**Documentação README**

**Estudo Dirigido 005 - Banco de Dados e POO**

**4. Etapas de Desenvolvimento**

**4.1 Modelo Lógico – Concepção e Justificativa**

**4.1.1 Identifique e descreva as entidades principais da aplicação: *Loja*, *Equipamento* e *Garantia*.**

As cinco entidades principais da aplicação são:

**Usuário:** Representa o dono da conta que utiliza o sistema.  
Cada usuário pode cadastrar lojas, equipamentos, garantias e documentos.

**Loja:** Armazena as informações sobre o estabelecimento onde o equipamento foi adquirido.  
Cada loja pertence a um usuário e pode ter vários equipamentos e documentos vinculados.

**Equipamento:** Representa um produto comprado pelo usuário, que terá documentos e garantias associadas.

**Garantia:** Contém os dados sobre a cobertura de um equipamento — seja de fabricante, estendida ou da loja.  
Cada garantia está vinculada a um equipamento e ao usuário.

**Documento:** Registra os arquivos digitais (nota fiscal, certificado, comprovante etc.) que comprovam a compra ou o vínculo da garantia.  
Pode estar ligado a uma loja, equipamento e/ou garantia.

**4.1.2. Liste os atributos de cada entidade e justifique sua escolha.**

**Atributos de Cada Entidade e Justificativas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entidade** | **Atributo** | **Tipo de Dado** | **Justificativa** |
| **Usuário** | id\_usuario | SERIAL (PK) | Identificador único do usuário. |
|  | nome\_usuario | VARCHAR(100) | Nome completo do usuário. |
|  | cpf | CHAR(11) | Identificação única no sistema. |
|  | email\_usuario | VARCHAR(150) | Canal principal de contato e login. |
|  | telefone\_usuario | VARCHAR(20) | Contato alternativo. |
|  | status | VARCHAR(10) | Indica se o usuário está ativo ou inativo. |
|  | senha | VARCHAR(255) | Armazena a senha criptografada. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entidade** | **Atributo** | **Tipo de Dado** | **Justificativa** |
| **Loja** | id\_loja | SERIAL (PK) | Identificador único da loja. |
|  | nome\_loja | VARCHAR(100) | Nome do estabelecimento. |
|  | cnpj | CHAR(14) | Identificação fiscal única. |
|  | endereco\_loja | VARCHAR(200) | Endereço físico da loja. |
|  | telefone\_loja | VARCHAR(20) | Contato comercial. |
|  | email\_loja | VARCHAR(150) | Comunicação eletrônica. |
|  | id\_usuario | INT (FK) | Liga a loja ao usuário responsável. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entidade** | **Atributo** | **Tipo de Dado** | **Justificativa** |
| **Equipamento** | id\_equip | SERIAL (PK) | Identificador único do equipamento. |
|  | nome\_equip | VARCHAR(100) | Nome ou descrição do equipamento. |
|  | data\_aquisicao | DATE | Data de compra. |
|  | marca | VARCHAR(50) | Marca do produto. |
|  | modelo | VARCHAR(50) | Modelo do produto. |
|  | numero\_serie | VARCHAR(100) | Número de série, usado para rastreamento. |
|  | preco | NUMERIC(10,2) | Valor de compra. |
|  | id\_loja | INT (FK) | Loja onde foi comprado. |
|  | id\_usuario | INT (FK) | Dono do equipamento. |
| **Entidade** | **Atributo** | **Tipo de Dado** | **Justificativa** |
| **Garantia** | id\_garantia | SERIAL (PK) | Identificador único da garantia. |
|  | data\_inicio | DATE | Início da validade da garantia. |
|  | data\_fim | DATE | Fim da cobertura. |
|  | tipo\_garantia | VARCHAR(20) | Classifica a garantia (fabricante, loja, estendida). |
|  | id\_equip | INT (FK) | Equipamento vinculado. |
|  | id\_usuario | INT (FK) | Usuário responsável. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entidade** | **Atributo** | **Tipo de Dado** | **Justificativa** |
| **Documento** | id\_documento | SERIAL (PK) | Identificador único do documento. |
|  | url | VARCHAR(255) | Caminho ou link para o arquivo. |
|  | tipo\_doc | VARCHAR(20) | Tipo de documento (nota fiscal, certificado, outro). |
|  | num\_doc | VARCHAR(50) | Número de controle (ex: número da nota). |
|  | data\_emissao | DATE | Data de emissão do documento. |
|  | id\_loja | INT (FK, opcional) | Loja relacionada. |
|  | id\_equip | INT (FK, opcional) | Equipamento vinculado. |
|  | id\_garantia | INT (FK, opcional) | Garantia vinculada. |

**4.1.3. Determine as chaves primárias e estrangeiras.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entidade** | **PK** | **FKs** | **Relaciona-se com** |
| **usuario** | id\_usuario | — | — |
| **loja** | id\_loja | id\_usuario | usuario(id\_usuario) |
| **equipamento** | id\_equip | id\_loja, id\_usuario | loja(id\_loja), usuario(id\_usuario) |
| **garantia** | id\_garantia | id\_equip, id\_usuario | equipamento(id\_equip), usuario(id\_usuario) |
| **documento** | id\_documento | id\_loja, id\_equip, id\_garantia | loja, equipamento, garantia |

**4.1.4. Crie o diagrama lógico (Draw.io, Miro ou similar) e salve como prints/modelo\_logico.png .**

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**4.1.5. Descreva no README uma justificativa técnica para as relações criadas (ex: "Uma loja pode vender vários equipamentos, logo a relação é 1:N").**

* **Usuário x Loja (1:N)**Um usuário pode cadastrar várias lojas (onde fez compras), mas cada loja pertence a um único usuário.
* **Usuário x Equipamento (1:N)**Um usuário pode ter vários equipamentos, mas cada equipamento pertence a apenas um usuário.
* **Loja x Equipamento (1:N)**Uma loja pode vender muitos equipamentos, mas cada equipamento é comprado em uma loja.
* **Equipamento x Garantia (1:N)**Um equipamento pode ter várias garantias (por exemplo, do fabricante e uma estendida), mas cada garantia pertence a apenas um equipamento.
* **Equipamento x Documento (1:N)**Um equipamento pode possuir diversos documentos (nota fiscal, certificado etc.), mas cada documento se refere a um único equipamento.
* **Garantia x Documento (1:N)**Uma garantia pode ter mais de um documento associado (certificados, comprovantes), mas cada documento está vinculado a uma única garantia.
* **Usuário x Garantia (1:N)**Cada garantia pertence ao usuário que registrou o equipamento, garantindo integridade dos dados.

**Reflexão: explique a diferença entre modelo conceitual, lógico e físico.**

Na construção de um banco de dados, a modelagem passa por três estágios principais: modelo conceitual, modelo lógico e modelo físico. Cada um desses modelos tem um objetivo distinto e se concentra em diferentes aspectos do sistema de dados.

**Modelo Conceitual:**

**Objetivo:** O modelo conceitual tem como principal objetivo **capturar a visão de alto nível do sistema** de informações, sem se preocupar com detalhes de implementação ou tecnologia. Ele foca no **significado** dos dados e nas **relações** entre as entidades.

**Características:**

* **Abstração**: Ele é mais abstrato e descreve as entidades e seus relacionamentos de forma geral, sem considerar aspectos de como os dados serão armazenados ou estruturados fisicamente.
* **Independente de tecnologia**: Não há preocupação com banco de dados específico (relacional, NoSQL, etc.). O modelo conceitual é agnóstico quanto à tecnologia de implementação.
* **Enfoque nas necessidades do negócio**: Tem como objetivo refletir as necessidades do negócio ou do sistema que se está construindo.

**Modelo Lógico:**

**Objetivo:** O modelo lógico é **uma tradução do modelo conceitual**, mas agora levando em consideração como os dados serão organizados no sistema de gerenciamento de banco de dados (SGBD). Ele se preocupa com a **estrutura dos dados** sem ainda entrar em detalhes de implementação física.

**Características:**

* **Estruturação detalhada**: Define as entidades e seus atributos, além das relações entre elas de uma maneira mais formal. Mas ainda não diz como os dados serão armazenados no disco.
* **Tecnologia dependente**: Embora o modelo lógico não seja tão dependente de tecnologia como o modelo físico, ele já leva em conta as limitações e características do tipo de banco de dados que será utilizado (relacional, por exemplo).
* **Definição de tabelas e chaves**: É onde são definidas as **chaves primárias e estrangeiras**, e as **relações entre tabelas**.

**Exemplo**: No modelo lógico de uma loja, você pode definir que a "Loja" tem um identificador único (chave primária), que "Equipamento" também tem uma chave primária e que há uma relação de 1:N entre "Loja" e "Equipamento", onde cada loja pode ter vários equipamentos.

**Modelo Físico:**

**Objetivo:** O modelo físico é o mais próximo da implementação real do banco de dados. Ele se preocupa com **como os dados serão armazenados fisicamente** no banco de dados, levando em consideração as especificidades do SGBD escolhido.

**Características:**

* **Desempenho e otimização**: Este modelo foca em como os dados serão armazenados para garantir **desempenho** e **eficiência** no acesso, leitura e gravação. Pode envolver a definição de índices, particionamento de tabelas e outras estratégias de otimização.
* **Especificações do SGBD**: O modelo físico considera as especificidades do SGBD utilizado (ex: MySQL, PostgreSQL, Oracle), como tipos de dados, índices e organização dos dados em disco.
* **Armazenamento físico**: É aqui que você decide como os dados serão armazenados em discos, quais índices serão criados para otimizar as consultas e qual a estrutura de arquivos será usada.

**Exemplo**: No modelo físico, você pode definir que a tabela "Equipamento" será armazenada em uma partição específica do banco de dados, que o campo "Preço" será armazenado como DECIMAL(10,2) e que um índice será criado sobre o campo "Nome" para otimizar as buscas por nome do equipamento.

**4.2 Modelo Físico – Implementação no PostgreSQL**

**4.2.1. Crie o banco app\_garantia no DBeaver.**

CREATE DATABASE app\_garantia;

**4.2.2. Construa o script schema.sql contendo as tabelas: Loja, equipamento, garantia**

Inclua restrições como NOT NULL , CHECK , UNIQUE , ON DELETE RESTRICT .

-- =========================================================

-- TABELA: usuario

-- =========================================================

**CREATE** **TABLE** usuario (

id\_usuario SERIAL **PRIMARY** **KEY**,

nome\_usuario **VARCHAR**(100) **NOT** **NULL**,

cpf **CHAR**(11) **UNIQUE** **NOT** **NULL**,

email\_usuario **VARCHAR**(150) **UNIQUE** **NOT** **NULL**,

telefone\_usuario **VARCHAR**(20),

status **VARCHAR**(10) **DEFAULT** 'ativo' **CHECK** (status **IN** ('ativo', 'inativo')),

senha **VARCHAR**(255) **NOT** **NULL**,

data\_cadastro **TIMESTAMP** **DEFAULT** CURRENT\_TIMESTAMP

);

-- =========================================================

-- TABELA: loja

-- =========================================================

**CREATE** **TABLE** loja (

id\_loja SERIAL **PRIMARY** **KEY**,

nome\_loja **VARCHAR**(100) **NOT** **NULL**,

cnpj **CHAR**(14) **UNIQUE** **NOT** **NULL**,

endereco\_loja **VARCHAR**(200),

telefone\_loja **VARCHAR**(20),

email\_loja **VARCHAR**(150),

id\_usuario **INT** **NOT** **NULL**,

**FOREIGN** **KEY** (id\_usuario) **REFERENCES** usuario(id\_usuario)

**ON** **DELETE** **CASCADE**

**ON** **UPDATE** **CASCADE**

);

-- =========================================================

-- TABELA: equipamentos

-- =========================================================

**CREATE** **TABLE** equipamentos (

id\_equip SERIAL **PRIMARY** **KEY**,

nome\_equip **VARCHAR**(100) **NOT** **NULL**,

data\_aquisicao **DATE** **NOT** **NULL**,

marca **VARCHAR**(50),

modelo **VARCHAR**(50),

numero\_serie **VARCHAR**(100) **UNIQUE**,

preco **NUMERIC**(10,2),

id\_loja **INT** **NOT** **NULL**,

id\_usuario **INT** **NOT** **NULL**,

**FOREIGN** **KEY** (id\_loja) **REFERENCES** loja(id\_loja)

**ON** **DELETE** **CASCADE**

**ON** **UPDATE** **CASCADE**,

**FOREIGN** **KEY** (id\_usuario) **REFERENCES** usuario(id\_usuario)

**ON** **DELETE** **CASCADE**

**ON** **UPDATE** **CASCADE**

);

-- =========================================================

-- TABELA: garantia

-- =========================================================

**CREATE** **TABLE** garantia (

id\_garantia SERIAL **PRIMARY** **KEY**,

data\_inicio **DATE** **NOT** **NULL**,

data\_fim **DATE** **NOT** **NULL**,

tipo\_garantia **VARCHAR**(20) **NOT** **NULL** **CHECK** (tipo\_garantia **IN** ('fabricante', 'estendida', 'loja')),

id\_equip **INT** **NOT** **NULL**,

id\_usuario **INT** **NOT** **NULL**,

**FOREIGN** **KEY** (id\_equip) **REFERENCES** equipamentos(id\_equip)

**ON** **DELETE** **CASCADE**

**ON** **UPDATE** **CASCADE**,

**FOREIGN** **KEY** (id\_usuario) **REFERENCES** usuario(id\_usuario)

**ON** **DELETE** **CASCADE**

**ON** **UPDATE** **CASCADE**

);

-- =========================================================

-- TABELA: documento

-- =========================================================

**CREATE** **TABLE** documento (

id\_documento SERIAL **PRIMARY** **KEY**,

url **VARCHAR**(255) **NOT** **NULL**,

tipo\_doc **VARCHAR**(20) **NOT** **NULL** **CHECK** (tipo\_doc **IN** ('nota\_fiscal', 'certificado', 'outro')),

num\_doc **VARCHAR**(50),

data\_emissao **DATE**,

id\_loja **INT**,

id\_equip **INT**,

id\_garantia **INT**,

**FOREIGN** **KEY** (id\_loja) **REFERENCES** loja(id\_loja)

**ON** **DELETE** **SET** **NULL**

**ON** **UPDATE** **CASCADE**,

**FOREIGN** **KEY** (id\_equip) **REFERENCES** equipamentos(id\_equip)

**ON** **DELETE** **SET** **NULL**

**ON** **UPDATE** **CASCADE**,

**FOREIGN** **KEY** (id\_garantia) **REFERENCES** garantia(id\_garantia)

**ON** **DELETE** **SET** **NULL**

**ON** **UPDATE** **CASCADE**

);

**4.2.3. Insira ao menos três registros em cada tabela ( inserts.sql ).**

-- =========================================================

-- INSERÇÕES NA TABELA: usuario

-- =========================================================

**INSERT** **INTO** usuario (nome\_usuario, cpf, email\_usuario, telefone\_usuario, senha)

**VALUES**

('Ana Souza', '12345678901', 'ana.souza@example.com', '(11)98888-1111', 'senha123'),

('Bruno Lima', '23456789012', 'bruno.lima@example.com', '(21)97777-2222', 'senha456'),

('Carla Mendes', '34567890123', 'carla.mendes@example.com', '(31)96666-3333', 'senha789');

-- =========================================================

-- INSERÇÕES NA TABELA: loja

-- =========================================================

**INSERT** **INTO** loja (nome\_loja, cnpj, endereco\_loja, telefone\_loja, email\_loja, id\_usuario)

**VALUES**

('TechStore', '12345678000199', 'Av. Paulista, 1000 - São Paulo/SP', '(11)4002-8922', 'contato@techstore.com', 1),

('Eletrônicos BR', '98765432000155', 'Rua das Flores, 200 - Rio de Janeiro/RJ', '(21)3003-4455', 'vendas@eletronicosbr.com', 2),

('MegaCell', '45678912000133', 'Av. Afonso Pena, 500 - Belo Horizonte/MG', '(31)3555-9090', 'suporte@megacell.com', 3);

-- =========================================================

-- INSERÇÕES NA TABELA: equipamentos

-- =========================================================

**INSERT** **INTO** equipamentos (nome\_equip, data\_aquisicao, marca, modelo, numero\_serie, preco, id\_loja, id\_usuario)

**VALUES**

('Notebook Dell Inspiron 15', '2023-01-15', 'Dell', 'Inspiron 15 3000', 'SN123456', 3500.00, 1, 1),

('Smartphone Samsung Galaxy S22', '2023-03-10', 'Samsung', 'Galaxy S22', 'SN987654', 4200.00, 2, 2),

('Impressora HP LaserJet 1020', '2023-05-25', 'HP', 'LaserJet 1020', 'SN567890', 900.00, 3, 3);

-- =========================================================

-- INSERÇÕES NA TABELA: garantia

-- =========================================================

**INSERT** **INTO** garantia (data\_inicio, data\_fim, tipo\_garantia, id\_equip, id\_usuario)

**VALUES**

('2023-01-15', '2025-01-15', 'fabricante', 1, 1),

('2023-03-10', '2024-03-10', 'estendida', 2, 2),

('2023-05-25', '2024-05-25', 'loja', 3, 3);

-- =========================================================

-- INSERÇÕES NA TABELA: documento

-- =========================================================

**INSERT** **INTO** documento (url, tipo\_doc, num\_doc, data\_emissao, id\_loja, id\_equip, id\_garantia)

**VALUES**

('https://docs.techstore.com/notafiscal123.pdf', 'nota\_fiscal', 'NF123', '2023-01-15', 1, 1, 1),

('https://docs.eletronicosbr.com/certificado\_s22.pdf', 'certificado', 'CERT987', '2023-03-10', 2, 2, 2),

('https://docs.megacell.com/manual\_hp1020.pdf', 'outro', 'DOC567', '2023-05-25', 3, 3, 3);

**4.2.4. Capture o resultado no DBeaver ( prints/consultas\_dbeaver.png ).**

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**Atividade investigativa: pesquise e aplique o uso de ON DELETE CASCADE e explique quando ele é útil ou perigoso.**

O **ON DELETE CASCADE** é uma restrição de chave estrangeira que, quando aplicada, especifica que, se uma linha na tabela referenciada for excluída, todas as linhas na tabela que fazem referência a essa linha também serão excluídas automaticamente. Em outras palavras, a exclusão de um registro "cascata" para todas as dependências relacionadas.

**Quando é útil?**

* **Útil** quando queremos garantir que a exclusão de um registro principal (como uma loja ou garantia) remova automaticamente todos os registros relacionados (como os equipamentos dessa loja ou com aquela garantia). Isso é prático para manter a integridade dos dados sem a necessidade de excluir manualmente as dependências.

**Caso de uso**: Se uma loja for excluída, todos os equipamentos dessa loja também serão excluídos automaticamente.

**Quando pode ser perigoso?**

* **Perigoso** se não houver um controle adequado, especialmente quando a exclusão em cascata pode remover dados importantes sem querer, como em um sistema de estoque ou financeiro. Por exemplo, se uma garantia ou loja for excluída, você pode acabar excluindo dados críticos de equipamentos sem perceber.

**4.3 Consultas SQL – Análise de Dados**

Crie e documente no README consultas que respondam:

**4.3.1. Quais equipamentos estão vinculados a cada loja?**

**SELECT**

l.id\_loja,

l.nome\_loja,

e.id\_equip,

e.nome\_equip,

e.marca,

e.modelo,

e.data\_aquisicao

**FROM** loja l

**JOIN** equipamentos e **ON** l.id\_loja = e.id\_loja

**ORDER BY** l.nome\_loja, e.nome\_equip;

**Explicação:**

Esta consulta faz um join entre loja e equipamentos, listando todos os equipamentos pertencentes a cada loja.

O ORDER BY organiza os resultados primeiro pelo nome da loja e depois pelo nome do equipamento.

**Uso na aplicação:**

* Exibir no painel administrativo a lista de equipamentos por loja.
* Gerar relatórios de inventário por unidade.
* Ajudar na gestão de ativos, especialmente quando há múltiplas filiais.

**4.3.2. Quais garantias vencem nos próximos 30 dias?**

**SELECT**

g.id\_garantia,

e.nome\_equip,

l.nome\_loja,

g.data\_fim AS data\_vencimento,

g.tipo\_garantia

**FROM** garantia g

**JOIN** equipamentos e **ON** g.id\_equip = e.id\_equip

**JOIN** loja l **ON** e.id\_loja = l.id\_loja

**WHERE** g.data\_fim **BETWEEN** CURRENT\_DATE AND (CURRENT\_DATE + INTERVAL '30 days')

**ORDER BY** g.data\_fim;

**Explicação:**

Filtra todas as garantias cujo término (data\_fim) ocorrerá dentro dos próximos 30 dias a partir da data atual.

**Uso na aplicação:**

* Geração de alertas automáticos para o time de suporte ou manutenção.
* Exibição em um dashboard de notificações para que a equipe possa agir preventivamente.
* Envio de emails ou notificações push para usuários responsáveis.

**4.3.3. Qual loja possui o maior número de garantias vencidas?**

**SELECT**

l.id\_loja,

l.nome\_loja,

**COUNT**(g.id\_garantia) **AS** qtd\_garantias\_vencidas

**FROM** loja l

**JOIN** equipamentos e **ON** l.id\_loja = e.id\_loja

**JOIN** garantia g ON e.id\_equip = g.id\_equip

**WHERE** g.data\_fim < CURRENT\_DATE

**GROUP BY** l.id\_loja, l.nome\_loja

**ORDER BY** qtd\_garantias\_vencidas **DESC**

**LIMIT 1**;

**Explicação:**

Conta quantas garantias já estão vencidas (data\_fim < CURRENT\_DATE) por loja, e retorna aquela com o maior número de garantias expiradas.

**Uso na aplicação:**

* Exibir em relatórios gerenciais quais filiais estão com mais garantias expiradas (indicando possíveis riscos de custo de manutenção).
* Ajudar a priorizar renovações ou ações corretivas.
* Geração de indicadores de desempenho (KPIs) por loja.

**4. Qual o tempo médio de garantia por loja?**

**SELECT**

l.id\_loja,

l.nome\_loja,

**ROUND**(**AVG**(g.data\_fim - g.data\_inicio)) **AS** media\_dias\_garantia

**FROM** loja l

**JOIN** equipamentos e ON l.id\_loja = e.id\_loja

**JOIN** garantia g ON e.id\_equip = g.id\_equip

**GROUP BY** l.id\_loja, l.nome\_loja

**ORDER BY** media\_dias\_garantia **DESC**;

**Explicação:**

Calcula a média de duração das garantias (em dias) por loja.

O cálculo é feito pela diferença entre data\_fim e data\_inicio, e o AVG() retorna a média de todas as garantias daquela loja.

**Uso na aplicação:**

* Geração de relatórios analíticos sobre o comportamento de compras e fornecedores.
* Avaliar se determinadas lojas costumam ter garantias mais curtas (pode indicar contratos piores ou produtos de menor durabilidade).
* Apoiar negociações futuras com fornecedores.

**4.4 POO – Representação das Entidades em Python**

**Crie classes correspondentes às tabelas dentro da pasta models/ .**

Cada classe deve conter:

* Construtor ( \_\_init\_\_ )
* Método \_\_str\_\_ para exibição legível
* Comentários de documentação

class Usuario:

    # Representa um usuário do sistema.

    def \_\_init\_\_(*self*, *id\_usuario*, *nome\_usuario*, *cpf*, *email\_usuario*,

*telefone\_usuario*=None, *status*='ativo', *senha*=None, *data\_cadastro*=None):

*self*.id\_usuario = *id\_usuario*

*self*.nome\_usuario = *nome\_usuario*

*self*.cpf = *cpf*

*self*.email\_usuario = *email\_usuario*

*self*.telefone\_usuario = *telefone\_usuario*

*self*.status = *status*

*self*.senha = *senha*

*self*.data\_cadastro = *data\_cadastro*

    def \_\_str\_\_(*self*):

        return f"Usuário: {*self*.nome\_usuario} ({*self*.email\_usuario}) - Status: {*self*.status}"

class Loja:

    # Representa uma loja cadastrada no sistema.

    def \_\_init\_\_(*self*, *id\_loja*, *nome\_loja*, *cnpj*, *endereco\_loja*=None,

*telefone\_loja*=None, *email\_loja*=None, *id\_usuario*=None):

*self*.id\_loja = *id\_loja*

*self*.nome\_loja = *nome\_loja*

*self*.cnpj = *cnpj*

*self*.endereco\_loja = *endereco\_loja*

*self*.telefone\_loja = *telefone\_loja*

*self*.email\_loja = *email\_loja*

*self*.id\_usuario = *id\_usuario*

    def \_\_str\_\_(*self*):

        return f"Loja: {*self*.nome\_loja} (CNPJ: {*self*.cnpj})"

class Equipamento:

    # Representa um equipamento vinculado a uma loja.

    def \_\_init\_\_(*self*, *id\_equip*, *nome\_equip*, *data\_aquisicao*, *marca*=None,

*modelo*=None, *numero\_serie*=None, *preco*=None, *id\_loja*=None, *id\_usuario*=None):

*self*.id\_equip = *id\_equip*

*self*.nome\_equip = *nome\_equip*

*self*.data\_aquisicao = *data\_aquisicao*

*self*.marca = *marca*

*self*.modelo = *modelo*

*self*.numero\_serie = *numero\_serie*

*self*.preco = *preco*

*self*.id\_loja = *id\_loja*

*self*.id\_usuario = *id\_usuario*

    def \_\_str\_\_(*self*):

        return f"{*self*.nome\_equip} ({*self*.marca} {*self*.modelo}) - Nº Série: {*self*.numero\_serie}"

class Documento:

    # Representa um documento vinculado a uma loja, equipamento ou garantia.

    def \_\_init\_\_(*self*, *id\_documento*, *url*, *tipo\_doc*, *num\_doc*=None,

*data\_emissao*=None, *id\_loja*=None, *id\_equip*=None, *id\_garantia*=None):

*self*.id\_documento = *id\_documento*

*self*.url = *url*

*self*.tipo\_doc = *tipo\_doc*

*self*.num\_doc = *num\_doc*

*self*.data\_emissao = *data\_emissao*

*self*.id\_loja = *id\_loja*

*self*.id\_equip = *id\_equip*

*self*.id\_garantia = *id\_garantia*

    def \_\_str\_\_(*self*):

        return f"Documento {*self*.tipo\_doc.upper()} ({*self*.num\_doc}) - Emitido em {*self*.data\_emissao}"

from datetime import date

class Garantia:

    # Representa uma garantia de equipamento.

    def \_\_init\_\_(*self*, *id\_garantia*, *data\_inicio*, *data\_fim*, *tipo\_garantia*,

*id\_equip*=None, *id\_usuario*=None):

*self*.id\_garantia = *id\_garantia*

*self*.data\_inicio = *data\_inicio*

*self*.data\_fim = *data\_fim*

*self*.tipo\_garantia = *tipo\_garantia*

*self*.id\_equip = *id\_equip*

*self*.id\_usuario = *id\_usuario*

    def \_\_str\_\_(*self*):

        dias\_restantes = (*self*.data\_fim - date.today()).days

        return f"Garantia {*self*.tipo\_garantia} - vence em {*self*.data\_fim} ({dias\_restantes} dias restantes)"

class GarantiaEstendida(Garantia):

    """

    Representa uma garantia estendida, herdando de Garantia.

    Atributos adicionais:

        empresa\_responsavel (str): Nome da empresa que oferece a garantia estendida.

        custo\_adicional (float): Valor pago pela extensão da garantia.

    """

    def \_\_init\_\_(*self*, *id\_garantia*, *data\_inicio*, *data\_fim*, *id\_equip*,

*id\_usuario*, *empresa\_responsavel*, *custo\_adicional*):

        super().\_\_init\_\_(*id\_garantia*, *data\_inicio*,

*data\_fim*, 'estendida', *id\_equip*, *id\_usuario*)

*self*.empresa\_responsavel = *empresa\_responsavel*

*self*.custo\_adicional = *custo\_adicional*

    def \_\_str\_\_(*self*):

        return (f"Garantia Estendida ({*self*.empresa\_responsavel}) "

                f"- Válida até {*self*.data\_fim} - Custo adicional: R${*self*.custo\_adicional:.2f}")

**4.5 Camada de Persistência – Conexão e CRUD**

**Crie o módulo database.py com uma classe Database responsável por:**

* Estabelecer conexão com PostgreSQL ( psycopg2 );
* Executar consultas SQL e retornar resultados;
* Tratar erros com try/except .

**Módulo database.py**

# Módulo database.py Responsável por gerenciar a conexão com o banco de dados PostgreSQL e executar consultas SQL

import psycopg2

class Database:

    """

    Classe responsável pela conexão e execução de comandos SQL no PostgreSQL.

    Métodos:

        - consultar(query): executa uma consulta SELECT e retorna os resultados.

        - executar(query, params): executa INSERT, UPDATE ou DELETE.

        - fechar\_conexao(): encerra a conexão com o banco.

    """

    def \_\_init\_\_(*self*):

        """Inicializa a conexão com o banco de dados PostgreSQL."""

        try:

*self*.conn = psycopg2.connect(

*dbname*="app\_garantia",

*user*="postgres",

*password*="sua\_senha",

*host*="localhost",

*port*="5432"

            )

            print("✅ Conexão com o banco de dados estabelecida com sucesso.")

        except Exception as e:

            print("❌ Erro ao conectar ao banco de dados:", e)

*self*.conn = None

    def consultar(*self*, *query*):

        """

        Executa uma consulta SELECT e retorna o resultado como lista de tuplas.

        """

        if not *self*.conn:

            print("Conexão não estabelecida.")

            return []

        cur = *self*.conn.cursor()

        try:

            cur.execute(*query*)

            resultados = cur.fetchall()

            return resultados

        except Exception as e:

            print("❌ Erro na consulta:", e)

            return []

        finally:

            cur.close()

    def executar(*self*, *query*, *params*=None):

        """

        Executa comandos de modificação de dados (INSERT, UPDATE, DELETE).

        Usa commit automático para persistir as alterações.

        """

        if not *self*.conn:

            print("Conexão não estabelecida.")

            return

        cur = *self*.conn.cursor()

        try:

            cur.execute(*query*, *params*)

*self*.conn.commit()

            print("✅ Comando executado com sucesso.")

        except Exception as e:

*self*.conn.rollback()

            print("❌ Erro ao executar comando:", e)

        finally:

            cur.close()

    def fechar\_conexao(*self*):

        """Fecha a conexão com o banco."""

        if *self*.conn:

*self*.conn.close()

            print("🔒 Conexão com o banco de dados encerrada.")

**Módulo Main.py**

Uma consulta que retorne equipamentos e garantias;

Impressão dos resultados em formato legível.

# Módulo principal da aplicação - Demonstra o uso da classe Database para consultar dados do banco e exibir informações

from database import Database

def main():

    # Instancia o objeto de conexão

    db = Database()

    # Consulta: equipamentos e suas garantias associadas

    query = """

        SELECT

            e.id\_equip,

            e.nome\_equip,

            e.marca,

            e.modelo,

            g.id\_garantia,

            g.tipo\_garantia,

            g.data\_inicio,

            g.data\_fim

        FROM equipamentos e

        JOIN garantia g ON e.id\_equip = g.id\_equip

        ORDER BY e.nome\_equip;

    """

    resultados = db.consultar(query)

    # Exibição dos resultados

    print("\n=== 📋 Lista de Equipamentos e Garantias ===")

    if not resultados:

        print("Nenhum registro encontrado.")

    else:

        for row in resultados:

            id\_equip, nome, marca, modelo, id\_gar, tipo, inicio, fim = row

            print(f"""

----------------------------------------

Equipamento: {nome} ({marca} {modelo})

ID Equip: {id\_equip}

Garantia: {tipo} (ID: {id\_gar})

Início: {inicio} | Fim: {fim}

""")

    # Fecha conexão ao final

    db.fechar\_conexao()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**4.6 Testes e Documentação**

**4.6.1. Capture a execução no terminal ( prints/execucao\_terminal.png ).**

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

**4.6.2. Reflexão Pessoal**

**Reflexão pessoal:**

* **O que aprendi neste estudo?**

Neste exercício, aprendi a:

* **Utilizar PostgreSQL com Python** para persistir dados e consultar informações de um banco de dados.
* **Criar classes em Python** para modelar entidades do banco de dados (POO), representando tabelas como objetos.
* **Trabalhar com a biblioteca psycopg2**, utilizando-a para interagir com um banco de dados PostgreSQL.
* **Estruturar um código modular** com camadas bem definidas (conexão, consulta, e processamento de dados).
* **Que erros enfrentei e como resolvi?**
* **Erros na integração do PostgreSQL com o DBeaver:** Tive que dar uma pesquisada pois estava acontecendo alguns erros na hora da execução do Banco de Dados
* **Erro na instalação do psycopg2**: Tive problemas com a instalação do pacote psycopg2, que não estava sendo encontrado no meu ambiente Python. Resolvi isso com a instalação do psycopg2
* **Como este exercício se conecta ao projeto integrador?**

Este exercício está diretamente relacionado ao meu **projeto integrador**, pois ele aborda a persistência de dados e a interação com bancos de dados, que é uma das funcionalidades essenciais para o projeto. A implementação de um sistema que armazena e consulta dados de maneira eficiente é uma parte fundamental do desenvolvimento de sistemas de gestão ou controle, como o que estamos pensando em criar no Projeto Integrador. Além disso, o conceito de camadas (modelagem, persistência e execução) e o uso de boas práticas de POO são fundamentais para garantir que o projeto seja escalável e bem estruturado.