AcheiPET

1. Introdução

1.1 Objetivo

Este projeto tem como objetivo criar um software para conectar instituições e ONGs de apoio a animais abandonados à interessados em adoção de animais de estimação. Estabelecendo também apoio a animais perdidos e achados.

1.2 Escopo

O sistema oferecerá interfaces para comunicação via postagens e feeds entre usuários e ONGs, permitindo o cadastro de organizações, usuários, e postagens sobre animais perdidos e abandonados achados.

As principais funcionalidades do sistema, e que tem portanto maior prioridade, são o cadastro das ONG's, tutores e voluntários, permitir que tutores divulguem pets perdidos e permitir que ONG's divulguem pets encontrados. Além disso, foram levantados requisitos de interesse das ONG's como permitir que publiquem notícias, que atualizem a situação de animais divulgados e permitir que vejam listas de interessados em voluntariar. Alguns requisitos que interessam aos voluntários são encontrar ONG's próximas e sinalizar interesse em voluntariar em ONG's específicas. No que diz respeito aos usuários não cadastrados, há interesse em visualizar notícias e visualizar informativos sobre procedimentos a adotar em caso de encontrar um animal em situação de rua.

1.3 Visão geral

Os próximos tópicos descrevem quais serão os requisitos e restrições utilizados para definir a arquitetura a ser implementada, bem como, quais atributos de qualidades serão priorizados e o porquê da escolha.

Quais os padrões arquiteturais serão utilizados conforme os atributos de qualidade selecionados e como funcionará o trade-off entre esses padrões arquiteturais, bem como o porquê da escolha dos padrões arquiteturais.

Quais e como as visões arquiteturais serão detalhadas e quais os pontos de vista da arquitetura serão utilizados para descrever as visões.

2. Contexto da Arquitetura

2.1 Funcionalidades e Restrições Arquiteturais

Id	Tipo	Descrição
RAS01	Requisitos Não-Funcionais	Integrar com o google maps API Maps JavaScript
RAS02	Requisitos Não-Funcionais	O Sistema deve estar ativo durante 99% do tempo
RAS03	Requisitos Não-Funcionais	O sistema deve ser intuitivo o suficiente para que um novo usuário consiga executar tarefas básicas sem a necessidade de treinamento, por meio de um fluxo lógico e orientações contextuais
RAS04	Requisitos Não-Funcionais	A API do sistema deve estar exposta e poderá ser acessada por sistemas externos de clientes, fornecedores e parceiros. Este acesso precisa ser seguro, com autenticação em nível do servidor e em nível da aplicação

2.2 Atributos de Qualidades Prioritários

Baseado nos requisitos funcionais e principalmente não funcionais do sistema, foram definidos os seguintes atributos de qualidade a serem priorizados na arquitetura do sistema:

2.2.1 Segurança

Como serão manipulados dados sensíveis dos usuários e terão áreas que precisarão de autorização de acesso, a arquitetura deve ser modelada com foco em proteger esses pontos.

2.2.2 Usabilidade

Segundo pesquisas do IBGE no site canaldopet a idade média dos tutores de pets brasileiros está entre 30 e 45 anos, com isso, um atributo de qualidade importante para facilitar a navegação do usuários pelo software

2.2.3 Manutenibilidade

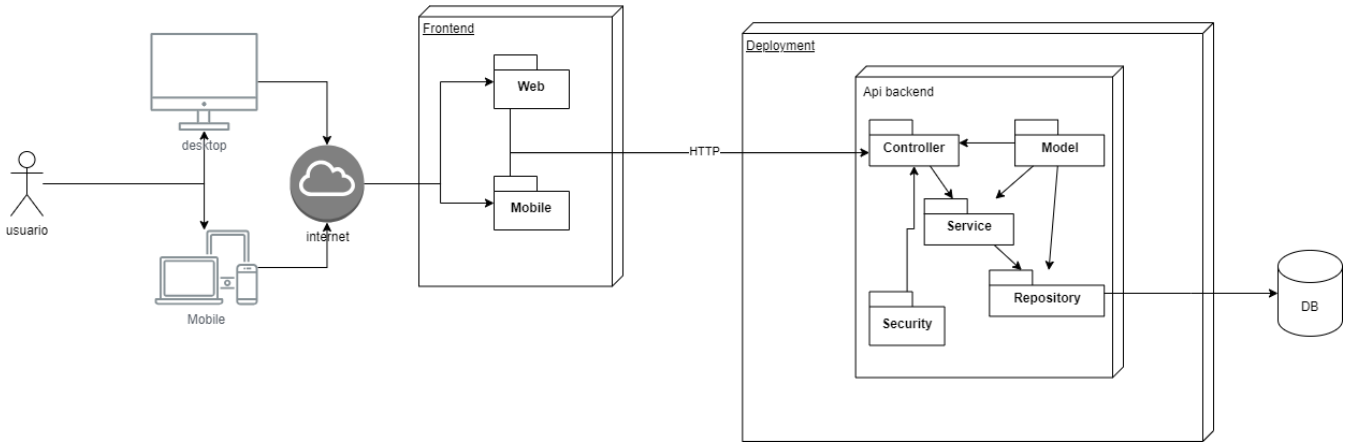
para facilitar a correção de erros e a adição de novas funcionalidades.

2.2.4 Portabilidade

Possibilitar a comunicação da api com várias interfaces do cliente, além de possibilitar também a mudança do banco de dados

2.3 Arquitetura geral

Abaixo tem contém um exemplo de geral do fluxo arquitetural esperado do sistema:



3. Decisões Arquiteturais

[Decisão Arquitetural 001]

Descrição: Foi definido o estilo arquitetural **MVC,** sendo o sistema dividido em uma api para facilitar a comunicação entre mais de um cliente.

Objetivo: Essa divisão foi feita para possibilitar **portabilidade** da api com vários clientes e estilo MVC para diminuir o acoplamento entre elementos internos facilitando a **manutenção**.

Motivação: Separar responsabilidades de front-end e de back-end , além de possibilitar correções de bugs e melhorias mais assertivas.

[Decisão Arquitetural 002]

Descrição: Além do estilo arquitetural MVC também foi definido o estilo de **Camadas** sendo as camadas principais Security, Repository e Service.

Objetivo: O estilo foi adicionado a arquitetura do sistema pois melhora pontos negativos do MVC e melhora os atributos de **Segurança** e **Manutenibilidade**.

Motivação: Melhorar a segurança e facilitar a correção de bugs e adição de melhorias no sistema

[Decisão Arquitetural 003]

Descrição: Na parte do front-end foi definido implementar clientes **web** e **mobile**

Objetivo: A decisão desses dois clientes foi feita para melhorar a **usabilidade** dos usuários que irão acessar o sistema.

Motivação: Melhorar a acessibilidade e facilitar o aprendizado dos usuários

4. Representação da Arquitetura

Para representar as decisões arquiteturais definidas anteriormente, serão utilizados os seguintes pontos de vista definidos:

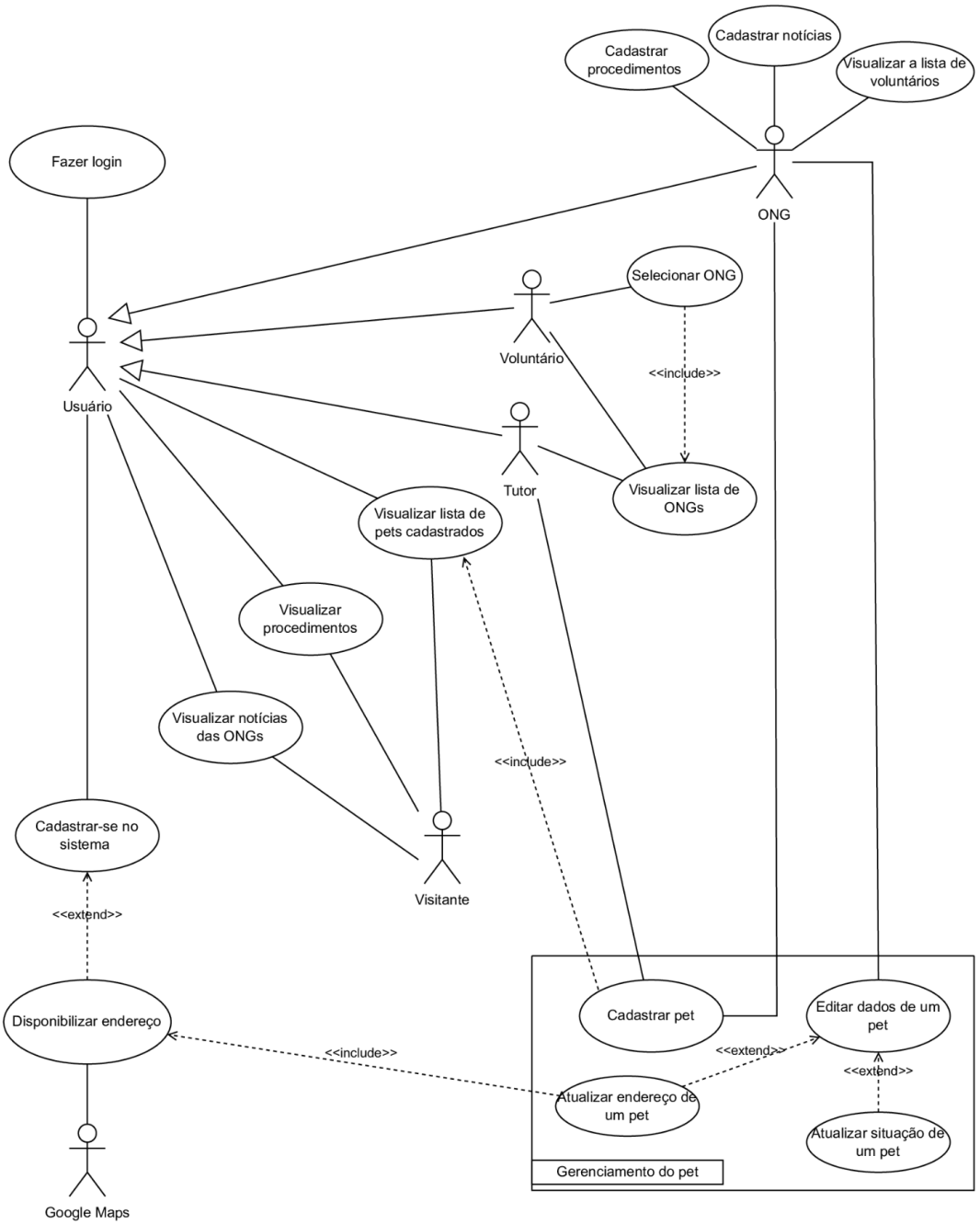
Ponto de vista	Visão	Diagrama(s)
Projetista	Desenvolvimento	Componentes
Desenvolvedor	Lógica	Classes
	Processo	Sequência
Implantador	Física	Implantação
-	Caso de uso	Caso de usos

Os tópicos a seguir detalham esses pontos de vista arquiteturais, bem como as visões e os metamodelos utilizados para representá-los.

5. Ponto de Vista dos Casos de Uso

5.1 Descrição

Para fornecer uma base para o planejamento da arquitetura das visões posteriores e dos demais artefatos que serão gerados no decorrer do ciclo de vida do projeto, foi gerado um diagrama de casos de uso com as funcionalidades que o software deve ter para os seus respectivos atores

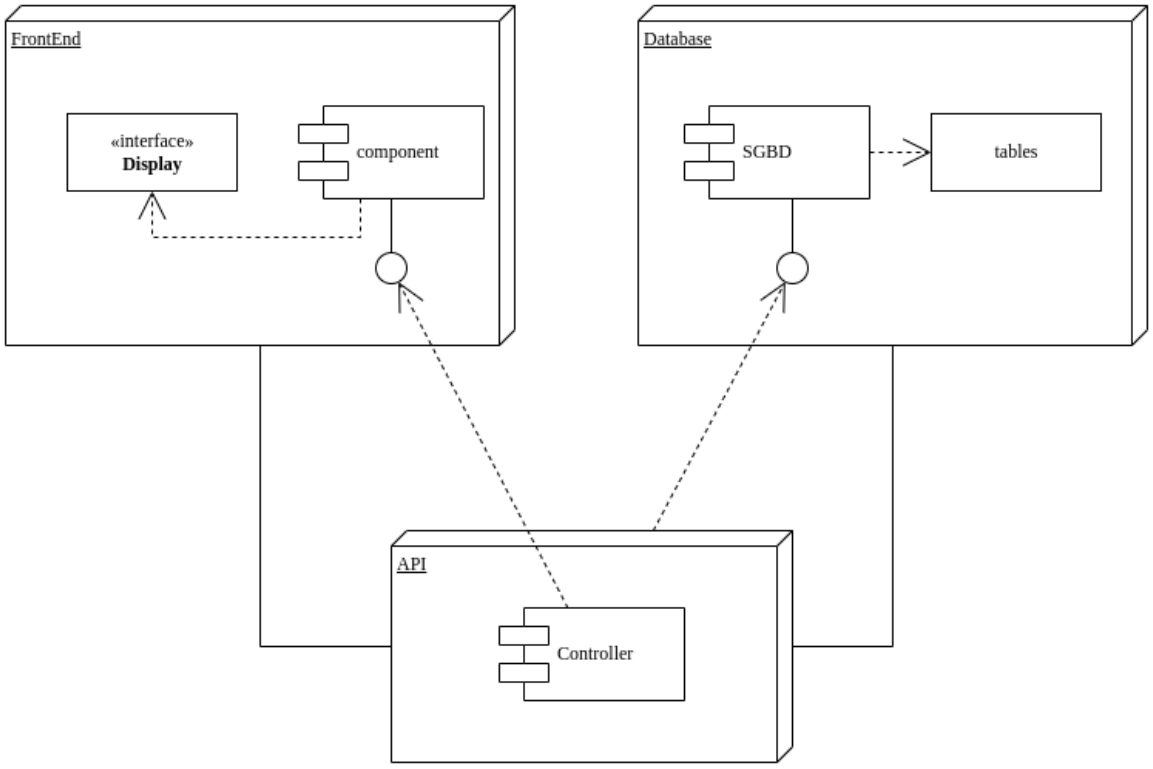

Cadastrar procedimentos
Cadastrar notícias
Visualizar a lista de voluntários
Fazer login
ONG
Selecionar ONG
Voluntário
Usuário
<<include>>
Tutor
Visualizar lista de ONGs
Visualizar lista de pets cadastrados
Visualizar procedimentos
Visualizar notícias das ONGs
<<include>>
Cadastrar-se no sistema
Visitante
<<extend>>
Disponibilizar endereço
Cadastrar pet
Editar dados de um pet
<<include>>
<<extend>>
<<extend>>
Atualizar endereço de um pet
Atualizar situação de um pet
Gerenciamento do pet
Google Maps

6. Ponto de vista físico

6.1 Descrição

A visão Física oferece um entendimento da distribuição física das tecnologias de software em dispositivos de hardware que as armazenam para disposição em ambiente de produção.

Para fornecer suporte ao aplicativo e sistema web, foi escolhido a implantação de uma API em um servidor que lidará com as requisições de um frontend, este também em um servidor de hospedagem online. Para armazenamento e persistência foi decidido pelo uso de um banco de dados sql, como é possível visualizar no diagrama de implantação abaixo.



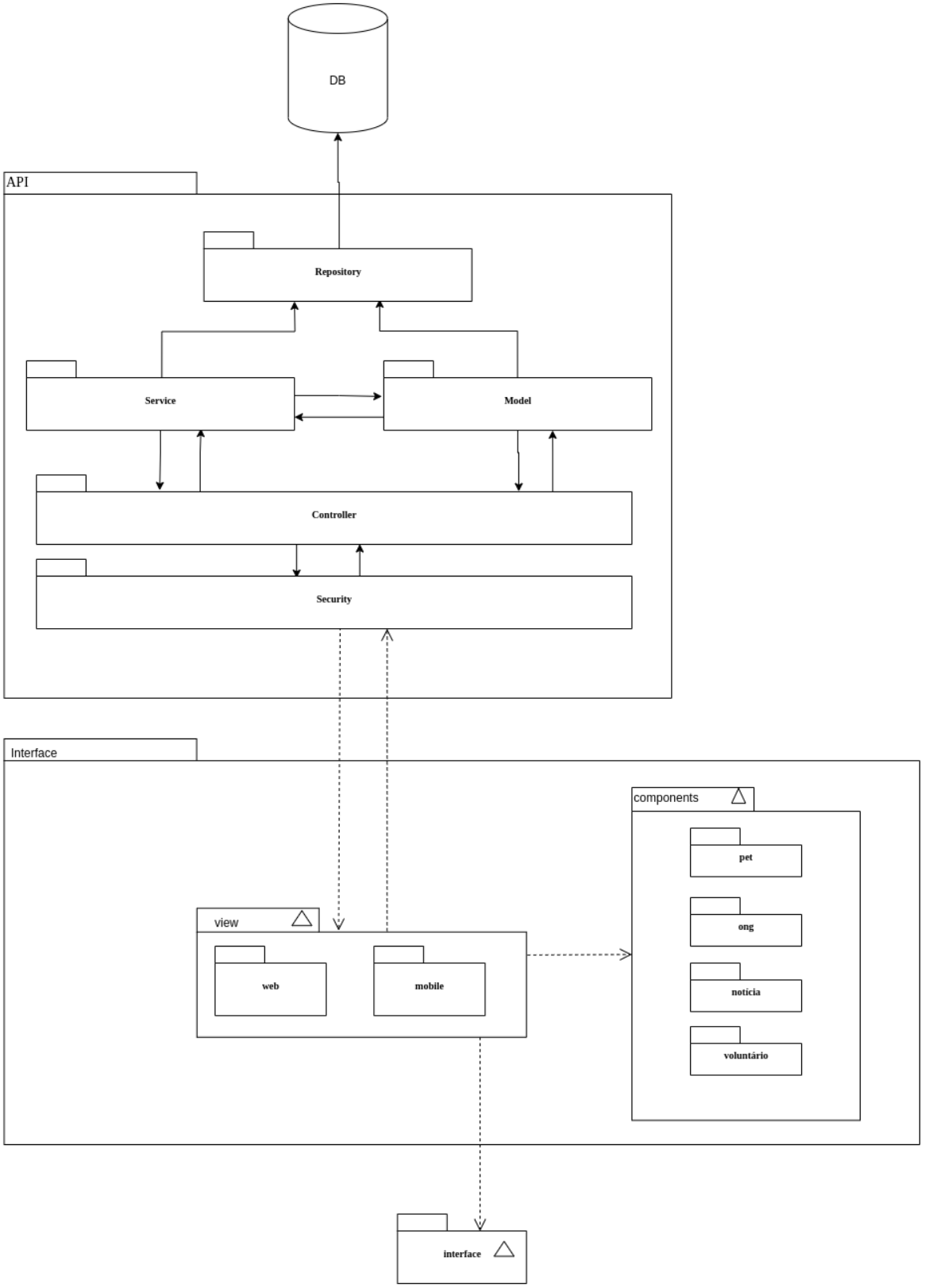
7. Ponto de vista do desenvolvimento

7.1 Descrição

A seguinte visão tem como objetivo especificar ao projetista os módulos do projeto de maneira abstrata, porém com a qual ele possa visualizar os devidos componentes do software e as interações entre eles e o fluxo de execução, facilitando a manutenção.

Foi optado neste projeto pelo uso de uma arquitetura MVC com camadas usando um controlador de requisições e uma camada de Model que fará o processamento dos dados. No diagrama abaixo é possível ver as interações entre os componentes da API e os meios externos de visualização em frontend e persistência de dados.

O diagrama de componentes a seguir tem como objetivo demonstrar o fluxo de execução e interações entre os módulos da aplicação


DB
API
Repository
Service
Model
Controller
Security
Interface
components
pet
ong
notícia
voluntário
view
web
mobile
interface

8. Ponto de vista do Desenvolvedor

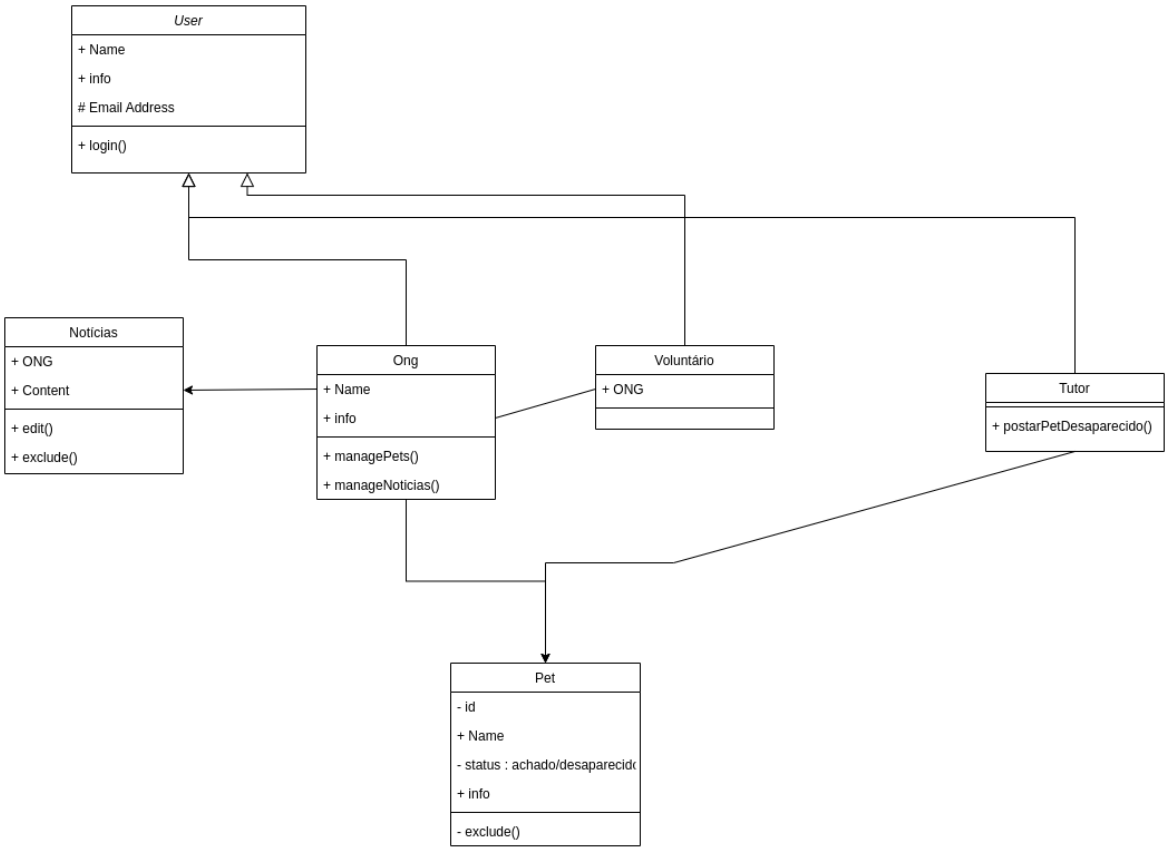
8.1 Visão Geral

O ponto de vista do desenvolvedor é direcionado aos projetistas e desenvolvedores do software e tem como objetivo definir as principais partes responsáveis por definir as funcionalidades e restrições do software, tal como as classes.

8.2 Visão Lógica

Esta visão tem como objetivo identificar as classes e interações entre os componentes lógicos do sistema, funcionalidades principais, regras de negócio e dependências entre eles.

Neste diagrama de classes foi definido como se estrutura as principais entidades do sistema, suas dependências e colaborações.



8.3 Visão do processo

A visão do processo tem como objetivo definir o processo que uma requisição irá percorrer no sistema, desde a ação que o usuário irá realizar até o seu retorno, vale

ressaltar que foi dado enfoque na camada de segurança para entrar em conformidade com o atributo de qualidade **Segurança** priorizado anteriormente.

8.3.1 Detalhamento do diagrama de sequência

O diagrama possui as classes user, view, security, model e repository para descrever a sequência de busca de dados, a classe security recebe a requisição do view(front) esperando um token caso esteja correto a requisição é prosseguida, caso contrário é retornado uma exceção para o usuário.