

iscte

INSTITUTO
UNIVERSITÁRIO
DE LISBOA

An effective and efficient web platform for monitoring, control, and management of drones supported by a new microservices approach

Jorge Rafael Cunha Santos

Master in Telecommunications and Computer Engineering

Supervisor

Doctor Pedro Joaquim Amaro Sebastião, Assistant Professor

Iscte - Instituto Universitário de Lisboa

December, 2020

Dedico este trabalho à minha família, o pilar da minha vida.

Sem eles nunca teria conseguido atingir este objetivo.

Por ti avó.

Acknowledgment

First of all, I want to express my gratitude to my supervisor Prof. Dr. Pedro Sebastião for the support, orientation and accompaniment given throughout the elaboration of this dissertation.

To the Institute of Telecommunications of Iscte-iul for providing its installations for the development of this project.

I thank my whole family for all the support given over the last few years. Without them I would not have been able to achieve the personal, academic, and professional successes that I now achieve. A special appreciation to my parents, António and Isabel, and to my uncles, Maria João, Carlos and Isabel.

To Carlos Saraiva for all the follow-up given during this dissertation, fundamental for this project to be concluded. For the opportunity given, the support and the motivation, thank you very much.

To my friends for the companionship and friendship that accompanied me on this long journey through the Iscte. Special thanks to Carolina Dionísio, Gonçalo Simões, Rui Passinhas, and Sara Ferreira for giving me fantastic years in this course. Without them, this passage through Iscte would not have had the same brilliance.

Thanks also to João Cardoso, João Antão and João Domingos for their motivation, support and help during these difficult months of confinement that coincided with the writing of this dissertation.

Finally, but most special, to my grandmother Maria Fernanda.

Resumo

Nos últimos anos tem-se assistido a um grande crescimento do uso de drones, sendo utilizados em diversas áreas como a da segurança, da agricultura ou da investigação. A existência de alguns sistemas que permite o controlo de drones à distância é uma realidade, porém, estes sistemas são bastante simples e direcionados a uma funcionalidade específica.

Esta dissertação propõe a elaboração de uma plataforma web feita em Vue.js e Node.js para controlar, gerir e monitorizar drones em tempo real. Usando uma arquitetura de microsserviços, o projeto proposto será capaz de integrar algoritmos que permitem a otimização de processos. A comunicação com os aparelhos remotos é sugerida via HTTP através das redes de 3G, 4G e 5G, e pode ser feita em tempo real ou através de agendamento de rotas. Esta dissertação aborda o caso dos incêndios florestais como um dos serviços que poderia ser incluído num sistema semelhante ao apresentado.

Os resultados obtidos com a elaboração deste projeto foram um sucesso. A comunicação entre a plataforma web com drones permitiu o seu controlo e monitorização à distância. A incorporação do algoritmo de deteção de incêndios na plataforma demonstrou ser possível uma análise em tempo real das imagens captadas pelo drone, sem intervenção humana.

O sistema proposto demonstrou ser uma mais valia ao uso de UAVs na deteção de incêndios. A arquitetura da aplicação desenvolvida permite que outros algoritmos sejam implementados, obtendo uma aplicação mais complexa e com clara expansão.

Palavras-Chave: Controlo de drones, Microsserviços, Node.js, Veículo Aéreo Não Tripulado, Vue.js

Abstract

In recent years there has been a great growth in the use of drones, being used in several areas such as security, agriculture, or research. The existence of some systems that allow the remote control of drones is a reality, however, these systems are quite simple and directed to specific functionality.

This dissertation proposes the development of a web platform made in Vue.js and Node.js to control, manage and monitoring drones in real time. Using a microservice architecture, the proposed project will be able to integrate algorithms that allow the optimization of processes. Communication with remote devices is suggested via HTTP through 3G, 4G, and 5G networks, and can be done in real time or by scheduling routes. This dissertation addresses the case of forest fires as one of the services that could be included in a system similar to the one presented.

The results obtained with the elaboration of this project were a success. The communication between the web platform and drones allowed its remote control and monitoring. The incorporation of the fire detection algorithm in the platform proved possible a real time analysis of the images captured by the drone, without human intervention.

The proposed system has proved to be an asset to the use of drones in fire detection. The architecture of the application developed allows other algorithms to be implemented, obtaining a more complex application with clear expansion.

Keywords: Drone Control, Microservices, Node.js, Unmanned Aerial Vehicles, Vue.js.

Contents

Acknowledgment	iii
Resumo	v
Abstract	vii
List of Figures	xiii
List of Tables	xv
List of Acronyms	xvii
Chapter 1. Introduction	1
1.1. Motivation and Context	1
1.2. Objectives	2
1.3. Structure of the Dissertation	2
1.4. Main Contributions	3
Chapter 2. State of Art	5
2.1. Software Architecture	5
2.1.1. Monolithic Architecture	5
2.1.2. Microservice Architecture	6
2.1.3. Monolithic vs Microservice Architecture	8
2.2. JavaScript	9
2.2.1. TypeScript	9
2.2.2. React	10
2.2.3. Angular	10
2.2.4. Vue.js	10
2.2.5. Angular vs React vs Vue	11
2.2.6. Node.js	12
2.3. Unmanned Aerial Vehicles	12
2.4. Database	13

2.4.1.	Structured Query Language	13
2.4.2.	Not Only SQL (NoSQL)	14
2.4.3.	MySQL / MariaDB	14
2.4.4.	Oracle Database	15
2.4.5.	MongoDB	15
2.4.6.	MySQL vs Oracle DB vs MongoDB	16
2.5.	Tests	16
2.5.1.	Jest	17
2.5.2.	Mocha & Chai	17
2.5.3.	ESLint and TSLint	18
2.6.	GitHub	18
2.7.	Security	18
2.7.1.	SSL Certificate	19
2.7.2.	PassportJS	20
2.7.3.	JSON Web Token	20
2.8.	Mobile Network Technologies	21
2.8.1.	3G, 4G and 5G	21
2.9.	Communication Protocols	22
2.9.1.	Hyper Text Transfer Protocol	22
2.9.2.	Hyper Text Transfer Protocol Secure	22
2.9.3.	Simple Mail Transfer Protocol	23
2.9.4.	MavLink	23
2.10.	Artificial Intelligence	24
2.11.	Machine Learning	24
2.12.	Ardupilot SITL	24
2.13.	Related Work	25
Chapter 3. System Implementation		29
3.1.	Platform Design	31
3.2.	Frontend Architecture in Vue.js	31
3.3.	Backend Architecture in Node.js	37
3.3.1.	Gateway microservice	40
3.3.2.	Notification microservice	41
3.3.3.	Image Processing Microservice	42

3.3.4. Drone Microservice	42
3.4. Database	43
3.4.1. Object Relational Mapper	45
3.5. Tests	46
3.6. Security	47
Chapter 4. Results	51
4.1. Web Platform	51
4.2. Drone Control	55
4.3. Fire Detection	56
Chapter 5. Conclusions	59
5.1. Main Conclusions	59
5.2. Future work	60
References	61

List of Figures

2.1 Monolithic Architecture	6
2.2 Microservice Architecture	7
2.3 Domain Validation (DV) Certificate	19
2.4 Organization Validation (OV) and Extended Validation (EV) Certificate	19
2.5 Without Certificate	19
2.6 JWT Authentication Diagram	20
2.7 MavLink Protocol Message Format	24
2.8 Ardupilot Running	25
2.9 Illustration of the Ardupilot by SITL	26
3.1 System Architecture	30
3.2 Devices List Mockup	31
3.3 Device Mockup	32
3.4 Change and Remove Permissions Mockup	32
3.5 Add Permissions Mockup	33
3.6 Dashboard Organization	33
3.7 Application Programming Interface (API) Code Example to Get Users	34
3.8 Sample Code for use of Components	35
3.9 Vue-router Implementation	36
3.10 Vuex Architecture	37
3.11 Authentication and UserData Organization	38
3.12 Backend Architecture	38
3.13 Node.js Architecture	39
3.14 Node.js Routing	39
3.15 Log File Structure	40

List of Figures

3.16	Gateway Architecture	41
3.17	Notification Architecture	41
3.18	Drone Architecture	43
3.19	Sample Code to Communicate with Drone via MavLink	43
3.20	UserData Relational Diagram	44
3.21	Notification Relational Diagram	44
3.22	Drone Relational Diagram	45
3.23	Registration of an Company using Sequelize	46
3.24	Unit Test of Drone Creation	47
3.25	Structured Query Language (SQL) Injection Scheme	48
3.26	Login Scheme with JSON Web Token (JWT)	49
3.27	JWT Structure	50
4.1	Flydren Website Homepage	51
4.2	Flydren Website Flight Automation Details	52
4.3	Flydren Website Object Recognition Details	52
4.4	Flydren Signup Page	53
4.5	Flydren Login Page	54
4.6	Dashboard Page	54
4.7	Drone List Page	55
4.8	Drone Edit Page	55
4.9	Drone Control Page	56
4.10	Drone Mission Simulation	57
4.11	Fire Algorithm	57

List of Tables

2.1 Basic information of Angular, React and Vue	11
2.2 Angular, React and Vue Statistics by GitHub	11
2.3 Comparison between MySQL, Oracle DB and MongoDB	16

List of Acronyms

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AWS	Amazon Web Services
CA	Certificate Authority
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DAO	Data Access Object
DBMS	Database Management System
DOM	Document Object Model
DTO	Data Transfer Object
DV	Domain Validation
ES5	ECMAScript 5
ES6	ECMAScript 6
EV	Extended Validation
GCS	Ground Control Station
HALE	High Altitude Long Endurance
HD	High Definition
HMAC	Hash-based Message Authentication Code
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IDE	Integrated Development Environment
IMAP	Internet Message Access Protocol
IMT-2000	International Mobile Telecommunications-2000
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token

List of Acronyms

KB	KiloByte
LTE	Long Term Evolution
MALE	Medium Altitude Long Endurance
MAV	Micro UAV
MavLink	Micro Air Vehicle Link
MIME	Multipurpose Internet Mail Extensions
MIMO	Multiple-Input and Multiple-Output
MUAV	Mini UAV
NoSQL	Not Only SQL
ORM	Object-Relational Mapping
OSI	Open System Interconnection
OV	Organization Validation
PL/SQL	Procedural Language/Structured Query Language
POP	Post Office Protocol
RAM	Random Access Memory
RDBM	Relational Database Management System
REST	Representational State Transfer
RFC	Request for Comments
RSA	Rivest-Shamir-Adleman
SITL	Software In The Loop
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SPA	Single Page Application
SQL	Structured Query Language
SSD	Single Shot Detector
SSL	Secure Socket Layer
TDD	Test-Driven Development
TLS	Transport Layer Security
TUAV	Tactical UAV
UAV	Unmanned Aerial Vehicle
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resourcer Locator
UTS	Unix Time Stamp

List of Acronyms

XML Extensible Markup Language

YOLO You Look Only Once

CHAPTER 1

Introduction

1.1. Motivation and Context

The forest area in the world represents approximately 31%, but over the last decades this value has been decreasing successively [1, 2]. The reasons for the losses of forest land are diverse, one of them being the disasters with forest fires. Forest fires are uncontrolled fires that have a very high destructive capacity, capable of destroying thousands of hectares of forest areas, naturally causing the destruction of flora, but also the destruction of infrastructure and, worst of all, the death of animals and humans. The carelessness of the population in the treatment and cleaning of the forests, the criminal hand and the inability of the security forces to control these fires contribute to them becoming real natural disasters when the wind and humidity conditions are favorable [3].

The existence of checkpoints in the forests has allowed fires to be detected more quickly, which allows a faster response to initial outbreaks of fire. The faster fires are fought the less damage occurs. Other detection methods, such as distributed sensors in the middle of the forests or the installation of cameras, theoretically help this detection. However, these efforts are insufficient and, considering the technological advances of recent years, new methods must be thought out and implemented. The use of a robust system that would be able to carry out active surveillance autonomously would be an added value for these situations.

Initially intended for military purposes, in recent years the Unmanned Aerial Vehicle (UAV) have captivated the ordinary citizen and the business sector. It is already possible to use this type of devices to make deliveries, to do private vigilance in properties or just make aerial filming for pure entertainment [4].

The use of these devices can be rewarding in the area of detecting and fighting forest fires. By having the ability to collect data and images in real time, drones can be used for continuous surveillance of forest areas. Through the collected data, which is sent to a server, it is possible to detect risk areas or even to detect fires through sensors and images captured by the drone.

1.2. Objectives

The objective of this dissertation is the development of a web platform for real-time control, management and monitoring of drones. Vaguely, it will be possible for a user to register the drone and be able to control it live via a computer or smartphone. This system would be very useful for situations related to the detection of forest fires, since the range of the drones would only be limited by the range of the mobile network in the territory and by the battery that the device itself possesses, overcoming constraints of this type of situations such as pedestrian access to isolated locations and difficult to reach areas.

Along with the real-time control of the drone, the possibility of scheduling the drone to make a certain route will be implemented. This functionality is very useful and important, taking advantage of the fact that it is not necessary the real time monitoring of a human being for the system to work, allowing these resources to be allocated to other areas.

The use of artificial intelligence would play a fundamental role in this system. The data collected by the drone, especially the images, need to be analyzed. Continuing the projects already developed by the Institute of Telecommunications of Iscte-iul, in Lisbon, Portugal, image processing algorithms have already been developed with the ability to detect fires. The objective of this project is also to include the results obtained from previous investigations carried out on this platform, in order to make it a robust and complete platform.

Although the development of the platform focuses essentially on the improvement of detection systems, its use can be generalized to other situations. A well implemented and thought out system allows the addition of new functionalities or algorithms.

1.3. Structure of the Dissertation

This dissertation is organized in a total of 5 chapters, with the respective descriptions:

- Chapter 1 - Introduction: introduces the context of this dissertation, as well as the objectives intended with this project.
- Chapter 2 - State of the Art: description of technology used or investigated before the implementation of this project and the analysis of related works
- Chapter 3 - System Implementation: description of the system architecture, and its implementation. Presentation of the project initially proposed for this dissertation.

- Chapter 4 - Results: demonstration of the objective results with the system developed
- Chapter 5 - Conclusion: presents the conclusions obtained with this dissertation and improvements that it should have.

1.4. Main Contributions

The work conducted in this dissertation contributed in two ways: one practical and the other scientific. On the practical side, the project developed proved to be able to be a robust but simple platform for managing drones through the internet. Its applications are diverse, including fire detection through the implemented algorithm. Its use can be by individual users or by companies.

This project allowed scientific contributions related to the development of platforms based on a microservice architecture for control, management, and monitoring of drones. The publications are:

- J. R. Santos, P. Sebastião, “An Effective and Efficient Web Platform for Monitoring, Control, and Management of Drones Supported by a Microservices Approach” in ICAST 2021: International Conference on Applied Science and Technology.
- J. R. Santos, P. Sebastião, “A Microservice Platform for Real-time Control and Monitoring of Drones” in Applied Sciences

CHAPTER 2

State of Art

In this chapter will be presented the various architectures, technologies, frameworks, libraries and methodologies studied for the development of the project proposed by this dissertation. Projects similar to the one proposed by this document will also be analyzed, presenting the advantages that the project presented here has.

2.1. Software Architecture

Two software architectures will be presented next: the monolithic architecture and the microservices architecture. The structure of each of these architectures will be illustrated, describing the advantages and disadvantages, culminating in a comparison between the two studied architectures.

2.1.1. Monolithic Architecture

The monolithic architecture is considering the standard way of developing an application. This approach consists of encapsulating all system services in the same basic code, for example the HTML and JavaScript pages that the user interacts, API, database access, system logic and request execution [5].

At the beginning of a project, the handling of this type of applications is quite simple and accessible: it has only one basic code which facilitates its initial development and scaling, and its level of complexity is relatively low [6]. It is possible to use only one database to handle all information, making transactions easier to manage. Figure 2.1 shows the example of a monolithic architecture.

For large-scale projects, this type of architecture can become a problem. Due to factors such as the large amount of code and high coupling, developing new features or fixing bugs can result in a high effort cost for development teams [6]. The simple implementation of a feature, which at first seems simple, can in fact affect multiple project files and components, making this task more complex and critical [5].

Continuous deployment is also affected in large projects that use this architecture. The construction of an artifact in this type of applications encompasses all the project's

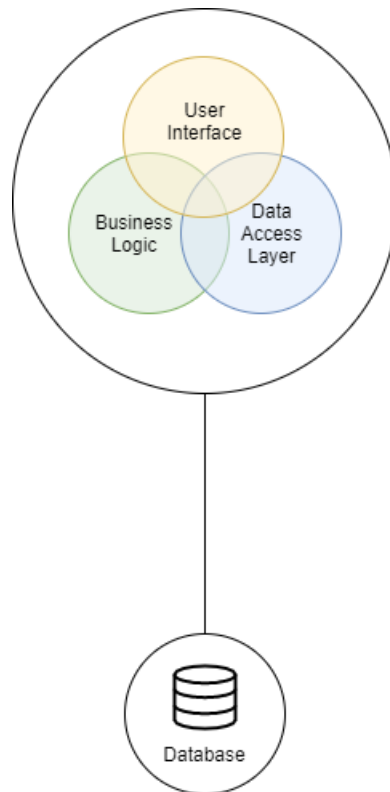


FIGURE 2.1. Monolithic Architecture

source code, not just a specific module. Thus, the passage to production of a new functionality has to be done with the project as a whole. This situation implies that the whole development team has to be in sync to avoid delays in the development of the application [7, 8].

On the other hand, the implementation of a monolithic project in the various development environments, such as the test environment or production environment, is simpler. Since only one artifact is used, its passage through the various stages of development is done in a continuous manner [5]. The development of integration tests in this type of projects is much easier compared to the microservice system, since all the project code is located in the same place [7].

2.1.2. Microservice Architecture

The architecture of microservices has been gaining space over the last years. An application based on this type of architecture is divided into several services, called microservices. The great advantage of this architecture is due to the easy scalability, project structure perception and the clean and readable code. The origin of this architecture is based on the software design called Service-Oriented Architecture (SOA) [9].

A microservice should only focus on a set of similar features and should be independent from other microservices. This is classified as a good modularity in code. Modularity means *"granting the application components separately and independently"*, thus allowing the development and evolution of each microservice not to be dependent of external factors. Figure 2.2 shows the example of a microservice architecture [8, 10].

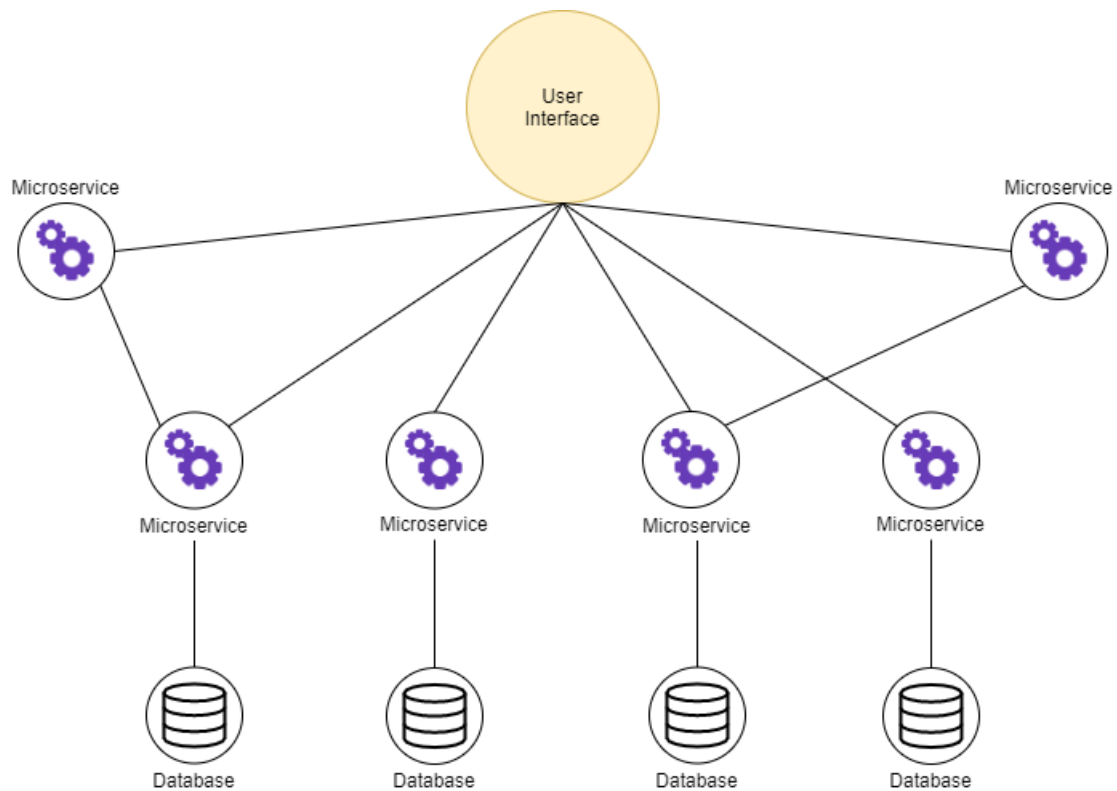


FIGURE 2.2. Microservice Architecture

The passage from a microservice based application to various development environments does not necessarily have to be done with all microservices at the same time. Each microservice generates an artifact that will be implemented independently [5, 9, 10]. This situation, on the other hand, has a negative side, as it is necessary to work with several artifacts, the time spent in the development of versions is quite high [11].

The communication between these microservices is well delineated because it is done through lightweight mechanisms, known as APIs, and there is no risk of a microservice executing internal calls from another microservice. This way, modularity is guaranteed in this architecture. However, a large number of communications between the microservices in a single request implies an increase in response time because APIs have limitations in number and response time [8].

2.1.3. Monolithic vs Microservice Architecture

Over the past few years many companies have migrated their applications from a monolithic architecture to a microservice architecture. Although the development of an application with this architecture is more expensive in the beginning, in the long term the advantages are evident. The main advantages and disadvantages of the two architectures studied are presented below [5, 6, 8, 9, 10].

Monolithic advantages

- Easy development.
- Easy to test, since the whole application integrates a single package.
- Simplicity and speed in deploying, since there is only one package.
- Technological knowledge is equitable, since all developers work on the same application and technologies.

Monolithic disadvantages

- In large and complex projects, the implementation of new features will take time due to high coupling.
- For a minimal change it is necessary to deploy the whole application.
- Introduction or evolution of technologies is complicated and time consuming.
- High CPU utilization.

Microservice advantages

- Low coupling.
- Problems in one of the services do not directly affect the others.
- Fast initialization of services.
- Changes to the code do not imply the deployment of the entire application, only the changed microservice.

Microservice disadvantages

- The structure of the application is more complex.
- End-to-end and integration tests are more difficult to perform.
- A larger development team is needed because each service can have its own technology.
- Initial application development is slow.

2.2. JavaScript

JavaScript is one of the most popular programming languages in the world. In the beginning it expressed itself in small segments of code to make web pages more dynamic and attractive to users, libraries and frameworks began to appear and changed the paradigm of this language [12], [13].

A framework is a set of libraries and tools that allow you to solve recurring problems, allowing the developer to focus on solving the actual problem and not rewriting code.

In 2010, with the development of AngularJS by Google, JavaScript frameworks began to stand out among the other technologies, opening space for other frameworks of the same kind to grow [14].

Most of these frameworks are made up of components. These components have intrinsically at their root the ability to communicate with each other, to store information and are encompassed within the flow of control and processes. The main difference between a library or a framework is what each one provides to the developer. While a library provides only one set of functions and methods, a framework offers a pre-implemented architecture, a set of processes and flows of an already prefabricated system, facilitating communication between the various modules/layers that the project itself needs [14, 15, 16].

2.2.1. TypeScript

TypeScript language is a super-set of JavaScript that includes a huge amount of features that are natively not available or that require a great implementation effort [17].

Created by Microsoft, this language better supports the use of Object-Oriented Programming, which includes the principles of inheritance, polymorphism, encapsulation and abstraction. In addition, it corrects a syntax flaw in JavaScript related to data types. While in JavaScript the creation of variables does not require the indication of the variable type, in TypeScript besides using the same syntax as JavaScript, it requires the developer to indicate the variable type. This improvement allows Integrated Development Environment (IDE)s to provide a richer environment for error detection and, consequently, enables better code interpretation by other users [17, 18].

TypeScript is an open-source project and is based on the ECMAScript 6 (ES6) standards, which allows compilation into JavaScript. Thus, development in TypeScript is done in .ts files that can be compiled to JavaScript, since browsers cannot process TypeScript directly [17].

2.2.2. React

React is a robust JavaScript library designed for the development of interactive interfaces without web pages. This library is one of the most used in the world, having a very simple learning process requiring only basic knowledge of HyperText Markup Language (HTML) and Cascading Style Sheets (CSS) to understand it. As it is a very complete library, it is commonly inserted in the context of frameworks [15, 19].

React, like other frameworks, is composed by components that, interlinked with each other, originate a cohesive and simplistic flow. It has a Data Binding feature, using an architecture called Flux, in which a control point makes the data management between and for the components.

Another feature highly valued in React is the concept of Virtual Document Object Model (DOM), a tree structure composed of simple objects, enhancing the performance of the application. When building a Virtual DOM, all objects in this structure are compared with those already existing in the browser, updating the components where there was a change, making the process effective and fast [15, 16, 19].

2.2.3. Angular

Angular is a JavaScript framework launched in September 2016, which allows web and mobile development. It was developed by Google, succeeding the AngularJS framework. Also known as Angular 2, this framework allows code writing in ECMAScript 5 (ES5) JavaScript, ES6 JavaScript, Typescript and Dart [14, 17].

Working with very recent languages and libraries, the browsers still have limitations, which involves some effort of Angular to ensure compatibility with older versions of browsers. This compatibility is guaranteed through transpilers that the framework itself already has properly implemented. On the other hand, there is a great effort in the community to adopt the ES6 standard, which allows the abolition of transpilers over time [14, 16, 20].

2.2.4. Vue.js

Vue.js, or just Vue, is a very recent progressive framework, compared to the others already compared, and is very versatile, easy to use and with a high performance. The development of this framework was thought to contain the most valued features of the most popular frameworks. Thus, Vue includes React's Virtual DOM and Angular's template construction [16, 21, 22].

Vue is essentially focused on the visual layer and allows an easy integration with other libraries in order to obtain the desired result. It also allows the development of Single-Pages Applications together with other libraries and dedicated tools [21].

2.2.5. Angular vs React vs Vue

Until a few years ago, the choice of which JavaScript framework to use was between Angular or React. But over the last few years, there has been a great growth in the use of another framework that has shuffled the accounts, the Vue.js.

The development community has a great impact on technological trends. Therefore, it is possible to obtain a perspective of the growth of these technologies using statistical data provided by platforms such as GitHub, the largest repository of projects in the world. Table 2.1 compares some of the characteristics between these JavaScript frameworks [16].

TABLE 2.1. Basic information of Angular, React and Vue

	Angular	React	Vue
Release date	2010	2013	2014
Approximately size (KiloByte (KB))	500	100	80
Stable version (Nov 2020)	10.2	16.6	2.6
Used by	Google, Wix	Facebook, Uber	GitLab, Alibaba

Analyzing the Table 2.2, it is possible to conclude that the interest between React and Vue are similar and high, comparing with the number of Angular observers. The popularity of Vue is also proven with the number of stars given to this project. It can also be seen that the contribution that Vue has is higher compared to Angular and React. The justification for this great difference is related to the companies that support each of the technologies: Angular and React have Google and Facebook, respectively, while Vue is supported by the open source community that has a huge number of users [23].

TABLE 2.2. Angular, React and Vue Statistics by GitHub

	Angular	React	Vue
GitHub #Observers	3.2k	6.7k	6.2k
GitHub #Stars	62k	151k	166k
GitHub #Contributors	1k	1.3k	300k

The choice between one of these frameworks is not consensual, each having pros and cons depending on the objective:

- Angular is the oldest framework, therefore the most mature among the three, having already a high amount of tested and operational tools. It is especially

indicated for large projects and with a team that knows TypeScript. However, the learning of this framework is large and complex.

- React is a framework widely accepted by the community and can be easily incorporated into projects already built. Because it allows some flexibility, it is widely used in small projects or start-ups.
- Vue, despite being the latest framework among these three, has a very flexible architecture and easy integration. It offers a great customization, having an overlapping of its functionalities like Angular and React, making a later transition very accessible.

2.2.6. Node.js

Node.js, or simply Node, is a JavaScript implementation developed to run in a server-side environment, so it is not browser dependent, unlike traditional JavaScript implementations. This type of implementation emerged with the growing need for server-side applications in JavaScript [24, 25].

Unlike other modern frameworks, Node processes do not rely on multi threading to support the simultaneous execution of logical processes, based on asynchronous and non-blocking input and output events. Node runs on a Google JavaScript V8 machine, which consists in transforming JavaScript code into another faster and more perceptible programming language by the computer [24, 25].

2.3. Unmanned Aerial Vehicles

UAV are flying devices that do not require a human on board to carry out flight plans. The drones can be controlled remotely by a human, through radio communication, or autonomously, using an embedded system and a set of sensors that allows you to make flight plans [26, 27].

Due to the wide variety of existing UAVs, the author in [28] distributes them into different categories, depending on their range, autonomy and achievable altitude:

- High Altitude Long Endurance (HALE) are aircraft that have a high autonomy of flight, which can exceed 24 hours. They can cover more than 5000km of distance, reach a maximum altitude of 15km and their size can go up to 35m. These UAVs are used in military missions, such as remote surveillance and air attacks.

- Medium Altitude Long Endurance (MALE) are drones with functions which are similar to HALE but have only a range of 500km and reach a maximum altitude of 5km. Their size can be up to 20m.
- Tactical UAV (TUAV) are drones like MALE and HALE with a range between 100 and 300km. Their size is very small (not exceeding 10m) and they have a maximum autonomy of 10h.
- Close-Range UAV are relatively small and very common devices in the military and civil area, used for different purposes. It can travel a distance of 100km and has an autonomy of 2 to 4 hours. Its size varies between 1 and 6 meters.
- Mini UAV (MUAV) are drones with a weight of under 10kg. They have an autonomy of only 2 hours and reach a maximum altitude between 150 and 300m. Their size does not normally exceed 3 meters.
- The Micro UAV (MAV), together with the MUAVs, are the most popular types of drones. Due to their small size that does not exceed 1 meter, their acquisition is quite affordable. The flight autonomy of an MAV is less than one hour and can reach a height of 250m. Its range does not exceed 20km.

This dissertation will focus mainly on MAV because they are the most common types of drones.

2.4. Database

2.4.1. Structured Query Language

SQL is the standard language used in Relational Database Management System (RDBM) relational database. This type of database has as main objective to relate the information stored in its tables to each other. It also allows a controlled management of data and system flow.

An SQL database is essentially built by tables, which in turn may have several rows or entries. Each row of a table is a distinct entity, and each column of the table is an attribute of that entity [29, 30].

SQL databases must respect the concept of ACID, which includes the integrity of the information and the system, the validation of transactions, the synchronization of all tables and the persistence of data after technical failures. The acronym ACID comes from the following concepts [29]:

- Atomicity: In which each transaction must be reversible if the action is not successful.

- Consistency: The stored data must respect the various rules defined.
- Isolation: The transactions are executed sequentially, which prevents a later action from catching an intermediate state existing in the previous multiple action.
- Durability: In case of system failure, if an action was successfully executed it must persist in the database.

2.4.2. Not Only SQL (NoSQL)

Databases in Not Only SQL (NoSQL), unlike the SQL database, are based on key value pairs, documents or even graph data. Another big difference is in the database schema, being a dynamic and versatile schema for when the data is not structural.

Because it is not a relational database, the construction of a NoSQL database is less demanding and has greater flexibility. It is possible to create documents without first defining a structure and each document can be its own structure [30, 31].

The scalability of the system also differs, being NoSQL scalable horizontally, allowing a large set of data and information is not a high effort for the Central Processing Unit (CPU) or Random Access Memory (RAM) of the system.

Most NoSQL systems do not use ACID transactions. To obtain another level of benefits such as scalability and resiliency, they use the BASE model in their systems [31]:

- Basically Available: The database has a very high availability.
- Soft-state: Even without any input during a period, the state of the system can change, very much a result of 'Eventually consistent'.
- Eventually consistent: The system guarantees that after a change in an object, all access requests will return the last updated value. This is if a failure occurs and taking into account the maximum size of the inconsistency window, which may vary depending on the system load [32].

2.4.3. MySQL / MariaDB

MySQL is one of the most popular relational database management systems as it is open source and free of charge for most of its functionalities. This service uses SQL as its language. This system was created in 1995 by David Axmark, Allal Larsson and Michael Widenius and was sold to Sun Microsystems in 2008 under the company MySQL AB. It is currently under Oracle domain since 2009 [33, 34].

Because it is a system with low hardware requirements, easy to implement and with a very good performance, it is the most used SQL system in the world by development fans

and small businesses. Other highly valued features of MySQL are the great compatibility with various drivers and the ability to be used in any platform [29, 30, 33].

After Oracle's acquisition of MySQL, the community version became more restricted, so one of the founders of MySQL, Michael Widenius, created the MariaDB project. MariaDB is a MySQL fork, i.e. it started as an independent project based on the MySQL source code, allowing the community to continue to use MySQL unrestricted and for free as well as to make contributions to the sustainability of the project. MariaDB keeps its version up to date with MySQL, being its compatibility 100% assured [29, 30, 34].

2.4.4. Oracle Database

Oracle Database is also one of the most popular database management systems, Database Management System (DBMS). The architecture of this database has a great optimization and performance, whatever the size of data in question.

Although it is an expensive DBMS, the confidence and security it has demonstrated justify the investment made to implement this service in a company. Being a multi-model database, it allows not only the creation of relational models, but also of documents, graphs and key values, some very famous features in NoSQL [35, 36].

Besides the simple database service, Oracle provides other services that allow completing and adding many compatible services such as Oracle Streams, Oracle Real Application Clusters. Oracle's Procedural Language/Structured Query Language (PL/SQL) extension integrates many other features that are valued in the developer community [35].

Oracle Database is the leading service in the market in this area, especially in the business environment, since this sector can support the costs.

2.4.5. MongoDB

MongoDB is a document-oriented database turning it into a NoSQL database and, as it is open-source, it is one of the most used nowadays. Because it is not a relational database, the insertion of a document does not imply the prior construction of any structure. Each document can be independent of everything [30, 31].

A document in a MongoDB database is in JavaScript Object Notation (JSON) format, allowing the document to have simpler values such as dates, numbers, or string, as well as more complex information such as lists of objects or pairs of key values.

MongoDB allows documents to be grouped into collections. It is this set of collections that forms the database. This information can be replicated to other servers or even be shared, through the sharding functionality that MongoDB integrates [31].

2.4.6. MySQL vs Oracle DB vs MongoDB

Table 2.3 presents the main differences between the analyzed databases. Comparing the databases portrayed here, we can see that there is a big difference between MongoDB, because it is non-relational, and MySQL and Oracle, which are both relational databases. The choice of the database for this project is based mainly on the most appropriate type of database. When seeking a more traditional database, where there is a relationship between the stored data, MongoDB excludes itself from the equation [37].

TABLE 2.3. Comparison between MySQL, Oracle DB and MongoDB

	MySQL	OracleDB	MongoDB
Primary database model	Relational DBMS	Relational DBMS	Document store
Secondary database model	Document store	Document store Graph DBMS	Search engine
Ranking	#2 Overall #2 Relational DBMS	#1 Overall #1 Relational DBMS	#5 Overall #1 Document store
Stable version (Nov 2020)	8.0	19c	4.4
License	Open source (MariaDB)	comercial	Open source
Data scheme	Yes	Yes	Schema-free

Following, and as previously described, the choice between the MySQL database and OracleDB fell on the MySQL database, more specifically in the open-source version MariaDB. The bet on MariaDB came about because it is a free version and, as the project presented in this dissertation is at an early stage and with little information flow, there is no need to invest in a more robust database like Oracle.

2.5. Tests

Tests in software development are an important factor in ensuring the quality control of the system. These have as main objective to guarantee that the requirements of the system are well implemented.

The use of tests allows for cost savings, whether monetary costs or time costs, because they allow the detection of errors or bugs much faster. A defect discovered in the production phase of a software has a much higher cost of correction when compared to discovering the same defect while in a test phase [38].

The use of testing in software development is seen as a quality standard by the technological community. These tests should include the various existing variants, being the most used [38, 39]:

- Black box testing: validating the functional requirements of the system. The code is not observed nor how it was implemented, focusing only on the results obtained by the various processes.
- Regression test: consists of testing each version, allowing the detection of errors resulting from the software version change.
- Unit tests: checks if smaller parts of the code, usually methods or a small set of methods, work properly. It is one of the most important tests when developing software.
- Integration tests: has as main objective to test the joint integrity of several parts of the code. Like unit tests, integration tests have a great importance in development.

2.5.1. Jest

Jest is a JavaScript framework for testing. Created by Facebook, it's a very complete and fast framework and doesn't need a complex configuration to use it. It was developed from the Jasmine framework.

It was initially developed with the purpose of testing the React framework, also developed by Facebook. However, its implementation has become quite complete and wide resulting in it being used in other platforms such as JavaScript, Node.js or Typescript [40].

2.5.2. Mocha & Chai

Mocha, as announced by the website itself, is a JavaScript framework targeted to Node applications. This framework allows testing in a simple and easy way, even asynchronous tests [41].

The Mocha project is an open-source project, allowing any developer to contribute to its growth. Its rich documentation and easy understanding make it one of the main testing frameworks used in Node.js [41].

When using Mocha in a project, the integration of Chai is recommended [42]. Chai is an assertion library that complements Mocha itself, giving the developer more options

for testing, making them more cohesive and effective [41, 42]. It also allows the implementation of HyperText Transfer Protocol (HTTP) order testing, very useful for projects that contain APIs [42]).

2.5.3. ESLint and TSLint

When developing a project, especially a large one, it is necessary to ensure the quality of the code developed. For this, there are several code quality tools on the market, which help you not only to correct errors, but also to prevent duplicate and confusing code. Two of these best-known tools are ESLint and TSLint.

ESLint is a lint tool for JavaScript that allows you to generate reports of good and bad practices existing in the project. You can configure ESLint to respect the JavaScript standards that exist in the community. ESLint is the most accepted open-source tool used by the community [43, 44].

TSLint is also a code quality tool and has features quite similar to ESLint [43]. The main difference is in the target language, Typescript. In order to respond to the calls made by the community, the developers of TSLint and ESLint decided to unify these tools. Thus, TSLint was deprecated in 2019 and ESLint started the *eslint-typescript* project with the support of the founder of TSLint [45].

2.6. GitHub

Git is a version control system created by Linus Torvalds, creator also of Linux [46]. A version control system is an essential tool for any project. It allows you to track changes made to the code, recording who changed it and, if necessary, reversing it [47].

These systems are highly recommended in medium and large projects, especially when there are continuous releases of versions, code corrections or implementation of new features [46].

GitHub is a company that provides the Git cloud service. The simple interface provides users, free access to numerous control and collaboration tools make it the most widely used Git service in the world [48].

2.7. Security

Security in any system should be considered as a priority to be implemented. In online systems, where anyone through the Internet can have access to the content, it is more important that the security implemented allows to guarantee the integrity, privacy and correct functioning of the whole system [49].

The existing servers on the Internet, which allow anyone to host a website, already have a robust security. However, certain aspects should be implemented on the website side and must be of concern to any web developer. In the following, some protocols or libraries will be presented that allow a better security in the communication and integrity of the whole system [49, 50, 51].

2.7.1. SSL Certificate

Secure Socket Layer (SSL) Certificates, which is issued by the Certificate Authority (CA), allows all communications with this site to be transferred through the HyperText Transfer Protocol Secure (HTTPS) protocol. SSL certificates can be of three types depending on the desired security level [52, 53]:

- DV Certificate: a certificate that can be obtained easily and at a very accessible price. It is directed to a domain or sub-domain.
- OV Certificate: it is also a certificate directed to a domain or sub-domain, however, its verification requires a higher level of security, essentially business.
- EV Certificate: is the certificate that guarantees the highest level of security and has a validity of only 2 to 7 days. It is aimed at businesses that require extreme confidentiality, such as bank access.

The images illustrated in figures 2.3, 2.4 and 2.5 show how the various types of SSL certificates are represented in the browser.

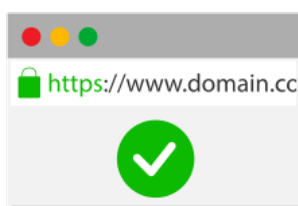


FIGURE 2.3. DV Certificate [54]

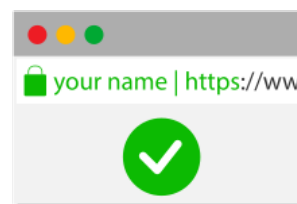


FIGURE 2.4. OV and EV Certificate [54]

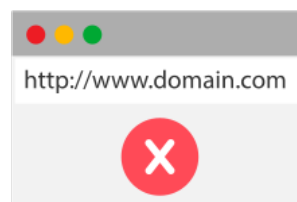


FIGURE 2.5. Without Certificate [54]

2.7.2. PassportJS

PassportJS is an authentication middleware for Nodes.js. Its sole purpose is to guarantee the authenticity of received requests through an easy and fast way to use. Through this implementation it is possible to configure a user's session, authentication strategy and configure the various authentication routes [55, 56].

PassportJS allows authentication to be done using the traditional method, using an email or username and the respective password. This information must be stored by the application in a database. It also allows integration with other more recent authentication mechanisms, OAuth. OAuth is a service that allows users to register and authenticate in an application using an external/third party application. This implementation is used recurrently through authentication using platforms like Google, Facebook or LinkedIn [55, 56].

2.7.3. JSON Web Token

The JWT is an Request for Comments (RFC) 7519, a security mechanism to realize authenticity between two parties through the use of a signed token [57]. This token is digitally signed using an Hash-based Message Authentication Code (HMAC) algorithm or using public and private Rivest-Shamir-Adleman (RSA) keys.

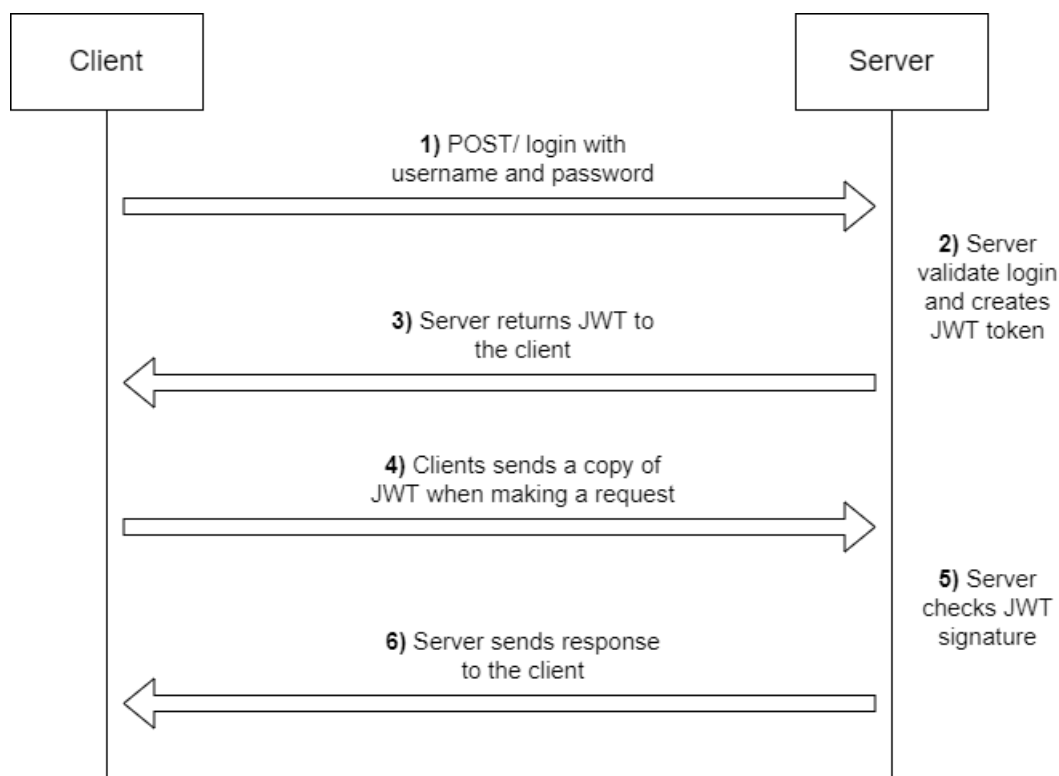


FIGURE 2.6. JWT Authentication Diagram

The operation mode using this mechanism is quite simple, as can be seen in the Figure 2.6. The server, after receiving the authentication request and subsequent validation, creates a token signed by the user that is sent to the user. From this moment on, whenever the user makes requests to the server, the latter must include the token in the message sent. This way, the server is able to verify the authenticity of the request received, ensuring that non-authenticated users cannot access restricted content [57, 58, 59].

2.8. Mobile Network Technologies

The technology used in mobile communications ceased many years ago to have exclusive use in the mobile market. The evolution of the technology, and the growing need, has allowed a high range of other products to incorporate this technology. We can characterize that a mobile communication technology is any wireless communication between two devices, which can be, among others, cell phones, computers, transmission towers or radios.

2.8.1. 3G, 4G and 5G

3G technology was the first great leap in mobile communication. With the global standard agreed at International Mobile Telecommunications-2000 (IMT-2000), the use of the Internet in cell phones became a daily act of the user. The mobile system compatible with the IMT-2000 standard is called Universal Mobile Telecommunications System (UMTS). The most used frequency bands are 900MHz and 2100MHz in Europe, Asia, Africa and Oceania and 850MHz and 1900MHz on the American continent [60].

The 4G/Long Term Evolution (LTE) technology has allowed communication and information exchange to take place with higher bandwidth and lower latency. This technology has revolutionized several digital sectors such as the use of audiovisual content, streaming and video calls. Although this technology has existed for many years, its coverage especially in rural areas is not guaranteed. The frequencies used by the 5G network are mainly 800MHz, 1800MHz and 2600MHz in EuroAsia and 700MHz, 1700MHz and 2300MHz in America [60].

The 5G technology under implementation presents a new digital technology, the Massive Multiple-Input and Multiple-Output (MIMO). With 5G, the data exchange speed will be much faster, with less latency and congestion [61]. The 5G will be the open door for the great growth of Internet of Things (IoT) in recent years, allowing high numbers of objects to be connected to each other without disturbances in the network. In contrast,

the range of an antenna will be smaller. This technology brings two frequency ranges: 450MHz to 6GHz and 24GHz to 52GHz [60, 61].

2.9. Communication Protocols

2.9.1. Hyper Text Transfer Protocol

HTTP is the protocol used by browsers to exchange information between a user and a server. This system is the basis of any communication on the Internet, which allows content to be accessible through an address, commonly called a Uniform Resourcer Locator (URL) [62].

All servers that host websites have designed in their programs the ability to receive HTTP requests, while browsers, the HTTP clients, can make these requests and consequently process the content received.

The protocol arose with the need to transfer information between a client and a server, so that there was a standardization in the distribution in this way, the first version of this protocol appeared in 1990, called HTTP/0.9. This first version had only one request method, the GET. Over the years other methods were incorporated and additional functionalities were implemented such as connection control, cache management and even articulation with other communication protocols used in the World Wide Web [63, 64]. Currently the most used version is HTTP/2.0, and the migration to version 3.0 is already happening [62, 65].

2.9.2. Hyper Text Transfer Protocol Secure

An existing problem with the transfer of information between the client and the server, or even between servers, is the necessity of ensuring that no content in the messages is viewed or changed. In this sense, there was the need to ensure that the exchange of information was [66]:

- private, the content should not be viewed by third parties except the recipient.
- integral, the content should not be altered or manipulated.

HTTPS is not exactly a new protocol, but the use of HTTP protocol over an encrypted SSL/Transport Layer Security (TLS) connection layer. In packets transmitted over the network, all content is encrypted, including the header and requests, and is only decrypted when the message reaches its destination [66, 67].

A connection between the client and the server, using HTTPS, only happens if the accessed website has an SSL Certificate. Browsers illustrate if the communication between

the browser and the server is secure through the https terms in the site's own URL or through an image of a lock [67].

2.9.3. Simple Mail Transfer Protocol

Simple Mail Transfer Protocol (SMTP) is a convention dedicated exclusively to sending emails over the Internet. Because it only refers to sending emails, it is usually used in conjunction with other protocols, such as Post Office Protocol (POP) or Internet Message Access Protocol (IMAP), when you want to have a sender-receiver system. In this case, the sender implements the SMTP protocol and the receiver one of the other protocols [68].

One of the gaps in this protocol is the inability to transfer files, such as images or videos, and the limitation of supporting only 7-bit American Standard Code for Information Interchange (ASCII) characters. In order to combat this problem, the Multipurpose Internet Mail Extensions (MIME) was developed. Thus, the vast majority of email services exchange messages using the SMTP protocol with the MIME format. Because there is such a close relationship between these two protocols, they are commonly called SMTP/MIME [68, 69].

2.9.4. MavLink

Micro Air Vehicle Link (MavLink) Communication Protocol is a message protocol for communication with drones, or between the components of a drone. This protocol allows two-way communication between the drone and the Ground Control Station (GCS). The control station sends commands to the drone, while the drone responds with measured data at least, such as location or images [70, 71]. It was initially released in 2009 by Lorenz Meier.

The MavLink protocol is a layer 2 protocol, in reference to the Open System Interconnection (OSI) layers, which is the data layer. It is responsible for starting the connection, indicating the logic for data transfer, and interrupting the connection [2, 71, 72]. The messages contained in this protocol are defined in Extensible Markup Language (XML) files and can be implemented in one of the programming languages supported by MavLink, some of them being C, C#, Java or Python.

All MavLink messages have an identifier, which relates the source station and the device. Because there is no guarantee that messages are delivered, frequent communication is required to ensure that the content has been delivered correctly. An advantage of this

protocol is that the messages are already encrypted. The size of these messages can vary between 8 Byte and 263 Byte, depending on whether the protocol is the first or second version of MavLink. In the Figure 2.7 it is possible to observe the structure of a MavLink message [70, 2].

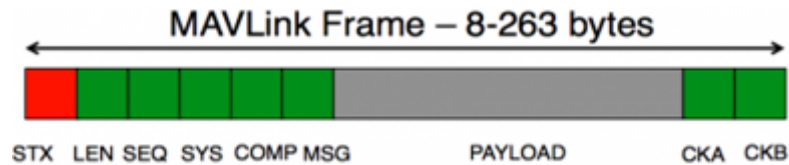


FIGURE 2.7. MavLink Protocol Message Format [70]

2.10. Artificial Intelligence

Artificial Intelligence is a system that can evolve through the acquisition of knowledge. Such a system is intended to imitate human intelligence, through behaviors such as reasoning, judgment, identification, perception, understanding, thinking, learning, problem solving, etc...

Artificial Intelligence can be used in countless areas and needs people with knowledge not only in computing, but also in philosophy, psychology or science, depending on the purpose of the machine created. The main focus of the last years has been essentially in the field of intelligence, education, process automation and the development of intelligent decision support systems [2, 73].

2.11. Machine Learning

Machine Learning is a field of artificial intelligence that allows applications to predict data with a high accuracy. As a rule, this prediction is possible with the learning of previous data, allowing the application to identify patterns and, consequently, make decisions autonomously and with minimal human intervention [2, 63].

The more data is provided to Machine Learning, the more accurate will be the conclusions presented. This type of applications are used in several areas such as security, privacy and simplicity of tasks.

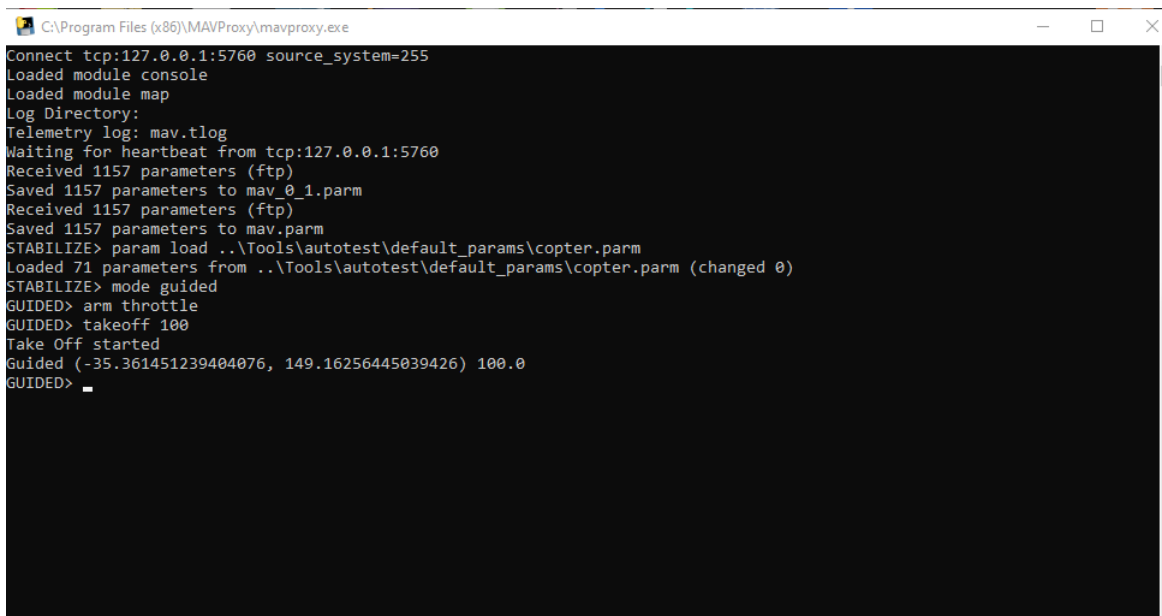
2.12. Ardupilot SITL

The Ardupilot is an open-source software of autopilot, i.e., intends to represent the control of a vehicle automatically. It can control almost all vehicles: airplanes, helicopters, cars, boats, drones, etc.... It is used mainly in research or in the simulation of prototypes

in other projects. Being an open-source project, the contribution given by the community is quite high, which allows a rapid and current evolution [74, 75, 76].

The Software In The Loop (SITL) Simulator allows you to run ArduPilot on your computer without using any special hardware. This simulator provides the user with a graphical interface that allows a better interpretation of the data collected by Ardupilot [77].

Figure 2.8 shows the Ardupilot running on a computer console, while in figure 2.9 the SITL simulator represents the Ardupilot through a graphical interface using a C++ compiler.



```

C:\Program Files (x86)\MAVProxy\mavproxy.exe
Connect tcp:127.0.0.1:5760 source_system=255
Loaded module console
Loaded module map
Log Directory:
Telemetry log: mav.tlog
Waiting for heartbeat from tcp:127.0.0.1:5760
Received 1157 parameters (ftp)
Saved 1157 parameters to mav_0_1.parm
Received 1157 parameters (ftp)
Saved 1157 parameters to mav.parm
STABILIZE> param load ..\Tools\autotest\default_params\copter.parm
Loaded 71 parameters from ..\Tools\autotest\default_params\copter.parm (changed 0)
STABILIZE> mode guided
GUIDED> arm throttle
GUIDED> takeoff 100
Take Off started
Guided (-35.361451239404076, 149.16256445039426) 100.0
GUIDED>

```

FIGURE 2.8. Ardupilot Running

2.13. Related Work

Over the last few years several systems for fire detection have been developed. In [78] three fire detection systems are demonstrated through image processing. Each system presented corresponds to a different type of image detection, and the one that obtained the best result in fire detection was the Faster R-CNN. Faster R-CNN is an object detection architecture developed by Ross Girshick, Shaoqing Ren, Kaiming He and Jian Sun in 2015. It is one of the most famous neural networks of convolution. The other architectures used were You Look Only Once (YOLO) and Single Shot Detector (SSD). The results obtained were positive. However, the system presented in this dissertation has the advantage that the image processing algorithm is in a cloud and can be accessed and used at any time.



FIGURE 2.9. Illustration of the Ardupilot by SITL

In [79] the authors approach one of the solutions used in fire detection and fighting: the use of artificial satellites. The study concludes that the development of satellites exclusively dedicated to image processing is too expensive. Therefore, the ideal would be the use of aerial devices that would be more economical and obtain the same result.

Fire detection should not be restricted to fire analysis only. In [80], the authors presented a proposal for a system to detect smoke. However, as concluded in the article, the use of only this system does not guarantee correct fire detection. Environmental pollution, especially in urban areas, makes the mission of these algorithms difficult. On the other hand, the development of a system that detects smoke and fire at the same time will increase the effectiveness in detection, reducing false alarms.

In [81], a proposal was made for a drone control system through a IoT platform developed on a Raspberry Pi. The communication between the platform and the drone is through internet, more specifically through Wi-Fi. The project presented in this dissertation will greatly enhance the idea sought by the authors in [81], transforming a basic and simple system, a complex platform, in a cloud, with the ability to manage users and control any device, provided it is connected to the Internet.

There are already some platforms that allow the control of UAVs in real time, such as <https://flytnow.com/>. This platform allows a user to associate a drone to his account and to control it through the internet. The functionalities of this system include the

Chapter 2 *State of Art*

communication with the drone through 4G or 5G and High Definition (HD) streaming. Being a platform directed to companies, the costs of membership are high, keeping away any normal user.

The objective of this dissertation is not only to develop and implement a control, management, and monitoring platform for drones, but also to have the ability to detect forest fires remotely. As there is no platform that offers this solution, the idea presented is innovative.

CHAPTER 3

System Implementation

For the implementation of this system, a microservice architecture was used which, among all the advantages already described in section 2.1, allows an independent development of each microservice. Seven microservices were developed according to the theme each one covered: Landing Page, Dashboard, Gateway, Auth and UserData, Drone, Notification, and Image Processing. The functionalities and objectives of each of these microservices are:

- Landing Page: main page of the website available to any internet user. Informs the features of platform.
- Dashboard: visual panel that presents information in a centralized way. It indicates processed metrics and allows the registration and monitoring of new drones.
- Gateway: middle ware microservice between frontend and backend microservices.
- Auth and UserData: responsible for storing user data. Also responsible for granting users access to the platform's pages.
- Drone: responsible for storing the drones' data. It communicates with the drones via Internet.
- Notification: microservice dedicated to creating and sending notifications to users.
- Image Processing: microservice responsible for processing images obtained by the drones.

The microservices and the databases were hosted on a cloud platform that allows the deployment, the execution of tests and a continuous monitoring of the application status. The server chosen was Heroku for being free. The addition of extras and new features is possible at very affordable prices or for free. There are other more complete services which offer better conditions, such as Amazon's Amazon Web Services (AWS) and Microsoft's Azure, but their cost of use is quite large, so they did not fit the expectations of this project [82, 83].

Of these seven microservices implemented, two of them are dedicated to providing information to the user, commonly called a website. In these two websites the framework

used for their development was Vue.js, which as already explained in the state of the art, is a framework with a great growth and very accepted by the community of programmers.

In the remaining five microservices, the JavaScript framework used was Node.js. Although the base language of this framework is JavaScript, the implementation of these microservices was in TypeScript. The structure of these microservices in Node.js is quite similar, but there are two cases that differ substantially: Gateway and Drone microservices. This difference will be detailed in sections 3.3.1 and 3.3.4, respectively.

The figure 3.1 illustrates how the architecture of this project was designed. Landing Page and Dashboard, to communicate with the backend services, always send their requests to the Gateway. The Gateway forwards the received requests to the respective microservice. Some of the backend microservices have a connection to an isolated database. A more detailed explanation will be given later in this chapter.

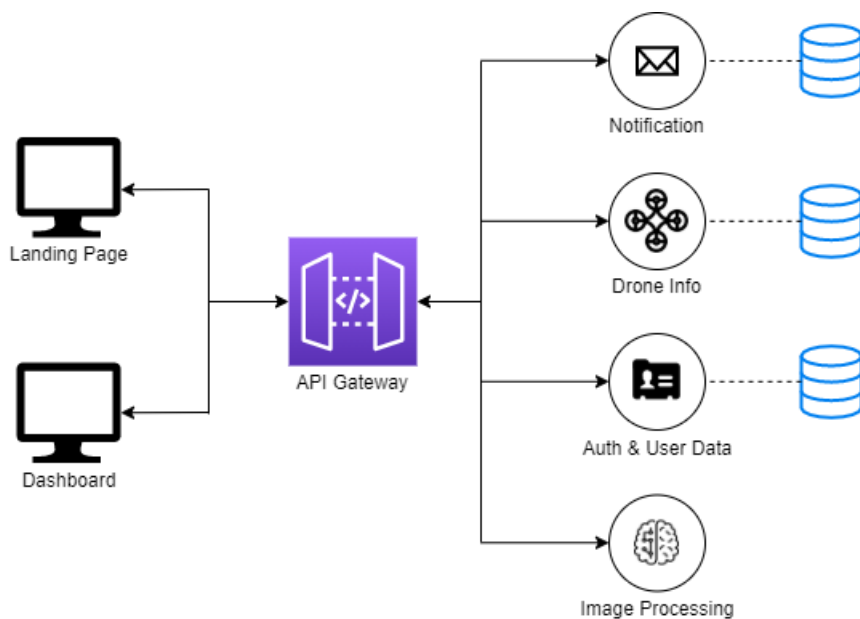


FIGURE 3.1. System Architecture

The implemented microservices use the HTTP protocol to perform their communications. However, the simple use of a communication protocol does not make a system efficient or well implemented: the HTTP protocol does not define any rules or principles for using its methods. Therefore, it is necessary to use the Representational State Transfer (REST) principles, a set of rules and good practices, to allow structured and organized communication. Systems that apply the REST principles are called RESTful.

3.1. Platform Design

The design of the platform had to go according to the expected functionalities for this website. This way, mockups were designed. A mockup is a template or a representation of a project, used to demonstrate the basic idea of a project. In mockups may already be represented attributes, values or colors, but it is not mandatory. The main objective of designing a mockup is to guarantee an initial approval of the system before starting to develop further.

Therefore, mockups were designed in a way to have a structured and organized idea of the website design. The pictures 3.2, 3.3, 3.4 and 3.5 illustrate some of the initial mockups that, of course, differ a lot from the result, but that were a beginning.

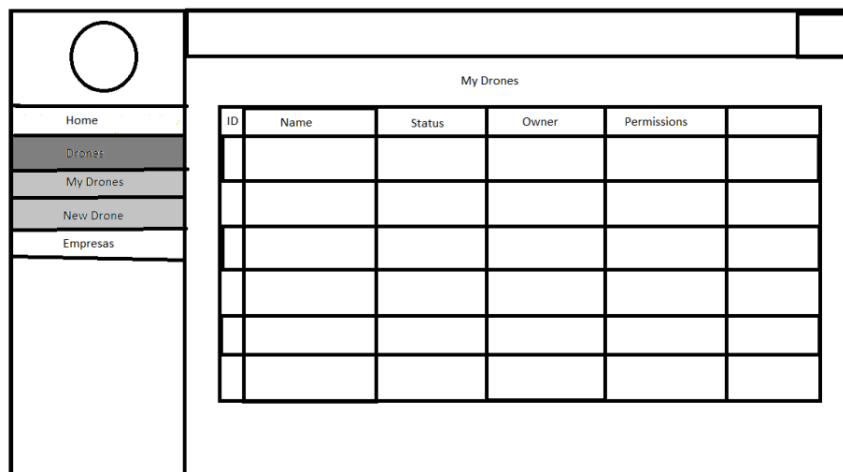


FIGURE 3.2. Devices List Mockup

The basic design of the platform developed comes from Vue.js libraries, as already explained in the section 2.2.4. Changes were then made that were considered necessary in order to meet the challenges inherent to this project. The final result can be seen in chapter 4.

3.2. Frontend Architecture in Vue.js

Vue.js is one of the fastest growing JavaScript frontend frameworks, as it allows great versatility, simple modularization and easy adaptation. The version of Vue.js used in this project was the 2.6, supported in JavaScript.

Like other JavaScript frameworks already explained in this dissertation, Vue uses the concept of component as a way to organize and structure the project. A component is a

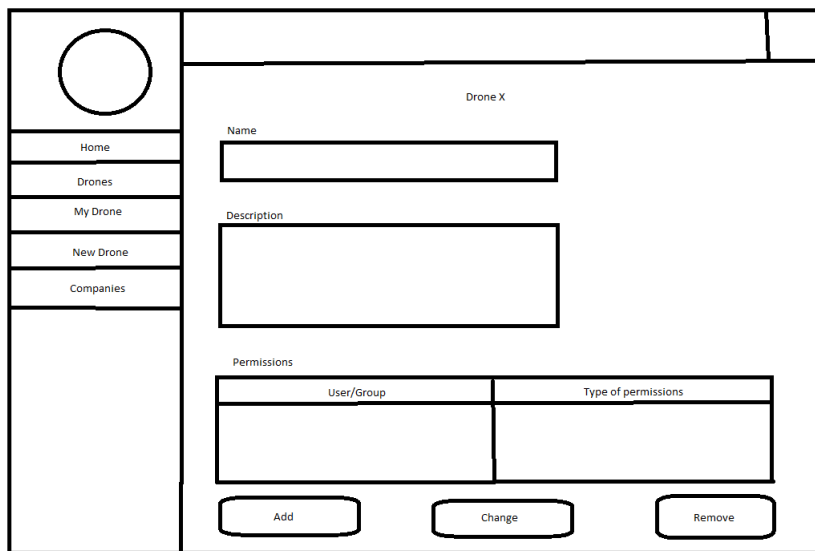


FIGURE 3.3. Device Mockup

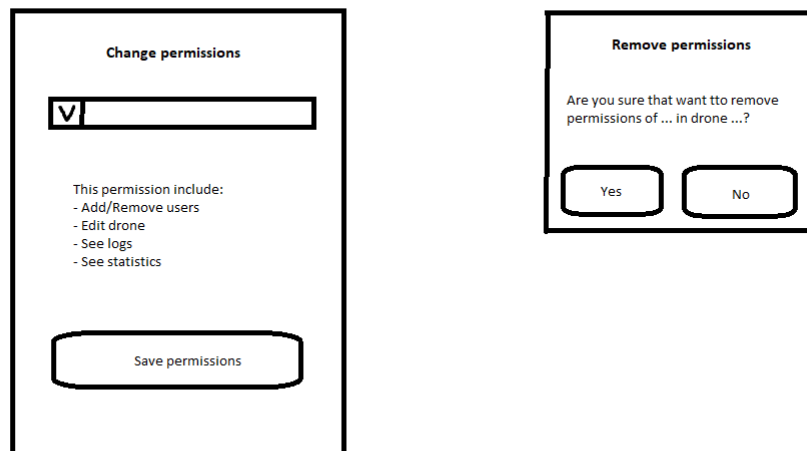


FIGURE 3.4. Change and Remove Permissions Mockup

reusable instance, created using the syntax of HTML with JavaScript code, which allows iterations and manipulations that the native HTML does not have. To be reusable, the components are created in their own files that allow their integration through specific tags, as will be demonstrated later.

Vue applications allow you to have several different programming syntaxes in a single .vue file: HTML code, used to create the skeleton of the web page, CSS, directed to the

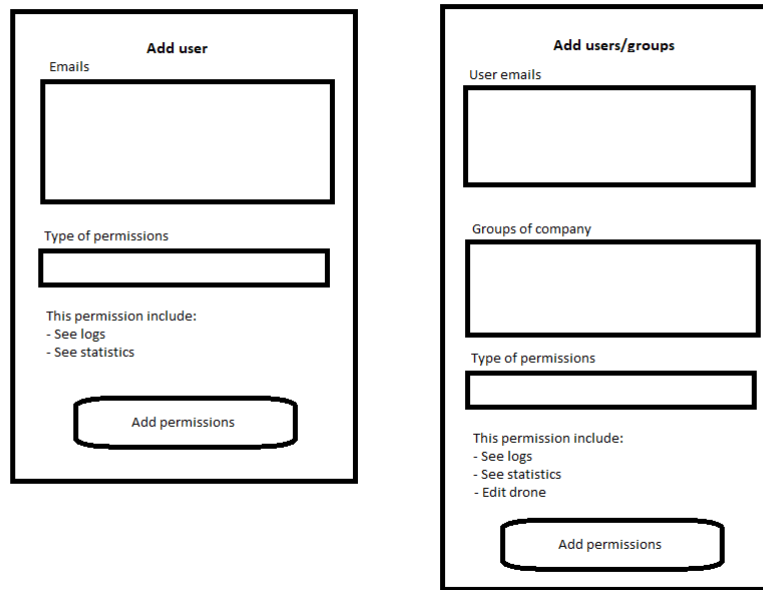


FIGURE 3.5. Add Permissions Mockup

look of the web page, and JavaScript, which allows to connect the web page to the logic of the application. In structural terms, the developed microservices that use Vue.js have a structure similar to the one illustrated in the 3.6 image.

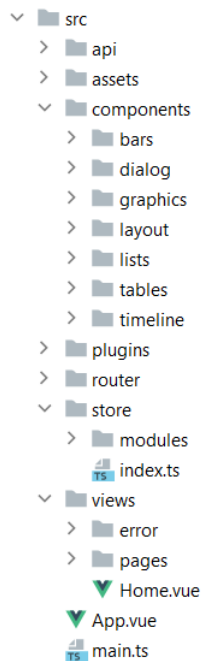


FIGURE 3.6. Dashboard Organization

The integration of Vue with libraries and CSS frameworks is possible, and greatly facilitated with the use of the Vuetify framework. This framework aims to provide a number of customized components for Vue applications. The use of this type of components allows simplifying and saving development time, giving the developer the opportunity to dedicate himself to other more relevant aspects. Although these components are already customized, there is the possibility of being customized by the developer, and this was the option taken throughout this project.

The two frontend microservices have an API that allows communication with the rest of the platform. The API is used whenever the application needs to make an information request for any backend service. As explained above, all requests made between the frontend and backend pass through the Gateway so that it is possible to make the appropriate validations when necessary.

The creation of these requests is done using the axios library, an HTTP client that allows making any of the HTTP requests, either in a browser or a Node.js server. The functions to make the requests are already pre-made, needing only the content of the request that is provided when it is invoked. Figure 3.7 shows two blocks of code that indicate how an API call is made in this project. The simplicity presented is the result of the good organization of the project.

```

async getUsers() {
  return await axios.axiosGet(url);
},
async getUser(userId: number) {
  return axios.axiosGetId( url: url + "/:id", userId);
},

function axiosGet(url: string) {
  return axios
    .get( url: baseUrl + "/" + url, config: { withCredentials: true })
    .then(resp => {
      return resp;
    })
    .catch(error => {
      return error.message;
    });
}

```

FIGURE 3.7. API Code Example to Get Users

The use of components as already mentioned, is a constant aspect in the development of these microservices. Each component is implemented in a single file and, in principle, are independent from each other. The exception appears when a component aggregates another component. The components can be reused quite easily and quickly by just importing them into the page to be used and then invoking it in HTML syntax, referencing its name between tags.

The figure 3.8 shows a code block refers to a situation of how the use of a component can be used. Using JavaScript syntax you can implement the component created in a separate file. Then, it is possible to include the component in the part of the code


```

<template>
  <div class="page-wrapper">
    <v-container id="devices" fluid tag="section">
      <v-row justify="center">
        <v-col cols="12">
          <plain-table-devices />
        </v-col>
        <div class="add-device-button">
          <CreateDevice />
        </div>
      </v-row>
    </v-container>
  </div>
</template>

<script>
import CreateDevice from "@components/dialog/CreateDevice";
import PlainTableDevices from "@components/tables/PlainTableDevices";

```

FIGURE 3.8. Sample Code for use of Components

dedicated to HTML, giving the name of the imported component between tags. In this case, it is demonstrated the inclusion of the component *PlainTableDevices*, referring to the device table, and the *CreateDevice* component, the button that appears on this page that allows the registration of a new drone.

A Vue.js component does not necessarily need to be a small block of code, nor should it be considered solely as an object that integrates into other pages already made. The web page can be considered a component, as was done in this application: the page component includes other smaller components.

Vue being a Single Page Application (SPA), can change the content presented on the page quickly, making the user experience very enjoyable. In order to use this capability, Vue uses a framework called vue-router that allows the management of the content to be made available to the user according to the URL address. The Figure 3.9 illustrates how this feature was implemented in the project, in which depending on the URL received, a component is associated that will be made available to the user.

The variable *requireAuth* is related to user authentication, which will be explained in section 3.6.

Finally, another framework that was implemented in this project was Vuex. Vuex is a state and data management standard in Vue.js applications. This framework works as a temporary database centralized in the application and accessible by all components, depending on the permissions granted to each.

```

    {
      path: "/dashboard",
      name: "Dashboard",
      component: () => import("../views/Home.vue"),
      meta: {
        requiresAuth: true
      }
    },
    {
      path: "/devices",
      name: "Devices",
      meta: {
        requiresAuth: true
      },
      component: () => import("../views/pages/Devices.vue")
    },
  ],

```

FIGURE 3.9. Vue-router Implementation

The great advantage of this type of store is related to the ability to update information shared by several components. At the same time, no component modifies this data directly, ensuring data consistency. This capability inserted in a Vue project allows that, if two components use the same variable, the change of the value of the variable by one of the components immediately leads to an update of the state of the variable in the other component. This makes the pages more interactive and provides a better user experience.

The Figure 3.10 illustrates the diagram of how this process works:

- (1) A component makes an order to the Vuex that it needs to obtain (or insert) information from the database.
- (2) In the Actions module, a request is made to the application's Gateway, using the APIs already explained. Once the request is made, it waits for the answer.
- (3) In case of a positive response, the content received is committed to the store, which is done through Mutations. These changes are reflected in the State.
- (4) Finally, the Component is constantly listening to the desired variables in the store, that is, in the State module. When it has the desired content, it makes it available to the user. In the situation where two components have the same variable, the change made by one of the components to the variable is reflected in State.

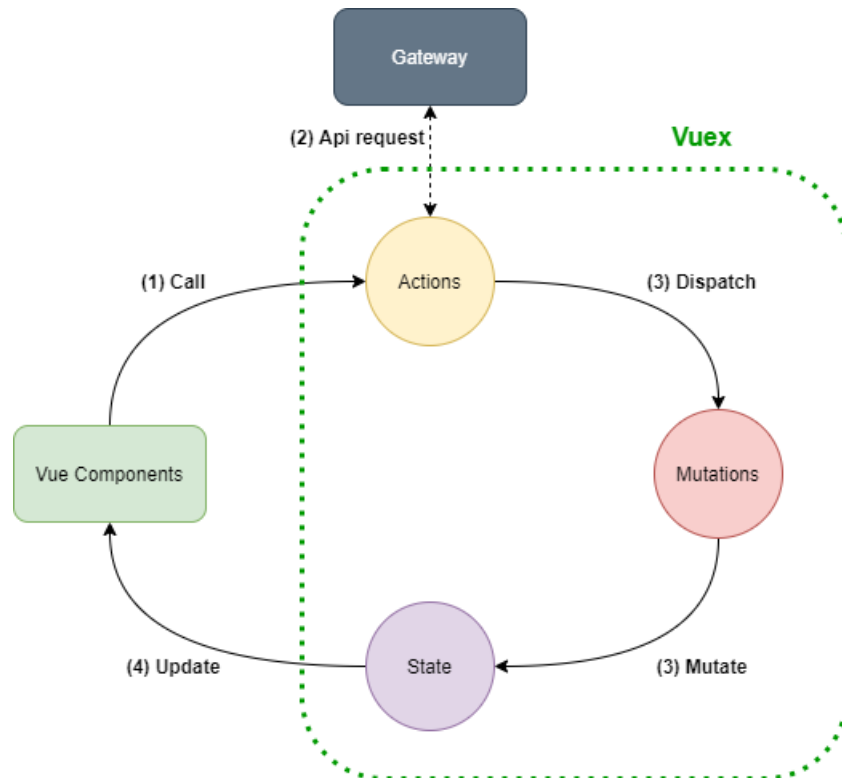


FIGURE 3.10. Vuex Architecture

The creation of a project in Vue.js is not necessarily complicated, but rather time consuming. However, the use of this type of frameworks brings advantages in the medium and long term, especially if the project starts to grow and become more complete.

3.3. Backend Architecture in Node.js

The backend microservices were developed in Node.js, being this one of the most used technologies for backend servers in JavaScript. The version used in the microservices was the 12.14 version, the most current at the start date of this project. It should be noted that the language used in these projects was not JavaScript, but a variation, TypeScript. The advantages of this language compared to the traditional JavaScript are described in section 2.2.1.

The architecture used in these microservices, with the exception of the Gateway's microservice and the Drone's microservice, as a result of the specific characteristics of these microservices, follow the architecture presented in figure 3.11. The same happens in the project organization, demonstrated in figure 3.12. Anyway, all good practices were respected in the development of these projects.

The reception of HTTP requests is possible through the Express.js framework, or simply express, which allows the construction of a web API in Node.js environments. These

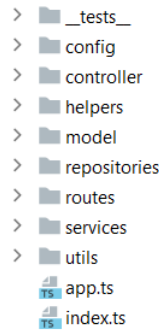


FIGURE 3.11. Authentication and UserData Organization

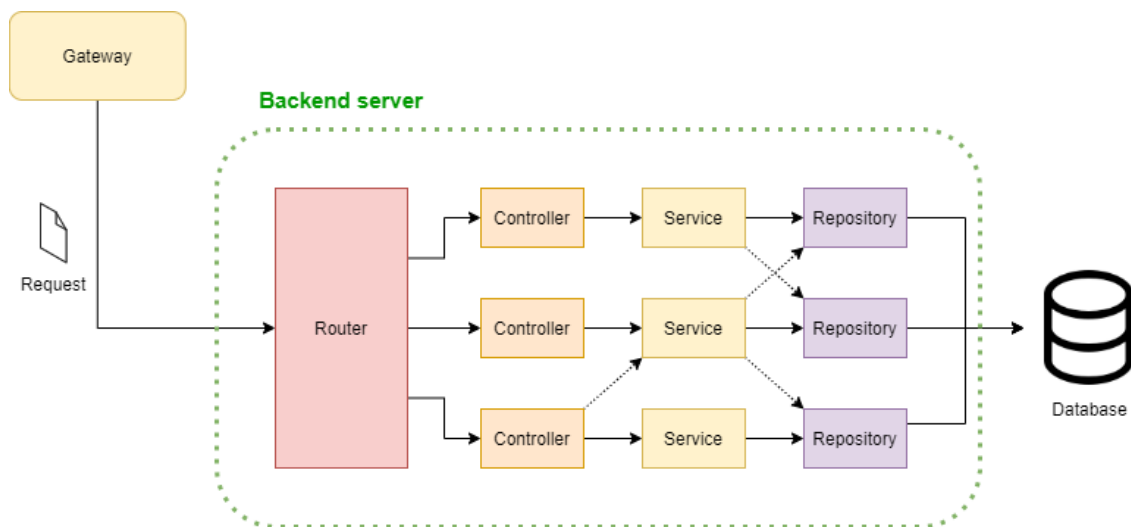


FIGURE 3.12. Backend Architecture

APIs can handle multiple requests simultaneously. As requests arrive, they are queued and processed consecutively as soon as possible. Each application has been configured with 4 threads pools to process the requests, i.e. the system can simultaneously process 4 processes at the same time. This value can be increased if justified. The 3.13 figure demonstrates how the Node.js architecture works in one of the developed microservices.

For each new request that arrives at the microservice, the URL prefix is analyzed, originating a forwarding of the request to one of the project controllers. This is described as routing and can be seen in figure 3.14.

In the controller, its first task is to validate the request body. This must respect a predetermined structure that is in accordance with the expected structure. This validation, which is done through a Data Transfer Object (DTO) is an added value in terms of security. Besides fulfilling its main purpose, which is to validate if the fields received in the body of the request are the expected ones, it checks if the content respects certain imposed rules. Thus, it is guaranteed that the content processed by the application

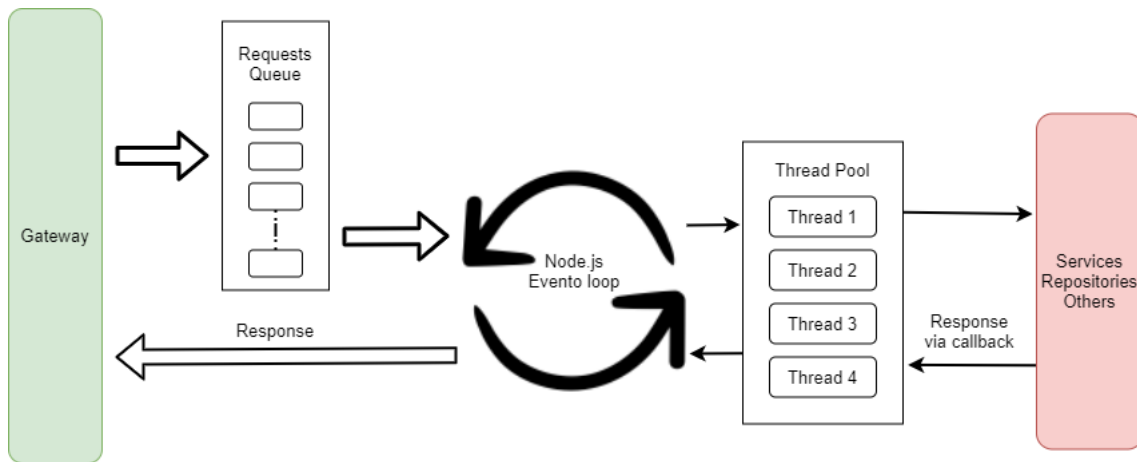


FIGURE 3.13. Node.js Architecture

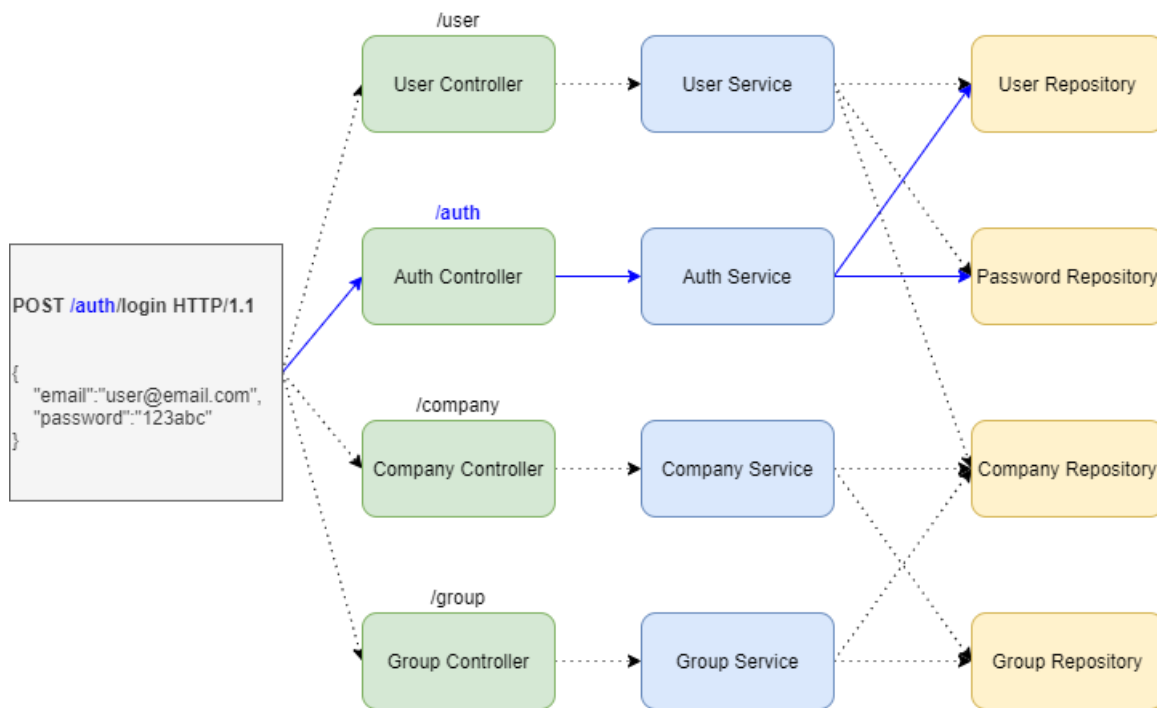


FIGURE 3.14. Node.js Routing

does not allow to make adulterations in the code. If the request has been filtered by the controller, it sends the request to the most appropriate service.

The service module is responsible for all the microservice logic. It is divided into several files depending on the type of objects processed. The content of the received request is validated again and, if possible, leads to the creation of a Model object, or alternatively Data Access Object (DAO). This new object created will be explained later in section 3.4.1.

Chapter 3 *System Implementation*

Any error or invalidation that occurs in the process is almost all treated in this module. The logs are inserted in a .log file whose name is the current date. The structure of the logs is shown in figure 3.15, which contains information such as the Internet Protocol (IP) and user id, and data that allows a follow-up of the executed processes. It is important to have a well structured log system. Any error that occurs during the process should be stored in a file for later analysis. The structure and the information contained in the files varies according to the microservice.

```
[2020-10-18T17:02:57.742] [INFO] SERVER - Server on port: 8085
[2020-10-18T17:02:57.964] [INFO] SERVER - Database connected.
[2020-10-18T17:03:03.371] [INFO] [AXIOS] [ID=null] [IP::ffff:127.0.0.1] [EMAIL=null] [FILE=AxiosService.ts] Get request sent successfully on operation VALIDATE USER COMPANY ID
[2020-10-18T17:03:03.413] [INFO] [DRONE_SERVICE] [ID=1] [IP::ffff:127.0.0.1] [EMAIL=null] [FILE=DroneService.ts] Drone created successfully;
[2020-10-18T17:03:14.005] [INFO] [DRONE_SERVICE] [ID=1] [IP::ffff:127.0.0.1] [EMAIL=null] [FILE=DroneService.ts] Drone founded.
[2020-10-18T17:03:23.528] [INFO] [AXIOS] [ID=null] [IP::ffff:127.0.0.1] [EMAIL=null] [FILE=AxiosService.ts] Get request sent successfully on operation GET GROUPS OF USER
[2020-10-18T17:03:23.543] [INFO] [DRONE_SERVICE] [ID=1] [IP::ffff:127.0.0.1] [EMAIL=null] [FILE=DroneService.ts] Drone updated.
[2020-10-18T17:03:26.498] [INFO] [DRONE_SERVICE] [ID=7] [IP::ffff:127.0.0.1] [EMAIL=null] [FILE=DroneService.ts] Drone founded.
[2020-10-18T17:03:36.196] [INFO] [AXIOS] [ID=null] [IP::ffff:127.0.0.1] [EMAIL=null] [FILE=AxiosService.ts] Get request sent successfully on operation GET GROUPS OF USER
```

FIGURE 3.15. Log File Structure

Associated to one service can be one or more repositories. Depending on the service, they may need to make queries, inserts, updates or deletions in the database. Repositories are classes responsible for managing these access requests. Their existence provides greater organization of the code. In the repository, each function is responsible for a single request to the database. The communication between the services and the repositories is done through a proper interface. A more detailed explanation of how communication with the database works is detailed in section 3.4.

3.3.1. Gateway microservice

Despite being developed in Node.js, the Gateway presents a different architecture, based on the functionalities that it needs to have. This microservice has as main responsibility the forwarding of requests received by the frontend to the backend, and consequent reverse forwarding with the response.

The reception of HTTP requests happens in the same way as other backend microservices, using the Express.js framework. And since the Gateway also needs to create HTTP requests, it contains in its code the necessary mechanisms to create requests via axios. Figure 3.16 illustrates the Gateway's architecture when it comes to message processing.

The Gateway also acts as an important role in validating requests, ensuring that the origin of requests comes from an authorized person. This information is explained in section 3.6.

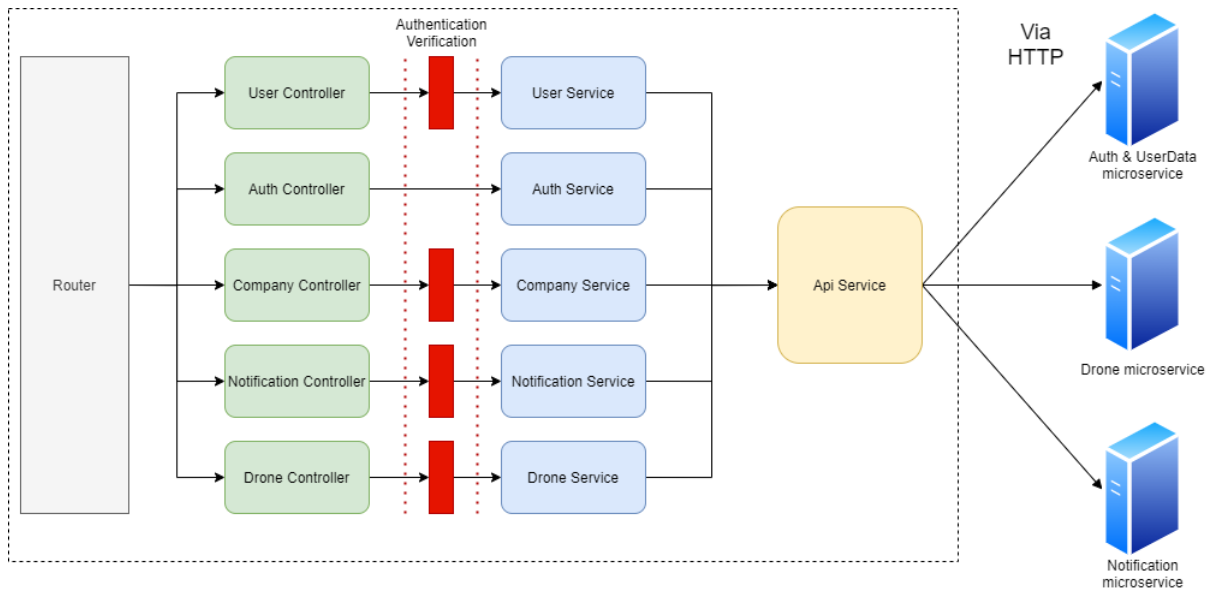


FIGURE 3.16. Gateway Architecture

3.3.2. Notification microservice

The notification microservice, besides including all the features already detailed in section 3.3 for being developed in Node.js, also includes the ability to send emails through the SMTP protocol. The figure 3.17 shows the architecture of this microservice.

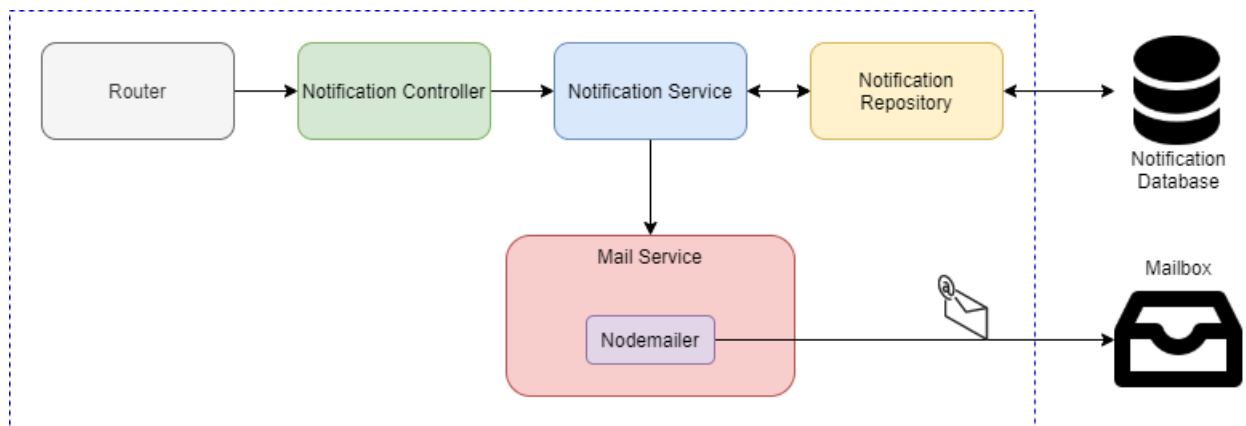


FIGURE 3.17. Notification Architecture

The nodemailer library allows the configuration and the sending of e-mails in Node.js applications. The emails sent can be of the type:

- Welcome to new users.
- Password recovery.
- Invitation to register on the platform.
- Invitation to join a group or a company.
- Fire detection alert.

The email used to send this information is associated with the domain acquired by the company Amen.pt.

3.3.3. Image Processing Microservice

This microservice has a different structure from the others since it also includes an algorithm. The algorithm made in python intends to represent a machine learning with the ability to identify forest fires through image processing. This machine learning was developed in another project made by the Institute of Telecommunications of Iscte-iul and adapted so that it could be integrated into a microservice platform [2].

This microservice, firstly, is developed in Node and presents the architecture already presented previously. It receives requests via HTTP, but by default this protocol cannot incorporate images in its requests. To get around this situation, it is necessary to use the *multipart/form-data* encoding that allows the transfer of binary files, in which an image is included, through the HTTP protocol.

The received images are stored in a server folder, *toProcess* folder, and the file name is defined according to the *unixtime-droneid.png* structure, where the first part corresponds to the date of reception, in Unix Time Stamp (UTS), and the second part to the drone id.

The algorithm is called by the Node thread responsible for processing the request. If the algorithm detects something suspicious, the microservice sends a request to the Notification microservice so that it alerts those device's responsible a suspicion of forest fire was detected. All processed images are moved to the *processed* folder.

3.3.4. Drone Microservice

This microservice has an architecture very similar to the one presented in section 3.3: it receives HTTP requests from the Gateway, processes them and has a database to store information, in this case drones. However, this microservice needs to actively communicate with drones. Therefore, in addition to the common features of a backend microservice, it uses the mavlink library to create MavLink requests to communicate with the drones. The implementation of the mavlink in this microservice is similar to that used in the Gateway.

MavLink requests are addressed to a destination drone, where the content of the message sent are commands that allow the drone to execute tasks. The Drone's microservice is

in constant contact with the drone so that it can store the collected data in the database. The figure 3.18 shows the architecture adopted for this microservice.

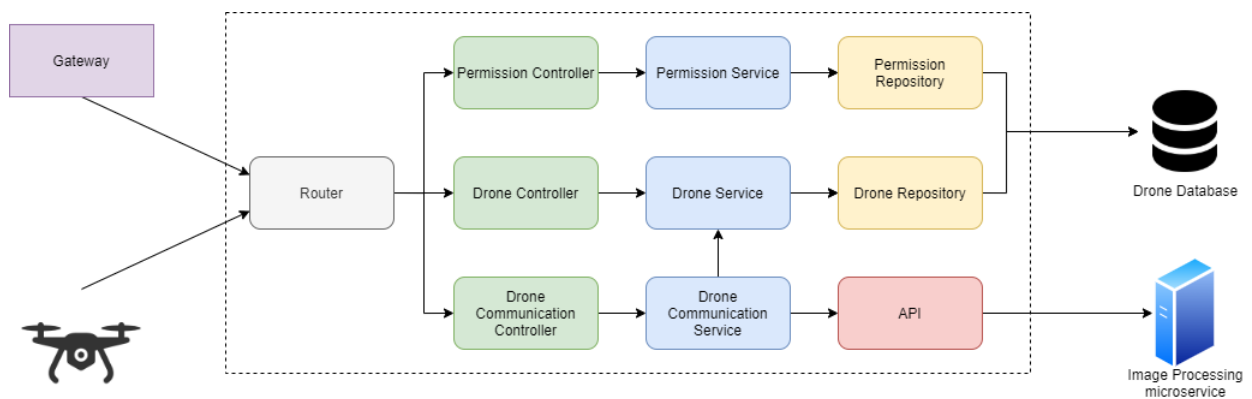


FIGURE 3.18. Drone Architecture

The figure 3.19 shows the code block responsible for creating the orders to be sent to the drone.

```

function send_command_long(param1, param2, param3, param4, param5, param6, param7, command, confirmation) {
  mav_parser.createMessage( msgid: "COMMAND_LONG", data: {
    'param1': param1,
    'param2': param2,
    'param3': param3,
    'param4': param4,
    'param5': param5,
    'param6': param6,
    'param7': param7,
    'command': command,
    'target_system': 1,
    'target_component': 1,
    'confirmation': confirmation
  }
),
  cb: function (message) {
    port_send_message(message);
    console.log(message);
  });
}
  
```

FIGURE 3.19. Sample Code to Communicate with Drone via MavLink

3.4. Database

The databases are associated with only a few backend microservices. These are the Auth & UserData microservice, the Notification microservice and the Drone microservice. Each one of these microservices has a unique database that is only accessible by itself.

The databases were developed in MariaDB, the free version of MySQL, and their design was done through the MySQL Workbench software that, besides allowing the

design, allows its export and especially the execution of a database locally. This allowed that during the development of the platform a local database was used instead of the production database that is hosted on the Heroku server.

The Figures 3.20, 3.21 and 3.22 illustrate the relational diagram of the various databases developed.

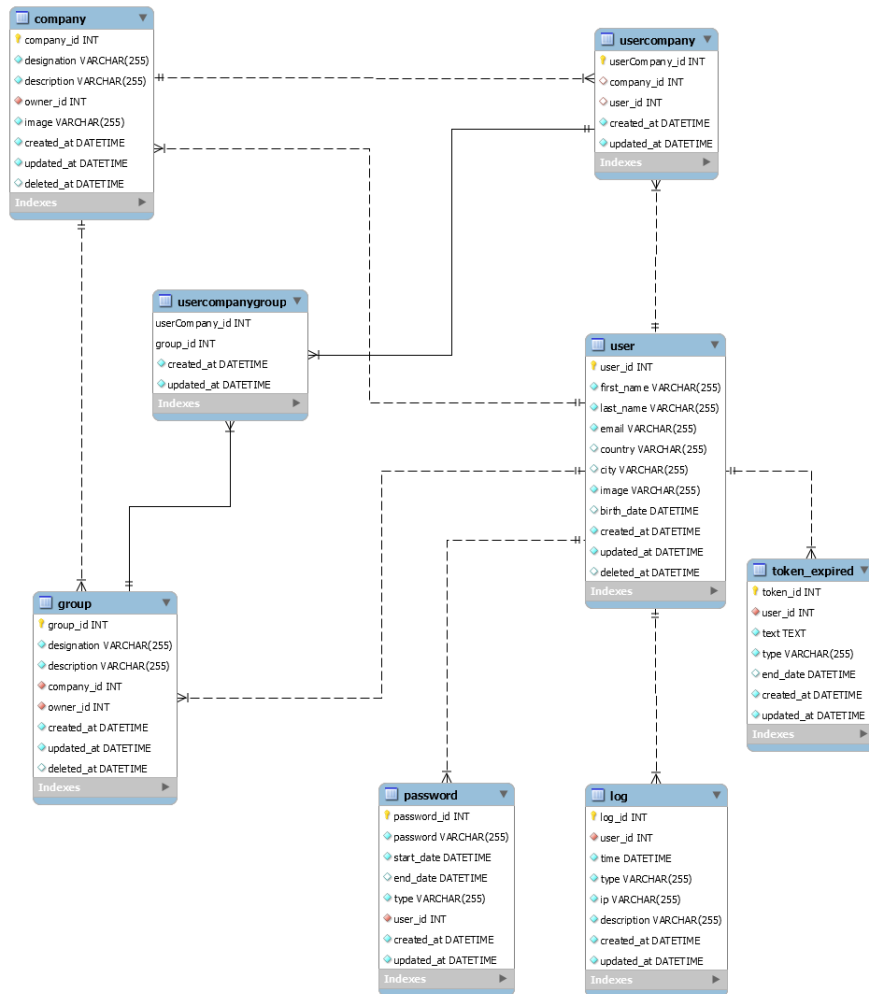


FIGURE 3.20. UserData Relational Diagram

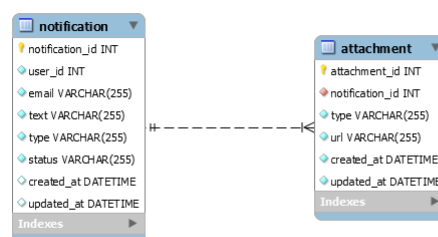


FIGURE 3.21. Notification Relational Diagram

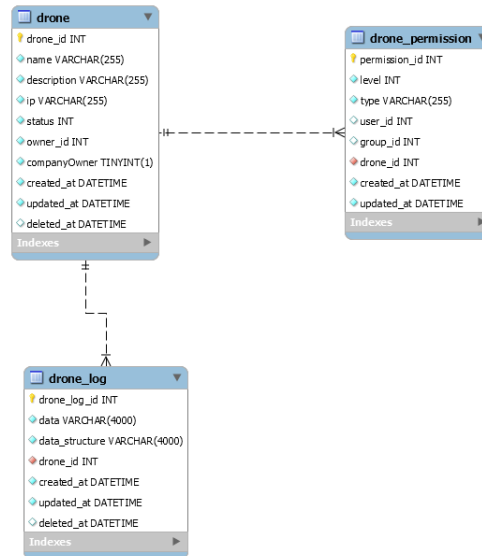


FIGURE 3.22. Drone Relational Diagram

At the start-up of each microservice, the connection to the respective database is initiated. Depending on the environment in which the application takes place, the connection is either made to the local database or to the production database.

In accordance with the rules of the European area, described in the General Data Protection Regulations, the data stored in the database are protected and are hosted on trusted servers. It also includes the right to forget, which allows the user to remove all data that allow their identification [84]. Thus, when the user makes this request, all fields containing sensitive data, such as name, image, and email, are replaced by a default text *'DELETED'*. This way we can ensure that the system continues to work correctly, as there was no abrupt removal of data, guaranteeing the rights of users registered on the platform.

3.4.1. Object Relational Mapper

To manage the database, an Object-Relational Mapping (ORM) called Sequelize was used. This promise-based library supports a wide database dialect, including MySQL, and has features and capabilities related to data transaction and the relationship between objects.

All the objects of the application, called Model, are built in ORM optics. This allows the manipulation of objects like User, Company or Drone to be more fluid and safer. One of the great advantages of using this type of objects is the transaction management, which allows rollbacks in case of failure in a certain step of the process, and the need not to use SQL code to make the requests to the database. In the 3.23 image it is possible to see

the create function, in the companies repository, which allows the creation of a company in the database receiving an ORM of type Company. In detriment of the typical SQL create command, in the code only the save method is invoked that automatically creates the object in the database.

```

create(company: Company, options?: any): Promise<Company> {
  return company.save(options)
    .then( onFulfill: result => {
      if (result) {
        return result;
      } else {
        return null;
      }
    })
    .catch( onReject: error => {
      throw error;
    })
}

```

FIGURE 3.23. Registration of an Company using Sequelize

These Sequelize ORM objects have in their class a direct association to the respective table in the database. First by indicating the name of the class, making the object Company correspond to the company table, and second by making a similar association between the parameters of the object and the columns of the table. The characteristics of the columns are also explicit in the ORM class, thus allowing a validation of data even before executing the action.

3.5. Tests

Throughout the project, unit tests and integration tests were developed, as a way to prove and certify that the implementation was correct. This allowed the development on the platform not to suffer significant delays, since it was not necessary to redo functionalities that could be working incorrectly.

The implementation of the tests, and the project code, followed the Test-Driven Development (TDD) approach. This approach consists in the development of the test-oriented software, consisting in the maximum realization of tests in order to contain all possible possibilities. The development process in TDD is the following [85]:

- (1) Write tests without having the code done.

- (2) Execute the tests and, as there is no code done, it will cause errors in all the tests.
- (3) Develop the functionality for the test until it passes.
- (4) Repeat the procedure of development of functionalities until all the tests of this functionality are positive.
- (5) Pass to the next functionality and repeat the process.

This way, and if the tests are well done and cover all the possibilities, either of success or of error, we guarantee that the writing of the code that is done later is correct. In figure 3.24 there is a code block that represents a test to the creation of drones on the platform. The tests were developed using the Jest framework, explained in section 2.5.1.

```
it( name: 'Drone created successfully', fn: async done => {
  const body = {
    name: "Drone 1",
    description: "Some description",
    owner_id: 1,
    companyOwner: false
  };
  const ip = '127.0.0.1';

  const result = await droneService.createDrone(body, body.owner_id, ip);
  expect(result).toBeInstanceOf(Drone);
  expect(result).not.toBe( expected: null);
  expect(result).not.toBe( expected: undefined);
  expect(result.name).toEqual(body.name);
  await result.destroy();
  done();
});
```

FIGURE 3.24. Unit Test of Drone Creation

3.6. Security

As it is a web platform, accessible to any user, it is necessary to ensure that the data processed by the application is safe. For this, several ways have been used to ensure that the entire system is able to prevent possible attacks to disable the platform or to access private information.

The first measure implemented was to obtain an SSL certificate. The use of this certificate on the website allows communications between users' browsers and the web server to be encrypted, as already explained in section 2.7.1. The type of certificate obtained was the simplest, DV Certificate. Because this is still a small platform with

little traffic there was no need to invest in a more robust certificate. This certificate was obtained at Amen.pt, a company accredited by several CA.

The use of Sequelize allows a greater security in the access to the database. One of the most common attacks on the database is through SQL Injection. This attack consists in the adulteration of variables to be used in SQL scripts in order to return something different, as shown in figure 3.25.

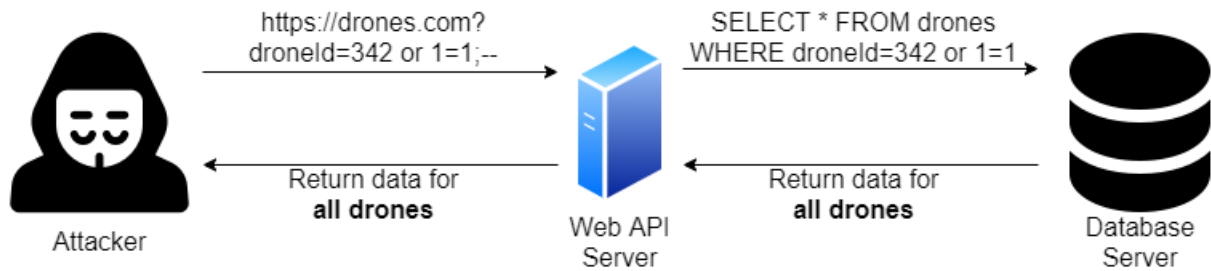


FIGURE 3.25. SQL Injection Scheme

Sequelize, although allowing the use of SQL scripts in the traditional way, its implementation in this project was more directed to the concept of ORM, explained in section 3.4.1. The use of these methods gives the possibility to make invocations to the database in a more secure way. Likewise, Sequelize already integrates in its base code mechanisms that prevent this type of attacks, so its use ensures greater security.

The authenticity check is done through the validation of the JWT, described in section 2.7.3. The use of this token in browser cookies is fundamental in accessing restricted pages, such as the dashboard, which should only be accessible if the login has been made.

As illustrated in figure 3.26, on the Landing Page login page, the user enters the correct data for authentication. These are sent to the Gateway, which redirects it to the Auth & UserData microservice. It is in this microservice that occurs, among other things, the verification of the password with that stored in the database. In case of success, a JWT token is created with the user id, email and IP, using an RSA private key. Then, the JWT takes the reverse route of the login request, being answered by Auth & UserData to the Gateway, where it will include the JWT in the cookies of the user response. This way, when the user is redirected to the dashboard page, they will already have the cookie in their browser.

Figure 3.27 shows how a JWT token is built. It is divided into three parts:

- Header - contains the algorithm used and the type of token, in this case RSA256 and JWT, respectively.

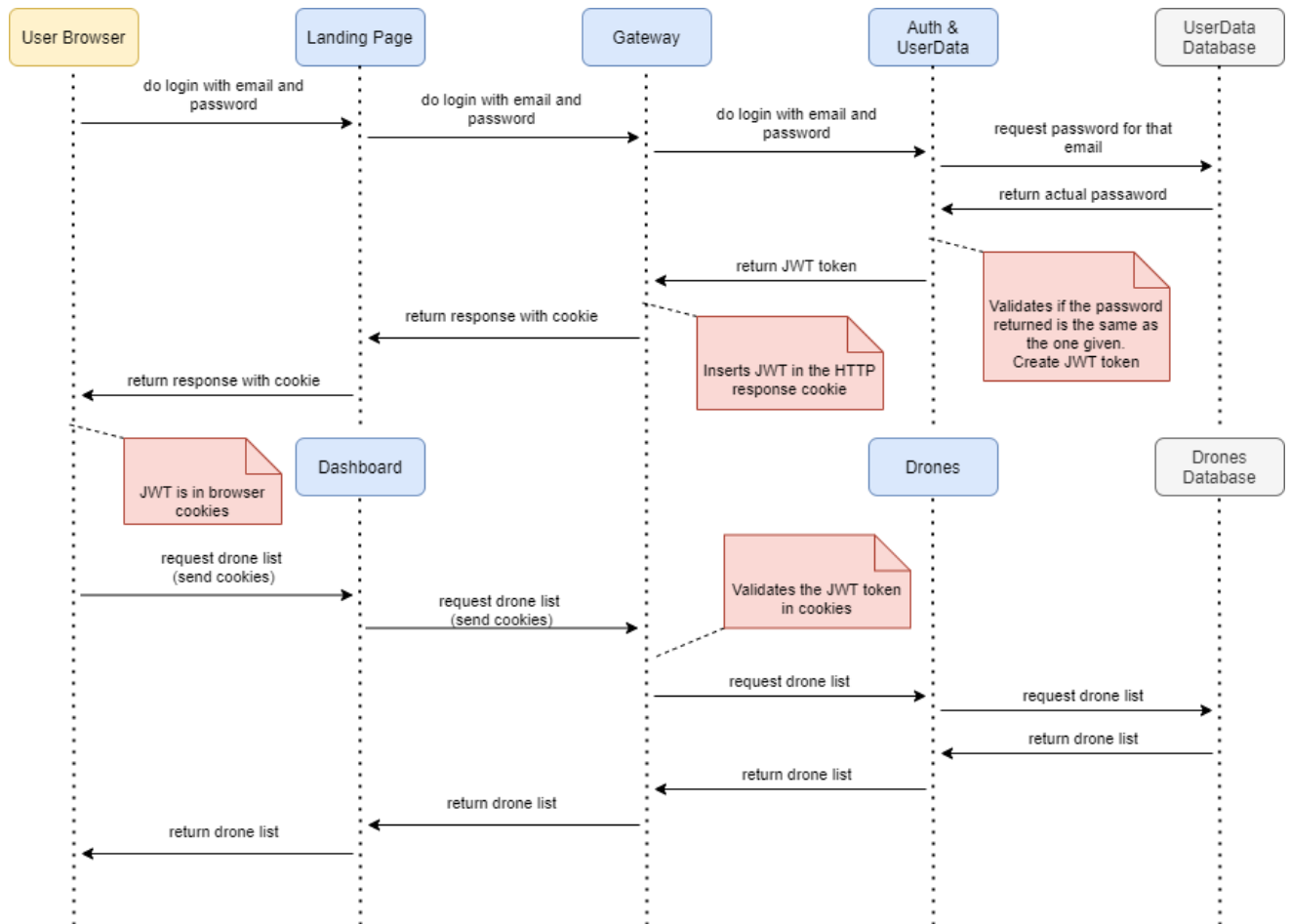


FIGURE 3.26. Login Scheme with JWT

- Payload: contains essentially the content (the user id, your email and your IP), but also additional information such as the creation date of the token and the expiration date, in UTS.
- Signature: contains the first two parts coded, so it is possible to check later if the message has changed. It also allows to verify that the origin of the token is who it says it really is, because it has the private key.

Some Dashboard pages may require the user to be authenticated. These accesses are indicated in figure 3.9 through the variable *requireAuth* which, if true, the Dashboard makes a verification request to the Gateway that the JWT cookie, if any, is valid.

The Gateway is responsible for validating the authentication tokens through the RSA public key. If it is possible to decrypt the JWT through that key it means that the token is a valid token on the platform, but an extra security validation is still needed: the IP. As an extra security measure, the IP stored in the JWT token must match the IP of the user who made the request. If everything is in accordance, the Gateway responds

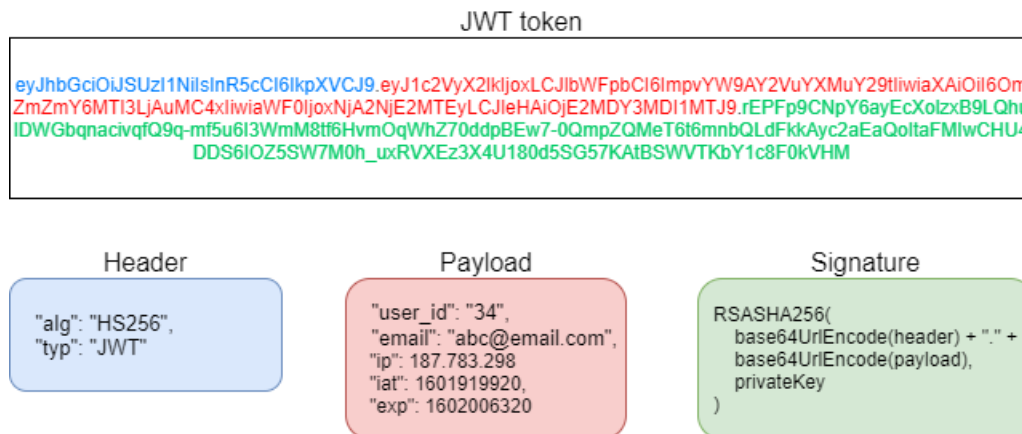


FIGURE 3.27. JWT Structure

positively to the validation requested by the Dashboard, thus allowing the user access to the rest of the platform.

A similar strategy has been adopted in the remaining requests that are made to the backend services. All HTTP requests that are made, pass through the Gateway that will do the JWT validation and will only forward the request to the specific microservice if it is valid. This allows a continuous validation in the access to the platform and its data.

At the same time that the JWT token is generated, another token called *refreshToken* is also generated. This token, also JWT, allows the renewal of the main token after its expiration. The JWT token has a maximum usage period of one day, unless the user logs out of the platform before. After that day, if the user continues with the token in the cookies, when accessing the platform, the token will have expired. In these situations, if *refreshToken* is in the cookies, there will be a renewal of the user's access by creating a new valid token. This mechanism is part of the JWT and prevents users from needing to make repetitive logins if the token has already expired.

CHAPTER 4

Results

The main results of the implementation of the previous chapter will be presented below, namely the web platform developed. The results obtained come from three different but interconnected sources. The developed platform, where the objective has always been to make it available online, was successfully achieved. The various microservices developed communicate between themselves perfectly and correspond to the expectations taken at the beginning of the development of this project. This platform includes the machine learning algorithm that allows image processing.

The results obtained in terms of communication with the drone, and the obtaining of its data, were made in a simulated environment. This environment would always be the first to be tested before performing open field tests with a real device. However, due to numerous circumstances this part of the project could not be completed. Nevertheless, the whole system is operational and working as desired.

4.1. Web Platform



FIGURE 4.1. Flydren Website Homepage

As already explained, the main objective of this dissertation was the development of a web platform that would allow the control of UAVs, more specifically MAVs. The

platform was developed locally and phased to the Heroku server, where it is currently active under the name Flydren and accessible by the domain <https://www.flydren.pt>.

In figures 4.1, 4.2 and 4.3 it is possible to see the main page of the platform, the Landing Page, where the basic information of the system, as well as the services available, can be found. The website is very receptive and can be accessed by computers, tablets, and smartphones. A page about the presented project and a contact form were also developed.

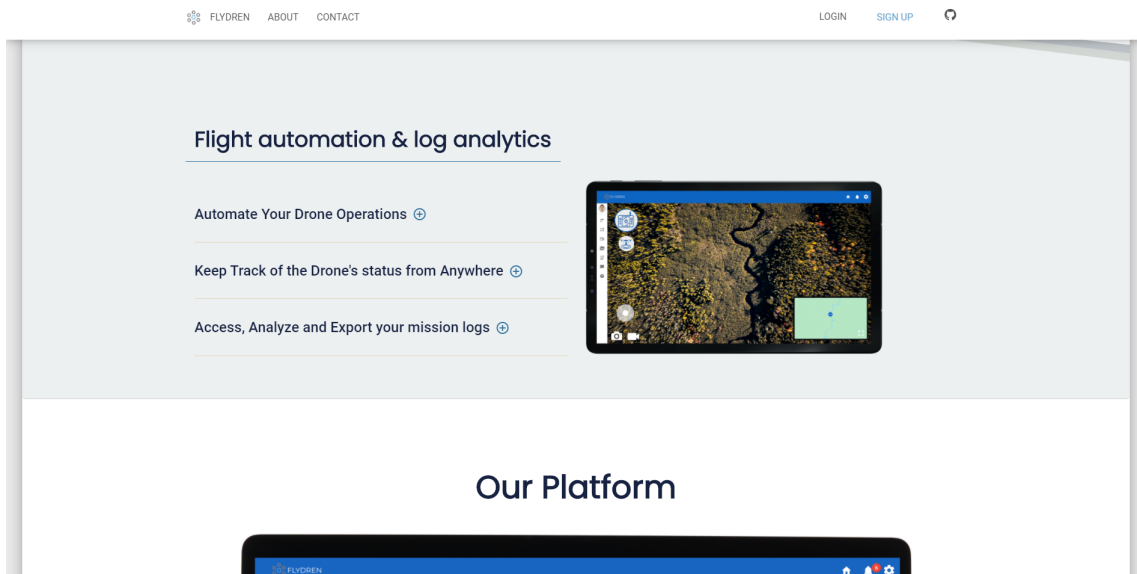


FIGURE 4.2. Flydren Website Flight Automation Details

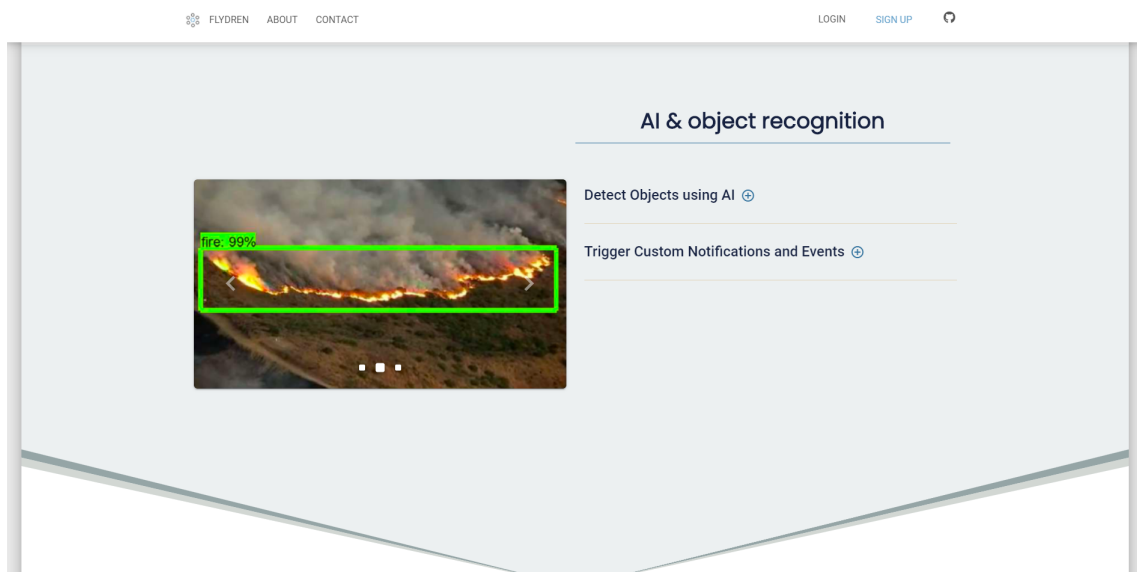


FIGURE 4.3. Flydren Website Object Recognition Details

The registration and the login of the platform were developed in the Landing Page microservice, as can be seen by figures 4.4 and 4.5. The registration on the platform is quite fast. The user needs to give some information such as name, email, country, city, and password. If the registration is successful, a confirmation message will appear on the page. An email will also be sent to the user. After logging in the platform, the user is redirected to another sub-domain, dedicated to Dashboard.

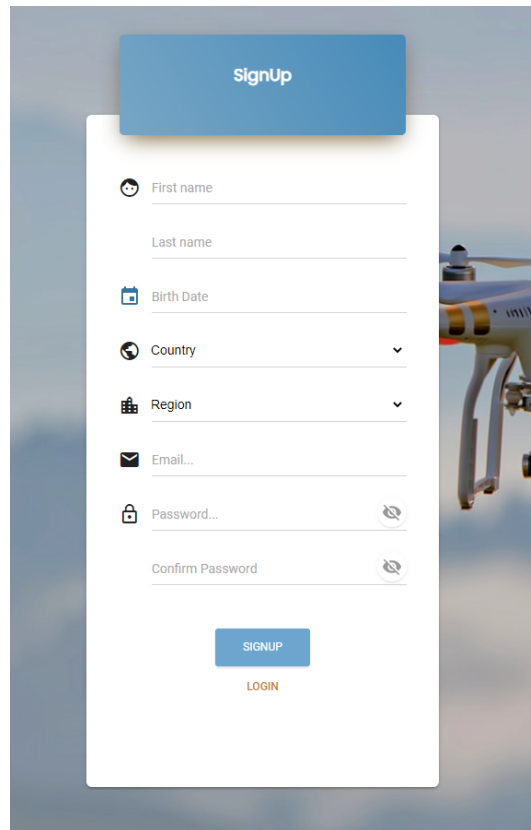


FIGURE 4.4. Flydren Signup Page

On the main page of the Dashboard, hosted in <https://dashboard.flydren.pt>, graphs with statistical information of the drones that the user has access are presented. As it is possible to see in figure 4.6, data such as the number of flights made, and the number of photos and videos captured by the devices are illustrated. It is possible to consult the log history, accessing the files directly.

The platform provides a list of drones that is presented on the main page of the Dashboard and on a page dedicated exclusively to this purpose, shown in Figure 4.7. The list contains all the drones that the user has access to, and the permission level is divided into three:

- Normal: the user only has access to the statistics that the drone generates.

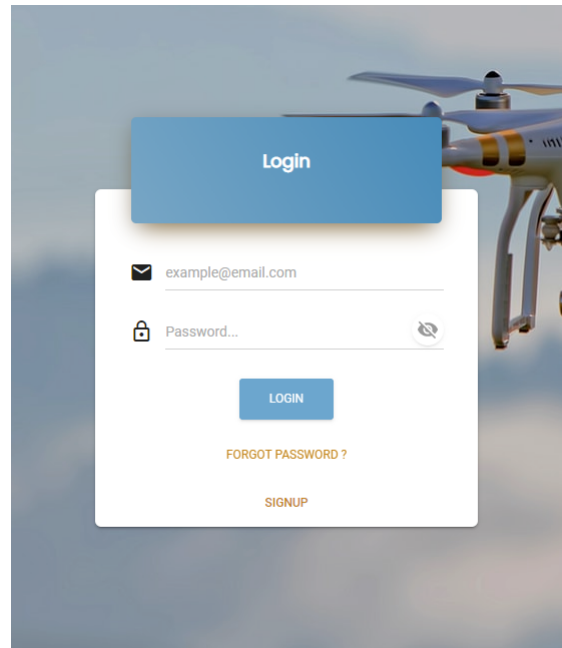


FIGURE 4.5. Flydren Login Page

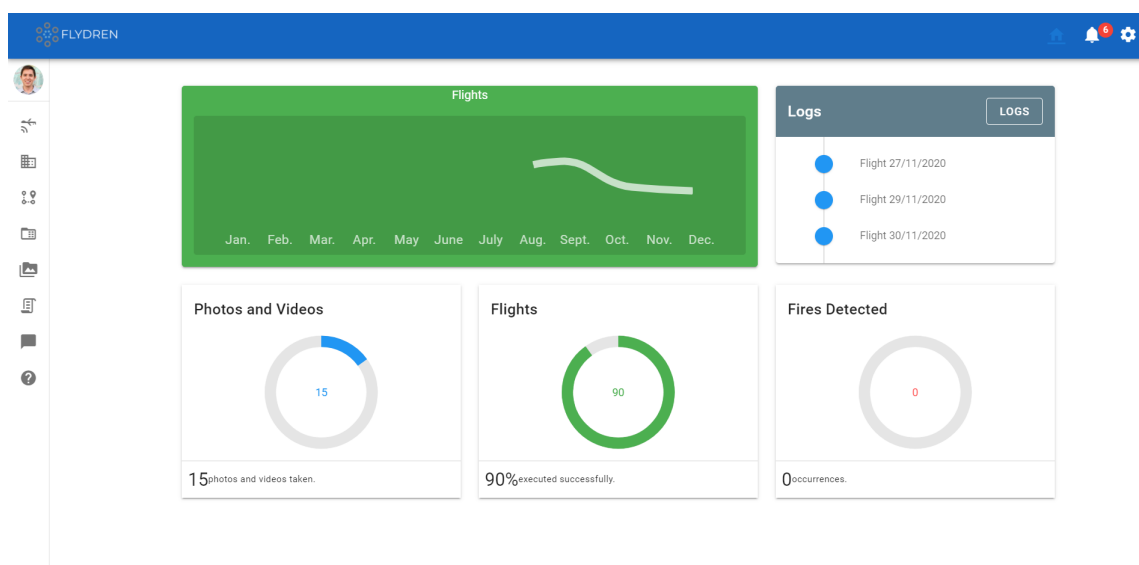


FIGURE 4.6. Dashboard Page

- Edit: the user can configure missions, edit the drone data and manage the drone permissions.
- Master: the user is the owner of the drone and has access to all the features. He can delete the drone.

On the devices page, the user can register a new drone by clicking the dedicated button. The user will be asked for some data about the drone to be registered, which can be changed later. Some of the fields of the drone are the name, description, IP, and the

Chapter 4 *Results*

list of users who have access to this drone. Figure 4.8 shows the permissions list of the Sintra Mountains drone.

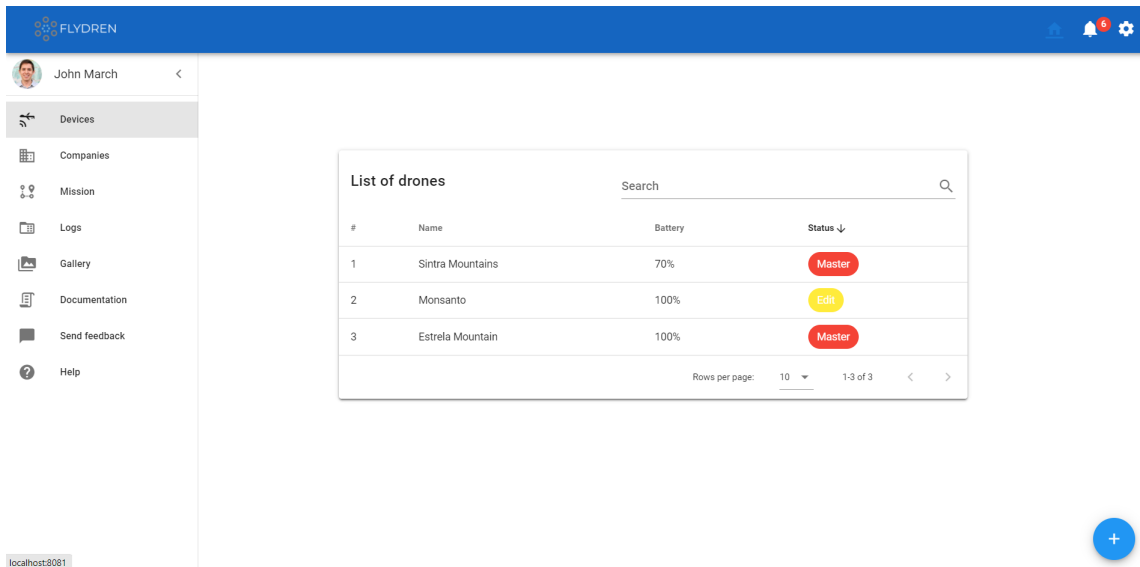


FIGURE 4.7. Drone List Page

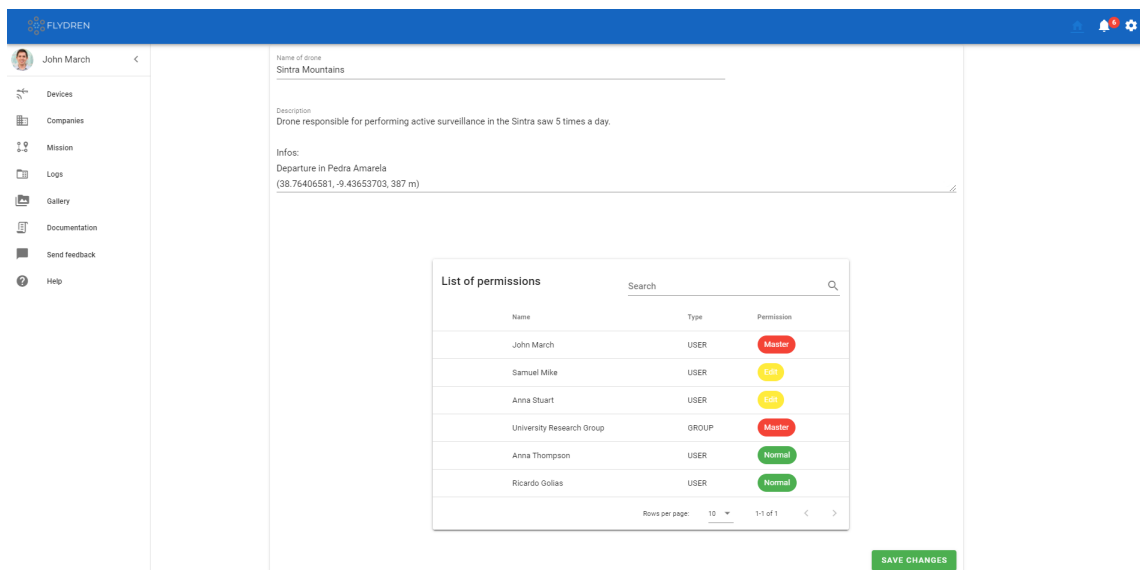


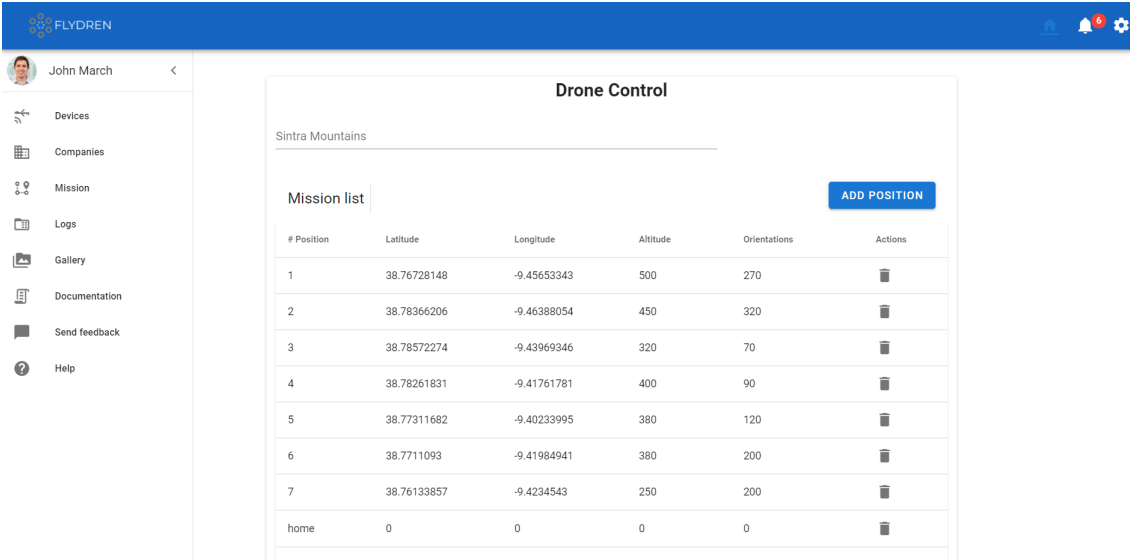
FIGURE 4.8. Drone Edit Page

4.2. Drone Control

The control of a device is only available to a user who has permissions to do it. Figure 4.9 shows how a mission for a drone is created. From the map that exists on the page, the user can draw the path that the drone will follow. For each point indicated on the map, the user must click on the extract point button to insert it in the table. The table contains all the points through which the drone will make the route. Each line of the

Chapter 4 Results

table has the coordinates of the point (longitude and latitude), the altitude of the drone and its orientation. Once the list of positions is finished, it is transformed into a file and sent to the drone.



# Position	Latitude	Longitude	Altitude	Orientations	Actions
1	38.76728148	-9.45653343	500	270	🗑️
2	38.78366206	-9.46388054	450	320	🗑️
3	38.78572274	-9.43969346	320	70	🗑️
4	38.78261831	-9.41761781	400	90	🗑️
5	38.77311682	-9.40233995	380	120	🗑️
6	38.7711093	-9.41984941	380	200	🗑️
7	38.76133857	-9.4234543	250	200	🗑️
home	0	0	0	0	🗑️

FIGURE 4.9. Drone Control Page

As soon as the drone is moving, on the map will appear the route that the drone is taking. Values measured by the drone are sent to the user. However, the values that each drone collects differ from drone to drone. This way, the user must indicate in the settings of the drone which message structure it returns, so that the information can be presented in a more user-friendly table.

For not having been able to use a real drone in the tests developed, open-source software was used that allowed simulating the drone. These softwares are the ardupilot and the SITL. The figures 4.10 represent the result of the communications made between the platform and the simulated drone, demonstrating the effectiveness of the project developed.

4.3. Fire Detection

As already mentioned in this dissertation, the results obtained were through the use of a simulated drone. Because it was not a real drone it was not possible to test or obtain results about the correct functioning of the whole project.

In the image 4.11 it is possible to see how the algorithm operates, as it has been previously demonstrated in other projects [2].

In order to obtain as many results as possible in this area, the sending of images to the image processing microservice was simulated. The results obtained with this test were

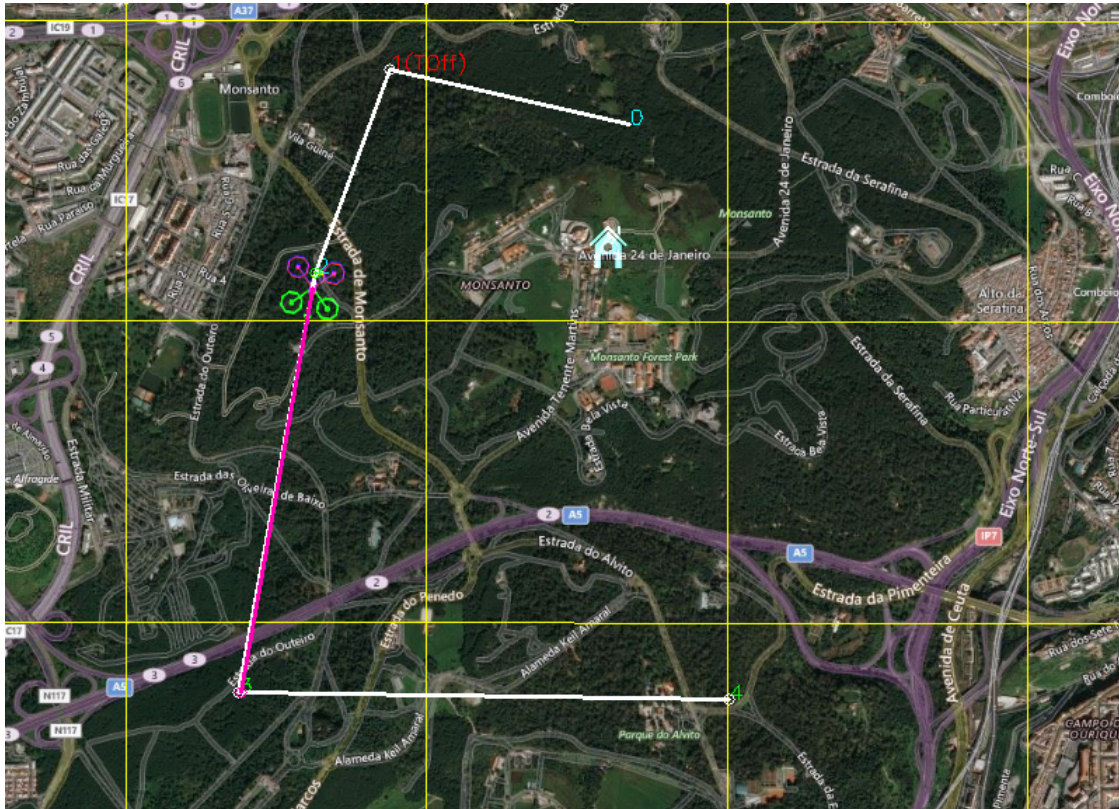


FIGURE 4.10. Drone Mission Simulation



FIGURE 4.11. Fire Algorithm [2]

like what had been obtained when using the algorithm in isolation and running locally. Thus, the effectiveness and efficiency of the image detection algorithm was not affected by its hosting in a cloud service or by the integration in a larger project.

CHAPTER 5

Conclusions

5.1. Main Conclusions

The main objective of this dissertation was the construction of a web platform capable of managing, monitoring and controlling drones in real time in order to detect forest fires, as a possible use case. The possibility of pre-programming the drones to perform a certain mission at a certain time was also one of the expected objectives for this dissertation. Finally, the integration of a machine learning algorithm with the ability to detect forest fires would be an added value for this project.

In general, the objectives proposed for this dissertation were accomplished and can be indexed and detailed in the following points:

- The development of a web platform that allows users to register their drones. The possibility of organizing users by groups and companies, allowing several users to have access to the same drone, was an added value implemented in this project.
- The architecture thought for the platform considered the possible escalation that can occur. The structure adopted allows the addition of new features without compromising the rest of the platform. The choice for a microservice architecture was fundamental for this possibility to exist.
- The research methodology and the development methodology were accomplished most of the time, as well as the good practices in software development.
- The control with the drone via internet, and through a web platform, was shown to be possible. Although it was only possible to use a simulated drone, it does not invalidate the work developed which, in theoretical terms, should not have any influence on the intended result.
- The integration of the machine learning algorithm in the platform was partially achieved. The fact that a real drone was not used in the system tests, does not ensure that the reception of images coming from the drone is working as desirable.

- The real-time control of the drone is not user-friendly. Although it is possible to trace the path of the drone using a map, the communication with the drone is done via commands that the user needs to understand.

5.2. Future work

Some improvements to the project developed in this dissertation should be made, as well as the development of new features:

- The tests of this system should not be restricted only to simulated drones. Tests should be made with real drones to validate the developed project.
- The communication between the drone and the platform must be protected. No security protocol has been implemented in this communication so the content of the messages between both parties is susceptible to external interference.
- The drone control page must be improved and made more user-friendly. The possibility to schedule a route for the drone to travel should also be developed and it should not be necessary for a user to give the exit order.
- It is expected that the drone sends images in real time to the platform. These images should be made available to users through the website. In addition, the video received must be forwarded to the image processing microservice for analysis.
- The upgrade of this system to a more complete and secure cloud server should be considered. While all security guarantees have been taken, the services provided by platforms such as Azure or AWS bring numerous advantages.
- The platform developed allows for the addition of new functionality or new algorithms with great ease. In order not to restrict this system to fire detection alone, its expansion should be considered.

References

- [1] X. Yang, S. Xiong, H. Li, X. He, H. Ai, and Q. Liu, "Research on Forest Fire Helicopter Demand Forecast based on Index Fuzzy Segmentation and TOPSIS," *2019 9th International Conference on Fire Science and Fire Protection Engineering, ICFSFPE 2019*, no. 2018, 2019.
- [2] C. Saraiva, "Automatic Fire Detection System: Autonomous Drone Supported by Artificial Intelligence," Master Thesis, ISCTE - Instituto Universitário de Lisboa, 2019.
- [3] "Global Fire Data," Accessed at 2020-08-29. [Online]. Available: <https://www.globalfiredata.org/regional.html>
- [4] K. Hartmann and K. Giles, "UAV exploitation: A new domain for cyber power," *International Conference on Cyber Conflict, CYCON*, vol. 2016-Augus, pp. 205–221, 2016.
- [5] M. Kalske, "Transforming monolithic architecture towards microservice architecture Department of Computer Science," Master Thesis, University of Helsinki, 2017. [Online]. Available: <https://helda.helsinki.fi/handle/10138/234239>
- [6] F. Ponce, G. Marquez, and H. Astudillo, "Migrating from monolithic architecture to microservices: A Rapid Review," *Proceedings - International Conference of the Chilean Computer Science Society, SCCC*, vol. 2019-Novem, 2019.
- [7] O. Al-Debagy and P. Martinek, "A comparative review of microservices and monolithic architectures," *18th IEEE International Symposium on Computational Intelligence and Informatics*, pp. 149–154, 2019.
- [8] L. De Lauretis, "From Monolithic Architecture to Microservices Architecture," *Proceedings - 2019 IEEE 30th International Symposium on Software Reliability Engineering Workshops, ISSREW 2019*, pp. 93–96, 2019.
- [9] T. C. Santos, A. Rodrigues, and P. Sardinha, "Adopting Microservices Migrating a HR tool from a monolithic architecture," Master Thesis, IST - Instituto Superior Técnico de Lisboa, 2018.
- [10] K. Stenroos, "Microservices in Software Development," Bachelor Thesis, Metropolia University of Applied Sciences, 2019. [Online]. Available: <https://www.theseus.fi/handle/10024/171777>
- [11] M. Zaymus, D. Author Zaymus, S. Salmikangas, and J. Assigned by Solteq Oyj, "Decomposition of monolithic web application to microservices School of Technology, Communication and Transport Degree Programme in Information and Communications Technology Decomposition of monolithic web application to microservices Degree programme Info," Bachelor Thesis, JAMK University of Applied Sciences, 2017. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/131110/Zaymus_{_}thesis.pdf
- [12] "JavaScript Mozilla Documentation," Accessed at 2020-04-15. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

References

- [13] “JavaScript Devdocs Documentation,” Accessed at 2020-04-15. [Online]. Available: <https://devdocs.io/javascript/>
- [14] Z. Brabec, T. Cerny, and J. Sebek, “On metadata extension to derive data presentations with angular 2,” *2016 6th International Conference on IT Convergence and Security, ICITCS 2016*, 2016.
- [15] M. D. Sousa and A. Gonçalves, “humanportal – Um caso de estudo usando React.js,” *15th Iberian Conference on Information Systems and Technologies (CISTI)*, no. June, pp. 24–27, 2020.
- [16] J. Voutilainen, “Evaluation of Front-end JavaScript Frameworks for Master Data Management Application Development,” Bachelor Thesis, Metropolia University of Applied Sciences, 2017. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/138668/Voutilainen_{_}Jaakko.pdf?sequence=1
- [17] “TypeScript Documentation,” Accessed at 2020-05-10. [Online]. Available: <https://www.typescriptlang.org/docs/>
- [18] “Why JavaScript - Serokell,” Accessed at 2020-06-03. [Online]. Available: <https://serokell.io/blog/why-typescript>
- [19] “React Documentation,” Accessed at 2020-03-05. [Online]. Available: <https://reactjs.org/>
- [20] “Angular Documentation,” Accessed at 2020-03-05. [Online]. Available: <https://angular.io/>
- [21] “Vue.js Documentation,” Accessed at 2020-03-05. [Online]. Available: <https://vuejs.org/>
- [22] “Angular vs React vs Vue Trends,” Accessed at 2020-04-19. [Online]. Available: <https://www.npmtrends.com/angular-vs-react-vs-vue>
- [23] “Front-end frameworks popularity,” Accessed at 2020-08-30. [Online]. Available: <https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190>
- [24] “Node.js Documentation,” Accessed at 2020-04-15. [Online]. Available: <https://nodejs.org/en/>
- [25] L. P. Chitra and R. Satapathy, “Performance comparison and evaluation of Node.js and traditional web server (IIS),” *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies, ICAMMAET 2017*, vol. 2017-Janua, pp. 1–4, 2017.
- [26] N. Santos, A. Raimundo, D. Peres, P. Sebastião, and N. Souto, “Development of a software platform to control squads of unmanned vehicles in real-time,” *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, no. 1, pp. 1–5, 2017.
- [27] A. S. Lima Raimundo, “Autonomous Obstacle Collision Avoidance System for UAVs in Rescue Operations,” Master Thesis, ISCTE - Instituto Universitário de Lisboa, 2016.
- [28] Reg Austin, *Unmanned Aircraft Systems*, wiley ed., 2010.
- [29] S. Palanisamy and P. Suvithavani, “A survey on RDBMS and NoSQL Databases MySQL vs MongoDB,” *2020 International Conference on Computer Communication and Informatics, ICCCI 2020*, 2020.
- [30] G. Ongo and G. P. Kusuma, “Hybrid Database System of MySQL and MongoDB in Web Application Development,” *Proceedings of 2018 International Conference on Information Management and Technology, ICIMTech 2018*, no. September, pp. 256–260, 2018.

References

- [31] M. Sharma, V. D. Sharma, and M. M. Bundele, "Performance Analysis of RDBMS and No SQL Databases: PostgreSQL, MongoDB and Neo4j," *3rd International Conference and Workshops on Recent Advances and Innovations in Engineering, ICRAIE 2018*, vol. 2018, pp. 22–25, 2018.
- [32] W. Vogels, "Eventually Consistent - Revisited," Accessed at 2020-06-14, 2008. [Online]. Available: <https://www.allthingsdistributed.com/2008/12/eventually-consistent.html>
- [33] "MySQL Documentation," Accessed at 2020-07-23. [Online]. Available: <https://dev.mysql.com/doc/>
- [34] "MariaDB Documentation," Accessed at 2020-09-01. [Online]. Available: <https://mariadb.com/kb/en/documentation/>
- [35] "Oracle DB Documentation," Accessed at 2020-03-12. [Online]. Available: <https://docs.oracle.com/en/>
- [36] W. Khan, W. Ahmad, B. Luo, and E. Ahmed, "SQL database with physical database tuning technique and NoSQL graph database comparisons," *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, pp. 110–116, 2019.
- [37] "DB Engines - MongoDB, OracleDB, MySQL," Accessed at 2020-11-15. [Online]. Available: <https://db-engines.com/en/system/MongoDB%3BMySQL%3BOracle>
- [38] Z. Kaprocki, V. Pekovic, and G. Velikic, "Combined testing approach: Increased efficiency of black box testing," *2015 IEEE 1st International Workshop on Consumer Electronics - Novi Sad, CE WS 2015*, pp. 76–78, 2017.
- [39] J. Vitovec, "Optimization of Recommender Systems," Bachelor Thesis, Czech Technical University in Prague, 2017.
- [40] "Jest Documentation," Accessed at 2020-05-21. [Online]. Available: <https://jestjs.io/docs/en/getting-started>
- [41] "Mocha Documentation," Accessed at 2020-05-03. [Online]. Available: <https://mochajs.org/>
- [42] "Chai Documentation," Accessed at 2020-05-03. [Online]. Available: <https://www.chaijs.com/>
- [43] K. F. Tomasdottir, M. Aniche, and A. Van Deursen, "Why and how JavaScript developers use linters," *ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pp. 578–589, 2017.
- [44] K. F. Tomasdottir, M. Aniche, and A. Van Deursen, "The Adoption of JavaScript Linters in Practice: A Case Study on ESLint," *IEEE Transactions on Software Engineering*, vol. 46, no. 8, pp. 863–891, 2020.
- [45] "TSLint Documentation," Accessed at 2020-04-30. [Online]. Available: <https://palantir.github.io/tslint/>
- [46] D. Spinellis, "Git," *IEEE Software*, vol. 29, no. 3, pp. 100–101, 2012.
- [47] F. F. Blauw, "The use of git as version control in the South African software engineering classroom," *2018 IST-Africa Week Conference, IST-Africa 2018*, pp. 1–8, 2018.
- [48] N. Kobayakawa and K. Yoshida, "How GitHub Contributing.md Contributes to Contributors," *Proceedings - International Computer Software and Applications Conference*, vol. 1, pp. 694–696, 2017.

References

- [49] Y. Yu, Y. Yang, J. Gu, and L. Shen, "Analysis and suggestions for the security of web applications," *Proceedings of 2011 International Conference on Computer Science and Network Technology, ICCSNT 2011*, vol. 1, pp. 236–240, 2011.
- [50] M. Bang and H. Saraswat, "Building an effective and efficient continuous web application security program," *2016 International Conference on Cyber Situational Awareness, Data Analytics and Assessment, CyberSA 2016*, 2016.
- [51] Divyaniyadav, D. Gupta, D. Singh, D. Kumar, and U. Sharma, "Vulnerabilities and security of web applications," *2018 4th International Conference on Computing Communication and Automation, ICCCA 2018*, pp. 1–5, 2018.
- [52] M. Kogce and N. E. Siseci, "A New Approach to Security of NTP via SSL Certificates," *1st International Informatics and Software Engineering Conference: Innovative Technologies for Digital Transformation, IISEC 2019 - Proceedings*, 2019.
- [53] K. A. Shaikh, A. Karthik Bhat, and M. Moharir, "A Survey on SSL Packet Structure," *2nd International Conference on Computational Systems and Information Technology for Sustainable Solutions, CSITSS 2017*, pp. 268–272, 2018.
- [54] "Amen.pt," Accessed at 2020-04-03. [Online]. Available: <https://www.amen.pt/>
- [55] "PassportJs Documentation," Accessed at 2020-04-26. [Online]. Available: <http://www.passportjs.org/>
- [56] "PassportJs Repository," Accessed at 2020-04-27. [Online]. Available: <https://github.com/jaredhanson/passport>
- [57] M. Jones, Microsoft, J. Bradley, P. Indentity, N. Sakimura, and NRI, "JSON Web Token (JWT)," RFC Editor, RFC 7519, 5 2015. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7519.txt>
- [58] "JWT Documentation," Accessed at 2020-04-30. [Online]. Available: <https://jwt.io/>
- [59] A. Alkhulaifi and E. S. M. El-Alfy, "Exploring Lattice-based Post-Quantum Signature for JWT Authentication: Review and Case Study," *IEEE Vehicular Technology Conference*, vol. 2020-May, pp. 0–4, 2020.
- [60] R. Zeqiri, F. Idrizi, and H. Halimi, "Comparison of Algorithms and Technologies 2G, 3G, 4G and 5G," *3rd International Symposium on Multidisciplinary Studies and Innovative Technologies, ISMSIT 2019 - Proceedings*, pp. 5–8, 2019.
- [61] Y. Li, P. Ren, and Z. Xiang, "A Dual-Passband Frequency Selective Surface for 5G Communication," *IEEE Antennas and Wireless Propagation Letters*, vol. 18, no. 12, pp. 2597–2601, 2019.
- [62] R. Fielding, UC Irvine, Gettys, Compaq/W3C, J. C. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC Editor, RFC 2616, 6 1999. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc2616.txt>
- [63] C. Dionisio, "Distributed sensing solution for home efficiency tracking," Master Thesis, ISCTE - Instituto Universitário de Lisboa, 2019.
- [64] G. Simoes, "Smart System for Monitoring and Control of Swimming Pools," Master Thesis, ISCTE - Instituto Universitário de Lisboa, 2019.

References

- [65] M. Belshe, BitGo, R. Peon, Google Inc, T. M, and Mozilla, “Hypertext Transfer Protocol Version 2 (HTTP/2),” RFC Editor, RFC 7540, 5 2015. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7540.txt>
- [66] E. Rescorla and RTFM Inc, “HTTP Over TLS,” RFC Editor, RFC 2818, 5 2000. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc2818.txt>
- [67] I. Stanivuk, V. Bjelic, T. Samardzic, and D. Simic, “Expanding lua interface to support HTTP/HTTPS protocol,” *2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications, TELSIKS 2017 - Proceeding*, vol. 2017-October, pp. 407–410, 2017.
- [68] J. Klensin, “Simple Mail Transfer Protocol,” Internet Requests for Comments, RFC Editor, RFC 5321, 10 2008. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc5321.txt>
- [69] R. Sureswaran, H. Al Bazar, O. Abouabdalla, A. M. Manasrah, and H. El-Taj, “Active e-mail system SMTP protocol monitoring algorithm,” *Proceedings of 2009 2nd IEEE International Conference on Broadband Network and Multimedia Technology, IEEE IC-BNMT2009*, pp. 257–260, 2009.
- [70] S. Atoev, K. R. Kwon, S. H. Lee, and K. S. Moon, “Data analysis of the MAVLink communication protocol,” *2017 International Conference on Information Science and Communications Technologies, ICISCT 2017*, vol. 2017-December, pp. 1–3, 2017.
- [71] I. Nurmawati, A. Affandi, and I. Pratomo, “Evaluation of AIS and MAVLINK Protocol Performance,” *Proceedings - 2020 International Seminar on Intelligent Technology and Its Application: Humanification of Reliable Intelligent Systems, ISITIA 2020*, pp. 338–344, 2020.
- [72] A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, “Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey,” *IEEE Access*, vol. 7, pp. 87 658–87 680, 2019.
- [73] X. Liu, “Artificial intelligence and modern sports education technology,” *Proceedings - 2010 International Conference on Artificial Intelligence and Education, ICAIE 2010*, pp. 772–776, 2010.
- [74] A. Allouch, O. Cheikhrouhou, A. Koubaa, M. Khalgui, and T. Abbas, “MAVSec: Securing the MAVLink Protocol for Ardupilot/PX4 Unmanned Aerial Systems,” *International Wireless Communications and Mobile Computing Conference*, pp. 621–628, 2019.
- [75] “Ardupilot Website,” Accessed at 2020-07-14. [Online]. Available: <https://ardupilot.org/index.php>
- [76] “Ardupilot Dev Website,” Accessed at 2020-07-14. [Online]. Available: <https://ardupilot.org/dev/index.html>
- [77] “SITL Documentation,” Accessed at 2020-09-20. [Online]. Available: <https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html>
- [78] S. Wu and L. Zhang, “Using Popular Object Detection Methods for Real Time Forest Fire Detection,” *Proceedings - 2018 11th International Symposium on Computational Intelligence and Design, ISCID 2018*, vol. 1, pp. 280–284, 2018.
- [79] F. A. Al-Wassai and N. Kalyankar, “Major Limitations of Satellite images Firouz,” *Journal of Global Research in Computer Science*, vol. 4, pp. 51–59, 2013.

References

- [80] X. Zhao, H. Ji, D. Zhang, and H. Bao, "Fire Smoke Detection Based on Contextual Object Detection," *2018 3rd IEEE International Conference on Image, Vision and Computing, ICIVC 2018*, pp. 473–476, 2018.
- [81] A. M. AbdElrahim Mohamed and A. Elmustafa AbuElgasim, "Controlling Drone – using IOT platform," *2019 International Conference on Computer, Control, Electrical and Electronics Engineering (ICCCEEE19)*, pp. 6–9, 2019.
- [82] B. H. Lee, E. K. Dewi, and M. F. Wajdi, "Data security in cloud computing using AES under HEROKU cloud," *2018 27th Wireless and Optical Communication Conference, WOCC 2018*, pp. 1–5, 2018.
- [83] A. S. Muhammed and D. Ucuz, "Comparison of the IoT Platform Vendors, Microsoft Azure, Amazon Web Services, and Google Cloud, from Users' Perspectives," *8th International Symposium on Digital Forensics and Security, ISDFS 2020*, 2020.
- [84] W. S. Blackmer, "General Data Protection Regulation," 2016.
- [85] Y. Zhang, "Test-Driven Modeling for Model-Driven Development," *IEEE Software*, vol. 21, no. 5, pp. 80–86, 2004.