RELATÓRIO - SISTEMA DE ADOÇÃO DE PETS

Alunos: Caio Torres, Gabriela Benevides, Antonio Augusto Diogo Araujo, Rayssa Machado e Eduardo Alves.

RESUMO
O objetivo deste relatório é fornecer uma visão detalhada do processo de desenvolvimento do sistema de adoção de pets, abordando tópicos relacionados à arquitetura do sistema, as tecnologias empregadas, aos desafios enfrentados durante o desenvolvimento, possíveis bugs identificados e o esquema físico do banco de dados.

1. INTRODUÇÃO

O sistema "Abrigo Cisco" é uma plataforma completa para adoção de pets, projetada para facilitar o processo de encontro entre animais que precisam de um lar e pessoas interessadas em adotá-los. Ele oferece funcionalidades tanto para usuários comuns em busca de um pet quanto para funcionários do abrigo que gerenciam os animais e os processos de adoção.

2. TECNOLOGIAS UTILIZADAS

Frontend:

- Linguagem: TypeScript
- Biblioteca Principal: React
- Estilização: Tailwind CSS e componentes da biblioteca Radix UI.
- Roteamento: React Router (react-router-dom).
- Gerenciamento de Estado/Cache de Dados: React Query.
- Formulários: React Hook Form (react-hook-form) com validação via Zod.
- Requisições HTTP: Axios
- Build Tool: Vite
- Outras bibliotecas: lucide-react (ícones), date-fns (manipulação de datas), sonner (notificações/toasts), recharts (gráficos), embla-carousel-react (carrossel).

Backend:

- Linguagem: JavaScript (Node.js)
- Framework: Express.js
- Banco de Dados: MongoDB com Mongoose como ODM (Object Data Modeling).
- Autenticação: JWT (JSON Web Tokens) com jsonwebtoken e bcryptjs para hashing de senhas.

- Upload de Arquivos: Multer
- Outras bibliotecas notáveis: cors (Cross-Origin Resource Sharing), dotenv (gerenciamento de variáveis de ambiente).

3. SOBRE O PROCESSO DE DESENVOLVIMENTO

3.1. DESAFIOS

- Integração Frontend-Backend: Garantir a comunicação eficaz e a correta tipagem de dados entre o frontend TypeScript e o backend JavaScript.
- Gerenciamento de Estado: Com funcionalidades como perfis de usuário, listagem de pets, solicitações de adoção e painel administrativo, o gerenciamento do estado da aplicação no frontend.
- Segurança: Implementar o JWT.
- Upload de Imagens: Lidar com o armazenamento de imagens dos pets.
- Validação de Dados: Garantir a consistência e validade dos dados em ambas as camadas.

3.2. MUDANÇAS

- Cadastro e Login com o Google: Como era uma feature mais complexa, requerendo uma conexão via API com o Google, optamos por não fazer essa implementação.
- Fazer doação para o abrigo: Essa implementação também tem um nível de complexidade mais alto, sendo necessário conectar-se a uma API para fazer o pagamento. Por isso, decidimos não implementar essa feature.

3.3. BUGS

- O formulário de Análise de Perfil deveria salvar os dados em uma tabela do banco de dados, mas foi erroneamente configurado para armazenar as informações em cookies do navegador porque não foi especificado que era uma tabela no banco do usuário.

4. ESTRUTURA DO BANCO DE DADOS

O banco de dados utilizado foi o MongoDB. O MongoDB é um banco de dados NoSQL orientado a documentos, que armazena dados em formato flexível (BSON/JSON). Sua arquitetura permite consultas ágeis, indexação eficiente e suporte a operações em tempo real, alinhando-se bem às necessidades de aplicações modernas.

- adoptions: Tabela para guardar as informações sobre adoções. Contém os atributos id (ObjectId), pet (ObjectId), user (ObjectId), Status (String), createdAt (Date), updatedAt (Date), AdoptionDate (Date).

- pets: Tabela para guardar as informações sobre os pets. Contém os atributos id (ObjectId), name (String), species (String), breed (String), age (Int32), gender (String), size (String), description (String), photos (Array), status (String), createdAt (Date), updatedAt (Date).

- profileanalysis: Tabela para guardar as informações dos formulários de análise de perfil. Contém os atributos id (ObjectId), userId (ObjectId), monthlyIncome (String), housingType (String), roomCount (Int32), hasPets (Boolean), petsDescription (String), hashTldren (Boolean), chTldrenCount (Int32), hoursAvailable (String), isComplete (Boolean), status (String), createdAt (Date), updatedAt (Date).

- users: Tabela para guardar as informações dos usuários cadastrados. Contém os atributos id (ObjectId), name (String), email (String), password (String), isEmployee (Boolean), createdAt (Date), updatedAt (Date).

5. TESTES

5.1. Teste de Unidade

Descrição**:** Testa métodos e funções isoladas no sistema.

Aplicações no projeto: Podemos aplicar teste de unidade ao método `requestAdoption` porque ele possui regras de negócio isoladas, como verificar se o pet está disponível e se o perfil do usuário está completo. Essas validações podem ser testadas individualmente, sem depender de outras partes do sistema.

5.2. Teste de Integração

Descrição: Verifica se os módulos do sistema funcionam corretamente juntos.

Aplicações no projeto: Formulário de cadastro de usuário + API de criação de conta + Banco de dados: Testa se os dados preenchidos no frontend são corretamente armazenados.

5.3. Teste de Sistema

Descrição: Avalia o sistema como um todo, simulando um fluxo real do usuário.

Aplicações no projeto: 3

- Simulação completa: Cadastro → Login → Escolher pet → Adotar pet → Verificar status atualizado.

- Cenário de erro: Tentativa de adotar um pet já adotado → Sistema deve impedir a ação e exibir mensagem adequada.

5.4. Teste de Aceitação

Descrição: Valida se o sistema cumpre os requisitos esperados pelo cliente.

Aplicações no projeto: Validação do fluxo de adoção: O sistema exige o preenchimento completo do formulário com dados obrigatórios antes de permitir a adoção.

5.5 Testes Manuais

Antonio - Funcionalidade testada: Solicitação de adoção.

1. Fazer login com um usuário que tenha perfil completo.
2. Acessar a lista de pets disponíveis.
3. Selecionar um pet e clicar em “Solicitar Adoção”.
4. Preencher as observações (opcional) e confirmar.

Caio - Funcionalidade testada: Cadastro de um novo usuário comum no sistema.

1. Acessar a página inicial do sistema.
2. Clicar no botão/link "Cadastrar" ou "Criar Conta".
3. Preencher o formulário de cadastro com os seguintes dados:
4. Nome: "Usuário Teste Cadastro"
5. Email: Um email válido e único (ex: usuariotestecadastro@example.com)
6. Senha: "senha123" (ou uma senha que atenda aos critérios de segurança do sistema, como mínimo de 6 caracteres)
7. Confirmar Senha: "senha123"
8. Clicar no botão "Cadastrar".

Fluxos alternativos:

1. Caso o usuário não preencha um dos campos, uma mensagem de erro será apresentada.
2. Caso o usuário coloque uma senha com menos que 6 caracteres, uma mensagem de erro será mostrada.
3. Se o e-mail inserido já tiver sido cadastrado, uma mensagem de erro será mostrada.

Rayssa - Funcionalidade testada: Busca e Filtros

1. Acessar a página inicial do sistema.
2. Clicar no botão “Vamos Começar!”
3. Buscar por nome ou raça ou usar os filtros de busca.
4. Caso haja correspondência, os pets serão devidamente apresentados.

Fluxo alternativo:

1. Se não houver, será exibida a mensagem “Nenhum pet encontrado com os filtros selecionados.”.

Eduardo - Funcionalidade testada: Preencher formulário de análise de perfil

1. Acessar a página inicial do sistema.
2. Fazer login com uma conta de usuário.
3. Clicar no seu perfil para abrir o menu dropdown.
4. Acessar a aba “Meu perfil”
5. Preencher o formulário.
6. Clicar no botão de salvar.

Fluxos alternativos:

1. Caso o usuário não preencha um dos campos obrigatórios, uma mensagem de erro será apresentada.

Gabriela - Funcionalidade testada: Criação de pet.

1. Acessar a página inicial do sistema.
2. Fazer login com uma conta de administrador (obs: para simplificar a lógica de nosso sistema, a regra de negócio para existir um admin é possuir @abrigocisco.com em seu email).
3. Acessar a área “Área do funcionário” destinada somente para funcionários.
4. Acessar a aba “Cadastrar novo pet”
5. Preencher as informações sobre o pet.
6. Clicar no botão de salvar pet.
7. Verificar na aba “Ver pets” se o pet foi devidamente cadastrado.

Fluxos alternativos:

1. Caso o usuário não preencha um dos campos obrigatórios, uma mensagem de erro será apresentada.

5.6 Testes (Desafio)

Foi feita a tentativa de desenvolvimento de um arquivo para testes automatizados utilizando as bibliotecas Jest e Supertest, com foco na verificação do fluxo de adoção de pets na API do projeto. A ideia foi simular e validar etapas como cadastro de usuário, autenticação com token JWT, cadastro de pet, verificação de autorização e solicitação de adoção, compondo um fluxo de teste de unidade, integração, sistema e aceitação completos.

Durante a execução do último teste de adoção, foi identificado um erro de autenticação (status 401), provavelmente relacionado à verificação do token JWT. O erro não foi resolvido a tempo da entrega, mas o arquivo **adoptionService.test.js** serviu como uma base funcional para evolução futura dos testes automatizados.

OBS: Para rodar o arquivo de testes, execute dentro da pasta do backend o seguinte comando:
npm test

O teste é estruturado dentro de uma suíte describe, com diversas etapas encadeadas em blocos it, seguindo a seguinte sequência:

1. **Cadastro de usuário comum**: Um novo usuário é registrado por meio da rota POST /api/auth/register, e sua resposta é validada quanto ao status HTTP e à presença dos dados do usuário.

2. **Autenticação do usuário**: O usuário recém-cadastrado realiza login via POST /api/auth/login, recebendo um **token JWT**, necessário para autenticações futuras.

3. **Tentativa de cadastro de pet sem autenticação**: É feita uma tentativa de cadastrar um pet sem fornecer o token, e espera-se que o servidor retorne um erro 401 (não autorizado), validando o controle de acesso.

4. **Cadastro e autenticação de administrador**: Um usuário administrador é cadastrado e autenticado, e seu token JWT é armazenado para operações privilegiadas.

5. **Cadastro de pet com autorização**: Utilizando o token de administrador, um pet é cadastrado com sucesso via POST /api/pets, e o ID retornado é armazenado para uso posterior.

6. **Solicitação de adoção**: Um usuário autenticado realiza uma solicitação de adoção para o pet cadastrado, por meio da rota POST /api/adoptions. A resposta é validada para garantir que a adoção foi registrada corretamente, com o status pendente e os dados do usuário e do pet associados.