Documentação do Desafio Técnico

Foi criado um projeto Restful de nominado “apiVenda” utilizando o framework Spring Boot e o banco de dados Postgres denominado, também, de “apiVenda”.

1. Modelo de Entidade e Relacionamento (MER)

Após análise do desafio foi elaborado o seguinte modelo:

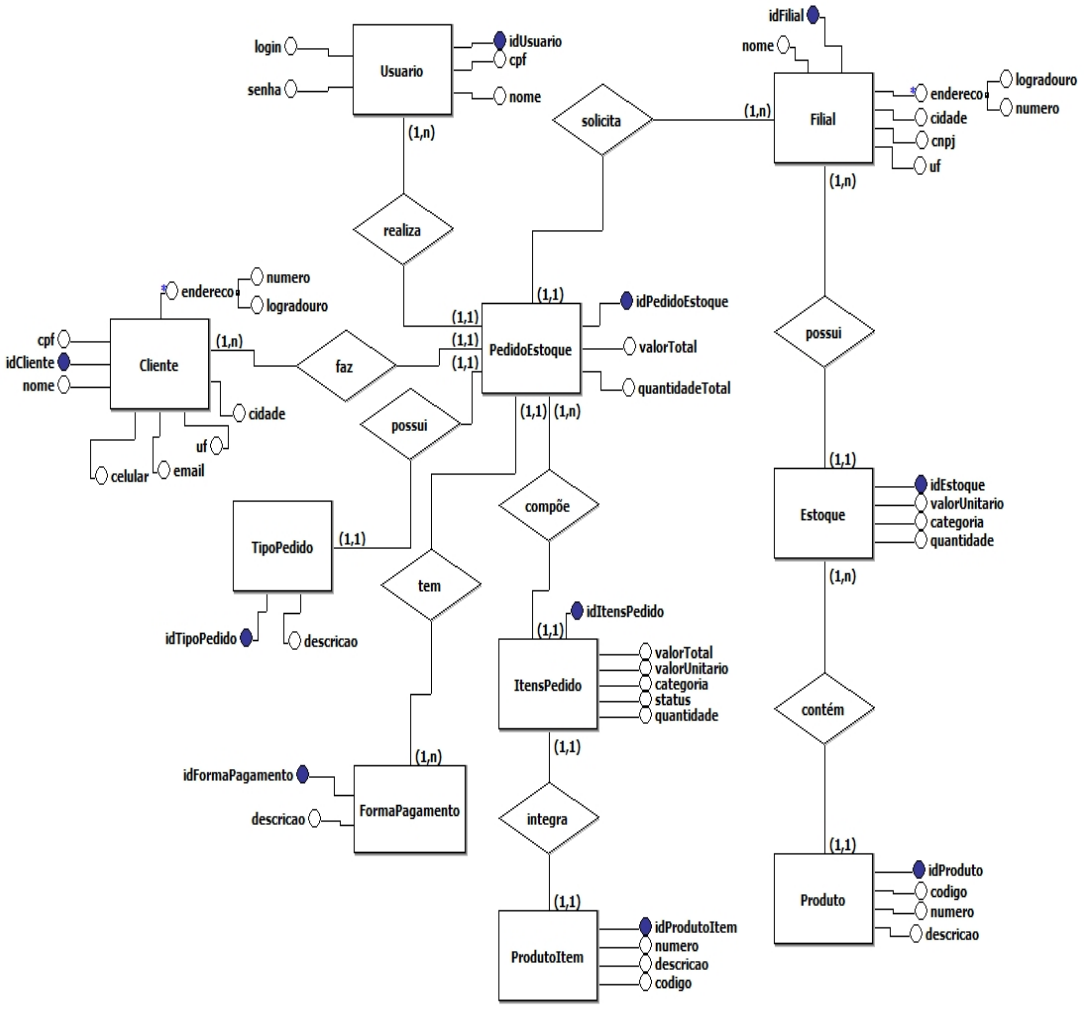

Figura 1: MER apiVenda

2. Modelagem do Banco de Dados

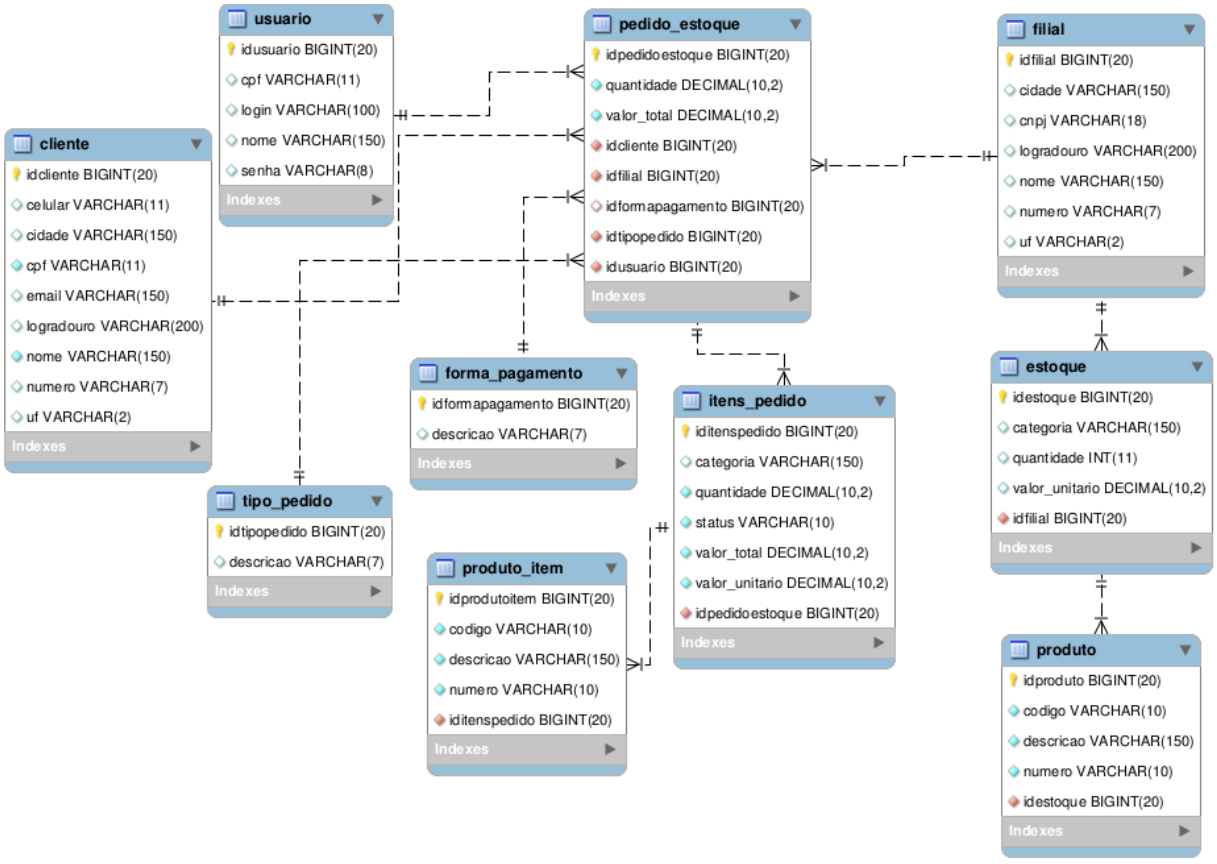

Figura 2: Modelo do Banco apiVenda

3. Script para estartar o banco de dados com os dados iniciais

O Script encontra-se na pasta “Documentação” juntamente com as imagens do MER e Modelagem do Banco de dados.

Para usar o script basta criar no postgres uma base de dados com o nome de “apiVenda” e fazer o restore com o script, pois o mesmo já traz consigo um conjunto de dados mínimos nas tabelas, com o intúito de agilizar o uso e consequente avaliação do projeto.

4. Bases JSON para Testes

Aqui coloquei alguns formados Json base de exemplo, que facilitarão a inserção e atualização de dados para teste em suas respectivas entidades no modelo elaborado e em uma sequência lógica que não acarretará em erros devido as regras denegócio impostas no texto do desefio.

Para testar o sistema utilizou-se no projeto a ferramenta “SWAGGER UI” (https://swagger.io/), no intuito de facilitar a avaliação. Para ter acesso a ferramenta, basta subir o projeto e no browser acessar o endereço: **http://localhost:8080/swagger-ui.html** e utiliza-lo, pois o mesmo é bastante intuítivo eprático.

========== Cadastros Básicos ==========
1. Cadastro de Usuário (usuario-resource)

```
{
  "nome": "Fulano da Silva",
  "login": "fulano",
  "senha": "123456",
  "cpf": "12345678909"
}
```

---
2. Cadastro de Cliente (cliente-resource)

```
{
  "nome": "Ciclano Oliveira",
  "cpf": "23412565476",
  "email": "fulano@gmail.com",
  "celular": "91987654355",
  "cidade": "Belém",
  "uf": "PA",
  "logradouro": "Cidade Nova 5 we-24",
  "numero": "302"
}
```

---
3. Cadastro de Forma de Pagamento (forma-pagamento-resource)

```
{
  "descricao": "Cartão"
}
```

---
4. Cadastro de Tipo de Pedido (tipo-pedido-resource)

```
{
  "descricao": "Entrada"
}
```

---
5. Cadastro de Filial (filial-resource)

```
{
  "nome": "Supermercado Mateus 1",
  "cidade": "Belém",
  "uf": "PA",
  "cnpj": "70250917000120",
  "logradouro": "Av. Presidente Vargas",
  "numero": "1000"
}
```

---
6. Cadsatro de Estoque (estoque-resource)

```
{
  "categoria": "Celular",
  "valorUnitario": 0,
  "quantidade": 0,
  "filial": {
    "idFilial": 1
  }
}
```

================== Realizar Pedido de Estoque ==========================
1. Abrir Pedido de Estoque (pedido-estoque-resource)

```
{
  "usuario": {
    "idUsuario": 1
  },
  "cliente": {
    "idCliente": 1
  },
  "filial": {
    "idFilial": 1
  },
  "tipoPedido": {
    "idTipoPedido": 1
  },
  "quantidade": 0,
  "valorTotal": 0
}
```

---
2. Inserir Itens do Pedido (itens-pedido-resource)

```
{
  "categoria": "Celular",
  "quantidade": 10,
  "valorUnitario": 2000,
  "pedidoEstoque": {
    "idPedidoEstoque": 1
  },
  "produtoItem": {
    "codigo": "123456",
    "numero": "654321",
    "descricao": "Ifone 5"
  }
}
```

5. Consultas SQL

- Escrever uma consulta que retorne todos os produtos com quantidade maior ou igual a 100

Resp.:

```
SELECT * FROM produto p
INNER JOIN estoque s ON s.idestoque = p.idestoque
WHERE s.quantidade >= 150.00
```

- Escrever uma consulta que traga todos os produtos que têm estoque para a filial de código 60

Resp.:

```
SELECT * FROM produto p
INNER JOIN estoque s ON s.idestoque = p.idestoque
AND s.idfilial = 60
WHERE s.quantidade > 0
```

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

Resp.:

```
SELECT  p.* , i.* FROM pedido_estoque p
INNER JOIN itens_pedido i ON i.idpedidoestoque = p. idpedidoestoque
INNER JOIN produto_item pi ON pi.iditenspedido = i. iditenspedido
AND pi.idprodutoitem = 7993
```

- Escrever uma consulta que liste os pedidos com suas respectivas formas de pagamento.

Resp.:

```
SELECT p.*, fp.* FROM pedido_estoque p
INNER JOIN forma_pagamento fp ON fp.idformapagamento = p. idformapagamento
```

- Escrever um consulta para sumarizar e bater os valores da capa do pedido com os valores dos ítens de pedido

Resp.:

```
SELECT p.*, SUM(i.valor_total) as total FROM pedido_estoque p
INNER JOIN itens_pedido i ON i.idpedidoestoque = p.idpedidoestoque
  GROUP BY p.idpedidoestoque HAVING SUM(i.valor_total) = p.valor_total
```

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

Resp.:

```
SELECT p.*, SUM(i.valor_total) as total, SUM(i.quantidade) as soma
FROM pedido_estoque p
INNER JOIN itens_pedido i ON i.idpedidoestoque = p. idpedidoestoque
GROUP BY p.idpedidoestoque HAVING SUM(i.quantidade ) > 10
```