Back-End em Spring Boot para um Serviço de Estoque

Paulo Weskley de Almeida Ferreira

1. **Introdução**

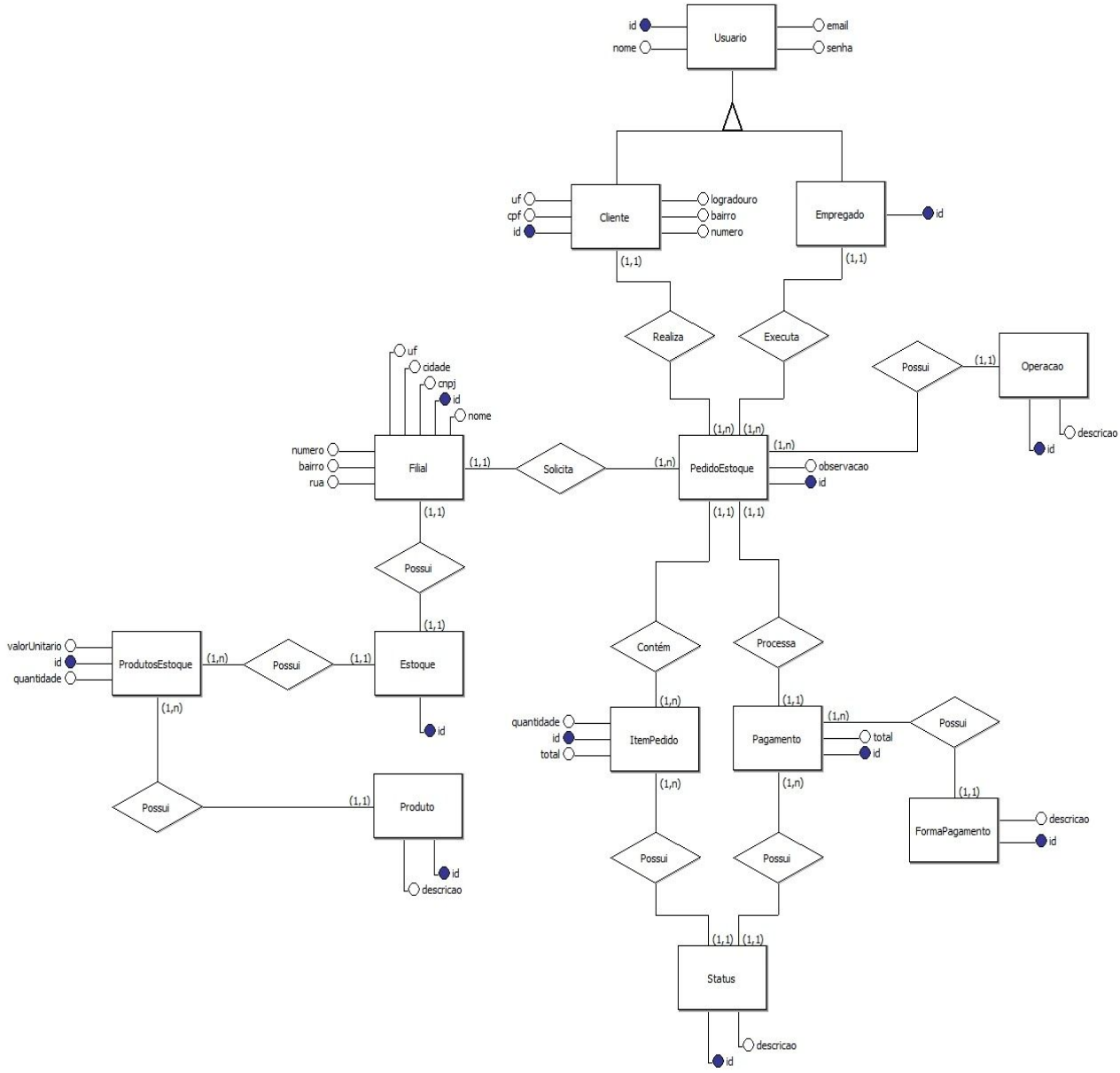
   A aplicação desenvolvida tem como intuito ser um serviço para sistemas de gestão de estoque, provendo informações necessárias para seus usuários realizar operações em uma loja.

2. **Contextualização**

   Este webservice foi pensado de acordo com a perspectiva do vendedor, onde ele pode atender um cliente e realizar a venda de seu produto ou realizar algum tipo de consulta no sistema para poder responder a algo para terceiros ou conhecimento próprio.

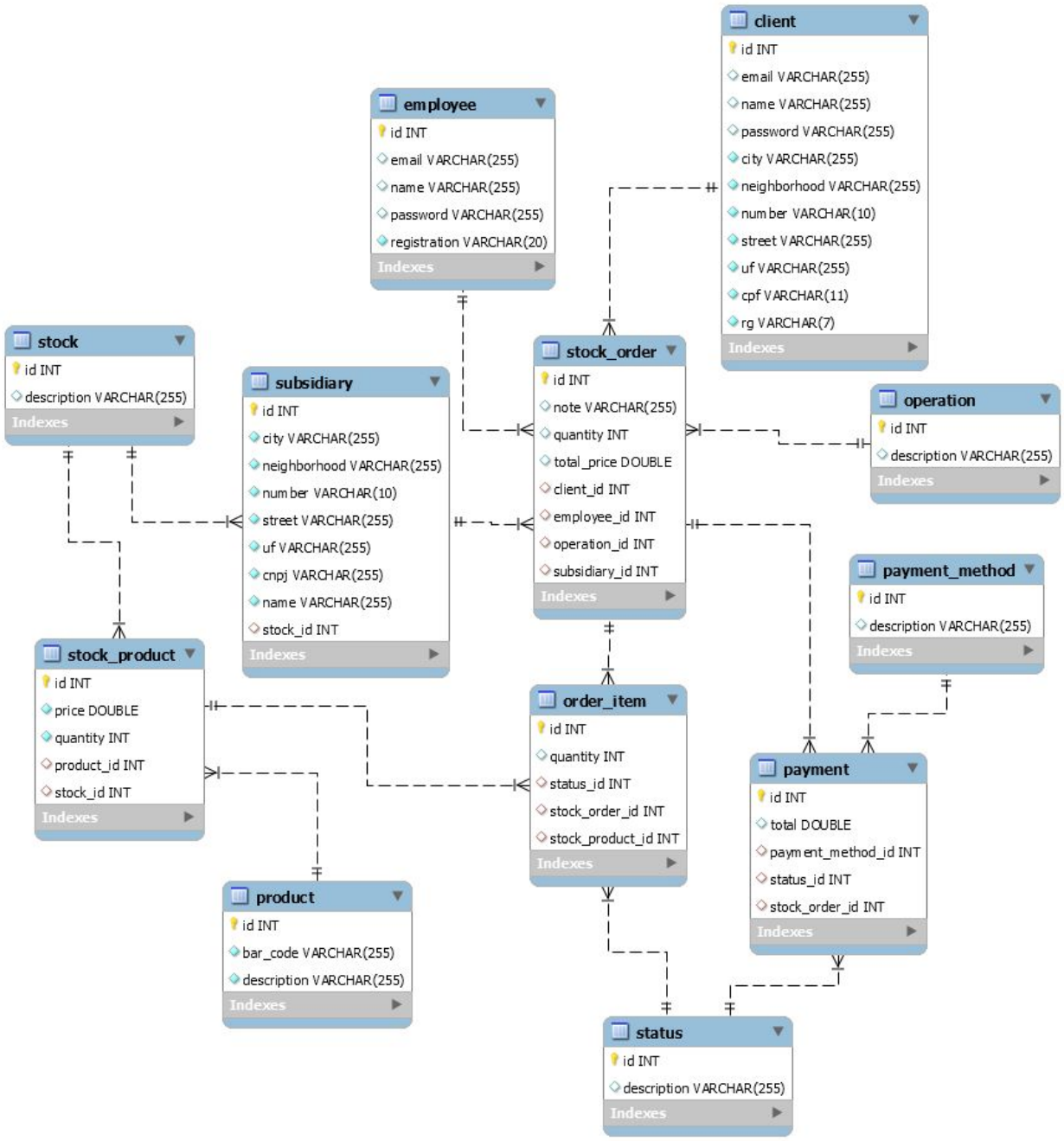
3. **Modelo Entidade-Relacionamento**

   Abaixo segue o modelo conceitual criado para servir de base para a criação do banco de dados:


Usuario
id
nome
email
senha
Cliente
uf
cpf
id
logradouro
bairro
numero
Empregado
id
(1,1)
(1,1)
Realiza
Executa
Possui
(1,1)
Operacao
descricao
id
Filial
uf
cidade
cnpj
id
nome
numero
bairro
rua
(1,1)
Solicita
(1,n)
PedidoEstoque
(1,n)
(1,n)
(1,n)
observacao
id
(1,1)
Possui
(1,1)
Estoque
id
(1,1)
Possui
(1,n)
ProdutosEstoque
valorUnitario
id
quantidade
(1,n)
Possui
(1,1)
Produto
id
descricao
(1,1)
(1,1)
Contém
Processa
(1,n)
(1,1)
ItemPedido
quantidade
id
total
Pagamento
(1,n)
total
id
Possui
(1,1)
FormaPagamento
descricao
id
(1,n)
(1,n)
Possui
Possui
(1,1)
(1,1)
Status
descricao
id

4. Diagrama Entidade-Relacionamento

Abaixo segue o diagrama do relacionamento entre as tabelas:



5. Tecnologias e Padrões Utilizados

Para o desenvolvimento da aplicação foi utilizado Java e o framework SpringBoot. O banco de dados utilizado foi o H2, que é um banco em memória e só existe no tempo de execução da aplicação. A aplicação é voltada a arquitetura REST, servindo como back-end para aplicações de front-end que venham a utilizar o serviço. No projeto para receber as informações via POST em muitos casos optou-se por utilizar o Padrão DTO, evitando o tráfego desnecessário de informações e otimizando o serviço. Foi utilizado também o padrão MVC com as camadas de Model, Repository, Service e Resources.

6. Funcionalidades

Foram implementadas diversas funcionalidades por meio do uso de API REST, dentre algumas, podemos citar:

6.1. Produtos:

- Listar todos;
- Consultar um produto específico pelo seu id ou código de barras;
- Incluir um novo produto na tabela de modelo para Produtos;
- Editar ou Excluir um produto existente.
-

6.2. Produtos no Estoque

- Colocar um novo produto no estoque;
- Editar um produto existente;
- Listar produtos por Estoque;
- Editar o preço ou quantidade de um produto;
- Excluir um produto do estoque;
-

6.3. Quanto às informações do estoque só é possível:

- Listar os estoques ou pesquisar algum pelo seu id;

6.4. Pedido de Estoque

- Lista todos os pedidos e por id;
- Cria um novo pedido no banco (Apenas de SAIDA);
- Exclui um pedido existente.

6.5. Pagamento

- Lista todos e por id;
- Cria novo pagamento;
- Exclui um existente;

6.6. Itens do Pedido

- Lista todos e por id;
- Cria um novo item para um pedido de estoque;
- Exlclui um pedido existente pelo seu id;
- Edita a quantidade ou o status do item;

6.7. Cliente

- Lista todos os clientes cadastrados;
- Pesquisa por id, rg ou cpf um cliente;
- Salva um novo cliente informado;
- Edita ou Exclui um cliente;

6.8. Funcionário

Lista todos os funcionarios cadastrados;

- Pesquisa por id ou matricula;
- Salva um novo funcionario informado;
- Edita ou Exclui um funcionario;

6.9. Classes de suporte

As classes de Operação, Status e Tipo de pagamento possuem apenas recursos para Listar todos os objetos do tipo escolhido ou filtrar um pelo id;

7. Scripts

Do total de 6 scripts foram criados apenas 5. Ficou faltando um entendimento melhor do que estava sendo pedido na penúltima query.

1. Produtos com quant. ≥ 100

```
SELECT * FROM STOCK_PRODUCT SP WHERE SP.QUANTITY >= 100;
```

2. Produtos para a filial 60

```
SELECT * FROM STOCK_PRODUCT SP JOIN SUBSIDIARY SUB ON
(SUB.STOCK_ID = SP.STOCK_ID AND [SUB.ID](http://sub.id/) = 60);
```

3. Todos os campos de item pedido e pedido estoque filtrando pelo produto de id 7993

```
SELECT * FROM ORDER_ITEM OI JOIN STOCK_ORDER SO ON
(OI.STOCK_ORDER_ID = [SO.ID](http://so.id/) AND OI.STOCK_PRODUCT_ID =
7993);
```

4. Todos os pedidos e formas de pagamento

```
SELECT * FROM STOCK_ORDER SO JOIN PAYMENT PAY ON ([SO.ID](http://so.id/) = PAY.STOCK_ORDER_ID) JOIN PAYMENT_METHOD PM ON (PAY.PAYMENT_METHOD_ID = [PM.ID](http://pm.id/));
```

5. Quantidade total de itens por pedido, para os itens que tenham quantidade maior que 10

```
SELECT OI.STOCK_PRODUCT_ID, SUM(OI.QUANTITY) AS TOTAL FROM STOCK_ORDER SO JOIN ORDER_ITEM OI ON (SO.ID = OI.STOCK_ORDER_ID) GROUP BY OI.STOCK_PRODUCT_ID HAVING SUM(OI.QUANTITY) > 10;
```