UNIVERSIDADE DO OESTE DE SANTA CATARINA - UNOESC
ÁREA DAS CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ALINE BAZOTTI
JÉSSICA KEMPER
PATRÍCIA STEFFEN

ADOTE UM AMIGO

São Miguel do Oeste - SC
2019

ALINE BAZOTTI
JÉSSICA KEMPER
PATRÍCIA STEFFEN

ADOTE UM AMIGO

Projeto Final apresentado como requisito parcial nas disciplinas de Engenharia de Software e Banco de Dados II, Curso de Ciência da Computação, Área das Ciências Exatas e Tecnológicas, da Universidade do Oeste de Santa Catarina, Campus de São Miguel do Oeste, Santa Catarina.

Orientadores: Profª. Franciele Petry;
Profº Roberson Junior Fernandes Alves.

São Miguel do Oeste - SC
2019

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho tem como objetivo o desenvolvimento de um site web capaz de intermediar e facilitar a adoção e o apadrinhamento de animais que não possuam uma moradia fixa. Podem tratar-se de animais abandonados, de rua, ou até mesmo animais os quais suas famílias não tenham mais condições de cuidar.

Outra funcionalidade é a possibilidade da arrecadação de donativos para ONGs que realizam trabalhos de cuidado com animais que estão passando por necessidades, sejam os encontrados na rua, sejam aqueles os quais os donos não tenham mais condições de manter. Permite ainda, realizar doações diretamente para os cuidados de animais específicos, os quais sejam postos no site como em situação de necessidade.

2 DESENVOLVIMENTO

Adote um Amigo é um site onde os usuários podem cadastrar animais que estão para adoção, e também encontrar um animal para adotar ou apadrinhar, unindo quem oferece com quem procura, além de possibilitar que sejam realizadas arrecadações de donativos para ONGs e animais necessitados.

O desenvolvimento do projeto deu-se inicialmente pelo levantamento dos requisitos do site, e após a documentação desses requisitos, dividiram-se as tarefas, as quais foram diagramadas no quadro Kanban, que facilitou a visualização das tarefas a serem realizadas, as em fase de teste, e as já finalizadas.

Adote um Amigo conta com uma base de dados para armazenar todos os usuários cadastrados e os animais que estão para serem adotados ou apadrinhados, além de armazenar os dados dos animais que já foram adotados ou apadrinhados. Um diferencial do site é que ele conta com um chat, através do qual o doador e o adotante ou padrinho poderão conversar. Esse chat facilita a comunicação entre as partes envolvidas, fazendo com que o administrador do site não necessite realizar a comunicação entre as duas partes.

2.1 REQUISITOS

A análise de requisitos é o processo de descobrir, analisar, documentar e verificar os serviços requeridos para um sistema e suas restrições operacionais. Esses requisitos podem variar de uma declaração abstrata de alto nível de um serviço ou de uma restrição de sistema.

Através do conceito de análise de requisitos foram verificados e documentados os seguintes requisitos para o projeto Adote um Amigo:

Tabela 1 – Lista de Requisitos

1	O site deve mostrar os animais disponíveis para adoção e apadrinhamento.
2	O site deve ter a opção de filtrar os animais por espécie, raça, localização, sexo, se é castrado.
3	Os detalhes de cada animal devem ser mostrados na mesma página da listagem principal após o usuário clicar para vê-los.
4	O site deve possuir um cadastro de adotante/padrinho com nome, telefone, e-mail, endereço, login e senha.
5	O site deve possuir um botão de "Apadrinhar" na listem do animal caso ele precise de um lar temporário.
6	O site deve possuir um botão de "Adotar" na listem do animal.
7	Para cada animal deve ter uma foto, uma descrição (raça, espécie, idade aproximada, sexo, se é castrado, se possui uma deficiência) e um histórico (por que está para adoção).
8	O site deve disponibilizar um chat para o doador e o adotante/padrinho conversarem caso o botão de "Adotar" ou "Apadrinhar" for pressionado.
9	O site deve listar em outra página os animais já foram adotados.
10	Cada animal deve poder ter os seguintes status de cadastro: para adotar e adotado.
11	O doador deve poder informar que o animal precisa de apadrinhamento.
12	O site deve possuir uma página listando os donativos que estão precisando. Podendo o doador do animal definir o item como urgente.
13	ONGs são prioridades, tudo que elas cadastrarem deve aparecer por primeiro.
14	A ordenação na listagem dos animais deve ser por cadastro mais antigo primeiro.

15	O site deve possuir um cadastro de doador com nome, telefone, e-mail, endereço, login, senha e se é uma ONG.

Fonte: Os Autores (2019).

Esses requisitos são de grande importância para o desenvolvimento do projeto, pois dão estrutura para tudo que será desenvolvido.

2.2 MODELO DE CASO DE USO

O modelo de caso de uso tem como objetivo principal descrever como os usuários interagem com o sistema, facilitando assim a organização dos requisitos. Ele dá uma visão externa do sistema, o qual deve ser capaz de se comunicar com as funcionalidades.

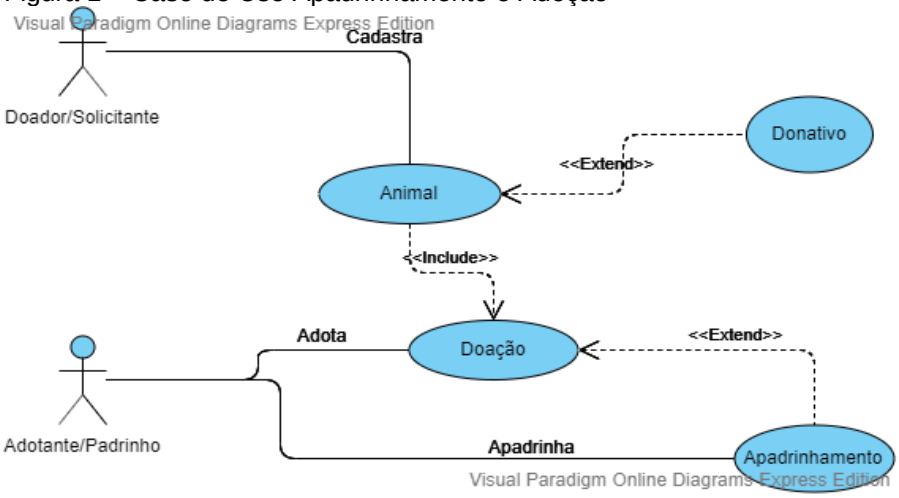
Outra descrição de Casos de Uso para Bornia (2005) é que

> Os casos de uso são os veículos de captura de requisitos e servem de base para a definição de requisitos funcionais. Além de facilitarem a visualização da aplicação, ajudam na delimitação do sistema. Servem como meio de comunicação com usuários finais e clientes por oferecerem uma visão dinâmica e fechada do sistema.

Além do objetivo principal desse tipo de caso, ele também descreve o que o sistema faz, mas não especifica como isso deve ser feito. Cada caso de uso pode ser definido através da descrição narrativa das interações que ocorrem entre os elementos externos e o sistema.

Na figura a seguir podemos visualizar um modelo de caso de uso:

Figura 1 – Caso de Uso Apadrinhamento e Adoção



Fonte: Os autores (2019).

Esse caso de uso descreve como os usuários do Adote um Amigo irão interagir com o sistema. Inicialmente um Doador/Solicitante irá cadastrar um animal, o qual poderá receber donativos ou ser cadastrado para adoção. Esse animal em adoção poderá ser apadrinhado por um Adotante/Padrinho ou ser adotado.

Portanto, pode-se dizer que o caso de uso do Adote um Amigo está demostrando como será feita a adoção e o apadrinhamento dos animais cadastrados, além de demostrar como é realizada a doação de donativos.

2.3 FLUXO DE CASO DE USO

O fluxo de caso de uso conforme Bornia (2005, p.25) “descreve todas as formas como um objetivo pode ser alcançado”. Esse objeto pode ser descrito na forma narrativa ou sob a forma de passos. Durante a descrição diversos pontos de decisão normalmente aparecem. Um ponto de decisão é uma situação onde o sistema ou ator deve decidir por que caminho seguir para atingir um objetivo. Durante a descrição eventualmente exceções podem ocorrer, levando a sequências alternativas que podem retornar a um fluxo que leva ao objetivo desejado ou então terminar sem sucesso.

Já os casos de uso descrevem concretamente o fluxo de eventos utilizando termos relacionados com o projeto do sistema, pode-se definir os seguintes casos de usos para o projeto Adote um Amigo.

Caso 01:

Caso de Uso – Doação/Apadrinhamento de Animal**.**
Objetivo: Colocar animal para adoção/apadrinhamento.
Atores: Site, Doador.
Pré-Condições: Doador estar autenticado.
Condição de Entrada: Doador seleciona a opção de cadastrar animal para adoção.

Fluxo Principal:

1. O sistema mostra um formulário com campos que devem ser preenchidos sobre o animal.

2. O doador preenche os dados do animal no formulário e envia.
3. O sistema verifica se o formulário foi preenchido corretamente. [Fluxo Alternativo 1]
4. O sistema cadastra o animal no banco de dados. [Fluxo Alternativo 2]
5. O sistema informa que o animal foi postado para adoção/apadrinhamento.
6. O sistema encerra o processo.

Fluxos Alternativos:

Fluxo Alternativo 1:

1. O doador não preenche corretamente o formulário
2. O sistema informa o local no formulário que foi preenchido incorretamente
3. O doador digita a informação no campo e envia
4. O processo volta ao passo 3.

Fluxo Alternativo 2:

1. O sistema não consegue cadastrar o animal no banco de dados por algum problema de conexão
2. O sistema informa ao doador que não foi possível cadastrar o animal e solicita que entre em contato com o suporte do site
3. O sistema encerra o processo

Caso 02:

Caso de Uso – Adoção/Apadrinhamento de Animal**.**
Objetivo: Adotar/apadrinhar um animal cadastrado.
Atores: Site, Adotante/Padrinho.
Pré-Condições: Adotante/Padrinho estar autenticado.
Condição de Entrada: Adotante/Padrinho escolhe um animal.

Fluxo Principal:

1. O sistema abre o WhatsApp do doador com o telefone dele [Fluxo Alternativo 1]

2. O Adotante/Padrinho troca mensagens com o Doador informando que tem interesse
3. O Adotante/Padrinho opta por adotar/apadrinhar o animal [Fluxo Alternativo 2]
4. O Doador acessa sua conta no site
5. O Doador acessa o cadastro do animal adotado/apadrinhado
6. O sistema mostra o formulário para edição dos dados do animal
7. O Doador informa no respectivo campo que o animal foi apadrinhado/adotado e salva a alteração
8. O sistema valida os campos [Fluxo Alternativo 3]
9. O sistema salva a atualização do formulário

Fluxos Alternativos:

Fluxo Alternativo 1:

1. O Doador ou o Adotante/Padrinho não possui WhatsApp
2. O sistema mostra as informações de contato do Doador
3. O Adotante/Padrinho entra em contato com o Doador
4. O processo volta ao passo 3

Fluxo Alternativo 2:

1. O Adotante/Padrinho opta por não adotar/apadrinhar o animal
2. O processo se encerra

Fluxo Alternativo 3:

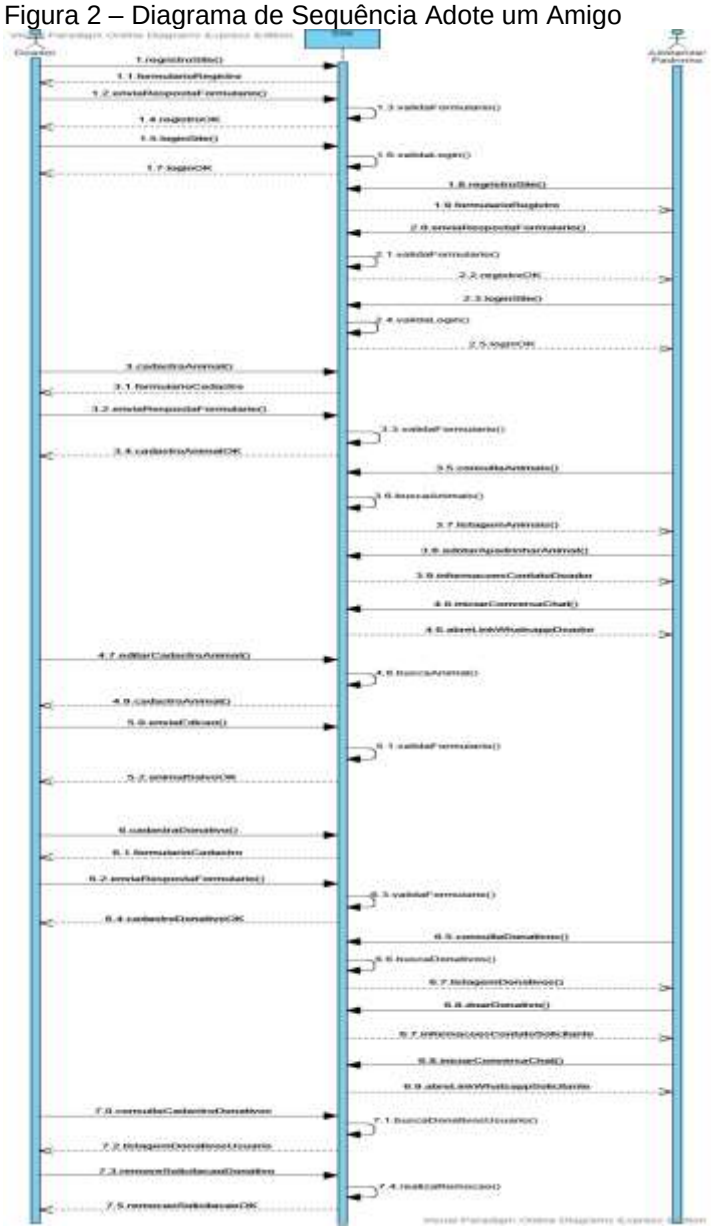
1. O doador não preenche corretamente o formulário
2. O sistema informa o local no formulário que foi preenchido incorretamente
3. O doador digita a informação no campo e envia
4. O processo volta ao passo 8.

2.4 DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência determina a sequência de eventos que ocorrem em um determinado processo, identifica quais métodos devem ser disparados entre os atores, os objetos ou classes envolvidas, e em que ordem. Esse diagrama depende da definição das classes e dos objetos, pois ao ser modelado podem-se identificar falhas e a necessidade de se declarar novos métodos que não haviam ainda sido identificados.

Apresenta-se na Figura 2, o diagrama de sequência do projeto Adote um Amigo com a identificação de seus principais elementos.

Figura 2 – Diagrama de Sequência Adote um Amigo



Fonte: Os Autores (2019).

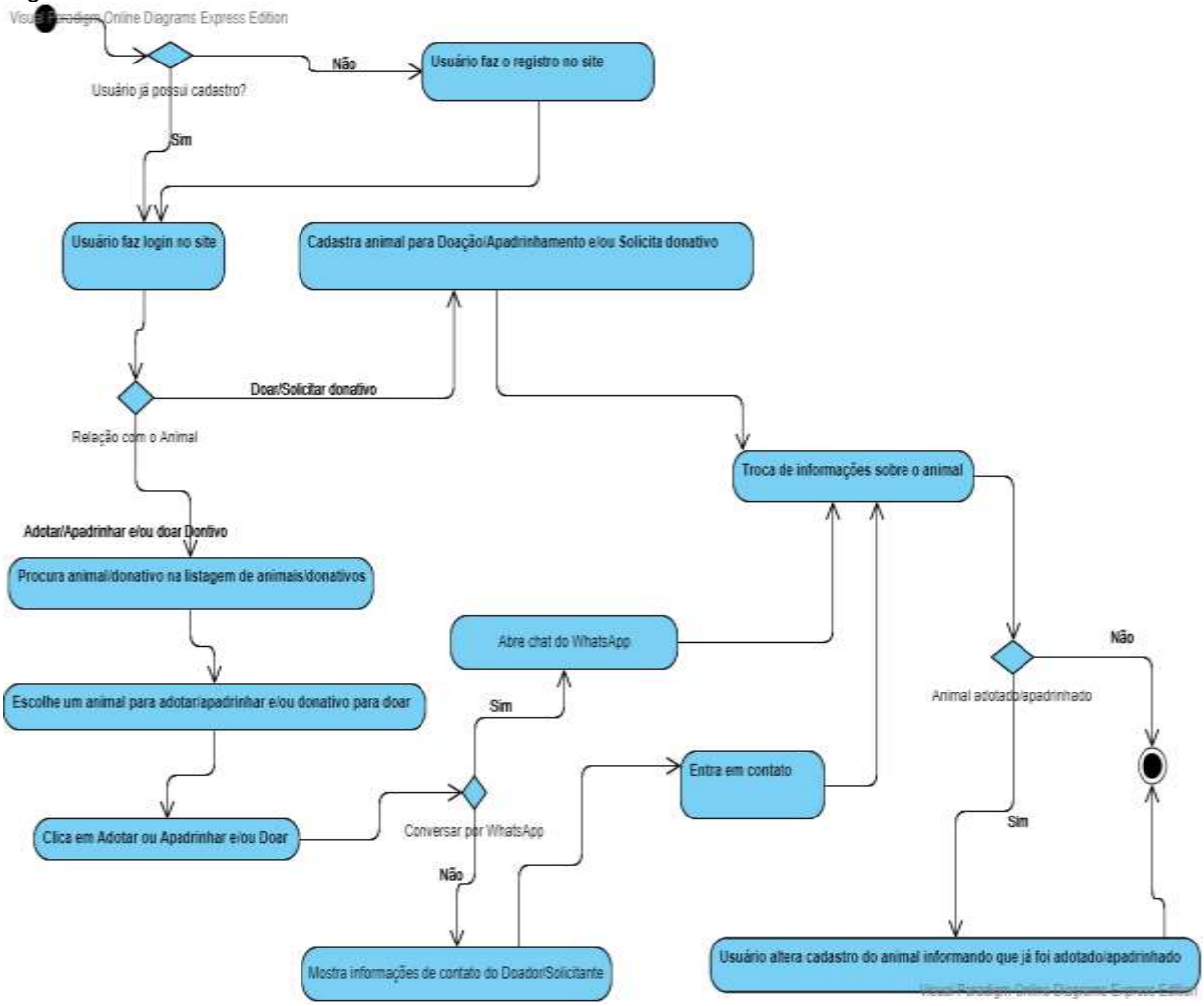
2.5 DIAGRAMA DE ATIVIDADES

“O diagrama de atividades é um diagrama UML utilizado para modelar o aspecto comportamental de processos. É um dos diagramas que mais sofreu mudanças em seu meta-modelo” (GUDWIN, 2000). Neste diagrama, uma atividade é modelada como uma sequência estruturada de ações, controladas potencialmente por nós de decisão e sincronismo.

Portanto, os diagramas de atividades descrevem todos os caminhos possíveis para os casos de usos de um sistema, levando em consideração todas as possibilidades.

Conforme disposto a seguir podemos observar o diagrama de atividade do sistema Adote um Amigo.

Figura 3 – Atividades do usuário



Fonte: Os Autores (2019).

Esse diagrama de atividade retrata todos os caminhos possíveis que os usuários poderão percorrer ao fazer uso do projeto. Ele possui sua inicialização no estado inicial, e após a inicialização da atividade ele verifica se o usuário possui cadastro. Caso não haja cadastro ele retorna para a atividade de cadastramento.

Entretanto, se houver cadastro, o usuário poderá realizar o login e em seguida escolher as opções de cadastrar animais ou até mesmo adotar, apadrinhar e realizar doações. Todas essas opções levarão à atividade de trocar informações sobre o animal, a qual pode dar finalidade a atividade ou alterar o cadastro do animal.

2.6 DIAGRAMA DE ESTADOS

O diagrama de máquina de estados, ou diagrama de estados, demonstra o comportamento de um elemento por meio de um conjunto finito de transições de estado (MONITORIA DE ENGENHARIA DE SOFTWARE, 2016).

Sua construção é recomendada apenas quando existir um certo grau de complexidade referente a transição de estados de um dos objetos envolvidos no processo.

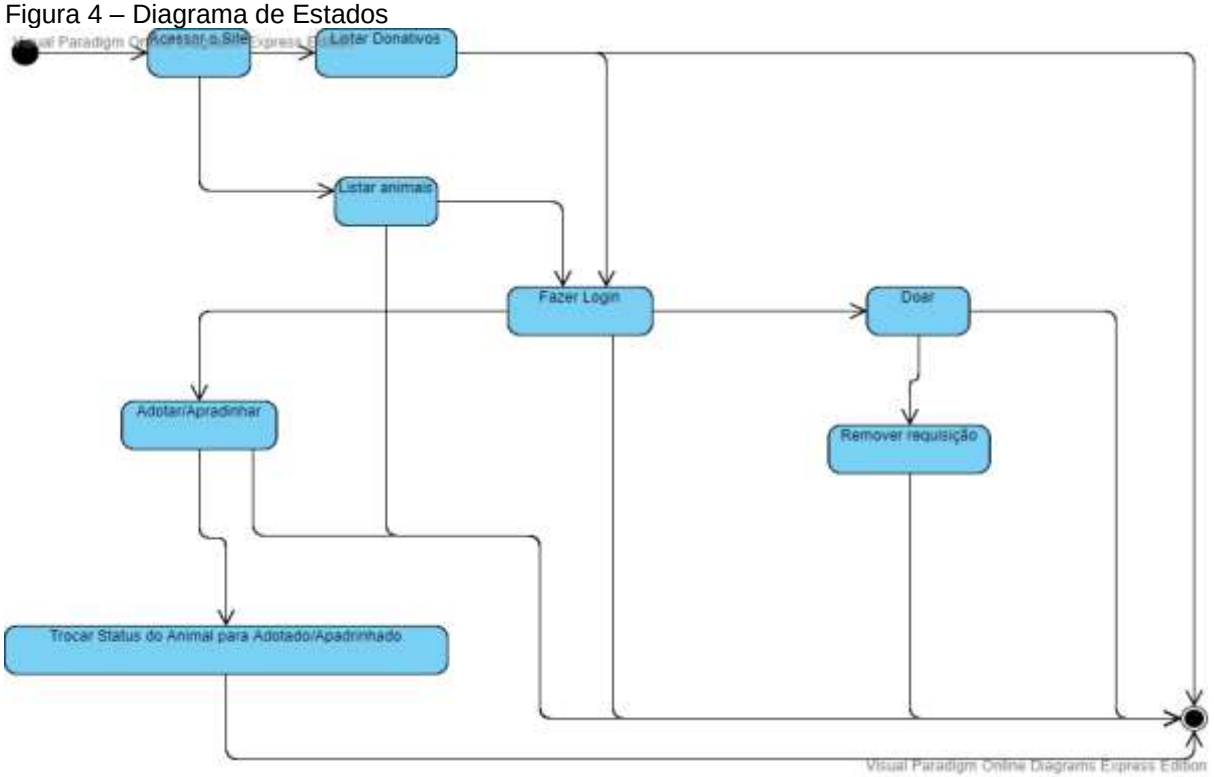
Esse diagrama possui algumas terminologias básicas, sendo elas:

- Evento;
- Estado;
- Transição;
- Objeto independente de estado;
- Objeto dependente de estado.

O evento é uma ocorrência significativa ou digna de nota. O estado, por sua vez, é uma condição de um objeto em determinado momento no tempo. A transição, um relacionamento entre dois estados. O objeto independente de estado, é um objeto que responde sempre da mesma maneira a um evento, e o objeto dependente de estado, um objeto que reage de maneira diferente aos eventos, dependendo do seu estado.

Após o esclarecimento de todas essas terminologias, podemos observar o diagrama de estados da Figura 4.

Figura 4 – Diagrama de Estados



Fonte: Os Autores (2019).

Abstrai-se que esse diagrama demostra todos os estados que o site Adote um Amigo possui. Esses estados possuem condições de um objeto em determinado momento no tempo, que são ligados por transições. Essas transições dão início a outro estado. Portanto, cada estado depende do seu estado anterior para dar sequência.

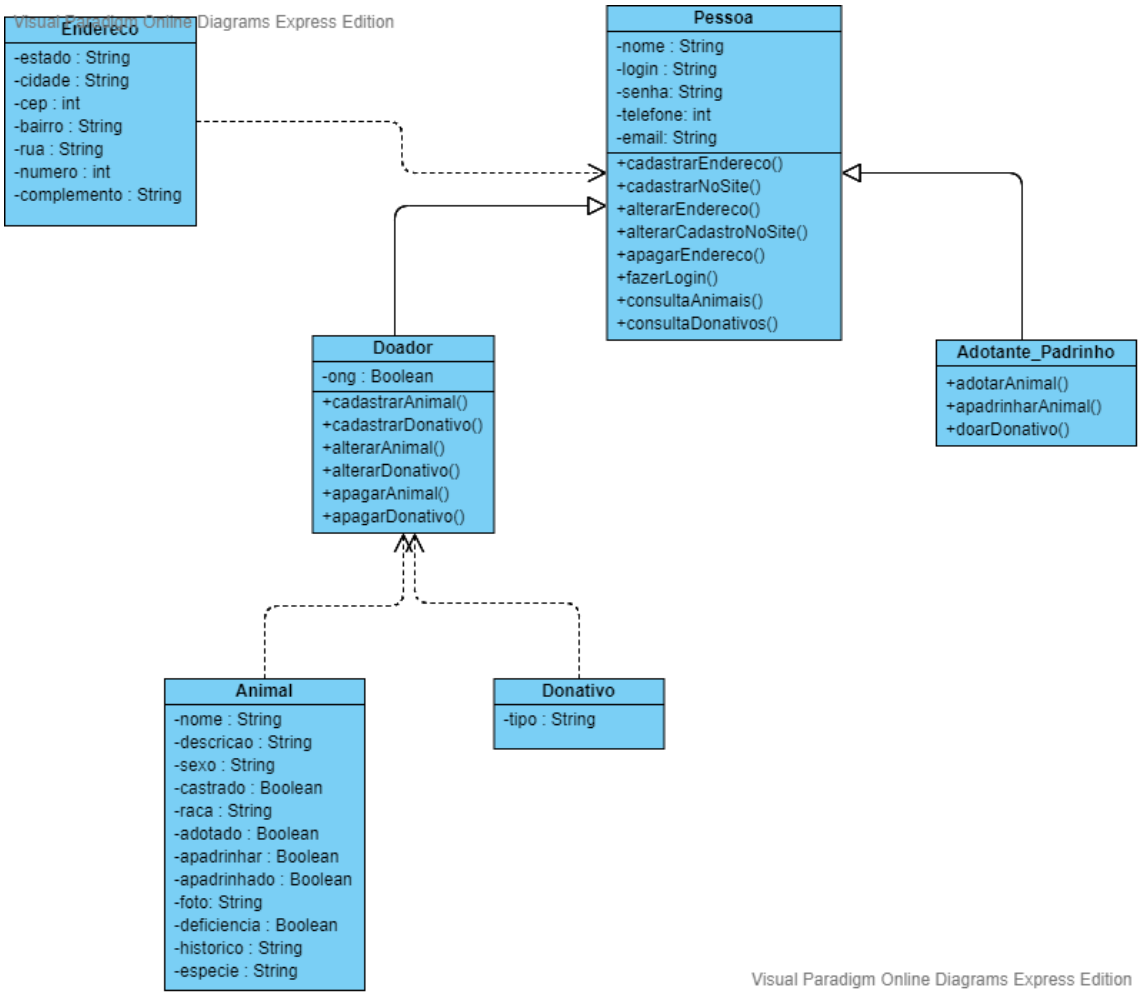
2.7 DIAGRAMA DE CLASSES

Um diagrama de classes é uma representação da estrutura e relações das classes que servem de modelo para objetos. “Podemos afirmar de maneira mais simples que seria um conjunto de objetos com as mesmas características, assim saberemos identificar objetos e agrupá-los, de forma a encontrar suas respectivas classes” (DEVMEDIA, 2016).

Uma classe é uma descrição de um conjunto de objetos que compartilham: atributos, operações, relacionamentos e semântica. Graficamente, uma classe é desenhada como um retângulo, a qual é composta por três partes, sendo elas o nome, os atributos e os métodos.

Pode-se observar essas três partes na figura a seguir:

Figura 5 – Classes do site



Fonte: Os Autores (2019).

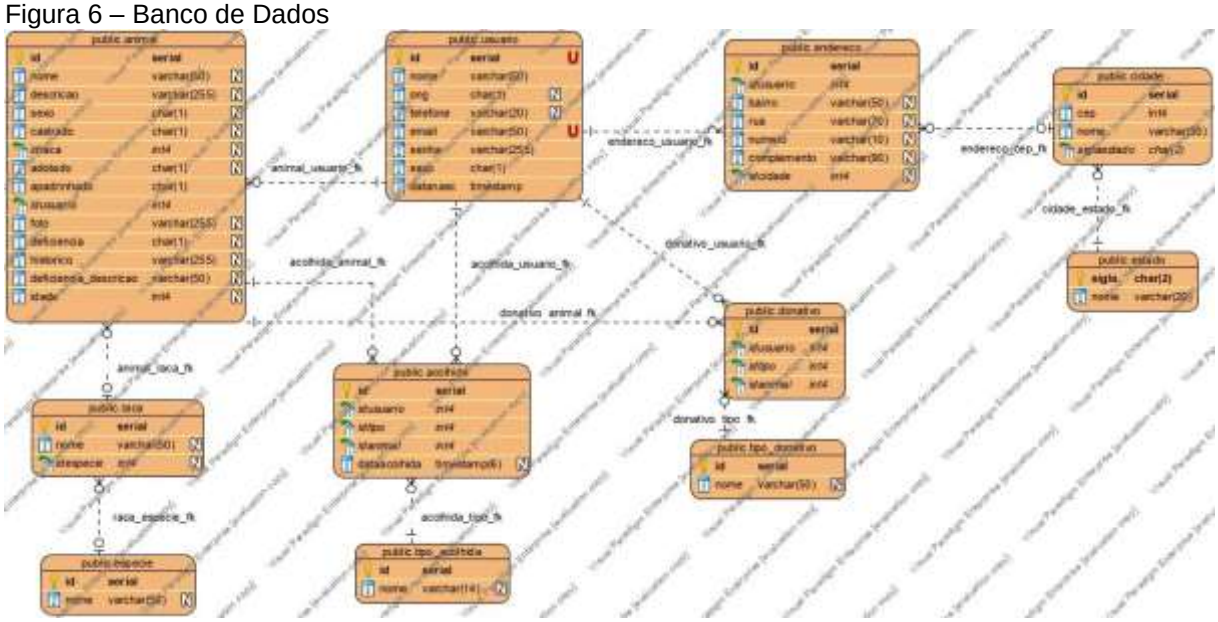
Esse diagrama de classes possui seis classes, sendo elas:

- Endereco;
- Pessoa;
- Doador;
- Adotante_Padrinho;
- Animal;
- Donativo.

Todas essas classes possuem os seus atributos e métodos. Esses atributos e métodos são utilizados para realizar o cadastramento e o login no Adote um Amigo, além de adotar, apadrinhar ou realizar doações para um animal.

2.8 MODELO LÓGICO RELACIONAL

Figura 6 – Banco de Dados



Fonte: Os Autores (2019).

2.9 IMPLEMENTAÇÃO DOS SQL'S

1) Relacione o código, nome e idade de usuários do sexo masculino que não moram em São Miguel do Oeste. Ordene o relatório de forma descendente pela idade dos usuários;

```
create
or replace
view relatorio1 as select
    u.id,
    u.nome,
    u.datanasc
from
    usuario u
join endereco e on
    (u.id = e.idusuario)
join cidade c on
    (e.idcidade = c.id)
where
    u.sexo = 'M'
    and c.nome <> 'São Miguel do Oeste'
order by
          datanasc desc;
```

2) Relacione os animais com código, nome, raça e espécie para animais não castrados e sem deficiência com mais de 2 anos de idade. Ordene o relatório do animal mais velho para o animal mais novo;

```
create
or replace
view relatorio2 as select
   a.id,
   a.nome,
   r.nome as raça,
   e.nome as espécie
from
   animal a
join raca r on
   (a.idraca = r.id)
join especie e on
   (r.idespecie = e.id)
where
   a.castrado = '0'
   and a.deficiencia = '0'
   and a.idade > 2
order by
   a.idade desc;
```

3) Relacione o código do usuário, nome do usuário e quantidade de apadrinhamentos do usuário. Ordene o relatório do usuário com mais apadrinhamentos para o usuário com menos;

```
create
or replace
view relatorio3 as select
    u.id,
    u.nome,
    count(a.id) as apadrinhamento
from
    usuario u
join acolhida a on
    (u.id = a.idusuario)
join tipo_acolhida ta on
    (a.idtipo = ta.id)
where
    ta.id = 2
group by
    u.id,
    u.nome
order by
          count(a.id) desc;
```

3) Relacione o código do usuário, nome do usuário, quantidade de apadrinhamentos, quantidade de adoções, quantidade de donativos para

usuários do sexo feminino, com mais de 30 anos e com mais de 5 apadrinhamentos. Ordene o relatório do usuário mais velho para o usuário mais novo.

```
create
or replace
view relatorio4 as select
	usera.id,
	usera.nome,
	usera.apadrinhamento,
	usera.adocao,
	usera.donativos
from
	(
		select u.id,
		u.nome,
		sum(case when ac.id = 2 then 1 end) as apadrinhamento,
		sum(case when ac.id = 1 then 1 end) as adocao,
		count(d.id) as donativos
	from
		usuario u
	join acolhida a on
		(u.id = a.idusuario)
	join tipo_acolhida ac on
		(a.idtipo = ac.id)
	join donativo d on
		(u.id = d.idusuario)
	where
		u.sexo = 'F'
		and u.datanasc < '1989-01-01'
	group by
		u.id,
		u.nome
	order by
		u.datanasc desc) as usera
where
		usera.apadrinhamento > 5;
```

2.10 SCRIPTS DA BASE DE DADOS

```
create database adoteumamigo;

CREATE table
	acolhida( id int(11) not null comment 'Id da acolhida',
	idusuario int(11) not null comment 'Id do usuário que cadastrou o animal',
	idtipo int(11) not null comment 'Id do tipo de acolhida, adoção ou
apadrinhamento',
	idanimal int(11) not null comment 'id do animal que foi acolhido',
	dataacolhida timestamp null comment 'data que o usuário alterou o cadastro
do animal para adotado ou apadrinhado',
	constraint acolhida_pk primary key(id) ) comment = 'Tabela com as
informações sobre a acolhida do animal, quanto feita';
```

```
create
	table
		animal( id int(11) not null comment 'id do animal',
		nome varchar(50) comment 'Nome do animal, se tiver',
		descricao varchar(255) comment 'Descrição de como é o animal, pelos
claros, escuros....',
		sexo char(1) comment 'Caso o sexo do animal foi indentificado
F - M',
		castrado char(1) default '0' comment 'Caso é identificado se o animal
é castrado
1 - sim
0 - não',
		idraca int(11) comment 'Id da Raça, chave estrangeira, caso tenha',
		adotado char(1) default '0' comment 'Se o animal foi adotado
1- Sim
0 -Não',
		apadrinhado char(1) default '0' not null comment 'Se o animal precisa
ser apadrinhado
1- Sim
0- Não',
		idusuario int(11) not null comment 'Usuário que cadastrou o animal',
		foto varchar(255) comment 'caminho da imagem',
		deficiencia char(1) default '0' comment 'O animal possui necessidades
especiais
1- Sim
0- Não',
		historico varchar(255) comment 'Como o animal foi encontrado',
		deficiencia_descricao varchar(50) comment 'Caso o animal possua
deficiência, qual',
		idade int(11) comment 'Idade do animal, caso saiba',
		constraint animal_pk primary key(id) ) comment = 'Tabela com as
informações do animal que foi cadastrado';

create
	table
		cidade( id int(11) not null comment 'id da cidade',
		cep int(11) not null comment 'cep da cidade',
		nome varchar(50) not null comment 'nome da cidade',
		siglaestado char(2) not null comment 'Sigla do estado, chave
estrangeira',
		constraint cidade_pk primary key(id) ) comment = 'Tabela com as
informações sobre a cidade usada no endereço do usuário';

create
	table
		donativo( id int(11) not null comment 'Id do donativo, chave
primária',
		idusuario int(11) not null comment 'Id do usuário que cadastrou o
pedido de doação',
		idtipo int(11) not null comment 'Id do tipo de donativo, chave
estrangeira',
		idanimal int(11) not null comment 'Id do animal que precisa do
donativo',
		constraint donativo_pk primary key(id) ) comment = 'Tabela com as
informações sobre os donativos solicitados';

create
	table
		endereco( id int(11) not null comment 'id do endereço',
```

```
            idusuario int(11) not null comment 'id do Usuário, chave
estrangeira',
            bairro varchar(50) comment 'nome do bairro',
            rua varchar(70) comment 'Nome da rua',
            numero varchar(10) comment 'número da residência',
            complemento varchar(90) comment 'Complemento do endereço',
            idcidade int(11) comment 'id da Cidade',
            constraint endereco_pk primary key(id) ) comment = 'Tabela com as
informações sobre o endereço do usuário';

create
      table
            especie( id int(11) not null comment 'Id da espécie',
            nome varchar(50) comment 'Nome da espécie, exemplo Cachorro',
            constraint especie_pk primary key(id) ) comment = 'Tabela com as
informações de espécies de animais';

create
      table
            estado( sigla char(2) not null comment 'Sigla do estado, exemplo RJ',
            nome varchar(20) not null comment 'nome do estado, exemplo Rio de
Janeiro',
            constraint estado_pk primary key(sigla) ) comment = 'Tabela com as
informações sobre o estado do endereço do usuário';

create
      table
            raca( id int(11) not null comment 'Id da raça do animal',
            nome varchar(50) comment 'Nome da raça, exemplo pitbull',
            idespecie int(11) comment 'Id da espécie, chave estrangeira',
            constraint raca_pk primary key(id) ) comment = 'Tabela com as
informações de raças';

create
      table
            tipo_acolhida( id int(11) not null comment 'id do tipo de acolhida,
chave estrangeira',
            nome varchar(14) comment 'Nome do tipo: Adoção, Apadrinhamento',
            constraint tipo_acolhida_pk primary key(id) ) comment = 'Adoção
Apadrinhamento';

create
      table
            tipo_donativo( id int(11) not null comment 'Id do tipo de donativo',
            nome varchar(50) comment 'Nome do tipo de donativo, exemplo Ração
Para Gatos',
            constraint tipo_donativo_pk primary key(id) ) comment = 'Tabela com
as informações sobre o tipo de donativo solicitado';

create
      table
            usuario( id int(11) not null comment 'Id do usuário',
            nome varchar(50) not null comment 'Nome do usuário',
            ong char(1) default '0' comment 'Se o usuário em questão pertence à
uma ong
1 - Sim
2 - Não',
            telefone varchar(20) comment 'Telefone do usuário',
```

```
            email varchar(50) not null comment 'e-mail do usuário, usado também
para login',
            senha varchar(255) not null comment 'senha para login',
            sexo char(1) not null comment 'F = Feminino
M = Masculino',
            datanasc timestamp not null comment 'Data de nascimento do usuário',
            constraint usuario_pk primary key(id),
            constraint usuario_un unique(id,
            email),
            constraint usuario_email_key unique(email) ) comment = 'Tabela com as
informações sobre o usuário';

alter table
      acolhida add constraint acolhida_animal_fk foreign key(idanimal) references
animal(id) on
      update
            no action on
            delete
                  no action;

alter table
      acolhida add constraint acolhida_tipo_fk foreign key(idtipo) references
tipo_acolhida(id) on
      update
            no action on
            delete
                  no action;

alter table
      acolhida add constraint acolhida_usuario_fk foreign key(idusuario)
references usuario(id) on
      update
            no action on
            delete
                  no action;

alter table
      animal add constraint animal_raca_fk foreign key(idraca) references raca(id)
on
      update
            no action on
            delete
                  no action;

alter table
      animal add constraint animal_usuario_fk foreign key(idusuario) references
usuario(id) on
      update
            no action on
            delete
                  no action;

alter table
      cidade add constraint cidade_estado_fk foreign key(siglaestado) references
estado(sigla) on
      update
            no action on
            delete
                  no action;
```

```
alter table
	donativo add constraint donativo_animal_fk foreign key(idanimal) references
animal(id) on
	update
		no action on
		delete
			no action;

alter table
	donativo add constraint donativo_tipo_fk foreign key(idtipo) references
tipo_donativo(id) on
	update
		no action on
		delete
			no action;

alter table
	donativo add constraint donativo_usuario_fk foreign key(idusuario)
references usuario(id) on
	update
		no action on
		delete
			no action;

alter table
	endereco add constraint endereco_cep_fk foreign key(idcidade) references
cidade(id) on
	update
		no action on
		delete
			no action;

alter table
	endereco add constraint endereco_usuario_fk foreign key(idusuario)
references usuario(id) on
	update
		no action on
		delete
			no action;

alter table
	raca add constraint raca_especie_fk foreign key(idespecie) references
especie(id) on
	update
		no action on
		delete
			no action;


CREATE INDEX usuario_datanasc_sk ON usuario(datanasc);
```

2.11 CONFIGURAÇÃO DE POLÍTICAS DE ACESSO

```
create user usuario1 with password 'usuario1';

grant select on relatorio1, relatorio2, relatorio3, relatorio4 to usuario1;
```

2.12 GATILHOS DE INTEGRIDADE OU AUDITORIA

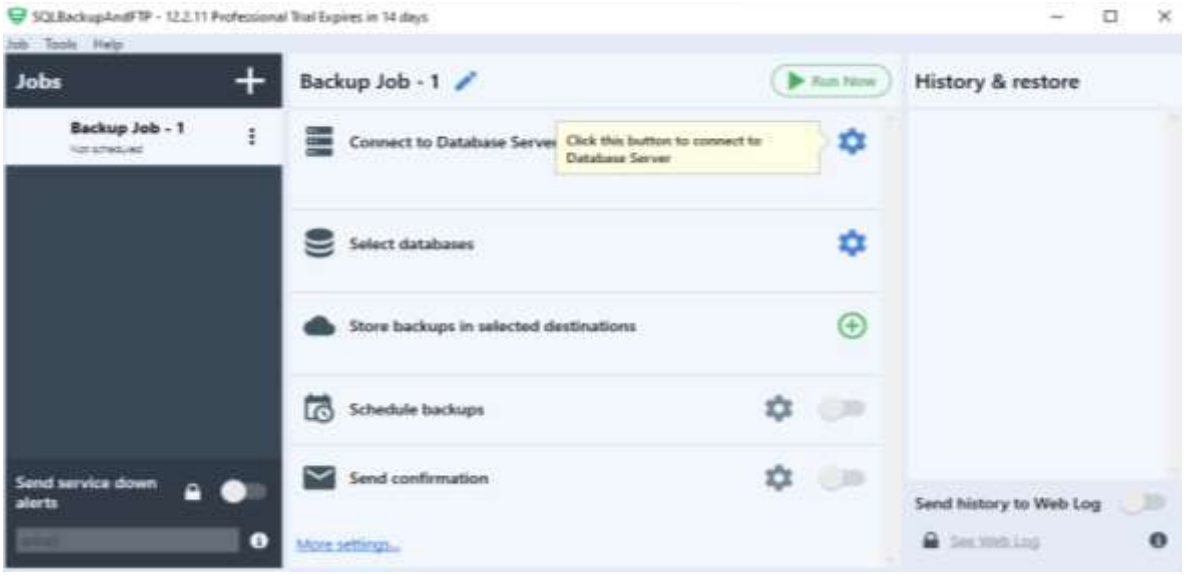
```
create or replace function gatilho()
returns trigger
as
$body$
begin
	if current_user = 'usuario1' then
		raise notice 'usuario1 tentou
		acessar dados da tabela %!', TG_RELNAME;
	end if;
	return null;
end
$body$
language plpgsql;

create trigger usuario1_fora_da_lei
before insert or update or delete
on
acolhida, animal, cidade, donativo, endereco, especie, estado,raca, tipo_acolhida,
tipo_donativo, usuario
   for each row execute procedure gatilho();
```

2.13 CONFIGURAÇÃO E POLÍTICAS DE BACKUP E RESTORE

Backup são cópias de segurança e de dados de um dispositivo de armazenamento a outro para que possam ser restaurados em caso da perda dos dados originais, o que pode envolver apagamentos acidentais ou corrupção de dados. Já o restore é o ato de se fazer uso dos dados armazenados recuperando-os.

Portanto, para realizar o backup e o restore de todos os dados do banco do projeto, utilizou-se o software SQL Backup and FTP. Esse software conta com diversas funcionalidades. Podemos observar algumas delas na figura a seguir:

Figura 7 – Tela inicial SQL Backup and FTP



Fonte: Os Autores (2019).

Essa ferramenta possibilita que seja feito o backup e o restore sem comandos através de códigos. A sua interface é simples e de fácil entendimento. Ela disponibiliza as funcionalidades de escolha da base do banco de dados, além da definição de um horário diário para o backup.

3 CONCLUSÃO

Neste trabalho abordou-se os diagramas e os requisitos do projeto Adote um Amigo. Além de todas as partes referentes ao banco de dados do projeto. Cumpriu-se todos os objetivos que foram documentados, o que ofereceu um grande aprendizado para todos os membros do grupo.

REFERÊNCIAS

BORNIA, Gabriel Silva. **Estruturação de Descrições de Casos de Uso através de Mecanismos de Extensibilidade da UML.** 2005. 212 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2005. Disponível em: <http://ucdesigner.sourceforge.net/dissertacao-2007-09-23.pdf>. Acesso em: 02 dez 2019.

DEVMEDIA (Org.). **Orientações básicas na elaboração de um diagrama de classes.** 2016. Disponível em: <https://www.devmedia.com.br/orientacoes-basicas-na-elaboracao-de-um-diagrama-de-classes/37224>. Acesso em: 04 dez. 2019.

GUDWIN, Ricardo R. **Diagramas de Atividade e Diagramas de Estado.** 2000. 13 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Dca-feec-unicamp, São Paulo, 2000. Disponível em: <http://www.dca.fee.unicamp.br/~gudwin/ftp/ea976/AtEst.pdf>. Acesso em: 03 dez. 2019.

MONITORIA DE ENGENHARIA DE SOFTWARE (Brasil) (Org.). **Diagrama de Máquina de Estados.** 2016. Disponível em: <https://monitoriadeengenhariadesoftware.wordpress.com/2016/06/19/diagrama-de-maquina-de-estados/>. Acesso em: 03 dez. 2019.