**UNIVERSIDADE NOVE DE JULHO**

**DIRETORIA DOS CURSOS DE INFORMÁTICA**

**Maria Eduarda Ferraz Silva - 324102259**

**Luana Dias da Silva Sacutti - 323200602**

**Guilherme Almeida da Silva - 324100245**

**PROJETO EM DESENVOLVIMENTO DE SISTEMAS**

Denguinho - Banho e tosa

**SÃO PAULO**

**2025**

**Maria Eduarda Ferraz Silva - 324102259**

**Luana Dias da Silva Sacutti - 323200602**

**Guilherme Almeida da Silva - 324100245**

**PROJETO EM DESENVOLVIMENTO DE SISTEMAS**

Denguinho - Banho e tosa

Trabalho apresentado à Universidade Nove de Julho, UNINOVE, em cumprimento parcial às exigências da disciplina de Projeto em Desenvolvimento de Sistemas, sob orientação do Prof. **Leandro Fernandes Da Mota**

**SÃO PAULO**

**2025**

**SUMÁRIO**

**Sumário**

[1.](#_30j0zll) OBJETIVOS 5

[2. MODELAGEM DE PROCESSOS DE NEGÓCIOS 5](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.3znysh7)

[3. PROCESSO DE DESENVOLVIMENTO DE SOFTWARE 5](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.tyjcwt)

[4. MODELAGEM ORIENTADA A OBJETOS 6](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.3dy6vkm)

[4.1. DIAGRAMA DE CLASSE 6](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.1t3h5sf)

[5. DESENVOLVIMENTO ORIENTADO A OBJETOS 6](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.4d34og8)

[5.1. TELA DO SISTEMAS 6](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.2s8eyo1)

[5.2. BANCO DE DADOS 6](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.17dp8vu)

[5.3. CÓDIGOS DO SISTEMA 6](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.35nkun2)

[6. CONCLUSÃO 7](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.26in1rg)

[7. REFERÊNCIAS 7](https://docs.google.com/document/d/14UHu2mX6IPoEpYuwxRw-eqTy-JBubqdi/edit#heading=h.lnxbz9)

### **1. OBJETIVOS**

### Segundo Lakatos e Marconi (2008, p. 220), "a especificação do objetivo de uma pesquisa responde às questões para quê? e para quem?". Partindo dessa premissa, o presente projeto tem como objetivo principal o desenvolvimento de um sistema desktop voltado para a gestão de agendamentos em um petshop, oferecendo uma solução prática e eficiente para o controle de serviços prestados aos animais de estimação.

### O sistema será desenvolvido em linguagem Java, utilizando o ambiente de desenvolvimento NetBeans, com interface gráfica amigável e banco de dados integrado para armazenamento seguro das informações. A aplicação tem como finalidade otimizar os processos internos do petshop, principalmente no que se refere ao agendamento de serviços como banho, tosa, consultas veterinárias, entre outros.

### O público-alvo do sistema são os pequenos e médios petshops que buscam uma ferramenta tecnológica acessível, organizada e intuitiva, capaz de melhorar o atendimento ao cliente, reduzir falhas operacionais e facilitar o gerenciamento do tempo e dos recursos.

#### Objetivo Geral:

### Desenvolver um sistema desktop em Java para agendamento e controle de serviços em petshops, promovendo organização, agilidade e melhoria na experiência dos clientes e colaboradores.

#### Objetivos Específicos:

### Implementar uma interface simples e funcional para cadastro de clientes, pets e serviços.

### Permitir o agendamento de horários com controle de datas e horários disponíveis.

### Criar um banco de dados local para armazenar de forma segura os registros dos agendamentos.

### Gerar relatórios básicos para consulta de serviços realizados e agendamentos futuros.

### Garantir facilidade de uso, visando usuários com pouca familiaridade com tecnologia.

### 

### **2. MODELAGEM DE PROCESSOS DE NEGÓCIOS**

Para representar o fluxo de trabalho de uma funcionalidade essencial do sistema, foi utilizada a notação BPMN (Business Process Model and Notation), que visa facilitar o entendimento dos processos tanto por desenvolvedores quanto por usuários não técnicos. A modelagem foi realizada com foco no processo de **agendamento de serviços** em um petshop.

Abaixo está a representação do processo "Realizar agendamento de um serviço para um pet", que inicia com o atendimento ao cliente e termina com a confirmação do agendamento no sistema:

**Etapas do processo:**

1. O atendente verifica se o cliente já está cadastrado no sistema.
2. Se o cliente não estiver cadastrado, realiza o cadastro.
3. Em seguida, verifica se o pet está cadastrado. Caso contrário, cadastra o pet.
4. O atendente seleciona o serviço desejado (banho, tosa, consulta etc.).
5. O sistema verifica a disponibilidade de horários.
6. O cliente escolhe o melhor horário entre os disponíveis.
7. O agendamento é confirmado e armazenado no sistema.
8. O cliente recebe a confirmação do agendamento.

A modelagem foi criada utilizando a notação BPMN 2.0, no site [bpmn.io](https://bpmn.io/), sendo fiel aos elementos padrões como tarefas, gateways, eventos e fluxos de sequência.

### 

### 

### **3. PROCESSO DE DESENVOLVIMENTO DE SOFTWARE**

Para o desenvolvimento do sistema de agendamentos para petshop, foi adotada a metodologia ágil **Scrum**, que se mostrou adequada por permitir entregas incrementais, reuniões frequentes e uma boa organização do time em ciclos curtos de desenvolvimento.

#### **Papéis da equipe:**

* **Product Owner (PO):** Maria Eduarda  
   Responsável por definir as funcionalidades do sistema, priorizar as tarefas e manter o backlog sempre alinhado aos objetivos do projeto.
* **Scrum Master:** Guilherme Almeida  
   Responsável por garantir que o time siga os princípios do Scrum, facilitando as reuniões e removendo impedimentos que possam comprometer o andamento do projeto.
* **Dev Team:** Luana Sacutti  
   Responsável pelo desenvolvimento técnico do sistema, incluindo a codificação, testes e documentação das funcionalidades.

#### **Duração das Sprints:**

O projeto foi dividido em **4 sprints**, cada uma com duração de **1 semana**, totalizando 4 semanas de desenvolvimento ativo.

#### **Product Backlog:**

| **ID** | **Funcionalidade** | **Prioridade** |
| --- | --- | --- |
| PB01 | Cadastro de cliente | Alta |
| PB02 | Cadastro de pet | Alta |
| PB03 | Cadastro de serviços | Média |
| PB04 | Agendamento de serviços | Alta |
| PB05 | Listagem e consulta de agendamentos | Alta |
| PB06 | Edição e exclusão de agendamentos | Média |
| PB07 | Relatórios de agendamentos | Baixa |
| PB08 | Interface gráfica amigável | Média |
| PB09 | Banco de dados integrado com o sistema | Alta |
| PB10 | Tela de login para segurança do sistema | Baixa |

#### **Planejamento das Sprints:**

**Sprint 1: Levantamento de requisitos e estrutura inicial**

* Definir objetivos e escopo do sistema
* Criar repositório e estrutura base do projeto em Java
* Criar modelo do banco de dados
* Iniciar desenvolvimento da tela de login
* Ferramentas de versionamento e organização do código

**Sprint 2: Funcionalidades básicas de cadastro**

* Implementar cadastro de clientes
* Implementar cadastro de pets
* Implementar cadastro de serviços
* Ajustar integração com o banco de dados

**Sprint 3: Agendamento e consulta**

* Desenvolver tela de agendamento com controle de datas
* Implementar listagem de agendamentos
* Implementar filtros e busca por cliente ou data
* Testes funcionais

**Sprint 4: Ajustes finais e relatórios**

* Criar relatórios simples de agendamentos
* Refatoração de código e correção de bugs
* Finalizar interface gráfica
* Documentação e preparo para entrega

#### **Exemplo de Ata de Daily Scrum**

**Daily Scrum – Sprint 2** **Data:** 12/05/2025  
 **Participantes:** Maria Eduarda (PO), Guilherme Almeida (SM), Luana Sacutti (Dev)  
 **Formato:** Reunião remota via Google Meet  
 **Duração:** 15 minutos

**1. O que foi feito desde a última daily?**

* Luana concluiu o cadastro de clientes e iniciou o cadastro de pets.
* Guilherme fez ajustes no banco de dados para permitir relacionamentos entre cliente e pet.

**2. O que será feito até a próxima daily?**

* Finalizar a tela de cadastro de pets.
* Iniciar o desenvolvimento do cadastro de serviços.

**3. Existe algum impedimento?**

* Foi identificado um problema com a persistência de dados dos pets, que será revisado por Guilherme com apoio da PO para garantir a regra de negócio.

### **4. MODELAGEM ORIENTADA A OBJETOS**

A modelagem orientada a objetos é fundamental para a estruturação e o desenvolvimento de sistemas, pois permite representar de forma clara os elementos e comportamentos do sistema, baseando-se em conceitos como classes, objetos, atributos, métodos, herança e encapsulamento.

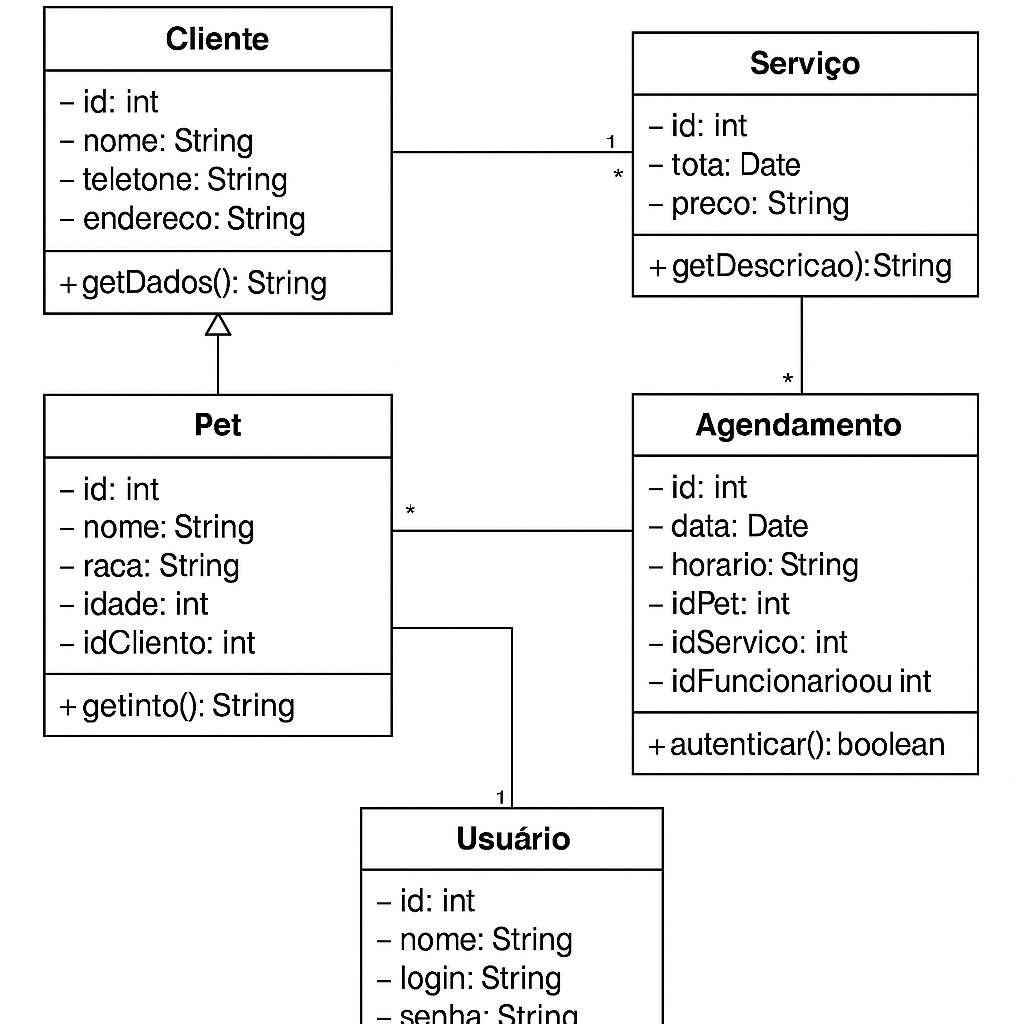
A seguir, são apresentados dois dos principais artefatos utilizados nessa abordagem: o **Diagrama de Classe**, que representa a estrutura estática do sistema, e o **Diagrama de Casos de Uso**, que descreve as interações entre os usuários e as funcionalidades do sistema.

### **4.2. DIAGRAMA DE CLASSE**

O diagrama de classes do sistema de agendamentos para petshop apresenta as principais entidades envolvidas e os relacionamentos entre elas. A modelagem foi feita de forma a garantir clareza na estrutura e reutilização de código, respeitando os princípios da orientação a objetos.

As principais classes modeladas são:

* **Cliente**: representa o tutor do animal.
* **Pet**: representa o animal de estimação do cliente.
* **Serviço**: representa os serviços oferecidos pelo petshop (banho, tosa, consulta etc.).
* **Agendamento**: representa a marcação de um serviço em determinada data e horário.
* **Usuário**: representa os funcionários que terão acesso ao sistema.



### **4.1. DIAGRAMA DE CASOS DE USO**

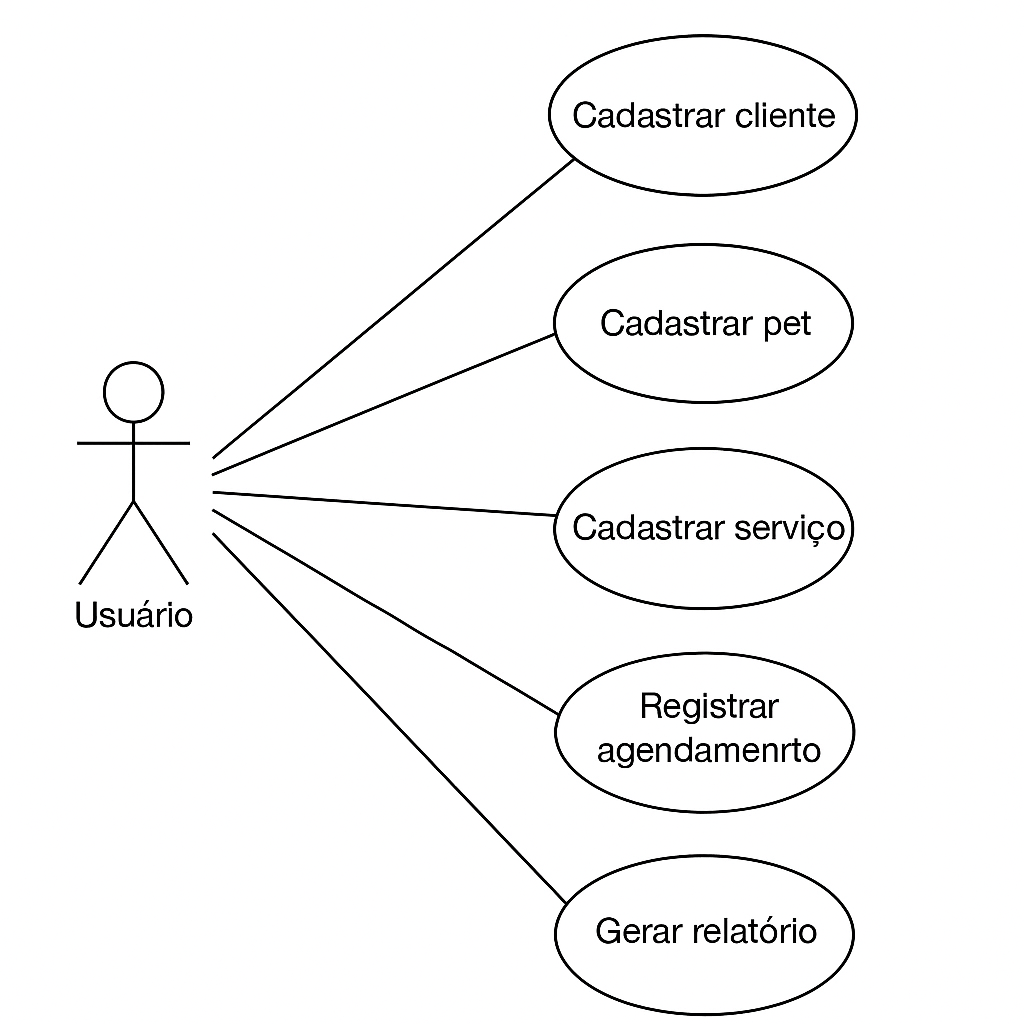
O diagrama de casos de uso mostra os principais cenários de interação entre os **atores** (usuários do sistema) e as **funcionalidades**. O objetivo é representar o que o sistema deve fazer do ponto de vista do usuário, sem entrar em detalhes de implementação.

Atores principais:

* **Funcionário**: responsável por cadastrar clientes, pets e agendar serviços.
* **Administrador** (opcional): responsável por acessar relatórios e configurar o sistema.

Casos de uso principais:

* Cadastrar cliente
* Cadastrar pet
* Cadastrar serviço
* Realizar agendamento
* Consultar agendamentos
* Editar ou cancelar agendamento
* Gerar relatórios (opcional)



## **5. DESENVOLVIMENTO ORIENTADO A OBJETOS**

O sistema desktop foi desenvolvido em **Java**, utilizando o paradigma de **Programação Orientada a Objetos (POO)**. Com isso, foram aplicados conceitos como encapsulamento, herança, abstração e polimorfismo para organizar as funcionalidades e promover a manutenção e evolução do projeto.

A proposta do sistema é atender às necessidades de um **petshop**, com foco em cadastro de clientes, animais, serviços e agendamento de atendimentos.

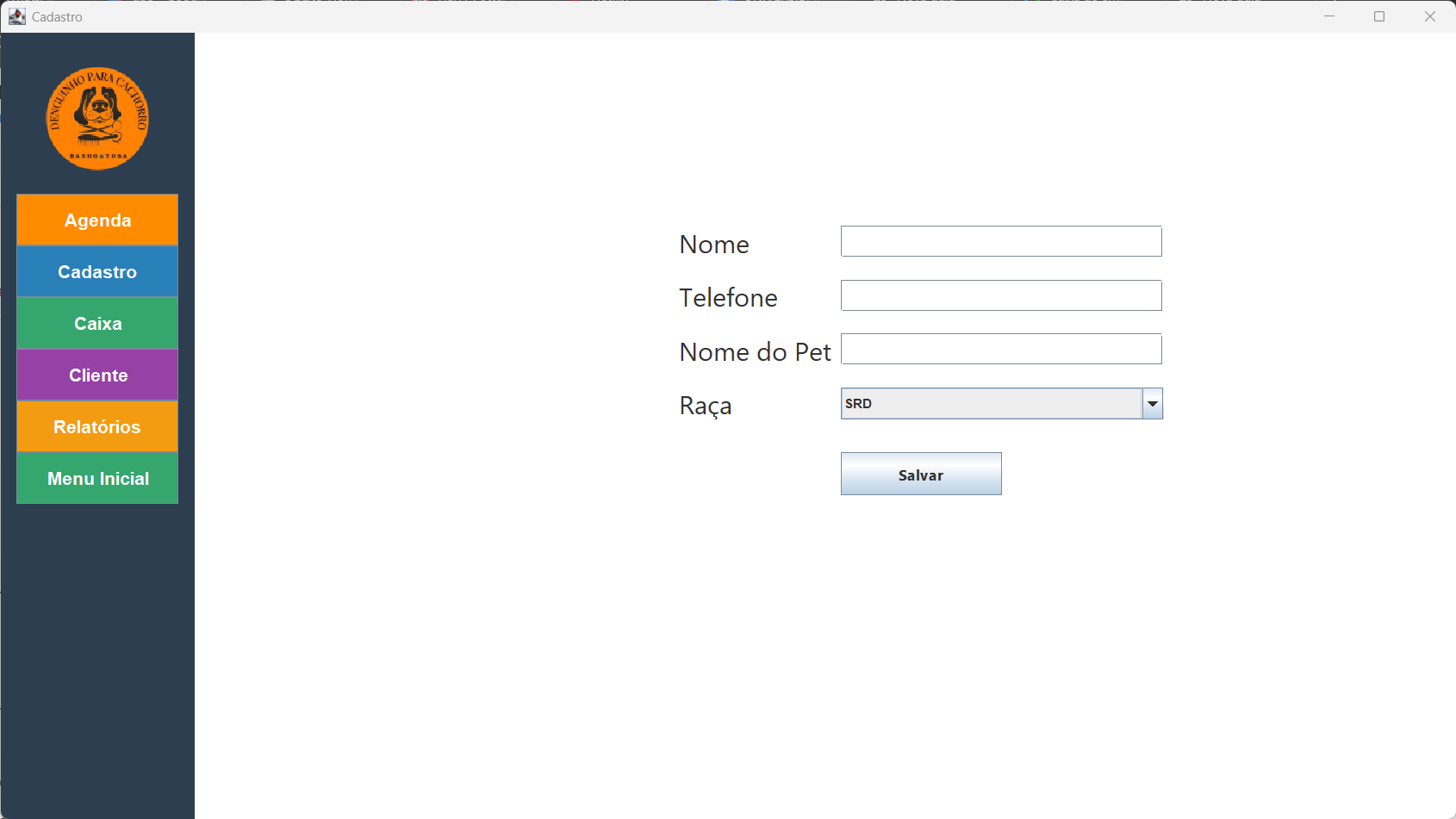
### **5.1. TELAS DO SISTEMA**

O sistema é composto pelas seguintes telas funcionais:

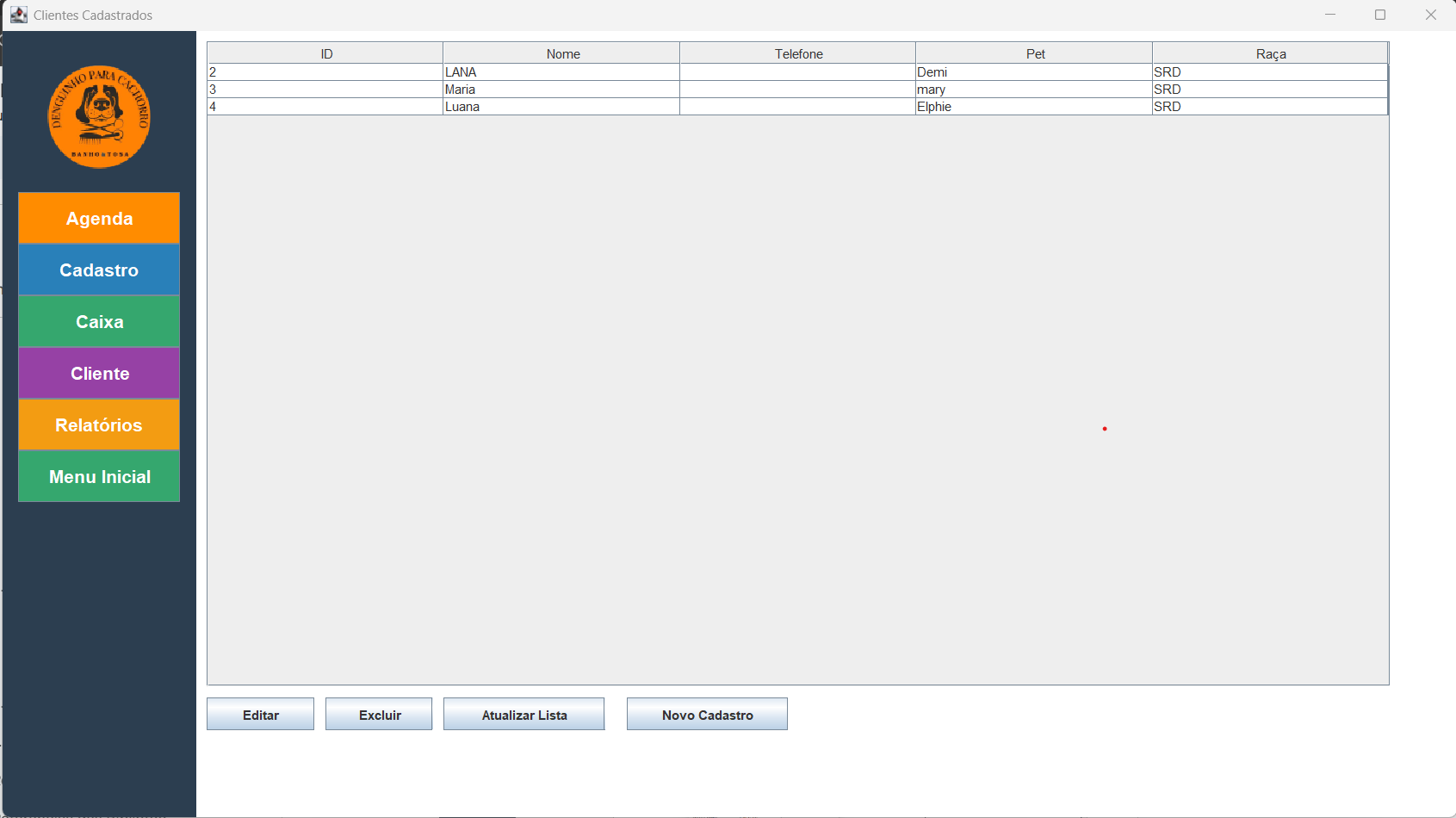
* **Tela Principal (Menu)** Apresenta ao usuário as principais funcionalidades do sistema, como acesso aos cadastros, agendamentos e relatórios (se implementado).

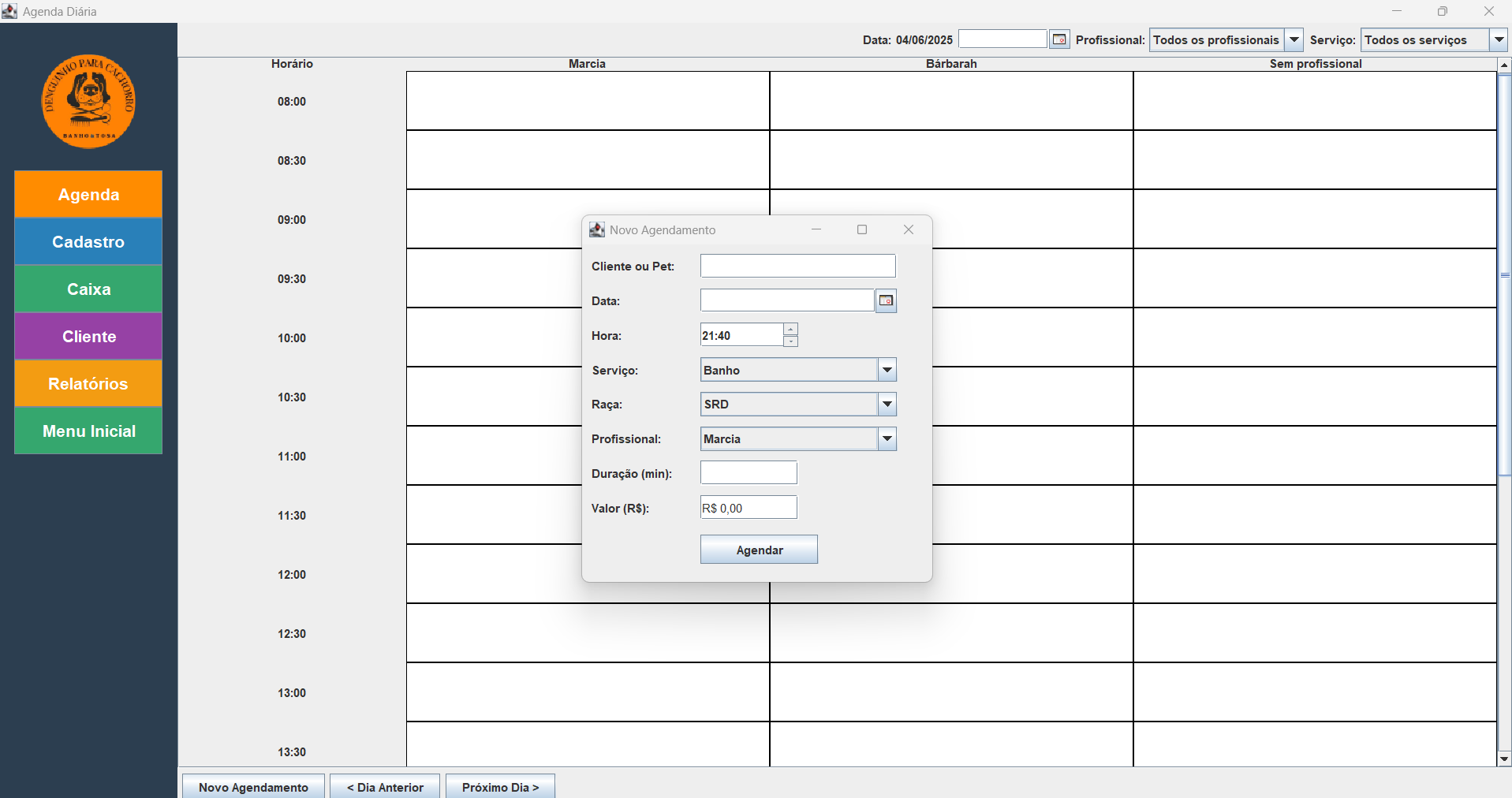


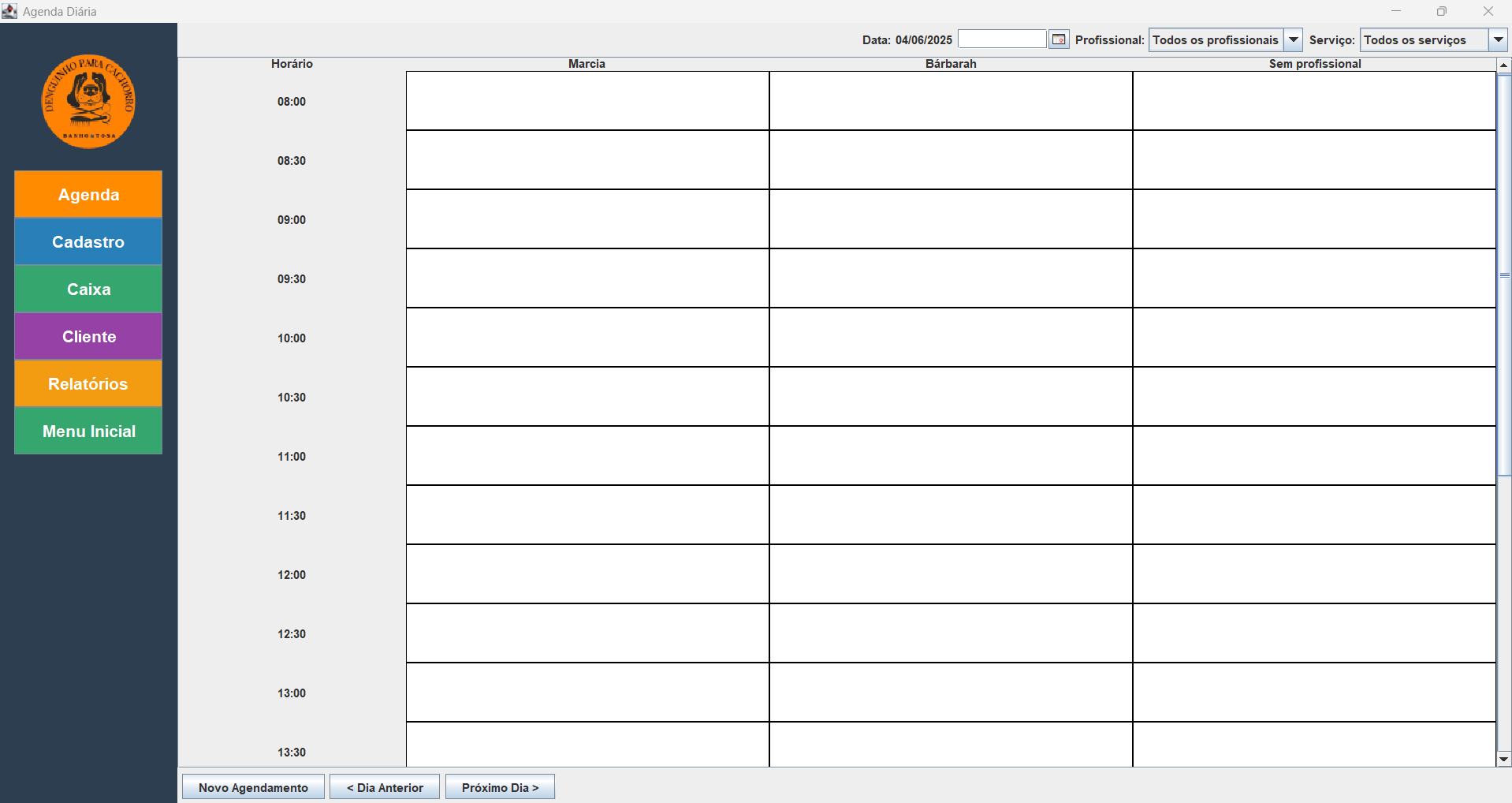
* **Tela de Cadastro de Cliente** Permite o registro de novos clientes, informando nome, telefone e endereço. Os dados ficam disponíveis para vínculo com os pets.



* **Tela de Cadastro de Pet** Permite cadastrar animais de estimação, associando-os a um cliente já existente. Os campos incluem nome, raça e idade.

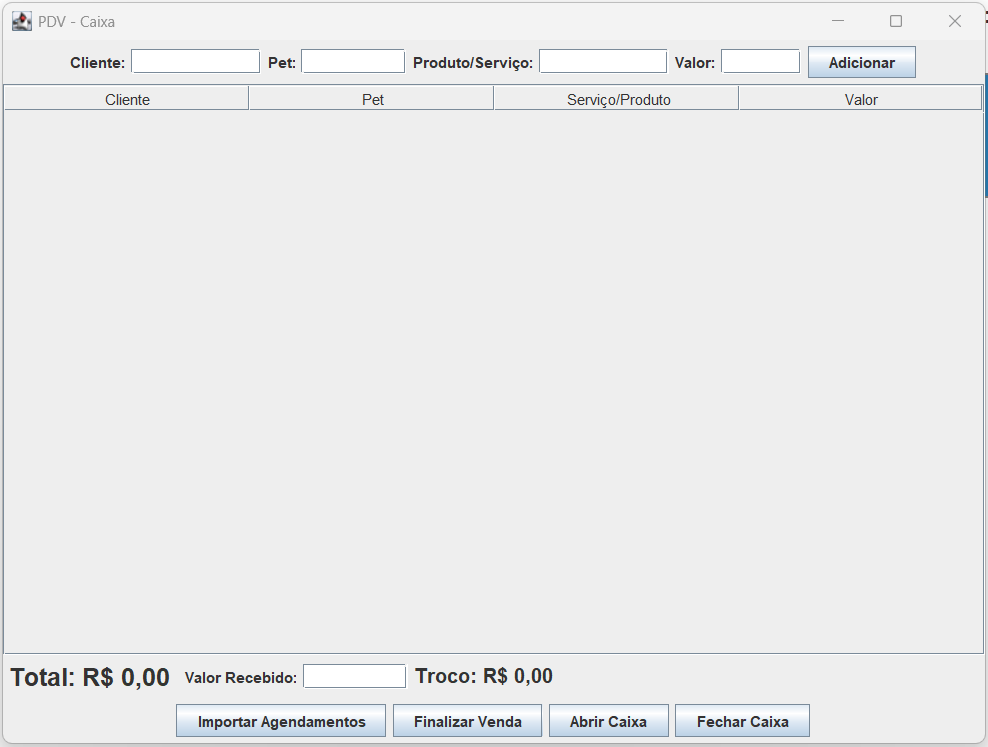


* **Tela de Agendamento** Permite registrar agendamentos de serviços para os pets, selecionando cliente, pet, serviço, data e horário.  
  
* **Tela de Consulta de Agendamentos** Permite visualizar todos os agendamentos realizados, com filtros por data, cliente ou pet.



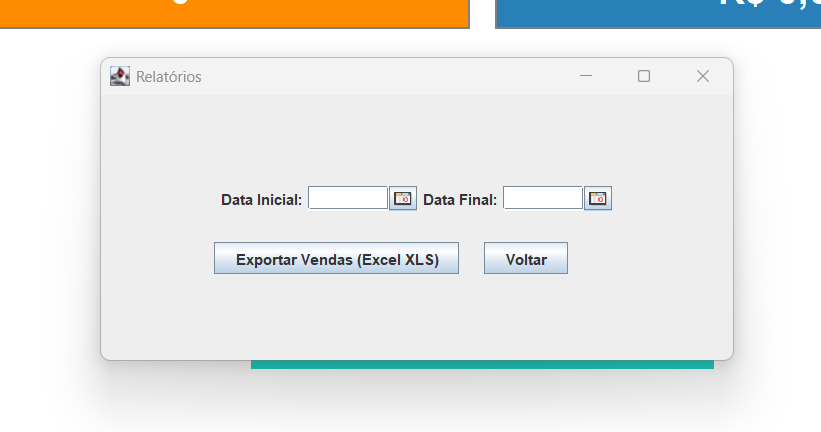
* **Tela de Caixa**

Permite registrar as vendas dos serviços prestados.



* **Tela de relatórios**

Permite criar relatórios com base nos agendamentos, saída de caixa e cadastro de clientes.



### **Link do código fonte**

<https://drive.google.com/drive/folders/1bYPjp2ls8M2xeXmnaxaYewPrdegG-c0V?usp=drive_link>

### **5.2. BANCO DE DADOS**

AgendamentoDAO.Java

public class AgendamentoDAO {

private static final String URL = "jdbc:sqlite:banco/banco.db";

private Connection conectar() throws SQLException {

return DriverManager.getConnection(URL);

}

public void salvar(Agendamento ag) {

String sql = "INSERT INTO agendamentos (cliente, nomePet, raca, dataHora, servico, duracao, profissional, valor) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

try (Connection conn = conectar(); PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setString(1, ag.getCliente());

stmt.setString(2, ag.getNomePet());

stmt.setString(3, ag.getRaca());

stmt.setTimestamp(4, ag.getDataHora());

stmt.setString(5, ag.getServico());

stmt.setInt(6, ag.getDuracao());

stmt.setString(7, ag.getProfissional());

stmt.setDouble(8, ag.getValor());

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

CaixaDAO.java

public class CaixaDAO {

private static final String URL = "jdbc:sqlite:banco/banco.db";

public CaixaDAO() {

criarTabelaCaixaSeNaoExistir();

}

private Connection conectar() throws SQLException {

return DriverManager.getConnection(URL);

}

public void criarTabelaCaixaSeNaoExistir() {

String sql = """

CREATE TABLE IF NOT EXISTS caixa (

id INTEGER PRIMARY KEY AUTOINCREMENT,

dataHoraAbertura TEXT NOT NULL,

dataHoraFechamento TEXT,

saldoInicial REAL NOT NULL,

saldoFinal REAL,

aberto BOOLEAN NOT NULL

);

""";

ClienteDAO.java

public class ClienteDAO {

private static final String URL = "jdbc:sqlite:banco/banco.db";

public ClienteDAO() {

criarTabelaSeNaoExistir();

}

private void criarTabelaSeNaoExistir() {

String sql = "CREATE TABLE IF NOT EXISTS clientes (" +

"id INTEGER PRIMARY KEY AUTOINCREMENT," +

"nomeCliente TEXT NOT NULL," +

"telefone TEXT," +

"nomePet TEXT," +

"raca TEXT)";

try (Connection conn = DriverManager.getConnection(URL);

Statement stmt = conn.createStatement()) {

stmt.execute(sql);

} catch (SQLException e) {

e.printStackTrace();

}

}

MovimentoDAO.java

public class MovimentoDAO {

private static final String URL = "jdbc:sqlite:banco/banco.db";

private Connection conectar() throws SQLException {

return DriverManager.getConnection(URL);

}

public void salvar(Movimento movimento) throws SQLException {

String sql = "INSERT INTO movimento (id\_Caixa, tipo, valor, descricao, data\_Hora) VALUES (?, ?, ?, ?, ?)";

try (Connection conn = conectar(); PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, movimento.getIdCaixa());

stmt.setString(2, movimento.getTipo());

stmt.setDouble(3, movimento.getValor());

stmt.setString(4, movimento.getDescricao());

stmt.setString(5, movimento.getDataHora().toString());

stmt.executeUpdate();

}

}

## **6. CONCLUSÃO**

O desenvolvimento deste projeto proporcionou à equipe a oportunidade de aplicar na prática os conhecimentos adquiridos ao longo do curso, especialmente em áreas como modelagem de processos, desenvolvimento orientado a objetos, uso da linguagem Java e estruturação de banco de dados.

Com a criação de um sistema para gestão de um petshop, foi possível compreender com mais profundidade o ciclo de vida de um software, desde a definição dos objetivos e levantamento de requisitos até a implementação das funcionalidades e testes. A utilização do Scrum como metodologia ágil também contribuiu para a organização das tarefas e o trabalho em equipe, permitindo entregas incrementais e melhoria contínua.

Através da aplicação dos conceitos de orientação a objetos, o sistema alcançou um bom nível de modularidade e clareza na estrutura de classes. A modelagem BPMN facilitou a visualização dos processos principais e o banco de dados relacional garantiu o armazenamento e integridade dos dados manipulados.

De modo geral, o projeto atingiu seus objetivos, entregando uma solução funcional, com potencial para evolução futura. Como ponto de vista final, pode-se afirmar que a experiência prática de desenvolver um sistema completo fortaleceu significativamente a base técnica da equipe e contribuiu para a formação profissional dos participantes.

## **7. Bibliotecas Utilizadas no Projeto**

O projeto **PetshopAgendamentos** utiliza as seguintes bibliotecas Java, que estão localizadas na pasta lib do projeto e são referenciadas automaticamente na execução do arquivo principal Denguinho.jar:

* **jcalendar-1.4.jar** Biblioteca para componentes gráficos de calendário (seleção de datas).  
   Utilizada para facilitar o agendamento de datas no sistema.
* **jxl.jar** Java Excel API.  
   Permite leitura e escrita de arquivos .xls (Excel) diretamente em Java, usada para importação e exportação de dados em planilhas.
* **sqlite-jdbc-3.49.1.0.jar** Driver JDBC para banco de dados SQLite.  
   Utilizado para conexão e manipulação do banco de dados local do sistema, que é um banco SQLite.

## **Referências Técnicas (Modelo ABNT para bibliografia de software e manuais)**

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 6023: Informação e documentação: referências: elaboração. Rio de Janeiro, 2002.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. *Metodologia do trabalho científico: procedimentos básicos; pesquisa bibliográfica, projeto e relatórios; publicações e trabalhos científicos*. 5. ed. São Paulo: Atlas, 2001.

SISTEMA DE BIBLIOTECAS PROF. JOSÉ STORÓPOLI. Universidade Nove de Julho. *Manual para Elaboração de Trabalhos Acadêmicos de acordo com ABNT*. Disponível em:<http://docs.uninove.br/arte/pdfs/Manual-Elaboracao-de-Trabahos-ABNT.pdf>. Acesso em: 30 set. 2016.

## **Referências das Bibliotecas utilizadas no projeto**

**jCalendar** TOEDTER, Kai. *JCalendar*. Versão 1.4. Disponível em: https://toedter.com/jcalendar/. Acesso em: 03 jun. 2025.

**Java Excel API (JXL)** BAXTER, Andy. *Java Excel API*. Versão 2.6.12. Disponível em: https://sourceforge.net/projects/jexcelapi/. Acesso em: 03 jun. 2025.

**SQLite JDBC Driver** Xerial. *SQLite JDBC Driver*. Versão 3.49.1. Disponível em:<https://github.com/xerial/sqlite-jdbc>. Acesso em: 03 jun. 2025.