Universidade de Aveiro
2022

Lucas Seabra

Plataforma de gestão para sistemas de transportes cooperativos

Management platform for cooperative transport systems

Universidade de Aveiro
2022

Lucas Seabra

Plataforma de gestão para sistemas de transportes cooperativos

Management platform for cooperative transport systems

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 José Alberto Gouveia Fonseca, Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor João Miguel Pereira de Almeida, Investigador do Instituto de Telecomunicações.

Este trabalho foi realizado no âmbito duma parceria com a A-to-Be - Mobility Technology e da FCT/MCTES através de fundos nacionais e quando aplicável, cofinanciado por fundos comunitários no âmbito do projeto UIDB/50008/2020-UIDP/50008/2020.

Dedico este trabalho aos meus pais pois eles sempre me apoiaram neste longo e memorável percurso académico.

agradecimentos / acknowledgements

Agradeço à minha familia que sempre acreditaram em mim, em especial aos meus pais pois sem eles nada disto seria possivel e por isso estou-lhes eternamente grato. À minha namorada e aos meus amigos pelo apoio incansável que me deram nesta etapa tão dificil.

Palavras Chave Platforma IoT, RSU, C-ITS, Cidades Inteligentes

Resumo

A rápida evolução da tecnologia levou ao aumento da investigação em domínios como dispositivos inteligentes. Como resultado, a tecnologia da *Internet of Things* tem-se expandido rapidamente e está a ser usada para resolver desafios que necessitam de vastas quantidades de dados de muitos dispositivos. Com isto, o conceito de Cidades Inteligentes começou a ficar mais popular tendo por objetivo estabelecer um ecosistema saudável em que o funcionamento da cidade é melhorado. A Comissão Europeia adotou posteriormente os Sistemas Inteligentes de Transporte Cooperativos (C-ITS), o que permite aos utilizadores de estrada e controladores de tráfego partilhar informações e coordenar as suas ações. Mais adiante, o grupo do Instituto de Telecomunicações (IT) colocou e continua a instalar infraestrutura de comunicações (RSUs) nas estradas portuguesas. Estes dispositivos inteligentes podem ter radares, câmaras ou outros elementos acoplados que ajudam a determinar os padrões de tráfego. Esta dissertação surge no contexto de oferecer uma plataforma C-ITS capaz de atender às necessidades dos investigadores do IT e fornecer-lhes *feedback* bem como aos utilizadores regulares de estrada sobre o comportamento ou alertas do tráfego. Para o desenvolvimento de um sistema como prova de conceito, uma abordagem centrada no utilizador foi adotada, começando com a identificação dos utilizadores alvo e a recolha de requisitos-chave gerados a partir de cenários de uso. O sistema desenvolvido implementou uma arquitetura baseada em micro-serviços, composto por vários módulos responsáveis por manipular o fluxo de câmaras, privilégios de administrador, dados de radares, criação de alertas e diversos fluxos de dados. O sistema apresentado mostrou-se capaz de fornecer informações úteis aos utilizadores e administradores. Através da utilização da plataforma desenvolvida, os utilizadores podem ver um conjunto de informações como gráficos, interagir com as RSUs instaladas nas rodovias, gerar e exibir alertas e interagir com a transmissão ao vivo da câmara.

Keywords IoT Platform, RSU, C-ITS, Smart Cities

Abstract

The rapid evolution of technology led to increased research in domains such as smart devices. As a result, the Internet of Things technology has expanded rapidly and is being used to solve challenges needing vast quantities of data from many devices. With this, the Smart City concept became more popular with the goal of establishing a healthy ecosystem in which the functioning of the city is enhanced. The European Commission subsequently adopted Cooperative Intelligent Transport Systems (C-ITS), which enables road users and traffic controllers to share information and coordinate their actions. In addition, the Institute of Telecommunications (IT) group has placed and continues to install Road Side Units (RSUs) on Portuguese roads. These intelligent devices may have radars, cameras, or other elements that aid in determining traffic patterns. This dissertation emerges in the context of offering a C-ITS platform capable of addressing the demands of IT researchers and delivering feedback to them and regular road users on traffic behavior or alerts. For the development of a proof-of-concept system, a user-centered design approach was adopted, beginning with the identification of target users and the collection of key requirements generated from usage scenarios. The developed system implemented a micro-service architecture. The proof-of-concept is composed by several modules responsible for handling the camera stream, admin privileges, radars data, creation of alerts and several streams of data. The presented system showed to be capable of providing useful information to the users and administrators. By using the developed platform, users may see a collection of information as charts, interact with the RSUs installed on the roadways, generate and display alerts, and interact with the camera's live stream.

Contents

List of Figures

List of Tables

List of Acronyms

ITS	Intelligent Transport Systems
ITS-S	Intelligent Transport Systems Station
C-ITS	Cooperative Intelligent Transport Systems
RSU	Roadside Unit
OBU	On-Board Unit
IoT	Internet of Things
CCAM	Cooperative, Connected and Automated Mobility
IT	Institute of Telecommunications
ETSI	European Telecommunications Standards Institute
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure
V2P	Vehicle-to-Pedestrian
V2X	Vehicle-to-Everything
DSRC	Dedicated Short Range Communication
LTE	Long Term Evolution
IVS	Intelligent Vehicle System
3GPP	3rd Generation Partnership Project
OSI	Open Systems Interconnection
BLE	Bluetooth Low Energy
5G	Fifth Generation
IEEE	European Telecommunications Standards Institute
CV2X	Cellular Vehicle-to-Everything
MQTT	Message Queuing Telemetry Transport
LDM	Local Dynamic Map
CAM	Cooperative Awareness Message
VAM	Vulnerable Road Users Awareness Message
CPM	Collective Perception Message
DENM	Decentralized Environmental Notification Message
IPv6	Internet Protocol Version 6
IPv4	Internet Protocol Version 4
DEN	Decentralized Environmental Notification
CA	Cooperative Awareness
VRU	Vulnerable Road Users
TCP	Transmission Control Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
UI	User Interface
UI	User Interface
API	Application Programming Interface
UC	Use Case
HTTP	Hypertext Transfer Protocol
JWT	JSON Web Token

CHAPTER 1

Introduction

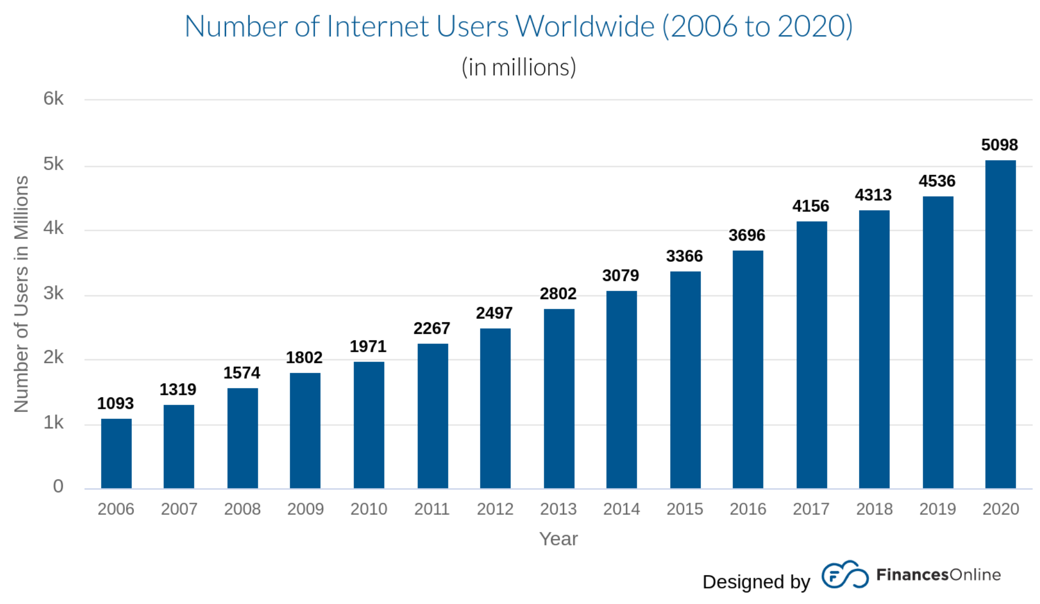
With the progression of time and, subsequently, of science, people's reliance on technology and also on the vehicles they use for transportation increased. If we consider combining these two, we may get millions of data that, once analyzed, can be displayed to provide feedback to individuals and assist them in recognizing traffic patterns.

1.1 Contextualization

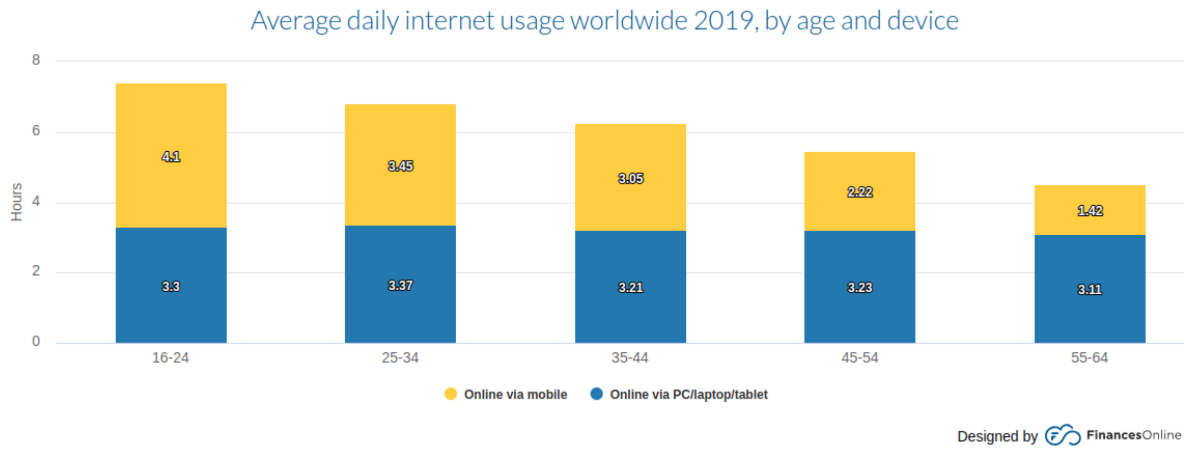
The most remarkable period after the industrial revolution was the digital age. This period was distinguished by the advancement of internet. Figure 1.1a depicts the number of users since 2006 to 2020 that are using internet and is possible to witness that roughly six out of every ten of the entire world's population currently have internet access [1]. In addiction, it is possible to conclude from figure 1.1b, that younger people spent more time on internet via mobile, i.e mobile apps, whereas older people spent more time on PCs or tablet, i.e web browsers/pages. Telecommunication industry exploded as well and after launch the new concept "smart phones" the use of this small devices, with high computational level, became more and more increasing hence is much more appeal to search the web via a gadget that allows portability.

In response to rising consumer expenditure, more research is being conducted not just on smartphones but also on other smart gadgets that may be used in other industries. By the middle of the 2010s, Internet of Things (Internet of Things (IoT)) technology was growing rapidly and being used to address problems requiring massive amounts of data from many devices. With the advancement of IoT technology, the Smart City concept is beginning to materialize. The objective of a Smart City is to create a thriving ecosystem where the operation of the city is improved, hence enhancing its efficiency, and where new possibilities emerge to enhance the lives of its residents [2].

Moreover, not only has people's communication changed drastically, but so has their mode of transportation. By the end of the first quarter of 2022, there will be around 1.45 billion vehicles on the planet, 1.1 billion of which will be passenger automobiles [3]. Following this,



(a) Number of internet users since 2006 until 2020 from [1]



(b) Number of hours each generation spends on internet via different platforms [1]

Figure 1.1: Figure (a) and (b) present the peoples usage on internet.

Cooperative, connected and automated mobility (Cooperative, Connected and Automated Mobility (CCAM)) philosophy was adopted and is characterized by the communication between vehicles, infrastructure, and other road users essential for enhancing the safety of future autonomous vehicles and ensuring their complete integration into the transportation system. This interaction falls within the scope of Cooperative Intelligent Transport Systems (Cooperative Intelligent Transport Systems (C-ITS)), which will enable road users and traffic controllers to exchange information and coordinate their actions using it.

1.2 Motivation

Prior to the realization of the project, and within the framework of the C-ITS platform designed by the European Commission, the IT group installed and continues to install RSUs throughout the Portuguese roadways. These smart devices may embed radars, cameras or other features that may help obtain traffic patterns. In addition, some of the most recent cars have been fitted with OBUs, which allow communication of the vehicles on the road and so making possible to provide awareness to the vehicles around him.

Even while the IT infrastructure is well-established in terms of device-to-device communication, it lacks the application-layer services necessary to give researchers and regular users with a visual representation of the data being sensed on the roadways. This is the most critical layer for establishing a Smart City solution, hence, if the acquired data is not made available to the public, it will be held without providing any advantages to the city.

This dissertation appears in the context of providing a C-ITS platform capable to address the needs for the IT researchers, i.e allow them to test the devices deployed on roads or the embedded gadgets within the RSUs, and, at the same time give feedback to them and to regular users of traffic behaviors or alerts on the roads.

1.3 Objectives

For the study presented in this dissertation, the following were the primary objectives:

- The system must support the streams of data produced by the C-ITS messages and allow the visualisation of them in real-time.
- Allow the system to support admin privileges making sure the researcher in the IT own features hidden from regular users.
- The system must be capable of support media stream hence admins may want to see the cameras stream.
- Researcher user may want to check RSU's status.
- Making possible to witness traffic behaviours through the use of the charts and allow users to choose the time frame.
- Create an application with information disclosure that is simple to use. This public service will feed a user- and system administrator-focused web application. This application must be very dynamic, educational, and user-friendly.

1.4 Structure of the dissertation

This dissertation's remaining structure consists of six further chapters grouped as follows:

- **Chapter 2: Fundamental Concepts-** Contains in-depth discussion of the most important ideas and subjects linked with Intelligent Transport Systems. In addition to presenting the related works within the scope of this dissertation's topic, the chapter concludes with a comparison of each connected work.
- **Chapter 3: Requirements and Solution Architecture-** Demonstrates the system's intended use case using a User-Centered Design methodology. Following the identification of the target users and the construction of Personas, Usage Scenarios enabled the generation of a set of requirements.
- **Chapter 4: Tools and Technologies-** This section describes the tools used in the development of the system and explains their selection.

- **Chapter 5: Proof-of-concept System -** Presents the proof-of-concept system and gives a in-dept explanation of how the components were developed.
- **Chapter 6: Tests and Results -** Presents the results obtained by testing several the main components.
- **Chapter 7: Conclusion and Future Work-** Conclusions are provided along with some suggestions for future work that may grow or benefit from this one.

CHAPTER 2

Fundamental Concepts and State of Art

This chapter discusses the underlying ideas of C-ITS and the current state of the art for Cloud C-ITS applications. In the first stage of this chapter, the standard established by the European community, namely ETSI, will be discussed. This standard defines a collection of messages and the conceptual structures that correspond to them. The second section of this chapter explores applications that use C-ITS technology and the ETSI standard with the goal of exhibiting, generally via web apps in pilots tests, how the adoption of these technologies and standards might improve people's lives. The chapter ends with a review of the current state of art.

2.1 ITS

Intelligent Transport Systems (ITS) was early developed by the Japanese in the 1980s and was called later as Japanese Intelligent Vehicle System (Intelligent Vehicle System (IVS)). Concurrently Siemens was also developing a pilot work on route guidance systems in Berlin and, in this case, the Europeans named it road transport informatics.

Despite the different names that both gave, the purpose was common: offer transport solutions by making use of information and telecommunications technologies [4]. While ITS aims at the integration of humans, vehicles and roads, C-ITS allows vehicles, roadside infrastructure, and traffic control centers to communicate directly [5]. These two, not only enable road safety performance and comfort, but also, tackle environmental issues that rise from traffic congestion through awareness of smoother traffic that could relieve traffic congestion.

C-ITS or V2X most important features are Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communications. The Dedicated Short Range Communication (Dedicated Short Range Communication (DSRC)) protocol is used to accomplish this [6]. Several communication formats have been proposed throughout the years to ensure information transmission inside the vehicular network such as: 3rd Generation Partnership Project (3GPP), Bluetooth Low Energy (BLE) and European Telecommunications Standards Institute (IEEE) 802.11p Wireless standard. However other options are being explored due to the Fifth Generation (Fifth Generation (5G)) appearance like 5G Cellular Vehicle-to-Everything (CV2X) and 5G Long Term Evolution (LTE) [7].

Following what has been explained above, a collection of technologies is necessary to enable V2X communications such as: V2I, V2V, and Vehicle-to-Pedestrian (V2P). These technologies are RSUs and OBUs. Detailed explanation of each technology is described in the sub subsection above.

2.1.1 RSU

Road Side Unit (V2I) are wireless access point which are scattered along the road. The communication between vehicles and RSUs takes place based on DSRC protocol. In this context, all RSUs communicate with each other via a cloud-based MQTT Broker, which links them to the central ITS-S Platform, in chapter 3 it's possible to see the overall concept. Figure 2.1 bellow shows an RSU developed in IT(Institute of Telecommunications of Aveiro).

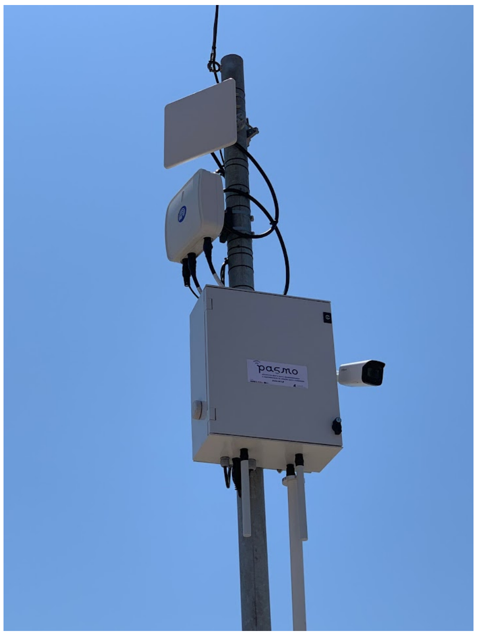


Figure 2.1: RSU with a camera installed and its components

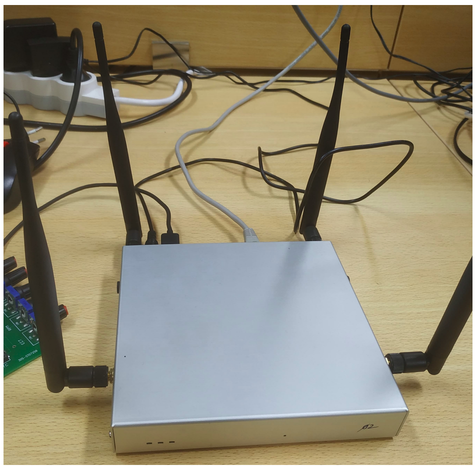


Figure 2.2: Example of an OBU developed by the IT group

2.1.2 OBU

On-board Units (OBUs) are used for V2V and V2I communications and are fitted inside the vehicle. In this context, a mobile application built in IT is connected to the OBU through a wireless access point and connects with a self-hosted MQTT broker on the OBU's premises. The mobile application acts as a graphical user interface (GUI), enabling for the production and viewing of events received by the ITS-G5 or the cloud MQTT broker, which the OBU receives and forwards to the local broker for visualization in the mobile application. Figure 2.2 depicts an example of an OBU.

2.2 ETSI ITS-G5

There are three standardization initiatives: ARIB in Japan, WAVE in USA and ITS-G5 in EU. All of them aspire to improve traffic safety and ITS dependability [8]. Nevertheless, we chose the ETSI ITS-G5 protocol stack for vehicular communications as it is a European project.

ITS-Station (ITS-S) implementation is defined by a multi-layered Open Systems Interconnection (OSI) model. In figure 2.3 is possible to see that there are 3 vertical layers plus Application layer and 2 horizontal layers [9].

The horizontal protocols are:

- **ITS Access Technologies**: Is in the lowest layer of the OSI model. For the physical and data communication layers, this layer encompasses multiple channels and related protocols. The access technologies are often used for internal and external communication within an ITS station.
- **Network and transport Layer**: Includes protocols for data transmission between ITS stations and between ITS stations and other network nodes. The routing of data transmission via intermediary nodes, as well as the effective dissemination of data across geographical areas, are all part of the ITS network protocols.

  Transport layer enable end-to-end data transmission as well as other services, such as reliable data transfer, flow control and congestion avoidance. It also allows Internet

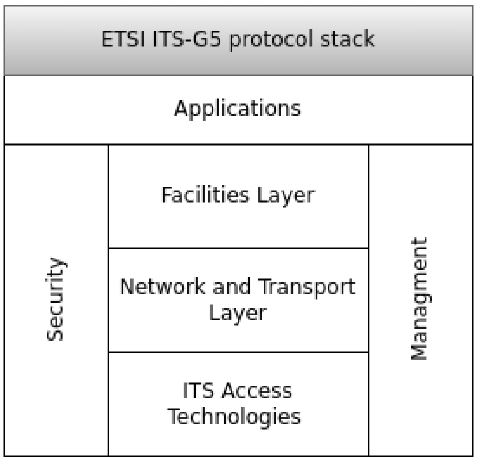


Figure 2.3: ETSI ITS-G5 protocol stack

Protocol Version 6 (IPv6) packet transmission, wo types of messages supported in this layeras well as interoperability issues of IPv6 and Internet Protocol Version 4 (IPv4).

- **Facilities Layer**: Offers a set of functions to aid in the development of ITS applications. In terms of communication, ITS facilities facilitate the construction and maintenance of communication sessions by enabling various forms of addressing to applications. The administration of services, which includes the discovery and download of services as software modules, as well as their management in the ITS station, is a significant facility.

  There are two types of messages supported in this layer - CAM and DENM messages. CAM are through Cooperative Awareness (Cooperative Awareness (CA)) basic service whereas DENM through Descentralized Environmental Notification (Decentralized Environmental Notification (DEN)) basic service.
- **Applications**: Is the upper layer and offers ITS applications relying on data provided by the underlying layer, e.g facilities layer. In the context of this project it is crucial as it provides a service that ensures the interconnection with the C-ITS CLOUD platform through a MQTT broker.

The vertical protocols are:

- **Security**: delivers security and privacy services, such as secure messaging at various layers of the communication stack, identity and security credential management, and features for secure platforms like: firewalls, security gateways, tamper-proof hardware, among other things.
- **Management**: service in charge of the setup of an ITS station, as well as cross-layer information exchange between the various levels and other activities.

2.2.1 CAM

CAMs are messages sent between ITS-Ss in the ITS network to generate and maintain awareness of one another and to facilitate cooperative vehicle performance upon its road network. The status and properties of the initial ITS-S are stored in a CAM. This data can be very useful to keep awareness of the environment around each ITS-S, hence status information for vehicle ITS-Ss contains data about the dimensions, vehicle type, and role in road traffic, among other things.

A common example that suits well the information status is, for instance, a receiving ITS-S can evaluate the collision risk with the originating ITS-S by comparing its own condition to the originating ITS-status, and if required, notify the vehicle's driver. [10]

As shown bellow in figure 2.4 which is compliant with the ETSI standard [11], it is possible to see the structure of a CAM message. It contains: [10]

- **ITS PDU header**: The ITS PDU header is a standard header that specifies protocol version, message type, and the ITS-S ID of the originating ITS-S.
- **Basic Container**: The basic container contains the following information about the originating ITS-S: the type of the originating ITS-S and the originating ITS-most recent geographic location as determined by the CA basic service at the CAM.
- **HF Container**: The Vehicle HF container stores all constantly changing (dynamic) vehicle ITS-S status information, such as direction or speed.
- **LF Container**: The Vehicle LF container stores non-changing or static vehicle data, such as the state of the outside lights.
- **Special Vehicle Container**: The special vehicle container provides information relevant to the ITS-S originating vehicle's function.

CAM structure be divided by two main elements: a header and a body. The CA basic service is a facility layer entity that is responsible for the CAM protocol's operation. It performs two functions: it transmits and receives CAMs. The CA basic service distributes the CAM by using the services offered by the protocol entities of the ITS networking and transport layer. Sending CAMs entails both their creation and transmission. The originating ITS-S creates the CAM during the generation process, which is subsequently sent to the ITS networking & transport layer for dissemination. When a CA basic service receives a CAM, it makes its content accessible to the ITS applications and/or other facilities inside the receiving ITS-S, such as a Local Dynamic Map (Local Dynamic Map (LDM)) [10].

The CA basic service manages the CAM generation frequency and it specifies the time interval between two consecutive CAM generations. The interval between CAM generation must not be less than $T\_GenCamMin$ = 100 ms, i.e a CAM generating rate of **10 Hz**. The delay between CAM generation must not exceed $T\_GenCamMax$ = 1000 ms, i.e a CAM generating rate of **1 Hz**. Another factor to consider is that the time needed to generate a CAM must be less than **50 ms** . The time needed to generate a CAM is defined as the interval between when the CAM creation is initiated and when the CAM is delivered to the networking transport layer.

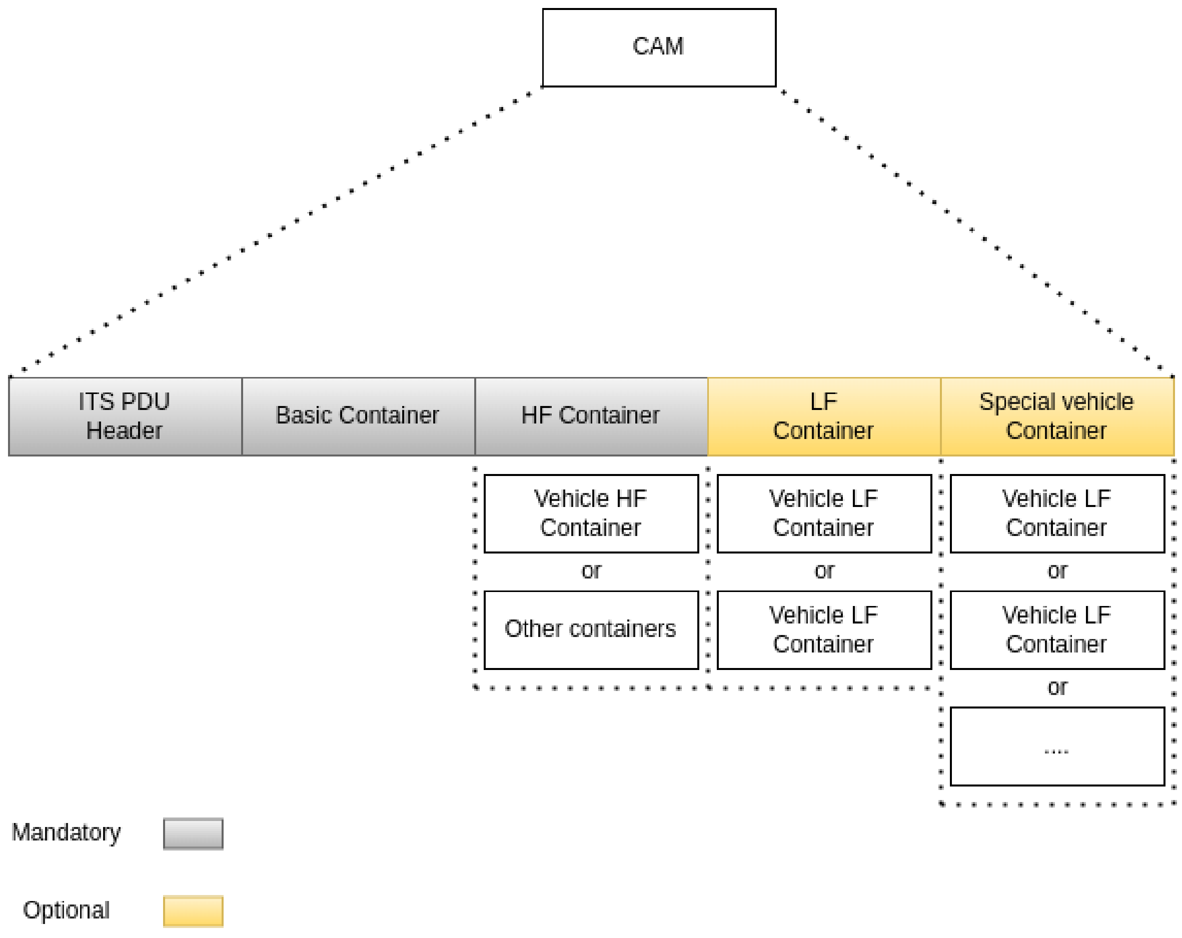


Figure 2.4: General structure of a CAM message

2.2.2 DENM

The facilities layer provides the DEN basic service, which is an application support facility. The Decentralized Environmental Notification Message is created, managed, and processed by it (DEN). A DENM comprises information on a road hazard or abnormal traffic circumstances, such as the type and location of the hazard.

A DENM is often transmitted to ITS-Ss in a geographic region via direct vehicle-to-vehicle or vehicle-to-infrastructure communications for an ITS application. If information about a road danger or traffic situation is determined to be useful to the driver, this ITS application may convey it to the driver. The driver is then in a position to take appropriate action in response to the scenario. [12]

Depending on the context, the application may request several types of DENMs, such as [12]:

- **New DENM**: When an event is detected for the first time by an originating ITS-S, a DENM is created by the DEN basic service. Each new DENM is given a unique identifier called actionID. Event parameters such as event location, event type, event detection time, and other attributes are provided via a new DENM.
- **Update DENM**: A DENM created by the DEN basic service that contains event

update information. The same originating ITS-S that generated the new DENM for the same event sends out an update DENM.

- **Cancellation DENM:**: A DENM that notifies of an event's completion. The identical ITS-S that produced the new DENM for the same event also transmits a cancellation DENM.
- **Negation DENM**: A DENM that advises the originating ITS-S of the end of an event for which a newer DENM has been received from another ITS-S. If the originating ITS-S has the capability to detect the termination of an event DENM has been previously declared by other ITS-Ss, a negation DENM may be used to announce the conclusion of an event.

The DEN basic service offers a DENM to the ITS networking & transport layer at the originating ITS-Sx. The DEN basic service must offer the ITS networking & transport layer with at least the protocol control information (PCI). If the receiving ITS-S is designated as the DENM dissemination destination, the ITS networking & transport layer transmits the received DENM to the DEN basic service. Moreover the general structure of a DENM is shown above in figure 2.5 . It contais a ITS PDU header and DENM payload that is defined as four fixed order parts: the management container, the situation container, the location container and the à la carte container[12].

- **ITS PDU header**: The ITS PDU header is a standard header that contains protocol version, message type, and the ITS-S ID of the originating ITS-S.
- **Management Container**: Includes information on the DENM protocol and its management.
- **Situation Container**: Includes information about the sort of incident that was identified.
- **Location Container**: Provides information on the event's location as well as a reference to that place.
- **À la carte Container**: Includes information unique to the use case that necessitates the transfer of extra data not found in the previous three containers.

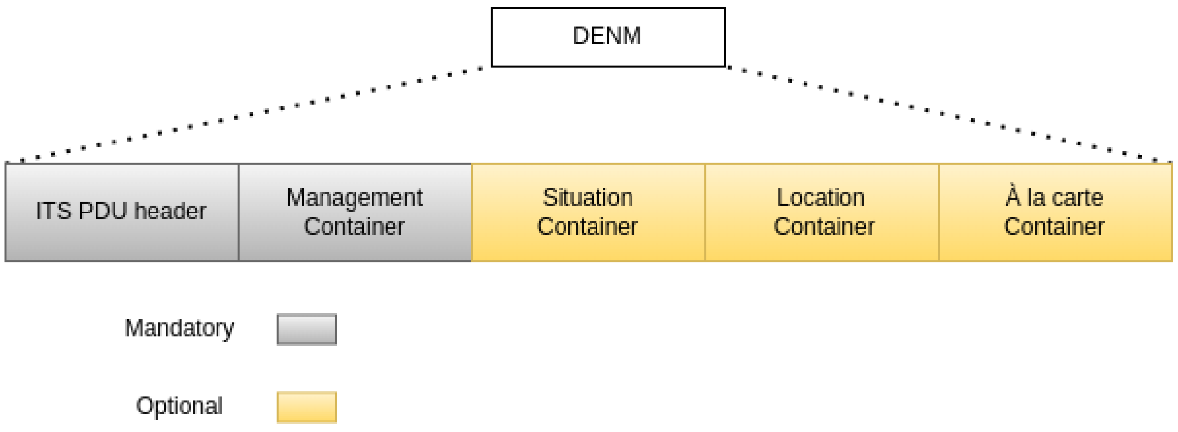


Figure 2.5: General structure of a DENM message

Table A.1 contains all the causes and sub-causes codes for the ETSI Use Cases. Further in Chapter 5, these use cases are shown while creating a DENM.

2.2.3 CPM

The Collective Perception Service is designed to allow ITS-Ss to exchange information about other road users and barriers identified by local perception sensors such as radars, cameras, and other similar devices. In this way, it intends to raise awareness among ITS-Ss by allowing them to contribute information about their seen items to the ITS-particular knowledge base.

The service specifies the Collective Perception Message (CPM), which enables the disseminating ITS-S to share information about identified objects. The communication contains details about the propagating ITS-S, its sensory capabilities, and the items it has detected. The message offers generic data components to characterize discovered objects in the propagating ITS-reference frame for this purpose [13].

This service has been standardized around the world, namely in Europe with the Cooperative Awareness Message (CAM) and in the United States with the Basic Safety Message (BSM). All V2X-enabled cars broadcast these signals at a variable frequency between 1 and 10 Hz, depending on the dynamic condition of the vehicle (current direction, location, and speed) and the assessed channel load [14].

Another important consideration is the CPM transmission rate. Because the frequency band dedicated to V2X communication is restricted, it is critical to appropriately adjust the generation rate and included objects in the CPMs in order to achieve a good trade-off between improving the environmental model and adding channel load, preventing channel congestion and lowering V2X communication performance [15]. Figure 2.6 shows the general structure of a CPM and it is described as follows [13]:

- **ITS PDU header**: The ITS PDU header is a standard header that specifies protocol version, message type, and the ITS-S ID of the originating ITS.
- **Management Container**: Contains information on the ITS-S Type and Reference Position. A moving ITS-S, including a vehicle, or a stationary ITS-S, such as an RSU, may propagate the message.
- **Station Data Container** : Contains the dynamic information of the originating ITS-S in the case of a CPM generated by a vehicle.
- **Sensor Information Container**: Offers the option of additionally providing information regarding an ITS-sensory capabilities. Different container descriptions are available to encode the attributes of a sensor depending on the station type of the originating ITS-S. Information from Sensors Containers are linked to each other at a lesser frequency than other containers.
- **Perceived Object Container**: For each item that an ITS-S has perceived, a Perceived Object Container may be generated. It gives information on the detected object's location in relation to the distributing station. It is also possible to offer classifications

and locations that are matched to road data. Only if objects have been discovered according to the inclusion criteria is this container type introduced.

- **Free Space Addendum Container**: Are provided in a description of computed free space to explain variations.

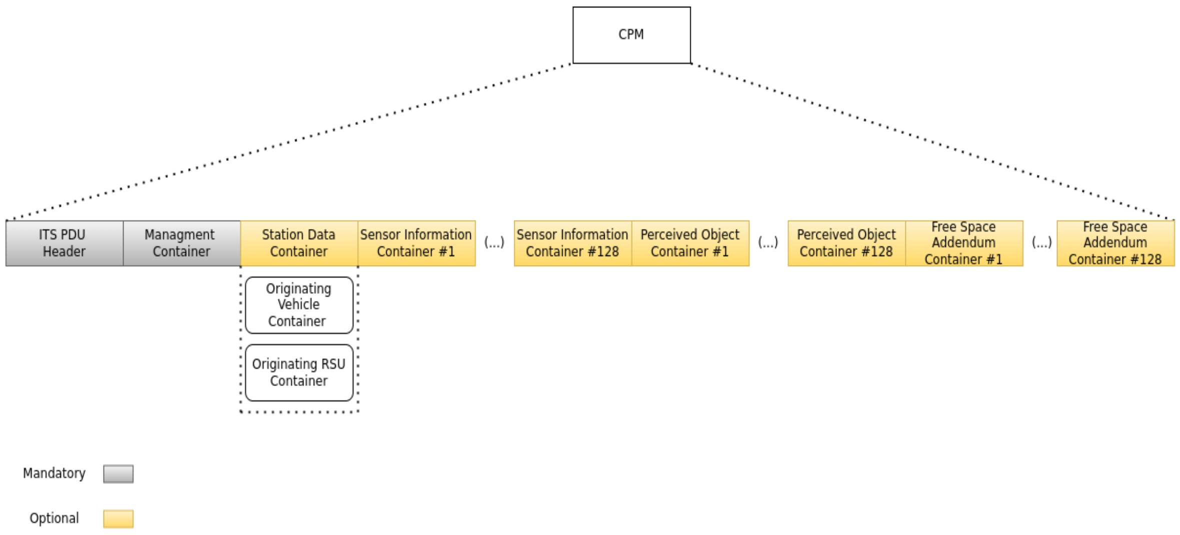

CPM
ITS PDU Header
Managment Container
Station Data Container
Sensor Information Container #1
(...)
Sensor Information Container #128
Perceived Object Container #1
(...)
Perceived Object Container #128
Free Space Addendum Container #1
(...)
Free Space Addendum Container #128
Originating Vehicle Container
Originating RSU Container
Mandatory
Optional

Figure 2.6: General structure of a CPM message

2.2.4 VAM

A necessary component of the road ITS refers to a group of road users who are particularly susceptible, including pedestrians, bicyclists and motorcyclists. To effectively engage in road safety-related ITS communication, these vulnerable road users send a constant repeating awareness message through the Vulnerable Road Users Awareness Message (VAM). VAMs are messages delivered by the Vulnerable Road Users (VRU) ITS-S to raise and sustain awareness of vulnerable road users who utilize the VRU system. A VAM contains status and attribute information of the originating VRU ITS-S. The content may vary depending on the profile of the VRU ITS-S.

The VRU basic service (VBS) is a facility layer object that is responsible for the VAM protocol's operation. It performs three primary functions: it manages the VRU role, sends and receives VAMs.

According to the first function, the VBS gets unsolicited signals from the VRU profile management entity about whether or not the device user is in a setting that qualifies as a VRU (e.g. pedestrian crossing a road) (e.g. passenger in a bus). VAM delivery consists of two activities: VAM generation and VAM transmission. When a VAM is generated, it is composed by the originating ITS-S and then provided to the ITS networking and transport layer for dissemination. The VAM is conveyed using one or more transport and networking protocols across one or more communications channels. When the VRU basic service receives a VAM, it makes the content of the VAM accessible to the ITS applications and/or other facilities inside the receiving ITS-S, such as a Local Dynamic Map (LDM) [16].

Figure 2.7 shows the general structure of a VAM and it is described as follows [16]:

- **ITS PDU header**: The ITS PDU header is a standard header that specifies protocol version, message type, and the ITS-S ID of the originating ITS-S.
- **Generation Delta Time**: Time corresponding to the time of the reference position in the VAM, considered as time of the VAM generation, modulo 2^{16}, (i.e. 65 536).
- **Basic container**: The basic container includes the following elements about the originating ITS-S: The ITS-S type from which it originated. To allow for the potential of transmitting the VAM through non-VRU ITS-S in the future, both data components are maintained separate. The most recent geographic location of the originating ITS-S as determined by the VBS during the ITS-S generation contains a positionConfidenceEllipse that indicates the measured position's accuracy at a 95% confidence level.
- **High Frequency Container**:All VAMs created by a VRU ITS-S should include at least one container with the VRU high frequency (VRU HF). The VRU HF container stores potentially volatile VRU ITS-S status information, such as direction or speed.
- **Low Frequency Container**: The VRU LF container stores data about the vehicle that is static or changes slowly, such as the profile or the state of the outside lighting.
- **Cluster Information Container**: This container should store the VRU cluster's information/parameters. The VRU cluster information container should include information on the cluster's ID, the shape of the cluster's bounding box, the cluster's cardinality size, and the profiles of the cluster's VRUs.
- **Cluster Operation Container**: The VRU cluster operation container must include information on the status and composition of the cluster. A cluster VAM transmitter or a cluster member (leader or ordinary member) may include this container. A cluster leader must include a VRU cluster operation container for disbanding (breaking up) the cluster. A cluster member's individual VAM must have a VRU cluster operation container for performing cluster operations such as joining and departing a VRU cluster.
- **Motion Prediction Container**: The VRU Motion Prediction Container stores information about the VRU's previous and prospective motion states. The VRU Motion Prediction Container must also contain information about the VRU's past locations, predicted future locations, safe distance indication between the VRU and other road users/objects, the possibility of the VRU's trajectory being intercepted by another VRU/object, the VRU's change in acceleration, heading changes, and changes in stability.

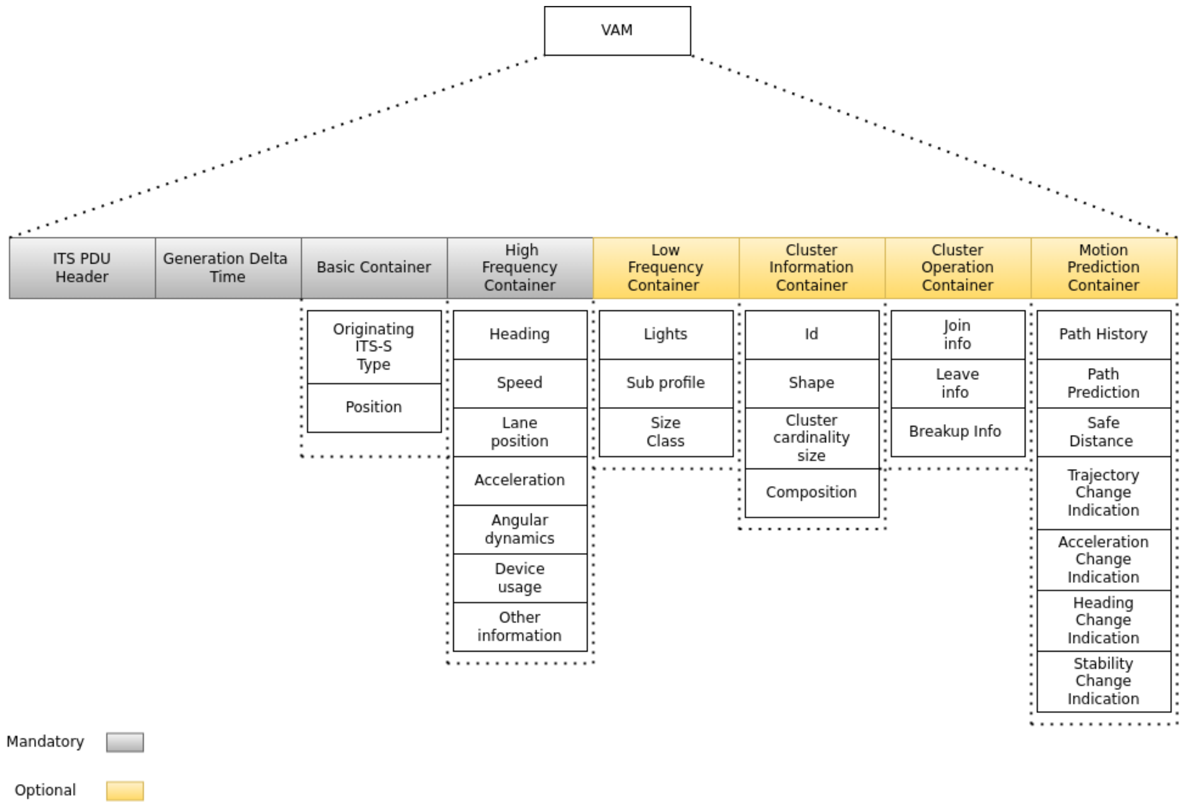


Figure 2.7: General structure of a VAM message

2.3 State of Art

The majority of C-ITS platforms remain private despite the fact that there are a number of platforms for testing C-ITS in existence today. Though some companies and institutions developed tools and technologies that in some way provide a better understanding of this testbedded scenarios. From a study of the relevant state of the art, significant elements have been identified. They are used to compare the systems to one another and to the proof-of-concept system that will be discussed in the remainder of this dissertation.

2.3.1 Synchronized Mobility 2023

IVS-KOM and ITS-S Pilot in Dresden were projects developed within the scope of "*Synchronized Mobility 2023*", led by Fraunhofer IVI. Synchronized Mobility appears with a purpose of using Cooperative Intelligent Transport Systems (C-ITS) thus, focusing on automated and connected driving in urban environments. Under this initiative many projects and research were developed - HarmonizeDD, SYNCAR, IVS-KOM, AULA and SePia. Nowadays more than 50 partners such as automotive companies, scholar researchers and public authorities are cooperating together in different research areas with complementary subjects. These projects received funding, not only by the automotive industry in Saxony, but also from different entities, such as: European Regional Development Fund (ERDF); state of Saxony as well the German Federal Ministry of Transport and Digital Infrastructure (BMVI) [17].

The **IVS-KOM** project is primarily concerned with the communications system. Its ambition is to provide a trustworthy reference platform based on heterogeneous radio technologies that supports a variety of ITS use cases in all traffic and network scenarios. In order to achieve this it is proposed reference components containing [18]:

- C-ITS Facilities
- Maneuvering Message Design
- Hybrid Communications Control
- Multi-standard DAB+ Receiver
- Smart Roadside Unit
- Smart Onboard Unit

As mentioned above the IVS-KOM project focuses in the communication system, whereas C-ITSPilot in Dresden [19] focuses on developing a modular architecture that enables swift execution of service applications via its central processing unit and in addiction to this, a cloud-based backend, a real-time C-ITS service platform coupled with a hybrid communication infrastructure. Fig 2.8 shows a centralized cloud service architecture supported by **Backend Services**.

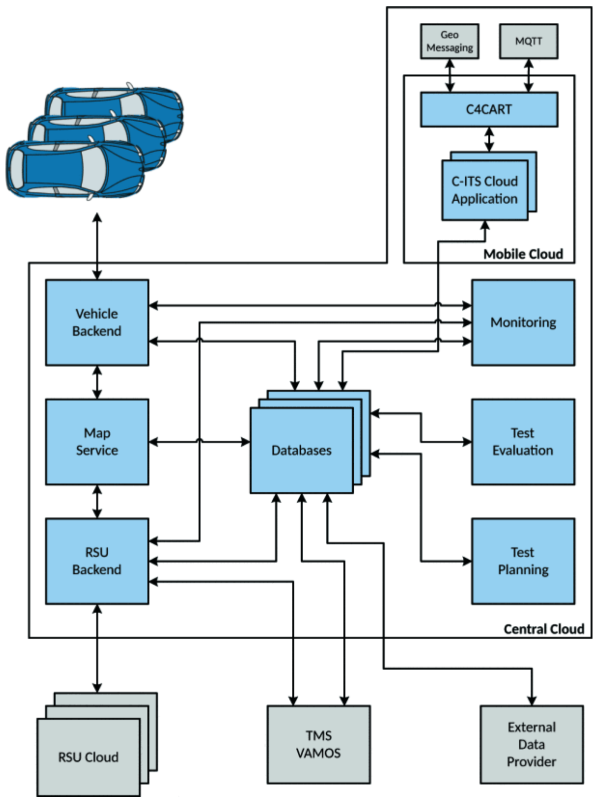


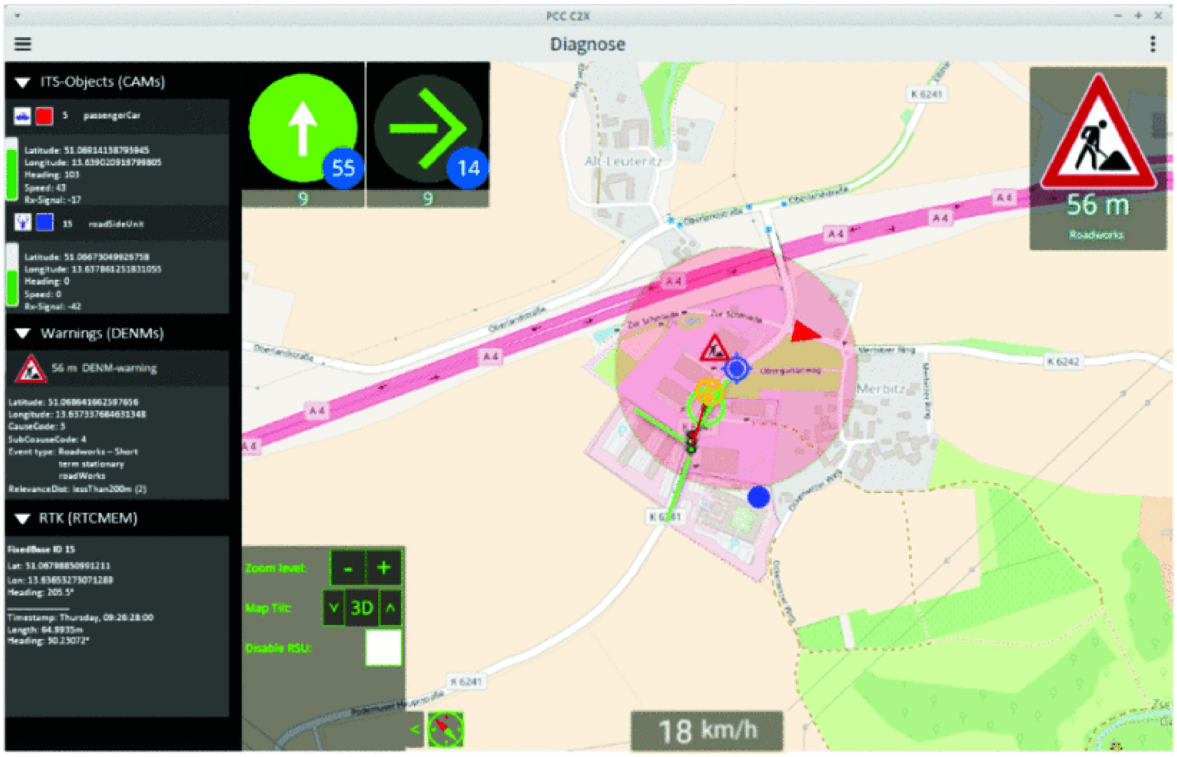
Figure 2.8: Architecture of the centralized cloud services from [19]

The cloud platform for the backend services is hosted inside Fraunhofer IVI's IT infrastructure, and the various services are delivered as scalable microservices. The vehicle and RSU backends integrate testbed data into central databases, enabling real-time monitoring of the Dresden Testbed. The central cloud then distributes semi-static and highly dynamic data to the RSUs in the form of MAP data created from a central database and TMS VAMOS-generated prognoses for the condition of traffic lights, respectively. Additionally, all data given by connected vehicles, RSUs, and external sources is made accessible to the C-ITS services installed in the Mobile Cloud (MC) instances, as well as information contained in different databases. The MC instances make use of the C4CART infrastructure. Simultaneously, communication is established with the linked automobiles using MQTT and/or a GeoMessaging service.

Furthermore, the core cloud's services are containerized with Docker and deployed using the Rancher cluster management tool on the cloud platform. The testbed's entities communicate through MQTT, which is handled by the central cloud via a clustered MQTT broker behind a load balancer. In terms of databases, a mix of PostgreSQL relational databases with the PostGIS extension and NoSQL databases is employed.

Later a testbed was made during the IVS-KOM project. The Dresden testbed consists of three public road corridors totaling approximately 20 kilometers in length: one near the airport, one near the university, and one around the city park. Each corridor is equipped with ten research RSUs built according to the IVS-KOM design and twenty-five productive RSUs from various vendors.

A real-time diagnostic tool has been created to evaluate new V2X services. It enables the representation of unstructured and visualized data, as well as the interpretation and validation of required functionality. The tool communicates with the OBU by Transmission Control Protocol (TCP)/Internet Protocol (IP) and may be controlled wirelessly via an Android mobile. It presents received V2X data in clustered detail and as a map. Fig 2.9 illustrates a scenario that includes both basic warning messages and advanced V2X applications such as GLOSA and HPCP. Each V2X item and event is detailed on the left and shown on the map on the right. On the top of GLOSA, the remaining phase duration and suggested speed are presented. For HPCP, the RTK field is presented on the left, and the corrected GNSS location of the ego-vehicle (green) is shown as a new orange cursor to emphasize the effect of position correction data [18].



Latitude: 51.06814158795945
Longitude: 13.63902091879805
Heading: 103
Speed: 43
Rx-Signal: -17

Latitude: 51.06673049926758
Longitude: 13.63786125183105
Heading: 0
Speed: 0
Rx-Signal: -42

Latitude: 51.06864106258765
Longitude: 13.63733768463134
CauseCode: 3
SubCauseCode: 4
Event type: Roadworks – Short term stationary roadWorks
RelevanceDist: lessThan200m (2)

FixedBase ID 15
Lat: 51.0679885099121
Lon: 13.6365327307128
Heading: 205.5°

Timestamp: Thursday, 09:28:28:00
Length: 64.8935m
Heading: 50.23072°

Figure 2.9: Test framework for the visualization of V2X messages and services from [18]

2.3.2 A Toolkit for Visualizing V2X Messages on the Smart Highway Testbed

The Smart Highway was supported by the European Union's Horizon 2020 project 5G-CARMEN and represents a more than 15 million euro investment. This project began in 2018 and delivers a cooperative, connected, and automated mobility (CCAM) platform that leverages the latest 5G advancements and enables cars to share speed, location, planned trajectories, and maneuvers by exploring distributed and centralized methods to cooperative lane merging [20].

The Smart Highway testbed [21] located in Antwerp, Belgium, makes available V2X systems for experimentation and evaluation, providing flexible support to different V2X access technologies simultaneously. Namely, ITS-5G for short-range communication and Cellular Vehicle to Everything (C-V2X) for short- and long-range communication. The primary contribution of this research is to provide a user-friendly toolbox for monitoring C-ITS Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs). Another application visualizes the current position of C-ITS things, such as automobiles, on a Local Dynamic Map (LDM).

Seven Roadside Units (RSUs) and one BMW X5 model are used. Each RSU is equipped with an NVIDIA Jetson processing station, three communication technologies, and a Global Positioning System (GPS) device. The BMW incorporates the same components into an internal unit called the Onboard Unit (OBU). Moreover, on the Smart Highway testbed the messages are generated in an RSU or OBU. They are then sent through MQTT via CAMINO to a message decoder installed in a car, which transmits the message data to a web application. CAMINO is a framework for managing vehicular communication that enables the hybrid use of the many technologies specified for V2X in Europe and the services that operate on top of them. DUST is an open-source framework that allows transport-independent apps to interact using a publisher/subscriber architecture. It allows CAMINO to link disparate applications over diverse networks ranging from the cloud to edge devices.

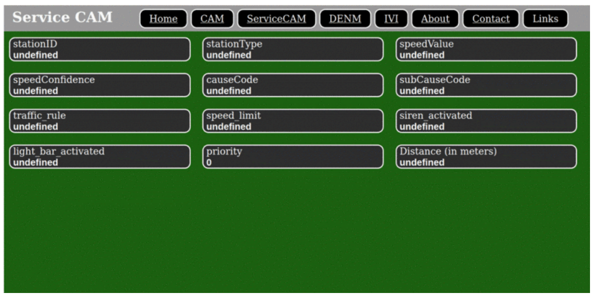
The CAM server offers a dashboard for CAM information. It uses the Spring Boot framework to execute the web application, which monitors CAM messages from nearby C-ITS roadside stations. It is divided into three distinct applications:

- **CAM generator**: Custom messages are created and sent through the CAM generator. The custom CAM messages are encoded in ASN.1 and transmitted to the CAMINO framework through MQTT over a DUST channel. CAMINO transmits CAM data to a car equipped with the CAM decoder and Web server apps.
- **CAM decoder**: The CAM decoder accepts bespoke CAM messages provided to CAMINO through a DUST channel by the CAM generator and records each decoded CAM message in a JavaScript Object Notation (JSON) file. Additionally, the decoded message is transmitted to the CAM web server through a DUST channel using ZMQ after being filtered according to predefined parameters.
- **CAM Web server**: The server runs the web application using the Spring Boot project, and it monitors all CAM received via the CAM decoder. Figure 2.10 depicts the dashboard's background color.These one may be changed from green to different increasing colors until it reaches red, according to a specified priority depending on the distance between the sender and recipient. The color corresponds to the severity of the CAM received. Additionally, based on the CAM data, it shows indicators on the dashboard to indicate the speed limit, an oncoming emergency vehicle, or a forced lane merge to the left or right.

The DENM visualizer acts as a dashboard for DENM information. Additionally, it analyzes DENM communications from nearby ITS-S stations on the road. It is divided into three distinct applications:

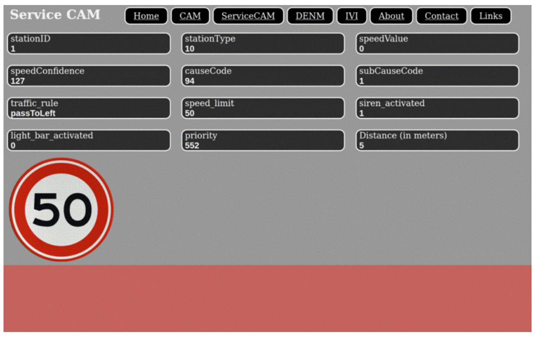
- **DENM Creator**: It encodes DENM messages in ASN.1 and transmits them over MQTT over a DUST channel to the CAMINO framework. CAMINO transmits DENM data to a car equipped with the DENM decoder and visualizer.
- **DENM Decoder**: The DENM decoder receives the bespoke DENM messages delivered to the CAMINO through DUST and records each decoded DENM message in a JSON file.
- **DENM Visualizer**: The DENM visualizer operates concurrently with the DENM decoder in the same vehicle. The visualizer runs the web application using the Spring Boot project, and it also monitors DENM signals arriving from the DENM decoder.

The LDM Display makes advantage of the Local Dynamic Map (LDM) standardized concept to create a real-time map of the vehicle's position and surrounds. Additionally, the



stationID
undefined
stationType
undefined
speedValue
undefined
speedConfidence
undefined
causeCode
undefined
subCauseCode
undefined
traffic_rule
undefined
speed_limit
undefined
siren_activated
undefined
light_bar_activated
undefined
priority
0
Distance (in meters)
undefined

(a) zero severity = green



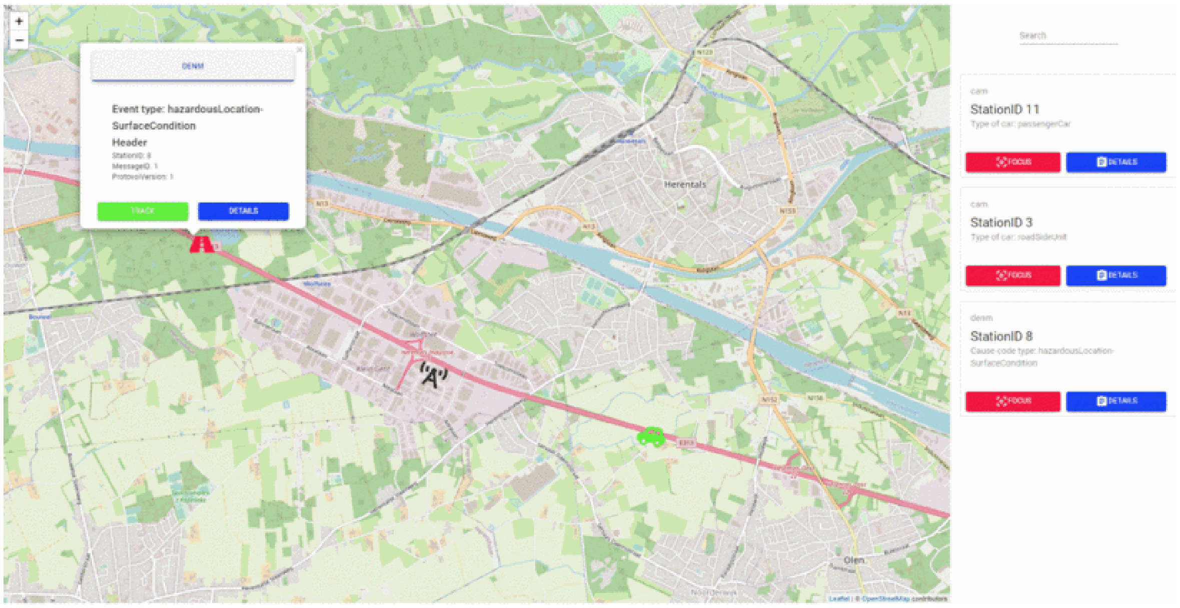
stationID
1
stationType
10
speedValue
0
speedConfidence
127
causeCode
94
subCauseCode
1
traffic_rule
passToLeft
speed_limit
50
siren_activated
1
light_bar_activated
0
priority
552
Distance (in meters)
5

(b) High severity = Current speed limit

Figure 2.10: CAM Dashboards in green and red status from [21]

program plots CAM and DENM messages sent by other cars or RSUs. In Figure 2.11 it is possible to see that at on the right side of the map is a panel displaying the received messages, from which the user can choose to follow the message's source on the map or to open a box with the message's information. LDM is divided into four distinct applications:

- **LDM sender**: Just like with the sender apps above, the LDM sender distributes customized CAM and DENM messages from an RSU. It encodes CAM and DENM messages using the ASN.1 standard and transmits them to the CAMINO framework through MQTT over a DUST channel. CAMINO transmits these CAM and DENM communications to a vehicle that is equipped with the LDM listener, back end, and front end.
- **LDM Listener**:The LDM listener receives and decodes the CAM and DENM messages sent by the DUST sender via a CAMINO DUST channel. Following decoding, the message is recorded in a JSON file and delivered to the LDM back end through DUST.
- **LDM back end**: The LDM back end verifies the message that was received from the LDM listener. After validation, it leverages the Spring Boot project to start the web application that serves the LDM front end.
- **LDM front end**: Is a responsive React application that periodically retrieves data from the back end. It makes use of the Openstreet map framework to show the map. The program displays messages according to the message type and protocol version specified in configuration files. Material User Interface (UI) is used to quickly prototype the application without having to worry about CSS design.


DENM

Event type: hazardousLocation-SurfaceCondition

Header

StationID: 8

MessageID: 1

ProtocolVersion: 1

cam

StationID 11

Type of car: passengerCar

cam

StationID 3

Type of car: roadSideUnit

denm

StationID 8

Cause code type: hazardousLocation-SurfaceCondition

Figure 2.11: Architecture of the centralized cloud services from [21]

2.3.3 Validation Experiences on Autonomous and Connected Driving in AUTOCITS Pilot in Madrid

AUTOCITS was born in 2016 and cost more than two million euros in total. The EU and other partners such as universities and companies such as Indra Sistemas participated in this initiative. This project seeks to accelerate the adoption of C-ITS in Europe by boosting interoperability for autonomous cars and strengthening the role of C-ITS as a catalyst for autonomous driving implementation. Autonomous driving is a critical area of study for Intelligent Transportation Systems (ITS). This kind of driving cannot be imagined without cars that are linked to both the infrastructure and to other vehicles, enabling the deployment of autonomous and connected driving [22].

The project will have a particular emphasis on the connections between connectivity and automation, especially as it relates to infrastructure and road safety. AUTOCITS will conduct a research with a pilot in the Core Network's (Atlantic Corridor) major urban nodes in Spain (Madrid), France (Paris), and Portugal (Lisbon). These pilots will include autonomous vehicles as well as V2X communications and their management inside a unified C-ITS architecture that incorporates traffic control centers (TMC). Madrid's pilot was conducted on a testbed area of the A6 highway, where V2X communications are being used to support C-ITS information. The studies examine the behavior of an autonomous vehicle participating in AUTOCITS as it rotates at high speeds around this testbed region, surveying the V2X coverage and data given by the C-ITS [23].

These three C-ITS services were chosen as sample use cases for extending the functionality of existing traffic control centers leveraging existing infrastructure and accessible services. These services include: Vehicles that are slow or stopped, roadwork warnings, and weather conditions. Furthermore, they are based on V2I communications and enable the demonstration of the advantages of updating the equipment now deployed on Spanish roadways. The services are dependent on a unique ITS architecture that connects traffic control centers (TCCs) to external and internal information providers and incorporates a C-ITS service management module. The service management module will be responsible for collecting the necessary information for service deployment and communicating it to the Road-Side Units (RSUs), which will relay it to the vehicles via their On-Board Units (OBU). The design and deployment of the AUTOCITS project's architecture are shown in figure 2.12. V2I communications are enabled via the use of ITS G5 (IEEE 802.11p) technology, geo-broadcasting, and Decentralized Environmental Notification Messages (DENM), which provide the essential information to activate the C-ITS services aboard. When this data is received in the car, it is used in two ways: to alert the human driver of the message through a human-machine interface, and to supply navigation data to the autonomous driving system's decision-making module.

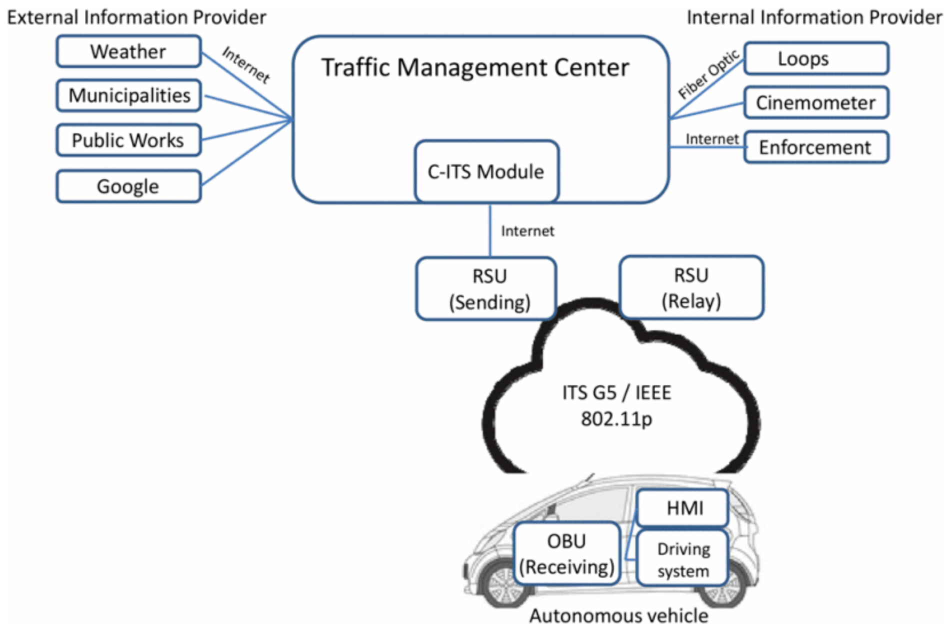


Figure 2.12: Architecture of AUTOCITS from[23]

The pilot's deployment scenario is the High Occupancy/BUS Lane that connects Madrid's urban node to the A6 highway, specifically the section between km 7 and 17, a section of road that allows for the circulation of buses, motorcycles, and cars. Additionally, 15 RSUs have been installed in this corridor. A Mitsubishi iMIEV with self-driving capability is one of the autonomous cars that will be utilized in the Madrid experiment.

Additionally, the vehicle presents a comprehensive architecture that supports steering, decelerating, and throttle, a path following system that maintains the trajectory defined in a high-definition map, a perception module that receives data from onboard sensors, a V2X communications system, and a decision-making module that adapts the path following to changing environmental conditions. Furthermore, a dual-antenna multi-constellation GPS, a Mobileye vision system for detecting road lanes, speed limit signals, pedestrians, and other cars on the road, and a Velodyne VPL-16 3D Lidar installed on the vehicle's roof to scan the vehicle's surroundings are included.

While monitoring the autonomous vehicle's journey, the TMC's C-ITS module sends the DENM message corresponding to "Public Works Warning". The message emission is set at a frequency of 4 Hz and with a location-based range of 1 kilometer, with the emission geo-networking center located at the position of the RSU 1. The message is conveyed by RSUs within the location - based range and rejected by the rest, in order to minimize flooding issues caused by the RSUs' connection being continuous. Figure 2.13 illustrates the coverage and continuity of data receipt throughout the test, as the autonomous vehicle navigates and receives C-ITS information from the glsglos:v2x infrastructure. The four dots indicate the locations of the AUTOCITS RSUs, while the route segments in the same color as the RSU indicate the territories served by each RSU.

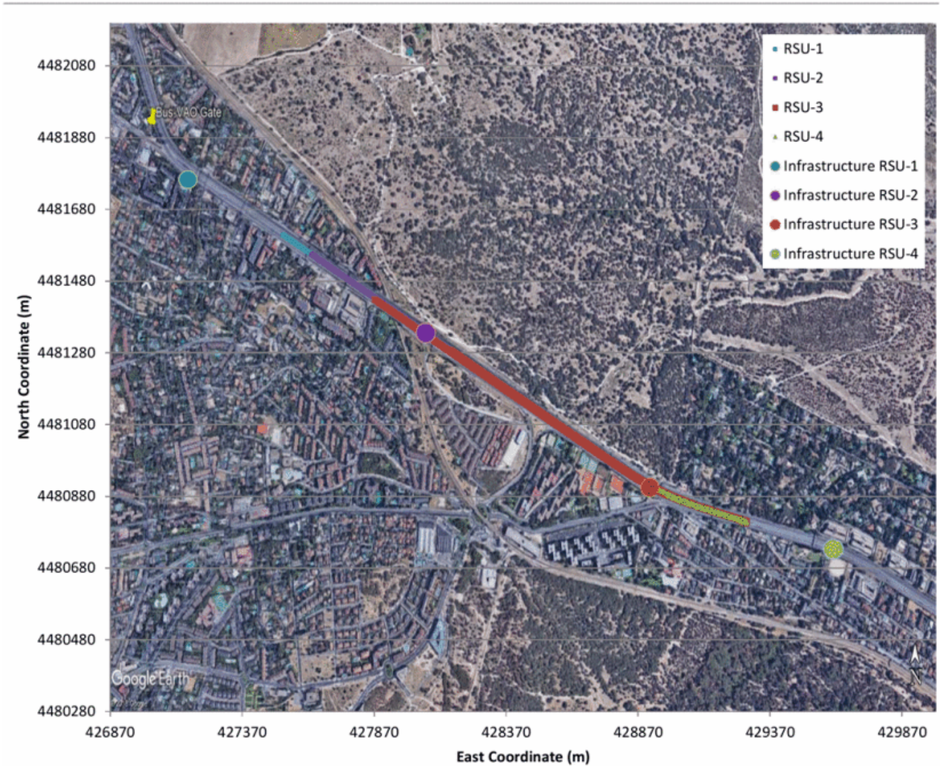


Figure 2.13: V2X coverage during the autonomous test route from[23]

2.3.4 A-to-Be experience: deploying a Portuguese highway C-ITS network under C-ROADS Portugal

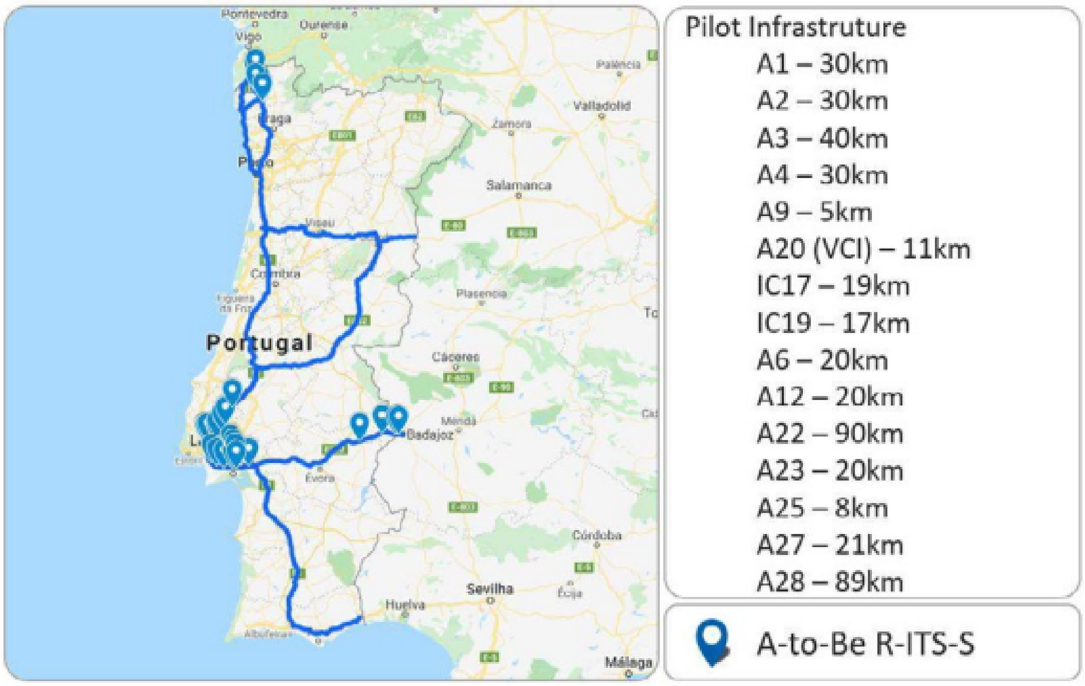
A-to-Be [24], powered by Brisa, has been creating state-of-the art mobility solutions for many years, including an end-to-end C-ITS framework that enables interoperability between various road operators and across borders. C-ROADS Portugal, the Portuguese national pilot that began in November 2017, has a four-year lifespan and is co-financed by the European Connecting Europe Facility Program. Its objective is to design, standardize, and implement large-scale C-ITS systems in Portugal. It works with 31 national partners that share the same goal of improving road safety in Portugal, promoting more sustainable mobility, and reducing greenhouse gas emissions from road transport. A-to-Be (a Brisa Group corporation dedicated to the development of technological solutions) is the project's partner, responsible for defining and coordinating the technical and functional specifications for the systems that comprise the C-ROADS architecture and demonstrate the services on the National highway network.

Five macropilots were launched with the goal of developing, harmonizing, and deploying C-ITS services along the Atlantic Corridor, with approximately 1000 kilometers of infrastructure covering the core and comprehensive network and an estimated 180 roadside ITS stations (RITS-S) and 162 vehicles in operation. The Portuguese network for C-ITS is one of the five macropilots. It proposes to expand the existing C-ITS network by deploying "Day 1" and "Day 1.5" services over a 460-kilometer stretch to cover ITS-G5 in the core (A1, A2, A3, A6, and A12) and comprehensive (A2, A22, A27, and A28) networks, including cross-border sections (A3, A28, and A6) and roads connecting the Lisbon (IC17 and IC19) and Porto (A4 and A20) metropolitan areas. A-to-Be did this by installing 32 R-ITS-S and use ten connected cars throughout a final network of 460 kilometers of roads equipped with 115 R-ITS-S and 64 V-ITS-S.

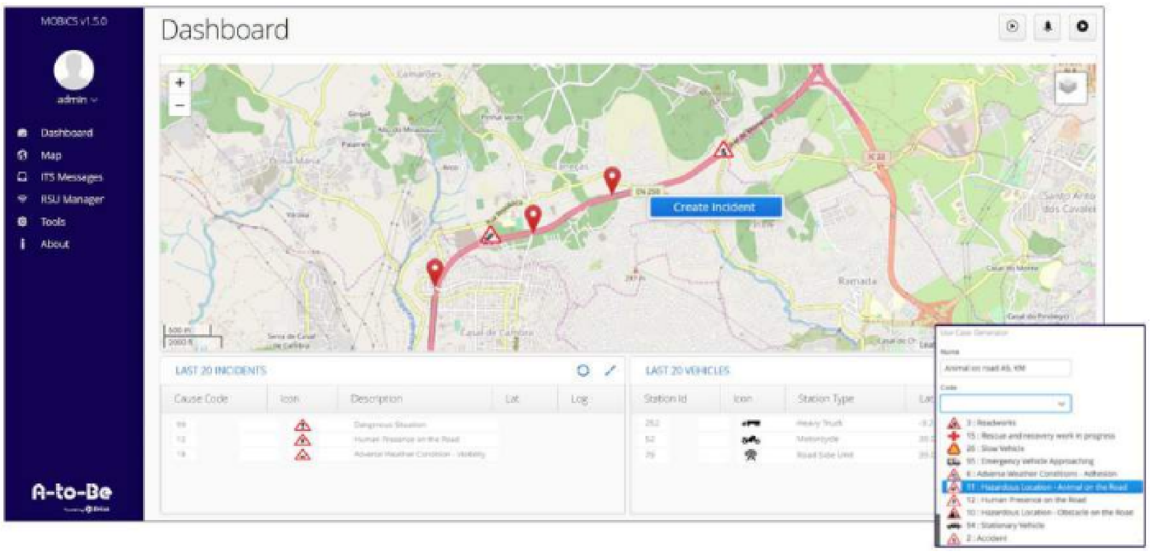
A-to-Be accomplished this by implementing an end-to-end solution that included every component of the C-ITS system, as seen in Figure 2.14a it is made up of A-to-Be V2X stations that may be installed on-board cars (V-ITS-S) or in roadside cabinets (R-ITS-S), and communicate through ITS-G5 radio or 4G cellular. Subsequently, communication between the roadside equipment and the traffic management system (A-to-Be Atlas) is accomplished via the use of an intermediate node known as A-to-Be MOBICS, which serves as the central ITS station (C-ITS-S). Additionally, this C-ITS infrastructure interfaces with national services provided by other project partners, including the National Public Key Infrastructure (NPKI) for security and the National Access Point (NAP) for data exchange.

As a consequence, an efficient C-ITS infrastructure dubbed A-to-Be MOBICS was built for effectively controlling and interacting with heterogeneous hardware and software. To monitor and troubleshoot particular systems, a user-friendly web interface was developed that allows for monitoring the status of R-ITS-S, configuring rules for specific events, creating/sending standard messages by choosing a specific R-ITS-S, and filtering out incoming events or awareness messages. Figure 2.14b shows an example of the Dashboard page.

Furthermore, several tests were made in order to verify the project's behavior. In Lisbon were covered communication's tests between V2I, I2V and V2V in a variety of application situations, including road weather alerts, construction projects, accidents, and infrastructure information. Report stated that throughout the duration of the testing, A-to-Be, CA messages were correctly received and presented. Additionally, they got 42 separate ITS-S DEN and IVI communications from various partners. Only four of the 42 messages received were not displayed.This scenario also confirmed the solution's rapid receipt and presentation of C-ITS messages, ensuring that it is entirely suitable for end users and real-world traffic scenarios. Besides that, other test were made on Portugal's A23 and A25 highways, respectively, in the interior of the country and close the Spanish border. A23 test stated that a significant amount of communications exchanged, including over 200 000 CAM, over 9000 DENM, and over 2000 CAM with toll-protected zones. A25 test also revel a high volume of communications



(a)



(b)

Figure 2.14: (a) A-to-Be's R-ITS-S deployment locations from [24].
(b) A-to-Be MOBICS dashboard view with events and R-ITS-S locations from [24]

exchange thus around 60 000 CAM and over 2000 DENM. Both of these tests allowed testing previously untested features like Protected Zones. Nevertheless, across the whole course of the testing, the CA messages were successfully received and displayed, and the emission power was reduced in the two protected zones specified in the A23. Additionally, A-to-Be accurately received and presented all ten individual C-ITS DEN and IVI messages from the various partners.

2.3.5 Simulation Framework for Edge-to-Cloud Orchestration in C-ITS

Aventi completed a Pilot-T pilot study with funding from the Research Council of Norway, which resulted in a request for a full project. The pilot project studied the obstacles and requirements to completing the major project of designing and testing SAM (Service Announcement Message) for the transportation system. These initiatives received more than 56 million euros in financing. Additionally, Aventi conceived and built a C-ITS platform based on internationally recognized standards and recognized IT solutions, as well as on significant pilot projects conducted in recent years in European C-ITS corridors. The C-ITS platform supports a variety of standard messaging services, including CAM (vehicle position, speed, etc.), DENM (traffic announcements), IVI (virtual road signs), SpAT (traffic lights), MAP (map information), and SAM (Service Announcement Message) [25].

The C-ITS platform in Norway [26] is described in detail, with an emphasis on a simulation framework for testing Edge-to-Cloud orchestration in the C-ITS platform. Edge-to-Cloud orchestration is critical in the C-ITS platform as it enables bidirectional communication between CAVs, edge-based roadside units (RSUs), and the cloud-based central C-ITS server. Moreover, a simulation framework was developed using sophisticated simulation tools and the C-ITS platform to evaluate scenarios that replicate genuine mobility patterns. So far they expanded the well-known open-source simulators SUMO (Simulation of Urban Mobility), OMNeT++ (an event-based network simulator), and Veins by including their own MQTT clients into the simulation code. These MQTT clients enable cars and RSUs in the simulation environment to communicate with the C-ITS platform through the MQTT protocol, utilizing the C-ITS standard data packets.

Figure 2.15 depicts a high-level picture of the C-ITS platform's architecture. Vehicles are "linked" in C-ITS through the C-ITS messaging services via their on-board units (OBUs) or mobile apps on their cellphones. Vehicles may "speak" to one another, roadside units (RSUs), and cloud-based traffic centers through the C-ITS messaging services. The edge devices (RSUs), OBUs, or smartphones communicate with the MQTT brokers in the MQTT Cluster to exchange data packages, which are then processed through the Kafka Cluster to the Database (DB) Cluster and/or Application services.

Furthermore, one kind of application service is the traffic center. A traffic center would feature a Web Server that would give a real-time map of linked automobiles. The traffic center may use the Web app to produce DENM messages such as a "Road work" notice for a particular place on the map. This DENM message "Road work" is sent to all cars that have subscribed to the C-ITS platform in order to receive such messages.

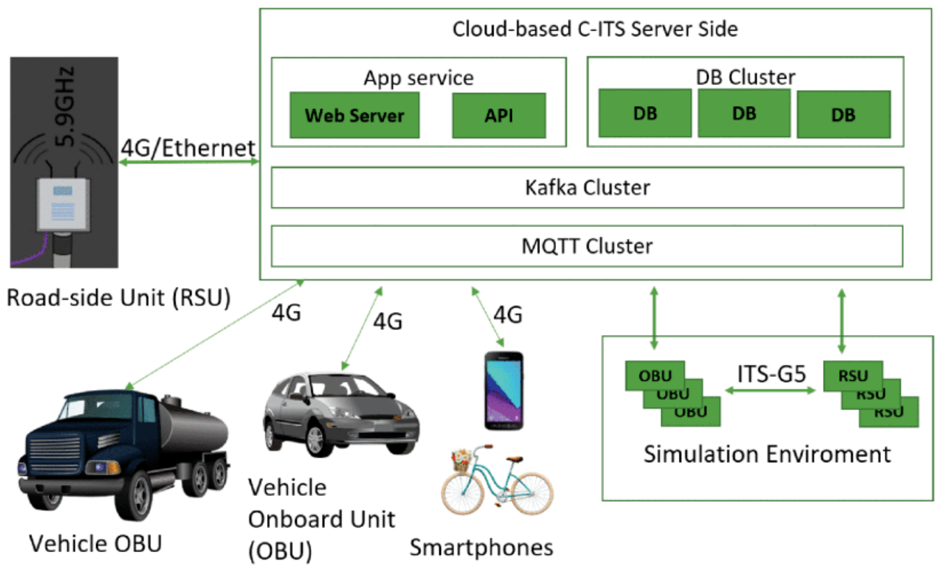


Figure 2.15: ITS-S platform from [26]

The simulation framework is composed of three major components: the construction of the transportation infrastructure (steps 1 and 2), the construction of C-ITS scenarios (steps 3 and 4), and the integration of the simulation environment with the C-ITS platform for testing Edge-Cloud orchestration (steps 5, 6). Figure 2.16 depicts the simulation procedure.

Futhermore, the Veins framework is constructed on top of two simulators: OMNeT++ (for network simulation) and SUMO (for road traffic simulation). Veins establishes the connection between these two simulators through a TCP socket by executing a Python script. This link supports the Traffic Control Interface (TraCI), which enables bidirectional coupling of road and network traffic simulations. When the two simulators run concurrently, vehicle movement in SUMO is represented as node movement in an OMNeT++ simulation. Then, nodes may engage with the ongoing road traffic simulation using the OMNeT++ environment, for example, to model the effect of InterVehicle Communications (IVC) on road traffic. In consequence, the Veins MQTT clients send the relevant live data from the simulation environment to the C-ITS server. More precisely, the simulation environment's live data is packed into C-ITS data packages for transmission to the C-ITS server's MQTT brokers (Edge-to-Cloud). In the other direction, data packages such as DENM, SPAT, or SAM may be delivered from the C-ITS server to the simulation environment's CAVs and RSUs.

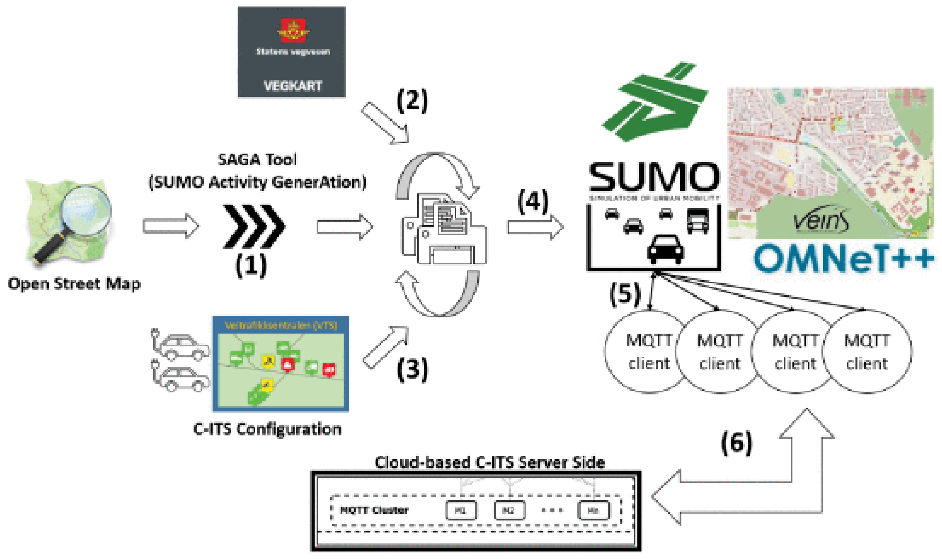


Figure 2.16: Simulation framework from [26]

2.3.6 Analysis of related work

From the study of the related work, certain significant aspects were retrieved and utilized to compare the systems in Tables 2.1 2.2. This does not take into mind the system performance for each feature but merely whether it is present. The final row of each table reflects the constructed proof-of-concept system that will be given later, but already offers a quick overview of where it stands relative to comparable projects.

System	CAM	CPM	DENM	VAM
C-ITS Pilot in Dresden [17]	✓		✓	
Smart Highway testbed [21]	✓		✓	
AUTOCITS Pilot in Madrid [22]			✓	
A-to-Be C-ITS framework [24]	✓		✓	
Aventi Pilot-T [26]		✓		
Developed System	✓	✓	✓	✓

Table 2.1: Related work comparison - Part 1

System	RSU's Map	Traffic metrics	Depict Real time events	radar camera stream
C-ITS Pilot in Dresden [17]	✓		✓	
Smart Highway testbed [21]	✓		✓	
AUTOCITS Pilot in Madrid [22]	✓		✓	
A-to-Be C-ITS framework [24]	✓		✓	
Aventi Pilot-T [26]	✓		✓	
Developed System	✓	✓	✓	✓

Table 2.2: Related work comparison - Part 2

System	Data Proce- -ssing	Data Base(s)	UI
C-ITS Pilot in Dresden [17]	✓	✓	✓
Smart Highway testbed [21]	✓		✓
AUTOCITS Pilot in Madrid [22]	✓		
A-to-Be C-ITS framework [24]	✓	✓	✓
Aventi Pilot-T [26]	✓	✓	
Developed System	✓	✓	✓

Table 2.3: Related work comparison - Part 3

Taking into account all of the investigated projects, we determined that a few essential elements were crucial for the development of a more acceptable and scalable platform. As a result, the C-ITS Pilot in Dresden, the A-to-Be C-ITS framework, and the Smart Highway testbed revealed a great deal more information to collect; consequently, in both of these systems it is possible to observe a back-end and front-end architecture capable of collecting the messages and processing them so that they may be consumed and displayed via an interface.

However, the A-to-be architecture seems to be the ideal solution, since the number of deployed RSUs was far more than those supplied to this project, and they also provide an excellent interface. The other initiatives vary mostly in terms of interface quality; therefore, the Dresden pilot had a complex interface and was difficult to use, while Smart Highway had a simpler back-end but a superior interface.

Moreover, the r AUTOCITS in Madrid revealed some intriguing features about the installations of the RSUs and the testing conducted on the DENM messages, but it lacked a defined architecture that explained how to handle the messages and an interface.

Last but not least, the Aventi project turned out to be less intriguing since it turned out to be a simulation environment and it is much more instructive to learn from real-world circumstances. Furthermore, the design was repetitive and the approach was unanticipated. The advantage was that each communication was likewise processed by a broker.

In general, when comparing the systems in terms of their support for C-ITS communications, all of them support DENM and CAM with the exception of the Aventi pilot. Additionally, the investigated systems share the ability to see the RSUs being tested and the availability of a user interface that allows the user to view them. Our developed system was designed with this in mind and includes support for all C-ITS communications as well as a platform for gaining control of the installed RSUs, the ability to observe traffic patterns, the display of all real-time messages, and the ability to see the cameras contained in the RSUs.

CHAPTER 3

Requirements and Solution Architecture

This section outlines the detailed architecture recommended to satisfy the requirements discussed in chapter 2 as well as the Use Cases necessary to address these demands. Firstly, to guide the present development, a user-centered design approach was used [27]. We begin by identifying target users and the development of personas and use scenarios. This allowed for the construction of a set of requirements. Following that, these requirements were employed to construct a system with the needed capabilities. Lastly, we present the architecture with a brief explanation of how the flow of all components behave so that it is possible to obtain a clear and better idea of how theses pieces of the puzzle, combined together, are important to achieve the desired outcome.

3.1 Personas and Goals

To form personas, it must be presumed that certain features supplied by this project are not available to the general public and are therefore intended for use by administrators at the Institute of Telecommunications (IT). Despite this, ordinary citizens may and should engage with the web application since it may assist them in reporting traffic information. As a result, two distinct personas were developed: “Tiago Cardoso“ and “Isabel dos Santos“. These two personas will not only have distinct demands, but will also aid in the later explanation of the events.

3.1.1 Tiago Cardoso, IT researcher

Figure 3.1: Tiago (from `https://www.pexels.com/`)

Age: 35
Job: IT Researcher
Family: Married and with one child
Location: Porto, Portugal
Education: PhD degree in "Low latency embedded systems"

Bio

Tiago Cardoso (3.1) was born on February 13, 1986, and lives in the city of Porto with his wife, Tania Oliveira, and their child, Martin with only 1 year old. Recently he received an invitation to join the IT team that had a major project regarding VANETs systems and they felt the necessity of adding a new member with some expertise in the area.

Pedro responded immediately to this request - sure, despite the fact that he travels a few kilometers to work, since Aveiro's IT is one of the best in Portugal in terms of network and hardware engineers, he knew he could not lose the opportunity.

Although this new project might sound attractive, Tiago realises that getting into a ongoing project is not easy to keep up at the beginning. Hence a project with so many embedded systems deployed in road sides involves coordination. Also keeping these systems working daily involves a lot research in documentation and, at the same time, understanding what component might not being working properly. Availability is a major on this project.

Goal: Provide Tiago with tools that will help him in the visualization and maintenance of the embedded systems deployed around the roads.

3.1.2 Isabel dos Santos, Businesswoman

Figure 3.2: Isabel (from `https://www.pexels.com/`)

Age: 40
Job: Investidor
Family: Single
Location: Coimbra, Portugal
Education: Degree in “Financial and Management”

Bio

Isabel dos Santos (3.2) was born on April 16, 1983 in Coimbra, Portugal, and has lived there ever since. Despite holding a business degree from the University of Coimbra, she quickly discovered that her love was rental properties. Following some successful investments in Coimbra, she expanded the business and currently owns a large amount of real estate throughout the country’s north and center.

As the growth occurs inside the Portuguese territory, Isabel is constantly going throughout the country to negotiate and handle these similar businesses on a personal level. Isabel, on the other hand, is a very informed individual when it comes to new technologies and a strong supporter for technological investments. If she could view the traffic on the roads she travels on a daily basis, it would be ideal since "time is money" and if she can avoid accidents or forced transits, she can arrive at her destinations quicker.

Goal: Provide Isabel with tools that will assist her in avoiding or anticipating roadblocks or other setbacks that might cost her the chance to close the deal.

3.2 Scenario(s)

Furthermore, several scenarios were built after determining the target user. These explain occurrences that influence the previous mentioned personas and demonstrate how the system assists the user in accomplishing her/his aspirations. These should provide a clearer picture of the system's primary objective and serve as a starting point for extrapolating the ultimate requirements.

3.2.1 Scenario 1

Tiago and the rest of his team have been assigned to verify the status of all RSUs deployed in the roads. Eventually a college alerts him that RSU 5 is not sending messages as it was supposed to. Tiago opens the dashboard and a map appears with **all RSUs deployed by IT** [$\rightarrow$ *REQ*2], later accesses the RSU list on the **RSUs tab** [$\rightarrow$ *REQ*3] and selects RSU 5 that will focus on the map to watch the RSU along with the **color corresponding to the severity** [$\rightarrow$ *REQ*4]. Then a **list of the ten recent messages appears** [$\rightarrow$ *REQ*14] and he decides to see each message to get a closer view.

3.2.2 Scenario 2

The IT team decides to make some tests concerning CAM messages. While two of Tiago's coworkers are in the field producing the tests, he gets to check if the messages are being displayed and therefore, working properly. After some tests, Tiago opens the platform and confirms that cars with **yellow pins** [$\rightarrow$ *REQ*6] are moving correctly around the RSU.

In addition, a second test was planned to confirm that the VAM messages are operational. After some tests Tiago starts to see a **person pin moving around the RSU** [$\rightarrow$ *REQ*7] and confirms that this message is working properly as well.

For the rest of the day additional tests were made and Tiago and his team would like to keep the information hidden from regular users and so Tiago goes to the platform and at the right selects the **check box to not display the RSU** [$\rightarrow$ *REQ*5] to regular users.

3.2.3 Scenario 3

Tiago along with his colleagues are interested in seeing if RSU 7 is displaying the CPM messages in a correct way so that it is possible to see the traffic in real-time. As a result, Tiago opens the platform and selects the RSU 7 on the **list of RSUs** [$\rightarrow$ *REQ*3] and sees **car's icons with blue** [$\rightarrow$ *REQ*11] describing their trajectory on the road. Afterwards, a colleague asks Tiago if the pins are moving properly and they decide to see the camera stream. Tiago and his colleague select the **Camera tab at the right of the screen** [$\rightarrow$ *REQ*12] and start seeing the camera in real-time.

3.2.4 Scenario 4

IT team is invited to make some tests with Brisa company and the key target are the DENM messages. Some members are in the roads preparing for the tests while Tiago is ready to trigger an alert. To make this he opens the platform and selects **Create Alert tab at the**

left [$\rightarrow REQ8$] of the screen. Tiago realises he first needs to **log in to access this feature** [$\rightarrow REQ1$]. Then selects the a DENM, e.g road works, and creates the alert. As a result a pin with the **icon selected is displayed in the map** [$\rightarrow REQ15$] where Tiago puts the pin.

3.2.5 Scenario 5

Isabel has a business to close near praia da Barra. She is on a rush and she would like to know how the traffic behaves at lunch time on Sundays. Isabel opens the platform and navigates through the map and eventually finds the RSU close to the road she will pass. Isabel's first move is to check if an **alert was sent** [$\rightarrow REQ9$] thus if the road is under construction she needs to find other circuit.

Afterwards selects the radar tab at the left and selects the RSU she is investigating. Isabel goes to the right tab and **selects the time interval to see the traffic** [$\rightarrow REQ13$] on last Sunday. After some comparison she understands that is best to travel by morning because at lunch time there is a lot of traffic, i.e many people travel to the beach at the beginning of the afternoon.

3.3 Requirements

The table 3.1 displays the functional requirements that were inferred from the situations given before. The last column indicates the degree of significance, which ranges from 3 (Extremely Important) to 1 (Moderately Important). Taking into account its overall significance and influence on the final system, this decision was made.

Req nº	Requirement	Importance
1	Create an authentication and authorization mechanism to ensure the administrator has features that are not available to public user.	3
2	Displaying all the RSU stations deployed by the IT team so that everyone can witness the location of each device.	3
3	Display the RSU stations deployed as a list, allowing fast search.	2
4	Enable the possibility to see the periodicity of the heartbeat messages sent by each RSU. Color semantics due to their severity.	3
5	The administrator(Admin) may withdrawn an RSU from the map. Therefore enable Admin to set/unset a RSU in the map.	1
6	Displaying the CAM messages sent by the RSUs in order to observe the cars in their proximity (cars pins marked as yellow).	2
7	Displaying the VAM messages detecting any kind of vulnerable road user(VRU) that is crossing any RSU. These messages are tagged with a person icon	2
8	Enable the possibility of the Admin to generate or schedule an alert through the web application	3
9	Display the alerts emitted by the Admin	1
10	Display the CPM messages. These messages are represented by a vehicle circle with a blue color.	3
11	Make possible to the Admin to watch and verify the RSUs cameras by providing a camera stream available to the RSU.	2
12	Provide to the users statistics up to date. Users can view detail metrics regarding the road's traffic.	3
13	Display the ten most recent heartbeat messages sent by each RSU.	3
14	Display the DENM chosen by the user serving as feedback that the alert was emitted successfully.	3

Table 3.1: System Requirements

3.4 Architecture

Using the data gathered above, we were able to develop a clear and objective representation of the system that would need to be given to meet all of these requirements. Having stated that, we developed an architecture to meet all requirements. The architecture can be seen below and it illustrates the outcome of an exhaustive research that, not only covers all UCs, but also, provides a reactive and versatile platform.

3.4.1 System Architecture

Figure 3.3 depicts the conceptual architecture by displaying the components and their interconnections. In the figure it is possible to witness that the system flow is divided in two key parts: **Roadside Infrastructure** and **Cloud Infrastructure**. The first is responsible for the system's communication and is composed by cars communication between each other or with the RSUs and later this data is populated to the central message broker in IT. This step was already implemented so the main focus is on the cloud infrastructure, whose responsibility is to make available the data produced in the infrastructure below. Therefore, the cloud infrastructure is composed by the following components:

- **Front-end**: Responsible for displaying the data collected and perceived by the back-end so that the end users can interact with the system.
- **Back-end**: Key component that can be divided in two main components: **Websocket files** and **Rest Full Application Programming Interface (API)**. The main purpose of this component is to make data processing and, in addiction to this, send the data to the front-end.
- **Message broker**: The centralized place where data is transferred so that the back-end may consume and process it later.
- **Data bases**: This module is composed by two data bases: **MongoDB TimescaleDB** whose goal is to store the data perceived so that it can later be used to display or to make evaluations.

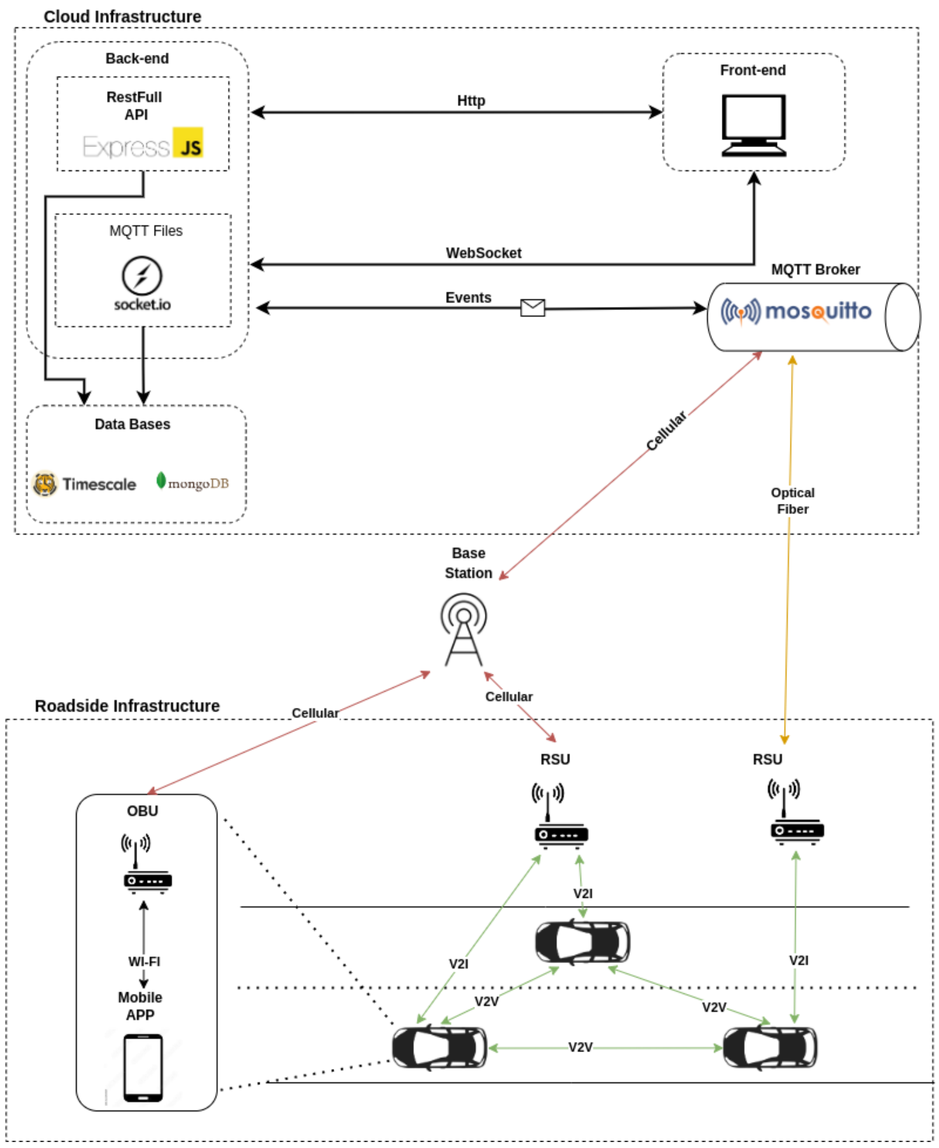


Figure 3.3: System Architecture

Having in consideration what was said above, a detailed architecture is provided for a better perception of how the main components communicate with each other. In short, we present a closer view of how the Cloud Infrastructure was implemented to attain the system needs.

3.4.2 Detailed Architecture

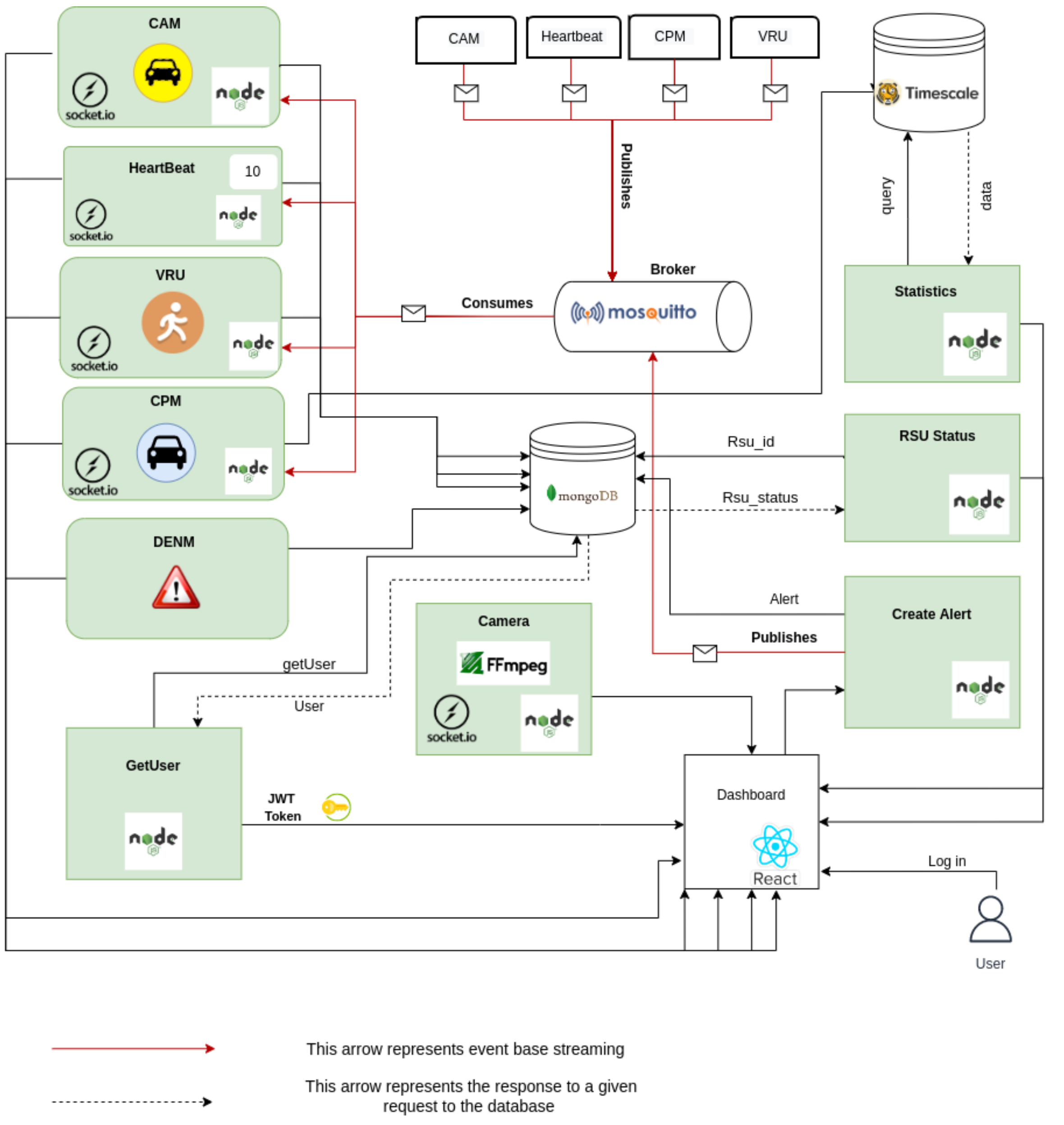


Figure 3.4: Detailed System Architecture

Figure 3.4 illustrates the flow of data collecting from beginning to end. This architecture can be divided in two main parts: the MQTT messages and the API developed. The first is responsible to deliver the messages published to the MQTT broker after consumed by it and the second is responsible to deliver features that afterwards the user might use.

As previously stated, RSUs collect information around them and are responsible for transmitting that information to the central broker or to the base station, which will subsequently transmit it to the broker. Following that, the data is published to the broker. Each message is published to a distinct topic that corresponds to the message's content.

This technique is made possible by the structure shown in Figure 3.5 where:

- **Queue type**: represents the message description (*in-queue* for messages published by the RSUs or *out-queue* for messages published by the platform).
- **Message format**: represents the message encoding type (e.g. JSON or XML).
- **StationId**: represents the ITS-S identifier number.
- **Message type**: represents the ITS messages types, e.g. CAM, DENM , VAM, CPM, Heartbeat.
- **Location quadtree**: represents the current location of the RSU following a quad-tree map system representation[28].

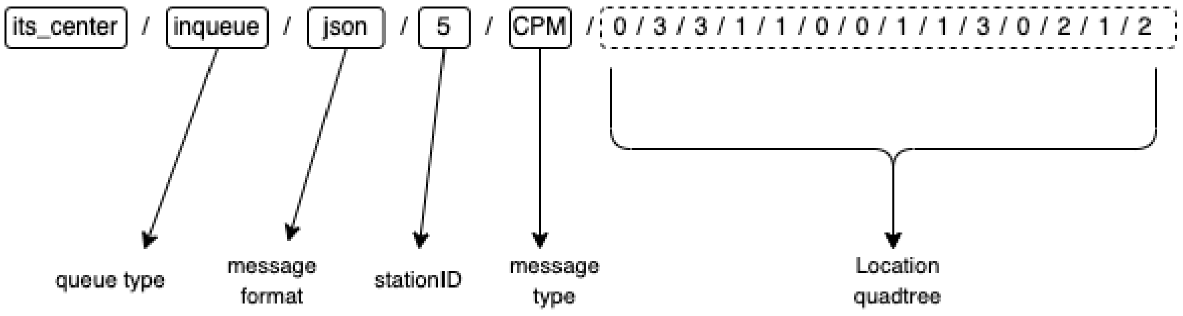

its_center / inqueue / json / 5 / CPM / 0 / 3 / 3 / 1 / 1 / 0 / 0 / 1 / 1 / 3 / 0 / 2 / 1 / 2
queue type
message format
stationID
message type
Location quadtree

Figure 3.5: MQTT topic stucture

Take the example shown in the Figures 3.5, where RSU 5 at position +40.6282327, -8.7336078 (World Geodetic System 84) published a message encoded in JSON and the coordinates are translated to its representative quad-tree, in this case, the system uses a quad tree of zoom level 14(fourteen numeric values generated from zero to 3). Figure 3.6 has a representation of how a quadtree works.

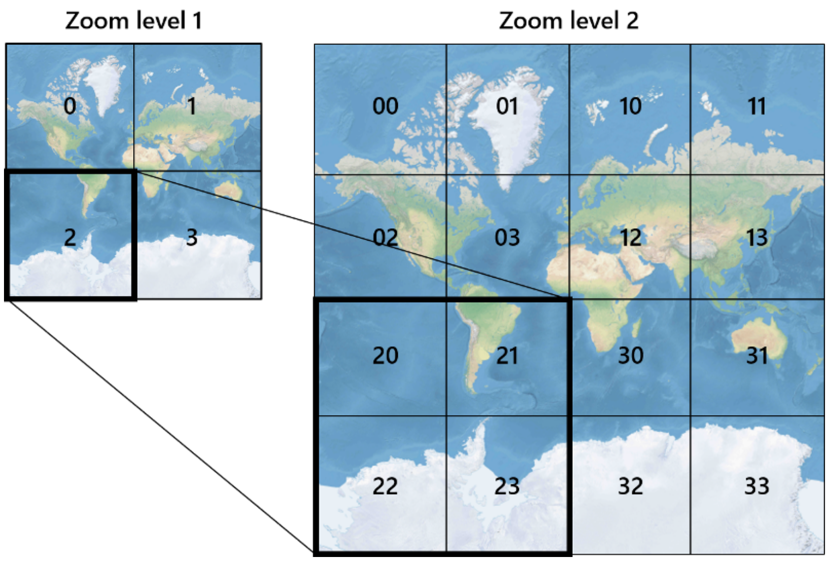


Figure 3.6: Quadtree example from[29].

After taking in consideration what has been said above the back-end needed to support all the MQTT messages, as shown in the architecture, it was designed to have each MQTT message running independently from each other, i.e, each green rectangle represents a Node.js file consuming a specific MQTT message on a specific topic. With this it is possible to apply some logic to the data being streamed and then sent back to the Front-end to be displayed. Each icon displayed within the rectangle represents the icons that would be displayed in the Front-end thus making possible to the user to distinguish how the messages are represented. All the MQTT messages are sent to the Front-end through the usage of the protocol Web-Socket (for detailed explanation see chapter 4) since it guarantees **event-based** communication between a client and a server [30], making possible real-time experience to the user.

In addition to this, the platform not only displays real-time messages but also supports features, namely: access camera in real-time, trigger alert, change the RSU status and real-time statistics. Starting from the very beginning, we needed to take in consideration that, depending on the user, some features are private to regular users and so an authentication mechanism has to be implemented in order to tackle this issue. In the architecture it is possible to see a green square that handles this issue by making use of the **JWT Token** mechanism [31].

Moreover, the user can see through the cameras embedded in the RSUs if the messages being displayed in the platform are corresponding to the expected. Since its about providing streaming/media two major issues needed to be addressed: real-time streaming and user experience. The RTSP (Real-time Streaming Protocol) was chosen to develop this feature as it is a client-server protocol (chapter 4). On top of that, we needed to provide great user experience hence it is expected that the platform does not lag or crash. In order to address this issue, we added a Camera endpoint that each time a user asks for a camera stream, it will ask the back-end if the camera exists and if it does exist, it will add a thread to run the service on a specif port. With this implementation it was possible to address this need and making it possible to use by more than one user without crashing the platform.

Additionally, it was required to create alerts and in order to support this feature we added a endpoint Trigger Alert. When the user wants to trigger an alert he/she needs to navigate to the the Alert tab and choose the corresponding cause code, the interval that the alert will take and the point in the map where he/she wishes to display the alert. Therefore when the user submits the alert a HTTP post request is made to the API that latter will publish the alert to the DENM topic. On the client-side the alert is displayed once the user publishes the alert.

Besides this we added a endpoint that simply says if the corresponding RSU is to be displayed - RSU Status endpoint, thus admin user might want to make some tests in the platform and not being interested in displaying them.

Last but not least, it is provided a Statistics endpoint and this endpoint is the only one that is public to any user as anyone might have interest in watching closely how the traffic flow behaves. To make this possible it was added a new database - timescale db. A highly scalable time-series database is perfect to address this need as the purpose of this feature is to show clearly and objectively how the traffic flow behaves over time.

In summary, this architecture shows that is was able to tackle all the requirements presented above in this chapter. Despite the fact that the back-end supports several services that run independently from each other the truth is that they all run on a node single process and consequently if the server goes down all the services go down as well. Note that this architecture has a pattern of micro-services architectures [32]. I.e the key components: back-end, databases and front-end run on different processes and so, they are independent from each other. We provide a detailed description of the technologies we used in Chapter 4.

CHAPTER 4

Tools and Technologies

This chapter discusses the technologies that were employed to achieve the desired outcome. The list of selected tools is shown in Figure 4.1. The chapter will also describe the methodology underlying their selection.

Figure 4.1: Technologies used throughout the development of the system

4.1 Client-server application development

This section is split into server-side and client-side subsets. The first focuses on describing how the uses cases previously described were implemented and how this information is subsequently sent to the end user. The last part focuses on the presentation of information generated by the server-side to the user.

4.1.1 Server-side

A lot technologies on these days provided libraries and frameworks that can respond to the system's needs, and one stands out for its high scalability and usability on constructing solid and fast back-end - **Node.js**.

Nodejs is an open-source, cross-platform, back-end JavaScript runtime environment that executes JavaScript code outside of a web browser and operates on the V8 engine [33]. Despite the fact it is single threaded, it can handle concurrent client's requests with ease as it uses event loop. It's known for being highly powerful and extremely fast handling client's requests.

Taking in consideration the system's needs, e.g: several data streams, potential use of websockets, creation of an API and media stream, we decided there was no better technology to approach this edge cases than Node.

In addition, Node is highly known for its popularity in providing thousands of libraries/frameworks such as **Express js**, **Socket IO**. The first, allows to develop Restful APIs within the Node ecosystem and is extremely easy to use, while the second enables the use of websockets. With these two frameworks not only it is possible to construct an API but also enable the use of websockets to handle the several data streams provided by the MQTT topics. To summarize, with only one technology/language it is possible to address all the use cases described in the previous chapter and, on top of that, enables the use of micro-services architecture.

Another key aspect while devolping the server side was concerning the authentication process so that an IT admininstrator (admin) could have its own features isolated from regular users, i.e, testbed scenarios or access to a camera within a RSU. As a result, it was decided to use the approach of a **JWT**. JSON web token (JWT) is an open standard [34] that specifies a small and self-contained method for sending JSON objects securely between parties. Due to its tiny size, a JWT may be swiftly conveyed by a URL, a POST parameter, or an HTTP header. In a nutshell, JWT is composed by:

- **Header**: Identifies the technique used to produce the signature. For example, HS256 shows that this token was signed using HMAC-SHA-256.
- **Payload**: A collection of claims. The JWT specification specifies seven Registered Claim Names, which are the standard attributes often found in tokens.
- **Signature**: Validates the token in a secure way. The signature is computed by encoding the header and payload using Base64url Encoding [35] RFC 4648 [36] and concatenating them with a period.

4.1.2 Client-side

An important system requirement was to display all the data provided by the messages, among other features. The technology chosen to develop this was React js. React is a JavaScript framework, open source, that allows to create great modern interfaces. It is managed by Facebook, a consortium of independent developers and businesses. Being solid and well-documented, installation and learning are quick and it has a big community.

Since it is required to display all RSUs as well as being able to track the vehicles in real time, a map was necessary to accomplish this requirements. **React-map-gl** [37] was the framework chosen. To display graphics regarding metrics of the traffic flow it was necessary to have libraries to accomplish this requirement: **recharts** [38] and **react-datepicker** [39]. The first enables access to the manipulation of bar's charts while the second provides the use of a calendar so that the user can be able to watch traffic metrics within a time frame he/she might want to choose.

4.2 Databases

The core system requirements are based in processing streams of data, having an API and provide traffic information within a time frame set by the user.

4.2.1 MongoDB

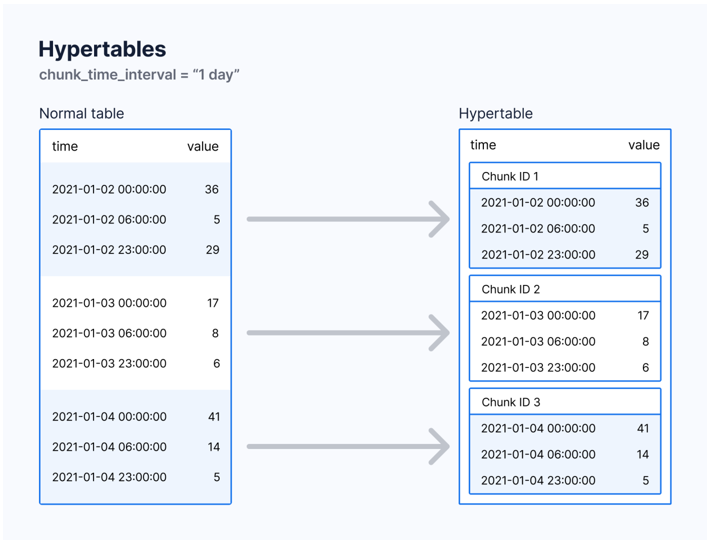
Since we are not dealing with data that needs to be related it became obvious that a NoSQL database was the best approach to deal with the API requests. Among several options the chosen one was MongoDB.

MongoDB [40] is an open source NoSQL database management application that can manage, store, and retrieve document-oriented data. It employs records consisting of documents containing a data structure formed of field and value pairs. The documents resemble JavaScript Object Notation, and groups of documents are referred to as collections.

4.2.2 TimescaleDB

Since the first issue is solved by Mongo the second issue regarded having a database capable of providing traffic metrics within a time frame. Mongo was first thought to handle this issue but sooner became clear that it would require a lot more work and complexity developing the queries. After some research it was decided to use TimescaleDB as it is a timeseries database.

TimescaleDB [41] is a PostgreSQL extension that includes all time-series optimizations and specialized functionality. It is well-known for its Hypertables, which are PostgreSQL tables with specific characteristics that make working with time-series data simple. However, behind the scenes, hypertables automatically divide your data into time-based chunks. Each hypertable (4.2) is composed by chunks, which are child tables. Each chunk is allocated a time interval and only stores data from that interval.



Normal table

time	value
2021-01-02 00:00:00	36
2021-01-02 06:00:00	5
2021-01-02 23:00:00	29
2021-01-03 00:00:00	17
2021-01-03 06:00:00	8
2021-01-03 23:00:00	6
2021-01-04 00:00:00	41
2021-01-04 06:00:00	14
2021-01-04 23:00:00	5

Hypertable

time	value
Chunk ID 1	
2021-01-02 00:00:00	36
2021-01-02 06:00:00	5
2021-01-02 23:00:00	29
Chunk ID 2	
2021-01-03 00:00:00	17
2021-01-03 06:00:00	8
2021-01-03 23:00:00	6
Chunk ID 3	
2021-01-04 00:00:00	41
2021-01-04 06:00:00	14
2021-01-04 23:00:00	5

Figure 4.2: Hypertable from [41]

4.3 Communication between modules

A message broker is software that allows applications, systems, and services to share information and interact with one another. Message broker does this by translating across formal messaging protocols. This enables interdependent services to "speak" directly with one another, despite being written in different languages or deployed on separate platforms.

A broker works as a middleware enabling real-time communication between different applications, services, and systems. As stated before, MQTT was chosen since it was already used by IT group, is known for being lightweight, very easy to use and is fully supported to JS frameworks, also provides a GUI (MQTT Explorer) witch is useful in case a user wants to see the data. Figure 4.3 depicts a typical scenario of MQTT message broker where data is published to a given topic, i.e data stream, to later be consumed by the subscriber.

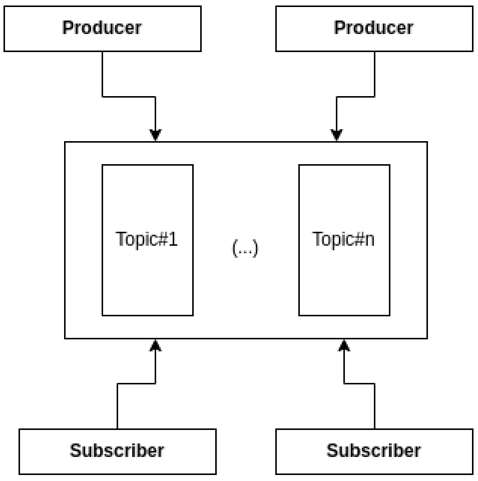


Figure 4.3: Scenario of a message broker

4.3.1 WebSocket

Another key technology responsible for the communication between the back-end and the front-end is WebSocket. As mentioned above it allows bidirectional, real-time communication between a server and a browser via a single TCP connection. With WebSocket, event-driven replies to messages are available without the need to poll the server.

Figure 4.4 depicts the WebSocket connection between a client and a server. To initiate a connection, the client sends a Handshake request to the server, which responds with a handshake response. Once the connection has been established, communication shifts to a binary protocol that is bidirectional. Both the client and server may now transmit and receive data in full-duplex mode. For the connection to be closed, just one party has to shut the channel.

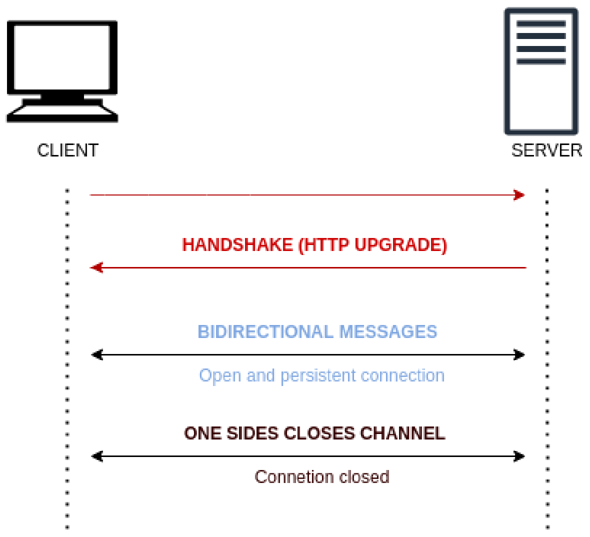


Figure 4.4: A diagram illustrating a WebSocket connection.

4.3.2 RTSP

Final key aspect in the back-end is regarding the access to the cameras embedded on a RSU. As stated before in chapter 3, the system is required to support media stream. Since the cameras support RTSP the choice was to implement this protocol, among other options such as WebRTC [42].

Real-Time Streaming Protocol (RTSP) is a communications protocol that provides the real-time delivery of media streams. It also includes techniques for requesting individual streams and regulating their playback. RTSP leverages TCP to ensure reliable transmission of streaming video material, and the protocol may also be used to initiate, pause, and terminate streaming. It may also be used to adjust the volume and playback speed.

Figure 4.5 depicts a scenario of the RTSP connection. In a nutshell the connection between browser and server is describe as:

- **DESCRIBE**: DESCRIBE consists of a rtsp url (rtsp:/...) and the kind of reply data that may be processed;
- **SETUP**: SETUP request provides the transport method for a single media stream.

- **PLAY**: A PLAY request will trigger the playback of one or all media streams.
- **Media Stream**: Media stream request will initiate the media from the server.
- **PAUSE**: A PAUSE request pauses one or all media streams briefly so they may be restarted with a PLAY request.
- **TEARDOWN**: A TEARDOWN request is used to end a session. It terminates all media streams and releases all session-related server data.

In order to reproduce the media stream it was required to have a software that could translate the data being streamed and the FFmepg software [43] was the most suited to address this need.

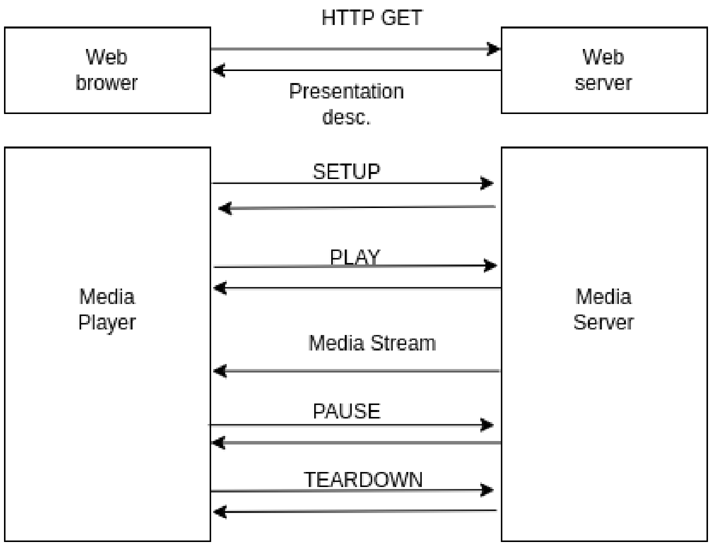
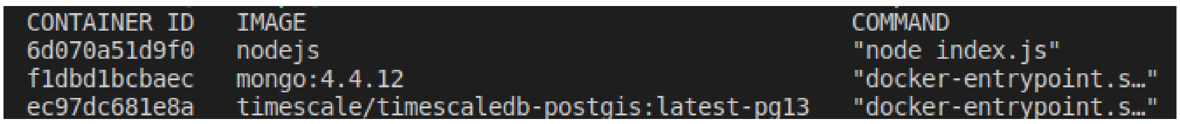


Figure 4.5: RTSP Protocol

4.4 Docker

One of the key aspects of this system is that it has a micro-service pattern [44]. Therefore, the best technology to obtain this behavior is Docker.

Docker is an open source platform for containerization. It helps developers to bundle programs into containers, which are standardized executable components that combine application source code with the operating system (OS) libraries and dependencies necessary to execute the code in any environment. Besides docker, Compose is used also since Compose is a tool for designing and executing Docker applications that cover several containers. Compose allows you to configure your application's services using a YAML file. Then, with a single command, you build and run all configuration-based services.



```
CONTAINER ID   IMAGE                                      COMMAND
6d070a51d9f0   nodejs                                     "node index.js"
f1dbd1bcbaec   mongo:4.4.12                               "docker-entrypoint.s…"
ec97dc681e8a   timescale/timescaledb-postgis:latest-pg13  "docker-entrypoint.s…"
```

Figure 4.6: Docker containers running for system's services

As a result, four services run independently from each other: mongoDB, timescalesDB, nodejs (back-end) and reactjs (front-end). Figure 4.6 depicts the current implementation of the micro-services by having three containers running independently. The front-end was not added to the docker compose due to testing scenarios as it is more productive to not access docker compose in a development mode situation.

CHAPTER 5

Proof-of-concept System

This chapter describes the intended proof-of-concept system based on chapter 3's design. It's a more in-depth look at the implementation of the system with a few diagrams that illustrate the data flows, as well as an explanation of the pieces of code responsible for the execution of the developed components.

5.1 Event based massages

Having in consideration that one of the major requirements was to support all data streams published to the MQTT broker, it was necessary to develop scripts that could listen to the given topic and, regarding the needs, implement the logic to make possible to see these streams in real-time on the front-end. Bellow it will be presented the implementation to achieve the desired outcome previously outlined.

Figure 5.1 depicts the scenario of a given message that publishes to the corresponding topic (identified by the type of message) that will be consumed by each independent JS file whose purpose is to parse the data and that latter will be transmitted via websockets and stored in a database simultaneously.

Having this in mind it seems clear that display data streams in real-time is quite straightforward and requires less work than the expected. Note that while **CAM** and **VAM** messages follow this behavior and therefore, no extra processes are made, other messages don't rely on just replaying the data fetched from the broker. Therefore, the remaining messages (Heartbeat, DENM, and CPM) must handle distinct edge cases.

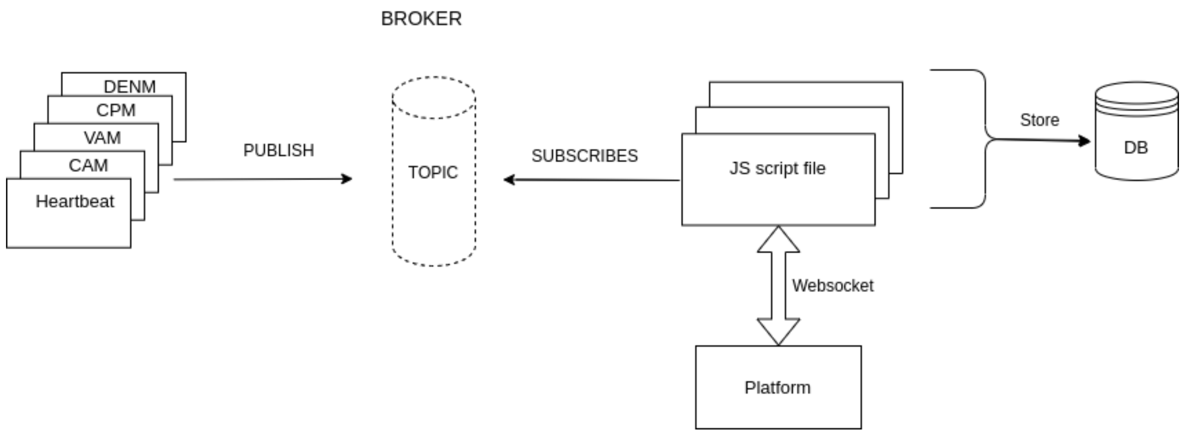


Figure 5.1: Diagram flow of data streams.

5.1.1 Up-to-date Heartbeat messages

Starting from the very beginning, Heartbeat messages have a significant influence on IT administrators, providing real-time input on the condition of a specific RSU. Basically the system keeps listening on the topic Heartbeat and as soon as it gets an heartbeat message it stores on a dictionary, where the key is its own RSU identifier(stationID), and keeps storing the message to that given RSU until it fills the array of ten position. Moreover, it needs to keep in mind that the major goal here is to obtain the status of a given RSU and, after 2h of not sending any message it is considered that it might have occured a problem and, after 12h of not receiving any message it is considered to have a problem, because of this the two dictionaries were created: oranges and reds, respectively (color semantics is related to the severity). Figure 5.2 shows how the flow of a RSU message status is being updated to the front-end.

As so, the messages are populated for one of the three dictionaries, and the way to make sure the time of a given message to receive a message has passed, is by using the "setTimeout()" callback functions. In a nutshell messages are previously stored on the regular dictionary, i.e rsuMap, that is consistently updating the RSU document on the database, and if after some time they are not receiving any more messages it will update the document on the database and set this one to orange or red, according to time that the last message was receive.

To identify the current state of the RSUS the setTimeout() function is used, in order to identify when some time has passed since each RSU received a message. A timeout per RSU is required, and cleared everytime the RSU receives a new message. After the first timeout, that detects if the station hasn't received a message in two hours and sets its status to orange, a new setTimeout is created for ten hours, that sets the status to red. Any message that any RSU receives will clear the timeout and set the status to green. In summary with dictionaries and setTimeouts it was possible to achieve this requirement in a clean way. The UI will be listening to the websocket channel, i.e RSU-update, so that it is possible to have a real time experience of providing feedback about the RSUs messages and status.

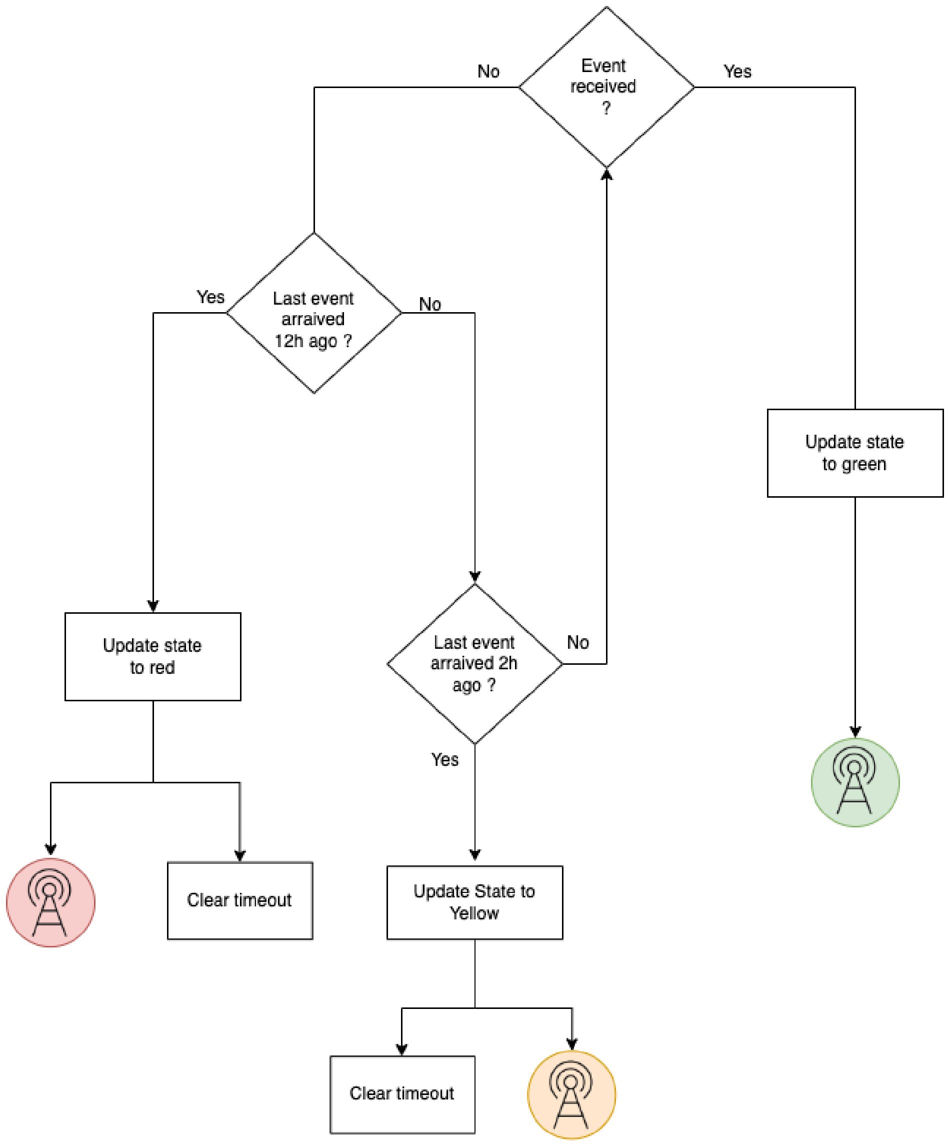


Figure 5.2: Diagram flow of how heartbeat messages are updated to the front-end

5.1.2 CPM format stored

The way CPM messages are stored in the database and displayed in real-time can be divided in two main parts: the message format and the message storage. The first part is responsible for parsing the data from the topic and transform it in values "readable" to the front-end and the database, hence the data fetched from the topic come in cm/s and the next position needed to be calculated every time. The second step is responsible for storing the corrected data into the database so that later these one may be used to calculate traffic metrics.

Timescaledb allows to insert multiple rows and strongly suggests to batch the data thus "the TimescaleDB engine batches the rows by chunk, and writes to each chunk in a single transaction" [45]. At the beginning it was thought that data could be "dump" to the hypertable. Sooner it became clear that this solution was not scalable thus in 12 hours it was obtained millions of records.

Consequently, it was decided to implement a store mechanism much more complex. Figure 5.3 depicts the scenario to approach this need, where we may have 1 or 10 cars perceived by the radar's RSU, each second. The definition of the CPM's script had this in mind and the way it operates is by breaking data perceived by the radar in time frame's chunks of 1 second. In order not to lose the data perceived until one second has reached, a dictionary was created to operate as a record of one second time interval, then it will populate an array of `BATCH_SIZE` length. When the array gets filed it will dump all the N rows into the hypertable. As a result, the database is updated much less frequently reducing it's load and increasing the overall platform scability.

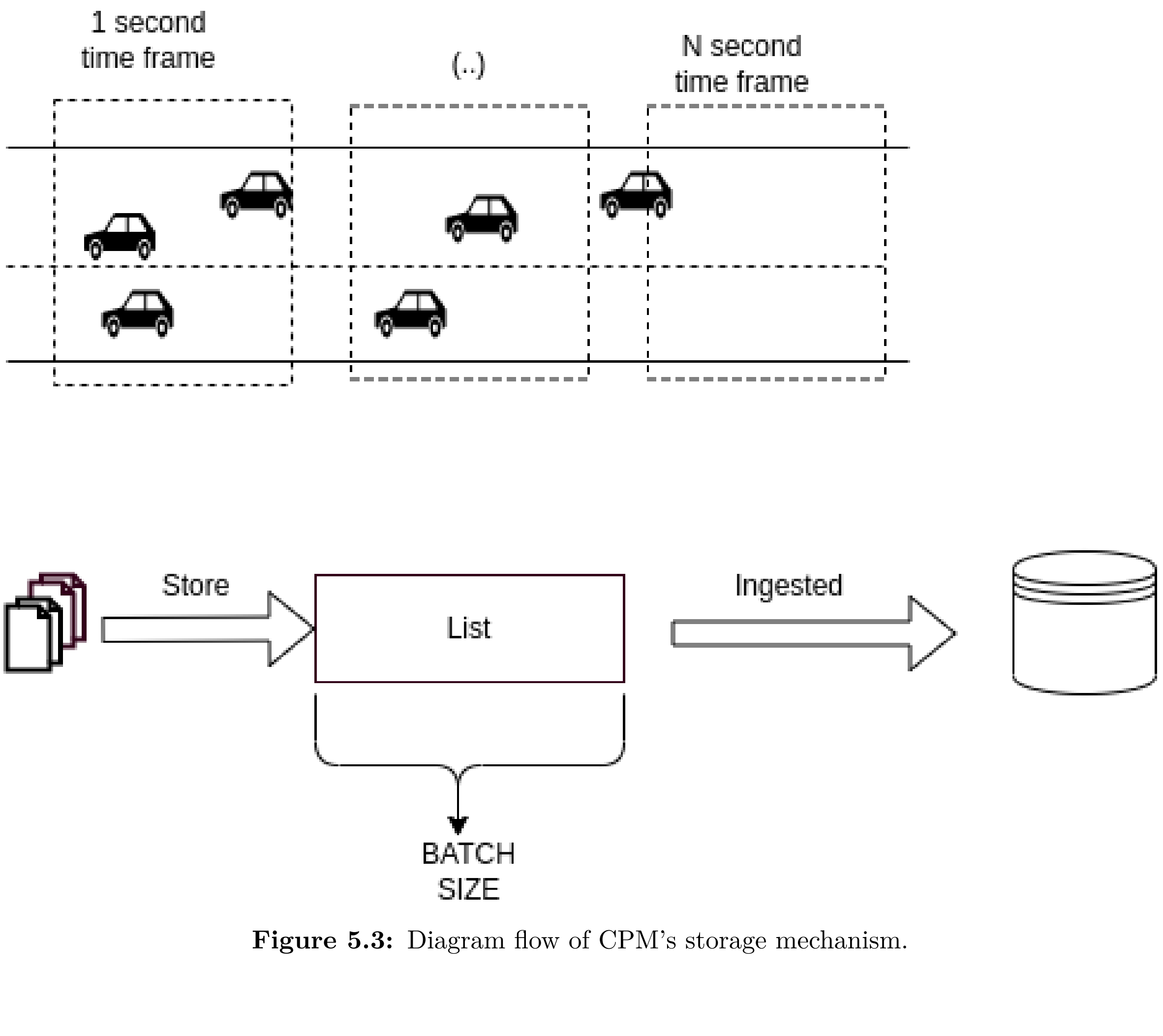


Figure 5.3: Diagram flow of CPM's storage mechanism.

5.2 API

5.2.1 Authentication and Privileges

As stated before, it was important to have an API that could deal with authentication and privileges thus adim features needed to be hidden from regular users. To address the first need, after the user logs in, an Hypertext Transfer Protocol (HTTP) request is made to the API that latter will ask the database if the given credentials exist. In case of success then a JSON Web Token (JWT) token is generated and given to the user, allowing him/her to navigate to any feature in the platform. Note that the JWT token is sent in the response header in order to set the browser cookies. Nowadays it is important to have in consideration security threats. Cross site scripting or XSS attacks [46] are very popular and in this context it can happen when the attacker is able to steal a user's session token. After gaining access to a session token, the attacker may often impersonate a user until the token expires or is revoked. To mitigate this, the HttpOnly flag is used in order to make the cookie inaccessible by JavaScript[47].

To resolve the privileges needed it was decided to add a Boolean attribute and, similarly to the authentication mechanism, in the front-end is verified in the cookies if the user has privileges, i.e the in case isAdmin is true.

Bellow it is possible to see figure 5.6 that depicts a scenario of how a request from the client is process within the API.

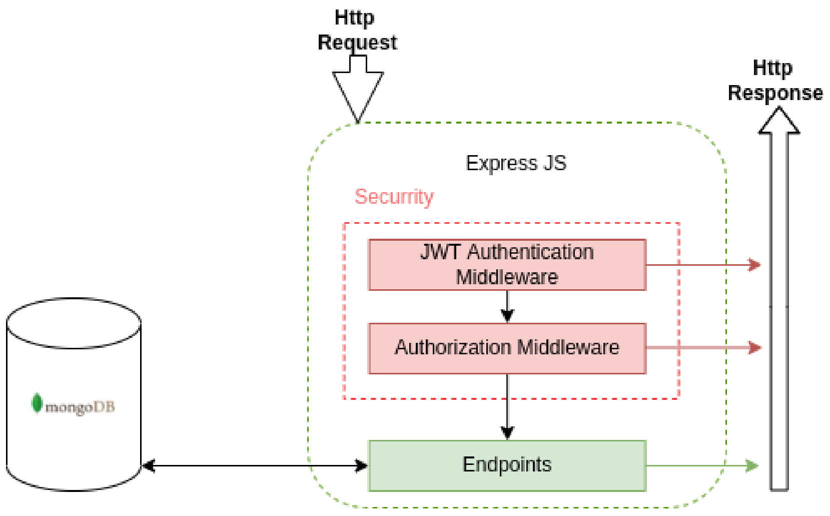


Figure 5.4: Node API with JWT Authentication and authorization mechanism

5.2.2 Media stream

This component is responsible to deliver media stream to the dashboard so that an admin could watch the camera stream. In order to make this possible it was required to use RTSP, as stated in chapter4. After some research on how to develop such component, it was found a solution that is shown on figure 5.5. The system is composed by a file containing the IP addresses of the RS, a video service responsible for creating a rtsp stream that uses FFMEPG [43] to process the given stream and outputs a file, in this case a mpegts file that will be deliverable to the websocket client that can then display it at the canvas player.

Nevertheless, it should be stated that, while this implementation seems a very easy and clean way, other concern arrived when developing the media stream. The UX (User experience) was poor hence if more than one user uses the stream the website got crashed and this need to be addressed. Therefore, it was decided to employ the use of threads. Node js has a library that supports threads [48] and they're very easy to implement. With this, before a stream gets created, a new flow was added to empower the quality of the stream in the UI. Figure 5.6 depict the scenario when a user (admin) request to see the camera of a selected RSU and 3 hypothesis can happen: stream does not exist on rsu; stream already running; stream being created by a new thread.

In a nutshell, user requests to see the camera , then the endpoint will check if it exists and, if does, it will verify if the stream is being used or needs to be created. If the stream needs to be created a thread will handle this process by getting the parameters sent in the http request so that, in the end could create a stream to listening on port 8000 + rsu identifier and the remaining specifications (i.e stream's Url, ffmpegOptions). At the client side a websocket client will listen to the channel and reproduce the data to a canvas and, as a result, the user gets to see in real time the camera stream.

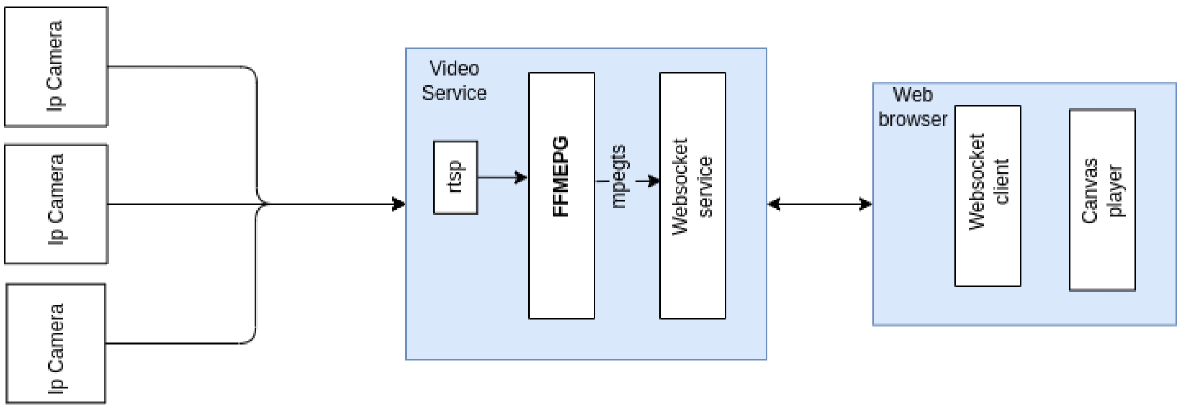


Figure 5.5: Diagram of media stream flow

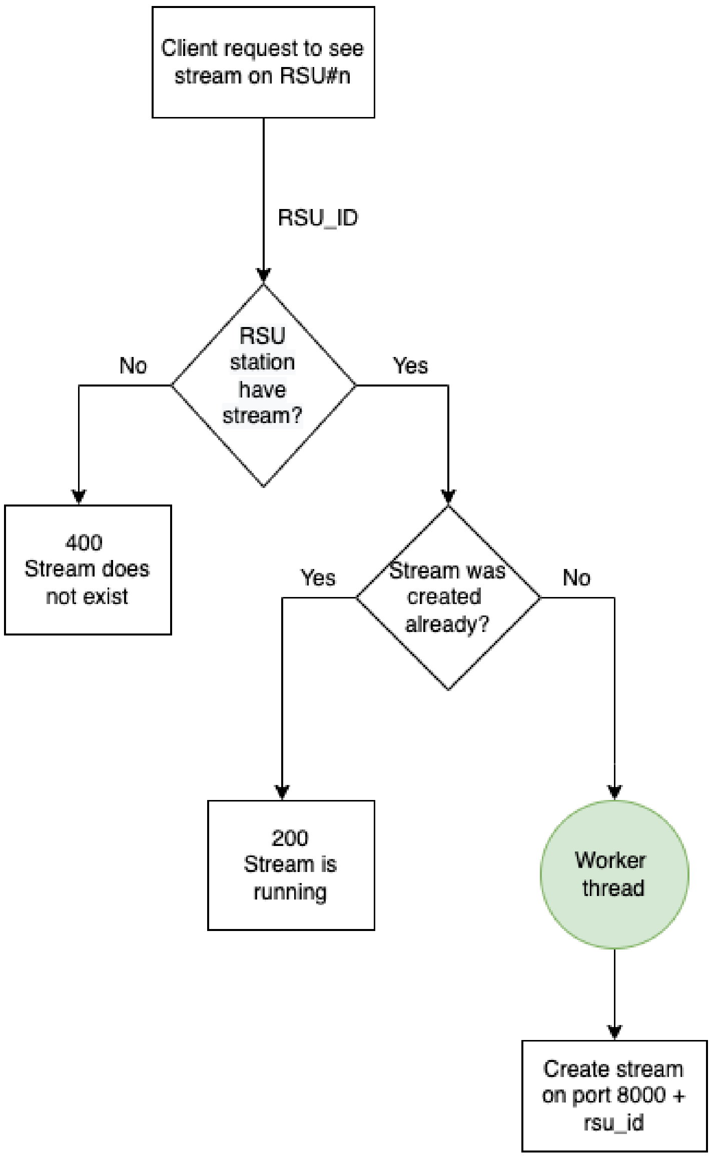

Client request to see stream on RSU#n
RSU_ID
RSU station have stream?
No
Yes
400 Stream does not exist
Stream was created already?
Yes
No
200 Stream is running
Worker thread
Create stream on port 8000 + rsu_id

Figure 5.6: Process of requesting a camera stream

5.2.3 Create Alerts

Unlike the other data streams, DENM messages are generated based on an event that needs to be created by an admin user. Such event is called Alert and it is a component, developed on the front-end, that has all DENMs causes and sub-causes that a user might send. Figure 5.7 depicts a scenario of creating an alert and the DENM generation.

When the user selects the Create Alert tab, he/she will be prompted to enter his/her credentials, and if he/she has administrative access, he/she will be presented with a form that lists the DENM causes and sub-causes. After defining the alert duration, an HTTP request to the Alerts/ endpoint is sent, which will send the alert to the broker and then publish to the topic. Additionally, it will store the alert and return to the user an HTTP response containing the alert information, which can then be used to display the alert with the icon in the user-selected position. Lastly, all users may access a DENM list on the left tab to see a very recent DENM sent; thus, a user may want to know whether the road he/she is about to drive is having construction or if an accident has occurred.

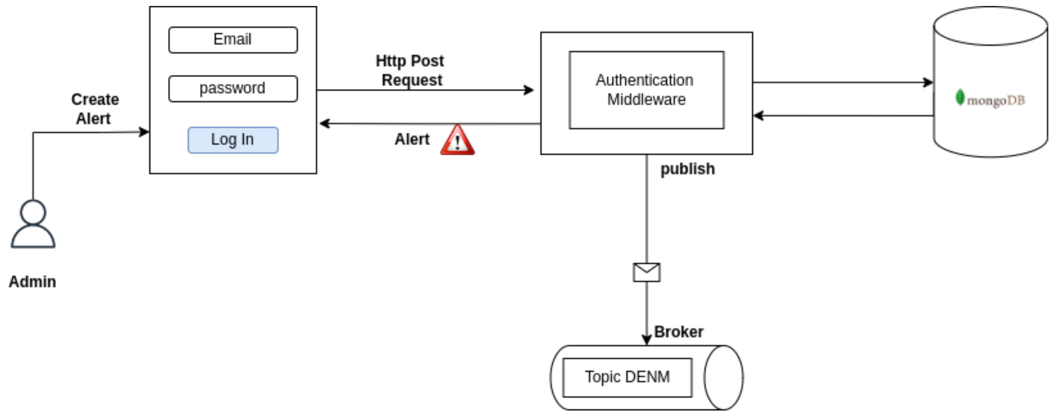


Figure 5.7: Diagram flow of a DENM generation

5.2.4 Statistics

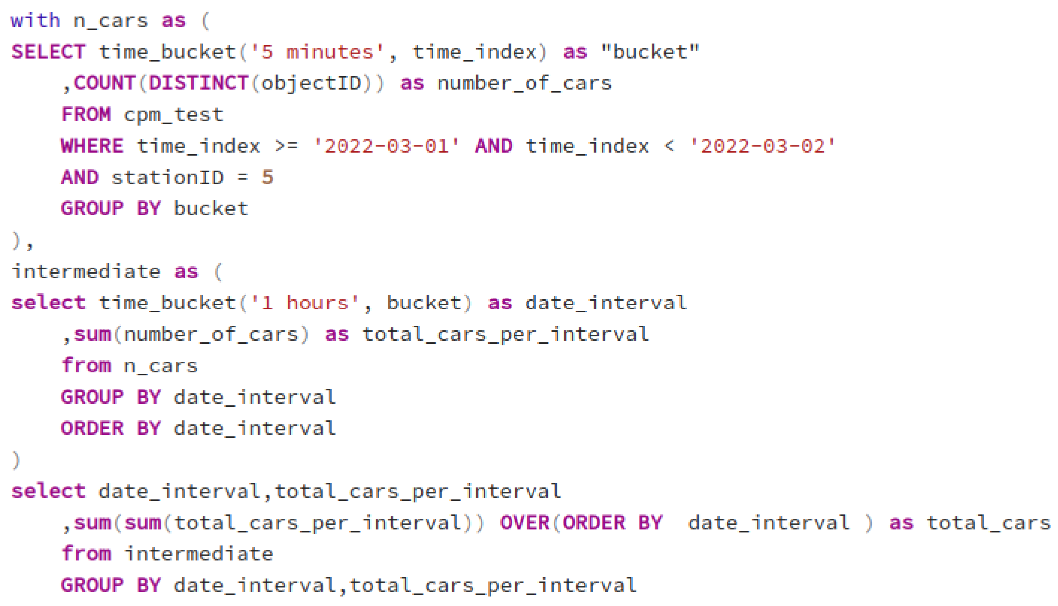
Last but not least, the statistics component appears as the last requirement for the system though it represents one of the major ambitions hence from here it is possible to provide traffic statistics to all users. As so, it was thought to allow the user interact with the platform and therefore, choose a radar to check the traffic metrics. On top of that, the user can and should consult in the calendar what time interval he/she might be interested to see. React js allows the use of this feature through the use of the library [39].

Later a HTTP request will be sent to the API endpoint with the time frame and RSU selected so that later they can be used to query the hypertable. Due to traffic metrics to graphics were take in consideration, the first is responsible to give the information of the average max and min speed within a time frame selected among with the number of cars in each time frame, while the second is responsible to give number of cars that are from: 0 to 50km/h; 50-90km/h; 90-120 km/h; >120 km/h. With this feature it is expected to give the user the most important traffic metrics historically.

As several edge cases where taken in consideration since displaying traffic velocities in one day may have 24 bars (1 bar for each hour), if we have in consideration other cases like from two to five days; or from one week to three weeks or from one month to eleven months and so on. Soon became clear that several functions need to respond to different time frames. For instance, if time interval is more than 2 days and less or equal of 2 weeks the $time\_bucket =$ 1day which subsequently gives a range of three to fifteen bares.

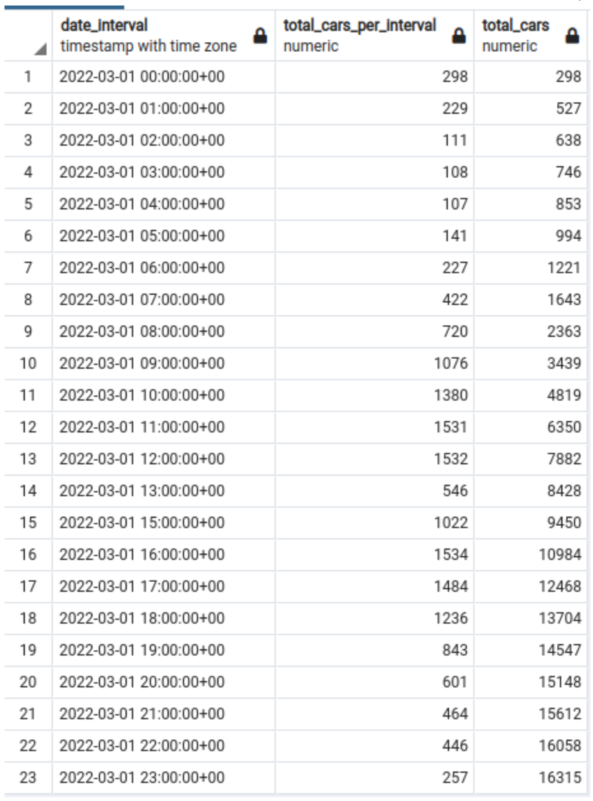
5.2.4.1 Timescale queries

Looking more closely to how the traffic metrics are performed it all comes to how the timescaledb queries are developed to attain these sort of needs. Starting from the beginning, it is known that now two graphic bars are presented to the user. The first starts to present the average minim and maximum velocities for each *time_bucket*. A time bucket basically



```sql
with n_cars as (
SELECT time_bucket('5 minutes', time_index) as "bucket"
    ,COUNT(DISTINCT(objectID)) as number_of_cars
    FROM cpm_test
    WHERE time_index >= '2022-03-01' AND time_index < '2022-03-02'
    AND stationID = 5
    GROUP BY bucket
),
intermediate as (
select time_bucket('1 hours', bucket) as date_interval
    ,sum(number_of_cars) as total_cars_per_interval
    from n_cars
    GROUP BY date_interval
    ORDER BY date_interval
)
select date_interval,total_cars_per_interval
    ,sum(sum(total_cars_per_interval)) OVER(ORDER BY  date_interval ) as total_cars
    from intermediate
    GROUP BY date_interval,total_cars_per_interval
```

(a)



	date_interval timestamp with time zone	total_cars_per_interval numeric	total_cars numeric
1	2022-03-01 00:00:00+00	298	298
2	2022-03-01 01:00:00+00	229	527
3	2022-03-01 02:00:00+00	111	638
4	2022-03-01 03:00:00+00	108	746
5	2022-03-01 04:00:00+00	107	853
6	2022-03-01 05:00:00+00	141	994
7	2022-03-01 06:00:00+00	227	1221
8	2022-03-01 07:00:00+00	422	1643
9	2022-03-01 08:00:00+00	720	2363
10	2022-03-01 09:00:00+00	1076	3439
11	2022-03-01 10:00:00+00	1380	4819
12	2022-03-01 11:00:00+00	1531	6350
13	2022-03-01 12:00:00+00	1532	7882
14	2022-03-01 13:00:00+00	546	8428
15	2022-03-01 15:00:00+00	1022	9450
16	2022-03-01 16:00:00+00	1534	10984
17	2022-03-01 17:00:00+00	1484	12468
18	2022-03-01 18:00:00+00	1236	13704
19	2022-03-01 19:00:00+00	843	14547
20	2022-03-01 20:00:00+00	601	15148
21	2022-03-01 21:00:00+00	464	15612
22	2022-03-01 22:00:00+00	446	16058
23	2022-03-01 23:00:00+00	257	16315

(b)

Figure 5.8: (a) Query format (b) Query result

allows to aggregate data within a time interval, e.g 5 minutes or 1 hour. So the first query is pretty straightforward.

Moreover, to a complement of the first bar a second was added: the number of cars per time interval. Figure 5.8a depicts the second query used to approach this requirement. At this point a lot of considerations had to be kept in mind: the number of cars are counted from

1 to 127 and consequently after the 127th car the next one would be the number 1 again. In order to count the number of cars within a time interval, first it was required to calculate the number of cars that are distinct from each other within a time interval that could be realistic, i.e a time interval of 5 minutes was choose hence, the traffic flow varies (most people work in prime time). Afterwards an intermediate step is added to aggregate the number of cars counted by each time frame of five minutes, i.e a sum of all cars counted in each five-minute time interval is aggregated in one hour time interval as $total\_cars\_per\_interval$. Finally the total number of cars ($total\_cars$) for the time interval required by the user is calculated by the sum of the $total\_cars\_per\_interval$.

5.3 UI features

This module offers the user interface through which it is possible to interact with the system. As previously indicated, the platform operates on a separate React-based process that has access to all system functionality. It is compatible with all major web browsers, therefore it is accessible on mobile devices while not being completely optimized for them.

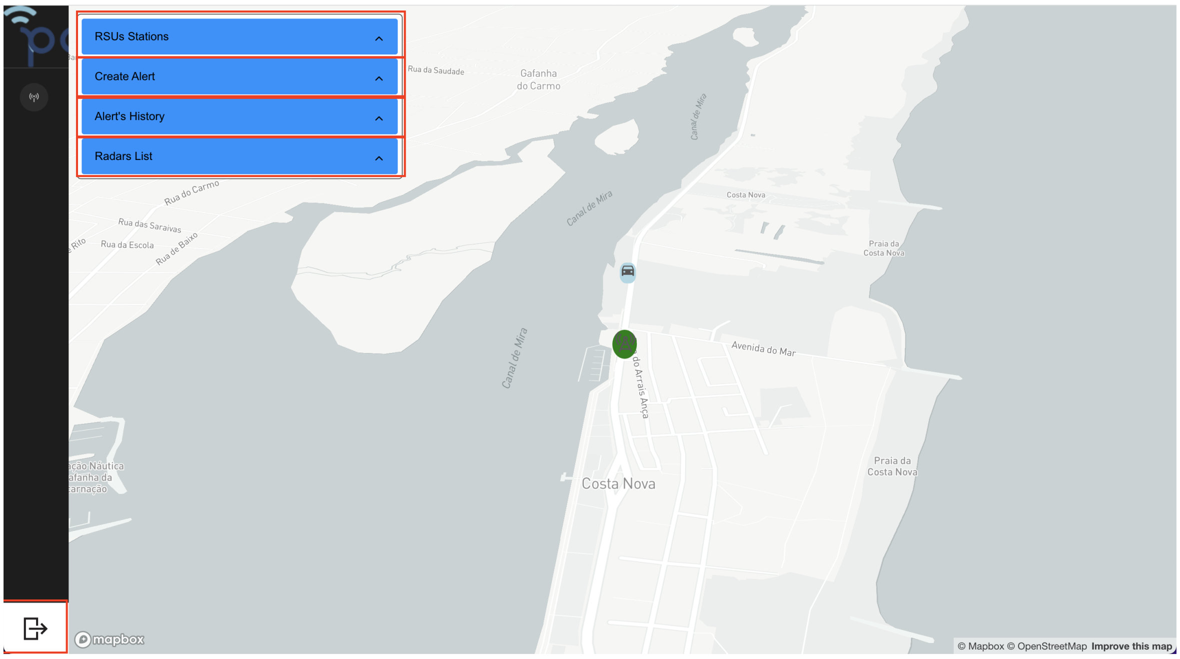
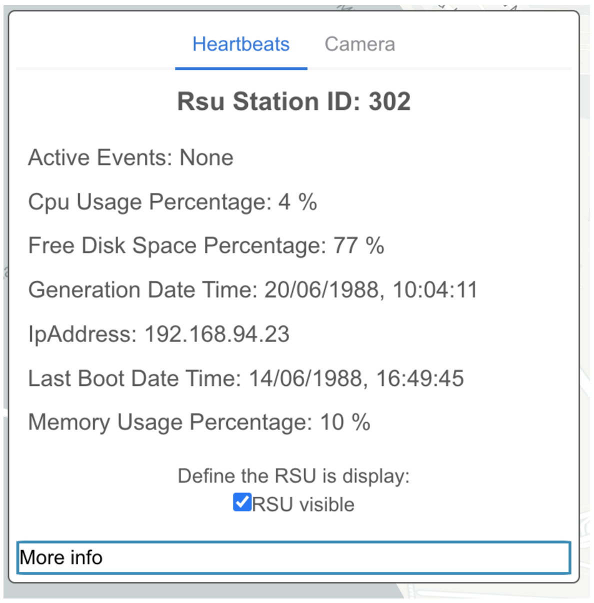


Figure 5.9: Main page visible to the Administrator

Since the Use Case (UC) is so specific it was decided to implement a web page from scratch. Therefore, it was implemented the best away to address this need by creating one page, RSU page, and within that page several components were developed, according with the use cases. Figure 5.9 depicts the main components developed with a red rectangle.

Consequently, the left tab may be separated into four distinct components. For the RSUs list, Figure 5.12a displays a situation in which a user may pick an RSU, and when this user does so, a right tab with information about the most recent message will emerge, and the administrator will have access to further capabilities (Figure 5.10). The administrator may hide the RSU from normal users, see the video feed, and view the most recent messages. Additionally two more components were developed to create and visualize the alerts. Figure 5.12c depicts the Create Alert tab containing a form and figure 5.12d depicts the alert's history tab where a user may check the alert's information. Lastly the Radars tab was added (Figure 5.12b)so that the user could choose a radar and watch the traffic statistics (Figure 5.11).


Heartbeats Camera

Rsu Station ID: 302

Active Events: None

Cpu Usage Percentage: 4 %

Free Disk Space Percentage: 77 %

Generation Date Time: 20/06/1988, 10:04:11

IpAddress: 192.168.94.23

Last Boot Date Time: 14/06/1988, 16:49:45

Memory Usage Percentage: 10 %

Define the RSU is display:
☑RSU visible

More info

Figure 5.10: Right tab with additional features

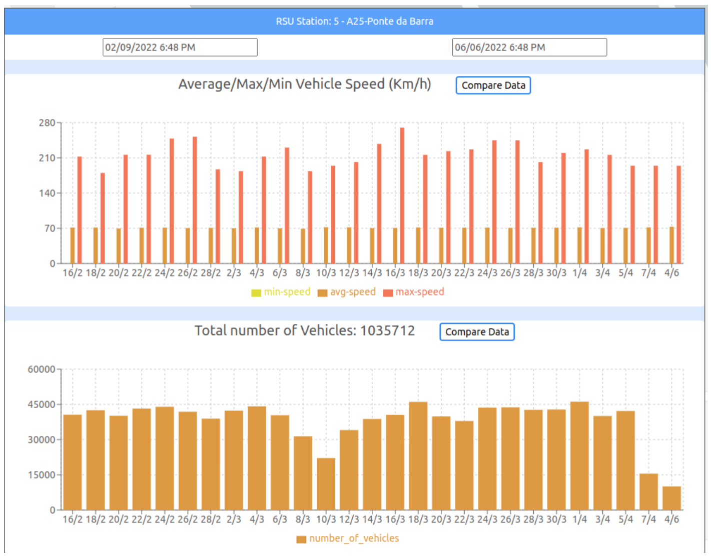
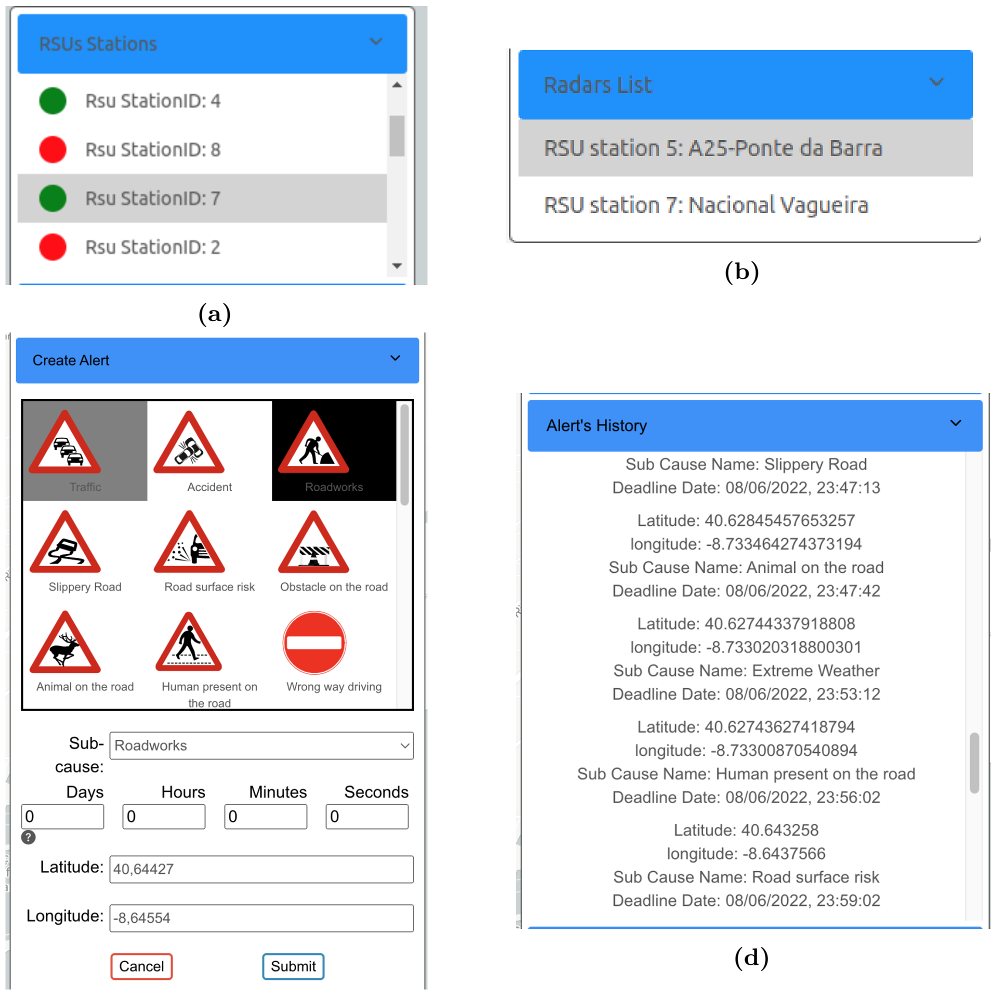


Figure 5.11: Charts based on the traffic metrics

Configuration instructions are described in B.



RSUs Stations

● Rsu StationID: 4

● Rsu StationID: 8

● Rsu StationID: 7

● Rsu StationID: 2

(a)

Radars List

RSU station 5: A25-Ponte da Barra

RSU station 7: Nacional Vagueira

(b)

Create Alert

Sub-cause: Roadworks

Days	Hours	Minutes	Seconds
0	0	0	0

Latitude: 40,64427

Longitude: -8,64554

(c)

Alert's History

Sub Cause Name: Slippery Road
Deadline Date: 08/06/2022, 23:47:13

Latitude: 40.62845457653257
longitude: -8.733464274373194
Sub Cause Name: Animal on the road
Deadline Date: 08/06/2022, 23:47:42

Latitude: 40.62744337918808
longitude: -8.733020318800301
Sub Cause Name: Extreme Weather
Deadline Date: 08/06/2022, 23:53:12

Latitude: 40.62743627418794
longitude: -8.73300870540894
Sub Cause Name: Human present on the road
Deadline Date: 08/06/2022, 23:56:02

Latitude: 40.643258
longitude: -8.6437566
Sub Cause Name: Road surface risk
Deadline Date: 08/06/2022, 23:59:02

(d)

Figure 5.12: Right tab with main features. (a) depicts the RSU list; (b) depicts the radar's list; (c) depicts the Alert form (d) depicts the Alert history tab

CHAPTER 6

Tests and Results

Taking in consideration the system development, numerous performance and usability tests were undertaken on the applications. Depending on the type of service, the used techniques will vary, and the results of those ways will be described in depth, along with an explanation of whether or not they result in a high level of performance.

6.1 Performance tests on database

An important feature of the system was the need to handle radar data. As mentioned before, timescaledb was selected since it has the functionality hypertables. Several experiments were conducted to get a comprehensive understanding of its power. Consequently, the data being added to the database was saved in a table. This table was a basic PostgreSQL table to which two indexes were added to accelerate the operation. Additionally, the same indexes were applied to the cloned secondary table. After the data was inserted into both databases, a script was developed to dynamically call the tables for each query. The script was executed five thousand times for each query per time period. Consequently, a csv file was uploaded, and afterwards, another script determined the mean and standard deviation based on this output.

Moreover, the tables 6.1 and 6.2 presented the results for each time interval. As a result, a third table was obtained 6.3 containing the speed up for each time interval for each query. In addiction to this, figures 6.1 6.2 are also provided and through them it is possible to see the average execution time per time interval.

With all this information it can be concluded that for a time interval of one day the speed up is brutal. Nonetheless, in the first query it is possible to see that, as the time interval gets longer, the execution times are better in relation to the regular table, though for seven and sixty days this showed otherwise. For the second query it is possible to conclude that the longer the time interval, the worse the speed up gets.

After having all in consideration, the hypertable proved to be always faster though some times it reveled a bit inconsistent. Although it has to be in mind that the queries diverge from each other and the second one reveled to be the more challenging.

First query - avg,min,max				
	Postgresql table results		Hypertable results	
Time Interval	Avg execution time (sec)	Standard deviation	Avg execution time(sec)	Standard deviation
1 days	1.5582	0.1184	0.2272	0.01693
3 days	2.0095	0.1181	0.9144	0.0992
5 days	2.8256	0.3671	1.2790	0.1187
7 days	3.4878	0.2411	1.5936	0.1434
15 days	4.9681	0.2665	1.8897	0.1906
20 days	6.7295	0.6305	2.1897	0.2381
30 days	8.1984	0.4983	2.5523	0.3188
60 days	10.3660	0.7334	5.670	0.5760

Table 6.1: Performance table for the first query

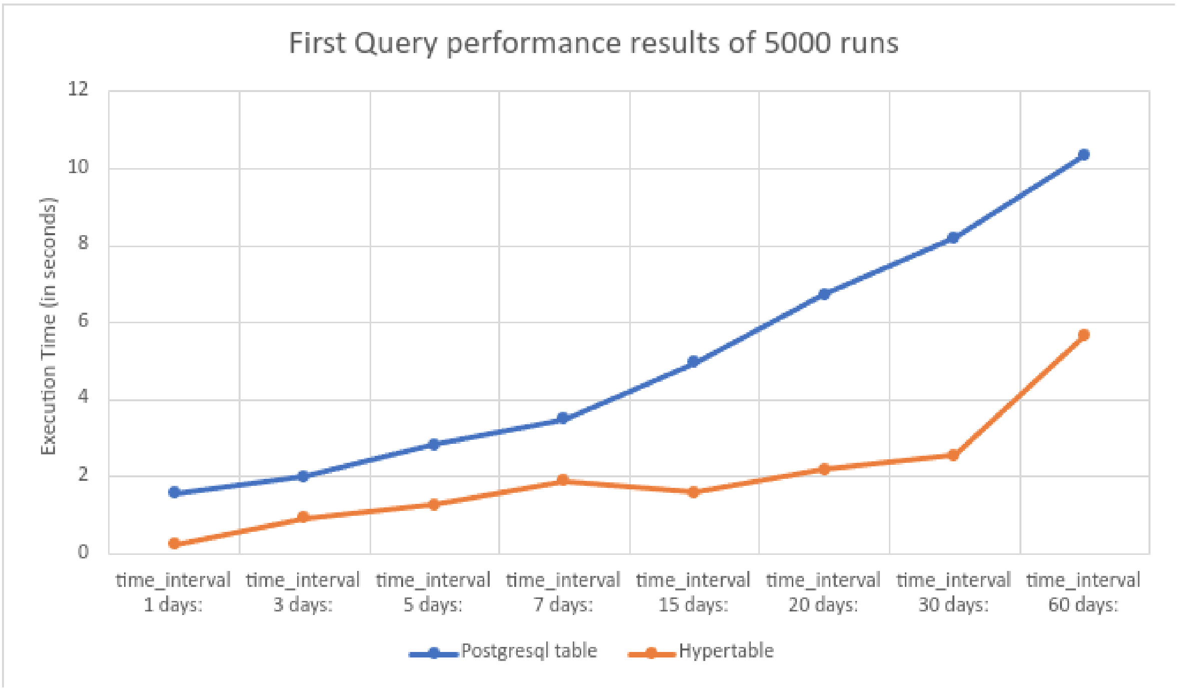


Figure 6.1: Performance results with 5000 executions for each time interval

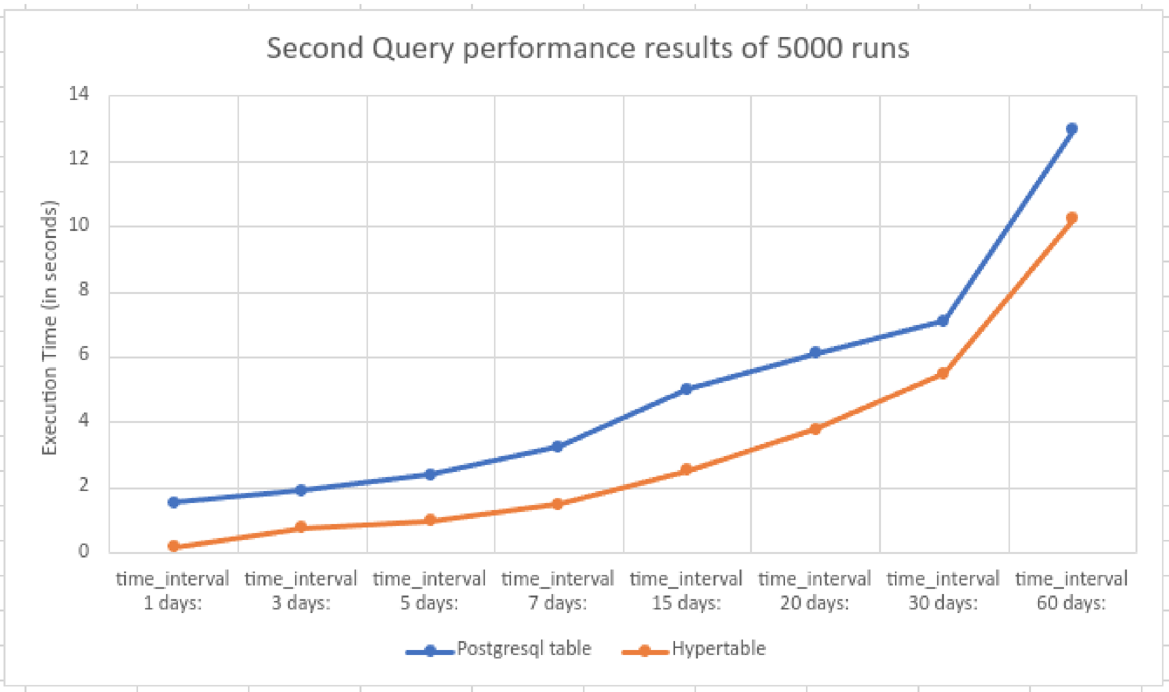


Figure 6.2: Performance results with 5000 executions for each time interval

Second query - Number of cars per time interval				
Time Interval	Postgresql table results		Hypertable results	
	Avg execution time(sec)	Standard deviation	Avg execution time(sec)	Standard deviation
1 days	1.5417	0.1319	0.1655	0.01482
3 days	1.9202	0.0951	0.7565	0.0415
5 days	2.3973	0.20901	0.9756	0.0518
7 days	3.2582	0.4170	1.4858	0.0792
15 days	5.0181	0.2331	2.5383	0.1005
20 days	6.108	0.2413	3.7911	0.1701
30 days	7.1150	0.5279	5.4987	0.3467
60 days	12.9627	2.41708	10.2382	0.4661

Table 6.2: Performance table for the second query

	Speed Up	
Time Interval	First Query Results	Second Query Results
1 day	6.858 x	9.32 x
3 days	2.20 x	2.53 x
5 days	2.20 x	2.45 x
7 days	1.84 x	2.19 x
15 days	3.12x	1.98 x
20 days	3.07 x	1.61 x
30 days	3.21 x	1.29 x
60 days	1.82 x	1.26 x

Table 6.3: Table showing the speedup of the hypertable against the regular postgresql table for each time interval

6.2 Usability Tests on UI

Usability testing is a method used in user-centered interaction design to assess a product via user testing. This is a crucial usability strategy, since it provides immediate feedback on how actual users interact with the system [49].

Therefore it was conducted a testing scenario with a set of users representing two different types: the users with no IT background and users with IT background. With these two groups it is guaranteed that either a regular user or a IT user can offer feedback on either the UI is forming as it should be. Cognitive Walkthrough and Heuristic evaluation were used to administer the tests. These give sufficient information for the assessor to understand how the users see the program.

6.2.1 Cognitive Walkthrough

A cognitive walkthrough is an organized method for assessing a product's usability. It entails the tester, who is not a user, asking four straightforward questions regarding how a particular user journey is executed.

Since none of the users had knowledge in the field of road side units and cooperative systems a brief contextualization was made. The average age of the user was about 46 years old and so, they reveled very confident to conduct the test hence had already a lot experience in apps or website. This population was choen because the goal here was to guarantee that the platform could be simple to use.

Before executing the tasks, the participants were given two to three minutes to familiarize themselves with the program. Note that the system has two target users and so the test was made to simulate a regular user and a admin user. Since the admin privileges give full access of the dashboard it was decided to create a set of tasks that could simulate a admin interacting with the UI. The tasks for each page are as follows:

- Login as an administrator.
- Once login, go to the map and explore it. Then find the RSUs deployed near Barra's beach.
- Access the RSU 171717 through the RSU list.
- Visualize the more recent message on the RSU selected.
- Explore the reaming messages.
- Access the radar's camera an see the message provided.
- Access the RSU 5 radars' camera and see the media stream.
- Interact with the cars moving across the RSU 5 by watching closely their speed.
- Go back to the normal system's state.
- Create an alert of a given cause and sub-cause, drag the pin around the RSU 5 and define 5 minutes of duration and finally submit the alert.
- Verify if the alert was correctly generated around RSU 5.
- Verify in alert history that the alert was emitted successfully.
- Go back to the normal system's state.
- Access radar list and choose one of the provided to interact with.

- Change the interval time and interact with the charts. Try to understand the graphics behaviour.
- Logout.

The number of users doing the tests was eight, and each participant was required to fill out a form after completing each activity. The form consisted of a series of questions assessing the difficulty of each assignment on a scale from 1 to 5, with 1 being the most difficult and 5 the least. In addition, figure 6.3 revealed that 75% of testers give the first assignment a score of 2 and the remainder a score of 3. In addition, figure 6.4 stats that 25% rank the creation of an alert as a 1, while 62.5% score it as a 2, and the remaining users score it as a 3. In addition, figure 6.5 revealed that 50% of users rated the task of visualize the alerts history a score of 3, 37% scored 2 and the remaining scored 4. The remaining tasks were graded between 4 and 5, therefore the total performance was satisfactory.

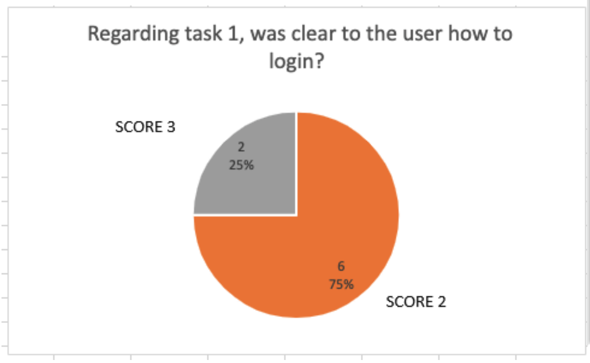


Figure 6.3: Depicts the scores of the login

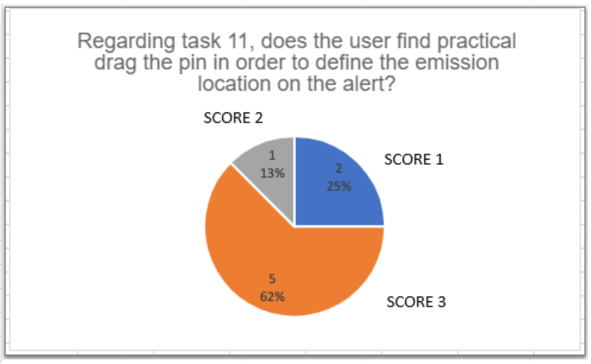


Figure 6.4: Depicts the score of the creation of the alert

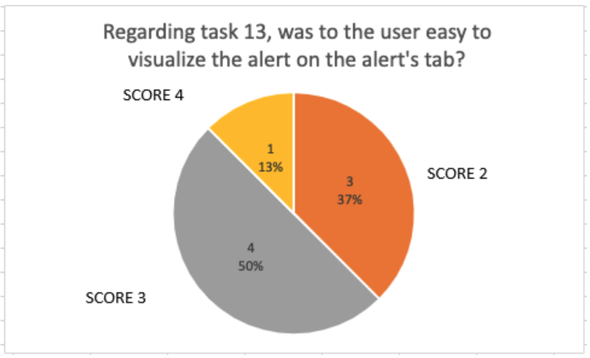


Figure 6.5: Depicts the score of the alert history

In conclusion, the testers discussed the tasks with the lowest scores. For the first, they felt that it was unclear how to login when the tab is not collapsed; for the second, they complained that a user had to look for the pin on the map and drag it to the desired location; and for the third, that it would be preferable to have ascending or descending order buttons to locate the most recent alerts.

6.2.2 Heuristic Evaluation

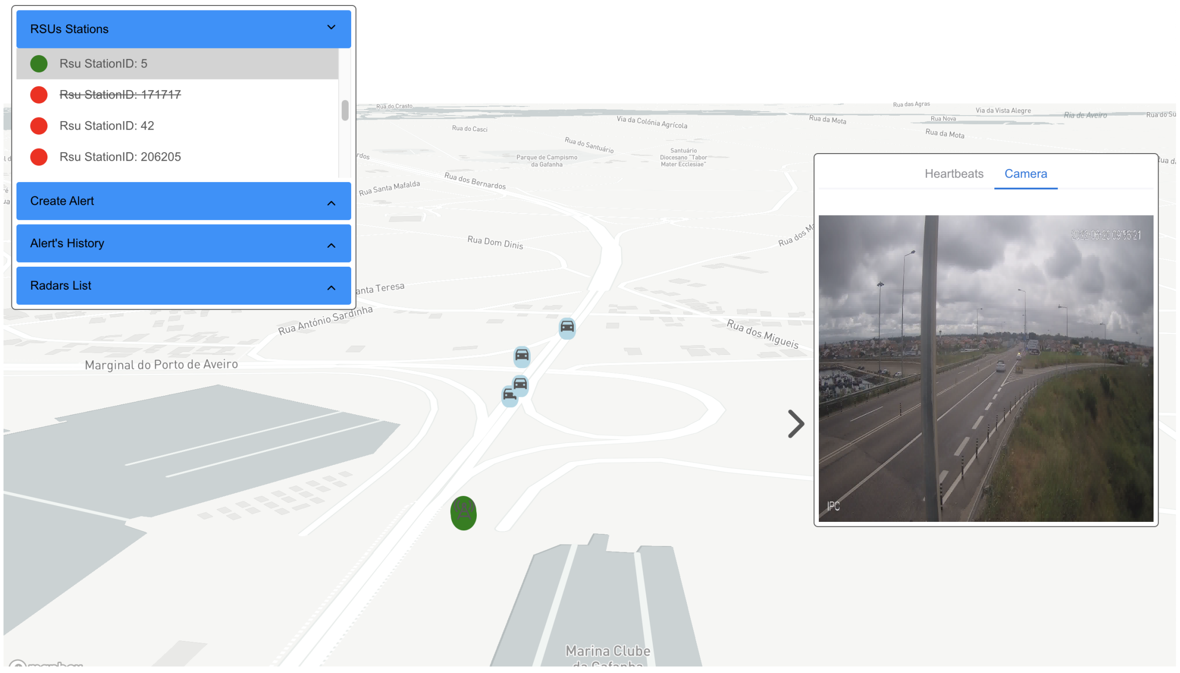
A heuristic evaluation is a software usability inspection technique that helps to uncover usability issues in the user interface design. It especially entails assessors reviewing the interface and assessing its conformance with established usability guidelines. Therefore the technique most used is through the use of 10 Laws of Usability by Jakob Nielsen [50].

These 10 Heuristic are the ones used for the tests, which provide the most complete general principles for interaction design. Since the users felt that the three tasks spoken above were the ones that required further attention, the test was focused on each of them. For the login the heuristic was "Visibility of system status" and the testers evaluated with a severity of 3. For the creation of an alert the heuristic was "Flexibility and efficiency of use" and the severity gave by the testers was 2. Lastly for the alert visualization on the tab the heuristic was "Visibility of system status" with a severity of 3.

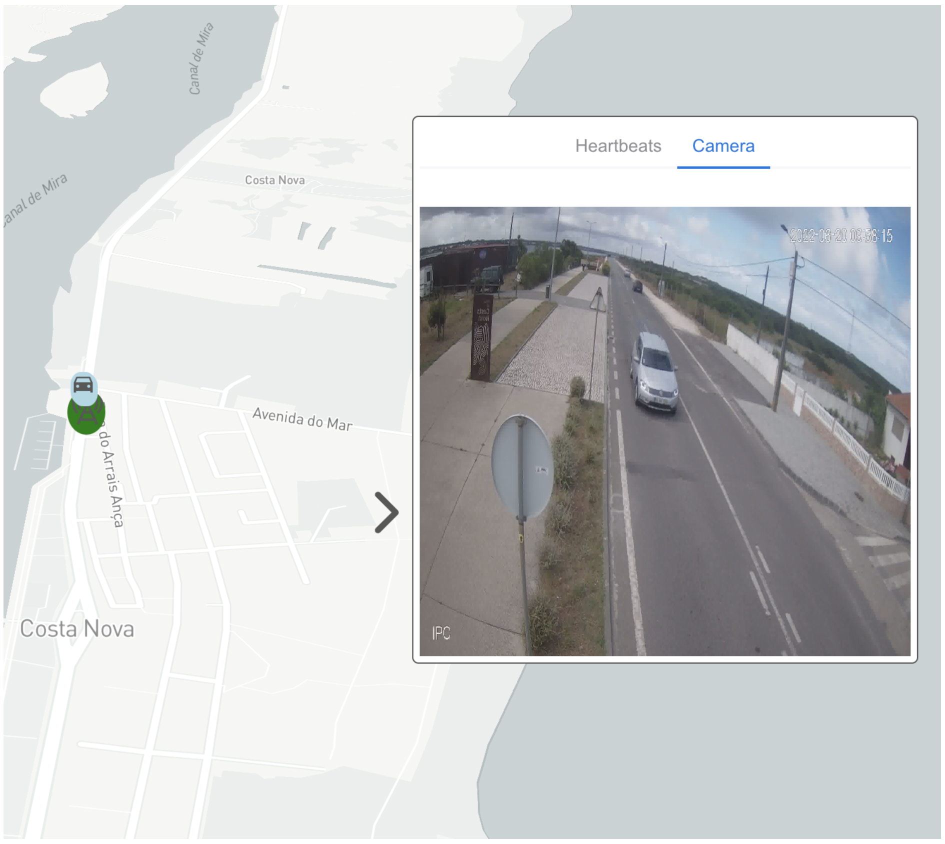
6.3 Camera Stream

As mentioned earlier, the camera stream was one of the most difficult features to build, thus the user experience and platform performance have to be considered when the stream is active. Numerous testing were undertaken to ensure that this feature was reliable and ready for usage at any moment, as everything seemed to go as planned. Early experiments were done using a publicly available RTSP stream (`rtsp://wowzaec2demo.streamlock.net/vod/mp4:BigBuckBunny_115k.mp4`). A second step was undertaken after multiple user experience and performance testing. During this phase, more rigorous testing were undertaken, hence they were conducted in IT facilities. Note that radar streams can only be seen in the IT lab, therefore this function is completely inaccessible to normal users other than IT administrators and field researchers.

In addition, there are two operational cameras placed in RSUs 5 and 7. As the primary objective of this feature was to determine whether the camera is synced with the radar, tests were conducted to ensure that this behaviour was implemented correctly. Therefore, after a series of tests, it was feasible to determine that the camera on RSU 7 was precisely synced with the radar, whereas the camera on RSU 5 displayed the automobiles detected by the radars with a lag. Recall that the radar data and camera stream are shown in real-time through websockets. Figures 6.6a and 6.6b depict the cases stated before and there it is possible to see that on RSU 7 the stream is synced with the radar's perceived vehicle whereas in RSU 5 the stream.



(a)



(b)

Figure 6.6: (a) RSU 5 shows some lag thus its CPM's appear on the road but on the camera only one car is capture. (b) RSU 7 shows that the camera is working as expected thus it detected the vehicle and the same is perceived by the radar.

CHAPTER 7

Conclusion and Future Work

This chapter finishes the dissertation with an overview of the whole body of work. The study results gathered from the research will then be presented. This chapter concludes with a discussion of future work that may be conducted in addition to the work outlined in the dissertation.

7.1 Conclusion

Taking into account the important elements to be created, we have the following: the display of C-ITS messages, the ability to have a platform with administrative capabilities, the visualisation of the camera's stream, and the charts.

Since these, all C-ITS messages are supported by the platform, but it must be stated that the VAM and CAM messages proved to be the least useful in the study, as they simply supplied the current location and it was not possible to derive a metric from such little information.

For DENMs, the implementation proved to be effective, thus each time a user makes an alert, it generates the event with the right payload, and the DENM is then sent to every user that uses the mobile app in that region.

Moreover, the CPM messages proved to be of top standard, so when participants were doing usability testing, they were pleased to see the automobiles in real time. The most important aspect of this was the graphics containing traffic statistics. This final one was likewise highlighted by the testers as one of the website's top features.

In addition, the heartbeat messages were a success, since the researchers found this feature to be highly beneficial for determining what may have gone wrong with the RSU.

For the remaining functionalities, the platform's ability to provide administrative rights enhances the security as envisioned. Regarding the radar cameras, it was one of the most difficult to execute, but the final results were better than anticipated, as it was posible to see whether or not the camera was synchronized with the radar.

Moreover, the implementation of the hypertable was a massive success, as the results shown, and with this technology it is feasible to create a scalable database capable of holding billions of records (right now the hypertable has a total of 54 millions rows, occupying nearly 6gb).

In conclusion, the proof-of-concept system was a success as all system criteria were satisfied. All the selected tools and technologies were flawless, however React lacked performance in specific instances, such as when there is a lot of traffic and charts are being shown at the same time, or while viewing the live stream.

7.2 Future Work

Following the work carried on in this dissertation others features can be added so that later the overall system can evolve and provide more detailed traffic behaviors. These features include:

- Fixing the features that got the worse feedback on the cognitive walkthrough.
- Development of statistics based on the CAM messages.
- Add a third query in the right tab to provide more detailed traffic measure. Chart with the number of cars per time interval for the scenarios: 0-50Km/h; 50-90Km/h; 90-120Km/h; $>$ 120Km/h
- Add more statistics regarding the number of cars for each direction, i.e north - south and south - north.
- Possibly add data compression on the hypertable when the data storage becomes much large, which would improve the database scalability.

APPENDIX A

Table A.1: Cause description and cause code assignment for ETSI use case

Begin of Table			
Cause code Description	Cause code	Sub Cause code	Sub Cause code Description
Traffic condition	1	0	Traffic
		1	Increased Traffic Volume
		2	Traffic Jam Slowly Increasing
		3	Traffic Jam Increasing
		4	Traffic Jam Strongly Increasing
		5	Stationary Traffic
		6	Traffic Jam Slightly Decreasing
		7	Traffic Jam Decreasing
		8	Traffic Jam Strongly Decreasing
Accident	2	0	Accident
		1	Multi-Vehicle Accident
		2	Heavy Accident
		3	Accident involving Lorry
		4	Accident involving Bus
		5	Accident involving Hazardous Materials
		6	Accident on Opposite Lanes
		7	Unsecured Accident
		8	Accident with Assistance Requested
Roadworks	3	0	Roadworks
		1	Major Roadworks
		2	Road Marking Works
		3	Slow Moving Road Maintenance
		4	Short term Stationary Roadworks
		5	Street cleaning

Continuation of Table A.1			
Cause code Description	Cause code	Sub Cause code	Sub Cause code Description
		6	Winter service
Adverse Weather condition	6	0	Slippery Road
		1	Heavy Frost on the Road
		2	Fuel on the Road
		3	Mud on the Road
		4	Snow on the Road
		5	Ice on the Road
		6	Black Ice on the Road
		7	Oil on the Road
		8	Loose Chippings
		9	Instant Black Ice
		10	Road Salted
Hazardous location Surface condition	9	0	Road surface risk
		1	Rockfalls
		2	Earthquake damage
		3	Sewer Collapse
		4	Subsidence
		5	Snow drifts
		6	Storm Damage
		7	Burst Pipe
		8	Volcano Eruption
		9	Falling ice
Hazardous location Obstacle on the road	10	0	Obstacle on the road
		1	Shed Load
		2	Parts of Vehicles
		3	Parts of Tyres
		4	Big Object on the road
		5	Fallen Tree
		6	Hub Caps
		7	Waiting vehicles
Hazardous location Animal on the road	11	0	Animal on the road
		1	Wild Animal on the road
		2	Herd of animals on the road
		3	Small animal on the road
		4	Large animal on the road
Hazardous location on the road	12	0	Human present on the road
		1	Children on roadway

Continuation of Table A.1			
Cause code Description	Cause code	Sub Cause code	Sub Cause code Description
		2	Cyclist on roadway
		3	Motorcyclist on roadway
Wrong way driving	14	0	Wrong way driving
		1	Wrong lane driving
		2	Wrong direction driving
Rescue and recovery work in progress	15	0	Rescue and Recovery Work in Progress
		1	Emergency Vehicle
		2	Rescue Helicopter Landing
		3	Police Activity Ongoing
		4	Medical Emergency Ongoing
		5	Child Abduction in Progress
Adverse weather condition - extremer weather condition	17	0	Extreme Weather
		1	Strong Winds
		2	Damaging Hail
		3	Hurricane
		4	Thunderstorm
		5	Tornado
		6	Blizzard
Adverse weather condition - visibility weather condition	18	0	Low Visibility
		1	Fog
		2	Smoke
		3	Heavy Snowfall
		4	Heavy Rain
		5	Heavy Hail
		6	Low Sun Glare
		7	Sandstorm
		8	Swarm of Insects
Adverse weather condition - Precipitation	19	0	Precipitation
		1	Heavy rain
		2	Heavy Snowfall
		3	Soft Hail
Slow vehicle	26	0	Slow vehicle
		1	Slow Maintenance vehicle
		2	Vehicle Slowing to Look at Accident
		3	Slow vehicle with Abnormal Load
		4	Slow vehicle with Abnormal Wide Load
		5	Slow vehicle convoy

Continuation of Table A.1			
Cause code Description	Cause code	Sub Cause code	Sub Cause code Description
		6	Low Sun Glare
		7	Slow snowplough
		8	Slow deicing vehicle
		9	Slow salting vehicle
Dangerous end of queue	27	0	Dangerous end of Queue
		1	Sudden end of Queue
		2	Queue over the hill
		3	Queue around the bend
		4	Queue in tunnel
Vehicle breakdown	91	0	Breakdown
		1	Lacking fuel
		2	Lacking battery power
		3	Engine problem
		4	Transmission problem
		5	Engine cooling problem
		6	Braking problem
		7	Steering problem
		8	Tyre puncture
Post crash	92	0	Post crash
		1	Accident without ECall triggered
		2	Accident w/ ECall manually triggered
		3	Accident w/ ECall automatically triggered
		4	Accident w/ ECall triggered without Cell access
Human problem	93	0	Human problem
		1	Glycemia problem
		2	Heart problem
Stationary vehicle	94	0	Stationary Vehicle
		1	Stationary Vehicle human problem
		2	Brokendown Vehicle
		3	Stationary Vehicle after Crash
		4	Stopped Public Transport
		5	Carrying dangerous goods
Emergency vehicle approaching	95	0	Emergency Vehicle Approaching
		1	Prioritized Vehicle Approaching
Hazardous location indication - Dangerous Curve	96	0	Dangerous curve
		1	Dangerous left turn curve
		2	Dangerous right turn curve

Continuation of Table A.1			
Cause code Description	Cause code	Sub Cause code	Sub Cause code Description
		3	Multiple dangerous curves
		4	Dangerous curves starting w/ left turn curve
		5	Dangerous curves starting w/ right turn curve
Collision risk	97	0	Collision risk
		1	Emergency Brake
		2	Pre Crash system engaged
		3	ESP engaged
		4	ABS engaged
		5	AEB engaged
		6	Brake Warning
		7	Warning
Signal violation	98	0	Signal violation
		1	Sign Violation
		2	Traffic Light Violation
		3	Traffic Regulation Violation
Dangerous Situation	99	0	Dangerous Situation
		1	Emergency Brake
		2	Pre Crash system engaged
		3	ESP engaged
		4	ABS engaged
		5	AEB engaged
		6	Brake Warning
		7	Collision Risk Warning
End of Table			

APPENDIX B

The service's configurations are depicted in B.1. The source code can be access in (`https://gitlab.es.av.it.pt/ITS/it2s-tms/`) and the following instructions are given to boot up the platform:

To run the back-end just need to download and install **Docker** and **docker-compose**, afterwards just type in the terminal docker-compose up if inside the Virtual Machine(VM) given by the Institute of Telecommunication(IT) otherwayse, it is need to install NodeJS. The following instructions specific the steps to run the application:

- To run the back-end inside the VPN IT just type in the terminal "docker-compose up" (in the path it2s-tms/backend/).
- To run the back-end locally install **Docker** and **docker-compose** and **nodejs** then just type in the terminal "jwtPrivateKey=123 node index.js" (in the path it2s-tms/backend/).
- To run the front-end just type in the terminal (assuming Node js is installed) "yarn start" (in the path it2s-tms/front-end/).
- To verify in real time the messages arriving at the broker install the MQTT explorer GUI (`http://mqtt-explorer.com/`). Afterwards connect to the mqqt socket as depicted in the table **B.1**.

Service	Service address	Service port
Node js	VM IT (private network)	8000
Mongo DB	VM IT (private network)	27017
Timescaledb	VM IT (private network)	5432
Mqtt	mqtt://ccam.av.it.pt	1883
React js	localhost	3000

Table B.1: Service's configurations.

Bibliography

[1] FinancesOnline, *Number of internet users in 2022/2023: Statistics, current trends, and predictions*, https://financesonline.com/number-of-internet-users/, (accessed 30/06/2022).

[2] A. D'Auria, M. Tregua, and M. Vallejo-Martos, "Modern conceptions of cities as smart and sustainable and their commonalities," *Sustainability*, vol. 10, no. 8, p. 2642, Jul. 2018. DOI: 10.3390/su10082642. [Online]. Available: https://doi.org/10.3390/su10082642.

[3] H. Company, *How many cars are there in the world in 2022?* https://hedgescompany.com/blog/2021/06/how-many-cars-are-there-in-the-world/, (accessed 30/05/2022).

[4] J. Andersen and S. Sutcliffe, "Intelligent transport systems (ITS) - an overview," *IFAC Proceedings Volumes*, vol. 33, no. 18, pp. 99–106, Jul. 2000. DOI: 10.1016/s1474-6670(17)37129-x. [Online]. Available: https://doi.org/10.1016/s1474-6670(17)37129-x.

[5] A. Paier, "The end-to-end intelligent transport system (its) concept in the context of the european cooperative its corridor," in *2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, 2015, pp. 1–4. DOI: 10.1109/ICMIM.2015.7117948.

[6] A. M and S. C. S, "RSU based efficient vehicle authentication mechanism for VANETs," in *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*, IEEE, Jan. 2015. DOI: 10.1109/isco.2015.7282299. [Online]. Available: https://doi.org/10.1109/isco.2015.7282299.

[7] M. Wolf and M. Eder, "V2i: Vehicle-to-infrastructure use cases and demonstrator,"

[8] D. Eckhoff, N. Sofra, and R. German, "A performance study of cooperative awareness in ETSI ITS G5 and IEEE WAVE," in *2013 10th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, 2013, pp. 196–200. DOI: 10.1109/WONS.2013.6578347.

[9] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 3: Network Architecture," *ETSI TS*, vol. 102, pp. 636–3, 2014.

[10] ——, "Intelligent transport systems (ITS); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service," *Draft ETSI TS*, vol. 20, no. 2011, pp. 637–2, 2014.

[11] ——, "Intelligent transport systems (ITS); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service," *Draft ETSI TS*, vol. 20, no. 2011, pp. 448–51, 2011.

[12] ——, "Intelligent transport systems (ITS); vehicular communications; basic set of applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service," *Draft ETSI TS*, vol. 20, no. 2011, pp. 448–51, 2014.

[13] ——, "Intelligent transport systems (ITS); vehicular communications; basic set of applications; Analysis of the Collective Perception Service (CPS); Release 2," *ETSI TS*, vol. 20, no. 2011, pp. 448–51, 2019.

[14] F. A. Schiegg, D. Bischoff, J. R. Krost, and I. Llatser, "Analytical performance evaluation of the collective perception service in IEEE 802.11p networks," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, May 2020. DOI: 10.1109/wcnc45663.2020.9120490. [Online]. Available: https://doi.org/10.1109/wcnc45663.2020.9120490.

[15] H.-J. Günther, R. Riebl, L. Wolf, and C. Facchi, "Collective perception and decentralized congestion control in vehicular ad-hoc networks," in *2016 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2016, pp. 1–8.

[16] ETSI, "Ts 103 300-3 - v2.1.2 - intelligent transport systems (its); vulnerable road users (vru) awareness; part 3: Specification of vru awareness basic service; release 2," 2021.

[17] S. Mobility, *Synchrone-mobilitaet*, `https://www.synchrone-mobilitaet.de/en/projects.html`, (accessed 05/02/2022), 2022.

[18] R. Jacob, M. Gay, M. Dod, *et al.*, "Ivs-kom: A reference platform for heterogeneous its communications," in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020, pp. 1–7. DOI: `10.1109/VTC2020-Fall49728.2020.9348580`.

[19] S. Strobl, M. Kloppel-Gersdorf, T. Otto, and J. Grimm, "C-ITS pilot in dresden – designing a modular c-ITS architecture," in *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, IEEE, Jun. 2019. DOI: `10.1109/mtits.2019.8883376`. [Online]. Available: `https://doi.org/10.1109/mtits.2019.8883376`.

[20] CORDIS, *5G for Connected and Automated Road Mobility in the European Union*, `https://cordis.europa.eu/project/id/825012`, (accessed 05/02/2022), 2022.

[21] E. D. B. e Silva, J. Vranckx, T. D. Bruyn, V. Charpentier, S. A. Hadiwardoyo, and J. M. Marquez-Barja, "A toolkit for visualizing v2x messages on the smart highway testbed," in *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, IEEE, Sep. 2021. DOI: `10.1109/ds-rt52167.2021.9576123`. [Online]. Available: `https://doi.org/10.1109/ds-rt52167.2021.9576123`.

[22] R. S. for Interoperability in the Adoption of Autonomous Driving in European Urban Nodes, *Auto c-its project*, `https://www.autocits.eu/`, (accessed 09/02/2022), 2022.

[23] J. E. Naranjo, F. Jimenez, J. J. Anaya, R. Castineira, M. Gil, and D. Romero, "Validation experiences on autonomous and connected driving in AUTOCITS pilot in madrid," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, IEEE, Jun. 2018. DOI: `10.1109/vtcspring.2018.8417888`. [Online]. Available: `https://doi.org/10.1109/vtcspring.2018.8417888`.

[24] J. P. Ribeiro, T. D. Dias, and L. T. Moura, *Deployment and interoperability tests for a portuguese network for c-its*, `https://www.a-to-be.com/wp-content/uploads/2022/02/2021_RIBEIRO_ITSWorld27_CRoads.pdf`, (accessed 12/02/2022), 2022.

[25] R. council of Norway, *Asam message services for c-its*, `https://prosjektbanken.forskningsradet.no/en/project/FORISS/296651?Kilde=FORISS&distribution=Ar&chart=bar&calcType=funding&Sprak=no&sortBy=date&sortOrder=desc&resultCount=30&offset=240&TemaEmne.2=Offentlig+sektor&source=FORISS&projectId=194071`, (accessed 19/02/2022), 2022.

[26] P. H. Nguyen, A. Hugo, K. Svantorp, and B. M. Elnes, "Towards a simulation framework for edge-to-cloud orchestration in c-ITS," in *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, IEEE, Jun. 2020. DOI: `10.1109/mdm48529.2020.00077`. [Online]. Available: `https://doi.org/10.1109/mdm48529.2020.00077`.

[27] A. Cooper, R. Reimann, and D. Cronin, *About Face3: The Essentials of Interaction Design.* Willey Publishing, 2007.

[28] Maptiler, *Maptiler tiles à la google maps*, `https://www.maptiler.com/google-maps-coordinates-tile-bounds-projection/#3/15.00/50.00`, (accessed 09/02/2022), 2022.

[29] M. Azure, *Azure zoom levels and tile grid*, `https://docs.microsoft.com/pt-pt/azure/azure-maps/zoom-levels-and-tile-grid?tabs=csharp`, (accessed 14/02/2022), 2022.

[30] Socket.io, *Introduction what socket.io is*, `https://socket.io/docs/v4/`, (accessed 14/02/2022), 2022.

[31] JWT, *JWT introduction to json web tokens*, `https://jwt.io/introduction`, (accessed 14/02/2022), 2022.

[32] M. Kamaruzzaman, *Architecture and its 10 most important design patterns*, `https://towardsdatascience.com/microservice-architecture-and-its-10-most-important-design-patterns-824952d7fa41`, (accessed 14/02/2022), 2020.

[33] nodejs.org, *Introduction what socket.io is*, `https://nodejs.org/en/about/`, (accessed 16/04/2022), 2022.

[34] M. B. Jones, *RFC 7519 json web token (jwt)*, https://datatracker.ietf.org/doc/html/rfc7519, (accessed 21/05/2022), 2015.

[35] Wikipedia, *Base64*, https://en.wikipedia.org/wiki/Base64#URL_applications, (accessed 21/05/2022).

[36] S. Josefsson, *RFC 4648 the base16, base32, and base64 data encodings*, https://datatracker.ietf.org/doc/html/rfc4648, (accessed 21/05/2022), 2006.

[37] REACT-MAP-GL, *REACT-MAP-GL*, https://visgl.github.io/react-map-gl/, (accessed 23/05/2022).

[38] recharts, *Recharts a composable charting library built on react components*, https://recharts.org/en-US/, (accessed 18/04/2022).

[39] R. Datepicker, *React Datepicker a simple and reusable datepicker component for react.* https://reactdatepicker.com, (accessed 24/05/2022).

[40] MongoDB, *MongoDB*, https://www.mongodb.com/, (accessed 18/04/2022), 2022.

[41] TimescaleDocs, *About hypertables*, https://docs.timescale.com/timescaledb/latest/how-to-guides/hypertables/about-hypertables/#hypertable-partitioning, (accessed 18/04/2022), 2022.

[42] WebRTC, *Webrtc contributors*, https://webrtc.org/, (accessed 03/06/2022).

[43] c. A complete cross-platform solution to record, stream audio, and video., *Ffpmeg*, https://ffmpeg.org/, (accessed 03/06/2022).

[44] AWS, *What are microservices?* https://aws.amazon.com/microservices, (accessed 03/06/2022).

[45] TimescaleDocs, *Insert data*, https://docs.timescale.com/timescaledb/latest/how-to-guides/write-data/insert/#insert-multiple-rows, (accessed 04/06/2022).

[46] KirstenS, *Cross site scripting (xss)*, https://owasp.org/www-community/attacks/xss/, (accessed 04/06/2022).

[47] C. Sidoti, *How httponly cookies help mitigate xss attacks*, https://clerk.dev/blog/how-httponly-cookies-help-mitigate-xss-attacks, (accessed 03/06/2022).

[48] Node.js, *Worker threads*, https://nodejs.org/api/worker_threads.html, (accessed 03/06/2022).

[49] J. Nielsen, *Usability engineering.* Morgan Kaufmann, 1994.

[50] nngroup, *10 usability heuristics for user interface design*, https://www.nngroup.com/articles/ten-usability-heuristics/, (accessed 20/06/2022).