CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE MOGI MIRIM

Alair Alves Junior
Daniela Sayuri Iwata
Hegon Roberto Fernandes Rosa
Pedro Henrique Dalben de Morais

ALIMENTADOR INTELIGENTE PARA CÃES

MOGI MIRIM
2019

Alair Alves Junior
Daniela Sayuri Iwata
Hegon Roberto Fernandes Rosa
Pedro Henrique Dalben de Morais

ALIMENTADOR INTELIGENTE PARA CÃES

Trabalho de Graduação apresentado ao Curso de Tecnologia em Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Mogi Mirim como pré-requisito para a obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Rita De Cássia Catini

Mogi Mirim
2019

RESUMO

O cão sempre foi amigo do homem, desde a idade das cavernas. Hoje é considerado um ente familiar e em decorrência disso possui muitas regalias. Para suprir essas regalias surgiram os pet shops, cujo número é crescente, isso em decorrência do fenômeno da humanização dos animais de estimação. Devido ao aumento do número de cães, de pet shops e da tecnologia, foi proposta a criação de um protótipo de alimentador automático para cachorros e um aplicativo para controlar o protótipo. A criação do protótipo teve como objetivo auxiliar na vida dos donos de cachorros, que frequentemente deixam de viajar em virtude de terem que alimentar seus animais. O protótipo desenvolvido utiliza linguagem de programação e controlador lógico programável para automatizar o alimentador, possuindo um reservatório de ração sólida e seu comedouro.

Palavras-chave: alimentador automático, cachorros e ração.

ABSTRACT

The dog has always been man's friend since the age of the caves. Today it is considered a family entity and as a result has many perks. In order to supply these perks, the pet shops, whose number is increasing, came about due to the phenomenon of the humanization of pets. Due to the increase in the number of dogs, pet shops and technology, it was proposed the creation of a prototype automatic dog feeder and an application to control the prototype. The creation of the prototype was aimed to helping the lives of dog owners, who often stop traveling because they have to feed their animals. The developed prototype uses programming language and programmable logic controller to automate the feeder, having a solid feed tank and feeder.

Keywords: automatic feeder, dogs and ration.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELA

LISTAS DE ABREVIATURAS E SIGLAS

CSS – Cascading Style Sheets

HTML – Hypertext Markup Language

IDE – Integrated Development Environment

IOT – Internet of Things

IP – Internet Protocol

NoSQL – Not Only SQL

PO – Product Owner

RDBMS – Relational Database Management System

SQL – Structure Query Language

UML – Unified Modeling Language

USB – Universal Serial Bus

SUMÁRIO

1. INTRODUÇÃO

Há muito tempo, os ancestrais selvagens dos cães domesticados entraram em nosso convívio e até hoje mantém uma amizade indissolúvel com os Homens. A humanidade compartilha com eles uma história extraordinária de como esses animais complementam cada aspecto de nossas vidas, seja lutando conosco, nos protegendo, trabalhando ao nosso lado e até para entender como nós sentimos. Os primeiros laços entre cães e humanos foram forjados quando a humanidade era composta por sociedades de caçadores coletores e muito tempo antes de domesticar qualquer outro animal, eles já ajudavam os homens durante a caça. Uma parceria que combinou a velocidade e a agilidade dos animais com a inteligência humana. Atuando juntos, tanto homem como o cão são mais bem-sucedidos e todos podiam comer. Assim os cães trouxeram uma reviravolta nessa etapa da evolução.

Carinhosamente sendo até mesmo chamado de melhor amigo do homem, os cães são animais domésticos muito populares. Centenas de raças de cães vivem na Terra e não há como negar que eles cativaram a população do planeta. As pessoas mimam seus animais e, muitas vezes, os tratam como membros da própria família. Em 2017, no Brasil, foram gastos U$ 20 bilhões no mercado Pet, segundo levantamento feito pela ABINPET (Associação Brasileira da Indústria de Produtos para Animais de Estimação).

No passado, os cães deixaram a vida selvagem tornando-se nossos aliados e, dessa forma, conquistaram o sentimento humano criando um fascínio das pessoas por seus animais de estimação. Contudo, o atual cenário de convívio entre cães e humanos tem sofrido com o comportamento das pessoas que desfrutam cada vez menos de tempo livre e de lazer se afogando em um oceano de compromissos “inadiáveis”. Nesse ambiente relacionado ao tempo onde rapidez significa eficácia, a tecnologia tem um importante papel de auxílio com as ferramentas de automação que substituem, provisoriamente, a presença humana.

Dessa forma, a automação foi usada para contribuir com um sistema automático de controle. Fazendo uso de microcontroladores capazes de processar e armazenar informações, utiliza uma linguagem de programação para a execução de instruções e orientação do mecanismo programado. Uma alternativa para implementar a automação é o Arduino. Uma plataforma de computação que ajuda

em projetos de eletrônica e programação com funcionalidade, praticidade e acessórios complementares.

Visto isso, o Alimentador Inteligente para cães tem como propósito facilitar essa ação mecânica de tutores que não podem, mas desejam acompanhar as refeições do animal e precisam deixar a ração disponível para alimentação.

Composto por compartimentos dedicados, o alimentador conterá sensores para identificar, quando necessário, a reposição de ração. Na divisão superior serão depositados os grãos de ração, onde um sensor fará a medição do nível verificando a quantidade de alimento calculada para o consumo diário. Atraídos por gravidade, os grãos deverão seguir de cima para baixo até um extrator onde serão depositados para que o animal tenha contato com o alimento. Para isso um sensor detectará a presença do cachorro identificando seu porte e liberando a ração apropriada.

Assim, o sistema oferecerá a possibilidade de trato automático liberando mecanicamente porções de ração em horários estabelecidos previamente e o envio de mensagens através de aplicativo para que o tutor possa acompanhar a alimentação.

Partindo da hipótese de que o agente causador dos problemas é a dificuldade de associar a rotina das pessoas com as necessidades alimentares de cães de estimação, acredita-se que o uso do alimentador inteligente deve auxiliar amplamente no processo de alimentação dos animais e, ao mesmo tempo, aproximar os tutores de seus cães durante um momento tão importante para ambos. O uso da tecnologia de automação e das linguagens de programação como elo fundamental de uma era nova, minimizando as distâncias, aumentando a qualidade de vida e otimizando o tempo, cada vez mais precioso na sociedade moderna.

2. METODOLOGIA DE PESQUISA CIENTÍFICA

Neste capítulo será exposto os conceitos de metodologia de pesquisa científica e os instrumentos de coletas de dados.

Prodanov e Freitas (2013) afirma que a pesquisa científica pretende explorar cientificamente um ou mais aspectos de um determinado assunto, com isso deve ser sistemática, metódica e crítica. E para a vida acadêmica, a pesquisa é um exercício de investigação para o trabalho e problemas sugeridos pelos orientadores.

Segundo Lakatos e Marconi (2003), todas as ciências têm característica pela utilização de métodos científicos, mas nem todos os métodos são ciência. Esses métodos permitem alcançar os objetivos traçando o caminho a ser seguido.

Conforme Gil (2002), a pesquisa tem objetivo de obter respostas aos problemas que são propostos, é desenvolvido na qual envolve inúmeras fases, desde a formulação do problema até a apresentação dos resultados. Para o desenvolvimento da pesquisa é através dos conhecimentos disponíveis e utilização cuidadosa de métodos, técnicas e outros procedimentos científicos. Para esquematizar uma pesquisa é preciso a formulação do problema, construção de hipóteses, determinação do plano, elaboração dos instrumentos de coleta de dados, operacionalização das variáveis, pré-teste dos instrumentos, seleção da amostra, coleta de dados, análise e interpretação dos dados e pôr fim a redação do relatório da pesquisa.

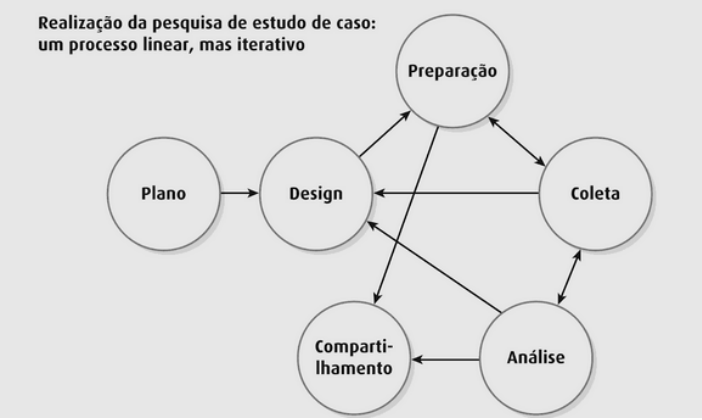
Para Prodanov e Freitas (2013), existem vários tipos de pesquisa, do ponto de vista dos procedimentos técnicos são elas:

- Pesquisa bibliográfica – é elaborada a partir de material já publicado, é importante atentar ao obter os dados para ver se é de confiabilidade já que coloca o pesquisador em contato direto com material escrito sobre o assunto da pesquisa. Toda pesquisa necessita de um referencial teórico então os outros tipos de pesquisa também envolve o estudo bibliográfico.
- Pesquisa documental – é baseada em materiais que não receberam um tratamento analítico ou que podem ser reelaborados de acordo com os objetivos da pesquisa.

- Pesquisa experimental – consiste em determinar um objeto de estudo, é onde define as formas de controle e observar os efeitos que a variável vai produzir no objeto, ou seja, o pesquisador refaz as condições de um fato a ser estudado.
- Pesquisa de campo – tem o objetivo de conseguir informações e conhecimentos através de um problema, assim procurando uma resposta ou hipótese para comprovar.
- Estudo de caso – com a ideia de coletar e analisar informações sobre um determinado indivíduo, uma família, um grupo ou uma comunidade de pessoas. Esquematizado na figura 1 o processo de desenvolvimento do estudo de caso.

> Consiste no estudo profundo e exaustivo de um ou poucos objetos, de maneira que permita seu amplo e detalhado conhecimento, tarefa praticamente impossível mediante outros delineamentos já considerados (GIL, 2002).

Figura 1 – Processo para o desenvolvimento da pesquisa de estudo de caso



Fonte: Yin (2015).

Para o projeto Tratador inteligente será utilizado o método de pesquisa bibliográfica e estudo de caso pois estudará algo específico e que irá proporcionar um avanço de conhecimento.

2.1. Instrumento de coleta

Prodanov e Freitas (2013) defende que o planejamento é uma ferramenta essencial para quem queira desenvolver uma pesquisa cientifica essa fase tem o objetivo de obter o máximo de informações sobre a realidade também será definido

o tipo de pesquisa, a população, as amostras que serão recolhidas quais serão os instrumentos de coleta de dados e a forma que iremos organizar e analisar os dados.

Segundo Prodanov e Freitas (2013) os instrumentos de coletas mais tradicionais são:

- Observação: a observação e utilizada para obter dados em determinados aspectos da realidade. Alguns exemplos de observação é: observação assistemática, observação sistemática, observação participante, observação não participante etc.
- Questionário: Visa fazer o levantamento primário de dados e a importância é a descrição verbal dos informantes. Tem como proposito elaborar uma sequência de questões que serão formuladas igualmente para todos informantes.

> O questionário deve ser objetivo, limitado em extensão e estar acompanhado de instruções que expliquem a natureza da pesquisa e ressaltem a importância e a necessidade das respostas, a fim de motivar o informante (PRODANOV E FREITAS, 2013).

- Entrevista: Realizada face a face pode ser usada ou não um roteiro de perguntas, tem como proposito a obtenção de informações de um entrevistado sobre um determinado assunto.

Para a entrevista é necessário que o entrevistador faça um plano para que não falte informações que deveriam ser colhidas. No Trabalho atual será utilizado entrevista e questionário como ferramenta de coleta de dados.

2.2. Cronograma

O alimentador inteligente para cães tem o propósito de realizar ação mecânica para trato dos animais com redução da ação de tutores que não podem, mas desejam acompanhar as refeições do animal, diariamente.

Composto por compartimentos específicos o projeto conterá sensores para monitorar os níveis de ração, microcontroladores para a quantidade de alimento calculada para o consumo diário e sensor que detectará a presença do cachorro identificando seu porte e liberando a quantia de ração apropriada.

Assim, o sistema oferecerá a possibilidade de trato automático liberando mecanicamente alimentos em horários estabelecidos e enviando mensagens através

de aplicativo para que o tutor possa acompanhar a alimentação. Abaixo, na tabela 1, segue o cronograma do desenvolvimento deste trabalho.

Tabela 1 - Cronograma das atividades desenvolvidas para o TG1

Meses	Desenvolvimento	Responsável
Fevereiro	Formação do grupo, escolha do tema	Alair/ Daniela/ Hegon/ Pedro
Março	Formulação e realizado a entrevista e questionário	Alair/ Daniela/ Hegon/ Pedro
Abril	Definição dos tópicos do TG	Alair/ Daniela/ Hegon/ Pedro
Maio	Desenvolvimento da estrutura dos tópicos do TG e pesquisa dos tópicos propostos	Alair/ Daniela/ Hegon/ Pedro
Junho	Entrega do trabalho e qualificação	Alair/ Daniela/ Hegon/ Pedro

Fonte: Elaborada pelos autores (2019)

3. REVISÃO BIBLIOGRÁFICA

Este trabalho apresenta a implementação de um protótipo de um alimentador automático para cães composto por compartimentos específicos e sensores de níveis. Utilizando microcontroladores e linguagem de programação, o sistema oferecerá a possibilidade de trato automático liberando mecanicamente porções de ração de maneira controlada e programada. Através deste protótipo será possível, determinar a quantidade de alimento, agendar horários para refeições, registrar e informar os tutores sobre a rotina de alimentação de seu animal.

Neste capítulo apresentará também uma explicação de como um sistema automático de controle pode ajudar através do uso de mecanismos que verificam seu próprio funcionamento, efetuam medições e introduzem correções através da automação.

3.1. Automação

A Automação é a substituição do trabalho humano ou de um animal por uma máquina, é a operação de uma máquina ou de um sistema automático ou através de um controle remoto com a mínima interferência do operador humano, é o controle de processos automáticos. Automático significa ter um mecanismo de atuação própria, que faça uma ação requerida em tempo determinado ou em resposta a certas condições como, por exemplo: a máquina de lavar para uma dona de casa, um robô para uma indústria ou um caixa eletrônico para uma pessoa comum fazer retirada de dinheiro (RIBEIRO, 1999).

Quando se fala em automação não se pode esquecer de falar de Automação Residencial (AR). Muratori e Dal Bó (2013) citam que por cerca de uns dez anos a automação residencial era visto por muitas pessoas como casa de luxo e com o passar do tempo isso foi se tornando cultura à pessoa adotar a tecnologia e usar a automação em prol de benéficos próprios e que muito daqueles conceitos de automação visto um dia como futurístico por muitos, hoje usa com muita naturalidade por muitas famílias e esses sistemas tornaram acessíveis.

3.1.1. Tipos de automação

Com o passar dos anos a automação foi se desenvolvendo entre diversos seguimentos fazendo com que as indústrias, prédios e até mesmo residências adotassem o uso dessa tecnologia. Fazendo com que a máquina trabalhe de forma autônoma, essa tecnologia traz conforto e o que antes era visto como luxo para alguns hoje se tornou algo essencial no dia a dia. A seguir apresenta-se alguns tipos de automações.

- **Automação Industrial**

A Automação Industrial é a chegada das máquinas na linha de produção de uma indústria seria usar a potência elétrica ou mecânica para acionar algum tipo de máquina e acrescentando a máquina uma inteligência artificial para que seja executada de modo mais rápido e eficiente e suas principais vantagens são econômicas e também a sua segurança. Citando Ribeiro (1999), com a chegada da automação foi à evolução nas indústrias já que foram substituídos empregados por máquinas fazendo com que diminua o quadro de funcionários e também as vantagens que as indústrias receberam com a automação já que como são máquinas que trabalham na linha de produção os maiores benefícios foram que uma máquina nunca reclama e também não faz greve e além de não pedi aumento de salário e não precisar de férias, mais como toda máquina possui suas limitações e seria uma limitada capacidade de tomar decisões, sempre terá que ser feito ajuste em suas operações e fazer sempre calibração periódica para garantir o melhor desempenho de exatidão e além da manutenção eventual para assegurar sua precisão e que a máquina não se degrade.

- **Automação Fixa**

A Automação Fixa é baseada na fabricação de um determinado produto quando deseja um elevado volume de produção. O maior exemplo são as indústrias automotivas, que realizam operações de usinagem em componentes de motores, da transmissão e nas diferentes peças que constituem a mecânica automotiva. Por serem equipamentos em geral, muito caros, e sua alta função de produtividade assim

os custos unitários resultantes são relativamente baixos se comparados com outros métodos de produção (PAZOS, 2002).

- **Automação Programável**

Citando Groover (2011), este tipo de automação foi projetado com a capacidade de modificar a qualquer momento qualquer comando ou operação que será executada por um programa e assim todas as alterações feitas será interpretada pelo sistema além de poder adicionar novos programas para criações de novos produtos facilitando a flexibilidade para variação e alteração nas configurações. Muito utilizado na produção de baixo e médio lotes, a cada novo lote de um produto diferente o sistema deverá ser reprogramado esse procedimento leva muito tempo até ser feito outra programação e até mesmo trocas de ferramentas como, por exemplo, máquinas de ferramentas numericamente controlada e robôs industriais e controladores lógicos programável.

- **Automação Flexível**

Afirma Groover (2011), que a automação flexível e uma extensão e uma melhoria da automação programável, pois ela e capaz de produzir uma grande variedade de peças ou produtos e com suas modificações de modelos de peças para outro sem perca de tempo. Com as modificações e reajuste do sistema e de suas configurações físicas sendo ferramentas, acessórios e configurações da máquina o sistema poderá assim produzir diferentes variações e planos de peças sem exigir que eles sejam produzidos em lotes, a diferença será entre as peças processadas pelo sistema não são significativas então o volume de alterações exigidas entre modelos é mínima facilitando a produção continua de um conjunto variado de produto além da flexibilidade para lidar com alterações e variações no projeto do produto.

- **Automação Residencial**

A Automação Residencial é um ramo da automação predial que se especializa no controle doméstico gerencia sistemas e obtém controle em equipamentos eletrônicos e eletromecânicos e reduz à necessidade de intervenção humana, ela coleta informações do ambiente por meio de sensores e assim analisa e tomam

decisões segundo um programa especifico e sendo assim essas decisões deverá ter uma ação tendo controle total do equipamento. (BOLZANI, 2010).

Para o tratador inteligente será feito o uso de automação residencial juntamente com sensores o tornando inteligente. Isso faz com que o usuário consiga ter o acesso ao equipamento de onde estiver através de uma página web. Ele será instalado em residências trazendo um conforto ao usuário quando for viajar e precisar ficar ausente alguns dias, o tratador é quem vai assumir o comando de alimentar o seu animal doméstico e fazendo com que saiba se está comendo corretamente.

3.2. Arduino

Segundo Banzi e Shiloh (2012), o Arduino é uma ferramenta eletrônica de computação física aberta, é uma plataforma open source onde qualquer pessoa tem acesso ao seu *hardware* e seu *software*, de fácil uso que utiliza um cabo USB (Porta Universal, em inglês Universal Serial Bus) para conectar no computador podendo fazer a programação direta do seu *software*.

Afirma McRoberts (2015), a maior vantagem que o Arduino tem sobre outras plataformas de desenvolvimento de micro controladores, é a sua facilidade e sua utilização, mesmo pessoas que não são da área, ou seja, qualquer um sem possuir técnica ou algum conhecimento possa rapidamente aprender o básico e por si só criar projetos mesmo que simples, em um intervalo de tempo relativamente curto.

Como foi dito o Arduino funciona como um microcontrolador e possui um microprocessador, dependendo do tipo de Arduino que utiliza, pode ser energizado por um plugue de conexão USB, por uma bateria de 9V ou por uma fonte de alimentação.

O Arduino, depois de programado, pode ser desconectado para trabalhar de forma independente, sendo capaz de controlar uma simples luz, um led, um ventilador e até mesmo um robô.

No Arduino é possível realizar vários tipos de projetos, podendo sempre inovar, mas assim como qualquer outra tecnologia há suas limitações de instruções, já que depende muito do modelo que está utilizando. A seguir apresenta o método de linguagem que pode ser utilizada no *software* de uma placa.

3.2.1. Métodos de Linguagem de Programação e *Software*

A comunicação padrão é via USB pode ser executado em uma plataforma Windows, Macintosh e Linux para fazer a programação nos sketches (rotina de boot e rotina de loop) para a placa, o código escrito será traduzido para a linguagem C e o compilador “avr-gcc”, do próprio Arduino, realiza a tradução final de seus comandos em uma linguagem que será compreendida pelo micro controlador (BANZI, 2012).

A linguagem do Arduino é a linguagem C/C++, possibilita adicionar bibliotecas que ajudam no desenvolvimento da solução. Através da programação pode-se definir pinos de entrada e saída, ler valores analógicos e digitais, com isso pode adicionar vários componentes, o *download* dessas bibliotecas muitas vezes está disponível no site do fabricante do componente e na própria IDE (*Integrated Development Environment*, em português Ambiente de Desenvolvimento Integrado) do Arduino tendo acesso a maioria das bibliotecas para os componentes (EVANS; NOBLE; HOCHENBAUM, 2013).

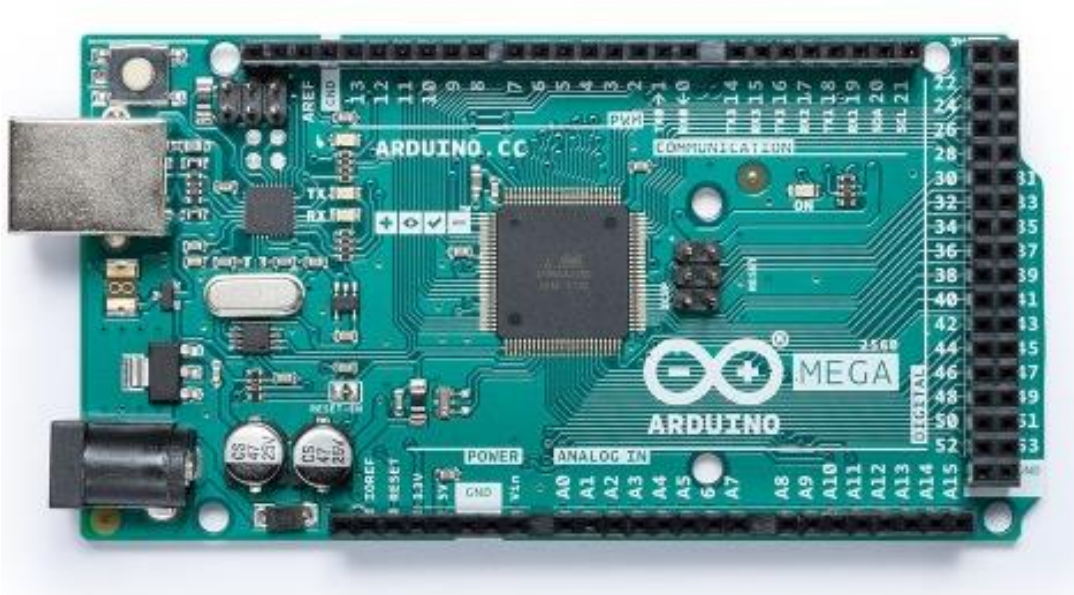
3.2.2. Tipos de Arduino

Conforme Arduino (2019) existem muitos modelos de Arduino, por isso o tipo de placa para o uso vai depender do projeto que vai ser desenvolvido e o número de portas também influenciará na escolha, é de fácil acesso com o *software.* De acordo com Monk (2013), por ser um projeto de placa aberto, qualquer pessoa pode criar uma placa de baixo custo tornando ainda mais barata, além delas serem compatíveis com as placas Arduino. A seguir apresenta-se alguns modelos oficiais do Arduino (2019):

- **Arduino Mega 2560**

É um dos mais velhos da família Arduino, foi criado na finalidade de executar projetos mais complexos e mais robustos, é muito recomendada e usada para projetos como impressoras 3D e projetos de robóticas já que fornece um aumento de performance de entrada e saída referente a outros Arduino, abaixo uma imagem do Arduino Mega 2560 na figura 2.

Figura 2 – Exemplo de Arduino Mega

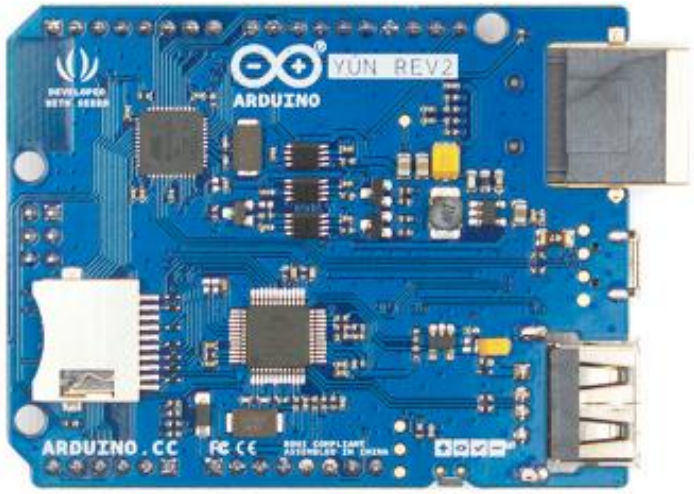


Fonte: Arduino (2019)

- **Arduino Yun**

Esta é umas das placas mais perfeitas para projetos de IOT (*Internet of Things* ou Internet das Coisas), tem um grande poder porque é baseada em um sistema Linux e possui opções de conexões muito avançadas. Possui entrada Wifi e Ethernet, além de ter uma biblioteca Bridge capaz de estender ainda mais seus recursos relacionados, além de acessar remotamente seus esboços, apresentado na figura 3.

Figura 3 – Exemplo de Arduino Yun



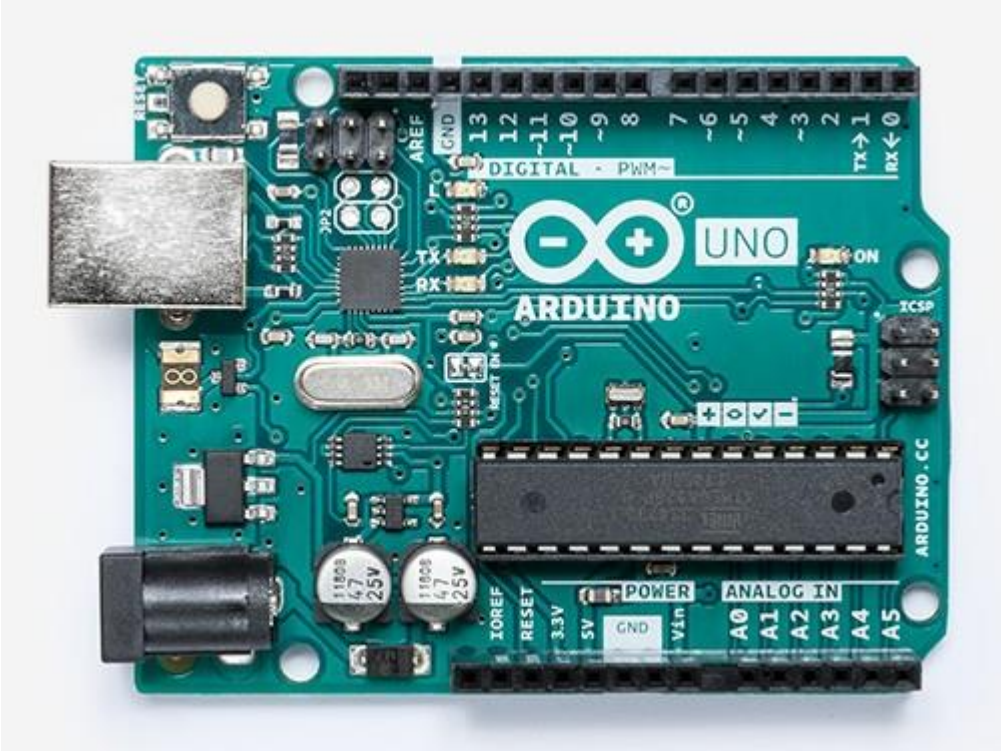
Fonte: Arduino (2019)

- **Arduino Uno**

O Arduino Uno seria a placa ideal para começar com eletrônica e codificação, é a placa mais robusta, mais usada, mais barata e eficiente para muito projetos, é a última geração de sua família já que é muito semelhante ao Duemilanove e ao Diecimila, só que ela possui uma memória ainda maior aos seus antecessores, adiante representado na figura 4.

Para a elaboração do projeto deste trabalho será utilizada uma placa neste modelo, pois possui as portas suficientes para ligarmos todos componentes que necessários.

Figura 4 – Exemplo de Arduino Uno



Fonte: Arduino (2019)

3.3. Engenharia de *software*

A engenharia de *software* é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até a manutenção dele, depois que entrou em operação (SOMMERVILLE, 2003).

Para PRESSMAN (1995) a engenharia de *software* é como uma “tecnologia em camadas”. Toda iniciativa deve ser apoiada por um compromisso com a qualidade. Acima da camada da qualidade encontram-se os processos, logo acima, os métodos e, acima destes, as ferramentas. Ao longo da história da engenharia de *software* foram sendo construídas ferramentas computadorizadas para apoiar o desenvolvimento e foram concebidos vários modelos de processos de *software* e nenhum pode ser considerado o ideal, devido às suas divergências.

Entretanto, para SOMMERVILLE (2003), todos compartilham de atividades fundamentais como especificação, projeto e implementação, validação, evolução e conclusão.

Esses processos são utilizados para coordenar projetos de desenvolvimento de *softwares* reais. Em Engenharia de *Software*, processos podem ser definidos para atividades como desenvolvimento, manutenção, aquisição e contratação de *software*.

O desafio mais comum dentro do processo de desenvolvimento é entregar um produto que atenda às necessidades reais do cliente, dentro do prazo e orçamento previstos (FAGUNDES, 2005). Como suporte a esta atividade, a Engenharia de *Software* oferece metodologias que auxiliam os desenvolvedores durante o processo de desenvolvimento, entre elas as metodologias tradicionais e as metodologias ágeis, que serão abordadas nos tópicos a seguir.

3.3.1. Métodos clássicos (tradicionais)

Na década de 1970 o desenvolvimento de *software* era executado de forma desorganizada, desestruturada e sem planejamento. O produto final era de má qualidade e que não correspondia às reais necessidades do cliente (PRESSMAN, 2001). A partir deste cenário surgiu a necessidade de tornar o desenvolvimento de *software* um processo estruturado, planejado e padronizado (NETO, 2004).

Segundo Neto (2004), no início dessa padronização foram mantidos conceitos típicos da Engenharia Civil, que ajudaram a sistematizar o processo de desenvolvimento de *software*. Conforme Fagundes (2005), o desenvolvimento de *software* era caracterizado como um processo descritivo, baseado em um conjunto de atividades predefinidas e planejadas.

As metodologias consideradas tradicionais são conhecidas como orientadas a documentação e têm como característica marcante dividir o processo em etapas ou fases bem definidas.

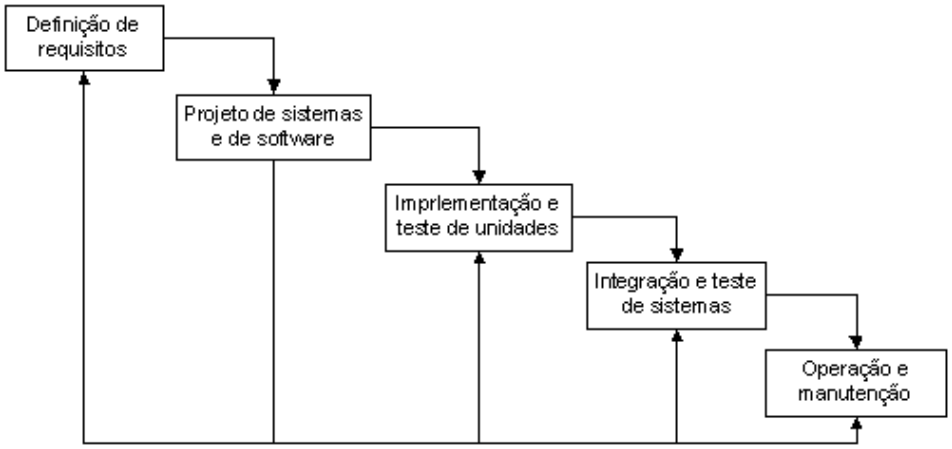
Muitas metodologias foram desenvolvidas sobre o Modelo Cascata (SOMMERVILLE, 2004). Esse é um modelo em que as fases são seguidas de maneira linear. Cada fase está associada ao seu término com uma documentação padrão que deve ser aprovada para que se inicie a etapa posterior e todo o processo é documentado (NETO, 2004).

De acordo com SOMMERVILLE (2003) e PRESSMAN (1995), existem diversos modelos de processo de *software*. Os mais conhecidos são:

- **Modelo cascata**

Este modelo foi idealizado em 1970 e tem como característica principal a sequencialidade das atividades: sugere um tratamento ordenado e sistemático ao desenvolvimento do *software*. Cada fase transcorre completamente e seus produtos são vistos como entrada para a nova fase; o *software* é desenvolvido em um longo processo e entregue ao final deste. O autor sugere laços de feedback, que permitem realimentar fases anteriores do processo, mas em geral o modelo cascata é considerado um modelo linear. Críticas ao modelo Cascata sugerem a inadequação deste a processos reais; em geral, há muito intercâmbio de informações entre as fases, e raramente ocorrem projetos onde não há concorrência das fases em si. Além disso, o modelo Cascata não leva em consideração questões modernas importantes ao desenvolvimento: prototipação, aquisição de *software* e alterações constantes nos requisitos, por exemplo. Abaixo mostra as fases do modelo cascata na figura 5.

Figura 5 – Modelo Cascata

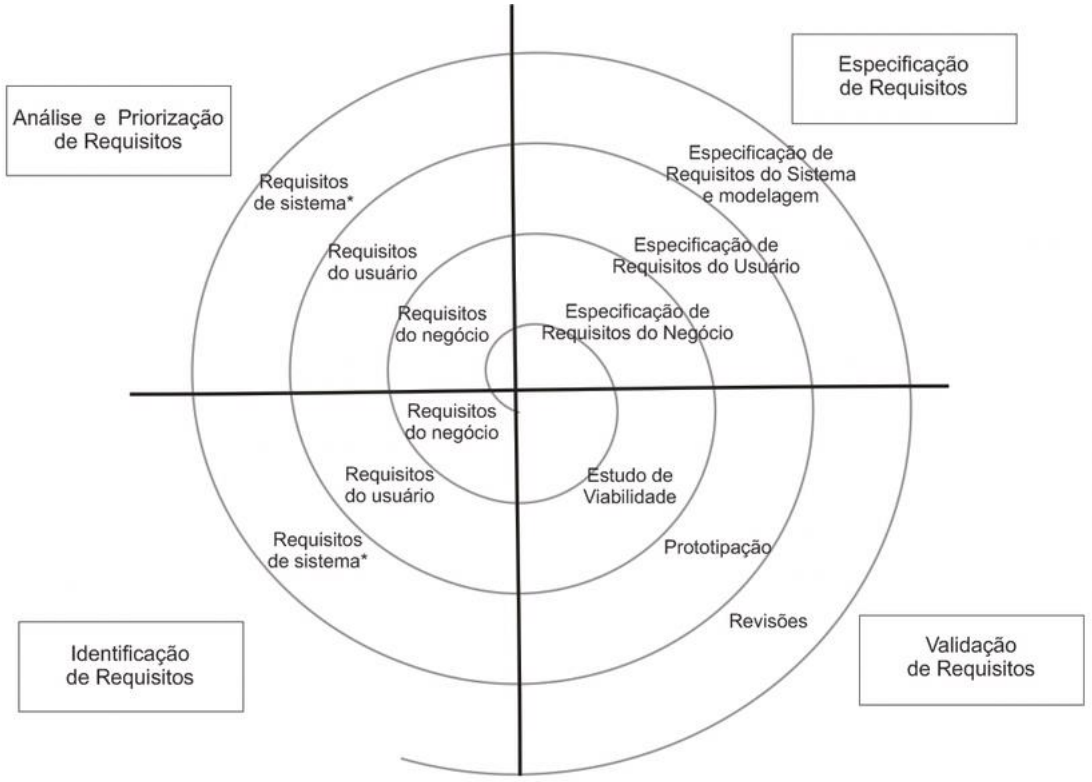


Fonte: SOMMERVILLE, 2004, p.38

- **Modelo espiral**

O modelo espiral é uma forma elaborada do modelo cascata, introduzido por um artigo publicado na IEEE Computer em maio de 1988. Foi sugerido um modelo evolucionário para o desenvolvimento de software, baseado em uma sequência de fases que culminam em versões incrementais do software. Determinação dos objetivos; Análise de risco e Desenvolvimento. Geralmente, modelos incrementais têm o objetivo de lidar melhor com um conjunto de requisitos incertos ou sujeitos a alterações. O modelo espiral parece mais bem adequado a projetos reais que o modelo cascata, representado na figura 6.

Figura 6 – Modelo Espiral



Fonte: Sommerville (2007).

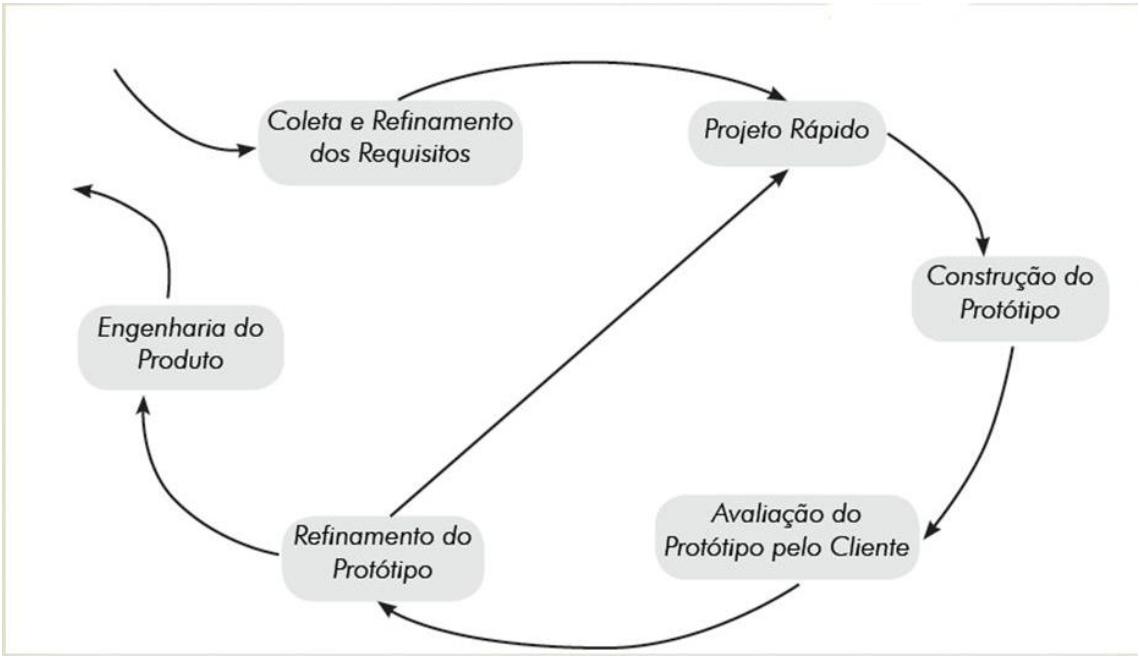
- **Modelo Prototipação**

O modelo Prototipação é um processo que tem como objetivo facilitar o entendimento dos requisitos, apresentar conceitos e funcionalidades do software. Desta forma, podemos propor uma solução adequada para o problema do cliente, aumentando sua percepção de valor.

Os protótipos também são grandes aliados das metodologias ágeis de desenvolvimento, uma vez que garantem maior alinhamento entre a equipe e o

cliente. Eles podem ser desenvolvidos em diferentes níveis de fidelidade: quanto maior ela for, mais o protótipo se assemelhará ao resultado entregue. No entanto, um protótipo de alta fidelidade leva mais tempo para ser criado ou modificado. A escolha do protótipo ideal varia de acordo com o nível de entendimento do negócio, a complexidade dos requisitos, prazo e orçamento para elaboração, na figura 7 aponta as fases deste modelo.

Figura 7 – Modelo Prototipação



Fonte: Pereira et al (2007).

3.3.2. Métodos ágeis

A partir da década de 1990 começaram a surgir outros métodos possibilitando uma nova abordagem de desenvolvimento, em que os processos adotados tentam se adaptar as mudanças com foco no apoio a equipe de desenvolvimento em seu trabalho (FAGUNDES, 2005). As metodologias ágeis surgiram como uma reação às metodologias tradicionais e tiveram como principal motivação criar novas alternativas.

O termo Metodologia Ágil se tornou mais conhecido em 2001, quando um grupo de 17 especialistas formou a Aliança Ágil que estabeleceu o Manifesto Ágil para o Desenvolvimento de Software.

De acordo com Fagundes (2005), para auxiliar as pessoas a compreender o enfoque do desenvolvimento ágil, esses especialistas criaram doze princípios de métodos ágeis de desenvolvimento de software.

O chamado Manifesto Ágil não é contra os modelos adotados pela abordagem tradicional, mas inclui técnicas utilizadas na Engenharia de Software que não seguem o padrão proposto pelas metodologias tradicionais (HIGHSMITH, 2002). Dessa forma, não rejeita os processos e ferramentas, a documentação, apenas mostra que eles têm importância secundária se comparado com os indivíduos e iterações e com a colaboração e respostas rápidas as mudanças.

Nas metodologias ágeis o cliente faz parte da equipe e acompanha o processo de desenvolvimento. A colaboração do cliente faz com que certos documentos tornem-se desnecessários e evita exagero, priorizando uma documentação apropriada evitando excessos e gerando somente o necessário (HIGHSMITH, 2002).

Outro ponto importante das metodologias ágeis é o enfoque nos aspectos humanos ao invés dos aspectos de Engenharia, reconhecendo a importância dos processos e ferramentas, mas ressaltando a importância do relacionamento entre os indivíduos que compõem o time do desenvolvimento de software ressaltando as boas práticas de programação. Propõem uma alternativa que busca a melhoria e a agilidade do processo, diminuindo a burocracia. A metodologia ágil possui costuma ser aplicada mais comumente em projetos não muito complexos que utilizam ciclos curtos (HIGHSMITH, 2002)

Existem muitos tipos de metodologias ágeis, mas vamos explicar mais detalhes no modelo SCRUM. Essa foi a ferramenta escolhida para utilização de gerenciamento do projeto, pois é o ideal para equipes pequenas. Facilitando a observação de todos os membros da equipe e sem grandes impactos nas mudanças das prioridades que podem ser manejadas com o menor esforço de alteração a cada ciclo.

3.3.2.1. A metodologia Scrum no desenvolvimento de sistemas

O desenvolvimento de um sistema pode ser comparado ao desenvolvimento de um projeto e o sucesso de ambos depende essencialmente da forma como foram

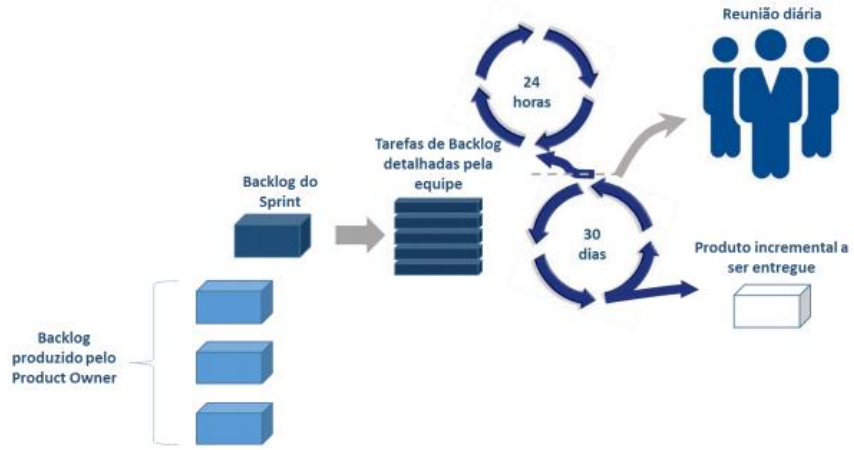
gerenciados. Neste contexto, a metodologia de gerenciamento SCRUM, conforme mencionado por Pereira et al. (2007), foi desenvolvido na década de 1990.

Dividindo o desenvolvimento dos sistemas em ciclos de curta duração (no máximo algumas semanas), tem como objetivo realizar entregas periódicas e rápidas. Esta metodologia traz vantagens, conforme Pereira et al (2007), dentre elas criar um ambiente favorável para definição de requisitos e inovação durante o desenvolvimento, tornando-o mais colaborativo e produtivo entre desenvolvedores e cliente, trazendo como resultado a entrega mais ágil de um produto mais bem adaptado à realidade do cliente. Além disso, a curta duração de ciclos também garante o constante planejamento e minimiza os riscos totais do projeto.

A metodologia define alguns papéis com diferentes atribuições entre os envolvidos no desenvolvimento do sistema e o ciclo é demonstrado de maneira simplificada na Figura 8.

O ciclo tem início quando o *Product Owner* (responsável pelo projeto) apresenta o *Backlog*, o que pode ser resumido como um conjunto de macro-tarefas a serem executadas. Em conjunto com a equipe de desenvolvimento, o PO seleciona as macro-tarefas, que serão incluídos no Sprint (ou ciclo), e a equipe de desenvolvimento será responsável por realizar reuniões diárias para atualização das tarefas já realizadas até o fim do sprint, quando o produto final é entregue.

Figura 8 – Ciclo do Scrum



Fonte: Pereira et al (2007).

É importante também destacar o papel dos membros da equipe envolvida no Sprint:

- Scrum *team* (ou equipe Scrum): responsável pela execução das tarefas de Backlog;
- Scrum master: responsável por garantir o cumprimento das regras e andamento do projeto;
- Cliente: participa na geração dos itens de Backlog selecionados pelo Product Owner;
- Gerente: responsável pela decisão final e definição dos requisitos do produto; Product Owner (PO): tem o papel de analisar o itens de Backlog determinados por clientes (tanto internos quanto externos), gerar uma especificação prática e inserir nos Sprints de acordo com a definição do Gerente.

Uma breve apresentação da metodologia se faz necessária para definir o papel dos participantes da equipe e realizar um paralelo entre as atividades realizadas pela equipe e o presente estudo. Algumas das tarefas realizadas principalmente pelo Gerente e PO poderão ser comparadas ao resultado do estudo, tais como a seleção de Backlog (identificação de uma necessidade do sistema), definição dos requisitos do produto.

3.4. UML

A UML (*Unified Modeling Language* ou Linguagem de Modelagem Unificada) é uma linguagem de modelagem por meio do paradigma de orientação a objetos sendo padronizada na Engenharia de *Software*.

Segundo Guedes (2006), a UML gera código por meio de diagramas UML, mas seu principal objetivo é auxiliar e definir um software através dos engenheiros, montando requisitos e a estrutura lógica. Por isso, é definida como uma linguagem de modelagem, sendo confundida com linguagem de programação. Seu processo é considerado dinâmico porque está em constante mudanças, seja para modificar, atualizar e melhorar o sistema.

Conforme Booch (2005), a UML é uma etapa para o desenvolvimento de um sistema, uma linguagem padrão na elaboração da estrutura, designado a visualizar, especificar, construir e documentar. A documentação é uma das formas de modelagem, bastante eficiente por ser detalhada e precisa.

A primeira fase do desenvolvimento de um sistema é o levantamento de requisitos. Nesta etapa o engenheiro obtém as informações referentes às necessidades do usuário e quais funcionalidades precisam ser fornecidas pelo software. Os principais desafios desta fase estão associados à dificuldade de compreender o que o cliente deseja e, graças às incertezas, definir melhor maneira de fazer um sistema de acordo com seu pedido.

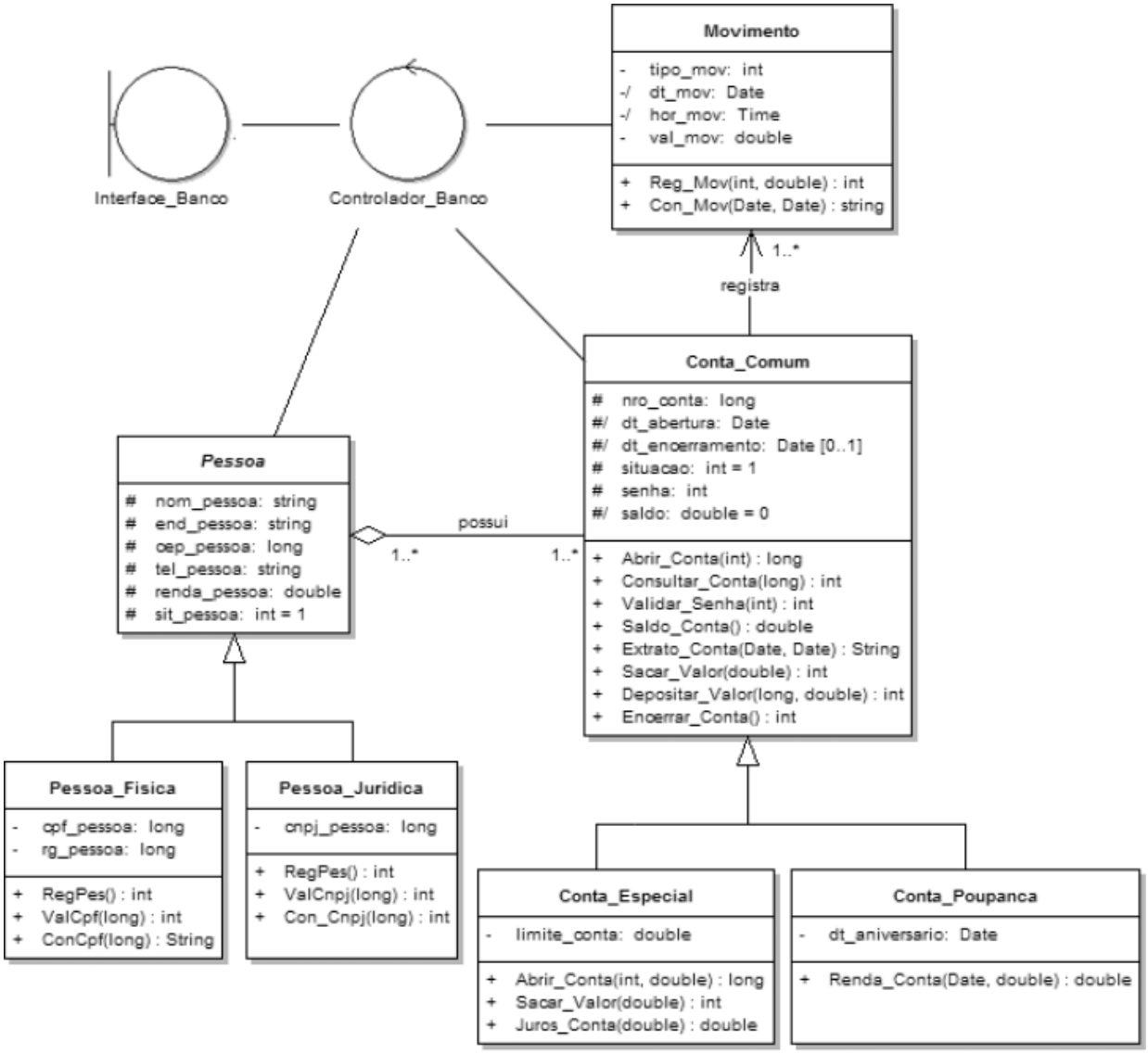
Guedes (2006) afirma que logo em seguida apresenta a análise de requisitos, separando as principais necessidades para compor o sistema, examinando, verificando e determinando se o sistema planejado está sendo atendida com o que o cliente pediu. A UML possui diagramas para analisar o sistema em diferentes níveis, analisando a organização estrutural e o comportamento do processo. Em seguida apresenta os tipos de diagramas da UML.

3.4.1. Diagrama de Classes

Booch, Rumbaugh e Jacobson (2012) dizem que os diagramas de classes são encontrados em sistemas modelados orientados a objeto, mostrando seus relacionamentos entre classes e interfaces, trocando ideias entre si. Assim como o próprio nome, define a estrutura das classes.

É denominado segundo Guedes (2006) como o diagrama mais utilizado e mais importante podendo ser utilizado como auxílio para outros diagramas, é determinado através de atributos e métodos por cada classe ligados por meio de relacionamentos, a seguir um exemplo de diagrama de classe abaixo na figura 9.

Figura 9 – Exemplo de Diagrama de Classes



Fonte: Guedes (2009)

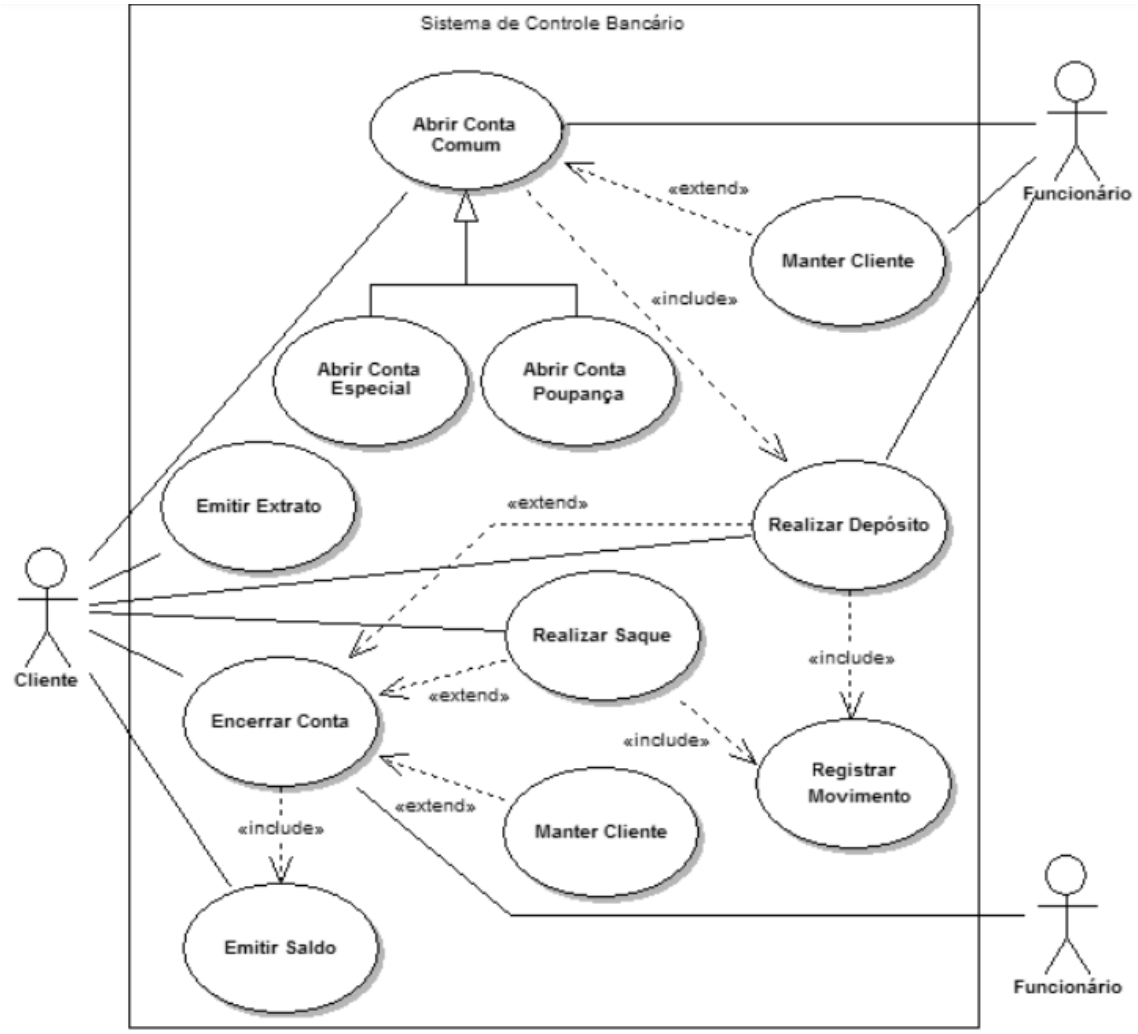
3.4.2. Diagrama de Casos de Uso

De acordo com Guedes (2006), esse diagrama é o mais geral e informal, utilizado como base para outros diagramas e ideia de como será conduzido pelo sistema, realizado nas fases de levantamento e análise de requisitos. É representado por atores, casos de uso e relacionamentos.

Para Booch, Rumbaugh e Jacobson (2005), é uma modelagem de aspecto dinâmico, têm como objetivo de modelar o comportamento de um sistema, um subsistema ou uma classe, visualizando especificando e documentando. O diagrama de casos de uso faz com que fique acessível e compreensível exibindo visão externa

sobre como os elementos pode ser utilizado, abaixo apresenta um exemplo de diagrama de casos de uso na figura 10.

Figura 10 – Exemplo de Diagrama de Casos de Uso



Fonte: Guedes (2009).

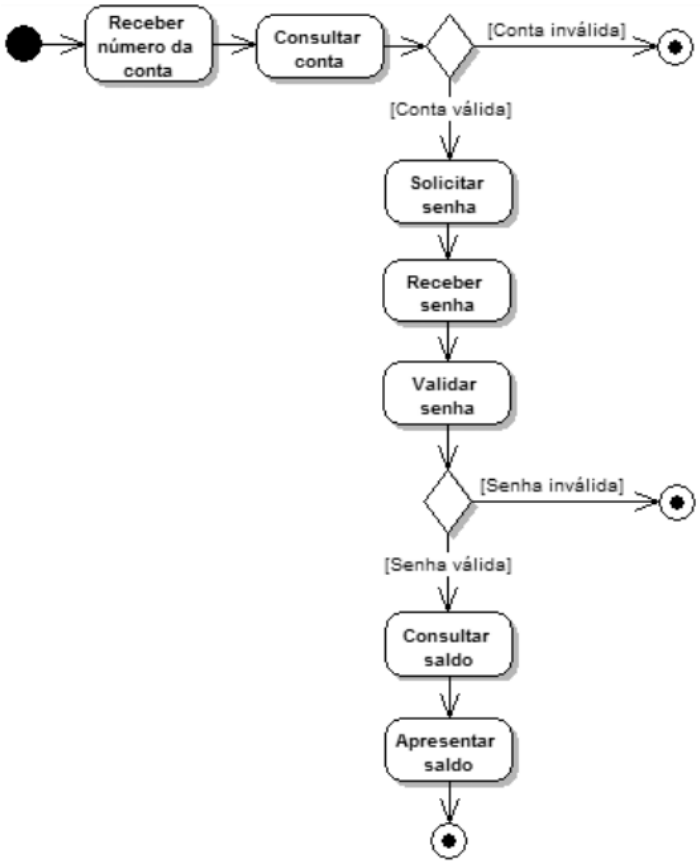
3.4.3. Diagrama de Atividade

Guedes (2006) afirma que o diagrama de atividade possui um estado inicial, estado final, transições e barras de sincronização, ou seja, se preocupa em descrever os passos ao serem percorridos para a conclusão de um método do sistema ou de uma parte do algoritmo específico.

Segundo Booch, Rumbaugh e Jacobson (2005), os diagramas de atividades mostram o fluxo de controle entre objetos e os dados de cada passo do sistema, pode fornecer uma visualização ampla e dinâmica, seu diagrama é conectado

através de setas, seus componentes são estado de ação, ponto de decisão, símbolo de início e fim da atividade. Preocupam descrever os passos a serem percorridos no sistema para ter uma atividade específica, segue abaixo um esquema utilizado com este diagrama.

Figura 11 – Exemplo de Diagrama de Atividade



Fonte: Guedes (2009)

3.5. Internet das coisas

Para Santos (2016), a Internet das Coisas ou da sigla inglês *Internet of Things* (IOT) começou com o avanço de sistemas embarcados, microeletrônica, comunicação e o uso de sensores. Também com a expansão da internet perceberam a possibilidade de poder automatizar o sistema ainda mais, fazendo com que esses sensores poderiam ser postos em locais em que há um difícil acesso. Com isso não teria a necessidade de ter alguém no local para poder recolher as informações do aparelho, dispositivos e sensores.

Segundo Oliveira (2017), a Internet das coisas vai além de conseguir fazer um sistema que liga uma lâmpada de uma casa através de um *smartphone* e não é apenas ligar as “coisas” via internet, mas sim fazer com que esses dispositivos tornem inteligente e muitas vezes autônomos ao ponto de saber quando ou não ligar. Quando pensamos em IOT temos que levar em consideração a comunicação através da Internet entre dispositivos e usuários fazendo com que os trabalhos dos usuários sejam automatizados.

Segundo Christophe et al. (2011), a internet das coisas tem como propósito conectar esses dispositivos à internet via IP (Internet Protocol). Isso pode gerar certa vulnerabilidade, caso sejam adotados o uso de senhas fracas, serviços de redes inseguros, interfaces mal desenhadas e a falta de atualização.

Quando usado os dispositivos e sensores em sistemas IOT os dados captados são transformados em informações. Com essas informações haverá um leque de possibilidades, por exemplo o uso dessas informações poderá alimentar uma inteligência artificial que ajudará nas tomadas de decisões, fazendo com que sempre estará à frente de possíveis problemas, e podendo implementar um sistema que automaticamente buscará uma solução. Isso fará com que o trabalho seja mais ágil e eficiente.

Entretanto um dos maiores problemas que enfrentamos ao usar os dispositivos da IOT é que precisam de fontes de energia. Atualmente esses dispositivos ou sensores são alimentados por baterias e muitas vezes elas são recarregáveis. As baterias não são adequadas para fazer essas tarefas pois esses dispositivos estão em locais com acesso difícil (dentro de outros dispositivos). Então todo *hardware* e o *software* deve ser projetado para ter uma eficiência energética alta (RERUM, 2015).

Uma das soluções estratégicas para resolver o problema de energia é fazer com que o sistema faça a colheita de energia [Ramos et al. 2012]. A ideia é transformar as fontes externas renováveis em fonte de energia para armazená-las em baterias recarregáveis. Essas fontes podem ser por exemplo, solar, térmica, eólica e cinética [Liu et al. 2013].

O projeto consiste em um tratador inteligente para cães, nesse sistema será acoplado vários sensores como exemplo, altura do animal, presença, pressão e nível. Estes sensores serão ligados em um Arduino Uno, que irá transmitir e receber

informações para e de um computador que terá um banco de dados (onde será armazenada e retirada as informações). Esse computador fará todo o controle inteligente do tratador, estará conectado via internet com um sistema web onde o usuário poderá ter acesso às informações dos sensores e do cachorro que será cuidado pelo tratador.

Para resolver os problemas de energia no projeto a alimentação será através de fontes ligadas a tomadas, já que esse tratador não tem o propósito de trocar de lugar todo tempo. Mas a ideia é que futuramente seja adicionado um sistema de baterias que poderá manter o equipamento por um certo tempo ligado fora da alimentação.

3.6. Linguagens de programação

Para que se tenha um desenvolvimento correto usando linguagens de programação, é necessário antes entender a lógica de programação. Pode-se defini-la como uma técnica para desenvolver uma sequência lógica em que atingiremos um determinado objetivo depois aplicamos essa sequência lógica em uma determinada linguagem de programação.

Segundo Lopes et al. (2002) a lógica de programação consiste em uma técnica de encadear pensamento para poder atingir um objetivo previamente determinado para quem deseja trabalhar com desenvolvimento essa é uma técnica inicial.

Para Said (2007), a lógica de programação consiste em fazer o uso correto de pensamentos para o raciocínio e simbolização nos processos de programação de computadores. A lógica de programação pode ser inicialmente pensada, passada para um papel e depois esse pensamento poderá ser aplicado em qualquer das inúmeras linguagens de programação.

Marçula (2005) diz que a linguagem de programação é entendida como um conjunto de palavras ou vocabulário e um englobamento de regras gramaticais, usado para instruir o computador a realizar tarefas, criando um programa. Cada linguagem de programação tem seu próprio conjunto de palavras e sintaxe.

Para a elaboração do projeto do presente trabalho de graduação, serão utilizados três tipos de programação: programação *web*, programação para servidor

e programação para o *hardware*. A Tabela 2 mostra qual será a finalidade de cada tipo de programação no trabalho.

Tabela 2 - Aplicação dos tipos de programação no projeto do Tratador Inteligente

Tipo de Programação	**Uso no projeto**
Programação *web*	Onde o usuário vai poder fazer o cadastro de seus animais, tratador e para ter todas informações
Programação para o servidor	Nesse servidor será programado toda a inteligente e automação do projeto e ficará responsável por controlar a web e o *hardware*.
Programação para o *hardware*	Programação do Arduino propriamente dito. Esse é o microcontrolador que vai receber dados, enviar dados dos sensores.

Fonte: Elaborada pelos autores (2019)

Para entender melhor como funcionam e diferenciar cada um dos tipos de programação que serão utilizados neste projeto, mais detalhes delas serão apresentados a seguir:

- **Programação web**

Na programação *web* as linguagens utilizadas serão HTML (*Hypertext Markup Language*), CSS (*Cascading Style Sheets*) e JavaScript. A ideia é fazer um sistema *web* responsivo onde o usuário poderá ter acesso a as informações de onde ele estiver, tanto usando um computador ou um dispositivo móvel.

Segundo Freeman (2008) tanto o CSS e o HTML são linguagens que se utiliza para criar páginas *web* HTML, trabalha com a estrutura do código enquanto o CSS diz ao *browser* como exibir o conteúdo e adiciona estilo a elas.

HTML ou *Hypertext Markup Language* é uma linguagem interpretada responsável em dizer ao *browser* o que precisa saber para exibir uma página web (estrutura). Para estruturar marcamos o conteúdo com tags, e então chamamos as tags correspondentes e seu conteúdo de elementos (FREEMAN 2008).

Conforme Flatschart (2011), o HTML é a principal linguagem utilizada exclusivamente para *web*, sendo possível criar parágrafos, listas, *links*, tabelas, formulários e outros elementos que poderão adicionar imagens e vídeos.

Juntamente com o HTML podem ser incluídos documentos *web* ou outras linguagens como o Java Script ou PHP que vão adicionar mais interatividades para o usuário e fazer acesso ao banco de dados (FLATSCHART 2011).

CSS é a abreviação de *Cascading Style Sheets* (em português Folhas de Estilo em Cascata) serve para controlar a apresentação do HTML (FREEMAN 2008). Flatschart (2011) afirma que o CSS é uma linguagem de estilo que exercerá o papel de formatação e apresentação do conteúdo (layout, cores, fontes).

JavaScript é a linguagem de *script* mais utilizada no mundo, a principal função é incorporar o comportamento dinâmico para páginas *web*, exemplo, animações e uma melhor interatividade com os usuários (DEITEL, 2017).

Flatschart (2011) defende que o JavaScript é uma linguagem de programação com uma sintaxe muito simples e pode adicionar fragmentos de código JS no HTML, esses fragmentos irão controlar a interatividade da página como menus, janelas e painéis.

- **Programação para o servidor**

Para que seja feito todo controle dos processos do *hardware* ou as informações para a *web*, a linguagem Python será a responsável em automatizar o processo e tornar o projeto inteligente.

Python é uma linguagem de altíssimo nível, orientada a objetos de tipagem dinâmica e forte, interpretada e interativa com uma sintaxe clara e concisa, isso torna uma linguagem muito mais produtiva. É um *software* de código aberto, a linguagem possui estrutura de alto nível com uma grande coleção de módulos prontos para uso. Possui vários *frameworks* criado pela comunidade (BORGES 2010).

Segundo Borges (2010), a linguagem Python assim como outras linguagens modernas possui alguns recursos como: geradores, introspecção, persistência, metaclasses e unidades de teste. Interpretada através de *bytecode* pela máquina virtual Python, com isso tornando o código portável para rodar em qualquer sistema. Além de ser uma linguagem orientada a objetos também é multiparadigma, ou seja, suporta programação modular e funcional.

Borges (2010), afirma que Python pode interagir com outras linguagens como a linguagem C e Fortran e apresenta muitas similaridades com as linguagens dinâmicas Perl e Ruby.

- **Programação para o *hardware***

O Arduino tem uma linguagem própria que é a mistura de C e C++, mais para a frente entraremos mais a fundo para saber como funciona a programação e o Microcontrolador.

A linguagem C foi desenvolvido no início da década de 1970 por Dennis Ritchie conhecido inicialmente como linguagem para desenvolvimento para o sistema UNIX até hoje em dia onde a maior parte dos sistemas operacionais são desenvolvidos em C e/ou C++. (DEITEL, 2017).

Segundo Davis (2016), C++ é uma linguagem de programação de baixo nível e orientada a objetos. Geralmente usada para a criação de jogos e *software* gráficos, é linguagem sucessor ao C, onde o desempenho é sua prioridade.

Davis (2016) diz que a maioria dos programas utilizados hoje em dia é escrito em C++, com sua alta agilidade para programação por ser uma linguagem orientada objetos se torna uma linguagem fácil de programação em alta escala.

3.7. Banco de dados

Segundo Date (2003), banco de dados é uma coleção de dados inter-relacionados, é um repositório para uma coleção de arquivos de dados computadorizado, ou seja, um armário de arquivamento tendo acesso para acrescentar, inserir, buscar, excluir, alterar e remover arquivos existente no banco de dados.

Conforme Heuser (2009), banco de dados surgiu para evitar redundância de informações solucionado através do compartilhamento de dados armazenando uma única vez, assim foi chamado de banco de dados, que é uma coleção de arquivos integrados atendendo um conjunto de sistemas aos usuários. O compartilhamento de dados tem a representação da estrutura interna dos arquivos sendo a mais complexa por atender às necessidades de diferentes sistemas e com isso criando o sistema de gerência de banco de dados (SGBD).

Silberschatz, Korth e Sudarshan (2012) afirma que o SGBD armazena e recupera informações de um banco de dados de maneira conveniente e eficiente, são projetados para administrar grandes informações por isso deve garantir segurança dos dados armazenados. As pessoas sem perceber interagem indiretamente com o banco de dados através de relatórios impressos como extratos de cartões de créditos, por exemplo. Com o crescimento da Internet, permite que os usuários interajam diretamente com as vindas dos caixas automáticos, interfaces de computadores por telefone, acesso de livraria online ou para inscrever nos cursos de uma universidade, assim aumentando o acesso direto às bases de dados do usuário.

Para Taylor (2016), os primeiros bancos de dados eram estruturados com modelo hierárquico apresentando restrições e estrutura rígida com isso ocorrendo problemas de redundância. Após, desenvolveram banco de dados de rede obtendo menos redundância portanto ainda apresentava complexidade estrutural. Até que mais tarde, desenvolveram o modelo relacional com estrutura fácil de compreender e com menos redundância, eliminando seus antecessores, e desenvolvendo a linguagem SQL (*Structured Query Language*, em português Linguagem de Consulta Estruturada) para operar o banco de dados relacional. Além disso, surgiu também o modelo não relacional conhecido como banco de dados NoSQL (*Not Only SQL*, traduzindo, não apenas SQL), não contendo estrutura e nem linguagem do modelo relacional.

Segundo o site OpServices (2019), os bancos de dados relacionais são armazenados em forma de tabela, composta por colunas, tuplas (linhas) ou registros, considerado de fácil inserção e recuperação. Seus principais sistemas são Oracle, SQL Server, MySQL e PostgreSQL.

Um banco de dados não relacional trabalha de maneira oposta ao modelo relacional, não podendo ser tabulado em linhas e colunas, atendendo para ambientes com imagens, mapas ou em soluções fundamentado em nuvem. Seus principais sistemas são MongoDB, Regis e Cassandra (OpServices, 2019). No presente trabalho será utilizado o sistema MongoDB.

Hows et al. (2015) afirma que o mundo onde utiliza um banco de dados seja simples, velocidade e escalabilidade, sem configurações complexas, isso e muito mais o MongoDB garante fazer. Com um novo conceito de banco de dados onde não se usa tabelas, linhas, esquemas e SQL. Se o programador já usou RDBMS

(*Relational Database Management System*, em português Sistema de Gerenciamento de Banco de Dados Relacional) o MongoDB promete ser mais eficiente com o conceito de banco de dados orientado a documentos.

Segundo MongoDB (2019), esse banco de dados foi projetado para atender as demandas de aplicativos modernos onde o programador tem a melhor maneira de manipular os dados (banco de dados orientado a documento) em que é permitido organizar os dados do jeito que desejar.

Para o projeto “Tratador inteligente para cães” foi optado pela utilização do MongoDB pois tem a ideia de uma manipulação simples, rápida e eficaz dos dados. Com isso a automação que fará as mudanças de dados com muito mais facilidade. Outro ponto importante é que o MongoDB utiliza o formato JSON, facilitando a comunicação com o Arduino.

4. ESTUDO DE CASO

O projeto consiste em fazer um sistema inteligente automatizado para tratar cães. Sendo composto por espaços físicos apropriados, o alimentador será equipado por sensores que ajudarão no monitoramento das condições da ração. Na parte superior, onde a ração será colocada, os grãos serão atraídos por gravidade, porém, contidos pelo *hardware*, somente deverão seguir até um extrator inferior depois que o sistema permitir, na figura 13 mostra como será a representação do projeto.

Tendo em vista à necessidade de alimentarmos os animais de criação com regularidade, que nem sempre é possível devido a nossa vida agitada, o tratador será projetado visando a tratar de cães quando o dono está ausente. E assim acompanhando as refeições de onde estiverem.

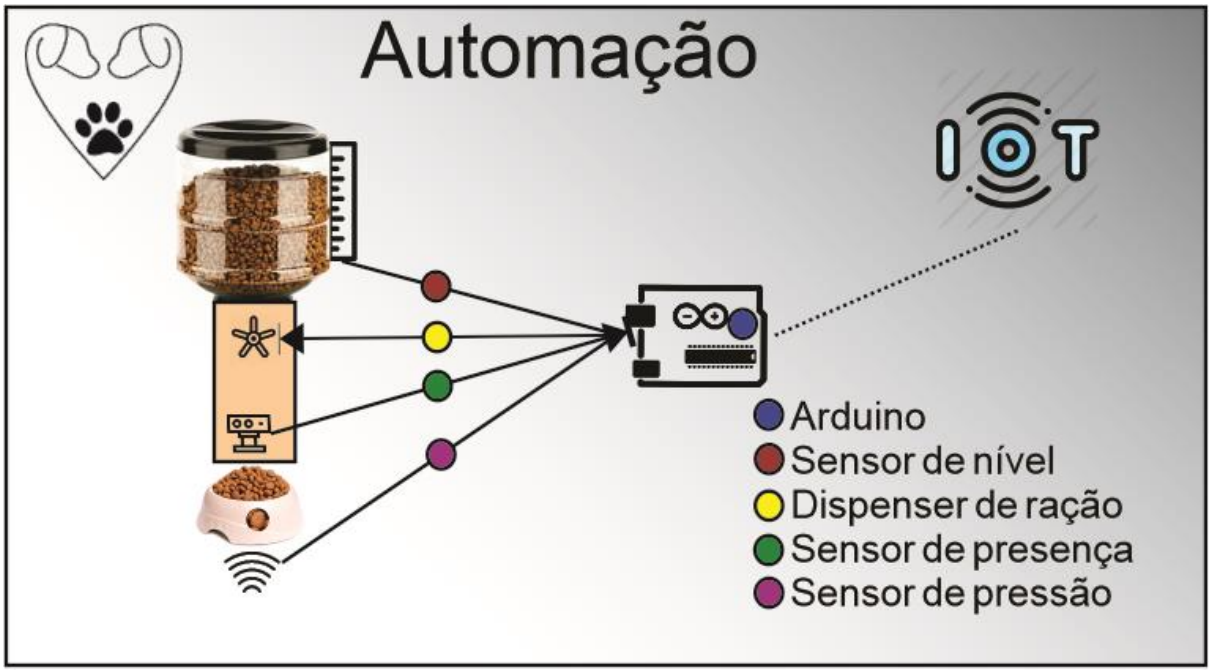
O tratador inteligente irá auxiliar uma família que tenha que viajar por alguns dias ou uma pessoa que não tem um tempo durante o dia para tratar adequadamente seu animal de estimação.

O sistema terá o propósito de utilizar uma base inicial de dados, visando o porte do cachorro, o consumo e a rotina de alimentação. Essa base inicial com o tempo irá sofrer uma evolução, que consiste em aumentar ou diminuir a quantidade de ração e o tempo em que o animal costuma comer.

O projeto será dividido em 3 sistemas (*hardware*, servidor e web). Para o *hardware* será usado um Arduino Uno que estará conectado alguns sensores de presença, de níveis e de pressão que farão as leituras dos dados, abaixo na figura 12.

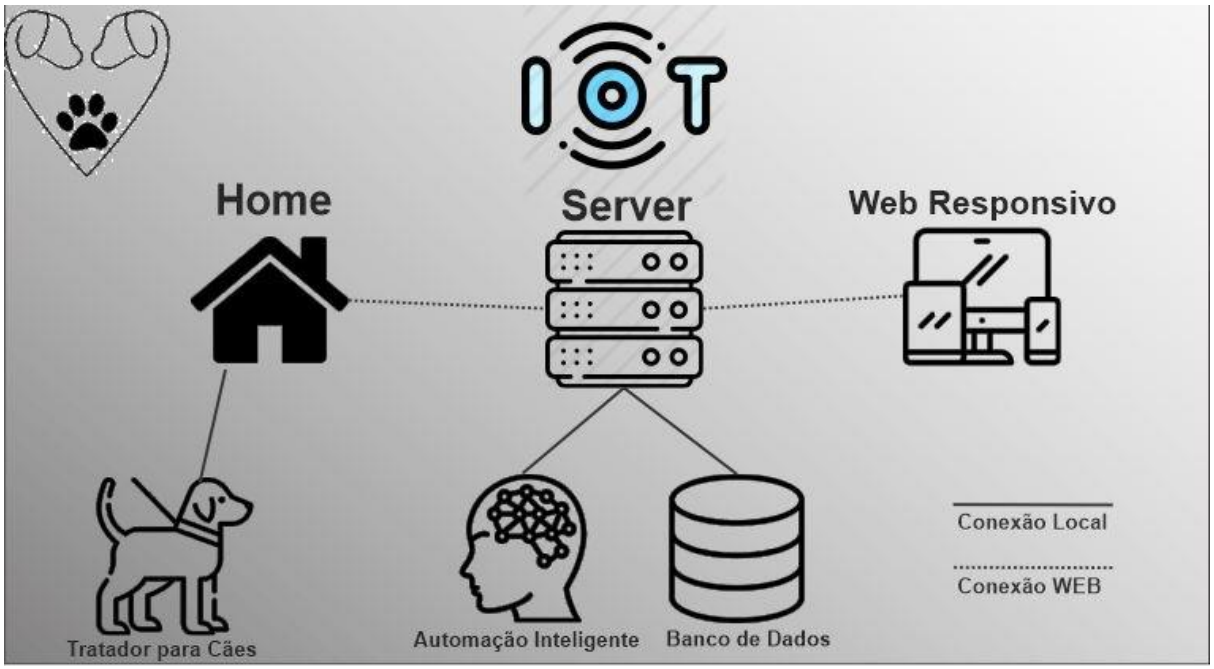
No Servidor faremos toda automação e inteligência do projeto, onde irá receber os dados do *hardware*, e enviar os dados, tudo via internet e, também no servidor será armazenado os dados em um banco de dados “MongoDB”. Na Web o usuário terá acesso a todas informações do cachorro, quando é a próxima alimentação, como foi as alimentações anteriores e caso ocorra algum problema será notificado.

Figura 12 – Representação dos componentes da Automação



Fonte: Elaborada pelo autor (2019).

Figura 13 – Representação dos componentes que formam a Internet das Coisas



Fonte: Elaborada pelo autor (2019).

4.1. Levantamento de requisitos

Para realizar o levantamento de requisitos neste projeto, será produzido entrevista, questionário, na qual entrevistaremos um veterinário para obter as informações sobre a alimentação e comportamento de um cachorro, e será apresentado também as fichas de requisitos.

4.1.1. Entrevista

A seguir apresenta a entrevista que realizamos com o veterinário Ricardo da Silva Araújo da empresa Confraria Animal Serviços e Produtos Veterinários Ltda, localizado em Mogi Mirim.

1. **Quantas vezes em média o cachorro se alimenta?**

O ideal para um cão ser alimentado é de 2 a 3 vezes ao dia com pequenas quantidades. Por exemplo um cão rottweiler come 2 quilos de ração ao dia, divide os 2 quilos por 3 e intercala.

2. **Como seria uma alimentação ideal para um cachorro? Quantidade por porte.**

Depende da raça. Por exemplo um cão de pequeno porte come 2 xícaras pequenas de ração intercalado 3 vezes ao dia, um cão de médio porte come 2 xícaras média de ração e o cão de porte grande depende muito da raça.

3. **Quais os problemas de usar um alimentador inteligente?**

Um tratador inteligente deve ser usado em casos como ausência grande do dono.

4. **O cachorro se adequa a um horário para alimentação?**

Um cão respeita rigorosamente os horários de ração e sabem a hora que vão comer.

5. Qual modo correto de armazenamento de ração?

Sempre armazenar em potes fechados com tampa, a ração tende a murchar e existem os perigos de ratos e baratas que causam doenças aos animais.

6. Quanto tempo a ração dura depois de sair de um pacote?

Depende do porte do cão.

7. Como evitar que os cachorros destruam o tratador inteligente?

Sendo um suporte fixado em paredes, seria interessante também um aviso como um “não” ou algum alarme rápido e não tão barulhento para não causar medo ao cão.

4.1.2. Questionário

Será mostrado o questionário realizado com o veterinário Ricardo da Silva Araújo da empresa Confraria Animal Serviços e Produtos Veterinários Ltda, localizado em Mogi Mirim.

1. Rações a granel podem ser misturadas? O contato de rações de marcas iguais ou diferentes podem contaminar ou alterar sua composição?

O granel não deverá ser misturado ao menos que o cliente queira misturar porque seu animal esteja acostumado a comer.

2. A estrutura do tratador será fixada na parede?

Sim.

3. O local de fixação da estrutura deve ser preservado de situações climáticas? Ou é resistente a fenômenos naturais corriqueiros?

Por tratar de um protótipo inicial, não precisará ser resistente.

4. O armazém de ração, o pote de rações será removível?

Sim, de fácil remoção para que possa fazer a limpeza.

5. O sistema também terá um app em que o dono possa ter controle do que está acontecendo?

Sim, poderão avisar caso precisa reabastecer o reservatório de ração, se o cachorro está comendo adequadamente, podendo monitorar se o cachorro está bem ou não.

6. O sistema permitirá enviar mensagem quando o tratamento for acionado para alimentar?

Sim.

7. Quantas refeições o sistema será capaz de efetuar para o cachorro?

Isso vai depender do comportamento do cachorro, deverá ser regulável no sistema.

8. Sobre a ração no pote, o que será feito?

Se isso ocorrer o sistema terá que na próxima vez dar menos ração ao cachorro, se adequando a quantidade.

9. Existe padrão de fabricação no tamanho da ração?

A importância está relacionada a fase de vida dos animais, para filhotes, adultos e idosos existem grãos diferentes, e o tamanho do cão também influência, como por exemplo, cães de raças grandes costumam comer com maior velocidade, quanto maior o cão, maior deve ser o bit de ração. A frente exclusiva da máquina deve ser diferente para cada linha, considerando esses pontos.

10. Quanto tempo a ração pode ficar exposta em um recipiente e sem prejudicar o animal?

Não existe um prazo estimado, é interessante avaliar o consumo diário do animal.

4.1.3. Fichas de requisitos

Para Sommerville (2007), requisitos são objetivos estabelecidos por clientes que definem as diversas propriedades de um sistema. Envolvem as restrições associadas a ele e estabelece um relacionamento entre objetivos, restrições e a especificação de um *software*.

Um conjunto de requisitos pode ser definido como uma condição ou capacidade necessária que um *software* deve possuir para que o usuário possa resolver um problema, atingir um propósito, para atender as necessidades, restrições de uma organização ou dos outros componentes do sistema (Sommerville, 2007).

A análise e especificação dos requisitos são iniciadas juntamente com a análise de um sistema, podendo se estender após a elaboração do documento de especificação do sistema e do planejamento do desenvolvimento, quando serão definidos os requisitos de um *software*. São atividades ligadas e que devem ser realizadas em conjunto. A análise é o processo de observação e levantamento dos elementos do domínio onde o sistema será introduzido. A especificação é a descrição detalhada do que o *software* deve fazer. A partir daquilo que foi analisado, apresentar uma solução de problemas que serão resolvidos pelo *software*. Determina quais as propriedades funcionais são necessárias (Filho, 2011).

O objetivo da definição dos requisitos é especificar o que o sistema deverá fazer e determinar os critérios de validação que serão utilizados para que se possa avaliar se o sistema cumpre o que foi definido. São divididas em dois tipos de requisitos: funcionais (RF) e não funcionais (RNF).

- **Requisitos Funcionais**

Segundo Sommerville (2007), requisitos funcionais descrevem as diversas funções que clientes e usuários querem que o software ofereça. Definem a funcionalidade desejada de um *software*. A funcionalidade pode ser descrita como a operação que pode ser realizada pelo sistema, seja através de comandos dos usuários ou pela ocorrência de eventos internos ou externos ao sistema.

> Esses requisitos dependem do tipo de *software* que está sendo desenvolvido, dos usuários a que o *software* se destina e da abordagem geral considerada pela organização ao redigir os requisitos. Eles descrevem as funções do sistema detalhadamente (SOMMERVILLE, 2007, p.81).

Para Filho, 2011, p.7, requisitos funcionais: “[...] representam os comportamentos que um programa ou sistema deve apresentar diante de certas ações de seus usuários.”.

Conforme as citações acima, um bom detalhamento dos requisitos é um ponto forte para obter as funcionalidades que se deseja do sistema.

Sommerville (2007) aponta alguns exemplos de requisitos funcionais:

- "O *software* deve possibilitar o cálculo de consumos diários, semanais, mensais e anuais".
- "O *software* deve emitir relatórios de consumo a cada quinze dias".
- “O *software* deve armazenar essas informações e disponibilizá-las ao usuário por um determinado período de tempo”.

A especificação de um requisito funcional deve determinar o que se espera que o *software* faça, sem a preocupação de como ele faz, no Apêndice A é apresentado um modelo de requisito funcional.

- **Requisitos Não Funcionais**

Requisitos não funcionais são qualidades globais de um *software*, como manutenibilidade, usabilidade, desempenho, custos e outros. Esses requisitos são descritos de maneira informal, controversa e difíceis de validar, como por exemplo; segurança e facilidade de uso (Sommerville, 2007).

> Os requisitos não funcionais são aqueles não diretamente relacionados às funções especificas fornecida pelo sistema. Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e espaço de armazenamento (SOMMERVILLE, 2007, p.82).

Segundo Bezzera (2007 p.24), usabilidade é “restrições sobre as plataformas de *hardware* e de *software* nas quais o sistema será implantado e sobre o grau de facilidade para transportar o sistema para outras plataformas”.

Sommerville (2007) cita exemplos de requisitos não-funcionais:

- "A base de dados deve ser protegida para acesso apenas de usuários autorizados".
- "O tempo de resposta do sistema não deve ultrapassar 30 segundos".
- "O tempo de desenvolvimento não deve ultrapassar seis meses".

Os requisitos exercem influência uns sobre os outros. A necessidade de se estabelecer os requisitos de forma precisa é crítica na medida em que o tamanho e a complexidade do *software* aumentam, no Apêndice B é apontado um modelo de requisito não funcional. E para realizar as fichas de requisitos funcionais e não funcionais, adotamos a utilização do Template Volare para a construção que será apresentado a seguir.

- **Template Volare**

Para Fischer (2001), introduzido a partir de 1995, o processo de especificação de requisitos chamado Volere fornece uma estrutura e rotas bem definidas sobre o conteúdo necessário para a identificação dos requisitos de um projeto. O processo é baseado na experiência de vários projetos de análise de negócios e busca constante de melhoria. O modelo proposto pelo método é composto por uma rota para o conhecimento dos itens necessários a fim de se especificar os requisitos para um produto.

O modelo conta com um checklist baseado na área de conhecimento de requisitos e fornece uma lista de questões que devem ser detalhadas para o levantamento dos requisitos do sistema, envolvidos no ambiente, usuários finais do produto, restrições da solução, escopo do trabalho e da estratégia, requisitos funcionais e de dados, e não funcionais, como aparência, usabilidade, desempenho, operacionais, manutenção, segurança, culturais, tarefas, custos, riscos, documentação entre outros que auxiliam no detalhamento do sistema (FISCHER, 2001, p. 87).

Este é um método que já foi adotado por milhares de organizações no mundo todo e seu modelo parte do princípio da sumarização de experiências em desenvolvimento de *software* para montagem de um modelo simples e completo. É um método completo de obtenção de requisitos, baseado nos casos de uso (FISCHER, 2001, p. 87). Este template é apresentado nas fichas de requisitos que podem ser encontradas no apêndice A e B.

REFERÊNCIA BIBLIOGRÁFICA

AGILE MODELLING. Disponível em: **<http://www.agilemodeling.com/>**. Acesso em: maio 2019.

APACHE TOMCAT. Disponível em:<**http://www.agilemodeling.com/**>. Acesso em: maio 2019.

ARDUINO. Disponível em: **<https://www.arduino.cc/>**. Acesso em: maio, 2019.

BANZI, Massimo; SHILOH, Michael. **Primeiros Passos com Arduino**. São Paulo: Novatec, 2012.

BOLZANI, Caio A. M. **Análise de Arquiteturas e Desenvolvimento de uma Plataforma para Residências Inteligentes.** 2010 155f. Tese (Doutorado) – Escola Politécnica da Universidade de São. Paulo. Departamento de Engenharia de Sistemas Eletrônicos, 2010.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. Rio de Janeiro: Elsevier Brasil, 2005.

BORGES, Luiz. E. **Python para desenvolvedores**. 2 ed. Rio de Janeiro: Novatec, 2010.

CHRISTOPHE, B.; VERDOT, V.; TOUBIANA, V. **Searching the 'web of things'. In: IEEE. Semantic Computing (ICSC)**, 2011 Fifth IEEE International Conference on. [S.l.], 2011. p. 308–315.

DATE, C. J. **Introdução a sistemas de banco de dados**. Rio de Janeiro: Elsevier, 2003.

DAVIS, Stephen R. **C++ para Leigos**. 7ªed. São Paulo: Berkeley, 2016.

DEITEL, Paul; DEITEL, Harvey. **Java: Como Programar**, 10ª ed. Pearson, 2017.

EQUIPE OpServices. **Quais os principais bancos de dados e quais suas diferenças?,** Porto Alegre, 26 fev. 2019. Disponível em: <https://www.opservices.com.br>. Acesso em maio de 2019.

EVANS, Martin; NOBLE, Joshua; HOCHENBAUM, Jordan. **Arduino em ação**. Novatec Editora, 2013.

FAGUNDES, Priscila B. **Framework para Comparação e Análise de Métodos Ágeis.** Dissertação de Mestrado. Universidade Federal de Santa Catarina. Florianópolis: 2005.

FLATSCHART, Fábio. **HTML5: embarque imediato**. Rio de Janeiro: Brasport, 2011.

FREEMAN, Elisabeth; FREEMAN, Eric. **Use a cabeça! HTML com CSS e XHTML**. 2 ed. Rio de Janeiro: Alta Books, 2008.

GIL, A. C. **Como elaborar projetos de pesquisa.** 4ªed. São Paulo: Atlas, 2002.

GROOVER, Mikell P. **Automação industrial e sistemas de manufatura**. Pearson Education do Brasil, 2011.

GUEDES, Gilleanes T. A. **UML: Uma abordagem prática**. 2 ed. São Paulo: Novatec Editora, 2006.

GUEDES, Gilleanes T. A. **UML 2: Uma abordagem prática**. São Paulo: Novatec Editora, 2009.

HEUSER, C. A. **Projeto de banco de dados: Volume 4 da Série Livros didáticos informática UFRGS**. Bookman Editora, 2009.

HIGHSMITH, Jim. **Agile Software Development Ecosystems**. Adisson-Wesley, 2002.

HOWS, David; MEMBREY, Peter; PLUGGE, Eelco. **Introdução ao MongoDB**. 1. ed. São Paulo: Novatec, 2015.

LIU, V., PARKS, A., TALLA, V., GOLLAKOTA, S., WETHERALL, D., and Smith, J. R. (2013). **Ambient backscatter: wireless communication out of thin air. ACM SIGCOMM Computer Communication Review**, 43(4):39–50.

MARCONI, M. A; LAKATOS E. M. **Fundamentos de metodologia científica.** 5ªed. São Paulo: Atlas, 2003.

MARÇULA, Marcelo; FILHO, Pio A. B. **Informática: conceitos e aplicações**. São Paulo: Érica, 2005.

MCROBERTS, Michael. **Arduino básico**. 2ª ed. Novatec Editora, 2015.

MONGODB. Disponível em: < https://www.mongodb.com/>. Acesso em: maio de 2019.

MONK, Simon. **Programação com Arduino**. São Paulo: Bookman, 2013.

MURATORI, José R.; DAL BÓ, Paulo H. **Automação Residencial conceitos e aplicações**. Minas Gerais: Educere, 2013.

NETO, Oscar N. S. **Análise Comparativa das Metodologias de Desenvolvimento de Softwares Tradicionais e Ágeis**. Trabalho de Conclusão de Curso. Universidade da Amazônia. Belém, 2004.

PAZOS, Fernando. **Automação de sistemas & robótica**. Rio de Janeiro: Axcel Books, 2002.

PEREIRA, Paulo; Torreão, Paula; Maçal, Ana S. **Entendendo Scrum para Gerenciar Projetos de Forma Ágil**. In.: Mundo PM, 2007.

PRESSMAN, Roger S. **Engenharia de Software**. Makron Books, 1995.

PRODANOV, C. C.; FREITAS, E. C. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico.** 2ªed. Novo Hamburgo: Feevale, 2013.

RAMOS, H. S., OLIVEIRA, E. M., BOUKERCHE, A., FRERY, A. C., and LOUREIRO, A. A. (2012). **Characterization and mitigation of the energy hole problem of many-to-one communication in wireless sensor networks. In Computing, Networking and Communications (ICNC)**, 2012 International Conference on, pages 954–958. IEEE.

RERUM (2015). **Advanced techniques to increase the lifetime of smart objects and ensure low power network operation**. RERUM.

RIBEIRO, Marco A. **Automação industrial. Apostila para curso de treinamento**, 1999.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN S. **Sistemas de banco de dados**. 6ª ed. Rio de Janeiro: Elsevier, 2012.

SOMMERVILLE, Ian. **Engenharia de software**. São Paulo: Addison Wesley, 2003.

SOMMERVILLE, Ian. **Engenharia de software**. 8ª ed. Addison Welsley, 2007.

TAYLOR, Allen G. **SQL para leigos**. 8ª ed. Rio de Janeiro: Alta Books, 2016.

YIN, Robert K. **Estudo de Caso-: Planejamento e métodos**. Bookman editora, 2015.

APÊNDICE

APÊNDICE A – Ficha de Requisito Funcional

Ficha de Especificação de Requisitos

(baseado no Template Volere**)**

Identificação do Requisito: RF01 – Cadastrar cachorro
Tipo de Requisito: Funcional
Caso(s) de Uso(s) vinculado(s): DCU02, DA02

Descrição: O sistema deverá ter um mecanismo de cadastro de cachorros com os seguintes dados: nome do cachorro, tipos de raça, porte, tipos de ração, horário da alimentação e quantidade sendo obrigatório o nome para a identificação do cachorro, porte para reconhecer ao aproximar do alimentador e o tipo de ração para controlar as refeições de cada cachorro. Qualquer usuário do sistema que possui o login e estiver logado, poderá realizar o cadastro. No final uma mensagem de "Cadastrado com sucesso!" deve ser exibida.

Justificativa: Atender à solicitação do cliente.

Solicitante: Ricardo da Silva Araújo.

Prioridade:
(X) Alta ( ) Média ( ) Baixa
Alta: Deve ter prioridade sobre as outras
Média: Devem ser realizadas após todas as de prioridade maior
Baixa: As últimas a serem realizadas

Material de Apoio: Entrevista

Histórico: Solicitação Inicial (04/04/2019)

Dados:

- nomeCachorro
- raca
- porte
- horarioAlim
- quantidade
- tipoRacao

APÊNDICE B – Ficha de Requisito Não Funcional

Ficha de Especificação de Requisitos

(baseado no Template Volere**)**

Identificação do Requisito: RNF01 – Sistema Operacional
Tipo de Requisito: Não Funcional
Caso(s) de Uso(s) vinculado(s): Não se aplica

Descrição: O sistema deverá funcionar no sistema operacional Microsoft Windows, versão 10, devidamente atualizada.

Justificativa: Atender à solicitação do cliente.

Solicitante: Ricardo da Silva Araújo.

Prioridade:
(X)Alta ( ) Média ( )Baixa
Alta: Deve ter prioridade sobre as outras.
Média: Devem ser realizadas após todas as de prioridade maior.
Baixa: As últimas a serem realizadas.

Material de Apoio: Entrevista.

Histórico: Solicitação Inicial (04/04/2019).

Dados: Não se aplica