Relatório Técnico - Especificações do Projeto

StoreGO

Unidade Curricular:	IES - Introdução à Engenharia de Software
Data:	Aveiro, 11 de janeiro de 2021
Estudantes:	93346: Alexandra de Carvalho 92972: Gonçalo Matos 93195: Hugo Almeida 91322: Isadora Loredo
Resumo:	Sistema de monitorização de clientes e produtos de uma loja física e automatização de compras

1 Introdução

Um dos principais objetivos do projeto consiste na exploração de conceitos ligados à monitorização e controlo de sistemas automatizados.

Para isto, serão aplicadas práticas de Engenharia de Software que vão desde a especificação de um produto até à implementação final do mesmo, aplicando práticas de trabalho colaborativo utilizando ferramentas como o *Github* para gestão e armazenamento de código e *Jira* para gestão de tarefas entre a equipa.

Será, ainda, abordada a construção de uma arquitetura que vá de encontro aos requisitos do projeto, sendo discutida a escolha de cada componente.

2 Conceito do produto

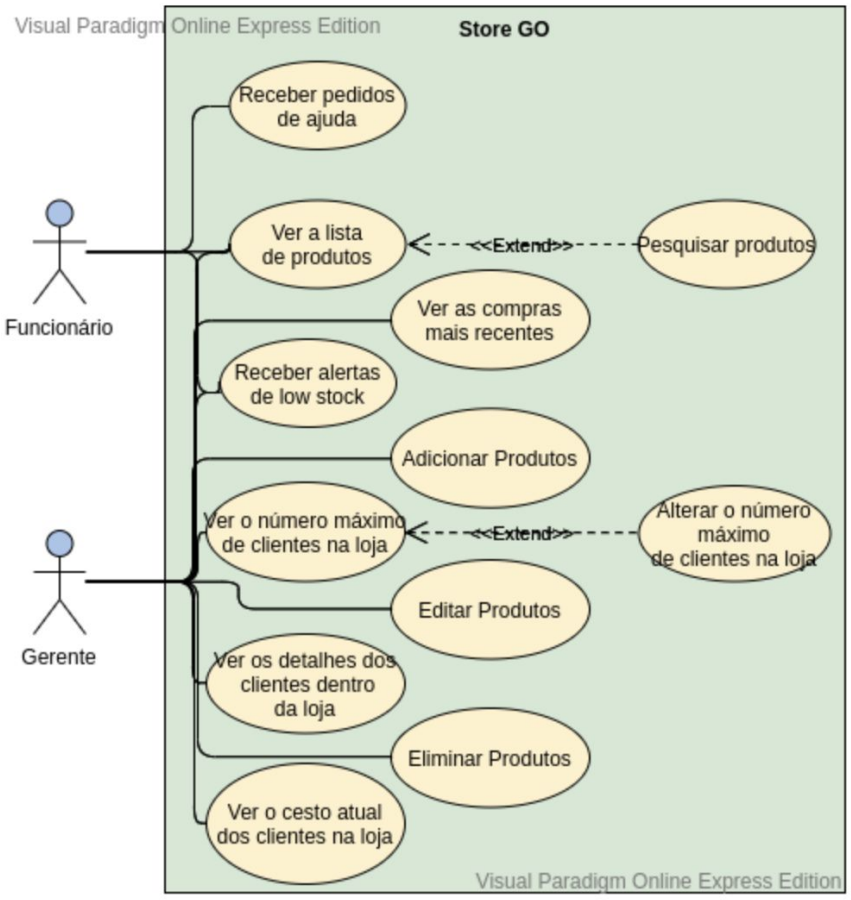
Visão

A aplicação tem a finalidade de gerir uma loja automatizada, ou seja, um estabelecimento que proporciona uma experiência de compras sem caixas de pagamento, ao reconhecer os produtos que um cliente retirou da prateleira e efetuar a cobrança no momento em que ele deixa a loja.

Ao chegar ao supermercado é detetada a sua entrada na loja e através dos sensores distribuídos pelo espaço são adicionados produtos ao seu carrinho virtual ou removidos se este os voltar a pousar. Terminadas as compras, não é preciso fazer nada: apenas sair da loja. Uma vez do lado de fora da loja a compra é finalizada.

Esta aplicação é similar ao supermercado inteligente da Amazon com o conceito “Just Walk Out“, sem filas e sem *checkouts*.

Tudo isto será visualizado através da aplicação *web* para Funcionários e Gerentes, cada um com diferentes permissões e ações possíveis:



Personas

Foram definidos dois atores principais do sistema:

Gerente

- Amélia Rodrigues tem 32 anos e é **gerente** da StoreGO

Funcionário

- Pedro Paulo tem 26 anos e é **funcionário** da StoreGO da Amélia

Cenários Principais

Monitorização dos clientes atuais na Loja

Um **Gerente** da loja, através da aplicação *web* vai conseguir visualizar em tempo real as informações relativas aos **clientes que se encontram na loja** bem como os seus **cestos de compras**. Além disso, este poderá limitar o número de clientes em simultâneo dentro da loja, sendo criada automaticamente uma fila de espera quando este número é ultrapassado.

Notificações de alterações de estado

O sistema, consoante a ocorrência de eventos (por exemplo limite de pessoas atingido, falta de *stock* e pedidos de ajuda de clientes) irá **notificar os seus atores**. Caso seja um **Gerente**, este receberá em tempo real notificações relativas a falta de *Stock* e limite de pessoas atingido. Quanto ao **Funcionário**, este receberá apenas notificações de pedidos de ajuda provenientes dos clientes.

Um **Gerente** consegue ainda visualizar todo o histórico de notificações que o sistema gerou.

Monitorização de Produtos

É possível visualizar todos os produtos incluindo informações relativas ao *stock* atual, categoria, etc. O **Gerente** consegue editar, adicionar e remover produtos bem como dar ordem para repor *Stock*. O **Funcionário** apenas consegue visualizar a listagem de produtos e fazer pesquisas/filtros.

Alteração de Preferências

Tanto o **Funcionário** como o **Gerente** vão poder alterar informações relativas à sua conta e escolher que tipo de notificações pretendem receber dentro do tipo de notificações permitido.

Gestão de Pedidos de Ajuda

Um **Funcionário** terá acesso à lista de pedidos de ajuda dos clientes que se encontram pendentes, podendo alterar o seu estado indicando que o pedido foi resolvido.

Histórico de Compras

Como **Gerente**, é possível aceder a todo o histórico de compras realizadas na loja.

Gestão dos Clientes do Sistema

O **Gerente** tem permissões para aceder à listagem de clientes da loja e por sua vez, ao histórico de compras desses mesmos clientes.

3 Notas de Arquitetura

Principais requisitos e Restrições

A escolha da arquitetura centra-se no endereçamento das seguintes questões:

- O sistema deve ser capaz de gerar e consumir dados automaticamente, simulando o que aconteceria numa loja em funcionamento.
- Devido às interações que os utilizadores podem realizar, o script *python* necessita de consumir os novos dados atualizados, de modo a garantir a congruência dos dados gerados. Por exemplo, quando é alterado o limite de pessoas dentro da loja não faz sentido gerem geradas mais pessoas e novos eventos para essas.
- Deve-se garantir que os dados gerados pelo simulador não são misturados no *message broker* com os dados recebidos.
- O sistema não pode permitir acesso por terceiros a dados confidenciais, necessitando de um sistema de autenticação.
- O sistema estará alojado numa máquina virtual fornecida pelo Docente.

Vista da Arquitetura

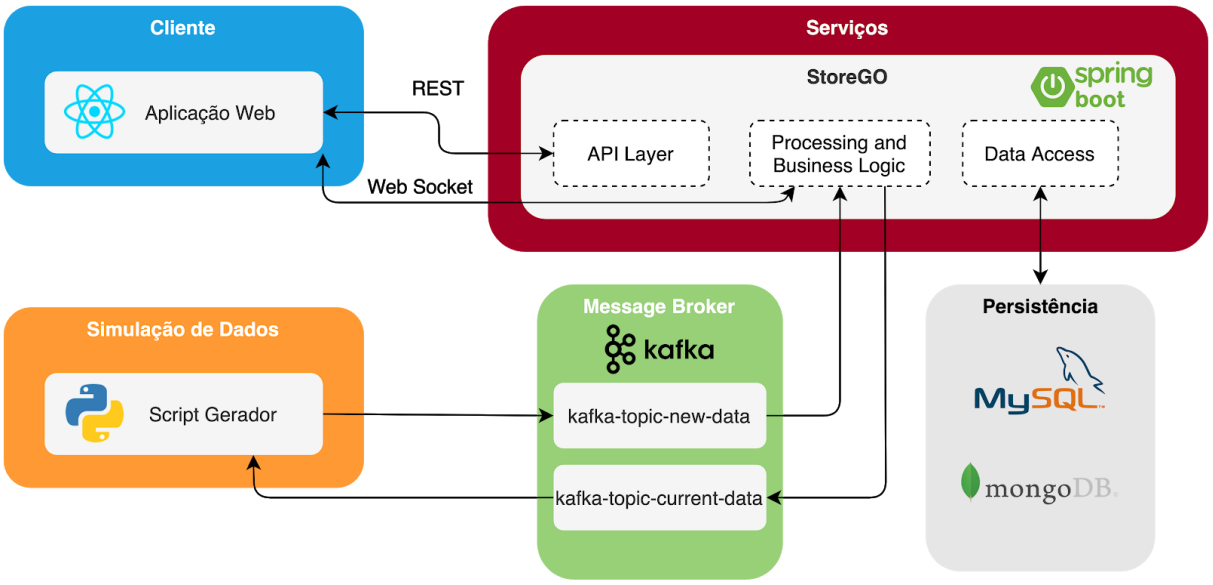
Aplicação Web: Escolheu-se a biblioteca de *JavaScript React* para construir a aplicação web do sistema devido às vantagens de desenvolvimento que esta apresenta face ao desenvolvimento de aplicações dinâmicas.

StoreGO: Será construído um serviço baseado em *Spring Boot* que será o ponto essencial do sistema. Este será composto por uma camada *REST API*, que permitirá consultar diversos dados do sistema para serem posteriormente apresentados na aplicação *web*. Existe ainda a camada do modelo de dados, que permite o mapeamento dos dados armazenados na base de dados para objetos *Java* a serem tratados e processados na camada de lógica de negócio.

Simulação de Dados: Foi definido que os dados consumidos pelo sistema seriam gerados através de um script em Python, que consumirá as novas atualizações de dados, essenciais ao seu funcionamento, através de um tópico do *Message Broker.* Os dados gerados serão publicados num outro tópico *Kafka* de modo a serem enviados para o sistema.

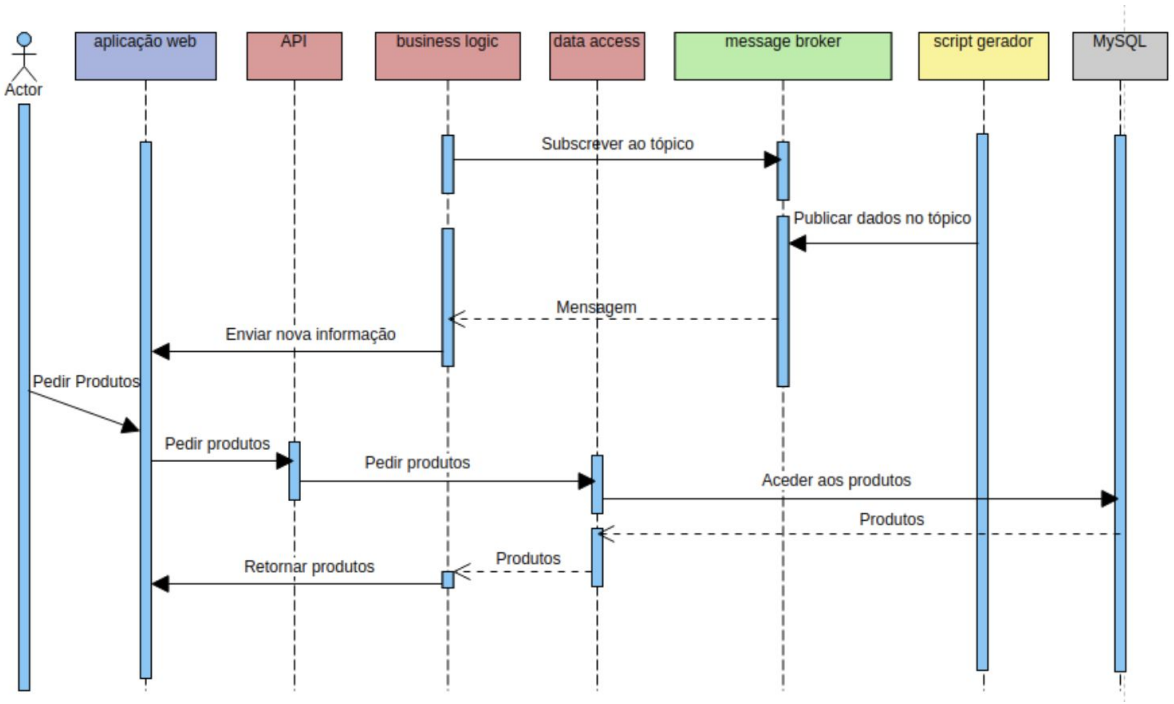
Message Broker: Foi escolhido *Kafka* como *message broker* uma vez que permite um sistema rápido e escalável de produtores-consumidores com um grande volume de dados. Existirão dois tópicos com diferentes objetivos. Um deles servirá para enviar dados que foram gerados de modo a serem guardados e processados pelo sistema. O outro servirá para atualizar os dados do gerador visto que ao longo do tempo poderão existir dados essenciais à geração que sofrerão alterações por parte dos utilizadores.

Base de Dados: Utilizado MySQL para armazenar de forma relacional os principais dados do sistema. Será também utilizado mongoDB para armazenar informações como logs de notificações que não necessitam de estar armazenados de forma relacional.

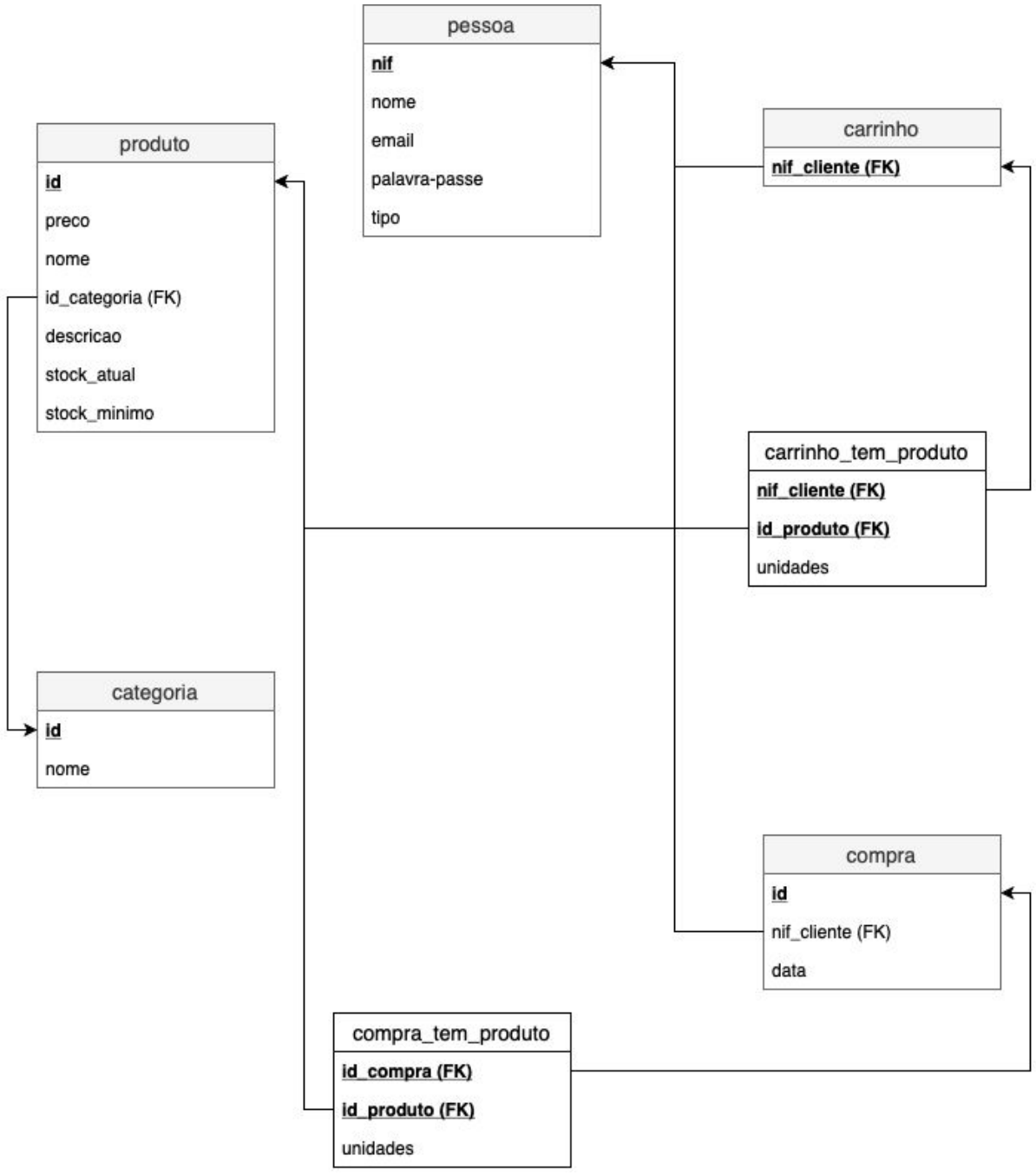


Exemplo de Interação entre Módulos

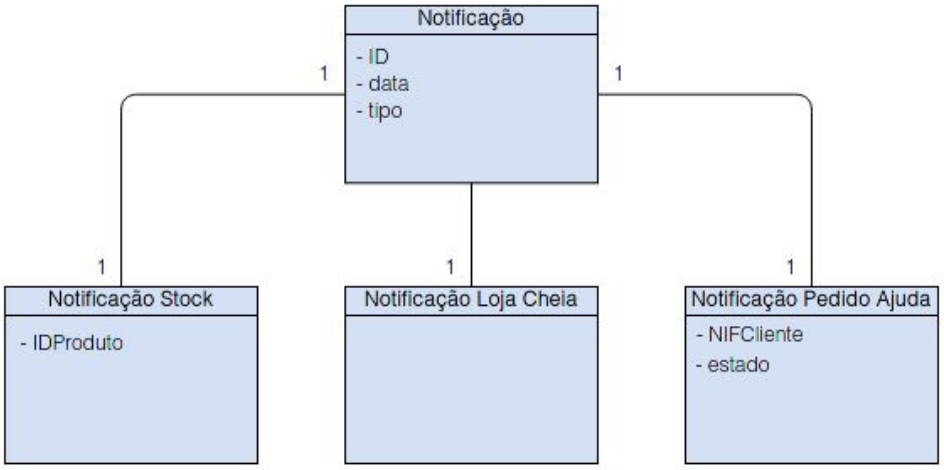
1. Os dados são gerados através do *Script* gerador, sendo publicados num tópico de *Kafka*
2. Estes dados são consumidos pela lógica de negócio implementada em *Spring Boot,* armazenando-os na base de dados
3. Através da ligação *Web Socket*, são enviadas à aplicação web novas informações ou ativadas notificações (por exemplo, limite de clientes na loja atingido)
4. O cliente através da aplicação *web* vê, por exemplo, os produtos disponíveis, sendo feito um pedido à *REST-API* para este propósito
5. Através da *REST-API* são enviados os dados pedidos, com um acesso à base de dados e passagem pela lógica de negócio
6. O gerente através da aplicação *web* dá ordem de reposição de stock, sendo enviada essa informação à API através de um pedido POST
7. Ao receber este pedido, as informações são alteradas na base de dados e é publicado no tópico correto os novos dados atualizados
8. O *Script Gerador* ao consumir estes dados através desse tópico, irá gerar novos dados de acordo, mantendo a congruência


Actor
aplicação web
API
business logic
data access
message broker
script gerador
MySQL
Subscrever ao tópico
Publicar dados no tópico
Mensagem
Enviar nova informação
Pedir Produtos
Pedir produtos
Pedir produtos
Aceder aos produtos
Produtos
Produtos
Retornar produtos

4 Perspetiva da Informação



MONGO-DB



5 Referências e Recursos

https://www.baeldung.com/spring-kafka