

# Software Design Document

## Projeto: Adoção de animais abandonados

Data	13/05/2023
Responsável	Igor Campos de Borba
Autor	Igor Campos de Borba
Doc ID	1
Localização	C:\Users\igor_\OneDrive\Documentos\UNISINOS\5º semestre\Projeto Integrador - Portal Web\1- Definição do tema do Projeto Integrador - Portal Web\Trabalho
Versão do template	0.0.1

## Histórico de revisão

Data	Versão	Autor	Descrição
13/05/2023	0.0.1	Igor Campos de Borba	<p>Redação de:</p> <ul style="list-style-type: none"><li>- Descrição do sistema</li><li>- Público e Stakeholders</li><li>- Regras de negócio</li><li>- Requisitos funcionais e não funcionais</li><li>- Limitações de software</li><li>- Visão funcional e visão técnica</li><li>- Definição de linguagens</li><li>- Resumo da arquitetura de software</li></ul> <p>Diagramas e ferramentas administrativas para análise de requisitos:</p> <ul style="list-style-type: none"><li>- Descrição de História de usuário</li><li>- Diagrama de Cenários</li></ul> <p>Diagramas para:</p> <ul style="list-style-type: none"><li>- Back-end - Diagramas UML: de classes e de objetos.</li><li>- Banco de dados - Diagrama entidade-relacionamento</li><li>- Wireframe de interface (UI)</li></ul>

## Revisor

Nome	Papel	Data
Igor Campos de Borba	Desenvolvedor de software	13/05/2023

## Sumário

Software Design Document .....	1
Histórico de revisão .....	2
Revisor .....	3
Introdução .....	5
Motivação para o projeto .....	5
Público e stakeholders .....	5
Stakeholders – descrições e responsabilidades.....	6
Recursos do software .....	6
Visão funcional .....	6
Visão técnica .....	7
Regras de negócio.....	7
Precedência e prioridade de caso de uso.....	8
Requisitos não-funcionais.....	8
Requisitos não-funcionais de produto.....	8
Requisitos não-funcionais de processo e legais .....	9
Requisitos funcionais .....	9
Requisitos funcionais .....	9
Escopo não contemplado .....	10
Linguagens .....	10
Diagramas .....	11
Análise de requisitos.....	11
Estrutura do sistema.....	12
Wireframe.....	16
Arquitetura de software .....	24
Padrão de 3 camadas (interface, lógica e persistência): .....	25
Planejamento de testes .....	26
Gerenciamento de mudanças de requisitos.....	26
Consulta MySQL para administrador.....	26
Readme de instalação e execução .....	27
PROGRAMAS PADRÃO para instalação:.....	30

## Introdução

Website para adoção de animais abandonados. O projeto visa desenvolver um portal para reunir ONGs e voluntários que defendem a causa animal. Na plataforma, as ONGs podem divulgar animais para adoção, além de compartilhar suas necessidades (ração, medicamentos, materiais de limpeza etc.) com outras ONGs que pudessem cooperar. Os voluntários também podem se informar sobre quando e qual tipo de ajuda as instituições precisariam.

## Motivação para o projeto

Quadro 1 – Motivação para o projeto

Causa	Animais abandonados têm dificuldade de serem adotados.
Quem é afetado	Adotantes, ONGs de pets e administrador desta empresa.
Uma boa solução poderia ser	<p>Website para conectar adotantes a ONGs que cuidam dos pets. Aplicação em 3 camadas:</p> <ul style="list-style-type: none"><li>• Camada de interface (front-end): página de internet com o catálogo de pets e contato das ONGs com a Instituição do software.</li><li>• Camada lógica (back-end): processamento dos dados que são persistidos no banco de dados.</li><li>• Camada de persistência de dados (banco de dados): dados de pets e adotantes salvos para serem encaminhados às ONGs entrarem em contato.</li></ul>

Fonte: Elaborado pelo autor.

## Público e stakeholders

### **Públicos-alvo:**

Entende-se por públicos-alvo o consumidor que o projeto almeja atingir.

1. Adotantes: pessoas físicas que se interessam em adotar um pet (cachorro, gato, passarinho etc.).
2. ONGs: Organizações/associações que cuidam de pets. São elas que fazem o processo de adoção.

### **Stakeholders:**

Entende-se por stakeholder funcionários e investidores envolvidos no projeto.

1. Administrador do sistema.
2. Desenvolvedor de software.

## Stakeholders – descrições e responsabilidades

No âmbito do cliente, pode-se citar:

Quadro 2 – Stakeholders – descrições e responsabilidades

Nome	Descrição	Responsabilidade
Administrador do sistema	É quem gerencia o contato entre o cliente e as ONGs de adoção	<ul style="list-style-type: none"><li>• Negociar com ONGs a divulgação de pets e encaminhamento de mantimentos doados.</li><li>• Verificar no sistema se há novos pedidos de adoção.</li><li>• Encaminhar para as ONGs o contato dos interessados na adoção.</li></ul>

Fonte: Elaborado pelo autor.

## Recursos do software

Estes recursos resumidos compõem as páginas e funcionalidades do website que influenciam o front-end, back-end e banco de dados.

- **Catálogo de pets** (mostrar catálogo de imagens no front. GET no back para recuperar recurso do banco de dados).
- **Formulário de adoção** (página do produto que envia POST/PUT de um cadastro/atualização de pedido de adoção para o banco de dados).
- **Home estática** (banner estático, texto com imagem com link para catálogo, resumo sobre a empresa e o processo de adoção).
- **Doações de mantimentos** (página com texto dos mantimentos necessários por ONG e botão que abre o e-mail no front).
- **Contato** (página para ONG se cadastrar para receber mantimento e contato para outros assuntos com botão que abre o e-mail no front).

## Visão funcional

Website para desktop que permite adotar pets, doar mantimentos e entrar em contato com a organização. O Sistema obterá dados do banco de dados do catálogo de pets e cadastrará novos pedidos de adoção dinamicamente. O website também possui páginas estáticas de doações de mantimentos (informações dos mantimentos que as ONGs necessitam para contato via e-mail) e home.

## Visão técnica

**FRONT-END** mostra catálogo (requisição GET do protocolo HTTP), página do produto tem formulário de pedido de adoção (nome, e-mail e telefone do adotante) que o usuário envia pedido (requisição POST para cadastrar pedido daquele usuário vinculado a aquele pet/produto). Páginas doação de mantimentos e de contato abrem e-mail do sistema operacional do usuário via protocolo mailto (doadores entram em contato via e-mail com a Instituição, ONGs contatam empresa para que se cadastrem e usuários contatarem para dúvidas gerais).  
Tecnologias: Angular, TypeScript, Angular Material UI, JavaScript, HTML e CSS.

**BACK-END** se comunica com o Front via API REST e com banco de dados via Hibernate (que implementa o padrão da JPA de requisições). Este back-end organiza a informação via paradigma de orientação a objetos para recuperar dados do banco de dados (catálogo de pets) e cadastrar pedidos de adoção.  
Tecnologias: Java e Spring Boot.

**BANCO DE DADOS:** O SGBD MySQL Workbench (paradigma relacional) armazena em tabelas (entidades) o catálogo de pets e adotantes. O Administrador do sistema consulta o banco de dados para verificar se há um novo pedido de adotante de um pet.  
Tecnologia: MySQL.

**SERVIDOR:** hospedagem em servidor compartilhado e gratuito da Railway em sistema operacional Linux, com interface no navegador. Espaço de 1GB em disco, tráfego por hora 500 horas por mês, servidor com 512mb de RAM, CPU com 2 núcleos.  
Path do endereço .railway.app

## Regras de negócio

Quadro 3 – Regras de negócio

ID	Regra de negócio	Descrição	Caso de uso resumido
RN1	Catálogo de pets	Todos os pets devem ter imagem e nome da ONG. Informações opcionais: porte, gênero, saúde, idade aproximada, temperamento.	Clareza da informação para auxiliar no processo de adoção.
RN2	Formulário de adoção de pet	Adotante deve informar nome, e-mail e telefone. Banco de dados deve registrar data e hora da solicitação.	Formulário de contato válido.
RN3	Arquitetura da informação deve estruturar as	Adotantes e ONGs devem entender o processo de adoção,	- Adotante: Entendimento do processo de adoção.

	informações de forma clara sobre processo de adoção, doação e mantimentos	doação e recebimento de mantimentos.	- ONG: Encaminhamento de contato de adotante para ONG.
--	---	--------------------------------------	---

Fonte: Elaborado pelo autor.

## Precedência e prioridade de caso de uso

Quadro 4 – Precedência e prioridade de caso de uso

Caso de uso	Precedência	Prioridade
Catálogo de pets implementado no back, banco e front-end para processo com adotantes e ONGs estar funcional.	1	Alta
Formulário de contato da página de pet deve receber informações válidas.	2	Média
Página de contato da ONG e dúvidas gerais deve estar implementado no front-end com envio para e-mail.	3	Média
Home deve direcionar para página de pets e ter um resumo sobre a empresa e processo de adoção.	4	Baixa

Fonte: Elaborado pelo autor.

## Requisitos não-funcionais

### Requisitos não-funcionais de produto

Quadro 5 – Requisitos não-funcionais de produto

ID	Requisito não-funcional	Descrição	Caso de uso resumido
PRNF1	Confiabilidade	Os dados obrigatórios devem ser enviados ao banco de dados, sem faltar nenhum campo requerido.	Todos os dados obrigatórios do adotante devem ser preenchidos.
PRNF2	Usabilidade – facilidade de uso para ONGs	Associados às ONGs devem entrar em contato facilmente com a Instituição do sistema	A ONG deve entrar em contato para divulgar pets disponíveis para adoção no site.
PRNF3	Guia de consulta SQL para administrador do sistema	Será enviado um documento de guia sobre quais comandos o administrador deve executar para verificar se há um novo adotante no banco de dados.	Encaminhamento de contato de adotante para ONG.



PRNF4	Licença GPL (General Public License)	Licença de software de código aberto para uso, estudo, modificação e distribuição.	-
-------	--------------------------------------	--	---

Fonte: Elaborado pelo autor.

## Requisitos não-funcionais de processo e legais

Quadro 6 – Requisitos não-funcionais de processo e legais

ID	Requisito não-funcional	Descrição	Caso de uso
ORNF1	Clareza da informação	Deve estar claro que o site divulga pets e qual ONG entra em contato com o adotante.	Entendimento do processo de adoção pelo adotante.
ORNF2	Termos de uso	Documento de termos de uso e de processamento de dados pessoais.	Requerimento Legal.
ORNF3	Prazo do projeto	O sistema front, back-end e banco de dados deve estar pronto até 22/06/2023	Prazo da disciplina.

Fonte: Elaborado pelo autor.

## Requisitos funcionais

### Requisitos funcionais

Quadro 7 – Requisitos funcionais

ID	Requisito funcional	Descrição	Caso de uso resumido
RF1	Catálogo de pets	- Front-end: catálogo de imagens e página do pet. - Back-end: receber requisição GET para obter pets no banco de dados.	Gerenciar informações de pets.
RF2	Formulário de adoção de pets	- Front-end: formulário com nome, e-mail e telefone. - Back-end: 1) receber requisição POST para cadastrar adotante do pet no banco de dados. 2) validar se já existe cadastro no banco daquele e-mail para aquele pet. Se já tiver no banco o cadastro, só atualizar registro com PUT e salvar	Gerenciar informações de adotantes.

		data e hora da solicitação de adoção.	
RF3	Página de doação de mantimento	Página com texto dos mantimentos necessários por ONG e botão que abre o e-mail da ONG no sistema operacional do usuário no front-end. A ONG pode optar por ao invés de ter um botão para o e-mail, substituir por um botão para o site/rede social da ONG.	Informar adotante dos mantimentos que ele pode doar para a instituição e direcionar o usuário para doação de mantimentos.
RF4	Página de contato	Página para ONG se cadastrar para receber mantimentos e para cadastrar divulgação de pets. E contato geral para outros assuntos com botão que abre o e-mail no front-end. Destinatário do e-mail é o administrador desta empresa.	Página para ONG se cadastrar para ela receber mantimentos e divulgar pets. Atender dúvidas gerais.

Fonte: Elaborado pelo autor.

## Escopo não contemplado

O que o software não possui de funcionalidade e alternativas:

- A home não terá conteúdo dinâmico. Ela será alterada estaticamente via código pelo desenvolvedor de software
- As páginas 1) de doação de mantimento e 2) de contato não salvarão os dados em banco de dados. Essas páginas abrirão o software de e-mail instalado na máquina do usuário.

## Linguagens

Versões LTS para evitar incompatibilidade entre linguagens e incrementos do sistema.

O website será feito apenas para desktop.

- Angular 12 e TypeScript 5.0.4
- SpringBoot 2.3.3 e Java 11
- MySQL 5.7.41

Motivo da escolha das linguagens e frameworks.

- Angular e TypeScript: orientada a objetos e tipada.

- SpringBoot e Java: orientada a objetos e configuração simplificada com as annotations e fácil integração com biblioteca Hibernate (mapeamento objeto-relacional).
- MySQL: paradigma relacional proporciona confiabilidade na persistência correta dos dados.

## Diagramas

### Análise de requisitos

O objetivo destes diagramas é auxiliar na análise de requisitos:

- **Descrição de história de usuário**

Quadro 8 – Descrição de história de usuário

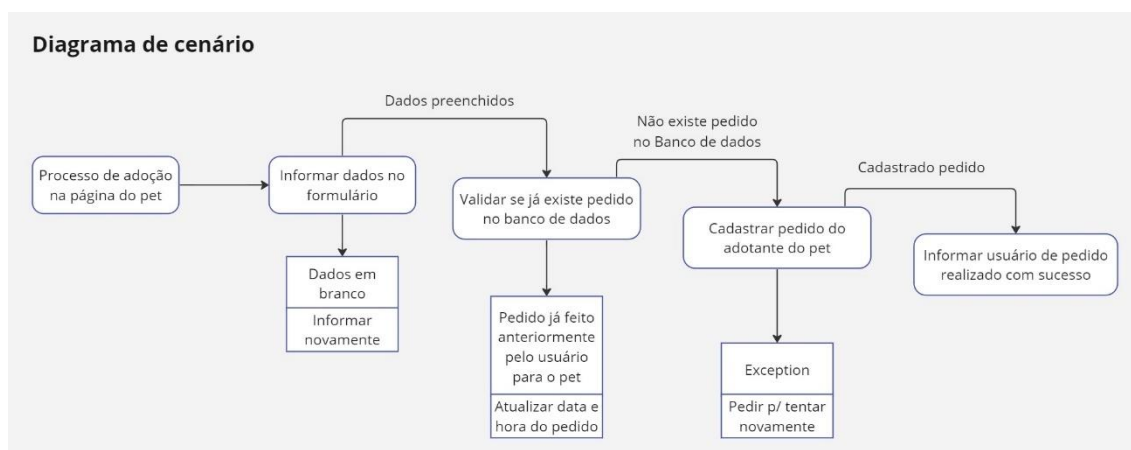
ID	História de usuário
H1	Como um <b>adotante</b> de pet eu quero <b>adotar pets</b> de modo que seja <b>fácil o processo</b> .
H2	Como um <b>adotante</b> de pet eu quero <b>doar mantimentos</b> de modo que <b>ajude na subsistência</b> dos animais.
H3	Como uma <b>ONG</b> eu quero <b>divulgar meus pets e pedir mantimentos</b> de modo que <b>amplie o alcance do público-alvo</b> .
H4	Como um <b>adotante</b> eu quero <b>tirar dúvidas do processo de adoção</b> de modo que eu compreenda.

Fonte: Elaborado pelo autor.

- **Diagramas de cenários**

Diagrama de cenário: processo de adoção na página do pet.

Figura 1 – Diagrama de cenário: processo de adoção na página do pet



Fonte: Elaborado pelo autor.

Diagrama de cenário: processo de doação de mantimento

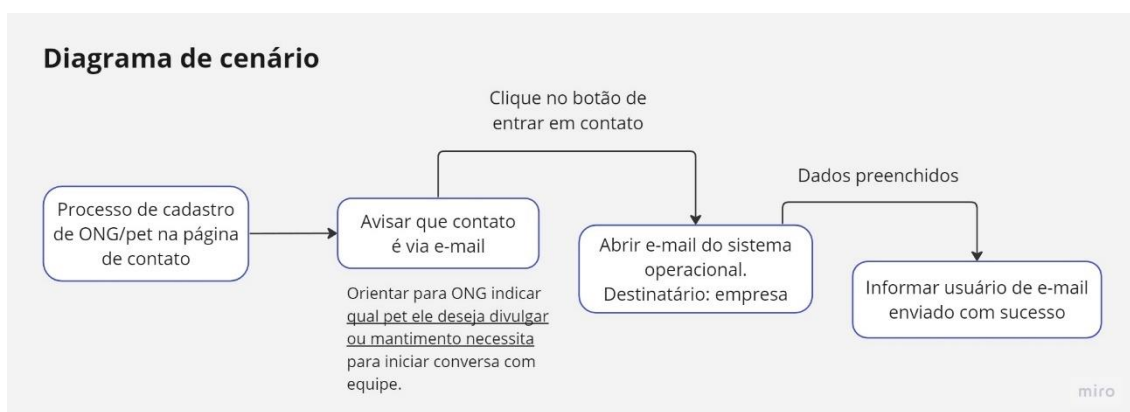
Figura 2 – Diagrama de cenário: processo de doação de mantimento



Fonte: Elaborado pelo autor.

Diagrama de cenário: processo de cadastro de ONG para divulgação de pet ou solicitação de mantimentos.

Figura 3 – Diagrama de cenário: processo de cadastro de ONG para divulgação de pet ou solicitação de mantimentos



Fonte: Elaborado pelo autor.

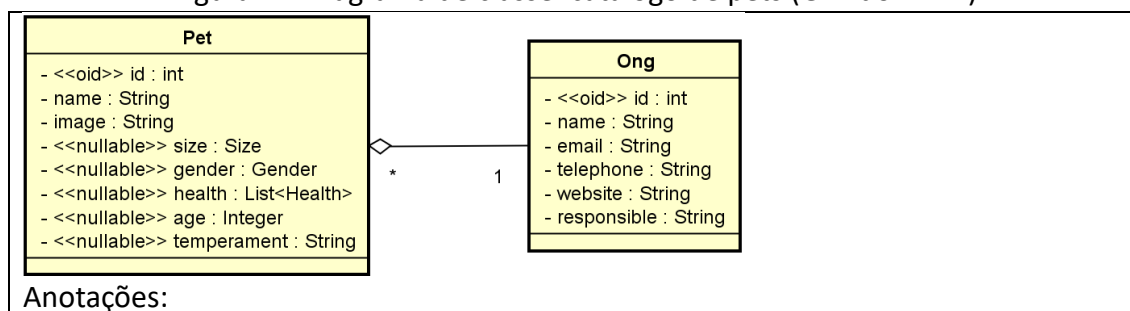
## Estrutura do sistema

O objetivo destes diagramas é estruturar a arquitetura do software lógica e visual:

**Back-end** – Diagramas UML de classes e de objetos.

## Diagrama UML de classes

Figura 4 – Diagrama de classe: catálogo de pets (GET do HTTP)



- Cadastrei a ONG porque é obrigatório que haja uma ONG que divulga o animal para um Pet existir no sistema.

Diamante branco (agregação em UML) porque uma ONG pode possuir vários Pets para adoção.

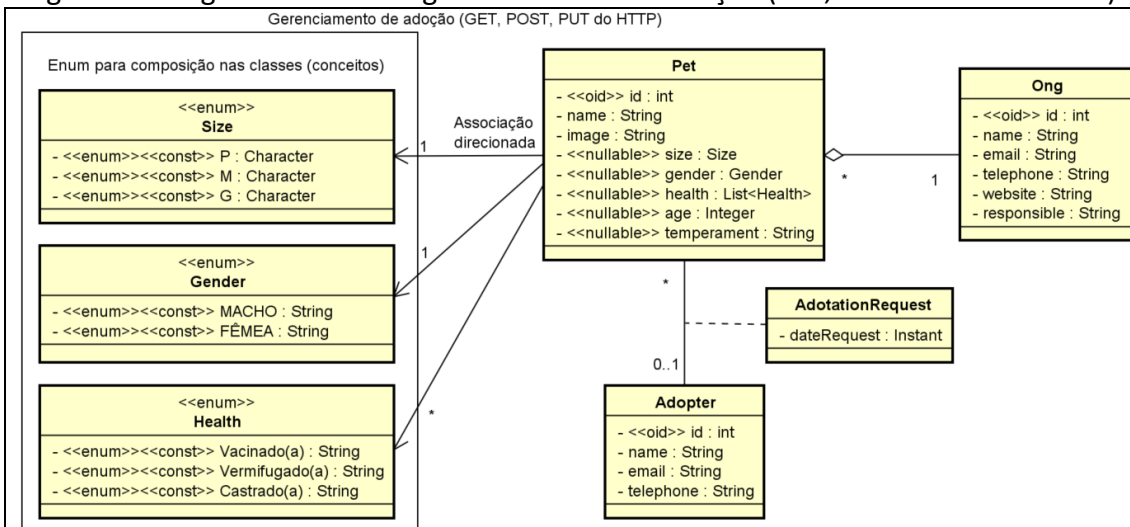
- Atributos com estereótipo <<nullable>> admitem valores null e é opcional o preenchimento. <<oid>> só indica a chave-primária.

- Atributo health é uma lista/array do tipo String porque o valor pode ser por exemplo Vacinado(a), Castrado(a), Vermifugado(a).

- Métodos get e sets são implementados para todos os atributos, exceto id que terá get.

Fonte: Elaborado pelo autor.

Figura 4 – Diagrama de classe: gerenciamento de adoção (GET, POST e PUT do HTTP)



Anotações:

- Coloquei o relacionamento a partir de zero em Adopter e Pet porque inicialmente um Pet não tem adotante e um adotante pode não ter nenhum pet num determinado momento.

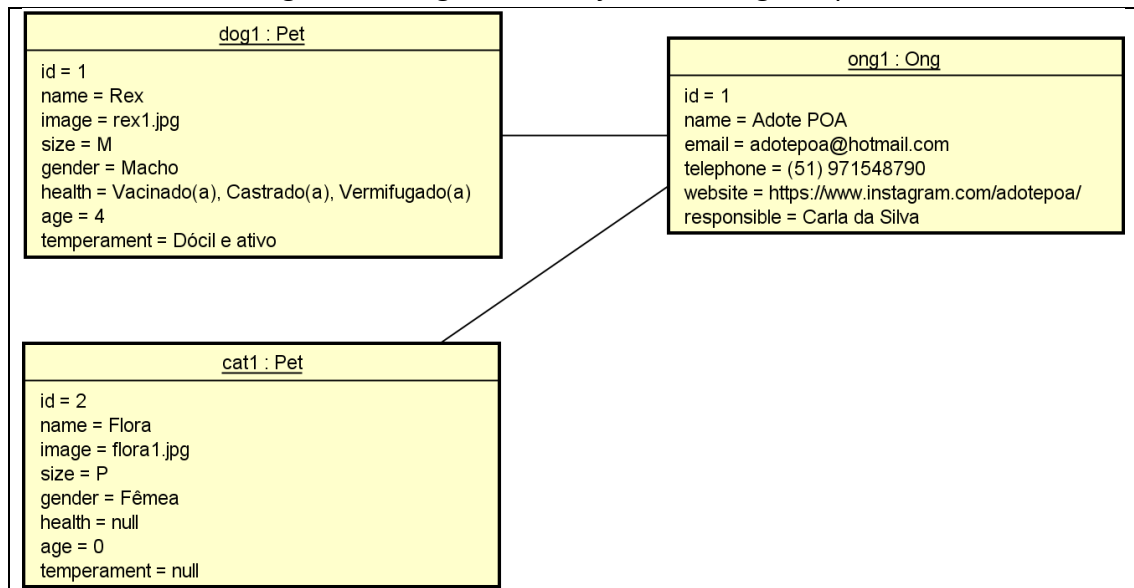
- AdotationRequest é a data e hora do envio do formulário do pedido de adoção.

- Os conceitos (classes) enum apenas o Pet possui a composição (associação direcionada).

Fonte: Elaborado pelo autor.

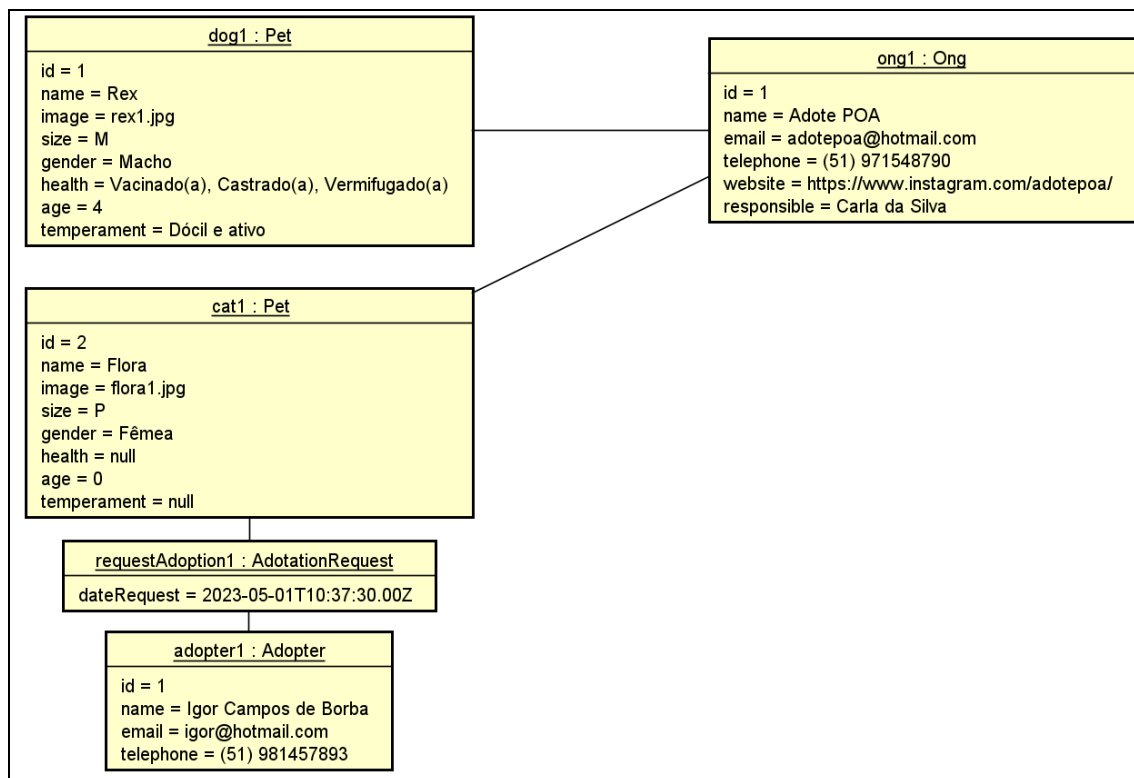
## Diagrama UML de objetos

Figura 5 – Diagrama de objetos: catálogo de pets



Fonte: Elaborado pelo autor.

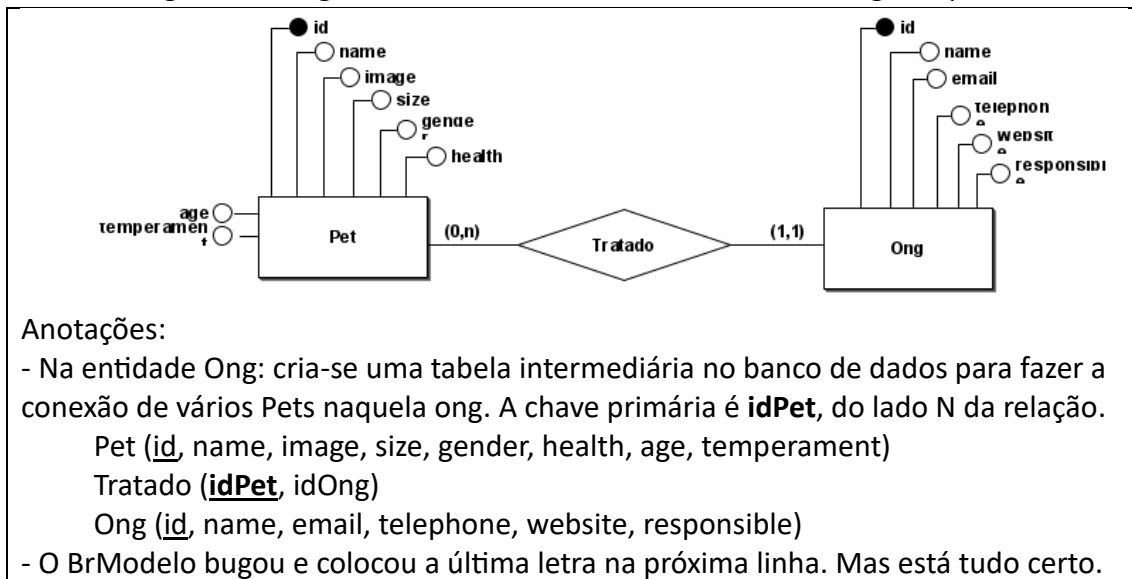
Figura 6 – Diagrama de objetos: gerenciamento de adoção



Fonte: Elaborado pelo autor.

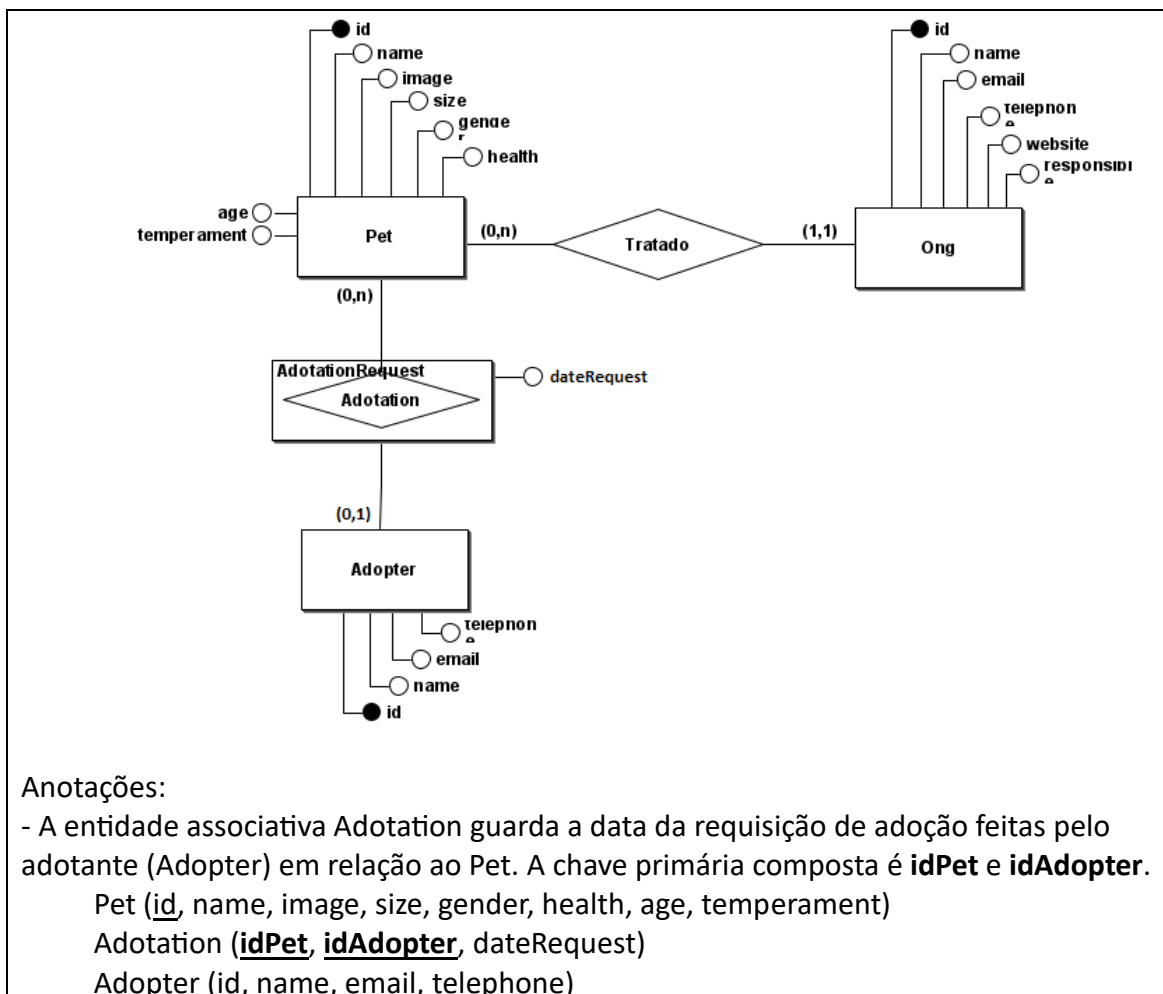
## Banco de dados – Diagramas entidade-relacionamento (DER).

Figura 7 – Diagrama de entidade-relacionamento: catálogo de pets



Fonte: Elaborado pelo autor.

Figura 8 – Diagrama de entidade-relacionamento: gerenciamento de adoção



- Na entidade Ong: é o mesmo caso de cima. Cria-se uma tabela intermediária no banco de dados para fazer a conexão de vários Pets naquela ong. A chave primária é **idPet**.

Pet (id, name, image, size, gender, health, age, temperament)

Tratado (**idPet**, idOng)

Ong (id, name, email, telephone, website, responsible)

Fonte: Elaborado pelo autor.

## Wireframe

Wireframe de média fidelidade, para desktop. Ele é uma variação de wireframe com layout para tornar mais inteligível a proposta, com média complexidade visual.

- **Composição visual**

Cores primárias: Branco, azul claro e azul escuro.

Ícones que remetem a animais domésticos.

Motivos: essas cores do tipo análogas transmitem sensação de calma, profissionalismo e confiança. Em conjunto compõem um visual minimalista, de modo que o azul escuro contrasta o CTA (botão de call-to-action). O branco é uma cor neutra que atua como uma área de descanso e destaca os elementos por meio da ampliação do espaço visual.



Figura 9 – Home – desktop



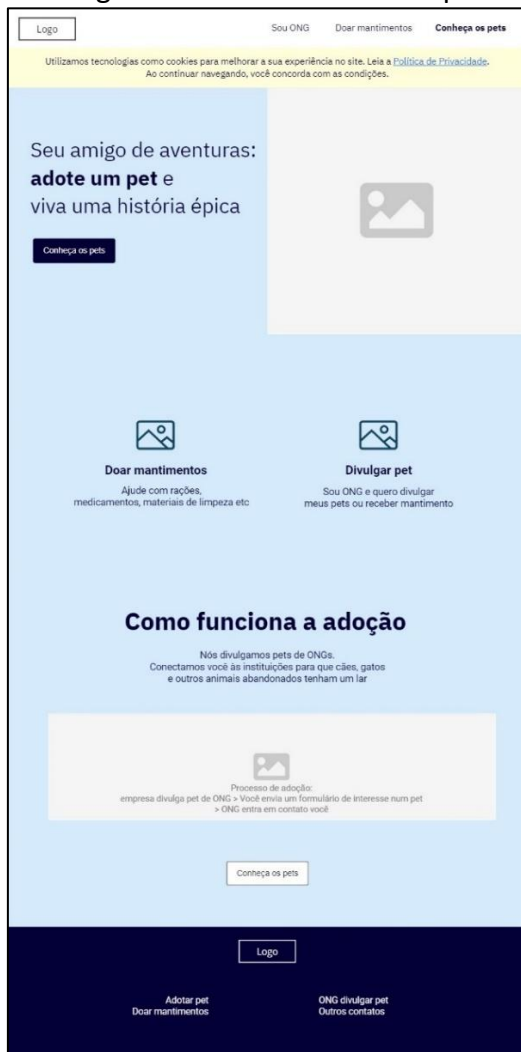
Fonte: Elaborado pelo autor.

Figura 10 – Home – mobile



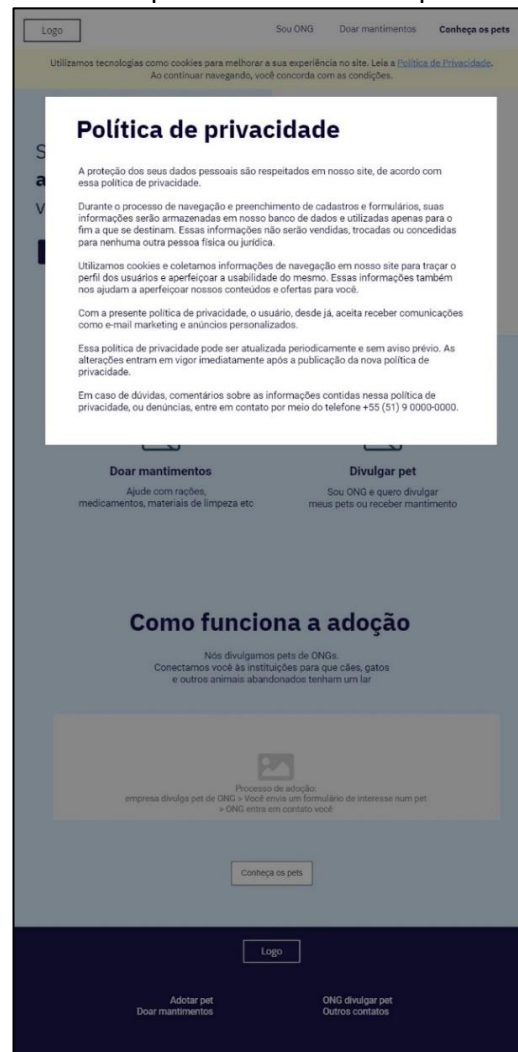
Fonte: Elaborado pelo autor.

Figura 11 – Cookies – desktop



Fonte: Elaborado pelo autor.

Figura 12 – Política de privacidade – desktop



Fonte: Elaborado pelo autor.

Figura 13 – Cookies  
– mobile



Fonte: Elaborado pelo autor.

Figura 14 – Política de privacidade  
– mobile



Fonte: Elaborado pelo autor.

Figura 15 – Catálogo de pets – desktop



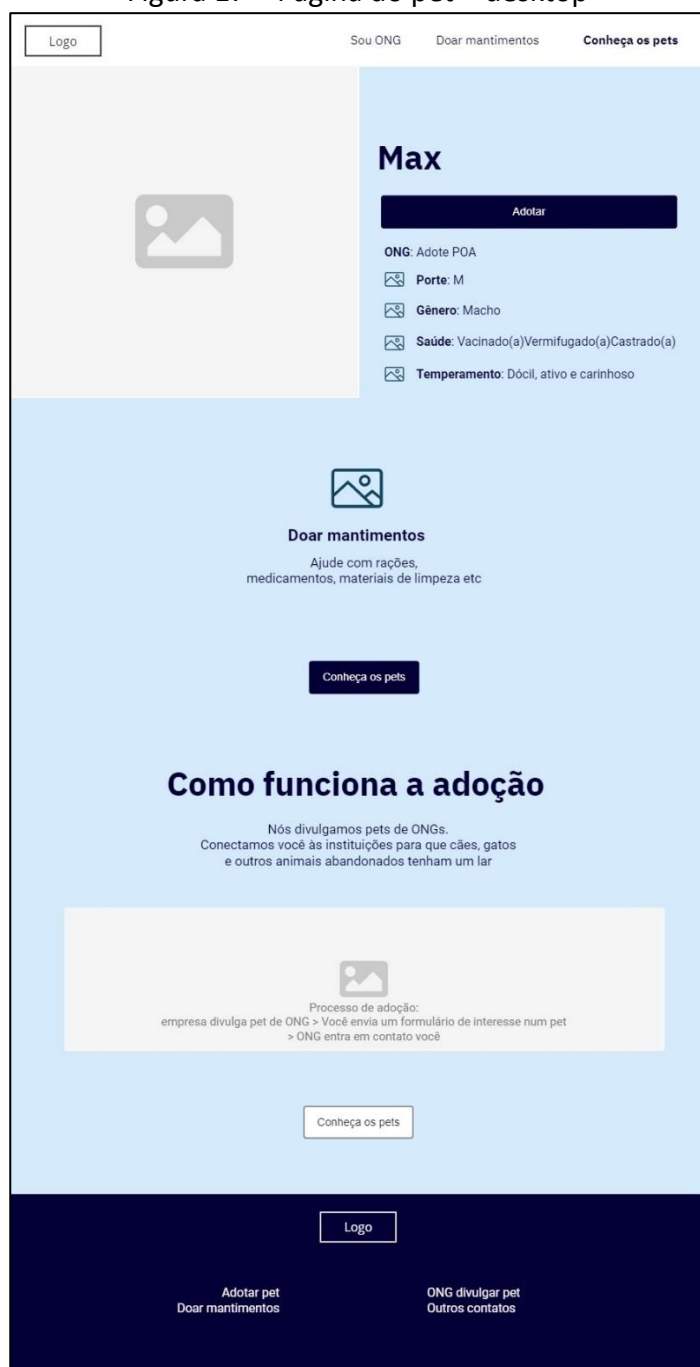
Fonte: Elaborado pelo autor.

Figura 16 – Catálogo de pets – mobile



Fonte: Elaborado pelo autor.

Figura 17 – Página do pet – desktop



Fonte: Elaborado pelo autor.

Figura 18 – Página do pet – mobile



Fonte: Elaborado pelo autor.

Figura 19 – Página pet – formulário – desktop

Logo

Sou ONG Doar mantimentos Conheça os pets

## Me adota

Nome

E-mail

Telefone

Enviar

Após o envio, aguarde a ONG entrar em contato.

### Doar mantimentos

Ajude com rações, medicamentos, materiais de limpeza etc

Conheça os pets

## Como funciona a adoção

Nós divulgamos pets de ONGs. Conectamos você às instituições para que cães, gatos e outros animais abandonados tenham um lar

Processo de adoção:  
empresa divulga pet de ONG > Você envia um formulário de interesse num pet  
> ONG entra em contato você

Conheça os pets

Logo

Adotar pet  
Doar mantimentos

ONG divulgar pet  
Outros contatos

Fonte: Elaborado pelo autor.

Figura 20 – Página pet – formulário – mobile

Logo

## Me adota

Nome

E-mail

Telefone

Enviar

Após o envio, aguarde a ONG entrar em contato.

### Doar mantimentos

Ajude com rações, medicamentos, materiais de limpeza etc

Conheça os pets

## Como funciona a adoção

Nós divulgamos pets de ONGs. Conectamos você às instituições para que cães, gatos e outros animais abandonados tenham um lar

Processo de adoção:  
empresa divulga pet de ONG > Você envia um formulário de interesse num pet  
> ONG entra em contato você

Conheça os pets

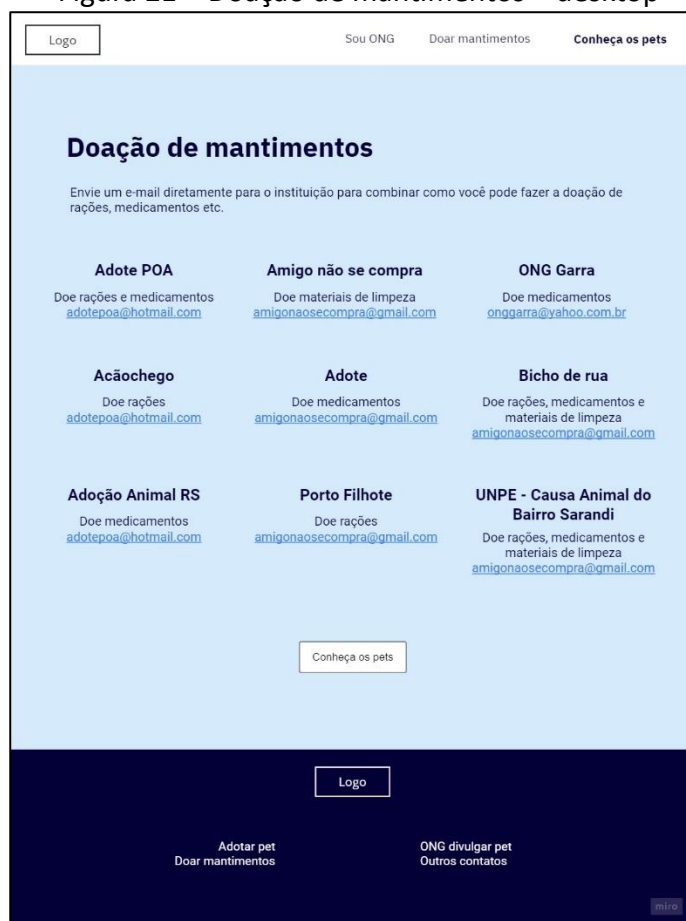
Logo

Adotar pet  
Doar mantimentos

ONG divulgar pet  
Outros contatos

Fonte: Elaborado pelo autor.

Figura 21 – Doação de mantimentos – desktop



Fonte: Elaborado pelo autor.

Figura 22 – Doação mantimentos – mobile



Fonte: Elaborado pelo autor.

Figura 23 – Contato geral – desktop



Fonte: Elaborado pelo autor.

Figura 24 – Contato geral - mobile



Fonte: Elaborado pelo autor.

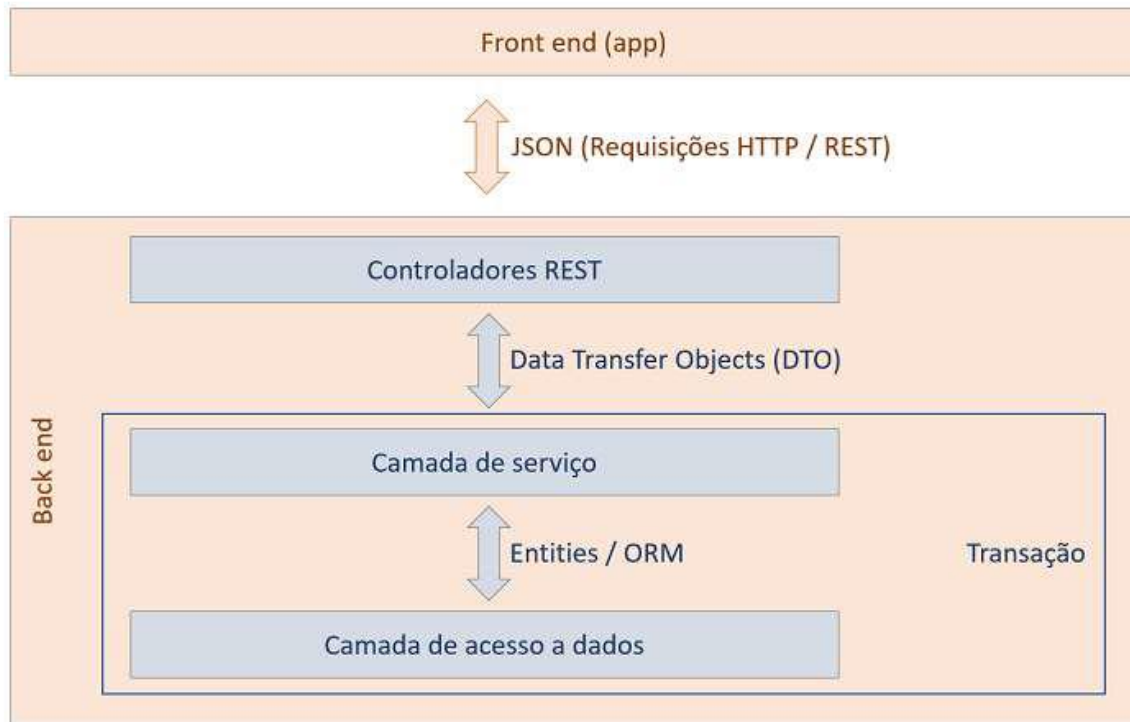
## Arquitetura de software

- **BACK-END:** Java usa padrão de projeto que implementa DTO para eu escolher os campos que quero do banco, JPA para mapeamento objeto relacional no lado do Java quando pego um dado do banco, padrão JPA é implementado pelo Hibernate que simplifica requisições SQL porque já estão pré-configuradas. Padrão REST para requisições, usando o protocolo HTTP.
- **BANCO DE DADOS:** MySQL armazena em entidades (tabelas) os dados interagidos com o back-end.
- **FRONT E BACK-END:** Arquitetura de projeto MVC.
- **FRONT-END:** arquitetura baseada em componentes para criar lógica dos componentes por meio do Angular. Angular Material UI para usar componentes pré-prontos.



Padrão de 3 camadas (interface, lógica e persistência):

Figura 25 – Padrão de 3 camadas



Fonte: Elaborado pelo autor.

- **Front-end:** Camada de apresentação e interação desenvolvida em Angular, JavaScript, HTML e CSS.
- **JSON:** formato que encapsula objetos com notação JavaScript para transmitir dados entre sistemas. Exemplo: um JSON é retornado como resposta para o GET (do HTTP) do catálogo no front-end.
- **Controller REST (resource):** expõe os endpoints para receber as requisições do front-end do protocolo HTTP.
- **DTO (data transfer objects):** eu defino quais dados do banco serão transmitidos ao front-end, por exemplo eu posso ignorar o responsável pela ONG. E separa a camada lógica do service com a apresentação do front.
- **Service:** parte lógica das regras de negócio, por exemplo a manipulação dos dados recebidos no front-end dos dados do adotante.
- **Entities (classes de domain) / ORM:** Mapeamento objeto-relacional. Mapeia a entity que é um objeto no Java para uma entity relacional do MySQL.
- **Camada de acesso a dados (repository):** Java faz a comunicação com o banco de dados por meio da implementação da interface JPRepository para ter acesso a métodos que facilitam o uso de operações básicas de CRUD, como o Read, Create e Update que eu utilizo para mostrar o catálogo de pet e adicionar/editar um adotante.

## Planejamento de testes

**Teste de classe de equivalência:** testar casos válidos e inválidos de inputs e verificar resultados dos outputs.

**Teste de usabilidade:** avaliar a facilidade de usar e a interface. Avaliação nocional e via Quant-ux e Google Page Speed.

## Gerenciamento de mudanças de requisitos

Processo de alteração do website:

1. O administrador solicita via e-mail a alteração de requisitos.
2. O desenvolvedor de software recebe a solicitação e a analisa para mensurar o impacto e necessidade.
3. O desenvolvedor negocia com o administrador a alteração.
4. O resultado da negociação é a decisão se ocorrerá ou não a mudança solicitada.

Nesse processo inclui-se cadastros e remoções de pets no Banco de Dados pelo desenvolvedor de software a pedido do administrador. O programador fará a inserção e deleção pelo Spring Boot. Isso porque há relacionamentos entre tabelas que não podem gerar inconsistência nos dados em caso de erro de manipulação das consultas.

## Consulta MySQL para administrador

O administrador tem apenas a responsabilidade de verificar no sistema se há novos pedidos de adoção por meio de consultas MySQL e encaminhar esses dados para as ONGs o contato dos interessados na adoção.

- Administrador consultar pedidos de adoção no Banco de Dados:

```
SELECT
  APA.PET_ID,
  APA.ADOPTER_NAME,
  APA.PET_NAME,
  AD.NAME,
  AD.EMAIL,
  AD.TELEPHONE
FROM
  ADOPTER_PET_ASSOCIATION AS APA
  JOIN ADOPTER AS AD ON APA.ID = AD.ID;
```

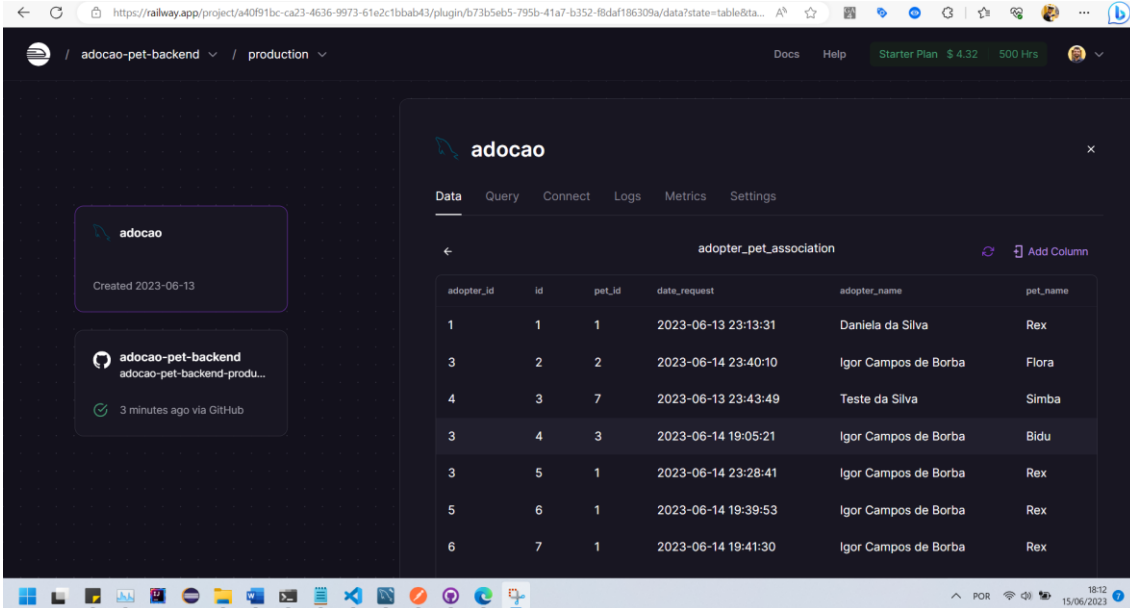
## Readme de instalação e execução

- Site disponível em: <https://adocaopet.vercel.app/>
- Vídeo de explicação do código e execução: <https://unisinios.instructuremedia.com/embed/9e3e5268-da33-4172-a02e-1325269dc03f>
- Código-fonte do Fron-end: <https://github.com/igorcamosdeborba/adocao-pet-frontend>
- Código-fonte do Back-end: <https://github.com/igorcamosdeborba/adocao-pet-backend>

Relatório: o trabalho está 100% concluído segundo este planejamento de software. E foi colocado no ar (feito deploy) em <https://adocaopet.vercel.app/>

Exemplos de execução:

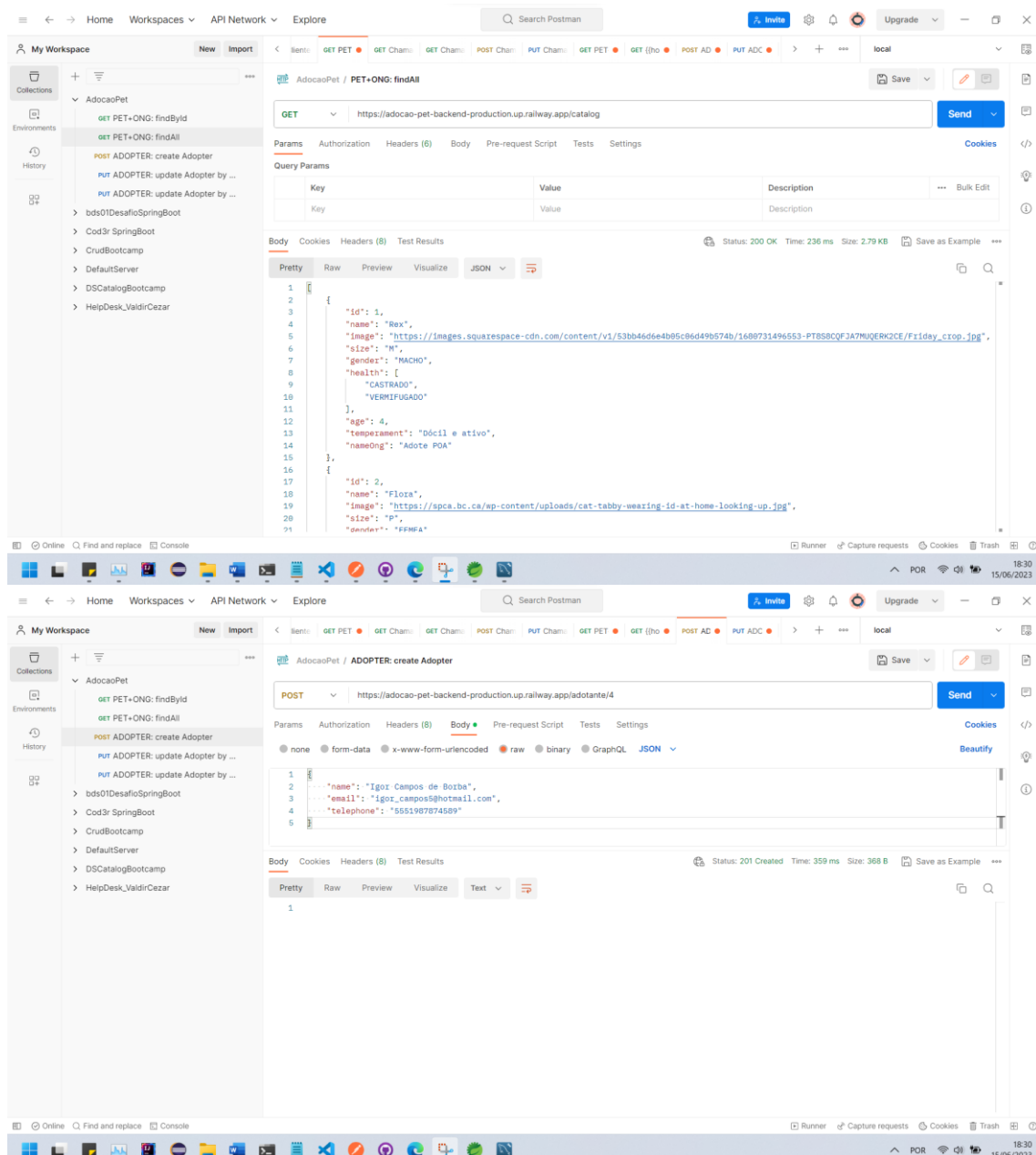
### Banco de Dados MySQL no Railway:



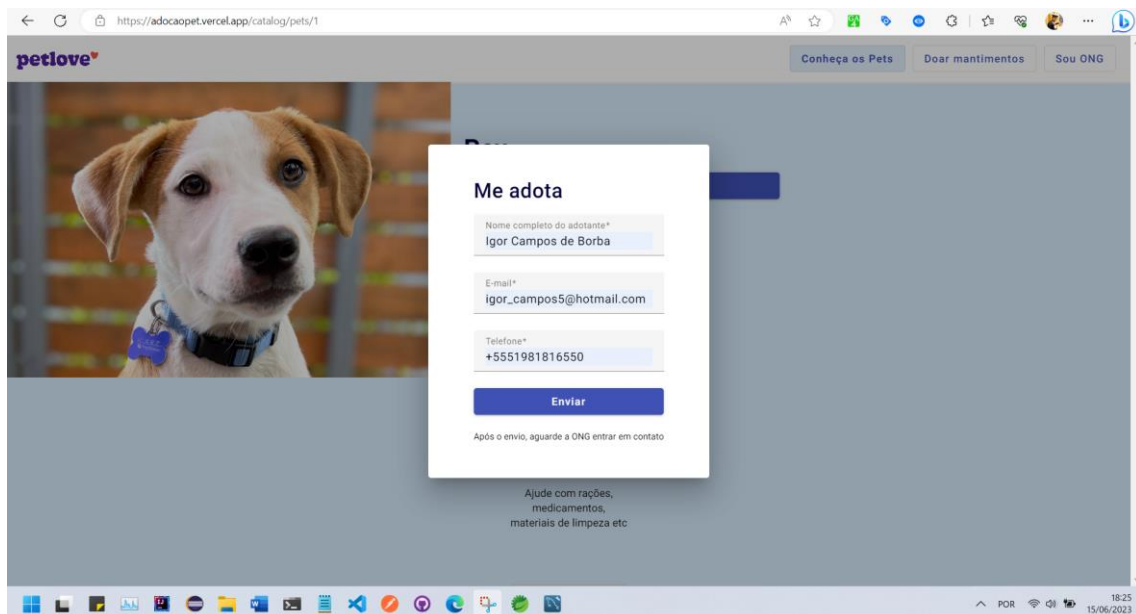
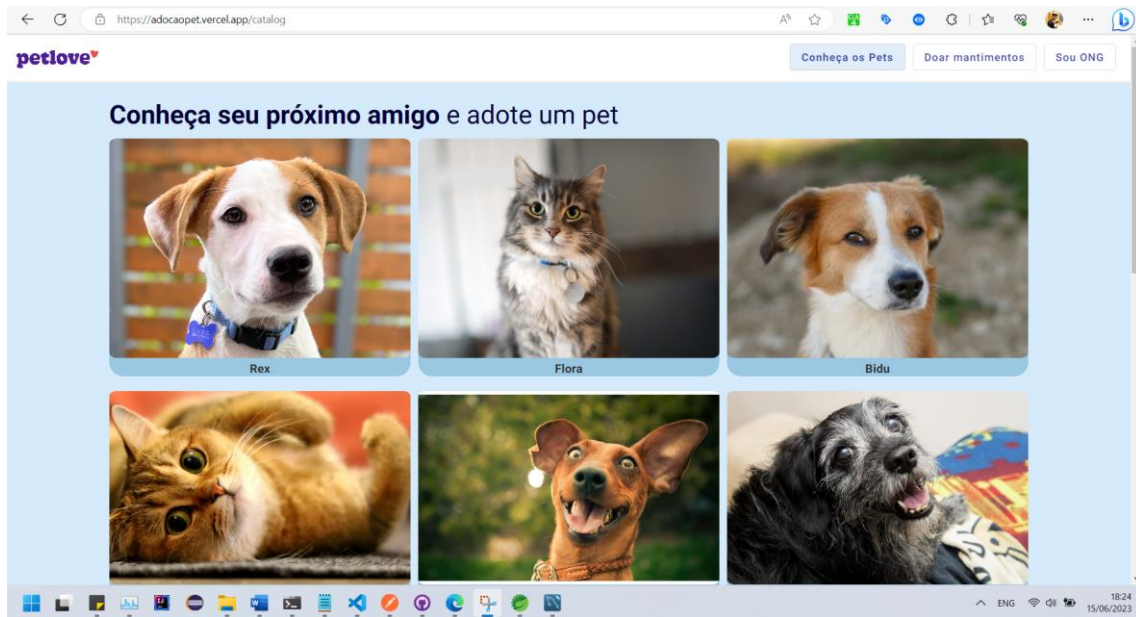
The screenshot shows the Railway app interface. On the left, there's a sidebar with a list of services: 'adocao' (Created 2023-06-13) and 'adocao-pet-backend' (adocao-pet-backend-produ... 3 minutes ago via GitHub). The main panel displays the 'adocao' database. Below the database name, there's a table named 'adopter\_pet\_association'. The table has the following columns: 'adopter\_id', 'id', 'pet\_id', 'data\_request', 'adopter\_name', and 'pet\_name'. The table contains 8 rows of data.

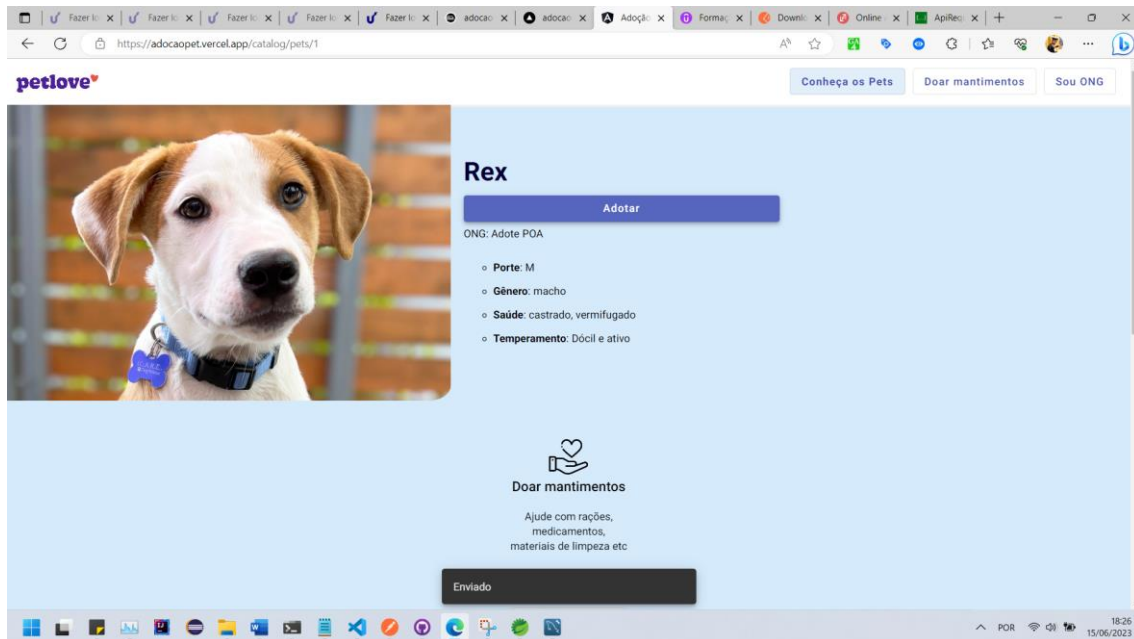
adopter_id	id	pet_id	data_request	adopter_name	pet_name
1	1	1	2023-06-13 23:13:31	Daniela da Silva	Rex
3	2	2	2023-06-14 23:40:10	Igor Campos de Borba	Flora
4	3	7	2023-06-13 23:43:49	Teste da Silva	Simba
3	4	3	2023-06-14 19:05:21	Igor Campos de Borba	Bidu
3	5	1	2023-06-14 23:28:41	Igor Campos de Borba	Rex
5	6	1	2023-06-14 19:39:53	Igor Campos de Borba	Rex
6	7	1	2023-06-14 19:41:30	Igor Campos de Borba	Rex

Java com Spring Boot – endpoints:



**Angular conteúdo dinâmico:**





## PROGRAMAS PADRÃO para instalação:

- Instalar NPM

`npm install -g npm@latest`

- Instalar Node última versão

<https://nodejs.org/en/download>

- Instalar NVM (Para mudar versão do Node em tempo de execução)

Download e instale nvm-setup.exe no link <https://github.com/coreybutler/nvm-windows/releases>

- Instalar node na versão 12.14.1

`nvm install 12.14.1`

- Usar o node na versão 12.14.1

`nvm use 12.14.1`

- Instalar plugin no Chrome ou edge para aceitar CORS origin em servidor local E ativar plugin clicando nele:

<https://chrome.google.com/webstore/detail/allow-cors-access-control/lhobafahddgcelffkeicbaginieeejlf/related?hl=pt>

- Instalar MySQL Community Server 5.7.41 (GPL)

<https://dev.mysql.com/downloads/mysql/>

- Instalar CLI do Eclipse Spring Tool Suite 4:

<https://spring.io/tools>

- Instalar Java JDK 11:

<https://www.oracle.com/br/java/technologies/javase/jdk11-archive-downloads.html>

- Instalar Postman:  
<https://www.postman.com/downloads/>

---

**Como executar:**

**ANGULAR:**

- Iniciar Angular liberando o CORS Origin  
ng serve --proxy-config proxy.config.js

- Acessar site:  
<http://localhost:4200/>  
OU  
<https://adocaopet.vercel.app/>

---

**Como executar:**

**SPRING BOOT e MYSQL:**

- Iniciar MySQL Workbench:  
criar banco de dados:  
> nome: root  
> senha: 123456  
> path e porta: 127.0.0.1:3306

- Iniciar Spring Boot:  
Clicar na seta verde em qualquer arquivo aberto do projeto.

- Testar endpoints pelo postman (colocar url na requisição e texto do body no campo body):

Requisição	URI
GET: PET+ONG - findAll	https://adocao-pet-backend-production.up.railway.app/catalog
GET: PET+ONG - findById	https://adocao-pet-backend-production.up.railway.app/catalog/pets/1
POST: ADOPTER - create/edit adopter and adopt pet	<p>https://adocao-pet-backend-production.up.railway.app/adotante/4</p> <p>Body: {   "name": "Igor Campos de Borba",   "email": "igor@hotmail.com",   "telephone": "5551987874589" }</p>