Sistema de Vendas em API com Laravel

Você foi contratado para desenvolver uma **API de Sistema de Vendas** utilizando o framework **Laravel**. O sistema deve permitir o cadastro e a manipulação de **clientes**, **produtos**, **vendas** e **itens de vendas**. A API deverá ter os seguintes requisitos e funcionalidades:

1. Criação das Models e Migrations

Para organizar os dados e o banco de dados, você deve criar as seguintes models e migrations:

Cliente

Crie uma **model** Cliente com os seguintes campos:

- **nome** (string)
- **email** (string, único)
- **telefone** (string)
- **endereco** (string)

Implemente a migration correspondente para a criação da tabela **clientes** no banco de dados.

Produto

Crie uma **model** Produto com os seguintes campos:

- **nome** (string)
- **codigo** (inteiro único)
- **preco** (decimal)
- **quantidade_estoque** (inteiro)

Implemente a migration correspondente para a criação da tabela **produtos** no banco de dados.

Venda

Crie uma **model** Venda com os seguintes campos:

- **cliente_id** (inteiro, chave estrangeira associada ao cliente)
- **data_venda** (timestamp)
- **subtotal** (decimal)
- **desconto** (decimal, valor do desconto)
- **total** (decimal, valor final após o desconto)

Implemente a migration correspondente para a criação da tabela **vendas** no banco de dados.

ItemVenda

Crie uma **model** ItemVenda com os seguintes campos:

- **venda_id** (inteiro, chave estrangeira associada à venda)
- **produto_id** (inteiro, chave estrangeira associada ao produto)
- **quantidade** (inteiro)
- **preco_unitario** (decimal)
- **subtotal_item** (decimal, valor total do item, que é quantidade * preco_unitario)

Implemente a migration correspondente para a criação da tabela **itens_venda** no banco de dados.

2. Criação dos Controllers

Crie os seguintes **controllers** para manipular as requisições da API:

ClienteController

- **Função store:** Para criar um novo cliente.
- **Função index:** Para listar todos os clientes cadastrados.
- **Função show:** Para exibir os dados de um cliente específico.
- **Função update:** Para atualizar os dados de um cliente.
- **Função destroy:** Para deletar um cliente.

ProdutoController

- **Função store:** Para criar um novo produto.

- **Função index:** Para listar todos os produtos cadastrados.
- **Função show:** Para exibir os dados de um produto específico.
- **Função update:** Para atualizar os dados de um produto.
- **Função destroy:** Para deletar um produto.

VendaController

- **Função store:** Para registrar uma nova venda.
    - Receberá um **cliente_id** e um array de **itens** (com os dados do produto, quantidade e preço).
    - Calculará o **subtotal** (somando os subtotais dos itens), o **desconto** (caso seja fornecido) e o **total** (aplicando o desconto, se houver).
    - Deverá atualizar o estoque dos produtos (diminuindo a quantidade de cada produto vendido).
- **Função index:** Para listar todas as vendas realizadas.
- **Função show:** Para exibir os detalhes de uma venda específica, incluindo os itens de venda.
- **Função update:** Para atualizar os dados de uma venda (caso necessário).
- **Função destroy:** Para deletar uma venda.

ItemVendaController

- **Função store:** Para registrar um item na venda.
    - Receberá o **venda_id**, o **produto_id**, a **quantidade** e o **preço_unitario** do produto.
    - Calculará o **subtotal_item** e o associará à venda.
- **Função index:** Para listar todos os itens de vendas registrados.
- **Função show:** Para exibir os detalhes de um item de venda específico.
- **Função update:** Para atualizar os dados de um item de venda.
- **Função destroy:** Para deletar um item de venda.

3. Funcionalidades Específicas

- **Cálculo de Subtotal e Total:**
    - Quando uma venda for registrada, o **subtotal** será calculado somando os subtotais de todos os itens da venda.

    - O **total** da venda deve ser calculado levando em consideração um possível **desconto**. Caso o desconto seja percentual, aplique o desconto sobre o subtotal. Caso o desconto seja fixo, subtraia o valor diretamente.
- **Controle de Estoque:**
    - A cada item vendido, o estoque do **produto** correspondente deverá ser atualizado automaticamente, diminuindo a quantidade em estoque.

4. Endpoints da API

Implemente os seguintes **endpoints** para expor a API:

Clientes

- POST /clientes: Criação de um novo cliente.
- GET /clientes: Listar todos os clientes.
- GET /clientes/{id}: Exibir um cliente específico.
- PUT /clientes/{id}: Atualizar os dados de um cliente.
- DELETE /clientes/{id}: Deletar um cliente.

Produtos

- POST /produtos: Criar um novo produto.
- GET /produtos: Listar todos os produtos.
- GET /produtos/{id}: Exibir um produto específico.
- PUT /produtos/{id}: Atualizar os dados de um produto.
- DELETE /produtos/{id}: Deletar um produto.

Vendas

- POST /vendas: Criar uma nova venda.
- GET /vendas: Listar todas as vendas.
- GET /vendas/{id}: Exibir uma venda específica.
- PUT /vendas/{id}: Atualizar uma venda (se necessário).
- DELETE /vendas/{id}: Deletar uma venda.

Itens de Venda

- POST /itens-venda: Criar um novo item na venda.
- GET /itens-venda: Listar todos os itens de venda.
- GET /itens-venda/{id}: Exibir um item de venda específico.
- PUT /itens-venda/{id}: Atualizar um item de venda.
- DELETE /itens-venda/{id}: Deletar um item de venda.

Requisitos Técnicos

1. **Validações:** A API deve realizar as validações necessárias:
    - Garantir que os campos obrigatórios estejam presentes.
    - Verificar se a **quantidade em estoque** é suficiente antes de registrar a venda.
    - Validar o formato do **email** e outras entradas.