Universidade de Aveiro Departamento de Eletrónica, Telecomunicações e Informática
2018

João Gonçalo Peixe Ribeiro

Pet Sense – Sistema de Monitorização de Animais em Hospitalização

Pet Sense – Animal Monitoring System with IoT

Universidade de Aveiro
2018

Departamento de Eletrónica, Telecomunicações e Informática

João Gonçalo Peixe Ribeiro

Pet Sense – Sistema de Monitorização de Animais em Hospitalização

Pet Sense – Animal Monitoring System with IoT

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Professor Doutor Ilídio Castro Oliveira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, em contexto de estágio em empresa na LVS S.A, com a supervisão do Mestre Luís Pinto.

o júri

presidente

Professor Doutor Joaquim Manuel Henriques de Sousa Pinto
Professor Auxiliar da Universidade de Aveiro

vogais

Professor Doutor Rui Pedro de Magalhães Claro Prior
Professor Auxiliar da Faculdade de Ciências da Universidade do Porto

Professor Doutor Ilídio Fernando de Castro Oliveira
Professor Auxiliar da Universidade de Aveiro

agradecimentos / acknowledgements

Aproveito este espaço para agradecer em primeiro lugar aos meus pais e irmão, pelo apoio e suporte incondicionais; em segundo à minha namorada pelo carinho e compreensão.

De seguida gostaria de agradecer ao meu orientador, Professor Doutor Ilídio Oliveira, pela orientação e disponibilidade manifestados ao longo da realização deste projeto. Ao Instituto de Engenharia Eletrónica e Telecomunicações de Aveiro e em particular ao Professor Doutor José Maria, pelo conhecimento e equipamentos partilhados.

Gostaria também de agradecer à Pet Universal e em especial ao Engenheiro Mestre Luís Pinto, não só por me terem dado a oportunidade de realizar este projeto, mas também pelo acolhimento. Um agradecimento ao Vítor Martins, com quem trabalhei proximamente.

Agradeço ainda a todos os meus colegas e amigos que me ajudaram e apoiaram ao longo do meu percurso académico.

palavras-chave Internet das Coisas, Processamento de *Streams*, Monitorização Animal.

resumo

A observação e tratamento de animais hospitalizados continua muito dependente de procedimentos manuais, especialmente no que diz respeito à colheita de sinais vitais (temperatura, frequência cardíaca, frequência respiratória e pressão arterial). Estes procedimentos manuais são dispendiosos em termos de tempo e afetam o bem-estar do animal.
Neste projeto, propomos o recurso a tecnologias IoT para monitorizar animais hospitalizados equipados com sensores que medem sinais vitais, com hardware acessível, e envio dos dados para um serviço na *cloud* que os analisa e armazena. O histórico dos valores e alarmes podem ser acedidos na web e incluídos na plataforma comercial da Pet Universal.
A arquitetura geral segue uma abordagem de processamento funcional, usando protocolos de telemetria para transportar dados e Apache Kafka Streams, analisando e lançando alarmes sobre potenciais condições de risco de acordo com a temperatura e pulsação.
O sistema foi totalmente implementado, embora com sensores de laboratório para simular os dispositivos a serem usados pelos animais. Conseguimos implementar um circuito de colheita e processamento de dados e integrar com o sistema de gestão clínica já existente.
A solução proposta oferece uma forma prática de monitorização continuada e de deteção de valores anormais de temperatura e frequência cardíaca em animais hospitalizados, tomando em conta as características do indivíduo monitorado (espécie e estado).

keywords IoT, Data Streaming, Animal Health Monitoring

abstract

The observation and treatment of animals in veterinary hospitals is still very dependent on manual procedures, including the collection of vital signs (temperature, heart rate, respiratory rate and blood pressure). These manual procedures are time-consuming and invasive, affecting the animal's well-being. In this work, we purpose the use of IoT technologies to monitor animals in hospitalization, wearing sensors to collect vitals, and low-cost hardware to forward them into a cloud backend that analyses and stores data. The history of observed vitals and alarms can be accessed in the web, included in the Pet Universal software suite.

The overall architecture follows a stream processing approach, using telemetry protocols to transport data, and Apache Kafka Streams to analyse streams and trigger alarms on potential hazard conditions.

The system was fully implemented, although with laboratory sensors to emulate the smart devices to be worn by the animals. We were able to implement a data gathering and processing pipeline and integrate with the existing clinical management information system.

The proposed solution can offer a practical way for long-term monitoring and detect abnormal values of temperature and heart rate in hospitalized animals, taking into consideration the characteristics of the monitored individual (species and state).

CONTENT

IMAGE INDEX

TABEL INDEX

ACRONIMS

IoT – Internet of Things

UI – User Interface

API – Application Programming Interface

SaaS – Software as a Service

DB – Database

HTTP – Hypertext Transfer Protocol

JSON – JavaScript Object Notation

IETF – Internet Engineering Task Force

AHMS – Animal Health Monitoring System

MOM – Message Oriented Middleware

GUI – Graphical User Interface

1 INTRODUCTION

Considered a mark to a new generation of technologies, the term "Internet of Things" (IoT) is the first concept needed to understand the so called "Industry 4.0" [1]. One common use case in IoT solutions is healthcare. In eHealth, monitoring the key vital signals of patients can benefit from using dedicated smart devices to measure and communicate the biometric values. These systems fall into an IoT approach when the biosensors send the data to a remote system for processing and decision-making. For example, patients can be monitored remotely while at home. An alarmistic function can alert nurses if the patient is suffering from an acute (or dangerous) condition.

Telemetry is also applied in a variety of areas, however, for the present work, we are interested in animal monitoring. A well-known example is present in wildlife and ecosystems being massively monitored by big companies such as Nacional Geographic[1], which use IoT systems to track the animal's position but also register bio-signals to study the animal's reactions to certain environment changes or preserve their ecosystem health, biodiversity distribution, patterns, and trends, across the globe. Progress in this area has rapidly developed, not only in terms of better equipment but also in the number of devices connected and the amounts of data transmitted [2].

The present dissertation addresses the development of "Pet Sense", a solution to monitor the vital signs of pets that are under hospitalization. The platform aims at collecting and analysing animal data, to speed up the routine operations and notify important events to the veterinary staff. In a broader sense, the motivation behind this project are our pets and how we can help and understand them better.

The project was proposed by Pet Universal[2], the company hosting the dissertation project, that already offers a software suite for general veterinary hospitals operations management.

1.1 Motivation

Pet veterinary and "Pet Care" institutions have been thriving through developed countries, and Portugal is not an exception as the majority of its population has a pet [3]. With the help of technology, humans can better care and understand pets as we still have a lot to learn about them [4].

Many scientific studies are done in areas such as pet psychology and pet behaviour either in captivity or in the wild, but no information was found in IoT technologies having served as a constant monitoring system for recovery of hospitalised pets (neither cats nor dogs).

Pet clinics have been integrating information technology solutions mainly for administrative purposes (client management, resource management, etc). To our best knowledge, the veterinary community in Portugal, Spain and Brazil is not using smart monitoring systems of animals during

[1] https://www.nationalgeographic.org/
[2] https://www.petuniversal.com/

hospitalization. This offers an opportunity to introduce technology to enhance data collection and alarm-detection.

The monitoring of animals in vet hospitals and clinics is done by periodically spending a human resource to physically check-up on the animal's vitals and registering it on a spreadsheet. Traditional veterinary monitorisation instruments are limited, intrusive and involve a slow and physical process, which is time-consuming and, therefore, very costly. This method may originate omissions in obtaining data and calculations. Furthermore, there is a clear difficulty in ensuring the necessary constant and thorough monitoring on the clinical state of the animal in hospitalisation. There is a struggle to prevent anomalies in high risk treatments, or at least perceive them in real-time, due to the rapid changes in the animal's state during recovery.

Pet Universal is a start-up in the Business Incubator of the University of Aveiro which produces and sells management and hospitalization software solutions to veterinary establishments as SaaS. It started in 2016 with the motto "We help those who help our pets". This project has that same objective and motivation. Pet's software aims to be an all-you-need software for any veterinary establishment (client-management, billing, hospitalisation, etc). In a broader sense, the motivation behind this project is our pets and how we can help/ understand them better.

A survey conducted by Pet Universal showed that, on average, 4 hours were spent weekly on monitoring hospitalised animals, and many mistakes were made daily. There is no way to predict or prevent risks in treatment other than the experience of the veterinary in question. The feedback collected by Pet Universal from vets around the country suggests that this project can overcome some shortcomings of existing software packages, by creating an independent platform of continuous collection of monitored data, for "real-time" observation and event detection that can be extended into the Pet Universal's product.

The next generation of products for pet care need to go beyond the paperless recording of clinical observations and care events and will address the active monitoring of vital parameters to monitor health values, alert for hazard conditions and feed predictive models of animal's health. The data gathering processes will feed data analysis routines to detect and notify, on near real-time basis, possible threatening hazards conditions and build models to enabling predictive analysis of animal health conditions based on observed trends.

The Pet Sense project will contribute to change the paradigm of the medicine sector veterinary surgeon, particularly at the hospitalization department, contributing to the improvement of veterinary care, increasing the success rate of treatments, saving the lives of more animals and optimizing and automating clinical/medical processes. It will contribute for the development of treatment and therapy of animal health, through smart wearables.

Pet Universal intends to continue to develop Pet Sense to present a product that is new to the market, which eliminates constraints (namely human failures in the diagnosis, analysis and continuous monitoring with the use of a nurse or veterinarian which is economically unfavourable).

1.2 Objectives

The main objectives of this work are:

To propose and implement a computing platform for sensor data collection, processing and long-term storage, following an IoT architecture;

Implement a basic alarmistic solution to find events of interest on the animal's data and raise notifications;

Integrate the IoT layer with the backend information system of the Pet Universal solution;

We seek an information system that continuously collects, processes, stores and presents data collected from biosensors mounted on hospitalized pets under treatment and require frequent monitoring. The solution should extend Pet Universal suite to build a rich record of each animal's vital signs.

On a more personal level, the objectives are to explore and learn about IoT architectures and technologies, have a full-time job experience working in a professional environment, and have the notion of what a year-long software engineering project is like.

1.3 Dissertation structure

This document contemplates seven chapters with the order as follows:

Chapter 1: Introduction - This chapter describes the context in which this project was originated, and why this project is beneficial in the specific area, as well as the initial goals for the project.

Chapter 2: State of the Art - This chapter sums up the history of monitoring animals, data stream processing tools and platforms that are currently on the market. This chapter aims at reflecting the research done by the author on the area of study. It gives more context, and reviews what has been done in the area, so far. Always with a correlation report to the project and on its limitations.

Chapter 3: Use Cases and System Requirements - This chapter describes which problems exist in the target scope, and how this project will contribute to solve them. Use cases are presented in detail.

Chapter 4: Architecture- This chapter presents the system architecture and how the system connects with other systems.

Chapter 5: Implementation - This chapter describes in detail how the different modules were implemented and how they connect with each other.

Chapter 6: Validation - This chapter presents the system's functional validation in terms of its use and describes results with simulated data based on real data from patients that were subjected to an veterinary hospitalization episode.

Chapter 7: Conclusion - This chapter presents the conclusions that were obtained after finishing the project, reviews which work was made and gives suggestions on several topics to future work.

2 STATE OF THE ART

The first phase of this project was to search the scientific community and the software market for similar solutions, and check whether comparable projects were under development thus getting insight about the subject of study.

2.1 Animal health monitoring

A big part of the development of software and computer systems has to do with the focus and purpose it will serve. In this example, this software is for humans but involves veterinary practises and knowledge.

2.1.1 Animal monitoring overview and applications

Monitoring is key on animal recovery at veterinary clinics and hospitals. Nevertheless, animal monitoring has many more applications and objectives. In 2017 a thorough list of computing and sensing technologies used in projects with animals (domestic, farm and wild animals) is described in reference [5]. The goal was to systematically review the existing literature on this scope and, as such, it served as one of the main sources of bibliographic review on the research made. We can classify the research works on animal telemetry we found in the literature according to three main groups: tracking wild animals, monitoring livestock and monitoring service dogs.

Tracking wildlife's geographical positions. These are the most common projects done as they enable studies in zoology (habitat environment observation and behaviour recognition [6] in wild animals) just by simply tracking the location of subjects in a given area (usually national parks), all of which use GPS systems. The concept of a WSN in animal monitoring was reviewed in 2016 [7] to show that it is also a very suitable solution for tracking the location of mounted sensors on animals in a natural park, using sensor nodes that aggregate information from segments of sensor devices. Reference [8] proved that tracking wild animals position in highly risk fire forests can aid forest guards in detecting fires epicentre within a three minute time window. It proves that IoT in animal monitoring enables helpful insight to professionals and their responsibilities: the system recognizes if the animals move in a distinctive manner (anomaly) and alerts the caregiver.

The second category is related to monitoring livestock for industrial purposes. In 2017, an information system was implemented to monitor vital signs, daily activities of cattle and alert veterinaries in the area in case of alarming values [9]. Many devices are used to monitor the cattle's individual well-being such as temperature sensor, three axis accelerometer sensor, heart rate sensor and humidity sensor, and a display. If the temperature of the animal is greater than 38,3ºC the display changes its content. WSNs are used also in reference [10] for behaviour pattern analysis in dairy

cows, using smart collars that track their position using an Inertial Measurement Unit (IMU). Other projects with the same objective use video monitoring to detect abnormal changes in behavioural patterns of cattle [11]. There was a study done [4] to build a system that monitors the heartbeat of horses through pulse sensors connected to a mobile app via Bluetooth, that stores the values in a cloud database and enables its visualisation through a website. Reference [12], demonstrates an implementation of a system that monitors animals in a zoo not only by tracking their position (if they leave their cages) through RFID (Radio Frequency Identification) and GPS, but also monitoring their well-being through temperature sensors (although this part is not very clear how they implemented it). There is also a thesis published by the University of Aveiro that describes an IoT system to monitor sheep herds geographical position to improve agriculture [13].

Finally, the third category is papers which represent studies and systems to monitor working and service dogs. A series of papers [14]–[16]describe in detail sensors and methodologies used on the creation of an IoT system to monitor service and working dog's vital signs, their environment, and improve canine-human communication on both directions. It is composed of three types of system: one for training SAR (Search and Rescue) dogs with little human interaction, an "intelligent" handle that vibrates according to environment and phycological state of the dog stimuli, for service dogs. And lastly, a SAR dog harness that keeps the dog safe with temperature sensors, gas sensors, GPS receivers, audio/video interfaces, and others. There was an article of 2016 in Tokyo that defines the system's design of a smart cage for small dogs that automate the process of cleaning the cages [17]. It as a camera that periodically captures images that are processed by an algorithm that identifies dejections and activates the mechanical tools that clean them. This is an example of how technology may help pet care takers at home or hospital in a more efficient way. Besides cleaning excrement/urine automatically, the system is capable of transmitting live images from the cage to a web app.

This set of papers contain important information about sensors, system architectures and tools, specific to dogs and the functions they are meant to perform. They are examples of systems which describe not only sensors that can be used, but also system architectures, processes, and tools that improve animal monitoring.

2.1.2 Key parameters in animal monitoring

Animal body-temperature may be measured by intrusive (by rectal, vaginal, vascular and digestive-tract sensors [18]) or extrusive methods (small hardware mounted on the animal skin/body). The normal temperature values for body-temperature in dogs vary between 38,33 Celsius and 39,3º Celsius. If it drops below 37,2ºC or rises above 40ºC we have fairly alarming values [19], depending on the breed/type. Puppies can vary a little outside these ranges. For instance, new born pups have a body temperature of 34.44ºC to 36.11ºC degrees and may not reach normal body temperature until they are about a month old.

For cats, the values are similar: between 37ºC and 40ºC are the normal values and in cats there is not much variation.

Temperature is very important in animal hospitalisation because it can translate the emotional state of the animal as well as an abnormal recovery [18]. Pregnancy in dogs usually lasts between 56 to 69 days, depending on the size and breed, through these values may change; the larger the animal, the longer it's pregnancy. When the temperature of the animal drops below 38ºC, she should deliver in the next twenty-four hours.

Heart rate or pulse are terms used to describe a clinical procedure that takes the number of heart beats per period, usually a minute. The result of this measure is called Beats Per Minute (bpm). It can be calculated with only two beats and the time elapsed between them. This way, it is possible to estimate a BPM value without having to spend a whole minute. Checking the frequency of the heart beats may tell us a lot about the animal phycological state (if the animal is calm it has low bpm values, or if it is enduring stress it has high bpm values). Unusual bpm records can also indicate many other conditions, such as blood loss, dehydration, fever and heatstroke. Tachycardia is a condition when an abnormally high bpm value is recorded. A slow pulse rate may be a sign of shock or heart disease. These conditions causes are pain, stress, congestive heart failure, shock, anaemia, infection, dehydration or blood clot.

Having said that, monitoring the animal's heart rate is critical to its well-being. According to veterinary doctors that collaborate with Pet Universal, we can catalogue cat's and dog's heart rates, in resting or relaxed state as shown in Table 1:

Species	**Size of Animal**	**Heart beats per minute**
Dog	Till 2-week-old Dog	160 – 220
	Till 1-year-old Dog	120 – 180
	Adult Very Small Dog	140 – 180
	Adult Small Dog	100 – 140
	Adult Large Dog	60 – 100
Cat	Adult Cat	120 – 220

Table 1- Beats per minute in dogs and cats

In general, the larger the dog, the slower the normal heart rate. And the same can be said about the respiratory rate. Dogs at rest have a normal respiration rate of 10 to 35 breaths per minute. The average dog at rest takes 24 breaths per minute. A normal, rested, non-panting dog should breathe between 10 and 35 times per minute. Smaller Dogs take between 12 and 24 breaths per minute. Dogs can also pant normally up to 200 pants per minute.

A normal cat takes between 20 to 30 breaths per minute, with a relaxed cat measuring on the lower end of the scale. If the rate is above 30, or if your cat is panting, it is alarming.

Blood pressure is a broad term to describe a clinical procedure that measures the amount of pressure the blood on the arteries' walls. It is usually measured with the heart rate. Doppler is the

most frequent and easy method of measuring the blood pressure. The result values of this measurement come in millimetre of mercury (mm Hg). The mean arterial pressure is calculated from a Systolic and Diastolic values, having the same normal values in dogs and cats, which is 60-100 mm Hg.

The following images (Image 1 and Image 2) show two equipment used in today's veterinary procedures.

Image 1- Doppler blood pressure measurement in a dog.

Source: http://www.vetfolio.com/todays-veterinary-practice/blood-pressure-monitoring-from-a-nursing-perspective-part-1-overview-of-blood-pressure-monitoring

Image 2- Feline patient connected to an oscillometer blood pressure monitor.

Source: http://www.vetfolio.com/todays-veterinary-practice/blood-pressure-monitoring-from-a-nursing-perspective-part-1-overview-of-blood-pressure-monitoring

Body temperature, heart rate, respiratory rate and blood pressure are the most frequent measurements taken in veterinary hospitals and clinics around Portugal, Spain and Brazil. But after the research done in scientific and commercial products, many more body and ambient metrics were found to be taken as well. Examples of parameters that have been sensed in dogs in the literature review are listed in Table 2.

Context	Parameter	Description
Cage	Ambient temperature	Taken from a thermometer inside the cage but no in touch of the animal. Measured in Celsius or Fahrenheit.
	Sound level/noise	Using a microphone inside the cage, it is possible to record the volume of the surrounding environment of the animal, high volumes or constant noise is highly stressful for a pet.
	Humidity levels	There is hardware that can measure the levels RH levels (relative humidity).
	Water/food level detection	Having a sensor on the bottom of the water/ food bowl checks if the animal needs replenishment.
	Cage weight	Smart weighing scale may be put below the cage and can be used for many purposes such as recognition of animal movements, or simply recognizing if the cage has an animal or is empty.
	Excrement detection	Either through camera recognition, weight scales or even certain gas sensors inside the cage, it is possible to indicate the presence of feces or urine inside the cage.

	Video recognition (Movement tracking)	Using cameras and tracking algorithms, it is possible to register not only the geographic position inside the cage but also the "geographical" or physical position of the animal (standing, seated, eating, standing on to legs, etc)
	Gas	Measures the presence of a certain gas, for example carbon dioxide inside the cage.
Animal	Body temperature	This measure can be divided into two: core-body-temperature and skin-body-temperature. Core-body-temperature is measured rectally and is a precise way to get the temperature of the animal. It is an intrusive procedure, unlike skin-body-temperature that can be measured by pressing temperature sensors against the animal's skin. It comes in Celsius or Fahrenheit.
	Heart rate	Heart rate measures the beats of the animal's heart per measure of time. Most common is beats per minute (BPM).
	Respiratory rhythm/frequency	This parameter is the count of the breaths (inhale and exhale) the animal takes per measure of time, usually a minute.
	Blood pressure	As seen in Image 1 and Image 2 there are different ways of measuring the blood pressure of an animal. Specific hardware is used to get two values: the systolic and diastolic values that come in mmHg (millimetres of Mercury).
	Saliva	Too much or too little saliva on a dog or a cat (for example) is a sign of distress.
	Growl or Barks frequency	Placing a microphone against the pet's throat is a simple way to record the frequency of its grows or barks. Too many vocalizations can be a sign of stress in a pet.
	Sweat level	Pets' skin is not like humans. The only place they "sweat" is between their paw toes. A humidity sensor can be placed there to monitor its "sweatiness".
	Posture patterns	Placing accelerometers along the animal body, and using recognition algorithms, it is possible to register the physical position of the animal (standing, seated, eating, standing on to legs, etc)
	Activity levels	The same accelerometers can be programmed to register the activity levels of the animal. If one is agitated or in stress it might move too much inside the cage.

Table 2- Monitoring Parameters in animal's cage and body

Many animals naturally have strict routines (like eating always the same amount of food per day and at a given hour). Smart monitoring system can use these parameters to cross-check if it's routines are being complied.

In Table 2, we describe certain types of sensors in the description. In the next section we define some of the sensors used in pets (cats and dogs).

2.1.3 Sensors for pet monitoring

Regarding the sensors placed on animals, there are huge amounts of devices that can be used to monitor them, and a lot of information can be taken from the literature review described above in section 2.1.1. This section intends to review the sensors used on those papers, specific to pets.

Smart collars is a term to describe animal collars that are capable of sensing some kind of metric from the animal, usually levels of movement or GPS signals. Smart collars are being commercialized for very specific purposes. Beacons installed in collars can be used to track the position of dogs and cats in a given area [13]. Beacons are close-ranged wireless devices with a unique id that uses Bluetooth radio frequency to communicate with other devices, usually a local "tower" receiver.

Other smart collars in the market use accelerometers to track the pet's physical activity throughout the day and present the data on a mobile app[3] (very similar to any fitness bracelet used for human fitness monitoring nowadays). Patterns such as sleeping, active or exercising are recognised, and the owner gets the feedback if his dog needs more, or less activity. Accelerometers are hardware that calculate the proper acceleration of a body, based on movement on axis. The last application of smart collar uses only one accelerometer, but if we gather 2 or 3 and set them along the animal body, we can define patterns of positions (laying down, eating/sniffing, standing on 2 legs, etc, [14], [20]. Inertial measurement units (IMU) may also be used to monitor the movement of the animal [16]. Accelerometers, gyroscopes and IMUs are specific kinetic sensors.

In general, the lower weight and volume of the collar, the better for the animal's comfort. However, there might be a trade off with processing power and communications capability [16].

Radio Frequency Identification (RFID) has also been used to track animal's location in urban areas [21]. Via close-proximity devices, the RFID tags on the stray's collars, give a sense of geographical position based on range.

According to [22], it is possible to build a biometric collar that measures the major vital-signs on dogs. Currently, in order to take an electrocardiogram (ECG) on an animal in hospitalisation, they undertake a clinical procedure where they have to shave fur out from some parts of the animal body, so that attached ECG electrodes may stick and not be prone to detachment due to the movement of the animal, and also to take clear readings. A project done at the North Carolina State University [22] propose a way to take an ECG skin reading without having to shave the animal. Using a photoplethysmogram (PPG) which is a volumetric optical measurement that is obtained by shining infrared light into tissue it is possible to detect the amount of light that is reflected to the photodiode

[3] https://www.fitbark.com/

that can be translated into heart beats. One of the cheapest photoplethysmograms in the market is Pulse Sensor[4].

Measuring temperature can be done with two main approaches: direct contact or radiation sensors [23]. Depending on the type of animal to be monitored, many types of sensor are best adequate. Most mammals are homeothermic (or warm-blooded) species, maintaining a stable body temperature and producing their own warmth. Direct contact sensors measure the body core temperature of the animal and radiation sensors (usually infrared) only take out the temperature of the skin surface.

There are huge number of contact temperature sensors in electronic devices/hardware, but they can be divided into two groups: thermocouples and thermistors [23].

Thermocouples are based on a thermoelectric effect that occurs when two dissimilar metals at different temperatures are close to each other, a voltage is produced, and the variation of that voltage can be translated into a temperature. These often emit errors, and often need re-calibration [23].

Thermistors are semiconductors from metal oxides that emit more precise and stable values of temperature [23]. As these sensors are analogic (thermoelectric variation), it is possible to convert their values into digital signals, which, in turn, can be translated into float values. They can be put in contact with the animal's skin therefore retrieving the value of the core body temperature of the animal.

In conclusion of chapter 2.1, animal sensing has been growing alongside zoology, veterinary and technological advances. The most advanced and innovative sensors as of 2017 are described in reference [18]. Ranging from simple sensors (movement and biosensors) to molecular bio/nano sensors which can detect various molecules and toxins in animals are being applied to livestock but can also be applied for pets. There are studies that present several disadvantages in the use of location and posture technology, namely in terms of energy consumption, precision and reliability, creating some barriers in its integration or application in some scenarios [14], [24].

2.2 Enabling technologies for IoT monitoring

This chapter briefly reviews the state of the art on IoT solutions and tools. In 2017, a series of a papers were published acknowledging many architectures and technologies in relation to IoT [25], presenting specific implementations for Big Data projects [26], and studying the integration of Cloud Computing in network architectures [27]. These articles served as a base for the following research and implementation.

IoT is not fully standardised, although there have been attempts from groups such as International Engineering Task Force (IETF)[5] and Eclipse IoT Working Group[6]. It all depends on the requirements of the project and its limitations. Moreover, "IoT system" is such a broad term, that there are many

[4] https://pulsesensor.com/
[5] https://www.ietf.org/
[6] https://iot.eclipse.org/

parts that it can be divided into, and even those are not linearly divisible. However, it is possible to make general comparisons and find consistency throughout the many aspects of IoT. This chapter will describe this general consistency and describe its parts.

The IETF is an open community that develops and promotes Internet standards. It is a group of researchers from public and private entities that gather periodically to review Internet state of the art, as well as convey discussions concerning the evolution of the Internet. IoT has been the focus of many of those discussion in recent years and since 1993 it has operated as a standards development.

With the goal to create an open IoT, many organizations and experts frequently get together in conferences. They form the Eclipse IoT Working Group, a collaboration that focuses on the development and promotion of open source IoT technology.

In recent years, the focus of developing IoT solutions has been directed for selling full-IoT Platforms (*ThingsBoard*)[7], followed secondly by Home Automation solutions and thirdly by Industrial Automation solutions[8]. eHealth is not yet the main focus of IoT investment due to the long-term nature of its return value, but solutions in this field are more and more common.

A consequence of this may also be found in the lack of research and test fields done in animal health monitoring systems (AHMS). This makes developing an AHMS more challenging to plan as there are no standardisations in IoT for animal eHealth or eHealth in general. Also, as we saw in the last section, there are lot of technologies mentioned but none specific for monitoring animals in hospitalization.

2.2.1 General view of IoT architectures

To understand the pipeline of information that is the basis of any IoT project, we first must talk about what a streaming pipeline is. A stream is an unbounded (of unknown or of unlimited size) continuous sequence of data sent from a transmitter to a receiver. The pipeline (represented by the long arrow in Image 3) is the description of that data flowing through the system. In a broader view, the literature review divides architectures in three main components [25]: the sensors/devices (perception layer), the system's gateways and server(s) network (network layer) and the software which will process and present information to the users (application layer). Each represented by a colour in Image 3Image 3.

In a typical "end-to-end" monitoring IoT architecture (from data gathering to information presentation), there would be many sensors at its foundation, in a physical area or a segment of the sensing layer (leftmost arrow in Image 3), generating and sending data periodically. As a consequence of such volume and varied sources, the presence of a local aggregator for each group of sensors is critical for the good routing and handling of the raw values [28]. This aggregator creates communication protocols between each sense point (device that has sensor(s)) and collects all the

[7] https://thingsboard.io/
[8] https://iot.eclipse.org/

messages sent. This aggregator then reads the messages and extracts the raw values of the sensors, creating the periodic message to be sent to the IoT system in the established format and in the form of a stream. At this point the aggregator serves as a gateway between the sensor layer and the network and transport layer (middle arrow in Image 3). The data is transported through computer networks using one or more protocols to a processing layer where it is analysed, stored and shared to the final users, usually in the form of dashboards. This is the processing and analysis part of the system, also called application layer, and it is usually done centrally in a server or a dedicated device, but it can also be distributed in a network (three rightmost arrows in Image 3), each having powerful software or technologies with a specific task. The first always involves the collecting of the messages on one or more ports, usually using a Message-Oriented Middleware (MOM) approach with the usage of a message broker for the streaming messages. They are responsible for routing the messages at a given time in the pipeline either for the database (storage) or to the next phase which is analysis. This segment is responsible for processing, logging, translating, and in many architectures, send alarms in case of anomalies. The last block refers to the availability of the data in a visual and searchable way, which enables the user to extract the information required.

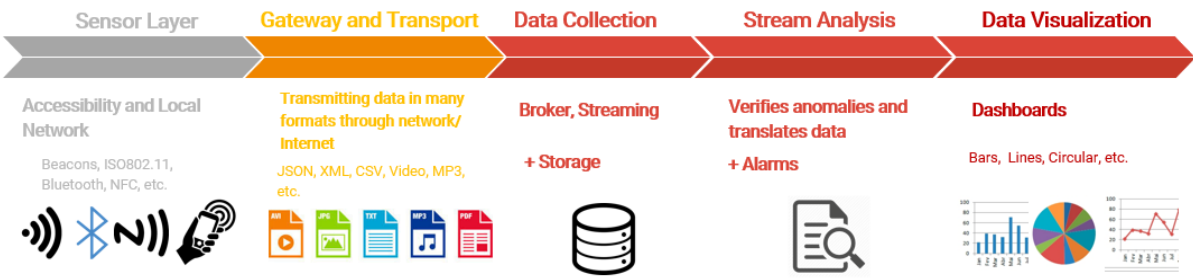

Image 3- General IoT monitoring architecture

These architectures are also called “event sourcing architectures”, “event-driven architectures” or even “message-oriented architectures”, and they focus on optimising the transport and processing of messages through a network so that the user can have a near real-time monitoring over a given subject or group of subjects. These messages are treated as events as they occur periodically or sporadically and represent a real-life event. As each message is produced at the pipeline’s start, a series of processes and technologies must guaranty that that same message gets to the pipeline’s end as quickly and as integrate as possible (it may not be in the original order or the same content, depending on the system’s objective).

This pipeline is rarely as sequential and flat as described. Often it has many processes that correlate with previous blocks and technologies (for example brokers and databases).

A study done by Eclipse[9] in April of 2018 shows that the most used programming language for device configuration (sensor layer) is C, for gateways (transport layer) is Java, and cloud applications (application layer) is also Java [29]. Furthermore, the same study shows that 71% of operating systems used in IoT are Linux-based servers [29].

[9] https://www.eclipse.org

2.2.2 Sensing layer

This layer is composed of hardware and communication protocols that make the bridge between the real-world parameters to be sensed and the creation of messages in a digital form. Sensors are devices that measure parameters of the physical world and interpret them into an analog or digital value, so that they can be transported and handled by code or other devices.

There are many ways to design an IoT sensing layer. If, for example, after gathering the values, each device sends data directly to an aggregator/ gateway, this layer uses a “Device-to-Gateway” architecture. There are some architectures where each sensor may directly send data to the cloud processing server called “Device-to-Cloud” architecture, and others where devices send data to each other called “Device-to-Device” architecture creating a Wireless Sensor Network (WSN).

The sensor device may be already able to connect to an aggregator or it must be configured (or wired) to do so. The communication between sensor device and gateway can be made through many communication protocols. There are wired and wireless communication protocols. The main wireless communication protocols used in IoT are: ZigBee, Bluetooth, RFID, IEEE 802.11, Beacons, NFC, among others.

Bluetooth is a common option to send data from devices to nearby aggregators. The short range of Bluetooth devices means that users must be in close range be able to communicate using this technology. This technology is often used in eHealth as it allows mobility and manoeuvrability to its subjects, and because devices establish a secure and consistent tunnel between sensing devices and the aggregator, being a lot more efficient on an energy consumption point of view.

When this data is gathered in a given timeframe, the aggregator usually serves as a gateway for the network layer. This local aggregator is the end of the sensor layer and the start of the network layer. There could even be many local aggregators routing messages to a main aggregator, and that would be part of the network layer. The aggregators are responsible for collecting and sending the data to the network through a specific protocol and in a pre-established data format (For example, sending JSON messages using Message Queue Telemetry Transport Standard protocol).

In 2015, a survey was made in order to enumerate some state of the art projects in IoT [28] and describe the efforts made for the standardisation of it into layers. It is important to mention that the Message Queue Telemetry Transport Standard (MQTT) standard is frequently mentioned as a specific telemetry standard for IoT projects and is the most used in the network layer.

MQTT is a messaging protocol that uses the publish-subscribe (pub-sub) model. The pub-sub model will be mentioned in the next section. MQTT is well suited for Device-to-Cloud architectures as it uses compression of the messages on a byte level. It relies on the Transmission Control Protocol (TCP) to transport data through the Internet, and it is a “fire-and-forget” pattern, which means that there is no guarantee of delivery, thus the state is not preserved making it a lightweight pattern. This pattern refers to the minimal quality of service (QoS) where the recipient does not acknowledge reception of the message.

IETF suggests some standards when building an IoT project besides MQTT. Among them are the following which are also widely used: Hypertext transport protocol (HTTP) and Advanced Message Queueing Protocol (AMQP).

HTTP is a communication protocol used as a standard for the Internet. It is based on the relation of hyperlink nodes of the internet and transfer of files. In 2015 a more secure and faster version, the HTTP/2 is now supported by major web servers and browsers but is still slower than MQTT.

AMQP is a message protocol that is often used in MOMs and like MQTT is uses streams of bytes to transport data. It is more powerful than MQTT as uses a much more complex and variety of features like routing and queuing, but this makes AMQP heavier and slower.

2.2.3 Aggregation, Processing and Visualizing layer

The application layer is the set of procedures that are responsible for routing, analysing, and parsing according to rules and algorithms coded. It must arrange data so that data can be presented visually and/ or create alarms. The following paragraphs will explain these procedures in the order described.

Message brokers are dedicated software that routes messages. A message broker is responsible for the distribution and sometimes temporary storage of messages that are sent to it. They are a piece of software that can be configured to behave according to the message's content. The way brokers route or store messages vary, but usually follow the publish-subscribe pattern: a publisher (or producer) is a programming class that generates messages and pushes them to the broker. A subscriber (or consumer) is a class that listens to the messages that it is interested in. The way it works is based on topics. Topics are "channels" the producers publish to and where consumers subscribe to, topics is a specific type of structured data. The producer sends messages to one or more topics independently of whether there are any subscribers to that same topic(s), these messages are temporarily and chronically stored, for the subscribers to retrieve them. Whenever the consumer subscribes to a topic, it receives all the messages that that topic contains.

Lambda architectures, illustrated in Image 4, use a dual method of batch and stream-processing to handle huge amounts of data and process it. The batch data model ingests and processes timestamped events/messages reading the messages and storing them temporarily, in order of entry. After that, the tool that implements this architecture, passes them to the speed layer. The speed layer reads the data and makes decisions based on programable algorithms, sending it to the next phase which can be a database, a file, or another application.

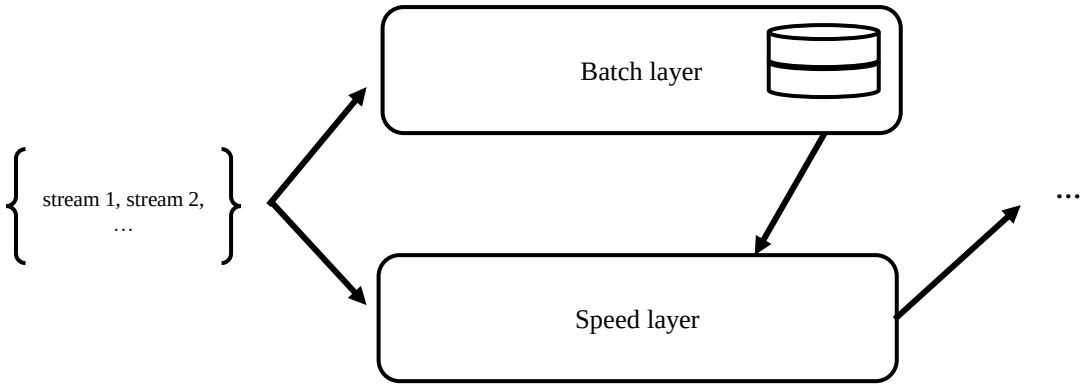

Image 4- Lambda architecture schema

The batch layer has a high throughput, using a distributed processing system with little latency. It aims to accurately process large quantities of data by generating temporary views from that data. This translates into a system without data loss that stores messages in a read-only database temporarily, updating existing views. It is also in this layer where rule-based algorithms are implemented (decision making). Apache Hadoop is becoming a standard batch-processing system used in most architectures. After being precomputed, data is sent to the speed layer.

The speed layer is responsible for providing views based on the most recent data. They are available almost immediately after data is received and can be replaced when the batch layer's views for the same data become available. Stream-processing technologies typically used in this layer include Apache Kafka Stream, Apache Storm, SQL-stream and Apache Spark.

There are a great deal of open-source or free brokers, and from the ones that explicitly work with MQTT protocol the following are the most used: RabbitMQ[10], Apache Kafka[11], Eclipse Mosquitto[12].

RabbitMQ[13] is one of the most used message broker in IoT projects [30], as it is compatible with most programming languages and protocols, due to its maturity in the market. It is mostly opensource offering distributed deployment, cloud services, plugins, etc. It also uses the pub-sub model with topics and was originally created to support the AMQP protocol. Publishers send messages to exchanges (pieces of software that route them), and consumers retrieve messages from queues

[10] https://www.rabbitmq.com/
[11] https://kafka.apache.org/
[12] https://mosquitto.org/
[13] https://www.rabbitmq.com/

(topics). These exchanges facilitate the process of routing the messages accordingly. Image 5 illustrates this tool's data flow.

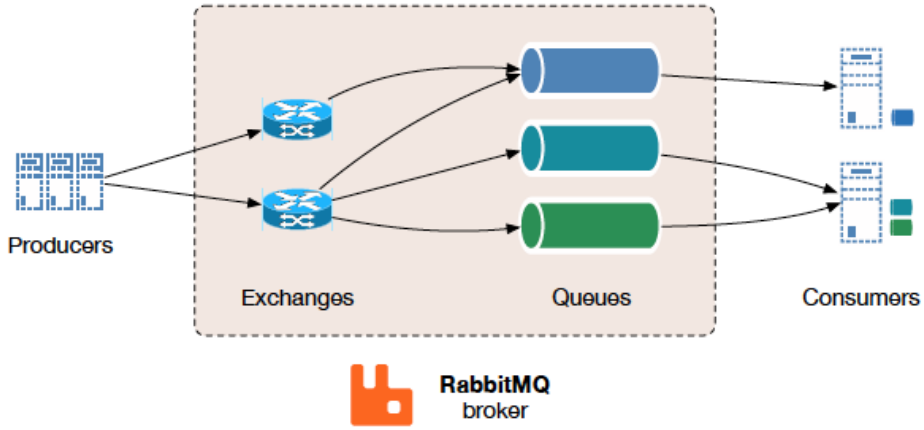

Image 5 - RabbitMQ layout
Source: http://kth.diva-portal.org/smash/get/diva2:813137/FULLTEXT01.pdf

Apache Kafka is designed for high volume messages and streams. It uses the pub-sub model in a durable, fast, and scalable way. Kafka makes use of another Apache tool named Zookeeper to store (similar to a logger) the messages that pass through. It runs in one or more server clusters that manage topics.

Every message in Kafka is viewed as a key-value. In contrast to RabbitMQ that tracks which messages were read by each consumer and only retain unread messages, Kafka retains all messages for a set amount of time, and consumers are responsible to track their location in each log (consumer preserves the state). It also makes replicas of the nodes it creates so that data is always recoverable and accessible. Image 6 shows the tool's architecture.

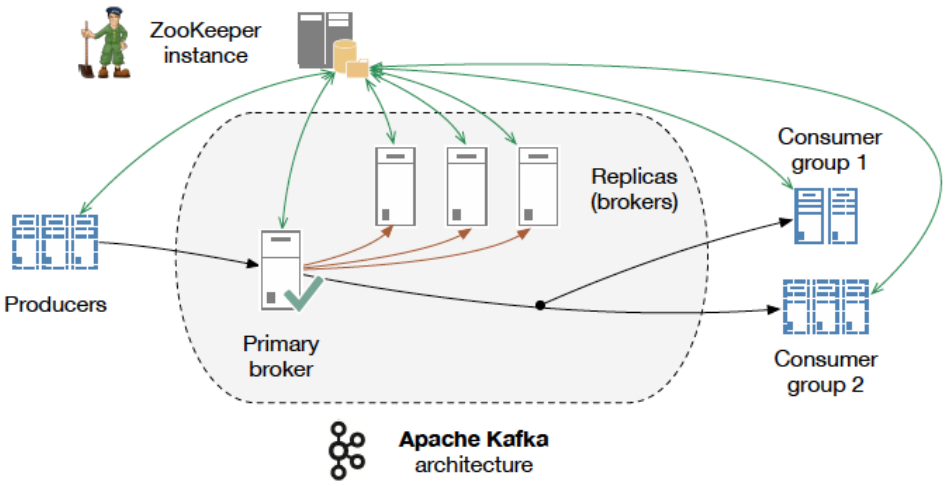

Image 6- Apache Kafka architecture

Source: http://kth.diva-portal.org/smash/get/diva2:813137/FULLTEXT01.pdf

Some of these brokers enable stream analysis within the tool. This is the case of Apache Kafka Streams that has been developed by an Apache spin-off called Confluent[14].

Confluent is trying to turn Kafka into a distributed streaming platform for Web and Mobile Apps, microservices, monitoring services, analytics, and more. It serves as a casing around Kafka through easy to follow "connectors" that enable developers to manipulate the way data enters and exits. These connectors simplify the configuration and integration of tools in a stream pipeline. It offers API's such as Producer, Consumer and Stream that are universal to all connectors.

The Stream API consists on the manipulation of the KStream object. This object is an abstraction of a record stream and allows the mapping of data records into a self-contained unbounded data set. Using the table analogy, data records are always interpreted as an insertion and each record adds rows with a different key. Another concept is the KTable object which is an abstraction of a changelog stream. Each new data record represents an update of the last value for the same record key. These API's allow stateless transformations like filter, map, foreach, group by, peek, select key, etc and stateful transformations like aggregating, joining, windowing, etc.

Eclipse Mosquitto is an open source iot.eclipse.org message broker that is specialized in using the MQTT. The Mosquitto project offers great documentation on implementing MQTT clients, as well as a useful command line UI.

In the application layer, after data analysis and stream processing, the data is ready for the next step which is visualization. Visualization is where the data is translated into information to be presented to the user in a readable and interactive way. Most often in monitoring systems, a dashboard or custom UI is shown. This is where the user must be able to quickly get feedback on the current state of the system and perform commands to interact with the system and its monitored subjects. There are many tools that can create a dashboard out of streams with intuitive and simple configuration. The most known are Grafana[15] and Kibana[16]. They are both open source tools that helps users create and change panel views with many sorts of graphical aids to present stream data (example: line graphs, bar graphs, tables, maps, etc.).

Grafana is one of the most referenced tools for visualization in monitoring and stream visualization. It offers good attention to detail, and it is used in computer monitoring systems in companies like PayPal[17] and eBay[18]. One of its main advantage is the integration with different backend and the focus in time series data.

[14] https://www.confluent.io/
[15] https://grafana.com/
[16] https://www.elastic.co/products/kibana
[17] https://www.paypal.com
[18] https://www.ebay.com

Kibana is integrated into the Elastic[19] stack which is a series of tools that form a simple stream pipeline more focused in monitoring computer systems through log streaming. Kibana is the "final" tool of that stack providing search, very thorough manipulation of the log messages, and a variety of graphical views. Many companies such as Netflix[20] and LinkedIn[21] use Elastic to manage and monitor logs.

Table 3 compares these two tools in a set of important features.

Feature	**Grafana**	**Kibana**
Many users	Different access user's privileges are allowed for free.	"Shield" must be bought to allow user differentiation.
Graph manipulation	Better manipulation.	Manipulation is not as powerful.
Learning curve	Integration is not always straightforward. Although it has a much more powerful interface.	The integration with Elastic is intuitive and easy to implement.
Focus	Temporal logs (containing timestamps).	More types of logs.
Enables alarms	Yes.	Yes, and it has integration with more tools.

Table 3- Comparison between visualisation tools

Both tools are great and can even be complementary as Grafana is compatible with Elastic stack. Its graphics are defined using a personalised manipulation that specify the source and content of each message.

2.2.4 Persistence layer

Persistence is the outcome related to the long-term storage of the data which makes it continuously available in a consistent form. A database is an efficient and secure way to get accurate queries to the data, ease of updating data without the risk of losing information when many data points are accessed at the same time (integrity and isolation during concurrent accesses).

Many IoT systems use time series databases to explore the time dimension in data. Apache Cassandra[22] database is a timestamp-oriented database owned by the Apache Foundation. Cassandra

[19] https://www.elastic.co/
[20] https://www.netflix.com
[21] https://www.linkedin.com
[22] http://cassandra.apache.org

is a distributed database management system and is a reference as a NoSQL (non-relational) database in time-series projects. It offers continuous availability (no point of failure), good performance, strong security, scalability and operational simplicity. Written in Java, originally built at Facebook and open-sourced in 2008 and bought by the Apache Foundation in 2010. Cassandra offers features of both key-value and column-oriented data stores and is very generic, meaning that it is not developed to fit into a certain type of project or specification.

Mastering a few concepts is needed to start working with Cassandra, such as: Cassandra Query Language (CQL), column families, wide rows, compact storage, partition keys, and secondary indexes. However, its learning curve is not as steep as other opensource database systems. Cassandra allows storage of messages in a wide range of formats, for example, we can tell Cassandra to store a JSON type message, and just map the fields we want to store.

In CQL, a parallelism is done to SQL where data is stored in “tables” containing “rows” of “columns”. The query “keywords” are very similar (SELECT, INSERT, UPDATE, DELETE, etc).

CQL supports 3 types of collections: Maps, Sets and Lists. This facilitates when inserting data when the developer or system architect does not care much about how data is stored. It is also possible to define a type structure by the developer, which is useful when the system has very uniquely structured data.

The tables are grouped in “keyspaces” which identify the group and enable commands to be applied to all tables contained. It is advised to use one keyspace by application, to create a hierarchical and faster way for commands to be passed to the many clusters. Each table is assigned to at least one keyspace and this is mandatory.

This is a powerful tool that enables full documents to be added into a database. With Cassandra 2.2, JSON support was added for keywords such as SELECT and INSERT.

With INSERT statements, the new JSON keyword can be used to enable inserting a JSON encoded map as a single row. The format of the JSON map should generally match that returned by a SELECT JSON statement on the same table. For example, to insert into a table with two columns named “myKey” and “myValue”, it is as easy as:

```
INSERT INTO mykeyspace.mytable JSON '{ "myKey": 0, "myValue": 0}'
```

Configuring and operating Cassandra is accessible to not very experienced developers and it stands as a good option in most types of timeseries projects, even with plugin and scripting integration.

2.3 Related work and products

According to a market study by Pet Universal, the most used veterinary management systems in Portugal are Qvet[23], WinVet[24], Marvet[25] and OranGestVet[26]. Estimates show that Qvet has around 45% of the market share, WinVet 30%, Marvet 15% and OrangeVet has 10%. Of these, only Qvet and OranGest Vet have hospitalization modules. According to Qvet and OranGest Vet's clients these modules are basic and limited, which makes them not making use of these features at all. This is the opportunity Pet Universal saw to complement the existing offer with a modern, intuitive and complete hospitalization solution. To the best of our knowledge, none of these two competitors are making an IoT system to monitor animals in hospitalization.

Qvet is a software for veterinary businesses that covers both management and hospitalisation parts. However, its main shortcoming drawback is that it does not offer any kind of registering for monitoring pets staying in hospitals. Qvet was recently bought by Logismart and is available as a SaaS globally. Qvet is a complete veterinary clinic and hospital software, currently in the "Spanish speaking" market, having more than 5,000 clients across 30 countries. Image 7 shows the hospitalization screen of their software.

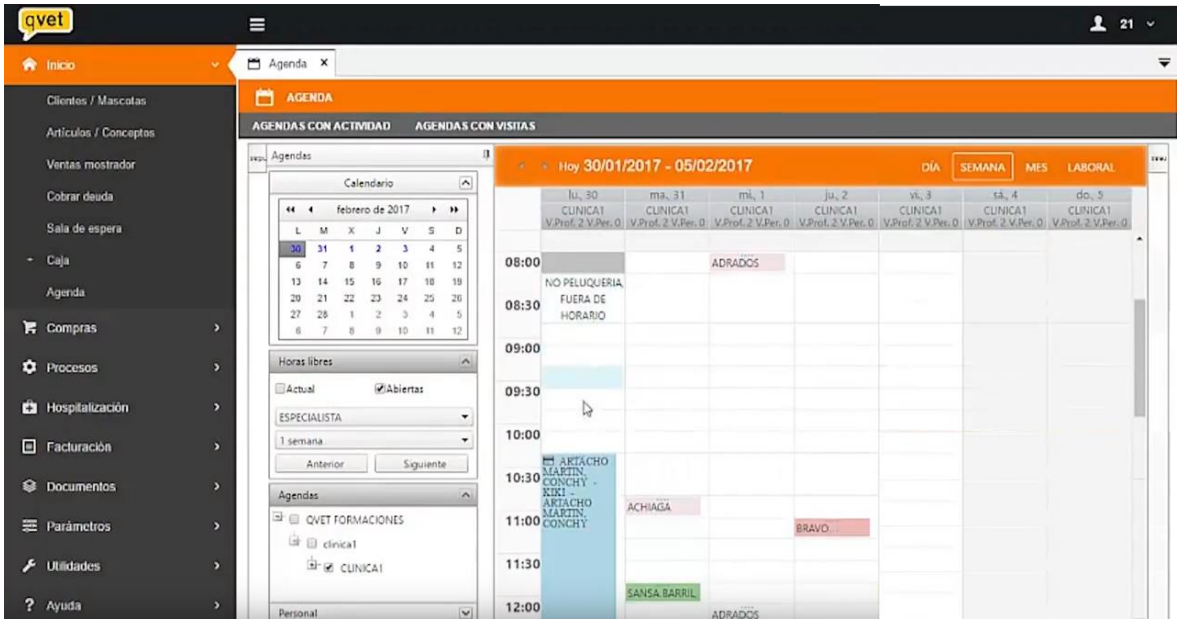

Image 7- Qvet hospitalization view

Source: www.qvet.net

OranGestVet by Magninsoft is a management and hospitalization software for veterinary businesses, also a SaaS. It has simpler views of the hospitalization episodes and tasks, it is only

[23] http://www.qvet.net
[24] http://www.winvet.es/
[25] https://www.logismart.pt
[26] http://www.magnisoft.pt

compatible with Windows. OranGestVet offers not only an online SaaS version but also a Windows native application. As for the hospitalization module, veterinaries complain that it is too basic and incomplete. This company focuses on providing the most wide solution (with a lot of modules) and not focus on each one for improvement.

The opportunity to be the first vet software company to implement an IoT system for monitoring animals would greatly benefit Pet Universal to gain leverage and market share. Outside the direct competition of Pet Universal, there are companies and products that are developing monitoring solutions for pets.

In an attempt to modernise the way owners monitor their pets and communicate with their veterinary professionals, one Portuguese company named CardioCão[27], developed a mobile application that was able to establish a personal communication between pet owners and veterinary professionals and also offers a way to monitor the respiratory and cardiac rates of pets. This mobile app interprets the human physical touch on the screen of the smartphone as animal heart beats. in which is an innovative way to enable animal vital signs readings. These readings are stored and presented to the user as a monitoring system that can be sent to a veterinary, whenever and wherever the user wants to.

PetPace[28] is a company created by vets in the United States of America, which monitors cats and dogs constantly with the use of smart collars. The product is composed of an Android Mobile App on the Google Play store, patented collar, and a corresponding gateway. The system collects biometrics such as temperature, heart rate, respiratory rate, position and stance, and even estimates Calories burned based on movement. It sends real-time data to the owner's and veterinary's smartphone with alerts in specific abnormal cases. Its architecture and functionalities are very similar to the objectives of this thesis, and the collar is the first of its kind for monitoring pets.

Another mobile App that gathers information about the pet is PetDialog[29], although in this one, the user has to put the values manually and is not automated (only for diabetes cases). PetDialog allows users to record exercise and nutritional intake, socialization and other activities, it also alerts for routine care such as vaccines or other planned activities. It is intended that users on their smartphones are able to track and send animal behaviour to veterinarians faster and more accurately.

These products provide insights on the needs of the vet and pet care markets and ideas of how the systems may be approached and developed. Relative to the literature review on science papers and theses, two relevant projects were found: an IoT system to monitor pets' health on a cannel, and "SheepIT", a University of Aveiro's thesis.

As a very relevant reference [31], a paper was published at the National Ilan University in Yilan, Taiwan, that intended to show that monitoring physiological and environmental metrics in an animal shelter may help increase the adoption rate of strays in the country. Using IoT technologies, the system consists on a WSN, with Arduinos, RFID and other sensors, and Cloud Computing, that

[27] http://www.cardiocao.pt/
[28] https://petpace.com/
[29] https://www.petdialog.co.uk/

transmits information to cannel managers and caregivers, providing decision assistance in case of emergency. The idea is to have evidences that the animals are healthy and well treated so that people feel more compelled to adopt, thus reducing the necessity of euthanize the animals. RFID tags are used to identify the stray animal, an Arduino single-board microcontroller and sensors are used as smart collars that build an WSN, where data is collected. The monitoring website shows the messages retrieved from the collars periodically requested from the database every 30 seconds. Every message is then calculated in an algorithm that attributes a colour to each row, if it normal blue is attributed, if it is an error message is comes in yellow and red is for alarming values. Ambient temperature, ambient humidity, and heart rate (ECG) is monitored and it is possible to visualize these values in a timeline graph.

There is a project developed in *Universidade de Aveiro* called SheepIT [13]. It is an IoT solution to monitor and manage sheep herds in viniculture fields. The paper contains insight on the overall architecture and specifically about MOM patterns, processing and analysis tools, as well as the alarmistic specifications used. They used collars that connect to beacons and transmit the relative location of each sheep in the field and sends that data to a server on the Internet. That data it processed using RabbitMQ and Apache Spark solutions and displayed to administrators.

3 PET SENSE USE CASES

Pet Sense is the name of the project of this thesis. The name derives from the Pet Universal name and the project's main function which is to "sense" data from hospitalized animals. As explained further, Pet Universal offers a set of modules each with a specific purpose, and Pet Sense is one of those modules.

Pet Sense is a system to gather values from various sensors placed on hospitalized pets, monitor their vital signs for potential hazard conditions, lowering the human intervention to a minimum. The values should be presented in Pet Universal software as well as the Mobile App.

The chapter describes the hospital stays processes, the prior software suite from Pet Universal, and the Pet Sense use case.

3.1 Animal hospitalization activities

To get a better understanding of the daily veterinary hospitalization activities, the following paragraphs describe what already exist in Pet Universal's clients and what the veterinary requirements are.

In a standard hospitalization episode, the pet owner arrives at the veterinary hospital and is escorted into a waiting room by arrival order. The veterinary receptionist registers the arrival in the Pet Universal software (calendar and scheduling software) completing the check-in and indicates the client to follow to the doctor's office when the vet is ready. The consultation appointment starts with all the physical examinations as the doctor checks the patient up as he sees fit. The doctor then registers all in the appointment issues and animal status on the Pet Universal software associated with the private client and patient sheet. If the dog needs surgery and/or recovery monitoring, the hospitalisation software module is initiated to register all the animal stay tasks to be made with the recovery calendarization. If not, the pet owner returns to the reception, or waiting room, paying before leaving. Image 8 illustrates these procedures in UML form.

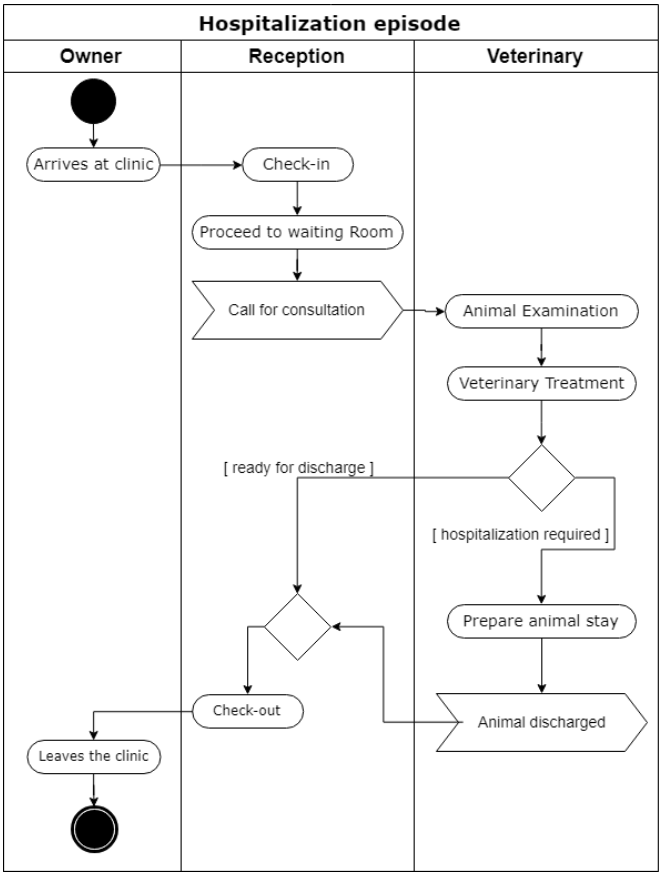

Image 8- Procedures on a veterinary hospitalization

After the pet is put to surgery or has taken medicine, the clinical stay is created with the information about the animal, the medicine administered and owner info. Medical tasks are added to the hospitalization episode, either periodical or exceptional, to be administrated on the patient throughout. They can vary from drug or serum administrations, manual tasks (clear cage, feed, etc), execute exams, and monitoring (vital signs check-ups).

This last process is the one which takes the longest to register. It is usually composed of four check-ups: physical exams (temperature, heart rate, blood pressure and respiratory rhythm), urine, feces, and vomiting. To take the temperature with a thermometer, at least 20 seconds are spent in animal manipulation, measurements and calculations; for the heart rate, using a stethoscope, another 20 seconds; for the blood pressure, another 40 seconds; respiratory rhythm takes about 5 seconds just by looking at the animal; all together it is about a minute and half spent on only one animal; if the clinic has 7 animals (the norm) there will be 10 minutes (at least) spend on one episode of monitoring. It comes as no surprise that physical monitoring is not so frequent as veterinaries would want, usually making three or less check-ups per day.

For example, the largest veterinary hospital in Portugal is *Restelo* and it has capacity for 60 hospitalised animals. Which means, if full capacity is achieved, 90 minutes per day per monitoring event. This is a great deal of time and expenses in staff. And usually is cut back for the same reasons.

Most veterinary establishments have an LCD monitor with a calendar-like view from Pet's software where they can see every animal currently hospitalised and the "task" ordered by time that they need to be done. Each "task" has at least three states represented by colours as Image 7 shows: scheduled (orange if it is on the current hour or colour grey if it is in the future), overdue (red) or done (green). Because of this fact, there is infrastructure for a monitoring sensor system and the costs would be minimal. A dashboard besides the calendar will greatly improve the "task" of the caregiver and for the Pet Universal perspective would not cost a thing, increasing sales and incrementing to its product's repertoire. Image 9 demonstrates the calendar view in the hospitalization module of Pet Universal software, "HOPI".

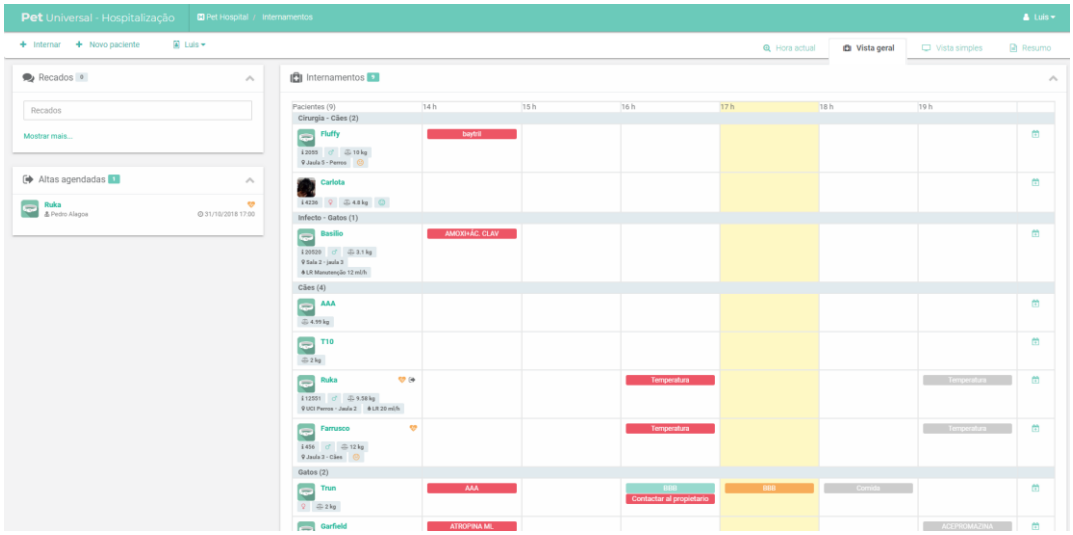

Image 9 - Calendar view of Pet Universal hospitalization module

The physical environment of pet vet facilities that have hospitalisation, generally have reception area, one or more vet offices, an open sky area for the animals, monitoring kennels, operating room, staff rooms and storage room. Animal stays are normally done indoors where animals spend most time in cages but eventually go to the "open sky spots" to exercise or excrement in a controlled environment.

3.2 The existing Pet Universal Platform

Pet Universal, or LVS S.A, is a technological start-up funded in May 2016 within the University of Aveiro. Created by former students of *Universidade de Aveiro*, two of which studied in *Departamento de Eletrónica, Telecomunicações e Informática* (DETI), Pet Universal is now a global company having clients in Portugal, Spain and Brazil. The core business of the company focuses on the development, integration and implementation of information systems for veterinary or pet establishments. It is a Software as a Service product that supports the management of veterinary organizations, clinics or projects. The company's main product (Pet Universal Platform) is a set of software modules for the veterinary sector as Image 10 illustrates, continuing to improve it and extending with new features (for example: billing, client management, etc).

Pet Universal started by selling only *Pet Hospitalização* module, growing in the Portuguese market these last 2 years; it was this year that it has been trying to increment its product, expanding it into an "all-in-one", "one-is-all-you-need" software for veterinary companies with the rest of the modules, depending on the client subscription package. Pet Sense aims to be a component in that software package.

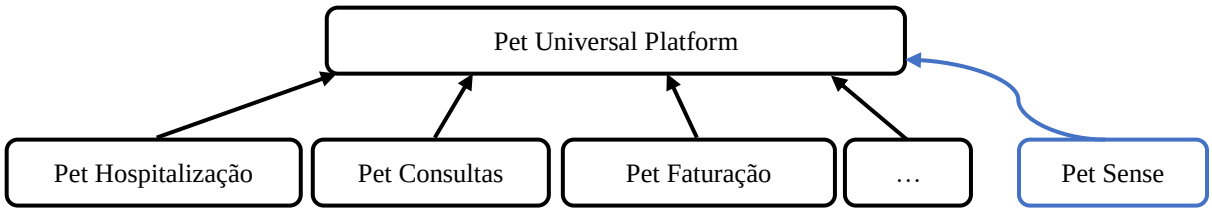

Image 10- Modules offered in the Pet Universal's platform.

The company values include continuous innovation to meet the needs of veterinary agents. The members that make up the Pet Universal team have a combined experience that is relevant to mentor or assist this project. In addition, Pet presents a valuable proposition as close as possible to the needs of the market and its final consumers, which are veterinary staff. The main features of *Pet Hospitalização* are: Scheduling of treatments for interned animals, storage of each animal's information, keeping track of tasks for hospitalised animals and all their attributes.

On the technical implementation, Pet's frontend and backend are built in Ember.js and is integrated as a cloud system. Ember.js is an open-source JavaScript web framework that allows developers to create single-page web applications by incorporating common idioms and best practices into the framework.

Pet Universal is a modular solution. The access to each module may vary with user permissions and with the subscription with Pet Universal. In Image 11 a full subscription is shown with the modules: Hospitalization, Staff, Patients, Customers, Appointments, Waiting room, Consultations, Laboratories, Imaging and Tasks.

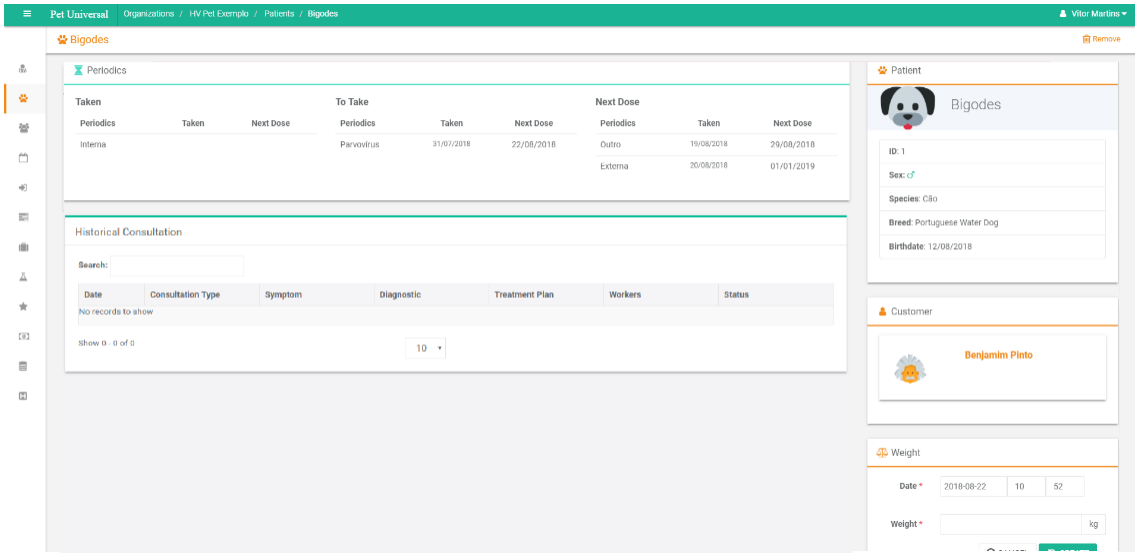

Image 11- Patient view of the Pet Universal software

The Veterinary Hospital of Aveiro (HVA) is the main veterinary partner and likely the first end-user of Pet Sense. It has the practical know-how to validate the product and contribute to its improvement. In this way, HVA provide the costumer perspective, enhancing the scientific and technical knowledge of Pet Sense, understanding the needs of end-users, in terms of control software, usability of wearables and hardware as well as its functionality and feasibility.

3.3 Functional description and use cases

Concerning the prior context, there is room for improvement in hospitalisation. After gathering the veterinary procedures, the planning of this project was thoroughly debated resulting into the functional requirements. The following list was created in collaboration with veterinary professionals, Pet Universal's staff and teachers from DETI:

- Collect real-time data from sensors in animals. Using the equipment placed on the animals, the system should enable a precise and regular monitoring of the hospitalised animals, without requiring constant care from veterinary professionals, in other words, the system should be automated. The sensor data should be sent in small intervals of time as a "stream" to the system.

- The data shall then be transported to a server or virtual machine through the Internet. Processing and forwarding the stream according to the animal type and sensed values.

- Store the data of each animal. The system should enable a way to extract the vitals' metrics history of each animal in its previous treatments, which means that there must be a database and webservices to extract data from.

- Visually present that data as information on a dashboard and allow basic interaction. The system should translate the data from the sensors into information readable and understandable by the veterinaries or caregivers. A dashboard with graphs and queries is required at this stage to provide the interaction layer.

- Alert the caregivers about alarmistic values in animal sensors. By allowing alarms to be sent to veterinary staff when irregular values are recorded, ensures that as soon as there is a potential threat to the animal's well-being is noticed of as soon as possible.

Pet Sense must be apt to handle different domestic animals (mainly dogs and cats) of different breed and size, within the hospitalization context, before and after receiving veterinary treatment. Another relevant question is that, the monitoring is constant (24h, 7 days a week), so the system should have continuous operation.

It is possible to infer from the requirements that the final user of this "product" are veterinary professionals whether they are veterinaries, vet nurses, or members of pet establishments that care for pets. Each interment has one or more people in change that gets the notification in case of an alarming event. Therefore, the system should enable the "full" view of all interned animals because it may be used by many caregivers at the same time.

The visualization of graphs and alert messages, built onto Pet's software's frontend, is to be displayed on the LCD monitors are typically located on the main corridor of the staff part of the clinic/hospital. In addition to the animal view as Image 11 showed, a monitoring dashboard is to be displayed. For each animal, there must be three visualisation areas representing the last/ current temperature and heart rate, another one where all temperatures and heart rates registered may be consulted (animal's history) and another with alarming values in a highlighted manner.

Apart from the LCD display, the system is also ready for all format displays such as smartphones or tables, that regularly are used by any veterinary personal during the day, to interact with the system.

The Android app simply lets the user log in onto the Pet Universal software, list all the patients in hospitalization and its information, subscribe to notifications/alarms of each one, and send monitoring values manually.

Pet Sense use cases are depicted in Image 12, Image 13, Image 14. Table 4, Table 5 and Table 6 describe the use cases in more detail.

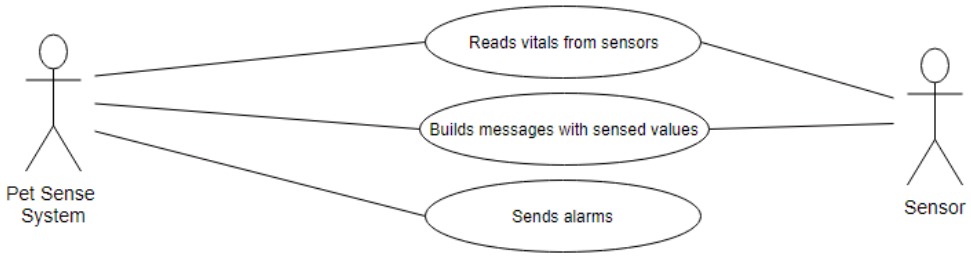

Image 12- Use case diagram of Pet Sense system

Use Case	Description
Pet Sense system reads sensor metrics.	When the system is set, it should be able to read values from sensors periodically.
Pet Sense system constructs messages.	The system should be able to build a JSON message with the values of the sensors.
Pet Sense system sends alarms.	The system should be able to send alarms to the smartphone and the front-end of Pet Universal.

Table 4- Use case description of Pet Sense system

In a system point of view, the data is extracted from sensors, a message is built, and alarms are sent with no human intervention.

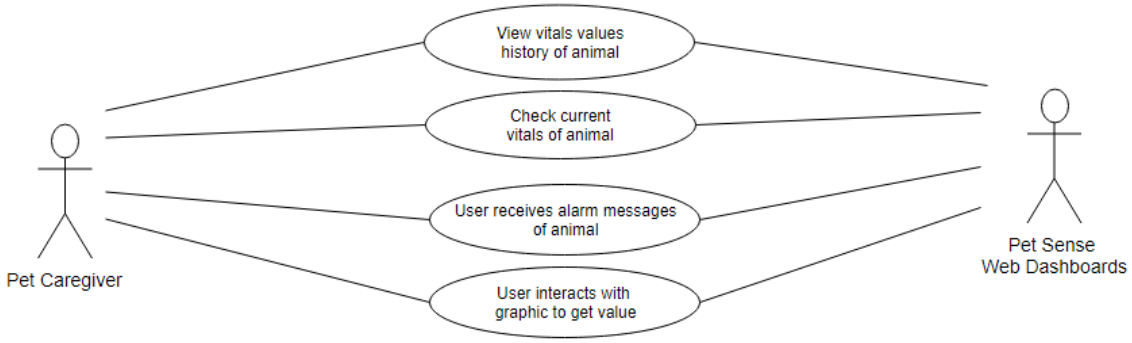

Image 13 - Use case diagram of Pet Sense front-end

Use Case	Description
Verify vitals history of animal.	After the user successfully logs in the Pet Universal platform and chooses to view a specific animal's history of vitals, the system should be able to display all registered values of vitals of the given animal, including normal and alarmistic ones. These values can be viewed in a table or a line graph.

Use Case	Description
User wants to check last vitals of animal.	After the authentication and chooses to view a specific animal details, the system should be able to display the last registered value of vitals of the given animal, or display nothing if the frontend is not receiving values from the system.
User receives alarm messages.	The system should be able to display the values through alarm pop-up messages. As mentioned in chapter 2.1.2 the normal values of temperature in pets are known and if the temperature rises or falls rapidly during recovery, it is crucial that the caregiver is notified immediately to make decisions of treatment.
User interacts with visualization graphs.	In the Pet Universal frontend, the user wants to check the evolution and peaks of the values of temperature and heart rate through a line graphic. By hovering the mouse over the specified point, the user should get the values of that point.

Table 5 – Use case description of Pet Sense front-end

In a user point of view there are a series of action that are expected to be performed on the web Pet Universal frontend.

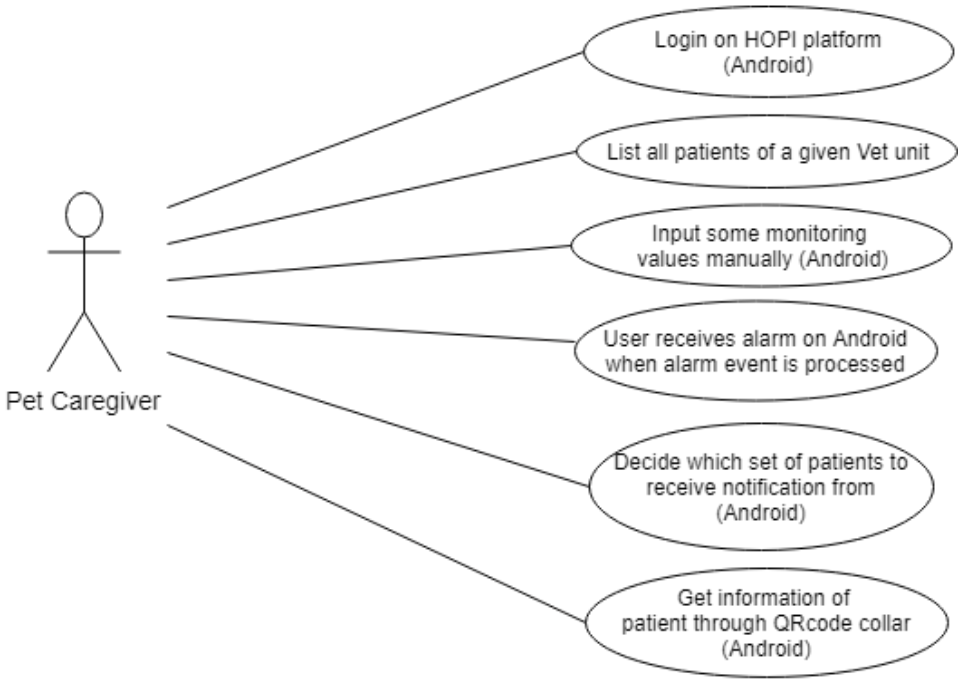

Image 14 – Use case diagram of Pet Sense Mobile

Use Case	Description
User Logs in on Pet Universal software.	This is the first interaction that the user has with the system. Assuming the user has registered previously, the login use case is responsible for identifying the user in the system for future refence (displaying specific data). This authentication is made via e-mail and password. The Pet Universal software system then returns a user ID and a token, which are used to get the corresponding animals and sensors correspondent of the authenticated user.
User chooses which patients he/she wants to receive notifications from.	Next to the list of patients an option of subscribing to its notifications must be present.
User receives notifications.	If a notification is on for a given patient the user should receive notifications of alarms related to that patient.
Input monitoring values manually.	In another application view, the user should be able to input the temperature and heart rate values of a given patient of the user's clinic. The user submits the values to the Pet Sense system.
User gets information about patient by clicking the patient's name.	On the list of patients, the user should be able to view more information about a specific patient.
User gets information about patient through QRcode.	Another way to view the information of a given patient is through the smartphone's camera using the QRcode technology and the backend of Pet Universal. The QRcode is sticked in the collar or cage of the patient and identifies the patient.

Table 6 – Use case description of Pet Sense Mobile

In a user point of view there are a series of action that are expected to be performed on the Pet Sense Mobile frontend.

3.4 Non-functional requirements

The requirements of the users of Pet Sense are quite homogeneous even across vet clinics and across countries due to the similarities of their services. In addition, the physical space across vet units is also comparable enclosed rooms with veterinary equipment, cannels, outdoor area, etc. For this system to be implemented, the technical requirements described in Table 7 were taken in consideration.

Requirements group	Requirement	Additional Description
Maintainability	The system should be deployed in existing clinics, with minor changes on the available infrastructure.	The software architecture and constraints should make the system easy to deploy in the clinic environment and add no extra requirement in addition to those of the base Pet Universal solution.
Compatibility	The system must integrate with Pet Universal tools suite.	The system should be integrated with API webservices.
Portability	The system should be able to be portable in terms of server/virtual machine.	The system should rely on virtualization techniques (Docker) for the deployment.
Autonomy	Once running, the system should continue with minimum human intervention possible.	As a monitoring system, it is expected that is runs as long as the hospitalization interment lasts.
Availability	Communications and data exchange should be available	The system should be able to allow data extraction constantly.
Durability	After the hospitalization episode is finished, data should be stored and remain unchanged	The system should be able to store data and should not allow changes in “perpetuity”.
Performance	Visualization should be as close to “real-time” as possible. Sensing should happen at least every 5 minutes.	It is expected that each message takes less than a second to present the message emitted by the sensor, and that it takes a measurement at least once every 5 minutes. This way, caregivers get much faster access to data than physical check-ups.
Data logging	Data from each tool is stored as logs.	Most tools used should enable logging methods for future debugging and registering.

Table 7- Technical requirements for Pet Sense

In terms of data processing, no requirements were specified. The budget for this project was not a priority, and so, is the choice was to use tools for processing and handling data, it must use opensource or free tools.

4 ARCHITECTURE

We propose an IoT solution deployed along Pet Universal's SaaS structure as a centralised cloud service for data analysis and processing. Inside the hospital/clinic it is only necessary to install the wearable sensors (smart collars) and gateway/ aggregator. After that, the aggregator will capture vital signs from hospitalized pets and will automatically connect to the Pet Sense server for processing. Storing and making API's available to the display that information to the caregiver in dashboards on the Pet Universal software's interface and other platforms including the Android App. Thus, suppressing the need for physically and periodically checks on the animals.

As an IoT solution integrated in monitoring animal's health, it involves a set of diverse software and hardware parts. This chapter describes the structure of Pet Sense system (architecture, how information should be handled, etc).

4.1 System structure

Pet Sense is a cloud-based IoT system integrated with Pet Universal's SaaS. It consists on a telemetry system that streams measurements from sensors mounted on animals and sends this information through an IoT-like architecture to the caregiver display (as shown in Image 15). This architecture is divided into 4 layers that are subdivided into smaller parts according to its function or physical representation.

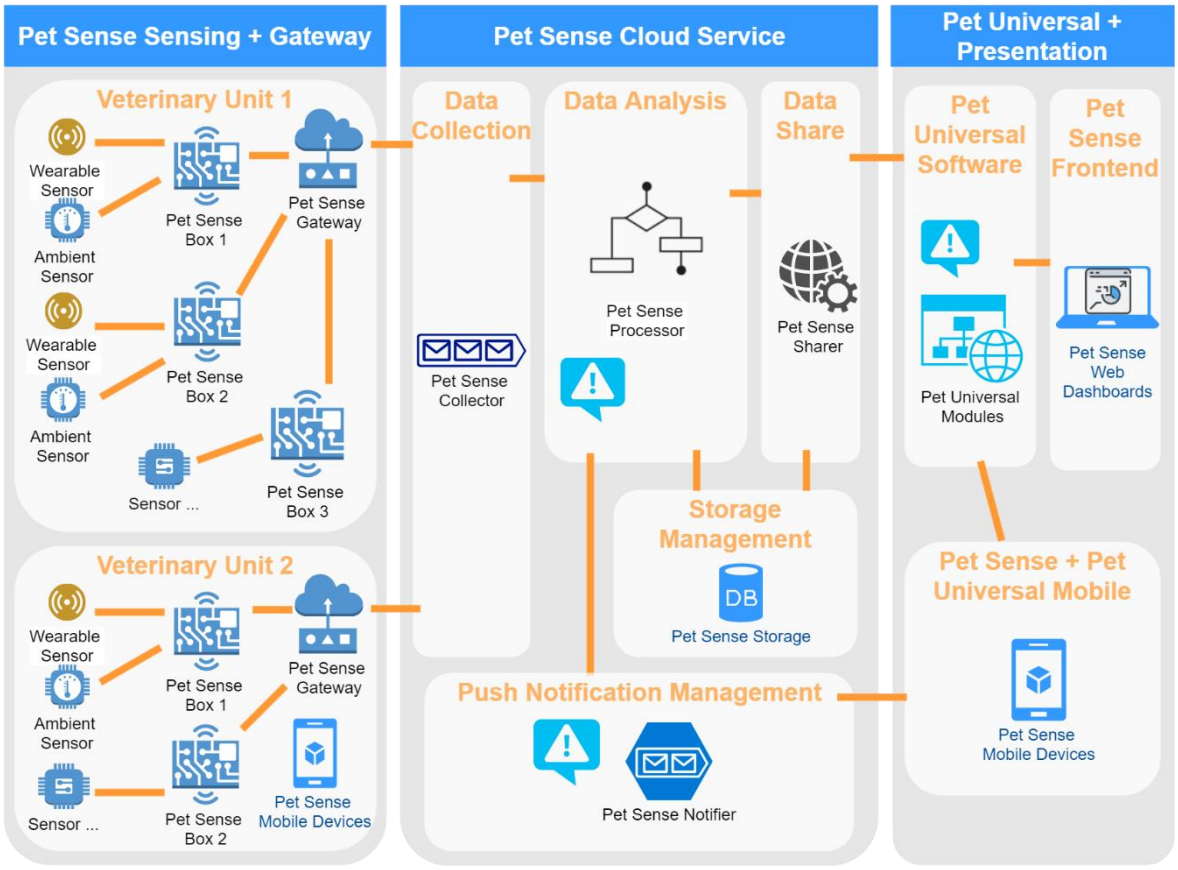

Image 15- Pet Sense system layered architecture

As wearable components, the sensors must be in contact with the animal body and/or its cage. These sensors are wired to a piece of hardware called Pet Sense Box that is able to sense both ambient and body metrics (when is only body metrics is usually called “Smart Collar” in animal IoT systems). The Pet Sense Box that is worn by one animal would gather all sensor data from that animal and its environment, transform it into structured, timestamped messages and periodically send them to the Pet Sense Gateway. In a standard veterinary unit, we expect to have many monitoring stays at the same time which means that there will be many smart collars connected to one Pet Sense Gateway. We propose that this connection uses Bluetooth as it is one of the most powerful and secure wireless protocols.

The Pet Sense Gateway’s purpose is to gather data from all Pet Sense Boxes and produce streams of messages that are sent to the Pet Sense Cloud Service. It serves as a communication gateway between the vet facility and the rest of the Pet Sense system. For each hospital/ clinic there is only one Gateway.

As Image 15 illustrates, the centre block “Pet Sense Cloud Service” is composed of 5 parts. The first component is the Pet Sense Collector, which is responsible to gather the messages from all the Veterinary Units that are in the system. The Pet Collector serves as a message broker and therefore must use a lightweight protocol (we use MQTT) and tool (we use Mosquitto) to be able to support such a load of data coming from several places at the same time. Mosquitto is one of the most used

MQTT message broker. It uses publish-subscribe pattern and is lightweight, easy to install and configure and comes with great community support.

Pet Sense Cloud Service stands as a centralised stream processing and data storage system in a server accessible through the Internet. With the help of opensource stream processing tools, the system should be able to receive, read, make decisions accordingly and make the messages available on a stipulated format (preferably a key-value format to simplify data readability).

After Pet Sense Collector, the messages should be passed into the Pet Sense Processor to be analysed by the algorithms in the code part where data will be read and stored/shared accordingly. This is the most important part of Pet Sense. The algorithm must be designed to read the content of the messages, find the sensor parameter values and if they are in the normal range, it should store and share these values, but if they are not, besides storing and sharing, it must push these values to a Pet Sense Notifier management system that immediately distributes the messages according to the source of the alarm (animal in risk).

Kafka Streams is proposed as the processing infrastructure. Apache Kafka seems promising as the future data analysis tool for streams with its KStream class capabilities. The learning curve for Kafka is easier than much of its competition and so, this section describes the processing methods that are to take place in the implementation. Kafka Streams uses the publish-subscribe pattern with topics and the name of a topic should be adequate to its purpose.

For the processing component many choices must be taken as the software reads the messages. First the system must extract the information from the messages: this is done by mapping the formatted messages and saving every data point onto the database. This method prevents data loss and enables future queries. After that, only the fields that are expected are read, for example: temperature, hear rates, animal type and animal state. With those values, conditional functions must be implemented to check if, according to animal state and animal type, the temperature and heart rate values normal or not. In Table 1, Table 9 and Table 10 representation of the types and states of the same animals can be correlated to get the variation expected (normal and abnormal) values of a given animal in the message. This is represented in the next chapter.

For storage, a NoSQL database will be used as it is best suited for this project as little effort is to be put into the conceptualization of the database, due to time constraints. All the messages must be stored in Pet Sense Storage for future consultations, normal values as well as the alert values with all the ACID properties (Atomicity, Consistency, Isolation, Durability).

This server must provide a way to export data from the database and make it available for the external applications, this module of Pet Sense is called Pet Sense Sharer. Data is to be consumed by other software for display purposes, and it must be as widely compatible as possible as described in the requirements chapter.

Pet Universal system is represented in Image 15 as a provider of webservices for the App, a consumer of the Pet Sense messages, and displayer of dashboards and information for each monitored animal.

For better understanding of the interaction between the user, Pet Sense and Pet Universal, Image 16 shows the interactions between these entities in an expected alarm use case.

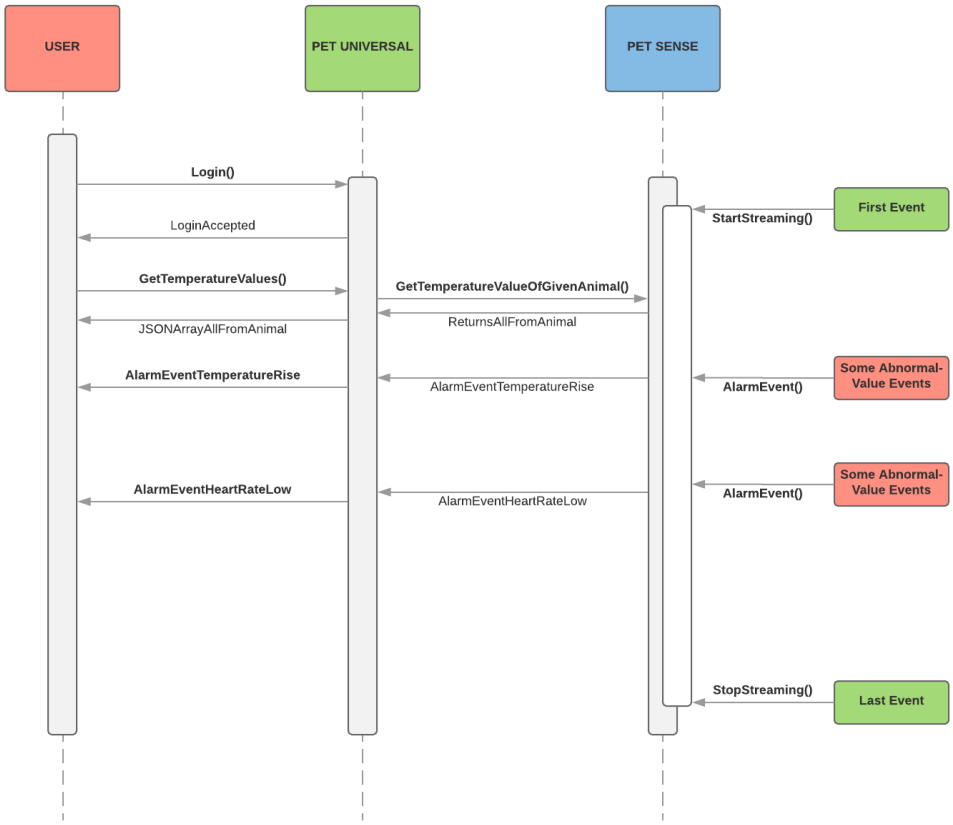

Image 16- Example of interactions between user, Pet Universal and Pet Sense

In the example represented in Image 16, the user logs in to Pet Universal platform and visits the patient page to check out the temperature values. Pet Universal then sends a request to Pet Sense for that given animal temperature history. The given animal is expected to have had the sensors streaming its first events before this point. The repetition of this request and its periodicity is Pet Universal's responsibility. Pet Sense system must be capable of handling a fair number of requests in a short amount of time.

The streaming continues even if the user is no longer on the patient's page (making requests). If any abnormal values are registered, the Pet Universal's system emits an alarm that is shown on any page the user is in. It may repeat for as long as the animal values are not stabilized, or until the streaming from the sensor is cut off.

The Pet Sense Presentation layer consists of the graphical views of information. They can be: label with the "current" or "latest" value monitored, a chart that shows the history of each parameter monitored (temperature, heart rate, etc) through time, a table with all values sensed (much like the database) and the alarm view (banner) which must be highlighted.

Through the Pet Sense Mobile, veterinary professionals will be able to receive the alarms on their smartphone devices at the moment the alarm is activated, and also be able to perform other features such as identifying the animal, listing all pet stays and tasks in his/her veterinary unit. In the event that the hospital/ clinic is not able to have sensors installed, another feature of the Pet Sense Mobile can be used which it may serve as a Pet Sense Gateway where the caregiver inputs the monitoring values manually into the App, and then it sends the values directly to Pet Sense Collector. In this last case, the input values are measurements done manually by the professional and not from mounted sensors.

4.2 Information View

Pet Universal system is a complex, mature software with thousands of classes and entities. Pet Sense increases that number with its own entities and concepts. For example, up until this point, Pet Universal never needed to register the animal's type or state in their backend system.

In this section, we describe the concepts and nomenclatures to better understand Pet Sense's entities, with the help of Image 17 (relation between concepts) and Table 8 (description). In Image 17 it is possible to distinguish between what was already implemented in Pet Universal software (red classes) and what was to be added (black classes).

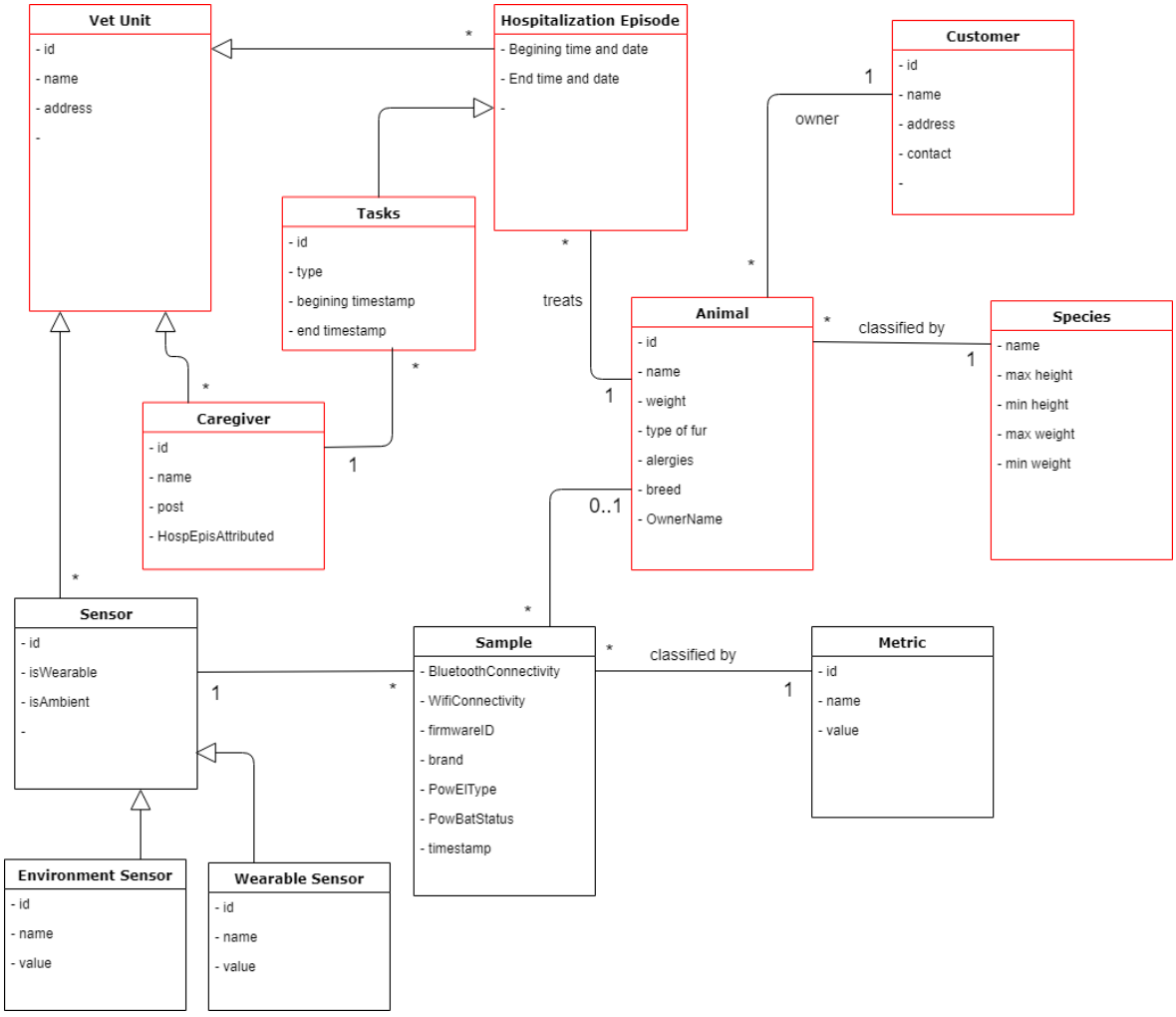

Image 17 - Pet Sense concept relations

Veterinary Unit has one gateway, many caregivers and many hospitalization episodes. The gateway allows connections from many devices which can have one or more sensors. This sensors perform measurments of metrics. The caregiver is responsible for complete tasks assigned to a hospitalization episode. Each hospitalization episode can have one or more caregivers responsible for administrating the tasks. One hospitalization episode has only one animal, an animal belongs to one owner and one species.

Entity	**Description**
Veterinary Unit	A physical place which may be a hospital, a clinic or a pet care centre. A Veterinary Unit employs Pet caregivers, installs sensors, and starts medical hospitalization episodes.
Hospitalization episode	A clinical stay is a state in which an animal is subjected to. To be medically interned is to be in captivity, usually inside a cage, after a critical medical condition is assigned. The caregiver must immediately start monitoring the animal. It has a start and end time and date and has one and only one animal associated.

Veterinary caregiver	A caregiver is a veterinary staff professional that overviews/ oversees the pet stay. Each caregiver has one or more stays at its responsibility. The caregiver is the person that will receive the alert notification.
Animal	An animal is the subject of observation/ monitoring. It may be a dog or a cat. It is associated with one stay. For each animal size and race, the values of normal body-temperatures and heart rate may vary. Because of this, different types of animal were created to classify animals based on weight. The weight is measured at the beginning of the hospitalization episode. See Table 9 and 10.
Species	Each pet is associated with a species (dog or cat).
Customer/ Owner	The owner represents the human that is legally responsible for the pet. One owner can have many pets, but each pet has only one owner.
Sample	This is the snapshot sample from that is taken on the device sensor periodically.
Device	This is the piece of hardware that is connected to one or more sensors.
Sensor	A sensor represents a hardware device that senses the vital signs. Each sensor has a type according to the metric it measures.
Environment/ Wearable	These represent the difference between a body sensor and a cage context sensor.
Metric	Monitoring parameter translated from the real world. It can be for example: body-temperature, heart rate (bpm), ambient-temperature, etc.

Table 8- Concepts in Pet Sense

The following Table 9 and Table 10 describe the numerical type and the numerical state codes, respectively, regarding how to catalogue and specify any animal in the system.

Type	**Animal**	**Description**
1	Micro dogs	These are also called “purse dogs” or “toy dogs”. These dogs weigh from 1kg to 3kg.
2	Cats	Cats are the second species of animal most cared for in vet establishments. They weigh around 2kg to 4,5kg and so deserve a separate type value.
3	Small dogs	These dogs are more common in urban areas and weigh between 3kg to 10kg.
4	Medium dogs	Medium dogs weigh from 10kg to 25kg. Examples are: Border collie, American Terrier, English Bulldog, etc.
5	Large dogs	Large dogs weigh from 25kg to 50kg. Examples are: German shepherd, Golden retriever, Dalmatian, etc.
6	Giant dogs	This type of dog is visually imponent and start to weigh at 50kg and no upper boundary, the heaviest breed has a normal weight of around 150kg. Examples are: Grand Danois, St. Bernard, Terranova, etc.

Table 9- Types and sizes of animals in Pet Sense

Having weight be the main factor for cataloguing is enough according to veterinary doctors, as pets with the same weight usually have similar vital sign values/ variations in temperature and heart rate (even though it may vary according to the race and not only in weight). Consequently, the type

described above is our first attribute of an animal. An interned animal also as a name, an ID, an owner, an age, etc. An animal may have one or more sensors attached to it, as described previously.

Moreover, there must be a metric to describe the state of the animal. For example, if the animal is a female and is pregnant, the temperature values might be higher than described “normal”, even though they are correct.

State	Animal	Description
1	Normal	Normal hospitalised state.
2	Pregnant	Pregnant animals might have their temperature higher than in their normal state.

Table 10- States of animal in Pet Sense

Studies show that the temperature in dogs rise insignificantly throughout pregnancy but drop considerably 24 hours before labour starts, as Image 18 shows. During pregnancy temperatures go between 37,7 ºC (~99ºF) and 38ºC (~100ºF), in the 48 hours before labour rise up to 38,6ºC (~101ºF) and then drop to 36,7ºC (~98ºF) 24 hours before labour starts.

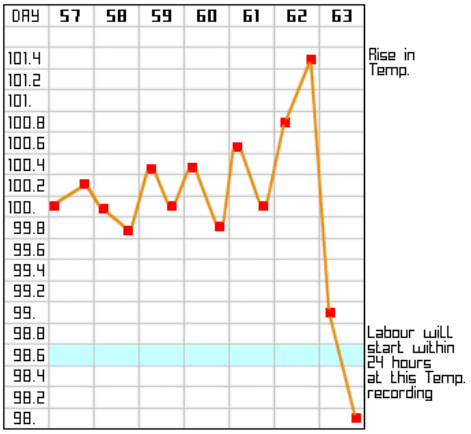

Image 18 - Body-Temperature of a dog seven days before delivery
Source: http://www.lowchensaustralia.com/breeding/pregnancy.htm

5 IMPLEMENTATION

In this chapter, the implementation will be explained, as well as some exploratory implementations that were made before developing the final solution. The practical aspects of the development will be described, highlighting relevant aspects of technologies.

5.1 Exploratory Implementation

The idea of this project was proposed by Pet Universal in cooperation with IEETA and nothing had been started either in Pet Universal nor in the university of Aveiro. And so, important choices had to be made throughout the thesis implementation in terms of how to process data and approach the alarmistic functions. Because of time and experience constraints, building a system to process streams from scratch was a gradual process of learning. One of the options of approaching the implementation of such system was to use only Java code in the Pet Sense Cloud Service (building a program that was able to collect, process and display alarms and data in a J2EE architecture manner). This would be time consuming and the results might not be as efficient to handle the huge loads of data in the pipeline. The decision was made to implement a Java system but with the help of tools available on the market specialised for this type of architecture. As mentioned before, one of the objectives of this dissertation was to explore and implement state of the art tools and frameworks. An IoT architecture was planned and implemented with the use of opensource or free tools, enabling the developer to create their own filters and triggers on certain points with his predilect programming language.

Due to the limited experience in such projects of the author, a gradual and experimental implementation of different tools was needed to better understand how the data is manipulated and carried throughout the pipeline.

The following sections describe the experimental approach to IoT enabling frameworks, prior to the system implementation. Besides the description of the tools, certain aspects are enumerated such as advantages, disadvantages and why they were or were not used in the final implementation.

5.1.1 Elastic stack

After the state of the art research, the Elastic stack was a promising framework for the project. It seemed like a good tool to start implementing to get a first approach on the overall pipeline of streams and as it allows the integration of plugins and other tools like Grafana. Many articles and blogs from experienced developers were found suggesting the Elastic stack as one of the best, most recent log processer, searcher and displayer for diagnosis of computer systems. At that time, it seemed as the best option and fast to implement (with lower learning curve). Logstash served as Pet Sense Gateway and routed the messages to Elasticsearch (Pet Sense Processor) who read the messages and stored them all into an Influx DB (Pet Sense Storage). Displaying the messages in graphs was no trouble for Kibana at this point.

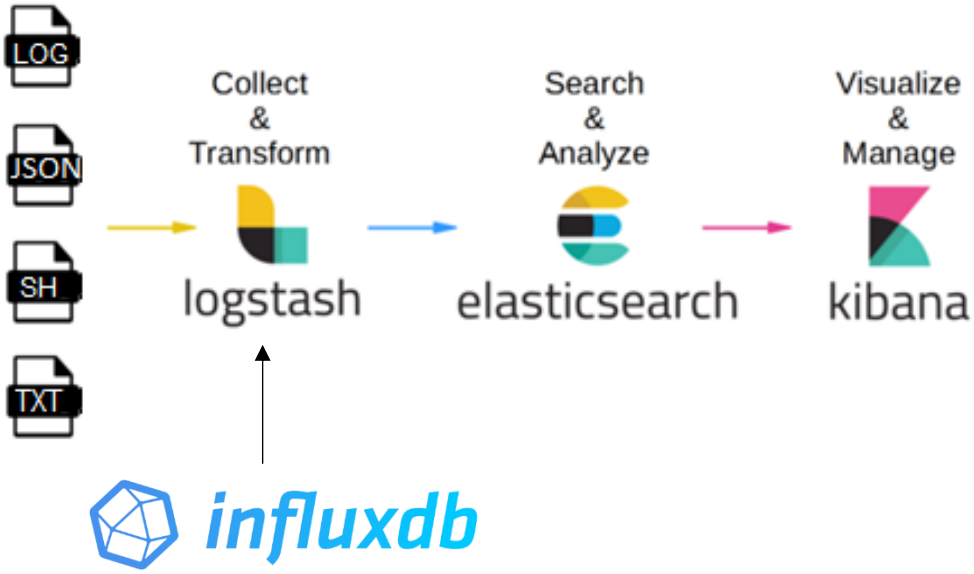

Image 19 - Elastic stack experimental setups
Source: https://misterhex.github.io/Monitoring-with-ELK/

All three components used (Logstash, Elasticsearch and Kibana) were configured (as Image 19 exemplifies) in localhost and different plugins of input and output for the data pipeline were tried: document oriented (system logs and txt files), JSON string messages (through command line scripts), directly from a database (InfluxDB), using TCP, etc. In every experiment, the stack was able to filter and rout JSON messages that were displayed as expected in the Kibana dashboard. The way these three tools work is that Elasticsearch needs to receive data in a pre-configured manner to manipulate data in an object level. For example: in a JSON formatted message, Logstash needs to parse each key-value in order for Elasticsearch to know how the data is formatted. Elasticsearch does not provide a GUI without the help of visualization tools (Kibana).

Logstash is a powerful filtering tool that enables specific configuration using three stages: inputs, filters and outputs. For input configuration, Logstash needs to know where to get the data from: files, syslog, Redis server, or other elastic tools like Beats. This can be a disadvantage as it is a limited way to get data. In some implementation experiments, files and syslog were used. For filtering,

plugins such as Grok and *geoip*, and commands such as mutate, drop, clone are the most documented in elastic. Grok plugin parses logs in a comparison with stipulated formats, it was used to parse JSON messages. *Geoip* tags each message with a geographical position, it was experimented briefly proving to be effective, but flooding the message with "unnecessary" data, it is most used for IoT movement tracking projects. For output, Logstash needs to know where to send data to. This was experimented in three ways: command line, Elasticsearch port and files. For debug purposes printing the parsed messages directly into the terminal prompt helped us view the immediate result. To proceed to the next phase in the data pipeline, the parsed messages were sent to Elasticsearch port. All these processes can be optimized with the use of codec specification. Examples are: json, plaintext, *avro* and command line.

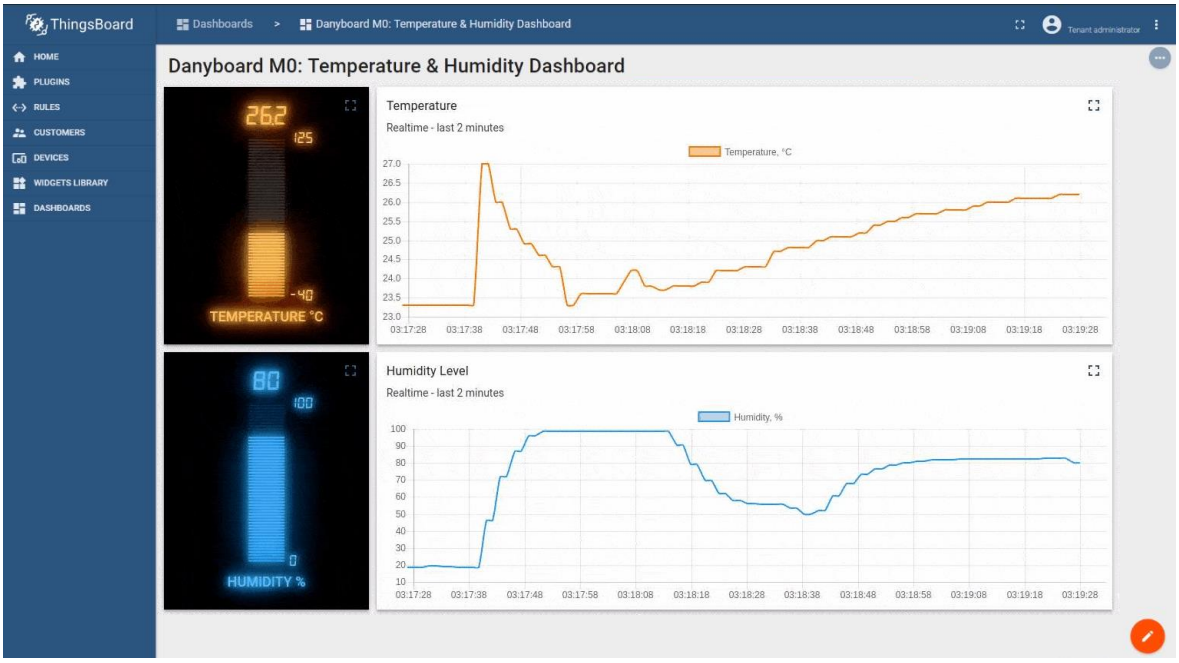

Image 20- ThingsBoard dashboard view
Source: https://www.thingsboard.io/docs/samples

This implementation was relatively easy: Logstash with a json codec reading from a MQTT port, parsing temperature values to Elasticsearch and Kibana with a line graphic with historic view of the temperature values.

However, when trying to create rules for processing the messages, Elasticsearch was very limited and insufficient. It was realized that the elastic stack is designed to improve queries and serve as a search engine for logs and was not possible to create alarmistic functions to output to other systems.

Elasticsearch was designed to improve monitoring on computer systems and diagnose potential threats. Using logs from servers or applications, the elastic stack is an effective and efficient way to diagnose and create performance reports according to CPU, memory, and so on. We learned that it was more focused on its searching features, not rule or stream content analysis, although it makes use of streaming and event-driven concepts.

To the present date we learned that elastic.co[30] is developing new capabilities to the stack by adding components such as "X-Pack" and "Elastic cloud". X-Pack is an extension that adds security, alerting, new graph features. Elastic cloud is a payed cloud service that allows deployment, operation, and scalability to Elastic solutions in the cloud.

5.1.2 ThingsBoard

ThingsBoard[31] is an online solution that enables the creation of some full-stack end-to-end IoT systems. In a drag-and-drop style, it allows the configuration of sensors and constrained devices, rules, triggers, and dashboards, which virtually simulate message exchange or connect to real devices. It offers a wide range of plugins and widgets that connect all components of the IoT system. Through rule-based decisions, ThingsBoard processes messages and displays information on a dashboard which has many options of graphs and display features. It also contains a unique way to describe the data format as well as the detailed description of the classes and entities that constitute the data, all in a drag-and-drop style on a webpage where the user sets the configurations for the different stages in a menu as Image 20 shows. It is a ready to use, free online platform can be integrated with any protocol or standard in IoT.

After implementing an end-to-end simulated "Temperature and Humidity" solution in it, it was possible to verify the value of this tool. Image 20 shows the dashboard view of this simulation, with a simulated Raspberry Pi 3 as a Pet Sense Gateway and two sensors (Temperature and Humidity). In this implementation experiment it was possible to create alarms with the alarm feature that broadcasts in e-mail form. This last feature was tested but it was never a requirement for this project (via e-mail). Besides receiving an email, the alarms were possible to be seen on the ThingsBoard environment and dashboard as "red flags" on certain points. After finishing an implementation with responsive simulated data being processed in real-time, it all seemed too easy, and too fast. Little was learned on how the stream values were being processed and how and when rules were applied at a code level.

ThingsBoard has a gentle learning curve as it is very intuitive to use and as it has easy-to-follow documentation with lots of tutorials. It is an excellent tool for businesses that want reliable and efficient IoT projects built rapidly, however, as an academic project, little focus would be done to what this implementation should be: the processing of streams.

5.1.3 Apache Spark vs Kafka Streams

Apache Spark is known to be the most powerful and complete platform for streaming projects compatible with Hadoop, since 2005. It is an opensource analytics platform specialized in distributed systems that allows users to run source code and analyse streams using Java or Python objects and

[30] https://www.elastic.co/
[31] https://thingsboard.io/

lambda expressions. Spark proudly announces to be the fastest state of the art tool in processing streams and its compatibility with other tools and environments.

Spark Streaming is the extension of Apache Spark that allows for the features of scalability and fault-tolerance to take place. It uses the Spark Engine to batch data as Image 21 shows.

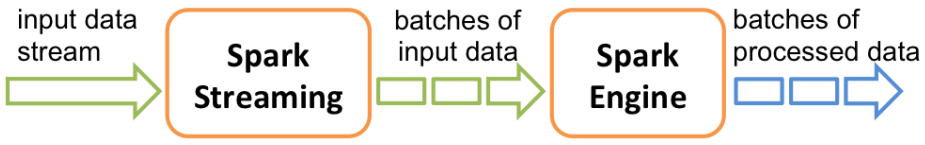

Image 21- Spark streaming and Spark engine relation
Source: https://spark.apache.org/docs/latest/streaming-programming-guide.html

Spark Streaming needs to be configured for input parsing of data with high-level functions such as map, reduce, join and windowing and output. It is widely used for IoT projects for stream processing in a variety of projects combined with RabbitMQ.

Nevertheless, new solutions like Kafka Streams are starting to show promising capabilities and be a fearless competition with new features and updates every other month. In comparison to Spark, Kafka engine is a more state of the art tool and seemed to have a lower learning curve. No experimental implementation was done for the Spark engine, due to the evaluation made in the literature review, downgrading it in comparison with Kafka Streams. Therefore, we decided to develop the final project with Apache Kafka, as the processing and central routing message broker of the system.

5.1.4 BITalino

The first implementation with a device that collects sensed data and transmits it via the Bluetooth protocol was done with a BITalino constrained device. To test the viability of a Bluetooth connected smart collar, a BITalino board was used to monitor the heart rate. BITalino is a human health focused constrained device that connects via Bluetooth. It has a set of sensors included in the board: Electromyography (EMG), Electrocardiography (ECG), Electrodermal Activity (EDA), Electroencephalography (EEG), Accelerometer (ACC) and actuators like one Light-Emitting Diode (LED), Pushbutton (BTN), Buzzer (BUZ), and of course an integrated Digital-to-Analog Converter (DAC). Image 22 shows BITalino board and the enumerated module parts. BITalino is not a medical device nor is was intended for medical purposes. It is envisioned to be sold as a tool for biomedical education, research and prototyping, and, like Raspberry Pi, makes inaccessible hardware available to everyone.

For the implementation of BITalino, an official C++ API was cloned and implemented in the Raspberry Pi where the Bluetooth connection is configured statically (using the device Bluetooth MAC address) and the code to get values from all module parts is ran. This library enables C++

applications device through a simple API interface. This API is composed of a header file (bitalino.h) and an implementation file (bitalino.cpp), and a sample test application in C++ (test.cpp). This last application ran and got raw analog values that have to be converted using sampling rates. There is also a Python API that does the same processes and is composed of only one implementation files (bitalino.py) and test application that was also implemented as a test.

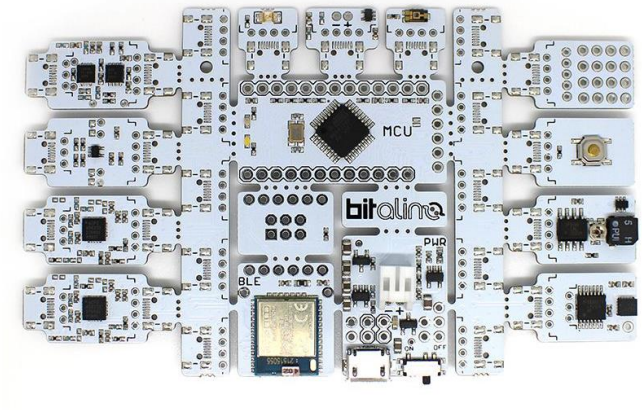

Image 22 - Revolution BITalino board

5.2 Implemented Solution

This section describes all the technologies and configurations made for the implementation of this project. It includes the system architecture and all its components, protocols, tools and programming languages used, from the collection of digital values from the sensors, through the centralized Pet Sense Cloud Service where the software developed for processing is, to the availability of the data and visualization parts. The implemented solution proves the viability of the project in a future real environment and a future Pet Universal product module.

It is expected that the final solution can demonstrate the viability of such project in a real scenario on a hospital environment. The implementation must also be able to simulate some scenarios of data being emitted by sensors with normal and abnormal values, aggregating them on a local gateway, carried using MQTT to a Kafka Streams client server that processes them and emits the appropriate alarms when expected. Besides this, it should also store all messages in a NoSQL database and provide access to the data remotely through webservices to be consumed by different types of frameworks and platforms of front-end.

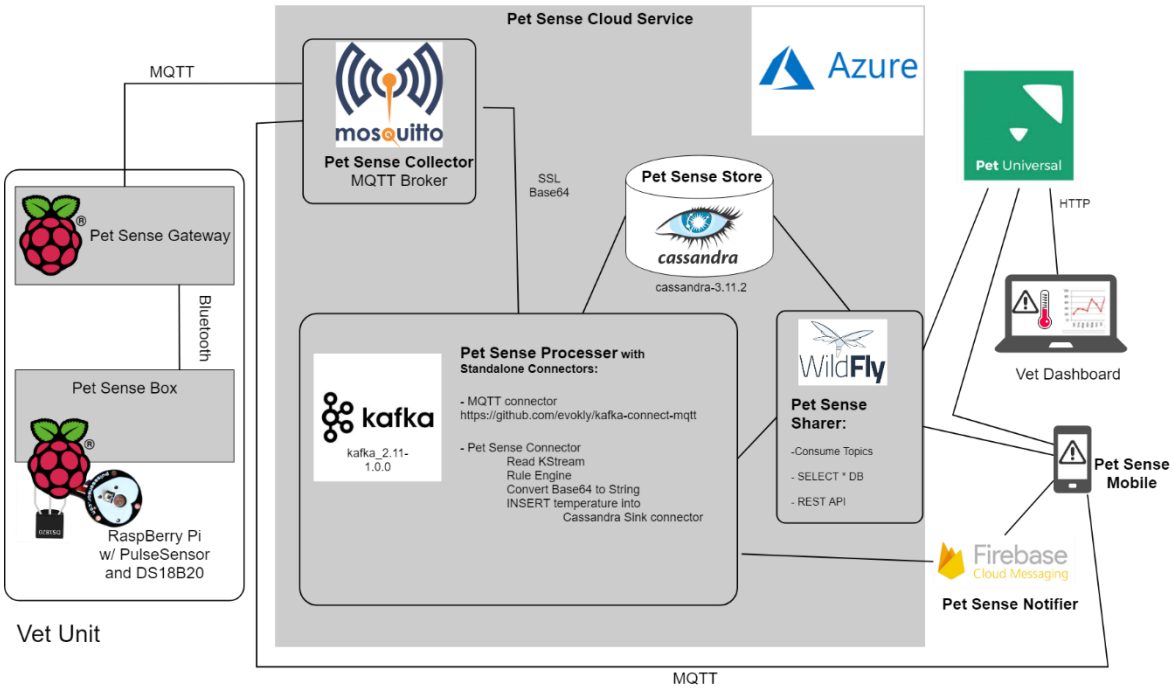

Image 23-Pet Sense system implementation

Image 23 is a representation of the Pet Sense system. It starts at the sensor and gateway layer (left side), where data is captured from real-life parameters, passed through a virtual machine where data is processed (centre square) and ends at the visualization part (right side). As it was not possible to build a real collar with evidence that the sensors were reading true values from real animals, for implementation purposes, an ambient temperature sensor (ds18b20) and a human heart rate sensor (Pulse Sensor) were used.

These sensors are wired to a Raspberry Pi (explained in 5.2.2) that serves as Pet Sense Box and another serves as Pet Sense Gateway (both connected via Bluetooth). The Raspberry Pi's would be present in the Veterinary environment. When both Raspberry Pi are booted Python scripts run automatically, connect to each other's' MAC address, and Pet Sense Box starts to collect data from all sensor sources, concatenate them into a JSON formatted string and periodically sends the messages to Pet Sense Gateway who forwards this to the other Raspberry Pi that sends them, with the MQTT protocol through the Internet, to the Pet Sense Cloud Service MQTT port.

This Pet Sense Cloud Service consists on a virtual machine from Microsoft Azure. An instance of Mosquito receives the messages using the MQTT protocol. When the virtual machine is booted, an Apache Kafka serves as the central message broker and with the help of a MQTT connector, subscribes to Mosquitto topics and stores all messages in real-time to a Kafka topic. This Kafka broker also runs another connector, developed by João Ribeiro, that consumes the data and uses Kafka Streams modules to implement the decision algorithm based of the message content. Once a message is collected by this connector, it is mapped and processed and passed into another topic. All messages are stored in a local Cassandra Database (Pet Sense Store) that stores both alarm and not alarm messages. Pet Sense Sharer consists of an instantiation of Wildfly (that also starts running during the VM boot) that provides webservices whenever called from the Pet Universal system to retrieve and consume all messages in a JSON format via HTTP. Red Hat Wildfly is configured to provide a deployment of a Java program that enables availability of Pet Sense messages through a REST API architecture and queries the database for the events requested. Also, in the Kafka broker, messages of alarm are sent directly to the Android smartphone of the users through Firebase Cloud Messaging service.

The Pet Sense Mobile's main purpose is to receive the alarms in real-time, but also to register some monitoring values using MQTT protocol, in case manual data insertion is necessary.

On the following sections each part of this architecture will be described and explained in detail.

5.2.1 Sensors

As seen in the literature [14], skin temperature can be used to get the core body temperature of a dog. At a first experiment, simple LM355 thermistor sensor was used, but it needed an Analog to Digital Converter (ADC), and the documentation to do this was not available and due to the lack of experience of the author in this field other option was chosen. One of the simplest temperature sensors, DS18B20, usually is bought with the ADC attached, and this was the case. This thermistor sensor, illustrated in Image 24, is cheap and often used for academic/domestic projects as it is broadly integrated with digital constrained devices. It has 3 pins: 3.3V, ground and the output.

Image 24- DS18B20 sensor

After the system was built with the temperature values only, a second parameter was introduced: heart beats per minute. For this, we would need a more developed piece of hardware that would retrieve a digital value with the estimation of heart rate in a minute, using a sample of two beats. An electrocardiography is an exam result that allows doctors to see three parameters from heart beats: Heart rate (BPM) which is the number of beats per minute; Inter-beat interval (IBI) which is the interval of time between beats; and the Heart rate variability (HRV) which is the report on the variation of the IBI along the duration of the exam. These may be related to emotions in case of high-frequency rates or low-frequency rates. The result of this exam can be presented in many ways but for the purposes of this implementation, only the value of BPM was used, which is the most important for alarm verifications.

PulseSensor[32] is a hardware product developed by World Famous Electronics LLC since 2011. As referred previously, it is a photoplethysmograms, which means that it uses a LED to get the variations in pulse signals. This sensor was available at the university and although it has poor documentation and code libraries online, opensource communities constantly suppress this issue. PulseSensor returns analog values, and so a MCP3008 ADC was added as shown in Image 25. PulseSensor also has 3 pins: 3.3V, ground and output.

5.2.2 Smart collar and gateway

Raspberry Pi is a constrained device which is a piece of hardware the size of a credit card that has limited CPU power and memory. It a widely used single-board computer and in this project, it is a way to obtain multi-purpose, mobile hardware. Raspberry Pi was created in 2006 in the University of Cambridge, originally designed to be a cheap way for public schools to teach how to code. It is a very common device in IoT projects due to its size and capabilities, consumption, etc. It is used as a microcontroller with specific purposes, but it can also be used as a personal "computer". This microprocessor typically runs Linux as an Operating System and is capable of running a graphical user interface and many interface programs at the same time.

Based on two opensource libraries in Python (one for PulseSensor and the other for the DS18B20), the author was able to create the final implementation of the Pet Sense Aggregator, by

[32] https://pulsesensor.com/

running a script that calls both digital inputs (GPIO pins in the Raspberry Pi) and creates a JSON message that is sent to the MQTT port on Pet Sense Cloud Service every second.

This python script is composed of important libraries (paho.mqtt.client, datetime, json and Pulsesensor) and consists in a while-true cycle that on each iteration it opens the file where the GPIO pin values are registered from the DS18B20 and synchronizes with the heart rate sensor class object that returns the last BPM value registered. A JSON message is constructed with the following values: id, temperature, heart rate, clinic timestamp, animal id, animal type, and animal state. These fields will be described, later in section 5.2.4. Then, the MQTT client publishes the string message to the IP “51.145.152.136” and port “1883” (Mosquitto server). The cycle sleeps for one second and the next is executed until the script is interrupted.

For the implementation, we used two Raspberry Pi 3 B+ to serve as Pet Sense Box and Pet Sense Gateway, with Bluetooth capabilities to communicate with each other, and also only one sensor of each type mounted to Pet Sense Box as Image 25 shows. The Pet Sense Box is able to gather the data from many sensors at a given same time, it can connect to Pet Sense Gateway via Bluetooth and have a temperature sensor and a BPM sensor mounted on the GPIO pins of the device. In Image 25, an illustration of how the wiring in the Pet Sense Box was made. The two sensors are powered by the Raspberry Pi with 3.3 Volts and share it in a parallel circuit.

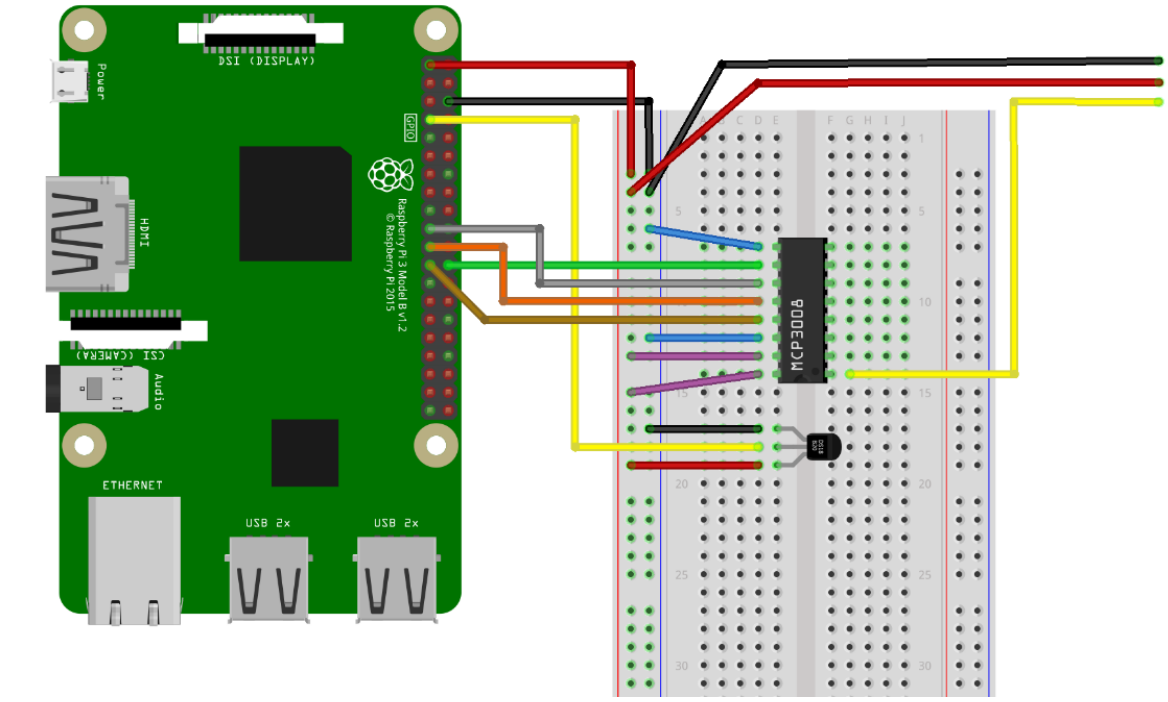

Image 25- Schema of the wiring done in Raspberry Pi Pet Sense Box

Pet Sense Gateway just receives the messages via Bluetooth and send it via MQTT to the 1883 port on the Pet Sense Cloud server.

5.2.3 Deployment and streams processing

A Microsoft Azure subscription was offered by Pet Universal to be used as a Pet Sense Cloud Service server, which will serve as a deployment and stream processing server. The virtual machine built was a B4MS Standard with 4 vCPUs, 16Gb of RAM and a 32Gb SSD disk, with an Ubuntu 16.04 Server operating system installed.

As Image 26 shows, Microsoft Azure offers a variety of online computational structures, it is a PaaS, IaaS, SaaS, and even DBaaS. The virtual machine package comes with an IP network configuration, DNS service, Firewall and Port management, and the possibility to connect other services. Inside Azure network, there is a private network where the virtual machine is mapped. The accesses were configured according to the Azure network, and are forwarded automatically.

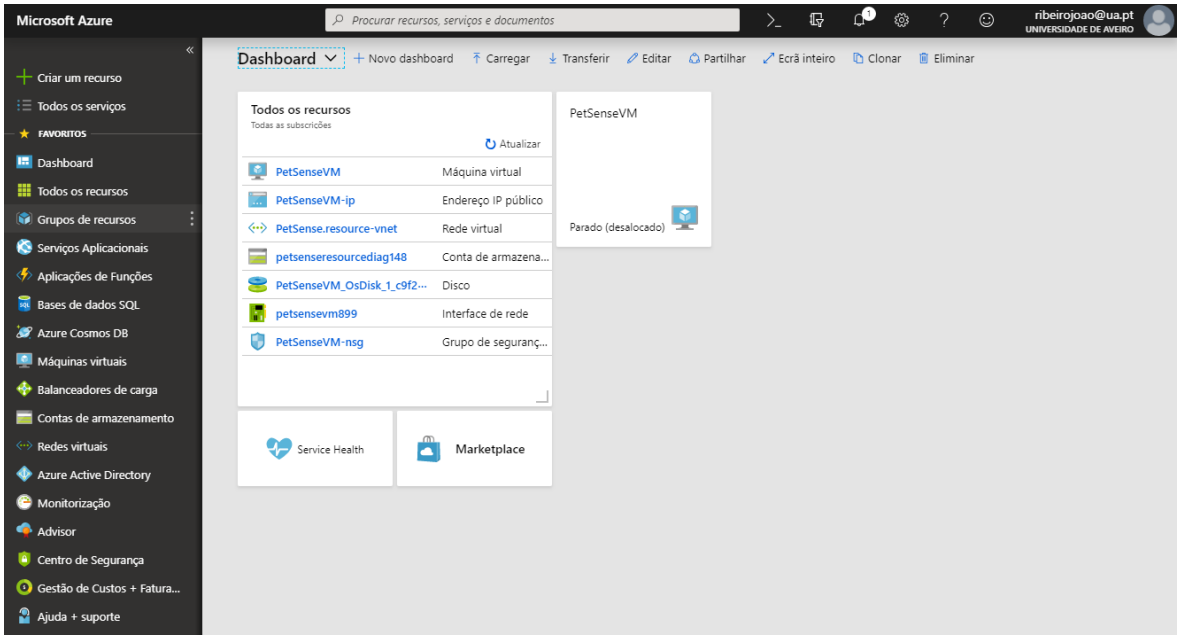

Image 26 - Microsoft Azure dashboard

Docker Compose was installed to provide portability and facilitate the control over the processing tools. Docker is a software that allows virtualization at the operating system level. This virtualization layer creates isolated “containers” that are able to build tools, libraries and configuration files, running as a single operating system kernel in a seemingly different environment. Containers make cloud migration effective as these are created from "images" which are often pulled from public repositories. As the tools installed in each container are viewed as images, it is easier to turn processes on and off, with docker-compose. Docker compose is a tool that defines and runs multi-container Docker applications. In Pet Sense Cloud Service, many tools need to be configured and so, their images were built using Docker Compose: Mosquitto, Confluent for Kafka, Cassandra (Storage) and Wildfly (Deployment and REST API). Some auxiliary tools were also installed such as Git (v.2.7.4).

With Java 8 installed and using the *curl* command, the docker-engine was installed with the version 18.06.1 and docker-compose with version 1.22.0.

As Pet Sense Collector, Mosquitto's job is to receive the messages in port 1883 and serve as a message broker to Kafka. This port was opened and configured to allow access at the Azure firewall. Mosquitto version 3.1.1 was installed pulling an image of docker. Mosquitto ensures that no message is lost if the subscriber (MQTT Kafka Connector) does not manage to get all messages at a given time.

Confluent 4.0.0 was installed as a docker compose image using the git clone command. Confluent offers an opensource platform that provides an easy way to manage the Kafka environment by having a command line interface (CLI) and enabling connectors to run code as "part of" the Kafka engine. Confluent community has been developing many connectors for all kinds of projects and for MQTT there were two, one of which was created by Evokly and was used in this project. Evokly is a platform that offers experience for specific needs for customizing deployments, and they created this connector for transporting all messages from the port to a Kafka topic. This connector written in Java is listening to the inputs from 1883 port and save them into a Kafka topic named "mqtt-topic".

Apache Kafka version 2.11-1.1.0 was installed as a docker image and configured to run on port 9092. Kafka is the central message broker of Pet Sense. As described in the state of the art chapter, it serves as a message broker that routes the messages according to their content. In Pet Sense's case, it runs the algorithm (described in the section 5.2.4) that checks the body-temperature, heart rate, animal type and animal state. All messages are forward to a Kafka topic named "all-topic" and stored in Pet Sense Store, described in the next section. If the algorithm finds abnormal values, these messages, besides being stored are forward to a specific topic named "alarms-topic", stored in the database but also forward to the according mobile smartphone. The next sections will describe these processed in a more detailed manner.

The Pet Sense Kafka Connector is composed of a set of java classes which contain the standard features for a sink connector, in this case: a Deserializer, and Serializer, Connector Task and Config a Service and Stream builder. This connector was created by the author (in Java) with the intention of translating (Deserializing) the messages in the MQTT topic, from Base64 into a String content message stored in a new topic where all messages are stored in json format (Serializer) and, at the same time, store and parse them as KStreams running the algorithm. The config call is responsible for setting the configuration properties for the database URL and port client. The connector class sets up the consumer tasks by starting or stopping the configuration of the connector. The connector task is the class where the data is retrieved from the configured topic and automatically sent to the instance of the implementation class as it is started or stopped. The implementation class is where the algorithm is executed as explained further. Image 27 shows the message flow that are transported between the Kafka connectors and Pet Sense Storage.

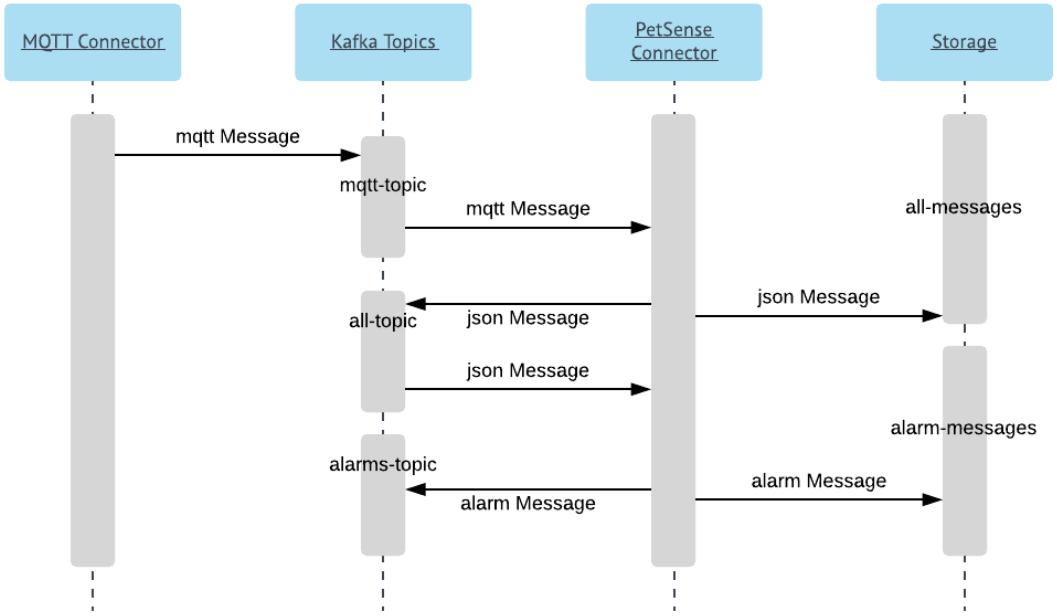

Image 27- Demonstration of how Pet Sense routs messages

These KStreams make use of two stateless transformations: map and foreach. With map, Kafka is able to map each key-value in the json message with little computational effort thereby making the deserialization from “mqtt-topic” to “json-topic” much easier. After this, the same KStream is read with the help of the foreach transformation, increasing the readability of the specific values we want to check. For this connector, org.eclipse.paho.client.mqttv3-1.0.2.jar, json-20151123.jar, kafka-streams.jar java libraries were used, to name a few. Readme files accompany each connector for better understanding and keep up with the minimal documentation quality assurance standards of code writing.

Finally, to finish describing the Azure virtual machine content, Wildfly 12.0.0 was installed for the deployment of Pet Sense Sharer which is a maven project that runs a REST API java architecture, pulling from the database all requests done by HTTP (described in 5.2.4). Wildfly is accessed through port 8080, which was opened and configured to allow connections at the Azure virtual machine firewall. This module starts running when the virtual machine is booted, and is independent of the other modules’ state, except for Pet Sense Storage, that is where it collects the data from.

Pet Sense Presenter is composed of a GUI module to present the graphs and data in a user-friendly way; and Wildfly maven program (Pet Sense Sharer) that provides the REST interface, thus not needing a GUI and not making part of the streaming pipeline.

5.2.4 Data management

This section describes the data management and the messages routing implementation.

It is important that each message is transported in a JSON format and using *Paho* library version 3-1.0.2. As described, the messages are always treated using JSON objects (either in Python or Java), and converted to String for transport (MQTT, KStreams and HTTP) or JSON arrays (in case of the webservice output having more than one message). The message payload should follow the following schema:

{id:<uuid>; temperature:<double/float>; heart_rate:<int>; sensor_timestamp:<String>; clinic_timestamp:<String>; animal_id: <int>; animal_type:<int>; animal_state:<int>}

The "id" identifies each message individually and are generated randomly on the gateway, *java.util.UUID* is a Java library that represents an UUID object which is a representation of alphanumeric characters and hyphens in the following regular expression:

Example of UUID: "123a4567-b89c-12d3-e456-464v54bgp00c"

Regular expression: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}

The "temperature" is the measurement of the patient's body-temperature in Celsius and it must a Double (java) or Float (python) format.

The "hr" represents the heart rate value of the patient's heart beats per minute sense in BPM and it must an Integer (java and python) format.

The "clinic_timestamp" is a String representation of the timestamp given by the aggregator when a message is created, it marks the message time of creation. These strings must follow the *SimpleDateFormat* library format from Java and more specifically, the "ISO 8601" format.

And finally, the three variables from the animal: "animal_id", "animal_type" and "animal_state". "animal_id" corresponds to the id of the animal on Pet Universal's backend, it is unique to an animal in the whole of Pet Universal's database and corresponds to the same animal in Pet Sense backend. It is an unique and unreplaceable value of one patient. It must be an Integer."animal_type" represents the type of animal described in Table 9. "animal_state" represents the state of animal described in Table 10. Which together they form the algorithm implicit in Table 11 as conditions of normal values.

After there is a new entry in the "all-topic" topic in Kafka, the Pet Sense connector reacts by consuming it instantaneously and starting the foreach method of KStream. For each key-value inside the JSON message, the algorithm reads if it stands as the temperature, heart rate, animal state or animal type. In any case, this is stored as a java variable in each cycle.

The parameters that integrate the algorithm are: animal state, animal type, temperature and heart rate (BPM). The alarm is emitted if certain conditions are not met, but in all cases if any topic called "alarms-topic" does not exist in the Kafka environment, one is created and if a table called

“alarmsTable” does not exist yet, a new one is also created. The following Table 11 depicts the normal ranges of temperature and heart rate values, that were used for the algorithm.

Animal state	**Animal type**	**Normal temperature values range (ºC)**	**Normal heart rate values range (BPM)**
1- Normal	1- Micro dogs	34,44 – 36,11	140 – 180
	2- Cats	36,8 – 38,8	120 – 220
	3- Small dogs	36 – 38	100 – 140
	4- Medium dogs	37 – 38,5	60 – 100
	5- Large dogs	38,33 – 39,3	60 – 100
	6- Giant dogs	38,33 – 39,3	60 – 100
2- Pregnant	1- Micro dogs	36,44 – 38,11	140 – 180
	2- Cats	38 – 40	120 – 220
	3- Small dogs	38 – 40	100 – 140
	4- Medium dogs	39 – 40,5	60 – 100
	5- Large dogs	39,33 – 40,3	60 – 100
	6- Giant dogs	39,33 – 40,3	60 – 100

Table 11- Normal temperature and heart rate values according to animal type and state

The first parameter to be checked is animal state. If animal state is pregnant (2) temperatures are usually higher about 2 ºC than normal values (1). Other animal states are not accepted for now. After that we need to check the type of animal, and only then we check the temperature and heart rate parameters as Table 11 describes. This algorithm is a set of choices based on a hierarchical logic from state to type to metric. Each metric depends equally from the state and the type of the pet.

After the analysis and processing, these messages must be shared in the Pet Sense Sharer to be accessed through webservices. The ensuing Table 12 enumerates the HTTP request paths and describes what they return to be manipulated and shown in the Pet Universal frontend (Pet Sense Dashboards).

Resource	Path and HTTP method	Request Parameters	Response
Animal	GET: animals/	-	A JSON array with all animals that are being monitored.
	GET: animals/{id}	Id: a valid internal identifier for an animal. This identifier is the same across both Pet Universal and Pet Sense backends.	A JSON object to describe the animal id, last temperature and BPM value registered and the timestamp with what it was registered.
Temperature	GET: animals/{id}/ temperatures/	Id: a valid internal identifier for an animal.	A JSON array with all temperature values registered of a given animal with id {id}.
	GET: animals/{id}/ temperatures?since=2018-06-18	Id: a valid internal identifier for an animal. Timestamp: with the ISO8601 format	A JSON array containing all temperatures of a given animal within a time period. Starting at the timestamp passed as a request parameter until the current time. It can be an array with zero objects or only one. The parameter of the timestamp triggers an implementation prepared to get only the values after that timestamp.
Heart Rate	GET: animals/{id}/hrs/	-	A JSON array with all BPM values registered of a given animal with id {id}.
	GET: animals/{id}/ hrs?since=2018-06-18	Id: a valid internal identifier for an animal. Timestamp: with the ISO8601 format	A JSON array containing all BPM values of a given animal within a time period. Starting at the timestamp passed as a request parameter until the current

			time. It can be an array with zero objects or only one.
Alarms	GET: animals/{id}/ alarms/	-	A JSON array with all alarm messages registered of a given animal with id {id}.
	GET: animals/{id}/ alarms?since=2018-06-18	Id: a valid internal identifier for an animal. Timestamp: with the ISO8601 format	A JSON array containing all alarm messages of a given animal within a time period. Starting at the timestamp passed as a request parameter until the current time. It can be an array with zero objects or only one.

Table 12 - Pet Sense Sharer APIs description

These return values are the product of calls (queries) made to the local database of Pet Sense (Pet Sense Storage) which implementation will be described in the next section.

5.2.5 Storage

According to the Eclipse organisation, 93% of Databases used in IoT are open-source [29], and this supports one of the first requirements of this project: the use of opensource or free tools for its development. Keeping that as an objective, at the planification phase of this project, only opensource databases were researched.

Although its main competitor InfluxDB seems to be the best option as a storage unit as it is the most used TSDB and because of its performance advantages. However, when trying to implement it after the first data pipeline was completed, in other words, switching CassandraDB for InfluxDB, the mapping had to be made differently on the Kafka Connector as InfluxDB does not allow json format to be simply mapped and passed into the DB. Having to translate all data to InfluxDB structure surely would take longer than the current implementation.

Apache Cassandra 3.11.2 was installed using docker compose and was configured to port 9042. Cassandra starts running on boot as a system service. This is a key module that must be one of the first to run. Without a database, Pet Sense system cannot store the message and cannot provide access to it via REST. This is how Pet Sense provides traceability, persistence and integrity.

Cassandra queries are made using CQL language which is based on SQL. Commands such as CREATE, INSERT, SELECT, FROM, WHERE were the main used commands to manipulate database data.

The CREATE keyspace or CREATE table commands, as the name implies, creates the keyspace and table for storage. Using the expression “IF NOT EXISTS” the Cassandra system is capable of checking if the keyspace and tables exist before creating and if so, the system executes the following commands on the existing keyspace and does not create duplicates. In this command the specification for the column names and types should be enumerated.

INSERT INTO table JSON is the line of commands that allows for the insertion of the key values in the form of a JSON string into the database. Here is an example:

INSERT INTO petsense.allTable JSON '{"id": "ab9de2c8-c784-43d0-b7e5-2b6167e83c1c", sensor_timestamp: "2018-06-17T18:45:12.000Z", "clinic_timestamp":"2018-06-17T18:45:12.000Z", "animal_state": 1, "temperature": 38.12, "hr": 71, "animal_id": 23, "animal_type": 3} ';

These two commands are use in the Pet Sense Connector for creating and input the data into Cassandra each message constituting a row. The SELECT, FROM and WHERE commands are executed much like SQL in Pet Sense Sharer, to extract the data from the database, as the following example shows:

SELECT id, clinic_timestamp, temperature, hr, animal_state, animal_type FROM petsense.allTable ;

There are two tables created in Cassandra for Pet Sense, mirroring Kafka topics: allTable and alarmTable. One contains all JSON messages, the other only the alarm JSON messages. Each message being a row in the DB and ordered by timestamp as Image 28 shows.

```
 id                                   | clinic_timestamp                | temperature | hr | animal_state | animal_type
--------------------------------------+---------------------------------+-------------+----+--------------+-------------
 ab9de2c8-c784-43d0-b7e5-2b6167e82b0c | 2018-06-17 18:46:00.000000+0000 |       38.15 | 71 |            1 |           3
 ab9de2c8-c784-43d0-b7e5-2b6167e82c04 | 2018-06-17 18:45:00.000000+0000 |        38.1 | 70 |            1 |           3
 ab9de2c8-c784-43d0-b7e5-2b6167e82a0c | 2018-06-17 18:44:00.000000+0000 |        38.2 | 72 |            1 |           3
 ab9de2c8-c784-43d0-b7e5-2b6167e82c1c | 2018-06-17 18:43:00.000000+0000 |          38 | 71 |            1 |           3

(4 rows)
```

Image 28 - Database ordered by timestamp

Calling the Cassandra java object with the *query* method, a String can be passed, as the previous example line of INSERT illustrated, to input the json string message into the DB and getting the result of Image 29

```
 id                                   | clinic_timestamp                | temperature | hr | animal_state | animal_type
--------------------------------------+---------------------------------+-------------+----+--------------+-------------
 ab9de2c8-c784-43d0-b7e5-2b6167e82b0c | 2018-06-17 18:46:00.000000+0000 |       38.15 | 71 |            1 |           3
 ab9de2c8-c784-43d0-b7e5-2b6167e83c1c | 2018-06-17 18:45:12.000000+0000 |       38.12 | 71 |            1 |           3
 ab9de2c8-c784-43d0-b7e5-2b6167e82c04 | 2018-06-17 18:45:00.000000+0000 |        38.1 | 70 |            1 |           3
 ab9de2c8-c784-43d0-b7e5-2b6167e82a0c | 2018-06-17 18:44:00.000000+0000 |        38.2 | 72 |            1 |           3
 ab9de2c8-c784-43d0-b7e5-2b6167e82c1c | 2018-06-17 18:43:00.000000+0000 |          38 | 71 |            1 |           3

(5 rows)
```

Image 29 - Database with new line ordered by timestamp

Pet Sense Sharer uses the SELECT command to query the DB and retrieve the JSONarrays as java objects with specific values.

As demonstrated, with Cassandra database, writing and reading data from it is effortless. The integration of Cassandra onto Kafka connector is well documented in Kafka and in Cassandra's documentation as it is one of the most coupled tools for Kafka.

5.2.6 Front End

The front-end of Pet Sense was created with the cooperation of Pet Universal developers to demonstrate the integration of Pet Sense and Pet Universal software suite. It is written in *EmberJS* and uses a graphic library called *ember-cli-chart*.

The Pet Sense front-end part is composed of 4 visualisations for each animal (current temperature and heart rate, a line graph with some past values of temperature, another one with the bpm, the table with all values for search, the alarm view and the history of alarms). It will appear at the bottom corner of the animal description web page.

One of the main objectives of this project is to integrate this software and infrastructure with the company's products. That objective was met on the Patients tab of Pet Universal software.

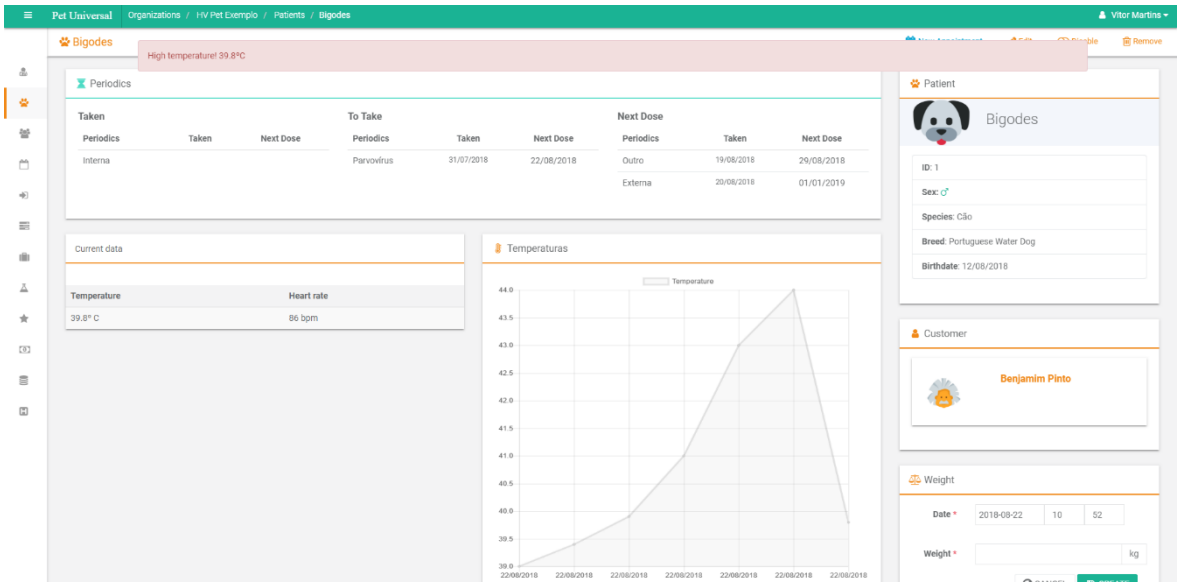

Image 30- Front-end of Pet Universal software

When the user selects a specific patient (one that is under Pet Sense's monitoring), a graph area is displayed with the history values and timestamps of temperature on the bottom centre of Image 30, and the current values of temperature and heart rate on the left. Every 5 seconds the front-end refreshes and through webservices available by Pet Sense, it queries if there are any alarming messages. If they exist, an alarm pop-up message appears on the top of the page with the reason of the alarm.

As an ongoing integration, the development of this feature is still being developed by Pet Universal team, and so, for demonstration and testing purposes, Kibana was implemented as an alternative for Pet Sense's frontend for the validation part of this project.

5.2.7 Android Application

An Android Native Application was developed for Pet Sense with four distinct purposes:

1. Caregivers receive notifications on their smartphone whenever an alarm is emitted by the system;
2. Veterinaries can manually input the temperature and heart rates from live animals from the app to the Pet Sense System. It would break the streaming paradigm, but it would serve as an alternative if the smart collars are not installed;
3. Aid veterinary professionals accessing Pet's software content and tasks;
4. Provide a new system of identification of patients using the smartphone.

The first feature allows the users to get notifications immediately after the alarm is processed in Pet Sense Processor, since there are some vet unit which rarely look at the monitor and have asked Pet Universal for a better solution. This was implemented with the help of Firebase Cloud Messaging.

Firebase is a Google solution that gives support to mobile apps in vast computational areas like Analytics, Databases, Notifications and Crash Reports. Firebase Cloud Messaging (FCM) is a service that allows the quick broadcast of notifications to mobile devices. It uses a method called "Push Notifications" to distribute mobile notifications to a group or a specific smartphone(s). It is designed for many platforms such as Android, iOS, Web applications and Unity.

The second feature is enabling vets to feed the system through the Pet Sense Mobile App with monitoring values that were physically measured, given that the hospitalization episode does not have smart collars mounted or available. This way, the app is able to input data to the system using the MQTT port for Mosquitto and the system pipeline receives this as part of the data stream messages that are processes and stored the same way.

By using the fcmjava-client-2.1 library, the Pet Sense connector is able to send each alarm message to the trusted smartphone which emits this notification in the system even if the Pet Sense App is not running. This emission is an HTTP request done on the java code of the Pet Sense connector level that is sent to the FCM service host (https://fcm.googleapis.com/fcm/send) with the specific authentication using public and private keys, that then is processed by the SDK of the FCM. This feature of Firebase even lets the web app tag the notification with a priority (high or normal). These notifications are in JSON format contained in a String.

This Android Pet Sense App was created with a compatibility minimum API of "19" which covers 90,1% of the software in Android smartphones and was tested with an Android smartphone of API 23 with an Android 6.0 version. It is composed of four Activity views: login with splash screen,

listing the clinics, a main view with three tabs (list hospitalization episodes, list tasks and submit monitoring values), and patient information view. Image 31 shows the "storyboard" of the Pet Sense Mobile.

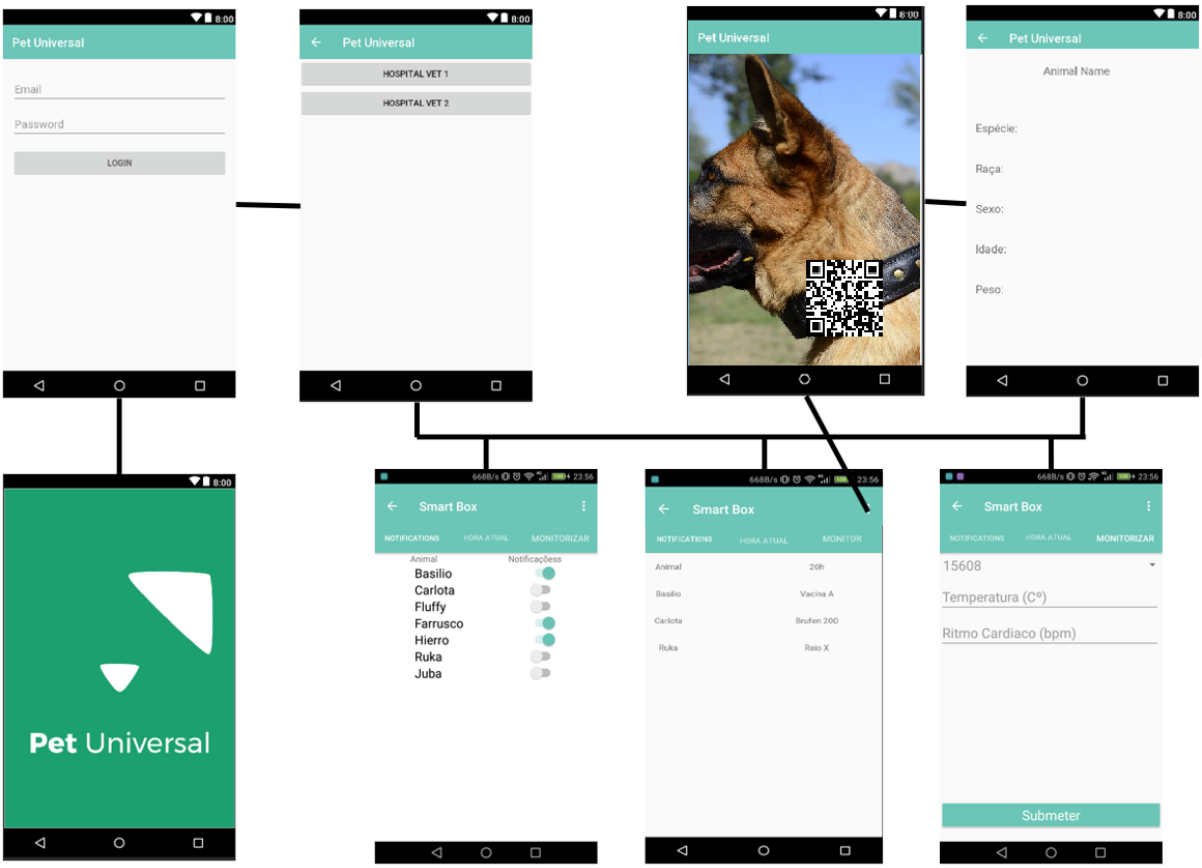

Image 31 - Storyboard of Pet Sense Mobile

Before the login page is shown, a splash screen appears for three seconds to present the brand and warn the user if an internet connection is not recognized, as this is a internet dependent app. The login page is composed of standard login features: two text views to input an email and the corresponding password, and a login button to attempt authentication. If the login is unsuccessful, an error pop-up message appears asking the user to try again. If the login is successful, a list of clinics/hospitals, that the user is associated with, appears in the form of clickable buttons, using webservices that communicate with Pet Universal's backend. Each button leading to the main view, where information varies according to the clinic chosen.

Once a clinic is selected, the user is led to the main page as Image 31 shows. The first tab contains a list of the stays of the given clinic id, represented by the patient's name. At the right side of each patient's name, a switch button represents the decision of whether the user wants to receive notifications of alarms from Pet Sense regarding that patient or not. Smartphone receives push notification but first checks if patient switch button is selected and displays notification if so. In Image 31 is an example where only Basilio, Farrusco and Hierro stays are set to be listening for alarms.

The second tab is still under construction by Pet Universal and as it is not the focus of this project, it will not be explored further. The third tab is the monitoring tab. It contains a selection bar on top which lets users pick the id of the animal they are monitoring, two number views to input the temperature and bpm values, and a submit button. Once all three fields are filled, the submit button must be pushed so that a MQTT message is sent to Pet Sense's virtual machine. Once pressed, a pop-up message will appear saying the message was sent successfully or unsuccessfully.

In the option menu on the top right corner of the main view there is an option to check the id of a given patient using the camera of the smartphone. Using QR code technology, the app is able to recognise a QR code that contains the id of the clinical stay of that animal. Each animal should have a unique id and a unique QR code. If the reading is successful, the user is lead to an information page where all information about that animal is retrieved from Pet Universal's backend and presented on screen, as Image 31shows.

All these views were built keeping in mind basic usability heuristics (back button for navigability).

6 SYSTEM VALIDATION

This is an academic project that needs to comply to the initial demands of Pet Universal, in the sense that, it must be testable and trust-worthy to integrate as one of their product modules. This chapter demonstrates the work done to prove the utility and operability of Pet Sense.

After implementing the system, efforts were made to decide the best way to validate Pet Sense. On one hand, Pet Universal wanted to put the system in "veterinary hands" as soon as possible, on the other hand we wanted to fully integrate Pet Sense as a module for Pet Universal. As first validation proof, in the previous section 5.2.6, Image 30, it was possible to see the integration with Pet Universal software. To prove the validation of the system as a true monitoring system capable of sending alarms, two more validations had to be made: mobile app receiving alarms as push notifications and feeding the system with datasets that portray real animal vital signs and simulate a fever or heart failure.

As we could not have applicable smart collars providing real animal vital signals we had to simulate them with a data set. This section will illustrate the dataset used and the result of the input of this dataset. It is expected that the system filters all messages and that the alarms are brought directly from the "alarmTopic" in Kafka and visualized inside Kibana. As it was not expected to be able to test the operability of the alarmistic feature in a real environment, a selection of simulated scenarios also had to be simulated in the datasets and demonstrated in the following sections.

6.1 Datasets

The data sets were built with a python algorithm that read temperature and BPM values from a file and then sent them periodically to the server (1 second apart). The file was composed of 100 lines with a float value for temperature, followed by an integer value of BPM separated only by a comma as Table 13 shows.

Temperature	BPM
68	120
68.10	120
…	…

Table 13 - Dataset of temperature and BPM of a dog type 4

The first 39 values were based on real values from an American Terrier dog of 38 kg monitored in *Hospital Veterinário de Aveiro* in early 2018. At the 40th line, temperature starts to rise rapidly from normal 38 to 42 in just 20 seconds. An alarm is expected here. The temperature then starts to

stabilise, simulating that the dog has been taken care of, and its temperature goes back to normal after 10 alarming messages.

6.2 Validation with datasets

The elastic stack was installed as a docker image version 6.3.1 to provide a demonstration of the Kafka messages and representing the Web Dashboards of Pet Universal front-end. Elastic was implemented only because it is a way to present the data, given that the access to Pet Universal software is restricted and under revision and development. Having experience in the implementation of this stack it seemed like a quick fix to this temporary deficiency.

Logstash was configured to get all content of Kafka topics, read it as JSON format and send it to Elasticsearch port 9300 to be parsed and make it available for Kibana. As Image 32 shows, the Kibana dashboard contains two views for the current values of heart rate (simple label view), the temperature (in the form of a ranged thermometer), a historic graph showing the amount of messages parsed in a given timestamp, and a line graph contain both temperature and heart rate values parsed in the last 2 minutes. This the dataset described, the system reacted as expected. The “alarmTopic” registered the 20 alarming messages as expected.

Using Kibana, it is possible to see the processed messages. Kibana view with normal values can be seen in Image 32.

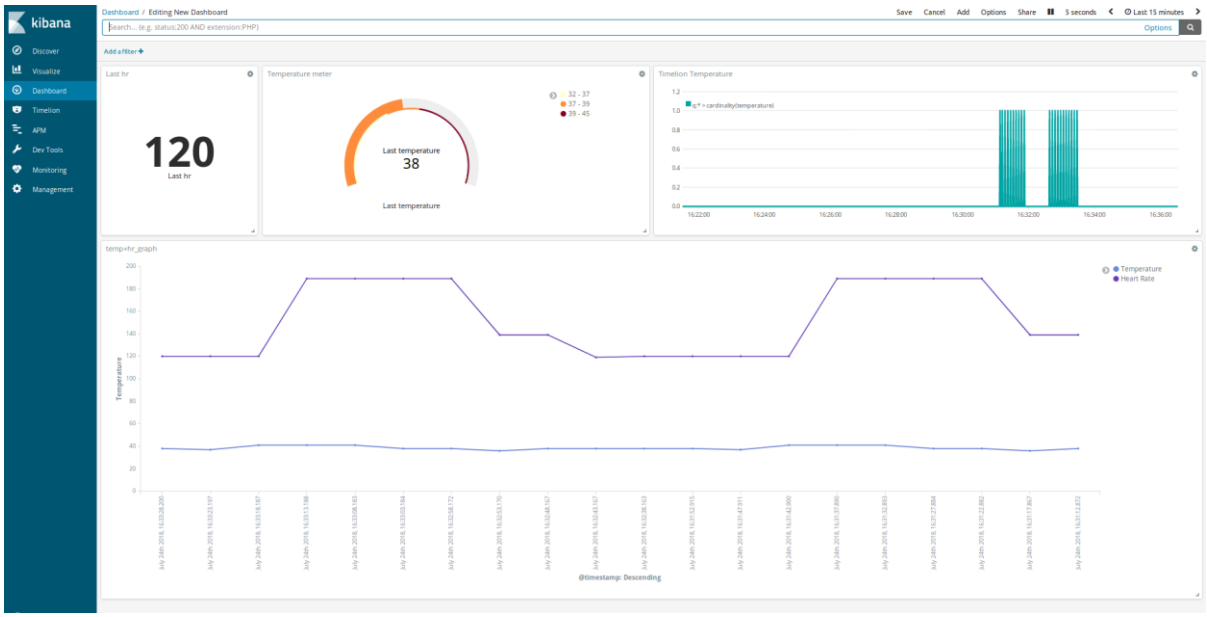

Image 32- Kibana screenshot of normal sensed values

As Image 33 demonstrates, the value of temperature is above normal and so the colour in the temperature view changes to red, meaning that the value is an alarm. The line graphic shows a rise in the temperature above the normal limit and then a regression to normal, this happens two times in the dataset, and happened two times in this simulation test, as expected.

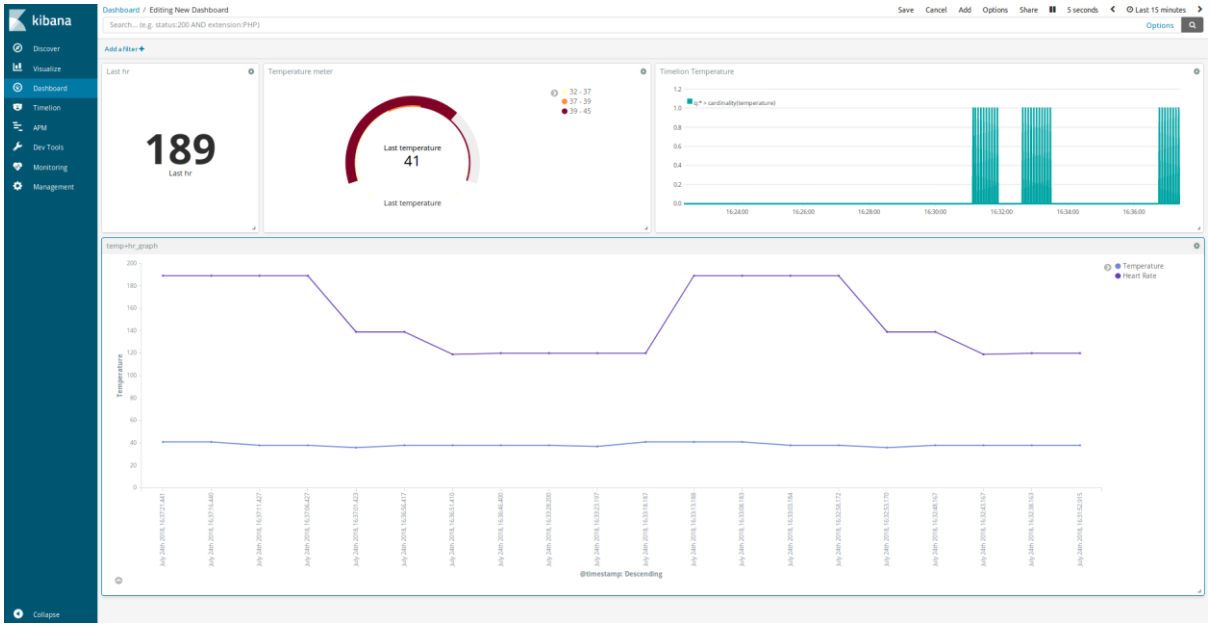

Image 33- Kibana screenshot of abnormal sensed values

7 CONCLUSIONS

The protection of animals, through regulation of best veterinary care provision, is present in modern societies. The digital transformation of hospital stays monitoring, as proposed in this project, aims to support the best practices in pet care. Innovation based on the Internet of Things concept has proven to generate economic value to many companies and organizations as it creates higher standards of efficiency on the control over “things”. The area of veterinary medicine offers great investment opportunities for IoT technologies and services.

The choice of implementing an IoT system is due to the fact that it provides extensibility via integration with other systems on the Internet; scalability of new sensors/ animals/clinics is easier to ensure; availability not only for vets all over the world, and also for the pet caregiver to check the state of his/her patient at any point in time and space; and finally standing as a state of the art solution.

The literary research done, captured a large range of scientific areas enabling a complete notion of the spectrum of the implementation purpose and context. It serves as reasoned justification for the choices made in the implementation. The process of experimenting according to the research, allowed the author to get a practical experience in IoT stream and logging opensource tools, which in term served as base for the choices in the final implementation.

As shown in the previous chapter, the system was validated in an environment with close-to-real data. With a simulated input where an alarm was expected, the system was able to perform as expected with “normal” messages, as well as with alarm messages, arriving as expected to the destinations (Kibana and Android), when expected. Also shown in the previous chapter this implementation is ready for full integration as a module in the Pet Universal’s product.

This project was created as a partnership between LVS company and the *Universidade de Aveiro* to form an internship that enabled a masters’ student to make part of the company’s team and develop software that would integrate the market product of the company. It was expected that the developer would feel part of the team and be provided with everything needed for the objectives of the implementation be met.

This chapter describes a retrospective on the work done throughout the year, from a technical to a theoretical perspective and even adding some personal views. The following sections are divided into different contexts and enumerating the author’s opinion on the research made and implementations.

7.1 Internship feedback

From a technical perspective, the objectives for this thesis were to do literary research on the topic, propose a software architecture, implement it, and provide Pet Universal's system access to the data. It was also an objective to learn as much as possible from a start-up business environment and team. The ten-month experience was an enriching way to get office experience and share knowledge about state of the art tools and software development issues, in both ways.

During the ten months in the Pet Universal environment, there was always support and feedback from the company's members. Every doubt or setback was often discussed together, giving helpful feedback and insight. The author was integrated as a member of the team, and actively participated in the company's activities and decisions. There was a lot learned in terms of business environment and business procedures for software that runs internationally. As well as software development cycle and software development practices.

This internship allowed the author to have first practical contact with work methodologies such as Kanban, Scrum and the Agile mindset. Pet Universal started implementing these methods right before the internship started and so, there were quite a few adjustments that were made throughout.

Scrum is a framework composed by a set of practices and tools in the workspace that allow the constant communication between the members, set milestones and divide the work load in smaller tasks. One the practices implemented by Pet Universal was the "daily meetings". Daily meetings allow the team to have daily feedback on the colleague's progress while sharing difficulties or obstacles each member has, in order to find solutions that unblock them together. This is an interactive way for the team to briefly discuss the day's work and refactor some bad practices/ problems.

The Kanban tool allows the participants to have a visual control over the work load of a project. It organizes the tasks according to its priority and state. Each task is linked to a team member and updated by him/her. This enables constant feedback on the progress of the project in a specific timeframe. This was a useful way to keep track of the tasks needed, in progress, blocked or done. This is an excellent practice, applicable in any context of team work and even in personal projects.

It was also the first time the author had practical contact to software development procedures such as the DevOps cycle and its tools. Pet Universal implements software engineering tools that promote code automation, monitoring, integration, testing, adequate to the client specifications. Software quality assurance systems that involve software repository (Visual Studio), versioning tools (Git), code level test specifications (Junit and Mock tests), Artifactory (JFrog), continuous integration (Jenkins). The software developed was delivered in Pet Universal's Visual Studio repository, making use of the DevOps toolchain.

At the end of the internship, the team was satisfied with the performance and development of the author, having fulfilled the initial requirements of the project. It was a fruitful cooperation that led to this thesis as a result.

7.2 Experienced difficulties

Along the way many projects suffer setbacks and delays. One of the first setbacks felt was the lack of experience in stream software systems, and the lack of focus on the scope of the project for the state of the art research (took too much time to realize that log monitoring systems and stream monitoring systems are not the same).

Not having the adequate hardware to test the “smart collars” in a living animal was detrimental to the final validation of the system. A lot of time and work had to be put into simulating such sensors with what was available in the university to be as close as possible to a “biometric collar”.

Regarding non-technical setbacks, it is difficult to work in a start-up as it is an ever-changing environment which can be challenging in many levels, and a true setback for stability of any project. Pet Universal changed office space two times during these ten months, participated in four technology and entrepreneur fairs and had three software releases, all this during processes of internationalization and rebranding. It is constantly improving the software and its modules: adjusting API interfaces and clearing the development user information database for memory crush purposes which is a setback for the front-end development and login accesses. As all professional setbacks, they make us more experienced and improve our carrier. The entrepreneurial mindset is something that is essential to succeed in any enterprise life gives us.

7.3 Lessons Learnt

On a more personal level, as a first job experience, there was a lot learnt on what working on a start-up company looks and feels like, in many aspects. One of the things learnt working in this environment is the importance of the communication between colleagues and having a relaxed work environment for the success of team projects. There is an increment in the sense of responsibility when developing software in comparison to university projects, as the consequences of the results are directly linked to client and investor satisfaction, thus affecting the overall success of the start-up.

As a year-long project a lesson was learnt about the importance of self-motivation and self-discipline in a one-man-team. Setting clear and objective tasks and milestones is key to deliver the project on time. Most tools and frameworks used in the implementation were the first contact the author had with them, contributing to the author’s knowledge about “full-stack” IoT tools and projects.

Also worth mentioning is the knowledge acquired concerning the veterinary context and clinic processes. Working with breadboards and sensor wiring was a novelty for the author who became an enthusiast about constrained devices and their immense applications.

7.4 Future work

On software projects, there is continuous evolution of computational innovations with more efficient and effective tools being released every day. The necessity for improvement and incremental updates in software is inevitable, especially for SaaS products that sell to the businesses that evermore demanding.

Of course, for this project to be integrated with Pet Universal's system and sold as a module, a smart collar must be built with all the sensors and Bluetooth connection, with the help of an electronic engineer. New features and usability tests could be planned and discussed with clients, such as:

- Replicating the mobile app for the iOS system. This would increase the use of the app through Pet Universal clients and Pet Sense users;
- Integrating a smart collar/device adaptable to the animal. In the state of the art a lot of projects were cited that had "animal-proof" devices integrated. In the future this is a wanted outcome. To continue to develop devices capable of integrating Pet Sense and also be wearable for pets, in a non-intrusive way;
- Adding more sensors to the "smart collar" and all that it inherits. Besides body-temperature, ambient temperature and heart rate, other metrics can be integrated in Pet Sense. Pet Sense has the capacity for this, only adding key values to the json messages and refactor little pieces of the implemented pipeline;
- Creating a mobile app that provides feedback to the pet owner on the well-being on his/her pet and recovery. Pet Sense is capable of providing a way for final users to subscribe to specific Kafka-topics, thus getting the vital signs of their pets in their smartphones, mimicking Pet Universal front-end.

As the present system is concerned, performance and load tests were not made for the final implementation, to test how much data the system can handle without losing any message or crushing the system. Furthermore, according to some literary research made, switching to a timestamp-oriented database would optimize the database's efficiency, as these are timestamp-oriented messages. The different types of load tests could be done according to the number of animals, vet units and smartphones. Confluent assures low latency as it supports "trillions of messages" and this was not tested. Regarding the number of smartphones, the implementation and tests only included one smartphone, but Cloud Messaging Firebase documentation mentioned a feature to allow group notifications of alarms to be sent synchronously (each clinic's smartphone belong to a certain clinic group).

Basic security and privacy issues were taken in consideration in implementation, but they were never a priority. In terms of cybersecurity, connecting devices to the Internet poses serious threats (DoS or fishing) to the protection of the data and infrastructure is a priority for the public product release. As one of the biggest challenges nowadays in IoT projects is the privacy of data, data protection might be a point to improve in this project with the use of cyphers to encrypt the messages,

and internet security measures like HTTPS TLS / SSL, for example. The Kafka MQTT connector used in the project is ready for the integration with TLS / SSL protocols.

When installed in a fair amount of vet hospitals and clinics, this project will allow for scientific studies on various fields, for example: building a central platform that cross-checks data from hospitals and clinics, creating a dataset of animals' vital signs generates opportunities for veterinary studies on how a given breed of pet that undertake the same veterinary treatment reacts to a certain drug/ treatment. With a lot of data on this, proof on how effective a certain treatment is on given breed is key to improve the way animal illnesses are treated in pets. And this could even be done through data science software (Machine Learning) that recognizes these patterns inside the Pet Sense system.

8 REFERENCES

[1] Centro Nacional de Cibersegurança, “A Internet das Coisas.” [Online]. Available: https://www.cncs.gov.pt/a-internet-das-coisas-iot-internet-of-things/. [Accessed: 10-May-2018].

[2] L. N. Robert Stackowiak, Art Licht, Venu Mantha, *Big Data and the Internet of Things, Enterprise Information Architecture for a New Age*. Apress, 2015.

[3] F. Azevedo, “Portugal é um país Pet-Friendly,” *GfK*, 2015. [Online]. Available: https://www.gfk.com/pt/insights/press-release/portugal-e-um-pais-pet-friendly/. [Accessed: 30-Nov-2017].

[4] A. Khelifi, R. Al Hamli, S. Al Tamimi, and R. Al Ali, “An Automated System for Monitoring Horses Vital Signs Using Heart Beat Sensors,” in *2017 Palestinian International Conference on Information and Communication Technology (PICICT)*, 2017, pp. 53–59.

[5] A. Jukan, “Smart Computing and Sensing Technologies for Animal Welfare: A Systematic Review,” *ACM Comput. Surv.*, vol. 50, no. Article 10 (April 2017), pp. 1–27, 2017.

[6] X. Liu, T. Yang, and B. Yan, “Internet of Things for Wildlife Monitoring,” 2015.

[7] J. P. Dominguez-Morales *et al.*, “Wireless Sensor Network for Wildlife Tracking and Behavior Classification of Animals in Doñana,” *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2534–2537, 2016.

[8] Y. G. Sahin and Y. Guneri, “Animals as Mobile Biological Sensors for Forest Fire Detection,” *Sensors*, vol. 7, no. 12, pp. 3084–3099, Dec. 2007.

[9] R. Umega and M. A. Raja, “Design and implementation of livestock barn monitoring system,” in *2017 International Conference on Innovations in Green Energy and Healthcare Technologies (IGEHT)*, 2017, pp. 1–6.

[10] K. Bhargava, S. Ivanov, C. Kulatunga, and W. Donnelly, “Fog-enabled WSN system for animal behavior analysis in precision dairy,” in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 504–510.

[11] T. T. Zin, I. Kobayashi, P. Tin, and H. Hama, “A General Video Surveillance Framework for Animal Behavior Analysis,” *Proc. - 2016 3rd Int. Conf. Comput. Meas. Control Sens. Network, C. 2016*, pp. 130–133, 2017.

[12] S. H. Kim, D. H. Kim, and H. D. Park, “Animal situation tracking service using RFID, GPS, and sensors,” *2nd Int. Conf. Comput. Netw. Technol. ICCNT 2010*, pp. 153–156, 2010.

[13] A. Tavares, “MONITORIZAÇÃO ANIMAL BASEADA EM TECNOLOGIAS IoT (SHEEPIT)”, Universidade de Aveiro, 2017.

[14] A. Bozkurt *et al.*, “Toward Cyber- Enhanced Working Dogs for Search and Rescue” in IEEE Intelligent Systems, 2014.

[15] S. Mealin *et al.*, “Towards the Non-Visual Monitoring of Canine Physiology in Real-Time by Blind Handlers” In Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology. ACM, 2015. p. 66.

[16] J. J. Majikes *et al.*, “Smart connected canines: IoT design considerations for the lab, home, and mission-critical environments,” *37th IEEE Sarnoff Symp. Sarnoff 2016*, pp. 118–123, 2017.

[17] P. Suksaengjun, D. Thanapatay, S. Nobuhiko, and J. Chinrungrueng, “The design of smart dog cage system,” in *2016 Management and Innovation Technology International Conference (MITicon)*, 2016, p. MIT-161-MIT-165.

[18] S. Neethirajan, “Recent advances in wearable sensors for animal health management,”

Sens. Bio-Sensing Res., vol. 12, pp. 15–29, 2017.
[19] Animal Planet, “What’s a dog’s normal body temperature?” [Online]. Available: http://www.animalplanet.com/pets/what-is-a-dogs-normal-body-temperature/. [Accessed: 15-May-2018].
[20] G. Lemasson, P. Lucidarme, D. Duhaut, and R. D. Of, “Real-time Detection of the Activity of a dog” in Nature-Inspired Mobile Robotics, 2013.
[21] J. S. L. Ting, S. K. Kwok, W. B. Lee, A. H. C. Tsang, and B. C. F. Cheung, “A Dynamic RFID-Based Mobile Monitoring System in Animal Care Management Over a Wireless Network,” in *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, 2007, pp. 2085–2088.
[22] R. Brugarolas *et al.*, “Wearable wireless biophotonic and biopotential sensors for canine health monitoring,” in *IEEE SENSORS 2014 Proceedings*, 2014, pp. 2203–2206.
[23] N. Sellier, E. Guettier, and C. Staub, “A Review of Methods to Measure Animal Body Temperature in Precision Farming,” *Am. J. Agric. Sci. Technol.*, 2014.
[24] Z. Sheng, S. Yang, Y. Yu, A. V Vasilakos, J. A. Mccann, and K. K. Leung, “A Survey on The IETF Protocol Suite for The Internet-of-Things: Standards, Challenges and Opportunities,” 2013.
[25] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, “A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications,” *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1–1, 2017.
[26] M. Marjani *et al.*, “Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges,” *IEEE Access*, vol. 5, pp. 5247–5261, 2017.
[27] J. Pan and J. McElhannon, “Future Edge Cloud and Edge Computing for Internet of Things Applications,” *IEEE Internet Things J.*, vol. 4662, no. c, pp. 1–1, 2017.
[28] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Commun. Surv. TUTORIALS*, vol. 17, no. 4, 2015.
[29] iot.eclipse.org, “Eclipse Paho.” [Online]. Available: https://www.eclipse.org/paho/. [Accessed: 01-Jun-2018].
[30] Pivotal, “RabbitMQ.” [Online]. Available: https://www.rabbitmq.com/. [Accessed: 24-Oct-2018].
[31] C. H. Huang, P. Y. Shen, and Y. C. Huang, “IoT-based physiological and environmental monitoring system in animal shelter,” *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, vol. 2015–Augus, pp. 317–322, 2015.