

DEPARTAMENTO DE INFORMÁTICA



**ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO**

Métodos e Técnicas de Suporte ao Desenvolvimento de Software

E-Commerce Application

Trabalho realizador por:

João Silva (8220024),

Luís Silva (8220025)

Estudantes do mestrado em Engenharia Informática

Supervisores/Orientadores

Professor: Célio Carvalho

Professor: Ricardo Santos

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Resumo

De forma a se colocar em prática todos os conhecimento teóricos adquiridos na Unidade Curricular de Métodos e Técnicas de Suporte ao Desenvolvimento de Software foi-nos proposto o seguinte trabalho. Este trabalho consistiu na realização de um projeto com um tema à escolha do Grupo, sendo que este deve respeitar os princípio de *Domain Driven Design* e um padrão de arquitetura em Microserviços.

Tendo em conta esta proposta, o nosso grupo procedeu à escolha do tema, sendo este baseado na elaboração de um sistema de *E-Commerce*, ou seja, uma loja de comércio eletrónico. Esta escolha teve por base o interesse do grupo pelo tema, bem como a capacidade de distinção da Aplicação em diversos Microserviços.

A nível de software, procedeu-se à utilização de Diversas Tecnologias, tais como o *SpringBoot/SpringCloud*, *Postman*, *RabbitMq*, *Docker*, *Minikube* e *Git*.

A nível de objetivos do presente trabalho destacam-se os seguintes:

- Conceber a arquitetura técnica de um projeto segundo o paradigma microserviços;
- Desenvolver e implementar uma solução tecnológica de acordo com as práticas e padrões associados a aplicações orientadas a microserviços;
- Argumentar decisões de arquitetura e justificar opções de implementação no contexto de desenvolvimento de arquitetura baseadas em microserviços.

Índice

1	Introdução	1
1.1	Contextualização	1
1.2	Motivação	1
1.3	Estrutura do Documento	2
2	Fundamentação Teórica	3
3	Tecnologias e Ferramentas	5
3.1	<i>Spring</i>	5
3.2	<i>IntelliJ IDEA</i>	5
3.3	<i>Postman</i>	6
3.4	<i>Swagger</i>	6
3.5	<i>Docker</i>	6
3.6	<i>PostgreSQL</i>	7
3.7	<i>RabbitMq</i>	7
3.8	<i>Kubernetes</i>	8
3.9	<i>Minikube</i>	8
4	Processos	8
5	Requisitos	10
5.1	Requisitos Funcionais	10
5.1.1	Registo na Aplicação	10
5.1.2	Autenticação na Aplicação	10
5.1.3	Edição de Perfil dos Utilizadores	10
5.1.4	Edição de <i>Wallet</i> dos Utilizadores	11
5.1.5	Listagem dos Produtos	11
5.1.6	Criação de Ordem de Encomenda	11
5.1.7	Criação de Pagamento de Encomenda	12
5.1.8	Criação de <i>Review</i> de Encomenda	12
5.2	Requisitos Não Funcionais	12

6	Desenvolvimento	13
6.1	Arquitetura Conceptual	13
6.2	<i>Use cases</i>	14
6.2.1	Casos de uso da aplicação para o <i>role</i> de Utilizador	14
6.2.2	Casos de uso da aplicação para o <i>role</i> de Administrador	14
6.2.3	Casos de uso da aplicação para o <i>role</i> de Fornecedor	15
6.3	Diagramas de Sequências	15
6.3.1	Microserviço de Autenticação	15
6.3.2	Microserviço de Utilizadores	16
6.3.3	Microserviço de Produtos	19
6.3.4	Microserviço de <i>Wallet</i>	22
6.3.5	Microserviço de Email	23
6.3.6	Microserviço de Ordens de Encomenda	24
6.3.7	Microserviço de Pagamento	25
6.3.8	Microserviço de Shipping	27
6.3.9	Microserviço de Reviews	29
6.4	Serviços	30
6.4.1	<i>Eureka-Server</i>	30
6.4.2	<i>API Gateway</i>	31
6.4.3	Autenticação	31
6.4.4	Utilizadores	32
6.4.5	<i>Wallet</i>	32
6.4.6	Produtos	32
6.4.7	Ordens	32
6.4.8	Pagamentos	33
6.5	<i>Shipping</i>	33
6.5.1	<i>Reviews</i>	33
6.5.2	<i>Email</i>	34
7	Infraestrutura	35
7.1	<i>Docker Files</i>	35
7.2	<i>Deployment Files</i>	36

7.3	<i>Minikube Dashboard</i>	36
8	Conclusão	38
	Referências	39

Lista de Figuras

1	Exemplo de Arquitetura Monolítica.	3
2	Exemplo de Arquitetura em Microserviços.	4
3	Documentação gerado pelo <i>Swagger</i> para o Microserviço de Produtos. . .	6
4	Exemplo da utilização de Comunicação Assíncrona com o <i>RabbitMq</i> como <i>broker</i>	7
5	Processo de criação de produtos.	9
6	Processo de criação de uma encomenda.	9
7	Diagrama de <i>Domain Driven Design</i>	13
8	Arquitetura microserviços desenvolvida.	13
9	Casos de uso da aplicação para o <i>role</i> Utilizador.	14
10	Casos de uso da aplicação para o <i>role</i> Administrador.	14
11	Casos de uso da aplicação para o <i>role</i> Fornecedor.	15
12	Diagrama de Sequências de Registo de utilizadores.	15
13	Diagrama de Sequências de Autenticação de utilizadores.	16
14	Diagrama de Sequências de Registo de utilizadores.	16
15	Diagrama de Sequências de Atualização de utilizadores por <i>Id</i>	17
16	Diagrama de Sequências de Atualização de utilizadores por <i>Id</i> no caso de Administradores.	17
17	Diagrama de Sequências de Eliminação de utilizadores por <i>Id</i>	18
18	Diagrama de Sequências de Busca de todos os utilizadores.	18
19	Diagrama de Sequências de Busca de todos os utilizadores por nome. . .	18
20	Diagrama de Sequências de Busca de todos os utilizadores por <i>Id</i>	19
21	Diagrama de Sequências de Criação de Produtos.	19
22	Diagrama de Sequências de Atualização de Produtos.	20
23	Diagrama de Sequências de Eliminação de produtos por <i>Id</i>	20
24	Diagrama de Sequências de Busca de todos os Produtos.	20
25	Diagrama de Sequências de Busca de todos os Produtos por categoria. . .	21
26	Diagrama de Sequências de Busca de todos os Produtos por Nome. . . .	21
27	Diagrama de Sequências de Busca de todos os Produtos por <i>Id</i>	21
28	Diagrama de Sequências de Criação de Wallets.	22

29	Diagrama de Sequências de Inserção de dinheiro na <i>Wallet</i>	22
30	Diagrama de Sequências de Extração de dinheiro na <i>Wallet</i>	23
31	Diagrama de Sequências de Busca de todas as <i>Wallet</i>	23
32	Diagrama de Sequências de Busca de todas as <i>Wallet</i> por <i>Id</i> de Utilizador.	23
33	Diagrama de Sequências de Criação de Email.	24
34	Diagrama de Sequências de Busca de todas os <i>Emails</i>	24
35	Diagrama de Sequências de Busca de todas as <i>Emails</i> por <i>Id</i>	24
36	Diagrama de Sequências de Criação de Ordem de Encomenda.	25
37	Diagrama de Sequências de Busca de todas as Ordens de Encomenda.	25
38	Diagrama de Sequências de Busca de todas as Ordens de Encomenda por <i>Id</i>	25
39	Diagrama de Sequências de Criação de Pagamento.	26
40	Diagrama de Sequências de Busca de todas os Pagamentos.	26
41	Diagrama de Sequências de Busca de todas os Pagamento por <i>Id</i>	26
42	Diagrama de Sequências de Criação de <i>Shipping</i>	27
43	Diagrama de Sequências de Busca de todas os <i>Shipping</i>	27
44	Diagrama de Sequências de Busca de todas os <i>Shipping</i> por Id de Utili- zador e Id de Ordem de Encomenda.	28
45	Diagrama de Sequências de Busca de todas os <i>Shipping</i> por <i>Id</i>	28
46	Diagrama de Sequências de Atualização do estado do <i>Shipping</i>	28
47	Diagrama de Sequências de Criação de Reviews.	29
48	Diagrama de Sequências de Busca de todas as <i>Reviews</i>	29
49	Diagrama de Sequências de Busca de todas as Reviews por <i>Id</i>	30
50	Exemplo da utilização do <i>Eureka</i> como <i>Load Balancer</i>	31
51	Exemplo da criação de <i>Docker File</i> no microserviço de utilizadores.	35
52	Verificação da Presença das imagens no <i>Docker Hub</i>	36
53	<i>Dashboard</i> do <i>Minikube</i> do presente projeto.	37

1 Introdução

Este capítulo tem como objetivo apresentar o enquadramento e a motivação do tema do trabalho, os objetivos e resultados esperados no seu desenvolvimento e a abordagem de investigação selecionada.

1.1 Contextualização

De forma a se colocar em prática todos os conhecimento teóricos adquiridos na Unidade Curricular de Métodos e Técnicas de Suporte ao Desenvolvimento de Software (MTSDS), do mestrado em Engenharia Informática, foi-nos proposta o planeamento e implementação de um Sistema à nossa escolha, que respeite os princípio de *Domain Driven Design* e um padrão de arquitetura em Microserviços

Seguindo os conteúdos lecionados nas aulas teóricas, este projeto foi dividido em várias fases:

- Definição do Grupos.
- Proposta do Tema.
- Descrição do tema e dos microserviços associados.
- Desenvolvimento da arquitetura baseada em microserviços, implementação e configuração da infraestrutura.
- Disponibilização no *Minikube*.
- Elaboração de relatório final.
- Apresentação do projeto desenvolvido.

1.2 Motivação

O presente projeto em si, a definição do tema e a própria arquitetura já são uma motivação, uma vez que estamos habituados ao contexto de Arquiteturas Monolíticas, ainda não tendo contacto com Arquiteturas em Microserviços.

Por outro lado, além da definição arquitetural em Microserviços, o fator de se proceder com a atualização de uma *Stack* tecnológica atualizada e que vai de encontro com

as necessidades do mercado de trabalho, através da utilização de *Java*, *Spring Boot*, *Docker*, *Postman*, *Swagger*, *Minikube* e *Kubernetes*.

1.3 Estrutura do Documento

O presente relatório encontra-se dividido em 8 capítulos, sendo estes a Introdução, Fundamentação Teórica, Tecnologias e Ferramentas, Requisitos, Processos, Desenvolvimento, Infraestrutura e Conclusão, respetivamente.

No primeiro capítulo e presente Capítulo, a Introdução, faz-se uma contextualização, explica-se as motivação e a Estrutura do presente trabalho.

No segundo capítulo, a Fundamentação Teórica, faz-se uma breve explicação sobre as arquiteturas Monolítica e arquitetura em Microserviços e estabelece-se uma comparação entre ambas.

No terceiro capítulo, Tecnologias e Ferramentas, apresentam-se as diversas definições e apresenta-se um contexto para a utilização das diversas Tecnologias e ferramentas.

No quarto capítulo, elaboram-se os requisitos funcionais e não funcionais da presente aplicação.

No quinto capítulo, Processos, apresenta-se uma breve explicação sobre os processos definidos para o seguinte trabalho.

No sexto capítulo, Desenvolvimento, além da explicação da arquitetura conceptual do projeto, apresentam-se também os diversos microserviços, a sua finalidade e o tipo de comunicação que estabelecem.

No sétimo capítulo, de Infraestrutura, explica-se os diferentes passos utilizados na implementação da infraestrutura do presente trabalho, tais como os processos de *deploys* das bases de dados e microserviços.

Por último, no oitavo capítulo, apresentam-se as conclusões, limitações e propostas de melhoria do presente trabalho.

2 Fundamentação Teórica

Nas últimas décadas, ao nível do Desenvolvimento de aplicações tem-se vindo a implementar uma nova arquitetura, que se prende pelo desenvolvimento de aplicações em Microserviços.

Os microserviços são peças pequenas, *self-contained* e executáveis, ou seja, Módulos de software de função única, com interface e operações bem definidas, que podem ser desenvolvidos em tecnologias distintas, independentes uns dos outros, que possam realizar uma interação entre si, de forma síncrona ou assíncrona. Este pressuposto apresenta uma vantagem relativamente ao monolítico, onde os componentes são muito acoplados entre si, e por isso dependentes, onde qualquer alteração necessita necessariamente compilar, testar e compilar toda a solução de uma vez.

Num sistema Monolítico, quando em comparação com um sistema em microserviços existe uma mais dependência entre as diversas partes do sistema (*frontend*, *backend* e base de dados).

Nas seguinte imagens, a Figura 1 e a Figura 2, verifica-se as principais diferenças do ponto de vista de arquitetura entre uma Arquitetura Monolítica e uma Arquitetura em Microserviços.

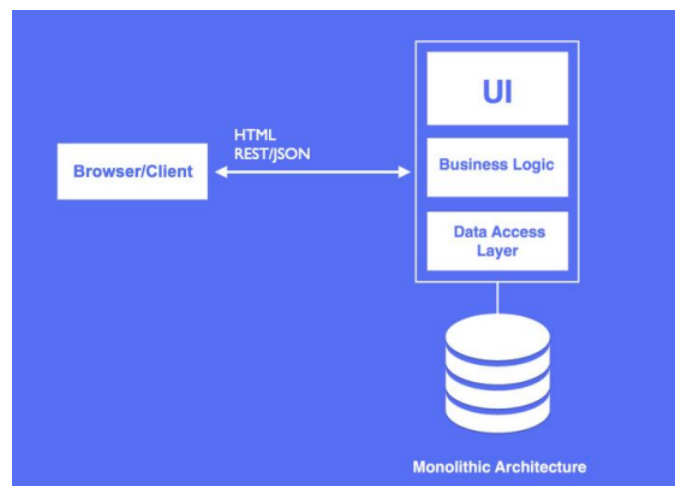


Figura 1: Exemplo de Arquitetura Monolítica.

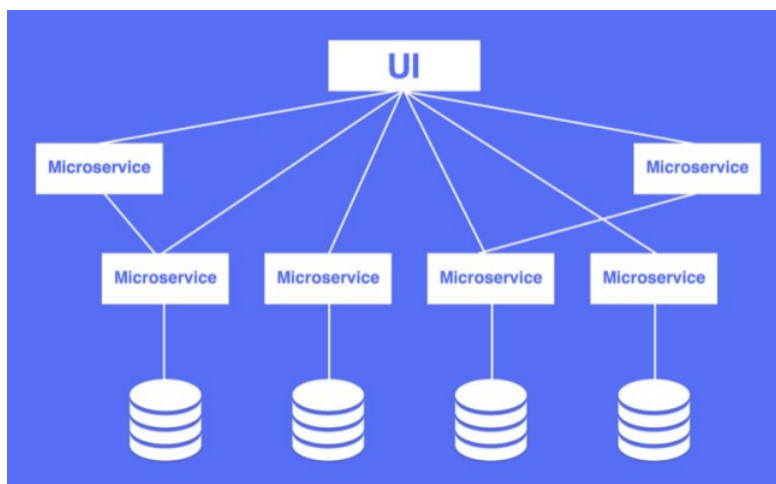


Figura 2: Exemplo de Arquitetura em Microserviços.

3 Tecnologias e Ferramentas

Na presente secção do Trabalho, apresentam-se as diferentes Tecnologias e Ferramentas utilizadas no desenvolvimento do seguinte trabalho prático, bem como os objetivos associados a cada uma.

3.1 *Spring*

O *Spring* define-se como uma *Framework* de desenvolvimento *open-source* para Java. Contém um conjunto de recursos e funcionalidade que simplificam e tornam mais rápido o desenvolvimento de aplicações. A framework é modular e flexível. Foi desenhada para a Programação orientada em objetos, sendo o objetivo de simplificar o desenvolvimento de aplicações em Java.

Esta framework é bastante relevante, uma vez que facilita o desenvolvimento de aplicações em diversos níveis, apresentando diversos módulos importantes, dos quais se destacam o *Spring Boot*, *Spring Cloud* e *Spring Data Jpa*.

O *Spring Boot* é o módulo que permite a simplificação do desenvolvimento de aplicações, o *Spring Cloud* apresenta um conjunto de ferramenta e bibliotecas que permite a facilitação de orquestração de Microserviços, com a criação de sistemas distribuídos (Load Balancing e Service Discovery), complementando a ação do *Spring Boot*.

O *Spring Data* é muito importante, uma vez que permite a simplificação do processo de acesso da bases de dados, que tanto podem ser relacionais, como bases de dados não relacionais, sendo que no presente trabalho utilizou-se o módulo *Spring Data Jpa* do *Spring Data*.

3.2 *IntelliJ IDEA*

O *IntelliJ IDEA* é um dos ambientes de desenvolvimento integrado (IDE), e foi o escolhido para a execução do seguinte trabalho.

A sua escolha foi importante, uma vez que este facilita bastante a criação de projetos Multi-Módulo em Maven, bastante suporte à linguagem Java, bem como um completa Integração com o repositório do *Git*.

3.3 *Postman*

O *Postman* é uma ferramenta muito importante durante todos os processos de desenvolvimento de aplicações, uma vez que permite o teste e a documentação de todos os pedidos via *API*. É bastante importante no processo de testes de uma aplicação e também pela possibilidade de orquestração entre a equipa, uma vez que permite a extração e partilha tanto das próprias coleções de testes, como das variáveis de ambiente.

3.4 *Swagger*

Swagger é uma ferramenta utilizada, tanto para a documentação, como para o posterior teste de *APIs* [1].

A título de exemplo, apresenta-se a imagem da Documentação gerada pelo *Swagger*, para os vários endpoints ao nível da API do microserviço de Produtos, na Figura 3.

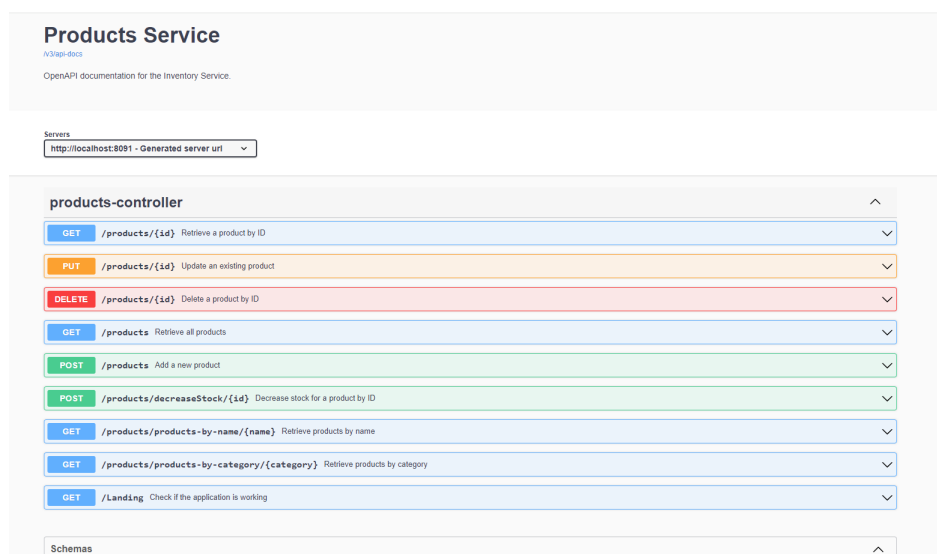


Figura 3: Documentação gerado pelo *Swagger* para o Microserviço de Produtos.

3.5 *Docker*

O *Docker* revelou-se uma das principais ferramentas na projeção e elaboração do presente trabalho, uma vez que foi utilizada na Gestão de *Containers*, tanto ao nível de bases de dados, como do *Broker* de mensagens assíncronas, bem como a sua integração com o *Spring Boot*, de modo a colocar cada Microserviço e as suas dependências no seu respetivo *Container*.

3.6 *PostgreSQL*

PostgreSQL é uma das diferentes existentes bases de dados relacionais, que apresenta a vantagem de ser livre de encargos económicos e *open-source*.

Por outro lado, esta suporta *queries* de forma relacional, através de SQL, e *queries* não-relacionais, através de JSON.

3.7 *RabbitMQ*

Ao longo do projeto, além da comunicação síncrona entre Microserviços, também se utilizou a comunicação assíncrona (nem sempre existe a necessidade de um microserviço responder de forma automática a outro), e é neste tipo de comunicação que se utiliza o *RabbitMQ*.

Um sistema de mensagens, permite a troca de mensagens entre os diferentes Microserviços de forma assíncrona e desacoplada (não existe dependências), ao contrário de um pedido *REST* síncrono. Este tipo de sistemas utiliza 4 componentes principais: o *Message Broker*, o *Producer*, o *Consumer* e a *Message*.

Na prática, os *Producers* enviam uma mensagem (Microserviço que envia mensagem), que é entregue ao *Message Broker* (*RabbitMQ*). Posteriormente, os *Consumers* (Microserviços que recebem mensagem) recebem estas mensagens do *Message Broker*, para processamento.

O *RabbitMQ* é um tipo de *broker* existente, onde existem várias *queue* de mensagens, que ficam à espera que os *Consumers* consumam as suas mensagens. Apresenta um funcionamento semelhante a um buffer, trabalhando ao seu próprio ritmo, não consumindo as mensagens todas simultaneamente.

Na figura 4 pode-se verificar o princípio de funcionamento previamente explicado da Comunicação Assíncrona, em que se utiliza o *RabbitMQ* como *broker* de comunicação.



Figura 4: Exemplo da utilização de Comunicação Assíncrona com o *RabbitMQ* como *broker*.

3.8 *Kubernetes*

O *Kubernetes* é um orquestrador de *containers* desenvolvido originalmente pela Google.

A sua principal valência consiste na possibilidade de implementar diversas aplicações de forma automatizada, bem como fazer o escalamento e monitorização das mesmas de forma simples. Desde 2014, é uma tecnologia *open-source*, sendo atualmente mantido pela *CNCF* (*Cloud Native Computing Foundation*).

3.9 *Minikube*

Minikube consiste numa aplicação que implementa um *cluster* de *Kubernetes* local de forma criar um ambiente de desenvolvimento similar a um *cluster*, tendo este apenas um *node*.

Este *node* é agnóstico ao nível de sistema operativo, podendo ser utilizado tanto em sistemas baseados em *Unix* como em *Windows*. A sua elevada popularidade, deve-se ao facto de permitir ao programador a possibilidade de realizar testes às suas aplicações preparadas para *Kubernetes* de forma rápida e intuitiva.

4 Processos

Nesta secção, apresentamos dois dos principais processos do desenvolvimento do projeto. Sendo eles o processo da criação de produtos e a criação de *orders* realizada pelos users.

Como se pode verificar na figura 5, os utilizadores iniciam o processo da criação de produtos, pois no nosso sistema, o *User* tem a possibilidade de para além de ter o cargo de Administrador, pode também ele ser Fornecedor. Sendo assim, poderá criar e editar produtos referentes à sua conta.

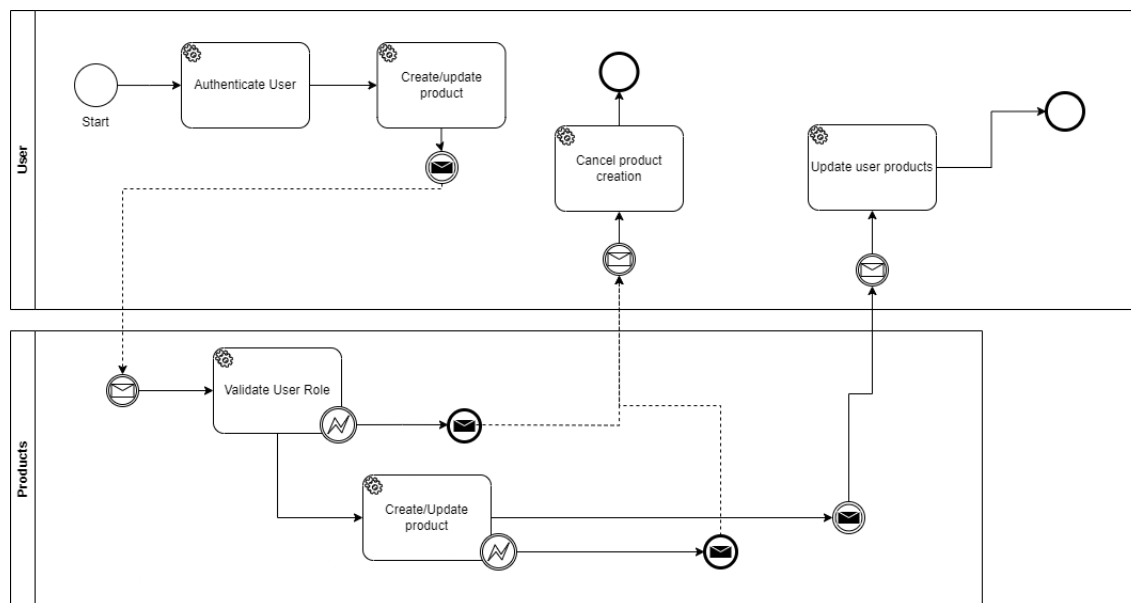


Figura 5: Processo de criação de produtos.

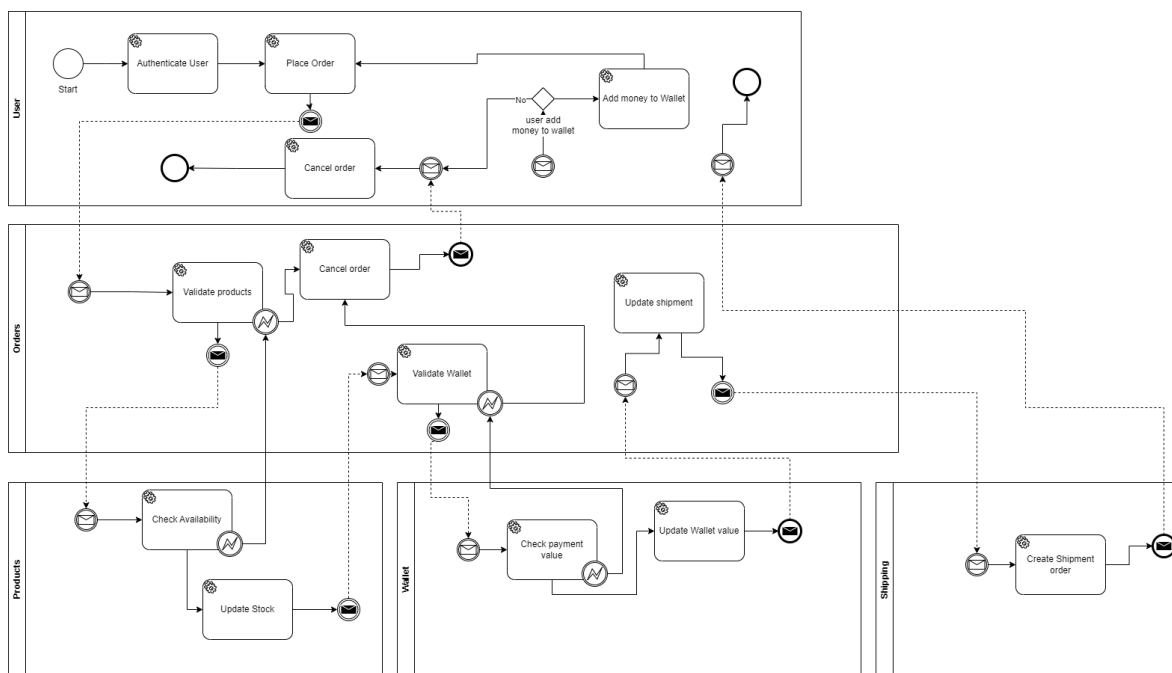


Figura 6: Processo de criação de uma encomenda.

5 Requisitos

5.1 Requisitos Funcionais

5.1.1 Registo na Aplicação

Definição de Requisitos de Utilizador

- O Utilizador necessita de fazer o registo na aplicação para a poder utilizar.

Definição de Requisitos de Sistema

- Aquando do registo, o sistema deve solicitar os dados necessários ao utilizador, nomeadamente o *e-mail* e *password*.
- O sistema não deve permitir o registo de utilizadores com o mesmo login.
- O sistema deve armazenar os dados do utilizador.

5.1.2 Autenticação na Aplicação

Definição de Requisitos de Utilizador

- O utilizador deverá introduzir o seu *login* e a respetiva *password* para iniciar sessão na aplicação.

Definição de Requisitos de Sistema

- sistema deve verificar a validade da tentativa de autenticação, verificando se o par *login* e *password* inseridos correspondem a algum utilizador existente no sistema. Caso não correspondam, o sistema não deve permitir a autenticação.

5.1.3 Edição de Perfil dos Utilizadores

Definição de Requisitos de Utilizador

- O utilizador deve conseguir editar as suas informações pessoais, nomeadamente o seu e-mail e password.

Definição de Requisitos de Sistema

- O sistema deve armazenar as alterações.

5.1.4 Edição de *Wallet* dos Utilizadores

Definição de Requisitos de Utilizador

- O utilizador deve conseguir colocar dinheiro na sua respetiva *wallet*.

Definição de Requisitos de Sistema

- O sistema deve armazenar as alterações.

5.1.5 Listagem dos Produtos

Definição de Requisitos de Utilizador

- O utilizador deve conseguir visualizar todos os produtos existentes no sistema.

Definição de Requisitos de Sistema

- O sistema deve apresentar a lista de todos os produtos existentes no sistema.

5.1.6 Criação de Ordem de Encomenda

Definição de Requisitos de Utilizador

- O utilizador deve conseguir criar uma ordem de encomenda com os produtos que deseja, na quantidade que deseja.

Definição de Requisitos de Sistema

- O sistema deve verificar se existem os produtos, e caso estes existam, deve fazer a verificação se tem quantidade suficiente.
- Uma vez validados todo o processo, o sistema deve guardar os dados relativos à ordem de Encomenda.

5.1.7 Criação de Pagamento de Encomenda

Definição de Requisitos de Utilizador

- O utilizador deve conseguir criar um pagamento para uma ordem de encomenda com os produtos que deseja, na quantidade que deseja.

Definição de Requisitos de Sistema

- O sistema deve verificar se o Utilizador apresenta dinheiro na sua carteira.
- Uma vez validados todo o processo, o sistema deve guardar os dados relativos ao Pagamento da Encomenda e colocar o *Shipping* no primeiro estado.

5.1.8 Criação de *Review* de Encomenda

Definição de Requisitos de Utilizador

- O utilizador deve conseguir criar *reviews* para os produtos existentes numa ordem de encomenda em seu nome.

Definição de Requisitos de Sistema

- O sistema deve verificar se a ordem de Encomenda foi paga, se o estado de *Shipping* é de que a encomenda foi entregue, e deve validar também se o produto existe na Ordem de encomenda.
- Uma vez validados todo o processo, o sistema deve guardar os dados relativos à *Review* da Ordem de Encomenda.

5.2 Requisitos Não Funcionais

- A aplicação deverá ser de uso fácil, com *endpoints* corretamente definidos.
- aplicação deverá estar disponível durante os 7 dias da semana, 24 horas por dia.
- O tempo de resposta da aplicação deve ser o mais curto possível para não prejudicar a confeção da receita nem influenciar de forma negativa a experiência do utilizador na aplicação.
- O sistema deve ser suportado por todos os browsers.

6 Desenvolvimento

6.1 Arquitetura Conceptual

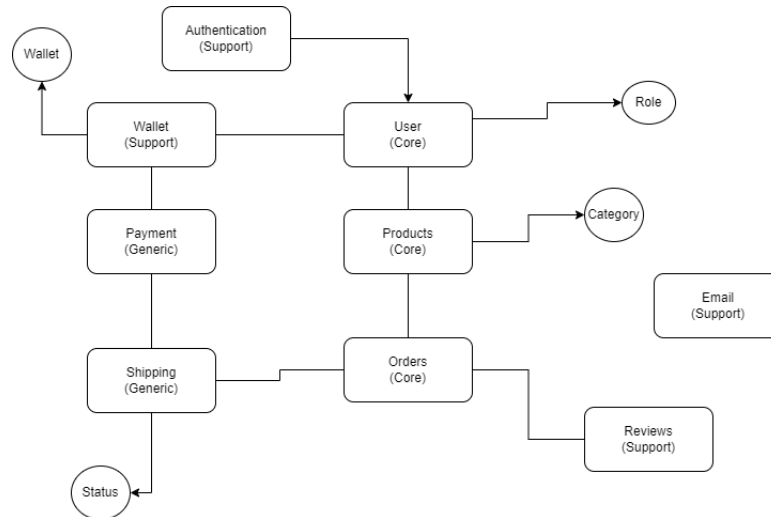


Figura 7: Diagrama de *Domain Driven Design*.

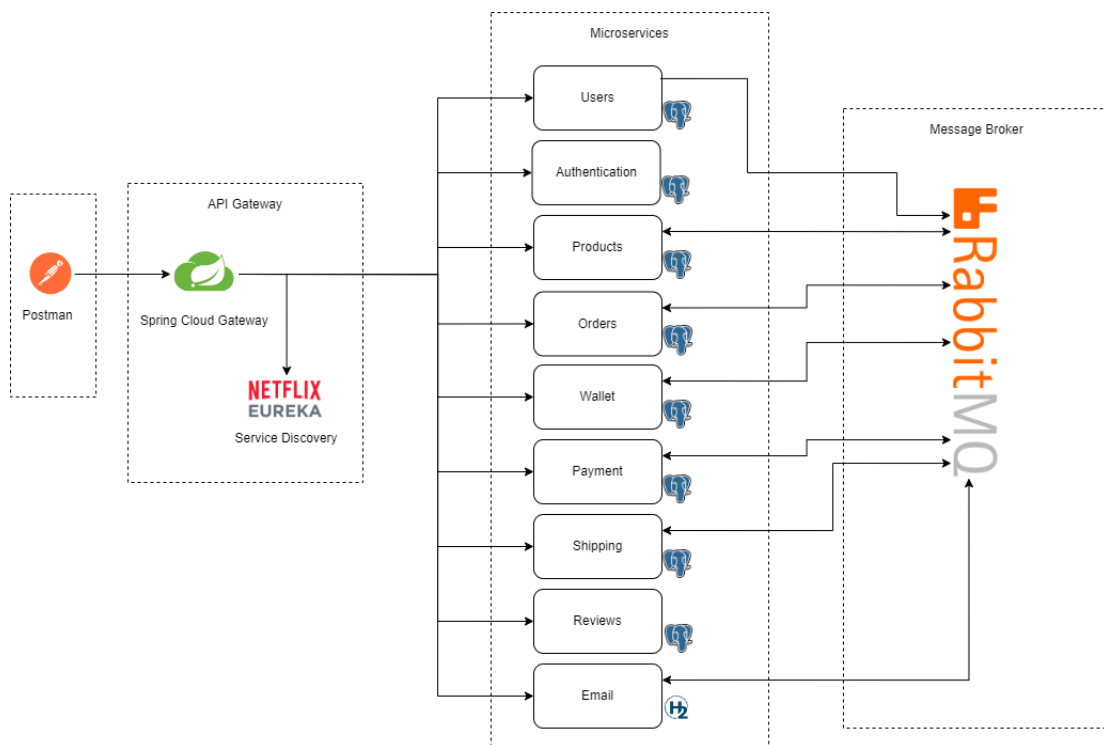


Figura 8: Arquitetura microserviços desenvolvida.

6.2 Use cases

Para as especificações do uso da aplicação, estes estão descritos, nesta secção, através da elaboração de diagramas de casos de uso.

6.2.1 Casos de uso da aplicação para o *role* de Utilizador

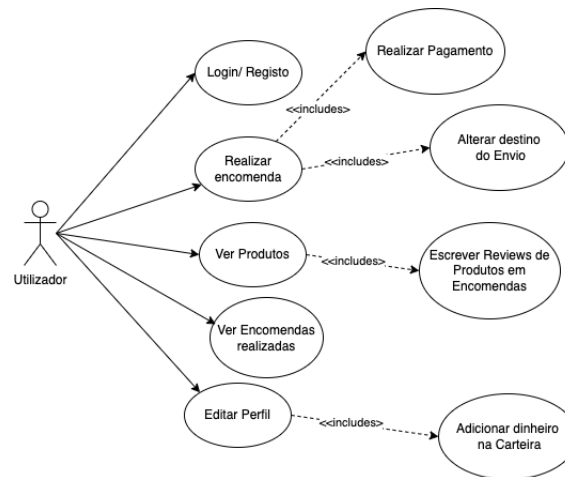


Figura 9: Casos de uso da aplicação para o *role* Utilizador.

6.2.2 Casos de uso da aplicação para o *role* de Administrador

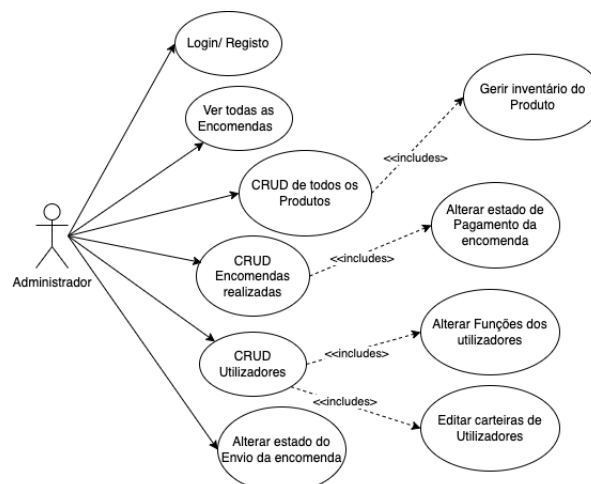


Figura 10: Casos de uso da aplicação para o *role* Administrador.

6.2.3 Casos de uso da aplicação para o *role* de Fornecedor

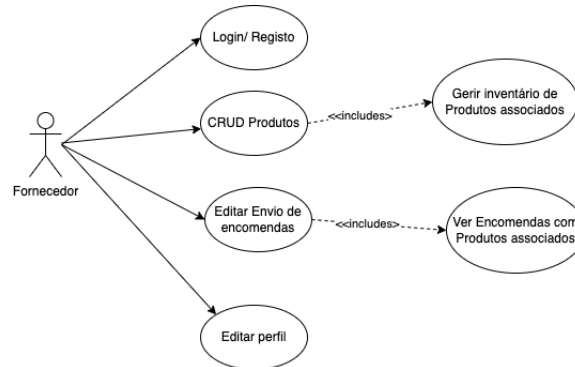


Figura 11: Casos de uso da aplicação para o *role* Fornecedor.

6.3 Diagramas de Sequências

Baseado na Arquitetura aplicacional proposta para este sistema de software, os diagramas de sequência de subsistema permitem representar de forma clara, simples e completa a interação entre os vários subsistemas da aplicação, bem como, qual o fluxo aplicacional e de dados durante cada *use case*.

6.3.1 Microserviço de Autenticação

Registo de Utilizadores:

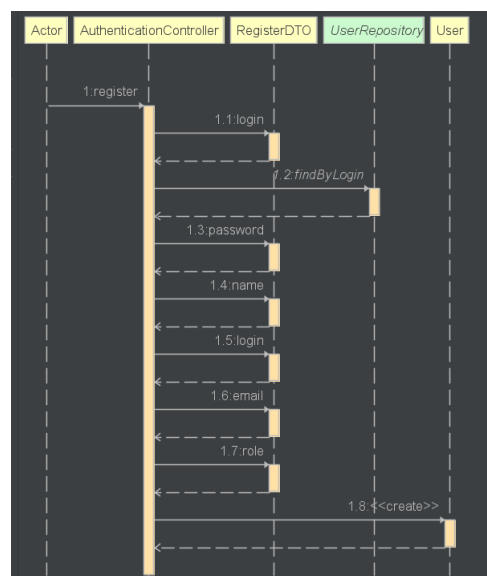


Figura 12: Diagrama de Sequências de Registo de utilizadores.

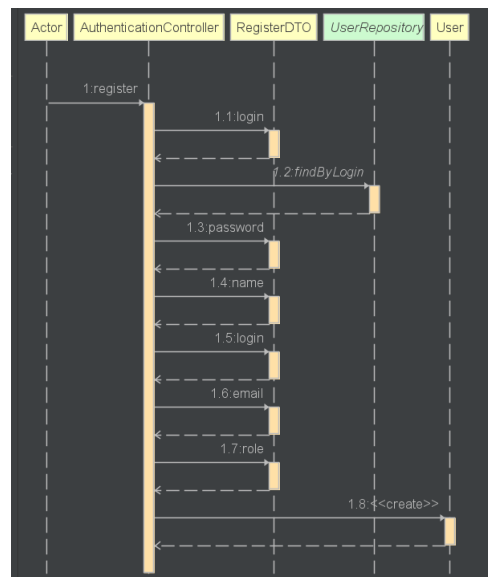
Autenticação de Utilizadores:

Figura 13: Diagrama de Sequências de Autenticação de utilizadores.

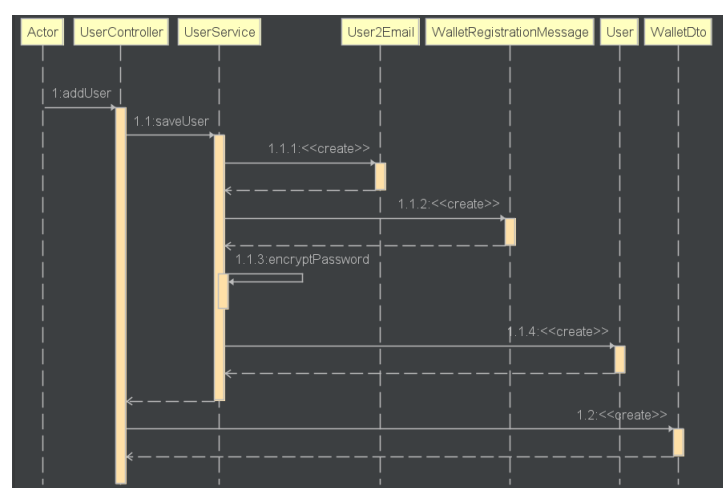
6.3.2 Microserviço de Utilizadores*Registo de Utilizadores:*

Figura 14: Diagrama de Sequências de Registo de utilizadores.

Atualização de Utilizadores:

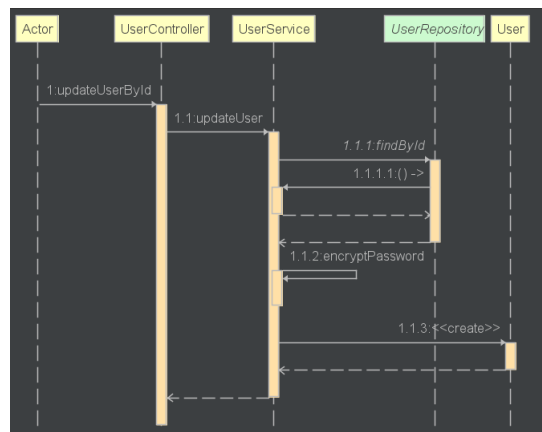


Figura 15: Diagrama de Sequências de Atualização de utilizadores por *Id*.

Atualização de Utilizadores por Administrador:

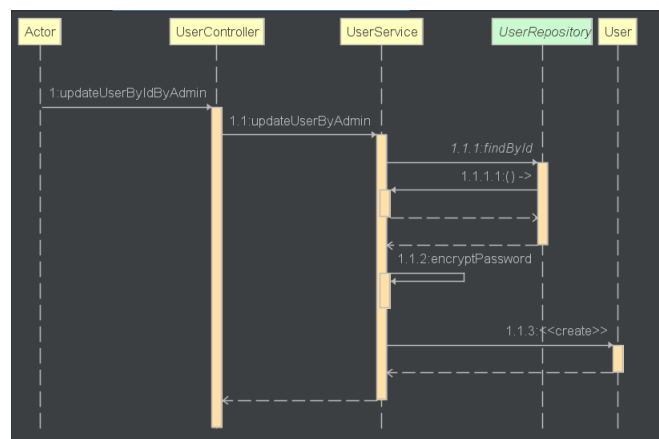


Figura 16: Diagrama de Sequências de Atualização de utilizadores por *Id* no caso de Administradores.

Eliminação de Utilizadores:

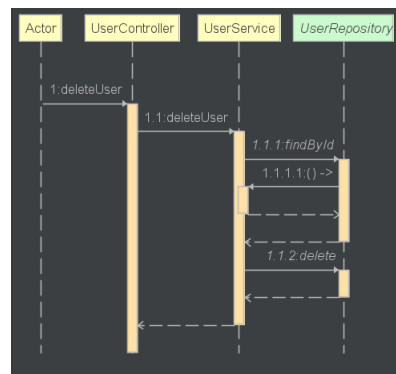


Figura 17: Diagrama de Sequências de Eliminação de utilizadores por *Id*.

Busca de todos os Utilizadores:

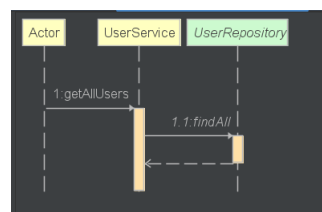


Figura 18: Diagrama de Sequências de Busca de todos os utilizadores.

Busca de todos os Utilizadores por nome:

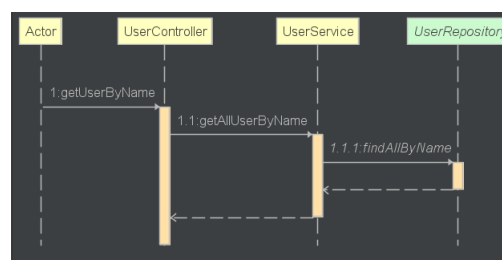


Figura 19: Diagrama de Sequências de Busca de todos os utilizadores por nome.

Busca de Utilizador por Id

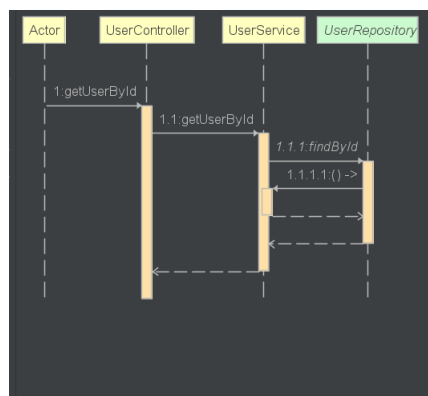


Figura 20: Diagrama de Sequências de Busca de todos os utilizadores por *Id*.

6.3.3 Microserviço de Produtos

Registo de Produtos:

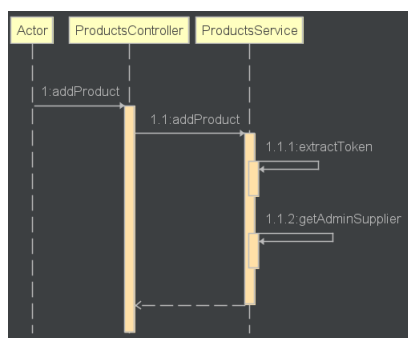


Figura 21: Diagrama de Sequências de Criação de Produtos.

Atualização de Produtos:

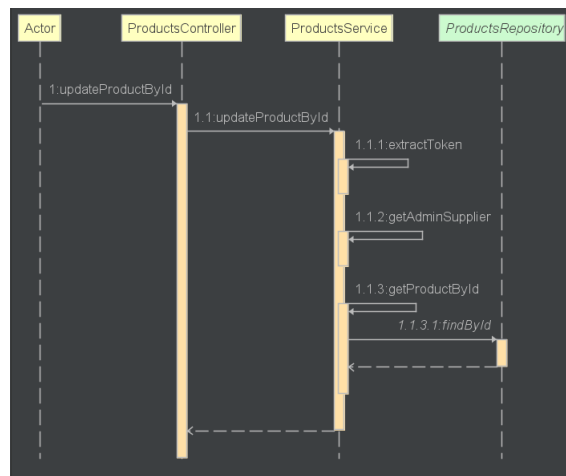


Figura 22: Diagrama de Sequências de Atualização de Produtos.

Eliminação de Produtos:

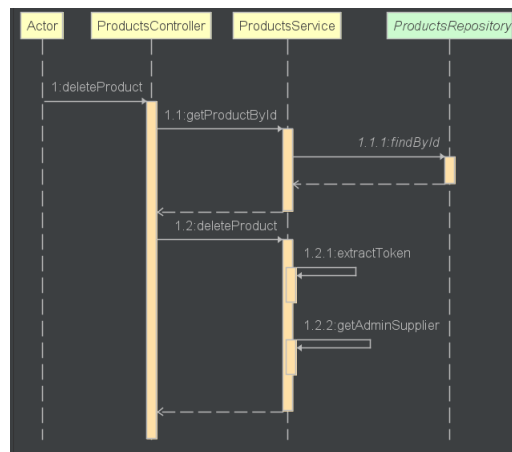


Figura 23: Diagrama de Sequências de Eliminação de produtos por *Id*.

Busca de todos os Produtos:

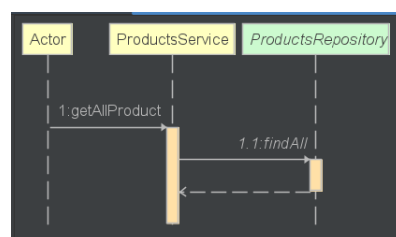


Figura 24: Diagrama de Sequências de Busca de todos os Produtos.

Busca de todos os Produtos por Categoria:

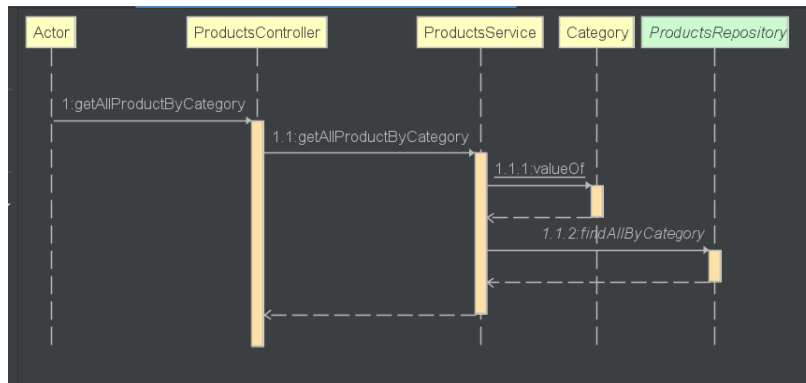


Figura 25: Diagrama de Sequências de Busca de todos os Produtos por categoria.

Busca de todos os Produtos por Nome:

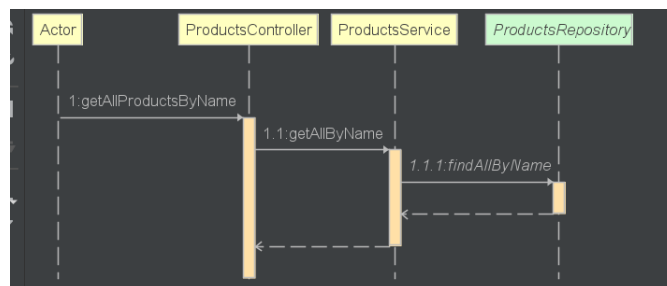


Figura 26: Diagrama de Sequências de Busca de todos os Produtos por Nome.

Busca de todos os Produtos por Id:

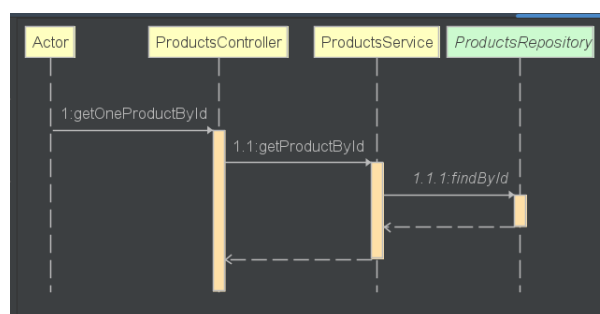


Figura 27: Diagrama de Sequências de Busca de todos os Produtos por *Id*.

6.3.4 Microserviço de *Wallet*

Registo de Wallet:

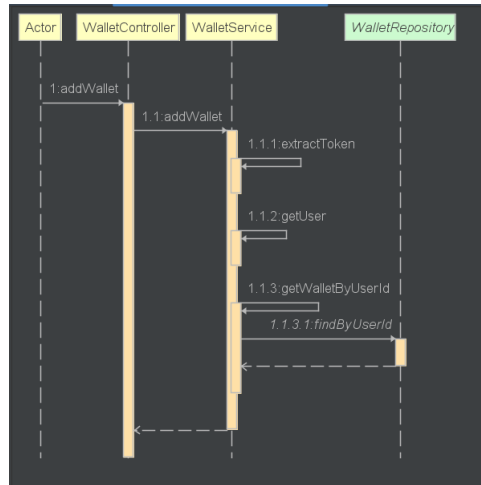


Figura 28: Diagrama de Sequências de Criação de Wallets.

Inserção de dinheiro na Wallet:

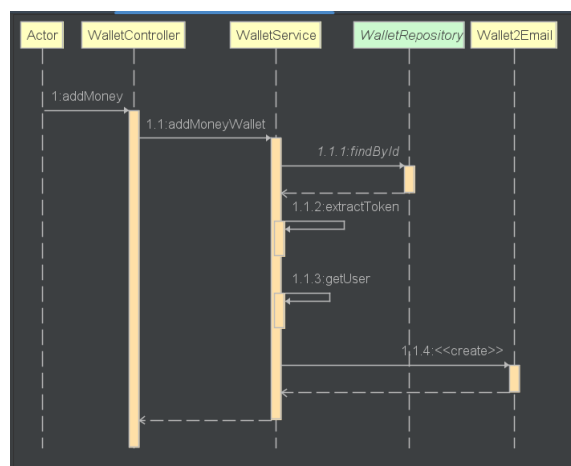


Figura 29: Diagrama de Sequências de Inserção de dinheiro na Wallet.

Inserção de dinheiro na Wallet:

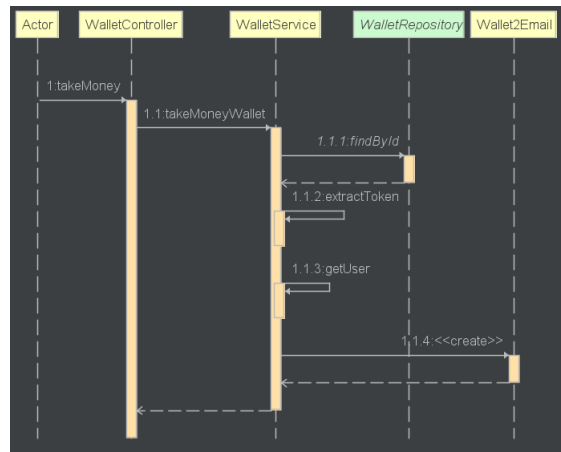


Figura 30: Diagrama de Sequências de Extração de dinheiro na *Wallet*.

Busca de todos as Wallet:

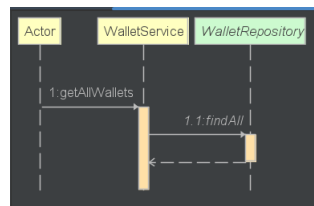


Figura 31: Diagrama de Sequências de Busca de todas as *Wallet*.

Busca de todos as Wallet por Id:

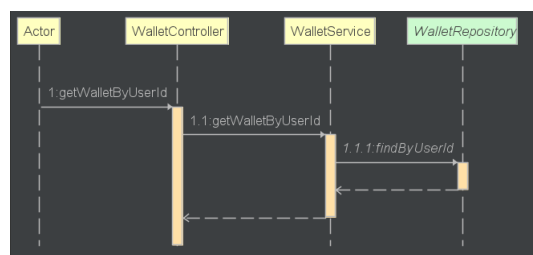


Figura 32: Diagrama de Sequências de Busca de todas as *Wallet* por *Id* de Utilizador.

6.3.5 Microserviço de Email

Registo de Email:

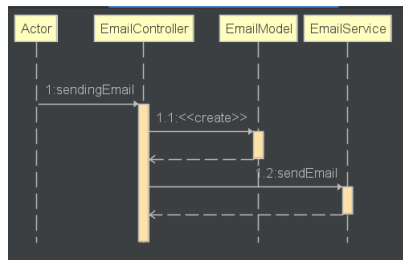


Figura 33: Diagrama de Sequências de Criação de Email.

Busca de todos os Emails:

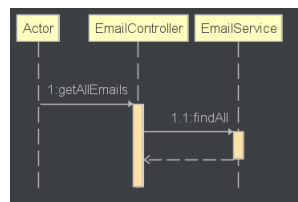


Figura 34: Diagrama de Sequências de Busca de todas os *Emails*.

Busca de todos as Emails por Id:

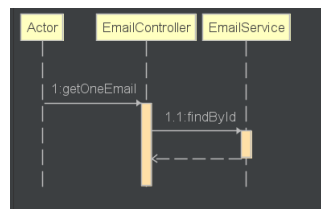


Figura 35: Diagrama de Sequências de Busca de todas as *Emails* por *Id*.

6.3.6 Microserviço de Ordens de Encomenda

Registo de Ordem de Encomenda:

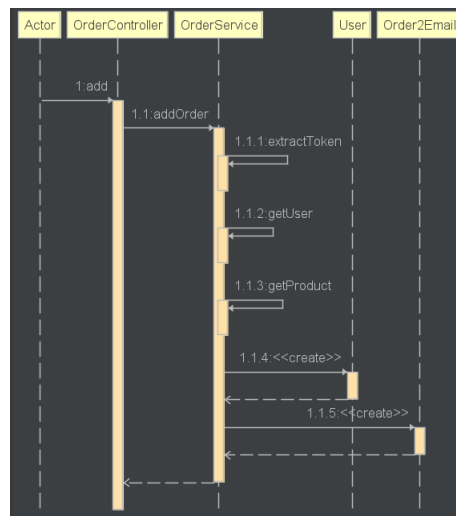


Figura 36: Diagrama de Sequências de Criação de Ordem de Encomenda.

Busca de todos as Ordens de Encomenda:

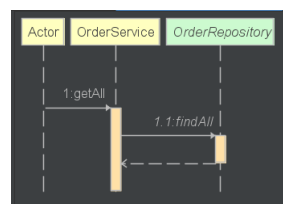


Figura 37: Diagrama de Sequências de Busca de todas as Ordens de Encomenda.

Busca de todos as Ordens de Encomenda por Id:

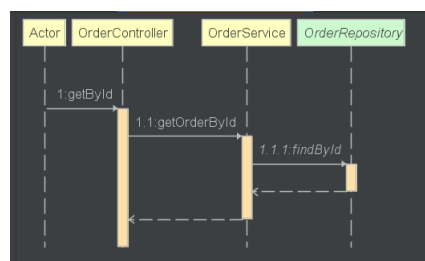


Figura 38: Diagrama de Sequências de Busca de todas as Ordens de Encomenda por *Id*.

6.3.7 Microserviço de Pagamento

Registo de Pagamento:

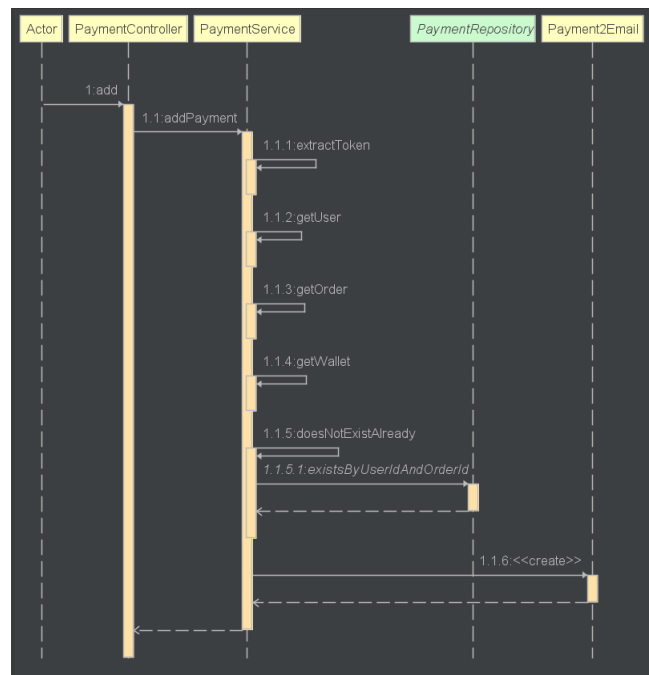


Figura 39: Diagrama de Sequências de Criação de Pagamento.

Busca de todos as Pagamento:

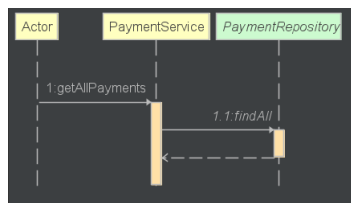


Figura 40: Diagrama de Sequências de Busca de todas os Pagamentos.

Busca de todos os Pagamentos por Id:

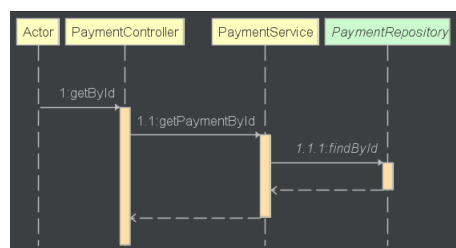


Figura 41: Diagrama de Sequências de Busca de todas os Pagamento por *Id*.

6.3.8 Microserviço de Shipping

Registo de Shipping:

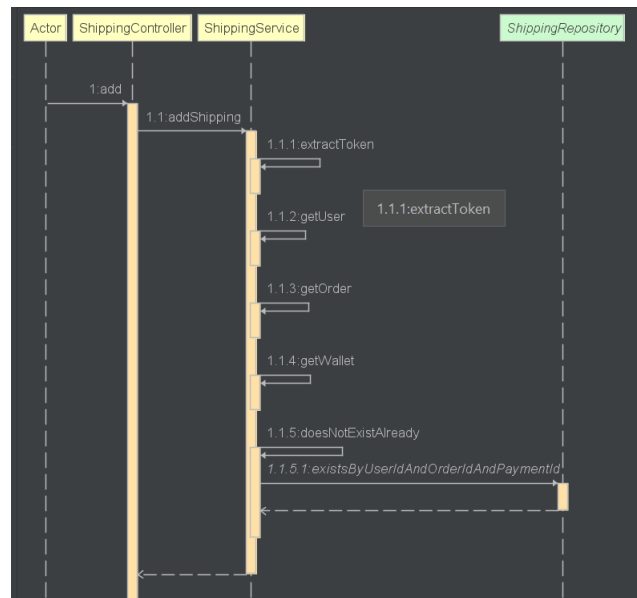


Figura 42: Diagrama de Sequências de Criação de *Shipping*.

Busca de todos as Pagamento:

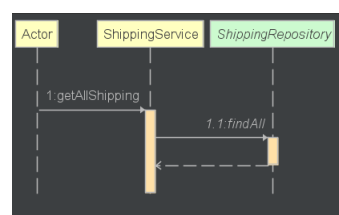


Figura 43: Diagrama de Sequências de Busca de todas os *Shipping*.

Busca de todos os Shipping por Id de Utilizador e Id de Ordem de Encomenda:

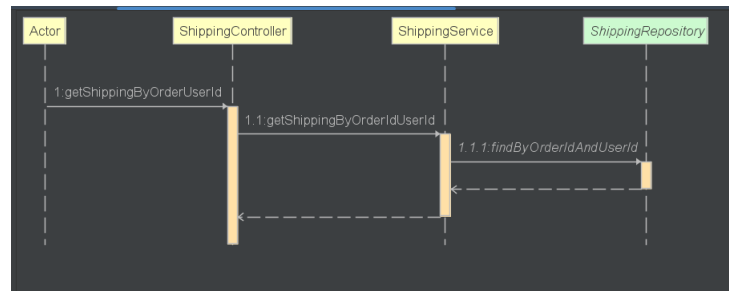


Figura 44: Diagrama de Sequências de Busca de todas os *Shipping* por Id de Utilizador e Id de Ordem de Encomenda.

Busca de todos os Shipping por Id:

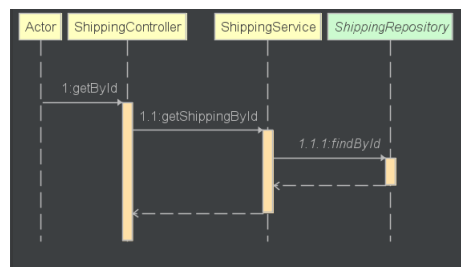


Figura 45: Diagrama de Sequências de Busca de todas os *Shipping* por *Id*.

Atualização do estado do Shipping

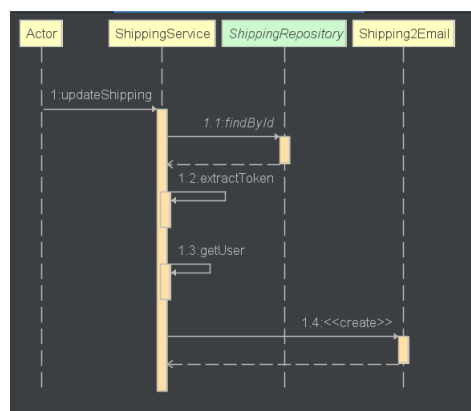


Figura 46: Diagrama de Sequências de Atualização do estado do Shipping.

6.3.9 Microserviço de Reviews

Registo de Reviews:

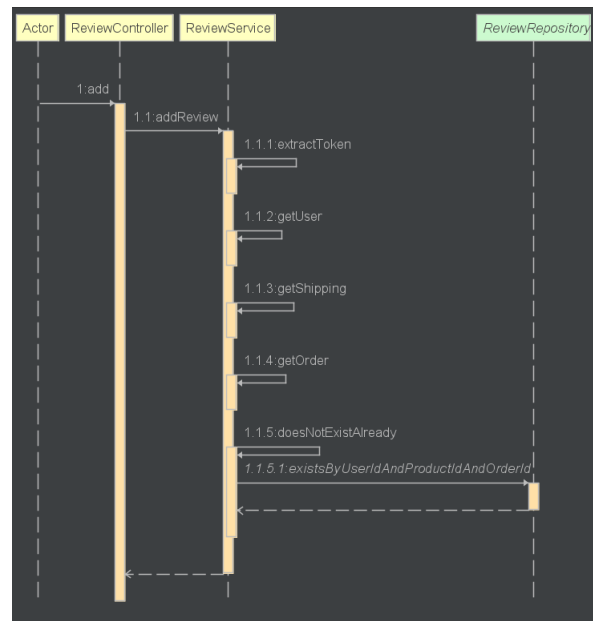


Figura 47: Diagrama de Sequências de Criação de Reviews.

Busca de todos as Reviews:

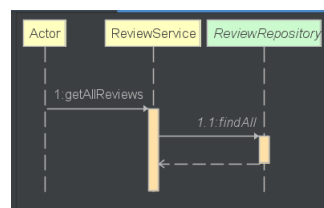


Figura 48: Diagrama de Sequências de Busca de todas as *Reviews*.

Busca de todos as Reviews por Id:

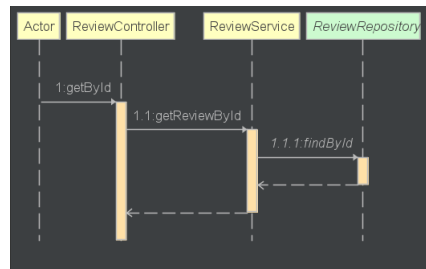


Figura 49: Diagrama de Sequências de Busca de todas as Reviews por *Id*.

6.4 Serviços

Na subsecção explica-se os diferentes Microserviços implementados ao longo do projeto, bem como a sua finalidade e tipo de comunicação utilizada pelo mesmo.

6.4.1 *Eureka-Server*

O *Eureka* trata-se de um microserviço fundamental que se comporta como uma espécie de servidor, que centraliza as informações de todos os outros serviços disponíveis, permitindo que estes sejam visualizados e disponibilizados.

Basicamente, o *Eureka Service Discovery* permite que os restantes microserviços se registem em instâncias de forma dinâmica, uma vez que podem se registar em mais do que uma, atuando basicamente como um *Load Balancer* que lida com as várias solicitações.

Na figura 50 pode-se verificar o padrão de funciona do *Eureka* como *Load Balancer* do sistema.

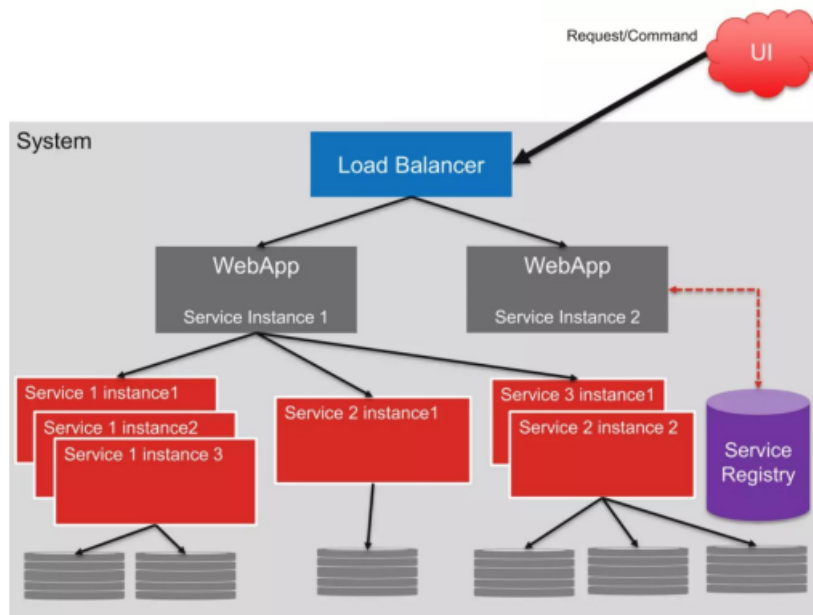


Figura 50: Exemplo da utilização do *Eureka* como *Load Balancer*.

6.4.2 *API Gateway*

API Gateway é um microserviço de tamanha importância, uma vez que é o primeiro ponto de contacto da aplicação, e onde são definidos todos os pedidos do sistema. A *API Gateway* além de agregar os diversos pedidos realizados ao sistema, também faz a sua divisão entre os diversos microserviços, ou seja, faz o roteamento e direcionamento dos pedidos para os microserviços corretos. Além disto também permite a autenticação e autorização antes do direcionamento, apenas permitindo o acesso a determinados microserviços consoante a autenticação correta e o tipo de role da autenticação.

6.4.3 Autenticação

Microserviço responsável pela parte relativo ao registo e autenticação de utilizadores de acordo com o seu tipo de *role*.

Após a verificação da existência de um utilizador registado, caso este exista, é gerado um *token* temporário que autoriza o acesso a partes da plataforma em que o seu tipo de *role* tenha acesso.

Este serviço faz comunicação via *REST* com a camada de dados em PostgreSQL, de forma a assegurar a persistência de dados.

6.4.4 Utilizadores

Microserviço responsável pela orquestração dos diversos utilizadores do Sistema. Este microserviço apresenta o *CRUD* relativo à gestão dos utilizadores do sistema, sendo que a parte relativa ao *update* e *delete* apenas pode ser realizada pelo próprio utilizador ou pelo administrador.

Este serviço faz comunicação via *REST* com a camada de dados em PostgreSQL, de forma a assegurar a persistência de dados.

6.4.5 *Wallet*

Microserviço que funciona como *subscriber* de mensagens assíncrona publicadas pelo processo de registo de Utilizadores, criando assim uma *wallet* com o seu valor a zero, aquando da respetiva criação de utilizadores.

Posteriormente, é possível de ser carregada com a quantidade monetária que o próprio utilizador entenda. É importante também a sua comunicação com o microserviço de Pagamento, uma vez que apenas é possível a realização de um pagamento, caso o utilizador apresente dinheiro da sua *wallet*.

6.4.6 Produtos

Microserviço que é utilizado com o intuito de fazer a gestão dos produtos e da quantidade existente de cada produto, onde é possível adicionar produtos caso a role do utilizador seja de *Supplier*, contudo, é possível a listagem de todos os produtos independentemente do tipo de *role*.

Este microserviço encontra-se em comunicação com o microserviço de Ordens, uma vez que faz a verificação se os produtos existentes numa ordem apresentam stock suficiente.

6.4.7 Ordens

Microserviço que é utilizado com o intuito de fazer a gestão das ordens de encomenda, onde é possível adicionar ordens caso a role do utilizador seja de *User*, contudo, é possível a listagem de todas as ordens de encomenda independentemente do tipo de *role*.

Este microserviço encontra-se em comunicação com o microserviço de Produtos, uma vez que faz a verificação se os produtos existentes numa ordem apresentam stock suficiente e apresenta também comunicação assíncrona com o microserviço de email, com o intuito de envio de notificações.

6.4.8 Pagamentos

Microserviço que é utilizado com o intuito de fazer a gestão dos pagamentos, onde é possível adicionar pagamentos caso a role do utilizador seja de *User*, contudo, é possível a listagem de todos os pagamentos independentemente do tipo de *role*.

Este microserviço encontra-se em comunicação com o microserviço de ordens, uma vez que apenas é possível realizar o pagamento de uma ordem de encomenda existente. Apresenta também comunicação de *wallet*, uma vez que apenas é permitida a efetivação de uma compra, caso o utilizador apresente dinheiro na *wallet*. Por outro lado, este serviço também comunica de forma assíncrona com o microserviço de email e com o microserviço de *Shipping*, onde após o registo de um pagamento, inicializa-se o processo de shipping com o seu primeiro estado de encomenda.

6.5 *Shipping*

Microserviço que é utilizado com o intuito de fazer a gestão do estado do *Shipping* das encomendas, onde é possível fazer a atualização do estado da encomenda, caso apresenta o role de *Supplier*.

Este microserviço recebe mensagens de forma assíncrona do microserviço de pagamento, e assim que é efetivado o pagamento, este cria um ordem de *shipping* no primeiro estado da encomenda.

Por outro lado, este microserviço também apresenta comunicação com o microserviço de *Reviews*, pois apenas é possível a realização de uma review, caso o estado do *shipping*, seja o estado final (Entregue ao utilizador).

6.5.1 *Reviews*

Microserviço que permite aos utilizadores que realizaram determinada encomenda, proceder com a avaliação dos produtos presentes na encomenda, contudo, apenas é

possível a avaliação de um produto numa encomenda, caso esta encomenda tenha sido paga e o estado da Encomenda seja de entrega realizada ao cliente.

Este serviço faz comunicação via *REST* com a camada de dados em Postgres, de forma a assegurar a persistência de dados.

6.5.2 *Email*

Microserviço que funciona como o principal *subscriber* do sistema de Mensagens assíncrona, que procede ao envio de emails para os emails definidos para os diversos utilizadores em ocasiões diversas: registo de utilizador, registo de encomenda, registo de pagamento, alteração do estado da encomenda.

Este serviço apenas se encontra relacionado com uma base de dados temporal H2, uma vez que o propósito é o de envio de notificação, não sendo necessária a persistência de dados.

7 Infraestrutura

Neste capítulo, apresentam-se os passos que foram seguidos na criação da Infraestrutura do projeto, através da utilização do *Docker*, *Minikube* e *Kubernetes*.

7.1 *Docker Files*

Numa primeira fase, começou-se com a criação de todos os *Docker Files*, sendo que foi criada um por cada Microserviço.

A título de exemplo deste tipo de Ficheiros, temos o implementação no Microserviço de Utilizadores, apresenta na figura 51, onde existem diversos comandos com diferentes funcionalidades.

- *FROM*: Permite indicar qual a imagem que será utilizada, onde se apresenta a versão do *Java*.
- *EXPOSE*: Permite indicar qual as portas que o serviço apresenta para fora.
- *ENTRYPOINT*: Permite indicar quais OS parâmetros de arranque do *Container*.

```
FROM openjdk:20
VOLUME /tmp
COPY target/users.jar users.jar
ENTRYPOINT ["java", "-jar", "./users.jar"]
EXPOSE 8012
```

Figura 51: Exemplo da criação de *Docker File* no microserviço de utilizadores.

Estes *Docker Files* permitem a criação de todas as imagens do *Docker* relativas a cada um dos diferentes Microserviços, que posteriormente serão submetida no *Docker Hub*. Este processo ajuda a facilitar o processo de implementação no *Minikube*.

Uma vez realizado o processo de submissão das imagens no Docker Hub, verificou-se a sua presença, tal como se verifica na Figura .

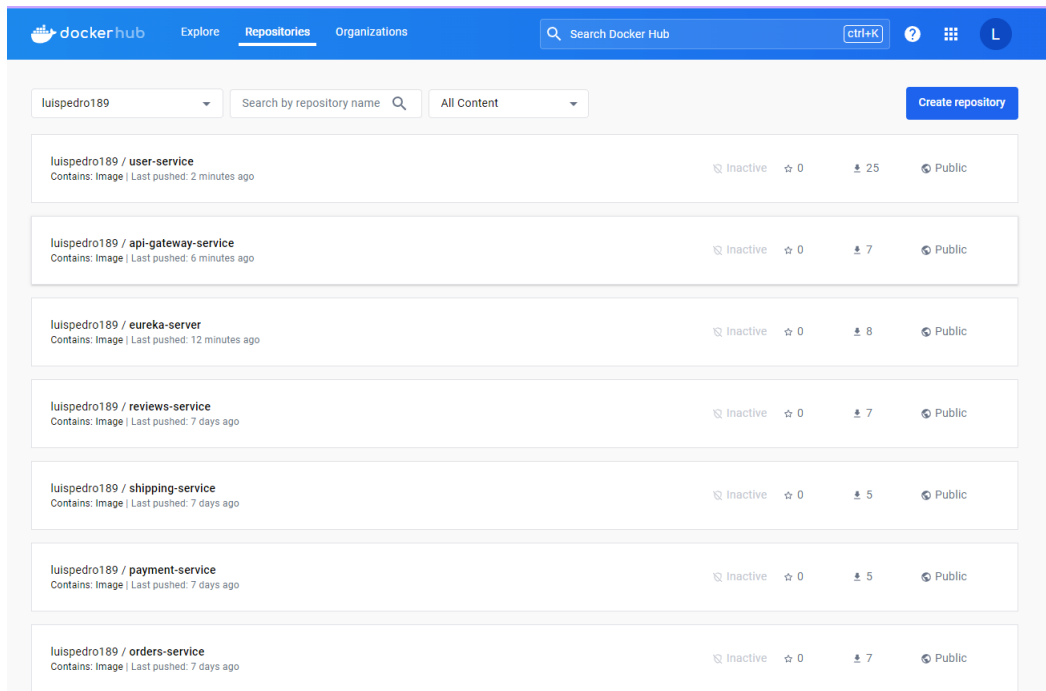


Figura 52: Verificação da Presença das imagens no *Docker Hub*.

7.2 *Deployment Files*

Numa segunda fase, procedeu-se com a criação dos ficheiro de *deployment* em formato *YAML*, tanto para as bases de dados, como para os próprios microserviços, sendo que foi criado um ficheiro para a base de dados e um para o serviço, para cada um dos serviços, e também para o broker de Comunicação Assíncrona, o *RabbitMq*.

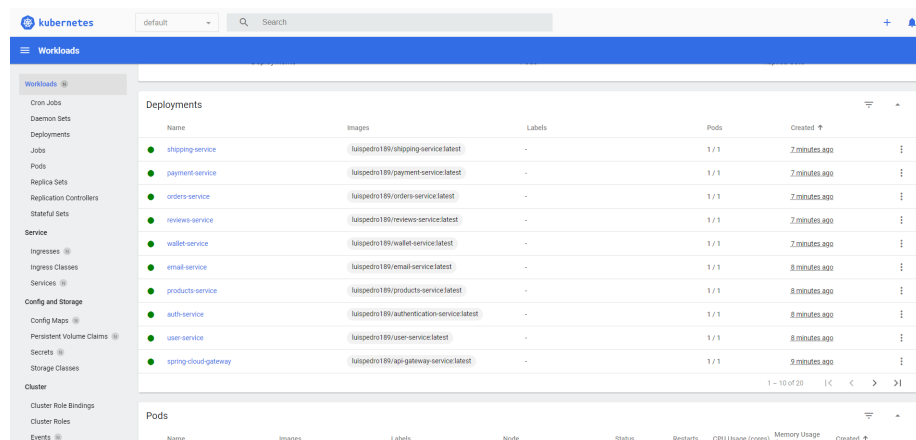
O ficheiro de *deployment* em formato *YAML* apresenta diversas informações acerca dos serviços, tais como a parte relativa à aplicação (*deployment*), a parte relativo ao serviço e as variáveis de ambiente.

7.3 *Minikube Dashboard*

O *Minikube Dashboard* apresenta uma Interface de visualização Gráfica, onde é possível, além da visualização de todos os recursos existentes no nosso cluster de *Kubernetes* local, desde os *Deployments*, *Pods*, Serviços, Configurações e volumes, também é possível verificar os seus *logs*, bem como fazer a sua eliminação, pelo que foi bastante útil a sua integração ao longo da realização do projeto.

A título de exemplo, apresenta-se a Figura z, onde se pode visualizar a Interface

gráfica do Minikube *Dashboard* apresentada no seguinte projeto.



The screenshot displays the Minikube Dashboard interface. The left sidebar contains a navigation menu with categories like Workloads, Service, Config and Storage, and Cluster. The main content area is titled 'Deployments' and shows a table of deployed services. Below this, a 'Pods' section is partially visible.

Name	Images	Labels	Pods	Created
shipping-service	luipedro189/shipping-service:latest	-	1 / 1	7 minutes ago
payment-service	luipedro189/payment-service:latest	-	1 / 1	7 minutes ago
orders-service	luipedro189/orders-service:latest	-	1 / 1	7 minutes ago
reviews-service	luipedro189/reviews-service:latest	-	1 / 1	7 minutes ago
wallet-service	luipedro189/wallet-service:latest	-	1 / 1	7 minutes ago
email-service	luipedro189/email-service:latest	-	1 / 1	8 minutes ago
products-service	luipedro189/products-service:latest	-	1 / 1	8 minutes ago
auth-service	luipedro189/authentication-service:latest	-	1 / 1	8 minutes ago
user-service	luipedro189/user-service:latest	-	1 / 1	8 minutes ago
spring-cloud-gateway	luipedro189/api-gateway-service:latest	-	1 / 1	8 minutes ago

1 - 10 of 20

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
------	--------	--------	------	--------	----------	-------------------	----------------------	---------

Figura 53: *Dashboard* do *Minikube* do presente projeto.

8 Conclusão

O presente projeto foi muito interessante, tanto ao nível do conceito do projeto em si, como do ponto de vista preparatório, teórico e prático, dado que permitiu que se adquirissem diversos conhecimentos tanto do ponto de vista de arquitetura como do ponto de vista de programação, tendo por base uma arquitetura em Microserviços, estando esta *stack* bastante atualizada para o que se promove a nível tecnológico no mundo industrial.

Apesar do referido anteriormente, também se apresentaram diversas dificuldades ao longo do projeto, tais como o desconhecimento inicial pelas tecnologias utilizadas, bem como a falta de conhecimento do tipo de arquitetura implementado, bem como desconhecimento relativo à infraestrutura, nomeadamente de *Kubernetes* e *Minikube*.

De uma forma geral achamos que o produto final corresponde aos objetivos definidos no início da elaboração do presente trabalho, bem como com a proposta inicial de arquitetura enviada aos docentes, contudo claramente poderia ser alvo de diversas melhorias, tais como a otimização do ponto de vista de programação, uma implementação mais aprofundada tanto ao nível de testes unitários, como do ponto de vista de segurança, e o desenvolvimento de outro tipo de testes, tais como testes de integração.

Por outro lado, também seria interessante a orquestração de fenómenos de *CI/CD* através de *Jenkins*, por exemplo, a disponibilização ao nível da *cloud* ou a utilização de projetos de monitorização, por exemplo com a utilização do *Grafana*.

Referências

- [1] Swagger - what is swagger? <https://swagger.io/docs/specification/2-0/what-is-swagger/>. Accessed on January 4, 2024.