Universidade de Aveiro
2021

João Manuel
André Coelho

Logistic and Mobility Management Holistic Platform
Plataforma Holística de Gestão Logística e de Mobilidade

Universidade de Aveiro
2021

João Manuel André Coelho

Logistic and Mobility Management Holistic Platform
Plataforma Holística de Gestão Logística e de Mobilidade

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Hélder Troca Zagalo, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president	**Prof. Doutor Augusto Marques Ferreira da Silva** Professor Associado da Universidade de Aveiro
vogais / examiners committee	**Prof. Doutor Fernando Joaquim Lopes Moreira** Professor Catedrático da Universidade Portucalense Infante D. Henrique
	Prof. Doutor Hélder Troca Zagalo Professor Auxiliar da Universidade de Aveiro

agradecimentos

Em primeiro lugar, quero agradecer aos meus orientadores, Doutor Hélder Zagalo, ao Cristiano Vagos e João Pedrosa, pela confiança que em mim depositaram, por toda a dedicação e pela motivação incondicional.

Aos meus amigos e colegas, por todo o encorajamento e amizade que proporcionaram durante este longo percurso.

Ao Gabriel e ao Pedro, um especial obrigado pela motivação, que para além de amigos, são colegas de trabalho a quem eu desejo muito sucesso.

Por último, o meu profundo e sentido agradecimento à minha família, por toda a força e por estarem sempre presentes ao longo da minha vida académica. Sem eles, não sei se conseguiria terminar esta etapa da minha vida.

palavras-chave Plataforma holística, mapa interativo, tempo-real, mobilidade, gestão logística, visualização de dados

resumo A monitorização em tempo-real é um requisito essencial para todo o tipo de sistemas, cujo objetivo é fornecer feedback ao utilizador da forma mais rápida possível, facilitando assim a tarefa de gestão. Este trabalho define e demonstra algumas soluções similares ao produto final solicitado pela empresa, bem como a análise de diferentes tecnologias. A análise e o estudo de diversas frameworks e bibliotecas resultam num trabalho mais complexo e de melhor qualidade, melhorando a experiência do utilizador. Passa também pela definição de uma proposta de solução, o seu desenvolvimento e testes finais. Esta dissertação tem como principal objetivo o estudo, a análise e a implementação das melhores metodologias para o desenvolvimento de uma plataforma holística de gestão logística e de mobilidade. Esta plataforma fornece uma solução frontend-backend de forma a ser possível a visualização num mapa as suas operações, dando suporte em tempo-real à equipa de gestão logística da empresa.

keywords Holistic Platform, interactive map, real-time, mobility, logistic management, data visualization

abstract Real-time monitoring is an essential requirement for all types of systems, whose objective is to provide feedback to the user as quickly as possible, thus facilitating the management task. This research work defines and demonstrates some similar solutions to the final product requested by the company, as well as the analysis of different technologies. The analysis and study of different frameworks and libraries result in a more complex and better quality work, improving the user experience. It also involves the definition of a solution proposal, its development and final testing phases. This dissertation has as its main objective the study, analysis and implementation of the best methodologies for the development of a holistic platform for logistics and mobility management. This platform provides a frontend-backend solution in order to be able to visualize their operations on a map, providing real-time support to the company's logistics management team.

Contents

List of Figures

List of Tables

Acronyms

ATC	Air Traffic Controller
IT	Information Technology
WebGL	Web Graphics Library
GPU	Graphics processing unit
CPU	Central processing unit
LISATM	Lisbon Air Traffic Control Management System
NAV	Navegação Aérea de Portugal
TRAVIC	TRAnsit VIsualization Client
AMQP	Advanced Message queuing Protocol
JMS	Java Message Service
FIFO	First In First Out
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
UI	User Interface
SDK	Software Development Kit
JS	JavaScript
JSX	JavaScript XML
XML	Extensible Markup Language
OSM	OpenStreetMap
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
REST	REpresentational State Transfer
YAML	Yet Another Multicolumn Layout

Chapter 1

Introduction

1.1 Context

Real-time monitoring is an essential piece in every type of system. The goal of real-time monitoring is to deliver as soon as possible instant feedback to the user and thus, making it beneficial. It is a system that can have customizable real-time notifications and that allows the immediate at-a-glance view and ensures an instant notification of potential issues. Consequently, the troubleshooting of the potential issues is faster and more frequent giving enough time to fix them without causing major meaningful damage. These features also benefit from an economic standpoint since it can reduce the time of fixing serious issues, there is a better chance of keeping the productivity higher and cutting revenue losses.

Nobody likes to waste time whether it is watching a football game or monitoring a real-time aircraft system. When someone is watching a football game, they expect to see every little detail of the match. When someone buys something on Ecommerce online stores, they expect to know immediately if something is out of stock so an alternative can be selected. In our personal lives, we expect as well real-time media consumption and online experiences [1].

In many situations, there is the necessity of keeping track of everything that is going on. It is possible to observe the importance of the example of the air traffic controllers (ATC) who direct aircraft on the ground and through controlled airspace and that can provide advisory services to aircraft in non-controlled airspace. Their primary purpose is to prevent collisions, organize and expedite the flow of air traffic, and provide information and support for pilots as referred in [2].

Lastly, real-time monitoring also allows to help building historic data over time which can be beneficial to identify patterns, investigate issues more effectively and predict outcomes [1].

Figure 1.1: Some types of Real-time Monitoring [3].

1.2 LUGGit

LUGGit[1] is a technological mobility platform that offers luggage collection and delivery service in real-time in several cities, carried out by drivers (denominated Keepers). The multi-platform allows anyone through a mobile app or by integrating with third-parties to request a Keeper (driver) in real-time to pick up their luggage and deliver it at the place and time they choose.

A Keeper is the driver who collects, stores and delivers luggage back to the customers. After collection on the location chosen by the customer, the luggage is stored in a logistics center, as the period of time between collection and delivery may justify keeping the luggage stored, freeing up space in the Keeper's vehicle for new services. Drafts are a premature version of Requests. A Draft can be made by a user and serves the purpose of showing the price and the details of locations and times of the Request. The user can decide whether to proceed and convert the Draft into a Request or to not use the service. An Operation Area is the geographical determined location where operations take place. Requests and Drafts can only occur within its borders.

The entire process related to the service, from the client request to the delivery by the Keeper and everything in between, must be monitored in order to accompany in real-time all of its phases. This allows an easy way to obtain general and personalized views about the status of the operations.

LUGGit uses two mobile applications: the user application, which requests the service; and the application of the Keeper, who is responsible for carrying out the service.

The platform focuses on the transport and storage of the luggage, allowing users (the majority tourists and travellers) to request the baggage collection in real-time through its mobile application, scheduling its delivery by the time and place the user pretends. In this

[1]https://luggit.app/

way, the user will be able to travel without its luggage, removing the responsibility and the discomfort when carrying it.

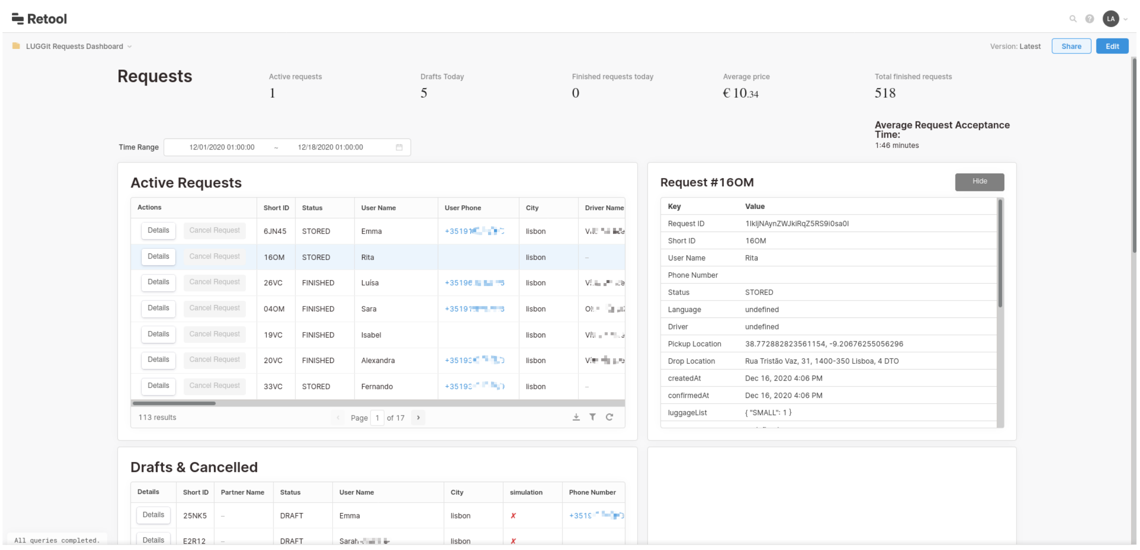
1.3 Problems and Challenges

The main problem of LUGGit is the lack of a good interface regarding the logistic operations of the company. Their current solution consists on a platform that is not customizable, disorganized and unresponsive, in regards to performance.

LUGGit currently does not have a platform capable of displaying the current operation of its fleet. It is not possible to show the map and knowing how the data analysis options are run and displayed. It is also not possible to know if the Keepers are synchronized.

As more data is required to be displayed and further logic ahead, using a third-party platform limits their usage, making it ineffective for custom integrations within the company's internal services. As a consequence of the rapid company's expansion, a solution providing a faster awareness of risks and errors in real-time during the operation was an important task.

The need to visualize useful data to improve the operations monitoring has become indispensable for this task. At this moment, LUGGit addresses the problem displaying their fleet and current Requests' data by using a logistic platform solution that is based on a drag and drop system called Retool[2].



Requests

Active requests	Drafts Today	Finished requests today	Average price	Total finished requests
1	5	0	€ 10.34	518

Average Request Acceptance Time:
1:46 minutes

Time Range 12/01/2020 01:00:00 ~ 12/18/2020 01:00:00

Active Requests

Actions	Short ID	Status	User Name	User Phone	City	Driver Name
Details Cancel Request	6JN45	STORED	Emma	+35191[redacted]	lisbon	[redacted]
Details Cancel Request	16OM	STORED	Rita		lisbon	–
Details Cancel Request	26VC	FINISHED	Luísa	+35196[redacted]	lisbon	[redacted]
Details Cancel Request	04OM	FINISHED	Sara	+35191[redacted]	lisbon	[redacted]
Details Cancel Request	19VC	FINISHED	Isabel		lisbon	[redacted]
Details Cancel Request	20VC	FINISHED	Alexandra	+35193[redacted]	lisbon	[redacted]
Details Cancel Request	33VC	STORED	Fernando	+35193[redacted]	lisbon	–

113 results Page 1 of 17

Request #16OM

Key	Value
Request ID	1IkljNAynZWJkiRqZ5RS9i0sa0l
Short ID	16OM
User Name	Rita
Phone Number	
Status	STORED
Language	undefined
Driver	undefined
Pickup Location	38.772882823561154, -9.20676255056296
Drop Location	Rua Tristão Vaz, 31, 1400-350 Lisboa, 4 DTO
createdAt	Dec 16, 2020 4:06 PM
confirmedAt	Dec 16, 2020 4:06 PM
luggageList	{ "SMALL": 1 }

Drafts & Cancelled

Details	Short ID	Partner Name	Status	User Name	City	simulation	Phone Number
Details	25NK5	–	DRAFT	Emma	lisbon	✗	+3519[redacted]
Details	E2R12	–	DRAFT	Sarah [redacted]	lisbon	✗	

All queries completed.

Figure 1.2: The Retool Platform Solution

Map coordinates are important features for analysis and for error detection and correction, for example, but in the context of the LUGGit platform, it is better to display the information in a way that is more easily understandable and recognizable. Visualizing this information in a map is more intuitive and can be better perceivable by anyone.

Also, having the possibility to visualize an end-to-end LUGGit Request directly on a map, displaying its key locations and entities that are responsible for the service in an interactive way, is less time consuming and more practical than searching on a table with many rows.

[2]https://retool.com/

LUGGit decided to propose this dissertation in order to solve this problem. The goal is to build a platform using open-source technologies, making it sustainable and highly customizable. Along with the eminent LUGGit's expansion, more information will have to be displayed in the dashboard simultaneously, which comes with a complex technological challenge of withstanding a high amount of elements.

The major challenge is to build a solution that better helps the company by allowing the operations team to know and view instantly the whole state of the operation.

Another challenge includes processing data that comes from LUGGit's API and goes directly to the dashboard. This part involves storing data in a cached system to retrieve data that is seemingly fast and fluid.

Dealing with new technologies I have never used before, is going to be challenging but a huge learning opportunity. These challenges consist of learning new technologies and programming languages that were not introduced in my academic career. This project will give me a better understanding of building a whole system, from the requirements to the development and the associated difficulties that come with it. In terms of personal growth, it will be highly beneficial, and at the same time, I will be able to help the company grow, as it is crucial in a startup.

1.4 Objectives

The main objective of this dissertation is to create and build a platform capable of being used in a real context that allows an effective way to provide a end-to-end visualization of the company's logistics operation, which operates in the context of real-time, in order to obtain a holistic view of the entire operation, as well as help in decision making through data analysis and visualization.

The solution currently used by LUGGit shows many sources of data including some internal ones: information about Keepers, partners, logistic centers, past and recent Requests and Drafts. The new solution must provide the same type of information, but displayed in a visual and a better organized way, gathering all data for real-time scenarios.

This monitoring must be shown in a platform that is easily readable and highly customizable for the purpose of allowing a close tracking of the informations related to the operations of LUGGit, and consequently facilitating the management and better decision making regarding to Keepers and services.

1.5 Document structure

The remainder of this document is organized as follows.

Chapter 2 presents an overview of background and related work regarding the topics covered by the work carried out, namely the importance of data visualization, real-time data visualization and , data processing, data visualization systems and related technologies required to build one system.

Chapter 3 presents the solution proposal and system design, architecture and its detailed elements and the system's wireframes.

Chapter 4 includes the development process, as well as the required tools and technologies used to build the solution. It also mentions features and how each component was built and some software and user tests to the platform.

Finally, chapter 5 presents the conclusions about this work, summarizing and discussing the main contributions. It is concluded with some ideas deemed relevant for future work.

Chapter 2

State of the Art

This chapter introduces the overview of holistic platforms of management in order to better understand what they represent and what they do.

Additionally, and considering the goals of this work, this chapter has the intention of describing the current knowledge about the topic matter through the analysis of similar or related published work. This analysis consists on debating the technologies of each solution, summary of recent research and its presentability.

2.1 Data Visualization and its importance

Human mind can understand and process visual information in a better and less time consuming way than a non-visual information (sound, for example). Our eyes are drawn to colours and patterns, making data visualization an ancient need. A team of neuroscientists from MIT has found that the human brain can process entire images that the eye sees for as little as 13 milliseconds. They also concluded that the fact that a person can do that at these incredible speeds, demonstrates that vision is finding concepts at all instants. During all day, the brain is trying to understand what we are seeing at each moment [4].

Since humans cannot translate binary code or quickly digest written information, data visualization helps to form a easier way to understand, highlighting the trends and outliers. A good graphical representation of data tells a story removing the noise from information and highlighting the useful part.

It is very common to think data visualization is a relatively modern technique used in technology but history has shown us that the earliest map-making and visual depiction, and later thematic cartography, was and still is important with tremendous applications in many fields, like science, medicine and many others. Many types of data visualization were important in order to make breakthroughs within the human race. Creating pictographic images, in Maya and Egyptian civilizations, to communicate within social classes and across generations, drawing maps to strategically conquer and explore more land (the most antique map dates from 2500 B.C) [5].

During the days when all roads led to Rome, the so-called Peutinger Map, 2.1, would have served as a handy guide to the Empire's transportation network. The Peutinger Map depicts the course of more 60,000 miles of Roman roads stretching from Western Europe to the Middle East. An additional section also shows India, Sri Lanka and other parts of Asia. Much like a modern travel guide, the map includes the locations of more than 500 cities along with some

3,500 other points of interest such as way stations, temples, forests, rivers and even spas [6].

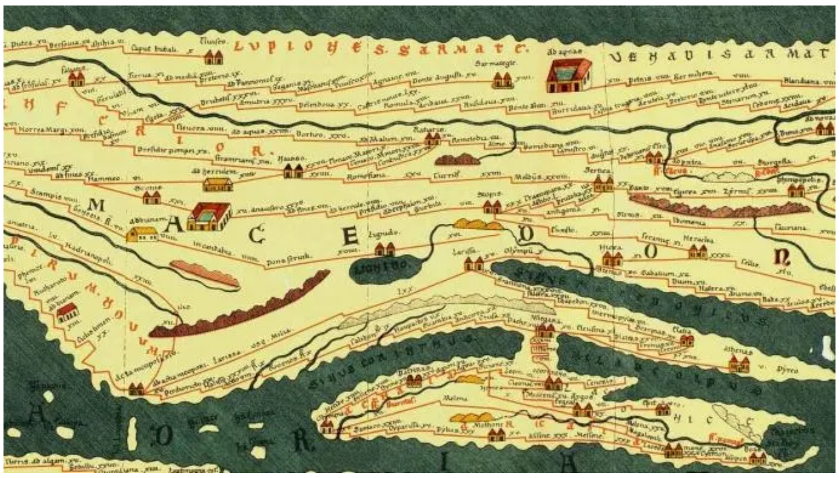


Figure 2.1: The Peutinger Map [6].

During the eighteenth century, the growing urban population in Europe started to attract the attention of local authorities. Decisions on the extension of existing cities or to build new cities became critical to make. Methods to assess and evaluate the situation were important and in 1782, a French mathematician named Charles Louis de Fourcroy published work using a visual diagram to classify towns by their surface area. The diagram is entitled Tableau Poléometrique and revolutionized the way to visualize data, using patterns, colors, and shapes [7].

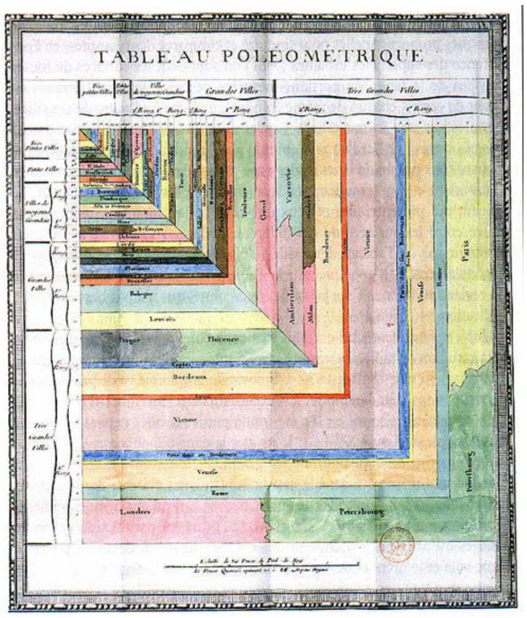


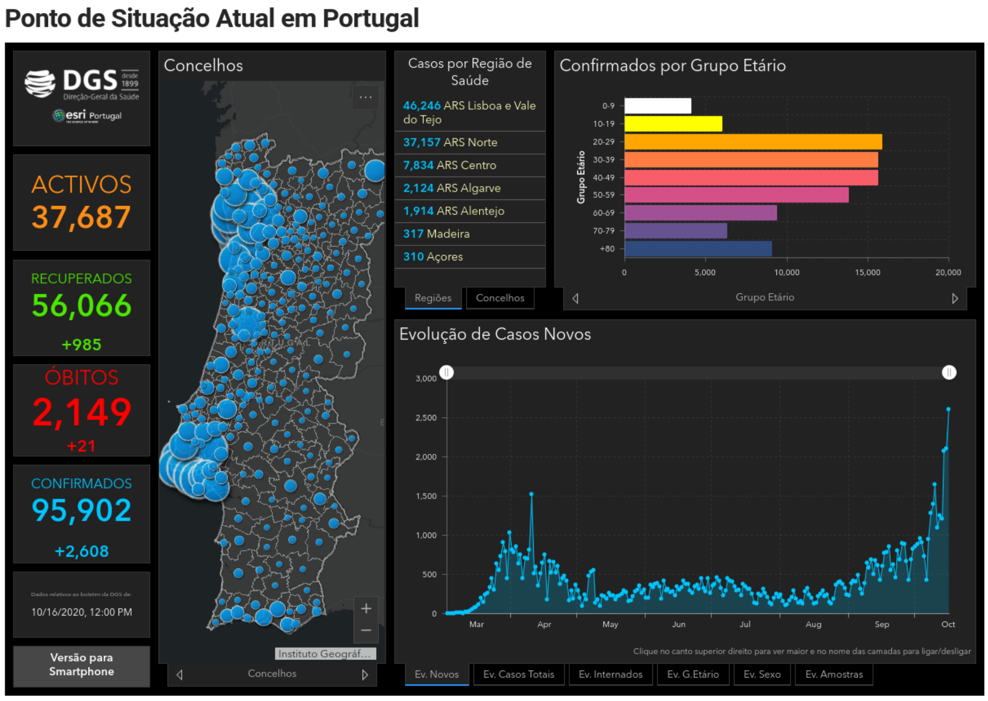
Figure 2.2: Tableau Poléometrique [7].

These and other breakthroughs contributed to the wide usage of data visualization today and the advancements in so many areas like mathematics, statistics, data collection and technology allowed for simpler and more efficient ways of visualizing data.

All organizations, from medicine to real estate, finance to government, use data visualization to drive their business decisions through data. They need to visualize the collected data in a way that shows what is relevant in the "bigger picture" [8]. Each organization type needs its own unique visualization strategy with a purpose in mind.

In healthcare, data analysis is present in a day-to-day basis. Physicians, nurses and other health professionals need to interact with health databases and interfaces, generate reports, and review patients' information. Their job was eased since data visualization appeared and it is even debatable that the health industry nowadays could not survive without it. Since every patient's record is digitalized, sometimes it is important to understand how many people in a country have a vaccine or how many will need a specific medicine and quickly visualize it. One study showed that the development of a dashboard to visualize electronic health record data resulted in a 65% reduction in time spent on data analysis in the first year alone, with further gains projected for the future [9]. Analysing large sets of data this way can save time and most importantly, help saving lives. In fast-paced medical settings, such as when physicians are reading Electroencephalographies (diagnostic test) in emergency rooms, the most important information needs to stand out quickly in a visualization, which can be obscured by a static of unnecessary data.

There is also a good example of the importance of data visualization in the middle of the coronavirus pandemic. An easy way to transmit the most important geographical and statistical informations about the daily report of the pandemic in Portugal to the general public, can be shown in the Figure 2.3. It is very intuitive to know which country areas have more people affected by the virus, the group age and how the number of new cases have been evolving since the beginning of the pandemic.



ACTIVOS 37,687

RECUPERADOS 56,066 +985

ÓBITOS 2,149 +21

CONFIRMADOS 95,902 +2,608

10/16/2020, 12:00 PM

Casos por Região de Saúde

46,246	ARS Lisboa e Vale do Tejo
37,157	ARS Norte
7,834	ARS Centro
2,124	ARS Algarve
1,914	ARS Alentejo
317	Madeira
310	Açores

Figure 2.3: Situation Report of Covid-19 in Portugal on Oct 16, 2020 [10].

Science aims to solve humankind problems and bring better solutions to our world. In order to do so, scientists must communicate with the population on quicker and simpler bases. A trend in science is to produce graphical abstracts to accompany the articles, allowing the reader to get an overview of the study in a glimpse. Software programs that enable scientists and designers to visualize microbiological forms and molecules by using interactive interfaces are becoming more common.

Data visualization can be surprising in the sense of realizing that one thing that is thought as not important is actually really important, and vice versa, especially in marketing. This also happens in other areas, like in the tech industry and health companies, for example. Without these tools, marketing would be just making a campaign and be done with it without any feedback, not knowing on what to improve next [11]. Some of the advantages that come with this technique are the possibility of knowing where customers are in the world, what social networks are driving the most traffic, whether or not customers are buying the product through a marketing campaign and how web traffic varies by time of day and day of the week. Data visualization may also be a great tool considering that when you have a statistic that proves that the product brings actual value and has improved the lives of its customers, that data will be a very important and valuable feature. It might be important to avoid the traditional pitch to sell, but instead share data with customers in the form of engaging data visualizations [11]. In fact, when companies are promoting a product, they want to present a simple approach to the customer, in order to let them know that it is the best product for them to purchase. The data insights from different test campaigns helps them to compare and apply the right one.

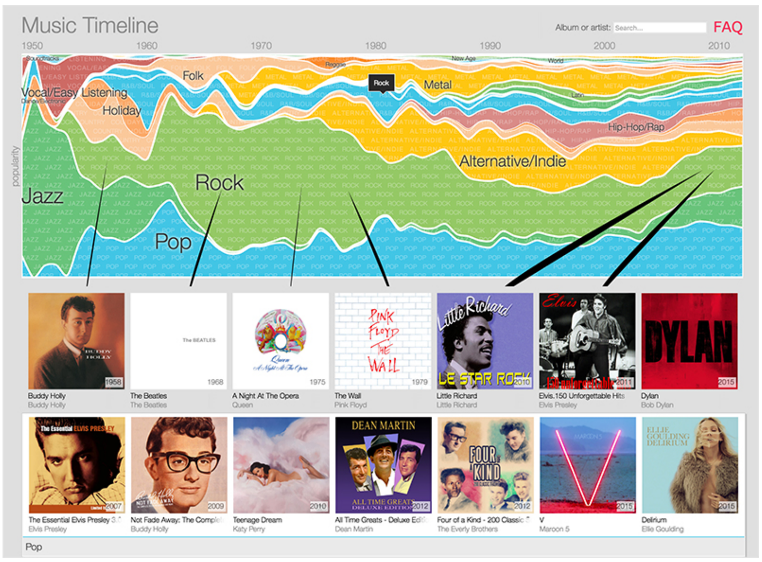


Figure 2.4: Advertising Campaign made in 2014 by Google [10].

In the advertising campaign 2.4 of the Google Play Music[1] app, data is used in an interactive way to show the most popular music genres among Google Play Music users. Using a fun and illustrative music timeline, users can click on each stripe to listen to different song genres, each with its own breakdown by artist and/or album.

[1]https://play.google.com/music/listen

There is a tremendous number of data visualization types, some of them being pie charts, maps, histograms, and tables and, for example, with the help of charts, trends can be identifiable over time, critical moments in the health of site traffic being able to react accordingly, and the biggest drivers of the business' success. Every company needs to maximize efficiency in order to stay competitive. Careful data analysis can offer critical insights on production and other vital measures.

As we can see, the communication of data in a visual manner comes with extensive usage cases due to this high number of types. Many companies rely heavily on data visualization to assess and improve their potential moves. Some of them employ a specific visual communication team to illustrate their data usually with custom interactive dashboards. Hiring a team to make the most of the available data is necessary in order to bridge the gap between analyst and knowledge as a simple and fast way of describing large data sets and complex phenomena. Since the data is now easily visualized by anyone, the analyst does not require to have deep knowledge on what is behind the data, thus being able to recognize patterns and hazards more efficiently.

Data visualization allows for a more efficient transfer of information within an organization which will serve to make a business more efficient. Data information is a really important feature for companies [12] and it has been proved to be the difference between the success and failure of a business. Using an efficient, effective and capable business intelligence tool that offers data visualization capabilities is a non-negotiable for companies looking to succeed into the future [12].

2.2 Static Reporting vs Real-time Reporting

With the introduction of new technologies, the real-time systems tend to be the ones that are most used globally. Static systems include static information or a set of resources that are generated periodically **??**. A static report illustrates trends, data and information over a predetermined period and it is usually used for the purposes of historical data analysis.

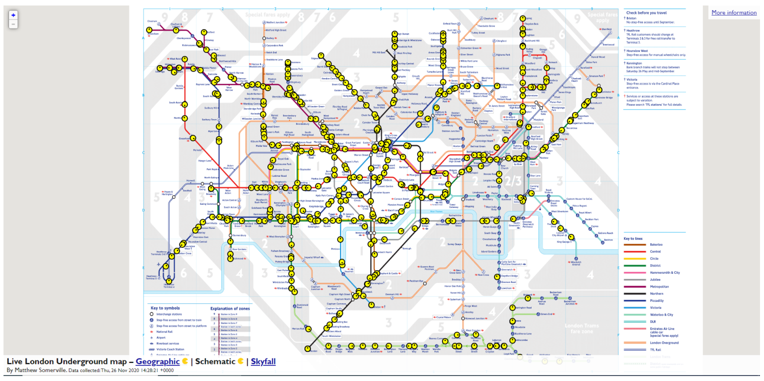
Contrarily, dashboards on real-time reports are dynamically created and they offer a continuously access to the most up to date information while enabling the user to interact with data through functionalities, such as interactive features, for example. These dashboards are continually refreshed and most of the dynamic real-time platforms are powered by machine learning capabilities, meaning that they are intuitive and live decision-making.

Real-time data is important when decisions in a company may be need to be made in seconds and there is a need to track and update information in the least time possible and that is why it the most system used by companies. It enables more accessibility to displayed data information, as the users can log into a dashboard from anywhere across multiple devices for instant analysis and more scalability, as it is possible to improve and update information over time.

2.3 Real-time Data Visualization

A variety of live public transit maps were developed recently, displaying the actual positions of certain vehicles. Several other transit agencies provide small live maps for their service range.

Live maps for various kind of vehicles are available nowadays in form of web applications, like for example the flight radar for planes and live train movement visualizations. There are also live maps for local traffic; for instance, subways in Munich, or tubes and buses in London. Several transit agencies provide position visualizations of their vehicles. One transit agency that displays a live map is the London underground[2]. It was built in August 2018 by Matthew Somerville[3], providing three map variants: geographic, skyfall and schematic.



Key to symbols
Explanation of zones
Live London Underground map – Geographic | Schematic | Skyfall
By Matthew Somerville. Data collected: Thu, 26 Nov 2020 14:28:21 +0000

Figure 2.5: Real-time London Underground Schematic[4] of Nov 26, 2020 [13].

Unfortunately, most existing live maps are either restricted to a very small area or to a certain type of vehicle (e.g. bus, metro). Moreover, most live maps do not feature smooth vehicle movements, but vehicles "jump" in certain intervals. However, global live maps would be beneficial in many aspects. From an operator's point of view this would provide a useful tool to supervise the complete system, track single vehicles, and observe non-scheduled stops or delays immediately. The coverage of an area (at a certain time) can also be estimated well if all vehicles passing through are displayed and also for evaluating how well-synchronized vehicles from different agencies can be. From a user perspective, the easy retrieval of all vehicles that are near-by can help to decide for a certain transportation mode or e.g. a particular bus to take. Also it can tell if you should hurry to catch your selected bus. In combination with a route planner, a live map can inform about delays of relevant vehicles neatly, and may give a hint about alternative transfers [14].

A person is needed to make sure everything is running as smoothly as possible and a platform with real-time information allowing easy access to edit some operations is essential. The most important feature that a visualization must have is that it should be interactive, which means that users should be able to interact with the visualization and personalize it. Whether be it time-frame updating or choosing which data to display, visualization must be adaptable to users' needs using relevant information.

One example of a real-time platform that integrates data across private and public clouds is Striim[5]. It is an end-to-end and enterprise-grade platform that delivers instant insights from high-volume and high-velocity data. It offers interactive and live dashboards with automatic refresh. It enables to view and compare historical data and keyword search on streaming data. The platform also ingests and processes structured and unstructured data from databases, log files, message queues and sensors.

[2]https://tfl.gov.uk/
[3]https://dracos.co.uk/
[5]https://www.striim.com/

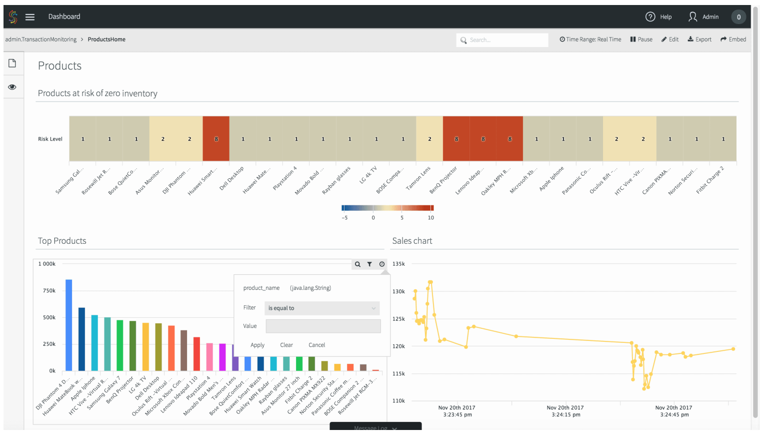


Figure 2.6: Dashboard of Striim Platform [15].

2.3.1 Advantages of Real-Time Data

In the sections above, real-time data has been described as one of the most important features considering using it in many areas, like in companies and in marketing, for example. It can have a very important impact on the outcome of a project and it is possible to have many advantages using real-time data [16]. Some of them are described below:

- Productivity

  Projects with real-time data visualization will have a better productivity, as the information flow and the decision-making is faster. The time that is saved can then be used in other areas that are more delayed, which will lead to a more productive outcome.

- Money

  Real-time data makes companies saving money. Instead of writing information data by pen and paper and using physical resources to do it, it can be used a real-time information technology solution. Reorganizing strategies and projects to solutions with real-time data will lower projects costs and flow of resources.

- Accuracy

  Inserting, updating and removing data manually can lead to human errors. These mistakes can be avoided with real-time data technologies. If the information is created, updated and shared in real-time, the information data will be more accurate and less prone to errors.

- Time

  Using solutions with real-time information will save time for workers and projects. Recording data in a project will save time to the worker, as the data will be created in less time.

- Decision-making

  If there is an error on information data, the flow of communication and information between workers and supervisors must be faster. With real-time data, decision making is improved and allows to make a better and faster decision with more precise results.

2.4 Data Processing in Data Visualization

Data processing is the conversion of data into usable and desired from meeting ones' needs. Processing data can be done manually or automatically, for example, with computers and the output can be plain text files, graphs and charts, tables, vector and image files, audios or other formats. These output forms depend on the software or method of the data processing used [17].

Even though it seems as simple as using the provided data to show what is pretended in the dashboard, it can be rather complex. Data often comes in a raw form and it is noisy, inaccurate, incomplete and sometimes contradicting. Therefore, it is important to familiarize with the process to clean and process data, to extract useful information out of it. The raw data is collected, filtered, sorted, processed, analyzed, stored and then presented in a readable format.

The collection of raw data is the first step of the data processing cycle and it is carried out across a variety of data types [18]. Data preparation is the process of sorting and filtering the raw data to remove unnecessary and inaccurate data. In the third step, the raw data is converted into a machine readable form and it is fed into the processing unit, usually in the form of an input source. The raw data is subjected to various data processing methods using algorithms in order to generate a desirable output. The data is finally transmitted and displayed to the user in a readable form like graphs, dashboards, video, documents, etc. The last step of this cycle is the storage, where data is stored for further use [19].

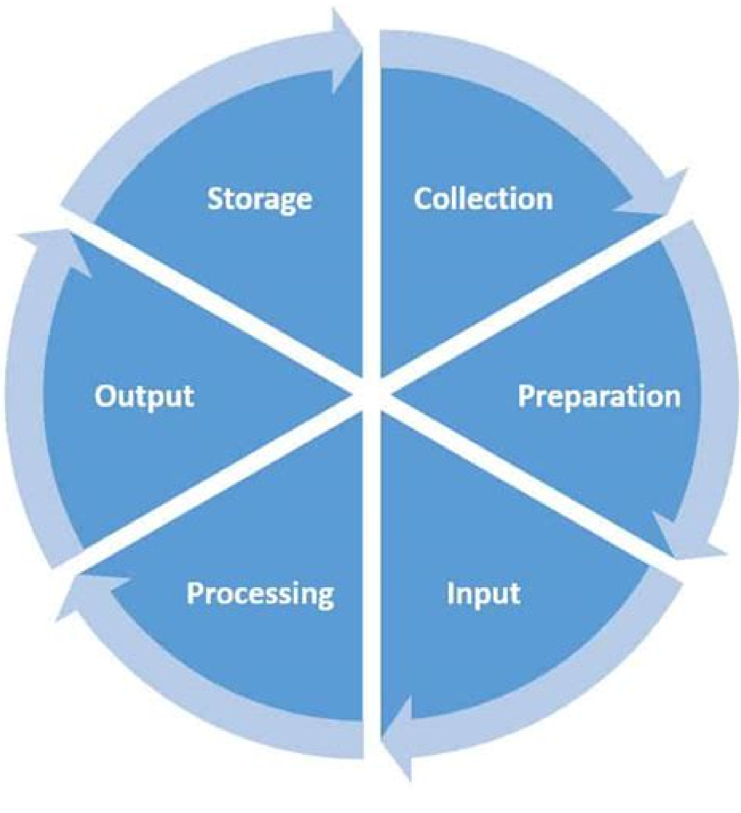


Figure 2.7: Data pre-processing steps [20].

2.4.1 Types of Data Processing

There are many methods and techniques used in data processing. The method used depends on the requirements, software and hardware used and time availability [17]. Some of them are described below:

- Batch Processing

  It is an efficient and sequential offline processing, where the information that is going to be organized is sorted into groups. The information data is processed by the order received, which helps in reducing the processing cost.

- Real-Time Processing

  It is a method that responds very quickly to the signals to acquire and process information. Although this method has a higher processing cost than the batch processing method, the results are displayed immediately and the time saved is maximum.

- Online Processing

  It needs an Internet and equipment attached to a computer. In this method, the data information is processed at the same time that it is received. It is mostly used for information recording and research.

- Distributed Processing

  It is mainly utilized by remote workstations connected to a big server, for example. Like ATMs, they use the same information and sets of instruction from a fixed software located at a specific place.

- Multiprocessing

  Multiprocessing is the most used type of data processing. It makes use of various available CPUs that can done tasks parallely within the mainframe, thus increasing efficiency and throughput.

- Time sharing

  This method consists of all users sharing the same CPU, but the time allocated to all the users might differ. It can be done by providing a terminal for their link to the main CPU and the time available is calculated by dividing the CPU time between all the available users as scheduled.

2.5 Data visualization Systems

Nowadays, the technology industry has a lot of real-time data monitoring systems. One of those examples is the platform used by air traffic control teams. The purpose is to visualize in real-time where the planes are and where they intend on landing [21]. This is greatly simplified when the visualization occurs in a map with animations and statistics in real-time. Being able to quickly check which flights are on time and which are not, it is not only useful but impressive. These teams try to maximize the overall efficiency of air space, minimizing the air traffic transfer and coordination costs between the different regional control centres and implementing better and more direct routes. This will lead to a more safety and security environment to all the logistic operations of an airport. The LISATM (Lisbon Air Traffic Control Management System) was developed by a NAV Portugal team and it is constantly changing towards up-to-date technology [22].

The University of Freiburg[6] together with geOps[7], a geospatial company, have created an interactive map of the world's major mass transit systems incorporating over 200 data feeds, showing the real-time movement of public transportation i.e. buses, trains, trams, etc. TRAVIC (Transit Visualization Client) provides movement visualization of transit data published by transit agencies and operators from all over the world. The movements are mostly based on static schedule data. The real-time data information is included in the visualization [14].

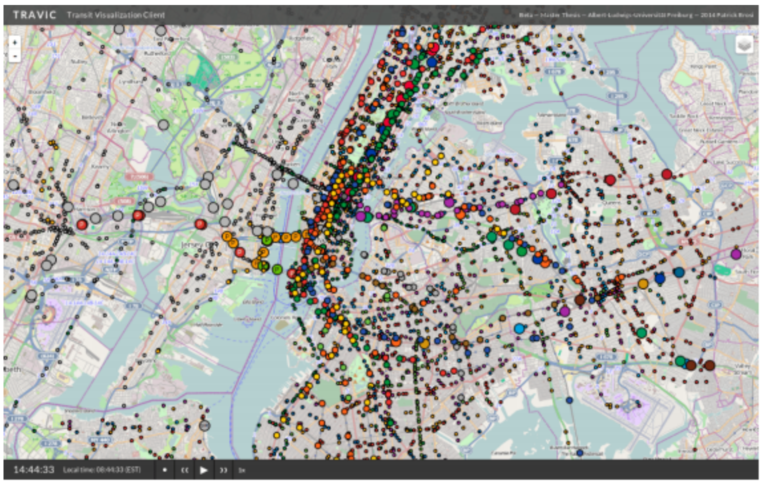


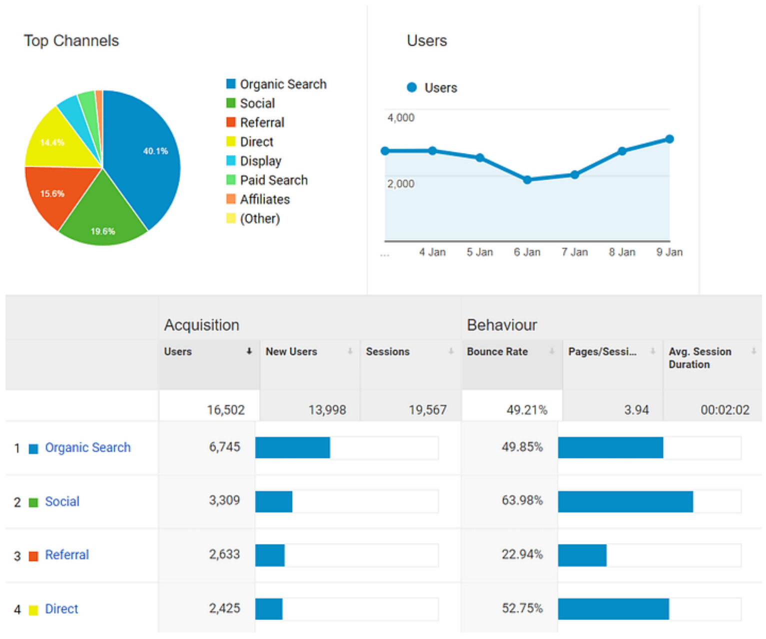
Figure 2.8: Real-time transit map of central New York in TRAVIC[23].

TRAVIC presents a client implementation that, in combination with a suitable client/server architecture, can display many thousands of smooth vehicle movements projected onto a map. TRAVIC makes use of Leaflet[8]. , an OpenSource JavaScript library for interactive web maps. Leaflet can handle most of the available map tile formats, but is mostly used with Google Maps or OpenStreetMap tiles.

Two well known business analytics services are from two of the biggest technology companies: Google[9] and Microsoft[10]. The former hosts a platform called Google Analytics[11] showcasing statistics on how users are using a certain platform. One of the most common use cases for using an analytics platform is to get a report of an online store traffic. This feature allows to monitor how many customers were attracted to the website from which source, like a campaign or a social media. It is also possible to know the users' geographic locations, how many pages are viewed from each location and how many people are actually buying from the ones that add to the cart. Also, measures from campaigns, events, social media posts' performance can be acquired and if they're driving traffic to your website or not. These statistics

[6]https://ad.informatik.uni-freiburg.de/
[7]https://www.geops.de/
[8]https://leafletjs.com/
[9]https://about.google/
[10]https://www.microsoft.com/
[11]https://marketingplatform.google.com/about/analytics/

can be helpful in analysing the user experience in the store.



	Acquisition			Behaviour		
	Users	New Users	Sessions	Bounce Rate	Pages/Sessi...	Avg. Session Duration
	16,502	13,998	19,567	49.21%	3.94	00:02:02
1 Organic Search	6,745			49.85%		
2 Social	3,309			63.98%		
3 Referral	2,633			22.94%		
4 Direct	2,425			52.75%		

Figure 2.9: Example of a Google Analytics Analytics dashboard [24].

Google Analytics tries to focus on colors and other data visualization types for an easier user experience. Any small company or freelancers, who are not tech-savy, will easily understand and make use of the traffic data on their website. This figure 2.9 shows where the most users are getting into the website and in which category, users spend the most time. It is easily readable that the most traffic comes from organic search but people that come from social media platforms, spend the most time.

The latter platform, Power BI [25], is a similar tool more focused on medium to large companies that enables users to monitor their business from every device at any time and shares insights from analysed data. Power BI is a cloud-based business analytics service and acts as a powerful as well as a flexible tool for connecting with and analyzing a wide variety of data.

Power BI's ease of use comes from the fact that it has a drag and drop interface. This feature helps to perform tasks like sorting, comparing and analyzing, very easily and fast. Power BI is also compatible with multiple sources, including Excel, SQL Server, and cloud-based data repositories which makes it an excellent choice for Data Scientists [26]. An innovative feature of this platform is the ability of building a custom data visualization dashboard on a drag and drop basis. Any person over viewing this tool, can choose which types of graphs to use, the amount of data to display and in which position to display it.

2.5.1 The Pipeline of Data Visualization

In the figure below, it is possible to visualize the steps of a typical iterative data visualization pipeline [27].

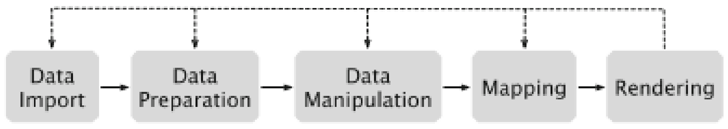


Figure 2.10: The data visualization pipeline [27].

The first step consists of retrieving data information from a specific source. Then, it is needed to prepare the imported data for visualization, by for example, normalizing all values and correcting errors. After the "Data Import" and "Data Preparation" steps, the data needs to be grouped and joined to select the information that it is required to be visualized. The "Mapping" process is to map the collected data to geometric primitives, like for example, points and lines, together with their attributes. Finally, it is needed to transform the geometric data into visual representations, like for example, plain text files, graphs and charts.

2.6 Related Technologies

In pursuance of obtaining a graphical interface displaying updated data in real-time within the framework of logistics operation, several technologies must be considered. Each technology must deliver detailed solutions as all important tasks make up a great overall solution. This section is presented with the necessary technologies to build a real-time holistic solution in order to provide a set of custom options in a web map framework. This system will give the real value in used data to issue a user friendly interface with lots of information.

2.6.1 Interactive Map Solutions

In an attempt to feature an easy-to-grasp visual compositing paradigm, interactive map solutions have to be considered. Apart from the need of displaying a web map, it is also necessary to show information on top of it, called layers. The layers enable developers and designers to quickly prototype through composition, while the framework offers comprehensive data handling and interaction mechanisms for building production-ready analytical solutions.

In terms of scalability, the solution should facilitate visualizing data at scale [28]. This should be done in a way that the visualization design presents relevant features and also filters out irrelevant data. The software should handle dynamic data, as to prevent unnecessary data processing while also providing the flexibility for fine grained controls over the procedure.

Usability is another feature that must be taken into consideration. It is hard to model the mapping between data and visual representations, especially when the dimension of the domain is large. A good composition model would help shorten the learning curve for decomposing an existing visualization design and reconstructing a new design that fits the platform's needs. When presented with a data set, one common practice for data visualization is to seek and apply existing designs for a quick prototyping. Besides the main functionality, the library should come with standalone examples that allow developers to plug in their data and search for salient patterns. These examples will not be the final product, but they are useful to start the exploration process of its technology.

A visualization framework should be a handy tool to facilitate the problem-solving process. Instead of building a solution with a steep learning curve, it is more desirable if prior knowledge of existing tools can be used. The framework should focus on the points that are not already solved and be interoperable with other existing solutions [29].

Considering all these points, some solutions that better suit LUGGit's needs will be referred below.

2.6.1.1 Deck.gl

Deck.gl[12] is a project, that was open sourced in 2016 by the company Uber[13], with the intention of visualizing all on-call cars in a metropolitan area, while monitoring hundreds of millions of moving points. It was specifically designed for exploring and visualizing data sets at scale and it lets us extract both historical and real-time insights from large and complex data sets, allowing us to think in 3D. Now it is a robust WEBGL-powered (Web Graphics Library) framework being highly customizable. All layers come with flexible APIs to allow programmatic control of each aspect of the rendering. All core classes such are easily extendable by the users to address custom use cases [30].

It is composed by an extensive documentation with lots of integrations and layers with multiple use cases and it is constantly getting new updates and improvements. It can be also integrated with other open-source mapping platforms, such as Mapbox and Google Maps.

2.6.1.2 Mapbox GL JS

Mapbox GL JS[14] is a JavaScript library for interactive, customizable vector maps based on WebGL. Its performance, real-time styling and interactivity features set the bar for anyone that wants to build fast and immersive maps on the web.

Layers are built with vector tiles and Mapbox styles. Rendering runs with WebGL so it is fast, even with a ton of data. Mapbox GL JS provides building blocks to add location features like maps, search, and navigation into any experience that is required.

This library is part of the cross-platform Mapbox GL ecosystem, which also includes compatible native SDKs (Software development kit) for applications on Android, iOS, macOS, Qt, and React Native.

2.6.1.3 Comparison between options

Mapbox GL JS became proprietary as of version 2.0 and cannot be used without accepting the Mapbox service terms. This move has produced shock waves in the geospatial industry and many organizations are using Mapbox GL JS directly or using a previous version and improving from it. Previously open source, Mapbox GL is a great solution, and could be used with many different data sources, so it became one of the best ways to render basemaps [31]. On the other hand, Deck.gl is supported by Uber and it is open source.

Both projects are maintained and updated regularly with an extensive amount of examples and detailed documentation. Outside of their official resources, Deck.gl leads with how to start guides as it is the most popular option. They use WebGL to render dynamically complex data by composing new visualizations into existing layers.

Despite of their lack of differences, Deck.gl and Mapbox GL JS are frequently used together to create compelling geospatial visualizations, each providing their unique advantages. They

[12]https://www.deck.gl/

[13]https://www.uber.com/

[14]https://www.mapbox.com/mapbox-gljs

serve somewhat different purposes but they do have similar features. Mapbox GL JS usually serves the purpose to build interactive maps, as Deck.gl is a library for visualizing large data sets.

2.6.2 Web Map Frameworks

The traditional way of displaying maps in a web browser is by downloading various images from a server and lay them side by side to make up a map. If any information on the map was changed, new images had to be downloaded and it was not done on the client. This meant lots of data transfer from the client and a slow presentation.

The introduction of WebGL[15] opens up a whole new world of delivering advanced graphics content to the end user in a web browser. Using this technology for displaying maps means only the source data is sent to the web browser where the map gets rendered using the device's GPU. This adds a number of benefits such as the ability of changing the map appearance on the client, adding new features to the map and often less data is transferred. However, it sets higher expectations of the client device's hardware as it needs to render the map at a high enough frame rate to not appear slow and unresponsive [32].

There are some web map frameworks, each one with its own advantages and disadvantages. A better knowledge of them provides us points to later decide the best one for this project.

2.6.2.1 Mapbox

The startup Mapbox was created as a part of Development Seed (team that creates data visualization solutions) in order to offer map customization for non-profit customers, in 2010.

Mapbox is the creator and a significant contributor to some open source mapping libraries and applications, including the Mapbox GL JS JavaScript library, the MBTiles specification, the TileMill cartography IDE, the Leaflet JavaScript library and the CartoCSS map styling language and parser.

Some known companies that used this map API are Foursquare, Lonely Planet, Facebook, the Financial Times, The Weather Channel and Snapchat. Since 2010, it has rapidly expanded the niche of custom maps, as a response to the limited choice offered by map providers such as Google Maps.

Developers will have access to comprehensive location information to create and customize dynamic and static maps, resources offered by the Mapbox tilesets. The styled maps can be embedded into web applications with the provided JavaScript library or to mobile apps.

The data is taken from open data sources, such as OpenStreetMap and NASA[16], and from purchased proprietary data sources, such as DigitalGlobe. The technology is based on Node.js[17], Mapnik[18], GDAL[19], and Leaflet[20].

Mapbox uses anonymised data from telemetry pings, such as Strava and RunKeeper, to identify likely missing data in OpenStreetMap with automatic methods, then manually applies the fixes.

[15]https://www.khronos.org/webgl/
[16]https://www.nasa.gov/
[17]https://nodejs.org/en/
[18]https://mapnik.org/
[19]https://gdal.org/
[20]https://leafletjs.com/

2.6.2.2 OpenStreetMap

Open Street Maps[21] is an open source project maintained by the OpenStreetMap Community that provides free editable maps of the whole world. This project's motivation was to make a restriction free mapping solution that can be used for commercial and non-commercial usage without any limitation. It was launched in 2004 by Steve Coast as a non-profit organization and its data is a perfect alternative compared to other solutions. Open Street Maps contains almost all the services that most web map frameworks offer with the advantage of customizability.

2.6.2.3 Google Maps

Google Maps[22] is a proud product of Google and is a major part of their services. The service was publicly launched in 2005 and since then has become a giant product with so many features, like for example user end API services such as routing, direction, map tiles design customization and many advanced layers.

The sudden increase of popularity of Google Maps began when people were able to integrate their map solution in their websites. Now it is common to see almost every single website with a section of Google Maps showing up their location in the view represented by some markers.

2.6.2.4 Comparison between options

There are many other web map integration APIs available to developers, but these three represent the most robust and flexible options. Regarding their similarities, they all have a good core map functionality, having static or dynamic maps. They show accuracy, in-depth geographic and local data in several views. The navigation displays real-time directions that account for traffic, route optimization and time of arrival estimation and they all have advanced search options, with automatic prediction and location-based suggestions. When competing to each other, the three web map solutions do not show any considerable advantage on these points.

Google Maps supports many types of data layers which include, for example, earthquakes, heat maps and fusion tables. It also provides data layers for traffic, transit and bicycle paths. Although Mapbox does not have as many data layers as the previous solution, the basic ones such as traffic and satellite are still provided. In OSM (OpenStreetMap) these layers are not part of the project and can only be fetched from Mapbox or third party services [33].

According to a developer who has had experience with both technologies, the possibility to combine the framework Mapbox GL JS with Mapbox, provides the best solution in terms of fast load speeds and performance compared to other solutions. Thanks to its tileset architecture and the optimization of Mapbox GL JS, it is expected that the integrated web map loads quickly and renders smoothly, particularly if complex sets of data are used [34].

Despite of new custom styles being rolled out, Google Maps API currently supports limited options for creating a unique look for an integrated map when compared with the other APIs. So contrary to Google Maps, there is OpenStreetMap, which is possible to download the whole solution and edit it without any restriction, due to being open source [33]. Finally, the Mapbox API provides the functionalities of adding objects, choosing the layers to be displayed and

[21] https://www.openstreetmap.org/

[22] https://cloud.google.com/maps-platform/

changing colors. Fonts can also be done with a few simple clicks and the resulting map is always a sight to behold [35].

Mapbox relies on collective mapping, with OpenStreetMap being its major source of data. While the OSM project has enjoyed remarkable growth over the past decade, its coverage in regions like India and China still needs some work [35]. Contrarily, Google Maps supplies excellent global and local data quality. Over the years of its existence, Google Maps has become the leader of mapping services and collected unfathomable volumes of information on the local level.

	Mapbox	**OpenStreetMap**	**Google Maps**
Data Layers	Incorporated (Not as many)	Incorporated (Not as many)	Incorporated (The most complete)
Direction Service	Mapbox Directions API	Third Party Services	Google Direction API
Performance	Fastest (when combined with Mapbox GL JS)	Intermediate	Slowest
Type	API provided by private company	Collaborative Open Source Project	API provided by private company
Map coverage	Weak Coverage	Less Coverage (like in United States and Japan)	More Coverage

Table 2.1: Comparative Table between Web Map Framework Solutions

2.6.3 Web Interface Frameworks

JavaScript[23] started out as a small scripting language in 1995 when the World Wide Web was still a new invention. When it all started, the requirements for this language were dynamism, a readily understood syntax and powerful . Now, for the eighth consecutive year, JavaScript has maintained its stronghold as the most commonly used programming language, according to a survey made by Stack Overflow[24] .

Before JavaScript was invented, web pages were shown with just HTML and CSS which meant the content was static. For example, if a website provided real-time updates of a sports event, a user could only see the updated information by refreshing the page. There was not a possible way of seeing the updates in real-time without having to reload the content manually. With the creation of JavaScript and its libraries and frameworks, the needed dynamism for real-time updates was introduced in websites. When changes occur to the underlying data, the framework renders the complete UI (User Interface) component. The displayed data should be up-to-date at any point in time. New concepts such as client-side rendering, server-side rendering and pre rendering came into discussion, changing the way websites are updated.

There are plenty of JavaScript solutions able to solve this problem. In this section, some of the most popular technologies that deliver reactions to the updated events in real-time are discussed and later compared.

[23]https://www.javascript.com/
[24]https://www.stackoverflow.com/

2.6.3.1 Angular

Angular[25] was originally created by Google in 2008. Back then it was referenced to as AngularJS and developed in plain JavaScript. This was at a time when the majority of websites where based on the multi-page application approach: when a user clicked on a link, the browser had to retrieve the requested HTML document from the server. Depending on the internet connection and the responsiveness of the server, it could take a fair amount of time until the user could view the new page. Gradually user devices increased in overall performance so that application logic could be executed in the browser. This led to the approach of single-page applications in many websites [36].

Angular, like many other frameworks, is component-based. This means that components are the main building blocks and they can display information, render templates and perform actions on data. The best practice suggests that components consist of three separate files: a HTML file for the template, a CSS file for the styling and a TypeScript[26] file for the controlling. By following this approach, a separation of concerns is implemented and it provides a more organized project structure and code [37].

2.6.3.2 React

React[27] is a JavaScript library for building user interfaces developed by Facebook. It earned a lot of popularity since its launch in 2013 being used by many websites like Netflix[28], Instagram[29], Airbnb[30], among many others. While React is often connected to web development, its core (the react package) is a standalone library and can be used in a variety of scenarios including native applications (iOS and Android). Only in combination with the react-dom, UIs for the web can be developed. JavaScript is an inherent part of the framework because it is the main development language [36].

The core of React is made of components and their composition. The overall goal is to transform a certain state of the application to a view which can be displayed in the browser.

2.6.3.3 Vue

Vue.js[31] is a front-end library written in JavaScript. This library allows an easily creation of web applications based on Model-View-ViewModel. Every application using the framework Vue.js consists of components that contain JavaScript code to be executed within the component, CSS style and HTML code that make the view layer.

Working with Vue.js begins with the creation of a new instance of Vue with the help of the constructor. This allows to visualize the changes that occur in a given component. Then, the Vue instance takes a template variable that will be used to get the view to render. Components, which are essentially parts of the user interface, help develop basic HTML elements and implement reusable code [38].

[25]https://www.angular.io/
[26]https://www.typescriptlang.org/
[27]https://www.reactjs.org/
[28]https://www.netflix.com/
[29]https://www.instagram.com/
[30]https://www.airbnb.com/
[31]https://www.vuejs.org/

Vue features a reactivity system that uses plain JavaScript objects and optimized re-rendering. Each component keeps track of its reactive dependencies during its render, so the system knows precisely when to re-render and which components to re-render.

Components extend basic HTML elements to encapsulate reusable code. At a high level, components are custom elements to which the Vue's compiler attaches behavior. In Vue, a component is essentially a Vue instance with pre-defined options.

2.6.3.4 Comparison between options

The solutions shown above were selected due to being the most used between the community in development and companies required to implement similar use cases. Nonetheless, other viable options were taken into consideration but they were not worth to pursue further. A benchmark comparison between these solutionswas made in great detail in 2020 [39] and a detailed comparison with compelling points between the referred frameworks was made in 2018 [36].

Regarding the differences of these frameworks, they do not differ very much as all of them are component based, which is already a relevant part of their overall philosophy. However, there are also some differences that can be subjective and it depends on the developer preference. These differences include strategies concerning file concepts (single versus multiple) or the main development language. While Vue is flexible, offering developers more than one way to write their code, Angular is more restrictive. Its official resources as well as most of the available tutorials and code snippets require the usage of TypeScript. React is also quite restrictive by enforcing the use of JSX (JavaScript XML) for developing even though it definitely is the most diverse approach of all three frameworks as JSX implies that HTML is strongly interconnected with JavaScript [36].

Vue and Angular share a relatively similar approach to structuring their components. They split up template (HTML), style (CSS) and logic (JS) although both offer the option to handle these parts either in one file or in three separate. The preference in Angular is the separation while Vue emphasises the single file approach even offering a special file extension for this purpose. Single file components have the advantage as it is easier for the developer to write components ensuring simplicity and re-usability. React focuses on having a single file architecture.

One important thing that is always mentioned when comparing these technologies is the classification. While Angular is the only fully-featured technology in this comparison which it is called a framework, React and Vue are just view libraries that are often named in the same contexts as frameworks because they offer best practices for a complete development setup. On one hand, while Angular provides a full set of homogeneous features and helpful strategies for development, it also leaves less room for own decisions. When a team or a company tends to adapt this framework, it means that a lot of mindset has to be adjusted to cover Angular's structure. On the other hand, React and Vue are very liberal and flexible. They can be used in various scenarios and technology can only be integrated to handle the view part of an application, for example. This freedom of choice, however, also requires a high level of responsibility and experience from the project leaders. All decisions have to be made on an elaborate basis and accounting for future proof [36].

Regarding the support topic, it is also important to analyze who supports these technologies. React is controlled by Facebook and Angular is controlled by Google. Despite being reassuring when companies are financially supporting a certain software, it has always to be considered that these companies are still based on profit which, of course, impacts their decision making. The project of Vue is completely supported by the open source community where the developers' requests and concerns have a higher chance of aligning with the users. Notwithstanding that it can be a great community, the risk of failure is higher in comparison with larger companies.

In terms of accessibility, Vue has a small learning curve being the easiest among the three. It focuses on well proven practices and offers a great documentation online which is co-developed with the community. React is easy to adapt for developers with a background in or an extensive knowledge of the JavaScript language as almost everything is adjusted to using JSX. Furthermore, the state management especially for larger projects can have a high complexity. Angular has the steepest learning curve among the discussed frameworks [36].

When talking about front-end development, performance is influential. However, it is very hard to find solutions that will work for every situation only based on speed. According to Levlin [39] in his thesis and to Camargos, Coelho, Aramuni, *et al.* [40], React has few weaknesses and performed consistently well in the benchmarks. From his benchmarks, React was the best performing of the three frameworks referred.

	Angular	**React**	**Vue**
Languages	TypeScript (Restrictive)	JSX(similar to HTML) (Flexible)	HTML, JavaScript (Flexible)
Structure	Preference on multiple file: template (HTML), style (CSS) and logic (JS)	Single file	Preference on single file (.vue)
Support	TypeScript (Restrictive)	JSX(similar to HTML) (Flexible)	HTML, JavaScript (Flexible)
Type	JavaScript Framework	Open Source JavaScript Library	Progressive JavaScript Framework
Easy to learn	Difficult	Moderate	Easy
Performance	Slowest	Fastest	Intermediate

Table 2.2: Comparative Table between Web Interface Framework Solutions

Chapter 3

Solution Proposal and System Design

After reviewing the state of the art regarding several data visualization solutions in real-time, it is expected with this dissertation to deliver the user a platform that envisions the data visualization in real-time with the most accurate results and also with a user-friendly design so that the user can quickly understand and observe the outcome.

The key to a successful project is a proper planning, design, and a thorough understanding of the requirements, from existing conditions to end goals, and afterward, proper planning and design. Consequently, this chapter will focus on all these topics, including requirements (both functional and non-functional), use cases, architecture, and design.

3.1 Solution Proposal

The solution proposal for this dissertation starts with planning the main use cases that will be used in the system design. Then, the system architecture will be designed with all the technologies involved and how they will be connected. The technologies that will be used will be the ones that are more efficient and with the outcome as most accurate as possible.

Before the platform development, there will be designed mockups for the various functionalities that the platform will include.

After that, the platform will start to be developed and reviewed with various iterations. It will be developed a platform to manage and visualize all data in real-time. It was decided to take the approach of building a "dashboard" type platform. This way, the user has easier control over options in a sidebar menu while continuing to see the main map.

Being able to receive real-time occurrences can genuinely make a difference when talking about hard decisions. For instance, if a customer requests a LUGGit's service, he expects it to be executed reasonably fast, but if there are no Keepers around that area, it can take longer than usual. In this situation, the LUGGit's operation team will take action and contact the fastest Keeper to take the Request.

After the platform development, various tests will be done in order to have a robust and precise real-time platform.

3.2 Stakeholder's Requirements

There are some important requirements defined by the LUGGit's head of operations for this project, like for example, the possibility of a simple visualization in real time of the various services and the access of their data information. Some functionalities that are required are the visualization of the Keepers in real time on the map, the delivery and the collection services, the Keeper's routes, the location of the emergent services (Drafts) and the response's data information in real time (availability and collection time). It is also important to have a search functionality by service status. Some other functionalities that are needed to develop in this project are the possibility of visualizing all the Keeper's data information when clicking on him or her. The data information of the Keeper that needs to be shown is the name, entity, vehicle's availability, affected service, affected time and contact line. When clicking on the User (on the delivery or collection point), it should be possible to view its data information, like the name, affected service, affected Keeper, estimated time of conclusion and contact line. When clicking on the Trajectory, it should be shown the data information of the time and distance of the route and an estimate if the route will be successfully fulfilled or not. Another important requirements are the creation of alerts that allow anticipate problems, such as services without responses, services with responses superior to N, delayed responses and the possibility of Keepers being out of time or out of the route.

3.3 Use Cases

Use cases are a representation of how users will perform tasks on the platform. It will be illustrated in the subsections below the various actions that users can perform.

Use cases are beneficial to this project since they clarify how the system should act while also helping to discuss what may go wrong. They are the first step of the system design, presenting a set of objectives, which may be used to calculate the system's complexity. In this case, LUGGit can then negotiate which functionalities become requirements and are built.

3.3.1 General Use Case

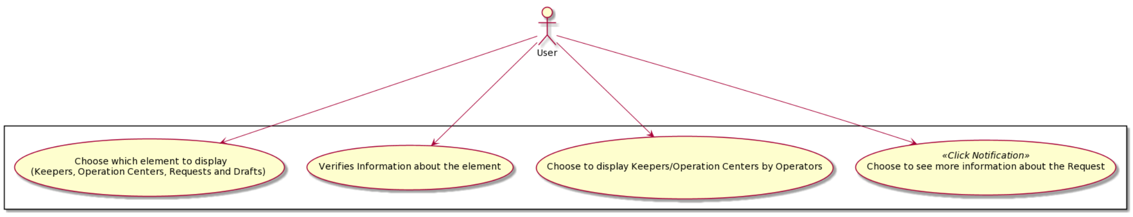

User
Choose which element to display
(Keepers, Operation Centers, Requests and Drafts)
Verifies Information about the element
Choose to display Keepers/Operation Centers by Operators
«Click Notification»
Choose to see more information about the Request

Figure 3.1: General Use Case

This use case describes what all the following use cases illustrate, which are based on the set of objectives discussed to build the platform. The user can choose which elements (Keepers, Drafts, Requests and Operation Centers) to be displayed on the map. He can also check information about any element, through a platform's action, allowing it to happen at any time. The user can choose what elements to be displayed on the map according to its Operators, namely Keepers and Operation Centers. Finally, the user receives a notification when a new Draft or Request occurs, which allows him to click on it displaying its corresponding locations.

3.3.2 Use Case 1

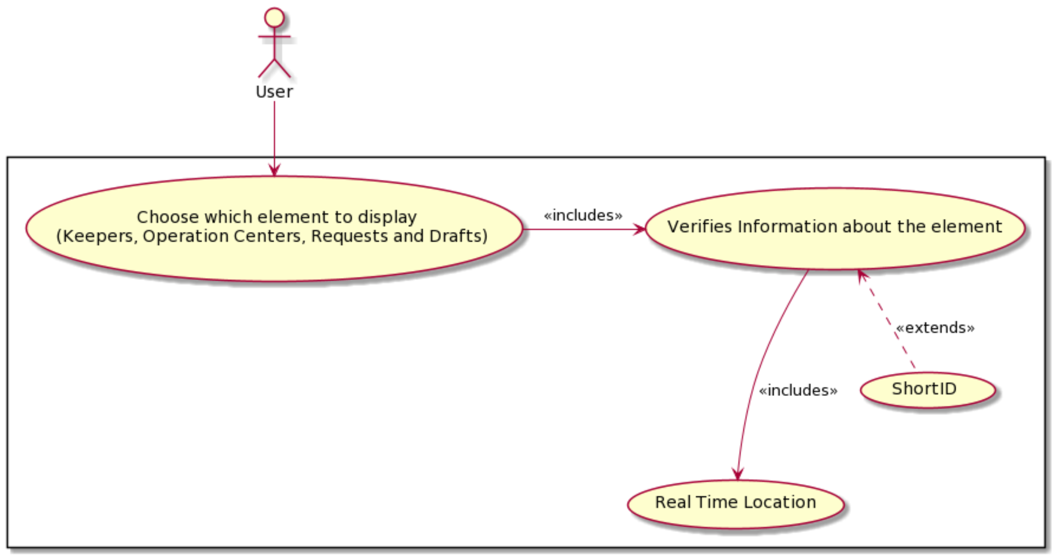

User
Choose which element to display
(Keepers, Operation Centers, Requests and Drafts)
«includes»
Verifies Information about the element
«extends»
ShortID
«includes»
Real Time Location

Figure 3.2: Use Case 1 - Filter out elements in the map view.

The user can choose which and how many Keepers, Operation Centers, Requests, or Drafts to display on the map. By doing so, he has the instant ability to see their respective locations and IDs (acro_list). The user is allowed to do this at any moment.

3.3.3 Use Case 2

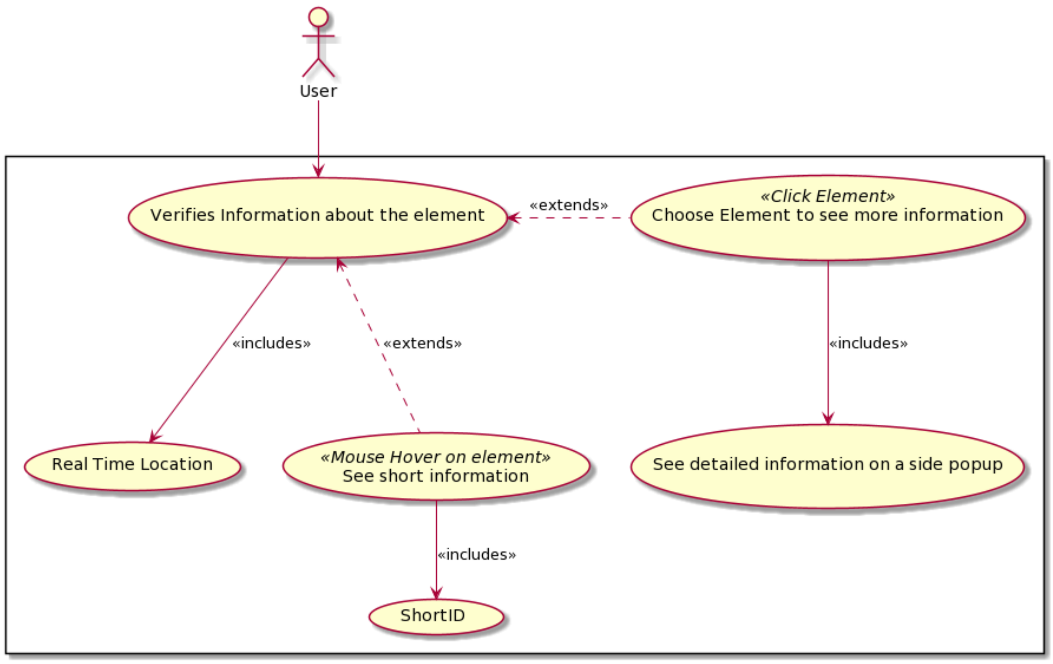


Figure 3.3: Use Case 2 - View elements' information (Keepers, Operation Centers, Requests and Drafts).

The use case 2 describes the platform user's action to know more about any element in the map view. With a mouse hover over each element, the user will see a detailed popup with short information. Upon mouse click event, the detailed information popup appears on the left side of the dashboard. The user can then interact with clickable elements to take advantage of dynamic interaction buttons.

3.3.4 Use Case 3

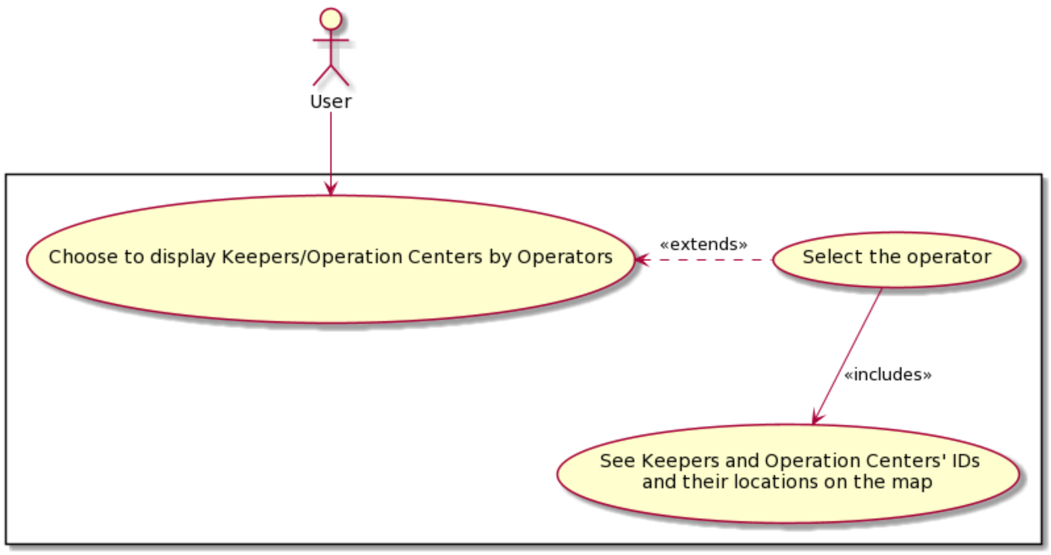

User
Choose to display Keepers/Operation Centers by Operators
«extends»
Select the operator
«includes»
See Keepers and Operation Centers' IDs and their locations on the map

Figure 3.4: Use Case 3 - Filter Keepers and Operation Centers by Operators.

The user can choose certain operators allowing Keepers belonging to that Operator to be displayed in the map highlighted by a specific color. Excluding an operator means that their associated operation centers and Keepers will no longer be displayed on the map.

3.3.5 Use Case 4

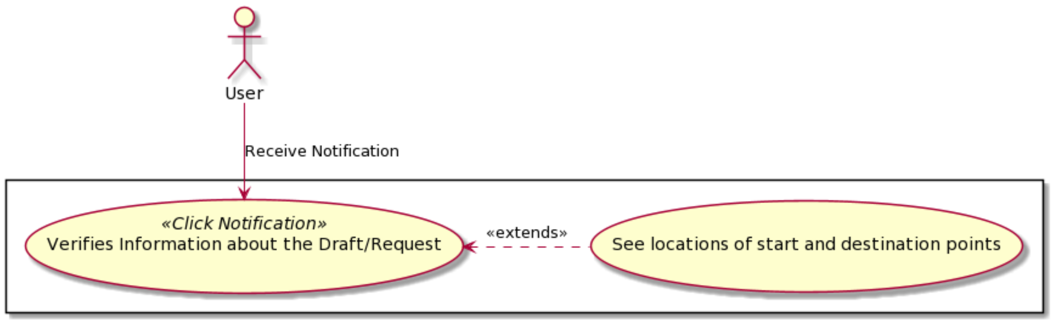

User
Receive Notification
«Click Notification»
Verifies Information about the Draft/Request
«extends»
See locations of start and destination points

Figure 3.5: Use Case 4 - View Notification of new Drafts and Requests.

Use case 4 describes that whenever a new Draft or Request occurs, as explained in section 1.2, a popup notification appears to warn the user of its occurrence. Once the user is notified, this notification can lead to the Draft/Request's location.

3.4 Non-Functional Requirements

Non-functional Requirements define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs [41].

These are the non-functional requirements of this project, considering the structure followed by LUGGit:

1. Availability - The platform can be used at any time of the day. The application will not be available in situations like unplanned system downtime and updates.

2. Redundancy - In the event of internal errors, the service must recover from them and continue their execution, providing the reason for its occurrence.

3. Security - The service should only allow authorized users to use it, and all communications must be made using network protocols that allow secure communication.

4. Performance - The system must have low response times at the loading stage of the website, and all the system features load time should not be more than one second for users. The overall platform must be optimized to prioritize performance as the smooth rendering of the map and the user's computer hardware can limit its elements. As the number of requests, Keepers, and Drafts grows, the platform must sustain the performance effectively without noticing any hiccups.

5. Scalability - The service must be scalable to match the speed of response expected (real-time), regardless of the amount of data that needs to be displayed.

6. Usability - The application interface has to be user-friendly and easy to use by any user, independent of its personal or professional background.

7. Documentation and Versioning - The project must provide documentation for the set of features and endpoints for easier use and understanding. It should also implement a sound versioning system.

3.5 System Architecture Design

Since LUGGit's prominent global architecture is implemented with the micro-service style, it was decided to follow this style for the system architecture implementation. This architecture style has a big advantage, as it is a service that can be built independently and the communication between blocks is apart.

This architectural style is an approach that structures an application as a highly maintainable collection, independently deployable, and loosely coupled. It brings many advantages to this project, such as each service being independent of each other and the testability, deployability, and maintainability are more straightforward than other architectures. They can be used in other projects without the need for many configurations. In Figure 3.6, it is presented the global architecture of the platform proposed to be implemented and it is divided into several components.

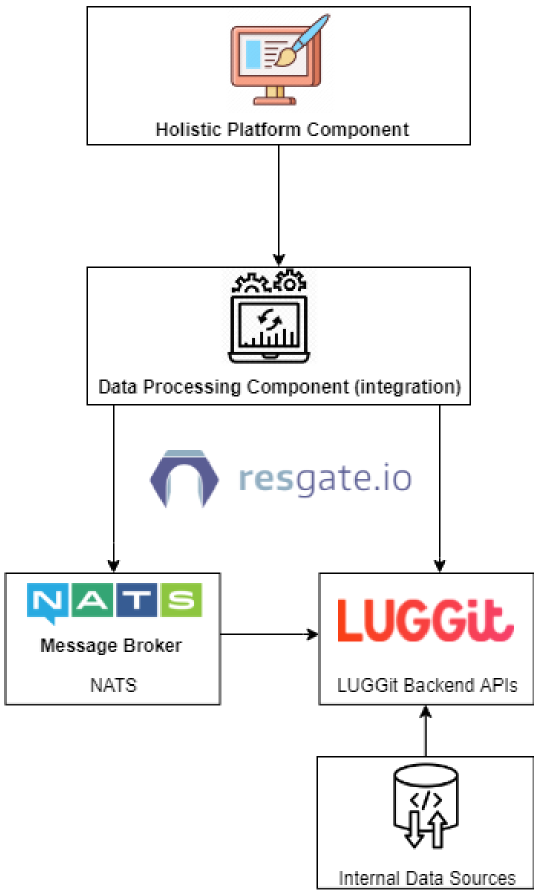


Figure 3.6: High-Level System Architecture.

The system architecture represents the entities in a typical workflow that are interconnected. In Figure 3.6, it is possible to see two components on the top, which are going to be developed in this project. The Holistic Platform Component retrieves information from the Data Processing Component, in order to be displayed to its user. The latter communicates to two entities: the first being NATS which consequently retrieves data from LUGGit's APIs and, the second being the company's Backend APIs directly. Finally, the LUGGit's Backend APIs get information from its internal data sources, which is outside of this project's scope.

The Data Processing Component is intended to be the core of data. It is responsible for receiving events and data from LUGGit's internal APIs, storing it in a caching system and then, transmitting it into the dashboard. This cached database provides data as soon as the dashboard is accessed to avoid slowdowns on startup, which will be explained in detail in the next chapter.

The other micro-service, represented in the Figure 3.6 as Holistic Platform Component, constitutes the UI platform that the user is going to use. This platform can listen to new events sent in real-time, process them into data objects, and display them. This service is vital since it turns raw data into visual elements that humans can easily recognize.

Furthermore, it will be discussed the characteristics of each component. Adding up the establishment of communication between these two components, we can then recognize a complete system that can process and display data.

3.5.1 Data Processing Component

This section will perform a detailed presentation on a typical service, presenting its inner structure, functional requirements, and connections with other services. This component, which was named as Data Processing Component, is the service that retrieves data from LUGGit's APIs and sends it to external sources while storing it. Therefore, it is a service to abstract all the data flow between services and the data coming from LUGGit's APIs. Whenever a service inside or outside of this project's context needs data from any operations' elements, it only needs to request it from the Data Processing component.

The component will retrieve real-time data, and by doing so, also handles the storage of information in the database. It is connected to LUGGit's APIs infrastructure and this always-on connection is a gateway for information to be streamed. In such a manner, this module is always listening to new potential data that might arrive, making it reliable for real-time updates. It then proceeds to store it in the database and send a new update to the clients who always listen for new data.

This approach creates a more generic service with great flexibility, thus improving data flow performance using a cache, data normalization and Publish/Subscribe scenarios.

3.5.1.1 Functional Requirements

1. The component must provide an accessible API through a URL.
2. The component must provide six API endpoints to obtain the lists of elements' data stored in the cache system. Each endpoint concerns about:
    - Logistic Centers, more specifically their IDs, names, locations and Operator IDs;
    - Operation Areas, including their IDs, names, and polygon locations form the available operation area in the map;
    - Operators, more specifically, their IDs, names, emails, phone numbers, list of Keepers and the list of Logistic Centers;
    - Keepers, specifically their IDs, names, locations, requests' list, luggage available capacity, rating, status, operator IDs, phone numbers, license plate numbers, brands, models and schedules;
    - Drafts, specifically their IDs, customer names, start and end times/locations, distance and luggage occupation in the automobile;
    - Requests, with the same information of Drafts also adding their designated Logistic Center ID, Keeper ID and status.
3. The service must retrieve data from LUGGit's API, check for errors and react accordingly. The data retrieval includes two forms: one that expects an event to arrive and is always listening, and another that requests data from an endpoint when required;
4. The service must provide an event-driven system in order to retrieve real-time data. A new event is sent from LUGGit's API to this service every time a new attribute changes. This event-based communication occurs when:
    - a new Keeper comes online, or when one of its attributes change, like location, status, requests list, capacity or schedule;

- a new Draft is submitted;
- a new Request is submitted, or when one of its attributes change, like Keeper ID, status or any other details.

5. The service must be able to store data seemingly in a cached system. This means that all data arriving from LUGGit's API is stored in a database, and after a predetermined amount of time, data is removed;

6. The service must send data through event-based communication as soon as it receives such information. This allows for a fast information transfer between LUGGit's servers and the platform component;

7. Each endpoint must have security and authentication protocols implemented to provide extra control on who can access these endpoints.

3.5.1.2 Internal Architecture

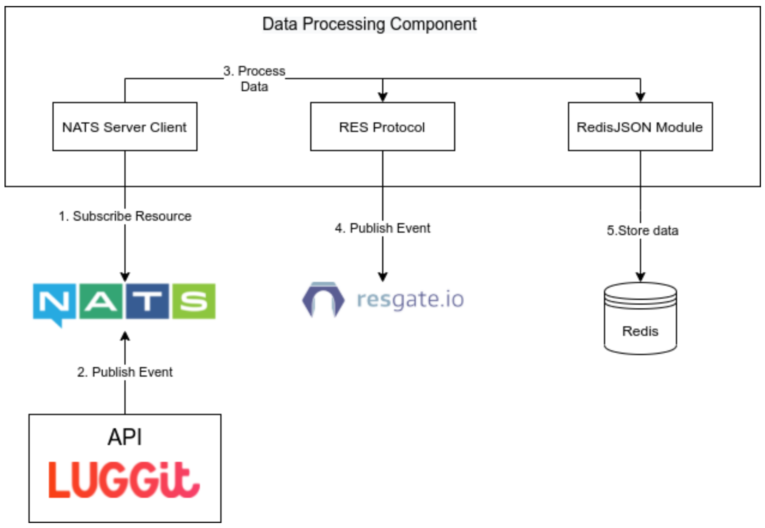


Figure 3.7: Data Processing Component Architecture

This section describes the internal architecture of the Data Processing Component. For an improved understanding of the diagram illustrated in the Figure 3.7, a numbered list representing each interaction will be presented below:

1. Subscribe Resource: It is the process of subscribing to a specific subject, representing the data produced by a resource. This process is initiated as soon as the Data Processing Component is started, and it is always running, waiting for new events to arrive;

2. Publish Event: The LUGGit's container generates a publishing process, which will send data to a previously defined message subject;

3. Process Data: It represents the data normalization and determining which data will be stored and which is helpful to send as an event;

4. Publish Event: Whenever the component receives new events, it will notify all clients and deliver the new content to them;
5. Store Data: For caching purposes, the new data will be stored in a Redis database using the RedisJSON module.

3.5.2 Holistic Platform Component

The Holistic Platform Component is the platform that the user will interact with, so it needs the most focus in the entire project. This component has built-in features to make the task of visualizing real-time data easier for the user. These features can bridge the gap between raw data and elements understood by the human eye. Instead of having a table with raw data, this interface comprises a map that displays elements representing the real-world scenario.

Just like the data processing component, this service is always listening for new events and it updates them in real-time. Then, data is automatically displayed when is recently updated. For instance, when a new Keeper's location is received, react updates it in the correct object using its ID, and it is instantly refreshed in the map view. Regarding data that is not updated as often, the platform sends a REpresentational State Transfer (REST) request of all the elements it needs on boot. This is done without many bottlenecks since the data is stored in a cache, making it faster to retrieve.

3.5.2.1 Functional Requirements

The platform must:

1. Be accessible through a Uniform Resource Locator (URL).
2. Lay out a map view of the operation areas.
3. Provide a map legend of all the present elements.
4. Include Keepers, Operation Centers, Drafts and Requests' real-time location with their start and destination points.
5. Show an optional side panel with detailed information about elements that are clicked on the map. The detailed information is dynamic and can be clickable, leading to other elements' side panel.
6. Show Keepers' activity information, including their contact, name, rating, and active Requests.
7. Real-time information about Drafts' activity, including their customer name, rating, status, and pending Requests.
8. Illustrate Requests' activity information, including their contact, designated Keeper and their status.
9. Show information about Operation Centers' activity, including their contact, designated Operator, status, and pending Requests.

10. Present Operators' activity information, including their contact, Keepers, Operation Centers and active Requests.

11. Filter in real-time what is displayed in the map, specifically Operators, Operation Centers, Keepers, Requests and Drafts.

3.5.2.2 Internal Architecture

The diagram illustrated in Figure 3.8 is the internal architecture of the Holistic Platform Component and has a numbered list representing each interaction that will be presented below:

1. Request Resources: As soon as the service boots up, it will request all the initial resources to populate the map with elements. This is done with REST protocol requests to the Data Processing Component.

2. Update application state: It is the process of creating new elements and their according states with the requested data in the previous step.

3. Access URL: This is where the user accesses the website to use it. Some data is already present since some elements' state was already updated.

Display website: This item is not numbered because after the user accesses the platform through an URL, the website is constantly being displayed.

5. Subscribe Resource: It represents the moment after the user has opened the platform and in which the service subscribes to new data changes.

6. Update application state: Once again, this is updating the elements' states when new data is received. Whenever the component is alerted that new events were published, it will process data and translate it into the existing objects already defined in step 2.

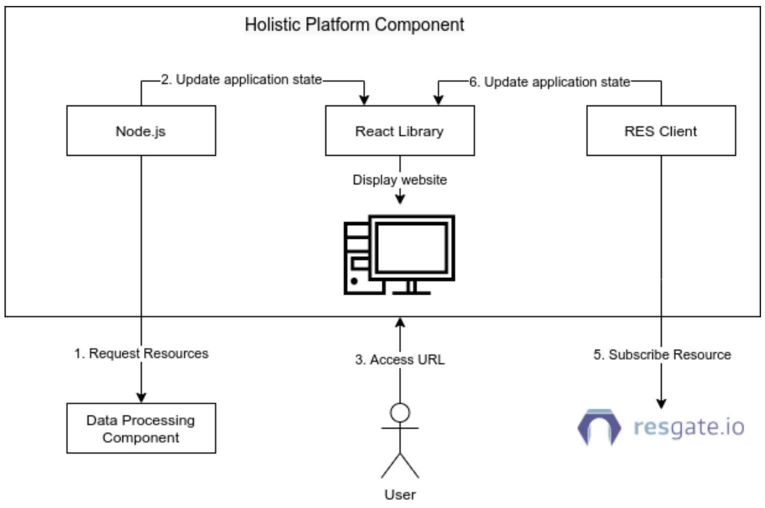


Figure 3.8: Holistic Platform Component Architecture

3.6 Mockups

The first steps of the solution proposal for this dissertation consists of the planning of the main use cases and the design of the mockups for the various functionalities that the platform will include.

Planning the main use cases that will be used in the system design and following with the mockups are important steps for the platform development. It is part of the system design work to investigate and prototype the UI concepts.

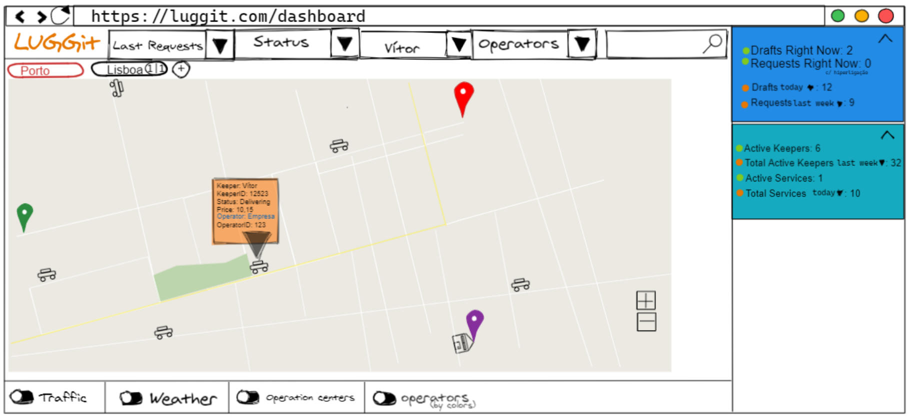
The objective is to interpret the defined use cases and the company's requirements to come up with mockups. These will bring value later on in the development process.

As a holistic logistics platform for LUGGit's operations, it will control several Keepers, Drafts, Requests, Operation Centers and Operators. It will be possible to search, add and remove from the view all of these elements and see more real-time information about them. Therefore, it is essential to keep all these interfaces easily accessible without compromising usability.

The mockups that are illustrated and explained in the subsections below are the first designed mockups, which were later sent to the company's team and after some reviews, there were made some changes.

3.6.1 Mockup 1 - First Landing Page

The mockup shown in Figure 3.9, was the first one to be made and it is possible to visualize that the center screen has a map with several elements inside it. It is possible to see multiple car icons that represent the Keepers and their real-time locations and three markers: the green one represents the pickup point of a request, the red one is the drop off location and the purple represents the location where luggage is stored. This final marker happens to be in an Operation Center, meaning that luggage is stored in that location. Then, a small popup above a Keeper shows some information about it, like its name, ID, status, Operator and Operator ID. Finally, there are three menus with options. The top bar has several options to filter Requests, Keepers, status and Operators, but it was later adjusted to a different location to improve usability. The sidebar menu is just displaying some information about Drafts and Requests and the bottom menu has different toggles about additional features to the map.



Keeper: Vitor
KeeperID: 12523
Status: Delivering
Price: 10,15
Operator: Empresa
OperatorID: 123

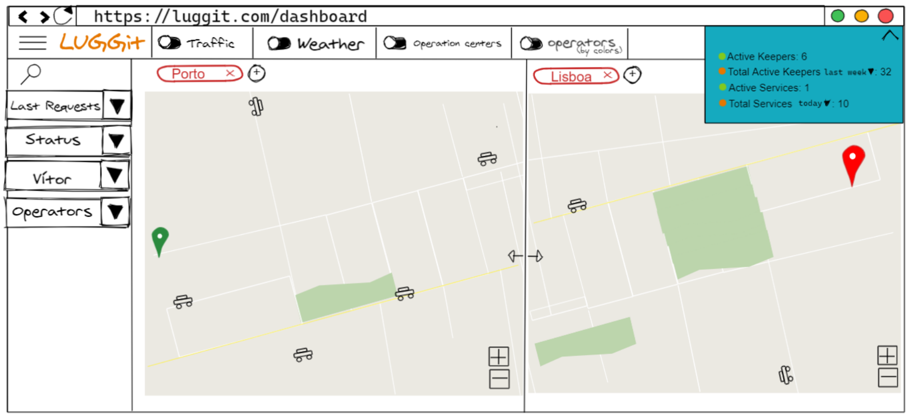
Drafts Right Now: 2
Requests Right Now: 0
c/ hiperligação
Drafts today ▼: 12
Requests last week ▼: 9

Active Keepers: 6
Total Active Keepers last week ▼: 32
Active Services: 1
Total Services today ▼: 10

Figure 3.9: Mockup 1 - Landing page

3.6.2 Mockup 2 - Operation Areas Page

The second mockup represented in Figure 3.10 already shows a different location for the search and the filter buttons (left side), which already improves the user's effectiveness and engagement in finding these options. This mockup has the primary purpose of demonstrating the splitting Operation Areas in the screen. This way, the user can pay attention to two cities simultaneously.



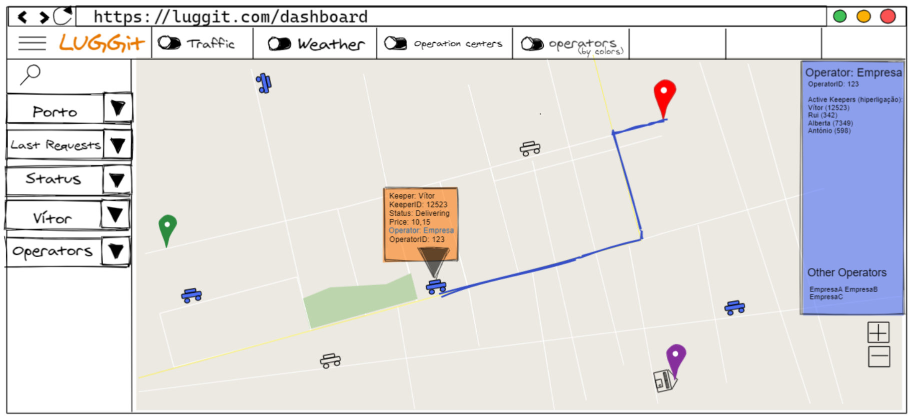
Active Keepers: 6
Total Active Keepers last week▼: 32
Active Services: 1
Total Services today▼: 10

Figure 3.10: Mockup 2 - Multiple Operation Areas at the same time

3.6.3 Mockup 3 - Keeper's Routes

In Mockup 3, in Figure 3.11, we can see what happens after a user clicks on a Keeper, in this case, Vítor. The Keepers that belong to the same Operator are highlighted with the same colour as the one clicked, and just like in the first mockup, there is an information popup on top of the car icon.

The main differences are the right sidebar that shows information about this particular Operator and the route this Keeper is taking at the moment. We can easily see that Vítor is going to the Request's drop off location and which route he will take.



Keeper: Vítor
KeeperID: 12523
Status: Delivering
Price: 10,15
Operator: Empresa
OperatorID: 123

Operator: Empresa
OperatorID: 123

Active Keepers (hiperligação):
Vítor (12523)
Rui (342)
Alberta (7349)
António (598)

Other Operators
EmpresaA EmpresaB
EmpresaC

Figure 3.11: Mockup 3 - Keeper's Route and Operator's Information

3.6.4 Mockup 4 - Operators Filtering

The mockup in Figure 3.12 has two active toggles: the Operations Centers and Operators (by colours). This means, respectively, that Operation Centers are shown on the map and that Keepers are coloured according to which Operator is active.

On the left sidebar menu, there is also the option to choose which Operators to display on the map, giving a greater sense of customization.

Each Operator is defined by a colour on the right sidebar, allowing a quick glimpse of how many Keepers by Operator and where they are on the map.

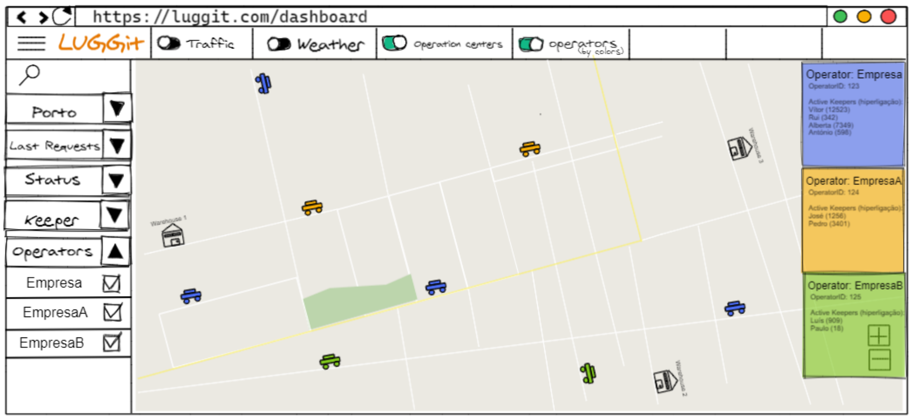


Figure 3.12: Mockup 4 - Highlight Operators with different colors

3.6.5 Mockup 5 - Additional features

The mockup in Figure 3.13 shows that only one toggle option is active, and it is the weather one. This is intended to check if there are more delays in a particular Operation Area due to the weather. This is where the map has more precipitation and less since there is a legend map with colour on the right side.

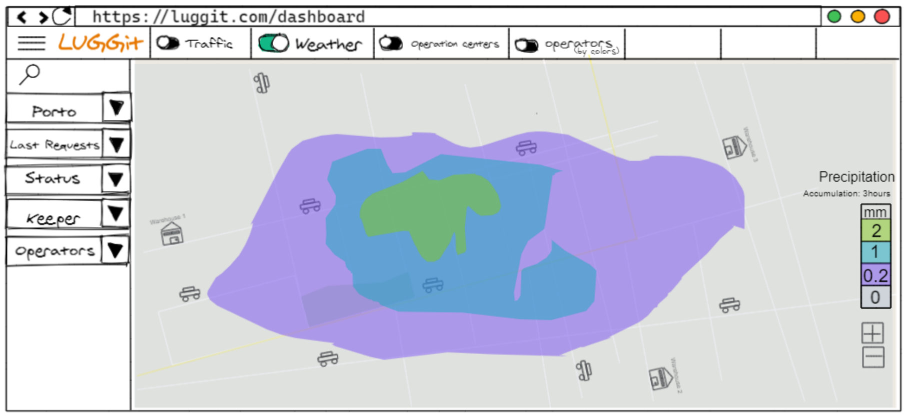
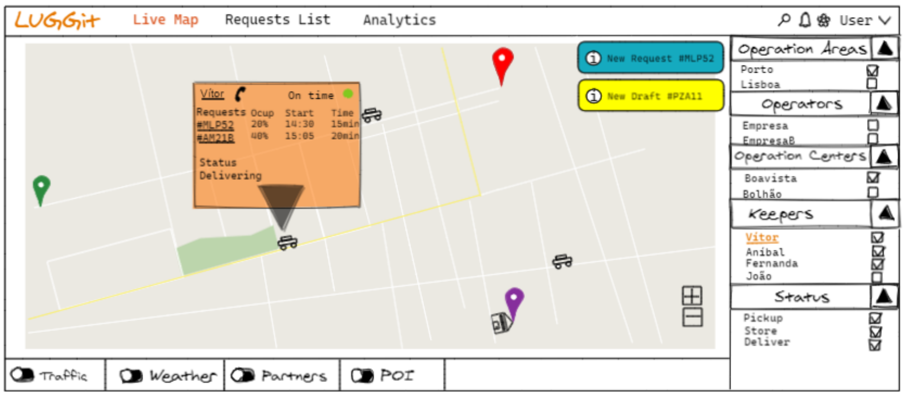


Figure 3.13: Mockup 5 - Additional feature of Weather

3.6.6 Mockup 6 - Landing Page final design

The final mockup in Figure 3.14 is the closest to the final version of the dashboard. It has a top bar menu with three possible pages within the website: "Live Map", "Requests List", and "Analytics", notifications, settings and search icons. Here, the options menu is on the right side with the drop-down form for each option. It is possible to hide/show whatever elements the user wants without leaving the page or opening a popup. The additional features toggle menu is now on the bottom since it is not as important as all the other primary attributes. The page seems much more organized and with a more appealing design than the previous mockups.

A significant difference is the presence of notifications. They appear on the top right side of the map and with a different colour, depending on if it is a Request or a Draft.



Vitor On time

Requests	Ocup	Start	Time
#MLP52	20%	14:30	15min
#AM21B	40%	15:05	20min

Status
Delivering

Figure 3.14: Mockup 6 - Notifications and Menus more defined

The design of the mockups was a very significant step for the platform development. It was designed the visual concept for the main functionalities that will be developed, but also with a user-friendly visual design.

Chapter 4

Development

This chapter intends to show the development of the solution proposal that was described in the previous chapter. This part will discuss the development process and its technical approach, focusing on the technologies used, the required tools, and the solution's functionalities. It is important to note that this implementation was carried out in a business environment. The paths to follow in the development of the solution were made in partnership with the company in question. Due to its experience in developing web applications, it was valuable to choose the best tools and technologies to develop and conceptualize the product itself.

4.1 Development Requirements

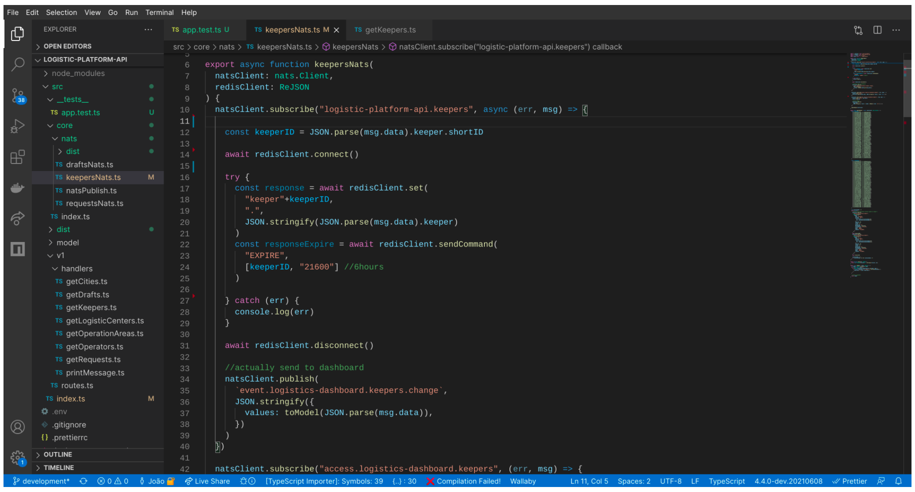
This section will cover technologies that were used to help address the workflow and objectives to be reached. I was given the freedom of choice to pick the technologies I figured that were best suited to find a solution to the problem. As this dissertation has to be in the company's context, some technologies were already integrated with the company's solutions.

Since most of these technologies were not taught in my academic course, I found in this dissertation an opportunity to be great for my personal growth as a developer. In addition to improving my skills and experience, what led me choosing these technologies, in particular, was their versatility.

4.1.1 Development Environment

The development environment chosen was Visual Studio Code[1], developed by Microsoft. This open-source editor is intended to be used with multiple languages and it was used on a Linux operating system.

[1] https://code.visualstudio.com/



```
export async function keepersNats(
  natsClient: nats.Client,
  redisClient: ReJSON
) {
  natsClient.subscribe("logistic-platform-api.keepers", async (err, msg) => {

    const keeperID = JSON.parse(msg.data).keeper.shortID

    await redisClient.connect()

    try {
      const response = await redisClient.set(
        "keeper"+keeperID,
        ".",
        JSON.stringify(JSON.parse(msg.data).keeper)
      )
      const responseExpire = await redisClient.sendCommand(
        "EXPIRE",
        [keeperID, "21600"] //6hours
      )

    } catch (err) {
      console.log(err)
    }

    await redisClient.disconnect()

    //actually send to dashboard
    natsClient.publish(
      `event.logistics-dashboard.keepers.change`,
      JSON.stringify({
        values: toModel(JSON.parse(msg.data)),
      })
    )
  })

  natsClient.subscribe("access.logistics-dashboard.keepers", (err, msg) => {
```

app.test.ts U
keepersNats.ts M
getKeepers.ts
src > core > nats > keepersNats.ts > keepersNats > natsClient.subscribe("logistic-platform-api.keepers") callback
LOGISTIC-PLATFORM-API

Figure 4.1: Development Environment with Visual Studio Code.

4.1.2 Planning, Project Management and Versioning

Clickup[2] was the tool used to provide task management and project planning. It is LUGGit's primary tool for project management, and as such, this project also took part in this platform. LUGGit follows a specific structure in development management, being that for each project is created a new section. In each section, it is possible to add new tasks and their priority and assigned workers. As it is possible to see in the example of this project's Clickup page in Figure 4.2, there is also a deadline for each task and respective progress.

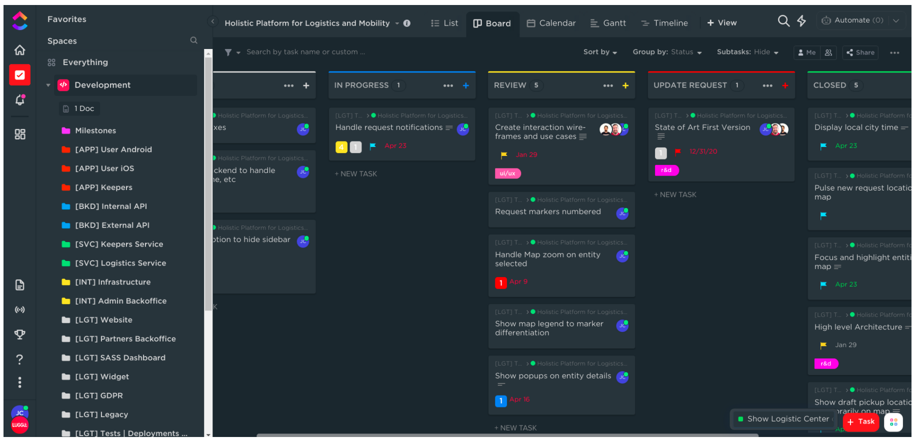
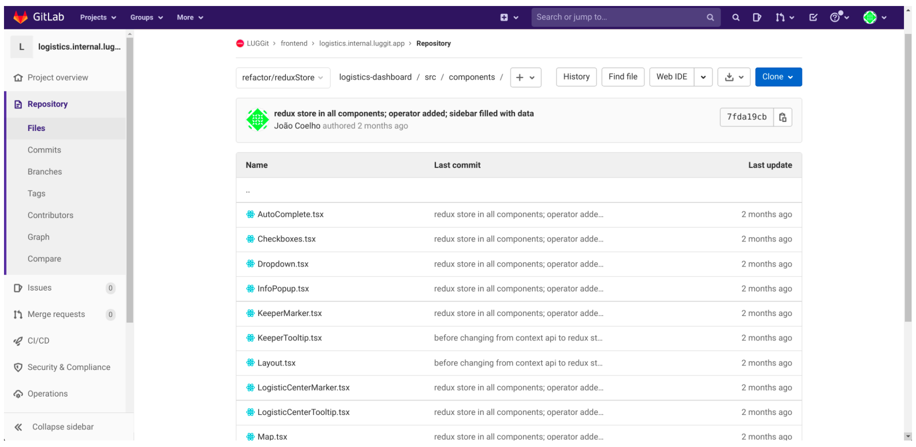


Figure 4.2: Clickup's Dashboard with this project's tasks.

For the storage of codebase, it was used GitLab[3] for the two different components. GitLab is an open source web-based tool that provides a Git repository manager. Git[4] is a free and open source distributed version control system and it was used for versioning control of both codebases. Each version is submitted using a branch, representing an independent

[2]https://clickup.com/
[3]https://gitlab.com/
[4]https://git-scm.com/

line of development. Branches modifications are recorded in the history using commits, each following the Conventional Commits[5] method, being a standard followed by the company for other projects. In Figure 4.3, there is an example of GitLab's commits for this project.



LUGGit › frontend › logistics.internal.luggit.app › Repository

refactor/reduxStore   logistics-dashboard / src / components /

redux store in all components; operator added; sidebar filled with data
João Coelho authored 2 months ago   7fda19cb

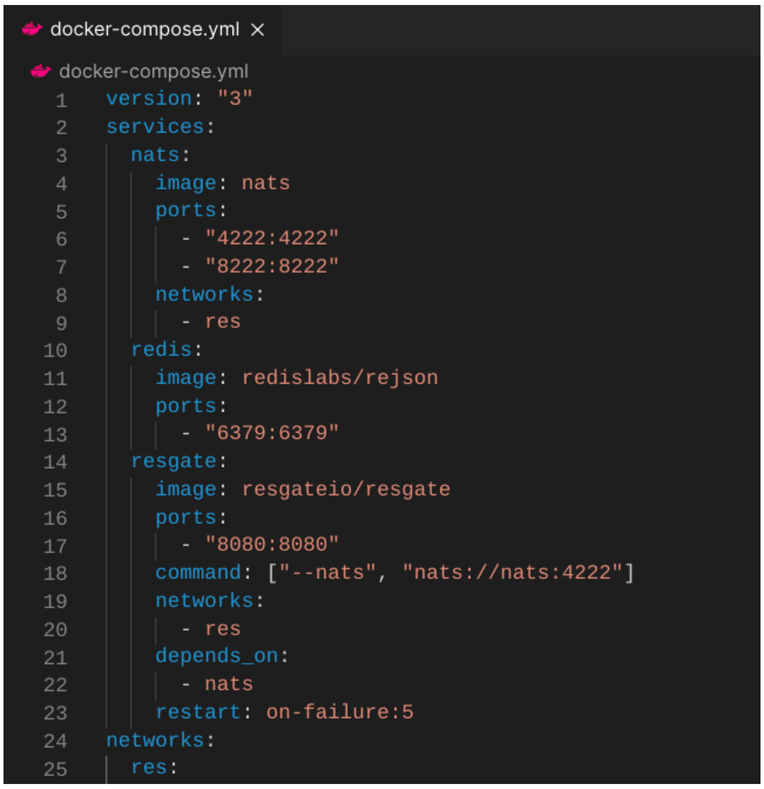
Name	Last commit	Last update
..		
AutoComplete.tsx	redux store in all components; operator adde...	2 months ago
Checkboxes.tsx	redux store in all components; operator adde...	2 months ago
Dropdown.tsx	redux store in all components; operator adde...	2 months ago
InfoPopup.tsx	redux store in all components; operator adde...	2 months ago
KeeperMarker.tsx	redux store in all components; operator adde...	2 months ago
KeeperTooltip.tsx	before changing from context api to redux st...	2 months ago
Layout.tsx	before changing from context api to redux st...	2 months ago
LogisticCenterMarker.tsx	redux store in all components; operator adde...	2 months ago
LogisticCenterTooltip.tsx	redux store in all components; operator adde...	2 months ago
Map.tsx	redux store in all components; operator adde...	2 months ago

Figure 4.3: GitLab's versioning system.

4.1.3 Local Code Testing and Development

After developing the Holistic Platform Component and the Data Processing Component, each executed in its own server, there was a need to integrate the communication between these components and the database, locally. This was done using Docker Compose[6], a tool that allows the instantiation of a stack of programs, using Docker as a virtualization tool. By this way, the installation, the configuration and the integration of external software components were easier to handle all at once.

[5]https://www.conventionalcommits.org/en/v1.0.0/
[6]https://docs.docker.com/compose/



```yaml
version: "3"
services:
  nats:
    image: nats
    ports:
      - "4222:4222"
      - "8222:8222"
    networks:
      - res
  redis:
    image: redislabs/rejson
    ports:
      - "6379:6379"
  resgate:
    image: resgateio/resgate
    ports:
      - "8080:8080"
    command: ["--nats", "nats://nats:4222"]
    networks:
      - res
    depends_on:
      - nats
    restart: on-failure:5
networks:
  res:
```

Figure 4.4: Docker compose file example.

4.1.4 Deployment

After talking about how each component was built and its features, this section will mention how the deployment process was done. The initial deployment strategy, throughout the development process, was to run everything locally. Firstly, running the Holistic Platform Component's server and, after that, the Data Processing Component's server and docker-compose with NATS and Resgate containers for the communication between the two.

On an intermediate stage of this project, there was the need to facilitate the continuous integration and deployment of all software, thus deciding to automate this process using GitLab CI/CD[7].

This tool provides a way of compiling, packaging, testing, and deploying the application, all in a single pipeline, and it is activated when a code change occurs. These tasks are called stages, and if one of them fails, the pipeline does not proceed and outputs an error.

After the pipeline process, the application is compiled into a Docker image to be executed in a container.

The rest of the configuration is composed of a set of YAML files, which are compiled by Helm[8] and deployed on Google Kubernetes Engine[9].

This way, all the deployment process is automated and executed in LUGGit's servers.

[7]https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/
[8]https://helm.sh/
[9]https://cloud.google.com/kubernetes-engine

4.2 Used Technologies

In this section, it will be mentioned which technologies were used and why they were the better options compared to others.

From now on, there will be some technological terminology referred throughout this chapter. The terms frontend and backend refer to the separation of concerns between the presentation layer (frontend), and the data access layer (backend) of a piece of software. An endpoint is an interface exposed by a communication channel, in order to provide data transfer when requested.

TypeScript[10] was the chosen language to develop both the frontend and the backend of this project. This decision was carried out because TypeScript is more reliable, easier to refactor and because the vast majority of LUGGit's projects were written in this language, allowing better supervision from my mentors and coherency throughout the codebase.

For the map solution, Mapbox was used as it provided long-term interoperability between the project's technologies and the tools needed for the platform. In parallel to this, the technology used to render interactive maps was Mapbox GL JS.

Other technologies used in web development were Next.js[11] and Reactjs[12], which allow the website to load faster.

Regarding the backend part of the solution, Koajs[13] was the web framework used to provide endpoints, processing data, storing data, and sending data in real-time.

The technology used to store data from the backend was Redis[14]. Redis implements an in-memory cache to decrease data access latency and it is supported by an extensive and helpful documentation.

The communication throughout the entire project was established using two technologies: NATS[15] and Resgate[16]. These technologies were used to provide fast and reliable real-time communication between the backend and the frontend was necessary. These are well-established projects among the developer communities.

4.2.1 TypeScript

TypeScript is an open source programming language developed and maintained by Microsoft that first appeared in 2012. It is seen as a strict syntactical extension of JavaScript, by being transcompiled to the latter. The source code of a program written in TypeScript is converted to an equivalent source code in JavaScript [42]. This attribute makes all existing JavaScript programs valid TypeScript programs.

TypeScript may be used to develop JavaScript applications for both client-side and server-side execution (as with Node.js). This allowed a single programming language used in the two different scenarios of this dissertation, backend and frontend.

While JavaScript is a dynamically typed language, which means that types are checked, and datatype errors are spotted only at the runtime, TypeScript introduces optional strong static typing: once declared, a variable does not change its type and can take only specific

[10]https://www.typescriptlang.org/
[11]https://nextjs.org/
[12]https://reactjs.org/
[13]https://koajs.com/
[14]https://redis.io/
[15]https://nats.io/
[16]https://resgate.io/

values. The compiler alerts developers to type-related mistakes, so they have no opportunity to hit the production phase[43]. This results in less error-prone code and better performance during execution. Actually, a study shows that 15% of all JavaScript bugs can be detected by TypeScript [44].

According to all the advantages mentioned above and maintaining a bigger coherency between all the frontend projects within the company, this language was chosen to develop the micro-services.

4.2.2 Next.js

Next.js is an open-source React frontend development web framework created by Vercel[17] and is now maintained by the community. It runs on top of Node.js[18] and provides functionalities such as server-side rendering and generating static websites for React-based web applications, allowing developers to create static and dynamic websites quickly.

Next.js is one of several recommended toolkits available when starting a new React app, all of which provide a layer of abstraction to aid in everyday tasks. However, the downside of traditional React apps is that the rendering is all done in the client-side browser. At the same time, Next.js is used to extend this functionality to include applications rendered on the server-side. This technology is clever enough to load the JavaScript and CSS needed for any given page. This makes for much faster page loading times, as a user's browser does not have to download unnecessary Javascript and CSS. This increases performance as there is less for the user's browser to download, and the user benefits from seeing the page content quicker [45].

4.2.3 Mapbox

Mapbox is one of the largest providers of custom-designed maps for websites and mobile apps, and this service is used by well-known delivery and transportation companies.

Mapbox uses openly available data to build its platform and supports a community of volunteer mappers, who help provide information and build data about locations. The platform uses OpenStreetMap[19] as the base map and lets developers add different markers, lines, and polygons, as well as layers from external sources (in GeoJSON, GPX, and other formats)[46]. This led to an easier decision, as this technology provided the necessary tools to develop the intended solution of this dissertation.

Mapbox is a good tool when it comes to mapping out clean and concise data with a medium-to-high level of complexity. Plus, its maps are pleasantly appealing. This mapping approach seemed to be the best to use for these reasons.

4.2.4 Mapbox GL JS

Mapbox GL JS is a JavaScript library that uses WebGL[20] to render interactive maps from vector tiles and Mapbox styles. It is part of the Mapbox GL ecosystem[47].

It has a real-time styling and it has many interactivity features that makes this library with a very good performance.

[17]https://vercel.com/
[18]https://nodejs.org/en/
[19]https://www.openstreetmap.org/
[20]https://get.webgl.org/

4.2.5 React

As already explained in chapter 2, React (also known as React.js or ReactJS) is an open-source, frontend JavaScript library for building user interfaces or UI components. It is component-based, meaning that each component manages its state and, together, builds a more complex UI. When we look at a webpage entirely built using React, we can see the UI is divided into multiple components, making the code easier to debug. For instance, having a navigation bar, sidebar, and a map as single components, their properties and functions are also separated.

React's main advantage is being declarative because it is possible to write the code, and the technology will efficiently update and render the right components when the data changes.

Also, it is essential to note that it is the most popular technology among frontend developers, and many LUGGit's projects were developed with React.

4.2.6 Koajs

Koa.js is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is an open source framework developed and maintained by the creators of Express.js, currently the most popular node web framework.

What differentiates Koa from other Node.js frameworks is that it has a simplified use, which is crucial as it serves as the middleware between LUGGit API and the frontend dashboard. Koa.js both simplifies and improves error handling by using middleware more effectively. The framework's built-in catchall for errors assists developers in avoiding website crashes. Even without writing additional code, programmers may report errors with a simple 'try/catch' command. Developers can also configure error handling in Koa.js by simply modifying the default setting [48].

Koa further allows developers to encapsulate the response and request objects into a single object using Context. The unified object helps developers build web applications and APIs more efficiently by using many helpful methods and accessors. When developing with JavaScript, there is even the option to create a context per request and refer to the context as the receiver in the middleware [49].

After mentioning these points, we can see that this framework has many advantages and it is a good framework to develop the server to make everything run smoothly, from the endpoints to the real-time publish-subscribe messaging models.

4.2.7 Redis

Redis, which stands for Remote Dictionary Server, is a fast, open-source, in-memory key-value data store for use as a database, cache, message broker, and queue.

All Redis data resides in memory, in contrast to databases that store data on disk. In-memory data stores avoid seeking time delays and access data in microseconds by eliminating the need to access disks. Redis features versatile data structures, high availability, transactions, on-disk persistence, and cluster support, making it simpler to build real-time internet-scale apps.

I choose to use Redis because it simplifies code by allowing fewer lines of code to store, access, and use data in applications. For example, if an application has data stored in a data structure and there is the need to store it in a data store, the Redis hash data structure stores the data accurately. A similar task on a data store with no hash data structures would require many lines of code to convert from one format to another. Redis comes with native data structures and many options to manipulate and interact with data. Over a hundred open source clients are available for Redis developers, and supported languages include Java, Python, C, C++, C#, JavaScript, Node.js, Ruby, R, Go, and many others [50].

4.2.8 NATS

NATS is an open-source messaging system developed and supported by Synadia[21]. It is a simple, secure and performant communications system for digital systems, services and devices.

Client libraries to interface with the server are available for dozens of major programming languages. NATS has many concepts and functionalities, but the main one implemented in this project is the publish-subscribe.

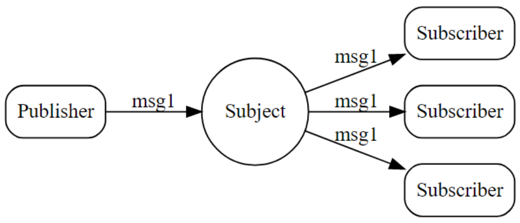


Figure 4.5: Architecture of publish-subscribe messaging model [51].

As we can see in the figure 4.5, this publish-subscribe message distribution model consists of a publisher sending a message on a subject and an active subscriber listening on that subject receives the message. This is an example of a NATS messaging model, among others available. The core design principles of NATS are performance, scalability and ease of use [52]. Therefore, it is perfect for a real-time event-driven solution where data must be instantly available to be displayed.

4.2.9 Resgate

Resgate is a real-time API gateway that allows synchronization between all application clients. Clients will connect to Resgate, using either HTTP or WebSocket, to make requests. These requests are sent to the micro-services that will serve each request over the NATS server, and Resgate will keep track of which resource each client has requested. Whenever there is a change to the data, the responsible micro-service sends an event. Resgate will use this event to update its cache and ensure each subscribing client is kept up-to-date.

4.3 System Development

It was decided to divide this project into two components for the reasons previously mentioned, one to gather and process data and the other to display the data visually in a web dashboard. Also, having separate components brings significant advantages of using these pieces of software and technology later in different context solutions.

[21]https://synadia.com

This section will refer to how each component was technically built, its functionalities/features and their purpose in the project.

4.3.1 Holistic Platform Component

This component's objective is to provide its functionalities in an web interface allowing the user to have a holistic overview of the company's logistics. These functionalities include: displaying visual feedback when a new Draft or Request is made in real-time, displaying where Keepers are placed on the Operation Area, and displaying where other elements are on the map, as well as others mentioned below. At the same time, he can better inform the customer regarding the Keeper's delay, while keeping track of his location, heading to the customer's pickup place.

The technologies used to develop this interface, considering all the research done before this point, were Next.js combined with React, Mapbox, and Mapbox GL JS. In order to make the connection between components, it was used Resgate.

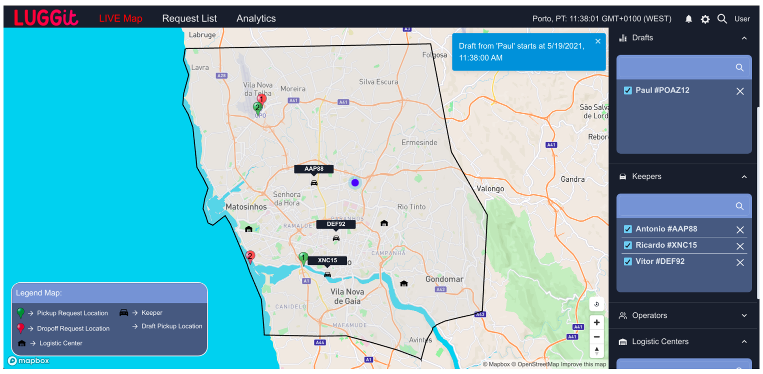

Draft from 'Paul' starts at 5/19/2021, 11:38:00 AM

Figure 4.6: Example of the dashboard with active Keepers, Drafts and Requests.

4.3.1.1 Features

It is possible to manage and visualize real-time data information in the platform. It was taken into consideration a user-friendly design and a platform that the user can use and manage it easily with various functionalities.

The user have the possibility of filter out elements from the map view, make requests and drafts and visualize information. When using the platform, the user can also have feedback by some appearing popups on specific mouse hovers. To facilitate the user's usability, it was also developed a city clock and a map legend.

These features will be explained with more details in the subsections below.

4.3.1.1.1 Filter

The purpose of the filter functionality is to be able to filter out elements from the map view. This feature can be accessed on the right sidebar, with a dropdown menu for each category, including Active Requests, Drafts, Keepers, Operators, Logistic Centers, and Operation Areas. There are two ways of removing an option from the sidebar, either by using the checkboxes or the remove button on one of the names/IDs displayed on each list, taking effect on the map.

If the user decides to use the checkbox, the icon associated with its name disappears from the map. The close button can be clicked, making the name disappear from the list and the map. If there is a need to add a new element to the map, the user can use the search bar. This search bar, when used, will show a list of available elements that can be added to the list and consequently to the map. It uses an auto-complete mechanism that diminishes the available list as long as the user keeps writing, narrowing it to the desired choice.

4.3.1.1.2 Request

Whenever a new Request enters the system, a green notification pops up on the top right corner of the map, warning the user. The notification has information about the customer's name, Request ID, and start time. Immediately, it will appear the Request markers as seen in Figure 4.12. The green marker represents the Request's pickup position. The red one represents the dropoff point of the luggage, and the number inside them has the purpose of helping the user identify which green marker corresponds to its associated red one.

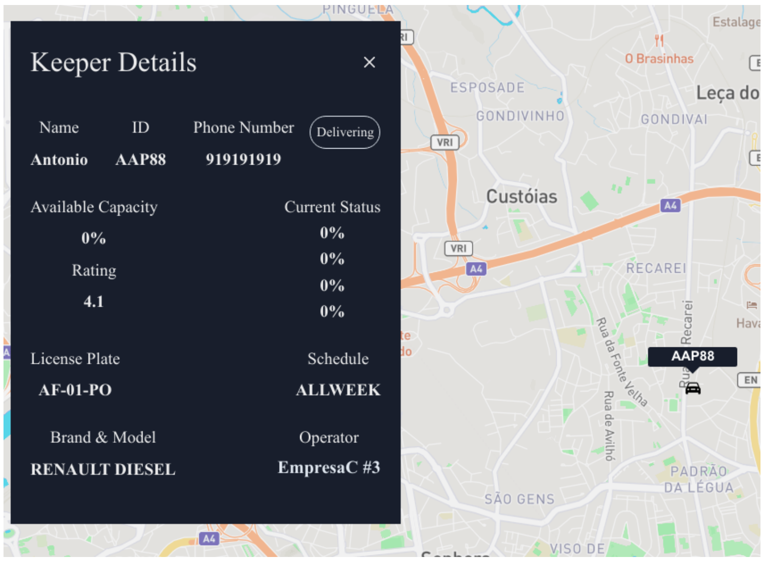
As soon as the Request shows on the map, both markers will blink for a short amount of time to make it easy for the user to notice it as a new Request.

4.3.1.1.3 Draft

Whenever a new Draft is executed, a similar notification pops up in the same place with a different color (blue). The notification contains the same information as a Request. The Draft's pickup location will appear on the map with a ripple effect circle so that the user can see, in real-time, where a possible Request might happen in the future. This can help with better tools for a possible LUGGit's service request from a customer. After a defined amount of time, the ripple effect will disappear and show a grey circle. All the popups will still work, allowing the user to see more information, and if the mouse hovers the circle, it will again show the ripple effect.

4.3.1.1.4 Information Sidebar

The information sidebar is located on the left side of the map for all elements, and it allows the user to visualize more information at a specific time while, at the same time, also viewing the element's real-time position. This section will display some figures in order to understand each one of them and how they work.


Keeper Details

Name	ID	Phone Number	Delivering
Antonio	AAP88	919191919	

Available Capacity
0%
Rating
4.1

Current Status
0%
0%
0%
0%

License Plate
AF-01-PO

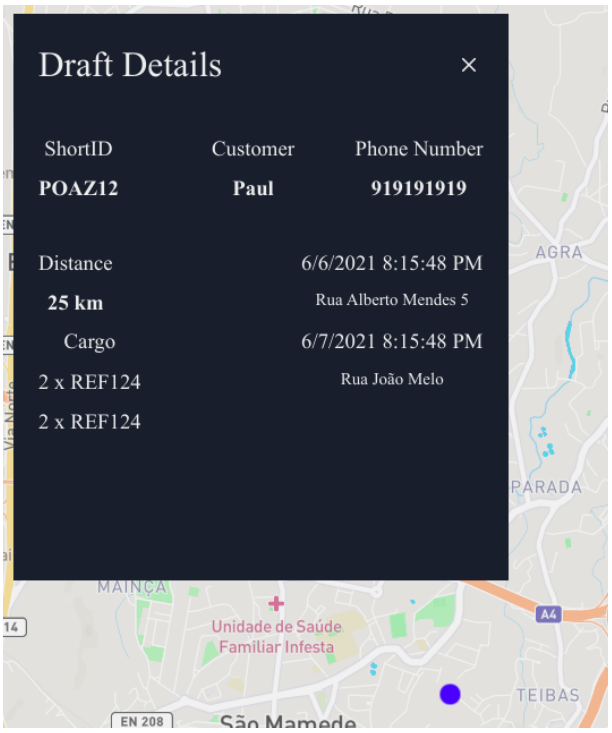
Schedule
ALLWEEK

Brand & Model
RENAULT DIESEL

Operator
EmpresaC #3

Figure 4.7: Keeper Details Information Sidebar.

After clicking in a Keeper's icon, the sidebar shown in Figure 4.7 opens up and it is possible to see the detailed information about that driver's operation. Apart from seeing his name, ID, current status, license plate, vehicle's brand, model, and schedule, some other helpful information can be interacted. This includes a phone number, so that the user can quickly call the Keeper if necessary. It also shows the Operator name and its ID, allowing to open a new information sidebar about it.


Draft Details

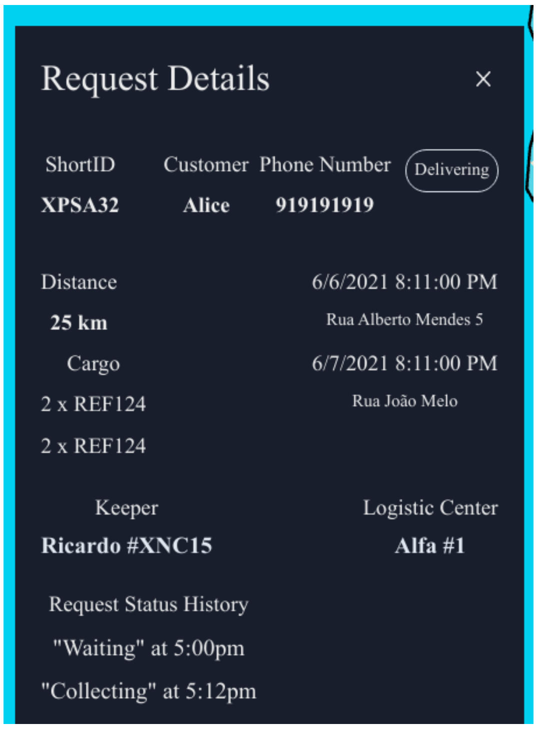
ShortID	Customer	Phone Number
POAZ12	Paul	919191919

Distance
25 km
Cargo
2 x REF124
2 x REF124

6/6/2021 8:15:48 PM
Rua Alberto Mendes 5
6/7/2021 8:15:48 PM
Rua João Melo

Figure 4.8: Draft Details Information Sidebar.

Like the Request details information sidebar, the Draft's one has similar information but without a few topics. In Figure 4.8, all essential information about a Draft is there, including its customer name, ID, phone number, and time and location of each of the points of a potential Request (pickup and dropoff).


Request Details ×

ShortID Customer Phone Number Delivering
XPSA32 Alice 919191919

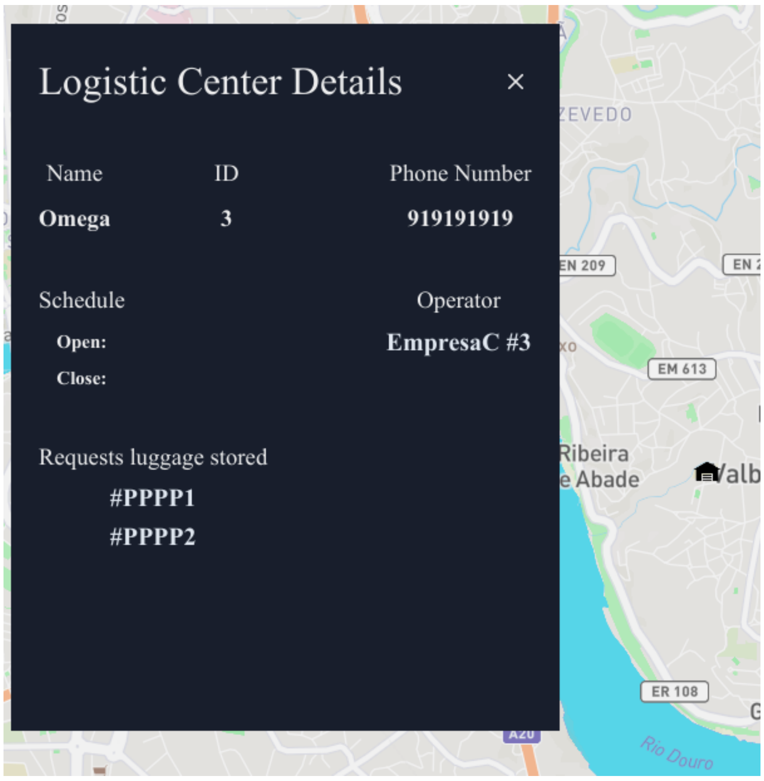
Distance 6/6/2021 8:11:00 PM
25 km Rua Alberto Mendes 5
Cargo 6/7/2021 8:11:00 PM
2 x REF124 Rua João Melo
2 x REF124

Keeper Logistic Center
Ricardo #XNC15 Alfa #1

Request Status History
"Waiting" at 5:00pm
"Collecting" at 5:12pm

Figure 4.9: Request Details Information Sidebar.

In Figure 4.9,there is an information sidebar from a random Request. Again, just like the Keeper's sidebar, there are some clickable elements, including phone number, Logistic Center, and its associated Keeper. If the user decides to click on the phone number, it redirects him to make a call through his phone, and if he clicks on one of the other options, the map zooms in to those respective elements, and a new sidebar is opened. Aside from these options, there is the usual information like its ID, customer name, and status. Specific information about this Request can also be found, such as the total distance, the current and previous status, how much luggage belongs to it, and the pickup and dropoff date and location.


Logistic Center Details ×

Name	ID	Phone Number
Omega	**3**	**919191919**

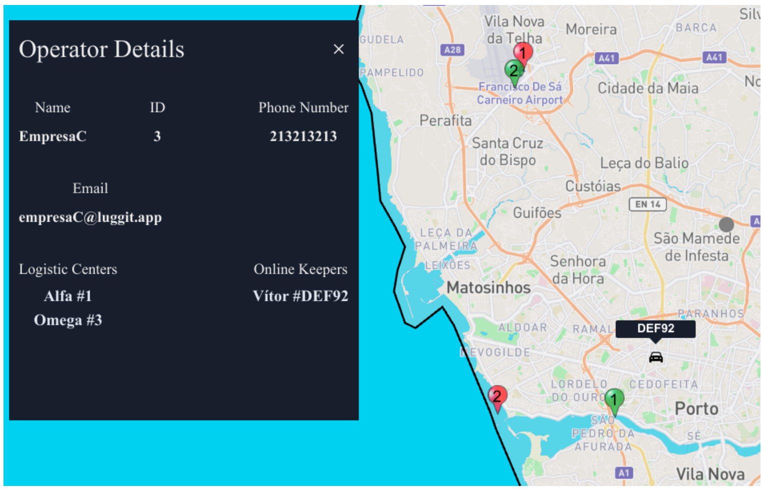
Schedule
Open:
Close:

Operator
EmpresaC #3

Requests luggage stored
#PPPP1
#PPPP2

Figure 4.10: Logistic Center Details Information Sidebar.

In Figure 4.10, there is the Omega's Logistic Center details information sidebar. There are two clickable options, namely the service phone number and the Operator. Clicking the Operator option, it zooms out to display all the icons belonging to that Operator, and it opens a new sidebar regarding it. It is also possible to see the Logistic Center's name, ID, schedule, and luggage stored in the building.


Operator Details ×

Name	ID	Phone Number
EmpresaC	**3**	**213213213**

Email
empresaC@luggit.app

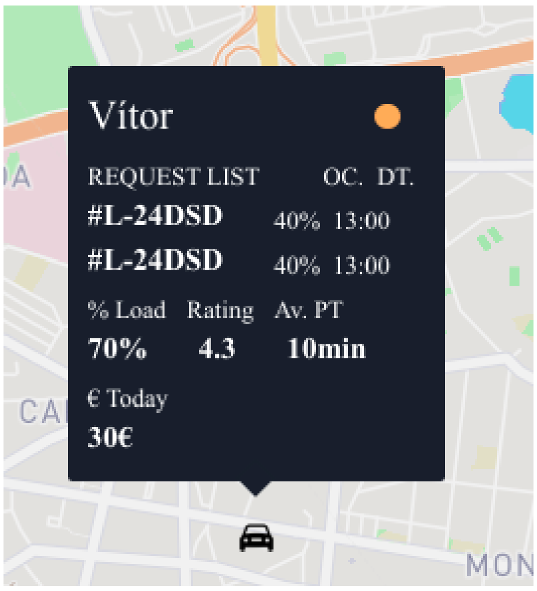
Logistic Centers	Online Keepers
Alfa #1	**Vítor #DEF92**
Omega #3	

Figure 4.11: Operator Details Information Sidebar.

There is the Operator information sidebar in Figure 4.11. From this figure, there is the map is zoomed out to show the icons belonging to this Operator and the sidebar. It shows its name, ID and, just like the others, it has clickable elements. These include all Logistic Centers, phone number, email, and online Keepers, allowing to interact with the system in a helpful way managing logistics.

4.3.1.1.5 Popups

Every element on the map, whether Request markers, a Draft marker, Operation Centers or Keepers, has a popup. This popup is activated when the mouse hovers the icon and displays a set of general information. This is a great way to view just the primary information about a specific element without opening the complete detailed information sidebar. Here are all of the available popups during a typical operating day.


Vítor

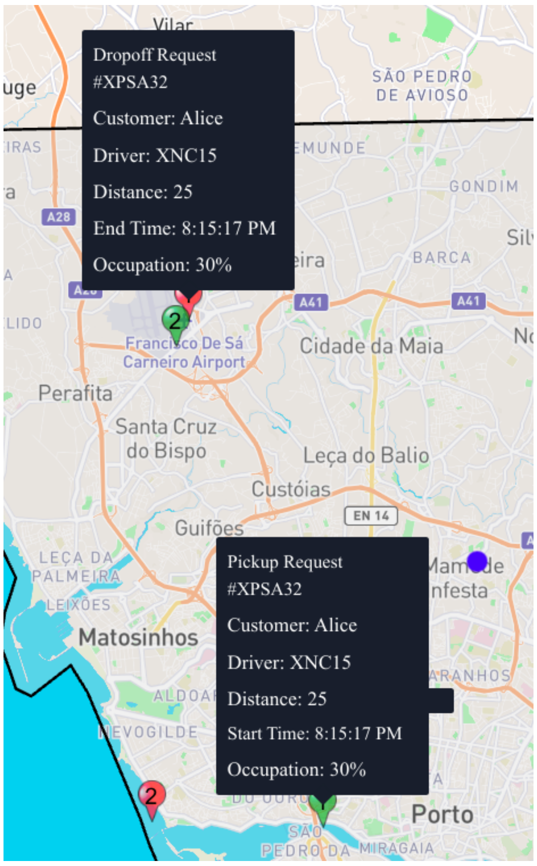
REQUEST LIST OC. DT.
#L-24DSD 40% 13:00
#L-24DSD 40% 13:00

% Load Rating Av. PT
70% 4.3 10min

€ Today
30€

Figure 4.12: Example of a Keeper's Popup.

In Figure 4.12, there is a Keeper's popup in service. It shows its name, capacity load of the vehicle, rating, average acceptance time, the profit earned that day, and its Requests list, including its ID, occupation, and departure hour. The significant difference between Keepers and other elements is that the Keeper always has an active popup. When the mouse hovers the car icon, it shows the popup that is in Figure 4.12; otherwise, it shows a similar popup to the Draft's one that is in Figure 4.14, with just its ID.


Dropoff Request
#XPSA32
Customer: Alice
Driver: XNC15
Distance: 25
End Time: 8:15:17 PM
Occupation: 30%

Pickup Request
#XPSA32
Customer: Alice
Driver: XNC15
Distance: 25
Start Time: 8:15:17 PM
Occupation: 30%

Figure 4.13: Example of a Request's Popup.

In Figure 4.13, it is possible to see how the popups' implementation was carried out. When the mouse hovers either one of the markers (red or green), two popups are activated: the dropoff location and the other the pickup point. This was intended for the user to have an easier time understanding which markers belong to each Request. In both popups, there is the primary information about a Request, including its ID, customer name, Keeper, distance, occupation, and start and end times.

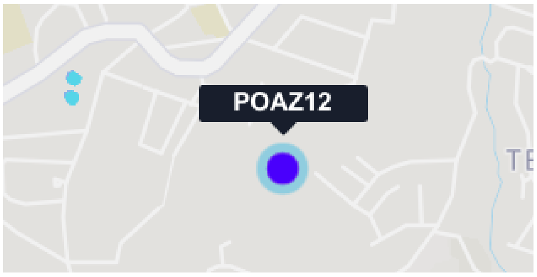


Figure 4.14: Example of a Draft's Popup.

In Figure 4.14, we can see a simple popup above the circle with an active ripple effect. The popup displays the Draft ID when the mouse hovers it to overwhelm the user with too much information about Drafts.

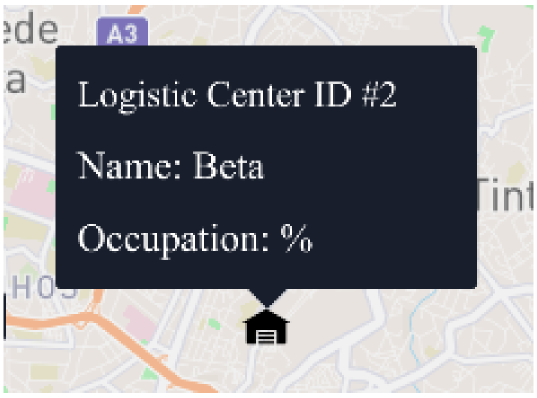


Figure 4.15: Example of a Logistic Center's Popup.

Regarding the Logistic Center Popup, it is the same as other popups. Above the logistic center icon, it shows information in regards to its ID, name, and current occupation, as seen in Figure 4.15.

4.3.1.1.6 Map Legend

The legend map is set up in the bottom left corner of the map, and it is helpful for a new user to know what each element means. The user, with a glance, can now understand the meaning of each icon displayed on the map. The legend map can be seen in Figure 4.6, showed previously.

4.3.1.1.7 City Clock

Although this feature seems simple, it was developed to facilitate the user's usability. Since Requests and Drafts have the times associated with them, the user needs a quick way to compare them. The local time city clock is displayed on the top right corner of the dashboard, as seen in Figure 4.6, and it is dynamic, changing according to the Operation Area displayed. It uses a light JavaScript library that looks up time zones given the name of a city. With the user's local time and the time zone of the given city, it is then calculated the time of the Operation Area's city.

4.3.1.2 Connection to the other Component

As mentioned earlier, the connection between components was achieved through Resgate and REST Endpoints. Whenever it is necessary to retrieve non-real-time data, the component makes a REST request. This data not only includes Logistic Centers, Operation Areas, and Operators but also incorporates data that is seemed as real-time, like Keepers, Drafts, and Requests. This is built to make the dashboard always have data available as soon as the user turns it on, allowing it to have fast boot time. Another option would be having to request data when the dashboard was accessed, which would make it slower to load.

In regards to real-time data, this component uses Resgate to listen for new events coming from outside sources. It listens to a topic already pre-defined and updates data when it is received.

4.3.2 Data Processing Component

The Data Processing Component provides the project's backend to support communication between LUGGit's APIs and the Holistic Platform Component. The backend has many features and functionalities, and it is an essential piece of work. The REST endpoints allow data retrieval from other components and can even be extended to other solutions. When a request comes through an endpoint, the provided data is fetched from the storage solution, as will be detailed below. For real-time data communication, events are sent on a pre-defined subject, with the purpose of clients getting that data almost immediately.

This section will include details about each endpoint and its purpose, an explanation of the event-based communication used, and storage.

4.3.2.1 REST Endpoints

This component possesses multiple endpoints for providing data to outside sources. Every endpoint is accessible and working through a defined URL containing a wealth of information, making it very easy to fetch information from this component's server. All information fetched from these endpoints comes with the JSON format, composed of the JSON API guidelines and the conventional status code and the rest of the requested data.

In Figure 4.16, it is possible to see the several endpoints that will be mentioned above.

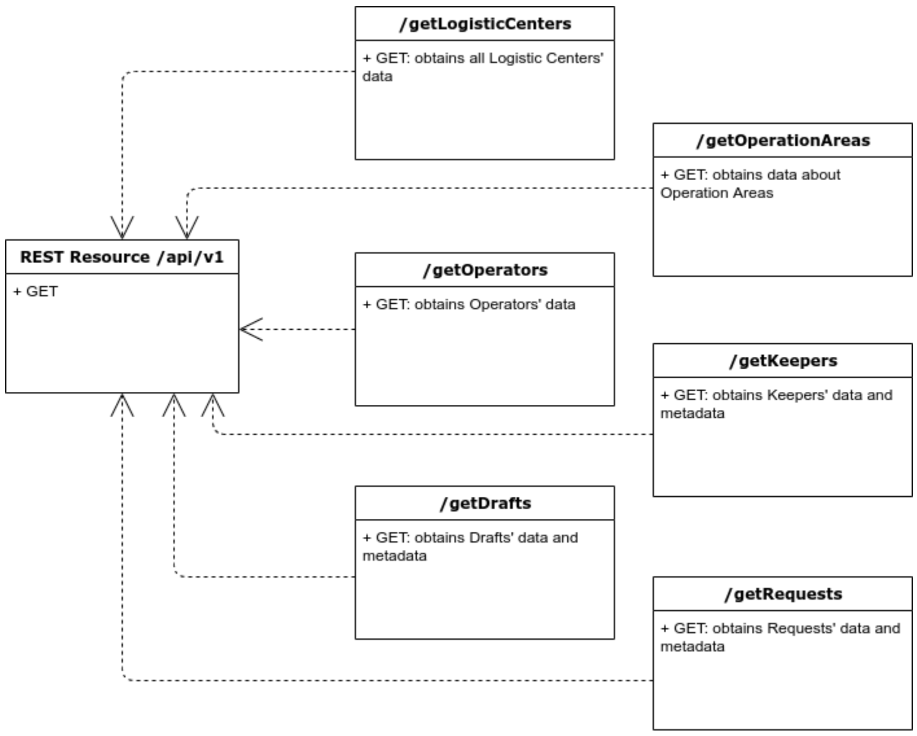

/getLogisticCenters
+ GET: obtains all Logistic Centers' data
/getOperationAreas
+ GET: obtains data about Operation Areas
REST Resource /api/v1
+ GET
/getOperators
+ GET: obtains Operators' data
/getKeepers
+ GET: obtains Keepers' data and metadata
/getDrafts
+ GET: obtains Drafts' data and metadata
/getRequests
+ GET: obtains Requests' data and metadata

Figure 4.16: Global Architecture of REST Endpoints.

getLogisticCenters

This endpoint's name is very straightforward as it supplies all the Logistic Centers' data. This data includes their IDs, names, locations, and Operator IDs. This information does not need to be updated regularly as it only changes when there are new Logistic Centers to LUGGit's operation.

getOperationAreas

The getOperationAreas endpoint is responsible for providing data regarding Operation Areas and their attributes. It consists of their IDs, names, and polygon locations, forming the available operation area on the map.

getOperators

This REST endpoint, when requested, delivers data about all Operators and their IDs, names, emails, phone numbers, list of Keepers, and the list of Logistic Centers.

getKeepers

This endpoint and the next two were built to provide a way to access data stored in the database. Most of these elements are updated in real-time through event-based communication, as will be discussed next, but there was also a need to retrieve static data as soon as the dashboard was accessed.

Keepers' data include their IDs, names, locations, requests list, luggage available capacity, rating, status, operator IDs, phone numbers, license plate numbers, brands, models, and schedules.

getDrafts

The getDrafts endpoint provides all stored Drafts in the database, specifically their IDs, customer names, start and end times/locations, distance, and luggage occupation in the automobile.

getRequests

Like the two previous endpoints, getRequests provides an address to retrieve information about Requests in the storage unit. Each Request has the same Drafts information, also adding its designated Logistic Center ID, Keeper ID, and status.

4.3.2.2 Event-based Communication

This communication type was used for real-time data fetching between the components built in this dissertation and between the Data Processing Component and LUGGit's APIs. This is done with the technology NATS through subject-based messaging and events. This subject is just a string of characters that form a name the publisher and subscriber can use to find each other.

Using a TypeScript Node.js client for the NATS messaging system, this component will subscribe to the defined subject from LUGGit's APIs to listen to new events. It will start listening as soon as the service starts, and it will always stay on. These events can arrive at any time, and when they do, the service will send the same event to another subject, occurring a data transmission ripple effect.

Three subjects will be talked about in further detail in this section.

Keepers

The subscription of new Keepers' events uses the subject "logistic-platform-api.requests" and will listen to all updates regarding Keepers. All new events include six different situations, and they are executed when LUGGit's API publishes them. The one that happens first is when a new Keeper goes online in the system. Immediately, a new event is published so that this platform can receive it in real-time. After that, all new information updated regarding that Keeper is published as a new event that the Data Processing Component then receives. Every time a new location, status, request list, capacity, or schedule are updated, real-time data is essential for the user to know.

After data is processed and normalized, it is sent through the subject "event.logistics-dashboard.keepers.change" to communicate with the other component. The "event." and ".change" are obligatory to use since the frontend component uses the technology Resgate to communicate. These words are a requirement of communication, and without them, it would not work.

Requests

When it comes to Requests, the subject used to communicate between LUGGit's API is "logistic-platform-api.requests". Apart from receiving events when a new Request is published, it also receives its Keeper ID, details about times and locations, or its status change. The rest of the steps are the same as the other topics.

Just like the Keepers topic, after data is normalized, it is sent through the subject "event.logistics-dashboard.requests.change".

Drafts

Since Drafts do not have much data to be changed, LUGGit's API sends the entire Draft's information and the verification if a new one is executed in the Holistic Platform Component. It subscribes to new events with "logistic-platform-api.drafts" and it publishes the same data with the subject "event.logistics-dashboard.drafts.change".

4.3.2.3 Storage

A big part of this component is storage, and for this area, it was decided to use Redis because of the reasons already mentioned in the previous chapter. Keeping storage as a cache means that all data is stored for a defined amount of time, called Time To Live, which is later removed not to overflow the database with unnecessary resources. Time to live directly impacts the server's load time (cached data loads faster), as well as content freshness (data cached for too long can become stale), and due to this, a resource's cache time to live should be adjusted based on how often it is used. Considering these points, it was decided that the amount of time data is stored in this cached database is 6 hours.

As soon as a new event is received, it is stored in the database using a Redis module called RedisJSON. This module allows storing, updating, and fetching JSON values from Redis keys, making it easier and faster to access data.

Conceptually, Redis is based on the key-value database paradigm meaning that every piece of data is associated with a key, either directly or indirectly. All data stored in this project follows the key-value structure. Every key is composed by its element type followed by the

ID; for instance, a Request with an ID "REF123" is stored in "requestREF123", facilitating the way data is retrieved.

4.4 Validation Tests

In this section it will be mentioned all the validation tests that were obtained throughout the development process, including user and technological tests. These include quality unit tests of all software components as well as tests done by the project's user/stakeholders. It will also incorporate user stories based on what the platform can provide to the user. These user stories will guide the unit tests, as well as the stakeholder's validation, focusing on each topic and on each functionality that the platform can provide.

4.4.1 User Stories

The user stories that are going to be described below led to the automatic tests of the platform. This system is designed to be used by a final user, meaning that a person with access to the credentials can use it. This user or an administrator is responsible for controlling LUGGit's logistic operations. He or she will be using the platform to understand better what happens in the company's operation area.

This dissertation's main focus is to build a holistic logistics platform, relying on the user interface to achieve it. The authentication process is not related to the objectives and challenges and, because of that, these user stories are based on the supposition that the user is already authenticated. These user stories are about the system administrator after he or she is authenticated and ready to use the dashboard:

1. As an authenticated user, I can see how many Keepers, Operation Centers, and Operators are in operation at any given time. I can also verify how many Drafts and Requests are active whenever I want.

2. As an authenticated user, I can check the locations of Keepers, Requests, Drafts, and Operation Centers at any given time.

3. As an authenticated user, I can filter out one or more Keepers between all available.

4. As an authenticated user, I can filter out one or more Operation Centers between all available.

5. As an authenticated user, I can filter out one or more Requests between all available.

6. As an authenticated user, I can filter out one or more Drafts between all available.

7. As an authenticated user, I can filter out Keepers according to their Operator.

8. As an authenticated user, I can verify when a new draft or request is made.

9. As an authenticated user, I can check Keeper's information about his vehicle occupation, his requests, and the according to times and positions.

10. As an authenticated user, I can check a Draft's information about its point locations, customer details, and times.

11. As an authenticated user, I can check a request's information about its point locations, customer's details, and its associated Keeper.

12. As an authenticated user, I can check an Operator's information about its Keepers and its Operation Centers.

13. As an authenticated user, I can check an Operation Center's information about its associated requests, capacity, and Operator.

4.4.2 Stakeholder

The stakeholder's feedback can make a big difference when talking about incremental improvements in functionalities, as well as when validating this dissertation's work. The development processes of this dissertation were iterated with the continuous help of LUGGit's logistic team and that was the best solution for the validation tests. This was a great help as it provided the essential feedback to improve. Every iteration and advancement was carefully analyzed and reviewed by their team. The stakeholder's feedback was provided according to the user stories pre defined before.

With those iterations and the feedback from LUGGit's logistic team, it was possible to make the platform better, in terms of efficiency, usability and overall design.

Regarding the user story number 2 in respect to the visualization of the Keepers in real time on the map, the stakeholder's feedback is that the Keeper moves correctly in real time, but there is no interaction when a Keeper is going online or offline. There is also a positive feedback by displaying the informations on "KeeperInfo", on "KeeperPopup" and the "Request List", that shows all the requests that are being made by the Keeper, when referring to the user story number 9. The collection and delivery services, the list of the services are being shown and the informations on "RequestInfo" and "RequestPopup" are fine.

The stakeholder also referred that the routes of the Keepers should be well defined and with different colors to have a better perception. The idea was to have the possibility to visualize the services and the locations that the Keepers were going through and could be a future work.

When performing the user story number 6, the locations of the emergent services (Drafts) are well displayed with all the information data, but could also have the responses data, the Keepers availability and the estimated collection time by the system. Conforming to the user story number 3, when clicking on the Keeper, it is possible to see his or her information data and it should also have displayed a synthesis of the services, the affected time and the route information. When referring to the user story number 10, regarding to the User, it is possible to visualize the distance and time data, but it also should be possible to see an estimation if the estimated time will be successfully fulfilled or not.

Concerning the alerts that provide to anticipate the problems, the alerts of new Requests and Drafts are well defined, but it should have, in future work, warnings of potential problems, like for example, if the Keeper has the car's luggage full, he or she should leave the bags in the warehouse or when happens a Draft out of the route of the Keeper.

In conclusion, the overall stakeholder's feedback is that the project has a good concept, the user stories pre defined before were reflected on the platform and it is better than the actual solution of operations with the objective of being in real time, but it should have a better visual design to facilitate the operation. The future work should consist in implementing warnings to be able to make decisions and have more informations displayed in the visual design.

4.4.3 Technological

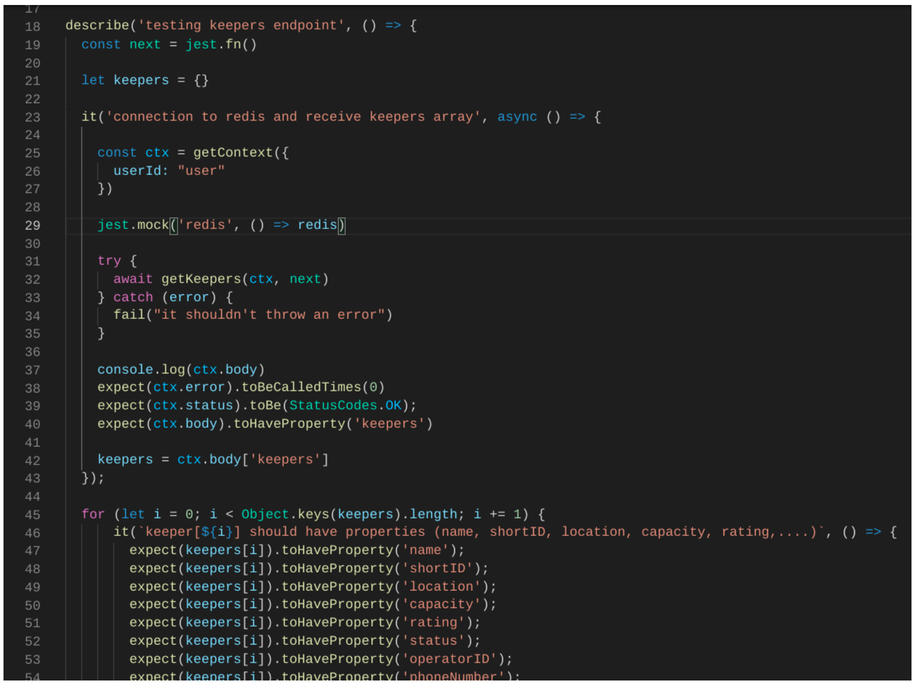
The technological tests were made to every part of software available, inside of Data Processing Component and Holistic Platform Component. These tests were built throughout the development process in order to facilitate and automate error checking in newly added system functionalities. Unit tests in Data Processing Component focused on testing endpoints and real-time communication, and tests in the Holistic Platform Component focused on every type of component, including buttons, activation of graphical control elements, and other usability features.

The main testing framework used for both Components was Jest[22], in conjunction with other more scenario-specific tools, like nats-mock[23] to test real-time communication, redis-mock[24] to test database connection and enzyme[25] to test React components' output. Jest is an open source unit testing framework for JavaScript that does not depend on other JavaScript frameworks or web browsers, and it is able to test both JavaScript versions used in website development and in Node.js-based applications.

These tools simulate connections to the database, input, or any other type of object that is required to execute a component. Initially, a controller is created, which will serve as the connection between the mock objects and the component to be tested. Then, the tests are created, which are composed of specific expectations, whether be it responses, errors, or data. Once all this is configured, the tests can be executed, and the results are displayed, showing if it fails or if it passes.

In Figure 4.17, it is possible to see an example of a unit test, which is performed in the Data Processing Component. This test, in particular, is testing an endpoint, more specifically its connection to the database and if the response obtained is the expected with all its properties.

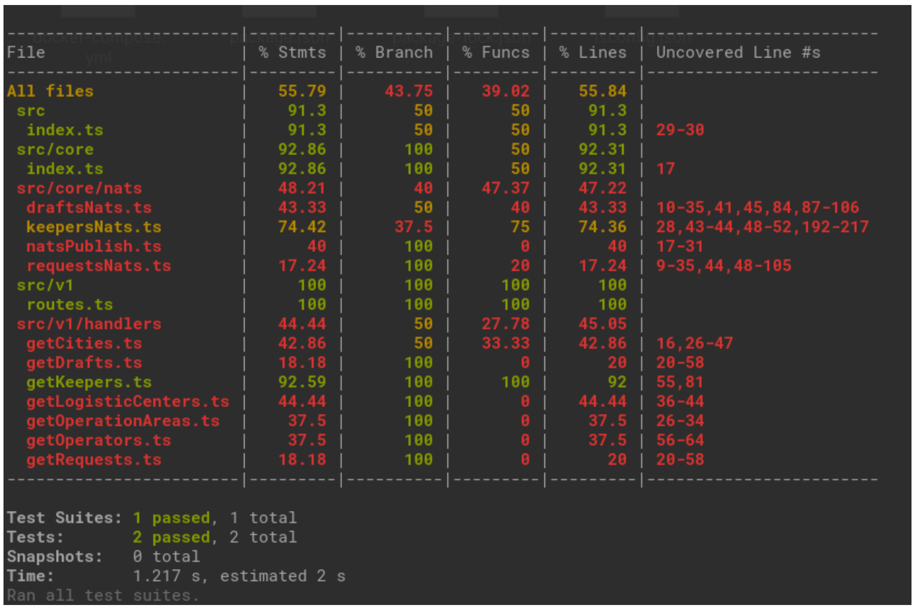
[22]https://jestjs.io/
[23]https://www.npmjs.com/package/mock-nats-client
[24]https://www.npmjs.com/package/redis-mock
[25]https://www.npmjs.com/package/enzyme



```javascript
describe('testing keepers endpoint', () => {
  const next = jest.fn()

  let keepers = {}

  it('connection to redis and receive keepers array', async () => {

    const ctx = getContext({
      userId: "user"
    })

    jest.mock('redis', () => redis)

    try {
      await getKeepers(ctx, next)
    } catch (error) {
      fail("it shouldn't throw an error")
    }

    console.log(ctx.body)
    expect(ctx.error).toBeCalledTimes(0)
    expect(ctx.status).toBe(StatusCodes.OK);
    expect(ctx.body).toHaveProperty('keepers')

    keepers = ctx.body['keepers']
  });

  for (let i = 0; i < Object.keys(keepers).length; i += 1) {
      it(`keeper[${i}] should have properties (name, shortID, location, capacity, rating,....)`, () => {
        expect(keepers[i]).toHaveProperty('name');
        expect(keepers[i]).toHaveProperty('shortID');
        expect(keepers[i]).toHaveProperty('location');
        expect(keepers[i]).toHaveProperty('capacity');
        expect(keepers[i]).toHaveProperty('rating');
        expect(keepers[i]).toHaveProperty('status');
        expect(keepers[i]).toHaveProperty('operatorID');
        expect(keepers[i]).toHaveProperty('phoneNumber');
```

Figure 4.17: Example of a Unit Test.

In the case of Data Processing Component, each unit test focuses on the functionality of one endpoint individually. This specific one is about getKeepers and it starts by creating a new user context and simulating a connection to the database, Redis. Then, it tries to call the method getKeepers with the previously created context as an argument. If it fails, it throws immediately an error, otherwise it moves to the next phase. After verifying that there are no errors, the status code is 'OK' and that it has a property named keepers, it checks if each object Keeper has all its elements accordingly, namely name, short ID, location and many others.



```
------------------------|---------|----------|---------|---------|------------------------
File                    | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
------------------------|---------|----------|---------|---------|------------------------
All files               |   55.79 |    43.75 |   39.02 |   55.84 |
 src                    |    91.3 |       50 |      50 |    91.3 |
  index.ts              |    91.3 |       50 |      50 |    91.3 | 29-30
 src/core               |   92.86 |      100 |      50 |   92.31 |
  index.ts              |   92.86 |      100 |      50 |   92.31 | 17
 src/core/nats          |   48.21 |       40 |   47.37 |   47.22 |
  draftsNats.ts         |   43.33 |       50 |      40 |   43.33 | 10-35,41,45,84,87-106
  keepersNats.ts        |   74.42 |     37.5 |      75 |   74.36 | 28,43-44,48-52,192-217
  natsPublish.ts        |      40 |      100 |       0 |      40 | 17-31
  requestsNats.ts       |   17.24 |      100 |      20 |   17.24 | 9-35,44,48-105
 src/v1                 |     100 |      100 |     100 |     100 |
  routes.ts             |     100 |      100 |     100 |     100 |
 src/v1/handlers        |   44.44 |       50 |   27.78 |   45.05 |
  getCities.ts          |   42.86 |       50 |   33.33 |   42.86 | 16,26-47
  getDrafts.ts          |   18.18 |      100 |       0 |      20 | 20-58
  getKeepers.ts         |   92.59 |      100 |     100 |      92 | 55,81
  getLogisticCenters.ts |   44.44 |      100 |       0 |   44.44 | 36-44
  getOperationAreas.ts  |    37.5 |      100 |       0 |    37.5 | 26-34
  getOperators.ts       |    37.5 |      100 |       0 |    37.5 | 56-64
  getRequests.ts        |   18.18 |      100 |       0 |      20 | 20-58
------------------------|---------|----------|---------|---------|------------------------

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.217 s, estimated 2 s
Ran all test suites.
```

Figure 4.18: Results of a Unit Test.

In Figure 4.18, it is possible to see all the files composed by the Data Processing Component on the left column. The files shown with a green color are the ones that were used to run this specific test and we can see that the test passed, meaning it ran successfully without any errors. Since the unit test was making sure every functionality of the endpoint getKeepers, we can acknowledge the endpoint works.

Chapter 5

Conclusion

At the end of this dissertation, it is time to talk about the general overview of this work project. At the beginning of this work I started by defining the problems, challenges and objectives, alongside explaining where LUGGit enter in the technology realm. Afterwards, I familiarized ourselves with the data processing, data visualization techniques and related technologies, that are standard tools in almost every holistic platform project. Then I started approaching the problem by defining use cases and requirements, with the logistics team's help. After that, it was easier to construe the system architecture and the platform's wireframes, which would later help the development process. The platform was designed and implemented with separation of concerns in mind, separating data processing from frontend service implementation. During this process, I came to the conclusion that feedback throughout each work iteration helps greatly the development process, making it faster and with better quality.

However, the main goal of this dissertation was to create a Holistic Logistics Platform in order to replace LUGGit's previous solution, which consisted in a non-customizable inefficient data displaying platform. In that regard, interaction with a customized working holistic logistics platform was accomplished, creating a complete set of workflows to display real-time data.

At the end of the development process, the platform was tested by LUGGit's operations team and the platform was adjusted according to their feedback. The backend part was thoroughly tested in all its features and services, with automated unit testing. Finally, the architecture design, allow extensibility by abstracting the data flow from the service, creating a content focused development where data access is effortlessly available.

5.1 Future Work

I believe this work has many different changes, adaptations, and experiments for future work. They were not possible due to time limits. Learning from the beginning all these new technologies can be very time-consuming, and because of that, I believe that I will be more efficient in the future.

To a large extent, this project was based on the frontend of the application, and it was left with some ideas of that matter that I would have liked to implement. These ideas are:

1. A helpful feature that could be implemented in the future is a routing system of each driver considering the traffic in real-time. If a Keeper is following a path that has more

traffic than another, the dashboard's user can alert the Keeper to take the better route. This system can be built using machine learning to improve the learnability of new routes, in view of efficiency.

2. The platform has two tabs that were not developed further, namely "Request List" and "Analytics". These features were not the main focus of a real-time holistic dashboard, however in the future, a page for each of these tabs would come handy to the analytics for LUGGit's team.

3. The application was made in a way, so that was intuitive. If we see the legend map, we can easily understand the map's information. Initially, this is perfect for the person who is using it, but it can be not very pleasant in the long run since the person gets experienced and does not need to see the Legend Map info anymore. So it could be interesting in the future to add the functionality hide/show the specific object.

4. Similar to the previous point, the adaptation I will talk about now also needs a hide/show specification. The adaptation for the future I am talking about is the possibility to hide/show the sidebar.

5. Regarding the Keeper route, it has the pickup and the dropoff locations. A future step-up would be to draw the path between them in a non-confusing way to the eye to be a little wiser to the user.

6. Usually, an application with a notification system can access previous ones, and, unfortunately, that is not the case with this application. Nevertheless, I believe it was not a crucial feature, so time was given to other matters.

7. Using other forms of data to benefit their visualization, can drastically improve logistics' performance and organization. For example, using meteorological data in relation to the number of requests and how it can impact them or using data about traffic in a certain area can indicate that it might not be the fastest route for a Keeper to take.

These are the improvements that, with the help of LUGGit's team, came up with. Nevertheless, there is the possibility of other experiments that can be exploited in the future to improve the system's capabilities.

Bibliography

[1] K. Jackson, “What is real-time network monitoring? | helpsystems,” 2020. [Online]. Available: https://www.helpsystems.com/blog/what-does-real-time-network-monitoring-mean-exactly (visited on 09/27/2020).

[2] “Chapter 2. general control,” [Online]. Available: https://web.archive.org/web/20100607105632/http://www.faa.gov/air_traffic/publications/atpubs/ATC/atc0201.html (visited on 12/30/2020).

[3] “Real-time monitoring,” [Online]. Available: https://www.next4biz.com/real-time-monitoring/ (visited on 06/22/2021).

[4] A. Trafton, “In the blink of an eye | mit news | massachusetts institute of technology,” 2014. [Online]. Available: https://news.mit.edu/2014/in-the-blink-of-an-eye-0116 (visited on 10/20/2020).

[5] M. Aparicio and C. J. Costa, “Datavisualization_acm_sigdoc_cdq-3-1-nov-2014.pdf,” 2014. DOI: 10.1145/2721882.2721883. [Online]. Available: https://www.researchgate.net/publication/269103846 (visited on 10/18/2020).

[6] “8 remarkable early maps - history,” [Online]. Available: https://www.history.com/news/8-remarkable-early-maps (visited on 10/19/2020).

[7] M. Mekhatria, “Tableau poléometrique - highcharts.” [Online]. Available: https://www.highcharts.com/blog/tutorials/156-tableau-poleometrique/ (visited on 11/11/2020).

[8] “How data visualization designers make content across industries,” [Online]. Available: https://killervisualstrategies.com/blog/category/data-visualization-in-industry (visited on 10/23/2020).

[9] J. G. Stadler, K. Donlon, J. D. Siewert, T. Franken, and N. E. Lewis, “Improving the efficiency and ease of healthcare analysis through use of data visualization dashboards,” *Big Data*, vol. 4, pp. 129–135, 2 Jun. 2016, ISSN: 2167-6461. DOI: 10.1089/big.2015.0059. [Online]. Available: http://www.liebertpub.com/doi/10.1089/big.2015.0059 (visited on 10/23/2020).

[10] “Ponto de situação atual em portugal - covid-19,” [Online]. Available: https://covid19.min-saude.pt/ponto-de-situacao-atual-em-portugal/ (visited on 10/16/2020).

[11] “3 ways the most successful marketers use data visualization,” [Online]. Available: https://killervisualstrategies.com/blog/digital-marketer-data-visualization.html (visited on 11/04/2020).

[12] "The analytics advantage we're just getting started." [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Deloitte-Analytics/dttl-analytics-analytics-advantage-report-061913.pdf (visited on 11/10/2020).

[13] M. Krisper, S. Oberauer, and T. Schlager, "Open data vis in browser," 2012. [Online]. Available: https://courses.isds.tugraz.at/ivis/surveys/ss2012/g3-survey-open-data-vis.pdf (visited on 11/26/2020).

[14] H. Bast, P. Brosi, and S. Storandt, "Travic: A visualization client for public transit data," vol. 04-07-November-2014, Association for Computing Machinery, Nov. 2014, pp. 561–564, ISBN: 9781450331319. DOI: 10.1145/2666310.2666369. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2666310.2666369 (visited on 11/14/2020).

[15] "Real-time data visualization and data exploration," [Online]. Available: https://www.striim.com/real-time-data-visualization-data-exploration/ (visited on 06/24/2021).

[16] R. Novotny, "What is real time data? how real time information is essential in the field," Aug. 2018, unpublished. [Online]. Available: https://esub.com/blog/what-is-real-time-data-why-real-time-information-is-essential-in-the-field/ (visited on 06/25/2021).

[17] "Everything about data processing | definition, methods, types & application," [Online]. Available: https://planningtank.com/computer-applications/data-processing (visited on 06/26/2021).

[18] M. Baron, "Everything about data processing | definition, methods, types and application," [Online]. Available: https://datascience.foundation/sciencewhitepaper/raw-data-collection-2020-principles-and-challenges (visited on 06/26/2021).

[19] "What is data processing: Types, methods, steps of data processing cycle," [Online]. Available: https://www.simplilearn.com/what-is-data-processing-article (visited on 12/17/2020).

[20] "What are the differences between data processing, data preprocessing and data wrangling?," [Online]. Available: https://www.i2tutorials.com/what-are-the-differences-between-data-processing-data-preprocessing-and-data-wrangling/ (visited on 06/28/2021).

[21] J. M. C. D. Cruz and R. Público, "Simulador de mensagens aftn e oldi-transmissãfo," 2010. [Online]. Available: https://repositorio.ul.pt/bitstream/10451/9256/1/ulfc104703_tm_Jo%c3%a3o_Miguel_Cruz.pdf (visited on 11/17/2020).

[22] "Lisatm," [Online]. Available: https://www.nav.pt/en/nav/air-navigation-services-1/traffic-management-systems/lisatm (visited on 11/05/2020).

[23] P. Brosi and H. Bast, "Real-time movement visualization of public transit data," 2014. (visited on 11/19/2020).

[24] . [Online]. Available: https://themeisle.com/blog/google-analytics-interface-explained/#:~:text=The%5C%20Google%5C%20Analytics%5C%20interface%5C%20explained,tables%5C%2C%5C%20charts%5C%2C%5C%20and%5C%20graphs (visited on 12/14/2020).

[25] "What is power bi?," [Online]. Available: https://powerbi.microsoft.com/en-us/what-is-power-bi/ (visited on 06/28/2021).

[26] “Data visualization with power bi - datacamp,” [Online]. Available: `https://www.datacamp.com/community/tutorials/data-visualisation-powerbi` (visited on 12/15/2020).

[27] X. Qin, Y. Luo, N. Tang, and G. Li, “Making data visualization more efficient and effective: A survey,” *The VLDB Journal*, DOI: `10.1007/s00778-019-00588-3`. [Online]. Available: `https://doi.org/10.1007/s00778-019-00588-3` (visited on 07/01/2021).

[28] H. Zhu, M. Zhu, Y. Feng, D. Cai, Y. Hu, S. Wu, X. Wu, and W. Chen, “Visualizing large-scale high-dimensional data via hierarchical embedding of knn graphs,” *Visual Informatics*, vol. 5, pp. 51–59, 2 Jun. 2021, ISSN: 2468502X. DOI: `10.1016/j.visinf.2021.06.002`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S2468502X21000292` (visited on 07/01/2021).

[29] Y. Wang, “Deck.gl: Large-scale web-based visual analytics made easy,” 2017. (visited on 12/26/2020).

[30] “Deck.gl | introduction,” [Online]. Available: `https://deck.gl/docs` (visited on 10/22/2020).

[31] “Our thoughts as mapboxgl js v2.0 goes proprietary | carto blog,” [Online]. Available: `https://carto.com/blog/our-thoughts-as-mapboxgl-js-2-goes-proprietary/` (visited on 12/23/2020).

[32] O. Eriksson and E. Rydkvist, “An in-depth analysis of dynamically rendered vector-based maps with webgl using mapbox gl js.” (visited on 11/04/2020).

[33] “Google maps vs open street maps comparison. who is winner?,” [Online]. Available: `https://agilestorelocator.com/blog/google-maps-vs-open-street-maps-comparison/` (visited on 12/28/2020).

[34] “From google maps to mapbox - dev,” [Online]. Available: `https://dev.to/korovka/from-google-maps-to-mapbox-4nfk` (visited on 12/28/2020).

[35] “Choosing a map api: Mapbox vs openstreetmap vs google maps,” [Online]. Available: `https://relevant.software/blog/choosing-a-map-amapbox-google-maps-openstreetmap/` (visited on 12/29/2020).

[36] im Studiengang Bachelor, B. PrÃ¼ferin, U. S. Zweitgutachter, M. B. geb Knoblauch, and E. Wohlgethan, “Supporting web development decisions by comparing three major javascript frameworks: Angular, react and vue.js,” 2018. [Online]. Available: `https://reposit.haw-hamburg.de/bitstream/20.500.12738/8417/1/BA_Wohlgethan_2176410.pdf` (visited on 12/08/2020).

[37] E. Saks, “Javascript frameworks: Angular vs react vs vue,” 2019. [Online]. Available: `https://www.theseus.fi/bitstream/handle/10024/261970/Thesis-Elar-Saks.pdf` (visited on 12/08/2020).

[38] “View of perfomance analysis of frameworks angular and vue.js,” 2019. [Online]. Available: `https://ph.pollub.pl/index.php/jcsi/article/view/1577/1263` (visited on 12/09/2020).

[39] M. Levlin, “Dom benchmark comparison of the front-end javascript frameworks react, angular, vue, and svelte,” Ãbo Akademi University, 2020. [Online]. Available: `https://www.doria.fi/bitstream/handle/10024/177433/levlin_mattias.pdf?sequence=2&isAllowed=y` (visited on 12/22/2020).

[40] J. G. C. D. Camargos, J. F. Coelho, P. Aramuni, and H. F. Villela, “Uma análise comparativa entre os frameworks javascript angular e react,” Sep. 2019. [Online]. Available: http://www.fumec.br/revistas/computacaoesociedade/article/view/7307 (visited on 12/23/2020).

[41] “Nonfunctional requirements - scaled agile framework,” [Online]. Available: https://www.scaledagileframework.com/nonfunctional-requirements/ (visited on 07/01/2021).

[42] “What is a transcompiler?,” [Online]. Available: https://www.computerhope.com/jargon/t/transcompiler.htm (visited on 05/13/2021).

[43] “Pros and cons of typescript: When and why it’s better than plain js | altexsoft,” [Online]. Available: https://www.altexsoft.com/blog/typescript-pros-and-cons/ (visited on 05/10/2021).

[44] “To type or not to type: Quantifying detectable bugs in javascript,” [Online]. Available: https://earlbarr.com/publications/typestudy.pdf (visited on 05/11/2021).

[45] “Next.js; what is it and why do we use it?,” [Online]. Available: https://www.clock.co.uk/insight/next-js-what-is-it-and-why-do-we-use-it (visited on 05/13/2021).

[46] “Which tool your logistics app needs: Mapbox or google maps platform,” [Online]. Available: https://yalantis.com/blog/mapbox-maps-ready-mobile-apps/ (visited on 05/14/2021).

[47] “Mapbox gljs,” [Online]. Available: https://www.mapbox.com/mapbox-gljs (visited on 05/14/2021).

[48] “Introduction to koa.js. what is koa.js? | by mohammed rishard | geek culture | medium,” [Online]. Available: https://medium.com/geekculture/introduction-to-koa-js-c332931c6b24.

[49] “What is koa.js?. as a cross-platform server-side runtime… | by mindfire solutions | medium,” [Online]. Available: https://medium.com/@mindfiresolutions.usa/what-is-koa-js-fc677df30795 (visited on 05/14/2021).

[50] “Redis: In-memory data store. how it works and why you should use it,” [Online]. Available: https://aws.amazon.com/redis/ (visited on 05/13/2021).

[51] “Publish-subscribe - nats docs,” [Online]. Available: https://docs.nats.io/nats-concepts/pubsub (visited on 05/13/2021).

[52] “Introduction - nats docs,” [Online]. Available: https://docs.nats.io/ (visited on 05/14/2021).