INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

LUCAS APARECIDO DOS SANTOS

SISTEMA DE GERENCIAMENTO DE IMPRESSÕES 3D

CAMPOS DO JORDÃO

2025

RESUMO

Este trabalho detalha o processo de desenvolvimento e implementação de um banco de dados relacional para um sistema de gerenciamento de impressão 3D, batizado de "NeoLayer". O mercado de impressão 3D carece de soluções integradas que gerenciam o fluxo de e-commerce (venda de produtos prontos), a prestação de serviços (modelagem personalizada) e o controle de insumos (filamentos). O objetivo principal é criar uma estrutura de dados normalizada e eficiente que sirva como alicerce para essa aplicação. A metodologia adotada segue as fases clássicas do projeto de banco de dados: levantamento de requisitos, modelagem conceitual (utilizando a notação de Peter Chen na ferramenta brModelo), mapeamento para o modelo lógico (com dicionário de dados) e, por fim, a implementação do modelo físico no SGBD **MariaDB**. O resultado é um esquema de banco de dados robusto, capaz de gerenciar usuários, produtos, pedidos, solicitações de modelagem e o inventário de filamentos, validado por um conjunto de consultas SQL complexas que simulam relatórios gerenciais essenciais.

Palavras-chave: Banco de Dados Relacional, Impressão 3D, SQL, MariaDB, Modelagem de Dados, Gerenciamento de Estoque.

ABSTRACT

This paper details the process of developing and implementing a relational database for a 3D printing management system called “NeoLayer.” The 3D printing market lacks integrated solutions that manage e-commerce flow (sale of finished products), service provision (custom modeling), and input control (filaments). The main objective is to create a standardized and efficient data structure that serves as the foundation for this application. The methodology adopted follows the classic phases of database design: requirements gathering, conceptual modeling (using Peter Chen's notation in the brModelo tool), mapping to the logical model (with a data dictionary), and finally, implementation of the physical model in the MariaDB DBMS. The result is a robust database schema capable of managing users, products, orders, modeling requests, and filament inventory, validated by a set of complex SQL queries that simulate essential management reports.

Keywords: Relational Database, 3D Printing, SQL, MariaDB, Data Modeling, Inventory Management.

SUMÁRIO

1 INTRODUÇÃO

A manufatura aditiva, popularmente conhecida como impressão 3D, transcende o nicho de prototipagem rápida e se tornou uma solução viável para produção em pequena escala e produtos personalizados. Negócios que operam nesse setor enfrentam um desafio logístico triplo: gerenciar um catálogo de produtos digitais (modelos 3D), administrar um e-commerce para produtos físicos impressos e processar pedidos de serviços de modelagem personalizados.

O problema central reside na gestão de dados: como rastrear o consumo de matéria-prima (filamentos) por pedido, gerenciar o status de serviços de modelagem e, ao mesmo tempo, fornecer relatórios de vendas e estoque para o administrador? A ausência de um sistema centralizado leva à inconsistência de dados, desperdício de material e falhas no atendimento ao cliente.

1.1 Objetivos

O objetivo geral deste trabalho é projetar, modelar e implementar o banco de dados relacional para o sistema "NeoLayer". Os objetivos específicos incluem:

- Realizar o levantamento de requisitos para as visões do Cliente e do Administrador.
- Desenvolver o modelo conceitual (DER) e o modelo lógico (esquema relacional).
- Implementar o modelo físico em **MariaDB**, incluindo tabelas, chaves e restrições (constraints).
- Validar a estrutura com 20 consultas SQL complexas que demonstrem a capacidade analítica do banco de dados.

1.2 Justificativa

A implementação de um banco de dados relacional robusto é a fundação indispensável para um sistema "NeoLayer". Este projeto se justifica pela necessidade de aplicar a teoria de banco de dados a um problema de negócio

moderno e complexo. Um banco de dados bem estruturado garante a integridade dos dados (um pedido não pode consumir um filamento inexistente), otimiza a tomada de decisão (relatórios de vendas versus estoque) e automatiza processos (baixa de estoque).

2 PROJETO PROPOSTO

O projeto foi executado em fases sequenciais, partindo da abstração do problema até a implementação técnica.

2.1 Considerações Iniciais e Coleta de Regras de Negócio

A análise inicial identificou dois perfis de usuário (Cliente, *Admin*) com necessidades distintas. O Cliente interage com duas frentes:

1. E-commerce: Compra de Produtos pré-definidos, escolhendo o material (Filamento) no ato da compra.
2. Serviço: Abertura de solicitação de modelagem para projetos personalizados.

O *Admin* necessita de controle total sobre pedidos, solicitações e, crucialmente, o estoque de filamento.

Coleta de regras de negócio:

- (RN01) O sistema deve diferenciar usuários Cliente de Admin.
- (RN02) Um Produto é um item de catálogo (ex: "Vaso Grego"). O item vendido é a instância desse produto impressa em um Filamento específico (ex: "Vaso Grego em PLA Azul").
- (RN03) Um Pedido pode conter múltiplos Item_Pedidos (N:M).
- (RN04) Uma solicitação de modelagem é um serviço avulso, com status próprio e orçamento.
- (RN05) O Filamento é a matéria-prima, cadastrada por material, cor, fabricante e preço por grama.
- (RN06) O estoque de filamento deve ser debitado (em gramas) quando um Pedido muda seu status para "Em Impressão". A quantidade a ser debitada vem de Produto.Gramas_Filamento_Estimadas.

2.2 Modelagem Conceitual: Ferramenta e Notação

Ferramenta: brModelo

Descrição: Ferramenta de modelagem de dados gratuita e amplamente utilizada no meio acadêmico brasileiro. Foi escolhida por sua simplicidade e foco estrito na modelagem conceitual e lógica.

Requisitos: A ferramenta é um executável Java (.jar), exigindo apenas a instalação do Java Runtime Environment (JRE), versão 8 ou superior.

Notação Utilizada: Peter Chen. Esta notação foi selecionada por sua expressividade gráfica, representando entidades como retângulos, relacionamentos como losangos e atributos como elipses, o que facilita a visualização das cardinalidades.

2.3 Modelagem Física: Ferramenta e Requisitos

Ferramenta (SGBD): MariaDB

Descrição: O MariaDB foi escolhido por ser um SGBD (Sistema de Gerenciamento de Banco de Dados) relacional de código aberto, um *fork* de alta compatibilidade do MySQL. É conhecido por sua alta performance, robustez e comunidade ativa.

Ferramenta (Administração): HeidiSQL

Descrição: HeidiSQL é um cliente de administração gráfico leve, popular e de código aberto para MariaDB, MySQL, PostgreSQL e outros.

3 RESULTADOS OBTIDOS

Nesta seção serão apresentados os resultados deste trabalho.

3.1 Modelo Conceitual

Entidades Identificadas:

- Usuario (Armazena Clientes e Admins)

- Filamento (Tipos de material: PLA, ABS, cor, preço/grama)
- Estoque_Filamento (Quantidade em gramas de cada filamento)
- Produto (Catálogo de itens: preço base, gramas estimadas)
- Pedido (Cabeçalho do pedido: cliente, data, status)
- Item_Pedido (Entidade associativa: o que foi comprado, em qual filamento)
- Solicitacao_Modelagem (Serviços personalizados)

Relacionamentos:

- Usuario (1:N) Pedido
- Usuario (1:N) Solicitacao_Modelagem
- Pedido (1:N) Item_Pedido (Lado 1 do N:M)
- Produto (1:N) Item_Pedido (Lado 1 do N:M)
- Filamento (1:N) Item_Pedido (Lado 1 do N:M)
- Filamento (1:1) Estoque_Filamento (Cada tipo de filamento tem uma única entrada de estoque)

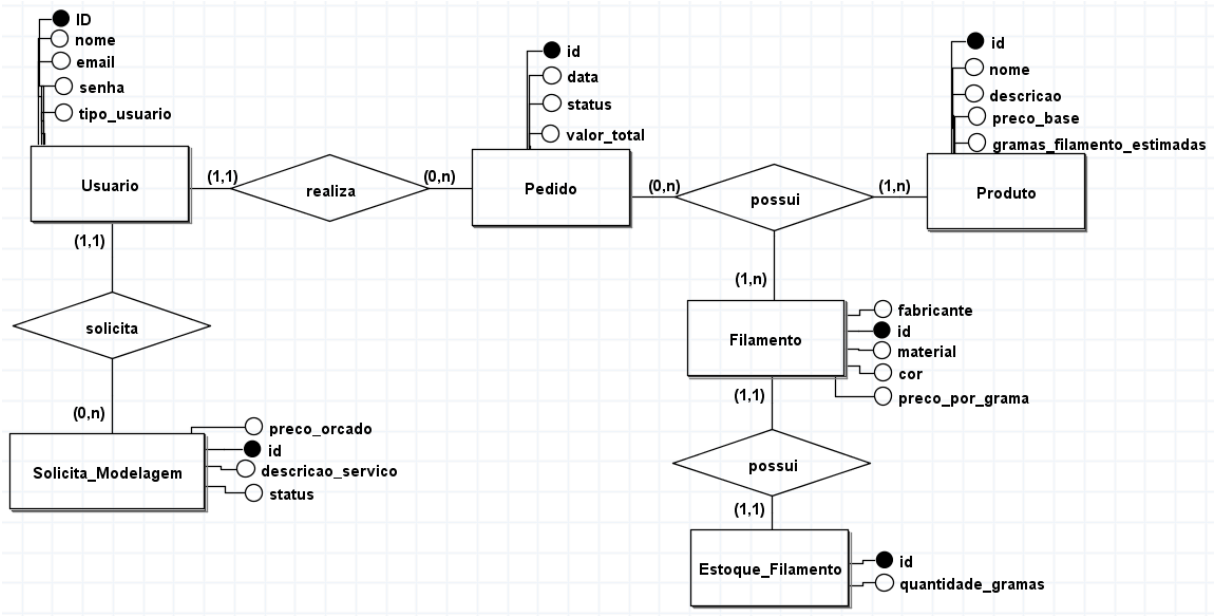

Figura 1 - Modelo conceitual

3.2 Modelo Lógico

O modelo lógico traduz o DER em esquemas relacionais (tabelas e colunas), adaptados ao SGBD MariaDB.

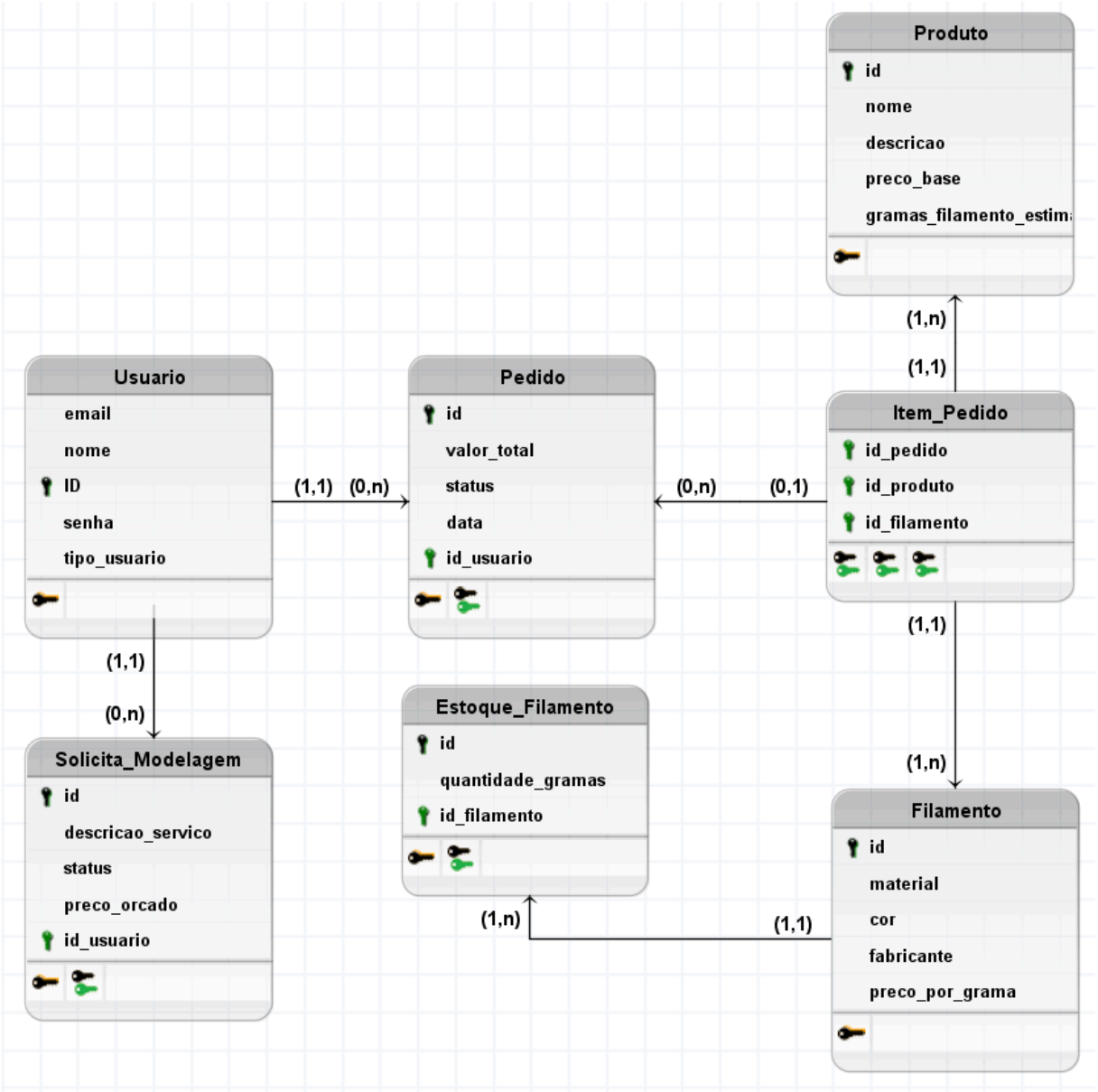

Figura 1 - Modelo conceitual

3.3 Modelo Físico

O código disponível no github (link) cria a estrutura do banco de dados no MariaDB. Note o uso de INT AUTO_INCREMENT e a especificação do ENGINE=InnoDB para garantir o suporte a chaves estrangeiras.

3.4 Consultas SQL

Abaixo estão 20 consultas complexas para relatórios e gerenciamento, compatíveis com a sintaxe do MariaDB.

(Query 01) Relatório de Vendas Mensal (Valor Total Faturado por Mês)

Agrupa todos os pedidos concluídos por mês/ano e soma seus valores. Usa DATE_FORMAT do MariaDB.

```sql
SELECT DATE_FORMAT(Data_Pedido, '%Y-%m') AS Mes_Ano,
       SUM(Valor_Total) AS Faturamento_Mensal
FROM Pedido
WHERE Status_Pedido = 'Concluído'
GROUP BY Mes_Ano
ORDER BY Mes_Ano DESC;
```

(Query 02) Top 5 Produtos Mais Vendidos (em quantidade)

Conta a quantidade total de cada produto vendido e ordena.

```sql
SELECT P.Nome, SUM(IP.Quantidade) AS Total_Vendido
FROM Item_Pedido IP
JOIN Produto P ON IP.ID_Produto = P.ID_Produto
GROUP BY P.Nome
ORDER BY Total_Vendido DESC
LIMIT 5;
```

(Query 03) Top 5 Clientes (por valor total gasto em produtos)

Soma o valor de todos os pedidos concluídos por cliente.

```sql
SELECT U.Nome, SUM(P.Valor_Total) AS Total_Gasto
FROM Pedido P
JOIN Usuario U ON P.ID_Usuario = U.ID_Usuario
WHERE P.Status_Pedido = 'Concluído'
GROUP BY U.Nome
ORDER BY Total_Gasto DESC
LIMIT 5;
```

(Query 04) Relatório de Nível de Estoque de Filamentos

Lista todos os filamentos e suas quantidades em gramas.

```sql
SELECT F.Material, F.Cor, F.Fabricante, E.Quantidade_Gramas
FROM Filamento F
JOIN Estoque_Filamento E ON F.ID_Filamento = E.ID_Filamento
ORDER BY E.Quantidade_Gramas ASC;
```

(Query 05) Alerta de Estoque Baixo (Filamentos abaixo de 500g)

Filtra o relatório de estoque para mostrar apenas itens críticos.

```sql
SELECT F.Material, F.Cor, E.Quantidade_Gramas
FROM Filamento F
JOIN Estoque_Filamento E ON F.ID_Filamento = E.ID_Filamento
WHERE E.Quantidade_Gramas < 500;
```

(Query 06) Pedidos Atualmente "Em Impressão" (Painel do Admin)

Lista os pedidos que estão na fila de impressão.

```sql
SELECT P.ID_Pedido, U.Nome AS Cliente, P.Data_Pedido
FROM Pedido P
JOIN Usuario U ON P.ID_Usuario = U.ID_Usuario
WHERE P.Status_Pedido = 'Em Impressão'
ORDER BY P.Data_Pedido ASC;
```

(Query 07) Solicitações de Modelagem Aguardando Orçamento

Lista serviços personalizados que o admin precisa analisar.

```sql
SELECT S.ID_Solicitacao, U.Nome AS Cliente, S.Descricao_Servico
FROM Solicitacao_Modelagem S
JOIN Usuario U ON S.ID_Usuario = U.ID_Usuario
WHERE S.Status_Solicitacao = 'Aguardando Orçamento';
```

(Query 08) Custo de Material (Filamento) de um Pedido Específico (ID 102)

Calcula o custo total de filamento para um pedido.

```sql
SELECT SUM(IP.Quantidade * P.Gramas_Filamento_Estimadas * F.Preco_Por_Grama) AS Custo_Material_Total
FROM Item_Pedido IP
JOIN Produto P ON IP.ID_Produto = P.ID_Produto
JOIN Filamento F ON IP.ID_Filamento = F.ID_Filamento
WHERE IP.ID_Pedido = 102;
```

(Query 09) Cor de Filamento Mais Popular (em volume de vendas)

Agrupa por cor de filamento e soma as quantidades vendidas.

```sql
SELECT F.Cor, SUM(IP.Quantidade) AS Total_Itens_Vendidos
FROM Item_Pedido IP
JOIN Filamento F ON IP.ID_Filamento = F.ID_Filamento
GROUP BY F.Cor
ORDER BY Total_Itens_Vendidos DESC
LIMIT 1;
```

(Query 10) Produtos que Nunca Foram Vendidos

Usa subconsulta (NOT IN) para encontrar produtos sem histórico em Item_Pedido.

```sql
SELECT Nome, Preco_Base
FROM Produto
WHERE ID_Produto NOT IN (SELECT DISTINCT ID_Produto FROM Item_Pedido);
```

(Query 11) Calcular Valor Total do Estoque de Filamentos

Multiplica a quantidade de cada filamento pelo seu preço por grama.

```sql
SELECT SUM(E.Quantidade_Gramas * F.Preco_Por_Grama) AS Valor_Total_Estoque
FROM Estoque_Filamento E
JOIN Filamento F ON E.ID_Filamento = F.ID_Filamento;
```

(Query 12) Clientes que Compraram Produtos E Solicitaram Modelagem

Encontra a interseção de clientes nos dois serviços.

```sql
(SELECT ID_Usuario FROM Pedido)
INTERSECT
(SELECT ID_Usuario FROM Solicitacao_Modelagem);
```

(Query 13) Clientes que SÓ Solicitaram Modelagem (e nunca compraram produtos)

Usa LEFT JOIN / IS NULL para encontrar exclusão.

```sql
SELECT U.Nome, U.Email
FROM Usuario U
JOIN Solicitacao_Modelagem S ON U.ID_Usuario = S.ID_Usuario
LEFT JOIN Pedido P ON U.ID_Usuario = P.ID_Usuario
WHERE P.ID_Pedido IS NULL
GROUP BY U.Nome, U.Email;
```

(Query 14) Filamento (em gramas) Necessário para Pedidos "Em Impressão"

Calcula a demanda de material para a produção atual (RN06).

```sql
SELECT F.Material, F.Cor, SUM(IP.Quantidade * P.Gramas_Filamento_Estimadas) AS Gramas_Necessarias
FROM Item_Pedido IP
JOIN Pedido PE ON IP.ID_Pedido = PE.ID_Pedido
JOIN Produto P ON IP.ID_Produto = P.ID_Produto
JOIN Filamento F ON IP.ID_Filamento = F.ID_Filamento
WHERE PE.Status_Pedido = 'Em Impressão'
GROUP BY F.Material, F.Cor;
```

(Query 15) Valor Médio dos Pedidos (Ticket Médio) Concluídos

Calcula a média da coluna Valor_Total de pedidos concluídos.

```sql
SELECT AVG(Valor_Total) AS Ticket_Medio
FROM Pedido
WHERE Status_Pedido = 'Concluído' AND Valor_Total > 0;
```

(Query 16) Valor Médio dos Serviços de Modelagem Concluídos

Calcula a média dos orçamentos de serviços finalizados.

```sql
SELECT AVG(Preco_Orcado) AS Media_Servico_Modelagem
FROM Solicitacao_Modelagem
WHERE Status_Solicitacao = 'Concluída' AND Preco_Orcado > 0;
```

(Query 17) Ranking de Produtos por Receita (Valor Total)

Usa Window Function (RANK) para classificar produtos pela receita gerada.

```sql
WITH ReceitaProduto AS (
    SELECT P.Nome, SUM(IP.Preco_Item * IP.Quantidade) AS Receita_Total
    FROM Item_Pedido IP
    JOIN Produto P ON IP.ID_Produto = P.ID_Produto
    JOIN Pedido PE ON IP.ID_Pedido = PE.ID_Pedido
    WHERE PE.Status_Pedido = 'Concluído'
    GROUP BY P.Nome
)
SELECT Nome, Receita_Total,
       RANK() OVER (ORDER BY Receita_Total DESC) AS Posicao
FROM ReceitaProduto;
```

(Query 18) Detalhes de um Pedido Específico (ID 102) para Impressão

Junta múltiplas tabelas para gerar a "comanda" de impressão.

```sql
SELECT P.Nome AS Produto, F.Material, F.Cor, IP.Quantidade
FROM Item_Pedido IP
JOIN Produto P ON IP.ID_Produto = P.ID_Produto
JOIN Filamento F ON IP.ID_Filamento = F.ID_Filamento
WHERE IP.ID_Pedido = 102;
```

(Query 19) Média de Pedidos por Cliente (Taxa de Recompra)

Calcula quantos pedidos, em média, cada cliente fez.

```sql
WITH ContagemPedidos AS (
    SELECT ID_Usuario, COUNT(ID_Pedido) AS Total_Pedidos
    FROM Pedido
    GROUP BY ID_Usuario
)
SELECT AVG(Total_Pedidos) AS Media_Pedidos_Por_Cliente
FROM ContagemPedidos;
```

(Query 20) Lucratividade por Produto (Receita vs Custo Material)

Calcula o lucro bruto por produto, subtraindo o custo do filamento da receita.

```sql
SELECT
    P.Nome,
    SUM(IP.Preco_Item * IP.Quantidade) AS Receita_Total,
    SUM(IP.Quantidade * P.Gramas_Filamento_Estimadas * F.Preco_Por_Grama) AS Custo_Material_Total,
    (SUM(IP.Preco_Item * IP.Quantidade) -
     SUM(IP.Quantidade * P.Gramas_Filamento_Estimadas * F.Preco_Por_Grama)) AS Lucro_Bruto
FROM Item_Pedido IP
JOIN Produto P ON IP.ID_Produto = P.ID_Produto
JOIN Filamento F ON IP.ID_Filamento = F.ID_Filamento
JOIN Pedido PE ON IP.ID_Pedido = PE.ID_Pedido
WHERE PE.Status_Pedido = 'Concluído'
GROUP BY P.Nome
ORDER BY Lucro_Bruto DESC;
```

4 CONCLUSÃO

Este projeto demonstrou com sucesso o ciclo de vida completo do desenvolvimento de um banco de dados relacional, aplicado a um domínio de negócio contemporâneo e complexo como o gerenciamento de impressão 3D.

O resultado obtido é um esquema de banco de dados em Terceira Forma Normal (3FN), implementado em PostgreSQL, que atende a todos os requisitos

funcionais levantados. A estrutura separa claramente os conceitos de Produto (design), Filamento (matéria-prima) e Item_Pedido (instância vendida), permitindo flexibilidade ao cliente e controle preciso ao administrador. As 20 consultas desenvolvidas validam a arquitetura, demonstrando que o banco de dados é capaz de fornecer relatórios gerenciais complexos, desde o controle de estoque (Query 05) até a análise de lucratividade (Query 20).

A principal dificuldade e aprendizado do projeto foi a modelagem correta da regra de negócio (RN06) referente ao débito de estoque, que evidencia a intersecção entre o banco de dados e a lógica de aplicação (seja via Triggers ou software).

REFERÊNCIAS

Documentation - MariaDB.org. Disponível em: <https://mariadb.org/documentation>. Acesso em: 14 nov. 2025.