Universidade Federal do Estado do Rio de Janeiro
Centro de Ciências Exatas e Tecnologia
Escola de Informática Aplicada

Desenvolvimento e Implementação de uma API para E-commerce

Jean Carlos Silva de Souza e Pedro Paulo Araujo de Andrade

Orientadora
Morganna Carmem Diniz

Rio de Janeiro, RJ – Brasil
Janeiro de 2024

Catalogação informatizada pelo autor

de Souza, Jean Carlos
d719 Desenvolvimento e implementação de uma API para e-commerce / Jean Carlos de Souza, Pedro Paulo de Andrade. -- Rio de Janeiro, 2024.
77p

Orientadora: Morganna Carmem Diniz.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2024.

1. Microsserviços. 2. API REST. 3. E-commerce. I. de Andrade, Pedro Paulo II. Carmem Diniz, Morganna, orient. III. Título.

API de apoio ao e-commerce

Jean Carlos Silva de Souza e Pedro Paulo Araujo de Andrade

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

_____

Morganna Carmem Diniz (UNIRIO)

_____

Mariano Pimentel (UNIRIO)

_____

Reinaldo Viana Alvares (UNIRIO)

Rio de Janeiro, RJ – Brasil.

Janeiro de 2024

Agradecimentos

Agradecimentos de Pedro Paulo de Andrade: Agradeço a todos os professores que pavimentaram o caminho até aqui e a minha esposa, Amanda Cristina de Andrade, pelo suporte na realização deste trabalho.

RESUMO

O comércio é uma prática antiquíssima e, como tudo, está em constante mudança. Atualmente, é muito comum comprarmos online, sem a necessidade de estar fisicamente em uma loja ou ser atendido por um vendedor. Essas aplicações web que utilizamos para compras online são nomeadas como *e-commerce*.

Este trabalho apresenta o desenvolvimento e a implementação de uma API REST, com arquitetura de microsserviços e de licenciamento livre, que pode ser acoplada a projetos de e-commerce de diferentes linhas de negócio.

Palavras-chave: Microsserviços, API REST, E-commerce.

ABSTRACT

Commerce is an ancient practice and, like everything, is constantly changing. Nowadays, it is very common for us to buy online, without the need to physically be in a store or be served by a salesperson. These web applications that we use for online shopping are called e-commerce.

The proposal of this work is an API REST, developed under a microservices architecture, with free licensing, that can be coupled to e-commerce projects from different lines of business.

Keywords: Microservices,.API REST, E-commerce.

Índice

Índice de Tabelas

Índice de Figuras

Capítulo 1: Introdução

1.1. Motivação

Com a popularização do acesso à internet e a evolução da *web*, o comércio se movimenta em direção à virtualização de suas atividades de relação com cliente − sejam relações de compra, venda ou pós-venda. Isso porque, segundo o G1 (2015, apud Springer, 2018) “no ano de 2000 havia 400 milhões de pessoas conectadas à internet, em 2015 este número já era superior a 3 bilhões”.

Tornou-se comum comércios de varejo terem, além de lojas físicas, sites para realizarem suas atividades comerciais. Pode-se citar marcas conhecidas no mercado como: Centauro, Casas Bahia, Carrefour, Americanas, Renner etc. Além destas, há marcas que realizam suas atividades inteiramente online, sem terem uma loja física, como, por exemplo: Kabuum, Terabyteshop, Shein, ShopFisio, Nerd Store etc. Nota-se que estas lojas abrangem diferentes mercados consumidores.

Além do crescimento do número de pessoas conectadas a internet, a pandemia do coronavírus (Covid-19), iniciada em 2020 e decretada terminada pelo OMS (Organização Mundial da Saúde) em 2023, acelerou o movimento de virtualização das atividades comerciais devido a necessidade de isolamento social para evitar a propagação do vírus.

De acordo com Almeida, Froemming e Ceretta (2020) apud SILVA et al. (2021):

> Esse novo contexto imposto pela crise sanitária gerou, por óbvio, reflexos no mercado de consumo bem como na atuação das empresas, que precisaram se reinventar e inovar, a fim de atender às novas demandas de atendimento aos consumidores e não serem sucumbidas pela crise

Posto isso, uma loja ter um site para ter contato com o cliente para realizar suas atividades comerciais passou de uma tendência para uma necessidade.

1.2. Objetivos

A virtualização dessas atividades se dá por meio de aplicações web, *e-commerce's*, termo conceituado como "uma abreviação de *eletronic commerce*, traduzindo comércio eletrônico" (Mendonça, 2016). Esta modalidade de comércio, outrora vista com desconfiança pelos consumidores, é considerada normal atualmente e é até preferível para alguns. De acordo com a pesquisa de Ebit-Nielsen (2020, apud SILVA et al., 2021),

> Em termos percentuais, a referida pesquisa evidenciou, no período referente ao primeiro semestre de 2020, um crescimento de 47% no faturamento das lojas online. Especificamente no decorrer do mês de abril de 2020, contexto inicial da pandemia no Brasil, o comércio eletrônico demonstrou um crescimento de 81% quando comparado ao mês de abril do ano anterior.

Ter um *e-commerce*, além ter se tornado necessário, é vantajoso, pois permite às empresas alcançarem mais clientes e serem capazes de servi-los, conforme MENDONÇA, 2016:

> Apesar da crise, o e-commerce brasileiro tem previsão de crescimento de 16% devido à quantidade de usuários cada vez maior e ao aumento da confiança em realizar as compras via internet (Mendonça, 2016).

Atualmente é comum os projetos de aplicações web (sites) se dividirem em duas sub aplicações que se comunicam. A primeira é a aplicação "*front end*" ou cliente, que é responsável por compor a interface gráfica apresentada para o usuário na tela. A segunda é a aplicação "*back end*" ou servidor, que é responsável por lidar e enviar as informações requisitadas pelo cliente ("*front end)*" que serão apresentadas para o usuário. As duas sub aplicações se comunicam por meio da internet.

Este presente trabalho irá desenvolver uma API. Uma API é uma aplicação que intermedia a comunicação entre dois softwares, neste caso, entre uma aplicação cliente e os bancos de dados a serem utilizados no desenvolvimento de *e-commerce's*. Portanto, a API atuará como um servidor.

Para facilitar as discussões sobre os requisitos do projeto, uma interface para uma empresa de varejo é usada como exemplo. Uma empresa de varejo tem como objetivo ter cada vez mais clientes para ter mais lucro. Logo, um *e-commerce* precisa estar sempre disponível para ser acessado e seu sistema deve ser pensado para receber um crescente número de acessos.

Para tal, é recomendado que o servidor tenha as seguintes características:

(a) disponibilidade (espera-se que o tempo de operação do sistema seja próximo de 100%);

(b) escalabilidade (facilidade em aumentar o atendimento para um número maior de clientes);

(c) manutenibilidade (facilidade de manutenção dos equipamentos sem a necessidade de parar todas as operações);

(d) tolerante a falhas (a ocorrência de falhas não torna o sistema inoperante).

Para o cenário acima, uma arquitetura de microsserviços para o servidor é mais recomendada, pois o sistema poderá escalar mais facilmente para atender uma demanda maior e será resiliente a falhas. Conforme explicitado por (Moreira; Beder, 2015), "às vantagens no uso de micro serviços podem ser caracterizadas como heterogeneidade tecnológica, resiliência, escalabilidade e facilidade de implantação".

Considerando o cenário discutido acima, este trabalho tem como objetivo apresentar um servidor web. Este servidor é baseado na arquitetura de microsserviços com as funções de compra, venda e inventário de produtos e poderá ser utilizado para o desenvolvimento de qualquer *e-commerce. O* servidor será distribuído sob a licença GNU GPLv3 (*GNU General Public License v3.0*) através de um repositório na plataforma *Github,* no endereço: https://github.com/pedroAndrad1/api-apoio-ecommerce. Sob essa licença, o software terá seu código fonte disponível, poderá ser modificado e redistribuído. Contudo, suas redistribuições deverão seguir a mesma licença.

> Os desenvolvedores que escrevem software podem lançá-lo nos termos da GNU GPL. Quando o fizerem, será software livre e permanecerá software livre, não importa quem mude ou distribua o programa. (Smith, 2022, traduzido por Fontenelle)

1.3. Organização do texto

O presente trabalho está estruturado, além deste capítulo inicial, em mais cinco capítulos.

- Capítulo 2: Definição do Sistema;
- Capítulo 3: Tecnologias;
- Capítulo 4: Documentação;
- Capítulo 5: Teste de uso;
- Capítulo 6: Conclusão.

Capítulo 2: Definição da API

Este capítulo apresentará as características da API de apoio ao e-commerce, a fim de descrever os requisitos que o sistema deverá cumprir.

2.1. Requisitos Funcionais

Os requisitos funcionais definem quais são as ações que o sistema deve realizar. Resumindo, o que o sistema deve fazer. A API de apoio ao e-commerce deverá cumprir os seguintes requisitos funcionais:

- RF001 - O sistema deve registrar produtos;
- RF002 - O sistema deve alterar produtos;
- RF003 - O sistema deve listar o inventário de produtos;
- RF004 - O sistema deve inativar um produto, mas não deletar seu registro;
- RF005 - O sistema deve ativar um produto (classificado como disponível para venda)
- RF006 - O sistema deve registrar pedidos de venda;
- RF007 - O sistema deve listar pedidos de venda;
- RF008 - O sistema deve permitir a alteração da quantidade disponível de um produto no inventário;
- RF010 - O sistema deve listar os produtos ativos;
- RF011 - O sistema deve listar os produtos ativos e inativos;
- RF012 - O sistema deve permitir registrar um item no inventário

2.2. Requisitos não funcionais

Diferente dos requisitos funcionais, os requisitos não funcionais definem aspectos técnicos que o sistema deve ter. Esses aspectos não são ações que o sistema deve realizar, mas sim características do sistema. Um exemplo de requisito não funcional seria: o sistema deve ter um tempo máximo de resposta de cinco segundos. A API de apoio ao e-commerce deverá cumprir os seguintes requisitos não funcionais:

- RNF001 - O sistema deve resistir a falhas, ou seja, não pode ficar totalmente fora de serviço;
- RNF002 - O sistema deve ser escalonável;
- RNF003 - O sistema deve ter um mecanismo de autorização OAUTH 2.0[1]

[1] *O protocolo de autorização OAuth 2.0 permite que terceiros tenham acesso limitado a um serviço HTTP em nome de um proprietário de recursos, com a permissão do proprietário de*

2.3. Casos de uso

2.3.1. Definição de caso de uso

Um caso de uso identifica as interações com o sistema. Um caso de uso abrange vários cenários. Existirá um cenário para representar a interação normal e outros para cada possível exceção (Sommervile, 2007 apud Domínguez, 2010). Um caso de uso contém os seguintes componentes:

- Objetivo:
  - Como o nome explicita, define qual o objetivo que as ações do caso de uso tem. Geralmente, este objetivo é um requisito funcional;
- Atores:
  - Os atores são aqueles que executam as ações;
- Pré-condições:
  - Pré-condições são exigências que devem ser satisfeitas antes que as ações do caso de uso possam ser executadas;
- Fluxo principal:
  - O fluxo principal é a sequência de ações esperada que aconteça. É um cenário onde o objetivo do caso de uso é alcançado.
- Fluxos alternativos:
  - Os fluxos alternativos são sequências de ações que ou não alcançam o objetivo do caso de uso, ou o alcançam por um caminho diferente do fluxo principal. Geralmente descrevem cenários de erro.

2.3.2. Lista de casos de uso

A seguir são listados casos de uso que descrevem o funcionamento da API de apoio a e-commerce e, a seguir, estes casos de uso são detalhados:

---

recursos (HARDT, 2012 apud BOUSSIENGUI, 2023). O cliente se identifica (autenticação) para o servidor e este retorna uma chave (token) que o cliente informará nas próximas requisições enviadas ao servidor.

1. **Realizar autenticação**
    a. Realizar autenticação para poder utilizar os recursos do servidor.
2. **Registrar produto**
    a. Salvar um produto novo.
3. **Alterar produto**
    a. Alterar as propriedades de um produto previamente salvo.
4. **Listar inventário**
    a. Listar todos os produtos salvos.
5. **Inativar produto**
    a. Trocar o status de um produto para inativo, ou seja, não disponível para venda.
6. **Ativar produto**
    a. Trocar o status de um produto para ativo, ou seja, disponível para venda.
7. **Registrar pedido de venda**
    a. Salvar um pedido de venda.
8. **Listar pedidos de venda**
    a. Listar todos os pedidos de venda salvos.
9. **Alterar inventário**
    a. Alterar as propriedades do inventário de produtos.
10. **Listar produtos ativos**
    a. Listar todos os produtos ativos.
11. **Listar todos os produtos**
    a. Listar todos produtos independe do status ativo ou inativo.
12. **Adicionar produto ao inventário**
    a. Adicionar um novo produto ao inventário

2.3.3. Descrição dos casos de uso

1. Realizar autenticação

Objetivo:

Autenticar usuário para ser possível utilizar os outros recursos.

Atores:

Aplicação Front End, KeyCloak.

Fluxo Principal:

1. O ator Aplicação Front End direciona o usuário que a está utilizando para se autenticar no KeyCloak;
2. O usuário da Aplicação Front End comunica suas credenciais para o KeyCloak;
3. O KeyCloak redireciona o usuário de volta para a Aplicação Front End e retorna para a Aplicação Front End um token para ser informado ao tentar utilizar os outros recursos da API de apoio ao e-commerce.

Requisitos não funcionais atendidos:

- RNF003 - O sistema deve ter um mecanismo de autorização OAUTH 2.0

2. Registrar produto

Objetivo:

Salvar um produto novo no banco de dados.

Atores:

Aplicação Front End.

Pré condições:

Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, as informações do novo produto junto do token de autenticação;
2. A aplicação devolve uma mensagem HTTP com status de sucesso e um texto sinalizando sucesso na operação.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, as informações do novo produto junto do token de autenticação;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido.
2. Fluxo Alternativo 2, produto com formato inválido:
    a. O ator envia para a aplicação, via requisição HTTP, as informações do novo produto junto do token de autenticação;
    b. A aplicação verifica que o formato do objeto produto é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o formato do produto é inválido.
3. Fluxo Alternativo 3, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, as informações do novo produto junto do token de autenticação;
    b. A aplicação não consegue salvar o produto no banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.

Requisitos funcionais atendidos:

- RF001 - O sistema deve registrar produtos

3. Alterar produto

Objetivo:

Alterar as propriedades de um produto salvo no banco de dados.

Atores:

Aplicação Front End.

Pré condições:

Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados e o produto deve estar salvo no banco de dados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, as informações a serem alterados no produto, junto do id do produto e token de autenticação;
2. A aplicação devolve uma mensagem HTTP com status de sucesso e um texto sinalizando sucesso na operação.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, as informações a serem alterados no produto, junto do id do produto e token de autenticação;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido.
2. Fluxo Alternativo 2, produto com formato inválido:
    a. O ator envia para a aplicação, via requisição HTTP, as informações a serem alterados no produto, junto do id do produto e token de autenticação;
    b. A aplicação verifica que o formato das informações a serem alteradas é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o formato das informações a serem alteradas é inválido.
3. Fluxo Alternativo 3, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, as informações a serem alterados no produto, junto do id do produto e token de autenticação;
    b. A aplicação não consegue salvar as alterações do produto no banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.
4. Fluxo Alternativo 4, produto inexistente
    a. O ator envia para a aplicação, via requisição HTTP, as informações a serem alterados no produto, junto do id do produto

e token de autenticação;

b. A aplicação não acha um produto com o id correspondente e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o produto não foi encontrado.

Requisitos funcionais atendidos:

- RF002 - O sistema deve alterar produtos;

4. Listar inventário

Objetivo:

Listar todos os produtos cadastrados com suas propriedades

Atores:

Aplicação Front End.

Pré condições:

Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
2. A aplicação devolve uma mensagem HTTP com status de sucesso e a lista de todos os produtos cadastrados.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido
2. Fluxo Alternativo 2, erro interno:

a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
b. A aplicação não consegue acessar os produtos no banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.

Requisitos funcionais atendidos:

- RF003 - O sistema deve listar o inventário de produtos;

5. Inativar produto

Objetivo:
Trocar o status de um produto de ativo para inativo e mantê-lo na base de dados.

Atores:
Aplicação Front End.

Pré condições:
Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto a ser inativado;
2. A aplicação devolve uma mensagem HTTP com um status de sucesso e uma mensagem que o produto foi inativado.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto a ser inativado;

    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido
2. Fluxo Alternativo 2, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto a ser inativado;
    b. A aplicação não consegue acessar os produtos no banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.
3. Fluxo Alternativo 3, produto inexistente
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto a ser inativado;
    b. A aplicação não acha um produto com o id correspondente e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o produto não foi encontrado.

Requisitos funcionais atendidos:

- RF004 - O sistema deve inativar um produto, mas não deletar seu registro;

6. Ativar produto

Objetivo:

Trocar o status de um produto de inativo para ativo.

Atores:

Aplicação Front End.

Pré condições:

Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto a ser ativado;
2. A aplicação devolve uma mensagem HTTP com um status de sucesso e uma mensagem que o produto foi ativado.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto a ser ativado;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido
2. Fluxo Alternativo 2, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto a ser ativado;
    b. A aplicação não consegue acessar os produtos no banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.
3. Fluxo Alternativo 3, produto inexistente
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto a ser ativado;
    b. A aplicação não acha um produto com o id correspondente e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o produto não foi encontrado.

Requisitos funcionais atendidos:

- RF005 - O sistema deve ativar um produto (classificado como disponível para venda)

7. Registrar pedido de venda

Objetivo:

Salvar um novo pedido de venda no banco de dados.

Atores:

Aplicação Front End.

Pré condições:

Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto, as informações sobre a venda e as informações sobre o comprador;
2. A aplicação devolve uma mensagem HTTP com um status de sucesso e uma mensagem indicando que a venda foi registrada com sucesso.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto, as informações sobre a venda e as informações sobre o comprador;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido
2. Fluxo Alternativo 2, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto, as informações sobre a venda e as informações sobre o comprador;
    b. A aplicação não consegue salvar a venda no banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um

erro interno.

3. Fluxo Alternativo 3, produto inexistente
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto, as informações sobre a venda e as informações sobre o comprador;
    b. A aplicação não acha um produto com o id correspondente e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o produto não foi encontrado.
4. Fluxo Alternativo 4, informações de comprador e/ou venda estão incorretas
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e o id do produto, as informações sobre a venda e as informações sobre o comprador;
    b. A aplicação identifica que as informações do comprador e/ou venda estão incorretas e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que as informações do comprador e/ou venda estão incorretas.

Requisitos funcionais atendidos:

- RF006 - O sistema deve registrar pedidos de venda;

8. Listar pedidos de venda

Objetivo:
Listar todos os pedidos de venda.

Atores:
Aplicação Front End.

Pré condições:
Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
2. A aplicação devolve uma mensagem HTTP com um status de sucesso e uma lista com todos os pedidos de venda registrados .

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido
2. Fluxo Alternativo 2, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
    b. A aplicação não consegue acessar o banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.

Requisitos funcionais atendidos:

- RF007 - O sistema deve listar pedidos de venda;

9. Alterar inventário

Objetivo:

Alterar a quantidade de um produto disponível no inventário.

Atores:

Aplicação Front End.

Pré condições:

Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados e o produto a ser alterado no inventário deve estar previamente salvo.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação, id do produto e a nova quantidade de um produto disponível;
2. A aplicação devolve uma mensagem HTTP com um status de sucesso e uma mensagem sinalizando que a operação foi realizada com sucesso.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação, id do produto e a nova quantidade de um produto disponível;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido
2. Fluxo Alternativo 2, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação, id do produto e a nova quantidade de um produto disponível;
    b. A aplicação não consegue acessar o banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.
3. Fluxo Alternativo 3, produto não encontrado
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação, id do produto e a nova quantidade de um produto disponível;
    b. A aplicação não consegue encontrar o pedido de venda no banco de dados devido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.

Requisitos funcionais atendidos:

- RF008 - O sistema deve permitir a alteração da quantidade disponível de um produto no inventário;

10. Listar produtos ativos

Objetivo:
Listar todos os produtos ativos.

Atores:
Aplicação Front End.

Pré condições:
Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
2. A aplicação devolve uma mensagem HTTP com um status de sucesso e uma lista com todos os produtos ativos.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido
2. Fluxo Alternativo 2, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
    b. A aplicação não consegue acessar o banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.

Requisitos funcionais atendidos:

- RF010 - O sistema deve listar os produtos ativos;

11. Listar produtos ativos

Objetivo:
Listar todos produtos independe do status ativo ou inativo.

Atores:
Aplicação Front End.

Pré-condições:
Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
2. A aplicação devolve uma mensagem HTTP com um status de sucesso e uma lista com todos os produtos ativos e inativos.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido
2. Fluxo Alternativo 2, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação;
    b. A aplicação não consegue acessar o banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que houve um erro interno.

Requisitos funcionais atendidos:

- RF011 - O sistema deve listar os produtos ativos e inativos;

12. Adicionar produto ao inventário

Objetivo:
Adicionar um produto ao inventário.

Atores:
Aplicação Front End.

Pré condições:
Os endereços do token de autenticação e dos bancos de dados devem estar previamente configurados.

Fluxo Principal:

1. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação e um novo item a ser salvo no inventário;
2. A aplicação devolve uma mensagem HTTP com um status de sucesso e uma mensagem sinalizando que a operação foi realizada com sucesso.

Fluxos alternativos:

1. Fluxo Alternativo 1, token inválido:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação, id do produto e a nova quantidade de um produto disponível;
    b. A aplicação verifica que o token é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o token é inválido
2. Fluxo Alternativo 2, erro interno:
    a. O ator envia para a aplicação, via requisição HTTP, uma mensagem com um token de autenticação, id do produto e a nova quantidade de um produto disponível;
    b. A aplicação não consegue acessar o banco de dados devido a um erro de conexão e devolve uma mensagem HTTP com status

de erro e uma mensagem sinalizando que houve um erro interno.

3. Fluxo Alternativo 3, item de inventário com formato inválido:
    a. O ator envia para a aplicação, via requisição HTTP, as informações do item de inventário junto do token de autenticação;
    b. A aplicação verifica que o formato do objeto produto é inválido e devolve uma mensagem HTTP com status de erro e uma mensagem sinalizando que o formato do item de inventário é inválido.

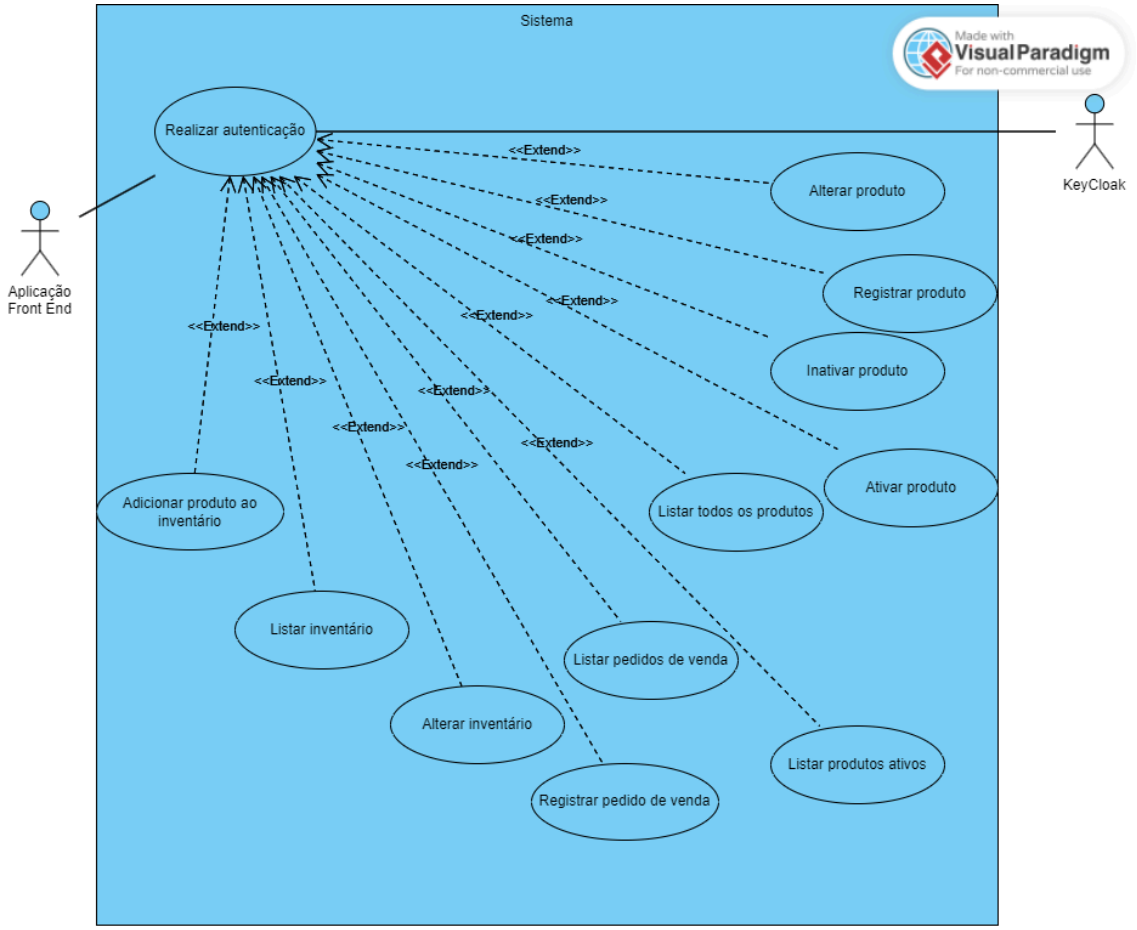
Requisitos funcionais atendidos:

- RF012 - O sistema deve permitir registrar um item no inventário;

2.3.4. Diagrama de casos de uso

O diagrama de casos de uso é uma representação visual de como os atores se relacionam com os casos de uso e como estes se relacionam entre si. A seguir, na Figura 3, o diagrama de casos de uso que representa os casos de uso apresentados para a API de apoio ao e-commerce:

Figura 3: Diagrama de casos de uso



Capítulo 3: Tecnologias

Neste capítulo serão apresentadas todas as tecnologias utilizadas no desenvolvimento da API de apoio ao e-commerce.

3.1. Arquitetura de Microsserviços

O princípio da arquitetura de Microsserviços é dividir o sistema em diversos subsistemas pequenos, coesos e independentes, nomeados como serviços. Cada serviço cumpre uma tarefa ou regra de negócio na aplicação (por exemplo, realizar a autenticação de um usuário) e, de preferência, sem precisar se comunicar com outros serviços.

> As vantagens no uso de micro serviços podem ser caracterizadas como heterogeneidade tecnológica, resiliência, escalabilidade e facilidade de implantação. (Moreira; Beder, 2015).

3.1.1. Comparação com Arquitetura Monolítica

A arquitetura Monolítica é a mais utilizada para o desenvolvimento de servidores web, ela consiste, a grosso modo, em todo o desenvolvimento ser feito em uma única aplicação executável. Embora ela apresente vantagens, ela também acarreta problemas como: crescente avanço da complexidade do código conforme o projeto evolui, dificultando a manutenção à longo prazo; ineficiência na escalabilidade do projeto, pois é necessário replicar todo o projeto em vez de apenas replicar as funcionalidades que são mais usadas; vulnerabilidade a falhas, pois se ocorrer um erro em alguma parte do código, todo o sistema fica indisponível.

> Em tradicionais sistemas monolíticos, principalmente em sistemas de grande porte, há um certo receio em realizar

> alterações em funcionalidades antigas e muitas vezes acopladas a outras funcionalidades, pois uma mudança pode afetar diversos módulos, podendo gerar erros em vários pontos, sem contar o custo de testar novamente todas as partes afetadas. (Moreira; Beder, 2015).

Estes problemas levaram a propostas de outras arquiteturas, dentre estas, a de Microsserviços. A API de apoio a e-commerce é implementada sob esta arquitetura.

3.1.2. Vantagens e Desvantagens

Conforme (Moreira; Beder, 2015), pode-se listar como vantagens do uso da arquitetura de microsserviços:

- Heterogeneidade tecnológica:
  - Em um cenário no qual os microsserviços utilizem a rede para se comunicar por meio de protocolo HTTP e troquem dados no formato JSON, a forma de implementação de cada microsserviço é irrelevante. Isso permite que a equipe escolha as tecnologias que melhor ampara as atividades do serviço. Por exemplo, um microsserviço pode ser escrito utilizando a linguagem Javascript e outro utilizando a linguagem Java. Ou, um microsserviço pode utilizar um banco de dados orientado a documentos e outro utilizar um orientado a grafos;

- Resiliência:
  - Cada microsserviço é implantado de forma separada. Isso significa que, em caso de falha em algum microsserviço, a aplicação não será comprometida como um todo, mantendo sua disponibilidade. Conforme (MOREIRA; BEDER, 2015), “Com micro serviços, caso um ou mais

falhe é possível isolar totalmente o problema. Com perda parcial das capacidades de um sistema";

- Escalabilidade:
    - Como cada microsserviço é implantado de forma separada, a escalabilidade do sistema é otimizada, pois não é necessário replicar todo o sistema, apenas os microsserviços que estão tendo maior carga de uso;

- Facilidade de implantação:
    - Cada microsserviço é um conjunto de código pequeno e coeso, um pequeno subsistema que compõe o todo. Dito isso, implantar um microsserviço se torna muito simples, já que são compostos por poucos arquivos e dependências.

Quanto às desvantagens em se utilizar uma arquitetura de microsserviços, podem-se citar:

- Dificuldade inicial no desenvolvimento:
    - Embora cada microsserviço deva ser um pequeno projeto, ainda assim, será preciso realizar as configurações iniciais para cada um destes projetos;

- Latência:
    - Apesar de ser ideal que cada microsserviço seja independente dos demais, em muitos casos um microsserviço terá que se comunicar com outro. Como essa comunicação se dá via rede, o tempo desta ação será adicionado ao tempo total de resposta da aplicação. Podendo haver uma latência maior que aplicações que

seguem outras arquiteturas, como a arquitetura monolítica, teriam;

- Complexidade de gerência:
    - Cada microsserviço, além de suas réplicas, terá um banco de dados próprio. É preciso que os dados entre os bancos estejam consistentes. Por exemplo, se um microsserviço faz uma alteração em seu banco de dados, é necessário que esta mudança seja feita em todas as suas réplicas e que essa informação seja consistente com as informações presentes nos bancos de dados de outros microsserviços. Além disso, existe a complexidade de gerência dos diversos times que fazem a manutenção dos microsserviços.

3.2. API REST e Servidor

A API de apoio ao e-commerce é definida como uma API REST. Uma API intermedia a comunicação entre dois softwares. Neste caso, entre uma aplicação Front End e os bancos de dados de cada microsserviço, REST significa Transferência Representacional de Estado. REST define um conjunto de funções como GET, PUT, DELETE e assim por diante, que os clientes podem usar para acessar os dados do servidor. Clientes e servidores trocam dados usando HTTP (Amazon Web Services, 2023). As API's no padrão REST tem requisições únicas, ou seja, um contexto de uma requisição não influencia em outra.

Um servidor, ou *back end*, aguarda a comunicação de outras aplicações (clientes). Conforme Teixeira, 2014: "Um servidor web é um processo que fica permanentemente aguardando por solicitações vindas dos clientes". Um cliente é definido como uma aplicação que inicia uma comunicação com um servidor.

Estes clientes podem ser tanto o navegador (*browser*) quanto um outro servidor web. Estas requisições podem ter diversos fins, como alterações no banco de dados, consultar informações contidas no banco de dados, autenticação de usuários, páginas HTML etc.

As API's são classificadas como servidores porque elas aguardam que algum cliente inicie uma comunicação com elas para devolver uma resposta. Lembrando que, uma API 1 pode mandar mensagem para outra API 2. Nesse caso, a API 1 é cliente da API 2, enquanto a API 1 é servidor de outro cliente que iniciou uma comunicação com ela.

3.3. Protocolo HTTP

O protocolo HTTP define o formato das mensagens trocadas entre cliente e servidor. Este protocolo será usado pela API de apoio ao e-commerce e pelo cliente que se comunicar com ela. Toda e qualquer informação que trafega entre os browsers e os servidores web o faz dentro de uma mensagem HTTP. (Texeira, 2014).

O formato da mensagem enviada pelo cliente deve ter o seguinte formato: A primeira linha (*Request line*) deve conter o verbo HTTP (indica a finalidade do requisição), a URL requisitada e a versão do protocolo HTTP; A segunda linha (*Request header*) contém campos sobre cliente que está fazendo a requisição, por exemplo, um token para autenticação; Por fim, a terceira linha (opcional) é o corpo da mensagem (*Request body*), que contém dados a serem tratados pelo servidor na requisição, como, por exemplo, informações a serem adicionados no banco de dados. A seguir, a figura 1 e a tabela 1 representam a estrutura da mensagem do cliente e os verbos HTTP com seus significados.

Figura 1: Representação da estrutura de uma mensagem HTTP do cliente

Fonte: (Ferreira, 2022)

Tabela 1: Verbos HTTP e seus significados

Verbo	**Finalidade**
GET	Solicita que seja retornado o recurso identificado pela URL
HEAD	Obtém informações sobre o recurso sem que o mesmo seja retornado ao cliente. Testa validade de links, acessibilidade e a data da última modificação
POST	Envia informações adicionais do cliente para o servidor no corpo da mensagem, por exemplo, dados digitados em formulários HTML
OPTIONS	Obtém as opções de comunicação disponíveis ou os requisitos associados ao recurso solicitado, sem necessariamente iniciar sua recuperação
PUT	Permite criar ou modificar um recurso no servidor web
DELETE	Solicita que o recurso identificado pela URL seja apagado do servidor web
TRACE	É usado para enviar uma mensagem de teste, do tipo ´ loopback, ao servidor
CONNECT	Reservado para comunicação com servidores proxy

Fonte: (Teixeira, 2014)

O formato da mensagem enviada pelo servidor, em resposta ao cliente, deve ter o seguinte formato: A primeira linha deve conter a versão do HTTP e o código do status da resposta (*Response code)*; A segunda linha deve conter o cabeçalho da resposta *(Response Header)*; Finalmente, a terceira linha deve conter o corpo da resposta (*Response body)*. A seguir, a figura 2 e tabela 2

representam a estrutura da mensagem do servidor e as classes de códigos de status com seus significados.

Figura 2: Representação da estrutura de uma mensagem HTTP do servidor

Fonte: (Ferreira, 2022)

Tabela 2: Classes de códigos de status e seus significados

Classe	**Descrição**
1xx	Finalidade informativa
2xx	Sucesso
3xx	Redirecionamento
4xx	Erro do cliente
5xx	Erro no servidor

Fonte: (TEIXEIRA, 2014)

3.4. E-commerce

A API de apoio ao e-commerce, como o nome explicita, tem como objetivo atuar como servidor para aplicações de e-commerce. O e-commerce é a prática comercial realizada por meios eletrônicos, mais comumente à internet, sem o comprador precisar ir a uma loja física para fazer uma compra. Esta modalidade de comércio já existia antes da internet, com compras e vendas

feitas por telefone e anúncios na televisão. Contudo, ganhou popularidade junto da popularização da internet.

> Desde o início da popularização da internet, milhares de empresas fixaram residência em sites na web, com um número crescente a cada ano. Desde então, o e-commerce vem ganhando espaço e importância entre as empresas que buscam inovar através desse tipo de estratégia e entre os consumidores que querem garantir preço baixo e praticidade às suas compras (Tassabehji, 2003; Makelainen, 2006, apud Mendes, 2013)

Para a empresa de varejo, ter um e-commerce tem vantagens como: economia de espaço físico e capacidade de atender mais clientes. Para o consumidor, comprar em um e-commerce tem vantagens como: facilidade para comparar preços e buscar opiniões de outros compradores e comodidade.

3.5. Spring Boot

O Spring boot é um framework open source para desenvolvimento de aplicações utilizando Java e foi utilizado para o desenvolvimento da API de apoio ao e-commerce. Ele facilita e agiliza o desenvolvimento de aplicações e micro serviços. Dentre suas capacidades o que mais se destaca é fornecer auto configuração para componentes trazendo agilidade e produtividade ao desenvolvimento. Principalmente para casos de um-para-um no uso de componentes; por exemplo, bancos de dados, o Spring Boot precisará somente de um arquivo de configuração que define a string de conexão, usuário e senha para fornecer ao desenvolvedor uma ambiente onde ele somente precisa se preocupar com a lógica de negócio.

3.6. Keycloak

O Keycloak é uma plataforma de gerenciamento de identidade e acesso de código aberto que oferece uma série de vantagens em comparação com a abordagem de desenvolver toda a autenticação de uma aplicação. Ao adotar o

Keycloak, os desenvolvedores podem aproveitar uma solução pronta e testada para autenticação e autorização, o que resulta em um desenvolvimento mais rápido, eficiente e seguro. A API de apoio ao e-commerce utiliza o Keycloack para os processos de autenticação e autorização.

O Keycloak oferece recursos de autenticação e autorização robustos, como autenticação de usuário, autenticação social e integração com provedores de identidade externos, como o Google e o Facebook. Esses recursos evitam a necessidade de implementar manualmente essas funcionalidades complexas, acelerando o processo de desenvolvimento.

Além disso, o Keycloak oferece um painel de administração intuitivo que simplifica o gerenciamento de usuários, grupos e permissões. Isso resulta em uma maior eficiência, pois os desenvolvedores podem controlar o acesso e a autorização, economizando tempo e esforço no desenvolvimento.

Outra vantagem do Keycloak é a sua capacidade de escalabilidade, suportando dimensionamento horizontal e distribuição em cluster. Isso significa que ele pode ser implantado em ambientes de alta demanda sem comprometer o desempenho, proporcionando uma solução eficiente e escalável.

Em termos de segurança, o Keycloak adota padrões modernos, como OAuth 2.0 e OpenID Connect, garantindo a integridade e a segurança dos dados transmitidos durante o processo de autenticação. Além disso, oferece recursos avançados de gerenciamento de permissões e grupos, permitindo um controle eficaz sobre quem pode acessar quais recursos nas aplicações.

Portanto, considerando a economia de tempo no desenvolvimento, a ampla gama de recursos disponíveis e a ênfase na segurança, é justificável afirmar que o uso do Keycloak resulta em um desenvolvimento mais rápido, eficiente e seguro para as aplicações que requerem recursos de autenticação e autorização.

3.7. Spring Cloud

As bibliotecas do pacote cloud do Spring fornecem ferramentas para a implementação de padrões de sistemas distribuídos, como configuração centralizada, descoberta de serviços, gerenciamento de mensagens e gateway de API. A API de apoio ao e-commerce utiliza o Spring Cloud para

implementar o microsserviço Gateway e o microsserviço de descoberta. Esses recursos prontos para uso evitam a necessidade de desenvolver toda a lógica do zero, resultando em uma significativa economia de tempo e esforço, além de permitir a reutilização de código previamente testado e validado.

Além disso, o Spring Cloud oferece mecanismos e abstrações para facilitar a comunicação entre os serviços distribuídos, como chamadas HTTP, balanceamento de carga e resiliência por meio de *circuit breakers*. Esses recursos contribuem para a eficiência e confiabilidade na interação entre os componentes do sistema.

Outra vantagem do uso das bibliotecas do Spring Cloud é o fato de fazerem parte de um ecossistema maior do Spring Framework, que possui uma ampla documentação, recursos online, fóruns e suporte disponíveis. Isso facilita o aprendizado e a resolução de problemas durante o desenvolvimento, garantindo uma implementação eficiente e bem sucedida.

3.8. Spring Data

A API de apoio ao e-commerce utiliza as bibliotecas do pacote data do Spring para acesso a dados, facilitando a conexão com bancos de dados relacionais e não relacionais, bancos hospedados na nuvem e MapReduce. O uso do Spring Data em conjunto ao Spring Boot agiliza o desenvolvimento de integrações com bancos de dados e permite a utilização do padrão JPA e da Biblioteca Hibernate.

3.9. JPA Buddy

O JPA Buddy é uma biblioteca para Java que simplifica o desenvolvimento de aplicativos utilizando a tecnologia Java Persistence API (JPA). A API de apoio ao e-commerce utiliza os recursos desta biblioteca para persistência de dados, mapeamento de entidades, consultas e transações ao banco de dados. A utilidade do JPA Buddy reside na redução da quantidade de código necessário para operações de persistência e no aumento da produtividade dos desenvolvedores. Além disso, o JPA Buddy possui uma funcionalidade de engenharia reversa que permite a geração automática de

classes de entidade e relacionamentos a partir de um banco de dados existente, facilitando a integração de um projeto com uma base de dados pré-existente. Essa funcionalidade agiliza o processo de criação de modelos de dados e acelera o desenvolvimento de aplicativos que utilizam JPA.

3.10. API Gateway

O API Gateway atua como um ponto de entrada único para os clientes, que enviam suas requisições para o Gateway. A API de apoio ao e-commerce utiliza-o para receber, processar e encaminhar essas requisições para os serviços adequados dentro da arquitetura distribuída. O Gateway também realiza a comunicação com o Keycloak, o sistema de gerenciamento de identidade e acesso escolhido por nós, para garantir a autorização correta dos usuários antes de direcioná-los aos serviços desejados. Além disso, o API Gateway realiza o mapeamento das URLs dos serviços, permitindo que os clientes acessem esses serviços de maneira simplificada e padronizada.

3.11. Spring Cloud Netflix

O serviço de descoberta implementado com o Spring Cloud Netflix (Eureka Server) facilita a localização e o registro de serviços em arquiteturas distribuídas. O Eureka Server atua como um repositório centralizado para serviços registrarem suas informações e obterem atualizações em tempo real. Os clientes Eureka se registram no servidor e consultam suas informações.

A API de apoio ao e-commerce utiliza o Eureka Serve para ser possível escalar horizontalmente os serviços de forma dinâmica, promovendo uma comunicação transparente entre os serviços e garantindo a resiliência do sistema.

3.12. Spring Cloud Config

O Spring Cloud Config (Config Server) é uma ferramenta para gerenciar as configurações de uma aplicação distribuída. A API de apoio ao e-commerce utiliza o Config Server como um repositório centralizado de configurações,

permitindo que os serviços obtenham suas configurações de forma dinâmica.

Os arquivos de configuração são armazenados em um repositório Git ou outro sistema de controle de versão, o que facilita o versionamento e o controle das configurações. Além disso, o Config Server suporta recursos avançados, como atualização automática das configurações e criptografia de dados sensíveis.

Essa abordagem oferece flexibilidade, facilita o gerenciamento e a padronização das configurações, e ajuda a garantir a consistência do ambiente distribuído. Em um ambiente que cresce de forma horizontal, o Config Server é a ferramenta que assegura a integridade das propriedades de configuração dos serviços e automatiza a manutenção dos arquivos de configuração.

Capítulo 4: Documentação

Este capítulo destina-se a explicar o passo a passo o uso da API, ilustrando algumas telas e demonstrando o comportamento do sistema. Os serviços precisam ser iniciados em uma ordem por haver dependência entre eles e é necessária a instalação do Mysql e do MongoDb para que sejam utilizados como bases de dados dos serviços. Após iniciada, a aplicação estará disponível localmente. A ordem de instalação são os tópicos .

4.1. Keycloak

Para poder iniciar os serviços é necessário baixar o Keycloak e iniciá-lo como descrito na documentação usando o comando correto para o sistema operacional que estiver sendo utilizado. Pode ser baixado em seu site oficial[2], nele pode ser encontrada a documentação[3] com todas as informações necessárias para configurar o Keycloak. Após iniciado, seguir para criação de um Realm. Na criação do Realm pode ser feita a importação de um arquivo json que contém as configurações do projeto. O json do Keycloak se encontra na pasta base do projeto[4] no Github. Incluímos um vídeo mostrando os passos para iniciar o Keycloak no Youtube[5].

4.2. Serviço de configuração

O serviço de configuração não depende do Keycloak para iniciar, portanto a ordem de inicialização não altera o fluxo. O serviço de configuração precisa da criação de cinco variáveis de ambiente para funcionar corretamente como podemos ver na imagem XX. As 5 variáveis que precisam ser criadas são:

- **Database_Drive**

[2] *https://www.keycloak.org/*
[3] *https://www.keycloak.org/documentation*
[4] *https://github.com/JeanCSDeSouza/sagui-parent*
[5] *https://youtu.be/eYXj3-qF1B8*

No ambiente de desenvolvimento estamos usando Mysql 8.0 e por este motivo o valor usado para essa variável é “com.mysql.cj.jdbc.Driver”. O valor dessa variável depende do banco que será utilizado. A aplicação está configurada para o Mysql, caso escolha usar Postgresql, SqlServer ou outro gerenciador de bancos de dados, a aplicação deve ser recompilada já que deve ser incluída no pom.xml a biblioteca do banco escolhido.

- **Database_Name**

O valor usado no desenvolvimento é “servico_configuracao” e esse é o nome do banco que precisa ser criado no gerenciador usando o bloco de códigos a seguir, este sql é genérico e, por tanto, pode ser aplicado a qualquer gerenciador de bancos de dados. A seguir o SQL para a criação do banco:

```
CREATE DATABASE IF NOT EXISTS ‘servico_configuracao’;
USE ‘servico_configuracao’;

CREATE TABLE IF NOT EXISTS ‘properties’ (
  `id` int NOT NULL AUTO_INCREMENT,
  `CREATED_ON` datetime DEFAULT NULL,
  `APPLICATION` varchar(255) DEFAULT NULL,
  `PROFILE` varchar(255) DEFAULT NULL,
  `LABEL` varchar(255) DEFAULT NULL,
  `PROP_KEY` varchar(255) DEFAULT NULL,
  `VALUE` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
)
```

- **Mysql_Url**

O valor desta variável no ambiente de desenvolvimento é “jdbc:mysql://localhost:3306/” se for mantido o MySql é este o valor que deve ser usado. Esta variável só precisa ser modificada caso a porta do Mysql seja diferente ou se o gerenciador de bancos de dados não for o MySQL

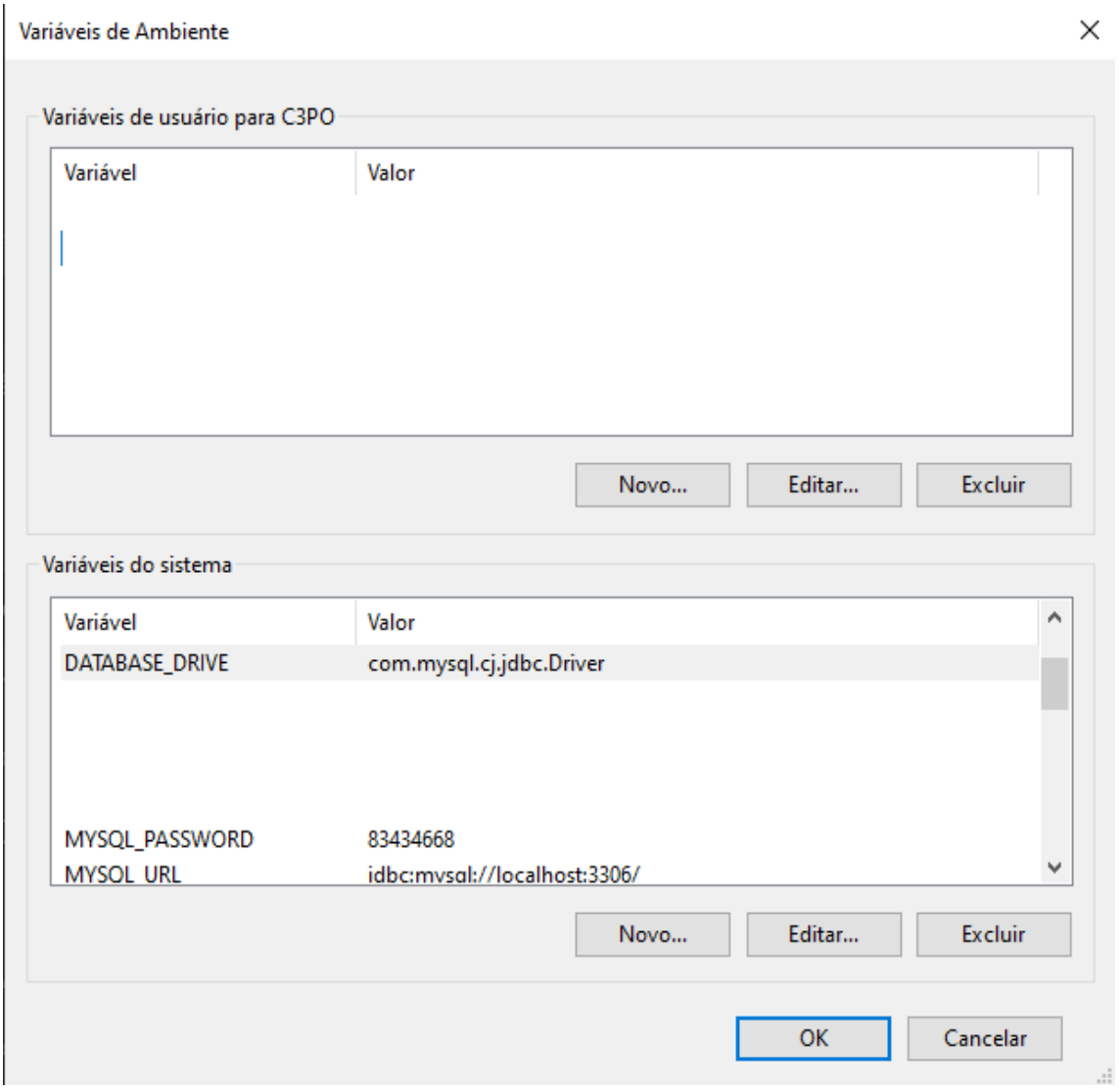
- **Mysql_Username**

Nome do usuário do banco com acesso de leitura.

- **Mysql_Password**

Senha do usuário definido em Mysql_Username.

Figura 4: Variáveis de ambiente do sistema no windows



Após a importação do projeto no IDE é preciso gerar o JAR executável para que possa ser dado deploy do serviço.

4.3. Serviço de descoberta

Após o projeto ser importado na IDE é preciso gerar o JAR executável para que possa ser dado deploy do serviço.

4.4. Api-gateway, serviço produto, inventário e pedidos

Após a instalação dos bancos, Keycloak, serviço de configuração e serviço de descoberta, todos os demais serviços só precisam ter seus JARs executáveis gerados para que possam ser iniciados, sem ordem pré definida.

4.5. Endpoints

Endpoints são segmentos de rota http nos quais um servidor web, neste caso a API de apoio ao e-commerce, disponibiliza para seus clientes acesso aos seus recursos.

Todos os endpoints a seguir, exceto /produtos (GET), esperam um Bearer token para autorização no header da requisição. O token deve estar em uma chave no header nomeada como “Authorization”. Caso não haja esta chave no header da requisição ou seu valor seja inválido, o acesso ao endpoint será negado. Dito isso, a API de apoio ao e-commerce disponibiliza ao seguintes endpoints para cada microsserviço:

4.5.1. Serviço de Produtos

/produtos:

- Método GET:
    - O endpoint /produtos no método GET retorna uma lista de todos os produtos que estão ativos, paginados. Ele aceita os parâmetros de rota, page e size. Sendo page para definir qual página deseja e size para definir quantos produtos tem a página. Ambos os parâmetros aceitam valores numéricos e tem valor 0 e 5, respectivamente, por padrão. Este endpoint não espera nenhum valor no corpo da requisição. O retorno deste endpoint estará no formato:
    - Formato de retorno:

```
{
  "produtos": {
    "_embedded": {
      "produtoDataList": [
        {
```

```
        "id": "string",
        "nome": "string",
        "descricao": "string",
        "preco": 0,
        "tipo": [
          "string"
        ],
        "imageUrl": "string",
        "ativo": true,
      }
    ]
  },
  "paginaAtual": 0,
  "totalPaginas": 0,
  "totalItems": 0
}
```


- Método PUT:
  - O endpoint /produtos no método PUT atualiza as propriedades de um produto previamente salvo e retorna o produto atualizado. O endpoint requer que no corpo da requisição haja um produto com um id existente e as propriedades que devem ser alteradas e retorna o produto com as propriedades alteradas. Lembrando que apenas é necessário enviar o id e as propriedades que devem ser alteradas, não sendo preciso enviar as que não serão alteradas.
  - Formato do objeto no corpo da requisição:


```
{
  "id": "string",
  "nome": "string",
  "descricao": "string",
  "preco": 0,
  "tipo": [
    "string"
  ],
```

```
  "imageUrl": "string",
  "ativo": true
}
```


- Formato do objeto no corpo da resposta:


```
{
  "id": "string",
  "nome": "string",
  "descricao": "string",
  "preco": 0,
  "tipo": [
    "string"
  ],
  "imageUrl": "string",
  "ativo": true
}
```


- Método POST:
  - O endpoint /produtos no método POST salva um novo produto e retorna este produto com um id. O endpoint requer que no corpo da requisição haja um produto sem um id.
  - Formato do objeto no corpo da requisição:


```
{
  "nome": "string",
  "descricao": "string",
  "preco": 0,
  "tipo": [
    "string"
  ],
  "imageUrl": "string",
  "ativo": true
}
```


  - Formato do objeto no corpo da resposta:


```
{
  "id": "string",
  "nome": "string",
```

```
  "descricao": "string",
  "preco": 0,
  "tipo": [
    "string"
  ],
  "imageUrl": "string",
  "ativo": true
}
```


/produtos/toogle-status-produto/{id}:

- Método PATCH:
    - O endpoint /produtos/toogle-status-produto/{id} no método PATCH atualiza o status de ativo do produto e retorna o produto atualizado. Se estiver ativo, ele passa a ser inativo e vice versa. Esse status define ser um produto está disponível para venda, apenas produtos ativos estão disponíveis. Este endpoint não requer um objeto no corpo da requisição, mas necessita que o id do produto a ser alterado seja passado junto a url (no lugar do {id}) no momento da requisição.
    - Formato do objeto no corpo da resposta:


```
{
  "id": "string",
  "nome": "string",
  "descricao": "string",
  "preco": 0,
  "tipo": [
    "string"
  ],
  "imageUrl": "string",
  "ativo": true
}
```


/produtos/{id}:

- Método GET:
    - O endpoint /produtos/{id} no método GET retorna um produto baseado no id recebido. Este endpoint não requer um objeto no

corpo da requisição, mas necessita que o id do produto a ser retornado seja passado junto a url (no lugar do {id}) no momento da requisição.

- Formato do objeto no corpo da resposta:

```
{
  "id": "string",
  "nome": "string",
  "descricao": "string",
  "preco": 0,
  "tipo": [
    "string"
  ],
  "imageUrl": "string",
  "ativo": true
}
```

/produtos/admin:

- Método GET:
  - endpoint /produtos/admin no método GET retorna uma lista de todos os produtos, tanto ativos quanto inativos. Este endpoint não espera nenhum valor no corpo da requisição. O retorno deste endpoint estará no formato:
  - Formato do objeto no corpo da resposta:

```
{
    "_embedded": {
     "produtoDataList": [
       {
         "id": "string",
         "nome": "string",
         "descricao": "string",
         "preco": 0,
         "tipo": [
           "string"
         ],
```

```
        "imageUrl": "string",
        "ativo": true,
      }
    ]
  },
}
```


4.5.2. Serviço de Inventário

/inventario

- Método PUT:
  - O endpoint /inventario no método PUT atualiza um item de inventário e retorna este item atualizado. Este endpoint requer que haja um item de inventário no corpo da requisição.
  - Formato do objeto no corpo da requisição:


```
{
  "id": 0,
  "skuCode": "string",
  "quantity": 0
}
```


  - Formato do objeto no corpo da resposta:


```
{
  "id": 0,
  "skuCode": "string",
  "quantity": 0
}
```


- Método POST:
  - O endpoint /inventario no método POST salva um item de inventário e retorna o item de inventário salvo. Um item de inventário está sempre relacionado a um produto (O id do produto deve estar na chave “skuCode” do item de inventário). Este endpoint requer que no corpo da requisição haja um item de inventário, sem o id.
  - Formato do objeto no corpo da requisição:

```
{
  "skuCode": "string",
  "quantity": 0
}
```

- Formato do objeto no corpo da resposta:

```
{
  "id": 0,
  "skuCode": "string",
  "quantity": 0
}
```

/inventario/admin

- Método GET:
  - O endpoint /inventario/admin no método GET retorna todos os itens de inventário salvos. Este endpoint não requer um objeto no corpo da requisição.
  - Formato do objeto no corpo da resposta:

```
{
  "_embedded": {
    "inventarioDataList": [
      {
        "id": 0,
        "skuCode": "string",
        "quantity": 0,
      }
    ]
  },
}
```

4.5.3. Serviço de pedidos

/pedidos

- Método GET:

- O endpoint /pedidos no método GET retorna todos os pedidos salvos. Este endpoint não requer um objeto no corpo da requisição.
- Formato do objeto no corpo da resposta:


```
{
  "_embedded": {
    "pedidoDataList": [
      {
        "id": 0,
        "pedidoNumero": "string",
        "linhaItemPedidoDataList": {
          "_embedded": {
            "linhaItemPedidoDataList": [
              {
                "codigoSku": "string",
                "preco": 0,
                "quantidade": 0,
              }
            ]
          },
        },
        "createdAt": "2023-12-02T05:36:56.920Z",
      }
    ]
  },
}
```


- Método POST:
  - O endpoint /pedidos no método POST salva um pedido no banco de dados. Ele requer um pedido no corpo da requisição e retorna este pedido com o id criado no momento de salvamento. O *array* “linhaItemPedidoDataList” é a lista de produtos contidos no pedido.
  - Formato do objeto no corpo da requisição:


```
{
```

```
  "pedidoNumero": "string",
  "linhaItemPedidoDataList": {
    "_embedded": {
      "linhaItemPedidoDataList": [
        {
          "codigoSku": "string",
          "preco": 0,
          "quantidade": 0,
        }
      ]
    },
  },
}
```


- Formato do objeto no corpo da resposta:


```
{
  "id": 0,
  "pedidoNumero": "string",
  "linhaItemPedidoDataList": {
    "_embedded": {
      "linhaItemPedidoDataList": [
        {
          "codigoSku": "string",
          "preco": 0,
          "quantidade": 0,
        }
      ]
    },
  },
  "createdAt": "2023-12-02T05:43:37.829Z",
}
```


/pedidos/detalhe/{id}

- Método GET:
  - O endpoint /pedidos/detalhe/{id} no método GET retorna um pedido baseado no id recebido. Este endpoint não requer um

objeto no corpo da requisição, mas necessita que o id do pedido a ser retornado seja passado junto a url (no lugar do {id}) no momento da requisição.

- Formato do objeto no corpo da resposta:


```
{
  "id": 0,
  "pedidoNumero": "string",
  "createdAt": "2023-12-02T05:53:31.155Z",
  "linhaItemPedidoList": [
    {
      "codigoSku": "string",
      "preco": 0,
      "quantidade": 0,
      "pedido": "string"
    }
  ]
}
```


Além desta documentação, após todos os serviços tiverem sido postos em produção, a API de apoio ao e-commerce gera uma documentação com mais detalhes sobre os endpoints disponíveis. Esta documentação fica disponível no endereço: http://{domínio de hospedagem}/webjars/swagger-ui/index.html.

Capítulo 5: Teste de Uso

Para exemplificar um uso da API de apoio ao e-commerce, foi desenvolvido um sistema de e-commerce para uma cafeteria fictícia que faz entregas de diversos tipos de cafés. A cafeteria foi nomeada como Coffee Delivery.

Este e-commerce é uma aplicação Front End, ou cliente, desenvolvida utilizando a biblioteca React[6], que utiliza o servidor API de apoio ao e-commerce para executar os casos de uso apresentados no capítulo dois. Como esta aplicação é um exemplo e não o foco deste trabalho, os detalhes acerca de seu desenvolvimento não serão aprofundados. Porém, ela está disponível no endereço: https://github.com/pedroAndrad1/rocketseat_ignite_coffee_delivery/tree/keycloack, na branch nomeada como “keycloak”. Ressaltamos que a aplicação Coffee Delivery não é o produto deste trabalho, sendo apenas uma aplicação para demonstrar como a API de apoio ao e-commerce será usada.

5.1. Execução do caso de uso número um, Realizar autenticação

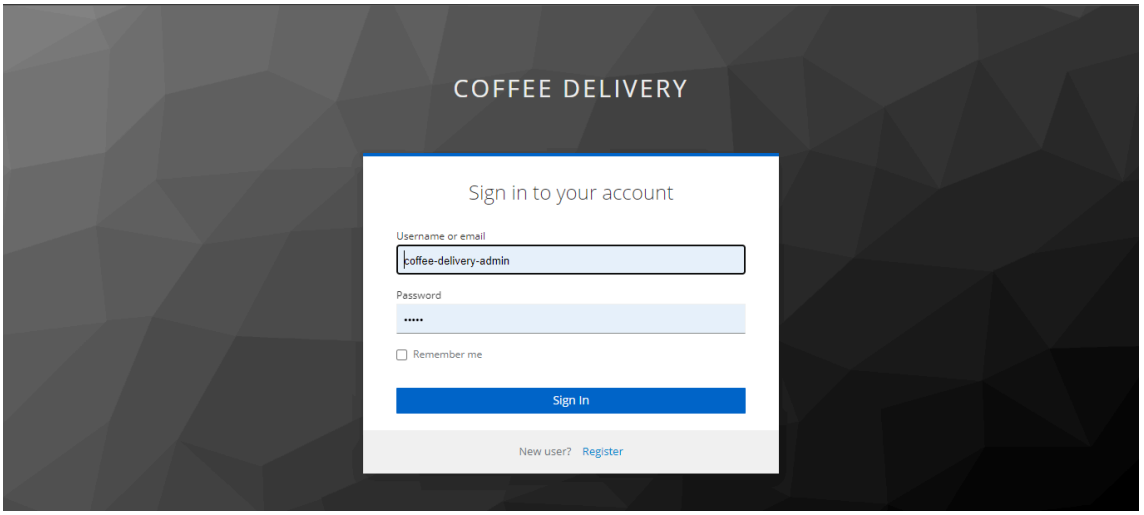
O usuário acessa a página Home do e-commerce Coffee Delivery (figura 5) e clica no botão de login, localizado no canto superior direito:

Figura 5: Homepage (usuário não logado)

[6] *React é um framework JavaScript criado pelo Facebook (atual Meta) que é usado para criar interfaces de usuário (UI) em aplicativos web. Ele é popular por ser fácil de usar, altamente flexível e escalável, e é usado por muitas empresas de tecnologia, incluindo o Facebook, Instagram e Airbnb (NEVES, 2023).*

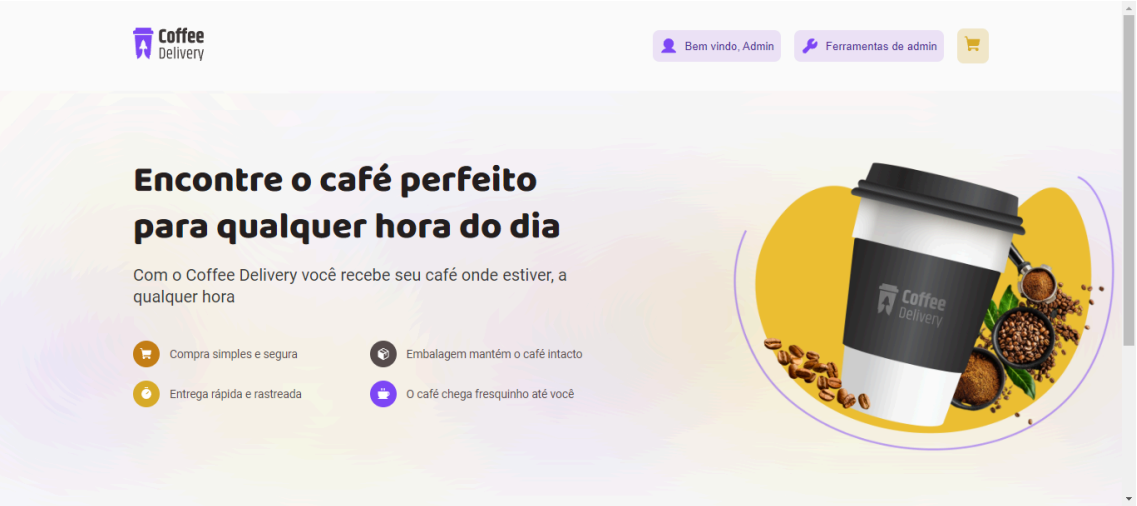
A aplicação Front End o redireciona para o KeyCloak (figura 6) e o usuário informa credenciais de administrador:

Figura 6: Tela de login do Keycloak



O KeyCloak redireciona o usuário de volta para a página Home (figura 7), enviando junto um token para ser utilizado para acessar os recursos da API de apoio ao e-commerce:

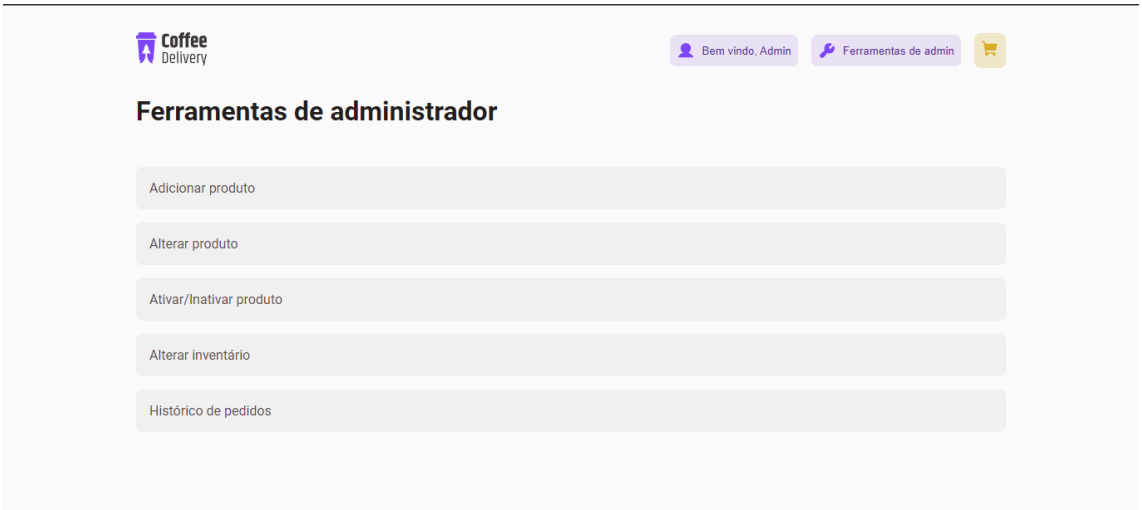
Figura 7: Home page (usuário logado)



5.2. Execução do caso de uso número dois, Registrar produto, e execução do caso de uso número doze, Adicionar produto ao inventário

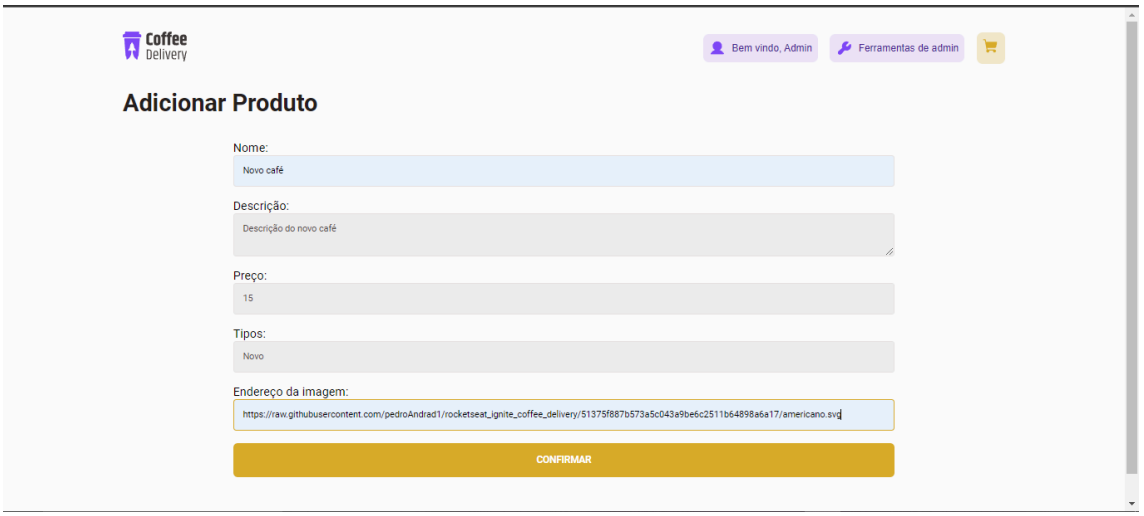
Na página de ferramentas de administrador (figura 8), o usuário seleciona a opção “Adicionar produto”:

Figura 8: Menu de ferramentas de administrador



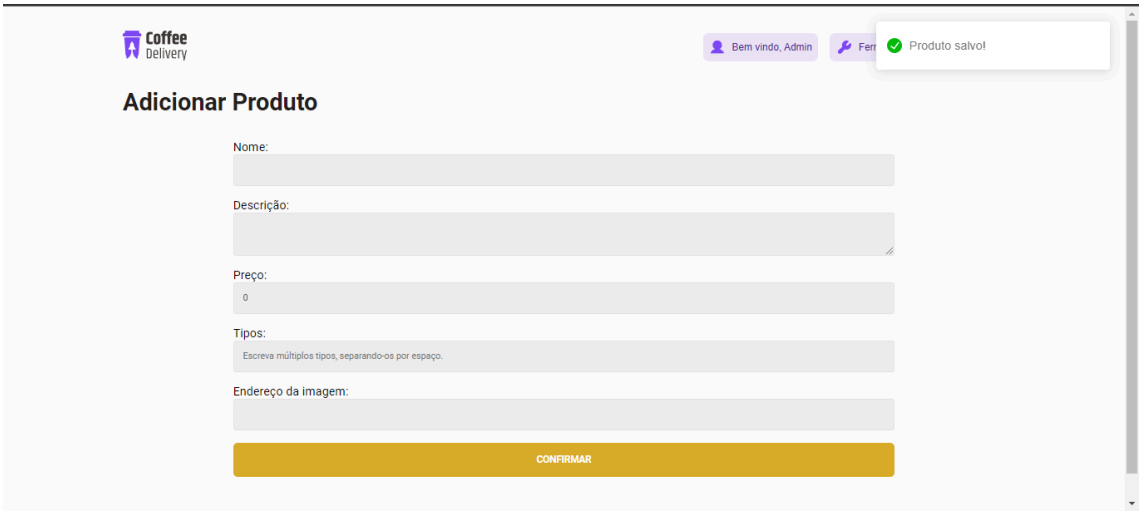
O usuário preenche o formulário de novo produto (figura 9) e clica em “Confirmar”:

Figura 9: Formulário para adição de produto (preenchido)



Os campos do formulário voltam ao estado inicial (figura 10) e é apresentada a resposta da API de apoio ao e-commerce:

Figura 10: Formulário para adição de produto (vazio)



Mensagens enviadas para a API de apoio ao e-commerce no momento do registro:

Figura 11: Mensagem para o endpoint /produtos POST, para salvar um novo produto.

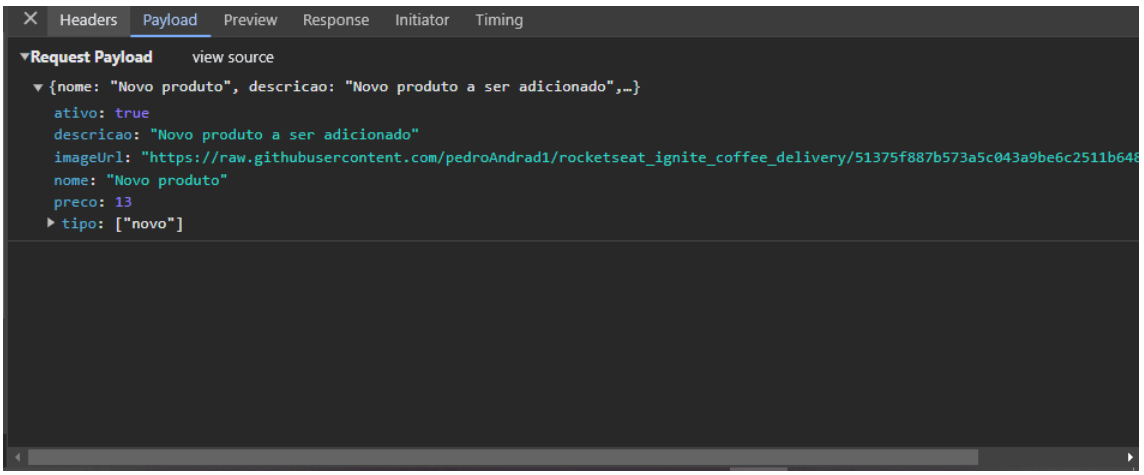
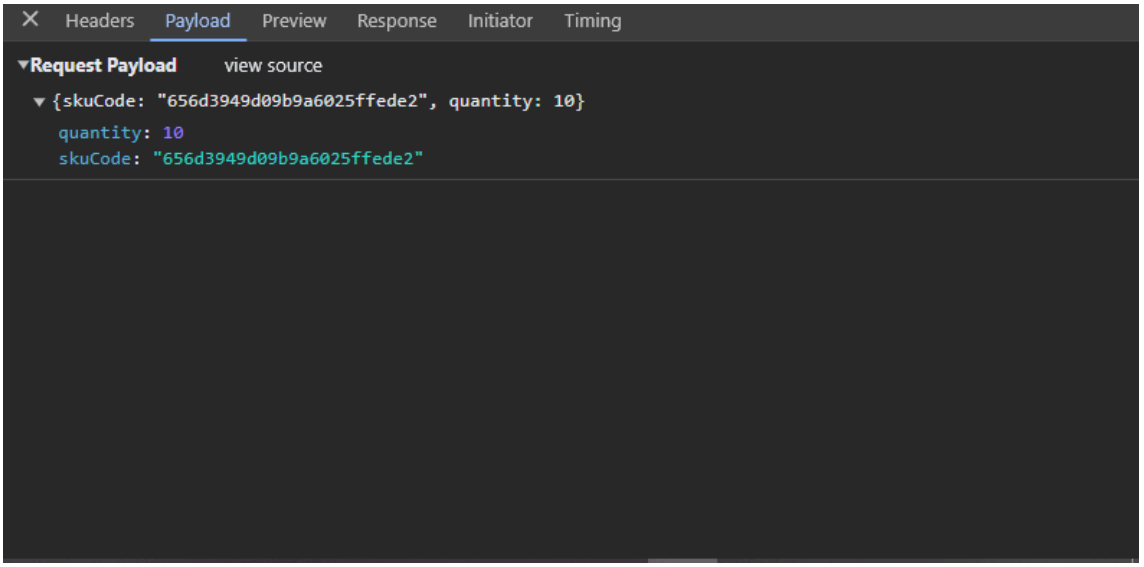


Figura 12: Mensagem para o endpoint /inventario POST, para salvar um novo item no inventário.



Respostas da API de apoio ao e-commerce:

Figura 13: Resposta do endpoint /produtos POST, para salvar um novo produto.

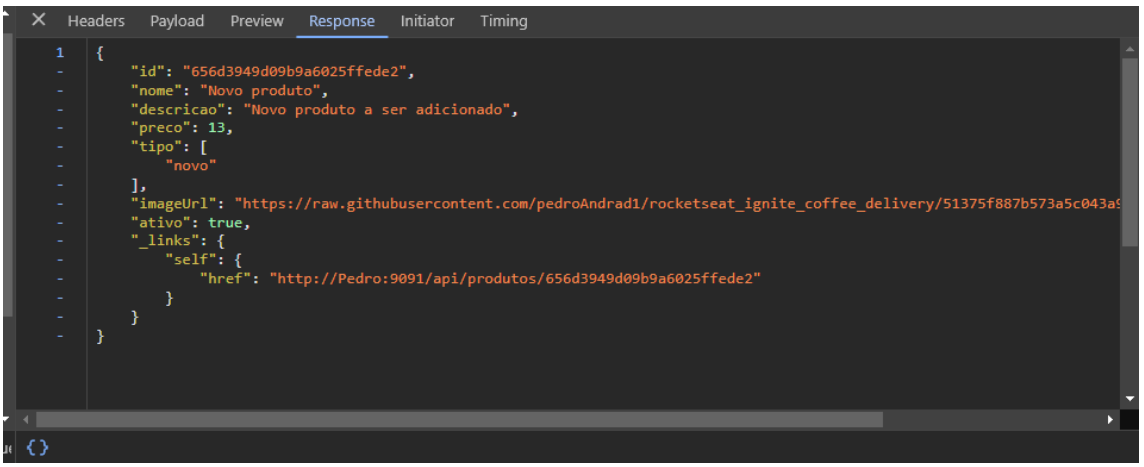
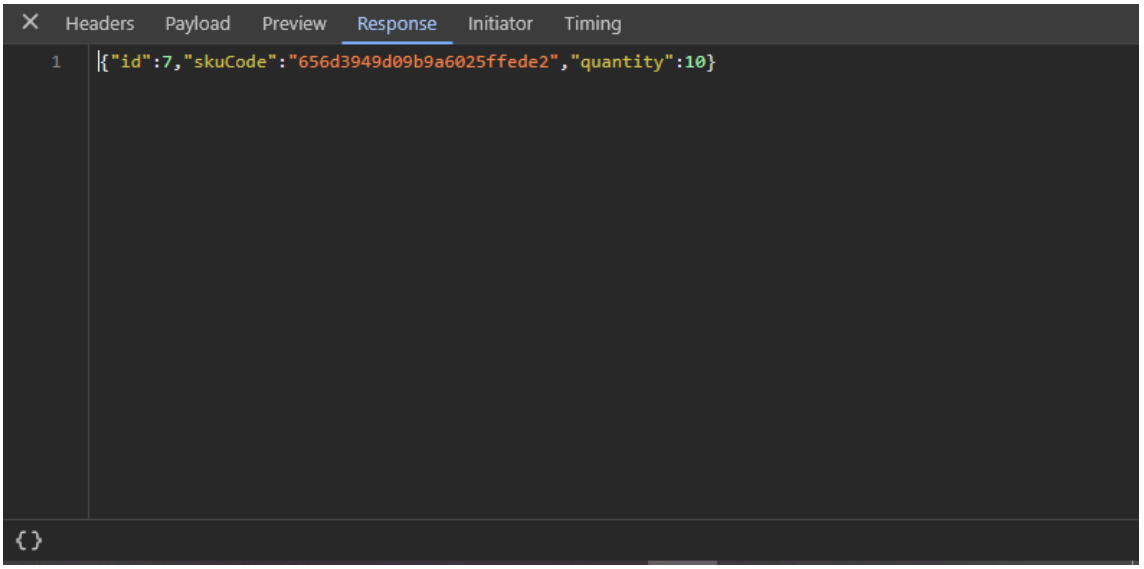


Figura 14: Resposta do endpoint /inventario POST, para salvar um novo item no inventário.



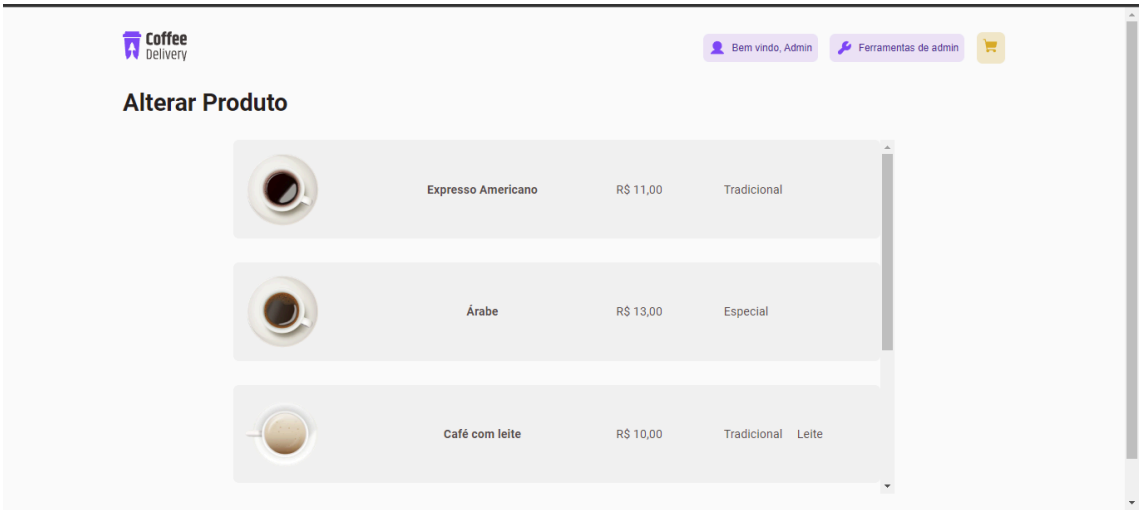
5.3. Execução do caso de uso número três, Alterar produto

Na página de ferramentas de administrador (figura 15), o usuário seleciona a opção “Alterar produto”:

Figura 15: Menu de ferramentas de administrador

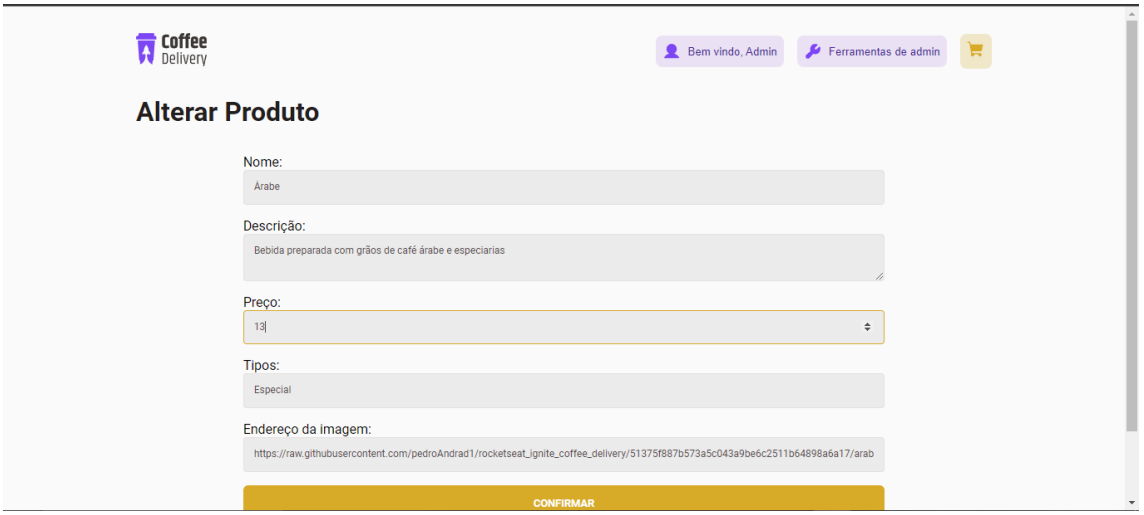
O usuário seleciona o produto a ser alterado:

Figura 16: Menu para seleção de produto a ser alterado



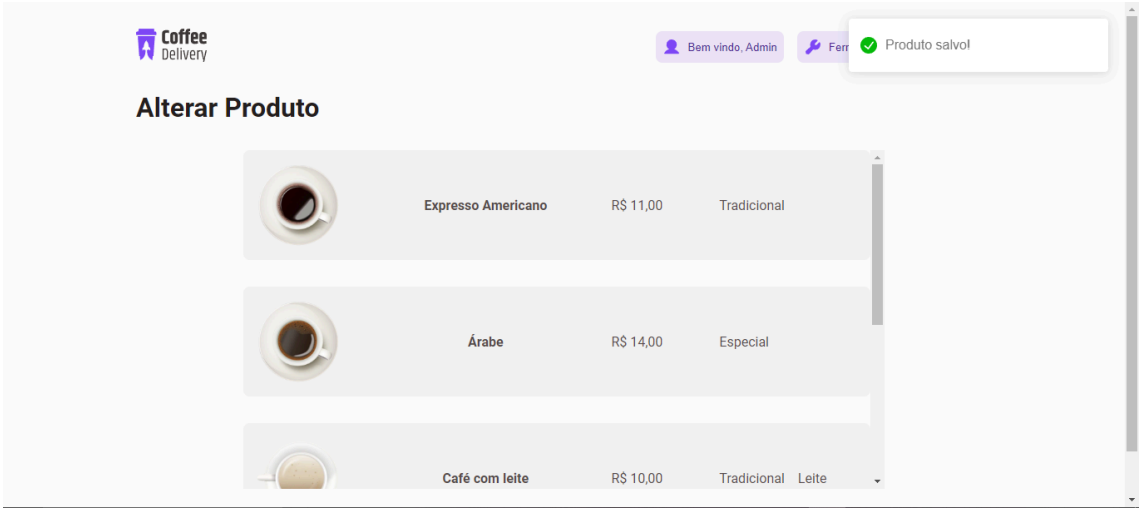
Os dados do produto são carregados em um formulário (figura 17), o usuário altera os dados que deseja no formulário e clicar em “Confirmar”:

Figura 17: formulário para alteração de produto (preenchido)



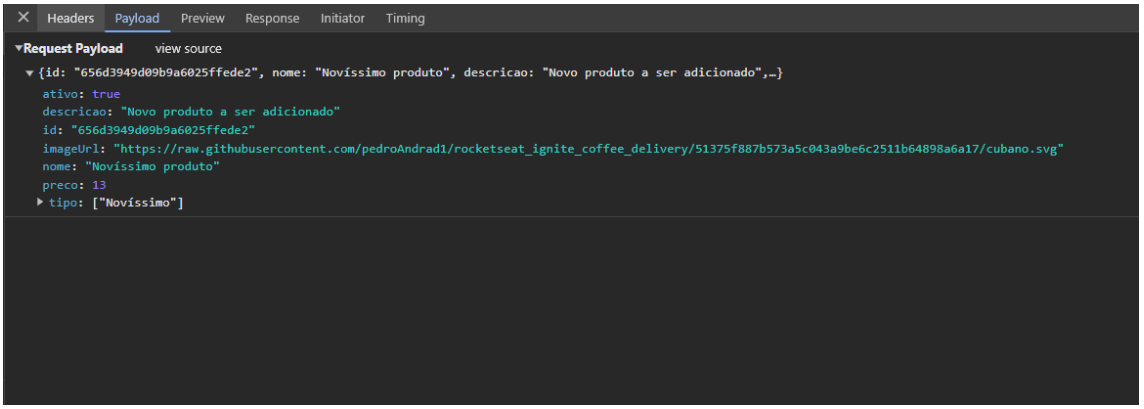
O usuário é redirecionado novamente para a lista de produtos (figura 18) a serem alterados e recebe uma notificação com a resposta da API de apoio ao e-commerce

Figura 18: Menu para seleção de produto a ser alterado, com mensagem de sucesso



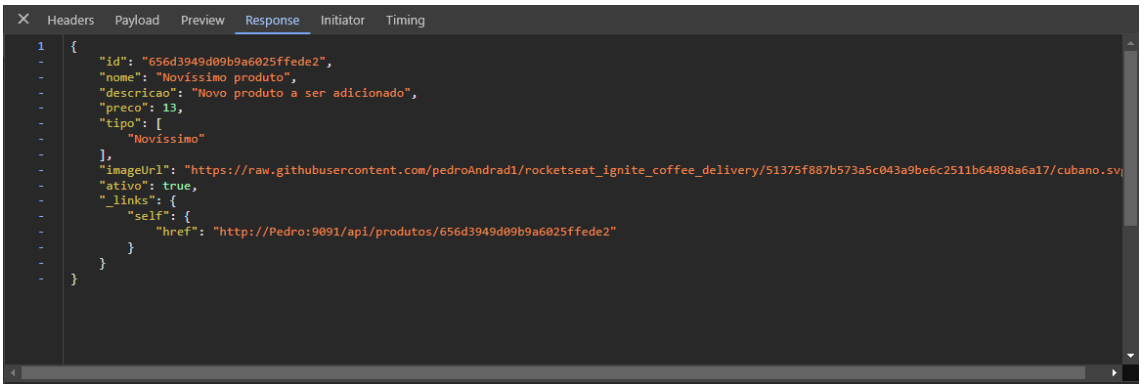
Mensagem enviada para a API de apoio ao e-commerce no momento da alteração:

Figura 19: Mensagem para o endpoint /produtos PUT, para atualizar um produto.



Resposta da API de apoio ao e-commerce no momento da alteração:

Figura 20: Resposta para o endpoint /produtos PUT, para atualizar um produto.



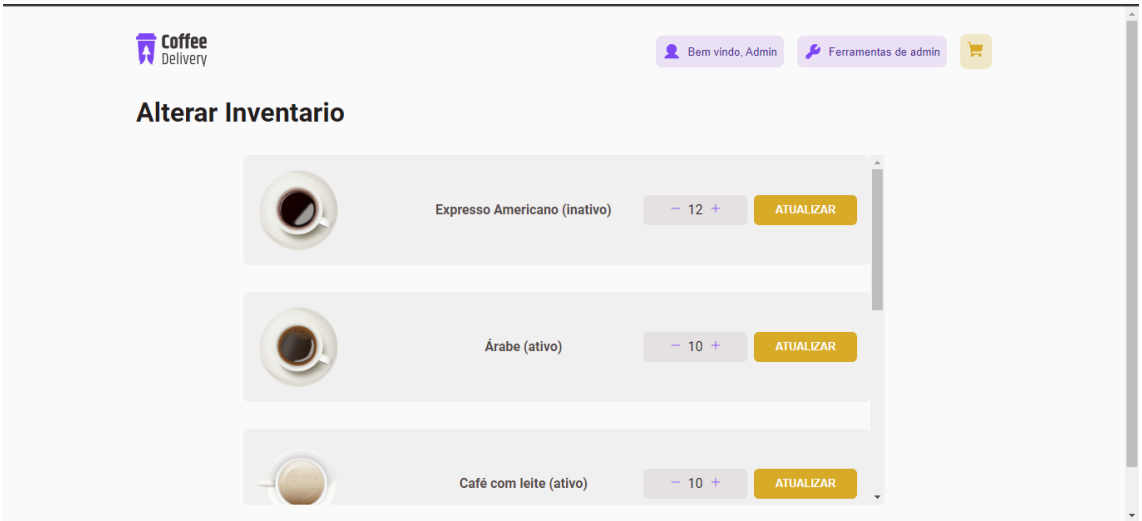
5.4. Execução do caso de uso número quatro, Listar inventário

Na página de ferramentas de administrador (figura 21), o usuário seleciona a opção “Alterar inventário”:

Figura 21: Menu de ferramentas de administrador

São listados todos os produtos contidos no inventário e suas quantidades:

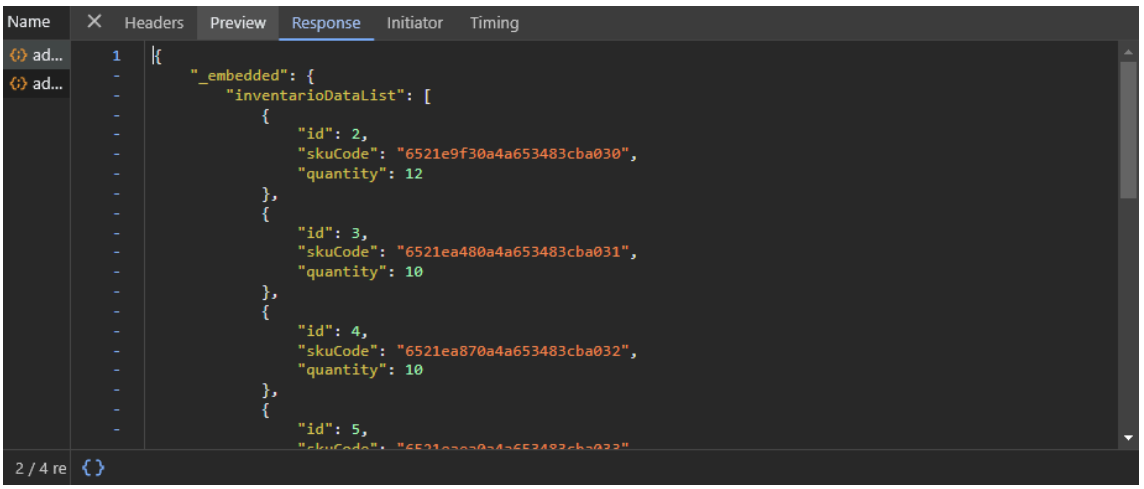
Figura 22: Menu para alteração de inventário



Mensagem enviada para a API de apoio ao e-commerce no momento da listagem: requisição para GET /inventario/admin, sem objeto no corpo da requisição.

Resposta da API de apoio ao e-commerce:

Figura 23: Resposta para o endpoint /inventario/admin GET, para listar inventário.

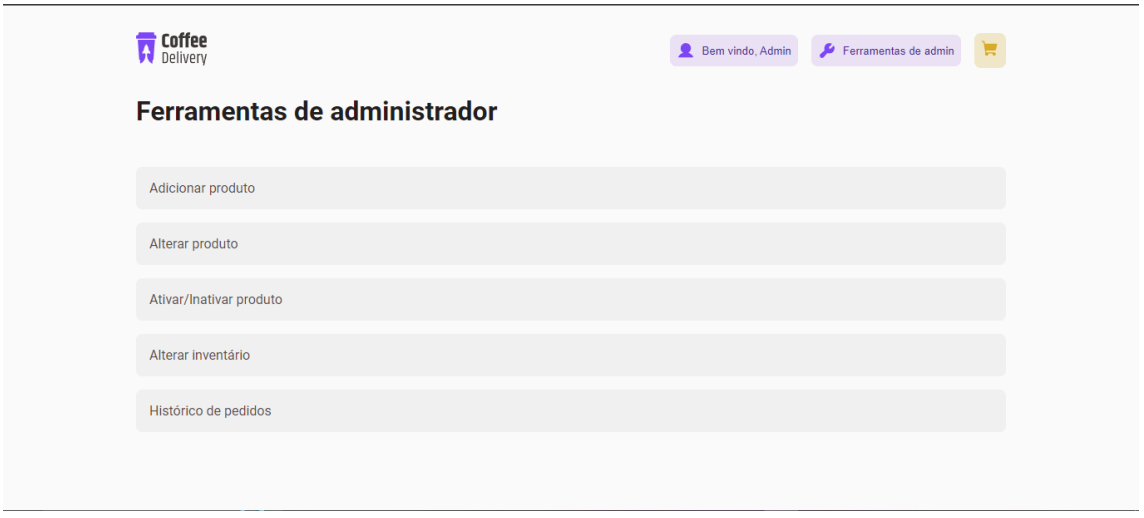


Obs: Na tela de listagem (figura 22) também foi feita uma requisição para /produtos/admin, a fim de conseguir a foto e o status do item dos produtos, relacionados aos itens de inventário, para mostrar em tela.

5.5. Execução do caso de uso número cinco, Inativar produto

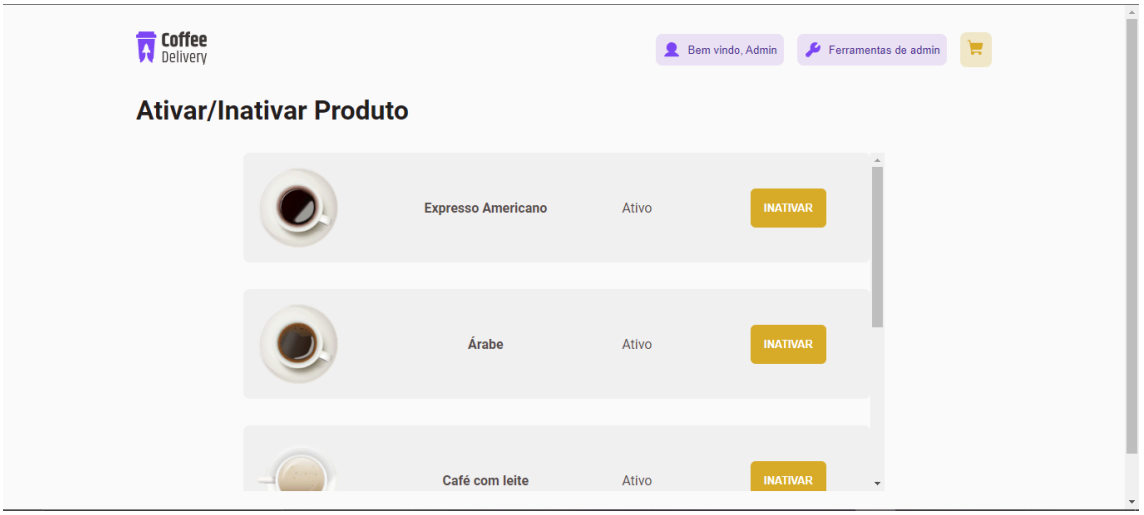
Na página de ferramentas de administrador (figura 24), o usuário seleciona a opção “Ativar/inativar produto”:

Figura 24: Menu de ferramentas de administrador



São listados todos os produtos e seus status:

Figura 25: Menu para ativação e inativação de produtos (com produtos ativos)

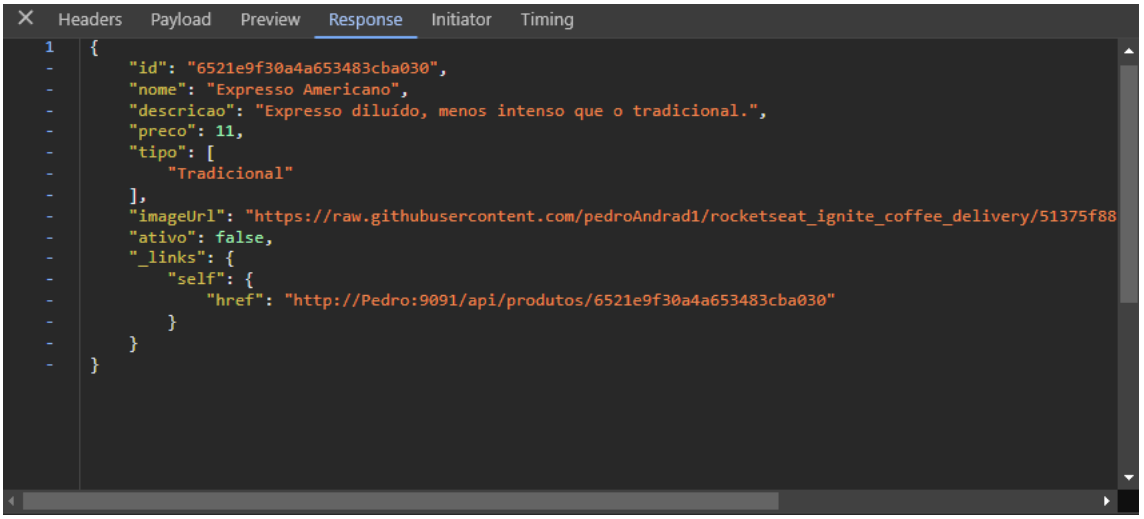


O usuário clica no botão de inativar para inativar o produto que deseja.

Mensagem enviada para a API de apoio ao e-commerce no momento do clique do botão: requisição para PATCH produtos/toggle-status-produto/6521e9f30a4a653483cba030, sem objeto no corpo da requisição.

Resposta da API de apoio ao e-commerce:

Figura 26: Resposta para o endpoint produtos/toggle-status-produto/6521e9f30a4a653483cba030 PATCH

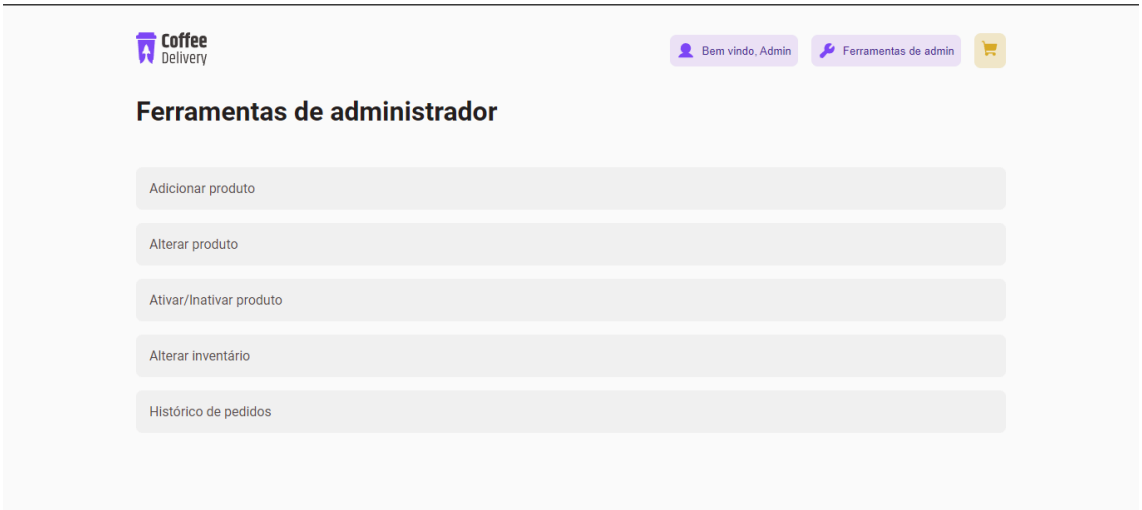


Obs: Na tela de listagem (figura 25) também foi feita uma requisição para /produtos/admin, a fim de conseguir a foto e o status do item dos produtos para mostrar em tela.

5.6. Execução do caso de uso número seis, Ativar produto

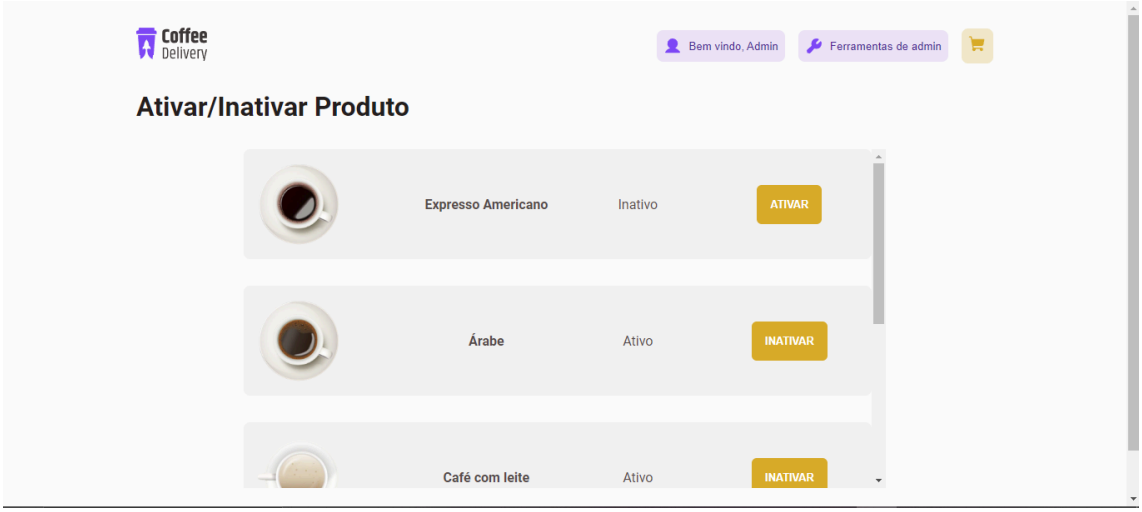
Na página de ferramentas de administrador (figura 27), o usuário seleciona a opção “Ativar/inativar produto”:

Figura 27: Menu de ferramentas de administrador



São listados todos os produtos e seus status:

Figura 28: Menu para ativação e inativação de produtos (com um produto inativo)

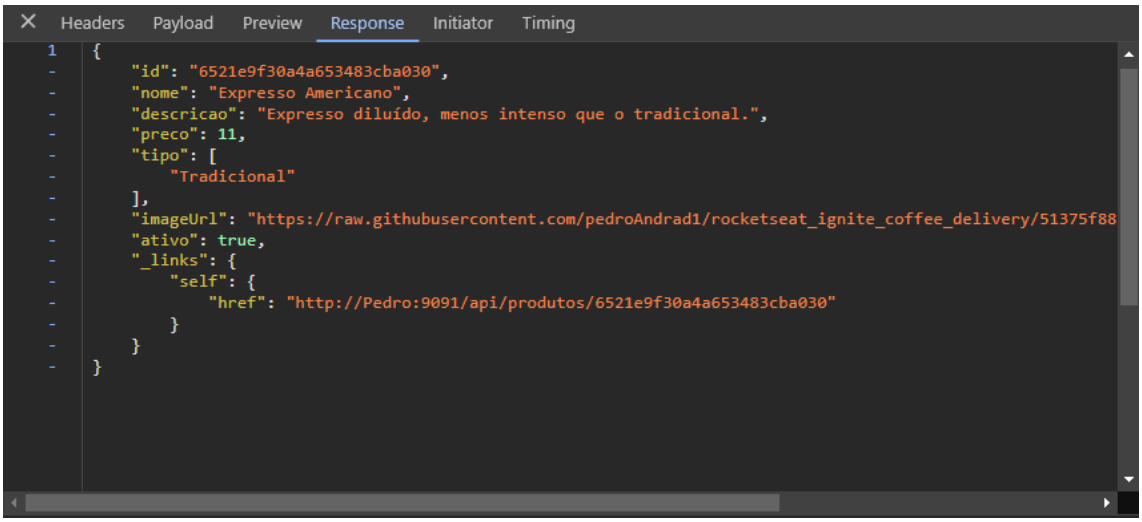


O usuário clica no botão de ativar para ativar o produto que deseja.

Mensagem enviada para a API de apoio ao e-commerce no momento do clique do botão: requisição para PATCH produtos/toggle-status-produto/6521e9f30a4a653483cba030, sem objeto no corpo da requisição.

Resposta da API de apoio ao e-commerce:

Figura 29: Resposta para o endpoint produtos/toggle-status-produto/6521e9f30a4a653483cba030 PATCH

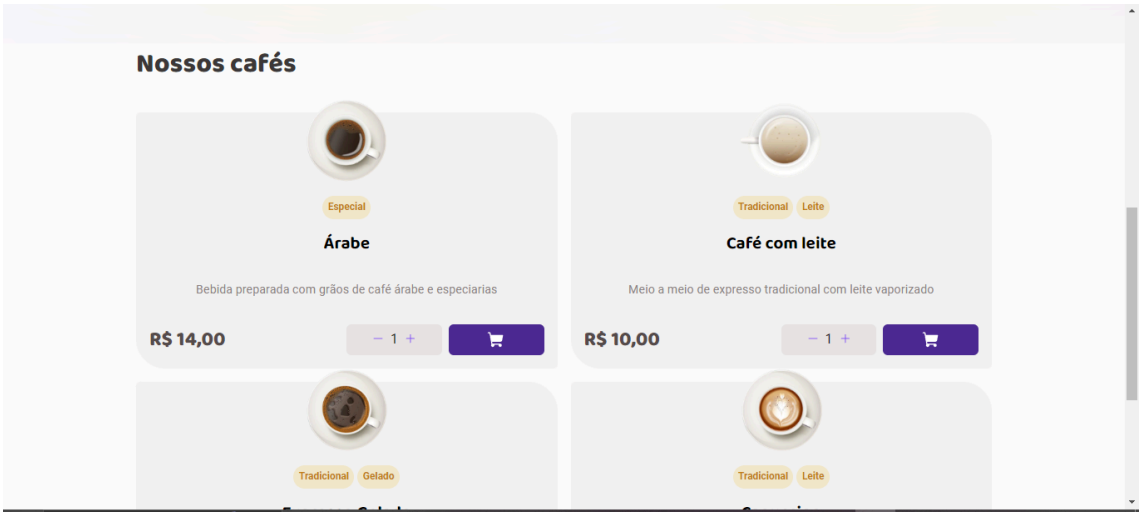


Obs: Na tela de listagem (figura 28) também foi feita uma requisição para /produtos/admin, a fim de conseguir a foto e o status do item dos produtos para mostrar em tela.

5.7. Execução do caso de uso número sete, Registrar pedido de venda

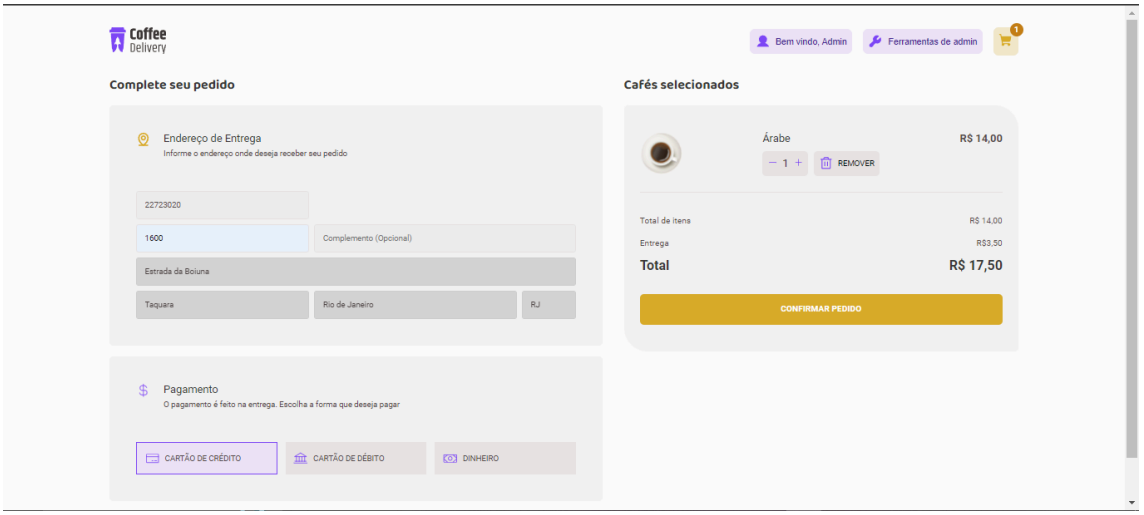
O usuário, já tendo realizado login (figura 30), seleciona na home quais produtos deseja comprar:

Figura 30: Lista de produtos da página Home



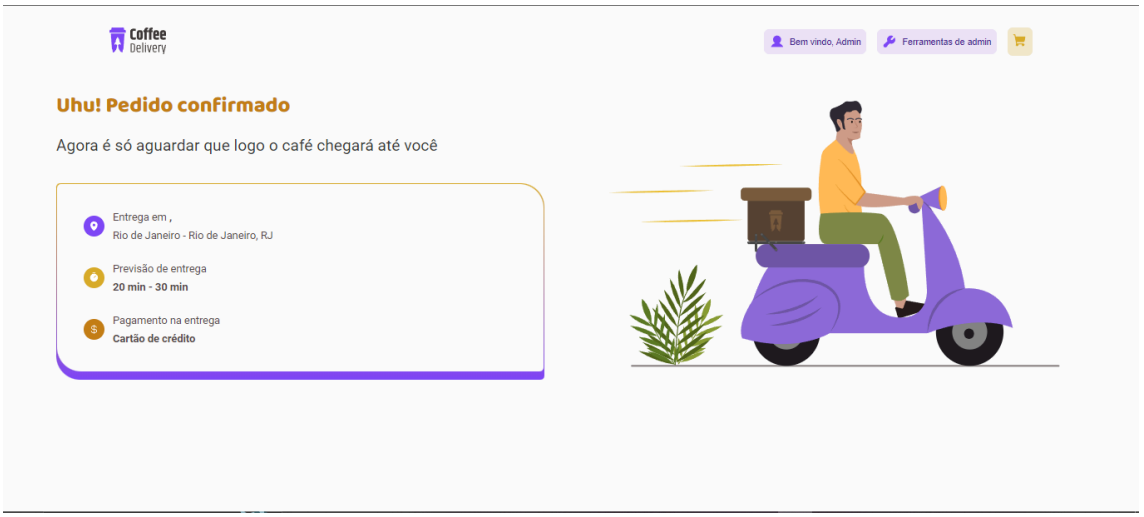
O usuário clica no ícone carrinho na canto superior direito da home e se direciona a página de carrinho:

Figura 31: Página de carrinho



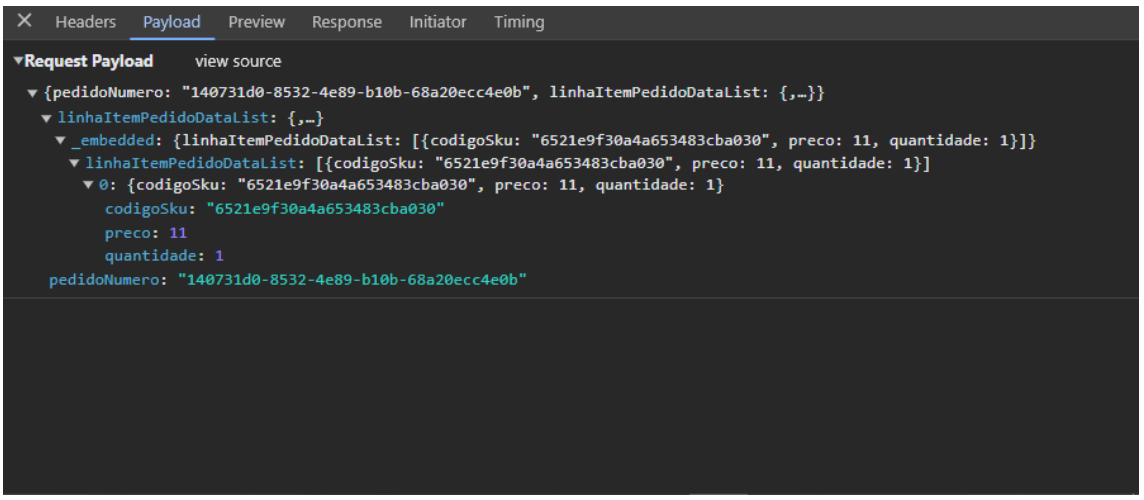
Na página de carrinho (figura 31), o usuário preenche o formulário com endereço para entrega, seleciona a forma de pagamento e clica em “confirmar o pedido” (As informações de endereço e forma de pagamento não são enviadas para a API de apoio ao e-commerce). Após a confirmação do pedido, o usuário é redirecionado para tela de confirmação (figura 32):

Figura 32: Página de pedido efetuado



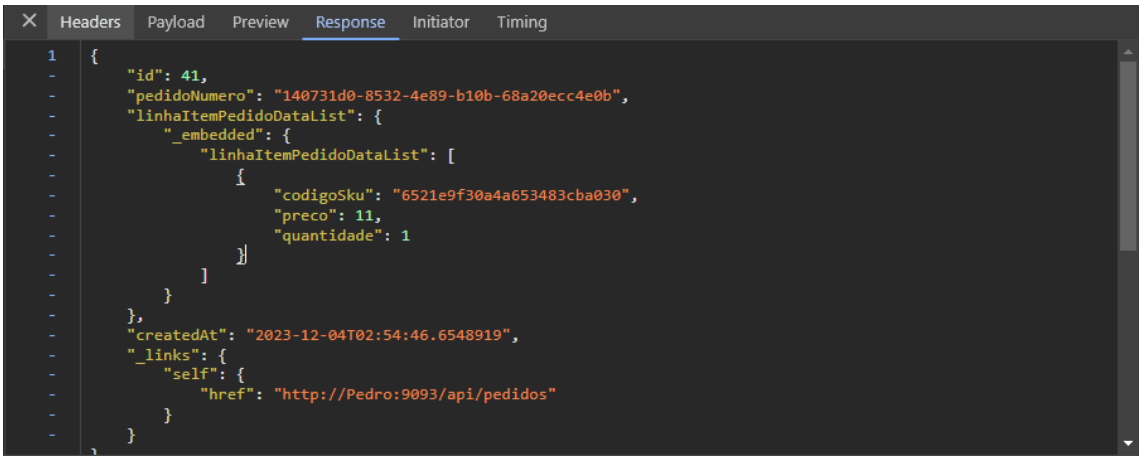
Mensagem enviada para a API de apoio ao e-commerce no momento do clique do botão:

Figura 33: Mensagem enviada para o endpoint /pedidos POST



Resposta da API de apoio ao e-commerce:

Figura 34: Resposta para o endpoint /pedidos POST



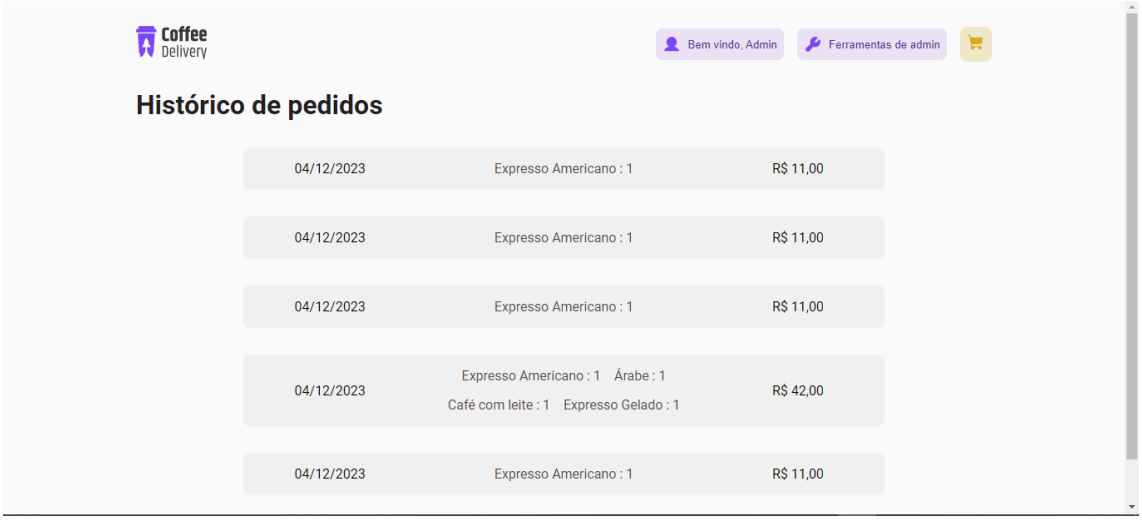
5.8. Execução do caso de uso número oito, Listar pedidos de venda

Na página de ferramentas de administrador (figura 35), o usuário seleciona a opção “Histórico de pedidos”:

Figura 35: Menu de ferramentas de administrador

O usuário é redirecionado para a página de histórico de pedidos (figura 36) e são listados todos os pedidos com id, produtos do pedido e valor total:

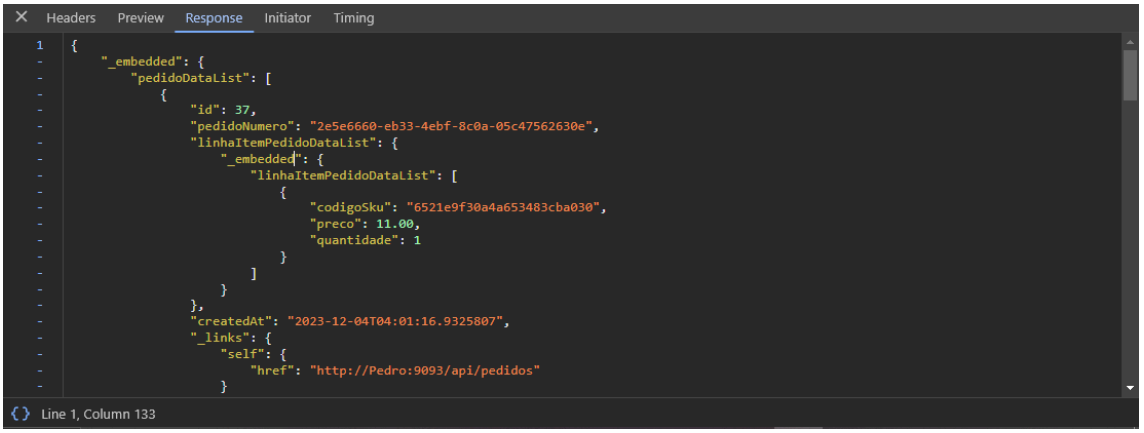
Figura 36: página de histórico de pedidos



Mensagem enviada para a API de apoio ao e-commerce no momento do clique do botão: requisição para GET /pedidos, sem objeto no corpo da requisição.

Resposta da API de apoio ao e-commerce:

Figura 37: Resposta para o endpoint /pedidos GET



Obs: Na tela de listagem (figura 36) também foi feita uma requisição para /produtos/admin, a fim de conseguir o nome dos produtos para mostrar em tela.

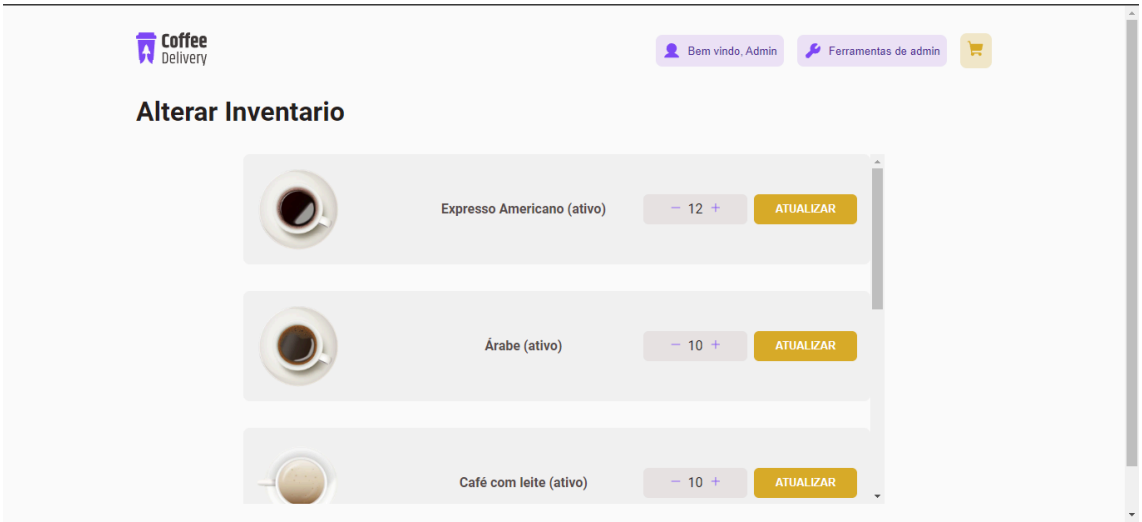
5.9. Execução do caso de uso número nove, Alterar inventário

Na página de ferramentas de administrador (figura 38), o usuário seleciona a opção “Alterar inventário”:

Figura 38: Menu de ferramentas de administrador

São listados todos os produtos contidos no inventário e suas quantidades:

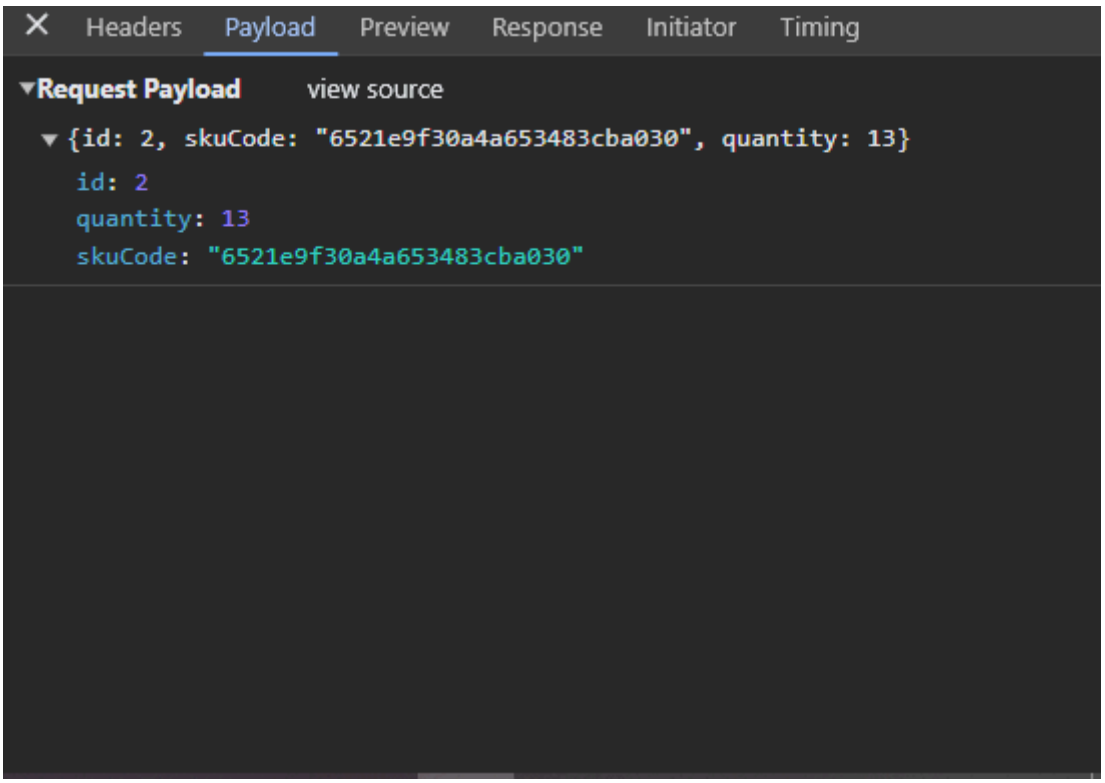
Figura 39: Menu para alteração de inventário



O usuário altera a quantidade do produto no inventário e clica em atualizar para atualizar o inventário.

Mensagem enviada para a API de apoio ao e-commerce no momento do clique do botão:

Figura 40: Mensagem enviada para o endpoint /inventario PUT



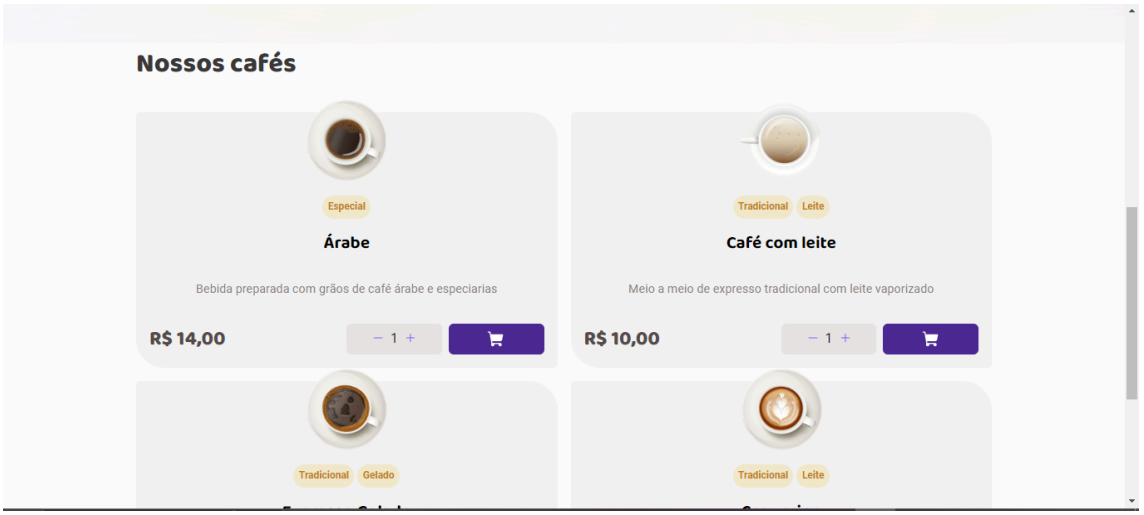
Resposta da API de apoio ao e-commerce:

Figura 41:Resposta para o endpoint /inventário PUT

5.10. Execução do caso de uso número dez, Listar produtos ativos

O usuário acessa a página Home da aplicação (figura 42) e nela estão listados todos os produtos ativos, ou seja, disponíveis para venda:

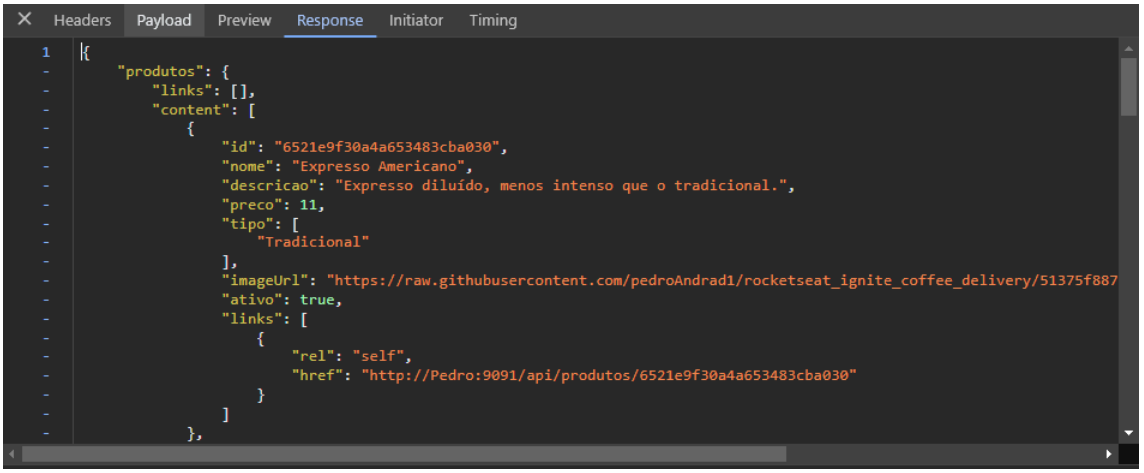
Figura 42: Lista de produtos da página Home



Mensagem enviada para a API de apoio ao e-commerce no momento da listagem: requisição para GET /produtos?size=99999, sem objeto no corpo da requisição.

Resposta da API de apoio ao e-commerce:

Figura 43: Resposta para o endpoint /produtos?size=99999 GET, para listar todos os produtos ativos.



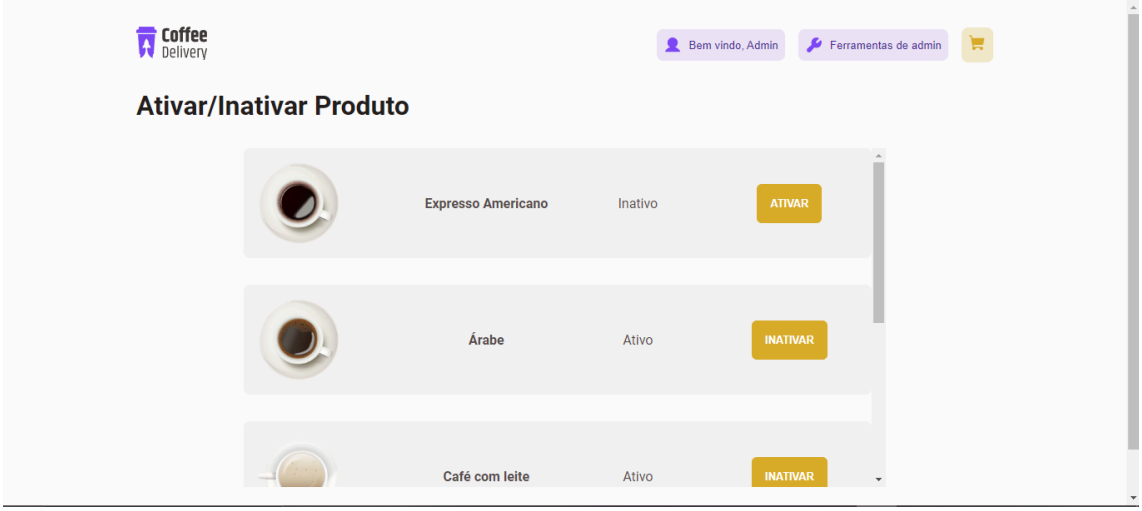
5.11. Execução do caso de uso número onze, Listar todos os produtos

Na página de ferramentas de administrador (figura 44), o usuário seleciona a opção “Ativar/inativar produto”:

Figura 44: Menu de ferramentas de administrador

São listados todos os produtos e seus status:

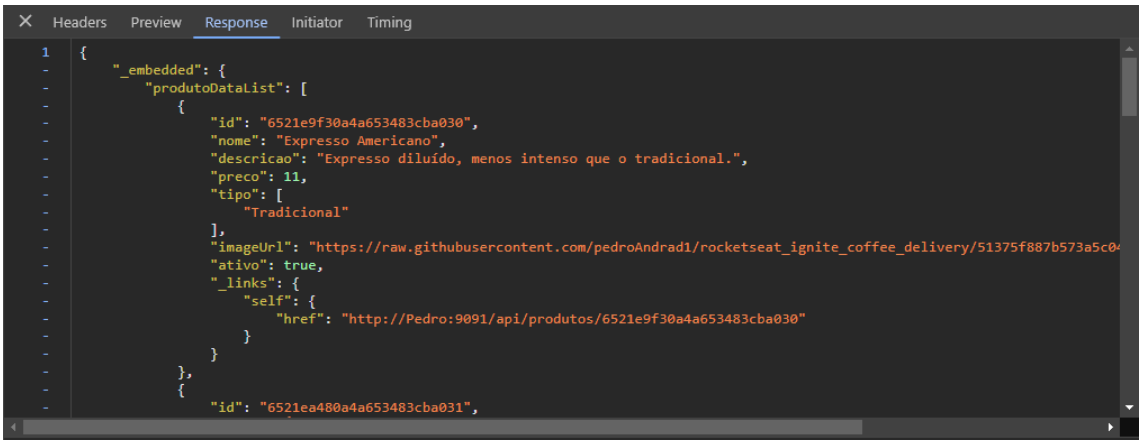
Figura 45: Menu para ativação e inativação de produtos



Mensagem enviada para a API de apoio ao e-commerce no momento da listagem: requisição para GET /produtos/admin, sem objeto no corpo da requisição.

Resposta da API de apoio ao e-commerce:

Figura 46: Resposta para o endpoint /produtos/admin GET, para listar todos os produtos ativos.



Capítulo 6: Conclusão

6.1. Considerações finais

Este trabalho apresenta a API de apoio ao e-commerce, um servidor web, desenvolvido sob a arquitetura de microsserviços, com o propósito de servir diferentes tipos de e-commerce, disponibilizando ações de controle de produtos, inventário e pedidos.

Observando o teste de uso, feito pelo o e-commerce fictício Coffee Delivery, onde ele executa todos casos de uso da API de apoio ao e-commerce, é possível concluir que a API de apoio ao e-commerce supre as necessidades básicas de um e-commerce: controle de produtos, de inventário e de pedidos.

A API de apoio ao e-commerce também é genérica o suficiente para poder ser usada no desenvolvimento de diversos tipos de e-commerce, desde que eles necessitem apenas de ações básicas de controle de produtos, inventário e pedidos. Todavia, é possível realizar alterações na API de apoio ao e-commerce para necessidades mais avançadas e específicas de um determinado e-commerce devido a licença na qual esta é distribuída. Ademais, a arquitetura de microsserviços, na qual a API de apoio ao e-commerce é desenvolvida, possibilita adicionar novas funcionalidades ao sistema de forma a haver um crescimento sustentável da complexidade deste.

6.2. Trabalhos Futuros

Sugerimos como trabalhos futuros a adição de novas funcionalidades ao servidor. Por exemplo, estatísticas acerca do funcionamento do e-commerce servido pela API de apoio ao e-commerce.

Referências Bibliográficas

JÚNIOR, O. A. **Arquitetura de Micro Serviços**: uma Comparação com Sistemas Monolíticos. Orientador: Marcus Williams Aquino de Carvalho. 2017. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Centro de Ciências Aplicadas e Educação, Departamento de Ciência Exatas, Universidade Federal da Paraíba (UFPB), Rio Tinto - PB, 2017. Disponível em: https://repositorio.ufpb.br/jspui/handle/123456789/3235. Acesso em: 11 fev. 2023.

OPUS SOFTWARE. **Micro Serviços**: Qual a diferença para a Arquitetura Monolítica?. São Paulo - SP, 17 mar. 2021. Disponível em: https://www.opus-software.com.br/micro-servicos-arquietura-monolitica/#. Acesso em: 12 fev. 2023.

MOREIRA, P. F. M.; BEDER, D. M. Desenvolvimento de Aplicações e Micro Serviços: Um estudo de caso. **Revista T.I.S.**: Tecnologias Infraestrutura software, São Carlos – SP, ano 2015, v. 4, ed. 3, p. 209-215, 2015. Disponível em: http://revistatis.dc.ufscar.br/index.php/revista/article/view/364. Acesso em: 12 fev. 2023.

CHIARADIA, L.F.C; MACEDO, D.D.J; DUTRA, M.L. Uma Proposta de Arquitetura de Microsserviços Aplicada em um Sistema de CRM Social. **Encontros Bibli**: revista eletrônica de biblioteconomia e ciência da informação, [*s. l.*], v. 23, ed. 53, p. 147-159, 2018. DOI DOI: 10.5007/1518-2924.2018v23n53p147. Disponível em: https://www.redalyc.org/journal/147/14762417014/14762417014.pdf. Acesso em: 13 fev. 2023.

ALMEIDA, V. S. **PLATAFORMA DE E-COMMERCE INTEGRADA A MICRO SERVIÇOS**. Orientador: Prof. Dr. Carlos Frederico M. C. Cavalcanti. 2022. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) -

Universidade Federal de Ouro Preto, Ouro Preto - MG, 2022. Disponível em: https://monografias.ufop.br/handle/35400000/3787. Acesso em: 15 fev. 2023.

SPRINGER, V. A. S. **FootBot**: Uma Arquitetura de Microsserviços na Nuvem para Predição de Resultados de Partidas de Futebol. Orientador: Leonardo Guerreiro Azevedo. 2018. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Universidade Federal do Estado do Rio de Janeiro, [*S. l.*], 2018. Disponível em: https://bsi.uniriotec.br/publicacoes-de-tcc/. Acesso em: 16 fev. 2023.

VOIGT, R.; JÚNIOR, O. O. B. Análise de desempenho de arquitetura SOAP e REST para comunicação entre sistemas. **Anais SULCOMP**, [*s. l.*], v. 8, 17 fev. 2017. Disponível em: https://www.periodicos.unesc.net/ojs/index.php/sulcomp/article/view/3120. Acesso em: 20 fev. 2023.

ARAÚJO, R. O. **UMA REVISÃO TEÓRICA SOBRE A ARQUITETURA DE MICROSSERVIÇOS**. Orientador: Prof. Dr. Joilson Alves Junior. 2021. Monografia de Especialização (Especialização em Arquitetura e Gestão de Infraestrutura de TI) - Universidade Tecnológica Federal do Paraná, [*S. l.*], 2021. Disponível em: http://riut.utfpr.edu.br/jspui/handle/1/29009. Acesso em: 24 fev. 2023.

MENDONÇA, H. G. E-Commerce. **IPTEC**: Revista Inovação, Projetos e Tecnologias, [*s. l.*], v. 4, ed. 2, p. 240-251, 18 out. 2016. Disponível em: https://periodicos.uninove.br/iptec/article/view/9361/4128. Acesso em: 25 fev. 2023.

SILVA, W. M *et al*. Marketing digital, E-commerce e pandemia: uma revisão bibliográfica sobre o panorama brasileiro. **Research, Society and Development**, [*s. l.*], ano 2021, v. 10, ed. 5, 13 maio 2021. DOI 10.33448/rsd-v10i5.15054. Disponível em: https://rsdjournal.org/index.php/rsd/article/view/15054. Acesso em: 3 jun. 2023.

TEIXEIRA, M. A. M. Interações Cliente-Servidor na Web. *In*: TEIXEIRA, M. A. M. **Suporte a serviços diferenciados em servidores web**: modelos e algoritmos. Orientador: Prof. Dr. Marcos José Santana. 2004. Tese (Doutorado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, [*S. l.*], 2004. Disponível em: http://www.deinf.ufma.br/~mario/producao/tese_swds.pdf. Acesso em: 12 jun. 2023.

FERREIRA, G. HTTP: Desmistificando o protocolo da Web. **Alura**, [*s. l.*], 22 nov. 2022. Disponível em: https://www.alura.com.br/artigos/desmistificando-o-protocolo-http-parte-1?gclid=CjwKCAjws7WkBhBFEiwAIi1680ejCEoQx3oH4bdGnz93w_itt-HO-febkh4gJ6L2kmSNwpLdZ6fsEhoC0goQAvD_BwE. Acesso em: 20 jun. 2023.

MENDES, L. Z. R. **E-commerce**: origem, desenvolvimento e perspectivas. Orientador: Cássio da Silva Calvete. 2013. Trabalho de Conclusão de Curso (Bacharelado em Ciências Econômicas) - Universidade Federal do Rio Grande do Sul, [*S. l.*], 2013. Disponível em: https://www.lume.ufrgs.br/handle/10183/78391. Acesso em: 28 jun. 2023.

SMITH, B. **Um guia rápido para a GPLv3**. [*S. l.*], 3 mar. 2022. Disponível em: https://www.gnu.org/licenses/quick-guide-gplv3.html. Acesso em: 3 jul. 2023.

BOUSSIENGUI, P. H. **Segurança no PagRN**: Implementação do processo de autenticação e autorização usando OAuth 2.0 e programação orientada a aspectos. Orientador: Ramon dos Reis fontes. 2023. Trabalho de Conclusão de Curso (Especialista em Tecnologia da Informação) - Universidade Federal do Rio Grande do Norte, [*S. l.*], 2023. Disponível em: https://repositorio.ufrn.br/handle/123456789/54873. Acesso em: 27 nov. 2023.

DOMÍNGUEZ, A. **Engenharia de Software: Unidade 1: Conceitos**. 2010. Material Didático (Bacharelado em Sistemas de Informação) - UNIVERSIDADE FEDERAL DE ALAGOAS, [*S. l.*], 2010. Disponível em:

https://educapes.capes.gov.br/bitstream/capes/177122/2/Material%20Didatico-Engenharia%20de%20Software.pdf. Acesso em: 27 nov. 2023.

NEVES, V. **React: o que é, como funciona e um Guia dessa popular ferramenta JS.** Dev em <T>, [*s. l.*], 17 jan. 2023. Disponível em: https://www.alura.com.br/artigos/react-js. Acesso em: 21 out. 2023.

AMAZON WEB SERVICES. **O que é uma API (interface de programação de aplicações)?**. [*S. l.*], 8 dez. 2023. Disponível em: https://aws.amazon.com/pt/what-is/api/#:~:text=di%C3%A1rias%20no%20telefone.-,O%20que%20significa%20API%3F,de%20servi%C3%A7o%20entre%20duas%20aplica%C3%A7%C3%B5es. Acesso em: 8 dez. 2023.