Validação da solução

A validação foi realizada por meio de:

- Testes manuais das funcionalidades principais no navegador e via ferramentas de API (como Postman);
- Simulação de fluxos completos de uso (cadastro, adoção, interesse);
- Criação de cenários no formato Gherkin, que permite descrever testes de forma clara e compreensível, orientada ao comportamento do usuário.

Estratégia de Validação Aplicada

Ambiente testado: Localhost com backend em Node.js, frontend React e banco PostgreSQL via Docker;

Ferramentas utilizadas: Navegador (Chrome), Postman (para testes de API), VS Code (para logs e correções);

Critérios de sucesso:

- A aplicação deve funcionar sem erros;
- O usuário deve conseguir realizar a jornada completa de adoção;
- Os dados devem ser salvos corretamente no banco;
- As mensagens de feedback devem ser claras.

Cenários de testes:

Testes manuais de interface

Funcionalidade: Cadastro de novo animal

Cenário: Protetor cadastra um novo animal

Dado que estou logado como protetor

E estou na página de cadastro de animal

Quando preencho o nome, idade, porte, personalidade e história

E envio uma imagem do animal

E clico no botão "Cadastrar"

Então o sistema exibe a mensagem "Animal cadastrado com sucesso"

E o animal aparece na listagem

Funcionalidade: Filtro de animais por personalidade

Cenário: Adotante busca animais brincalhões

Dado que estou na página de adoção

E seleciono o filtro "Brincalhão"

Quando clico em "Buscar"

Então a lista de animais exibida deve conter apenas animais com personalidade brincalhão

Funcionalidade: Registro de interesse na adoção

Cenário: Adotante demonstra interesse por um animal

Dado que estou visualizando o perfil de um animal

Quando clico em "Tenho Interesse"

E preencho o formulário com meu nome, telefone e informações do lar

E clico em "Enviar"

Então o sistema confirma o envio com a mensagem "Obrigado! O protetor entrará em contato em breve."

Testes Manuais de API

1. Autenticação

Endpoint: `POST /api/login`

Objetivo: Autenticar protetores e retornar token de acesso.

Cenários:

- **Sucesso (200 OK):**

```
// Request
{ "email": "protetor@email.com", "senha": "senha123" }

// Response
{ "token": "protetor-auth", "email": "protetor@email.com" }
```

- **Credenciais Inválidas (401 Unauthorized):**

```
// Request
{ "email": "invalido@email.com", "senha": "errada" }

// Response
{ "mensagem": "Credenciais inválidas" }
```

- **Erro Interno (500 Internal Server Error):** Erro interno do servidor

2. Gerenciamento de Animais

Objetivo: Listar, filtrar, editar, cadastrar e deletar os animais

Endpoints:

- **GET /api/animais**
  - **Sucesso (200 OK):** Retornar array de animais.
  - **Erro (500):** Simular falha no banco de dados.
- **GET /api/animais/:id**
  - **Sucesso (200 OK):**

```
// Exemplo para /api/animais/3
{ "id": 3, "nome": "Thor", "idade": 4, ... }
```

  - **Erro (404 Not Found):** ID inexistente (ex: `/api/animais/9999`).

- **POST /api/animais**
  - **Sucesso (201 Created):**

```
// Request (multipart/form-data)
{
  "nome": "Rex", "idade": 3, "porte": "Médio",
  "especie": "Cachorro", "foto": [arquivo.jpg]
}

// Response
{
  "mensagem": "Animal cadastrado com sucesso!",
  "animal": { "id": 99, ... }
}
```

  - **Erro (400 Bad Request):** Campos obrigatórios ausentes (ex: `nome`, `especie`).
  - **Erro (500):** Falha no upload da imagem ou banco de dados.
- **PUT /api/animais/:id**
  - **Sucesso (200 OK):** Atualizar parcialmente (ex: alterar `idade`).
  - **Erro (404):** ID inexistente.
- **DELETE /api/animais/:id**
  - **Sucesso (200 OK):**

```
{ "mensagem": "Animal deletado com sucesso!" }
```

Erro (500): Erro ao deletar animal

3. Registro de Interesse

Endpoint: `POST /api/interesse`

Objetivo: Registrar interesse em adoção e notificar o protetor por e-mail.

Cenários:

- **Sucesso (201 Created):**

```
// Request
{
"nome": "João Silva",
"telefone": "51999999999",
"tipoLar": "Casa com pátio",
"outrosAnimais": true,
"animalId": 5
}

// Response
{ "mensagem": "Interesse enviado com sucesso!" }
```

  - Validação: E-mail deve ser enviado ao protetor.
- **Erro (400 Bad Request):** Campos obrigatórios ausentes (ex: `animalId`).
- **Erro (404):** `animalId` inexistente.
- **Erro (500):** Falha no envio do e-mail ou banco de dados.

Endpoint: `GET /api/interesse`

- **Sucesso (200 OK):** Retornar lista de interesses registrados.

---

4. Localização (Estados e Municípios)

Endpoints:

- **GET /api/estados**
  - **Sucesso (200 OK):**

```
[
  { "id": 23, "sigla": "CE", "nome": "Ceará" },
  { "id": 35, "sigla": "SP", "nome": "São Paulo" }
]
```

- **Erro (500):** Falha na integração com a API do IBGE.

- **GET /api/estados/:estado/municipios**
  - **Sucesso (200 OK):**

```
// Exemplo para /api/estados/SP/municipios
[
  { "id": 3550308, "nome": "São Paulo" },
  { "id": 3509502, "nome": "Campinas" }
]
```

  - **Erro (404):** Sigla de estado inválida (ex: `/api/estados/XX/municipios`).
  - **Erro (500):** Falha na integração com o IBGE.