Centro Universitário Capital
Federal Unifecaf

Sistema de Agendamento para Barbearia:

Um Projeto Integrador com Java e PostgreSQL

Alex Tavares de Oliveira

Joice Barbosa Santos

Lucas David Pereira Esteves

Luiz Henrique de Almeida Santos

Matheus Bondezan de Sousa

Orientador: Marcus Vinícius Camillo Gália

Disciplinas: Object Oriented Programming

Turma: 3ºMA

Taboão da Serra - SP

2025

Centro Universitário Capital
Federal Unifecaf

Objetivo e Proposta do Projeto

Este projeto integrador tem como objetivo principal **consolidar os conhecimentos adquiridos no 3º semestre do curso de Análise e Desenvolvimento de Sistemas**, por meio do desenvolvimento de um **sistema funcional completo**.

A proposta consiste em aplicar, na prática, os conceitos de:

- **Arquitetura de software**
- **Programação orientada a objetos**
- **Modelagem de dados**
- **Estruturas de dados**
- **Integração com banco de dados**

Para isso, o grupo escolheu como tema a criação do **sistema "Barbershop"**, uma aplicação desktop para o **agendamento de serviços em barbearia**, desenvolvida em **Java**, utilizando **JFrame (Swing)** como framework de interface gráfica e **PostgreSQL** como banco de dados.

O projeto também promove o uso de **design de software ágil** e práticas de back-end modernas, proporcionando uma experiência prática que prepara os alunos para os **desafios do mercado de trabalho**.

Centro Universitário Capital
Federal Unifecaf

Atendendo aos Requisitos da Expo Tech 2025

1. Modelagem de Dados

📌 Objetivo

Definir a estrutura lógica do banco de dados do sistema *Barbershop*, contemplando as **tabelas**, **relacionamentos** e **regras de negócio**, com foco na persistência dos dados relacionados a usuários, agendamentos, cadastros e histórico administrativo.

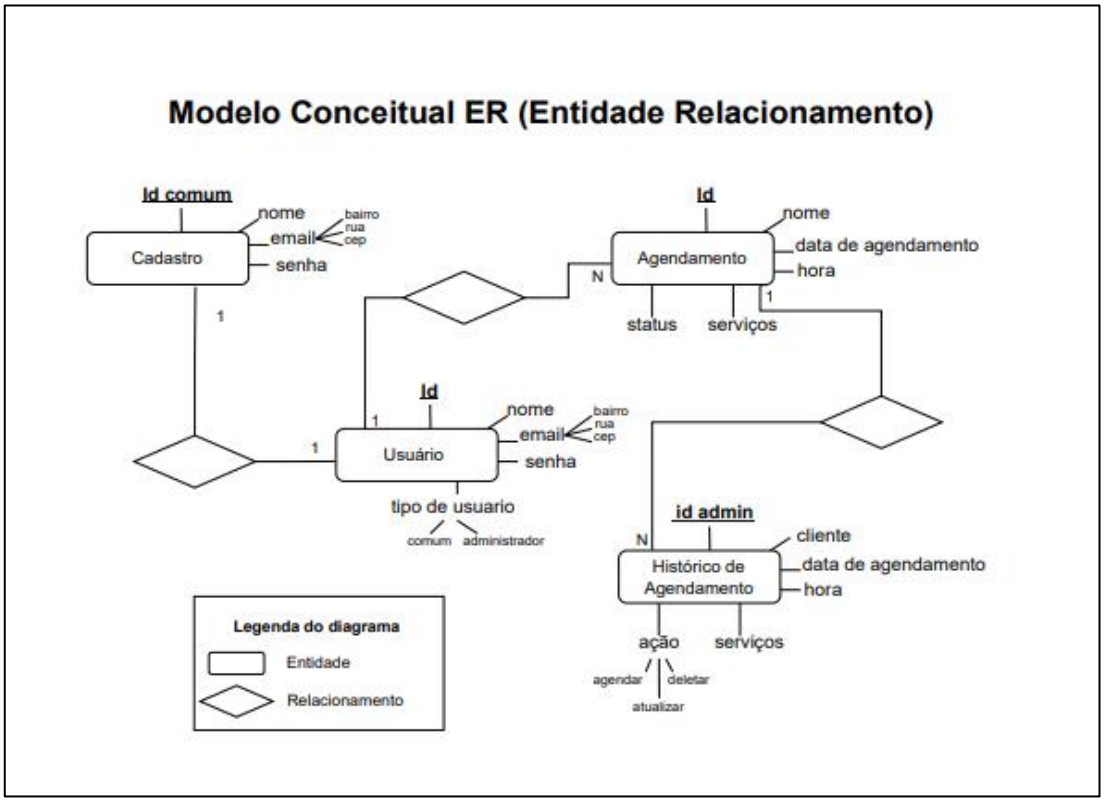

Figura 1: Modelo Gráfico ER

Relacionamentos

- **Usuário** pode realizar vários **Agendamentos** (1:N)
- **Usuário** possui um único **Cadastro** (1:1)
- **Agendamento** pode conter várias **Ações no Histórico de Agendamentos** (1:N)
- **Administrador (Usuário)** pode registrar várias **Ações no Histórico de Agendamentos** (1:N)

Centro Universitário Capital
Federal Unifecaf

Modelo Lógico

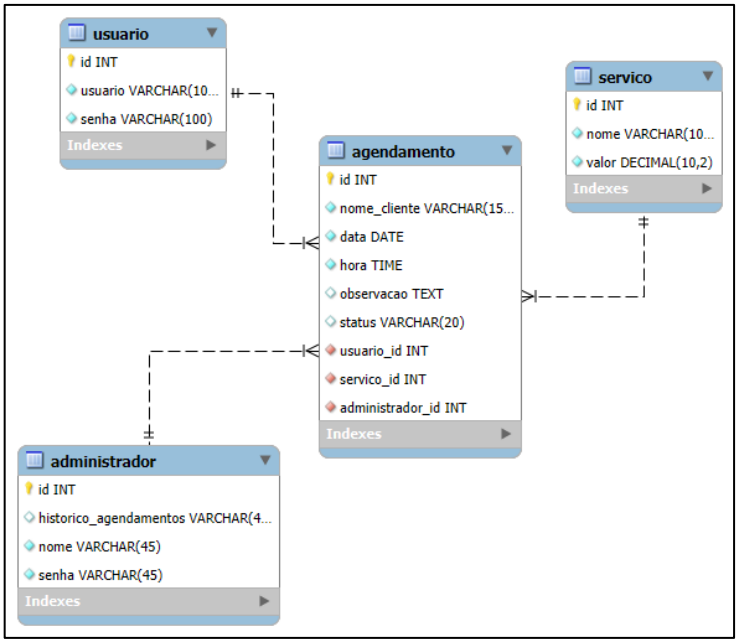

Figura 2: Diagrama ER **PostgreSQL**

Modelo Físico em SQL

```
-- Tabela de Usuários
CREATE TABLE Usuario (
    id_usuario INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    senha VARCHAR(255) NOT NULL,
    tipo_usuario ENUM('comum', 'admin') NOT NULL
);

-- Tabela de Cadastro (dados adicionais do usuário)
CREATE TABLE Cadastro (
    id_cadastro INT PRIMARY KEY AUTO_INCREMENT,
    nome_completo VARCHAR(150) NOT NULL,
    cpf VARCHAR(14) UNIQUE NOT NULL,
    telefone VARCHAR(20),
    endereco TEXT,
    id_usuario INT,
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario)
);
```

```
-- Tabela de Agendamentos
CREATE TABLE Agendamento (
    id_agendamento INT PRIMARY KEY AUTO_INCREMENT,
    data DATE NOT NULL,
    hora TIME NOT NULL,
    status VARCHAR(20) DEFAULT 'pendente',
    observacoes TEXT,
    id_usuario INT,
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario)
);
```

2.Fluxograma

Um fluxograma foi criado (conforme especificado no resumo) representando:

- Abertura do sistema
- Login
- Redirecionamento para interface de usuário comum ou administrador
- Fluxo de agendamento
- Fluxo de cadastro
- Operações do administrador (confirmar, editar, excluir agendamentos)

Figura 3: Fluxograma

3.Interface para o Usuário (Desktop)

Criada com **Java Swing (JFrame)**:

- **Login**
  - Campos: usuário e senha
  - Botões: "Cadastrar" e "Entrar"
- **Cadastro**
  - Campos: usuário, senha, confirmar senha
  - Botão: "Cadastrar"
- **Agendamento (Usuário)**
  - Campos: nome, data, hora, observação
  - Combobox: tipo de valor e tipo de serviço
  - Botões: "Agendar", "Voltar"
- **Tela Administrativa (Admin)**
  - Visualização em tabela
  - Botões: "Atualizar", "Agendar", "Deletar"
- **Tela Inicial**
  - Menu com opções: "Cadastre-se", "Agende já"

4.Validação de Dados

- Campos obrigatórios não aceitam valores vazios
- Tipos de dados são validados (ex: não aceita letras em campos de data)
- Confirmação de senha no cadastro
- try-catch com tratamento para:
  - NumberFormatException
  - SQLException
  - IOException

5.Linguegem de Programação

- Todo o sistema foi desenvolvido em **Java**

❖ **Boas Práticas de POO Aplicadas**

Conceito	Aplicado?	Exemplo
Encapsulamento	✅	Uso de private + getters/setters
Herança	✅	Usuario extends Pessoa
Polimorfismo	✅	Construtores sobrecarregados
Abstração	✅	Classe Pessoa é abstract
Tratamento de erros	✅	try-catch com Logger

6. Integração Front-end e Back-end

- Utilizou-se **JFRAME (interface)** que oferece funcionalidades como:

- **Barra de título:** exibe o título da janela do aplicativo.
- **Controles da janela:** fornece botões para minimizar, maximizar e fechar a janela.
- **Painel de conteúdo:** atua como um contêiner para adicionar outros componentes da GUI, como botões, rótulos e campos de texto.
- **Manipulação de eventos:** gerencia interações do usuário, como cliques do mouse e entradas do teclado.
- **Gerenciamento de layout:** organiza os componentes dentro do quadro usando gerenciadores de layout.

- **Operações de janela:** define o comportamento quando o usuário fecha a janela, como sair do aplicativo ou descartar o quadro.

7. Autenticação de Usuário

- Login e senha validados contra banco de dados
- Diferenciação entre usuários comuns e administradores (ENUM)

8. Conexão com Banco de Dados

- Conexão via JPA com **PostgreSQL**
- Persistência completa das entidades e histórico de ações

Funcionalidades CRUD implementadas

- **Create**: cadastro de usuários, agendamentos
- **Read**: visualização de agendamentos e histórico
- **Update**: edição de agendamentos (admin)
- **Delete**: exclusão de agendamentos (admin)

9. Perfis de Usuário

- Usuario Comum:
  - Visualiza e realiza agendamentos
- Administrador:
  - Confirma, edita e exclui agendamentos
  - Visualiza histórico completo de ações

Ferramentas Utilizadas

- **Linguagem:** Java
- **IDE:** NetBeans
- **Framework de Interface:** JFrame (Java Swing)
- **Banco de Dados:** PostgreSQL
- **Gerenciamento de Layout:** Gerenciadores Swing (FlowLayout, BorderLayout, etc.)

- **Validações:** Try-Catch + estruturas condicionais

Atendimento aos Requisitos:

Requisito	Implementado?	Detalhes
1. Modelagem de Dados	✅	Tabelas com relacionamentos definidos, integradas via PostgreSQL
2. Fluxograma	✅	Representa o fluxo entre telas: Login, Cadastro, Agendamento, Admin
3. Interface para o Usuário (Desktop)	✅	Interface em Java Swing (JFrame), menus, campos de texto, botões
4. Validação de Dados	✅	Campos obrigatórios, confirmação de senha, tipos de entrada verificados
5. Linguagem de Programação	✅	Java (usando NetBeans)
6. Integração Front-end e Back-end	✅	Lógica Java conectada diretamente à interface JFrame e banco PostgreSQL
7. Autenticação de Usuário	✅	Login com validação no banco de dados
8. Conexão com Banco de Dados	✅	PostgreSQL com CRUD completo
9. Perfis de Usuário	✅	Comum (cliente) e Administrador