

Atividade em grupo – 3º ano TECH 2024

Mensageria para o e-commerce da Natalia

Entregas (*Deliverables*):

1. Documentação da implementação do microserviço: o prazo de entrega para todos os grupos é dia **10 de maio**, fim do dia, via plataforma Odette.
2. Container no Docker para integração com o Kafka nas máquinas locais:
Microserviço de testes, turma 3E – na aula do dia **13 de maio**
Microserviço de testes, turma 3D – na aula do dia **15 de maio**

Demais microserviços: na aula do dia **10 de maio**

Os microserviços de teste terão o tempo de uma aula a mais porque dependem da integração com os demais microserviços.

Dicas úteis para documentação de microserviços:

<https://www.pythian.com/blog/technical-track/top-three-considerations-documenting-microservice>

<https://www.quora.com/What-is-the-best-way-to-document-microservices-in-a-distributed-system>

Obs: A implantação (deploy) em uma infraestrutura de nuvem não faz parte da entrega desta prática. Poderá ser realizada em atividade futura, quando a Germinare tiver a infraestrutura disponível para trabalho.

Sumário

Atividade em grupo – e-commerce da Natalia	2
Objetivo.....	2
Protótipo do front-end.....	2
Microsserviços.....	3
Plataformas	3
Estrutura de tópicos do Kafka	3
Funcionamento dos microsserviços.....	4
Antifraude	4
Venda de produtos adquiridos de terceiros	5
Pagamentos.....	5
Venda de encomendas.....	6
Testes	6
Banco de dados no PostgreSQL.....	6

Atividade em grupo – e-commerce da Natalia

A Natalia tem um e-commerce que vende roupas e lembrancinhas:

- (1) venda de mercadorias de terceiros – as roupas que ela revende
- (2) encomendas de lembrancinhas – sem manutenção de itens em estoque

Objetivo

Este trabalho visa implementar uma solução para o e-commerce da Natalia usando Kafka como middleware, com foco na comunicação entre os microsserviços com o uso da mensageria.

Demais aspectos podem ser considerados atendidos com funcionalidade mínima, bastando explicar ao restante da classe quais são as restrições de funcionamento.

Protótipo do front-end

O protótipo do Front-end está no Figma

(https://www.figma.com/team_invite/redeem/EeTzBRIECQipSStpg0nuhW).

O objetivo do protótipo é ilustrar um fluxo de operação mínimo no e-commerce.

Restrições: A operação da loja tem dois fluxos principais, que não se misturam: venda de roupas e encomenda de lembrancinhas. Quem abre um carrinho de produtos e desiste, precisa recomençar o processo a partir da *landing page* do e-commerce. O mesmo acontece quando é aberto um carrinho de encomenda de lembrancinhas.

Microserviços

Serão implementados cinco:

1. Antifraude
2. Venda de produtos adquiridos de terceiros
3. Processamento de pagamentos
4. Venda de encomendas
5. Testes

O detalhamento das funcionalidades é feito em um mapa mental na plataforma Miro:

[Turma 3E](#)

[Turma 3D](#)

O detalhamento de cada um foi indicado separadamente, no item **Funcionamento dos microserviços**.

Plataformas

Docker	Para execução independente de configurações de rede.
Kafka	O middleware
Github	Distribuição do código-fonte; versionamento de software.
Node.js	recomendável, para aplicação do conteúdo da disciplina de programação assíncrona.
Outras linguagens de programação	Deve-se garantir a interoperabilidade do microserviço com o middleware implantado em Kafka.
PostgreSQL	Banco de dados com cadastro de clientes, produtos, pedidos e carrinhos de compra. Implantado no Render.
Opcional: plataformas de teste tipo K6	Testes de carga do sistema.

A configuração do Kafka é a mesma usada em aula, com um cluster de 3 brokers, uma instância de KafkaUI e uma instância do Zookeeper.

Estrutura de tópicos do Kafka

Os tópicos para troca de mensagens entre microserviços são os seguintes:

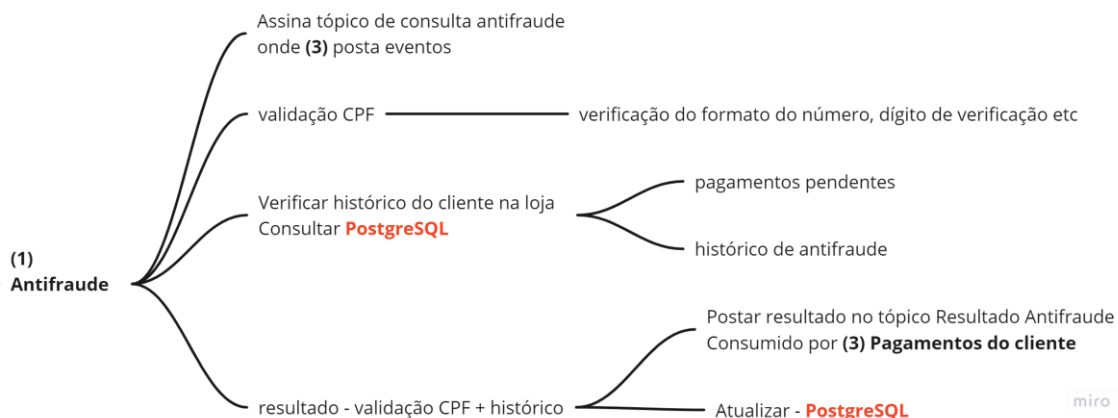
Tópicos	Nome a ser dado na implementação	Produtor	Consumidor
Consulta antifraude	cons_antifraude	3 – Pagamentos	1 – Antifraude
Veredito antifraude	vered_antifraude	1 – Antifraude	3 – Pagamentos
Reserva de estoque	res_estoque	2 – Venda de produtos	Kafka Connect (consulta ao banco de dados)
Baixa de estoque	bx_estoque	2 – Venda de produtos	Kafka Connect (atualização do banco de dados)
Requisição de cobrança	req_cobra	2 – Venda de produtos 4 – Encomendas	3 – Pagamentos

Tópicos	Nome a ser dado na implementação	Produtor	Consumidor
Status do pagamento	status_pgto	3 – Pagamentos	2 – Venda de produtos 4 – Encomendas
Requisição de envio	req_envio	2 – Venda de produtos 4 – Encomendas	5 – Testes
Código de rastreamento	cod_rastreio	5 – Testes	2 – Venda de produtos 4 – Encomendas
Disparo de ordem de produção	dispara_prod	4 – Encomendas	5 – Testes
Aviso de produção concluída	prod_ok	5 – Testes	4 – Encomendas
Requisição de processamento de pagamento	req_pgto	3 – Pagamentos	5 – Testes
Resultado de processamento de pagamento	result_pgto	5 – Testes	3 – Pagamentos

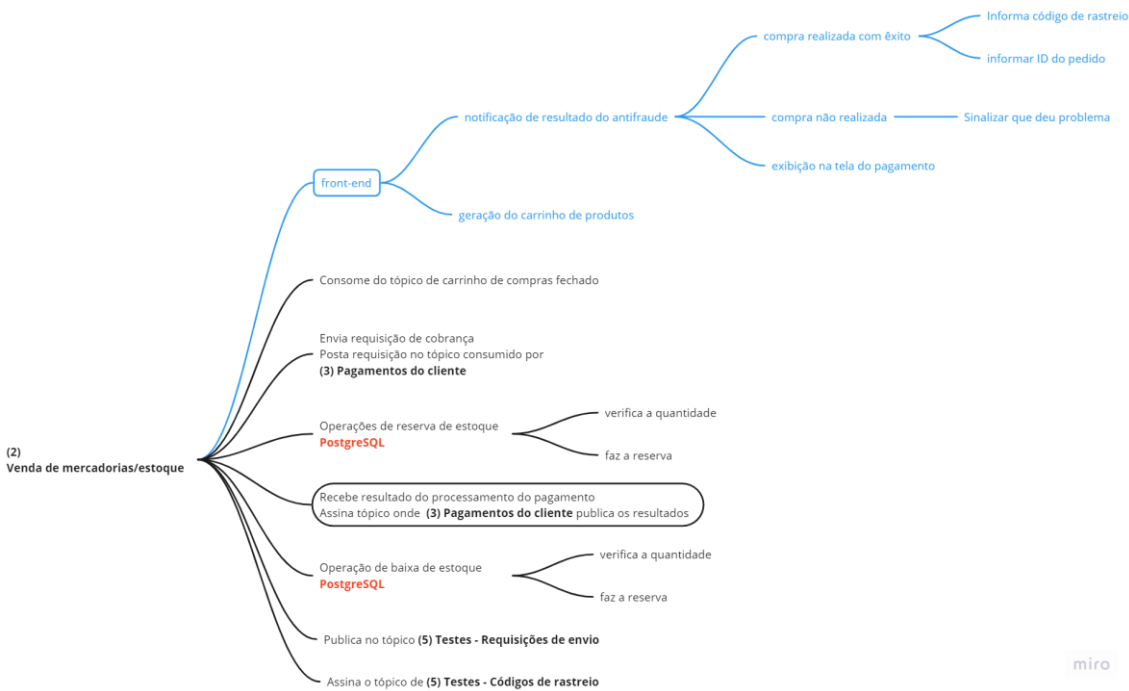
Funcionamento dos microserviços

Os requisitos mínimos de funcionamento dos microserviços é informado na forma dos mapas mentais, a seguir. As funcionalidades que não aparecem no protótipo fornecido no Figma podem ser implementadas da forma mais simples possível, lembrando que o foco do trabalho é a comunicação via mensageria, não a UX.

Antifraude

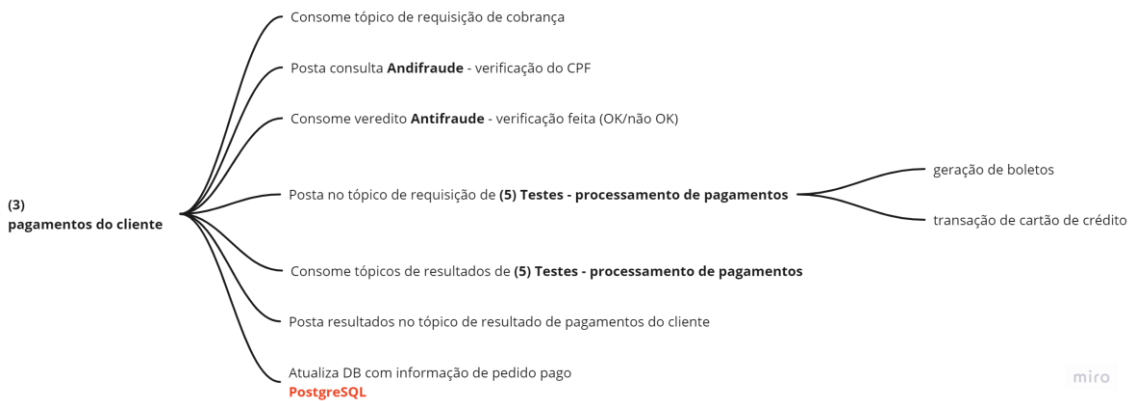


Venda de produtos adquiridos de terceiros



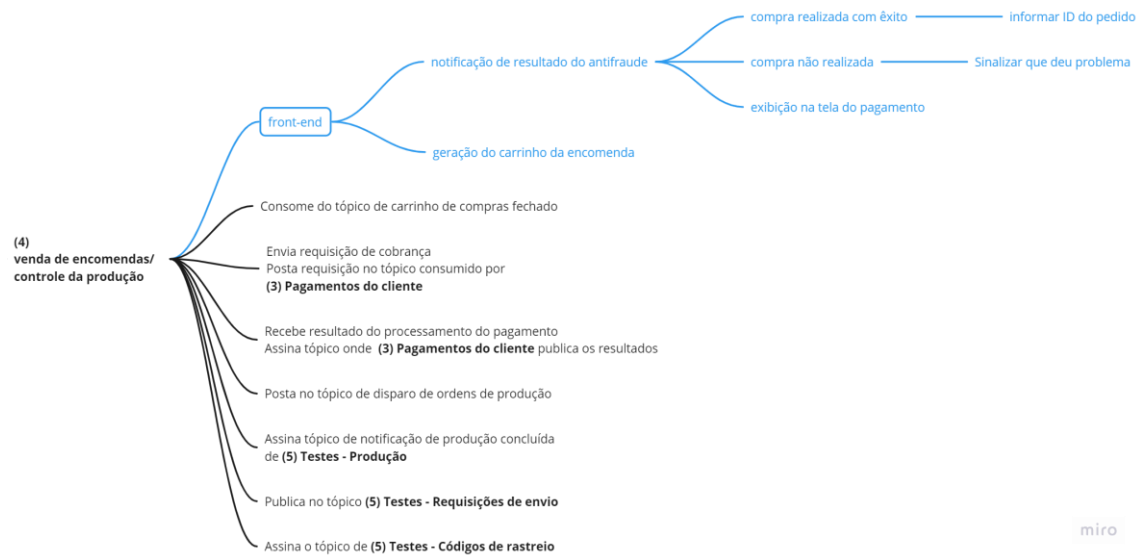
miro

Pagamentos

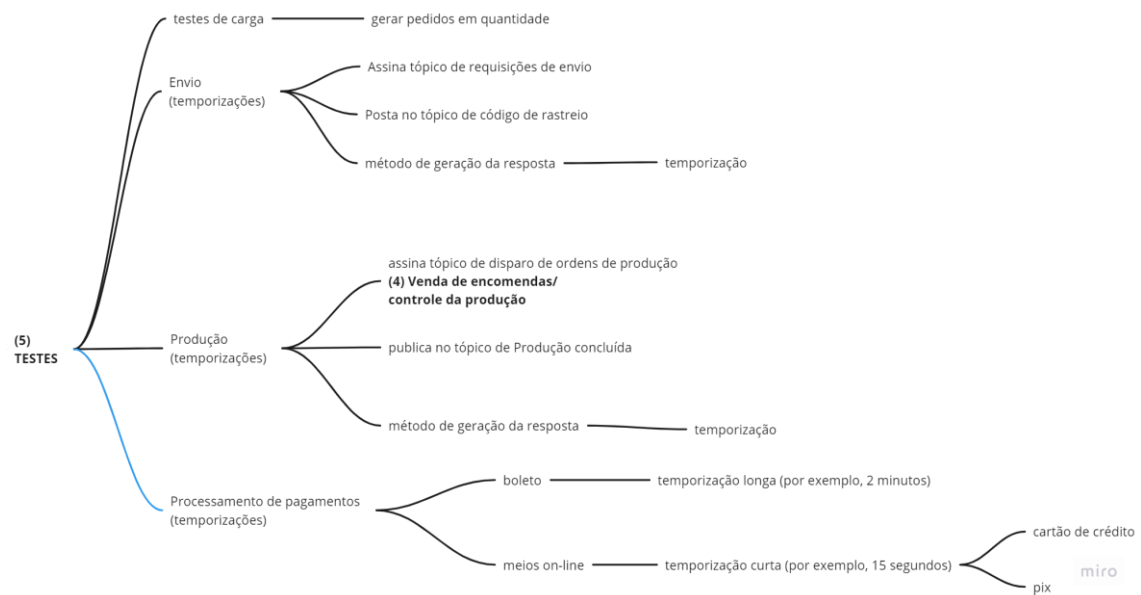


miro

Venda de encomendas



Testes



Banco de dados no PostgreSQL

O banco de dados está implantado no PostgreSQL, com os seguintes dados de acesso:

- dbname: dbecommerce_n9hn
- user: dbecommerce_n9hn_user
- password: 0Scqh1LQR7sKG1EYNBaBmx4VPWIrvtF
- host: dpq-co7upesf7o1s738n3u5g-a.oregon-postgres.render.com



A partir do dia 22 de abril:

1. A tabela **produto** passa a se chamar **produtos**.
2. As tabelas do banco de dados do e-commerce estão definidas com nomes de colunas mais amigáveis à programação, conforme indicado a seguir.

As definições das tabelas ficaram assim (chaves primárias realçadas em amarelo):

Nome da tabela: **clientes**

Nome da coluna no PostgreSQL	Tipo no PostgreSQL	Descrição dos dados
cpf_cli	bigint	CPF do cliente, formato inteiro longo. Chave primária.
nome_cli	character varying (100)	Nome do cliente
email_cli	character varying (50)	endereço de e-mail do cliente
celular_cli	character varying (15)	número de celular do cliente
end_res_cli	character varying (250)	endereço residencial do cliente
senha_cli	character varying (15)	senha do cliente na loja
data_cadastro_cli	date	data de cadastro do cliente, formato AAAA-MM-DD
niver_cli	date	aniversário do cliente (opcional), formato AAAA-MM-DD
trusted_cli	boolean	score do cliente: true = cliente OK, false = cliente com problema comercial (CPF falso ou compra anterior com problema)

Nome da tabela: **produtos**

Nome da coluna no PostgreSQL	Tipo no PostgreSQL	Descrição dos dados
id_prod	character varying (15)	ID Produto
nome_prod	character varying (100)	Nome "amigável"
tipo_prod	character varying (15)	Tipo (mercadoria adquirida de 3os/encomenda)
descr_prod	character varying (250)	Descrição
valor_venda_prod	numeric	Valor de venda
qtd_estoque_prod	integer	Quantidade em estoque
custo_prod	numeric	Custo

Nome da tabela: **pedidos**

Nome da coluna no PostgreSQL	Tipo no PostgreSQL	Descrição dos dados
id_ped	integer	ID do pedido. Chave primária.
data_ped	date	Data do pedido
CPF_cli_ped	bigint	Cliente que fez o pedido, identificado pelo CPF
id_car_ped	integer	Carrinho de compras deste pedido
valor_ped	numeric	Valor total do pedido
transp_ped	character varying (100)	Transportadora do pedido
rastreio_ped	character varying (100)	Código de rastreamento do pedido
end_entrega_ped	character varying (250)	Endereço de entrega do pedido

Nome da tabela: **carrinhos**

Nome da coluna no PostgreSQL	Tipo no PostgreSQL	Descrição dos dados
id_car	integer	ID do carrinho de compras. Chave primária.
prod_car	integer	Produto no carrinho; um registro para cada produto
qtd_prod_car	integer	Quantidade do produto; um registro para cada produto
desconto_car	numeric	Desconto aplicável ao carrinho

O arquivo Excel com as definições das colunas está atualizado no GitHub.

Por favor, atualizem os nomes das colunas nos seus códigos.

⚠ Os privilégios do banco de dados foram alterados para permitir que todos possam fazer tudo:



Restrições: O banco de dados foi criado no PostgreSQL usando Python com a biblioteca **psycopg**. Caso seja necessário redefinir as tabelas, basta executar o notebook **GerenciaBD_Python.ipynb**.