UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CAMPUS PAU DOS FERROS
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA

ALAN ALMEIDA DA SILVA - 2024011434
DALTON FIRMINO CAMPOS - 2024011414
IVERTON EMIQUISON RIBEIRO DE BESSA - 20204011408
PAULINA JÚLIA COSTA DE OLIVEIRA - 2023023644

SISTEMA DE GERENCIAMENTO DE MERCADO

PAU DOS FERROS – RN
2024

Introdução

Este documento tem como objetivo apresentar a estrutura e a funcionalidade do sistema de gerenciamento de produtos desenvolvido em Django. O sistema foi projetado para facilitar a administração de produtos, controle de estoque e gerenciamento de vendas, permitindo uma gestão eficiente e organizada.

1. Produto

A classe **Produto** representa um produto disponível no sistema de gerenciamento. Ela armazena informações relacionadas ao produto, como nome, código, descrição, valor de venda e detalhes adicionais.

I. Campos:

- **nome** (CharField):
- **codigo** (CharField):
- **descricao** (TextField):
- **valor_venda** (DecimalField):
- **detalhes** (TextField):

II. Métodos:

- **total_estoque** (@property):
  - **Descrição**: Calcula o total de itens em estoque para o produto.
  - **Comportamento**: O total é calculado somando todas as movimentações de estoque associadas ao produto. Retorna 0 se não houver movimentações.
- **str()**:
  - **Descrição**: Retorna uma representação em string do produto, utilizando o nome do produto.

2. Estoque

A classe **Estoque** representa a movimentação de estoque para um produto. Ela armazena informações sobre a entrada e saída de produtos, incluindo a data da movimentação, peso, valor de compra, quantidade e data de vencimento.

I. Campos:

- **produto** (ForeignKey):
- **data_movimentacao** (DateTimeField):
- **peso** (DecimalField):
- **valor_compra** (DecimalField):
- **quantidade** (IntegerField):
- **vencimento** (DateField):

II. Métodos:

- **clean()**:
  - **Descrição**: Valida os dados antes de salvar, garantindo que a quantidade de itens no estoque não seja negativa.
  - **Comportamento**: Lança uma ValidationError se a quantidade for negativa.
- **str()**:
  - **Descrição**: Retorna uma string que representa a movimentação de estoque, incluindo o nome do produto e a quantidade

3. **Caixa e Gerente**

As classes **Caixa** e **Gerente** estendem a classe AbstractUser, representando um usuário no sistema com funcionalidades adicionais de grupo e permissões. Essas classes são usadas para gerenciar usuários que têm acesso a funcionalidades específicas do sistema.

I. Campos:
- **groups** (ManyToManyField):
- **user_permissions** (ManyToManyField):

4. **Venda**

A classe **Venda** representa uma transação de venda no sistema, permitindo registrar informações sobre o vendedor, o valor total da venda e a data da transação. A classe utiliza campos genéricos para vincular o vendedor, permitindo que diferentes tipos de usuários possam ser associados à venda.

I. Campos:
- **vendedor_content_type** (ForeignKey):
  - **Descrição**: O tipo de conteúdo do vendedor (referência ao modelo do vendedor).
- **vendedor_object_id** (PositiveIntegerField):
  - **Descrição**: O ID do objeto do vendedor, que será usado junto com o tipo de conteúdo para identificar o vendedor.
- **vendedor** (GenericForeignKey):
  - **Descrição**: Um campo genérico que permite referenciar diferentes modelos de vendedor.
  - **Relacionamento**: Combinado com vendedor_content_type e vendedor_object_id.
- **valor_total** (DecimalField):
- **data_venda** (DateTimeField):

5. **ItemVenda**

A classe **ItemVenda** representa um item específico que foi vendido em uma transação de venda. Cada item está associado a uma venda e a um produto, armazenando informações sobre a quantidade e o valor unitário do produto vendido.

I. Campos:
   - **venda** (ForeignKey):
   - **produto** (ForeignKey):
   - **quantidade** (PositiveIntegerField):
   - **valor_unitario** (DecimalField):

6. **signals**

Os **signals** são usados para executar ações automaticamente em resposta a eventos específicos no modelo ItemVenda, como a criação, atualização ou exclusão de itens de venda.

I. Verificar Estoque Antes da Venda
   - **Descrição**: Este sinal é disparado antes de salvar um objeto ItemVenda. Ele verifica se há estoque suficiente do produto para a quantidade que está sendo vendida.
   - **Comportamento**:
     - Obtém o produto da instância do ItemVenda.
     - Verifica o total de estoque disponível.
     - Se a quantidade solicitada exceder o estoque disponível, lança uma ValueError, informando a quantidade disponível e a quantidade solicitada.
     - Atualiza a quantidade no estoque correspondente ao produto vendido, excluindo ou ajustando as movimentações de estoque conforme necessário.

II. Definir Valor Total da Venda Após Salvar um Item
   - **Descrição**: Este sinal é disparado após salvar um objeto ItemVenda. Ele recalcula o valor total da venda à qual o item pertence.
   - **Comportamento**:

- Chama a função handle_venda_changes, que calcula o valor total da venda somando o valor de todos os itens associados à venda e atualiza o campo valor_total da venda.

III. Definir Valor Total da Venda Após Deletar um Item

- **Descrição**: Este sinal é disparado após excluir um objeto ItemVenda. Ele recalcula o valor total da venda e reverte a quantidade no estoque.
- **Comportamento**:
  - Chama a função handle_venda_changes para atualizar o valor total da venda.
  - Reverte a quantidade do produto no estoque correspondente ao item vendido. Se não houver movimentação de estoque, cria uma nova entrada de estoque para o produto com a quantidade retornada.