INSTITUTO FEDERAL
Bahia
Campus Vitória da Conquista

CLÍNICA VETERINÁRIA - FILA
RELATÓRIO DE DESENVOLVIMENTO

Ariane Rocha dos Santos
Brendo Gomes Prates
Gabriela Carrera Gomes
Ícaro Silva Rodrigues dos Santos
Mariana Moreira Santos

Novembro, 2022
Vitória da Conquista, Bahia

INSTITUTO FEDERAL
Bahia
Campus Vitória da Conquista

CLÍNICA VETERINÁRIA - FILA

Relatório de desenvolvimento de aplicativo apresentado como requisito parcial para obtenção de aprovação na disciplina de Linguagem de Programação II, do Curso de Bacharelado em Sistemas de Informação, do Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Campus Vitória da Conquista.

Prof. Me. Alexandro dos Santos Silva

Ariane Rocha dos Santos

Brendo Gomes Prates

Gabriela Carrera Gomes

Ícaro Silva Rodrigues dos Santos

Mariana Moreira Santos

Novembro, 2022

Vitória da Conquista, Bahia

SUMÁRIO

1. INTRODUÇÃO

A utilização das estruturas de dados no desenvolvimento de software é imprescindível. Diante disso, foi sorteado no ambiente universitário pelo professor Alexandro Silva, ambientes cotidianas os quais estão presentes estruturadas de dados, estudados na disciplina Linguagem de Programação II, com a finalidade de que os discentes desenvolvam um programa de computador utilizando a linguagem de programação Java e os conhecimentos adquiridos ao longo na disciplina.

Diante do pressuposto, este projeto foi desenvolvido com a finalidade de implementação da estrutura de dados do tipo fila para uma aplicação de uma clínica veterinária. Desse modo, o programa dispõe de algumas funcionalidades básicas essenciais para o desenvolvimento de uma fila em uma clínica animal, sendo esses o cadastramento de um novo animal solicitando para tanto o apelido, tipo de animal (canino, felino ou roedor) haja vista que na clínica trabalha-se apenas com esse três tipos de animais, nome do dono, data de nascimento e serviço desejado, que pode ser vacinação, castração, check-up ou atendimento de urgência e emergência, sendo este último serviço considerado prioritário. Além dessas especificações, o programa deverá mostrar o próximo atendimento, às consultas estatísticas e realizar o atendimento tratando adequadamente de distinguir os atendimentos urgentes e normais, a ordem de chegada e gerenciamento das senhas.

2. CONTEXTUALIZAÇÃO

Para o desenvolvimento do programa, foi desenvolvido um diagrama de classe, técnica utilizada para facilitar as exigências solicitadas pelo cliente antes da codificação e para complementar o desenvolvimento da codificação utilizamos o Github, software de versionamento de código e, os aplicativos WhatsApp e Discord, para realizar a comunicação entre os membros envolvidos, onde todos puderam apresentar sua opinião e propor melhorias para o desenvolvimento do projeto. Assim, nas reuniões para discutirmos a respeito do projeto, constatou-se que seria necessário o desenvolvimento das seguintes classes:

Classe ClinicaVet : Possui toda a lógica / regra de negócio das funcionalidades do programa. Nela há 16 métodos:

- cadConsulta: Método responsável por inserir um novo atendimento na fila normal ou prioritária.

- chamaProximoAtendimento: Método responsável por verificar se a fila está vazia e remover o atendimento de uma fila (iniciando da fila prioritária e não havendo nenhum animal na fila prioritária, chamando/removendo da normal).

- getProximoAtendimento, getProximoAtendimentoFilaPrioritaria, e getProximoAtendimentoFilaNormal: Todos os métodos possuem a mesma função de

retornar o próximo atendimento a ser realizado - mas sem remover na fila - (iniciando da fila prioritária e não havendo nenhum animal na fila prioritária, chamando da normal).

- verificarProximoAtendimento: Método responsável por verificar se um animal passado por parâmetro será o próximo da fila a ser atendido.

- relAnimalFilaPrioritaria, relAnimalFilaNormal: Imprime um relatório de quantos animais há em uma fila específica (prioritária ou normal).

- separaAnimalPorFaixa, relAnimalFaixa: Métodos responsáveis por criar a lógica e gerar o relatório que calcula os animais atendidos com base na faixa etária.

- relAnimalEspera: Imprime um relatório de quantos animais há em espera por tipo de serviço.

- relPermanenciaMedia: Imprime um relatório com o tempo médio de espera nas filas.

- animalJaExiste: Verifica se o animal já foi ou não cadastrado com base no seu apelido e nome de seu dono.

- cadastrarAnimal: Realiza o cadastro do animal apenas se o animal não houver sido cadastrado antes.

- getAnimais: Pega a lista de animais cadastrados.

-temAnimais() : Serve para verificar se há animais cadastrados no sistema

Codificamos outra Classe para podermos tratar do Atendimento

- Atendimento: Classe responsável por conter as informações importantes de um atendimento:

- isPrioritario: Método que verifica se um atendimento será ou não prioritário.

É sempre bom reforçar a importância do encapsulamento no desenvolvimento da POO para isso codificamos os métodos:

- Métodos setters:

setSenha, setHoraConsulta, setAnimal: Responsáveis por mostrar os dados de senha, hora em que a consulta foi realizada e o objeto animal.

- Métodos Getters:

- getSenha, getNomeServico, getAnimal, getPermanecia: Responsáveis por pegar os dados de senha, nome do serviço a ser executado, objeto animal e cálculo de quanto tempo o animal ficou na fila.

A solução pensada para trabalharmos com a faixa etária dos animais foi desenvolver uma Classe específica para isso.

- AnimalPorFaixa: Classe do tipo DTO, focada em transferir dados. Além de incrementar os valores inseridos nela.

Como o objeto principal para o nosso projeto são os animais os quais são nossos clientes. Para isso, é necessário implementar uma classe para trabalharmos com eles, assim temos a classe

-Animal: Classe que trata dos atributos e métodos dos animais , ela encontramos o no construtor os atributos apelido, tipoAnimal, dataNascimento e dono, como nosso software é baseado na OO, por conseguinte o encapsulamento é primordial temos os métodos getters e setters dos atributos citados anteriormente e por fim temos o método dados o que retorna um toString melhorado contendo os dados do animal.

Para podermos gerenciar nossa estrutura de dados do tipo Fila (Queue), decidimos desenvolver uma classe para isso a denominamos de Fila

-Fila: Classe responsável por tratar do gerenciamento da Fila de Atendimento, nela encontramos os métodos:

Inserir : Método que é responsável de adicionar um novo cliente na fila de atendimento

ChamarProximo: Método responsável por retirar o última cliente atendido e chamar o novo cliente para o atendimento

VerProximo : Método que espia quem será o próximo a ser atendido

Vazia: Esse método informa se a nossa fila de atendimento está vazia e retorno um valor booleano.

TamFila : Esse método serve para nos informar o tamanho da nossa fila de atendimento.

Iterator: Método para percorrer a Coleção de dados Queue

A última classe implementada foi a Main , que recebe esse nome devido ser a classe onde ocorre a comunicação com o usuário final.

-Main: É onde o programa realmente ocorre de fato, nessa classe é a que capturamos os dados dos usuários para que o nosso software funcione de fato. Nele contém um menu com as opções as quais o usuário deverá ler e utilizar o periférico de entrada teclado, para o armazenamento na memória RAM , uma vez que nessa etapa do projeto ainda não trabalhamos com a implementação de nenhum tipo de banco de dados. Dessa maneira, a medida que o usuário final for informando o que deseja o software deve atendê-lo de forma eficiente sem contextualização ambígua impedindo a compreensão e não gerando exceções de funcionamento resultado no travamento, quebra do sistema.

Segundo os conhecimentos adquiridos durante o curso de sistemas de informação, no Instituto Federal de Educação, Ciência e Tecnologia da Bahia, aprendemos que nenhum software é totalmente seguro, mas seguindo nossos conhecimentos utilizamos as melhores práticas conhecidas em nosso nível de desenvolvimento para satisfazer as necessidades do cliente e vale ressaltar que essa é a primeira versão do software, por conseguinte, ao longo da evolução do projeto podemos repensar melhor e encontrar não só vulnerabilidades como também eventuais melhorias na codificação do aplicativo.

3. IMPLEMENTAÇÃO

É de conhecimento geral que a linguagem de programação Java em seu pacote java.util já disponibiliza a estrutura de dados Queue, que traduzida do inglês significa fila. Assim, sabe-se que todos os métodos básicos necessários para a aplicação são encontrados (adicionar, remover, verificar se está vazia).

Para o desenvolvimento do programa foi necessário a utilização de classes, interfaces e coleções já disponíveis no pacote java.util. As linhas abaixo demonstram as explicações da utilização de cada uma delas e as tabelas com a apresentação da aplicação no projeto.

No que se refere a Classe denominada como Fila, primeiramente fazermos o uso de coleções, para o correto funcionamento de um programa é indispensável , sendo assim é necessário realizar a importação da Classe Queue, desta maneira adicionarmos essa na Classe Fila, a qual é responsável por trabalhar no gerenciamento das filas no programa , o código abaixo exemplifica a importação da classe Queue e a utilização no software.

```
1 import java.util.Queue;
2 public class Fila{
3 private int NumeroFila;
4 private char TipoFila;
5 private Queue<Atendimento> Fila;
```

Tabela 1: Exemplo de importação e utilização da Coleção Queue

Encontramos na primeira linha da tabela a importação da Classe Queue, que é encontrada no pacote java util da biblioteca java, assim já podemos fazer o uso desta classe que conta com os métodos Enqueue (Método responsável de adicionar um elemento na Fila),Queue (Método que retira o primeiro item da fila e retorna uma referência), Peek (Método que mostra o primeiro elemento na fila). Já na linha 5ª(quinta linha) da tabela , ocorre a utilização da Classe Queue que é responsável pelo enfileiramento , neste procedimento é trabalhando usando a lógica FIFO (First in First Out), nota -se que entre os sinais matemáticos de maior que e menor que <> , temos a classe Atendimento, pois nela que iremos trabalhar a estrutura de dados fila.

Para o desenvolvimento da fila contamos também com a utilização da Interface LinkedList (double linked list), foi escolhida essa interface devido a sua performance entre os métodos add, remove, get e set, comparado com o uso do ArrayList. A tabela a seguir demonstra a importação e a utilização desta interface.

```
1 import java.util.Queue;
2 import java.util.LinkedList;
```

```
3  public class Fila{
4  private int NumeroFila;
5  private char TipoFila;
6  private Queue<Atendimento> Fila;
7   public Fila(char tipoFila) {
8  NumeroFila = 0;
9  TipoFila = tipoFila;
10 Fila = new LinkedList<Atendimento>();
```

Tabela 2: Exemplo de importação e utilização da Interface LinkedList

Na segunda linha de código encontra- se a importação da Interface LinkedList disponível no pacote java.util da biblioteca do Java. Note-se que na linha 10 do exemplo de código acima, temos a instanciação de um novo atendimento utilizando a Interface LinkedList e esse objeto é atribuído a Fila de atendimento.

Por fim, para realizar a interação na Coleção, foi utilizado o Objeto Iterator, servindo com um loop em uma coleção, esse objeto contribui não só para a facilitação de compreensão no código como também é uma forma mais elegante de percorrer uma coleção de objetos. Para fazermos o seu uso é necessário primeiramente realizar a importação. A tabela 3, a seguir, mostra esse processo.

```
1  import java.util.Queue;
2  import java.util.LinkedList;
3  import java.util.Iterator;
4  public class Fila{
5  …
```

Tabela 3: Exemplo de importação do Objeto Iterator

Na primeira linha de código, encontramos a importação do Objeto Iterator, assim como a Classe Queue e a Interface LinkedList também está disponível no pacote java.util da api da linguagem Java.Para percorrermos a coleção utilizamos o Interator, a tabela abaixo, exemplifica seu uso na classe Fila do projeto.

```
45  …
46 public Iterator<Atendimento> getFilaIterator() {
47  return Fila.iterator();
48 }
```

Tabela 4: Exemplo de utilização do Objeto Iterator

Outra classe que fazermos para o desenvolvimento do programa foi a Classe ClinicaVet, nela achamos mais prático e ágil utilizar o atalho da api Java , sendo esse digitar o nome da biblioteca Java.util e adicionar um asterisco (*), invés do nome da classe , Objeto ou Interface, Listas , etc desejada, pois dessa maneira podemos fazer o uso de tudo que está

presente na api sem se preocupar com com a importação. A tabela a seguir mostra como é feita essa utilização.

```
1  import java.util.*;
```

Tabela 5: Exemplo de importação de tudo presente no pacote java.util

Todavia, para simplificar a compreensão da codificação, citamos a seguir as utilizações que fizemos na Classe ClinicaVet. Vale ressaltar que, é imprescindível a utilização das classes desenvolvidas por nós ao longo da codificação, então para gerenciarmos essas classes precisamos de alguma estrutura de armazenamento, desta forma, fazermos o uso da interface List ,utilizada para atuar na manipulação dos elementos internos e suas posições ocupadas.Todavia, como se trata de interface é obrigatório garantir a flexibilidade da sua aplicação, garantir que o mesmo objeto possa ser instanciado de maneiras distintas em pontos distintos da aplicação, por isso fazemos o uso da Classe ArrayList . A tabela abaixo apresenta a importação da interface e da Classe no programa

```
6   private List<Atendimento> atendidos;
7   private List<Animal> animais;
8    public ClinicaVet() {
9   filaNormal = new Fila('N');
10   filaPrioritaria = new Fila('P');
11    atendidos = new ArrayList<Atendimento>();
12     animais = new ArrayList<Animal>();
13   }
```

Tabela 6: Exemplo de uso da Interface List e da Classe ArrayList

Nas linhas 6 e 7 encontra-se a criação das listas de recebimento das Classes Atendimento e Animal para fazermos o gerenciamento dos animais atendidos e dos animais cadastrados. E nas linhas 11 e 12 encontra-se a instanciação das Classes utilizando a Classe ArrayList da Api da linguagem de programação Java.

E para percorremos as listas, fazermos o uso do Objeto Iterator E iterator()método pode ser usado para obter um Iterador para qualquer coleção, e do método hasNext usado para verificar se há algum elemento restante na lista. A tabela abaixo, mostra a utilização desses métodos na codificação.

```
77 public AnimalPorFaixa separaAnimalPorFaixa(Iterator<Atendimento>
78 atendimentoIterator){
79  Date agora = new Date(System.currentTimeMillis());
80   int crianca = 0, adulto = 0, idoso = 0;
81    while (atendimentoIterator.hasNext()){
82 …
```

Tabela 7: Exemplo do trecho de utilização dos métodos hasNext e iterator

Na linha 77, vemos a utilização do Objeto Iterator para podermos interagir na Coleção e nas linha 81 na estrutura de repetição while encontramos os métodos iterator e hasNext para percorrer na Coleção citada anteriormente.

É essencial a utilização das datas no desenvolvimento de software, seja para ser capturado no sistema operacional utilizado pelo usuário ou seja para que o usuário do sistema forneça dados como data de nascimento, ano, dia e mês. Diante do exposto , para o nosso sistema da clínica veterinária não foi diferente, fizemos o uso da classe Date da linguagem de programação Java, não só para capturar a data de nascimento dos animais como também para detectarmos a hora decorrente do sistema operacional do usuário. Devido seu uso ser primordial utilizamos Classe Date do pacote java util não apenas em uma única classe mas sim em sua grande maioria, assim em nosso software encontra-se a importação nas Classes : Animal , ClinicaVet ,Atendimento e Main.

Como a importação ocorre da mesma forma, na tabela abaixo temos a importação da Classe Date na Classe Animal

```
1  import java.util.Date;
```

Tabela 8: Exemplo da importação da Classe Date da Api do Java

Utilizamos também essa classe para capturar a data atual do sistema, a tabela a seguir mostra um exemplo de utilização na Classe Atendimento.

```
21     HoraEntrada = new Date(System.currentTimeMillis());
22    }
```

Tabela 9: Exemplo da capturada da hora do sistema Operacional

Desta forma capturamos a hora do sistema operacional do usuário em milissegundos. Como é de conhecimento de geral , as linguagens de programação são desenvolvidas na linguagem inglesa e por conseguinte sabe-se que o formato de dado da língua inglesa é diferente do nosso idioma , portanto precisamos tratar corretamente essa diferença de padrão. Assim, fizemos o uso das Classes DateFormat e SimpleDateFormat, para podermos realizar a

formatação das Datas . A tabela, na sequência, mostra como é feita a importação e como foi utilizado o código nessas Classes.

```
1  import java.util.Date;
2  import java.text.DateFormat;
3  import java.text.SimpleDateFormat;
```

Tabela 10: Exemplo da importação das Classes DateFormat da Api do Java

Encontramos na linhas de código 3 e 4 as importações das Classes citadas anteriormente, sendo a primeira utilizada formatar e analisar datas de maneira sensível ao local e a segunda , presente na linha 3 é usada para ajudar você a formatar e analisar datas para qualquer localidade. Normalmente, sua utilização ocorre no o auxílio de uma variável do tipo String, devido ser realizada uma conversão . A tabela , a seguir, mostra o modo o qual foi utilizado no programa , na Classe Main

```
66  String data = entrada.nextLine();
67  Date dataNascimento = null;
68  try {
69  dataNascimento = new SimpleDateFormat("dd/MM/yyyy").parse(data);
```

Tabela 11: Exemplo da utilização da Classe DateFormat

Na linha 68, nota -se que na na variável dataNascimento atribuímos o valor da variável data que é do tipo String como podemos visualizar na linha de código 66 e utilizamos o parse para solucionar esse problema com as datas.

Ademais, sabemos que em um sistema de computação erros e exceções são comuns durante o desenvolvimento, por isso, é interessante analisarmos com cuidado e providenciarmos cuidar o mais rápido possível destes erros, antes que o sistema fique no ar e o sistema quebre na mão do usuário final. Por isso, utilizamos a importação da Classe ParseException, para tratar de possíveis erros na data. A tabela abaixo mostra a sua importação para a Classe Main.

```
1    import java.text.ParseException;
```

Tabela 12: Exemplo da Classe ParseException da Api do Java

A sua utilização ocorre no método main , informando que poderá ocorrer uma exceção. A tabela abaixo mostra sua utilização.

```
11  public static void main(String[] args) throws Exception {
12    entrada = new Scanner(System.in);
13    clinica = new ClinicaVet();
14    boolean sair = false;
15    do {
16      try{
17     …
```

Tabela 13: Exemplo da utilização da Classe import java.text.ParseException

Note que na linha 11 ocorre o uso da palavra reservada throws que é utilizada quando falamos de exceções, em seguida temos a palavra Exception que se refere ao tipo de exceção que falamos.

Para concluir, sabe-se que o software é desenvolvido para pessoas e para isso necessitamos da interação do usuário final , podemos realizar a comunicação com o cliente de diversas formas mas a escolhida foi através da classe Scanner. Na tabela abaixo, temos o exemplo de importação dessa Classe , utilizada na nossa Classe denominada como Main .

```
4   …
5  import java.util.Scanner;
6  entrada = new Scanner(System.in);
```

Tabela 14: Exemplo da importação da Classe Scanner

Observe que, na linha de código 5 , ocorre a importação da Classe Scanner que está no pacote java util da Api do java. Para realizar seu uso, veja na linha de código 6 , é realizada a instanciação que será atributo na variável entrada. Nesta classe podemos capturar os dados digitados pelo usuário pelo periférico de entrada teclado. Na tabela a seguir, encontramos sua utilização.

```
18  System.out.println("********************* MENU **********************");
19   System.out.println("* O que deseja fazer?                    *");
20   System.out.println("* 1. Cadastrar novo animal               *");
21   System.out.println("* 2. Agendar Atendimento                 *");
22   System.out.println("* 3. Realizar Atendimento                *");
23   System.out.println("* 4. Consultar proximo Atendimento       *");
24   System.out.println("* 5. Consultar proximo Atendimento por Fila    *");
25   System.out.println("* 6. Consultas de caráter estatístico    *");
26   System.out.println("* 0. Encerrar programa                   *");
27    System.out.println("************************************************");
28    int op = entrada.nextInt();
29    entrada.nextLine();
```

Tabela 15: Exemplo de utilização da Classe Scanner

Na linha 28, encontramos uma variável chamada op, a qual armazenará um número inteiro, que será digitado pelo usuário e na linha posterior temos a codificação para capturar apenas o enter do teclado, resolvendo a problemática da leitura de um número inteiro e uma String na linguagem de programação Java.

Portanto, com o uso das Classes, Interfaces presentes no pacote Java. util podemos realizar o desenvolvimento do código fonte da Clínica Veterinária, para saber mais detalhes disponibilizamos o código fonte na plataforma de hospedagem de código fonte github , o qual poderá ser acessado através do do link https://github.com/Gabidiela/ClinicaVeterinaria/tree/master , e lá também encontra o diagrama de Classe utilizado para o desenvolvimento do programa.

4. ORIENTAÇÕES DE USO

Esta seção é dedicada à apresentação de orientações direcionadas aos utilizadores do aplicativo desenvolvido, seguindo o passo a passo a seguir será possível de maneira simples e fácil o manuseio do aplicativo.

Primeiro passo:

Ao iniciar o programa aparecerá um menu com diversas funções onde o usuário deverá inserir um número entre (0 a 6), cada número corresponde a uma funcionalidade diferente do programa, fica a critério do usuário que tipo de serviço ele irá querer utilizar, veja a seguir o menu inicial do aplicativo.

```
********** MENU **********
* O que deseja fazer?                          *
* 1. Cadastrar novo animal                     *
* 2. Agendar Atendimento                       *
* 3. Realizar Atendimento                      *
* 4. Consultar proximo Atendimento             *
* 5. Consultar proximo Atendimento por Fila    *
* 6. Consultas de car□ter estat□stico          *
* 0. Encerrar programa                         *
************************************************
```

Ao digitarmos (1):

Essa função trata do cadastramento do animal no sistema, onde para fazer o cadastro do animal será pedido primeiramente o (Apelido) nome do animal, em seguida será pedido o tipo que o animal em questão pertence 1) para canino 2) para felino 3) para roedor, depois o nome do dono a quem o animal pertence, e por fim a data de nascimento do animal, feito isso o cadastro do animal será feito com sucesso.

```
1
Informe os dados do animal
Apelido: Bob
Tipo do animal:
1) canino
2) felino
3) roedor
2
Dono: Marcos
Data de nascimento (dd/mm/aaaa): 11/09/2018
Animal cadastrado com sucesso
```

Ao digitarmos (2):

Essa função serve para agendar o atendimento do animal, onde irá primeiramente aparecer o tipo de serviço a ser agendado, o usuário deve digitar um número de (1 a 5) o número 5) serve caso o usuário deseje retornar ao menu inicial, os números de (1 a 3) são para serviços comuns realizados na clínica, 4) para agendar um serviço de (urgência/emergência) de um animal, assim um animal com um agendamento feito por essa função, será colocado em uma fila de prioridade, assim recebendo atendimento primeiro que os que foram registrados em um atendimento comum, em seguida irá aparecer uma lista com todos os animais cadastrados no sistema, e deve ser selecionado o valor numérico correspondente ao animal a qual deseja ser agendado o atendimento.

```
2
Selecione o servi□o
1. Vacinar Animal
2. Castrar Animal
3. Check-up
4. Atendimento de Urg□ncia/Emerg□ncia
5. Voltar
3
Selecione o Animal que para realizar o antendimento:
0) Bob | Marcos
0
Cadastro bem sucedido!
```

Ao digitarmos (3):

Essa função chama o próximo animal da fila para que possa ser realizado o atendimento.

```
3
Por favor, compare□a ao consult□rio o pr□ximo a ser atendido:
Senha: N.1
animal: Bob
Dono: Marcos
Tempo de perman□ncia na fila: 15.845 minutos
Atendimento realizado com sucesso
```

Ao digitarmos (4):

Essa função serve para verificar se um animal específico será o próximo a ser chamado, após selecionarmos essa opção será pedido o nome do animal, em seguida o nome do dono, lembrando que, o animal em questão deve já ter sido cadastrado anteriormente para que possa ser possível ser realizado a consulta.

```
4
Insira o apelido do animal:
bob
Insira o nome do dono:
Marcos
Este animal n□o □ o proximo a ser chamado
```

Ao digitarmos (5):

Essa função é um pouco parecida com a função anterior só que, ao invés de nos mostrar se um animal específico é o próximo a ser atendido, ele nos mostrar qual é o próximo animal a ser atendido de uma fila específica, onde o usuário deve escolher entre 1) para mostrar qual é o próximo animal a ser atendido da fila normal, e 2) para mostrar qual é o próximo animal a ser atendido da fila prioritária.

```
5
Selecione a fila:
1) Normal
2) Prioritaria
1
Pr□ximo animal a ser atendido:
Senha: N.1
O apelido do Animal □:Bob
o tipo do animal □:felino
a data do nascimento □:Tue Sep 11 00:00:00 BRT 2018
O dono do animal □:Marcos
Tempo de permanência na fila: 15.383383333333333 minutos
```

Ao digitarmos (6):

Essa função trata de características estatísticas, onde o usuário terá que introduzir um valor numérico de (1 a 4) para será exibida informações que vão auxiliá lo, sendo elas 1) para mostrar a quantidade de animais presentes em cada filha, 2) a quantidade total de animais por faixa etária, 3) para ser exibida a quantidade de animais que estão em espera de atendimento, e 4) para mostrar o tempo médio de espera na fila.

```
6
**************************************************
* Selecione a op□□o desejada                     *
* 1. quantidade de animais em cada fila          *
* 2. quantidade total de animais por faixa et□ria *
* 3. quantidade total de animais em espera       *
* 4. tempo m□dio de permanência na fila          *
**************************************************
4
O tempo m□dio de permanência na fila foi de: 15.845
```

Ao digitarmos (0):

Fazendo isso o programa será encerrado.

Esperamos que com essa orientação de uso, seja possível utilizar sem problemas o aplicativo, e espero que tenham gostado, o programa foi feito de maneira muito clara e facilitada para que qualquer pessoa possa utilizá-lo, com essa orientação creio que as futuras funcionalidades do programa ou mudanças serão de fácil compreensão do usuário.

5. CONCLUSÕES

Através do desenvolvimento deste trabalho chegamos a conclusão que para o desenvolvimento de um excelente software devemos realizar diversas pesquisas em diversas fontes na web verificando se estas são confiáveis, realizar a leitura da documentação da linguagem a qual escolhemos para o desenvolvimento do código é primordial para encontrar possíveis soluções. Ademais, compreendemos que a comunicação não só com os membros da equipe é um fator importante para eficácia no desenvolvimento como também buscar o monitor da disciplina e pessoas mais experientes para auxiliar nessa missão.

Outro aspecto importante para ser analisado é o processo das fases de desenvolvimento, haja vista que para a melhor compreensão das solicitações do cliente o processo de diagrama de classe, leitura em diversos momentos do dia das solicitações auxilia na facilitação de raciocínio lógico para melhores soluções dos problemas previstos.

Em uma última análise, para o melhoramento desse projeto para futuros discentes, solicitamos ao professor a trabalhar mais em sala ou nas listas,provas e outras trabalhos o diagrama de classe para melhor compreensão do projeto, devido não ter sido muito explorado neste semestre tivemos muita dificuldade em realizar a primeira parte do projeto , haja vista ainda não sermos familiarizados com nenhum padrão de projeto , isso não quer dizer que iríamos produzir de maneira adequada até porque ainda não temos o nível de conhecimento necessário mas serviria também como uma base mais rica de conhecimento.

Por fim, agradecemos ao nosso mestre profº Alessandro, por oferecer essa oportunidade de desenvolver um projeto praticamente real que ampliou nossa mente para visualizar problemas cotidianos os quais vemos todos os dias e não nos auto avaliamos capazes de realizar devido a inexperiência.