UNIVERSIDADE CIDADE DE SÃO PAULO
LABORATÓRIO DE BANCO DE DADOS AVANÇADO

BRUNO VINICIUS CHAGAS DE MEDEIROS COSTA
KAYKE GONÇALVES DE SOUSA
KAUÃ LOURENÇO DA SILVA
LUIZ HENRIQUE
NINA AREAL CEZARIO
RYAHN PAULO SOBRAL JESUS
STENIO LUCAS DA SILVA

DESENVOLVIMENTO E MODELAGEM DE APLICAÇÃO: DOGSTOCK

SÃO PAULO
2025

BRUNO VINICIUS CHAGAS DE MEDEIROS COSTA
KAYKE GONÇALVES DE SOUSA
KAUÃ LOURENÇO DA SILVA
LUIZ HENRIQUE
NINA AREAL CEZARIO
RYAHN PAULO SOBRAL JESUS
STENIO LUCAS DA SILVA

DESENVOLVIMENTO E MODELAGEM DE APLICAÇÃO: DOGSTOCK

Trabalho de desenvolvimento e modelagem de aplicação em formato web de gestão de estoque, com uso de APIs e integração com Power BI.

SÃO PAULO

2025

SUMÁRIO

RESUMO

O projeto consiste na modelagem e construção de um aplicação web para registros e movimentação de estoque, capaz de persistir dado de produtos, fornecedores, movimentações (e seus tipos, como saída e entrada etc.), utilizando APIs em Python para o backend, HTML, CSS E JavaScript para o frontend, e banco de dados MySQL hospedado em serviços de nuvem como Amazon Web Services (AWS). Por fim, esses dados serão apresentados de forma estratégica via PowerBI.

1. OBJETIVO

O objetivo central da aplicação DogStock é permitir ao usuário visualizar e registrar a suas movimentações de estoque, de forma a tomar decisões estratégicas com base nas vendas e movimentações, receber alertas quando seus produtos estão com a quantidade abaixo do limite pré-definido, e definir pontos de ação com a integração com PowerBI.

Desta forma, será possível analisar qual o fornecedor com maior custo-benefício, quais os produtos mais vendidos e outros tipos de insights de dados valiosos para a gestão de qualquer empresa, sem restrição de ramo de atividade.

2. ESCOPO DA APLICAÇÃO

Backend	Python + FastAPI
Frontend	HTML, CSS e Javascript
Banco de Dados	Banco de dados MySQL hospedado em AWS
Visualização estratégica de dados	PowerBI
Repositório e controle de versão	Git + Github

O banco de dados hospedado em AWS será alterado através de requisições via API Python (FastAPI), com os dados de acesso armazenados em arquivos “dotenv”, visando políticas de segurança da informação. A escolha de serviços cloud para host do banco visa facilitar o trabalho de integração com PowerBI, uma vez que se utilizado bancos de dados local, seria necessário a troca constante dos arquivos a cada alteração, o que poderia atrasar o desenvolvimento.

O frontend, responsável por apresentar ao usuário a interface, fará uso de HTML, CSS e JavaScript para realizar as requisições na API hospedada localmente ou em servidor Linux.

2.1 Orçamento do Projeto

O projeto será orçado em custo zero, visto que tanto Amazon Web Services quanto outros serviços de Cloud (tal como Azure, da Microsoft e Google Cloud) possuem contas do tipo “free-tier”, com créditos em U$D 200 para cobrir os custos da aplicação. Visto que o projeto não será utilizado comercialmente e o tráfego será pouco/mínimo, os créditos serão mais que suficientes. O backend e frontend poderá também ser hospedado localmente ou via instâncias de servidores EC2 Linux (Ubuntu), em serviços de Cloud como os mencionados anteriormente.

2.2 Stakeholders

Dentre os stakeholders envolvidos no projeto, incluem-se os

1. *fornecedores;*
2. *clientes;*
3. *usuário gerente da aplicação*.

3. REGRAS DE NEGÓCIO – CONTROLE DE ESTOQUE

ID	Módulo / Seção	Requisito	Descrição / Regra de Negócio
RN-01	Categorias	Evitar duplicidade de categorias	O sistema não deve permitir o cadastro de duas categorias com a mesma descrição.
RN-02	Produtos	Associação obrigatória a uma categoria	Todo produto deve estar vinculado a uma categoria previamente cadastrada.
RN-03	Produtos	Descrição obrigatória	Não é permitido cadastrar produtos sem uma descrição detalhada.
RN-04	Produtos	Estoque não pode ser negativo	O sistema deve impedir saídas que resultem em quantidade negativa em estoque.
RN-05	Produtos	Restrição de exclusão	Um produto só pode ser excluído se não houver movimentações associadas.
RN-06	Produtos e Fornecedores	Vinculação obrigatória a fornecedor	Todo produto deve estar associado a pelo menos um fornecedor cadastrado.
RN-07	Produtos e Fornecedores	Múltiplos fornecedores permitidos	Um produto pode ter mais de um fornecedor vinculado simultaneamente.
RN-08	Movimentações	Atualização de estoque em entradas	Ao registrar uma movimentação de entrada, o sistema deve somar a quantidade ao estoque atual do produto.
RN-09	Movimentações	Atualização de estoque em saídas	Ao registrar uma movimentação de saída, o sistema deve subtrair a quantidade, desde que haja saldo suficiente.
RN-10	Movimentações	Registro de data e hora	Toda movimentação deve conter data e hora exatas do registro.
RN-11	Movimentações	Vinculação obrigatória a tipo de movimentação	Cada movimentação deve estar associada a um tipo previamente definido no sistema.
RN-12	Tipos de Pagamento	Obrigatoriedade de tipo de pagamento nas vendas	Toda venda deve indicar o tipo de pagamento utilizado.
RN-13	Tipos de Pagamento	Permitir múltiplos tipos de pagamento	O sistema deve permitir mais de um tipo de pagamento na mesma venda.
RN-14	Tipos de Pagamento	Restrição de exclusão de tipos utilizados	Não é permitido excluir um tipo de pagamento que já tenha sido usado em movimentações.

4. REQUISITOS DO SISTEMA

4.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que o sistema DogStock deve oferecer para atender às necessidades operacionais. A Tabela 1 apresenta os requisitos funcionais organizados por categoria e prioridade de implementação.

Tabela 1 – Requisitos Funcionais do sistema DogStock

Código	Categoria	Descrição	Prioridade
RF-01	Cadastro	Permitir o cadastro de produtos, categorias e fornecedores.	Alta
RF-02	Estoque	Controlar entradas e saídas de estoque com atualização automática.	Alta
RF-03	Vendas	Registrar pedidos e associar múltiplos tipos de pagamento.	Alta
RF-04	Relatórios	Gerar relatórios de movimentações, produtos e vendas por período.	Média
RF-05	Acesso	Gerenciar perfis de usuário com permissões específicas (admin, operador, fornecedor).	Alta
RF-06	Exportação	Permitir exportação de dados em formato CSV.	Baixa
RF-07	Consulta	Consultar estoque com filtros por categoria e fornecedor.	Média
RF-08	Reposição	Solicitar e aprovar reposição de produtos.	Média
RF-09	Histórico	Exibir histórico de movimentações e vendas por cliente.	Média
RF-10	Exclusão Segura	Impedir exclusão de produtos com movimentações registradas.	Alta

4,2 Requisitos Não Funcionais

Os requisitos não funcionais definem restrições e qualidades que o sistema deve possuir, como desempenho, segurança, usabilidade e manutenibilidade. A Tabela 2 apresenta os requisitos não funcionais categorizados e priorizados.

Tabela 2 – Requisitos Não Funcionais do sistema DogStock

Código	Categoria	Descrição	Prioridade
RNF-01	Desempenho	O sistema deve processar consultas de estoque.	Alta
RNF-02	Desempenho	O sistema deve suportar até 50 transações de entrada/saída de estoque por minuto sem degradação significativa de desempenho.	Alta
Código	**Categoria**	**Descrição**	**Prioridade**
RNF-03	Usabilidade	A interface deve ser simples e intuitiva, permitindo que usuários com conhecimento básico realizem operações.	Alta
RNF-04	Usabilidade	O frontend, construído com HTML, CSS e JavaScript, deve ser responsivo, funcionando adequadamente em navegadores desktop e dispositivos móveis.	Alta
RNF-05	Usabilidade	Mensagens de erro devem ser claras, com sugestões de ações (ex.: "Código do produto inválido. Digite um código válido.").	Média
RNF-06	Segurança	Os dados do banco de dados (ex.: informações de produtos e usuários) devem ser protegidos.	Alta
RNF-07	Disponibilidade	Backups do banco de dados devem ser realizados manualmente ou automaticamente a cada semana, com tempo de restauração inferior a 2 horas.	Média
RNF-08	Manutenibilidade	O código Python deve seguir boas práticas para facilitar a manutenção por desenvolvedores futuros.	Média
RNF-09	Manutenibilidade	O frontend (HTML, CSS, JavaScript) deve ser organizado em arquivos separados, com comentários claros, para facilitar atualizações.	Média
RNF-10	Conformidade	O sistema deve permitir a exportação de dados de estoque em formato CSV para auditorias simples, quando necessário.	Baixa

5. CASOS DE USO

Os casos de uso representam os principais fluxos de interação entre os usuários e o sistema DogStock. A partir do documento técnico, foram identificados os seguintes atores e funcionalidades:

5.1 Atores

- **Usuário:** Realiza consultas de estoque e solicita reposições.
- **Administrador:** Aprova reposições e registra pedidos.
- **Fornecedor:** Envia produtos, registra recebimentos e entradas de mercadorias.

5.2 Funcionalidades por Ator

Tabela 3 – Casos de uso por ator

Ator	Caso de Uso	Descrição
Usuário	Consultar Estoque	Visualiza os produtos disponíveis com filtros por categoria e fornecedor.
Usuário	Solicitar Reposição	Solicita ao administrador a reposição de produtos com estoque mínimo.
Administrador	Aprovar Reposição	Valida e autoriza pedidos de reposição.
Administrador	Registrar Pedido	Formaliza pedidos de compra junto aos fornecedores.
Fornecedor	Enviar Produto	Realiza o envio físico dos produtos solicitados.
Fornecedor	Registrar Recebimento	Confirma o recebimento de pedidos e atualiza o status da entrega.
Fornecedor	Registrar Entrada de Produto	Atualiza o sistema com a entrada de novos produtos no estoque.

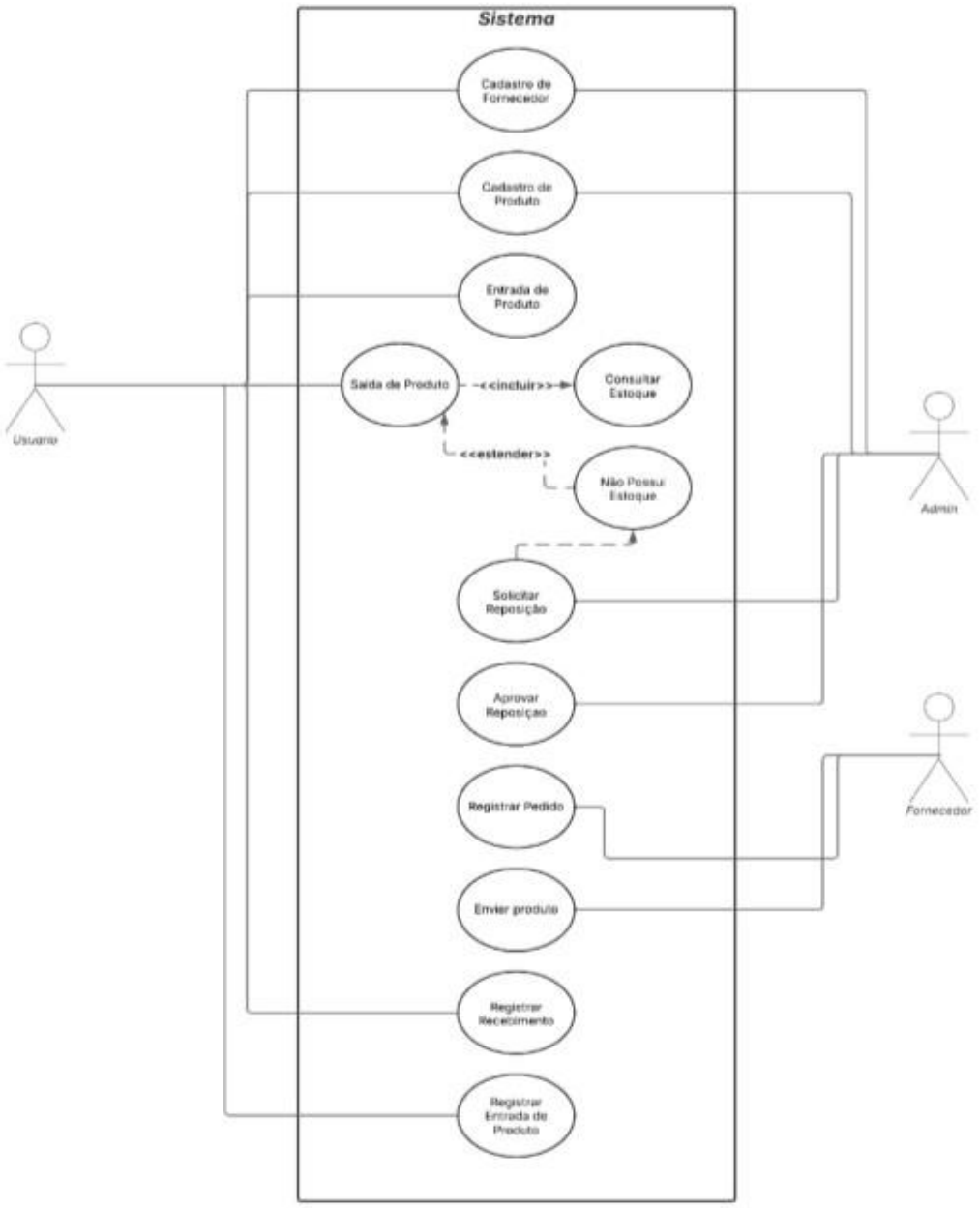

Figura 1 – Diagrama de Casos de Uso-Relacionamento do sistema DogStock (casosdeuso.png)

6. DIAGRAMA ENTIDADE - RELACIONAMENTO

O diagrama de entidades do sistema DogStock representa a estrutura relacional entre os principais componentes da base de dados. Ele é essencial para garantir integridade, consistência e rastreabilidade das informações.

6.1 Entidades Identificadas

Com base no documento **Diagrama Estoque.pdf**, foram identificadas as seguintes entidades:

- **Cliente:** Armazena dados de compradores.
- **Fornecedor:** Contém informações de empresas fornecedoras.
- **Produto:** Inclui descrição, preço unitário, quantidade em estoque e vínculo com fornecedor.
- **Pedido:** Representa a solicitação de compra, vinculada ao cliente e fornecedor.
- **Item_Pedido:** Detalha os produtos incluídos em cada pedido.
- **Movimentação_Estoque:** Registra entradas e saídas com data, tipo e quantidade.

6.2 Relacionamentos

- Um **cliente** pode realizar vários **pedidos**.
- Um **pedido** pode conter múltiplos **itens**.
- Cada **item_pedido** está vinculado a um **produto**. • Um **produto** pode ter múltiplas **movimentações**.

Um **fornecedor** pode fornecer vários **produtos**.

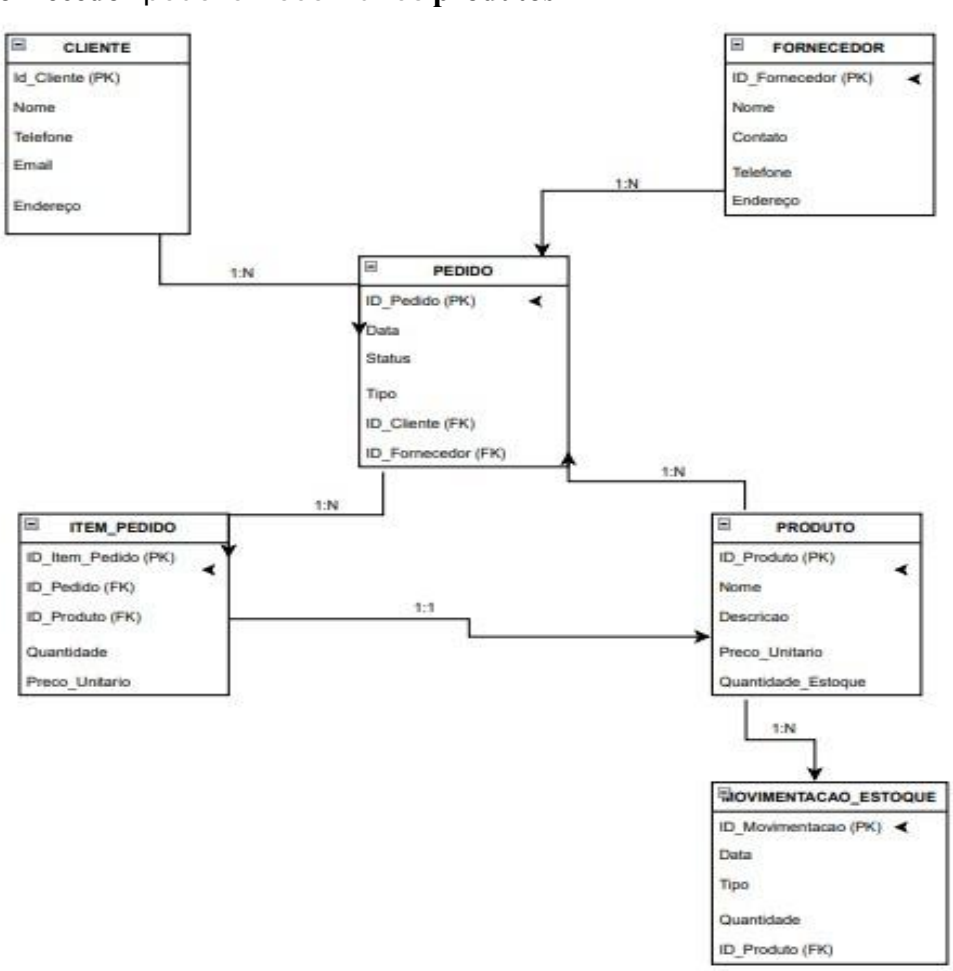

Figura 2 – Diagrama Entidade-Relacionamento do sistema DogStock (Inserir imagem do diagrama conforme o arquivo Diagrama Estoque.pdf)

8. DICIONÁRIO DE DADOS

TABELA	CATEGORIA			
DESCRIÇÃO	Tabela de referência que armazena os tipos de categoria de produto.			
CAMPOS				
NOME	DESCRIÇÃO	TIPO DE DADO	TAMANHO	Restrições de domínio (PK, FK, Not Null, Check, Default, Identity)
id	Código de identificação da categoria.		-	Primary Key / Identity
descricao	Nome descritivo da categoria.	VARCHAR	(100)	Not Null

TABELA	PRODUTOS			
DESCRIÇÃO	Tabela que armazena os produtos gerenciados pela aplicação.			
CAMPOS				
NOME	DESCRIÇÃO	TIPO DE DADO	TAMANHO	Restrições de domínio (PK, FK, Not Null, Check, Default, Identity)
id	Código de identificação da categoria.	INTEGER	-	Primary Key / Identity
nome	Nome do produto gerenciado.	VARCHAR	(100)	Not Null
medida	Medida do produto gerenciado (quilograma, metragem, unidade etc.).	VARCHAR	(100)	Not Null
qtd_disponivel	Quantidade de produtos disponíveis em estoque.	DECIMAL	(10, 3)	Not Null
qtd_minima	Quantidade mínima para alerta de falta de produtos.	DECIMAL	(10, 3)	Not Null
categoria_id	Chave estrangeira referenciando o id da tabela Categoria, referente a categoria dos produtos.	INTEGER	-	FK
status	Produto ativo ou inativo para operações.	VARCHAR	(100)	Not Null
fornecedor_id	Chave estrangeira referenciando a tabela fornecedor, referente ao ID do fornecedor.	INTEGER	-	FK

produto_id	Chave estrangeira referenciando a tabela produtos, referente ao ID do produto.	INTEGER	-	FK

TABELA	PRODUTOS_FORNECEDORES			
DESCRIÇÃO	Tabela que cruza produtos e fornecedores, permitindo que varios fornecedores sejam vinculados ao mesmo produto.			
CAMPOS				
NOME	DESCRIÇÃO	TIPO DE DADO	TAMANHO	Restrições de domínio (PK, FK, Not Null, Check, Default, Identity)
fornecedor_id	Chave estrangeira referenciando a tabela fornecedor, referente ao ID do fornecedor.	INTEGER	-	FK
produto_id	Chave estrangeira referenciando a tabela produtos, referente ao ID do produto.	INTEGER	-	FK

TABELA	MOVIMENTACOES			
DESCRIÇÃO	Tabela que registra movimentações de entrada ou saída e seus detalhes.			
CAMPOS				
NOME	DESCRIÇÃO	TIPO DE DADO	TAMANHO	Restrições de domínio (PK, FK, Not Null, Check, Default, Identity)
id	Código de identificação da movimentação.	INTEGER	-	Primary Key / Identity
produto_id	Chave estrangeira referenciando a tabela produtos, referente ao ID do produto.	INTEGER	-	FK
quantidade	Quantidade de itens movimentados.	NUMERIC	(10, 3)	Not Null
data	Data da movimentação.	TIMESTAMP		Not NuLL
tipo_mov_id	Chave estrangeira referenciando a tabela tipo_movimentacao, referente ao ID do tipo de movimentação.	INTEGER		Not Null

preco_venda	Preço de venda da movimentação.	NUMERIC	(12, 2)	Nullable
preco_compra	Preço de compra da movimentação.	NUMERIC	(12, 2)	Nullable
fornecedor_id	Chave estrangeira referenciando a tabela fornecedor, referente ao ID do fornecedor.	INTEGER		FK
tipo_pag_id	Chave estrangeira referenciando a tabela tipo_pagamento, referente ao ID do tipo de pagamento.	INTEGER		FK

TABELA	TIPO_PAGAMENTO			
DESCRIÇÃO	Tabela de referência para os tipos de pagamentos usados na movimentação.			
CAMPOS				
NOME	DESCRIÇÃO	TIPO DE DADO	TAMANHO	Restrições de domínio (PK, FK, Not Null, Check, Default, Identity)
id	Código de identificação do tipo de pagamento.	INTEGER	-	Primary Key / Identity
descricao	Nome descritivo do tipo de pagamento.	VARCHAR	(100)	Not Null

TABELA	TIPO_MOVIMENTACAO			
DESCRIÇÃO	Tabela de referência para os tipos de movimentação (entrada, saída etc.).			
CAMPOS				
NOME	DESCRIÇÃO	TIPO DE DADO	TAMANHO	Restrições de domínio (PK, FK, Not Null, Check, Default, Identity)
id	Código de identificação do tipo de movimentação.	INTEGER	-	Primary Key / Identity
descricao	Nome descritivo do tipo de movimentação.	VARCHAR	(100)	Not Null

9. MODELO LÓGICO DE DADOS

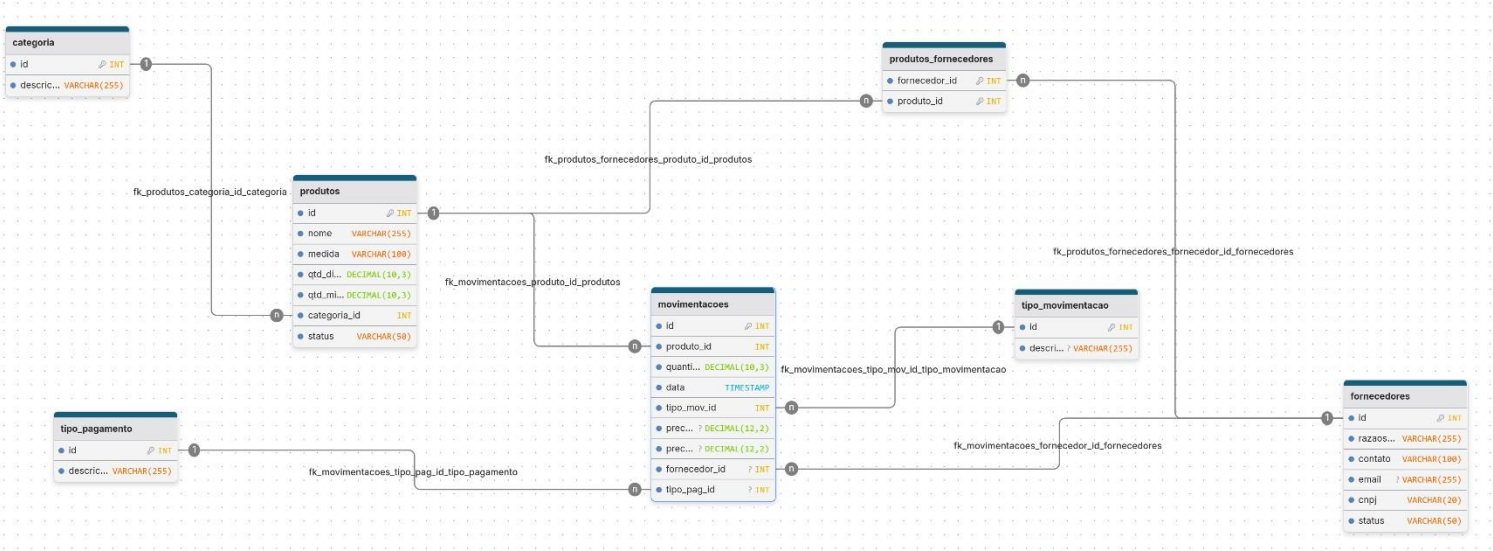

Figura 1: Modelagem lógica do banco de dados DogStock

9.1Tabela: "categoria"

A tabela categoria armazena as categorias dos produtos disponíveis no estoque. Possui duas colunas:

- id: identificador único da categoria, do tipo inteiro, com auto incremento, funcionando como chave primária.
- descricao: descrição da categoria, do tipo texto (VARCHAR 255), obrigatória. Cada produto será vinculado a uma categoria através do campo categoria_id na tabela produtos, estabelecendo um relacionamento de muitos-para-um (N:1).

9.2 Tabela: "tipo_pagamento"

A tabela **tipo_pagamento** registra as formas de pagamento disponíveis no sistema. Suas colunas são:

- id: identificador único do tipo de pagamento, inteiro com autoincremento, **chave primária**.
- descricao_: descrição do tipo de pagamento, VARCHAR(255), **obrigatório**.

Essa tabela é utilizada nas movimentações para registrar a forma de pagamento de compras.

9.3 Tabela: "produtos"

A tabela produtos contém informações sobre os produtos gerenciados pelo sistema. Suas colunas são:

- id: identificador único do produto, do tipo inteiro, autoincrementado, chave primária.
- nome: nome do produto, do tipo VARCHAR(255), obrigatório.
- medida: unidade de medida do produto, VARCHAR(100), obrigatório.
- qtd_disponivel: quantidade atual em estoque, do tipo DECIMAL(10,3), obrigatório.
- qtd_minima: quantidade mínima para alerta de reposição, DECIMAL(10,3), obrigatório.
- categoria_id: referência à categoria do produto, do tipo inteiro, obrigatório, com chave estrangeira referenciando categoria(id).
- status: situação do produto, VARCHAR(50), obrigatório, indicando se está ativo ou inativo.

A tabela produtos possui um relacionamento de muitos-para-um (N:1) com a tabela categoria, garantindo que cada produto esteja associado a uma categoria existente.

9.4 Tabela: "movimentacoes"

A tabela movimentacoes registra todas as entradas e saídas de produtos no estoque, incluindo vendas, compras e ajustes. Suas colunas são:

- id: identificador único da movimentação, inteiro com auto incremento, chave primária. • produto_id: referência ao produto movimentado, inteiro, obrigatório, chave estrangeira para produtos(id).
- quantidade: quantidade movimentada, DECIMAL(10,3), obrigatório.
- data: data e hora da movimentação, TIMESTAMP, com valor padrão CURRENT_TIMESTAMP, obrigatório.
- tipo_mov_id: referência ao tipo de movimentação, inteiro, obrigatório, chave estrangeira para tipo_movimentacao(id).
- preco_venda: preço de venda do produto, DECIMAL(12,2), opcional.
- preco_compra: preço de compra do produto, DECIMAL(12,2), opcional.
- fornecedor_id: referência ao fornecedor, inteiro, opcional, chave estrangeira para fornecedores(id).
- tipo_pag_id: referência ao tipo de pagamento, inteiro, opcional, chave estrangeira para tipo_pagamento(id).

Essa tabela estabelece os seguintes relacionamentos:

- Muitos-para-um com produtos
- Muitos-para-um com tipo_movimentacao
- Opcionalmente, muitos-para-um com fornecedores
- Opcionalmente, muitos-para-um com tipo_pagamento

9.5 Tabela: “produtos_fornecedores”

A tabela produtos_fornecedores é uma tabela de relacionamento muitos-para-muitos (N:N) entre produtos e fornecedores. Suas colunas são:

- fornecedor_id: referência ao fornecedor, inteiro, obrigatório, chave estrangeira para fornecedores(id).
- produto_id: referência ao produto, inteiro, obrigatório, chave estrangeira para produtos(id).

A chave primária é composta pelos campos (fornecedor_id, produto_id), garantindo que cada combinação de produto e fornecedor seja única. Esta tabela permite registrar quais produtos são fornecidos por quais fornecedores.

9.6 Tabela: “tipo_movimentacao”

A tabela tipo_movimentacao registra os tipos de movimentações possíveis no estoque, como compra, venda ou ajuste. Suas colunas são:

- id: identificador único do tipo de movimentação, inteiro com autoincremento, chave primária.
- descricao: descrição do tipo de movimentação, VARCHAR(255), opcional.
  Essa tabela é utilizada para categorizar cada registro da tabela movimentacoes.

9.7 Tabela: “fornecedores”

A tabela “fornecedores” armazena informações sobre os fornecedores de produtos. Suas colunas são:

- id: identificador único do fornecedor, inteiro com autoincremento, chave primária.
- razao_social: nome ou razão social do fornecedor, VARCHAR(255), obrigatório.
- contato: contato principal, VARCHAR(100), obrigatório.
- email: endereço de e-mail do fornecedor, VARCHAR(255), opcional.
- cnpj: CNPJ do fornecedor, VARCHAR(20), obrigatório.
- status: situação do fornecedor, VARCHAR(50), obrigatório.

A tabela permite identificar fornecedores e vinculá-los a produtos e movimentações.

9.8 Normalização do Banco de Dados

O banco de dados *dogstock* foi projetado de acordo com os princípios de normalização até a terceira forma normal (3NF), com o objetivo de garantir a integridade, consistência e eficiência no gerenciamento das informações.

Todas as tabelas que compõem o banco de dados foram analisadas e verificou-se que atendem integralmente aos requisitos da 3NF:

- Tabelas com estrutura simples (categoria, tipo_movimentacao, tipo_pagamento) apresentam atributos atômicos que dependem exclusivamente da chave primária, sem qualquer ocorrência de dependência transitiva.

- Tabelas com relacionamentos complexos (produtos, fornecedores, produtos_fornecedores, movimentacoes) garantem que todos os atributos não-chave dependam diretamente da chave primária da respectiva tabela. Os relacionamentos entre as tabelas são estabelecidos por meio de chaves estrangeiras, assegurando a integridade referencial sem gerar redundância de dados.

10. ÍNDICES

10.1 Índices da tabela produtos

O índice idx_produtos_nome foi criado para acelerar as buscas de produtos pelo nome, o que é essencial quando a API precisa localizar rapidamente um produto específico ou realizar buscas parciais. Já o índice idx_produtos_categoria_id otimiza os JOINs com a tabela de categorias, facilitando a listagem de produtos agrupados por categoria nos endpoints da API.

10.2 Índices da tabela fornecedores

O índice idx_fornecedores_razao_social acelera a busca de fornecedores pelo nome, sendo útil para filtros ou funcionalidades de autocomplete na API. O índice único uq_fornecedores_cnpj garante que cada fornecedor possua um CNPJ exclusivo, evitando duplicidades e permitindo consultas rápidas por esse documento.

10.3 Índices da tabela produtos_fornecedores

O índice idx_pf_produto facilita a listagem de todos os fornecedores relacionados a um determinado produto, enquanto o índice idx_pf_fornecedor agiliza a consulta de todos os produtos fornecidos por um fornecedor específico. Mesmo que a chave primária da tabela seja composta por (produto_id, fornecedor_id), esses índices simples ajudam nas buscas que envolvem apenas um dos campos isoladamente.

10.4 Índices da tabela movimentacoes

O índice composto idx_movimentacoes_produto_data foi criado para permitir a busca eficiente das movimentações de um produto dentro de um período específico, que representa a consulta mais frequente realizada pela API. Já o índice idx_movimentacoes_tipo acelera o filtro das movimentações de entrada ou saída, facilitando a geração de relatórios e dashboards de forma rápida.

11. VIEWS

11.1 VIEW: vw_lista_produtos:

```
CREATE VIEW dogstock.vw_lista_produtos AS

SELECT p.id, p.nome, c.descricao FROM dogstock.produtos p
INNER JOIN dogstock.categoria c on c.id = p.categoria_id
```

Esta view tem como objetivo fornecer uma listagem completa de todos os produtos cadastrados, associando cada produto à sua respectiva categoria. Através dela, é possível visualizar rapidamente quais produtos pertencem a quais categorias, facilitando a organização do catálogo e a consulta para relatórios ou sistemas de exibição de produtos.

11.2 VIEW: vw_produtos_alerta

```
CREATE VIEW dogstock.vw_produtos_alerta AS

SELECT p.id, p.nome, c.descricao,
CASE
        WHEN p.qtd_disponivel < p.qtd_minima THEN 'Abaixo do estoque mínimo'
          ELSE 'Quantidade OK'
END AS status_estoque
FROM dogstock.produtos p
INNER JOIN dogstock.categoria c on c.id = p.categoria_id
```

A finalidade desta view é identificar produtos cujo estoque disponível encontra-se abaixo da quantidade mínima estabelecida. Além de apresentar os produtos e suas categorias, ela indica o status do estoque, informando se está “Abaixo do estoque mínimo” ou se a “Quantidade está dentro do esperado”. Essa informação é essencial para alertar gestores sobre a necessidade de reposição de produtos e para apoiar decisões de controle de estoque.

11.3 VIEW: vw_produtos_fornecedores

```
CREATE VIEW vw_produtos_fornecedores AS
SELECT
  p.id AS produto_id,
  p.nome AS produto,
  c.descricao AS categoria,
  GROUP_CONCAT(f.razao_social SEPARATOR ', ') AS fornecedores
FROM dogstock.produtos p
INNER JOIN dogstock.categoria c
```

```
  ON c.id = p.categoria_id
LEFT JOIN dogstock.produtos_fornecedores pf
  ON pf.produto_id = p.id
LEFT JOIN dogstock.fornecedores f
  ON f.id = pf.fornecedor_id
GROUP BY p.id, p.nome, c.descricao
```

O objetivo desta view é organizar os produtos por categoria, permitindo consultas segmentadas e filtradas de forma eficiente. Ela apresenta os produtos junto com a categoria correspondente, facilitando a visualização e o gerenciamento dos produtos agrupados por tipo, sendo especialmente útil para aplicações em interfaces e APIs que exigem filtragem por categoria.

11.6 VIEW: vw_estoque_baixo_por_categoria

```
CREATE VIEW vw_estoque_baixo_por_categoria AS
SELECT
  c.id AS categoria_id,
  c.descricao AS categoria,
  COUNT(p.id) AS total_produtos_abaixo_minimo,
  GROUP_CONCAT(p.nome SEPARATOR ', ') AS produtos_abaixo_minimo
FROM dogstock.produtos p
INNER JOIN dogstock.categoria c ON c.id = p.categoria_id
WHERE p.qtd_disponivel < p.qtd_minima
GROUP BY c.id, c.descricao;
```

A finalidade desta view é apresentar, por categoria, os produtos que estão com estoque abaixo do nível mínimo recomendado. Ela fornece o total de produtos em alerta e lista os nomes desses produtos de forma consolidada. Essa visão permite que a equipe de estoque priorize rapidamente os produtos que necessitam de reposição e otimize o gerenciamento de inventário.

11.7 VIEW: vw_movimentacoes

```
CREATE VIEW vw_movimentacoes AS

SELECT m.produto_id,

 p.nome,

m.quantidade,

m.data, tm.descricao AS TipoMovimentacao,

m.preco_venda,

m.preco_compra,
```

```
m.fornecedor_id,
tp.descricao AS TipoPagamento
FROM dogstock.movimentacoes m
INNER JOIN dogstock.produtos p ON m.produto_id = p.id
INNER JOIN dogstock.tipo_pagamento tp ON m.tipo_pag_id = tp.id
INNER JOIN dogstock.tipo_movimentacao tm ON m.tipo_mov_id = tm.id
```

A finalidade dessa view é de exibir os registros da tabela movimentações com nome de produto e descrições de tipo de pagamento e movimentação, visto que na tabela original, existem apenas os IDs, impedindo uma leitura eficiente.

12. TESTES DE VALIDAÇÃO

12..1 Tabela categoria

O nome da categoria (descricao) não pode ficar vazio, e não pode existir duas categorias com o mesmo nome.

```
-- Inserção válida
INSERT INTO categoria (descricao) VALUES ('Ração');

-- Inserção inválida: descricao nula
INSERT INTO categoria (descricao) VALUES (NULL);

-- Inserção duplicada
INSERT INTO categoria (descricao) VALUES ('Ração');
```

Somente o primeiro INSERT deve ser aceito; os outros dois devem falhar.
RESULTADO: Como o esperado.

12.2 Tabela produtos

Nome, medida, quantidade disponível, quantidade mínima, categoria e status não podem ficar vazios. As quantidades não podem ser negativas, e a categoria precisa existir na tabela categoria.

```
-- Inserção válida
INSERT INTO produtos (nome, medida, qtd_disponivel, qtd_minima, categoria_id, status)
VALUES ('Ração Adulto', 'Kg', 100, 10, 1, 'Ativo');

-- Inserção inválida: quantidade negativa
INSERT INTO produtos (nome, medida, qtd_disponivel, qtd_minima, categoria_id, status)
VALUES ('Ração Filhote', 'Kg', -5, 10, 1, 'Ativo');

-- Inserção inválida: categoria inexistente
INSERT INTO produtos (nome, medida, qtd_disponivel, qtd_minima, categoria_id, status)
VALUES ('Ração Especial', 'Kg', 10, 5, 999, 'Ativo');
```

Só o primeiro INSERT deve funcionar; os outros dois vão falhar, garantindo que as regras sejam respeitadas.

RESULTADO: Como o esperado.

12.3 Tabela tipo_movimentacao

A descrição do tipo de movimentação não pode ficar vazia.

```
INSERT INTO tipo_movimentacao (descricao_mov) VALUES ('Compra');
INSERT INTO tipo_movimentacao (descricao_mov) VALUES (NULL); -- inválido
```

O segundo INSERT não deve ser aceito.

RESULTADO: Como o esperado.

12.4 Tabela fornecedores

Razão social, contato, CNPJ e status não podem ficar vazios. Cada CNPJ precisa ser único, e o e-mail precisa ter um formato válido, como “nome@dominio.com”.

```
-- Inserção válida
INSERT INTO fornecedores (razaosocial, contato, email, cnpj, status)
VALUES ('Fornecedor A', 'João', 'contato@fornecedor.com', '12.345.678/0001-90', 'Ativo');

-- Inserção inválida: CNPJ duplicado
INSERT INTO fornecedores (razaosocial, contato, email, cnpj, status)
VALUES ('Fornecedor B', 'Maria', 'contato2@fornecedor.com', '12.345.678/0001-90', 'Ativo');

-- Inserção inválida: email inválido
INSERT INTO fornecedores (razaosocial, contato, email, cnpj, status)
VALUES ('Fornecedor C', 'José', 'email_invalido', '98.765.432/0001-10', 'Ativo');
```

Somente o primeiro INSERT deve passar.

RESULTADO: Como o esperado.

12.5 Tabela produtos_fornecedores

O produto e o fornecedor precisam existir nas tabelas certas, e a combinação de produto + fornecedor não pode se repetir.

```
INSERT INTO produtos_fornecedores (fornecedor_id, produto_id) VALUES (1, 1);
INSERT INTO produtos_fornecedores (fornecedor_id, produto_id) VALUES (999, 1); -- fornecedor
inexistente
INSERT INTO produtos_fornecedores (fornecedor_id, produto_id) VALUES (1, 999); -- produto
inexistente
INSERT INTO produtos_fornecedores (fornecedor_id, produto_id) VALUES (1, 1); -- duplicado
```

Apenas o primeiro INSERT deve funcionar.

RESULTADO: Como o esperado.

12.6 Tabela tipo_pagamento

A descrição do tipo de pagamento não pode ficar vazia.

```
INSERT INTO tipo_pagamento (descricao) VALUES ('Cartão');
INSERT INTO tipo_pagamento (descricao) VALUES (NULL); -- inválido
```

O segundo INSERT deve falhar.

RESULTADO: Como o esperado.

12.7 Tabela movimentacoes

Produto, quantidade e tipo de movimentação não podem ficar vazios. A quantidade precisa ser maior que zero. Preço de venda e compra, quando preenchidos, não podem ser negativos. Produto, tipo de movimentação, fornecedor e tipo de pagamento precisam existir nas tabelas correspondentes.

```
-- Inserção válida
INSERT INTO movimentacoes (produto_id, quantidade, tipo_mov_id, preco_venda, preco_compra,
fornecedor_id, tipo_pag_id)
VALUES (1, 10, 1, 15.00, 10.00, 1, 1);

-- Inserção inválida: quantidade negativa
INSERT INTO movimentacoes (produto_id, quantidade, tipo_mov_id)
VALUES (1, -5, 1);

-- Inserção inválida: produto inexistente
INSERT INTO movimentacoes (produto_id, quantidade, tipo_mov_id)
VALUES (999, 5, 1);

-- Inserção inválida: tipo_mov inexistente
INSERT INTO movimentacoes (produto_id, quantidade, tipo_mov_id)
VALUES (1, 5, 999);

-- Inserção inválida: preço negativo
INSERT INTO movimentacoes (produto_id, quantidade, tipo_mov_id, preco_venda) VALUES (1, 5, 1, -
10);
```

Somente o primeiro INSERT deve funcionar, garantindo que as regras sejam respeitadas. **RESULTADO:** Como o esperado.

13. DESEMPENHO E OTIMIZAÇÃO DO BANCO DE DADOS

O banco de dados DogStock foi estruturado para oferecer integridade relacional e alta eficiência em operações de leitura, escrita e análise. As estratégias de performance a seguir visam otimizar consultas frequentes, reduzir tempo de resposta e garantir escalabilidade com o aumento de dados.

13.1. Índices

Foram criados índices específicos para acelerar consultas e junções entre tabelas, considerando os principais padrões de acesso identificados no sistema:
Produtos **idx_produtos_nome** → otimiza buscas textuais por nome do produto.
idx_produtos_categoria_id → acelera junções com a tabela categoria e filtros por categoria.
Fornecedores **idx_fornecedores_razao_social** → melhora o desempenho em pesquisas por razão social.
Produtos_Fornecedores **idx_pf_produto** → otimiza junções e consultas de fornecedores por produto.
idx_pf_fornecedor → melhora junções no sentido inverso (produto → fornecedor).
Movimentações
idx_movimentacoes_produto_data → acelera consultas por produto e intervalo de datas, padrão mais frequente nas APIs de histórico e relatórios.
idx_movimentacoes_tipo → otimiza filtros por tipo de movimentação (entrada ou saída de estoque).
Esses índices reduzem significativamente o custo de varredura de dados (full scan) e melhoram o tempo de resposta em endpoints de listagem, relatórios e dashboards.

13.2. Normalização e Estrutura

O modelo segue Terceira Forma Normal (3FN), eliminando redundâncias e minimizando o volume de dados armazenados. Essa abordagem reduz bloqueios em operações de escrita e garante consistência sem comprometer a performance em consultas complexas.

13.3. Integridade Referencial

O uso de chaves estrangeiras assegura consistência entre as tabelas, mas pode impactar a performance em operações em massa. Mitigações aplicadas:
Índices criados sobre colunas de chaves estrangeiras (como **categoria_id, produto_id e fornecedor_id**);
Evita-se o uso de ações em cascata (ON DELETE/UPDATE CASCADE) em tabelas de alto volume, como movimentacoes, para não gerar bloqueios desnecessários.

13.4. Consultas e Views

As consultas mais frequentes — especialmente as relacionadas a movimentações e histórico de produtos — foram projetadas para se beneficiar dos índices compostos (**produto_id, data_movimentacao**). Recomenda-se:
Evitar SELECT * e trazer apenas as colunas necessárias; Criar views otimizadas para relatórios, agregando campos calculados e reduzindo a necessidade de junções repetitivas; Filtrar preferencialmente por colunas indexadas.

13.5. Escalabilidade e Manutenção

Para sustentar o crescimento do volume de dados:
Avaliar o particionamento da tabela **movimentacoes** por data (mensal ou anual); Implementar rotinas de arquivamento para registros históricos antigos; Executar periodicamente:
ANALYZE TABLE movimentacoes; OPTIMIZE TABLE movimentacoes;
para atualizar estatísticas e reorganizar índices.
O log de consultas lentas (**slow_query_log**) deve permanecer ativo em ambiente de produção, permitindo análise contínua e ajuste de queries com alto custo.

13.6. Considerações Finais

As práticas de indexação e normalização aplicadas asseguram que o DogStock mantenha desempenho consistente mesmo sob alto volume transacional. A manutenção periódica e o monitoramento de queries são essenciais para preservar o equilíbrio entre integridade relacional e performance operacional.

14 PERMISSÕES DO BANCO DE DADOS

O controle de permissões no banco de dados DogStock foi definido com o objetivo de assegurar a integridade e a segurança das informações armazenadas, adotando o princípio do menor privilégio (least privilege principle). Dessa forma, cada usuário tem acesso apenas aos recursos necessários para o desempenho de suas funções, reduzindo o risco de alterações indevidas e garantindo rastreabilidade das operações. Foram criados quatro usuários principais: dbadmin, powerbi, registro e consulta. Cada um possui permissões específicas conforme sua função dentro do sistema, descritas a seguir.

14.1 Usuário dbadmin

O usuário dbadmin representa o administrador do banco de dados, sendo responsável pela criação, manutenção e controle de todos os objetos e usuários. Ele possui privilégios completos (ALL PRIVILEGES) sobre todas as tabelas e estruturas do banco, incluindo comandos de criação (CREATE), modificação (ALTER), exclusão (DROP), definição de índices e concessão de permissões (GRANT OPTION).
O uso desse usuário é restrito a atividades administrativas e de manutenção, devendo ser acessado apenas por conexões seguras. Sua existência é fundamental para o gerenciamento do ambiente, mas seu uso deve ser cuidadosamente controlado, evitando que operações cotidianas sejam realizadas com privilégios excessivos.

14.2 Usuário Power BI

O usuário powerbi foi criado especificamente para a integração do banco de dados com ferramentas de Business Intelligence (BI), permitindo a extração de informações para relatórios e dashboards.
Por questões de segurança e consistência dos dados, esse usuário possui apenas permissão de leitura (SELECT) em todas as tabelas necessárias. Ele não tem permissão para realizar inserções, exclusões ou atualizações, nem para modificar a estrutura do banco.
Essa limitação garante que o processo de análise de dados seja totalmente seguro, impossibilitando que consultas executadas pelo Power BI alterem ou corrompam os registros originais. Assim, o usuário powerbi atua de forma isolada, servindo exclusivamente à visualização de informações.

14.3 Usuário registro

O usuário registro é utilizado pelas APIs responsáveis pelo registro de informações no sistema DogStock. Ele possui permissões de inserção (INSERT), atualização (UPDATE) e leitura (SELECT) sobre as tabelas operacionais, como produtos, fornecedores e movimentacoes.
Entretanto, o usuário registro não possui permissão para exclusão (DELETE) nem para alterações estruturais. Essa configuração foi adotada para evitar a perda acidental de

dados e garantir que as aplicações externas mantenham a integridade das informações inseridas no banco.
Dessa forma, o usuário registro é essencial para o funcionamento das operações diárias do sistema, mas sem comprometer a segurança do ambiente de dados.

14.4 Usuário consulta

O usuário consulta foi criado para permitir acesso restrito aos dados por meio de consultas diretas, sem risco de modificação. Ele possui apenas permissão de leitura (SELECT) em todas as tabelas do banco, sendo voltado para fins de auditoria, monitoramento e verificação de informações.
O usuário é identificado como consulta, com senha Unicid!dg25, e foi configurado para acesso limitado e seguro. Assim como o powerbi, ele desempenha papel de apoio à análise de dados, mas com uso direcionado a consultas pontuais e verificações internas.
Com essa estrutura de permissões, o banco de dados DogStock mantém um equilíbrio entre segurança, controle e eficiência operacional, assegurando que cada usuário atue apenas dentro do escopo necessário à sua função. Além disso, recomenda-se a atualização periódica das credenciais e o monitoramento contínuo dos logs de acesso para reforçar a segurança do ambiente.

15. SCRIPTS DE CRIAÇÃO

15.1 Criação do Banco

```
CREATE DATABASE IF NOT EXISTS dogstock;
USE dogstock;
-- ----------------------
-- Table: categoria
-- ----------------------
CREATE TABLE IF NOT EXISTS categoria (
id INT NOT NULL AUTO_INCREMENT,
descricao VARCHAR(255) NOT NULL,
PRIMARY KEY(id)
);
-- ----------------------
-- Table: produtos
-- ----------------------
CREATE TABLE IF NOT EXISTS produtos (
id INT NOT NULL AUTO_INCREMENT,
nome VARCHAR(255) NOT NULL,
medida VARCHAR(100) NOT NULL,
qtd_disponivel DECIMAL(10,3) NOT NULL,
qtd_minima DECIMAL(10,3) NOT NULL,
categoria_id INT NOT NULL,
status VARCHAR(50) NOT NULL,
PRIMARY KEY(id),
CONSTRAINT fk_produtos_categoria FOREIGN KEY (categoria_id)
    REFERENCES categoria(id)
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
);

-----------------------
-- Table: tipo_movimentacao
-- ----------------------
CREATE TABLE IF NOT EXISTS tipo_movimentacao (
id INT NOT NULL AUTO_INCREMENT,
descricao VARCHAR(255) ,
  PRIMARY KEY(id)
);

- ----------------------
Table: fornecedores
-- ----------------------
```

```
CREATE TABLE IF NOT EXISTS fornecedores (
id INT NOT NULL AUTO_INCREMENT,
razao_social VARCHAR(255) NOT NULL,
contato VARCHAR(100) NOT NULL,
email VARCHAR(255) ,
cnpj VARCHAR(20) NOT NULL,
status VARCHAR(50) NOT NULL,
  PRIMARY KEY(id)
);

-- ----------------------
-- Table: produtos_fornecedores
-- ----------------------
CREATE TABLE IF NOT EXISTS produtos_fornecedores (
fornecedor_id INT NOT NULL,
produto_id INT NOT NULL,
   PRIMARY KEY(fornecedor_id, produto_id),
  CONSTRAINT fk_produtos_fornecedores_fornecedor FOREIGN KEY
(fornecedor_id)
     REFERENCES fornecedores(id)
     ON UPDATE NO ACTION
      ON DELETE NO ACTION,
  CONSTRAINT fk_produtos_fornecedores_produto FOREIGN KEY (produto_id)
     REFERENCES produtos(id)
     ON UPDATE NO ACTION
      ON DELETE NO ACTION
);

-- ----------------------
-- Table: tipo_pagamento
-- ----------------------
CREATE TABLE IF NOT EXISTS tipo_pagamento (
id INT NOT NULL AUTO_INCREMENT,
descricao VARCHAR(255) NOT NULL,
   PRIMARY KEY(id)
);

-- ----------------------
-- Table: movimentacoes
-- ----------------------
CREATE TABLE IF NOT EXISTS movimentacoes (
id INT NOT NULL AUTO_INCREMENT,
produto_id INT NOT NULL,
quantidade DECIMAL(10,3) NOT NULL,
```

```
data TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
tipo_mov_id INT NOT NULL,
preco_venda DECIMAL(12,2) ,
preco_compra DECIMAL(12,2) ,
fornecedor_id INT,
tipo_pag_id INT,
  PRIMARY KEY(id),
  CONSTRAINT fk_movimentacoes_produto FOREIGN KEY (produto_id)
    REFERENCES produtos(id)
    ON UPDATE NO ACTION
   ON DELETE NO ACTION,
  CONSTRAINT fk_movimentacoes_tipo_mov FOREIGN KEY (tipo_mov_id)
    REFERENCES tipo_movimentacao(id)
    ON UPDATE NO ACTION
     ON DELETE NO ACTION,
  CONSTRAINT fk_movimentacoes_fornecedor FOREIGN KEY (fornecedor_id)
    REFERENCES fornecedores(id)
    ON UPDATE NO ACTION
     ON DELETE NO ACTION,
  CONSTRAINT fk_movimentacoes_tipo_pagamento FOREIGN KEY (tipo_pag_id)
    REFERENCES tipo_pagamento(id)
    ON UPDATE NO ACTION
     ON DELETE NO ACTION
);
```

15.2 Script de Criação de Índices

```
INDICES SQL
-- ========================
-- 1. PRODUTOS
-- ========================
-- Busca rápida por nome do produto
CREATE INDEX idx_produtos_nome ON produtos(nome);

-- JOIN rápido com categorias
CREATE INDEX idx_produtos_categoria_id ON produtos(categoria_id);

-- ========================
-- 2. FORNECEDORES
-- ========================
-- Busca por razão social do fornecedor
CREATE INDEX idx_fornecedores_razao_social ON fornecedores(razao_social);
```

```
-- CNPJ único, garante integridade e consultas rápidas
CREATE UNIQUE INDEX uq_fornecedores_cnpj ON fornecedores(cnpj);

-- ========================
-- 3. PRODUTOS_FORNECEDORES
-- ========================
-- JOINs comuns produto → fornecedor
CREATE INDEX idx_pf_produto ON produtos_fornecedores(produto_id);

-- JOINs comuns fornecedor → produto
CREATE INDEX idx_pf_fornecedor ON produtos_fornecedores(fornecedor_id);

-- ========================
-- 4. MOVIMENTACOES
-- ========================
-- Consulta rápida por produto e período (mais usada na API)
CREATE INDEX idx_movimentacoes_produto_data
ON movimentacoes(produto_id, data_movimentacao);

-- Filtros rápidos por tipo de movimentação (entrada/saída)
CREATE INDEX idx_movimentacoes_tipo ON movimentacoes(tipo_movimentacao);
```

16. DOCUMENTAÇÃO TÉCNICA

16.1 Tabela: "categoria"

A tabela categoria armazena as categorias dos produtos disponíveis no estoque. Possui duas colunas:

- id: identificador único da categoria, do tipo inteiro, com auto incremento, funcionando como chave primária.
- descricao: descrição da categoria, do tipo texto (VARCHAR 255), obrigatória.

Cada produto será vinculado a uma categoria através do campo categoria_id na tabela produtos, estabelecendo um relacionamento de muitos-para-um (N:1).

16.2 Tabela: "tipo_pagamento"

A tabela **tipo_pagamento** registra as formas de pagamento disponíveis no sistema. Suas colunas são:

- id: identificador único do tipo de pagamento, inteiro com autoincremento, **chave primária**.
- descricao_tpag: descrição do tipo de pagamento, VARCHAR(255), **obrigatório**.

Essa tabela é utilizada nas movimentações para registrar a forma de pagamento de compras.

16.3 Tabela: "produtos"

A tabela produtos contém informações sobre os produtos gerenciados pelo sistema. Suas colunas são:

- id: identificador único do produto, do tipo inteiro, autoincrementado, chave primária.
- nome: nome do produto, do tipo VARCHAR(255), obrigatório.
- medida: unidade de medida do produto, VARCHAR(100), obrigatório.
- qtd_disponivel: quantidade atual em estoque, do tipo DECIMAL(10,3), obrigatório.
- qtd_minima: quantidade mínima para alerta de reposição, DECIMAL(10,3), obrigatório.
- categoria_id: referência à categoria do produto, do tipo inteiro, obrigatório, com chave estrangeira referenciando categoria(id).
- status: situação do produto, VARCHAR(50), obrigatório, indicando se está ativo ou inativo.

A tabela produtos possui um relacionamento de muitos-para-um (N:1) com a tabela categoria, garantindo que cada produto esteja associado a uma categoria existente.

16.5 Tabela: “movimentacoes”

A tabela movimentacoes registra todas as entradas e saídas de produtos no estoque, incluindo vendas, compras e ajustes. Suas colunas são:

- id: identificador único da movimentação, inteiro com auto incremento, chave primária.
- produto_id: referência ao produto movimentado, inteiro, obrigatório, chave estrangeira para produtos(id).
- quantidade: quantidade movimentada, DECIMAL(10,3), obrigatório. • data: data e hora da movimentação, TIMESTAMP, com valor padrão CURRENT_TIMESTAMP, obrigatório.
- tipo_mov_id: referência ao tipo de movimentação, inteiro, obrigatório, chave estrangeira para tipo_movimentacao(id).
- preco_venda: preço de venda do produto, DECIMAL(12,2), opcional.
- preco_compra: preço de compra do produto, DECIMAL(12,2), opcional.
- fornecedor_id: referência ao fornecedor, inteiro, opcional, chave estrangeira para fornecedores(id).
- tipo_pag_id: referência ao tipo de pagamento, inteiro, opcional, chave estrangeira para tipo_pagamento(id).

Essa tabela estabelece os seguintes relacionamentos:

- Muitos-para-um com produtos
- Muitos-para-um com tipo_movimentacao
- Opcionalmente, muitos-para-um com fornecedores
- Opcionalmente, muitos-para-um com tipo_pagamento

16.6 Tabela: “produtos_fornecedores”

A tabela produtos_fornecedores é uma tabela de relacionamento muitos-para-muitos (N:N) entre produtos e fornecedores. Suas colunas são:

- fornecedor_id: referência ao fornecedor, inteiro, obrigatório, chave estrangeira para fornecedores(id).
- produto_id: referência ao produto, inteiro, obrigatório, chave estrangeira para produtos(id).

A chave primária é composta pelos campos (fornecedor_id, produto_id), garantindo que cada combinação de produto e fornecedor seja única. Esta tabela permite registrar quais produtos são fornecidos por quais fornecedores.

16.7 Tabela: “tipo_movimentacao”

A tabela tipo_movimentacao registra os tipos de movimentações possíveis no estoque, como compra, venda ou ajuste. Suas colunas são:

- id: identificador único do tipo de movimentação, inteiro com autoincremento, chave primária.

- descricao: descrição do tipo de movimentação, VARCHAR(255), opcional.

Essa tabela é utilizada para categorizar cada registro da tabela movimentacoes.

16.8 Tabela: "fornecedores"

A tabela fornecedores armazena informações sobre os fornecedores de produtos. Suas colunas são:

- id: identificador único do fornecedor, inteiro com autoincremento, chave primária.
- razaosocial: nome ou razão social do fornecedor, VARCHAR(255), obrigatório.
- contato: contato principal, VARCHAR(100), obrigatório.
- email: endereço de e-mail do fornecedor, VARCHAR(255), opcional.
- cnpj: CNPJ do fornecedor, VARCHAR(20), obrigatório.
- status: situação do fornecedor, VARCHAR(50), obrigatório.

A tabela permite identificar fornecedores e vinculá-los a produtos e movimentações.

16.9 Normalização do Banco de Dados

O banco de dados *dogstock* foi projetado de acordo com os princípios de normalização até a terceira forma normal (3NF), com o objetivo de garantir a integridade, consistência e eficiência no gerenciamento das informações.

Todas as tabelas que compõem o banco de dados foram analisadas e verificou-se que atendem integralmente aos requisitos da 3NF:

- Tabelas com estrutura simples (categoria, tipo_movimentacao, tipo_pagamento) apresentam atributos atômicos que dependem exclusivamente da chave primária, sem qualquer ocorrência de dependência transitiva.

- Tabelas com relacionamentos complexos (produtos, fornecedores, produtos_fornecedores, movimentacoes) garantem que todos os atributos não-chave dependam diretamente da chave primária da respectiva tabela. Os relacionamentos entre as tabelas são estabelecidos por meio de chaves estrangeiras, assegurando a integridade referencial sem gerar redundância de dados.