DOCUMENTAÇÃO DA SOLUÇÃO PROPOSTA

Lucas Raphael Fernandes Ferreira

lucasraphael.fernandes@hotmail.com

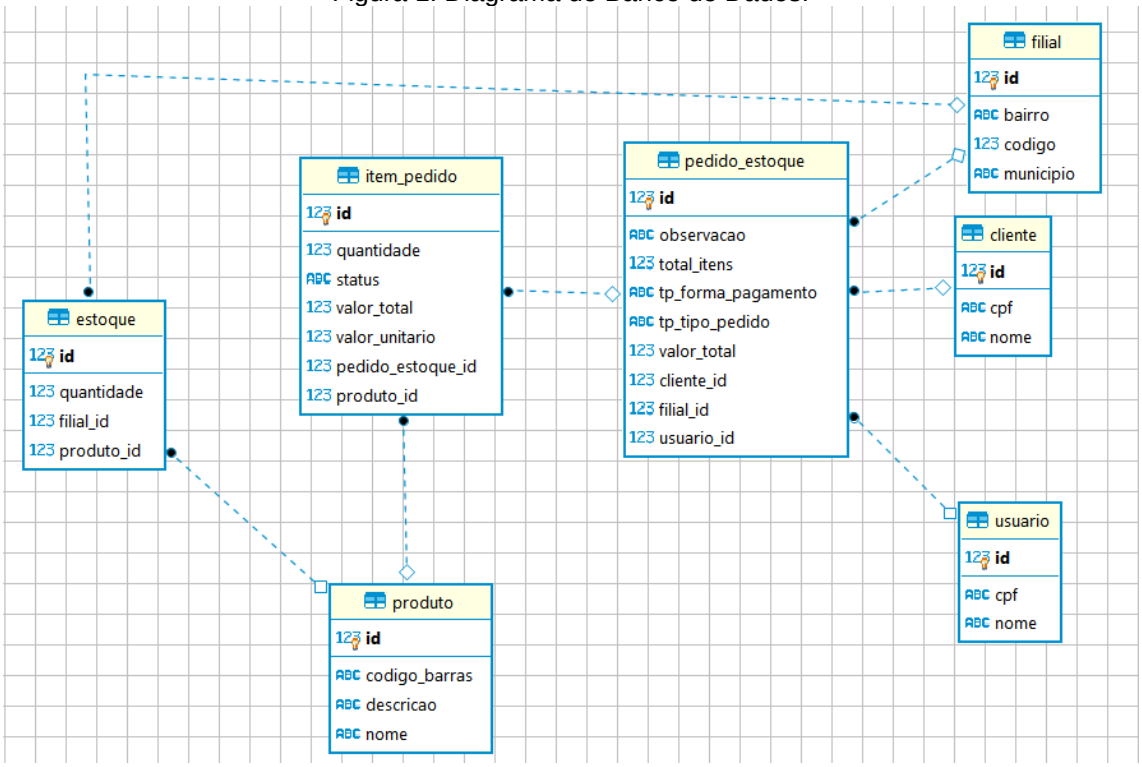
1. COMPETÊNCIA DE BANCO DE DADOS

O banco de dados foi projetado de forma a atender os requisitos apresentados no contexto do negócio. Dessa forma, cada um dos domínios apresentados foi representado da melhor maneira identificada pelo autor deste projeto. Foi utilizado um banco de dados H2 para o desenvolvimento, desta forma, ao executar a aplicação, o mesmo estará disponível através da url "http://localhost:8080/h2". Os parâmetros para o acesso são:

- URL: jdbc:h2:file:~/h2db/ithappens;
- User Name: admin;
- Password: admin.

1.1 Diagrama

Figura 1: Diagrama de Banco de Dados.



Fonte: Autor.

1.2 Consultas

Neste tópico estão exibidas as consultas que foram solicitadas no desafio.

- Consulta que retorne todos os produtos com quantidade maior ou igual a 100:

```
SELECT * FROM PRODUTO
JOIN ESTOQUE ON ESTOQUE.PRODUTO_ID = PRODUTO.ID
WHERE ESTOQUE.QUANTIDADE >= 100;
```

- Consulta que traga todos os produtos que têm estoque para a filial de código 60:

```
SELECT * FROM PRODUTO
JOIN ESTOQUE ON ESTOQUE.PRODUTO_ID = PRODUTO.ID
JOIN FILIAL ON FILIAL.ID = ESTOQUE.FILIAL_ID
WHERE FILIAL.CODIGO = 60
```

- Consulta que liste todos os campos para o domínio PedidoEstoque e ItensPedido filtrando apenas o produto de código 7993:

```
SELECT * FROM PEDIDO_ESTOQUE JOIN ITEM_PEDIDO ON
ITEM_PEDIDO.PEDIDO_ESTOQUE_ID = PEDIDO_ESTOQUE.ID
WHERE ITEM_PEDIDO.PRODUTO_ID = 7993
```

- Consulta que liste os pedidos com suas respectivas formas de pagamento:

```
SELECT * FROM PEDIDO_ESTOQUE
```

- Consulta para sumarizar e bater os valores da capa do pedido com os valores dos itens de pedido:

  *SELECT PE.ID, PE.VALOR_TOTAL AS VALOR_TOTAL_PEDIDO, IP.VALOR AS VALOR_ITEM_SOMADO FROM PEDIDO_ESTOQUE PE JOIN (SELECT PEDIDO_ESTOQUE_ID, SUM(QUANTIDADE * VALOR_UNITARIO) VALOR FROM ITEM_PEDIDO GROUP BY PEDIDO_ESTOQUE_ID) IP ON IP.PEDIDO_ESTOQUE_ID = PE.ID;*

- Consulta para sumarizar o total dos itens por pedido e que filtre apenas os pedidos no qual a soma total da quantidade de itens de pedido seja maior que 10;

  SELECT PE.TOTAL_ITENS, IP.QUANTIDADE AS QUANTIDADE_TOTAL_ITENS FROM PEDIDO_ESTOQUE PE JOIN (SELECT PEDIDO_ESTOQUE_ID, SUM(QUANTIDADE) AS QUANTIDADE FROM ITEM_PEDIDO GROUP BY PEDIDO_ESTOQUE_ID) AS IP ON IP.PEDIDO_ESTOQUE_ID = PE.ID WHERE IP.QUANTIDADE > 10;

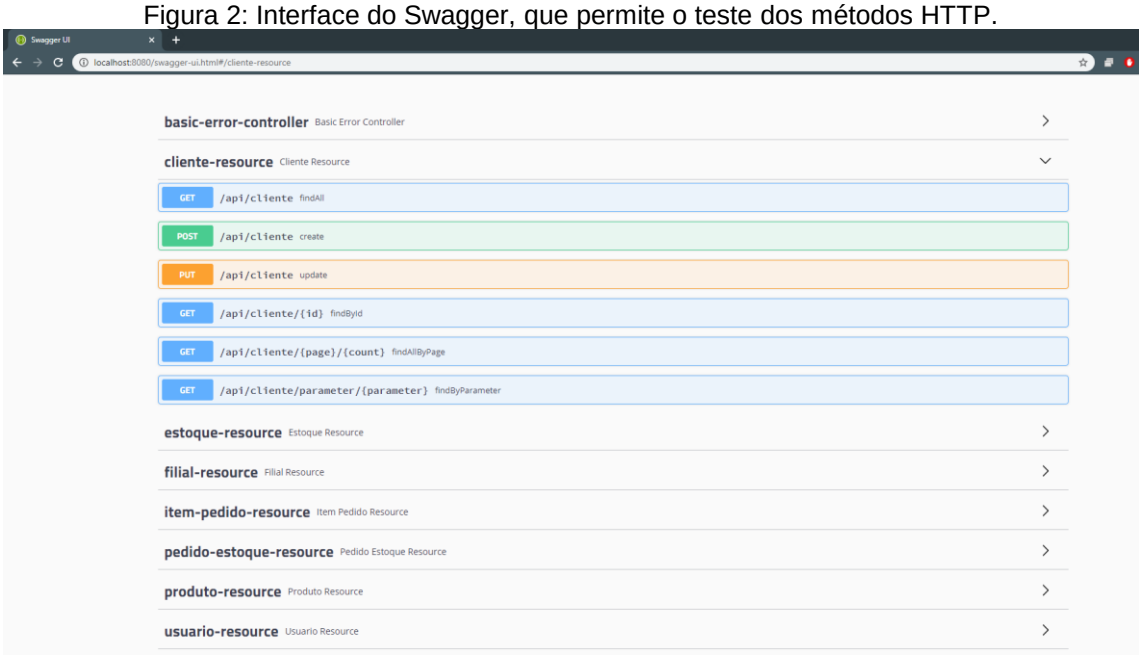
2. COMPETÊNCIA DE DESENVOLVIMENTO

O código foi feito em Java, utilizando a tecnologia Spring Boot para desenvolver uma API que disponibiliza os recursos para o funcionamento da aplicação. Dessa forma, é possível acessar os métodos da aplicação via HTTP Requests.

Para facilitar seu uso, foi utilizado o framework Swagger, que auxilia na documentação dos métodos HTTP, tornando mais prático seu entendimento, as entradas necessárias e também consultar o resultado.

O Swagger também disponibiliza uma interface gráfica para teste dos métodos, que são acessíveis através do caminho "http://localhost:8080/swagger-ui.html", ficando disponível ao executar a aplicação.

Figura 2: Interface do Swagger, que permite o teste dos métodos HTTP.



Fonte: Autor.

Através da imagem, podemos identificar os domínios da aplicação separadamente. Por exemplo, o domínio CLIENTE é representado pelo módulo cliente-resource e assim por diante.

3. FLUXO DA APLICAÇÃO SEGUINDO O CONTEXTO DO NEGÓCIO

Seguindo o contexto do negócio, apresentado na descrição do desafio, as etapas necessárias para a utilização correta da aplicação são:

- Criar uma filial: Através do domínio filial-resource – CREATE;
- Criar um usuário: Através do domínio usuario-resource – CREATE;
- Criar um cliente: Através do domínio cliente-resource – CREATE;
- Criar uma venda: Através do domínio pedido-estoque-resource – CREATE, informando o cliente*, a filial*, o usuario*, tipo de pedido (TpTipoPedido) como “SAIDA”. Não é necessário informar os itens, nem o valor total ou mesmo a quantidade, tendo em vista que conforme serão adicionados os itens posteriormente, esses valores vão sendo atualizados conforme solicitado na descrição do desafio. OBS*: Para

informar os campos cliente, filial e usuário basta informar o ID de cada um deles;

- Criar um Produto: Através do domínio produto-resource – CREATE;
- Criar um Estoque: Através do domínio estoque-resource – CREATE. No caso de venda de produtos, é necessário informar a quantidade em estoque antes de adicionar os Itens ao Pedido de VENDA. Caso seja um pedido do tipo ENTRADA, um estoque não é necessário, pois ao processar o pedido o estoque será criado com a quantidade informada no pedido;
- Adicionar Item a um pedido: Através do domínio pedido-estoque-resource – ADD ITEM, informando um ItemPedido (previamente cadastrado ou não). Para isso, também é necessário informar a quantidade de itens e o valor unitário, para que seja calculado o valor total do ItemPedido e atualizado o valor do PedidoEstoque, bem como a quantidade de itens do pedido. OBS: A quantidade mínima de produtos é 1, e a máxima, no caso de VENDA, é o valor em estoque do produto para aquela filial;
- Processar um Pedido: Através do domínio pedido-estoque-resource – PROCESSAR, informando o ID do PedidoEstoque e a forma de pagamento no formato de enum (“A_VISTA”, “BOLETO” ou “CARTAO”). Feito isso, as quantidades do estoque serão atualizadas, bem como os status dos itens do pedido, que de “ATIVO” serão agora “PROCESSADOS”;

Também estão disponíveis as opções:

- Remover ItemPedido: Através do domínio pedido-estoque-resource – REMOVER ITEM, informando o ID do ItemPedido. Neste caso, o status do ItemPedido será atualizado para CANCELADO, os valores totais do pedido e quantidade serão atualizados e não levarão em conta os itens com esse status. Ao processar este PedidoEstoque, os estoques também não serão influenciados por estes;
- Diversas outras funcionalidades: Outras funcionalidades também foram criadas na API como atualizar recursos, listar, listar de forma paginada,

dentre outras funções. Estas não estão definidas no escopo do desafio, mas foram adicionadas para atender as necessidades em possíveis cenários.

4. INICIALIZAÇÃO VIA DOCKER

Foi utilizado o Docker para a disponibilização da aplicação. Dessa forma, alguns procedimentos são necessários:

- Gerar o .jar da aplicação: No diretório “implementação/teste-ithappens” executar o comando “mvn package –DskipTests”. OBS: Durante o envio do teste para o GitHub foi efetuado o build para a disponibilização do arquivo .jar para que este processo não seja necessário;
- Iniciar o container: no diretório “implementação” existe um arquivo docker-compose.yml. Através deste arquivo é possível iniciar o container. Para isso, executar o comando “docker-compose up --build –d”;

Importante ressaltar que para o desenvolvimento da aplicação, foi utilizado um banco em memória H2. Dessa forma, ao finalizar a execução do container, os dados são perdidos.