INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
CAMPUS SÃO PAULO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ALEX PASSOS DA SILVA
GIULIA MATTOS GOES BRAGA

SISTEMA DE CONTROLE PARA UMA CLÍNICA DE DOAÇÃO DE SANGUE

SÃO PAULO
2025

ALEX PASSOS DA SILVA
GIULIA MATTOS GOES BRAGA

SISTEMA DE CONTROLE PARA UMA CLÍNICA DE DOAÇÃO DE SANGUE

Trabalho apresentado ao Programa do Curso de Bacharelado em Sistemas de Informação, do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Campus de São Paulo, como requisito parcial para aprovação na disciplina SPOBDD2 – Banco de Dados, sob orientação do Professor Doutor Francisco Verissimo Luciano.

SÃO PAULO
2025

Resumo

A proposta deste trabalho é apresentar um sistema para uma clínica de doação de sangue, com foco na implementação prática de um banco de dados que dê suporte às principais funcionalidades esperadas pelos usuários (profissionais ou pacientes). O sistema busca tornar os processos internos mais ágeis, contribuindo tanto para a produtividade da equipe quanto para a experiência do cliente.

Palavras-chaves: Clínica de doação de sangue, banco de dados, gestão de informações.

Sumário

Introdução

Com a expansão dos meios digitais em praticamente todos os âmbitos da vida da população global, tornou-se imprescindível a adaptação dos processos de organizações e empresas para esse contexto. Muitos negócios hoje operam de maneira integralmente online por conta das novas formas de trabalho, consumo e estilo de vida em geral, que emergiram graças ao avanço das tecnologias da informação.

No caso das empresas da área de saúde, não aconteceu de forma diferente. Mesmo que ainda não seja possível realizar todo tipo de procedimento médico remotamente, os serviços de saúde também vêm se modernizando com o passar do tempo para acompanhar essas transformações. Nos últimos anos, tornaram-se comuns as consultas e agendamentos online, e em muitos casos, o principal ou único meio disponível para acesso ao atendimento. Diante dessa tendência, torna-se essencial visar a usabilidade do sistema, garantindo que grupos com menor familiaridade com tecnologias, como a população idosa, não sejam excluídos ou tenham sua autonomia reduzida no acesso aos serviços.

Nesse contexto, desenvolvemos um sistema voltado para uma clínica de doação de sangue, com o objetivo de ampliar o alcance dos serviços oferecidos e, consequentemente, contribuir para o aumento do número de doadores.

Solução proposta

Para o sistema proposto, a usabilidade foi tratada como um requisito central, visando assegurar que todos os potenciais usuários do website (doadores e profissionais) possam utilizá-lo de forma simples e eficiente. Além disso, priorizamos desempenho e confiabilidade por meio da adoção do MySQL como Sistema Gerenciador de Banco de Dados (SGBD), bem como o desenvolvimento de um layout responsivo utilizando HTML, CSS e JavaScript.

A definição rigorosa dos requisitos funcionais e não funcionais orientou a construção de um sistema capaz de oferecer uma experiência satisfatória ao usuário, favorecendo sua fidelização. Ao mesmo tempo, buscou-se otimizar a rotina dos profissionais da clínica, contribuindo para processos internos mais ágeis e eficazes.

Objetivos

Objetivo geral

Apresentar um sistema para uma clínica de doação de sangue que atenda as necessidades levantadas por doadores e pela equipe da clínica, como gestão e triagem de doadores, agendamento 100% online e acessibilidade à informação.

Objetivos específicos

- Disponibilizar agendamento de doações;
- Registrar e consultar o histórico de doações de cada usuário;
- Controlar o intervalo mínimo entre doações para cada doador;
- Permitir o cadastro e exclusão de campanhas de doação por administradores;
- Registrar triagens clínicas e doações por profissionais de saúde;
- Garantir a segurança e confidencialidade dos dados dos usuários;
- Oferecer uma interface acessível para diferentes perfis de usuários.

Tecnologias

A escolha das tecnologias para o desenvolvimento do sistema foi guiada pela necessidade de criar uma solução eficiente, segura e de rápida implementação, considerando o tempo disponível e a complexidade do projeto. Optamos por ferramentas modernas, amplamente utilizadas no mercado e com grande suporte da comunidade, o que facilita a resolução de problemas e garante maior produtividade no desenvolvimento.

Back-end

Para o desenvolvimento do back-end, utilizamos JavaScript com a plataforma Node.js, que permite a execução de código no lado do servidor. Em conjunto, adotamos o framework Express.js, que fornece uma estrutura leve e flexível para a criação de rotas, gerenciamento de requisições HTTP e desenvolvimento de APIs RESTful.

O uso de Node.js e Express.js também favorece a produtividade, já que a mesma linguagem (JavaScript) pode ser aplicada tanto no front-end quanto no back-end, reduzindo a curva de aprendizado e acelerando o desenvolvimento. Além disso, a vasta quantidade de bibliotecas disponíveis no ecossistema do npm permite integrar funcionalidades como a autenticação de usuários de maneira prática.

Banco de dados

O banco de dados utilizado para a implementação do nosso sistema será o MySQL, um SGBD que se alinha aos objetivos do projeto por seu desempenho e confiabilidade. Essas características são essenciais para cumprir o objetivo vigente pois, em se tratando de uma clínica, é necessário que seu sistema possa comportar um número alto de usuários simultâneos, principalmente em tempos de campanhas de doação de sangue, em que se espera um aumento no número de usuários. É indispensável que o sistema funcione corretamente e de forma consistente para que não haja desincentivo do seu uso e consequente perda de doadores.

A comunicação entre o banco de dados e o sistema será feita através do pacote mysql2, específico para o SGBD utilizado.

Controle de versão

O repositório do projeto foi hospedado no GitHub, permitindo o acesso remoto ao código-fonte, bem como o acompanhamento das modificações realizadas ao longo do tempo. O uso dele contribuiu para a manutenção da qualidade do software, visto que possibilita a visualização do histórico de alterações, integração das contribuições de forma segura pelos branches, reversão de alterações problemáticas, comparação de versões e documentação da evolução do sistema de maneira estruturada.

Front end

A interface do usuário foi desenvolvida com foco na usabilidade, buscando ser amigável e intuitiva. Essa preocupação é especialmente relevante em uma aplicação de clínica de doação de sangue, cujo público-alvo é amplo, abrangendo indivíduos com idades entre 16 e 69 anos.

Utilizamos HTML, CSS e JavaScript para estruturar, estilizar e adicionar interatividade às páginas do site.

Infraestrutura

A infraestrutura do sistema foi planejada para garantir disponibilidade, desempenho e segurança durante o desenvolvimento e utilização da aplicação. O ambiente foi configurado para suportar a execução do back-end em Node.js, o gerenciamento dos dados por meio do MySQL e a comunicação entre essas camadas através do pacote mysql2.

O versionamento do código-fonte foi realizado com GitHub, permitindo organização, rastreabilidade e integração entre diferentes etapas do desenvolvimento.

Cronograma de atividades

- Semana 1 (até 06/10) - Levantamento de requisitos e análise de requisitos;
- Semana 2 (até 13/10) - Objetivos, tecnologias, banco de dados e primeira entrega;
- Semana 3 (até 03/11) - Projeto de software;
- Semana 4 (até 24/11) - Implementação e testes;
- Semana 5 (até 05/12) - Elaboração de vídeo e demais entregáveis.

Revisão da literatura

Requisitos

Para definir os requisitos do sistema, realizamos uma pesquisa exploratória baseada na análise de sites e plataformas já existentes voltadas para agendamento e gestão de doações. Através dessa investigação online, foi possível identificar padrões de funcionalidades, boas práticas de usabilidade e recursos essenciais que atendem às necessidades dos usuários e profissionais envolvidos nesse tipo de serviço.

Análise de requisitos

O sistema proposto tem como objetivo principal gerenciar o processo de doações de forma eficiente, segura e acessível, atendendo tanto usuários doadores quanto profissionais de saúde e administradores. A análise dos requisitos funcionais e não funcionais permite compreender as funcionalidades essenciais e os critérios de qualidade esperados.

Requisitos Funcionais

- O usuário deve conseguir fazer o próprio cadastro no sistema com os dados pessoais e, depois, fazer o próprio login no sistema;
- O usuário pode visualizar o seu histórico de doações;
- O sistema deve verificar a elegibilidade do usuário, checando o peso, a idade e as condições de saúde. Caso o usuário não seja elegível, ele deve ser impedido de marcar horários no sistema;
- O usuário pode atualizar seus dados pessoais, como endereço, telefone e condições de saúde;
- O sistema deve permitir que o usuário agende uma doação em datas e horários disponíveis;
- O sistema deve registrar cada doação realizada pelo usuário;

- O sistema deve impedir o agendamento de uma nova doação caso o tempo mínimo entre doações de um mesmo usuário ainda não tenha passado;
- O sistema deve permitir que administradores cadastrem e excluam campanhas de doação;
- O sistema deve permitir que profissionais de saúde registrem a triagem clínica e doação;
- O sistema deve permitir que administradores cadastrem novos profissionais de saúde;

Requisitos Não Funcionais

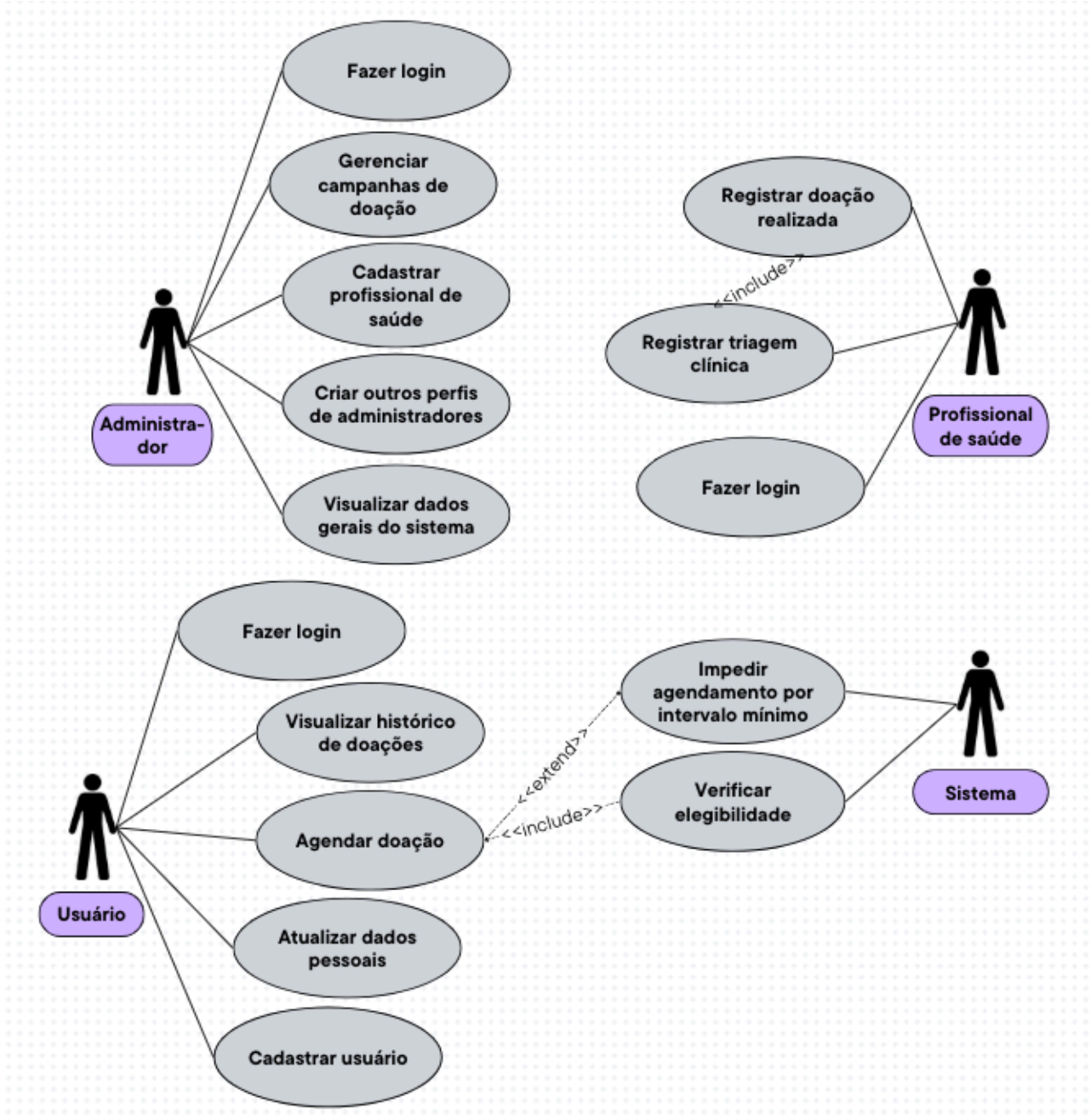
- Desempenho: O sistema deve responder às requisições dos usuários em até 2s e suportar o acesso de vários usuários simultâneamente;
- Disponibilidade: O sistema deve ficar no ar 24h por dia, em todos os dias da semana;
- Segurança: O sistema deve garantir a confidencialidade dos dados dos usuários;
- Usabilidade: Deve ser acessível para usuários com diferentes níveis de conhecimento tecnológico;
- Manutenibilidade: Boa documentação e boas práticas para desenvolver o código são essenciais para atualizar e corrigir erros no futuro, sem complicações;
- Portabilidade: Deve funcionar em diferentes navegadores e dispositivos.

Projeto de software

Casos de uso

Diagrama

Figura 1 - Diagrama de casos de uso do sistema da Clínica de doação de sangue.



Fonte: elaboração própria.

Definições do UC

UC01 - Cadastrar usuário

Ator: Usuário

Objetivo: Criar um novo perfil no sistema.

Pré-condições: Nenhuma.

Pós-condições: Usuário cadastrado e armazenado na base de dados.

Fluxo principal

1. O usuário acessa a página de cadastro.
2. O usuário informa seus dados pessoais (nome, email, senha, idade, peso, condições de saúde, etc.).
3. O sistema valida os dados informados.
4. O sistema cria o cadastro.
5. O sistema confirma a criação da conta.

Fluxos alternativos

- **E-mail já cadastrado:** O sistema detecta e-mail duplicado → Exibe mensagem de erro → Retorna ao passo 2.
- **Dados inválidos:** Sistema encontra campos inválidos/ausentes → Solicita correção → Retorna ao passo 2.

UC02 – Login de usuário

Ator: Usuário

Objetivo: Acessar o sistema.

Pré-condições: Usuário deve estar cadastrado.

Pós-condições: Sessão iniciada.

Fluxo principal

1. O usuário acessa a página de login.
2. O usuário informa e-mail e senha.
3. O sistema valida as credenciais.
4. O usuário obtém acesso ao sistema.

Fluxos alternativos

- **Senha incorreta:** Sistema rejeita → Exibe mensagem → Retorna ao passo 2.
- **Conta inexistente:** Sistema alerta que o usuário não existe → Oferece criação de conta.

UC03 - Login de administrador

Ator: Administrador

Objetivo: Acessar o sistema com privilégios de administrador.

Pré-condições: Administrador cadastrado no sistema.

Pós-condições: Sessão iniciada.

Fluxo principal

1. O usuário acessa a área de login do administrador;
2. Usuário informa email e senha;
3. O sistema valida;
4. Usuário obtém acesso ao sistema.

Fluxo alternativo

- **Dados incorretos:** Exibição de mensagem de erro.

UC04 - Login de profissional

Ator: Profissional

Objetivo: Acessar o sistema.

Pré-condições: O profissional deve estar cadastrado.

Pós-condições: O profissional obtém acesso ao sistema.

Fluxo principal

5. O usuário acessa a área de login do profissional;
6. Usuário informa email e senha;
7. O sistema valida;
8. Usuário obtém acesso ao sistema.

Fluxo alternativo

- **Dados incorretos:** mensagem de erro exibida.

UC05 - Cadastro de novos administradores

Ator: Administrador

Objetivo: adicionar novos administradores ao sistema.

Pré-condições: O administrador deve estar logado.

Pós-condições: Nenhuma.

Fluxo principal

1. O administrador acessa a área restrita;
2. O administrador seleciona a opção “Adicionar novo administrador";

3. O administrador informa os dados;
4. Sistema valida.

UC06 – Visualizar histórico de doações

Ator: Usuário

Objetivo: Consultar registros de doações anteriores.

Pré-condições: Usuário logado.

Pós-condições: Nenhuma alteração de estado.

Fluxo principal

1. O usuário acessa a área de histórico.
2. O sistema busca todas as doações do usuário.
3. O sistema apresenta lista com datas, volumes e tipo sanguíneo.

UC07 – Verificar elegibilidade do usuário

Ator: Sistema

Objetivo: Determinar se o usuário pode agendar doação.

Pré-condições: Usuário deve ter dados pessoais cadastrados.

Pós-condições: Status de elegibilidade atualizado.

Fluxo principal

1. O sistema analisa peso, idade e condições de saúde.
2. O sistema determina se é elegível ou inelegível.

Fluxo alternativo

- **Usuário não elegível:** Sistema marca usuário como não elegível e impede agendamentos.

UC08 – Atualizar dados pessoais

Ator: Usuário

Objetivo: Atualizar telefone, endereço, saúde etc.

Pré-condições: Usuário logado.

Pós-condições: Dados atualizados no sistema.

Fluxo principal

1. O usuário acessa a área de edição.
2. O usuário altera os campos desejados.
3. O sistema valida as mudanças.
4. O sistema salva os dados.

5. O sistema confirma a atualização.

Fluxo alternativo

- **Dados inválidos:** Sistema rejeita → Solicita correção.

UC09 – Agendar doação

Atores: Usuário

Objetivo: Agendar uma doação em data e horário disponíveis.

Pré-condições:

- Usuário logado.
- Usuário elegível.

Pós-condições: Agendamento registrado.

Fluxo principal

1. O usuário acessa a agenda.
2. O sistema exibe datas/horários disponíveis.
3. O usuário escolhe data e horário.
4. O sistema verifica se o tempo mínimo entre doações já passou.
5. O sistema cria o agendamento.
6. O sistema envia confirmação (e-mail/SMS).

Fluxos alternativos

- **Usuário inelegível:** Sistema impede acesso → Exibe motivo.
- **Intervalo mínimo não cumprido:** Sistema bloqueia agendamento.

UC10 – Registrar doação realizada

Ator: Profissional de saúde

Objetivo: Registrar oficialmente uma doação após realização.

Pré-condições: Deve existir um agendamento aprovado.

Pós-condições: Doação registrada no histórico do usuário.

Fluxo principal

1. O profissional registra no sistema a doação realizada.
2. O sistema solicita dados: data, volume, tipo sanguíneo.
3. O sistema salva o registro.
4. O histórico do usuário é atualizado.

UC11 – Impedir agendamento por intervalo mínimo

Ator: Sistema

Objetivo: Bloquear agendamento quando intervalo mínimo não foi cumprido.

Pré-condições: Usuário tem histórico de doações.

Pós-condições: Agendamento não é criado.

Fluxo principal

1. Usuário tenta agendar.
2. O sistema compara a data atual com a data da última doação.
3. Caso o intervalo seja insuficiente, o sistema bloqueia.
4. Mensagem de impedimento é apresentada.

UC12 – Gerenciar campanhas de doação

Ator: Administrador

Objetivo: Criar e excluir campanhas.

Pré-condições: Administrador autenticado.

Pós-condições: Campanhas atualizadas.

Fluxo principal

1. Administrador acessa menu de campanhas.
2. Escolhe entre adicionar ou excluir.
3. Sistema valida os dados.
4. Sistema salva mudanças.
5. Sistema confirma operação.

UC13 – Registrar triagem clínica

Ator: Profissional de saúde

Objetivo: Registrar triagem.

Pré-condições: Profissional autenticado; usuário com agendamento.

Pós-condições: Status atualizado (aprovado/reprovado).

Fluxo principal

1. Profissional seleciona agendamento do usuário.
2. Profissional informa os resultados da triagem.
3. Sistema registra as informações.
4. Profissional marca aprovado ou recusado.
5. Sistema salva e confirma.

UC14 – Cadastrar profissional de saúde

Ator: Administrador

Objetivo: Adicionar novo profissional ao sistema.

Pré-condições: Administrador logado.

Pós-condições: Profissional registrado.

Fluxo principal

1. Administrador abre tela de cadastro de profissional.
2. Preenche dados (nome, registro profissional, contato).
3. Sistema valida.
4. Sistema salva.

Fluxo alternativo

- **Dados inválidos:** mostra mensagem de erro e solicita o preenchimento adequado.

UC15 - Visualizar dados gerais do sistema

Ator: Administrador

Objetivo: Consultar números relacionados às atividades registradas no sistema.

Pré-condições: Administrador logado.

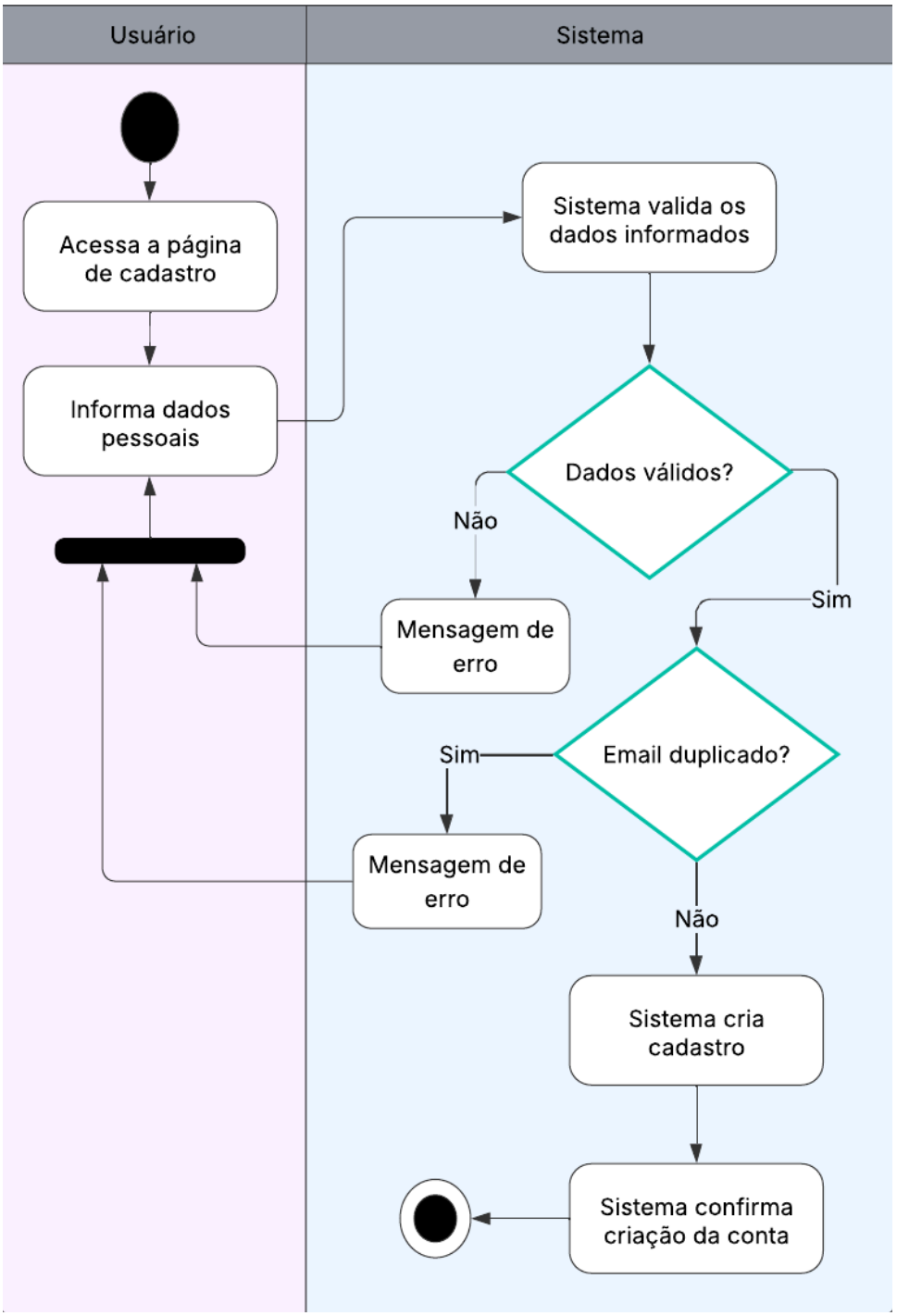
Pós-condições: Nenhuma.

Fluxo principal

1. Administrador entra na área restrita;
2. Consulta área “Dados gerais”.

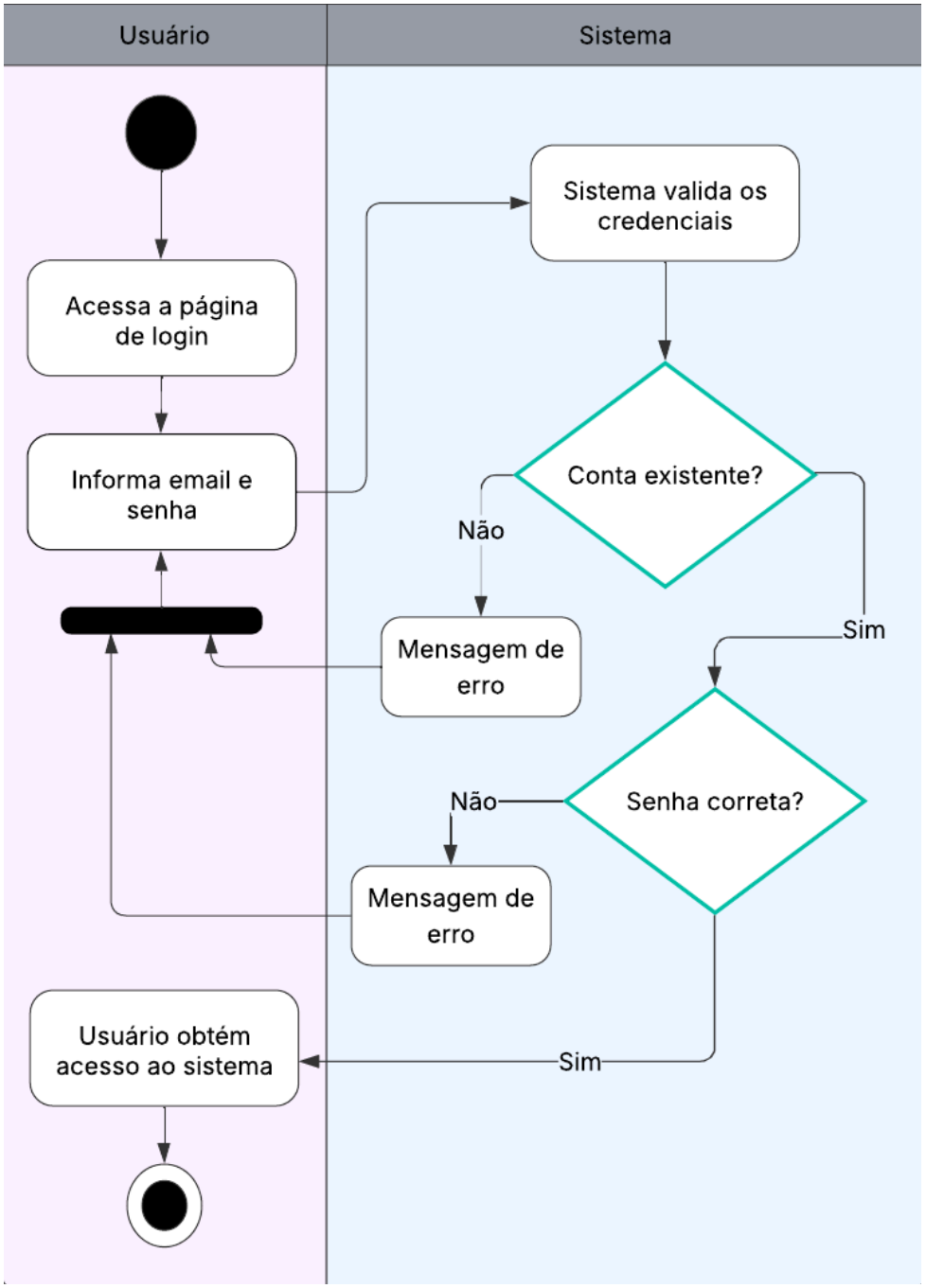
Diagramas de atividades

Figura 2 - Cadastro de usuários (UC01).



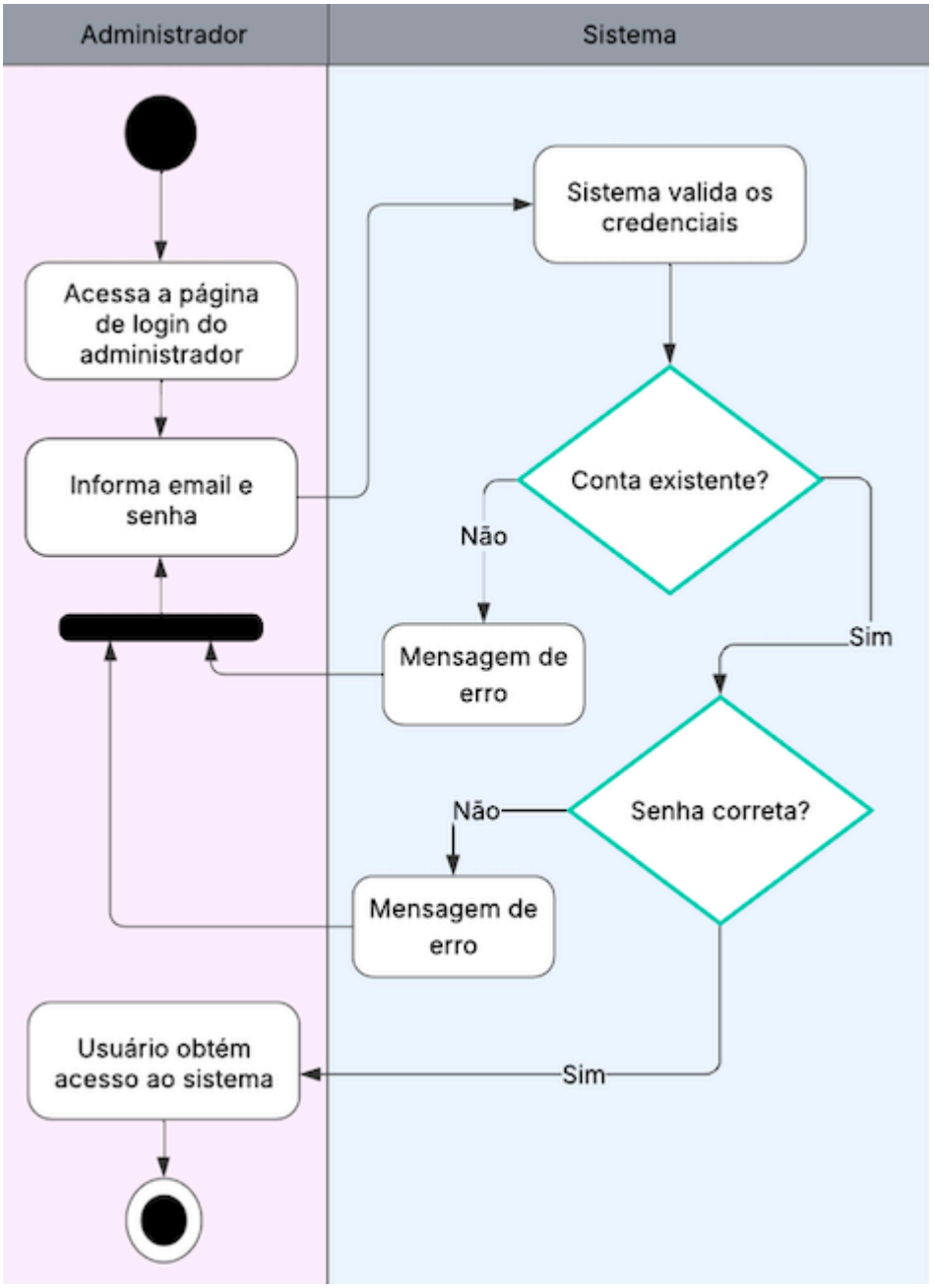
Fonte: elaboração própria.

Figura 3 - Login de usuários (UC02).



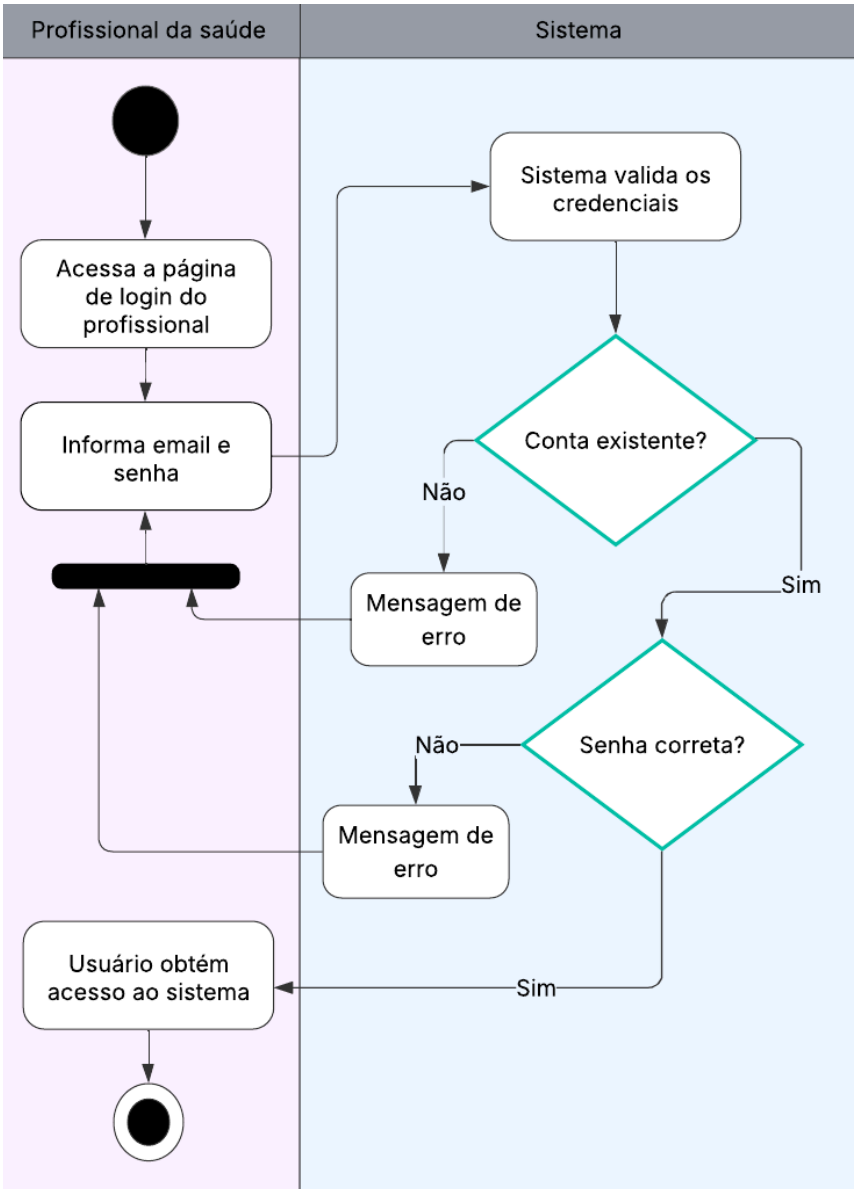
Fonte: elaboração própria.

Figura 4 - Login do administrador (UC03).



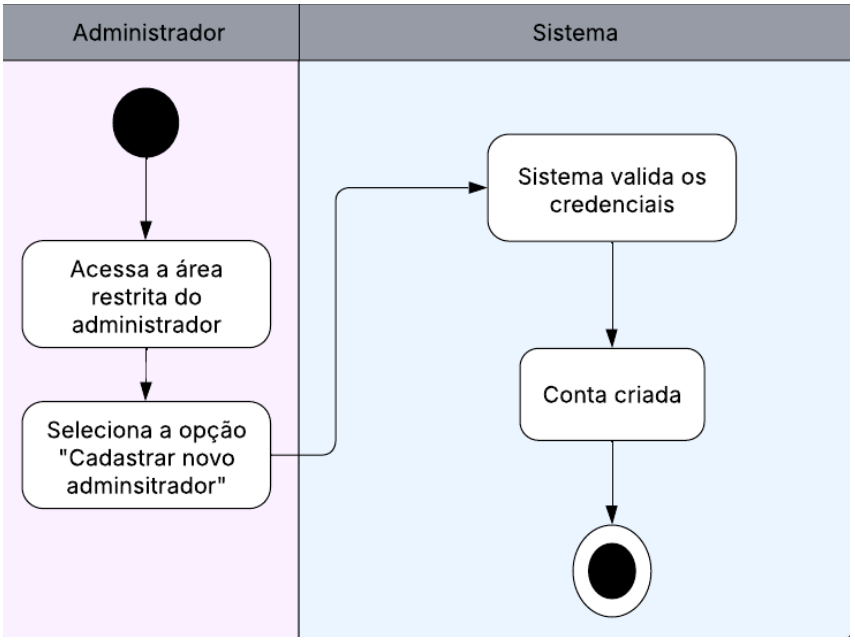
Fonte: elaboração própria.

Figura 5 - Login do profissional (UC04).



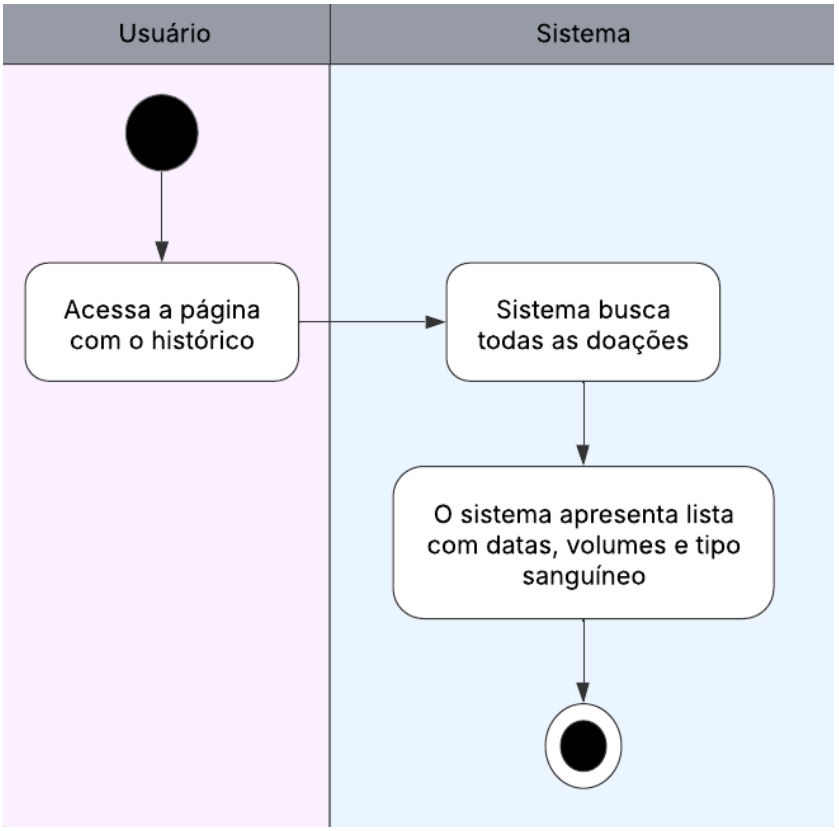
Fonte: elaboração própria.

Figura 6 - Cadastro de novos administradores no sistema (UC05).



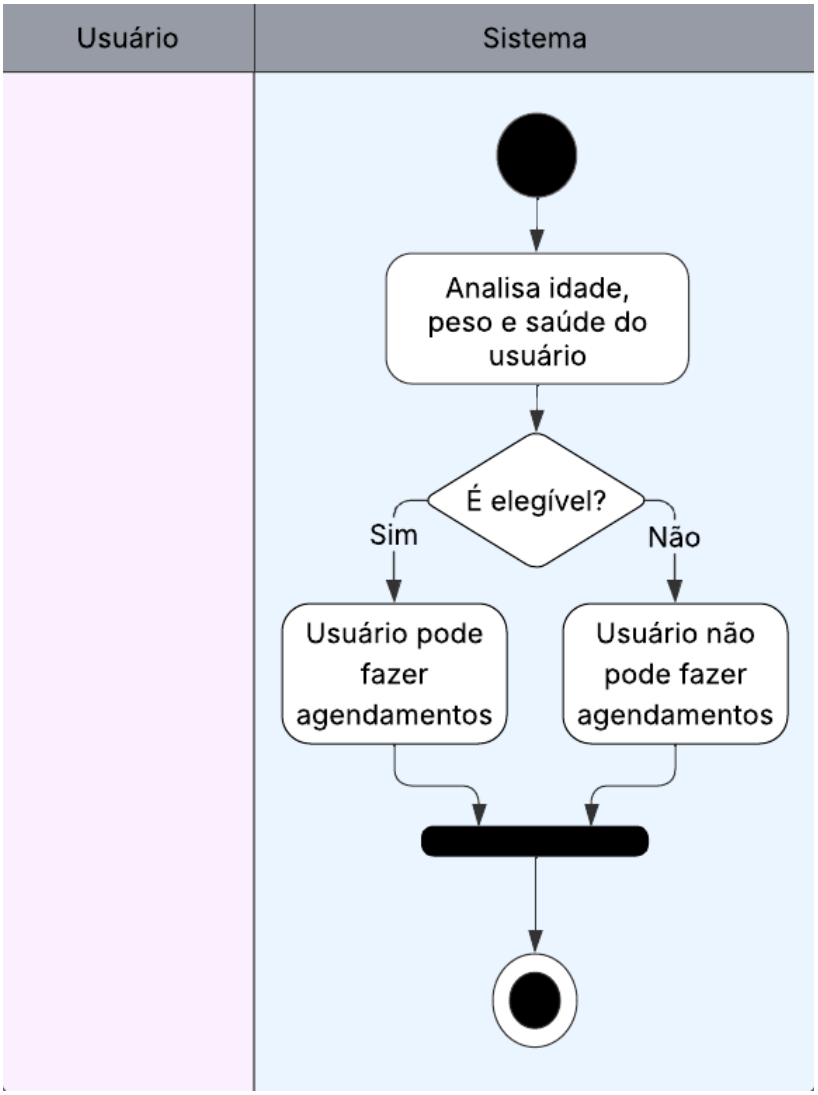
Fonte: elaboração própria.

Figura 7 - Visualização do histórico de doações (UC06).



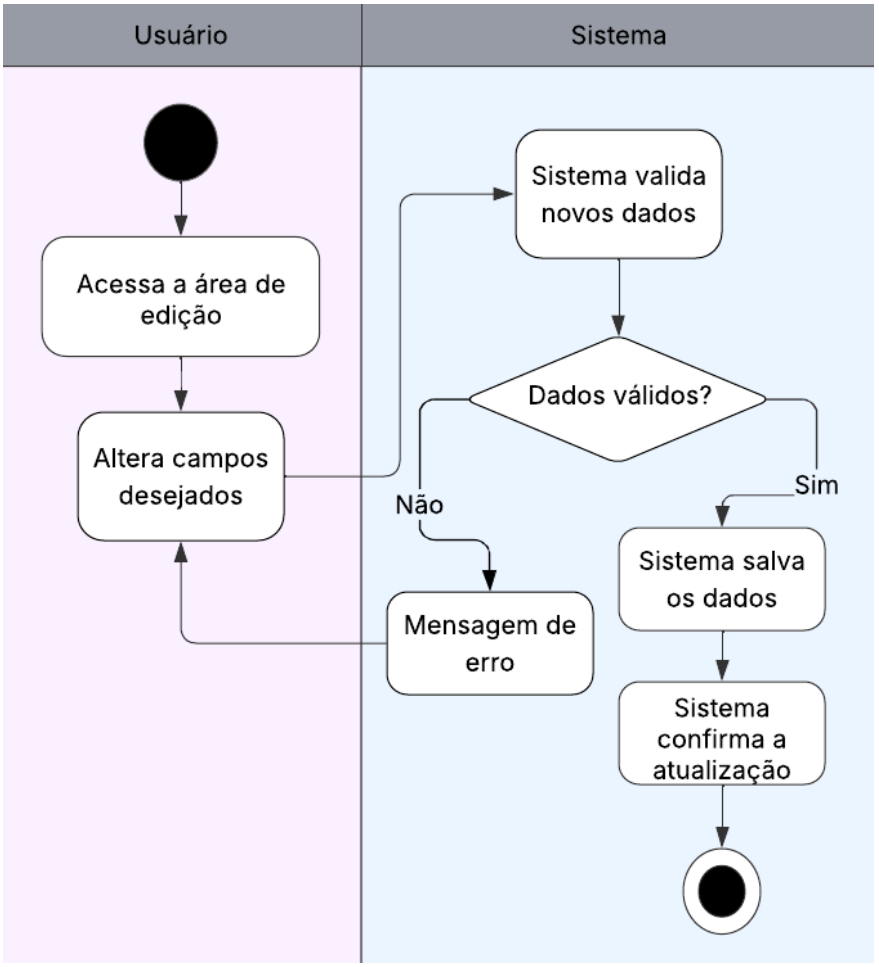
Fonte: elaboração própria.

Figura 8 - Verificar elegibilidade do usuário (UC07).



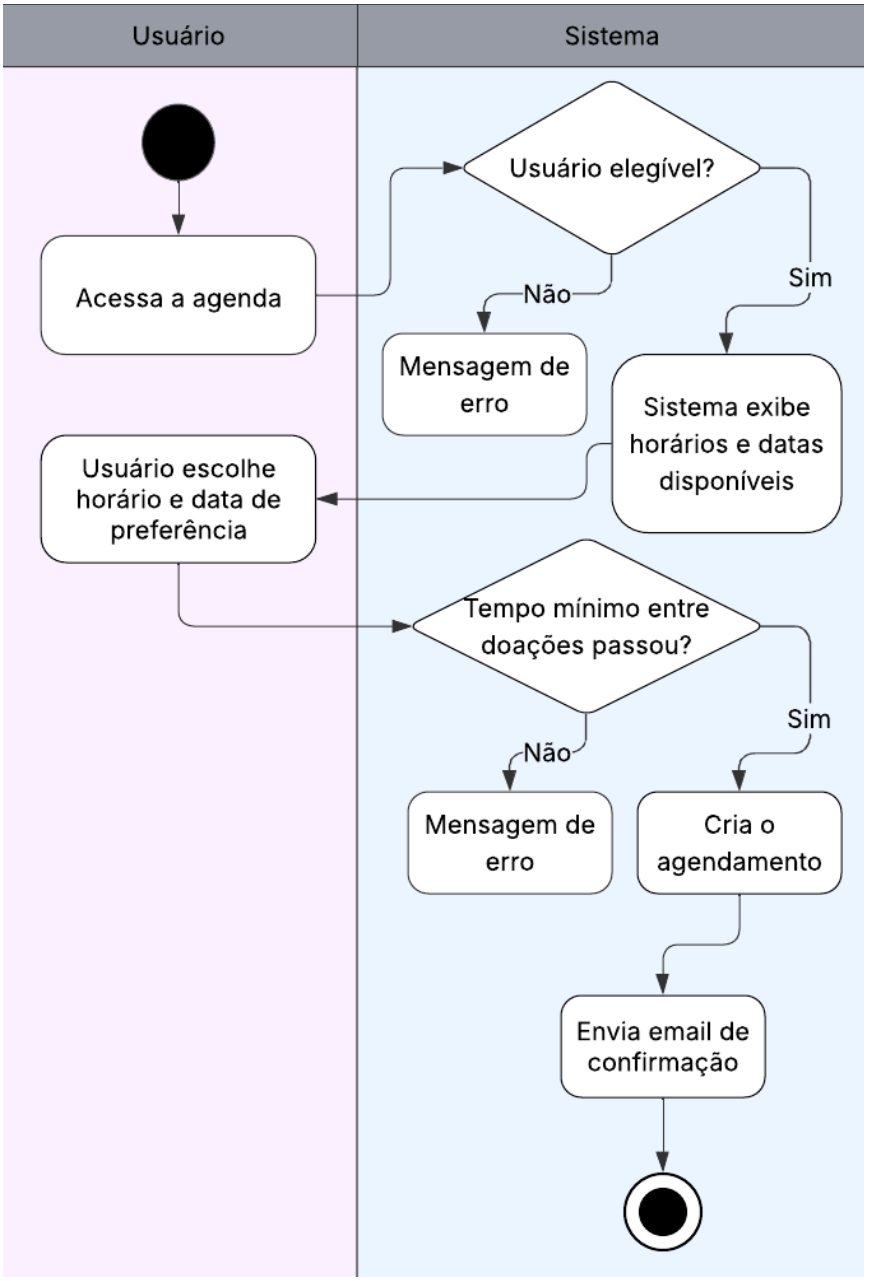
Fonte: elaboração própria.

Figura 9 - Atualizar dados pessoais (UC08).



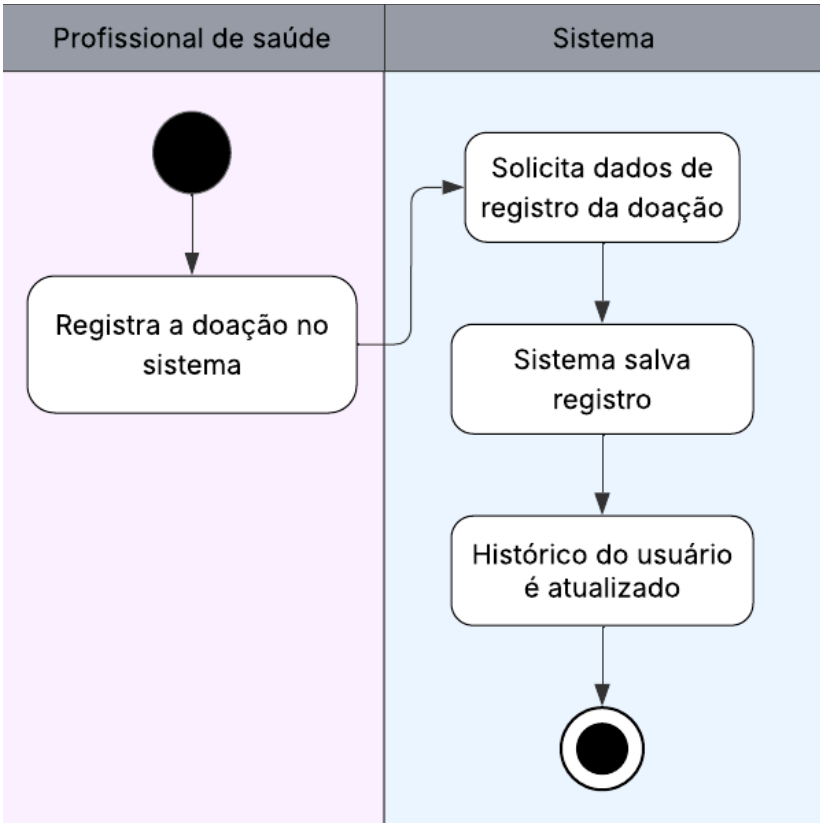
Fonte: elaboração própria.

Figura 10 - Agendar doação (UC09).



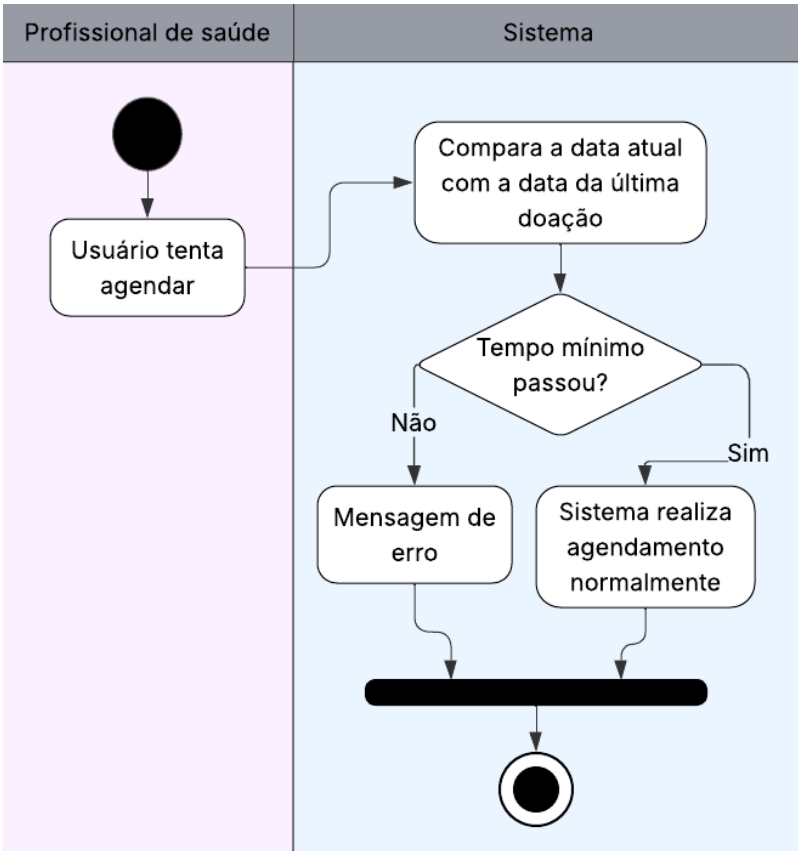
Fonte: elaboração própria.

Figura 11 - Registrar a doação realizada (UC10).



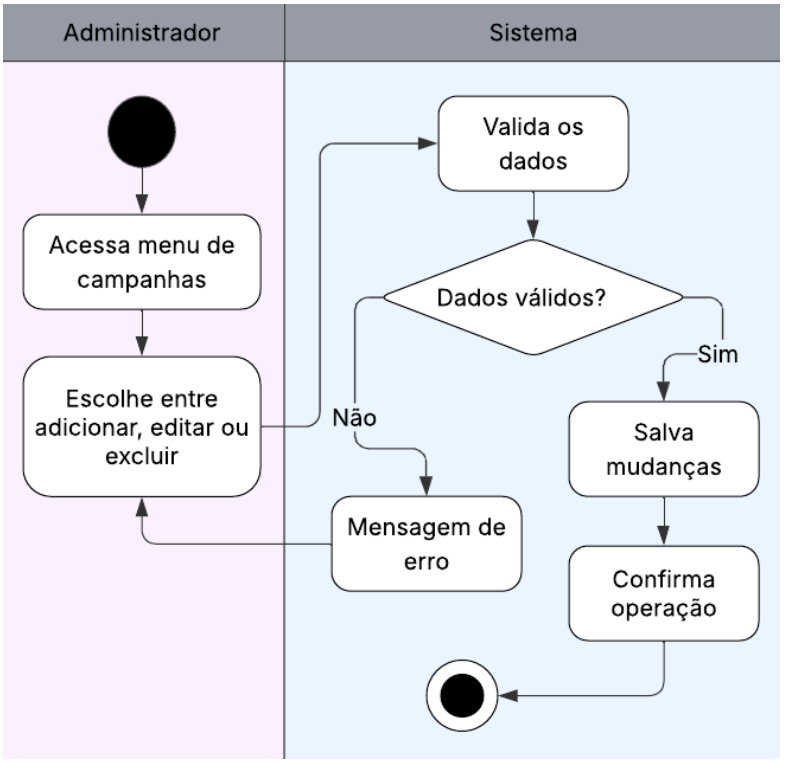
Fonte: elaboração própria.

Figura 12 - Impedir agendamento por intervalo mínimo (UC11).



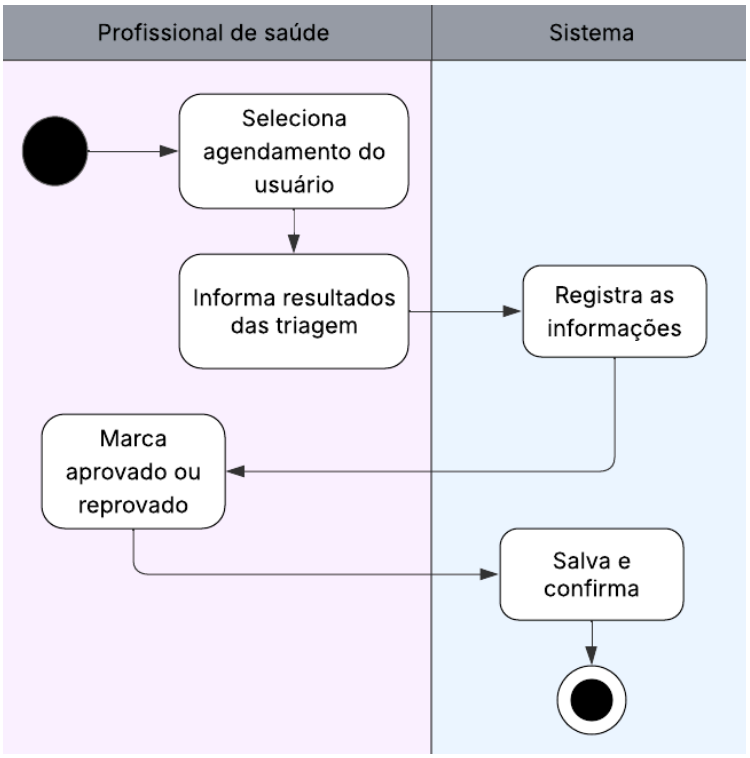
Fonte: elaboração própria.

Figura 13 - Gerenciar campanhas de doação (UC12).



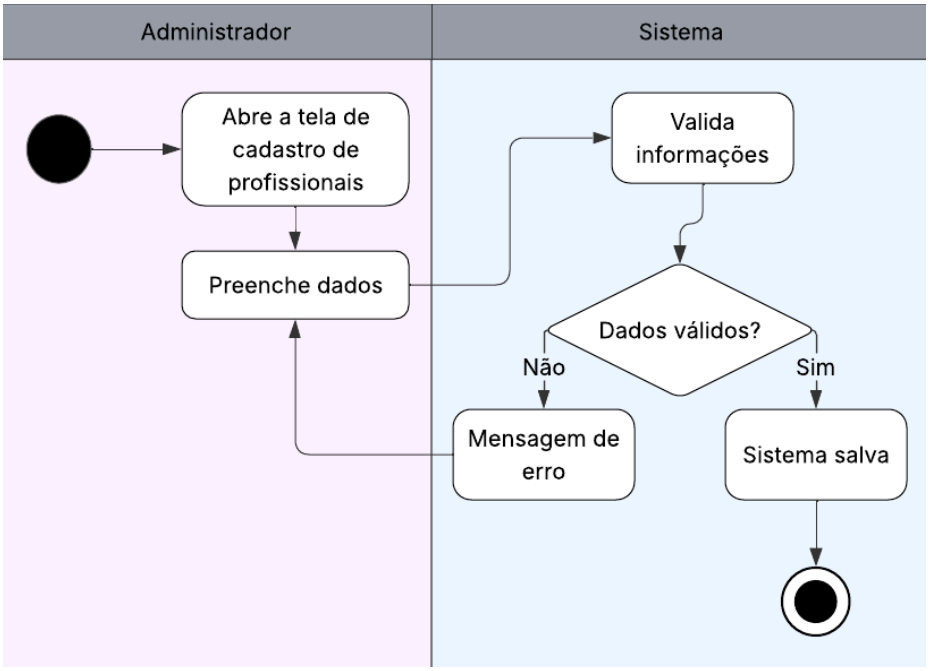
Fonte: elaboração própria.

Figura 14 - Registrar triagem clínica (UC13).



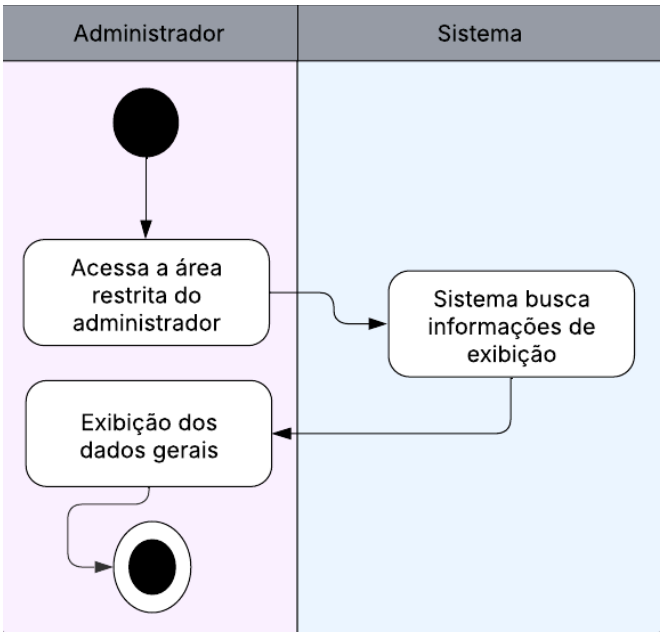
Fonte: elaboração própria.

Figura 15 - Cadastrar profissional de saúde (UC14).



Fonte: elaboração própria.

Figura 16 - Visualizar dados gerais do sistema (UC15).

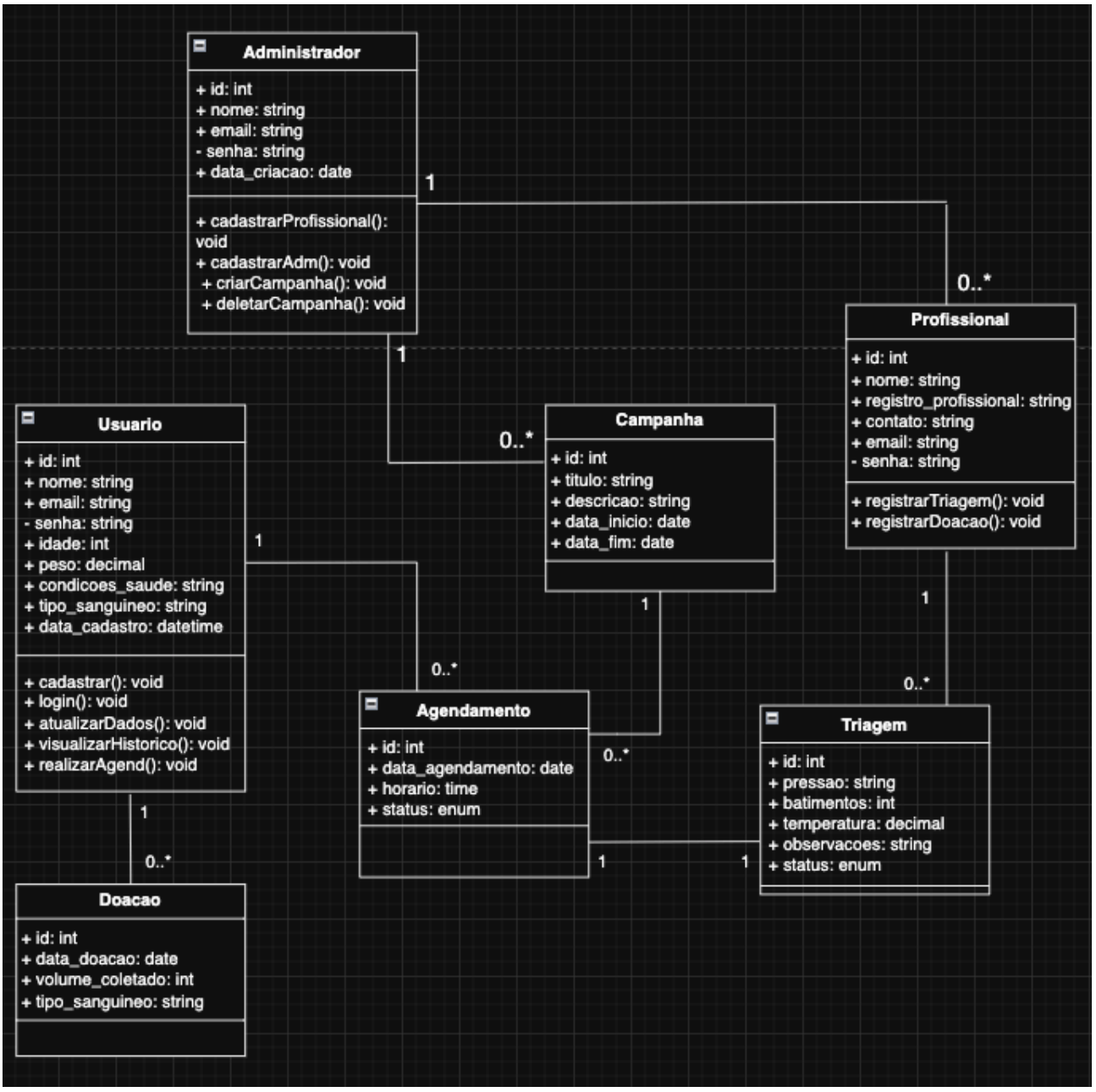


Fonte: elaboração própria.

Diagramas de classes

Identificamos sete classes essenciais no nosso sistema: usuário, campanha, profissional, doação, agendamento, administrador e triagem.

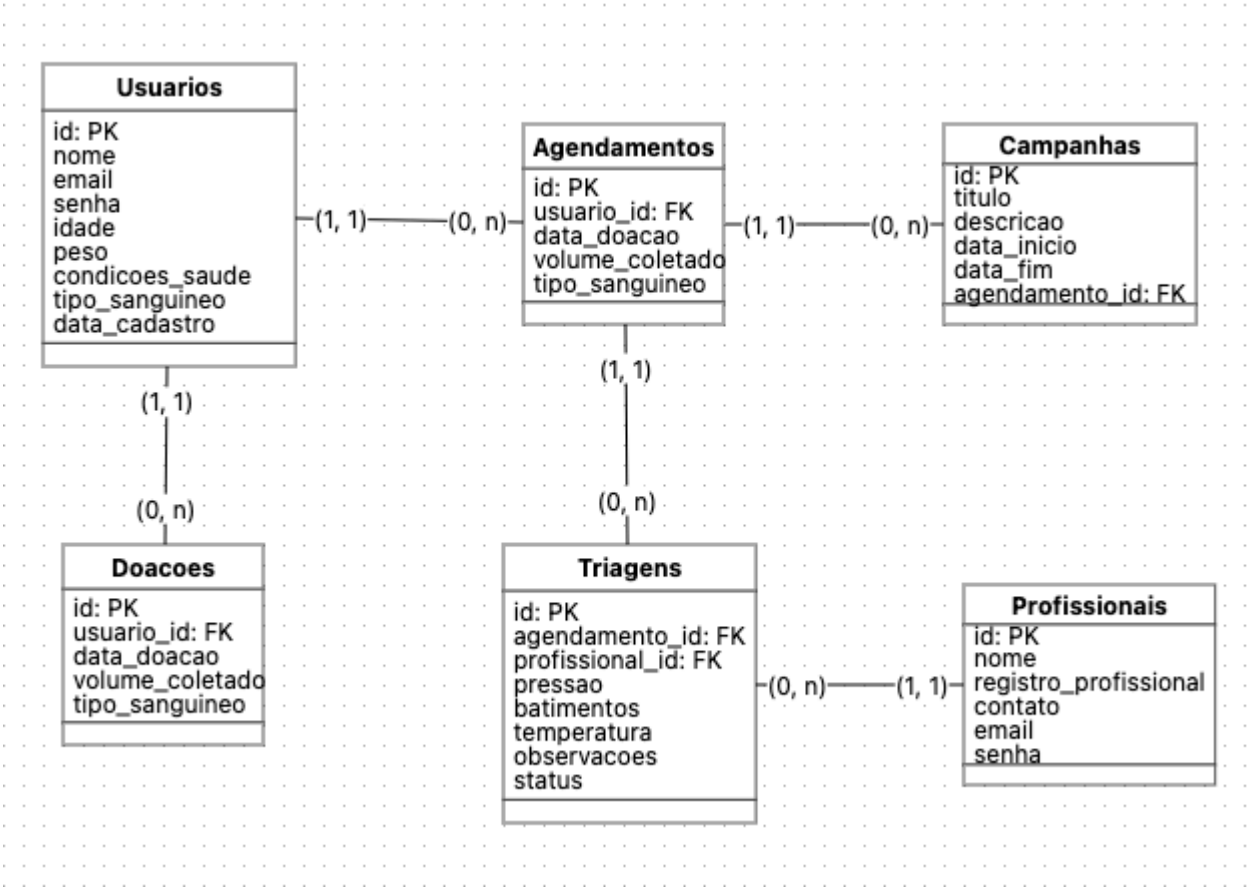
Figura 14 - Diagrama de classes do sistema.



Fonte: elaboração própria.

Projeto Lógico

Figura 15 - Projeto lógico do sistema.



Fonte: elaboração própria.

Projeto Físico

Criação do Banco, Tabelas e Relacionamentos

```
CREATE DATABASE IF NOT EXISTS doacao_db;
USE doacao_db;

--  TABELAS

-- Tabela: Usuarios
CREATE TABLE usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  nome VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  senha VARCHAR(255) NOT NULL,
  idade INT NOT NULL,
  peso DECIMAL(5,2) NOT NULL,
  condicoes_saude TEXT,
  tipo_sanguineo VARCHAR(3) NOT NULL,
  data_cadastro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela: Profissionais
CREATE TABLE profissionais (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  registro_profissional VARCHAR(50) NOT NULL,
  contato VARCHAR(50),
  email VARCHAR(100),
  senha VARCHAR(255)
);

-- Tabela: Campanhas
CREATE TABLE campanhas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  titulo VARCHAR(100) NOT NULL,
  descricao TEXT,
  data_inicio DATE NOT NULL,
  data_fim DATE NOT NULL
);

-- Tabela: Admins
CREATE TABLE admins (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
```

```
  email VARCHAR(120) UNIQUE NOT NULL,
  senha VARCHAR(255) NOT NULL,
  data_criacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela: Agendamentos
CREATE TABLE agendamentos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  usuario_id INT NOT NULL,
  data_agendamento DATE NOT NULL,
  horario TIME NOT NULL,
  status ENUM('PENDENTE','CONFIRMADO','CANCELADO') DEFAULT 'PENDENTE',
  campanha_id INT NULL,
  FOREIGN KEY (usuario_id) REFERENCES usuarios(id),
  FOREIGN KEY (campanha_id) REFERENCES campanhas(id)
);

-- Tabela: Triagens
CREATE TABLE triagens (
  id INT AUTO_INCREMENT PRIMARY KEY,
  agendamento_id INT NOT NULL,
  profissional_id INT NOT NULL,
  pressao VARCHAR(20),
  batimentos INT,
  temperatura DECIMAL(4,1),
  observacoes TEXT,
  status ENUM('APROVADO','REPROVADO') NOT NULL,
  FOREIGN KEY (agendamento_id) REFERENCES agendamentos(id),
  FOREIGN KEY (profissional_id) REFERENCES profissionais(id)
);

-- Tabela: Doações
CREATE TABLE doacoes (
```

```
 id INT AUTO_INCREMENT PRIMARY KEY,
 agendamento_id INT NOT NULL,
 usuario_id INT NOT NULL,
 data_doacao DATE NOT NULL,
 volume_coletado INT NOT NULL,
 tipo_sanguineo VARCHAR(3) NOT NULL,
 campanha_id INT NULL,
 FOREIGN KEY (usuario_id) REFERENCES usuarios(id),
 FOREIGN KEY (agendamento_id) REFERENCES agendamentos(id),
 FOREIGN KEY (campanha_id) REFERENCES campanhas(id)
);
```

Functions

```
DELIMITER //
CREATE FUNCTION pode_doar(p_usuario_id INT)
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
   DECLARE ultima DATE;
   DECLARE peso_usuario DECIMAL(5,2);
   DECLARE idade_usuario INT;

   -- pega a data da última doação
   SELECT MAX(data_doacao)
   INTO ultima
   FROM doacoes
   WHERE usuario_id = p_usuario_id;

   -- pega peso e idade do usuário
   SELECT peso, idade
   INTO peso_usuario, idade_usuario
```

```
    FROM usuarios
    WHERE id = p_usuario_id;

    -- verifica peso mínimo
    IF peso_usuario < 50 THEN
        RETURN FALSE;
    END IF;

    -- verifica idade mínima e máxima
    IF idade_usuario < 18 OR idade_usuario > 65 THEN
        RETURN FALSE;
    END IF;

    -- se nunca doou e atende aos critérios
    IF ultima IS NULL THEN
        RETURN TRUE;
    END IF;

    -- verifica intervalo entre doações (120 dias)
    RETURN DATEDIFF(CURDATE(), ultima) >= 120;
END //
DELIMITER ;
```

Views

```
CREATE OR REPLACE VIEW vw_agendamentos AS
SELECT
    a.id,
    a.usuario_id,
    u.nome AS usuario,
    a.data_agendamento,
    a.horario,
```

```
    a.status,
    c.titulo AS campanha
FROM agendamentos a
LEFT JOIN campanhas c ON c.id = a.campanha_id
JOIN usuarios u ON u.id = a.usuario_id;

CREATE OR REPLACE VIEW vw_doacoes AS
SELECT
    d.id,
    d.usuario_id,
    u.nome AS usuario,
    d.data_doacao,
    d.volume_coletado,
    d.tipo_sanguineo,
    c.titulo AS campanha
FROM doacoes d
LEFT JOIN campanhas c ON c.id = d.campanha_id
JOIN usuarios u ON u.id = d.usuario_id;

CREATE OR REPLACE VIEW vw_triagens AS
SELECT
    t.id,
    t.agendamento_id,
    u.nome AS usuario,
    p.nome AS profissional,
    t.pressao,
    t.batimentos,
    t.temperatura,
    t.observacoes,
    t.status,
    a.data_agendamento
FROM triagens t
JOIN agendamentos a ON a.id = t.agendamento_id
```

```
JOIN usuarios u ON u.id = a.usuario_id
JOIN profissionais p ON p.id = t.profissional_id;
```

Triggers

```
DELIMITER //
-- Impede agendamento duplicado por usuário no mesmo dia
CREATE TRIGGER trg_agendamento_unico
BEFORE INSERT ON agendamentos
FOR EACH ROW
BEGIN
   IF EXISTS (
      SELECT 1 FROM agendamentos
      WHERE usuario_id = NEW.usuario_id
        AND data_agendamento = NEW.data_agendamento
        AND status != 'CANCELADO'
   ) THEN
      SIGNAL SQLSTATE '45000'
         SET MESSAGE_TEXT = 'Usuário já possui agendamento neste dia.';
   END IF;
END //
DELIMITER ;

DELIMITER //
-- Cria doação automaticamente após triagem aprovada
CREATE TRIGGER trg_criar_doacao
AFTER INSERT ON triagens
FOR EACH ROW
BEGIN
   IF NEW.status = 'APROVADO' THEN
      INSERT INTO doacoes
                  (agendamento_id, usuario_id, data_doacao, volume_coletado, tipo_sanguineo,
```

```
campanha_id)
        SELECT
            a.id,
            a.usuario_id,
            CURDATE(),
            450,
            u.tipo_sanguineo,
            a.campanha_id
        FROM agendamentos a
        JOIN usuarios u ON u.id = a.usuario_id
        WHERE a.id = NEW.agendamento_id;
    END IF;
END //
DELIMITER ;
```

Procedures

```
-- CRUD USUARIOS
DELIMITER //
CREATE PROCEDURE criar_usuario(
  IN p_nome VARCHAR(100),
  IN p_email VARCHAR(100),
  IN p_senha VARCHAR(255),
  IN p_idade INT,
  IN p_peso DECIMAL(5,2),
  IN p_condicoes TEXT,
  IN p_tipo VARCHAR(3)
)
BEGIN
  INSERT INTO usuarios (nome, email, senha, idade, peso, condicoes_saude, tipo_sanguineo)
  VALUES (p_nome, p_email, p_senha, p_idade, p_peso, p_condicoes, p_tipo);
END //
DELIMITER ;
```

```
DELIMITER //
CREATE PROCEDURE atualizar_usuario(
  IN p_id INT,
  IN p_nome VARCHAR(100),
  IN p_email VARCHAR(100),
  IN p_senha VARCHAR(255),
  IN p_idade INT,
  IN p_peso DECIMAL(5,2),
  IN p_condicoes TEXT,
  IN p_tipo VARCHAR(3)
)
BEGIN
  UPDATE usuarios
  SET nome = p_nome,
      email = p_email,
      senha = p_senha,
      idade = p_idade,
      peso = p_peso,
      condicoes_saude = p_condicoes,
      tipo_sanguineo = p_tipo
  WHERE id = p_id;
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE remover_usuario(IN p_id INT)
BEGIN
  DELETE FROM usuarios WHERE id = p_id;
END //
DELIMITER ;

-- CRUD PROFISSIONAIS
```

```
DELIMITER //
CREATE PROCEDURE criar_profissional(
  IN p_nome VARCHAR(100),
  IN p_registro VARCHAR(50),
  IN p_contato VARCHAR(50),
  IN p_email VARCHAR(100),
  IN p_senha VARCHAR(255)
)
BEGIN
  INSERT INTO profissionais (nome, registro_profissional, contato, email, senha)
  VALUES (p_nome, p_registro, p_contato, p_email, p_senha);
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE remover_profissional(IN p_id INT)
BEGIN
  DELETE FROM profissionais WHERE id = p_id;
END //
DELIMITER ;

-- CRUD CAMPANHAS
DELIMITER //
CREATE PROCEDURE criar_campanha(
  IN p_titulo VARCHAR(100),
  IN p_descricao TEXT,
  IN p_inicio DATE,
  IN p_fim DATE
)
BEGIN
  INSERT INTO campanhas (titulo, descricao, data_inicio, data_fim)
  VALUES (p_titulo, p_descricao, p_inicio, p_fim);
END //
```

```
DELIMITER ;

DELIMITER //
CREATE PROCEDURE remover_campanha(IN p_id INT)
BEGIN
  DELETE FROM campanhas WHERE id = p_id;
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE listar_campanhas()
BEGIN
  SELECT * FROM campanhas ORDER BY data_inicio DESC;
END //
DELIMITER ;

-- CRUD ADMINS
DELIMITER //
CREATE PROCEDURE criar_admin(
  IN p_nome VARCHAR(100),
  IN p_email VARCHAR(120),
  IN p_senha VARCHAR(255)
)
BEGIN
  INSERT INTO admins (nome, email, senha)
  VALUES (p_nome, p_email, p_senha);
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE remover_admin(IN p_id INT)
BEGIN
  DELETE FROM admins WHERE id = p_id;
```

```
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE listar_admins()
BEGIN
  SELECT id, nome, email, data_criacao FROM admins ORDER BY nome;
END //
DELIMITER ;

--  PROCEDURES - AGENDAMENTO (WORKFLOW COMPLETO)

-- Criar agendamento
DELIMITER //
CREATE PROCEDURE registrar_agendamento(
    IN p_usuario_id INT,
    IN p_data DATE,
    IN p_horario TIME,
    IN p_campanha_id INT
)
BEGIN
    INSERT INTO agendamentos (usuario_id, data_agendamento, horario, campanha_id)
    VALUES (p_usuario_id, p_data, p_horario, p_campanha_id);
END //
DELIMITER ;

-- Confirmar agendamento
DELIMITER //
CREATE PROCEDURE confirmar_agendamento(
    IN p_agendamento_id INT
)
BEGIN
    DECLARE v_status VARCHAR(20);
```

```
    SELECT status INTO v_status
    FROM agendamentos
    WHERE id = p_agendamento_id;

    IF v_status IS NULL THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Agendamento não encontrado.';
    END IF;

    IF v_status = 'CANCELADO' THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Agendamento cancelado não pode ser confirmado.';
    END IF;

    IF v_status = 'CONFIRMADO' THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Agendamento já está confirmado.';
    END IF;

    UPDATE agendamentos
    SET status = 'CONFIRMADO'
    WHERE id = p_agendamento_id;
END //
DELIMITER ;

-- Cancelar agendamento
DELIMITER //
CREATE PROCEDURE cancelar_agendamento(
    IN p_agendamento_id INT
)
BEGIN
    DECLARE v_status VARCHAR(20);
```

```
    SELECT status INTO v_status
    FROM agendamentos
    WHERE id = p_agendamento_id;

    IF v_status IS NULL THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Agendamento não encontrado.';
    END IF;

    IF v_status = 'CANCELADO' THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Agendamento já está cancelado.';
    END IF;

    UPDATE agendamentos
    SET status = 'CANCELADO'
    WHERE id = p_agendamento_id;
END //
DELIMITER ;

--  TRIAGEM E DOAÇÃO

DELIMITER //
CREATE PROCEDURE registrar_triagem(
    IN p_agendamento INT,
    IN p_profissional INT,
    IN p_pressao VARCHAR(20),
    IN p_batimentos INT,
    IN p_temp DECIMAL(4,1),
    IN p_obs TEXT,
    IN p_status ENUM('APROVADO','REPROVADO')
)
```

```
BEGIN
    INSERT INTO triagens (agendamento_id, profissional_id, pressao, batimentos, temperatura, observacoes, status)
  VALUES (p_agendamento, p_profissional, p_pressao, p_batimentos, p_temp, p_obs, p_status);
END //
DELIMITER ;
```

Proposta comercial

Nosso objetivo é desenvolver um sistema de controle e gerenciamento para uma clínica de doação de sangue, possibilitando a sua modernização, a melhoria da experiência de doadores e profissionais, com a entrega de um ambiente digital eficiente para agendamentos, triagens e gestão de doações. Para cumprir o nosso objetivo, focamos em usabilidade, desempenho, segurança (pelo uso do SGBD MySQL) e ampliação do alcance dos serviços prestados pela clínica. A solução proposta consiste em um sistema composto por:

- Módulo de cadastro e autenticação de usuários (doadores, administradores e profissionais de saúde);
- Agendamento online de doações, com controle automático de elegibilidade e intervalo mínimo;
- Gestão de campanhas de doação, incluindo criação, edição e exclusão;
- Triagem clínica digital, permitindo registro e aprovação/recusa de doações;
- Histórico de doações do usuário, atualizado automaticamente;

Com essas funcionalidades, o sistema será capaz de facilitar processos internos que antes demandavam trabalho manual e repetitivo. Após a implementação, garantimos que haverá:

- Maior eficiência nos processos da clínica;
- Redução de filas e tempo de espera;
- Aumento no número de doadores ativos;
- Melhor experiência do usuário;
- Centralização de informações;
- Facilitação de campanhas e comunicação.

Considerações Finais

O desenvolvimento do sistema para a clínica de doação de sangue permitiu explorar conceitos essenciais de análise, modelagem e implementação de banco de dados, bem como a

integração entre back-end, front-end e persistência de dados. A construção da solução evidenciou a importância de um levantamento de requisitos bem estruturado, da modelagem UML e da utilização de boas práticas de engenharia de software.

O sistema proposto atende às principais necessidades identificadas, como o agendamento online de doações, o controle de elegibilidade dos usuários e o registro de triagens clínicas por profissionais. Ao priorizar usabilidade, segurança e desempenho, o projeto contribui para a modernização dos processos da clínica, facilitando o trabalho dos profissionais e oferecendo uma experiência mais eficiente aos doadores.

Além disso, a implementação do banco de dados em MySQL demonstrou-se adequada ao contexto, garantindo organização, integridade e confiabilidade das informações. O uso de tecnologias amplamente difundidas no mercado reforçou a viabilidade do projeto e possibilitou uma estrutura flexível para futuras expansões.

Por fim, o trabalho evidencia a relevância de soluções tecnológicas no incentivo à doação de sangue, ampliando o alcance dos serviços e promovendo maior engajamento. O sistema desenvolvido representa um passo importante em direção à digitalização desses processos, ao mesmo tempo em que abre espaço para evoluções e aprimoramentos futuros.

Referências Bibliográficas

LUCIANO, Francisco Veríssimo. SPOBDD2 – Banco de Dados 2 - Notas de aulas disponíveis no Ambiente Virtual de Aprendizagem Moodle do IFSP – Campus São Paulo. São Paulo: 2024.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de Software: Uma Abordagem Profissional. 9. ed. Porto Alegre: AMGH, 2021.