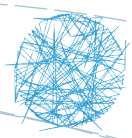
DM



LARSyS
Laboratory of Robotics
and Engineering Systems

arditi

Real-time Application Programming Interfaces for Depicting Aquatic Internet of Things

MASTER DISSERTATION

Fábio Mateus Rodrigues Abreu
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

December | 2020

Real-time Application Programming Interfaces for Depicting Aquatic Internet of Things

MASTER DISSERTATION

Fábio Mateus Rodrigues Abreu
MASTER IN INFORMATICS ENGINEERING

ORIENTATION
Marko Radeta

CO-ORIENTATION
Filipe Magno de Gouveia Quintal

FCEE

Mestrado em Engenharia Informática

Real-time Application Programming Interfaces for Depicting Aquatic Internet of Things

Fábio Abreu

supervisionado por
Prof. Dr. Marko Radeta and Prof. Dr. Filipe Quintal

17 de Maio de 2021

Abstract

Although recent years portray an increase demand for Internet of Things (IoT) applications in aquatic setting, there is a lack of standardization in collecting and displaying these data to a wider set of audiences ranging from marine biologists, whale-watching companies and environmentalists. More flexible APIs and long-range data access are necessary, providing the facilitated remote access to the data, while reducing significantly the cost of fuel and time when obtaining the data from oceanic settings.

The main goal of this thesis is to produce the robust back-end and an API for: (i) managing the IoT devices to be applied in aquatic setting; (ii) obtaining the status and the telemetry in real-time; and (iii) visualizing the collected data from IoT devices such as temperature, pressure, humidity, luminosity, GPS position, etc.

The final product advances the state of the art in back-end development for collecting, storing and displaying larger datasets (e.g. collected telemetries, radio transmission data) in Single-page applications (SPAs). It will, moreover, use the latest back-end and front-end development techniques (e.g. React.JS, Laravel) while optimizing database querying, and providing the real-time access to the data on any device, and without the need of refreshing the page.

Keywords: Internet of Things · Software Engineering · Application Programming Interfaces · Backend Development · LoRa · Biodiversity Assessments · Information Visualization.

Resumo

Embora nos últimos anos tenha existido um aumento no desenvolvimento de projetos na área da Internet das coisas (IoT) em ambientes aquáticos, não existe uma padronização na recolha e exibição dos dados obtidos com esses mesmos projetos de modo a possibilitar o seu aproveitamento por um conjunto diverso de utilizadores, que variam desde os biólogos marinhos, passando pelas empresas de observação de baleias até aos ambientalistas. Para que tal seja possível são necessárias APIs mais flexíveis e acesso a dados de longo alcance, fornecendo acesso remoto facilitado aos dados, reduzindo significativamente o custo de combustível e tempo ao obter os dados de configurações oceânicas.

O principal objetivo desta tese é desenvolver um back office robusto e uma API para: (i) gerir os dispositivos de IoT a serem utilizados em ambientes aquáticos; (ii) obter o estado e a telemetria em tempo real; e (iii) visualizar os dados recolhidos pelos dispositivos de IoT como por exemplo, temperatura, pressão, humidade, luminosidade, posição do GPS, etc...

O produto final contribui para o avanço da tecnologia, pois providencia um back office para recolher, guardar e exibir um grande conjunto de dados (por exemplo, multimédia recolhida, telemetrias, dados de transmissão de rádio) em aplicações de uma única página (SPAs). Além disso, utilizará as mais recentes técnicas de desenvolvimento de back-end e front-end (por exemplo, React.JS, Laravel), otimizando a consulta à base de dados e fornecendo o acesso em tempo real aos dados em qualquer dispositivo, e sem a necessidade de atualizar a página.

Keywords: Internet das Coisas · Engenharia de Software · Interfaces de Programação de Aplicações · Desenvolvimento de Backoffice · LoRa · Avaliações da Biodiversidade · Visualização de Informações.

Acknowledgements

This study is part of LARGESCALE project with grant no. PTDC/CCI-CIF/32474/2017 by Fundação para a Ciência e a Tecnologia (FCT) and Portuguese National Funds (PIDDAC). It is also supported by the funding by project UID/EEA/50009/2019 and UIDB/50009/2020 (ARDITI/LARSyS) as well as by Programa de Cooperación INTERREG V-A España-Portugal MAC (Madeira-Azores-Canarias) 2014-2020, throughout project INTERTAGUA (Interfaces Aquáticas Interativas para Deteção e Visualização da Megafauna Marinha Atlântica e Embarcacoes na Macaronésia usando Marcadores Rádio-transmissores), with Ref: MAC2/1.1.a /385. Moreover, conducted studies were supported by the additional funding from project MITIExcell - EXCELENCIA INTERNACIONAL DE IDT&I NAS TIC (Project Number M1420-01-01450FEDER0000002), and project SUBMARINE (Simulators for Understanding the Biodiversity using Mobile and Augmented Reality as Interactive Nautical Explorations), provided by the Regional Government of Madeira. Also, it was supported by the project INTERWHALE - Advancing Interactive Technology for Responsible Whale-Watching, with grant no. PTDC/CCI-COM/0450/2020 by Fundação para a Ciência e a Tecnologia (FCT). Moreover, it is supported with the data provided by the PROYECTO MARCET (MAC/1.1b/149) and MARCET II (MAC2/4.6c/392) "Fomento de la actividad ecoturística de observación de cetáceos como modelo de desarrollo económico sostenible mediante la protección y conservación de estas especies y su puesta en valor como patrimonio natural de la Macaronesia", project Marinfo[1] and Observatório Oceânico da Madeira (OOM)[2]. Finally, project is supported by the Wave Lab[3] - interactive technologies for depicting the anthropogenic impact on marine biosphere.

[1] `https://marinfo.lsts.pt/en/home`
[2] `https://oom.arditi.pt/index.php?page=home`
[3] `http://wave.arditi.pt`

Table of Contents

List of Figures

List of Tables

Nomenclature

ADC Analog-to-Digital Converter
API Application Programming Interface
$APRS$ Automatic Position Reporting System
BLE Bluetooth Low Energy
CLS Collecte Localisation Satellites
$CNES$ Centre National d'Etudes Spatiales
CPU Central Processing Unit
CSI Camera Serial Interface
DAB Digital Audio Broadcasting
DAC Digital-to-Analog Converter
DOM Document Object Model
ECG Electrocardiogram
EEG Electroencephalogram
$EEPROM$ Electrically-Erasable Programmable Read-Only Memory
$GPIO$ General Purpose Input/Output
GPS Global Positioning System
GSM Global System for Mobile
GUI Graphical User Interface
HD High-Definition
$HDMI$ High-Definition Multimedia Interface
$HTML$ Hypertext Markup Language
$ICSP$ In-Circuit Serial Programming
IDE Integrated Development Environment
IO Input/Output
IoT Internet of Things
IP Internet Protocol
JSX JavaScript XML
LAN Local Area Network
$LPWAN$ Low-Power Wide-Area Network
LTE Long Term Evolution
$LXDE$ Lightweight X11 Desktop Environment
MCU MicroController Unit
$MQTT$ MQ Telemetry Transport
MVC Model-View-Controller
$NASA$ National Aeronautics and Space Administration
$NOAA$ National Oceanic and Atmospheric Administration
NRT Near Real-time
ORM Object-relational Mapping
PAM Passive Acoustic Monitoring
PHP PHP: Hypertext Preprocessor

PWM Pulse Width Modulation
RAM Random Access Memory
$REST$ Representational State Transfer
SDK Software Development Kit
SIM Subscriber Identity Module
SPA Single Page Applications
SPI Serial Peripheral Interface
$SRAM$ Static Random Access Memory
SUS System Usability Scale
UHF Ultra High Frequency
UI User Interface
USB Universal Serial Bus
USV Unmanned Surface Vehicle
VHF Very High Frequency
WSN Wireless Sensor Network

1 Introduction

While ubiquotous devices are mostly applied in the land-based settings, there is an interest in deploying such devices in aquatic setting. What is generally known, is that such devices remain challenging to be deployed due to the external factors (e.g. waterproofing, wind scales, battery autonomy, corrosion, etc.). While some of the devices can withstand these challenges, there are two important issues which appear when retrieving the data. Typically, most of the devices store the data to their own internal memory, creating an additional cost and effort (time, fuel, human resources. etc), where the user needs to return back to the device, to retrieve the data and exchange battery. Secondly, there is no interoperable solution for collecting and displaying the data, providing diverse technologies in both software and hardware.

In this dissertation, study tackles the aforementioned challenges by providing the universal backend application programming interface, capable for retrieving the data in near real-time. Such application is able also to collect data from diverse kind of IoT devices (and not only), allowing them to be sent directly to a single cloud point, using 3G, Bluetooth, LoRa, etc.

1.1 Motivation

With the global warming being a pertained issue, nations of the world are failing to reach a mutual agreement and climate change is being contested. An increase of ocean literacy is needed to educate the wider population, and increase the sustainable actions. In a recent IMF report, whales are not anymore shown to be protagonist of the USD 4 billion economy, but also an important player in the reduction of CO2. However, such knowledge does not reach the average user (e.g. tourist on a whale watching trip). The necessity for platforms portraying the high level perspective of ocean occurrence is needed, where the diverse sensory input data can be portrayed (salinity, ph, temperature, pressure, or whale migration coordinates etc).

Today, as it will be seen throughout this dissertation, the existing interoperable interfaces, capable for portraying such aquatic data and present that collected data to the wider public, including the marine biologists and researchers, who can use such platforms to obtain the high level perspective of ocean dynamics, and thus, create more sustainable behaviour and incentives on governments, which in return can provide conservation areas (e.g. changing the ship trajectory due to the potential occurrence of juveniles whales and dolphins) have some limitations that prevent their use by a large number of users. These existing systems are too complex, closed and expensive in order to be possible to integrate a device in a simple way, without spending too much time and money and to be able to store and interpret the different types of data that the device in question is capable of collecting. This is the main focus of this dissertation, seeking to develop a system that is able to counter these currently existing limitations and that allows a fast, simple and low-cost integration with sufficient flexibility to configure the types of data collected that are necessary.

1.2 Research Questions

- **RQ1 - How to successfully receive the environmental telemetry in NRT using LoRa?**

Manuscript will produce an apparatus based on LoRa messages and HTTP requests, capable of sending and receiving the payloads, from nodes (end devices) to gateways (middleman), and forwarding to the back-end for further analysis.

- **RQ2 - How marine ecologists evaluate such apparatus?**
  In this question, study will use a microcontroller to simulate the realistic case scenario as well as to analyze the usability of the system by the marine experts and computer engineers, when exploring the API for an IoT device.

1.3 Objectives

Tailored to the aforementioned research questions, this manuscript studies the feasibility of back-end development as a scalable infrastructure, capable to receive multiple sensory input, ranging from existing to novel IoT devices. To reach this goal, several objectives are outlined:

- **[O1. Architecture]** Design of the backend architecture
  This objective will study the ideal and modular architecture which is capable to receive diverse sensory input.
- **[O2. API]** Implementation of the dynamic API structure
  In this objective, goal is to receive the most versatile variable data, having the possibility to configure them in the backoffice.
- **[O3. GUI User]** Provide the visual appealing interface in backoffice, allowing the users to manipulate and explore the collected data.
- **[O4. GUI Admin.]** Provide the interface to perform administrative settings including: creating, reading, updating and deleting the existing devices, capable of communicating with the provided API.

1.4 Structure of the Document

This manuscript analyzes the contribution of software technologies when applied to aquatic setting, providing an interface to gather, store and depict the bio-telemetry assessments. Introductory chapter (Section 1) analyzed issues and opportunities to perform biodiversity assessments, outlining research questions, objectives and contributions of the study. Conversely, prior work and State of the Art analysis reports the known technologies and the effort by the scholars in their applications (Section 2). In further, proposed research apparatus as an IoT device, including the study setup and data inquiry, is described in Section 4. Lastly, Section 5 report the obtained insights from the performance of the proposed apparatus, summarizing the core contributions in Section 6.

2 Related Work

In this section, dissertation reports the current challenges and technologies which are observed in the State of the Art (Section 2.2) section, describing the pros and cons of the diverse solutions for performing biodiversity assessments. Moreover, Prior Research (Section 2.1) reveals the effort by the scholars from both ubiquotus computing and marine ecology, when applying and analyzing such technologies.

2.1 Prior Research

With the objective to satisfy the aforementioned constrains, the goal of this dissertation is to use existing IoT solutions, capable for transmitting and depicting the data obtained from aquatic settings, using LoRa communication. In this section, several topics are analysed, in order to understand the prior effort made by scholars in the following areas: (i) Bio-Telemetry Assessments - describing how the data are obtained from larger distances when applied to biodiversity, (ii) IoT in Marine Environment Monitoring - describing current effort in the IoT applications in aquatic environment, and (iii) Potential of LoRa - describing some of the studies using the long-range protocol and efforts applied in different areas.

2.1.1 Bio-Telemetry Assessments

In one study [29], is described the development of an advanced and non invasive lightweight compliant environmental monitoring system, named "Marine-Skin". This is an IoT device for fish species made from soft-polymers that can adhere to any skin type in order to actively sense and log data. This device has a soft-packaging and endurance to operate in an highly saline water, such as Red sea, in a depth of 2 km for multiple weeks. After performing some tests on sharks, sea bream and also on common goldfishes, this device proved to be non invasive and does not interfere with the species normal behavior, independently of its size or shape.

Also, another study [27] presents the habitat and distribution preferences of 17 Bugio petrels. Between June 2007 and July 2010, a total of 24 GLS-immersion loggers were deployed. These loggers had been attached to the tarsus of each bird using a single cable-tie and also a metal ring sealed with silicone. These biologgers collect the light, that is used for the geolocation and also the salt-water immersion, which is used for activity analysis. From the 24 loggers, only 17 were retrieved which means 70% of successful recovery rate. All birds remained in North Atlantic waters during the the pre-laying exodus, incubation and chick-rearing periods. It was also concluded that exists a high individual variability in migration strategies. Where identified 5 wintering areas: one in central South Atlantic, one close to Cape Verde, two at the Brazilian Coast and another at the southeast coast of the United States.

Moreover, in [4] is described how the authors used tri-axial accelerometer biollogers to predict accelerometric behaviours (swimming, resting, foraging, bursting, and coasting) of bonefish at Eleuthera, Bahamas. The swim flume was also used in order to estimate the relationship between

the speed and the acceleration of the bonefish. For this experience, 5 fishes where tagged with the accelerometer loggers for 5 days. After performed the study, the authors concluded that the bonefish spends the major part of time (57%) resting, 26% swimming and only 17% coasting. In terms of swimming speeds, bonefish exhibited first slow swimming speeds, with an average of 0.18 m/s with some occasional burst swimming to speeds ranging from 4.3 to 6.4 m/s. The study also concluded that the water temperature has influence in the activity of bonefish, which becomes more active at lower temperatures.

In [17], is presented a study to the Arapaima, which is a giant breathing-air fish, after being released in recreational fishing. This study was made in Guyana, and target the Arapaima, wich is a giant air-breathing fish that had been targeted by recreational fishing on the last years. Due the unknown conservation status of this specie, and with the goal to evaluate the sustainability of this activity, a study was conducted using tri-axial accelerometer biologgers before the captured fishes been released, allowing the authors to monitor post-release survival and behaviour. These experience targeted 26 fishes, and from these only 2 of them died at the monitoring time, the death reason was related to the fact that the fishes do not came to the surface to breath air and drown. For the survivor had been calculated the breathing frequency and the overall dynamic body acceleration, that had been used by the machine learning system to the the classification.

Another study, [9], describes the performed study in cheetahs using biologgers. This study has been conducted with the goal of recover long-term concurrent measurements of activity and body temperature of 5 free-living cheetahs over 7 months, at Namibia. The measures were made with implanted sensors, that had been inserted in the cheetahs surgically, after they had been darted and anesthetized. With the collected data was possible to establish some relations between the physical activity, the body temperature, the usual part of day to hunt and also the spent time to hunt.

Also in [16] is described the experiences of development and using of three different generations of biologgers, in this case implantable biologgers. The tests had been made in Eurasian brown bears and also in American black bears. Using these devices was possible to do some investigations about the underlying mechanisms for winter survival, and made some discoveries such as the extreme respiratory sinus arrhythmias that results in an energy conservation but provides an adequate circulation to stay alertness and could response in a fight or flight situation. Were also registered some extreme heart rate variations, such as a 39.4 s asystole and a 240 beats/min sinus tachycardia in brown bears and a 33.8 s asystole and a 261 beats/min sinus tachycardia in black bears. Was also possible to understand better, combining this data with the coordinates collected by a GPS collar, the impact of humans and also some environmental stressors such as drones, road crossings, hunting and others.

Moreover, [31] describes the collaborative work between zoologists, engineers and computer scientists in order to design and implement a device that was capable to collect audio data accurately and continuously and use it attached to the already existing lions tracking collars produced by Africa Wildlife Tracking. This device was attached to eight lions at Zimbabwe and collects not only

audio, using a mono-electret microphone, but also the data collected by a triaxial accelerometer and magnetometer until it's battery fail, which takes between 4 and 10 days. The collected data is used as source of a classification method that has 5 behavioural states: eat,drink, slow, fast and stationary. Using all the features to do the classification, were achieved some really good results with an average per-class precision of 98.5%. This Precision was higher in slow, fast and stationary behaviours with 99% of precision and a little bit lower at drinking and eating behaviour with a precision of 96.2%.

Another approach is described in [10], in this study is presented the design and development of a sensing and logging system with the name of "BioLogger". This system can monitor and record various types of psychological signal, such as EEG , ECG , skin temperature and respiration rate. This system was built with energy efficiency considerations in order to improve the battery life time and be more efficient. The BioLogger have been tested using chickens and all the data measurements worked as expected, and also had been made tests in putting the chickens in different environments.

In [2], is studied if the behaviour of Oystercatcher, which is a shorebird, can be classified using accelerometers. Due the power consumption needed to collect data using accelerometers, it is also analyzed the possibility of do this behaviour classification using the instantaneous speed measurement that is collected by the same biologgers but demands less energy than accelerometers. Three Oystercatchers have been equipped with solar powered biologgers that include a GPS, a tri-axial accelerometer and also an instant speed measure. Eighteen different behaviours have been identified, and with the collected data some classification trees were developed. After all the collected data analysis was possible conclude that the use of instant speed was useful to the classifying of a flying Oystercatchers, but was not so useful to distinguish the movements when the Oystercatchers are not flying.

In [1], is described how the increase of Cortisol levels impact the behaviour of male smallmouth bass engaged in paternal care. At the paternal care period, the male smallmouth bass, which is a freshwater fish, expend a lot of energy swimming in order to protect the brood from predators. This results in activation of the hypothalamic-pituitary-interrenal axis which produces cortisol. In order to study this situation, the effects of experimental manipulation of stress hormone cortisol have been examined on behaviour activity and also on behaviour of male smallmouth bass. In 6 fishes have been attached tri-axial accelerometers, in order to use them as control group and in 10 the levels of cortisol were elevated using implants and had also been attached tri-axial accelerometers. From these 10, 5 have a low cortisol level and the other 5 have a high cortisol level. The results concluded that the cortisol-treated males have a lower physical activity and it could result in a lose of their brood.

In order to make it easier to assimilate and compare the information presented throughout this section, table 1 was constructed.

Authors	Year	Technology	Collected Data	Species
Shaikh et al.	2019	Polydimethylsiloxane	Water temperature Water salinity Depth	Wobbegong shark Seabream Goldfish
Ramírez et al.	2013	GLS-immersion loggers	Light (for geolocation) Salt-water immersion	Petrels
Brownscombe et al.	2014	Tri-axial accelerometer biologgers	Acceleration (x, y, z) Temperature	Bonefish
Lennox et al.	2018	Tri-axial accelerometer biologgers Machine learning	Acceleration (x, y, z)	Arapaima
Hetem et al.	2018	StowAway XTI, mlog_T1A	Body temperature Locomotor activity	Cheetah
Laske et al.	2018	Implantable biologgers, GPS collar	Heart rate, Locomotor activity, Geolocation	American black bear Eurasian brown bear
Wijers et al.	2018	Tri-axial accelerometer biologgers Machine learning	Audio, Accelerometer, Magnetometer	Lion
Hu	2009	Implantable biologgers, MSP430F2274	ECG, EEG, Respiration rate, Skin temperature	Chicken
Oosterbeek et al.	2010	Tri-axial accelerometer biologgers, GPS	Accelerometer, Geolocation, Speed	Oystercatcher
Algera et al.	2017	Tri-axial accelerometer biologgers	Accelerometer	Smallmouth bass

Table 1. Bio-Telemetry Assessments comparison

After analyzing all the research papers presented in this section, we find that there is a variety regarding to the types of data collected by the devices used for the studies. This variety of data should be considered for the development of the platform so that it is possible to predict some type of data that requires special processing, such as the coordinates, where a map with its coordinates should be generated.

2.1.2 Potential of LoRa

With the goal of prevent the overusing and also reduction of the discarded fishing gear, the study [15] reports the implementation of an IoT system that is capable of reporting position and type of fishing gear and also the name of it's user to the fishing boat and also to the control center using LoRa communication. The paper also reports all the calculated link budgets required to implement the desired marine IoT radio network. After the performed tests and after all the calculus made, the conclusion is that the antennas quality is very important to achieve a long range communication using LoRa.

In [3], is presented an analyzes in terms of performance and efficiency of an already existing LoRa transceiver, the XRange device from Net-Blocks that has a SX1272 LoRa transceiver and a low-power ARM micro controller, the STM32L151. There is made a description of it's features and also is demonstrated how this transceiver can be used to build an efficient wide-area network. In particular is demonstrated the implementation of a six LoRa nodes network that is capable of cover 1.5 ha in a built up environment, which achieves a potential life-time of approximately two years using only two AA batteries and delivering the data within 5 seconds and with a reliability of 80%, concluding that LoRa is a good choice for a long range communication if the power consumption is a requirement.

Moreover, in [11], Ao Huang et al. reported an implementation scheme of a marine wireless monitoring system that is based on LoRa and also on MQTT.This network structure is different from the traditional network architecture due the use of LoRa and MQTT combined to achieve the final result. The first used network is LoRa to interconnect the gateways with the sensor nodes and after that, the collected data was sent to the server trough MQTT. To visualize the collected data the back-end management continuously refreshes the monitoring page and allows the users to have a real time monitoring experience.

Another study, [6] where is described an infrastructure that is designed and developed in order to support IoT deployment in health care area targeting mostly the Alzheimer's and dementia's patients. The proposed approach is based on the LoRaWAN protocol stack, using unlicensed frequencies and allowing the use of very low-power radio devices, which makes the use of this technology a rational choice for IoT communication. The paper describes the design of a complete LoRaWAN-based infrastructure, which contain features that had been partly decided according with caregivers, including the capacity of collecting data for future clinical studies, as well as outdoor tracking and also fall recognition. As future work is targeted the night motion surveillance and also the indoor tracking, in order to control the patients also at their home. This project describes some LoRaWAN use advantages such as the achieved range and the free of charges use of this communication technology, but also describes some technological limitations of its use which makes its use limited.

Sanchez-Iborra et al. presents the development of a boat tracking and monitoring system based on LoRa [28]. Regarding the power consumption, range and cost issues of IoT communication in the ocean, the authors of this study decided to implement the communication using LorA, one of the most prominent LPWAN solution. This can be an interesting solution for small sailboats, recreational boats or radio control ships where the energetic restrictions exists. The paper provides a comprehensive overview of the developed solution and also a discussion of the benefits of using this kind of solution when applied to maritime scenarios. The test, made using sailboats have also achieved good results, obtaining good levels of coverage and also link-reliability with limited power consumption. Also the performed tests ensure that this is a good solution to use for IoT communications near to the shore.

Kais Mekki et al. [21] presents a comparison between the three leading LPWAN technologies which are Sigfox, LoRa and also NB-IoT. In this paper, is made a comprehensive and comparative study between these technologies, in order to understand the advantages and disadvantages of each one and also the application scenarios where each one will fit better. It concludes that any of these technologies is an efficient solution to connect smart, autonomous, and heterogeneous devices, but Sigfox and LoRa provides better battery lifetime capacity and also cost. On the other hand, NB-IoT has benefits in terms of quality of service and latency.

In [32] is studied the feasibility of developing a low-cost IoT network based on LoRa. To achieve these low-cost aspect the development was made using off-the-shelf components and also open source software.It is described the used tesbed, which includes gateways, end devices and also some sensors. The achieved results of LoRa network over the 915 MHz unlicensed ISM band in an indoor and also outdoor environment are also presented. After the results are analyzed it concludes that even in a harsh propagation environment, for example, when the gateway is inside a concrete building, the network is able to achieve a good coverage. More specific results conclude that the indoor coverage is enough to cover an seven-story office building with just a minimal packet drop; in the case of an outdoor environment the situation is different due the environmental dependencies. Despite of that, was achieved a communication of 4.4Km with only 15% of packet drop. Was also concluded that the networks parameters such as packet size and spreading factor greatly affects the coverage, for example with a payload size of 1 byte was achieved a packet drop of only 5% and using a payload of 242 bytes the packet drop leads to 90%.

Moreover, in [20] is described an IoT based health monitoring solution where the collected data is send to an analysis module using LoRaWAN technology. The main focus of this health monitoring system is the control of blood pressure, glucose and temperature in people that lives in rural areas, where the cellular network coverage can not provide the data transmission. The main goal using this kind of communication is reduce the long trips costs for the people that live in that kind of areas and need to move to the healthcare facilities in order to provide the collected data. After several experiments, in order to evaluate the capacities of LoRaWAN, this study concludes that the average covered area is around 33 Km^2 when the LoRa gateway is placed in a place with 12m of height from the ground. The study also concludes that the power consumption of this communication technology is at least ten times lower than the other long range cellular solutions, such as GPRS/3G/4G.

In [12] have been designed and developed six Air Quality IoT devices, each one with four different Particulate Matter sensors and deployed at two different location is the city of Southampton in the UK. The used communication protocols in these devices is LoRaWAN, in order to test the city scale LPWAN coverage. After made the tests, the study concludes that the designed and developed devices can be used in the city; that some low-cost Particulate Matter sensors are viable for monitoring air quality and for detecting Particulate Matter trends; and also concluded that LoRaWAN can be used as communication technology where connectivity is an issue.

Another study, [5] describes how to control the entry of fishing vessels in Marine Protected Areas. The Marine protected Areas are reserved areas for reproduction of marine species, where fishing activities are a threat for it sustainability. In order to protect these areas against illegal fishing, an IoT based system that control the vessels entry was developed. The Haversine method is used by the system for calculate the distance between the protected area and the vessel. To do the communications was used LoRa, that was capable to transmit the location of the vessel to the base station within a few kilometers. The system as also a web interface integrated with google maps that allows the users to monitor the entry of the vessels in the Marine protected Areas.

Li et al. [19] present a solution for monitoring sailing using LPWAN. In this study is focused the transmission performance of LoRa technology with the goal to apply LoRa technology to Sailing Monitoring System. The tests had been made in Brazil Olympics sailings in order to evaluate if the performance of packet loss rate and the coverage can be achieved using LoRa, which was achieved. It also concludes that the parameters such as bandwidth and spread factor, influences the coverage and the data transmission time.

In [4], is described how the world record of communication distance using LoRa has been broken, not only one time but twice in the time interval of only 5 hours. Servet is a science project that is supported by Ibercivis and also by a growing community of volunteers. The project goal is to give to the interested people the opportunity of conduct experiments using weather balloons. In one of this experiments day, on 13th of July 2019, where launch 7 balloons, tracked using LoRa, APRS and also satellite, with different sizes at Alfamen, Spain, with a total of 20 experiments from different people. The first world record breaker, achieving 741km and beating the previous record of 702km, was beaten by the probe named PAPe I, that was built by Jose Manuel Cuesta. This experiment actually beats the 702km record several times during the flight, but the 741km mark was achieved when this device made a connection with a receiver that was located in Lisbon, Portugal. The new world record of 766 km, was set by a small probe made by Enrique Torres, the Diana I. This achievement was made when the Diana I was flying at 24859 meters above Ariza, Spain, and it's communications had been received at a ski resort, on a mountain, registering the distance of 766 Km. This is actually a good indicator that this kind of communication technology can be used at the ocean IoT devices due the achieved and verified distances at this event.

In order to make it easier to assimilate and compare the information presented throughout this section, table 2 was constructed.

[4]Retrieved from `https://www.thethingsnetwork.org/article/lorawan-distance-world-record`, on % today

Authors	Year	Technology	Cost	Range	Known Issues
Kwak et al.	2018	LoRa, NB-IoT	-	25 Km	Power consumption, Antenna height
Bor et al.	2016	LoRa NetBlocks XRange SX1272	35 $	-	Waterproof
Huang et al.	2019	LoRa, MQTT, ATmega2560	-	4 Km	Range
Mea et al.	2020	LoRa, PollicIoT	-	31 km	High Frequencies, Large Payloads, Can't use for emergencies
Sanchez-Iborra et al.	2018	LoRa, Arduino mini Pro, NeoN8M GPS,		4 Km	Range
Mekki et al.	2018	Sigfox, LoRa, NB-IoT	-	-	-
Yousuf et al.	2015	LoRa, Raspberry PI 3, Arduino UNO, Arduino MO, FRDM-KL46Z, Arduino M0 Pro	-	4.4 Km	Large Payloads
Mdhaffar et al.	2017	LoRa, Raspberry Pi 3,	-	33 Km	Continuous Transmission
Johnston et al.	2019	LoRa, Raspberry Pi 3,	900 $, 1000 $	-	Large Payloads, Weatherproof
Candido et al.	2019	LoRa, uC32,	-	9.23 Km	-
Li et al.	2017	LoRa, YL-800IL, YL-900IL	-	2 Km	Range

Table 2. Potential of LoRa Comparison

After analyzing all the research papers presented in this section, we find that LoRa is a technology that despite having some limitations, such as long transmission times when the payload is large, using high frequencies and having a moderate consumption of energy, it also has great potential for use in marine environments regarding to its ability to reach long distances.

2.1.3 IoT in Marine Environment Monitoring

Aside from biotelemetry asessments, there are other reported studies when understanding the current status of deploying the IoT solutions in the aquatic setting. One of them [13] describes a novel architecture for enabling on-line communication in marine environment monitoring sys-

tems. The paper focuses on a set of communication technologies that range from the Wi-Fi and LTE protocols to IoT related low data rate communication standards. In order to achieve the best power consumption, in this paper there are also described some taken actions to achieve the desired results, such as switching off all the non used peripherals and also the hungry interfaces that consume a lot of energy, suggesting an option in using LoRa.

Also, [23] describes the design of a marine environment monitoring system based on open source software. This kind of approach offers a low budget implementation due the use of open source tools that are free to use. The proposed system also target the presentation of the collected data to final users in order to be analysed by professionals, or just observed by a non professional user.

Djajadi et all [30], describes the use of UAV as an alternative to the static IoT common monitoring devices such as buoys. This kind of controlled device provide new types of data and also ease of use, in this specific case the developed UAV collects video. The main goal of the above mentioned manuscript is to provide an easy to deploy, flexible and cost effective WSN for the marine environment monitoring, in this case using LoRa to deal with the communication between the UAV and the receivers. In the paper is described the entire process, from the design to the implementation and also are presented tests in a real world scenario.

Moreover, in [26] is also described the development of a USV the Seamote. This device is the first low-cost and also long-range radio controlled USV that is based on IoT and also on LoRa with the goal of be used to collect marine environmental telemetry. This USV is available to collect humidity, GPS position, footage, temperature and also provide an interface for mobile remote controlling. This paper describe the implementation of the Seamote USV, the Seamote Bridge (that is the middle point between the controller and the USV) and also the Seamote App (that was used to control the Seamote USV). As in the present paper the used technology to perform the communication between the devices is LoRa.

Another study, [24], describes the implementation of an IoT device that is capable of monitoring the water pollution levels. The device was built using a Waspmote micro-controller board and some sensors that can measure the temperature, pH, Oxidation- Reduction Potential and Conductivity of the water. The data was stored locally using an SD card and also was sent to the cloud using GSM communication. As in this manuscript, our collected data is sent to the server, however the communication method is different. In this case the use of GSM has some costs and having the issue with the antennas coverage.

There have been substantial work when collecting the data from the remote deployments. In one of the reported work [7], authors describe the development of an IoT module using the micro controller ATMega 2560. On it, some sensors and one Wi-Fi module are coupled, and used to easily deploy in specific geographical area. The equipped sensors are capable to recover environmental data such as content of carbon monoxide, content of carbon dioxide, luminosity, air temperature and air humidity. Such collected data are sent to the cloud trough Wi-Fi and it is plotted in

graphs to ease the users to monitor the trends. As the data are stored in the cloud, they can be easily accessed trough the web. After the study is complete, the conclusion was that this device performs adequately in indoor as in outdoor environments, however such kind of device still has the data transmission as a limitation, and has not been applied in oceanic setting, as proposed in this manuscript.

In [25], is presented the design, deployment and testing of a PAM to use not from land, as the major of them, but in the whale watching experiences. The POSEIDON, is a low-cost PAM for nautical citizen science and real-time acoustic augmentation that uses machine learning in order to identify samples of whales and dolphins. Actually this is an example of how this manuscript solution could be integrated with the developed API, providing media files also, in this case audio.

Also, [14] describes the design, implementation and testing of a real time monitoring IoT system that collects oxygen and nitrogen reduction in water, the water temperature and also the illumination with sensors placed outside and inside the water. These collected data is sent periodically at specific intervals of time to a server that saves it to a database. This information is used to calculate the quantity of zooplankton and phytoplankton in the studied area and after that the system pretend to provide to the end users a visual information based on the collected data.

Another study, [8] describes the development of the Marine Icing Monitoring System that has been deployed on a Marine Atlantic ferry and also in two offshore supply vessels.This project has a visual based technology in order to monitoring the accumulation of ice on the the vessels or at the rigs where the huge amount of it could be a problem. The architecture of the project consists in a CPU that is connected with two cameras with high resolution pointed to the point where the ice control is required, depending of the requirements. This is a standalone system that do not need maintenance. All the components are weatherproofed so the decision where to do the installation is a easier due the characteristics of the material. The cameras are controlled by computer and take pictures every twelve minutes which are stored on the hard drive. This kind of system could also be adapted in order to monitor not only the icing on the vessels but also to monitor the melting of the iced zones and use our API to share and store that kind of information.

Another study, [22] presents the developed work in order to monitor the water quality. This IoT project has the goal of monitor the water quality, controlling its pH and its variations. The measured values are sent to the corresponding authorities. To implement this solution it was used an Arduino board, some sensors to get the pH values and to transmit the data is was used a GSM module. The performed tests where made at the municipal water tanks and also at the drinking tank reservoir. As future work, the authors propose sent the data to the cloud for a global monitoring of water quality, this could be achieved using the developed API at this manuscript.

In order to make it easier to assimilate and compare the information presented throughout thissection, table 3 was constructed.

Authors	Year	Technology	Collected Data
Kazdaridis et al.	2017	Wi-Fi, LTE, MK20DX128, LoRa	Water Quality
Park et al.	2018	LoRa, Arduino Mega 2560	Air Temperature, Humidity, Water Temperature
Trasviña-Moreno et al.	2017	LoRa, PIC24FJ128GC006, ARM® Cortex-A8	Video capture
Radeta et al.	2019	Pycom LoPy4, LoRa, PySense	Temperature Humidity, GPS Position, Footage
Prasad et al.	2015	GSM, Wasp mote microcontroller	Temperature, pH, Oxidation- Reduction Potential, Conductivity,
Djajadi et al.	2016	ATMega 2560, Wi-Fi	Carbon Monoxide, Carbon Dioxide, Luminosity, Air Temperature, Air Humidity
Radeta et al.	2018	Machine Learning, Raspberry Pi 3, Hydrophone, GoPro	Vocal Call Acoustic Samples
Kim, Nam Ho	2016	GSM, ARM-Cortex-M4	Oxygen Reduction, Nitrogen Reduction, Water Temperature, Illumination
Gagnon et al.	2009	Camera, Satellite Phone	Photo Capture
Moparthi et al.	2018	GSM, Arduino	pH

Table 3. Marine Environment Monitoring Comparison

After analyzing all the research papers presented in this section, all the collected data by the devices presented was summarized and analyzed in order to prepare the developed system

After analyzing all the research papers presented in this section, all the information collected by the different aquatic devices was analyzed and summarized in order to verify if the developed system would be able to process it and show it to users.

2.1.4 Summary of the Related Work

After the research and analysis necessary to write the previous sections, it is indisputable that the areas of Bio-Telemetry, Marine Environment Monitoring and the use of LoRa for data communication have increased over the years. At the moment, the main technological challenges on these areas are the following:

- The miniaturization of the equipment used, mainly due to the batteries that are needed and the flotation bodies.
- Equipment's resilience, mainly due to temperature changes that can be either very low or very high; the depth to which they are subjected and consequently to the pressure variation; attacks by predatory animals; and friction in the environment or even with the animal itself.
- The bandwidth, which is sometimes not enough to transmit all the desired data, which brings limitations to the projects.
- The transmission frequency, which is sometimes too low for what is intended. Sometimes the objects of study move at great speeds and their most interesting movements to be studied may not have been transmitted.
- The variety and quality of the sensors, which sometimes due to their size are not possible to add to the projects, or else they provide values that do not correspond to the truth, thus leading to false results.

All these aspects mentioned above will be used to some extent in the development of this project, so that we can avoid some errors reported in the various articles studied and somehow use the advice given in each one of them. In this way we hope to build an improved system compared to those that already exist.

2.2 State of the Art

Hereinafter, it will be presented the actual state of three different kind of technologies and techniques related to this dissertation. These parts include: (i) overview of commercial products capable for collecting the data from the oceans, (ii) the IoT micro controllers, which have the potential to be applied in this project, (iii) the bio-telemetry systems, describing the ways how usually the collected data is transmitted, (iv) the web application technologies, that will be used to build the API, to receive the data, including the back office that will be used to depict and manipulate the received data.

Devices on Market. Moreover, existing devices on the market that are capable to collect marine data are described. State of the Art devices will be presented and detailed, including the comparison between them in terms of type (underwater, above water, etc), collected data types (temperature, humidity, pressure, etc), as well as the price. Another comparative analysis will be in depicting pros and cons of each one. The objective is to study these devices and understand the best practices to receive, depict and present the data, collected from existing IoT devices. Study will yield the solution to portray the collected data from the pre-selected micro controller, tailored to perfom aquatic biodiversity assessments.

IoT Alternatives. After listing and describing the existing devices, the next area to explore is the IoT micro controllers that can be used in aquatic projects. The most used IoT microcontrollers will be explored and compared in order to find which one will fit the necessities of this project. They will be also compared in terms of price, size, power consumption, wireless data transfer protocols, memory and other features that are relevant for this dissertation. After depicting all the necessary information about them, research will further select the best micro controller, based on power consumption (as a low consumption level leads to a longer battery life time), the size (due the limitation in terms of space) and the existing wireless data transfer protocols (it is required a long range data transfer protocol, due the environment were the device will be used).

Affordable Biotelemetry. Another important topic is to evaluate the possibilities in terms of the existing wireless data transfer protocols. This part is important due the environment were the data is collected, the ocean. The goal is to find the suitable protocol to transfer the data from the devices to the API in terms of range (it must have a large range) and cost (must be cheap). After analyse the existing possibilities these two factors are the most important to choose which protocol will be used.

Modular SPAs. In terms of web application, there are numerous frameworks/tools that can be used for this project. As in the previous topics, comparison between some of the most popular ones will be provided. Here is important to define the most adequate approach in order to achieve a good connection between the 2 different parts of the web application: the API and the back office. The goal is to allocate frameworks that work well together and also allow the creation of a dynamic API, that will be used to collect data from all the devices. Moreover, they should provide the creation of a SPA that will be used to manage all the devices that communicate trough the API and display the received data.

2.2.1 Internet of Underwater Things

Nowadays, with the pertained environment issues such as global warming, endangered species, pollution, and climate changes, there is a great concern in the collection of environmental data. In this chapter, the IOT marine devices that exists on the market will be analyzed in order to understand what kind of data they are capable in collecting in order to study how such data could be collected trough the solution that is being built throughout this dissertation. The study of this devices will be important to develop a robust API structure, that is flexible and well structured, so that it can receive as many types of data as possible, and also to manage how this data can be presented on the back-office. The devices that will be studied and analyzed in this section are: (i) Deeper Smart Sonar CHIRP+[5], (ii) Poseidon[6], (iii) SeaMote[7], (iv) Triton[8], (v) WaveFinder[9] and

[5]Retrieved from `https://deepersonar.com/uk/en_gb/products/smart-sonar-chirp-plus`, on 24-06-2020

[6]Retrieved from `http://wave.tigerwhale.com/kits/iot`, on 24-06-2020

[7]Retrieved from `http://wave.tigerwhale.com/kits/iot`, on 24-06-2020

[8]Retrieved from `http://wave.tigerwhale.com/kits/iot`, on 24-06-2020

[9]Retrieved from `https://www.miros-group.com/products/wavefinder/`, on 24-06-2020

(vi) Wave & Current Radar[10].

The Deeper Smart Sonar CHIRP+ is a GPS enabled, Wi-Fi fish finder that uses CHIRP technology. This device allows the users to quickly locate target species holding spots, pinpointing predator fish and fishing in extreme depths. As this device is using GPS, if the collected data was sent to the platform that will be developed in this thesis, the user could be able to consult which places that contain more fish, aiding the fishermen in allocating new fishing spots.

Poseidon is an IOT device that is used for real-time augmentation of cetaceans watching experiences. It collects acoustic samples (clicks, moans and whistles) and trough machine learning it predicts what kind of taxa is emitting the collected sound. If this device sends the data to the proposed dissertation SPA, it will be possible to know where and which kind of cetaceans exist in one area. Such may improve the growth of the cetaceans community (for example changing of the boat routes not to interfere with the migration routes of cetaceans).

Similarly, SeaMote is a long-range radio controlled USV that is based on LoRa and IOT. It was built with the objective of collect marine environmental telemetry such as temperature, humidity, GPS position and underwater footage. The obtained information, as it collects the GPS position and water temperature, could be used for monitoring the water temperature in a determined region.

Another device is the WaveFinder, a microwave radar that measures the distance between the antenna (that is located in a fixed place, such as a bridge or a jetty) and the water surface with a millimetre accuracy (+/- 1mm) in all weather conditions. These kind of information, if shared on the proposed dissertation platform could be used to monitor the ocean level and to prevent floods.

Also, Wave & Current Radar is a high performance device for the measurement of the surface currents and directional wave spectra. With a monitoring range of 180º this radar provides accurate measurements in harsh weather situations of any kind. Usully this kind of devices are used on Petrol platforms but could also be used on the coast or on boats. The data provided by this devices, if available to the general public on the platform that is being described in this manuscript, could be used to for example to monitoring the surface currents.

2.2.2 IoT Microcontrollers as an Opportunity for Sensing Oceans

Similarly to aforementioned existing devices, recent increase in development boards suggest popular usage of IoT microcontrollers and presence on the market. Nowadays, the challenge seems to be in analyzing which of the options is best suited to the intended development. Hereinafter, 4 different manufacturers of microcontrollers/microcomputers are analyzed: (i) Arduino, due its reputation between the developers and also the hobbyists; (ii) Photon, due to the experience that our research group has with the system; (iii) LoPy4, due to its new interest and launch on the market; and (iv) Raspberry Pi, for the same reasons of (i). In some of these manufacturers, more

[10]Retrieved from `https://www.miros-group.com/products/wave-and-current-radar/`, on 24-06-2020

than one model will be analyzed in order to find the best option for the intended implementation.

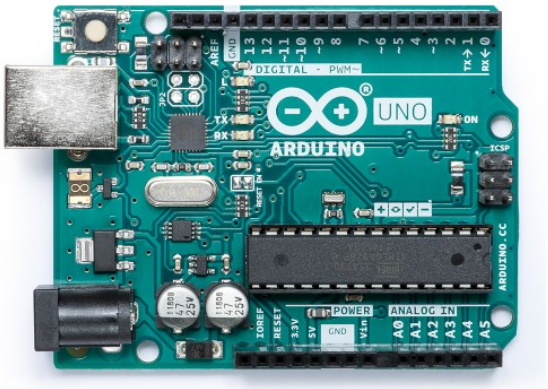


Fig. 1. Arduino UNO

Regarding Arduino, it is an open source tool based on easy-to-use combination between hardware and software. Arduino boards can read numerous kind of sensors, such as detecting click on a button, a light or a temperature selection, a message from some app, and many others. Such sensors can be easily turned into an actuator for activating a led, turning on a motor, playing some sound, etc. The behavior of the board is set programmatically, defining a set of instructions on the microcontroller that exists in the board. To set this instructions, Arduino programming language (based on Wiring) is used, and the Arduino Software (IDE), that is based on Processing. Since 2005, Arduino has been developing easy to use, open-source and low cost physical computing platforms that are based on a simple MCU board, including the environment to write the software on the board. The used language to program is C or C++. The model to be analyzed is Uno (see fig. 1), which is the most frequently used and most documented model of all Arduino micro controllers. Uno is a micro controller board based on the microchip ATmega328P that has 3 pools of memory: (i) 32Kb of flash memory, of which 0.5Kb is used by boot loader and the rest is free to use; (ii) 2Kb of SRAM that is where the variables are created and manipulated when the programs runs; and 1Kb EEPROM that can be used to store long-term information. It has a recommended operating voltage of 5V but it works with an input voltage between the range of 3V to 12V. This device has 14 digital I/O pins which 6 of them provide PWM output and has also 6 analog input pins. In terms of consumption, it consumes around 50 mA when in active state, and around 35 mA when is in sleep mode without external outputs. All these features in a device fits into 68.6mm of length, 53.4mm of width and weights 25g. Arduino Uno has a USB connection, a power jack, an ICSP header and a reset button. It does not have any kind of wireless connection using just the microcontroller, but there are some separated boards that allow the use of some wireless protocols. This is adequate due the variety of opportunities that are possible to be leveraged, however at the same time it requires more space, more connections and more cost to buy the boards separately.

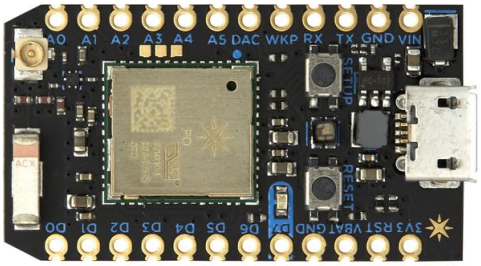


Fig. 2. Particle Photon

Photon (see fig. 2) is a tiny IoT device, developed by Particle Industries Inc. It only has 36.58x20.32x6.86mm in dimensions and weights only 5g. With this device, it is possible to write the firmware in a local or web IDE and also deploy it with ease. It is also possible to build mobile or web applications using the ParticleJS and their mobile SDKs. The main usage of this device is to build connected projects or products for IoT. Their community is growing, which provides more support, more developed libraries and more interaction between its' users. Another interesting feature is that they are connected to their cloud, the Device Cloud, and it is free to be used for the first 100 devices. There, it is possible to perform the device management, execute firmware updates, allow integrations, provides some developer tools, and also guarantee a fully managed connectivity [11]. The Photon uses a powerful STM32 ARM Cortex M3 micro controller with a 120Mhz clock. In terms of memory, it has 1Mb of flash memory and 128Kb of RAM . Regarding the wireless communication, it only comes with a Cypress BCM43362 Wi-Fi chip that allows communication trough Wi-Fi. Still, there is possibility to obtain other wireless modules to be used with the Photon, however it requires more space, more costs and more effort to connect them all. In further, Photon is equipped with 18 digital I/O, 8 analogs inputs (ADC), 2 analog outputs (DAC), 2 SPI and one USB connector. The power for this device is provided trough USB connector or directly via source power pin and it should be between 3.6V and 5.5V. With a 5V as power input and with the Wi-Fi turned on, its consumption is, in average, 80mA and in a deep sleep it is only around 80μA.

[11] `https://store.particle.io/collections/mesh`

Fig. 3. Pycom Lopy 4

The LoPy4 (see fig. 3) is a powerful and compact board developed by Pycom. It has a Espressif ESP32 chipset, which offers a very good combination of power and flexibility. This device can act as a LoRa nano gateway and a multi-bearer (LoRa, Sigfox, Wi-Fi and Bluetooth) development platform suitable for all Sigfox and LoRa networks in the world. The way to program it is through Pymakr plugins, using MicroPython, which allows a rapid and easy programming IoT application development and also an extra resilience with network failover. The most important feature of LoPy4 is the 4 wireless connection possibilities that it is ready to use: LoRa, Sigfox, WiFi and Bluetooth and also its power consumption. It works with an input voltage between 3.3V and 5.5V and in terms of power usage it depends of what wireless protocols is being used (see table 4).

Wireless Type\State		**Active**		**Standby**
Wi-Fi		12mA		5μA
Lora		15mA		1μA
Sigfox	Europe	Rx mode	Tx mode	0.5μA
		12mA	42mA	
	Australia, New Zealand	Rx mode	Tx mode	0.5μA
	South America	12mA	120mA	

Table 4. Lopy4 Power Consumption

The purchase of a LoPy4 includes 1 year of free Sigfox connectivity, however for the development purposes solely. For a deployment/commercial usage, there is a payable service that can be contracted. In terms of memory it has 4Mb of RAM and also has 8Mb of flash memory. Regarding the number of inputs and output it can have up to 24 GPIO , and all of these in a device with only 55mm x 20mm x 3.5mm and with a weight of 7g.

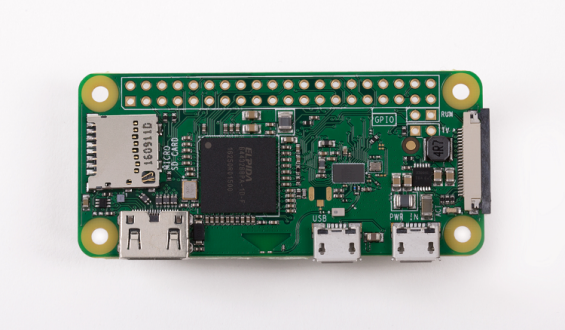


Fig. 4. Raspberry Pi Zero W

The last analyzed device is the Raspberry Pi Zero W (see fig. 4), this IoT micro controller has all the functionality of the original Raspberry Pi Zero and is also equipped with a 2.4GHz 802.11n wireless LAN , a Bluetooth Classic 4.1 and BLE connectivity. That is the reason to analyze this particular version instead of the previous version, the Raspberry Pi Zero. The used language to program Raspberry is Python (as with LoPy4), which is an easy to use (easy to read and write). This device has a 1GHz, single-core BCM2835 CPU chip, in terms of memory it has 512Mb of RAM, for the flash memory should be plugged a MicroSD card. In terms of input voltage it must have a 5V input voltage supplied via micro USB, and it consumes around 120mA on idling and around 160mA when loading LXDE . In what concerns to external non wireless communication, it has a micro USB, a USB On-The-Go port, allows 1080P HD video and stereo audio via mini HDMI , a CSI camera connector and has also 40-pin GPIO, unpopulated. This is a small device with only 65mm x 30mm x 5mm and with a weight of 7g.

After analyzing and exploring all the aforementioned devices, and in order to compare the most important aspects of them, the table 5 was made using the specs that will be more important for this project:

Device	**Price** ($)	**Size** (mm)	**Weight** (g)	**Input Voltage** (V)	**Current** (mA)		**Wireless Protocols** (included on board)
					Active	Sleep	
Arduino Uno	22	69x53x13	25	5	50	35	-
Photon	19	37x20x7	5	3.6-5.5	80	0.08	Wi-Fi
Pycom LoPy 4	38	55x20x4	7	3.3-5.5	20	0.025	LoRa, Sigfox, Wi-Fi, Bluetooth
Raspberry Pi Zero W	10	65x30x5	7	5	160	120	Wi-Fi, Bluetooth

Table 5. IoT devices comparison

After analyzing the table and all the collected information about each studied device, the conclusion is that the best device to use in this project is Pycom LoPy4, because it provides long-range and free communication using LoRa, has a low energy consumption and all the other

features (price, memory and size) are satisfactory for the project requirements.

2.2.3 Real-time Telecommunication Protocols

There are numerous wireless ways to transfer data between devices, to operate in a short range, in a long range, some of them are free, others have some cost, while others are free just for certain proposes. There are some of them that require a high energy consumption, encompassing others that require just a low energy level to perform adequately, For this dissertation,it will be only considered the ones that allow a long range communication. Such systems could be divided in two groups: the long duration, that ranges from weeks to years which is normally made using satellites or acoustic telemetry. They have low data density and spatio-temporal resolution, a limited number of sensors. Normally, it is very intrusive, which may cause injuries and the devices are rarely ever recovered; and on the other hand, there exists the short duration devices, which range from hours to days. Such are usually based on satellite telemetry, contain much more density of data, require much more sensors, may be less intrusive using suction cups, harnesses or glue, and the used devices must be recovered due the amount of data that they have and also, normally are more expensive. In this section, 5 such protocols will be compared in order to find the one that will serve the project requirements better: (i) VHF due its power in terms of range, (ii) Satellite, using Argos, regarding the wide coverage area that is covered by this technology, (iii) Cellular, due to the large number of users it has, (iv) LoRa, because it allows a long range distance communication and also since some IoT devices are ready to use it, and (v) Acoustic telemetry, regarding the use on sub-aquatic species.

VHF is the electromagnetic spectrum of the radiation with a wavelength between 1m and 10m and a frequency that varies between 30Mhz and 300MhZ.
This kind of communication is used normally for DAB , FM radio broadcasting (usually between 88Mhz and 108Mhz), television transmission (using also UHF), land navigation systems, air traffic control communications, two-way land mobile radio systems (military, emergency, business or private use) and also for amateur radio.
This kind of communication propagates by line-of-sight and ground-bounce paths, and can achieve 160 Km of range. This kind of waves can penetrate the building walls and may be received inside of it. However, in urban areas the propagation is more challenging, due to the reflections caused by the buildings. Nevertheless, they still can reach great distances in aquatic settings.
The bigger disadvantage of this communication system is the need of permissions to use it, where it is not as easy to obtain the necessary authorizations.

Another option is to use Argos, which is a satellite based system created in 1978 by CNES , by NOAA and NASA . The original purpose of Argos was to collect and relay oceanographic and meteorological data around the world. In order to commercialize, maintain and operate the system, in 1986, CNES created a subsidiary, the CLS and nowadays, CLS is the exclusive Argos services provider which is used by diverse sectors, such as: sustainable fisheries, mining, environmental monitoring, maritime surveillance and fleet management. This system can be divided in 4 parts:

- Platforms - It could be any equipment that has an Argos-certified transmitter. This devices uploads periodic messages to Argos satellites, these messages are characterized by the transmission frequency, which is 401.650MHz +- 30kHz and must be stable in order to calculate the location using the basis of Doppler effect; and also by the repetition period, which is the difference of time between the two messages. Each platform has it's own identification number.
- Satellites - These emitters are at an altitude of 850 km, and are responsible for receive the messages from the platforms, store that messages and also relay them back to the Earth.
- Receiving Stations - It exists nearly 70 stations that receive data relayed from the satellites and transmit them to the processing centers. Regarding this kind of method, Argos provides a worldwide coverage.
- Processing Centers - These centers are responsible for calculating the locations, verifying the message quality, the time-tagging, data processing, and other tasks in order to ensure the correct working of the system to provide a good service to the costumers. It exists two global Argos processing centers, one is located it Toulouse, France and the other is located near Washington, DC, USA.

One disadvantage of this service is that it has the high cost. Although Argos is a non-for-profit system, they have to guarantee that the revenues cover the costs of the entire operations, and that means that it additional has costs for the users.

Celular communication is a bidirectional wireless form of communication that is distributed over land areas, that are typically called "cells". Each cell is served by, at least, one fixed transceiver. A cell normally uses a different set of frequencies that differs from the neighbour's one. Such different sets of frequencies are further used to avoid interference and provide a better quality to each cell. These cells, when joined together, provide a radio coverage on a large geographic area. Any device can communicate with other devices or with transceivers anywhere in the network, even that the device is moving along more than one cell, the communication will still be made.
The most common usage of this kind of communication are the smartphones communication using the mobile operators transceiver stations. These kind of network could be used by any device that is equipped with a SIM card.
Since that the same frequency could be used for multiple transmissions, as long that the devices are in different cells, this kind of network offers more capacity than a single transmitter, and also, as the communication is made with the towers and not with a satellite nor with a transmitter, the required power is lower due the distance. Another advantage of this kind of network is the fact that the covered area could be enlarged with the constructions of more cell towers.
With the past of the years this kind of technology have been improved, passing from 1G up to the new 5G, with an incredible improvement in terms of speed, data transfer capacity and costs. This kind of communication has an associated cost,depending of the country, operator and chosen plan. Another disadvantage of this kind of communication is the coverage on remote areas, and the ocean is one of those remote places.

LoRa, that is the short representation for "Long Range", is a radio frequency technology that allows long range communication, using a very reduced power consumption level.
The modules send and receive data from specific gateways (similar to Wi-Fi, however, with much

more range power) that redirects that data trough IP to local or remote servers.
LoRaWAN is the used protocol by LoRa that defines the architecture of the system, as well as the communication parameters that will be needed. This protocol implements the working details, security, service quality, power adjustment, in order to maximize the battery duration of the modules, and the types of applications on both, server and module side.
The main usage of such kind of communication is for the IoT devices projects, that are typically used in a challenging access places and due that, they require a long range communication and a low level consumption. This way, the number of times that the battery of devices needs to be replaced or recharge is reduced. The LoRa devices and the open LoRaWAN protocol enable smart IoT applications communications that have benefits in some major problems of the world, such as natural resource reduction, energy management, infrastructure efficiency, pollution control, disaster prevention and some more application areas. The number of use cases is high and the most commons are smart houses, smart cities, smart agriculture, smart supply chain and logistics, and smart metering. The number of connected devices is over 100 million in 100 countries and growing.

The acoustic telemetry, as the name suggest, is a method for transmitting information across open space using sound. This kind of telemetry is used to collect information of sub aquatic species as fishes or turtles attaching pingers (acoustic transmitter tags) to them. This kind of technology allows the researchers to conduct survival studies, monitor the migration routes and analyze the behaviour of the species. Such tags can have many sizes and shapes, different battery duration times that range up to 10 years. Moreover, they can collect more data than only the location of the studied individual species, but also depth and water temperature and could be surgically implanted inside the animal or externally attached if possible. Each tag emits unique sound pulses that are received and interpreted by the hydrophone of underwater tracking stations, also know as receivers. These tracking stations convert the sound in data and use the sent and arrival time to calculate the position of the tag, providing the information to the researcher in real time. The frequency of the emitted sound is different depending of the environment, and for oceanic usage it usually lower than 100 KHz. For rivers or lakes instead, it is higher depending on the situation. This kind of telemetry has some limitations of use, such as the sub aquatic usage only and also the costs of the service.

After exploring the possibilities and challenges explained in these section and in order to be easier to compare the technologies, the following comparison had been made:

Type	Frequency range	Fees	Power consumption	Coverage
VHF	30MHz - 300MHz	-	Moderated	Good
Argos	401.650MHz +- 30KHz	Has	Low	Worldwide
Cellular	1850MHz - 1990 MHz 824MHz - 894MHz	Has	Moderated	Limited
LoRa	433 MHz 915MHz 923MHz	-	Low	Good
Acoustic	Bellow 100 KHz - Saltwater Above 100 KHz - Freshwater	Has	Depends	Limited

Table 6. Real-time Bio-Telemetry comparison

After analyzing the protocols, LoRa was selected due to its adequate relation between all the most important points for this project, the long range, the low cost, the low power consumption and other aforementioned reasons. The other methods have some issues that make with them are not a good solution for this project such as the permissions that VHF requires, the costs and hardware that Argos requires, the coverage issues of cellular antenna and their costs, and also the limitations of usage and cost of acoustic telemetry.

2.2.4 Overview of Web Applications

Nowadays, the number of tools to help the web applications development is huge, the problem is really to choose the one that will offer more advantages to the project that the user wants to implement. In this particular case will be necessary to develop an API to receive the data collected by the IoT devices, a back-office to manage (create, edit and delete) them and also a SPA to present the received reports to the users. The frameworks give to the developers a set of technical tools that make their development easier and faster. With the use of a frameworks the programmers are available to structure the application in standard patterns; built-in some classes such as database, user input validation, session, and others; test the quality of the developed code; easy third party integration; fast deployment mechanisms and some others interesting features that make the life of the programmer much more easier. In these section there will be described, analyzed and compared three PHP frameworks that are built on MVC architecture: (i) Codeigniter, (ii) Laravel and (iii) CakePHP that will be used to build the API and back-office. In order to develop the SPA there will be also analyzed two tools: (i) ReactJS and (ii) Angular.

2.2.4.1 API and Back end

Codeingiter first release was in 2006, it is the second released framework from the ones that will be analyzed in this manuscript, and is still used. In terms of releases it is not so much active as Laravel, the version 2 was released in 2011 and then only 4 years latter comes the version 3 that

is the used one until now, the version 4 is in development and they don't give a specific release date. Codeigniter announces that this new version will change dramatically the direction that this framework has been developed, in order to match the other frameworks on the market. This framework is mostly popular in some countries of Asia, such as India, Bangladesh and Pakistan. It is structured following the MVC model and is based on object-oriented programming but the programmers could use it according to their needs. It does not provide direct support for composer, but it is possible to achieve that integration and after that use all this tool functionalities. This framework has a good official documentation, has a forum for the users and it is easy to find questions about it on Stack Overflow, wich is good for developers. Codeigniter provides some built-in functionalities but it does not provide the same level of third party libraries as the others frameworks that will be analyzed in this section. In terms of database it is ready to use relational databases such as PostgreSQL or MySQL but if the goal is to use non relational databases such as Mongo it is hard to achieve and maintain. It has it's own ORM which is enough for the basics but is not so complete as the one that Laravel provides. In terms of API development it does not support the direct implementation, but exists some third party libraries which can be integrated with Codeigniter in order to achieve the API development. A very good aspect of this framework is that it is easy to start using and has a good learning curve.

Laravel first version was released in 2011, after the others frameworks that will also be analyzed in this study, but that did not stopped this framework to become the most popular PHP framework at the moment. It is mainly used by web developers of US, Canada and Europe countries. In terms of structure, it follows the MVC, it also has a command line tool called Artisan which allows the developers to easily create controllers and models, manage the database migrations, setup schedule tasks, create common mechanisms like the login, password recovery and some others. The development of this framework is active and constant. In the past three years there are 2 major updates per year. Regarding to the online help for this framework, it has it's official documentation which is very complete and well organized, as Laracasts where is possible to find tutorials and also where developers can set their own questions and wait for help and there is a lot of Laravel related questions on Stack Overflow and other developers platforms. There are a lot of libraries that are compatible with Laravel in Packagist that the only thing that is needed is composer them and use it. This framework also offers it's own ORM , the Eloquent. This tool can easily query the database with just some definitions and make simple or complex operations with a low quantity of code. It is also possible manage the database migrations and set up some seeds to populate it with real data or with dummy data just to have some rows on it. Regarding to the API support, Laravel offers a small framework, the Lumen, and also has support for REST built in with API routes. In terms of front end, Laravel has a the Blade template engine which is good for the front-end developers who don't want to learn PHP.

CakePHP was the first framework, from the ones analyzed in this section, and it was release in 2005. The version 2 was release in 2011 and the version 3 on 2015, actually the version 4 is in development, but there is available a beta version. In terms of structure it is also MVC, CakePHP was inspired on Ruby on Rails structure. In terms of documentation it has it's own documentation and is really easy to find help searching on the internet. It comes with composer and due that

is really easy to add new libraries to the developing platform. Similar to the other frameworks explored before, CakePHP comes with it's own ORM and some query builder tools that help with the database interaction, but they only provide support to relational databases, for non relational it is needed third party integration's. Regarding the implementations of APIs, the structure of CakePHP allows it but it demands too much work for the things work well and for that reason it is not worth to use CakePHP to API development. This framework allows Twig templates which is great for UI developers.

Framework	**Codeigniter**	**Laravel**	**CakePHP**
Database Model	Relational Object-Oriented	Object-Oriented	Object-Relational Document-Oriented
Programming Language	PHP	PHP	PHP
License	MIT License	MIT License	MIT License
Template Language	PHP proprietary	Blade Template Engine Template	PHP
Programming Paradigm	Component-Oriented	Object-Oriented Event Driven Functional	Object-Oriented Event Driven Functional
Supported VCS	Git Github	Git	Git
Stack Overflow Questions	65.5K	136K	30.7K
GitHub Stars	17.9K	57.2K	8.1K
GitHub Forks	7.8K	17.7K	3.4K

Table 7. PHP frameworks comparison

After analysing all the tools that will be more useful for this kind of project and comparing the PHP frameworks in terms of API support, ease of development, database integration, structure, support, updates and third party integration's, the chosen framework is Laravel due its good balance between all the analysed points. Moreover, it is also because it has been already used in other parts of the platform where the this module will be integrated.

2.2.4.2 Front end

ReactJS is an open source JavaScript library for the web development. It is maintained by the Facebook, Instagram and other companies and also by a huge community of individual developers. ReactJs is a component based library, and each component implements a render method that takes input data and return the view to display. The goal of using this tool is divide the web application into a small components that can be exported and imported into another components, which makes the code more organized and the use of same component in different parts of the project. Each component have its properties, that are passed by the parent component (the one where the component is instantiated) and also has its own states, that changes dynamically with

the interaction of the user. Using this two mechanisms is possible to define the behaviour of the component based on its states and also on its properties dynamically and without the need to refresh the entire page, just the specific component.
ReactJs allows also the developers to use other libraries and frameworks, and there is also, many reactJS ready to use libraries that had some usual components already build, such as collapsible sections, tables, carousels, accordions, and some other.
It is not mandatory, but is possible to use JSX to develop the components. JSX is an extension of the JavaScript language and is based on ES6, which is translated into a regular JavaScript at runtime, this can result into a incompatibility with some older browsers that dont support ES6, but this can be solved using a transpiler. JSX allows the developers to write HTML elements in JavaScript and place them into DOM without the need of use specific JavaScript methods for that, such as createElement or appendChild.

AngularJS is an open source MVC JavaScript framework for web development that is maintained by Google. This framework adapt and extend the traditional HTML to a better experience with dynamic content with a two-way data-binding which allows an automatic synchronization of models and views. This kind of behavior abstracts the DOM's manipulation and improve the tests to the code.
AngularJS has a huge global community support, where developers, designers and enthusiasts constantly collaborate and contribute to the community grow, increasing the reliability and credibility of the framework, this framework is full-fledged and can run in any browser, also this framework learning curve is low which is good to start a new project.

Technology	**ReactJS**	**AngularJS**
Author	Facebook community	Google
Type	Open source JS library	Fully-featured MVC framework
Toolchain	High	Low
Learning curve	Low	High
Language	JSX	JS and HTML
Rendering	Server Side	Client Side
Packaging	Strong	Weak
Architecture	None	MVC
Data Binding	Uni-directional	Bi-directional
DOM	Virtual	Regular

Table 8. SPA tools comparison

After this study and comparison in terms of the best SPA tool, the one that will be used is ReactJS due to its functionalities that will bring to the project a modern and rapid front-end development, due its fast learning curve and, once again, since this tool is already being used in other parts of the website where this component will be applied.

3 System

This part of the dissertation contains a description of how the system parts are implemented, which is used and how it works.

In order to be easier to understand and more organized the description of the different parts of the system are separated in the below sections.

3.1 Overall Architecture

This part of the dissertation describes all the existing components in the system in a more high level way, and its objective is to show which are the different components, how they are connected and what are their interactions.

The main objective of the developed system is to allow the collection of the most varied types of data, that the existing aquatic IoT devices are able to collect. The communication between the devices and the backoffice will be done through an API that should be developed with a focus on flexibility and versatility in order to be able to process and store the various types of reported data. In order of this flexibility and versatility be possible, there will be a backoffice that will allow users to configure the types of data that each device will report, the units of those types of data, and which devices will report them. After the reports are effectively processed and saved, it will be possible for users to make their query through an interface that will have as one of the main objectives to be as simple and intuitive as possible.

The architecture of this system requires four distinct elements in order to be fully utilized: a web application, a database, a REST API and IoT devices that collect marine biodiversity data. The target is the Iot devices, however in reality, others can be integrated with the system as long as the technology used allows them to make requests to the developed API. In this project, the first three components were developed and the objective is for existing devices that do this type of data collection to be integrated with the system, in order to have a common repository where users can configure their devices and consult information, allowing them to navigate the devices' reports in real time. Below is the description of each component and the types of connection it establishes with the other components:

- **IoT Devices** - This component is the starting point to have marine life information in the system, so that it can be consulted by users. These are the devices responsible for collecting the most diverse types of marine life data, and can be of different types: tags, buoys, implants, and others that have already been mentioned in this manuscript. In order for these to be integrated with the system, they must have the ability to communicate with the REST API developed for this purpose and must also be correctly configured in the back office. If these requirements are not met, the device will not be able to communicate and obtain the data. These devices can only establish communication with the API, and there is no direct communication with the other components. As previously mentioned, these devices were not developed, and the objective of the project was the development of the other three components.

- **REST API** - This REST API was developed so that the devices could communicate the collected data to the system. As such, it only has one endpoint for that purpose. This is the link between the devices and the system, being responsible for validating the data sent by the device. If these are in accordance with what is expected and with what is configured in the backoffice, it will save the reports in the database, allowing them to be used by the system. This element does not establish a connection with any other components other than those previously described.

- **Database** - The database can be seen as the central point of the system. This is where all information related to units, data types, device types, devices and reports is saved. Such is responsible for storing the data transmitted by the API, which comes from the devices; it is responsible for providing the API with the necessary data for the validation of the device that is making contact with the API, and is responsible for providing the web application with the data it has, allowing for making changes, additions or deletions if the user requests it in the web application.

- **Web Application** - This component allows the user to interact with the system, such as consultation, creation, editing and deletion of data. The web application requires login to be used and is divided into several menus so that its use is intuitive. This component only interacts directly with the database, in order to obtain the necessary data to present to the user and also editing, creating or deleting operations on the data that are present.

Figure 5 illustrates the four components and their connections so that it is simpler and more intuitive to understand how they are interconnected.

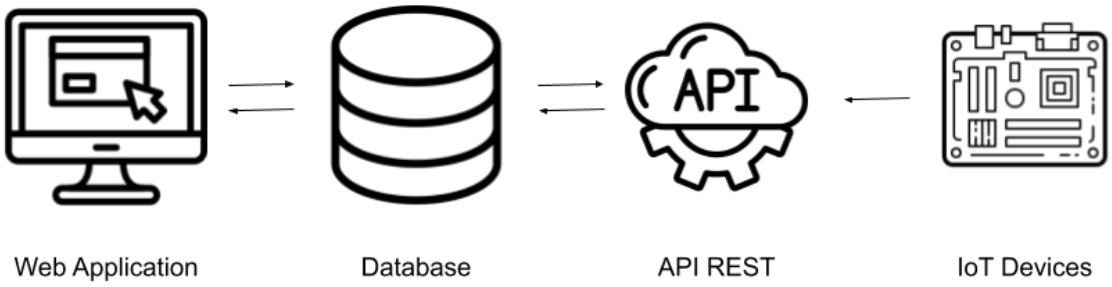


Fig. 5. General Architecture of the System

3.2 API Architecture

In this section, implementation process of the API is described that will be used by the devices to transfer the collected data to the system. It will be also presented the process of how the communications are processed by the API.

In order to make possible the reports communications from the devices to the system, an API had to be developed. This API REST was developed using Laravel, since it allows an easy and fast implementation and also it was to be used for the graphical user interface development, that is described in the next section.

As the objective of creating the API is only the communication of reports, and there is no need for third parties to obtain information, only one endpoint has been implemented for the creation of reports. The created endpoint is `http://wave.arditi.pt/api/reports`, and in order to successfully save the reported information in the system the HTTP requests must follow the below mentioned rules:

- The http verb must be a POST.
- In order to identify which device is performing the request, it is required a Header parameter with the name "id-device" that contains the id of the device in the system.
- The request body must be an array of objects, and each object must follow the structure "key":"value" as indicated in figure 6. Even if device only performs a single report, it must be an array with one object or the API will reject the request.
- If the device is capable to save the timestamp when the collection of data was performed, a parameter with the name "collected_at" can be sent in each object of the body, otherwise the system will assume the collected data as the current time. If defined in the object, this field must be a timestamp.

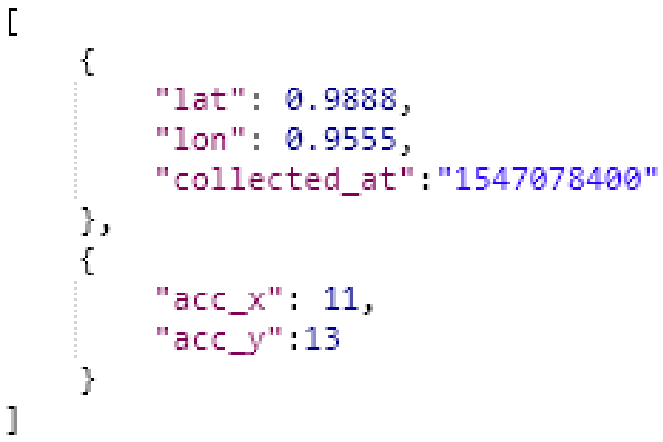


Fig. 6. API Request Body Example

To be able to communicate successfully with the API, it is not enough to follow the rules mentioned above. For everything to work seamlessly, it is also necessary to have some settings on the back office side so that the system is capable to process the information received in each request. Here are the necessary settings in the back office:

- The units that will be needed must exist.
- The data types that will be needed must exist.
- The device must be created and configured. The configuration is an important step, mainly the configuration of the data types that the device will send. The data types must be configured and the key property of each one must match the key sent in the request body. If, because of some reason, the keys do not match, the value will not be stored.

In terms of how the API processes each request, the mechanism is the following:

1. The request is received by the API.
2. The system checks if the request has the header. "collected_at", in order to identify the device that is making the request. If this header is not provided, the API returns the error: "Device identifier is required". It could happen that the header exists and it is filled, however the provided id does not exist in the database. In this case, the returned error is: "Invalid device identifier".
3. The device information is loaded from the database, and this information includes the data fields that are expected in the reports
4. All the reports are retrieved from the body and a loop starts until all the reports are processed. In each report is checked if exists the "collected_at" key. If not, it will be created with the current value.
5. A new loop is started inside the already existing loop, this time it is a loop trough the device data types, where the system tries to find the key sent at the report in the configured data types keys of the device. If the a match occurs, the value is stored into the database. If not, it is ignored and proceeds to the next element.
6. When the loop trough the retrieved reports ends, it returns this success message: "Reports inserted with success".

For a more visual understanding of how the API processes the requests, the figure 7 contains the flowchart of the process.

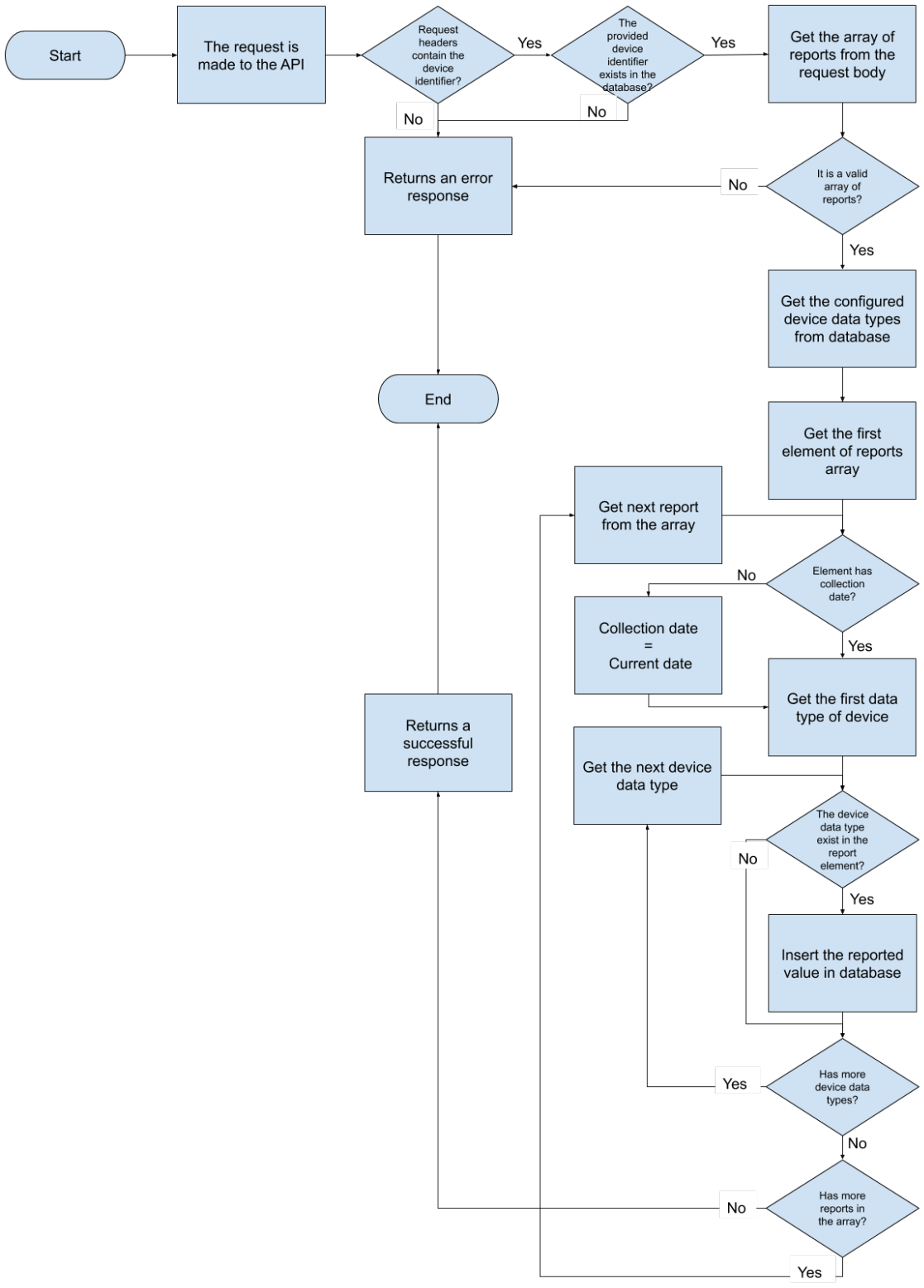


Fig. 7. Api Flowchart

3.3 Graphical User Interface

In below, GUI is provided, depicting the several screens which were used throughout the user study, containing several administration menus, which facilitate the administration of IoT devices, units, data types and device types.

The menus are divided in two groups: (i) one that contains the configurations of the units and data types that will be used by the system, and (ii) other that contains the devices specifications. In order to access the menus, it is necessary to have a valid account. The registration and login mechanisms that are required to achieve restriction of only authorized users have access to this part of the website, and it was already implemented in the web application where the dissertation development system was integrated.

3.3.1 Units Menu

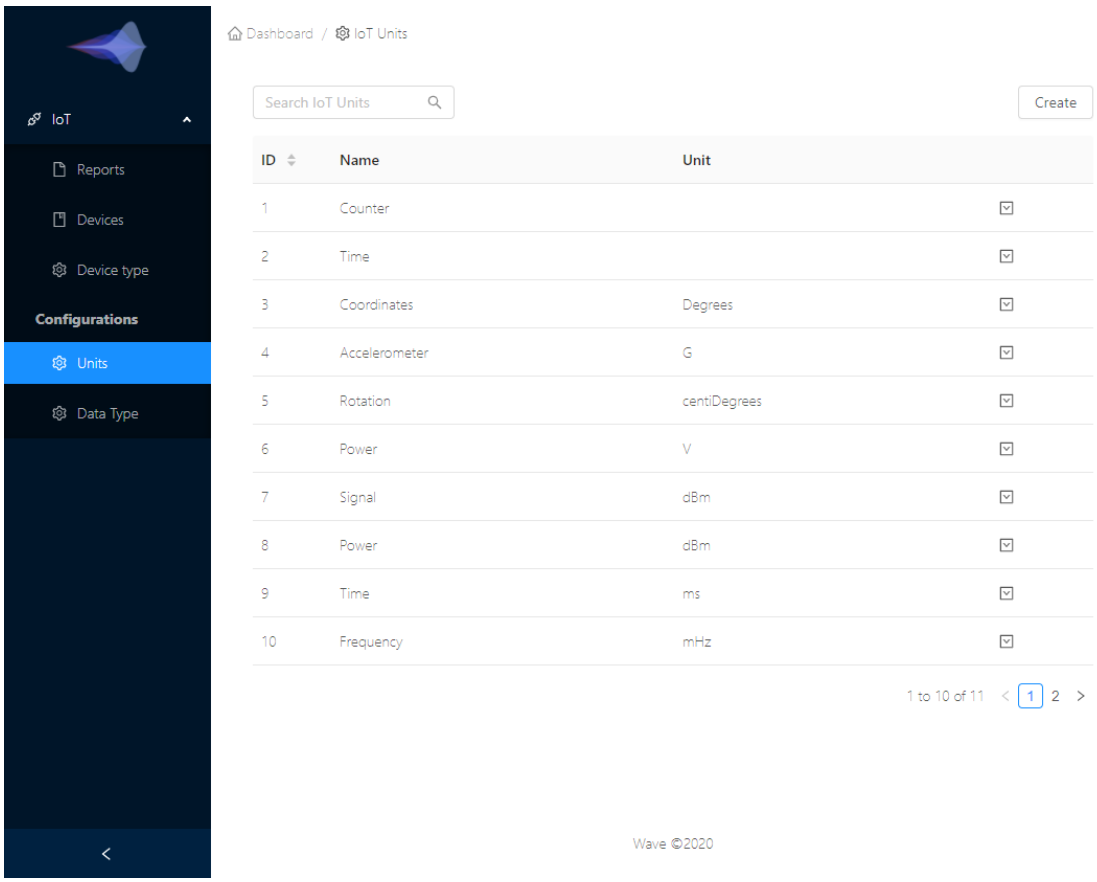


Fig. 8. Configuration Units Menu

The Units menu (fig. 8) is where it is possible to manage all the units used at the system. When chosen, a query is performed in the database, and all existing units are retrieved. The user is redirected to an area where it is possible to consult all units in the system, showing ten units at a time on the screen, and the rest in pagination mode. In this case, the following attributes are

listed: ID, name and unit.

In order to provide to the user some tools to query the unit that they are looking for more quickly, the table has the ability to be reordered in ascending or descending order by ID with just one click on its header. This could be useful to consult the last created units with just one click instead of going to the last page of results.

There is also a search feature that can receive the name or the unit. When the button with the magnifying glass icon is clicked, a query is made to the database, using Laravel, and it is checked whether the data entered by the user corresponds to a unit or to a unit name. It is not necessary for the values to be the same, just that a part of the entered value is equal to a part of a stored value, for example, searching for "er" will return as a result "Power" and "Counter."

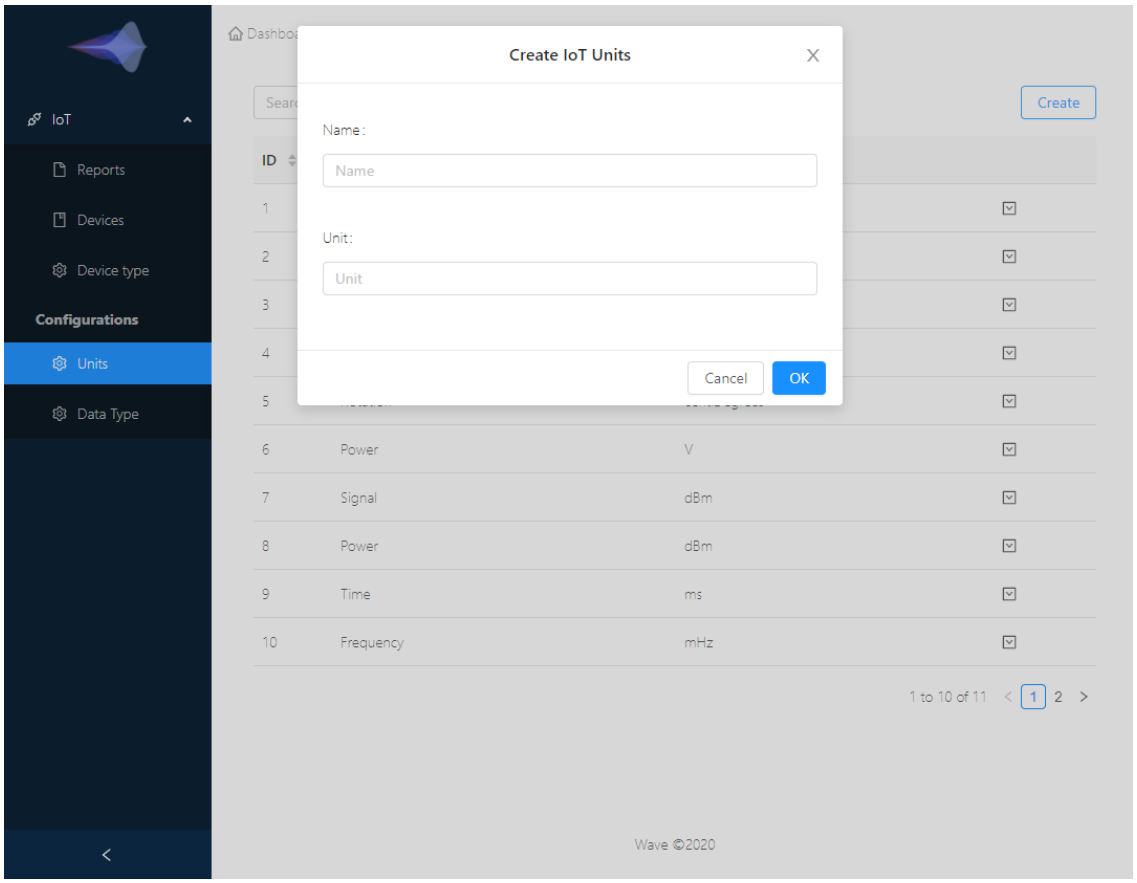


Fig. 9. Creation Form of New Unit

There is also a button for the insertion of new units in the system. When clicked, a popup is displayed (fig. 9) that contains the form with every field that are necessary for the creation of a new unit, in this case in specific a field for the name and another for the unit which are both mandatory.

If the "Cancel" or close button are clicked, the data entered are discarded and the popup is closed, if the "Ok" button is clicked, the data is entered in the database, the popup is closed and

the list will be updated, already showing the new result.

There is also the possibility to edit a specific unit, or even delete it. To perform these actions, there is a button at the end of each line that contains these two options, relative to the unit that is displayed on that same line.

The data editing is done in a popup with the same creation fields, but with the pre-filled data corresponding to the unit being edited.The edited values will only be saved in the database after confirmation of the edition. If the edition is canceled or the popup is closed, the values will remain unchanged. The deletion, and to prevent any errors, needs to be confirmed through a popup that asks for confirmation of the deletion, and only after being confirmed that the unit is to be deleted, it is removed from the database.

3.3.2 Data Type Menu

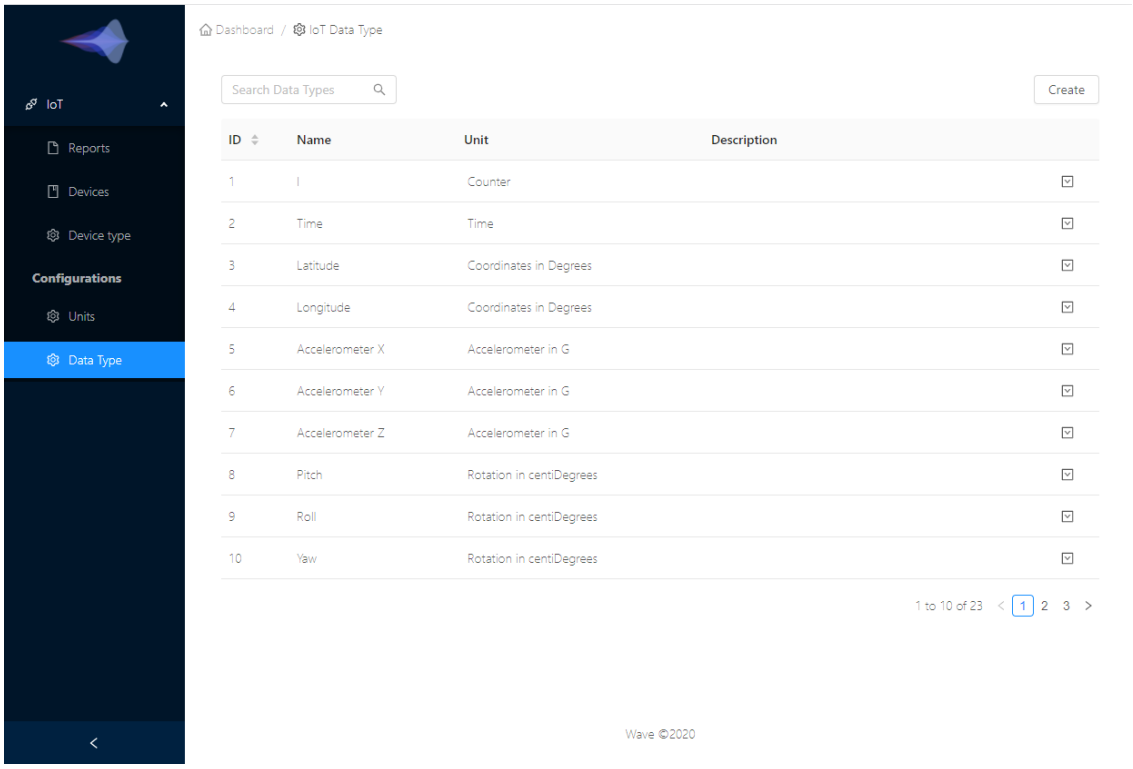


Fig. 10. Configuration Data Type Menu

The Data Type menu (fig. 10) is the menu where the management of all data types is made. As in the Units menu, when the page is loaded, a query is made to the database that obtains all the results of the existing data types and is presented a table with the ten first existing data types and the pagination where is possible consult the remaining results. The table contains the identifier, the name of the data type, the respective unit and its description.

This menu also has the possibility of re-order the existing data types in ascending or descending order by id, with a single click at the column. The search mechanism at the left-top corner can perform a search using the name of the data type, and once again, it only need to match a part of the name of the data type, not the entire name.

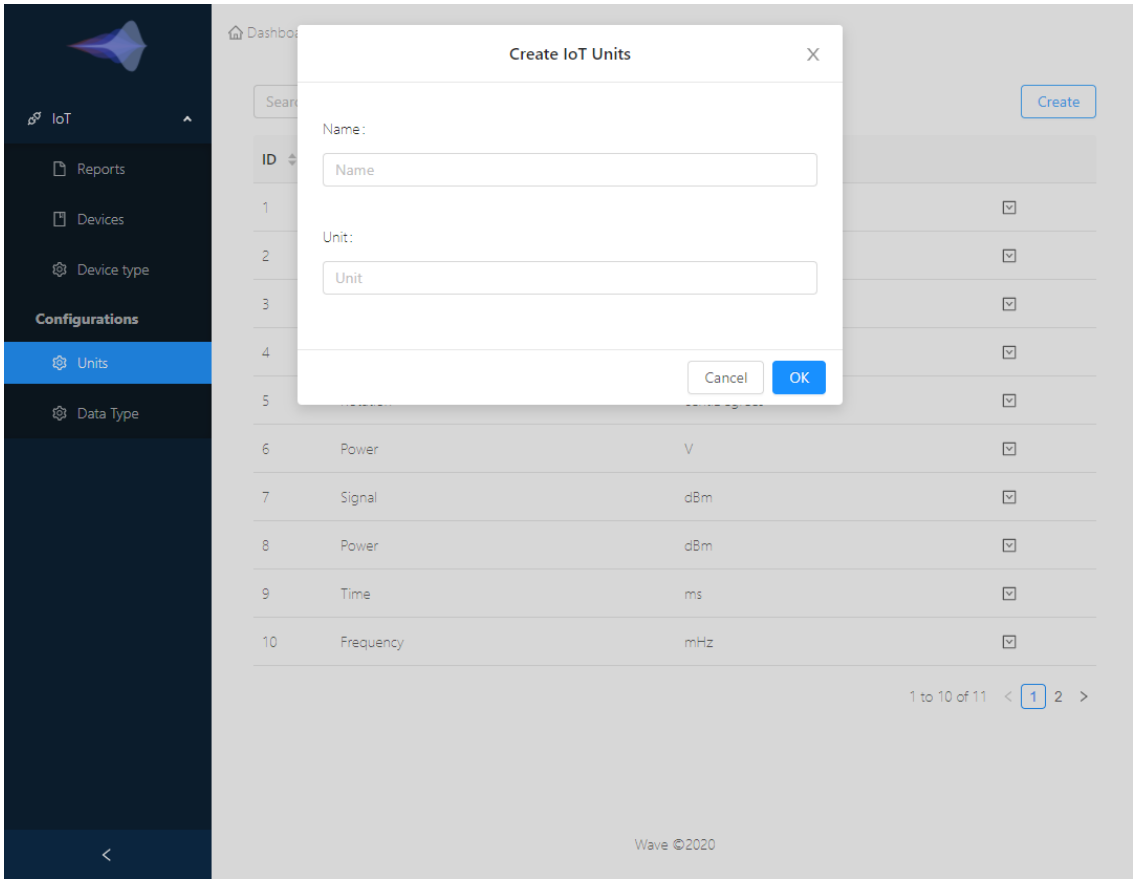


Fig. 11. Creation Form of New Data Type

When the create button is pressed, the behaviour of the system is similar to the units creation: the popup appears (fig. 11) with the form of the creation of a data type that are all mandatory: the name, description and the select option to choose the unit of the data type, that are already in the database.

With this approach, is possible use the same unit to define multiple data types, for example, with the unit Temperature in °C is possible to create the data types Water Temperature, Device Temperature, Creature Temperature, and as many data types that uses temperature as unit as the user wants. If the popup is closed or the creation is canceled, there is no data inserted in the database and the popup is closed. To save the new data type it is necessary click the "Ok" button, after that click, if every field is correctly filled, the new data type will be inserted

in the data base, the popup will be automatically closed and the list of data types will be refreshed.

3.3.3 Device Type Menu

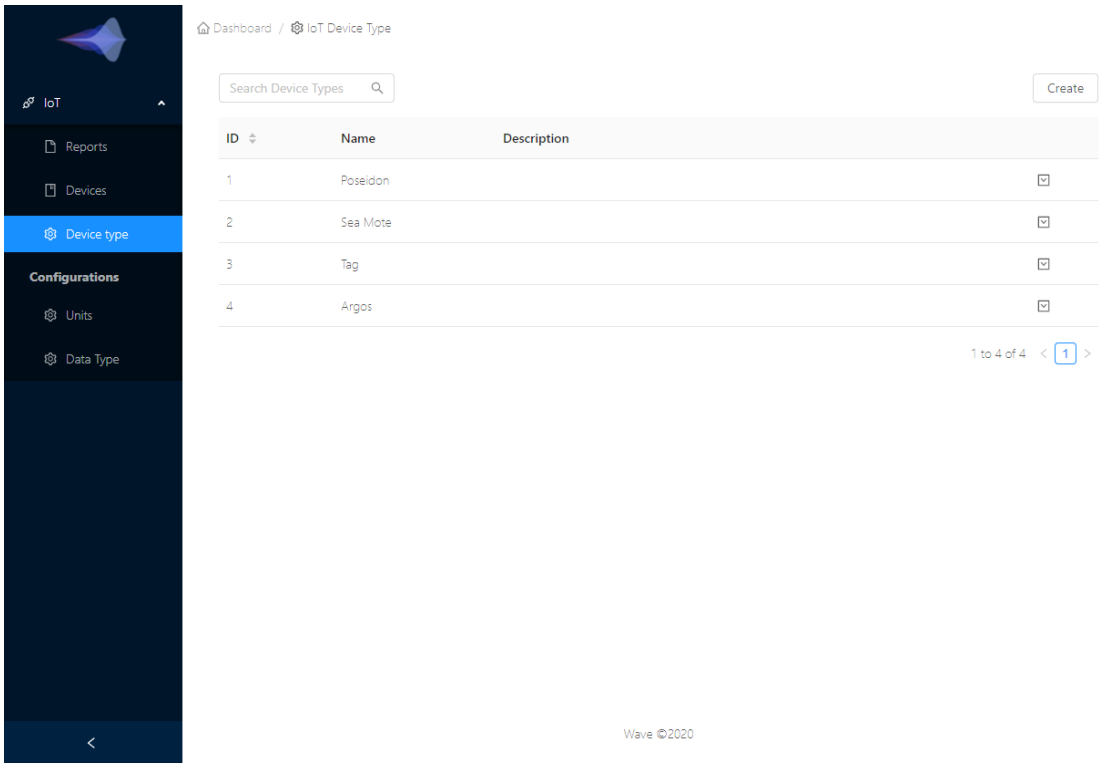


Fig. 12. Device Type Menu

The Device Type menu (fig. 12) has the same structure as the described before with a table that show the first ten device types and the remaining can be consulted clicking in the pagination. The table contains the details of the device types, such as the identifier, the name and the description. There is also a search feature that is capable of search by the name of the device type and does not need to match the entire name.

The reorder by id feature with one click in the column header is also implemented in this menu to provide an easy way of consult the last created device types.

There is also a creation button in the top right corner of the screen, that shows a popup with the form that contains the necessary fields to create a new device type. In this menu it is also possible to edit or delete a specific type of device using the buttons at the end of each line. When clicking on edit, a popup appears with the same form used to create a new device type, but this time with the data pre-filled with the existing values in the database. The edited data will only

be saved in the database if the confirmation button is pressed, otherwise the changed data will be discarded.The deletion mechanism, as in the previous menus, has implemented a confirmation feature to prevent the device type from being deleted by mistake.

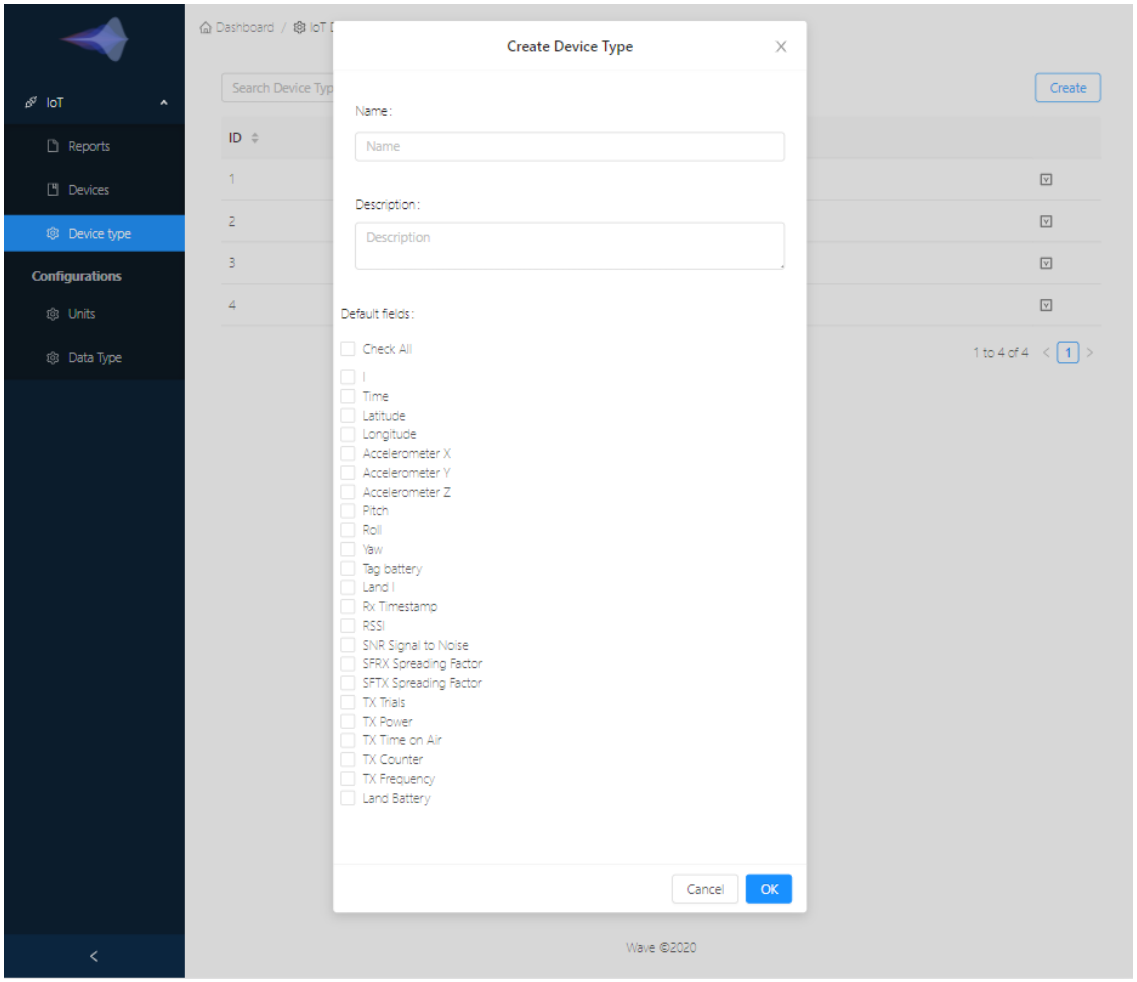


Fig. 13. Creation Form of New Device Type

In terms of structure of the creation and editing form, this menu has a more complex form than that of the previous menus. This form (fig. 13) is composed by a name and description field and also by a default fields group. This group of default fields will be used to pre-fill the Data Types used by each device of each type, in order to make the process of a device creation easier and faster.

The mechanism implemented in this form, more precisely the group of checkboxes, was developed with the purpose of saving the data types that a given device type will use. Therefore, when selecting a specific data type in this form, it should appear in the form used to create a device, if the device type is the same as the one where that field type was selected. The form also have a

"Check All" option to easily check or uncheck all the check-boxes with only one click.

3.3.4 Devices Menu

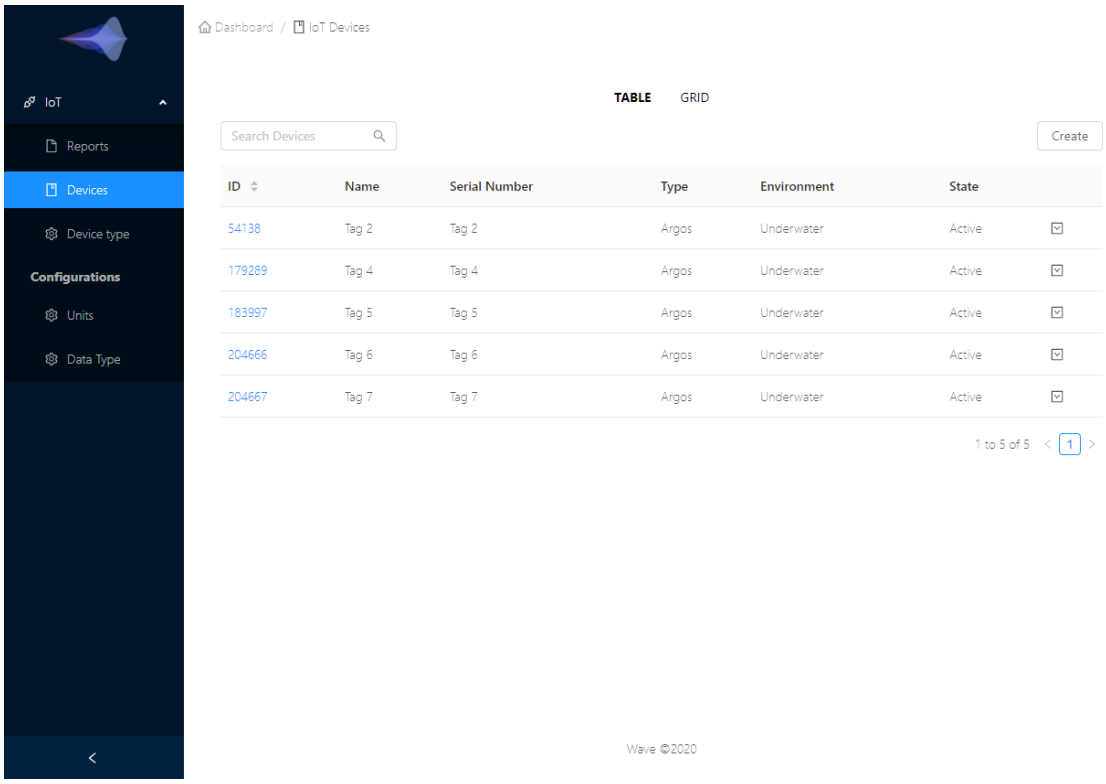


Fig. 14. Devices Menu - Table View

The Device menu (fig. 14) is similar to the menus described previously but with some more features. The main difference of this menu is the way that data is presented. On the previous menus the data was presented only in a table format, and in the current menu, in addition to the listing in the table there is also the possibility to consult the various devices in a grid view.To alternate between the two views it is only necessary to click on the desired option which is located in the center of the upper part of the screen.

The table listing presents the first twelve devices and some of their properties, such as id, name, serial number, device type, environment type and status. To consult the remaining devices is just necessary to click at the desired page in the pagination section.To help the user to consult the last created devices, the reorder mechanism is also implemented in this menu. The search bar is also implemented at the top left, that allows the user to search by the device name or the serial number of device, as in the previous menus, if some part of the saved name or serial number matches the

search input, is enough to show the results.

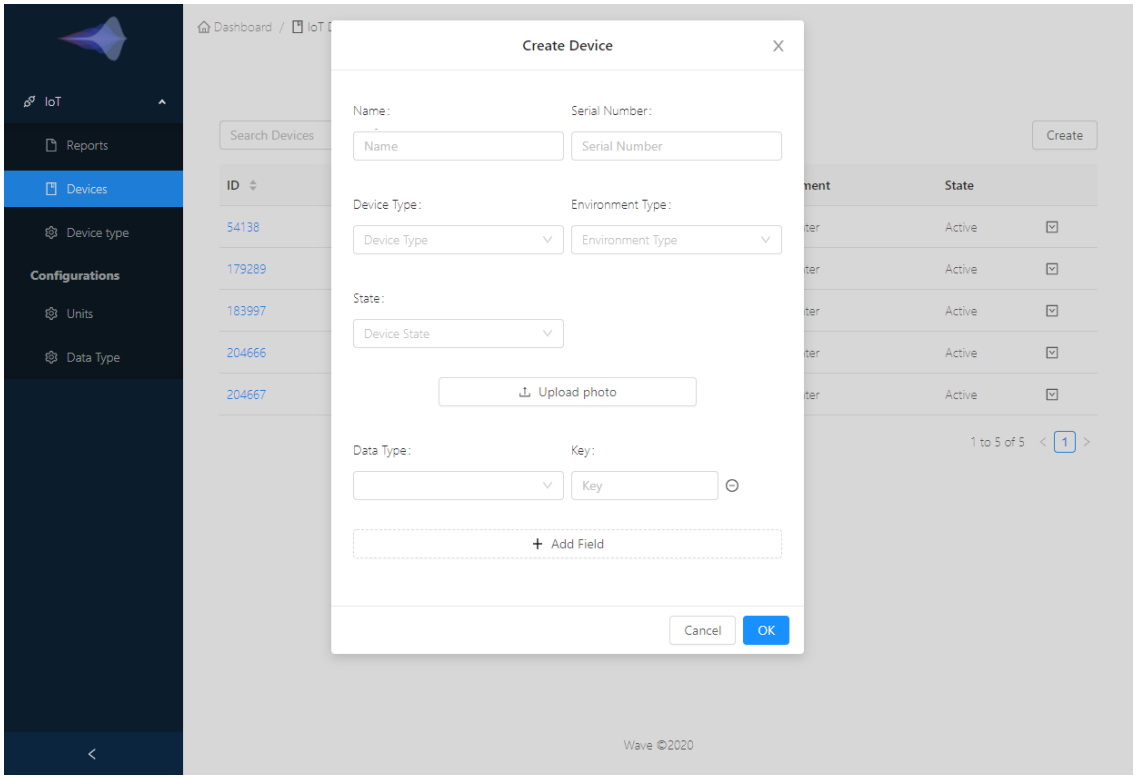


Fig. 15. Creation Form of New Device

At the top left, as usual, exists the create button, that allows user to create a new device. When this button is clicked the popup with the creation device form is shown, as figure 15 show. The existing fields are name, serial number, device type, type of environment, status, photo of the device, and the fields that will be sent in the reports performed by this device. This last field is actually made up of two fields: the data type and the key. The key field is the name of the field sent by the device, while the data type field is the corresponding field with a more readable name to help the end user to understand with type of data it is. For example, data type could be "Water Temperature" and the corresponding key could be "waterTemp" or "water_temp" or another value that is used at the device implementation. These association between the data type and the key is basically used to do the automatic mapping between the field names used in the device implementation and the data types configured at the system back office when a report is communicated to the API. When the type of device is changed, a query is automatically made to the database that obtains the pre-defined data types for the type of device chosen. These data types are inserted in the data types of the device being created or edited and the user can remove or add others, if necessary. The respective keys must be filled in manually.

If the "Cancel" or close button is pressed, the data will not be saved and the popup will be closed, if the "Ok" button is pressed and all the data is correctly defined the new device will be saved, the popup closes and the listing will be updated with the new device.

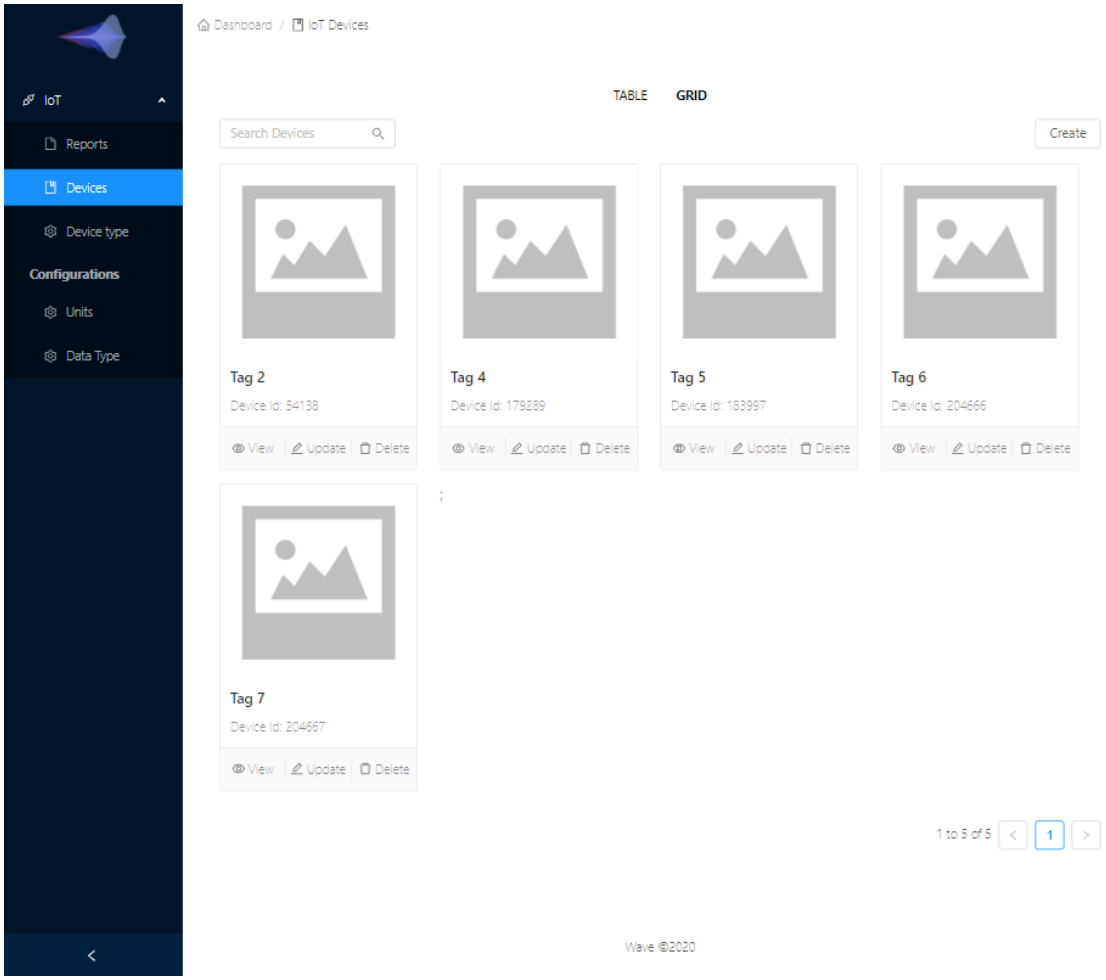


Fig. 16. Devices Menu - Grid View

The grid view (fig. 16) presents the information in a more visual way, showing the photograph of the device, its name and its id. There is also the possibility to edit and delete a specific device. At the behavioral level, these actions are the same as those implemented on the previous page, that is, when editing the same editing popup appears and when deleting the user is asked to confirm the deletion popup.

In both views of this menu, it is also possible to perform a more detailed view of a given device. To check the device in more detail, just click on the desired device id (table view) or in the "View" button (grid view) and the system will show a page with more detailed information.

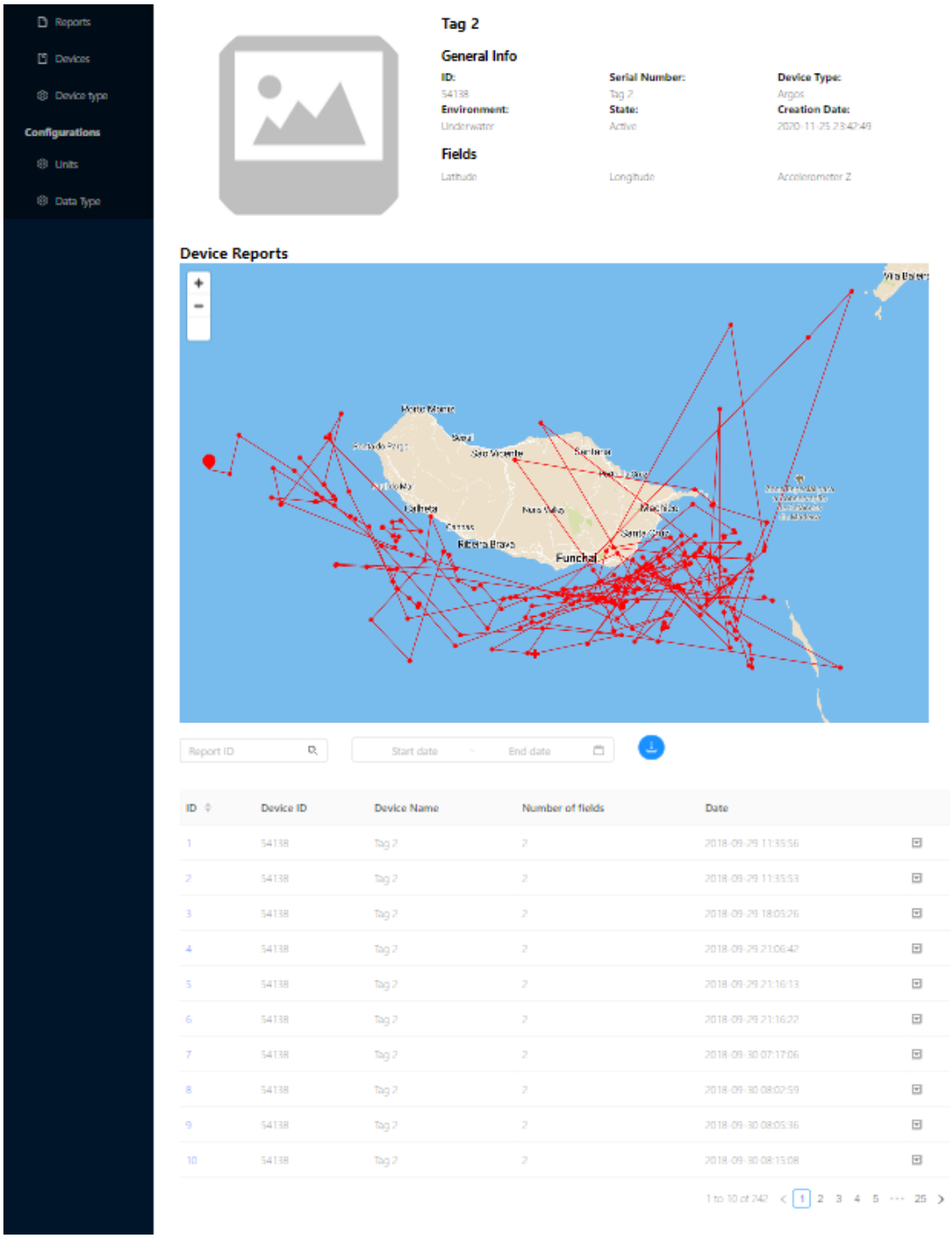


Fig. 17. Device Page

This new page (fig. 17) shows, at the top, a section with the links to consult the next and the previous device information, after that a section with the photo of the device, its name, its id, its serial number, the type of device, the environment, the status, the date of creation and the list

of fields that this device will transmit to the system. After this section with information related to the device, a map is shown where the various reports made by the device are marked, as well as a list in the form of a table with that same information. In order to make it easier to search for a single report, or even for a set of reports, filtering mechanisms have also been implemented, namely: the report identifier, and a minimum and maximum date for the report to have occurred. In the map, the reports are connected between them, following the order of happening. In order to make it easier to interpret which was the first and last reports, the first report is marked with a cross icon and the last one with a location pin icon, while the intermediate ones are marked with a simple circle. To know more about the report that is represented in the map, the icons are clickable, and presents in a popup the most important data, such as the report id, the name and id of the device that performed the report, the coordinates of the report and also a link to a page to view the report in more detail. In the table there is also the possibility of view more information of the report if the id is clicked, which will present the same page that will appear when the link at the popup shown when a point in the map is clicked; and also it is possible delete a specific report. To complete the deletion, and with the goal of avoid unexpected deletions, it is necessary confirm the popup that will appear. Another tool that exist in this page is the export tool. The export button that is located after the filter options creates a CSV file with all the reports performed by the device that is being consulted and fit the filter options, in a simpler way, it exports all the results that are currently being shown. The exported information is the report id, the id and the name of the device that performed the report, the date when the data was collected by the device, the date when the data was communicated to the system, and the list of fields in the form of a key-value pair.

3.3.5 Reports Menu

The last existing menu is where it is possible to consult the reports made by the devices and consult the data reported on them. In this menu (fig. 18), similar to what exists in the "Devices" menu, there are two different types of views to consult the data: table view and map view, which can be changed with a simple click on the desired view that is centered at the top of the screen. When loaded, the menu has by default the table view that shows to the users the following information: id, device id, device name, number of fields that exists in the report and date when the data was collected by the device. In addition to these data, there is the possibility of eliminating a specific report, for some reason that may occur. As in the previous menus, once again, to complete the deletion, it is necessary to confirm the popup. This menu also includes several filtering mechanisms that are common to both views, in order to facilitate the user when he wants to find a specific report or a set of reports. The available filters are the report id, the device id, the device name, and the minimum and maximum date on which the report occurred. There is also and export mechanism that generates and downloads a .CSV file with all the reports that satisfy the current filter search. The file contains the same information as described in the previous menu, that is, the report id, the id and the name of the device that performed the report, the date when the data was collected by the device, the date when the data was communicated to the system, and the list of fields in the form of a key-value pair. It is also possible to consult a specific report, where all the information related to it is shown in a more descriptive way, as can be seen in fig. 20. To this page

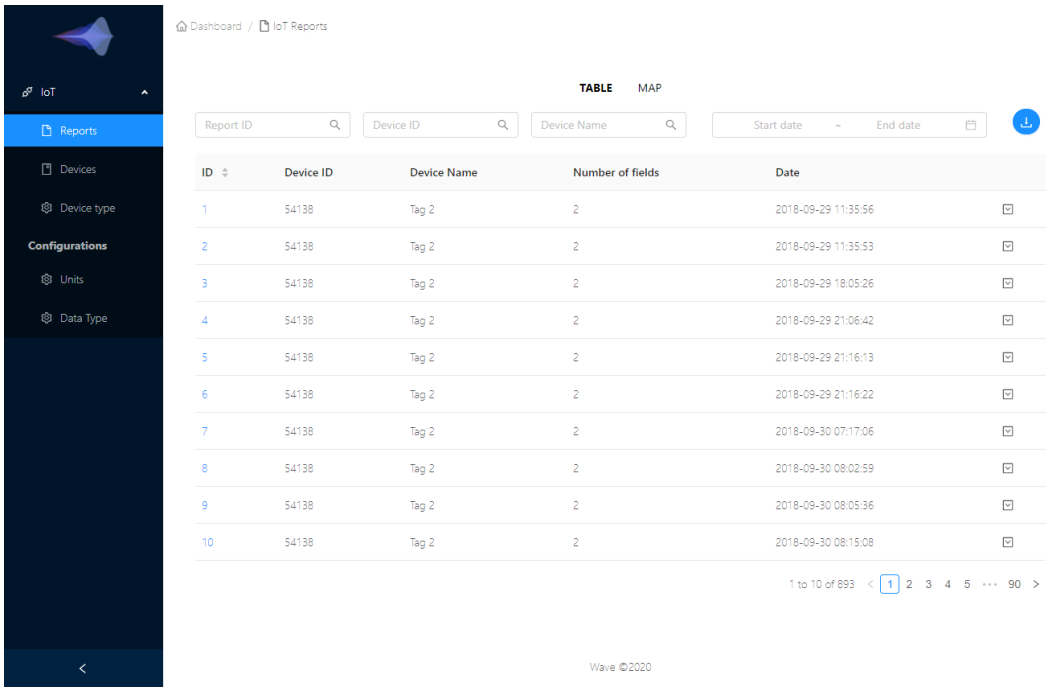


Fig. 18. Reports Menu - Table View

be shown, it is only necessary to click on the desired report id and the user will be redirected.

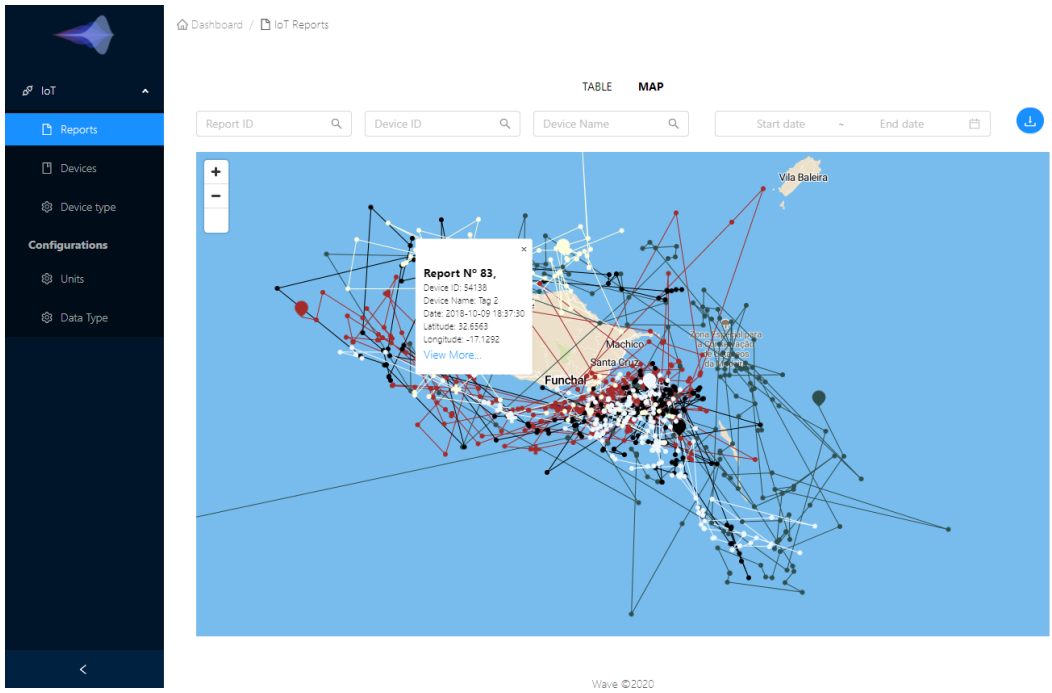


Fig. 19. Reports Menu - Map View

In the map view (fig. 19) it is possible to view the coordinates where the report data was collected. In order to make data interpretation even more visual, they are separated by device, being identified by different colors. They are also linked in order of occurrence through a straight line, with the icon in the form of a cross indicating the position of the first report that occurred for that device and the icon in the form of a location pin indicates the last known position of the device. All circular points represent intermediate reports. Was also implemented a popup that appears if any of the report icons is clicked. In this popup more information is shown, such as the report id, the name and id of the device that originated the report, the report date, it's latitude and longitude and also a direct link to the page for a specific report. It is important to note that all the filters available in the table view, and also the exporting mechanism are also available in map view.

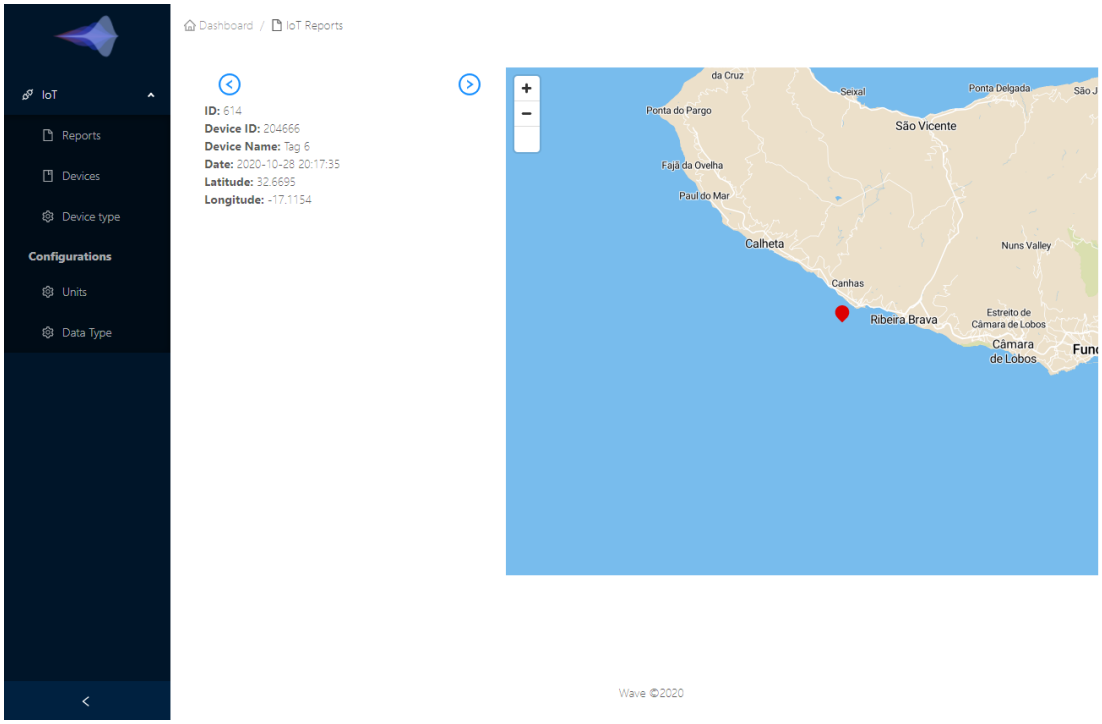


Fig. 20. Report Page

The page to consult a specific report (fig. 20) is made up of three sections. The first section, which is in the upper left corner of the page, is where the direct links to the next and previous reports pages are, if it exists. After that is other section where the information related to the report is shown, with the following data being presented: id, id and name of the device that generated the report, date on which the information was collected by the device, and finally the list of all data that exist in the report, being always shown in the form of "data type": "value". The last section is where the map is shown with an indication of the location where the report occurred.

This graphical user interface, as mentioned before in this manuscript, was implemented using the Laravel framework and React JS. The design of it was inspired in the website were it is incorporated, to improve the user experience and to have a uniform layout in all the web application.

The back-end of the project uses Laravel and some of its native tools such as QueryFilters to a faster and easier filter of results when querying the database, Models to do the mapping from the database table content to an object that will be used in the front-end and also back-end, JsonResource, that convert the model to a json object and Maatwebsite for the exportation of reports. It has also implemented migrations that creates the database structure required to the system with a Laravel command.

In terms of front-end it was used React JS and to also two libraries: the Ant Design of React [12] and Mapbox GL [13]. The Ant Design of React was used to render the biggest parts of the components, such as tables, buttons, text inputs, check boxes, select drop-downs, cards, pagination and popups. The Mapbox GL was used to create the maps, render the coordinates, the lines that join the various reports and the icons that mark them.

3.4 Database Architecture

In order for the system to be completely dynamic in terms of configurations of units, devices and information reported in the database, it was necessary to follow a structure that allowed this flexibility. Therefore, the tables were developed so that users could create, edit and delete their types of units, data and devices, using the developed interface, but it is also possible to directly manage in the database the types and environments of devices. The used database is an MySQL database because the data to be stored is structured.

The tables developed were as follows:

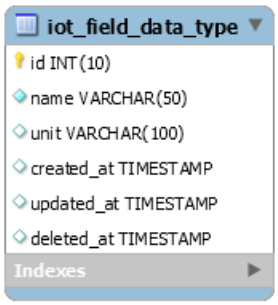


Fig. 21. Table iot_field_data_type

- **iot_field_data_type** - This table stores the records of existing units, which can be consulted in the Configuration Units menu of the web application. These values will be used to

[12] https://ant.design/docs/react/introduce
[13] https://docs.mapbox.com/help/tutorials/use-mapbox-gl-js-with-react/

define in which units the data is. The existing columns are as follows:

- **id** - Identifier of the unit type
- **name** - Name of the unit
- **unit** - Unit of the unit
- **created_at** - Date when the Unit was created
- **updated_at** - Date of unit last update
- **deleted_at** - Date when the unit was deleted

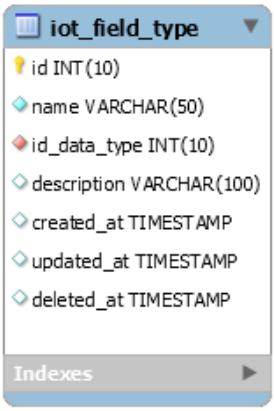


Fig. 22. Table iot_field_type

- **iot_field_type** - This is the table responsible for saving the data types that can be consulted in the corresponding menu. The information stored in this table will be used to identify the type of data to be communicated by the devices, including the unit and the name. The unit associated with the data type is obtained through the foreign key "id_data_type". The existing columns are as follows:

  - **id** - Identifier of the data type
  - **name** - Name of the data type
  - **id_data_type** - Foreign key that identifies the unit of the data type
  - **description** - Description of the data type
  - **created_at** - Date when the data type was created
  - **updated_at** - Date of data type last update
  - **deleted_at** - Date when the data type was deleted

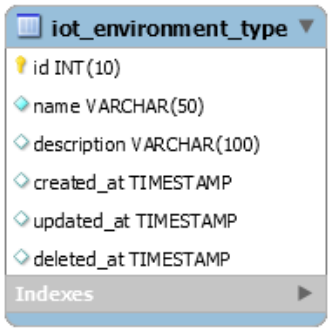


Fig. 23. Table iot_environment_type

- **iot_environment_type** - This table stores the existing environments where the devices will operate. This table can only be edited directly in the database an its values will be used by the table "iot_device". The existing columns are as follows:
  - **id** - Identifier of the environment type
  - **name** - Name of the environment type
  - **description** - Description of the environment type
  - **created_at** - Date when the environment type was created
  - **updated_at** - Date of environment type last update
  - **deleted_at** - Date when the environment type was deleted

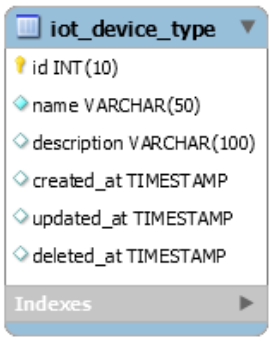


Fig. 24. Table iot_device_type

- **iot_device_type** - This table contains the types of existing devices and it content can be consulted in the "Device Type" menu. This same menu is where device types can be added, removed or edited. The values in this table will be used by the "iot_device" table to pre-define the data types that are used for a specific type of device, in order to simplify the process of creating a new device. The existing columns are as follows:
  - **id** - Identifier of the device type
  - **name** - Name of the device type

- **description** - Description of the device type
- **created_at** - Date when the device type was created
- **updated_at** - Date of device type last update
- **deleted_at** - Date when the device type was deleted

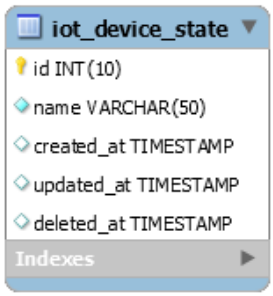


Fig. 25. Table iot_device_state

- **iot_device_state** - Here is where the possible states of a device are stored. This table can only be edited directly in the database. Its content will be used will be used by the "iot_device" table. The existing columns are as follows:

  - **id** - Identifier of the device state
  - **name** - Name of the device state
  - **description** - Description of the device state
  - **created_at** - Date when the device state was created
  - **updated_at** - Date of device state last update
  - **deleted_at** - Date when the device state was deleted

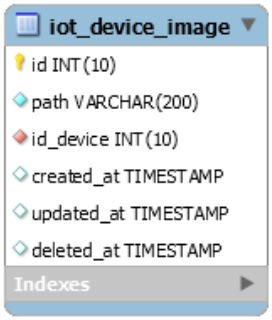


Fig. 26. Table iot_device_image

- **iot_device_image** -This table is where the information related to the device images will be stored. The table does not contain the images, but the information necessary for the system to locate the desired image. In order to identify which device each image belongs to, this table

has a foreign key with the device id to which it is associated. The existing columns are as follows:

- **id** - Identifier of the device image
- **path** - Path where the image is stored
- **id_device** - Foreign key that identifies the device with which the image is associated
- **created_at** - Date when the device image was created
- **updated_at** - Date of device image last update
- **deleted_at** - Date when the device image was deleted

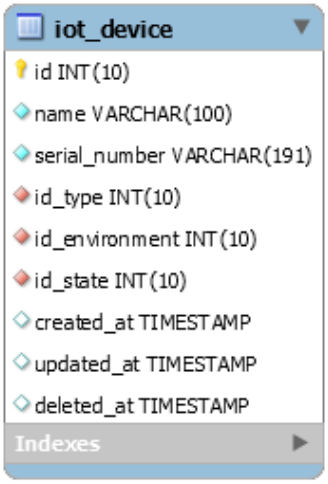


Fig. 27. Table iot_device

- **iot_device** -This is the table that contains all the devices that exist in the system and that can be consulted and managed from the respective menu. This is a table that contains several foreign keys in order to be able to identify the type of environment, device and state of each device. The data in this table will be used by the "io_device_report" table to identify which device originated each report. The existing columns are as follows:

  - **id** - Identifier of the device
  - **name** - Name of the device
  - **serial_number** - Serial number of the device (should be unique)
  - **id_type** - Foreign key that identifies the device type with which the device is associated
  - **id_environment** - Foreign key that identifies the environment type with which the device is associated
  - **id_state** - Foreign key that identifies the device state with which the device is associated
  - **created_at** - Date when the device was created
  - **updated_at** - Date of device last update
  - **deleted_at** - Date when the device was deleted

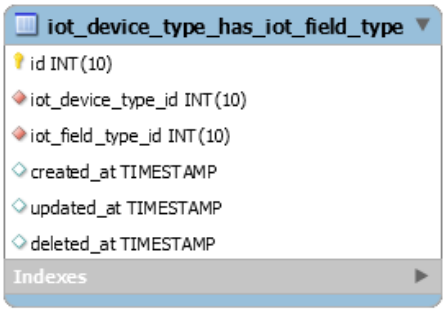


Fig. 28. Table iot_device_type_has_iot_field_type

- **iot_device_type_has_iot_field_type** - This is the relational table that allows the store of data types associated with each type of device. To make this possible, this table contains two foreign keys, one for the device type and another for the data type. The existing columns are as follows:
  - **id** - Identifier of the relation between the device type and the data type
  - **iot_device_type_id** - Foreign key that identifies the device type
  - **iot_field_type_id** - Foreign key that identifies the data type
  - **created_at** - Date when the relation was created
  - **updated_at** - Date of relation last update
  - **deleted_at** - Date when the relation was deleted

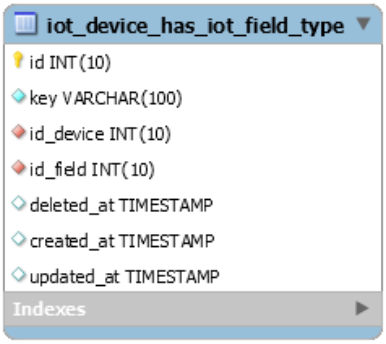


Fig. 29. Table iot_device_has_iot_field_type

- **iot_device_has_iot_field_type**- This is the relational table that allows the store of data types effectively associated with each device. To make this possible, this table contains two foreign keys, one for the device and one for the data type. The information in this table is used to process the data for each report made by a device. The existing columns are as follows:

  - **id** - Identifier of the relation between the device and the data type
  - **key** - Key word that is used to mapping the data type in database with the data that is reported by the device

- **id_device** - Foreign key that identifies the device
- **id_field** - Foreign key that identifies the data type
- **created_at** - Date when the relation was created
- **updated_at** - Date of relation last update
- **deleted_at** - Date when the relation was deleted

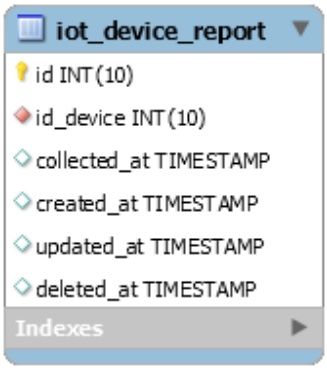


Fig. 30. Table iot_device_report

- **iot_device_report** - The reports made by the devices are stored in this table. This table only contains information of the date on which the reported data were collected and reported, and which device it is associated with trough the foreign key "id_device". The collected data types values are not stored in this table. The existing reports can be consulted in the respective menu. The existing columns are as follows:

  - **id** - Identifier of the report
  - **id_device** - Foreign key that identifies the device with which the report is associated
  - **created_at** - Date when the data of this report was collected by the device
  - **created_at** - Date when the report was created
  - **updated_at** - Date of report last update
  - **deleted_at** - Date when the report was deleted

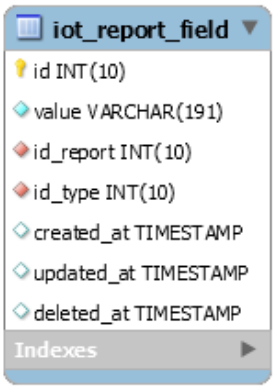


Fig. 31. Table iot_device_report_field

- **iot_report_field** - It is in this table that the values of the reported data types are stored. For the value to be related to reporting and data types, there are foreign keys from the "iot_device_report" and "iot_field_type" tables, respectively. The values stored in this table can be consulted on the report page built for this purpose. The existing columns are as follows:

  - **id** - Identifier of the report field
  - **value** - Value of the field
  - **id_report** - Foreign key that identifies the report with which the report field is associated
  - **id_type** - Foreign key that identifies the data type with which the report field is associated
  - **created_at** - Date when the report field was created
  - **updated_at** - Date of report field last update
  - **deleted_at** - Date when the report field was deleted

As described in the previous section, the tables are related to each other through relationships using foreign keys or relational tables, in order that performance is higher. The relational schema can be consulted in the next image.

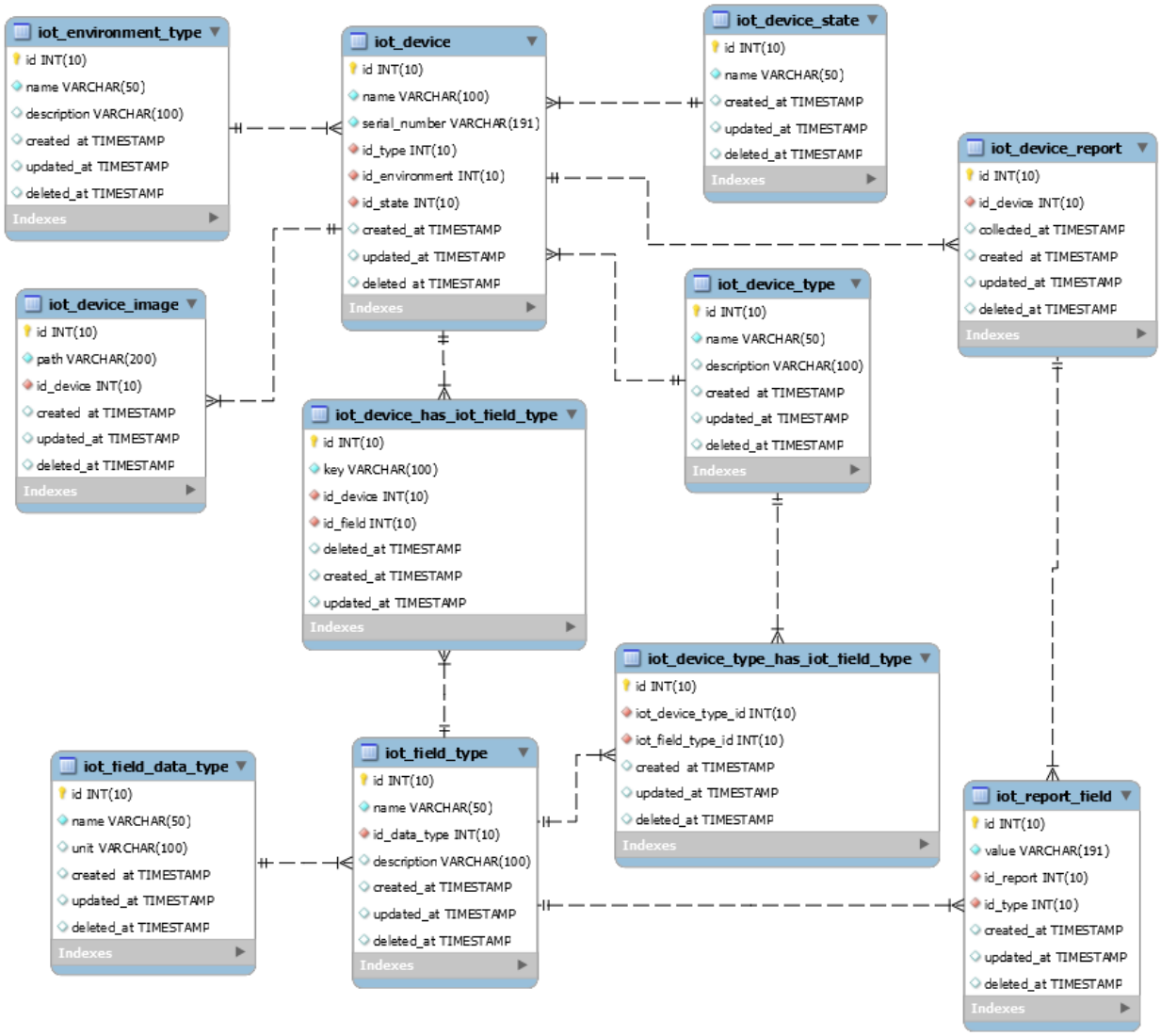


Fig. 32. Database Relational Schema

This database with only fourteen tables allows the system to have all the flexibility idealized so that the requirements initially thought are met and the system has a good reading and writing performance.

3.5 Hardware

This section describes how the Lopy4 device was used to make requests to the API in order to transmit data to the system via the API.

As it was not possible to integrate with real devices that were currently in full operation and collecting data, a laboratory simulation was carried out to simulate the communications that would

be made from real devices in order to test the API and also to populate the system with real data. The process of obtaining the used real data is described in the section 4.1.

After obtain the csv files with the real data, they were stored in an memory card that was used in the Lopy4. It was also implemented a program in this device that loops the existing files, and for each existing file it creates a POST request to the developed API in order to communicate the existing data in the files to the system.

For each file, the program defines the request header "id-device" with the id that the device has in the source and creates a body with the fields:

- "lat" - that contains the latitude where the collection of data was performed by the device
- "lon" - that contains the longitude where the collection of data was performed by the device
- "collected_at" - that contains the timestamp when the collection of data was performed by the device

After the request is ready to be sent the connection with the API is made and the data is collected by the system. In order to visualize better the developed flow, it can be consulted in fig 33.

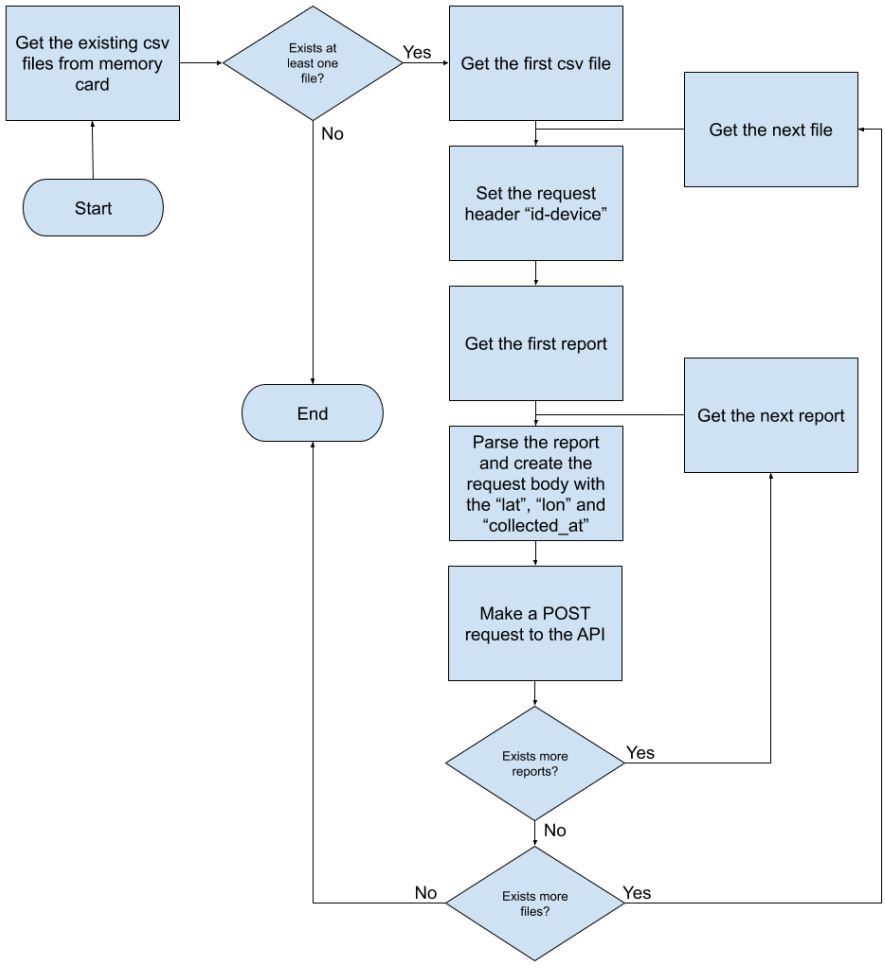


Fig. 33. Lopy4 Behaviour Flowchart

This was the way to get around the limitation of not being able to test with a real device, not only because of the difficulty in finding one, but also because of the pandemic situation experienced today. The choice to create this program on a Lopy4 instead of being made through a program that runs on a computer was due to the fact that it tests effectively with an IoT device, which are the main targets for using the API.

All the devices, data types and units that were used in this process were configured previously in the back office of the system.

3.6 Data Types

For this particular study, only the GPS coordinates (Latitude and Longitude) were obtained. But in order to assess the biodiversity from remote areas, back-end system is implemented to collect as many data types, as desired. In order to have some initial values already configured in the database the following list of units was created:

- **Counter** - Integer that can be used for any data type that does not have a unit and wants to indicate a count.
- **Coordinates** (Degrees) - Number that indicates a coordinate, whose precision is variable. Its unit is indicated in degrees.
- **Accelerometer** (G) - Number that will be used to measures proper acceleration, that is the acceleration of a body in its own instantaneous rest frame. The used unit is G-forces.
- **Rotation** (centiDegrees) - Number that indicates the rotation, expressed in centiDegrees.
- **Power** (mAh) - Number defined to indicate the power in milliampere hour.
- **Signal** (dBm) - Is the unit of level used to indicate that a power level is expressed in decibel-milliwatts.
- **Time** (ms) - Unit used to measure the time in milliseconds.
- **Frequency** (Hz) - Created unit to measure the frequencies in Hertz.
- **Temperature** (°c) - Unit to measure the temperature in degrees Celsius.

To use the above mentioned units, the data types are also pre-populated with initial data, based on the predetermined types of data that the IoT devices will report:

- **Report Number** (Count) - The internal device number of the collected report.
- **Collection Interval** (Time) - The time needed to perform the collection of data that the report contains.
- **Latitude** (Coordinates) - Latitude were the collection was made.
- **Longitude** (Coordinates) - Longitude were the collection was made.
- **Accelerometer X** (Accelerometer) - Acceleration value along the X axis
- **Accelerometer Y** (Accelerometer) - Acceleration value along the Y axis
- **Accelerometer Z** (Accelerometer) - Acceleration value along the Z axis
- **Pitch** (Rotation) - Rotation around the side-to-side axis.
- **Roll** (Rotation) - Rotation around the front-to-back axis.
- **Yaw** (Rotation) - Rotation around the vertical axis.
- **Device Battery** (Power) - Power of the device battery.
- **Device Temperature** (Temperature) - Temperature of the device when the data is collected.

- **Water Temperature** (Temperature) - Temperature of the water when the data is collected.
- **Animal Temperature** (Temperature) - Temperature of the animal when the data is collected.
- **Rx Timestamp** (Time) - Internal timestamp that contains the last received packet.
- **RSSI** (Signal) - Received signal strength.
- **SNR Signal to Noise** (Signal) - Signal to noise ratio
- **SFRX Spreading Factor** (Counter) - Data rate (in the case of LORAWAN mode) or the spreading factor (in the case of LORA mode) of the last packet received.
- **SFTX Spreading Factor** (Counter) - Data rate (in the case of LORAWAN mode) or the spreading factor (in the case of LORA mode) of the last packet transmitted.
- **TX Trials** (Counter) - Number of tx attempts of the last transmitted packet.
- **TX Power** (Power) - Power of the last transmission
- **TX Time on Air** (Time)- Time on air of the last transmitted packet
- **TX Counter** (Counter) - Number of packets transmitted
- **TX Frequency** (Frequency) - Frequency used for the last transmission.
- **Land Battery** (Power) - Battery power of the receiver located in land.

It is important to emphasize that these are pre-filled values that serve as a basis for them to serve as an example for the creation of future units and data types, or even to test a device whose reported information is already present in these configurations. It is possible at any time to add more units or data types according to the need for each specific device.

4 Evaluation

This section describes the obtained sample data for portraying the collected data from aquatic IoT and also the description of the performed user studies.

4.1 Sample Data

Coordinates were obtained from public available dataset [14] This dataset was proposed by the marine biologists to be the state of the art mechanism for gathering the biotelemetry from diverse types of cetaceans. However, after internal consultations with the two experts, such platform was found to be cumbersome to be used, which further motivates the proposed dissertation. Collection date, GPS latitude and longitude was downloaded into an CSV file, from which further uploaded onto an IoT device LoPy4, previously depicted in Figure 3. Goal of this approach is to simulate the realistic obtained data from the field.

After the development of all the necessary mechanism to import the data obtained using the Lopy4 via API, and after this communications have been made successfully, the database obtained the following five IoT devices and its reports:

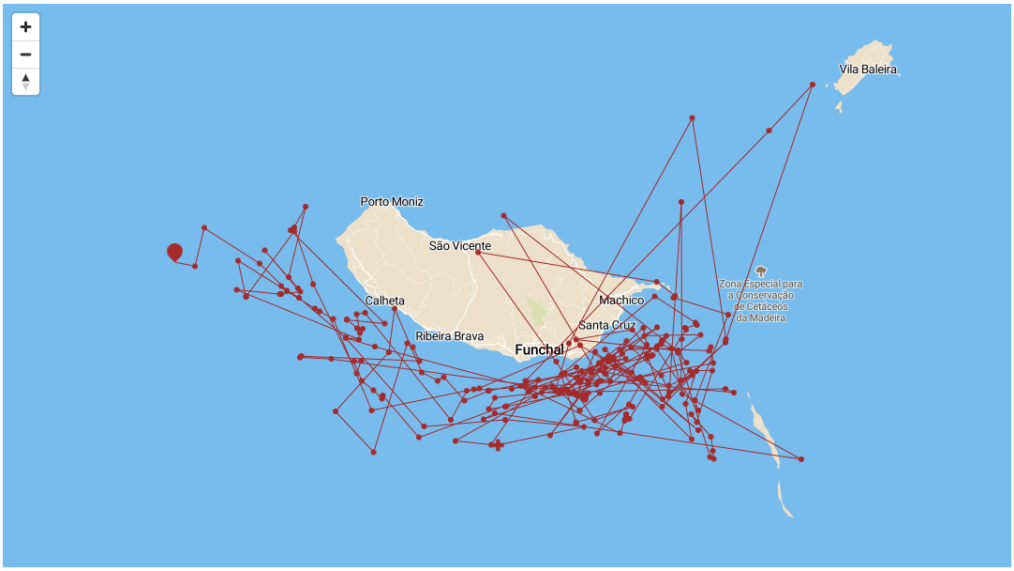


Fig. 34. Reports of Tag 2

- **Tag 2** - This device, with the Id 54138, has 242 reports associated. The reports ids goes from the 1 to 242 and they are performed between 2018-09-29 and 2018-10-27. These reports only

[14] Supported by the projects MARCET, MARINFO, OOM, and MARCET2. Data are accessible at: `https://www.movebank.org/cms/webapp?gwt_fragment=page%3Dsearch_map_linked%2CsensorTypeId%3D82798%2CstudyIds%3D886013997%2Clat%3D32.7115981735054 2%2Clon%3D-17.026564306640434%2Cz%3D10`

have the collected date and the coordinates where the report was collected. The map generated when consulting the reports of this device is illustrated in the figure 34.

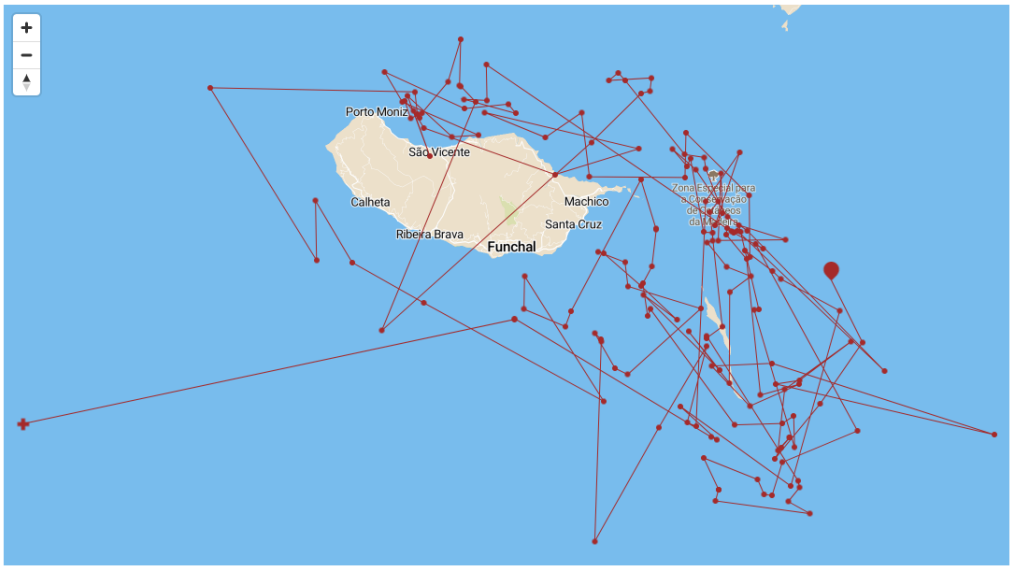


Fig. 35. Reports of Tag 4

- **Tag 4** - This device, with the ID 179289, has 167 reports associated. The reports id goes from the 243 to 410 and they are performed between 2019-05-30 and 2019-06-15. These reports only have the collected date and the coordinates where the report was collected. The map generated when consulting the reports of this device is illustrated in the figure 35.

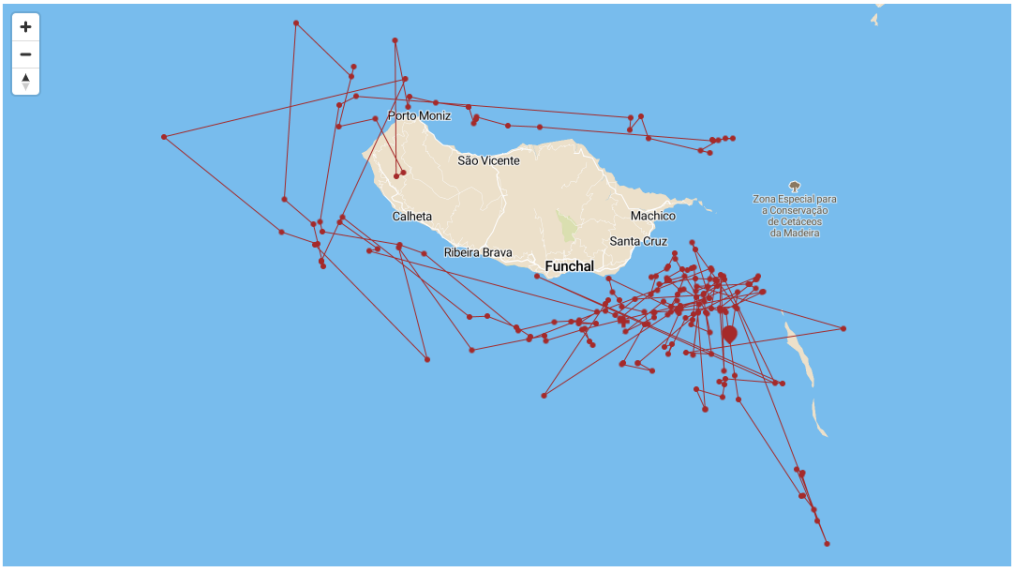


Fig. 36. Reports of Tag 5

- **Tag 5** - This device, with the ID 183997, has 201 reports associated. The reports id goes from the 411 to 612 and they are performed between 2020-01-26 and 2020-02-16. These reports only have the collected date and the coordinates where the report was collected. The map generated when consulting the reports of this device is illustrated in the figure 36.

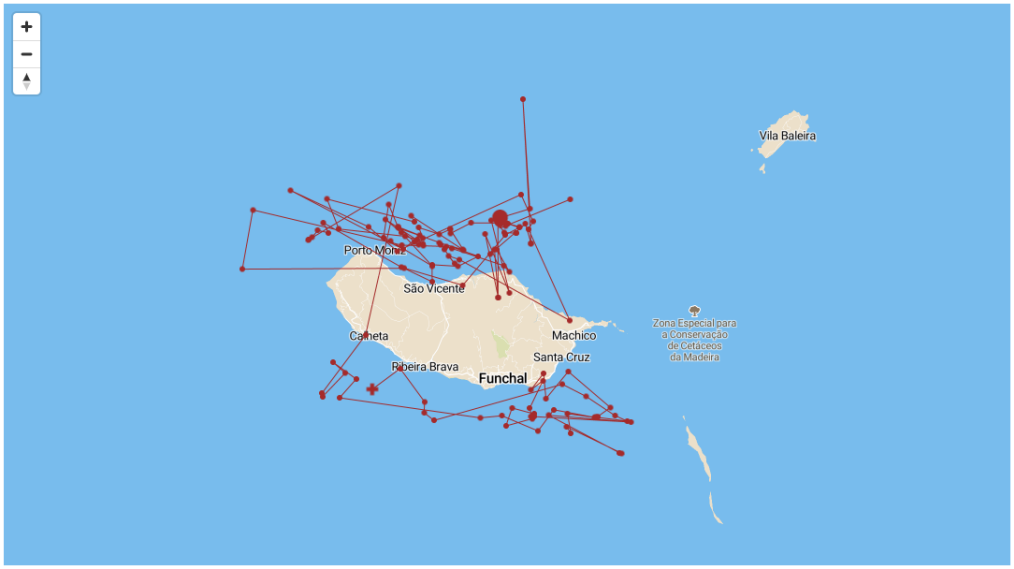


Fig. 37. Reports of Tag 6

- **Tag 6** - This device, with the ID 204666, has 132 reports associated. The reports id goes from the 613 to 745 and they are performed between 2020-10-28 and 2020-11-10. These reports only have the collected date and the coordinates where the report was collected. The map generated when consulting the reports of this device is illustrated in the figure 37.

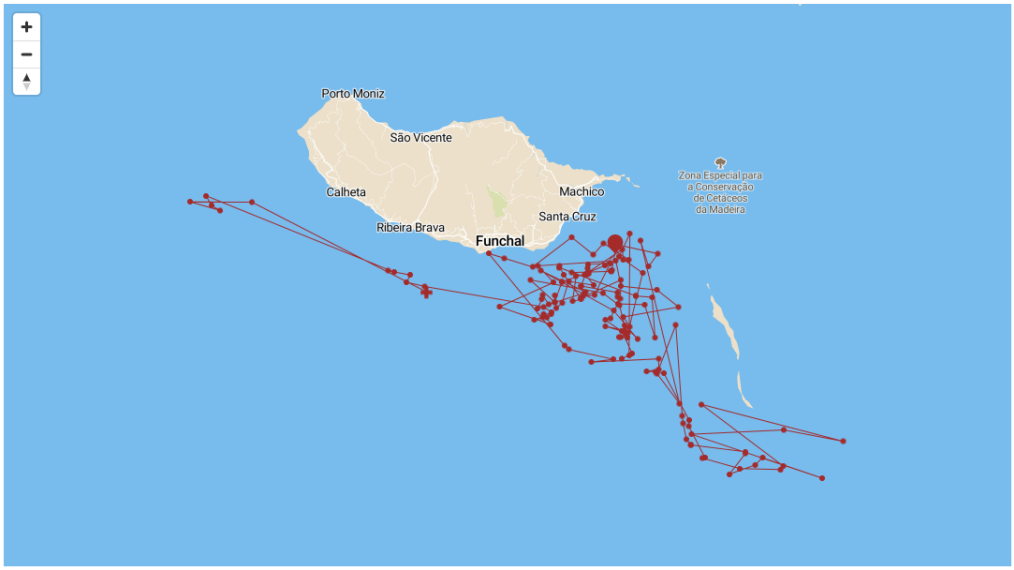


Fig. 38. Reports of Tag 7

- **Tag 7** - This device, with the ID 204667, has 147 reports associated. The reports id goes from the 746 to 893 and they are performed between 2020-11-09 and 2020-11-20. These reports only have the collected date and the coordinates where the report was collected. The map generated when consulting the reports of this device is illustrated in the figure 38.

In total the system has 5 devices, with a total of 893 reports ranging from 2018-09-29 to 2020-11-20.

4.2 User Study

In order to check if the system is enough intuitive to use with the implemented design, a first user study has been conducted only with a marine biologist, with the goal to collect suggestions to improve the system and in a second phase perform more tests with more users.

After collecting the inputs of the first user test and made some of suggested changes, a second user study was performed with two different groups of users:

- Nine Software Engineers, that have experience using many different types of layout.
- Two Marine Biologists, that have the knowledge of marine species and field experience using systems to retrieve information from the environment.

The first user study was conducted asking to the user to complete some tasks and giving some feedback of the usability of the menus. The tasks that were requested are intended to simulate the necessary steps to configure a device so that it is ready to work with the implemented system. The tasks were as follows:

- Go to the website `http://wave.arditi.pt/`
- Register a new user

- Go to Configuration Units Menu
- Create a new unit
- Go to Configuration Data Type Menu
- Create a new Data Type
- Go to IoT Device Type Menu
- Create a new device type
- Go to IoT Devices Menu
- Create a new device

The second performed user study has some more steps due the changes made and also because all the features had already been implemented:

- Go to the website `http://wave.arditi.pt/`
- Register a new user
- Go to Configuration Units Menu
- Create a new unit
- Go to Configuration Data Type Menu
- Create a new Data Type
- Go to IoT Device Type Menu
- Create a new device type
- Go to IoT Devices Menu
- Create a new device
- Consul the map view
- Filter the results performed by the device with the name "Tag 2"
- Explore the coordinates
- Consult a specific report information
- Download the report data of that device

In order to measure the usability of the system, a SUS [18] test was conducted. The SUS is a 10 item questionnaire for measuring the usability of a system and it was invented by John Brooke in 1986. The used questions are following:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Each question of this SUS, has 5 possible answers and uses the following scale:

- **Strongly Disagree**: 1 point

- **Disagree**: 2 points
- **Neutral**: 3 points
- **Agree**: 4 points
- **Strongly Agree**: 5 points

After the questionnaires were completed, the data was saved and the score was calculated for each questionnaire. The score is a value between 0 and 100, and is obtained as follows:

1. Sum all the odd-numbered question results and subtract 5 to it.
2. Sum all the even-numbered question results and subtract from 25 the obtained sum.
3. Sum the obtained result in point 1 with the obtained result in point 2.
4. Multiply the result obtained in 3 by 2,5.

The SUS score ranges of values used as reference for the usability classification are the following [18]:

- **More than 80.3** - Excellent result. The user loved the system usability.
- **Between 68.1 and 80.3** - Good result. The system is well built, but there is space for improvements.
- **68**: Satisfactory result. This is the average of results.
- **Between 51 and 67.9**: Poor Result. The system is usable but it must be improved in order to get better results.
- **Less than 51**: Negative result. The system should be rethought and that should be a priority.

As described in section before, some usability tests were carried out for different types of users. In order to have an assessment by someone who may become a frequent user of the system, two persons in the marine area were asked to carry out the test.

One of them was Rui Prieto da Silva [15], born in Luanda in 1969, he is graduated in Marine Biology in university of Algarve, in 1997 and did his PhD degree in University of the Azores in 2014. During his career he developed research on marine mammal behaviour and ecology in many Research and development projects developed through the the Department of Oceanography and Fisheries/ Universidade of the Azores, the Institute for the Conservation of Nature, the multipolar RD&I consortium Marine and Environmental Sciences Centre (MARE) and the Inter-Universities consortium Institute of Marine Research (IMAR). He has also published 45 articles in Peer Reviewed Journals, 3 books and has thousands of citations.

The other was Marc Fernandez Morron [16] , born in Spain in 1981. He graduated in Marine Scienses and as soon as he finished his master's degree he started working in the field as a biologist in Costa Rica. Since then, he has played an active role in marine research, with several articles, international collaborations and lectures, having also been a professor at the University of the Azores. Marc is an Principal investigator (PI) of Monicet (a citizen science platform, allowing

[15] https://orcid.org/0000-0002-0354-2572
[16] https://www.researchgate.net/profile/Marc_Fernandez

the reports of marine species by the uploaded photographs of dorsal fins and flukes from the sea-vessels.[17])

Both users are the experts marine ecologists which conduct research with cetaceans and marine megafauna, with whom several conference calls were made, in order to understand the challenges in depicting the collected data.

Other user sample that performed the user tests were computer engineers with ages between 23 and 34 years old.

The first test was performed by Rui Prieto when the system was not completed, with the goal of receiving a feedback of what could be done to improve the actual system. The SUS results of it were the following:

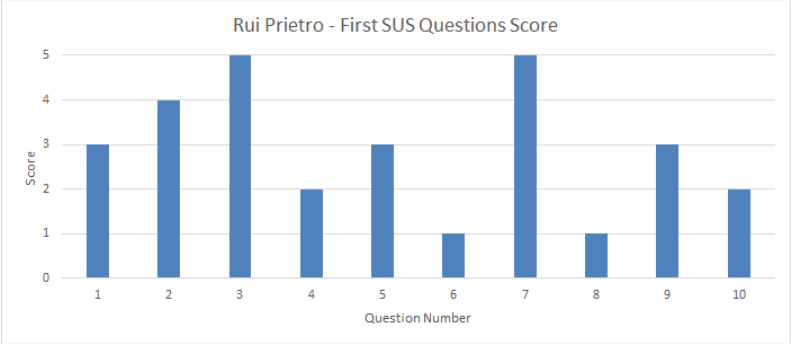


Fig. 39. First SUS performed by Rui Prieto

In this first SUS, the obtained score was 72,5 with is already good. After finish the implementation of the missing features Rui performed a new user test in order to compare the results with the first one, and the results were the presented in the figure 40:

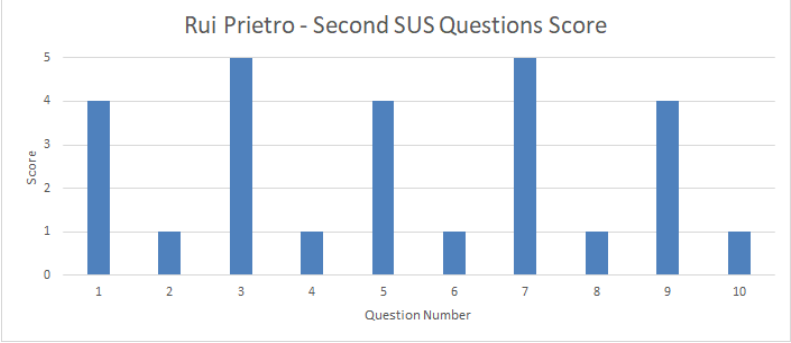


Fig. 40. Second SUS performed by Rui Prieto

[17] see `http://www.monicet.net/en`

In this second test, the obtained score was 92,5 which is a substantial improvement comparing with the first one. I am certain that this difference is due the implementation of some suggestions and also because the system was completely function which made the user experience much more complete.

Following graph contains the average score results and their standard deviation of the tests carried out by the two marine biologists. For the elaboration of these graphs, Rui Prieto's second test was used, not the first.

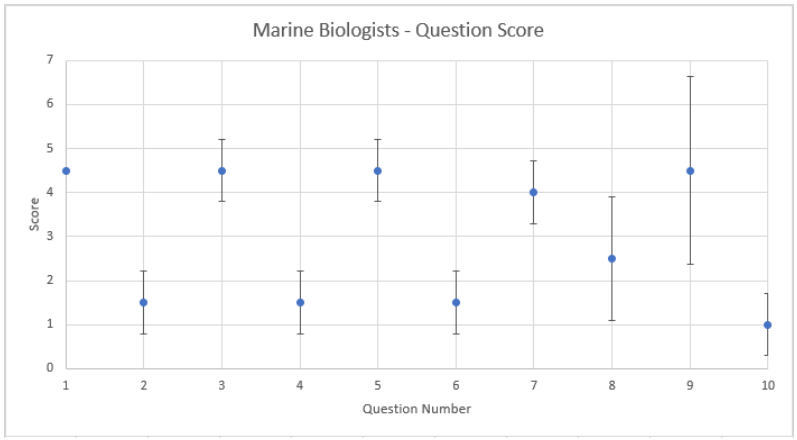


Fig. 41. Question Results Of Marine Biologists

The SUS result of the tests carried out by the marine biologists gave an average of 85 as we can see in figure 42.

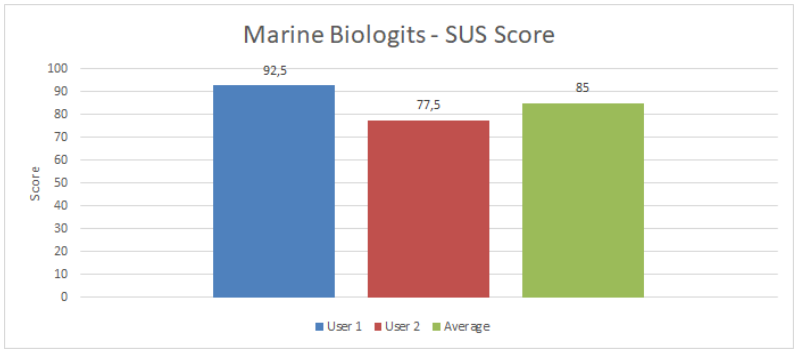


Fig. 42. SUS Results Of Marine Biologists

The average score results and their standard deviation to the ten questions in the SUS questionnaire made by software engineers can be found in the following figure:

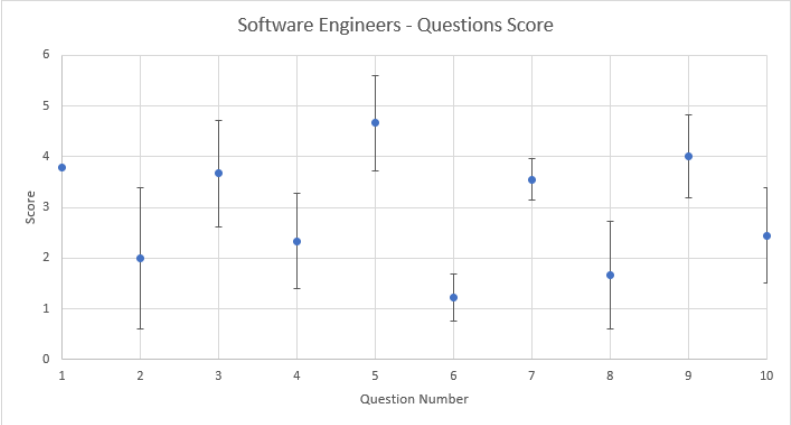


Fig. 43. Question Results Of Software Engineers

The result of the SUS carried out by software engineers gave a slightly lower average than that of marine biologists, with an average of 82.5. Although a little lower it remains a good result. The results of each user and their average can be found in figure 44

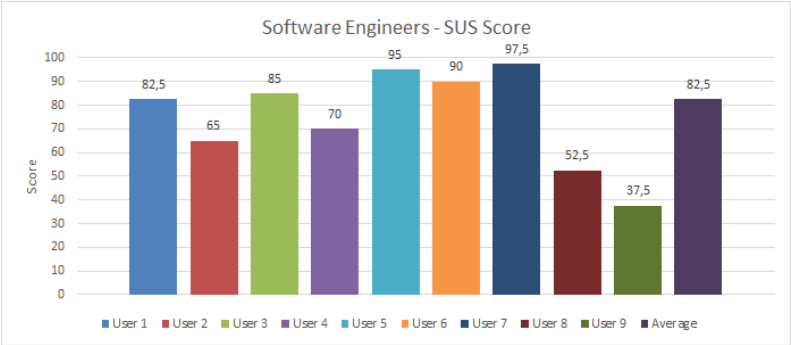


Fig. 44. SUS Results Of Software Engineers

In order to have an overview, with the results of all tests, of all users, a graph was created that illustrates these results and can be consulted in figure 45.

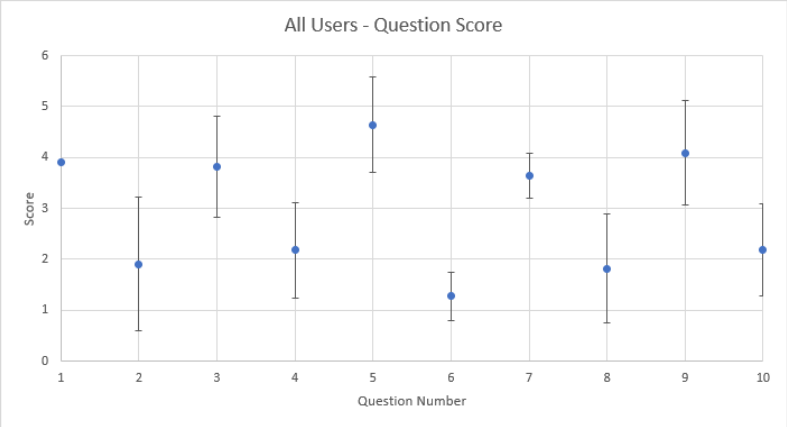


Fig. 45. Question Results Of All Users

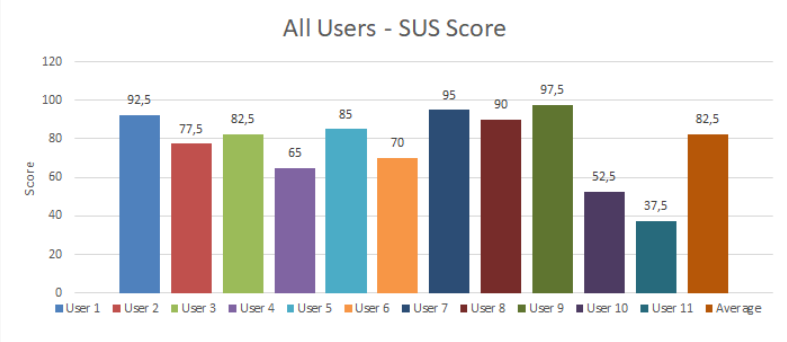


Fig. 46. SUS Results Of All Users

The same approach was made to the final result of SUS, and the resulting graph can be consulted in figure 46. Where it is possible to view individual results and also the average of results.

5 Discussion

In below, thesis summarizes obtained results, outlining constraints and proposing future studies for further improvement of the real-time API for marine ecology and aquatic IoT.

5.1 Results Interpretation

This section presents the interpretation of the data obtained that were presented in section 4.

The data obtained from public repositories, and imported into the system from the simulation of a real device using a Lopy4 was undoubtedly positive. Thanks to this data, it was possible to test all system components and validate their correct functioning. The reports were all transmitted via API, and consulted from the GUI's report menu to verify correct import. This methodology required rigorous checks to prove that the imported data was indeed correct, which was verified. Thus being able to affirm that the system works as expected, both at the level of the API, the database and the web application, as well as the integration of IoT devices that after the tests carried out with Lopy4 I verify that they can be integrated without many efforts.

The import of these reports also served to verify that the platform's performance was maintained, even with 893 performed reports in its database, each one with 2 fields (latitude and longitude), there is no slowdown in the database query operations. These 893 reports are already a good basis for showing the platform's potential. It should be noted that the points that appear on land and not on water are due to an error in obtaining the coordinates by the device and not due to errors in the developed application, or at the importation process of the reports.

At the level of user testing, the results were also satisfactory. There was an SUS improvement from the first test carried out by Rui Prieto to the second, which means that the improvements and new features have taken effect and allow users to have a more intuitive and practical use of the system. The tests carried out by marine biologists showed a higher SUS average than that obtained by software engineers, as would be expected, but this is not necessarily bad news, since in general the obtained result was 82.5 which is much higher than the average used as a reference that is 68 [18]. This result indicates that the usability test by users was successfully passed.

In addition to the SUS scores, these tests also allowed me to obtain direct feedback from users and some suggestions for improvements proposed by them are described in the section 6.

5.2 Research Contributions

In this section we will attempt to answer the RQ stated at the start of this manuscript.

- **RQ1 - How to successfully receive the environmental telemetry in NRT using LoRa?**
  Clearly, reported research provided the proof-of-concept in using the combination of REST API

and LoRa gateway receivers, which forward the data packets from the IoT devices directly to the backend. Further research should verify the software using the in-situ deployment, instead of the simulated device. Also, further experiments should verify the retrieval of the data from the IoT devices, using encryption layers through joining (e.g. using Over-the-Air Activation (OTAA) or Activation by Personalization (ABP), securing further the data transmission.

- **RQ2 - How marine ecologists evaluate such apparatus?**
  The feedback given by the marine biologists to the system was very positive, in addition to the good result of the usability test by both (Rui Prietro and Marc Fernandez), their favourite aspects was related to the way that the data was shown to the users and the fact that this system has a flexibility that allows integration with many different types of devices. Some suggestions for improvements were also given and added to section6 of the manuscript.

5.3 Constraints

Although thesis provided the first pipeline for integrating diverse IoT devices, there is a significant space for improvements. For instance, since the scope of the thesis was to provide the pipeline for the aquatic IoT devices to be used, tests were performed with the simulating microcontrollers in-vitro. Conversely, real-time usage of such tags with the provided telecommunication structure (senders, relays and gateways) should further test the proposed API pipeline.

6 Conclusion

Research provided in this thesis contributes directly to the creation of the real-time Application Programming Interface (API), allowing the multimodal and customized data input for diverse types of state-of-the-art tags. In the proposed work, thesis creates the real-time API for depicting the Aquatic Internet of Things. Such system advances the state of the art in the biotelemetry systems, surpassing the limitations of the current public software (e.g. MoveBank), which remain cumbersome to be used.

Introduction (Section 1) targeted the importance of having the need of a software to assist the marine ecologists in their biodiversity monitoring, by managing diverse types of sensors and their data input. Related Work (Section 2) described typical Aquatic Internet of Things devices as sensors, as well as possible telecommunication protocols (such as LoRa) in allowing the retrieval of such monitoring devices remotely. System (Section 3) depicted the used and implemented backoffice (comprised from front-end and back-end), allowing the marine ecologists and other stakeholders to manage the IoT devices as well as to manipulate with the obtained data inquiry. Methodology (Section 4) depicted the conducted user study with 2 marine ecologists as well as with 9 computer engineers, when using such system, with obtained data from the device and ratings by the end-users in Results (Section 4).

Currently the system is complete in operation and ready for the first real devices to be integrated, but there are always improvements that can be made, or new features that can be added. The ideas presented below are the results not only of ideas that emerged during the development of the project, but also suggestions from users who carried out usability tests. All together resulted in the following list of future work:

- **Receive files in the reports** - Currently it is only possible to collect report data with a value that can be represented in text, and it is not possible to receive videos, images, sound files, or any other type of files, which represents a limitation for the system. Although this type of information currently collected is sufficient for most of the devices that collect marine information, it would be interesting to implement a mechanism that allows the reception of files through the API and their subsequent visualization when consulting the reported data. This implementation would bring to the system an even greater versatility and open doors for integrations with devices that collect this type of information.

- **Creation of new endpoints in API Rest** - This improvement consists of the creation of new endpoints in the API so that it becomes more complete and it is possible for third parties to obtain or edit existing information and not just create new reports. Endpoints may be implemented to obtain data from a specific device, from all existing devices or from those that satisfy a certain sent condition in the request, for example to be in the active state, or to have been created between two specific dates. An endpoint can also be created that will do the same for units, for data types, device types and reports. In this way the information will be accessible in applications that intend to integrate the solution developed in this project in some way. Still at the API level, another possible improvement would be the implementation of an endpoint that allows the device itself to register and configure itself automatically

in the system through a request, without the need for configurations in the back office by a user.

- **Importing and exporting data** - This improvement comes in the sense of allowing integration with devices that are already being used for data collection and already have several data reported and intend to consult not only the new data on the platform, but also the old ones. Another case in which it would be advantageous to have this type of tool would be for cases in which it is only intended to visualize existing data and not to collect new ones. In order for this consultation to be possible, a mechanism capable of importing and interpreting a given file in CSV, or other formats must be developed and inserting this information into the database. It would also be interesting to export data in more than one format. Currently the export of reports is done in CSV, but the export could be implemented in more formats and the user would choose the desired one when exporting the data.

- **Data visualization** - In order to allow the user to view the reports even more intuitively, appealing and interactive, some changes could be made to the report query maps. These changes would include the introduction of tools that allow the user to choose the colors and the information shown on the map according to each device, to choose which information they want to appear (first report icon, intermediate report icons, last report icons, line between reports). Another interesting feature would be to develop an animation that would appear on the map in order of occurrence and show the user the route of the device from the beginning to the end, making the user's experience more visual and making it easier to understand which path was taken by the device. Another improvement that could be made is the implementation of other ways of visualizing information, for example through a graph. This implementation would give to the system even more flexibility and versatility.

- **Change history** - In order to be able to better control the actions taken by users on the various components, it could be interesting to have a history that kept that information. This information, after analyzed and studied, could serve to implement changes that would lead to better use by users, or even to ascertain the cause of any changes in system configurations. It would also be interesting to associate the user responsible for the creation of a unit, a data type, a field type or a device to that same creation.

- **Alert system** - This would be one of the most important future implementation, as it was one of the improvements suggested by users who participated in usability tests. The idea of this feature was to have an alert management menu configurable according to the needs of different users. In this new menu it would be possible for users to configure which types of alerts are intended, which devices would be targeted by this configuration and how they would be alerted (email, whatsapp or other tool). Some of the suggested alerts were:

    - **Report value alert** - This alert consists of defining, for a given device and for a given data type, a value and a condition (equal, major, minor, an interval, etc ...) and if that condition is verified in one of the reports of that device, it would be generate a notification for the user. For example, an alert would be defined for when the value of the data type

"water temperature" was higher than 26°C, if a report was detected in which the value of that data type was higher than 26°C an alert would be generated and sent to the user.

- **Area alert** - This alert can be very useful if the user wants to be more attentive to a device if it is in a certain area, or outside of it. The idea is to allow the user to define, on a map, one or more areas of their choice and choose whether to be alerted when a report is made inside or outside that defined area. In this way it will be possible for the user to pay attention to the specific device only when it is in an area that is relevant to the study being conducted.

- **Time without response alert** - With this type of alert implemented, the user will be able to define a maximum time interval that, if it is exceeded without receiving a report from a specific device, will generate an alert. It could be an important alert to detect any malfunction or anomaly in the device.

- **Predict future values of reports** - An interesting feature to implement in the system would be, through artificial intelligence, to predict the result of future reports. Based on the analysis of the reports already made, the system could predict, for example, the migration route of a certain animal that was being monitored. This is just an example of an application of this feature, but its use can cover more fields than just the prediction of coordinates.

With the improvements described above the system would be even more complete and with a much more immersive and personalized operation, appealing even more to its use.

Asides from the implemented software, which was hypothesized and verified to bring the ease to marine biologists to gather remote marine data, working on this dissertation was enriching both personally and academically. Indeed, marine life has never been a topic that aroused the special interest to the dissertation author, however after starting to work on the project the author started to have more interest in the theme. There is a world of possibilities not yet explored in this area and with the advancement of technology it is more simple and important to contribute. The author of dissertation also had s contact with new technologies, which will certainly be useful in further professional career. Among the technologies used, the one found most challenging was the programming of LoPy4, which was used to import the data into the system, through the proposed real-time API mechanism. Finally, author hopes that this work developed fulfills its objective and that it can be used by marine biologists and other stakeholders enthusiastic about the topic.

References

1. Dirk A Algera, Jacob W Brownscombe, Kathleen M Gilmour, Michael J Lawrence, Aaron J Zolderdo, and Steven J Cooke. Cortisol treatment affects locomotor activity and swimming behaviour of male smallmouth bass engaged in paternal care: A field study using acceleration biologgers. *Physiology & behavior*, 181:59–68, 2017.
2. Roeland A Bom. *Can speed and tri-axial acceleration measured by biologgers be used to classify oystercatcher behaviour*. PhD thesis, Master's thesis, University of Amsterdam, The Netherlands, 2010.
3. Martin Bor, John Edward Vidler, and Utz Roedig. Lora for the internet of things. 2016.
4. Jacob W Brownscombe, Lee FG Gutowsky, Andy J Danylchuk, and Steven J Cooke. Foraging behaviour and activity of a marine benthivorous fish estimated using tri-axial accelerometer biologgers. *Marine Ecology Progress Series*, 505:241–251, 2014.
5. Francis Jerric Candido, Rojay Flores, and Percival Forcadilla. Haversine method and lora for monitoring entry of fishing vessel in marine protected areas. In *2019 7th International Conference on Information and Communication Technology (ICoICT)*, pages 1–4. IEEE, 2019.
6. Vincenzo Della Mea, Mihai Horia Popescu, Dario Gonano, Tomaž Petaros, Ivo Emili, and Maria Grazia Fattori. A communication infrastructure for the health and social care internet of things: Proof-of-concept study. *JMIR Med Inform*, 8(2):e14583, Feb 2020.
7. Arko Djajadi and Michael Wijanarko. Ambient environmental quality monitoring using iot sensor network. *Internetworking Indonesia Journal*, 8(1):41–47, 2016.
8. R Gagnon, M Sullivan, W Pearson, W Bruce, D Cluett, L Gibling, and LF Li. Development of a marine icing monitoring system. In *Proceedings of the International Conference on Port and Ocean Engineering Under Arctic Conditions*, number POAC09-87, 2009.
9. Robyn S Hetem, Duncan Mitchell, Brenda A De Witt, Linda G Fick, Shane K Maloney, Leith CR Meyer, and Andrea Fuller. Body temperature, activity patterns and hunting in free-living cheetah: biologging reveals new insights. *Integrative zoology*, 14(1):30–47, 2019.
10. Sheng Hu and Jindong Tan. Biologger: A wireless physiological sensing and logging system with applications in poultry science. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4828–4831. IEEE, 2009.
11. Ao Huang, Mengxing Huang, Zhentang Shao, Xu Zhang, Di Wu, and Chunjie Cao. A practical marine wireless sensor network monitoring system based on lora and mqtt. In *2019 IEEE 2nd International Conference on Electronics Technology (ICET)*, pages 330–334. IEEE, 2019.
12. Steven J Johnston, Philip J Basford, Florentin MJ Bulot, Mihaela Apetroaie-Cristea, Natasha HC Easton, Charlie Davenport, Gavin L Foster, Matthew Loxham, Andrew KR Morris, and Simon J Cox. City scale particulate matter monitoring using lorawan based air quality iot devices. *Sensors*, 19(1):209, 2019.
13. Giannis Kazdaridis, Polychronis Symeonidis, Ioannis Zographopoulos, Thanasis Korakis, Katja Klun, and Nives Kovac. On the development of energy-efficient communications for marine monitoring deployments. In *2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*, pages 271–274. IEEE, 2017.
14. Nam Ho Kim. Realtime monitoring system for marine red tide and water-bloom based on internet of things. *Smart Media Journal*, 5(1):130–136, 2016.
15. Jae-Min Kwak, Se-Hoon Kim, and Seong-Real Lee. Design of marine iot wireless network for building fishing gear monitoring system. *The Journal of Advanced Navigation Technology*, 22(2):76–83, 2018.
16. Timothy G Laske, Alina L Evans, Jon M Arnemo, Tinen L Iles, Mark A Ditmer, Ole Fröbert, David L Garshelis, and Paul A Iaizzo. Development and utilization of implantable cardiac monitors in free-

ranging american black and eurasian brown bears: system evolution and lessons learned. *Animal Biotelemetry*, 6(1):13, 2018.

17. Robert J Lennox, Jacob W Brownscombe, Steven J Cooke, and Andy J Danylchuk. Post-release behaviour and survival of recreationally-angled arapaima (arapaima cf. arapaima) assessed with accelerometer biologgers. *Fisheries Research*, 207:197–203, 2018.
18. James R Lewis. The system usability scale: past, present, and future. *International Journal of Human–Computer Interaction*, 34(7):577–590, 2018.
19. Lingling Li, Jiuchun Ren, and Qian Zhu. On the application of lora lpwan technology in sailing monitoring system. In *2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pages 77–80. IEEE, 2017.
20. Afef Mdhaffar, Tarak Chaari, Kaouthar Larbi, Mohamed Jmaiel, and Bernd Freisleben. Iot-based health monitoring via lorawan. In *IEEE EUROCON 2017-17th International Conference on Smart Technologies*, pages 519–524. IEEE, 2017.
21. Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. A comparative study of lpwan technologies for large-scale iot deployment. *ICT express*, 5(1):1–7, 2019.
22. Nageswara Rao Moparthi, Ch Mukesh, and P Vidya Sagar. Water quality monitoring system using iot. In *2018 Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, pages 1–5. IEEE, 2018.
23. Sun Park, JongWon Kim, and ByungRae Cha. Design of marine environment monitoring system based on open source softwares. In *2018 International Conference on Information Networking (ICOIN)*, pages 492–494. IEEE, 2018.
24. AN Prasad, K Al Mamun, FR Islam, and Haq Haqva. Smart water quality monitoring system. In *2015 2nd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*, pages 1–6. IEEE, 2015.
25. Marko Radeta, Nuno J Nunes, Dinarte Vasconcelos, and Valentina Nisi. Poseidon-passive-acoustic ocean sensor for entertainment and interactive data-gathering in opportunistic nautical-activities. In *Proceedings of the 2018 Designing Interactive Systems Conference*, pages 999–1011, 2018.
26. Marko Radeta, Miguel Ribeiro, Dinarte Vasconcelos, Jorge Lopes, Michael Sousa, João Monteiro, and Nuno Jardim Nunes. Seamote-interactive remotely operated apparatus for aquatic expeditions. In *IFIP Conference on Human-Computer Interaction*, pages 237–248. Springer, 2019.
27. Iván Ramírez, Vitor H Paiva, Dilia Menezes, Isamberto Silva, Richard A Phillips, Jaime A Ramos, and Stefan Garthe. Year-round distribution and habitat preferences of the bugio petrel. *Marine Ecology Progress Series*, 476:269–284, 2013.
28. Ramon Sanchez-Iborra, Ignacio G Liaño, Christian Simoes, Elena Couñago, and Antonio F Skarmeta. Tracking and monitoring system based on lora technology for lightweight boats. *Electronics*, 8(1):15, 2019.
29. Sohail Faizan Shaikh and Muhammad M Hussain. Marine iot: non-invasive wearable multisensory platform for oceanic environment monitoring. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 309–312. IEEE, 2019.
30. Carlos A Trasviña-Moreno, Rubén Blasco, Álvaro Marco, Roberto Casas, and Armando Trasviña-Castro. Unmanned aerial vehicle based wireless sensor network for marine-coastal environment monitoring. *Sensors*, 17(3):460, 2017.
31. Matthew Wijers, Paul Trethowan, Andrew Markham, Byron Du Preez, Simon Chamaillé-Jammes, Andrew Loveridge, and David Macdonald. Listening to lions: animal-borne acoustic sensors improve bio-logger calibration and behaviour classification performance. *Frontiers in Ecology and Evolution*, 6:171, 2018.

32. Asif M Yousuf, Edward M Rochester, and Majid Ghaderi. A low-cost lorawan testbed for iot: Implementation and measurements. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 361–366. IEEE, 2018.