioteca Java Junit5 para realizar testes unitários e testes de integração.

É possível concluir que a aplicação se mostrou satisfatória no seu propósito, realizando assim uma exibição de um catálogo de produtos online elegante e de fácil gerenciamento de produtos e demais entidades. Entretanto, algumas melhorias são necessárias no sistema, como o desenvolvimento de uma funcionalidade do cliente solicitar orçamento baseado na pesquisa realizada. Outra melhoria seria desenvolver esse mesmo sistema em uma versão mobile, visto que foi utilizado React e Java, e assim sendo, seria totalmente compatível agora utilizar esse código como base para desenvolver uma aplicação mobile.

# REFERÊNCIAS

APIS RESTFUL. **Aprendendo arduino**, 2019. Disponível em: <https://aprendiendoarduino.wordpress.com/2019/10/27/api-rest/>. Acesso em: 17 jullho de 2023.

ASTAH. Leverage the power of software modeling. **ASTAH**, 2023. Disponível em: <https://astah.net/>. Acesso em: 10 de maio de 2023.

BARRO, Bruna. O que é uma API RESTful e porque isso importa. **HOSTINGER**, 2023. Disponível em: <https://www.hostinger.com.br/tutoriais/api-restful>. Acesso em: 20 de maio de 2023.

BAULDUNG. The DTO Pattern (Data Transfer Object). **BAULDUNG**, 2022. Disponível em: <https://www.baeldung.com/java-dto-pattern>. Acesso em: 20 de setembro de 2023.

CENTRAL DE AJUDA DO JAVA. **Java**, 2022. Disponível em: <https://www.java.com/pt-BR/download/faq/index_general.html>. Acesso em: 05 de junho de 2023.

CIGAM. Cigam software de gestão. **CIGAM**, 2023. Disponível em: <https://www.cigam.com.br/>. Acesso em: 15 de maio de 2023.

COMUNIDADE WEB BRASILEIRA. Design Patterns em JavaScript-Observer. **TABLELESS**, 2023. Disponível em: < https://tableless.com.br/design-patterns-em-javascript-observer/>. Acesso em: 15 de Agosto de 2023.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo (Org.). **Métodos de pesquisa**. Porto Alegre: UFRGS, 2009. Disponível em: <http://www.ufrgs.br/cursopgdr/downloadsSerie/derad005.pdf>. Acesso em: 10 de maio de 2022.

GIL, A.C. **Como elaborar projetos de pesquisa**. 4.ª Ed. São Paulo: Atlas, 2002.

HELP RESOURSES AND TIP. **Bring it all Together**, 2022. Disponível em: <https://docs.netlify.com/get-help/resources-and-tips>. Acesso em: 5 de junho de 2022.

INTERNET SOCIETY. The RFC Series. **RFC EDITOR**, 2023. Disponível em: <https://www.rfc-editor.org/rfc/rfc9110.html >. Acesso em: 20 de setembro de 2023

IBM. Modelos e Diagramas UML. **IBM**, 2023. Disponível em: <https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=models-uml-diagrams>. Acesso em: 10 de setembro de 2023

JETBRAINS. Intellij IDEA – O principal IDE para Java e Kotlin. **INTELIJ IDEA**. 2023. Disponível em: <https://www.jetbrains.com/pt-br/idea/>. Acesso em: 10 de maio de 2023.

JUNIT. The 5th major version of the programmer-friendly testing framework for Java and JVM. **JUNIT**, 2023. Disponível em: <https://junit.org/junit5/>. Acesso em: 20 de setembro de 2023.

JWT. JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securety between two parties. **JWO**, 2023. Disponível em: <https://jwt.io/>. Acesso em: 20 de setembro de 2023.

KHADRA, Fadi. React-toastify. **GITHUB**, 2023. Disponível em: <https://github.com/fkhadra/react-toastify>. Acesso em: 20 de outubro de 2023.

META OPEN SOURCE. React, a JavaScript library for building user interfaces. **LEGACY REACT JS**, 2023. Disponível em: < https://legacy.reactjs.org/>. Acesso em: 05 de Agosto de 2023.

META OPEN SOURCE. React, the library for web and native user interfaces. **REACT**, 2023. Disponível em: <https://react.dev/>. Acesso em: 10 de Agosto de 2023.

MIT. Build fat, responsive sites with bootstrap. **GETBOOTSTRAP**, 2023. Disponível em: <https://getbootstrap.com/>. Acesso em: 20 de agosto de 2023.

MOZILLA. Cross-Origin Resource Sharing (CORS). **MOZILLA**, 2023. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>. Acesso em: 25 de setembro de 2023.

MOZILLA. JavaScript. **MOZILLA**, 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 20 de setembro de 2023.

MOZILLA. Métodos de requisição HTTP. **MOZILLA**, 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>. Acesso em: 20 de setembro de 2023.

MOZILLA. Trabalhando com JSON. **MOZILLA**, 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/JSON>. Acesso em: 20 de setembro de 2023.

MOZILLA. Window: localStorare property. **MOZILLA**, 2023. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage/>. Acesso em: 20 de outubro de 2023.

OAUTH 2.0. RFC 6750: OAuth 2.0 Bearer Token Usage. **OAUTH**, 2023. Disponível em: <https://oauth.net/2/bearer-tokens/>. Acesso em: 20 de maio de 2023.

ORACLE. Class Date. **ORACLE**, 2023. Disponível em: <https://docs.oracle.com/javase/8/docs/api/java/util/Date.html>. Acesso em: 20 de maio de 2023.

ORACLE. Interface List. **ORACLE**, 2023. Disponível em: <https://docs.oracle.com/javase/8/docs/api/java/util/List.html>. Acesso em: 20 de maio de 2023.

ORACLE. Java Documentation. **ORACLE**, 2023. Disponível em: <https://docs.oracle.com/en/java >. Acesso em: 20 de maio de 2023.

PAWLOWSKI, C.S, ANDERSEM, H.B., TROELSEN, J. e SCHIPPERIJN, J. **Children's Physical Activity Behavior during School Recess: A Pilot Study Using GPS, Accelerometer, Participant Observation, and Go-Along Interview**. Disponível em <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0148786>. Acesso em 06 de junho de 2022.

POSTGRESQL. About. **POSTGRESQL**. 2023. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 20 de setembro de 2023.

POSTMAN. Postman is an APi platform for buiding and using APIs. **POSTMAN**, 2023. Disponível em: <https://www.postman.com/>. Acesso em: 10 de outubro de 2023.

RED HAT. Hibernate ORM. H**IBERTANETE**, 2023. Disponível em: <https://hibernate.org/orm/>. Acesso em: 30 de maio de 2023.

REFSNES DATA. CSS tutorial. **W3SCHOOLS**, 2023. Disponível em: <https://www.w3schools.com/css/>. Acesso em: 15 de setembro de 2023.

REFSNES DATA. HTML tutorial. **W3SCHOOLS**, 2023. Disponível em: <https://www.w3schools.com/html/>. Acesso em: 15 de setembro de 2023.

REFSNES DATA. Java Interface. **W3SCHOOLS**, 2023. Disponível em: <https://www.w3schools.com/java/java_interface.asp>. Acesso em: 20 de maio de 2023.

SARJEANT, John Jakob. Client for the browser and node.js. **AXIOS**, 2023. Disponível em: < https://axios-http.com/>. Acesso em: 25 de agosto de 2023.

SPRING. Spring Boot Overwiew. **Spring**, 2022. Disponível em: <https://spring.io/projects/spring-boot>. Acesso em: 15 de maio de 2023.

SPRING. Spring Boot Reference Documentation. **SPRING**, 2023. Disponível em: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. Acesso em: 15 de maio de 2023.

SPRING. Spring Boot maven Plugin Documentation. **SPRING**, 2023. Disponível em: <https://docs.spring.io/spring-boot/docs/current/maven-plugin/reference/htmlsingle/>. Acesso em: 25 de maio de 2023.

The Apache Software Foundation. Apache Mavem Project. **MAVEN**, 2023. Disponível em: <https://maven.apache.org/what-is-maven.html>. Acesso em: 20 de maio de 2023.

THE WORLD´S MOST ADVANCED OPEN SOURCE RELATIONAL DATABASE. **POSTGRESQL**, 2023. Disponível em: <https://www.postgresql.org>. Acesso em: 20 de junho de 2023.

TWILIO. Twilio para empresas. **TWILIO**, 2023. Disponível em: <https://www.twilio.com/pt-br/>. Acesso em: 10 de outubro de 2023.

TYPESCRIPT. TypeScript is JavaScript with sintax for types. **TYPESCRIPTLANG**, 2023. Disponível em: <https://www.typescriptlang.org/>. Acesso em: 10 de agosto de 2023.

VMWARE, Tanzu. Spring makes Java simple. **SPRING**, 2023. Disponível em: <https://spring.io/>. Acesso em: 20 de maio de 2023.

WHAT IS HEROKU. **Heroku**, 2023. Disponível em: <https://www.heroku.com/what>. Acesso em: 16 de junho de 2023.