Politecnico di Torino

POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis
a.y. 2021/2022

Enabling geographic data exploitation for the Cooperative, Connected and Automated Mobility

Supervisors

Prof. Bartolomeo MONTRUCCHIO

Ing. Daniele BREVI

Ing. Edoardo BONETTO

Candidate

Matteo MINOTTI

October 2022

Abstract

Urbanization is imposing radical changes in the planning and management of cities compared to those in which we are used to living. The Smart City concept is based on the IoT and aims to resolve the critical issues of today's cities, identified by six macro areas. Among them, Smart Mobility promotes the connection between vehicles and road infrastructures and relies on cooperation by sharing relevant sensor information. However, a sensor-based approach could not be very robust, as the hardware could fail or the data analysis could provide incorrect information.

This thesis deals with geographic data management (GIS) within Cooperative and Connected Automated Mobility (CCAM). The main idea is to leverage geographic data to support drivers and traffic operators in making smart, safe, and sustainable decisions. This project focuses on creating a software framework called Geographic Map Information Provider (GMIP), capable of providing access to geographic data to be exploited in CCAM contexts, such as onboard units (OBU), roadside units (RSU), and traffic control centers (TCC). The framework provides services and tools for querying and viewing geographic information, acting as an interface to the data made available by map providers. In particular, the adoption of this solution allows the integration of multiple map providers within GMIP, thus enabling a provider-agnostic approach.

The development centralizes on bringing innovation by implementing new additional advanced features that could be helpful to a large number of real-world use cases in the C-ITS field. From the architectural point of view, GMIP comprises three main components: server, client, and storage. The server side embraces a modular design composed of several Docker containers for exposing RESTful microservices to CCAM applications. The comparison of multiple web frameworks has anticipated the implementation of the services to choose the most promising and guarantee a flexible and optimized solution to be integrated into embedded systems. On the client side, GMIP focuses more on providing tools to TCC operators. It comprehends a web application for managing and visualizing ETSI C-ITS messages through an HMI. Moreover, the database component stores both server and client-side configurations. Concluding, a final validation step has verified the framework's advanced features by implementing some significant usage scenarios.

Acknowledgements

"A Manuela e Paolo, per avermi insegnato cosa significano la passione e la dedizione per ciò che si ama e per avermi permesso di inseguire il mio sogno incondizionatamente.

A Gianfranco e Fausta, per aver instillato in me i loro migliori valori e preziosi insegnamenti, anche se il destino ci ha diviso prematuramente.

A Simona e Serena, per aver condiviso con me molti momenti di svago ed esperienze culinarie, e per avermi sempre viziato con dolcetti quando diventavano la meta temporanea delle mie escursioni in bicicletta.

A Daniele ed Edoardo per avermi supportato (e sopportato) in modo esemplare lungo questo cammino, e per avermi motivato nei momenti difficili.

A me, per non aver mai smesso di credere in me stesso."

Table of Contents

List of Tables

List of Figures

List of Algorithms

Acronyms

3GPP
3rd Generation Partnership Project

ADAS
Advanced Driver-Assistance System

AI
Artificial Intelligence

AMQP
Advanced Message Queuing Protocol

API
Application Programming Interface

ATIS
Advanced Traveller Information System

AWGN
Additive White Gaussian Noise

BSSID
Basic Service Set IDentifier

CA
Cooperative Awareness

CAN
Controller Area Network

CCAM

Connected, Cooperative and Automated Mobility

C-ITS

Cooperative-ITS

CLI

Command Line Interface

CP

Cooperative Perception

CPU

Central Processing Unit

C-V2X

Cellular-V2X

DEN

Decentralized Environmental Notification

EMS

Emergency Management System

ESRI

Environmental Systems Research Institute

ETSI

European Telecommunications Standards Institute

GCS

Geographic Coordinate System

GeoJSON

Geographic JSON

GIS

Geographic Information System

GIScience

Geographic Information Science

GIS-T

GIS for Transportation

GL

Graphic Library

GMIP

Geographic Map Information Provider

GPS

Global Positioning System

GNSS

Global Navigation Satellite System

GPU

Graphics Processing Unit

HMI

Human-Machine Interface

HTML

HyperText Markup Language

HTTP

HyperText Transfer Protocol

IEEE

Institute of Electrical and Electronics Engineers

IoT

Internet-of-Things

ISP

Internet Protocol

ISP

Internet Service Provider

ISO

International Organization for Standardization

ISO/TC

International Organization for Standardization/Technical Committee

ITS

Intelligent Transportation System

IVI

Infrastructure Vehicle Information

JSON

JavaScript Object Notation

JVM

Java Virtual Machine

LC

Lane Centering

LDW

Lane Departure Warning

LDM

Local Dynamic Map

LIDAR

Laser Image Detection And Ranging

MAPE

Map Extended

MEC

Multi-access Edge Computing

MFU

Monthly Free Usage

MQTT

Message Queuing Telemetry Transport

MVP

Minimum Viable Product

NASA

National Aeronautics and Space Administration

OBU

On Board Unit

OSM

OpenStreetMap

POI

Point Of Interest

REST

Representational State Transfer

RAM

Random Access Memory

RLT

Road and Lane Topology

RPC

Remote Procedure Calls

RSU

Road Side Unit

SAE

Society of Automotive Engineers

SDK

Software Development Kit

SOAP

Simple Object Access Protocol

SPA

Single Page Application

SPATE
: Signal and Phase Timing Extended

SRS
: Software Requirements Specification

SSE
: Server Sent Event

TC
: Technical Committee

TCC
: Traffic Control Center

TEN-T
: Trans-European Transport Network

TLM
: Traffic Light Maneuver

TSP
: Travelling Salesman Problem

TTM
: Time To Market

UWP
: Universal Windows Platform

UC
: Use Case

UX
: User Experience

V2C
: Vehicle-to-Cloud

V2D
: Vehicle-to-Device

V2I

Vehicle-to-Infrastructure

V2G

Vehicle-to-Grid

V2N

Vehicle-to-Network

V2P

Vehicle-to-Pedestrian

V2V

Vehicle-to-Vehicle

V2X

Vehicle-to-Everything

W3C

World Wide Web Consortium

WLAN

Wireless LAN

WMS

Web Map Service

WMTS

Web Map Tile Service

XML

eXtensible Markup Language

Chapter 1

Introduction

The results achieved by the technological research have always been able to radically change our lives, especially in the modern era, aiming at simplifying people's everyday life habits. These innovations have led to the introduction of new technologies and the continuous improvement of those already used - in every field - in terms of quality, functionality and lately also sustainability. Over the past few years, the **Internet-of-Things (IoT)** has become one of the most important and pervasive projects of the 21st century. This technology can be described as a network of smart physical objects - or «Things» - that are capable of exchanging information. The shared data is predominantly coming from their sensors and it is transmitted over both long and short range communication, i.e., respectively the Internet and Wi-Fi, Bluetooth and Zigbee channels. The concept just introduced is simple and easy to generalize, which is why the engineers were able to apply it in many fields, allowing the birth of new concepts related to the use of smart objects.

Modern society is experiencing a great **urban transition**, namely the migration of people from rural areas to cities. Of all the reasons, the search for new opportunities is the main one. According to a study conducted by the United Nations - Department of Economic and Social Affairs in 2018 [1], more than 55% of the world's population lived in urban settlements that year, with an estimated 5% increase for 2030. To better understand the numbers, immediately after the Second World War, the urban population was around 30%. This phenomenon has caused urban infrastructures to suddenly need changes and optimizations, giving birth to many ideas deeply related to IoT, including the broad concept of **Smart City**.

Smart cities are conceived to solve many of the problems of conventional cities. Their design is aimed at a profound change and improvement in the quality of life compared to the current scenario. To define development strategies it is necessary to introduce six main areas of competence, or indicators, together with their respective sub-categories, as shown in 1.1. Each category will be responsible for analyzing its own critical aspects and promoting any smart solutions. They are presented below.

Briefly, **economic** and **government** models will have to deal with promoting innovation and global connections and providing online administration services for all citizens. The monitoring of public health and urban safety combined with the concept of inclusiveness will be key points in support of **society** and for a better **living**. The **environmental** indicator, one of the most important along with mobility, will provide sustainable solutions for the creation of energy and buildings. Furthermore, it will have to monitor and reduce harmful emissions and waste of any type of resources. In conclusion, **mobility** innovation will have to focus on public infrastructures and on the efficiency and safety of transport networks. The **Smart Mobility** will be the starting point for the development of my thesis.

Figure 1.1: The concept of Smart City

1.1 Connected, Cooperative and Automated Mobility

The main role of smart mobility is to innovate and globally optimize transport networks, be they roads, railways, airports or waterways. To do this, **Intelligent Transportation System (ITS)** have quickly become a substantial means of solving major infrastructure problems. In the current literature, only road transportation systems will be analysed.

We now want to briefly introduce Europe's existing transportation network system to highlight its critical aspects and explain how ITS concepts could be used as intelligent solutions, whether they are already applied or not. The **Trans-European Transport Network (TEN-T)** is very complex, offers the main transport methods, and extends homogeneously throughout the whole territory. The entire infrastructure has been designed both for the transport of goods and for that of people. A significant statistic dates back to 2019 [2], according to which the road carried around 76% of European freight transport. This means that, especially near cities, the traffic efficiency is inevitably affected by this percentage, causing congestion and polluting emissions. The transport of people can be divided into two scenarios, the individual and the collective. As for the individual, the importance of critical issues depends on the scenario. For example, daily car use for home to work commuting is highly inefficient, especially if traveling alone. On the other hand, short-range urban movements carried out by modern electric vehicles (bicycles, scooters, etc.) are more efficient but require the creation of appropriate tracks for their use. Collective transport is by definition more efficient than the previous one. However, optimizing available routes and transit times is essential to maximize this efficiency and avoid transportation with few or no passengers. It is also essential that these vehicles are provided with a sustainable method of locomotion. To summarize, smart mobility's role is to promote **transport safety**, **sustainability**, and **traffic efficiency**, trying to achieve these goals with the contribution of all road «actors».

The full collaboration between vehicles and infrastructures is a fundamental resource for creating a globally shared vision. This result will help provide the ability for drivers and vehicles to make intelligent decisions to best adapt to any situation. That is why, in the past years, the concept of **Connected, Cooperative and Automated Mobility (CCAM)** has been introduced as a suitable solution in the Cooperative-ITS domain. The main idea is to create a standard for interaction between vehicles, infrastructure, cloud networks, pedestrians, and other entities. These standards must be adopted by all applications that want to support the cooperation. Its features are based on a vehicular communication system called Vehicle-to-Everything (V2X), which refers to a vast set of cooperation scenarios.

Below is the list of specific technologies that have been introduced as a standard by 3rd Generation Partnership Project (3GPP) [3], considering some recent trends:

- **Vehicle-to-Infrastructure (V2I)** supports the communication between vehicles and local infrastructures, typically called Road Side Unit (RSU). Usually, it is used to share information about traffic or events.
- **Vehicle-to-Vehicle (V2V)** enables information exchange among vehicles. The CA messages generated by the vehicle's OBU are an application example.
- **Vehicle-to-Cloud (V2C)** relies on an Internet connection to retrieve and save data in cloud-based platforms, such as firmware updates, infotainment settings, etc.
- **Vehicle-to-Pedestrian (V2P)** shares data between vehicles and pedestrians. It has been introduced for safety reasons, e.g., in proximity of pedestrian crossings.
- **Vehicle-to-Device (V2D)** allows communication with smart devices, such as smartphones. Recent applications have seen the introduction of keyless vehicles, the latter being replaced by mobile phone apps.
- **Vehicle-to-Network (V2N)** provides information exchange with data centers, other vehicles, and road facilities, exploiting the mobile network.
- **Vehicle-to-Grid (V2G)** helps smart grids in balancing hybrid and full-electric vehicles' recharge loads.

Over the years, the European Telecommunications Standards Institute TC dealing with ITS has developed many services that infrastructures can use for cooperative applications [4]. Among these, we can mention

- **Cooperative Awareness (CA)**, to support the ITS stations cooperation
- **Decentralized Environmental Notification (DEN)**, to notify about an event occurrence
- **Traffic Light Maneuver (TLM)**, to share traffic safety-related information with Signal and Phase Timing Extended messages
- **Road and Lane Topology (RLT)**, to provide a detailed description of the road with Map Extended messages
- **Infrastructure Vehicle Information (IVI)**, to represent the road signage
- **Cooperative Perception (CP)**, to share environment perception

Figure 1.2 shows the roles of road infrastructures in promoting a cooperative scenario. In the following chapters, we will analyze and discuss the implementation of some use cases related to V2I and V2V.

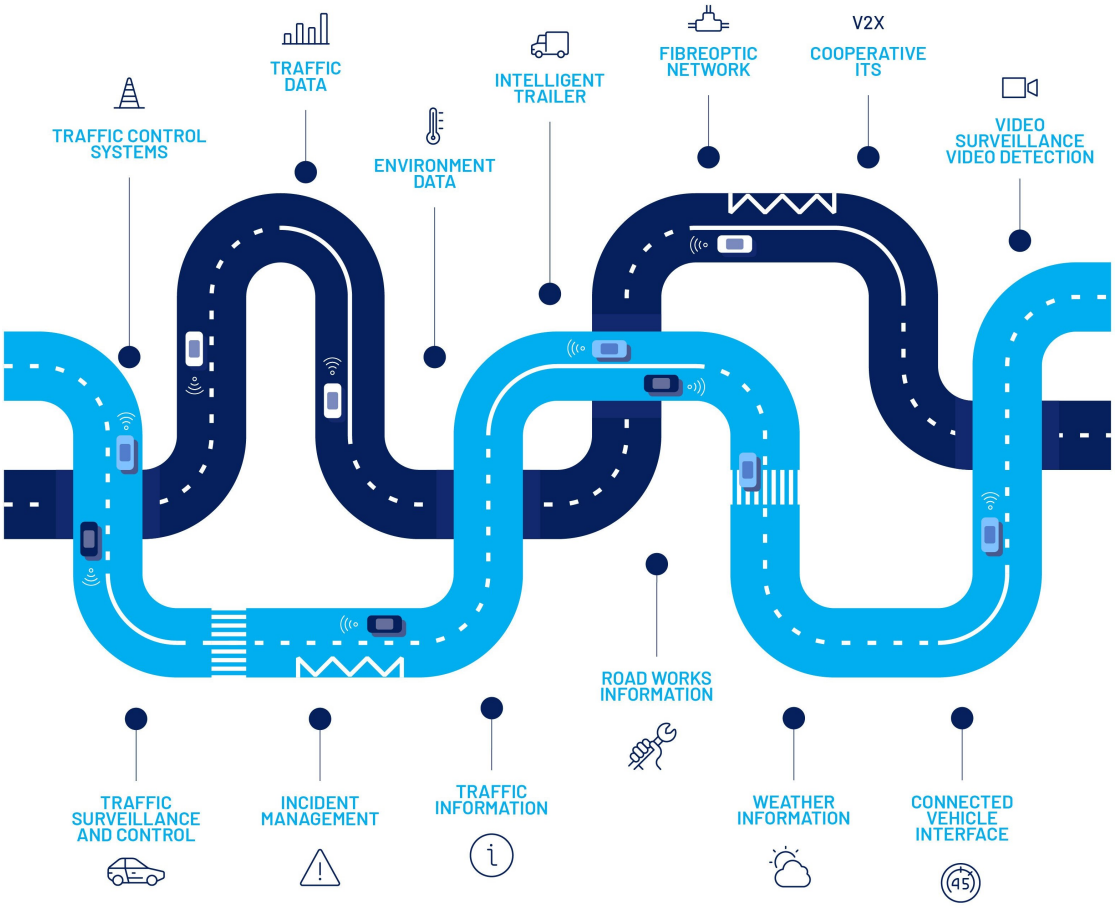


Figure 1.2: Conventional and Digital Infrastructures [5]

1.2 Thesis objective

Digitization has allowed significant innovations in the management of any data. The so-called **Geographic Information Systems (GISs)** are a clear example of this transition. They are specialized databases devised for storing vast amounts of information in the geospatial field. Unlike a few decades ago, today, we can claim to have a world atlas in our pockets. Thanks to these systems and geographic data visualization and analysis tools, we can use geographic data in digital format, even on small devices. According to Thill [6], since the late 1980s, GIS has benefited

research and management of transport networks. This has been possible because transportation systems can be well modeled and optimized by geodata. The exploitation of these data for the improvement of CCAM features will be the subject of this document.

Geographic Map Information Provider (GMIP)

This thesis work will address some problems related to the existing road transport systems by implementing some of the concepts suggested by ITS, exploiting GIS data on both the vehicle and infrastructure sides. The goal of this study is to create a framework that is capable of providing geographic information to CCAM applications. This project can be considered a software framework [7] as it will provide services, reusable components, and tools for the exploitation of GIS in C-ITS scenarios. From now on, we will refer to this system using **Geographic Map Information Provider (GMIP)**.

GMIP will consist of a microservices component and a map visualization tool. The former, called **«GMIP - C-ITS Engine»**, will expose REST APIs designed for processing geographic data. The component will have to guarantee access to geographic data by integrating some map providers and will act as an abstraction layer to facilitate users to retrieve helpful geographic information to support CCAM applications. Furthermore, describing and examining some of the most relevant CCAM use cases will help identify a list of innovative functionalities to be implemented to promote the cooperation of vehicles and road infrastructures by employing geographic data. The latter, called **«GMIP - C-ITS Visualizer»**, will consist of a graphical interface to allow the management and visualization of ETSI C-ITS messages in real time.

The framework will be organized in standalone docker images to give more flexibility from the architectural point of view. Moreover, the code will be divided into modules to allow correct and efficient maintenance. Finally, GMIP will implement procedures related to the manipulation of geographical positioning, road information, route management, and the visualization and management of vehicular messages based on **European Telecommunications Standards Institute (ETSI)** standards.

Below are some real world applications to demonstrate GMIP's value in the field.

Let's imagine we are in a vehicle traveling on the road. The onboard Global Positioning System (GPS) will track your movements. It can happen that, for many reasons, the generated coordinates are influenced by noise, and the larger the noise, the more difficult it will be to associate the vehicle's path with the effectively traveled road. The framework under study will provide a map matching service that also takes into account the history of the vehicle positions, e.g., by applying the Kalman Filter, to improve the overall results.

Another example of the same scenario relates to managing the travel route. To move from point A to point B, set the path on the navigator and start. What happens, however, when events occur along the way? Traditional navigators are not smart enough to report them, while most smartphones' applications can provide only general information about events. An intelligent approach, on the other hand, would be able to redirect traffic to avoid congestion. To do this, GMIP will exploit information coming from C-ITS Decentralized Environmental Notification messages, thus promoting their interaction with GIS systems.

1.3 Thesis outline

The thesis is structured as follows:

- **Chapter 2** introduces the GIS systems, as well as the state-of-the-art of their application in the Intelligent Transportation System field;
- **Chapter 3** defines the map providers and compares the solutions currently on the market, analyzing the services provided;
- **Chapter 4** proposes a list of requirements that should be implemented by applications that exploit geographic data to allow cooperation between vehicles and road infrastructures;
- **Chapter 5** analyzes and compare different solutions for the application from an architectural point of view;
- **Chapter 6** explains the solution adopted and the related implementations, first comparing some frameworks used to structure the application;
- **Chapter 7** discusses the results obtained and the problems encountered during the phases of the work
- **Chapter 8** summarizes the work done and suggests any possible future improvements

Chapter 2

Geographic Data and Information for Cooperative-ITS

Before introducing the analysis and implementation of the GMIP framework, it is necessary to briefly explain how a Geographic Information System works and which modules it is composed of. Details about existing concepts and the **state-of-the-art** of Cooperative-ITS applications exploiting GIS will also be provided.

2.1 Geographic Information System

The study of geographic information is the responsibility of the GIScience discipline. This research area was introduced in the 1990s to develop new concepts for improving GIS systems.

2.1.1 Overview

Geospatial data and information standards have been defined by International Organization for Standardization/Technical Committee (ISO/TC) 211. These sources represent an implicit or explicit binding with a geographic location, i.e., relative to Earth, and they are stored in Geographic Information System databases.

Data

Entity representation includes discrete objects and continuous geographic fields, while events, processes, and masses are typically modeled by analysis. Data structures describe their geometry, i.e., position and shape, and other attributes.

They can be referenced by introducing a key index corresponding to a space-time position. Like the last piece of a puzzle, geographic data is the essential component of the analyzed system. Without them, all analysis and visualization tools would be worthless. For this reason, correctly collecting such information represents the central aspect of managing GIS systems. An impeccable model of reality cannot be obtained, but it is essential to pay attention to the quality of the data to get an excellent approximation. However, compromises must be made to define an acceptable quality depending on the area of use. Below is a list of metrics [8] to take into consideration:

- **accuracy** denotes the degree of similarity between a represented measurement and the actual one
- **precision** indicates how well defined a representation is
- **uncertainty** suggests the presence of errors or inaccuracies in the representations. The propagation of uncertainty could affect the results of spatial analyzes.
- **vagueness** describes how inherently vague the definition of an entity aspect is
- **completeness** indicates whether the entity is described in all its aspects
- **currency** indicates whether the data represents the most recent version of the entity
- **consistency** describes whether the entity representations correctly match each other

Chapters 3 and 7 will highlight some critical issues encountered in the attempt to exploit the data made available by map providers. In particular, the subject of discussion will be the vagueness, completeness, and currency of data.

2.1.2 GIS for Transportation

GIS systems are mainly used to carry out spatial analyzes. This study allows the understanding of spatial objects and events by observing their topological, geometric, and geographic properties. It has applications in many fields of research. From the point of view of transportation, its contribution is mainly oriented to the study of geometric networks.

Transportation planning and **infrastructure and service management** have benefited from geographic visualization tools since their introduction in GIS systems [9]. In the first case, they facilitate accessibility and environmental impact

studies and provide tools for designing balanced multimodal road networks. Regarding the second, they are used to manage and improve the existing infrastructures over the years.

Fleet and logistics control concerns the management of real-time transport infrastructure information. It is the field most associated with Intelligent Transportation System and has dramatically improved the GIS systems optimizing their performance for their usage in real-time applications. This field includes planning the route of public means of transport and private vehicles, thanks to the equipping of navigation systems, where the management of meteorological and traffic data can help to calculate the best route. Emergency management is also facilitated since it is now possible to compute a time-optimal path to the destination.

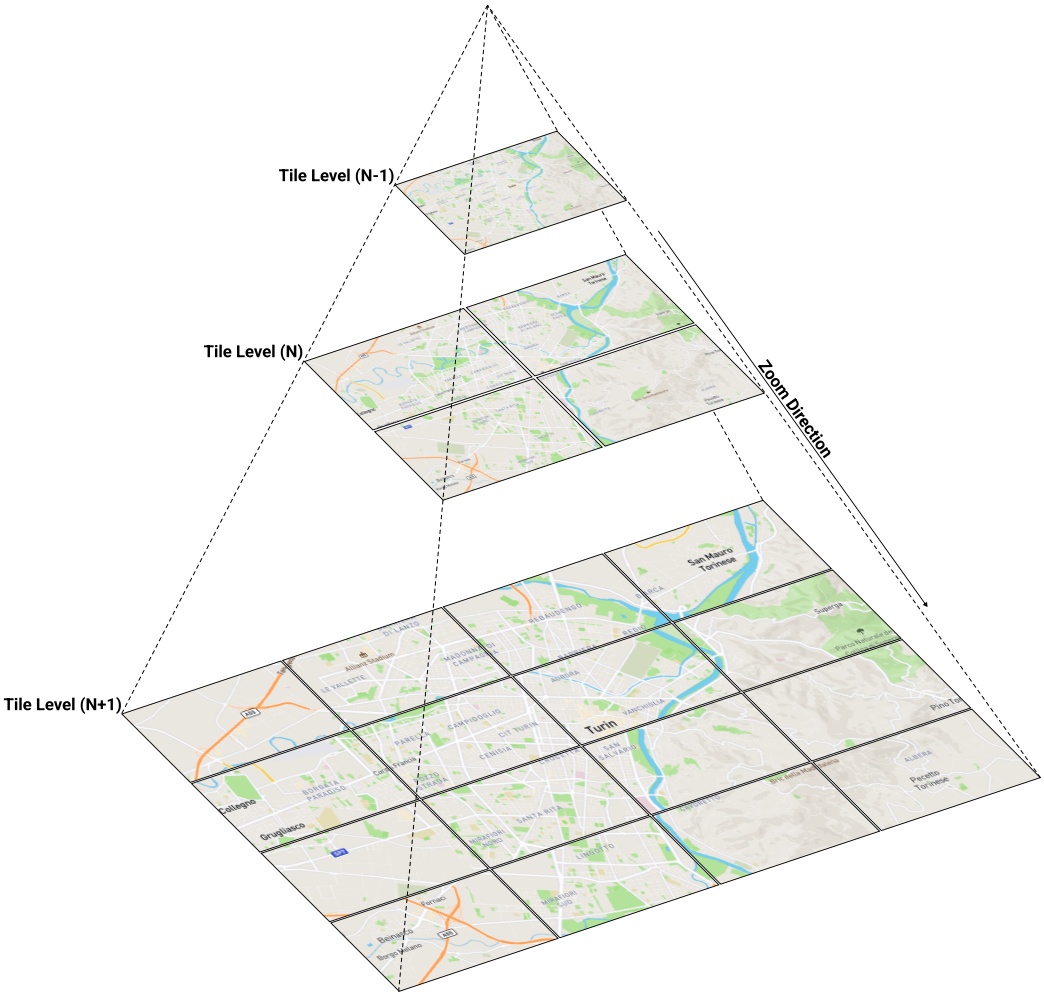


Figure 2.1: Web Map Tile Service concept

2.1.3 Web Map Tile Service

The first versions of the maps available on the Internet were based on technologies that required downloading huge size **pre-calculated images** stored on the server side. The innovation made it possible to introduce new standards, among which Web Map Tile Service (WMTS) stands out. This protocol adopts a «*divide-et-impera*» approach to solve the old methods' problems by splitting the map image into tiles, usually 256 pixels per side, and storing them in a **quadtree** data structure. Figure 2.1 represents the typical pyramid structure of a complete quadtree. Each layer corresponds to a zoom level, as explained in table 2.1. By zooming an area, the parent tile is divided into four smaller tiles until reaching the 23^{rd} level. Generally, this allows for a faster and smoother experience and reduced bandwidth consumption.

Zoom Level	Tiles Number	Tile Width [km]	Example Area
0	1	40041.472	Earth
1	4	20015.087	
2	16	10007.543	
3	64	5003.772	Continent
4	256	2501.886	Large islands
5	1 024	1250.943	
6	4 096	625.471	Large rivers
7	16 384	312.791	Small country
8	65 536	156.340	
9	262 144	78.170	Wide area
10	1 048 576	39.141	Metropolitan area
11	4 194 304	19.570	City
12	16 777 216	9.785	Town
13	67 108 864	4.893	Village
14	268 435 456	2.446	
15	1 073 741 824	1.223	Buildings
16	4 294 967 296	0.556	Street
17	17 179 869 184	0.334	Park
18	68 719 476 736	0.111	Trees
19	274 877 906 944	0.056	Crossings
20	1 099 511 627 776	0.028	Mid-sized building
21	4 398 046 511 104	0.014	
22	17 592 186 044 416	0.007	

Table 2.1: Zoom levels available on a vector tiled map [10]

Each tile is addressed by a value, or spatial index, which can be generated considering the two most used methodologies. The first is based on the concept of **quadkey**, which is nothing more than a one-dimensional index that identifies its position in the quadtree. The alternative, called **geohash**, uses letters and numbers to encode a geographic location based on Z-order curves.

The original idea uses **raster tiles**, simple bitmaps saved as one of the most common image extensions. Recently WMTS has been further improved by introducing the concept of **vector tiles**. This solution associates a mathematical interpretation to each geometric element, thus reducing the space required for data storage. The images are therefore replaced by textual descriptions, e.g., as a GeoJSON-like format, which must be interpreted on the client side, requiring performing devices. Tables 2.2 and 2.3 show the differences between the two approaches [11].

Advantages	Disadvantages
Server-side rendering	Large size
Pre-computed	High bandwidth required
	Slow loading
	Rough motion
	Low resolution

Table 2.2: Advantages and disadvantages of raster tiles

The aim of the raster tiles was also to make the service usable on mobile devices, which at the time had limited performance. However, they need a fast and reliable mobile internet network to be able to download all the pictures. The exponential growth of mobile devices' performances is leveraged to provide faster and smoother service, which is a good improvement from the User Experience (UX) point of view.

Advantages	Disadvantages
Customization	Client-side rendering
Small size	
Low bandwidth required	
Fast loading	
Smooth motion	
High resolution	

Table 2.3: Advantages and disadvantages of vector tiles

2.2 Applications in the ITS field

Subsection 2.1.2 briefly presented the Geographic Information Systems' benefits to the transport field. This section will discuss some of the most innovative results over the past 30 years. In particular, the recent Cooperative-ITS applications will be included in the analysis, as the cooperation of both road users and infrastructures has proved to be fundamental for the effectiveness of smart mobility.

2.2.1 Infrastructures management

The geographical data analysis can improve road infrastructure design and maintenance processes. This subsection presents some of the most exciting results available in the literature.

Anomaly detection on roads

Concerning infrastructure management, the periodic maintenance of roads and bridges is essential to maintain a good level of road safety, thus avoiding traffic congestion. In addition, lately, vehicles are equipped with **Advanced Driver-Assistance System (ADAS)** technologies to assist the driver and promote safer driving. In this regard, the Society of Automotive Engineers (SAE) assigns a driving automation level, among the six available, to each ADAS system based on the complexity of the support provided. Among the most primitive functions, we can find the Lane Departure Warning (LDW) and the Lane Centering (LC). These approaches are highly dependent on the wear conditions of the road lane strips. To anticipate this problem, in «Measurement and Management of the Lane Markings' Stripping Ratio from In-vehicle Camera Image»[12] the authors propose a methodology for measuring and estimating the paint degradation ratio. Their system acquires information on the vehicle position and images of the front sensors from the CAN. First, a phase of re-sampling synchronizes the data and associates each image with its relative position. The GIS system is then used to store and display the characteristics of the roads examined.

Atmospheric events and poor maintenance can cause varying dimensions potholes on the road surface. Reporting these cavities can help the authorities in charge to intervene promptly. Moso et al. [13] propose the detection of these anomalies by analyzing the vehicle's maneuvers described by C-ITS Cooperative Awareness messages generated by the OBUs. Their study uses GIS modules to extract, match and visualize vehicle trajectories using **PostGIS** and **QuantumGIS**. This demonstrates the extreme flexibility and usefulness of GIS tools both in providing solutions and validating results by enabling the visual representation of geographic-related data.

Facilities arrangement analysis

The cooperation between vehicles and road infrastructures foresees that they can exchange helpful information for safety and to avoid traffic congestion. However, the infrastructural equipment, e.g., the RSU, must be positioned in such a way as to provide homogeneous coverage and, at the same time, limit the costs of installation and subsequent management. Storsaeter et al. [14] have implemented an application capable of optimizing this process by minimizing the number of units used and maximizing the area covered by them. The tool, integrated into QuantumGIS, considers both LPWAN communications and 4G mobile networks. It estimates the layout starting from a generic solution provided by the user and some details relating to the antennas and geographic information of the terrain, vegetation, and roads.

By generalizing the study, one can think of optimizing the layout of any building. The sale of full-electric and hydrogen-powered vehicles has forced the construction of new areas dedicated to refueling. In this regard, in 2011, a solution [15] was proposed for the optimal arrangement of hydrogen refueling stations. Considering the extension of a region of the state of Connecticut, it was possible to minimize the driving time of a person to reach the nearest station. The GIS tools helped create the starting model, displaying information about existing filling stations, population density, traffic volumes, and more.

Residential parking monitoring

Cities today are home to up to hundreds of thousands of vehicles. A study conducted by Confused.com in 2022 [16] attributes Italy a ratio of over 650 vehicles per 1000 inhabitants. This inevitably translates into the need to create vehicle parking areas in urban areas, from the periphery to the center. For the correct management of these areas, a group of students from the Brno Technological University proposed a system [17] based on GIS for dynamic parking control. A vehicle equipped with a video camera and a GNSS system is guided along specific routes to take, geolocate and store road images in a GIS system. The solution aims to prevent prolonged stops and thus promote the rotation of occupied parking spaces, favoring residents and tourists.

2.2.2 Transportation enhancement

This subsection describes the approaches adopted in the literature regarding improving and optimizing public and private transport by exploiting geographical data.

Commuters

Every day, millions of students and workers worldwide use means of transport for their home-school or home-work journeys. Adopting public transport is fundamental to passenger transport's sustainability, safety, and efficiency. The availability of reliable public transport increases people's adherence to it, thus decreasing the number of private vehicles on the streets. In this regard, in 1999, Kjellström and Regnér [18] showed that a considerable distance between home and university has negative impacts, albeit small, on the number of university enrollments. This can be exacerbated if public transport is limited or absent. However, public transport can also have downsides. Qoradi et al. [19] analyze a use case in their university to decrease the daily trip time of their campus bus transportation. Their solution uses GIS techniques and GPS sensors to notify students of the upcoming bus arrival. The first phase of data collection involves the anonymous localization of the addresses of the student's homes. The next step uses the geofencing technique, which associates a surrounding area with each address, defined by precise rules. During the journey of the buses, their onboard GPS sensors will be able to trigger a particular action when one of the areas specified above is accessed. For students, the area represents all the places reachable within 5 minutes of travel, starting from their homes. This way, they can be notified when the bus is about to arrive to avoid delays due to prolonged stops. Their solution made it possible to reduce the duration of bus stops, going from 55 to just 14 minutes.

Sustainability

In recent years, awareness of the negative impacts of human activities on the environment has grown enormously. Emissions from internal combustion engines contaminate the air, causing serious environmental damage, especially near large population centers. Compared to the past, technological evolution has made it possible to optimize these engines, generating significantly lower emissions. However, it is still essential to consider aspects such as the altitude profile of road routes. Ojeda et al. have devised a system capable of predicting fuel consumption for heavy vehicles [20]. The work is based on the contextualization of the roads, i.e., the collection, through GIS systems, of data relating to the characteristics of the streets belonging to a specific route. This data is then applied to build a speed profile as accurately as possible. The speed information, together with the physical model of the vehicle, will ultimately provide the fuel consumption estimate.

In 2021, the developers of Google Maps introduced an eco-friendly routing method [21] to accelerate the process of reducing road transport emissions that are harmful to humans and the environment. This feature, made available in Europe a few weeks ago, identifies the route with the lowest fuel consumption to reach the destination. In particular, the artificial intelligence model, trained with real-world

datasets, can predict fuel consumption considering different types of engines and driving conditions.

Autonomous vehicles

The creation of self-driving vehicles includes very demanding research and tests in real-world scenarios. Their control is based on the processing of an enormous amount of data, which can be obtained in real-time from the sensors or retrieved from external databases. The Intelligent Speed Adaptation System, or ISA, is designed to allow the vehicle to adapt its speed according to the signage and the context in which it is located. There are two leading commercial solutions. The first approach is based on the recognition and interpretation of horizontal and vertical road signs, thanks to the cameras installed on the vehicles. The alternative takes advantage of the data provided by the GIS tools. Puthon et al. [22] propose a system capable of exploiting both knowledge to strengthen the results obtained individually. The vehicle is geolocated, and the track is associated with the nearest road, from which the data relating to speed limits are subsequently extracted. This information is then compared with that from the signs. A significant increase in the effectiveness of speed limit determination has been demonstrated, as outdated geographic information can be replaced by signage, and vice versa, failed image detections can rely on stored data.

Speaking of decision-making while driving, GIS systems can prove useful in reducing the lateral positioning error of vehicles, as described in [23], while another essential support concerns lane change [24].

2.2.3 Users cooperation

This subsection discusses the applications of GISs for extracting information that humans can exploit. The long journey towards a future of collaborative and autonomous cars foresees human support in the initial stages. For this reason, in the last decade, there has been much talk about «social nudging» [25] and its effects on people's decisions. For example, Google's eco-friendly routing system leverages nudge theory to make people aware of the efficiency of their road trips.

Advanced Traveller Information System (ATIS)

Since introducing advanced infotainment systems in vehicles and smartphones, satellite navigation functions have ensured great travel support [26]. In general, ATIS systems have been designed to assist travelers and drivers in planning their routes using these electronic devices. One of the first systems dates back to 1997 when Peng [27] presented a solution for elaborating the fastest route between two points, considering the sum of the time spent walking, waiting, using public

transport, and driving. The research was carried out in the following years to improve the service by completing its functions, e.g., calculating the shortest route and a primitive cost function based on the estimated road traffic. In 2011 Zhang et al. [28] proposed a system capable of considering the use of different means of transport along the route. To date, all major map providers offer extremely dynamic navigation solutions, allowing the user to add multiple constraints to the problem.

Further research related to Emergency Management System (EMS) has allowed the computation of the optimal path to reach a specific structure in the surrounding area, for example, a police station or a hospital. In this regard, the geographical analysis of the routes taken by special vehicles made it possible to highlight the critical issues of route planning in highly variable traffic conditions [29]. Ke-jun et al. [30] developed a GIS-based architecture to automatically generate a rescue plan for injured people following a road accident.

Urban traffic monitoring

One of the main objectives of Intelligent Transportation System is undoubtedly aimed at improving the efficiency of city traffic. To do this, existing scenarios need to be deeply analyzed and congestion avoidance strategies implemented. In [31], the authors present a GIS-based solution for real-time traffic data visualization. Government bodies and traffic control systems can then analyze this information for proper infrastructure growth and management.

From the users' point of view, some map providers analyzed in the next chapter offer the integration of dedicated layers to visualize traffic on maps. The most common example concerns Google Maps. The application identifies slowdowns or congestions, whether real-time or history-based events, showing road sections with a different color to highlight their presence. However, this service is relatively primitive, as it does not provide automated cooperation between the actors of the road network but offers advice to users on the «right» decisions to be made. Furthermore, in some areas, it seems not to be as efficient, as reported by Jain and Jain in their paper [32]. They suggest an approach based on analyzing vehicle flows recorded by the cameras of the traffic management centers to overcome reliability problems. Users can set the desired path on a smartphone map application to receive notifications about possible alternative routes when critical events happen.

Driving style analysis

GIS can also increase users' awareness of the sustainable and safe use of their private means of transport. Astarita et al. [33] propose a gamified approach to encourage user participation. Their system comprises users' smartphones and a central server equipped with GIS tools. The goal is to collect information on smartphones'

position and speed, analyze them and define the drivers' driving model. The analysis considers a set of geographic data from GIS, e.g., the estimation of fuel consumption is based on the vehicle velocity and the slope in the neighborhood of its position. Thus, the drivers can be aware of their driving style and receive notifications of any behavior they can improve.

2.2.4 Wrap up

The research has shown how Geographic Information Systems is helpful for the transportation field's innovation. Since their introduction, they have provided tools for **managing road infrastructures**, such as detecting road surface anomalies, arranging road infrastructures, and monitoring city parking. Over the years, the research moved towards **enhancing both public and private means of transport**. For example, adopting Advanced Driver Assistance Systems in vehicles increases human safety by supporting drivers in dangerous conditions. About that, GIS can help in providing more robust controls in lane detection-based approaches. The most recent concepts rely on the **cooperation of users, vehicles and road infrastructures**. The idea is to increase drivers' awareness of their surroundings to help them make intelligent decisions. For instance, primitive applications of traffic-based routing date back to 1997.

Compared to the literature, this project adopts a more flexible approach. In particular, the GMIP framework wants to facilitate vehicles' cooperation by enabling the exploitation of geographic data coming from different map providers and interacting with the ETSI C-ITS messaging service stack. Moreover, most studies resulted in offline environment analysis for optimal results, while GMIP proposes real-time support to the users.

Chapter 3

Maps Providers

The GMIP must leverage geographic data to provide relevant services and information. It is, therefore, necessary to identify which geographic service providers are currently available by exploring and categorizing the geographic services they offer. The framework proposed by this study will have to rely on geographic data provided by GIS systems, potentially worldwide. For this reason, comparing these results can give helpful information for choosing a good set of data providers to integrate into the application, exploiting the peculiarities of each of them and making the system more robust.

3.1 Definition

As discussed in the previous chapter, Geographic Information System (GIS) can store a massive amount of geospatial data. It is no longer a matter of a simple map visualization, but it is now required to overlay objects and events for which geographical information has value. From the transportation network point of view, it could be related to the roads, pedestrian crossings, bus stops, traffic lights, road side units, gas stations, and many other entities. The information can be easily exploited by implementing visualization software and tools to provide online or offline cartography services and customized features. For example, assume to have information about the location and status of traffic lights in a neighborhood. The digital cartography made available by GIS systems would allow their position to be displayed on the map, possibly updating their status in real-time.

All stored data must meet specific quality requirements. They must be truthful, accurate, and as detailed as possible. Above all, they need to be updated periodically. Outdated, incomplete, or inconsistent data would make geographic services unusable, which would not be able to provide a satisfactory result.
To clarify, we can compare the management of such systems to making a pizza.

The map data represents the flour, water, yeast, and seasoning, while the services exploit the ingredients to create geographic knowledge, i.e., make the pizza. It is clear that to cook a good pizza, you need to rely on high-quality ingredients, and the same goes for the services of the maps.

Anyone who can provide third parties accesses to map-related data and algorithms is called a **Map Data Provider**. Companies or foundations such as Google, Microsoft, MapBox, and OpenStreetMap are examples of this.

3.2 Services

Map providers usually do not provide direct access to geographic data but prefer to create services that return an altered version. These algorithms can be of different complexity, from the simple retrieval of the information about a point of interest to the complex routing between two or more points. We now want to introduce a list of the services the considered map providers offer, split into categories reflecting the type of data generated.

3.2.1 Cartography

Cartography involves analysis and design for making and handling maps. The digital version requires the adequate transformation of data collections from GISs into a specific structure. **Web mapping** defines as using maps via the Internet, and several technologies describe its implementation details following different approaches. The WMS is the oldest and is now obsolete. Its output expected the server-side generation of images. Recently, it has been replaced by WMTS, which approach requires the cooperation of both servers and clients to handle tiles.

Static map It is the most basic service in this category and allows to retrieve a portion of the map as an image. The static data is identified by a central location, image size, zoom level, and map layer type. It is often also possible to add markers to highlight important positions. This solution is usually chosen for a simple and optimized user experience.

Dynamic map Cloud-based services are a must when there is the need to visualize dynamic information. The dynamic map service gives you complete control for viewing and managing your data. This solution is perfect for all those scenarios requiring the automatic management of events and the possible geographical analysis of the data obtained, e.g., the real-time visualization of traffic.

Layers customization The geographic data stored in the GIS does not define the representation rules of the elements. For this reason, layers have been introduced to apply a specific style to the map. Generally, the available layers allow satellite view, road networks view, or a hybrid version. Recently, many providers have introduced more in-depth customization regarding the elements and colors displayed.

Street view The possibility of virtually browsing an atlas using the smartphone was undoubtedly a significant technological breakthrough. The street view was a further innovation; it allowed people to «travel» around the world, thanks to the association of 360° images from the real world with geographic information.

3.2.2 Routing

In general, routing defines a line connecting two or more points belonging to a network. In transportation, the line describes the streets that the vehicle will have to travel. The route can be optimized according to different criteria, minimizing time, distance, cost, or other variables. Furthermore, it may have to respect some constraints. Graph theory is used to solve this problem, using Dijkstra's algorithm or its variants.

Directions This service allows you to request more or less detailed information about the routing between a starting position, an arrival position, and any passage points. The solution can be calculated for a specific vehicle type. Moreover, it can take into account a desired departure or arrival time. Some providers allow the exclusion of map areas and road types.

Distance matrix The distance matrix is essential when it is necessary to analyze the routing from multiple sources to multiple destinations. Calculating travel times and distances allows you to quickly solve complex problems, such as determining the fastest route to reach one of the refueling stations in the area.

Map matching Modern GPS devices have significantly improved from the point of view of accuracy. Still, in big cities and especially near the so-called «urban canyons», their precision can dramatically degrade. The map matching allows associating inaccurate tracks to the road route and can be beneficial for visual analysis and the vehicle itself.

Isochrone and isodistance From Greek, iso means equal. These functions allow you to calculate map areas that can be reached within a specific travel time (chrone) or a certain distance. It is beneficial for defining regions subject to events

or areas where a particular service is available, e.g., urban food delivery services. The output data is usually defined as a polygon in GeoJSON format.

Optimization problem Standard directions compute the fastest route, minimizing the traveled distance and defining a well-defined transit order at waypoints, including the destination. Some providers allow the development of time-optimized solutions modeled by the Travelling Salesman Problem (TSP), where the transit order is also a problem variable. This functionality has many applications in transporting freight and delivering products to the end user, as it enables optimizing vehicle movements.

3.2.3 Places

The geographical representation of locations associates a place with a pair of coordinates, latitude, and longitude. These measurements refer to the Geographic Coordinate System (GCS), the most widely used spatial reference system. The latitude, phi for short, represents the angle between the equatorial plane and a generic position. By convention, it can take any value between -90 and +90, respectively, from the south pole (90° S) to the north pole (90° N). Longitude, or lambda, refers to the angle between the prime meridian, Greenwich, and a generic position. In this case, the values are included between -180 and +180, both coinciding with the prime antimeridian. The western hemisphere is identified by the negative values, while the eastern hemisphere references the positive ones.

Geoquery The geoquery service allows you to search for in-depth information on a generic place or nearby business categories. The data can include reviews, opening hours and days, photos, and more.

Geocoding and reverse Geocoding is the ability to provide the geographic coordinates associated with a location based on a descriptive query, be it an address or a point of interest. When you search for an address, the application displays its location by translating the text into coordinates. Conversely, the inverse ability allows you to trace the address given a pair of geographic coordinates.

Geolocation This service allows you to estimate the geographical position of an object. In particular, it implements solutions to track devices connected to the Internet. The primary approach, defined by the W3C organization, attempts to provide the result with a best accuracy-oriented policy based on four methodologies. The policy exploits GPS sensors, mobile networks, wifi networks, and IP addresses.

Timezone Given a location, its time zone can be determined based on tile map technology.

Elevation This service allows you to retrieve information about altitude expressed concerning the local sea level.

3.3 Main choices

3.3.1 Google Maps

Google is an American multinational founded in 1998 that deals with a vast set of technologies, mainly related to the world of information technology. In February 2005, he released the first version of his web mapping platform and called it Google Maps. The application is available in more than 256 countries, supporting approximately 60 languages. It consists of modules written in C++ for the back end and JavaScript for the front end. A proprietary license protects the implementations.

The products made available are usable through HyperText Transfer Protocol API or specific Software Development Kits (SDKs). The APIs are defined according to the REST protocol, while the SDKs are available in different languages and for the leading mobile operating systems, namely Android and iOS. The pricing policy defines an overall cost that scales according to the number of requests received. In addition, a $200 charge-less bonus is available each month. No offline feature is available at the moment.
Below is the list of available services, where prices are referred to a minimum usage of 1000 requests per service.

Cartography

- Static and dynamic map, starting at $2 and $7 per month respectively
- Layers customization, for free
- Street view, starting at $7 per month

Routing

- Directions, starting at $5 per month
- Distance matrix, starting at $5 per month
- Map matching, starting at $10 per month

Places

- Geoquery, starting at $17 per month
- Geocoding, starting at $5 per month
- Geolocation, starting at $5 per month
- Timezone, starting at $5 per month
- Elevation, starting at $5 per month

3.3.2 OpenStreetMap

The OSM project was conceived in 2004 by an English entrepreneur whose primary objective was to propose an open source alternative for managing and visualizing geographic data. A few years later, the OSM Foundation was created to support the project and encourage collaborations. Given the nature of the project and with the help of users worldwide, the application currently supports more than 96 languages. The project mainly concerns data management rather than data visualization, hence the Open Database License.

The implementations are numerous, as well as the supported programming languages. SDK and HTTP API are the main approaches for using the services. The entire platform is made available free of charge. Some services are served by third-party companies, which manage their servers in exchange for money. Alternatively, self-hosting solutions can be employed. A limited set of offline features, such as map rendering and routing, is available at the moment.
Below is the list of available services.

Cartography

- Static and dynamic map
- Street view, by integrating Mapillary

Routing

- Directions, by integrating OSRM, GraphHopper, CartoType, ORS or Valhalla
- Distance matrix
- Map matching
- Isochrone and isodistance
- Optimization problem

Places

- Geoquery, by integrating Nominatim
- Geocoding, by integrating Nominatim
- Geolocation, by integrating Nominatim
- Timezone
- Elevation, by integrating Open-Elevation

3.3.3 MapBox

MapBox is the newest provider. The American company was born in 2010 to provide customizable maps to non-profit organizations. Thanks to the periodic help of some investors, the company has grown a lot. The platform is available in more than 240 countries, but only about 30 languages are currently supported. A proprietary license protects the products designed by the MapBox team, but over the years, they have created and contributed to many open source projects. The geographic data it exploits comes mainly from open-source data such as OSM and National Aeronautics and Space Administration (NASA).

Features can be used on Android, iOS, or web pages. Major modern programming languages are supported. Each product's pricing scales according to the monthly usage, including a free volume tier. Limited offline routing is available.
Below is the list of available services, where prices are referred to a minimum usage of 1000 requests per service.

Cartography

- Static and dynamic map, starting at $1 and $5 per month respectively
- Layers customization, for free
- Street view, by integrating Mapillary

Routing

- Directions, starting at $2 per month
- Distance matrix, starting at $2 per month
- Map matching, starting at $2 per month
- Isochrone, starting at $2 per month
- Optimization problem, starting at $2 per month

Places

- Geoquery, starting at $0.75 per month
- Geocoding, starting at $0.75 per month
- Geolocation, by exploiting browser W3C API
- Timezone, starting at $1.50 per month
- Elevation, starting at $1.50 per month

3.4 Alternatives

3.4.1 Other considered choices

Bing Maps It is part of the Bing suite of the American multinational Microsoft. The project was presented in 2005 and has undergone significant updates. The countries supported are more than 250, while the languages are about half. Thanks to the respective SDKs or HTTP APIs, the platform can be used on Android, iOS, UWP, and web pages. The prices are remarkably high, even though this provider also adopts a pay-as-you-go policy.

Here WeGo Launched in 2006 by Nokia under the name Ovi Maps, it subsequently changed its name following the acquisition by several multinationals. The platform supports more than 30 languages and is available in approximately 180 countries. They offer almost all the services listed in 3.2. Moreover, their products are commercial and protected by a proprietary license. The SDKs and the supported languages are available for development in Android and iOS environments or web platforms. It provides limited offline routing. The usage cost varies according to the incoming requests, and the prices are slightly above average.

3.4.2 Worth a mention

ArcGIS It includes a family of software and services developed by the ESRI company. The first version dated back to 1999 and had only a GIS system. The services offered implement cartography, routing, and geocoding capabilities, with a significant focus on spatial data analysis.

Leaflet It is an open-source library written in JavaScript. It provides web mapping applications for tile-based maps. It was created in 2011, and the MapBox team currently supports it.

MapTiler MapTiler is a project of the Swiss company Klokan Technologies. It mainly offers cloud services for viewing and analysing geographic data and search and geocoding services.

MapLibreGL This open source project was born following the recent transition of MapBoxGL to a proprietary license. It is a Graphic Library (GL) written in JavaScript and provides tools for visualizing tile-based maps.

OpenLayers It is an open-source library for browser-based viewing of tile-based maps. It was created in 2006 and is written in JavaScript.

3.5 Comparison

To make a correct selection of the leading map providers to be integrated into the framework, a comparison was made between those mentioned in 3.3 and 3.4.1. General information and details on the available services have been divided to provide a more explicit analysis.

3.5.1 General information

The following discussion is referred to the data in Table 3.1.

Tyically, proprietary licenses ensure that projects are constantly updated and have a low bug rate. However, open source projects promote the collaboration of the developer community in creating new features and fixing any bugs, as well as allowing them to be used for free. In particular, the OSM community is very active and offers many well-established libraries. Additionally, some companies, such as MapBox and MapTiler, actively support some open source projects. For this reason, the type of license did not affect the selection.
Thinking about a framework that can be used worldwide, the data related to global support showed a lower coverage, albeit with a high number, in Here WeGo.
As for the cost of the services provided, all providers use a pay-as-you-go policy, i.e., pay what you consume. Comparing the Monthly Free Usage (MFU) plans, Bing Maps offers less than the average free usage. Also, together with Here WeGo, they were more expensive considering the individual services.
All providers are compatible with major mobile platforms, namely Android, iOS, and web platforms. Since Bing Maps is a Microsoft product, it is also available for Universal Windows Platforms (UWPs).
Each provider supports Java, Kotlin, Swift, and Objective-C. OSM, being open source, is the most complete and heterogeneous project, providing libraries in

various languages. Bing Maps is the only one to provide .NET support, while Here WeGo only supports the first four mentioned.
The main types of layers are supported by all providers. OSM is the only one that does not yet implement a layer for visualizing traffic in real-time. However, only two of them, Google Maps and MapBox, allow customization.

From this analysis, it can therefore be seen that Google Maps, OSM, and MapBox are the most complete and cheapest platforms among those analyzed. They can be used all over the world and they support major modern programming languages and modern platforms as well. A final detail concerns the platforms for the management of services and consumption. Google Maps and MapBox seem to take care of this aspect more than their competitors. It is allowed to create authentication tokens that can be associated with a set of services. Cost monitoring enables service suspension if a set threshold is reached.

3.5.2 Services

The following discussion is referred to the data in Table 3.2.

Essential mapping services are available to all map providers. On the other hand, the customization of the layers seems to be integrated only by MapBox and Google Maps. The latter is the only one to provide a native street view service, while Bing Maps only shows the most famous points of interest. However, it is possible to include the service in other providers using the Mapillary API. As far as routing is concerned, there are no differences in the list of services offered. However, there are significant differences regarding the implementations, i.e., the number of available parameters. The choices introduced in 3.3 seem to provide more customizable services. The only significant difference between the services of the Places category concerns geolocation. Google Maps and Here WeGo are the only ones to offer a native BSSID-based approach. A possible alternative solution is identified by the W3C API mentioned in 3.2.3, which the main browsers are equipped with. Routing Places

Summarizing, we can again identify the most promising map providers in Google Maps, OSM, and MapBox. Compared to the competition, they offer modern and innovative solutions. They also seem to look after the UX aspect better. They allow the creation of modern and aesthetically accurate HMIs thanks to the detailed customization of the layers. In general, their services allow the exploitation of many parameters, thus offering a better generalization of the API. For example, the routing services of MapBox and OSM enable the provisioning of a list of prohibited areas, that is, areas that cannot be crossed. This allows the generalization of the routing between two points, taking into account possible events.

	Google Maps	OpenStreetMap	MapBox	Bing Maps	Here WeGo
License	Proprietary	Open Db	Proprietary	Proprietary	Proprietary
Coverage	256+	Worldwide	246+	251+	180+
Pricing					
Scaled	•	n.d.	•	•	•
MFU	28k	n.d.	50k	10k	30k
Platforms					
Android	•	•	•	•	•
iOS	•	•	•	•	•
UWP				•	
Web	•	•	•	•	•
Languages					
JavaScript	•	•	•	•	•
Node.js	•	•			
Python	•	•	•		
Go	•	•			
C++		•		•	
.NET				•	
Layers					
Satellite	•	•	•	•	
Street	•	•	•	•	•
Hybrid	•	•	•	•	•
Traffic	•		•	•	•
Terrain	•	•	•	•	•

Table 3.1: Providers general information comparison matrix

	Google Maps	OpenStreetMap	MapBox	Bing Maps	Here WeGo
Cartography					
Static map	•	•	•	•	•
Dynamic map	•	•	•	•	•
Custom layers	•		•		
Street view	•			•	
Routing					
Directions	•	•	•	•	•
Distance matrix	•	•	•	•	•
Map matching	•	•	•	•	•
Isochrone	•	•	•	•	•
Optimization	•	•	•	•	•
Places					
Geoquery	•	•	•	•	•
Geocoding	•	•	•	•	•
Geolocation	•				•
Timezone	•		•	•	•
Elevation	•		•	•	•

Table 3.2: Providers services availability comparison matrix

Chapter 4

GMIP Requirements

In general, the design of a product requires a complete and in-depth analysis of the requirements it will have to meet. This analysis is vital for the initial definition of possible real Connected, Cooperative and Automated Mobility applications. The requirements will be subsequently used to validate the implemented functionalities. The chapter is structured as follows:

- in **Section 4.1** an overview of the requirements and scenarios to which they may refer is presented;
- in **Section 4.2** are introduced the use cases for the On Board Unit scenario;
- in **Section 4.3** the use cases for the Road Side Unit scenario are explained;
- in **Section 4.4** are shown the use cases for the Traffic Control Center scenario;
- in **Section 4.5** the definition of the framework requirements is provided.

4.1 Overview

To begin identifying the requirements that the framework will have to meet, two fundamental questions must be answered. How can geographic information be helpful for the cooperation of vehicles on the road network? What entity could exploit such data to address the critical aspects of existing road transport networks?

4.1.1 Definition

In general, requirements can be interpreted as physical or functional needs that must be met by those who want to implement them. They can be of different types, depending on the level of detail they provide, and they must be verifiable.

The Institute of Electrical and Electronics Engineers (IEEE) in [34] defines the Software Requirements Specification (SRS) as an accurate analytical study of the needs of a system, optionally carried out before implementing its core features, starting from a high-level overview of business key points. This process's primary goal is to reduce unwanted redesign phases by providing a clearer view to the project team. As a further benefit, it helps analyze business indicators, such as the project's cost and risk estimation.

All the requirements presented in the following sections represent an elaborate and detailed version of the business requirements. The latter, in fact, only provide high-level concepts that must be decomposed into user requirements and subsequently into product requirements to detail their needs. Product requirements can be functional or non-functional. The former detail the functions of the product, describing the work done and the user's actions. They are usually characterized by the form « *The system must do ...*». On the contrary, the non-functional ones identify the product's properties, introduce the task's actors, and explain the user experience. They are generally written as « *The system shall be ...*».

Generally, the evaluation process does not end, but it can be iterated several times during the life of the software. This is because the requirements may have been poorly defined, and it is, therefore, necessary to change or eliminate them; in other cases, it is needful to introduce new ones. Figure 4.1 shows the life cycle.

Process analysis is the starting point. From the framework point of view, it included the analysis of services and data offered by map providers. The **requirements elicitation** came before their definition and required answering the first question introduced above. The available data must therefore act as an inspiration for the **definition** step, where functional and non-functional requirements are shaped. Afterward, a preliminary validation is done with the **use cases analysis**. This stage describes a real-life walkthrough by following the defined rules. Note that the last two steps can be reversed, as it might be helpful to build the specifications from the use cases. The **visualization and validation** step performs a further and complete review by introducing software testing techniques. A Quality Assurance team, if any, usually accomplishes it. Each specification is then traced, as well as any of its changes, for monitoring the efficiency of development progress and assessing the impacts of any incoming changes. Finally, the performance of the existing software, i.e., the **baseline**, is monitored to compare the results with the expected behavior and possibly manage the **changes**.

This thesis project has faced the first five steps. It started by analyzing the services and data offered by the map providers in Chapter 3. This study, together with the analysis of the use cases, facilitated the conceiving of more specific needs. Consequently, this has enabled the identification of multiple functional requirements to satisfy the use cases applicable in the scenarios proposed in the next section. Finally, the results obtained were validated in Chapter 7.

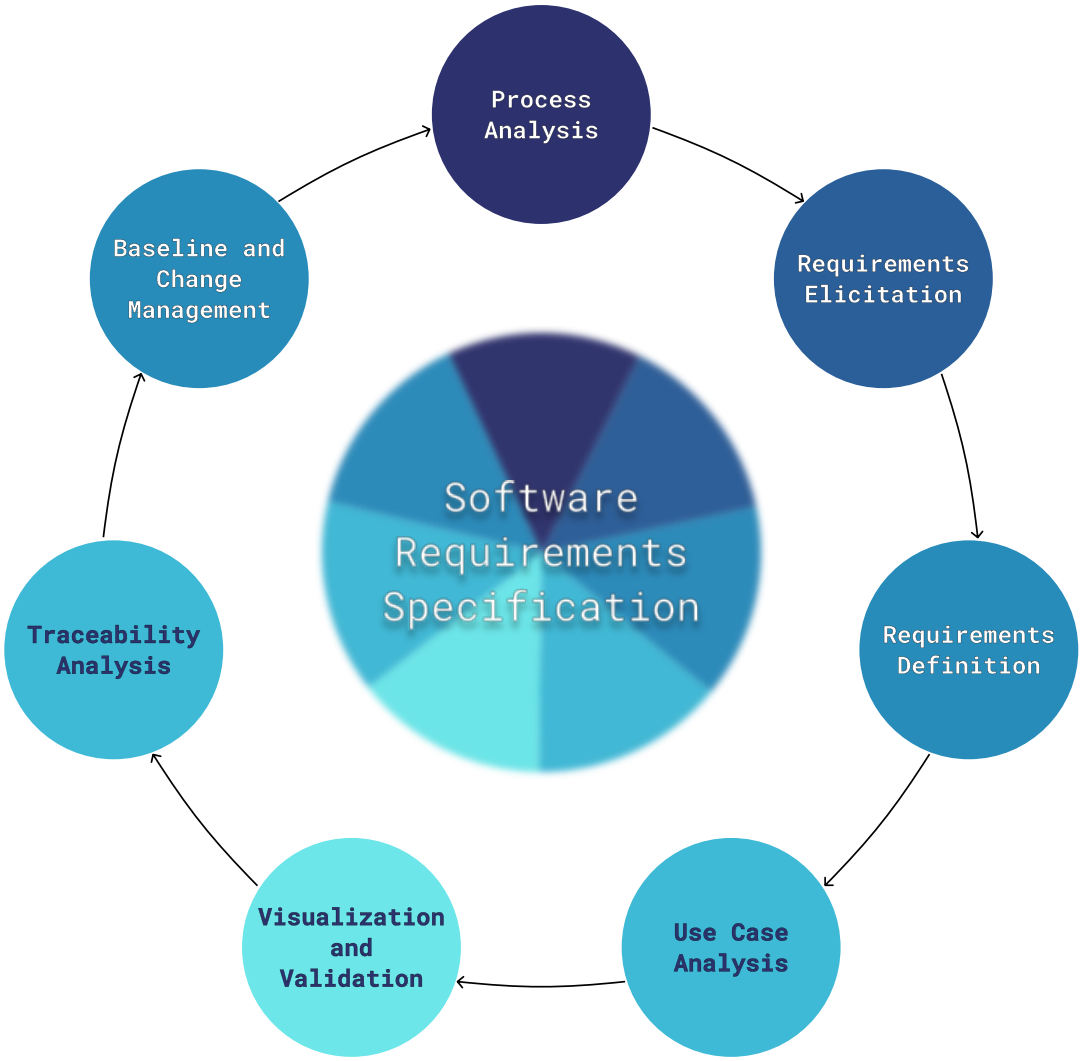


Figure 4.1: IEEE Software Requirements Specification lifecycle [34]

4.1.2 Cooperative-ITS scenarios

Connected, Cooperative and Automated Mobility technology defines **infrastructure**, **roads**, and **vehicles** as the main actors of this innovation. They will be in charge of cooperating by providing and gathering valuable data to and from other entities. For each of the mentioned actors will now be presented the devices used to fulfill the requirements.

On Board Unit (OBU)

The OBU is an embedded system designed primarily to allow bidirectional communication in the C-ITS environment, i.e., the ETSI ITS related standards. This component can interact with devices inside the vehicle, e.g., through the Controller Area Network (CAN). Moreover, it can carry out extra-vehicular transmissions through multi-technology transceivers, offering dedicated short-range and long-range communications. The 802.11p WLAN and C-V2X standards are used in the first case, while the second relies on the mobile network, e.g., 4G, 5G. The OBU implements the ETSI ITS-G5 communication stack, that provides message creation, coding, and filtering capabilities and stores relevant and environment safety-related information in the local data store called Local Dynamic Map (LDM). Finally, it can interact directly with the driver or vehicle through HMIs and ADASs.

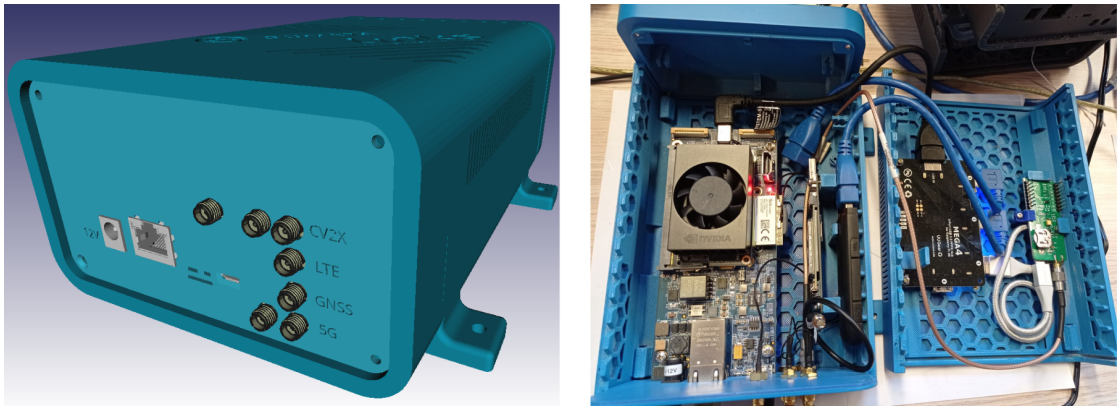


Figure 4.2: The On Board Unit designed by LINKS [35]

Road Side Unit (RSU)

The RSU is a device for enabling infrastructure cooperation and supporting vehicles with limited communication capabilities. It allows sharing helpful information with vehicles, exploiting protocols based on V2I communication. Usually, it has the same set of components described for the OBU. Instead, advanced versions of RSUs can incorporate sensors for sharing additional knowledge. In addition, the integration of GPU enables events' autonomous detection and provides far-edge computing capabilities. The installation of these units requires the analysis of spatial coverage to ensure optimal distribution. About that, GISs are essential to provide a visual analysis of the areas concerned [36]. Recently an attempt [37] has been made to assign the system to the coverage of specific quad keys. In this way, there is a direct correspondence between the two entities, facilitating management and dissemination of V2X communications.

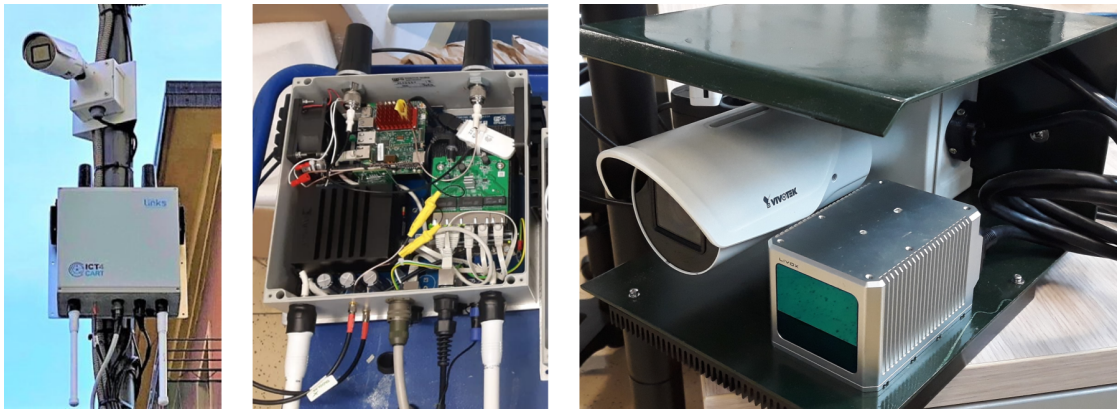


Figure 4.3: The Road Side Unit designed by LINKS [35] for ICT4CART [38]

Traffic Control Center (TCC)

It is a crucial component of intelligent mobility management systems. This entity deals with traffic monitoring and management, thanks to specialized operators' help and automatic controls of urban equipment. Supervision ensures more significant levels of safety and efficiency in road networks. Furthermore, the data collected allows the processing of reports for statistical purposes. This is very important when identifying any criticalities of the local road network. Traffic handling is partly obtained by notifying the events' presence to the roadside units in the appropriate region.

Figure 4.4: Swarco's Traffic Control Center in Romania [39]

4.2 Relevant use cases for OBU

In this section, we want to introduce the most significant use cases regarding the use of geographic data to support cooperation between vehicles.

OBU.01.00	**Event Reporting**
Need	Notification of an unexpected and dangerous event detected by the vehicle
Actors	• Vehicle's OBU • C-ITS communication stack
Required data	• Geographic coordinate • Event information
Storyboard Consider a generic means of transport moving on the road. Suddenly, the vehicle detects a dangerous situation and must notify the nearby vehicles about its occurrence. Thus, it asks the GMIP for geographic data relevant to the event and then forwards them to the appropriate C-ITS stack facility. For instance, GMIP could compute the paths towards a road accident.	

Table 4.1: Event Reporting Use Case Description

OBU.01.01	**Wrong-way Driving Detection**
Need	Notification of a wrong-way driving event detected by the vehicle
Actors	• Vehicle's OBU • C-ITS communication stack
Required data	• Geographic coordinate • Event information
Storyboard Consider a generic means of transport moving on the road. Suddenly, the vehicle detects a wrong-way driving situation and must notify the nearby vehicles about its occurrence. Thus, it asks the GMIP for geographic data relevant to the event and then forwards them to the appropriate C-ITS stack facility.	

Table 4.2: Wrong-way Driving Detection Use Case Description

OBU.02.00	**Message Broker Discovery**
Need	Knowledge of connection information to communication channels for C-ITS services exploitation
Actors	• Vehicle's OBU
Required data	• Geographic (list of) coordinates
Storyboard Consider a generic vehicle traveling along a specific path. In C-ITS, long-range communication channels adopt a pub-sub approach. Thus, the vehicle must subscribe to a message broker to receive the messages. However, each message broker is responsible only for a specific geographic area. The OBU asks GMIP for the list of quadkeys used by the path for retrieving the correlated message brokers.	

Table 4.3: Message Broker Discovery Use Case Description

OBU.03.00	**GNSS Enhancement**
Need	Knowledge of the vehicle's current position projected to the nearest road.
Actors	• Vehicle's OBU
Required data	• Vehicle position (latest OR history) • Eventual signal confidence
Storyboard Consider a generic vehicle equipped with a geolocalization sensor traveling in an urban area. The GNSS signal quality may degrade, e.g., approaching so-called «urban canyons», producing inaccurate measurements. The vehicle position must match with a road to avoid misleading reference points. It asks for the projection of the coordinates to the GMIP.	

Table 4.4: GNSS Enhancement Use Case Description

OBU.04.00	**Event-aware Routing**
Need	Possibility to change the route according to the events occurring along it.
Actors	• Vehicle's OBU • C-ITS communication stack
Required data	• Vehicle position • Destination position • Events information
Storyboard Consider a generic means of transport traveling along a specific road route. Along the way, it could run into slowdowns or even traffic congestion due to the presence of road works, possible accidents, or other reasons. Such events are reported by C-ITS DEN messages. In this case, the vehicle asks the GMIP framework to calculate an alternative route.	

Table 4.5: Event-aware Routing Use Case Description

OBU.05.00	**Vehicle-related POIs**
Need	Knowledge of the list of vehicle-related facilities for a given area.
Actors	• Vehicle's sensors • Vehicle's OBU
Required data	• Vehicle position • POI data
Storyboard Consider a generic means of transport moving on the road. While monitoring its status, the vehicle detects the need to stop at a waypoint, for example, because it is necessary to refuel. The GMIP receives the request to look for the refueling station, mechanic, and other facilities in the surroundings, to prompt the driver for a temporary destination.	

Table 4.6: Vehicle-related POIs Use Case Description

OBU.06.00	**Human-related POIs**
Need	Knowledge of the list of structures, that can be useful to humans, for a given area.
Actors	• Human • Vehicle's infotainment • Vehicle's OBU
Required data	• Vehicle position • POI data
Storyboard Consider a generic road means of transport. The driver could need to reach a specific point of interest, such as a restaurant for taking lunch or a supermarket for shopping. To choose its preferred destination, the human uses an HMI to ask the GMIP framework for shops, restaurants, bars, and other buildings in the surroundings.	

Table 4.7: Human-related POIs Use Case Description

OBU.07.00	**Road Intersection Knowledge**
Need	Knowledge of the characteristics of a given road intersection.
Actors	• Vehicle's OBU • C-ITS communication stack
Required data	• Geographic coordinate • Vehicle information
Storyboard Consider a generic means of transport moving on the road. When approaching an intersection, the vehicle may require some information about its topology, e.g., the allowed directions or how many lanes is composed of. In this case, the vehicle asks to GMIP for relevant information, for example, by exploiting C-ITS MAPE messages.	

Table 4.8: Road Intersection Knowledge Use Case Description

OBU.08.00	**Pedestrian Crossings Knowledge**
Need	Knowledge of the position of pedestrian crossings for a given road or area.
Actors	• Vehicle's OBU • C-ITS communication stack
Required data	• Geographic coordinate
Storyboard Consider a generic means of transport moving on the road. For safety reasons, the vehicle may need some information about the presence of nearby pedestrian crossings. In this case, the vehicle asks to GMIP for relevant information by exploiting maps provider knowledge or, nearby intersections, C-ITS MAPE messages	

Table 4.9: Pedestrian Crossings Knowledge Use Case Description

OBU.09.00	**Cycling Lanes Knowledge**
Need	Knowledge of the presence of any cycling lane and its characteristics for a specific road.
Actors	• Vehicle's OBU • C-ITS communication stack
Required data	• Geographic coordinate • Vehicle information
Storyboard Consider a generic means of transport moving on the road. For safety reasons, the vehicle may need some information about the presence of nearby cycling lanes. In this case, the vehicle asks to GMIP for relevant information by exploiting maps provider knowledge or, nearby intersections, C-ITS MAPE messages.	

Table 4.10: Cycling Lanes Knowledge Use Case Description

OBU.10.00	**Road Lanes Knowledge**
Need	Knowledge of the number and topology of lanes for a specific road.
Actors	• Vehicle's OBU • C-ITS communication stack
Required data	• Geographic coordinate • Vehicle information
Storyboard Consider a generic means of transport moving on the road. The vehicle may require some information about the lanes' composition, e.g., their number, position, and allowed maneuvers. In this case, the vehicle asks to GMIP for relevant information by exploiting maps provider knowledge or, nearby intersections, C-ITS MAPE messages.	

Table 4.11: Road Lanes Knowledge Use Case Description

OBU.11.00	**Road Surface Knowledge**
Need	Knowledge of the road surface type and its conditions along a specific path.
Actors	• Vehicle's OBU
Required data	• Geographic coordinate • Vehicle information
Storyboard Consider a generic means of transport traveling along a specific path. The vehicle may need information about road surface type and quality, e.g., to avoid specific road sections that could be difficult to go through. In this case, the vehicle asks to GMIP for relevant information to exploit.	

Table 4.12: Road Surface Knowledge Use Case Description

OBU.12.00	**Road Signage Knowledge**
Need	Knowledge of the road signage that must be observed by a specific vehicle along a specific path.
Actors	• Vehicle's OBU • C-ITS communication stack
Required data	• Geographic coordinate • Vehicle information
Storyboard Consider a generic means of transport traveling along a specific path. The vehicle may need information about prescriptions, warnings, or directions to follow, e.g., the limitations of the vehicle's speed on the road. This knowledge is provided by map providers and C-ITS IVI messages. In this case, the vehicle asks to GMIP for relevant information.	

Table 4.13: Road Signage Knowledge Use Case Description

OBU.13.00	**Road Vehicle Limits Knowledge**
Need	Knowledge of the types of vehicles that can travel along a specific path.
Actors	• Vehicle's OBU
Required data	• Geographic coordinate • Vehicle information
Storyboard Consider a generic means of transport traveling along a specific path. The vehicle may need some information about the presence of limitations of vehicle categories or characteristics on the road. In this case, the vehicle asks to GMIP for relevant information to be exploited.	

Table 4.14: Road Vehicle Limits Knowledge Use Case Description

4.3 Relevant use cases for RSU

In this section, we want to introduce the most significant use cases regarding geographic details provisioning about events detected by road infrastructures.

RSU.01.00	**Road Event Detection**
Need	Knowledge of the occurrence of both planned and unexpected events in a given area.
Actors	• The RSU • C-ITS communication stack
Required data	• Event position • Event information
Storyboard Consider a road side unit. This infrastructure supports vehicles in their drive and cooperation within its relevant area. For example, it can detect a dangerous event on the road. In this case, the RSU could ask GMIP for relevant geographic characteristics about the event to forward them to the appropriate C-ITS stack facility.	

Table 4.15: Road Event Detection Use Case Description

RSU.02.00	**Vulnerable Road User Detection**
Need	Knowledge of jaywalkers, cyclists and other vulnerable users in a given area.
Actors	• The RSU • C-ITS communication stack
Required data	• Event position • Event information
Storyboard Consider a road side unit. This infrastructure can detect road events, e.g., pedestrians and cyclists on the road. In this case, the RSU could ask GMIP for details about designated areas, such as pedestrian crossings and cycling lanes, to check event safety. Then, it forwards the relevant data, if unsafe, to the appropriate C-ITS stack facility.	

Table 4.16: Vulnerable Road User Detection Use Case Description

4.4 Relevant use cases for TCC

This section will introduce the most significant use cases for crafting, handling, and visualizing geographic information in traffic control centers.

TCC.01.00	**Road Vehicles Monitoring**
Need	Monitoring of vehicles in a given area.
Actors	• The TCC • C-ITS communication stack
Required data	• Area position (bounding box OR quadkey)
Storyboard Consider a generic traffic control center. It will be responsible for monitoring a specific geographic area. By exploiting CA messages, a web map can show a real-time representation of the stations' position and other helpful information, such as their speed profile. Accordingly, the operator can use the GMIP HMI to subscribe to the local communication channel and visualize messages from the C-ITS stack.	

Table 4.17: Road Vehicles Monitoring Use Case Description

TCC.02.00	**Road Events Monitoring**
Need	Monitoring of events in a given area.
Actors	• The TCC • C-ITS communication stack
Required data	• Area position (bounding box OR quadkey)
Storyboard Consider a generic traffic control center. It will be responsible for monitoring a specific geographic area. By exploiting DEN messages, a web map could visualize events' locations and other information, such as their cause and relevance area. Hence, the operator can use the GMIP HMI to subscribe to the local communication channel and visualize messages from the C-ITS stack.	

Table 4.18: Road Events Monitoring Use Case Description

TCC.03.00	**Traffic Lights Monitoring**
Need	Monitoring of traffic lights in a given area.
Actors	• The TCC • C-ITS communication stack
Required data	• Area position (bounding box OR quadkey)
Storyboard Consider a generic traffic control center. It will be responsible for monitoring a specific geographic area. Exploiting SPATE messages could offer the visualization of traffic lights on a web map. Moreover, their real-time status could be visualized. The operator can use the GMIP HMI to subscribe to the local communication channel and visualize messages from the C-ITS stack.	

Table 4.19: Traffic Lights Monitoring Use Case Description

TCC.04.00	**Road Event Creation**
Need	Creation of both planned and unexpected road events.
Actors	• The TCC • C-ITS communication stack
Required data	• Road positions • Events information
Storyboard Consider a generic traffic control center. It will be responsible for monitoring a specific geographic area. The notification about events, such as road works, accidents, traffic jams, and parades, can be shared with vehicles via DEN messages. The operator can use the GMIP HMI to define the paths to the event, describe its characteristics, and request to disseminate the message to the C-ITS stack.	

Table 4.20: Road Event Creation Use Case Description

TCC.05.00	**Road Signs Notification**
Need	Notification of both temporary and permanent signage and status for a given road.
Actors	• The TCC • C-ITS communication stack
Required data	• Road positions • Signage information
Storyboard Consider a generic traffic control center. It will be responsible for monitoring a specific geographic area. The notification of temporary and permanent information about enforced autonomous driving levels, vehicle speed limits, road works layout, etc., can be shared with vehicles via IVI messages. The operator can use the GMIP HMI to create these messages and request their dissemination to the C-ITS stack.	

Table 4.21: Road Signs Notification Use Case Description

TCC.06.00	**Road Topology Notification**
Need	Notification of topology-related information for a given road.
Actors	• The TCC • C-ITS communication stack
Required data	• Road positions • Road characteristics
Storyboard Consider a generic traffic control center. It will be responsible for monitoring a specific geographic area. The description of road intersections can be shared with vehicles via MAPE messages. The operator can use the GMIP HMI to create these messages and request their dissemination to the C-ITS stack.	

Table 4.22: Road Topology Notification Use Case Description

4.5 Framework Requirements Specification

The previous section defined the use cases of the GMIP framework by picturing its possible applications in real Cooperative-ITS scenarios. This approach has facilitated the process of identifying the requirements the framework will have to meet. As a result, four principal contexts of use have been identified based on the geographic information provided. Below is the list.

- **position** concerns the analysis and manipulation of positions information, such as the vehicle's geolocation and the search for points of interest;
- **road** is related to the provisioning of streets topology and characteristics knowledge;
- **route** includes innovative routing solutions, as well as algorithms for the exploitation of information coming from road side units;
- **event** regards the receipt, storing, processing, visualization, creation, and the request for delivery of ETSI C-ITS messages.

Identifier	Category	Name	Use Cases
GMIP.1.1	Position	Map Matching	OBU.03.00
GMIP.1.2	Position	Filter-enhanced Map Matching	OBU.03.00
GMIP.1.3	Position	Vehicle Heading Control	OBU.01.01 OBU.03.00
GMIP.1.4	Position	Points of Interest Retrieval	OBU.05.00 OBU.06.00
GMIP.2.0	Road	Topology Description	OBU.07.00 OBU.08.00 OBU.09.00
GMIP.2.1	Road	Characteristics Description	OBU.10.00 OBU.11.00 OBU.12.00 OBU.13.00 RSU.02.00
GMIP.3.0	Route	Fastest Path Computation	OBU.04.00
GMIP.3.1	Route	Alternative Path Suggestion	OBU.04.00
GMIP.3.2	Route	Path Quadkeys Analysis	OBU.02.00 OBU.04.00

Identifier	Category	Name	Use Cases
GMIP.4.0	Event	C-ITS Event Traces Crafting	OBU.01.00 RSU.01.00 TCC.04.00
GMIP.4.1	Event	C-ITS Message Input Handling	TCC.01.00 TCC.02.00 TCC.03.00
GMIP.4.2	Event	C-ITS Message Info Processing	TCC.01.00 TCC.02.00 TCC.03.00
GMIP.4.3	Event	C-ITS Message Info Visualization	TCC.01.00 TCC.02.00 TCC.03.00
GMIP.4.4	Event	HMI for Geographic Data Crafting	TCC.04.00 TCC.05.00 TCC.06.00
GMIP.4.5	Event	C-ITS Message Creation	TCC.04.00 TCC.05.00 TCC.06.00
GMIP.4.6	Event	C-ITS Message Delivery	TCC.04.00 TCC.05.00 TCC.06.00
GMIP.4.7	Event	C-ITS Message Storage	TCC.04.00 TCC.05.00 TCC.06.00

Table 4.23: Framework Requirements Specification

It is fundamental to underline that the list of both use cases and requirements described in this chapter is not exhaustive, i.e., it does not represent them as a whole. Accordingly, further digging and gathering phases about CCAM scenarios could extend the list. Table 4.23 summarizes all the relevant requirements identified, grouping them by context. Below is an explanation of the methodology used to define them by examining use cases.

Commercial vehicles' GNSS sensors are typically reasonably accurate. However, their accuracy can quickly degrade when approaching urban canyons and highly concentrated urban areas. **Map Matching (GMIP.1.1)** and **Filter-enhanced Map Matching (GMIP.1.2)** attempt to improve the satellite navigation signal measurements, as described in **GNSS Enhancement (OBU.03.00)**, by exploiting ready-to-use map provider features and tailored algorithms. The services offered by map providers identify the need to calculate the nearest road in two distinct

functionalities. Usually, Nearest Road uses a single coordinate, while Map Matching handles a list of coordinates. Precisely, **GMIP.1.1** includes both.

The requirement **Vehicle Heading Control (GMIP.1.3)** wants to address both use cases **Wrong-way Driving Detection (OBU.01.01)** and **OBU.03.00**. The geographic detection of wrong-way driving must rely on accurate GNSS measurements since it is necessary to match the vehicle's position with the nearest road. Similarly, **Points of Interest Retrieval (GMIP.1.4)** focuses on the need to find specific places in the surroundings that could be helpful for vehicles **Vehicle-related POIs (OBU.05.00)** and humans **Human-related POIs (OBU.06.00)**.

The road context will likely picture numerous scenarios where drivers and vehicles can exploit geographic knowledge. Therefore, the use cases from **Road Intersection Knowledge (OBU.07.00)** to **Road Vehicle Limits Knowledge (OBU.13.00)** and **Vulnerable Road User Detection (RSU.02.00)** helped define two different requirements. Topology-related information are provided by **Topology Description (GMIP.2.0)**, whereas **Characteristics Description (GMIP.2.1)** is responsible for gathering general information.

A contemporary use case involving route information is undoubtedly related to the computation of the fastest path from a source to a destination point, taking traffic-related knowledge into account. That will be possible by implementing the requirement **Fastest Path Computation (GMIP.3.0)** in the framework. An improved version, **Event-aware Routing (OBU.04.00)**, would also consider sudden dangerous events on the path by computing an alternative way to avoid the relevant area, as declared by **Alternative Path Suggestion (GMIP.3.1)**. As exemplified in **Message Broker Discovery (OBU.02.00)**, when traveling along a path, it is necessary to subscribe to any local message broker to receive relevant data notifications. Accordingly, requirement **Path Quadkeys Analysis (GMIP.3.2)** involves the calculation of route's quad keys.

In **Road Event Reporting (OBU.01.00)**, **Detection (RSU.01.00)**, and **Creation (TCC.04.00)** it is considered the need to notify the occurrence of a dangerous road event. Creating such messages requires information about the location and cause of the event. Furthermore, since the event will be considered relevant only for nearby vehicles, it is appropriate to highlight the paths, or traces, heading towards it. The **C-ITS Event Traces Crafting (GMIP.4.0)** requirement, therefore, aims to compute these routes automatically.

The monitoring of C-ITS messages, described in **Road Vehicles Monitoring (TCC.01.00)**, **Road Events Monitoring (TCC.02.00)**, and **Traffic Lights Monitoring (TCC.03.00)**, is a complex scenario that requires the implementation of several tasks. First, a message broker subscription is necessary for receiving the data. Moreover, it is mandatory a decoding phase since message transmission includes data encoding due to safety and bandwidth issues. Finally, to represent the message payload's geographic information, it is demanded the implementation

of a visualization tool. For this reason, their examination has conceived **C-ITS Message Input Handling (GMIP.4.1)**, **C-ITS Message Info Processing (GMIP.4.2)** and **C-ITS Message Info Visualization (GMIP.4.3)** requirements. Similarly, creating and managing such messages requires different processes. **HMI for Geographic Data Crafting (GMIP.4.4)** accounts for gathering geographic information, such as indicating a position or drawing a relevant area. At the same time, **C-ITS Message Creation (GMIP.4.5)** and **C-ITS Message Delivery (GMIP.4.6)** take charge of the messages creation and delivery pipeline. Concluding, requirement **C-ITS Message Storage (GMIP.4.7)** implicates the use of databases as storing C-ITS messages might be required, e.g., for inspecting historical data.

Chapter 5

GMIP Architecture

Defining the product's structure is a mandatory step after analyzing the requirements. It is, therefore, necessary to identify the modules responsible for implementing the functionalities and establishing their interoperability at a high level.
«Architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.» [From 40]

Below is a brief introduction of the topics addressed in the chapter:

- **Section 5.1** presents the preliminary analysis accomplished to define the main aspects of the framework;
- **Section 5.2** explains the advanced services architectural solution;
- **Section 5.3** examine the visualization tool architectural solution;
- **Section 5.4** compares and analyzes the advantages and disadvantages of employing different solutions.

5.1 Preliminary analysis

In this analysis, we want to introduce some of the fundamental concepts related to software development that have been considered. These aspects helped resolve some initial doubts about the overall design idea and in understanding the approach to be adopted in the subsequent writing of the implementations. It is essential to underline that in software development, there is no right or wrong architectural solution. Still, it is a matter of how the solution harmonizes with the given requirements, making the most of its components.

5.1.1 Design approach

Standalone component

Software architecture defines a module as standalone when it can operate autonomously and independently from other implementations. This is usually achieved by creating executable programs. With this approach, the component would expose available features directly to the end user. Most of the geographic data that the GMIP framework will manipulate will be provided by the platforms mentioned in 3.3. But, if these platforms were no longer reachable for any reason, the framework could no longer operate independently. Therefore the definition just introduced cannot be fully respected.

Among the advantages offered by this solution, one of the most helpful is being almost ready for use. The rules are simple: install the software, check if it is correctly configured, and, if necessary, make it reachable from the Internet. Another considerable advantage is modularity. Imagine a computer system made up of various modules, including this one. If the component is no longer needed due to a change in requirements, it could be easily removed without causing problems to the entire system. Speaking of software updates, the framework could even be updated remotely without impact on the system. Another advantage is related to the management of the incoming requests load. This component can be «scaled», i.e., cloned into numerous instances to process all requests as quickly as possible. Designing a standalone component also offers advantages in the development stages. Using Docker is one of them. In this way, implementations can be developed regardless of the details of the target operating system, effectively introducing a portable application that can be deployed through docker-enabled machines.

However, selecting an autonomous and independent component can also present disadvantages, based on the software requirements. For the framework, the services will be offered through APIs, so a suitable infrastructure must be adopted to expose them, inevitably adding overhead to code development. Introducing layers for security, monitoring, and other middleware could also increase the code volume.

Advantages	Disadvantages
Ready for use	Infrastructure overhead
Modularity	Middlewares overhead
Scalability	
Upgradeability	
Environment isolation	

Table 5.1: Advantages and disadvantages of a standalone component

Integrable library

The complementary approach is obtained by creating a library that can be integrated into third-party applications that need an interface to the maps provider. In this case, the services offered by the framework would be implemented by classes, functions, and tools made available to developers, almost like a Software Development Kit (SDK).

The first advantage of this solution is obtained indirectly from the nature of the library itself. In fact, the kit can also provide algorithms and tools for low-level data manipulation, in addition to the functionality defined by the requirements. Another potentiality of the library concerns the shareability between applications. A dynamic model would allow you to share code and save memory, thus favoring its use on low-resource systems, such as embedded systems. In general, this solution is more open to the concept of extensibility. An open-source scenario could allow third-party developers to extend the available services and tools. These extensions would allow greater code customization while leaving the main logic intact.

Despite these advantages, the main problem with the library is that it is not ready for use, as it is designed to be inserted into third-party systems. From a commercial point of view, this can slow down the Time To Market (TTM). Moreover, modularity is no longer applicable when exploiting this solution. The library integration prevents it from being easily removed when no longer needed, thus adding unused code to the system. Another critical point is the practical side of the implementations. They will be utterly dependent on the solutions adopted in external systems; therefore, the library will have to be written in different programming languages to meet best the needs of those who will use it. Upgradeability and maintainability are two other closely related negative aspects. In the first case, the update process can be difficult and time-consuming, considering that it cannot be performed remotely. Similarly, fixing bugs can be very challenging for those who create the library and those who integrate it. The former will have to keep all the different versions of the library. After updating the library dependency, the latter must validate their systems carefully.

Advantages	Disadvantages
Shareability	Modularity
Extensibility	Upgradeability
Tools provisioning	Maintainability
	Environment dependency

Table 5.2: Advantages and disadvantages of an integrable library

Choice

The solution adopted for the GMIP framework bases its implementation on a set of standalone components. The primary motivation concerns modularity, a fundamental characteristic of modern and well-designed software. In addition, a ready-to-use component can easily integrate into external systems without introducing a direct dependency on it. This approach also allows the creation of customized systems consisting of the desired components and nothing more. Finally, standalone components facilitate the update stages, which can perform even remotely.

5.1.2 Development approaches

The creation of the GMIP framework arises from providing geographic data to CCAM applications to promote and strengthen cooperation between vehicles and road infrastructures. As anticipated in the previous chapters, GMIP will act as an interface for the map providers. Therefore, it will embrace a **«multi-provider»** strategy to support communication with the different available systems. For this reason, both services and visualization tools will require generalized implementations. In particular, each service will have to define the data needed for the computation and the data generated, i.e., the input and output data structures. Similarly, the visualization tool will have to supply an interface for user interaction within different web maps. The first version of GMIP will support the providers introduced in section 3.3, i.e., Google Maps, OpenStreetMap and MapBox.

Below is a brief explanation of the two criteria considered at the beginning of the development phase. In particular, a brief evaluation led to understanding which approaches to adopt for developing GMIP functionalities.

Features over generalization

The first approach is based on the identification of the map provider who offers more types of services and allows more customization. Once identified, the application can be developed, considering it the only source of access to geographic data. Typically, this solution is not very effective in generalization, as it is not sure that the same services offered by other providers need the same input data and accept the same format. However, keeping the implementations as generalized as possible and considering an accurate analysis of the providers' documentation can help.

Generalization over features

The opposite criterion favors implementing the features individually, focusing on the peculiarities of each provider. For example, identifying the most basic functionality and exploring the differences between map providers allows the creation of simple

but complete data structures. The primary advantage of this solution is undoubtedly the better generalization of data structures. However, from a purely business point of view, this methodology contrasts with the Minimum Viable Product (MVP) concept.

Choice

The development of the framework has adopted a hybrid approach. At first, more emphasis was placed on generalization, implementing the most elementary feature for all providers, and then implementing all the remaining services. This criterion allowed a suitable generalization of the data structures from the beginning.

5.1.3 Framework components

The framework will have to promote the cooperation of vehicles and infrastructures within the Connected, Cooperative and Automated Mobility. First, it is worth mentioning the three primary cooperative players mentioned in X, i.e., OBUs, RSUs, and TCCs. Thus, GMIP will consist of two main components. The first component must expose REST APIs to third-party users, e.g., CCAM applications. In addition, it will be responsible for retrieving and manipulating geographic data to support vehicles and road infrastructure. Instead, the second software will consist of a tool for monitoring V2X communications to traffic control centers. Furthermore, this tool will allow the creation and management of such messages.

5.2 Advanced services for vehicles and facilities

This section wants to introduce the architectural solution used for GMIP's advanced services and explain the reasons behind this choice. This component aims to offer advanced geographic services to CCAM applications. Given that these services will be installed mainly in OBUs and RSUs, it will be necessary to consider any limitations regarding hardware performance and characteristics of the 5G mobile network. Hence, software design will have to promote the efficient use of resources across the board.

Recently, the implementation of web applications has seen a clear shift from **monolithic solutions** to small and scalable **distributed systems** [41]. This trend facilitates code development and maintainability by creating loosely coupled modules. As a result, it enables modularity and system customization, e.g., by allowing the exclusion of features not required from the deployment configuration. Talking about scalability, in *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise* [42], the author introduces the «Scale Cube» for discussing the three scalable axes of an application. Newman

further improves it by adding a fourth dimension. First, of course, **vertical scaling** is not achievable on embedded systems. Moreover, the framework does not require **data partitioning**, which usually applies to complex systems with massive databases. Thus, mixing **horizontal** and **functional scaling** could be a good solution for obtaining a more flexible and lightweight approach for GMIP.

Based on this preliminary analysis, the framework will expose advanced features by implementing APIs through microservices. In general, the type of communication will comply with the client-server model, as GMIP will receive clients' requests and process a response in a synchronous fashion. Data exchange will be carried out by web APIs, exploiting the HTTP protocol. Regarding this, it is necessary to analyze which architectural style should be used for implementing such services.

The introduction of HTTP and the consequent implementation of technologies for data transmission has facilitated the exchange of any information. Since then, SOAP, REST, GraphQL, and RPC have been the most used ones [43].

The **Simple Object Access Protocol** appeared in the late 1990s, proposing the XML format for data exchange. While not currently recommended for use due to its verbosity in representing the messages, the community considers it a legacy protocol, as most of the first financial information systems are currently using it.

In 2000, Fielding unveiled a more flexible solution called **Representational State Transfer** [44]. His goal was to create a resource-driven application and suggest additional formats, such as JSON. Over the years, due to its simplicity and reduced verbosity, this latter approach has become the *de facto* standard for web APIs implementation.

A further improvement emerged in 2015, with **GraphQL** [45] being the first query language to apply a dynamic schema-building concept. Its idea is to use a textual description to define an output object containing only the required data, thus optimizing the bandwidth usage.

The first version of **Remote Procedure Calls** dates back to 1998 when it used XML. New technologies, such as gRPC [46], adopt JSON for efficient and high-performance client interaction with functions installed on a server, e.g., suitable for command-oriented microservices.

Considering that the advanced services will also be available for OBU and RSU, it is necessary to prefer an efficient computation and data transfer technology. To summarize, although SOAP is still widely used, it is an inefficient solution in terms of network bandwidth. RPC is the fastest protocol; however, it is not prone to code reusability and strongly depends on implementation details. REST is currently the most used technology; it is highly versatile and offers the right compromise between RPC speed and SOAP verbosity. GraphQL further optimizes REST bandwidth consumption but requires more time to be comprehended due to its complexity. GMIP will implement RESTful APIs to maintain a flexible approach to changes in map providers' data structures, being careful in designing lightweight messages.

5.2.1 Architecture

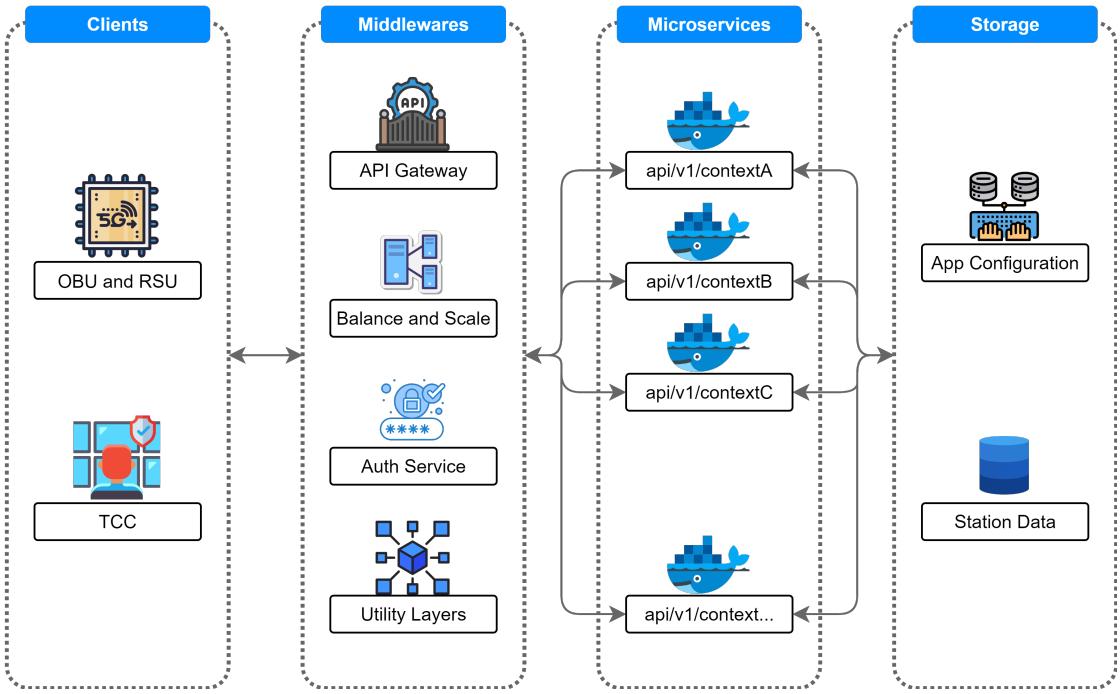


Figure 5.1: Advanced APIs: the microservices-oriented architecture

Figure 5.1 depicts a comprehensive architecture for the framework's advanced APIs component. It comprises four dashed rectangles characterizing the application's interaction layers, where the closest to the end-user is the one named «Clients». Below is a more detailed description of each layer.

Clients The client-server architectural model describes the bidirectional communication between two computer systems, in which the client usually begins the interaction through requests. GMIP's microservices will provide advanced geographic functionality to clients. As already illustrated, there are three major players in the C-ITS area: OBU, RSU, and TCC. Each of them will benefit from these services by acting as a client.

Middlewares Although it is not among the objectives of the thesis work, including a potential layer of middleware utilities in the architecture design can help depict the whole scenario. Since this component adopts a microservices solution, it will need an API gateway to facilitate access to third-party applications. In addition, for safety reasons, an authentication layer should secure services and data from unwanted interactions. Distribution management rules, e.g., network

balancing, application scaling, and so on, are typically applied to such systems to achieve a desired Quality of Service (QoS) requirement. However, further discussing these concepts is not integral to work.

Microservices Microservices are the core of the framework's advanced API services component. They will consist of lightweight web servers, exposing the RESTful APIs for each context defined in section 4.5. Furthermore, they will be responsible for interacting with the supported map providers and exchanging data with the storage layer. These microservices will be embedded in dedicated software containers to facilitate deployment and execution regardless of the target infrastructure. In particular, Docker [47] will be the basis of these packages. In short, this technology allows sharing applications easily and quickly by embedding the code and its dependencies in containers. It exploits the concept of virtualization at the application level to share the host operating system's kernel and run as classic processes. The Docker engine provides access to the host operating system functions to the container. Figure 5.2 illustrates the internal structure of a generic Docker container.

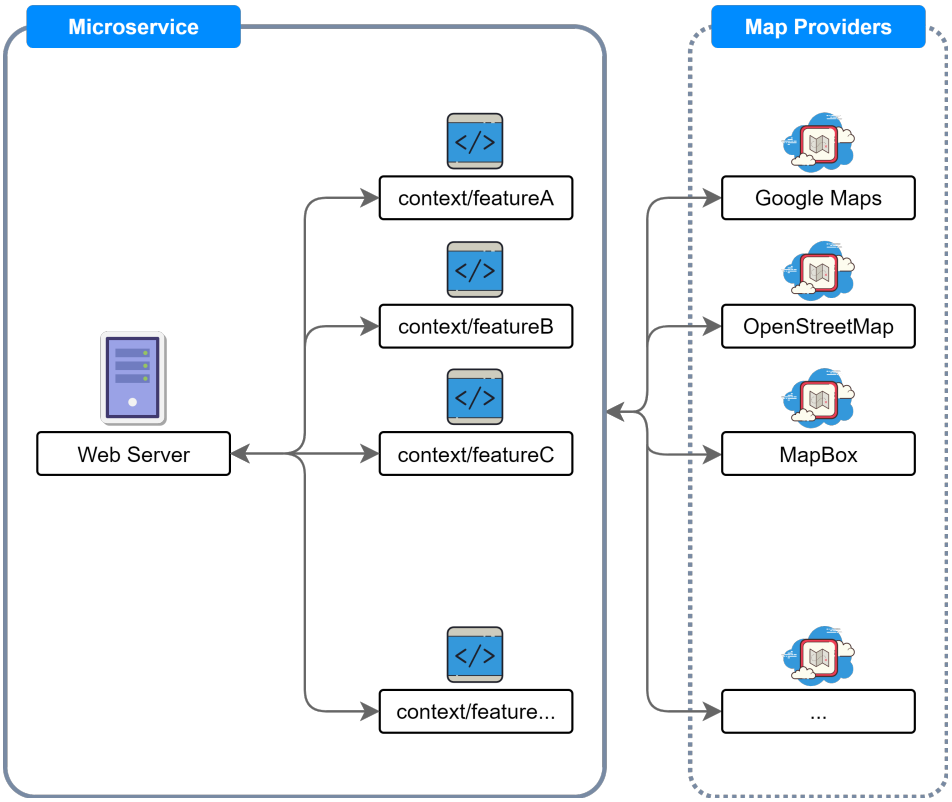


Figure 5.2: Advanced APIs: an overview of a GMIP's microservice structure

Storage A data storage layer is an almost indispensable component in a computer system. However, the solutions considered may differ depending on the type and amount of information. For example, files can store small system configurations, while databases are usually related to the need to store large amounts of sensitive, dynamic, and complex data. Unlike simple files, such systems require ad-hoc query languages for information retrieval by automatically and optimally managing concurrent access to data.

5.3 Visualization tool for C-ITS

This section introduces the architectural solution used for GMIP's visualization tool. This component will consists of an HMI showing geographic information about vehicles and road infrastructures. Furthermore, it will exploit the advanced services component to supply the framework's core features to the Traffic Control Centers. To contextualize, the OBUs and RSUs are responsible for generating messages and requesting their dissemination to the appropriate stack facility, installed on an edge server that implements the Multi-access Edge Computing (MEC) approach. This server has the task of immediately forwarding these messages to other nearby vehicles. Furthermore, it is possible to adopt a data sharing system, e.g., using a cloud-side message broker to forward them from the MEC server to third-party users. The TCC can be considered a third-party user. It monitors the entire road infrastructure in a specific geographical area. GMIP's visualization tool would assist TCC operators in their supervision by enabling the real time display of ETSI C-ITS messages on a map and providing access to advanced services for creating geographic information, e.g., road event traces.

5.3.1 Architecture

From the software architecture point of view, unlike for the advanced services, it is necessary to consider a publisher-subscriber model as it is suitable for managing the asynchronous vehicular message, which in this case are represented by vehicular messages. Figure 5.3 shows the architectural solution proposed for the ETSI C-ITS message display and monitoring tool. The system consists of two main applications on the back and frontend sides, respectively. The first application will take care of receiving messages and forwarding them to the client via WebSockets. The second, i.e., the web application, will have to integrate dynamic map services, e.g., those provided by map providers, for visualizing the messages.

External Sources Before starting the architectural design, it is worth asking which types of external sources will transmit the messages. Considering different

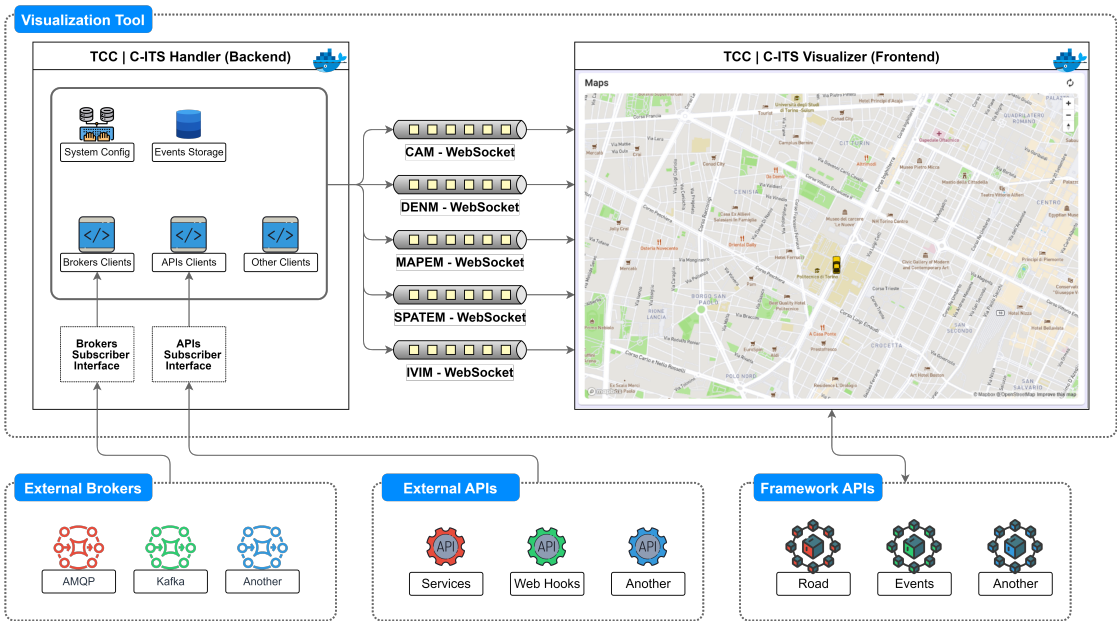


Figure 5.3: Visualization tool: the publisher-subscriber architecture

sources helps depict the whole usage scenario by introducing an abstraction layer for the receiver, to be translated into interfaces code side. As previously exemplified, once the MEC receive the messages, it can forward them through a cloud-side message broker. Another possibility is using specific webhooks and callbacks or web services based on the client-server model. Possible use of the latter concerns the search for old messages stored in the database. However, it is best to avoid using such services as they operate synchronously, which collides with the real-time operation.

Messages Handler This back-end application, i.e., the handler, is called «**GMIP - C-ITS Handler**». Its role is to receive messages and forward them to the different clients connected to it. Therefore, its implementation will have to consider the previously mentioned data sources and adopt a generalized approach, e.g., by developing interfaces, to provide simple but complete management of the receiving channels. It is important to remember that these events require real-time visualization; thus, the handler-clients communications will perform through WebSockets. Its most primitive version will only consist of managing its configuration and messages forwarding. In a more advanced version, it will also be able to provide REST APIs for user authentication, server health check, and others. Furthermore, it will be possible to store the events in an ad-hoc database to make them available for future analysis.

Messages Visualizer This application, i.e., the visualizer, is the most complex module of GMIP's visualization tool and is called **«GMIP - C-ITS Visualizer»**. It must manage large amounts of data, updating and showing information in real-time on a map. Furthermore, unlike the backend solution, this software will potentially be used by different clients. It will therefore have to implement an optimal solution even for the less performing ones. The architectural choice must pay attention to the resources used, optimizing their consumption while maintaining an approach independent of the operating system used. For this reason, a Docker container will host a single-page web application implementing the graphical interface. The interface will allow selecting the desired map provider to ensure global coverage of the service. Information about vehicles, road side units, traffic lights status, and dangerous events will appear on the screen. Furthermore, it will be possible to filter these entities and eventually limit their visualization to run the tool from mobile devices.

5.4 Alternatives

In this section, we want to briefly mention some alternatives to the architectural solutions presented in 5.2.1 and 5.3.1, highlighting the reasons for not considering these approaches for the implementation.

5.4.1 Monolith-based solution for advanced services

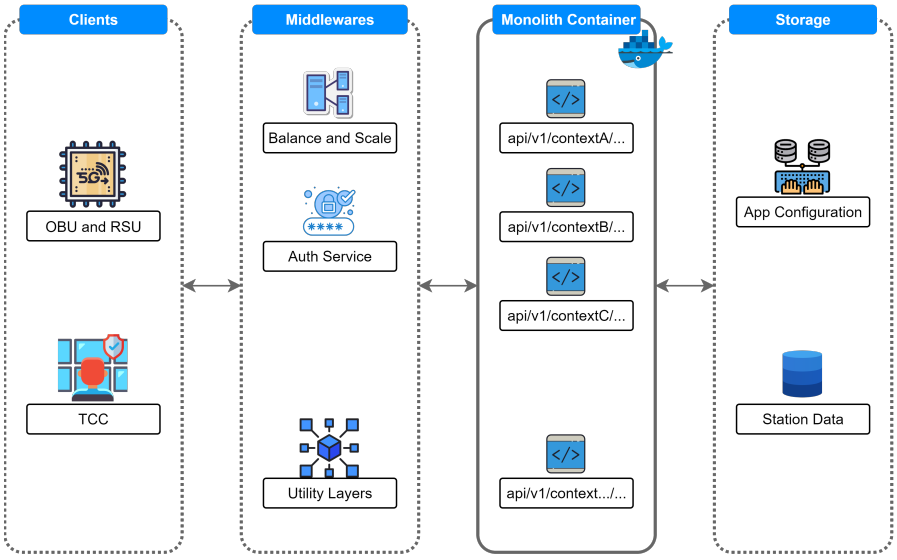


Figure 5.4: Advanced APIs: the monolith-based architecture

Figure 5.4 shows the advanced services model based on a monolithic structure. This approach differs from microservices as a single Docker container has replaced the third layer. Accordingly, the system will consist of a single web server, which will take care of all traffic from third-party applications. As easily guessed, this approach simplifies the management of containers and does not require using an API gateway, but it also has many disadvantages. First, the system can no longer be customized, i.e., choosing the desired modules and excluding the others will no longer be possible. Then, the scaling rules can no longer refer to the single microservice but will have to consider the entire component. Finally, the system will be slower due to a more complex load balancing. This solution also has negative implications from a development point of view. Developers' contribution to a single module can be more complicated and exponentially increase the probability of introducing bugs into the system, especially in large teams with high interactions.

5.4.2 Desktop application for the visualization tool

This architectural solution concerns the component for displaying ETSI C-ITS messages. At first glance, a desktop application can be more professional than a web page. However, this approach has some significant disadvantages that preclude its adoption. A desktop application strongly depends on the operating system in which the developers forge it. The creation of a desktop application strongly couples it with the development operating system. In this case, Docker cannot be used to solve the problem, as it allows interaction via the Command Line Interface (CLI), but this component also requires interaction via HMI. Furthermore, lately, the development of web applications has evolved a lot, thanks also to the affirmation of different frameworks supported by massive communities of developers.

Chapter 6

GMIP Implementation

Before starting the implementation of features, it is good practice to analyze existing technologies to see if the solutions will facilitate the low-level development details. As discussed in the previous chapter, GMIP's advanced services component and part of the vehicle message visualization tool will consist of a web application, so the analysis must consider web frameworks. Furthermore, in the OBU and RSU scenarios, the application will be installed on embedded systems, with access to limited and shared resources, both from a hardware and energy point of view. Therefore, it must optimize its performance by adopting a memory-efficient programming language. Similarly, the visualization tool must pay attention to the resources used to provide a fast, fluid, and interactive User Experience to the end user, regardless of the device being used to access the web page.
Below is a summary of what will be discussed in the chapter:

- in Section 6.1 an overview of the development phase setup is presented, including an extensive analysis of existing web frameworks;
- in Section 6.2 are detailed the advanced services implemented for supporting vehicle and road infrastructures;
- in Section 6.3 some implementation details of the vehicular messages' visualization tool are examined.

6.1 Overview

From a technical point of view, the start of the software development phase is a crucial moment. Thus, it is imperative to choose the technologies to rely on to facilitate the development after outlining the architectural solution. In addition, as already mentioned, the components offered by the framework will run on limited-performance devices and rely on the mobile network for communications.

6.1.1 Fundamentals

This preliminary analysis aims to consider several programming languages to compare the efficiency of the related web development frameworks. In particular, it aims to identify the technology able to optimally manage the available hardware resources to provide fast and reliable advanced services. This methodology will guarantee excellent performance even on devices that must comply with well-defined design constraints, such as the containment of energy consumption. All the C-ITS entities illustrated in 4.1.2 will leverage the GMIP advanced services component. Therefore, unlike Traffic Control Center, OBUs and RSUs will need optimized implementations.

Programming languages and web frameworks

In the modern era, programming languages are evolving quickly, and new ones are continually emerging. However, designing an optimal architectural solution in software development often takes a long time. Thus, building software frameworks can help developers quickly implement basic functionalities and facilitate infrastructure management. As already mentioned in the introduction of the thesis, a software framework consists of a set of executable modules, libraries, services, and tools. Accordingly, it is common for software frameworks to be designed on top of programming languages to take advantage of their peculiarities by exploiting ad-hoc tools and modeling concepts.

Considering web applications, the proliferation of projects adopting this concept has led to the rapid definition of new templates and guidelines for implementing web APIs and SPAs, each with its strengths and weaknesses. In particular, C++ [48], GoLang [49], Java [50], Kotlin [51], Python [52], and Rust [53] are the choices included in the comparison. About that, be aware that the analysis does not consider all the programming languages employed for web development but chooses the modern and most used languages at the moment.
The following paragraphs briefly describe these languages, highlighting the main characteristics of those evaluated in the comparison.

C++ It appeared for the first time in 1985, intending to extend the programming language most used in that period, namely C. It is a compiled language and offers object-oriented, functional, and generic programming paradigms. Its design aims to provide a suitable solution for system programming and embedded software, mainly taking care of computational efficiency and reducing footprint in terms of memory resources used. However, these characteristics and their consequent rapid expansion make it a remarkably universal language today. When it comes to implementing web applications, the different frameworks available have not always proved easy to use and complete in their components.

GoLang It is a relatively recent language, introduced in 2009 by three Google engineers. Go is a compiled language and supports the object-oriented paradigm and concurrent programming. The designers' goal was to devise an easy language capable of increasing productivity while relying on some fundamental characteristics already present in other languages: static typing, readability, and efficient management of multiprocessing. Thanks to its excellent performance, especially demonstrated in the latest versions, it is one of the most used modern languages for web-oriented solutions.

Java It is an object-oriented programming language, born in 1995, and designed to allow its execution on different hardware architectures without requiring a new compilation. In particular, the Java code is compiled in bytecode and interpreted by the Java Virtual Machine (JVM) at runtime. Furthermore, the incredible support of the developers' community has boosted its evolution and adoption of functional, reflective, and concurrent paradigms. Java is excellent at modeling any object using classes and generalizing their behavior. Due to its simplicity, it still is among the world's top 3 most used languages [54].

Kotlin It is a modern high-level programming language and stands for Java's leading alternative. In particular, JetBrains' project founds on Java and aims at improving some of its critical aspects while still needing the JVM. Its main distinctions concern type inference, which allows for writing more concise code, the absence of primitive types, and null safety.

Python It is a programming language designed in the early 1990s. It is a high-level language and supports multiple paradigms, including object-oriented, functional, and reflexive programming. Python's philosophy focuses on readability and adopts a structured and well-defined indentation system. Thanks to its simplicity, over the years, it has often been used by non-programmers for data analysis and task automation. However, like any interpreted language, it allows for rapid functionality development, compromising performance at runtime. Furthermore, frameworks aimed at developing Python web applications are constantly growing.

Rust It is a programming language introduced in the late 2000s, designed to prioritize concurrency and secure memory management. Like Go, it favors structs instead of classes and supports the functional paradigm. Over the years, it has undergone a significant evolution, especially in system programming. In 2022, Linus Torvald announced its introduction in the first versions of the Linux 6 kernel. The focus on memory management and concurrency makes it a great candidate for web server implementation.

The wrk tool

The first component of the GMIP framework will expose REST APIs. Consequently, the application will consist of a set of microservices, i.e., lightweight standalone web servers. A server must handle multiple incoming connections, providing the clients with the requested data as soon as possible. The best way to evaluate its performance requires the analysis of its ability to handle large loads of requests. The «wrk» project is an HTTP benchmarking tool [55] for running personalized test sessions on web servers. Their results comprise metrics describing the system's **response time** and **processing capacity** performance. In particular, the latency considers the average and maximum values, the standard deviation, and the time distribution, while the latter measures the number of requests successfully managed.

Hardware architecture

Performance tests aim to highlight any criticalities of the considered web frameworks regarding memory usage and response time. Thus, a generic environment might be acceptable for their execution. In particular, this thesis work employed the development environment: a **Desktop computer** composed of 8-cores, 16-threads CPU up to 3.7GHz, and 48GB of RAM running at 3000MHz. For a clearer understanding of the results obtained, it is helpful to consider an additional hardware configuration whose performances must be comparable to the actual hardware. In this circumstance, the project chose a **Raspberry Pi 3 Model B+** [56] embedded board equipped with a 4-core 1.4GHz CPU and 1GB of RAM.

6.1.2 Comparison

This subsection presents the results of numerous performance tests executed by considering multiple technological solutions. However, before analyzing the results, it is necessary to discuss some fundamental points.

First, the difficulties encountered in using web frameworks written in C++ led to evaluate some *hybrid* solutions. Specifically, these systems aim at adopting simple, well-designed frameworks while maintaining high performance for core functionality. Hence, they consist of two application layers written in different languages. The core module implements the algorithms in C++, while the web server infrastructure relies on Python, GoLang, or Java web frameworks. Consequently, software tools such as Simplified Wrapper and Interface Generator (SWIG) [57] ensure their interoperability. However, an almost negligible or even negative performance improvement, together with the effort required to implement hybrid solutions, designated the exclusion of these results from the final comparison.

Then, some web frameworks initially written in Java are now available in Kotlin, so the study will consider them as a single solution, selecting the most efficient one.

Finally, each application created to test web frameworks consists of a Docker image to facilitate its development and management. The implementation models a simple REST API that supports the GET method and returns a «text/html» response to the caller with the string «**Hello, World!**». The Raspberry board uses the balena operating system (balenaOS [58]), based on the Yocto Project [59]. In particular, the design of this operating system offers an essential and fast environment to allow the deployment of Docker containers on embedded systems. In addition, it offers tools for quick installation of Docker images remotely.

In summary, this comparison will include some of the most promising open-source web frameworks for each language presented at the beginning of this section.

Web Framework	**Code Language**	**Latency 99^{th} Percent. [ms]**	**Memory Max Heap [MB]**	**Requests per sec**
Gearbox	Go	10.3	9.5	595 724
Fiber	Go	6.7	9.2	542 143
Fiber	*Hybrid*	7.5	10.9	523 943
Gin Gonic	Go	7.3	11.4	279 153
Ktor	Kotlin	12.8	540.0	266 306
Echo	Go	6.4	5.6	258 480
Oat++	C++	7.8	2.4	235 549
Lithium	C++	0.1	4.8	224 863
Javalin	Java	20.5	509.0	221 025
Kratos	Go	9.5	8.7	205 743
Drogon	C++	13.6	1.8	166 640
POCO	C++	0.3	0.3	138 614
Crow	C++	13.0	1.9	137 818
Warp	Rust	4.4	4.1	86 491
Actix	Rust	3.2	5.0	73 313
Spring	Java	49.6	279.0	59 338
Salvo	Rust	8.1	4.1	56 287
Blacksheep	Python	25.0	40.3	49 281
Gotham	Rust	8.2	4.1	48 512
FastAPI	Python	24.6	41.0	39 696
Rocket	Rust	16.2	4.4	23 257
Falcon	Python	37.5	36.6	19 239
Sanic	Python	39.0	40.1	18 466
Quarkus	Java	31.7	152.0	10 411
Flask	Python	1100.0	23.5	3 510

Table 6.1: Desktop - Third-party web frameworks comparison

Web Framework	**Code Language**	**Latency 99^{th} Percent. [ms]**	**Memory Max Heap [MB]**	**Requests per sec**
Fiber	Go	59.6	3.6	11 635
Oat++	C++	304.7	9.4	5 082
Warp	Rust	101.5	2.9	4 017
Blacksheep	Python	1220.0	2.4	335

Table 6.2: Raspberry Pi 3 - Third-party web frameworks comparison

Table 6.1 summarizes the results obtained from the first test sessions, performed on the Desktop computer, while Table 6.2 includes the results obtained using the Raspberry. For the sake of brevity, the second table contains only the results of the best framework for each language. However, in both cases the **wrk** configuration employed 12 threads to open 200 HTTP connections over a 30 second test span.

The first table shows that the web frameworks written in Go are exceptionally performing, as they cover four of the first five positions. Also, Ktor demonstrates decent performance in terms of requests per second. However, its memory consumption is very high, as well as the other solutions in Java. As expected, Rust and C++ are the languages with the best memory management. However, in C++, the absence of well-designed web frameworks compromises its performance concerning the number of requests processed per second. About Rust, it showed acceptable yet disappointing results compared to solutions written in Go.Python proved to be a big disappointment. Despite almost proper memory management, the latency and number of successfully processed requests per second are much worse than mid-table solutions. Go, Java, Kotlin, and Python are languages whose memory management is in charge of a garbage collector. The analysis of the maximum heap memory consumption column shows that all of them, except Go, use much more memory than a highly efficient language like C++. Accordingly, this highlights how, especially in Java and Kotlin, the presence of a garbage collector is deficient when it is necessary to guarantee high performance.

Regarding the tests conducted on the embedded board, there are no particular critical issues, as the order of magnitude of the requests processed by the slower solution is still acceptable, i.e., more than a thousand per second. Furthermore, there is no reference to the Java/Kotlin test as this solution was found memory inefficient and consequently not applicable in an embedded environment.

Go turned out to be the most balanced and performing language. In general, comparing the latency and memory values with Rust and C++ frameworks, there is a higher but still moderate memory consumption and a latency distribution around lower values. For this reason, the component of the advanced services framework will adopt a framework written in Go. However, the chosen framework will be Fiber and not Gearbox, as the latter is nowadays poorly maintained.

6.2 Advanced services for vehicles and facilities

This section will present the advanced services implemented for the 5.2.1, this is one of the two primary components of GMIP. It adopts a modular approach by dividing your application into multiple self-contained modules called microservices. These microservices will exploit the Go Fiber [60] web framework to implement a web server and expose the REST APIs. Fiber proved to be an optimal solution during the testing phase, enabling extreme performance and facilitating feature design. The developer community periodically contributes to the development and correction of the project. Furthermore, its documentation is complete and perfectly explains all the essential information. From the technical point of view, it offers a simple but fast API routing engine. Finally, it supports WebSockets, OpenAPI [61] specifications, and multiple middlewares, such as authentication, logging, and restriction of incoming requests. Concerning the OpenAPI support, each microservice will provide the details of its REST resources, giving access to the HTML page of the definitions.

Before listing GMIP's advanced services, it is necessary to introduce the concept of «polyline». Google designed the Polyline encoding [62] for storing a list of coordinates as a string. The algorithm enables a better representation of the data and reduces its size by losing some information. The two versions of the algorithm, Poly5, and Poly6, can save coordinates with up to 5 and 6 decimal digits, respectively. However, this is an acceptable compromise as the accuracy achieved with five digits is around 1.2 meters. GMIP adopts this coding to optimize the amount of data exchanged and to incorporate the coordinates in the path parameters without incurring errors.

6.2.1 Server side

This subsection delves into the implementation details, describing the features developed. Each microservices mentioned above will be responsible for a specific geographic data category, following the contexts defined in 4.5. In particular, the microservices will implement features for the retrieval and manipulation of position, road, route, and event-related geographic information. The following paragraphs will include some OpenAPI-inspired tables that define each advanced service.

Focusing on a more practical aspect, it is worth remembering that GMIP will support multiple map providers before discussing the advanced services. Therefore it is necessary to provide a more or less innovative methodology for selecting the provider to retrieve geographic data. The first version of the framework will base this choice on the value of a custom HTTP header, leaving the caller with the task of setting it and defining OpenStreetMap as the default provider.

Position API

The framework's advanced services development started by implementing features for exploiting geographic information related to coordinates and locations. In this regard, the Nearest Road service (6.3) allows the projection of a generic position on the nearest road. As already discussed in 4.5, this can help provide the vehicle with correct information about the road traveled. However, the procedure of this elemental feature enforces the services provided by the supported map providers. It requires a single input coordinate with polyline encoding to project it onto the nearest road if any.

API.01.01	**Nearest Road**
GET position/*:pp*/road/nearest	
Request Parameters	• pp → position polyline
Request Body	none
Response	• nearest road coordinate

Table 6.3: Nearest Road API Specification

Another useful feature supplied by map providers is the ability to match the last N positions of the vehicle with the nearest road, i.e., Map Matching. However, note that each map provider limits the number of locations to $N = 100$, which is quite limiting considering the frequency of generation of ETSI C-ITS CA messages.

GMIP makes available the Map Matching service (6.4) for this purpose. In addition, the framework proposes an enhanced version with the Filtered Map Matching service (6.5). It aims to improve the results of the former with an ad-hoc implementation while still exploiting the features made available by map providers. In particular, the filter-based version wants to mitigate any significantly noisy GNSS measurement by implementing a Kalman filter [63] to estimate the vehicle's path. Thus, the map matching input data will consist of filter estimations.

API.01.02	**Map Matching**
GET position/*:pp*/road/matching	
Request Parameters	• pp → path polyline
Request Body	none
Response	• matched road coordinates

Table 6.4: Map Matching API Specification

By way of example, the workflow consists of the following steps. The vehicle starts moving. The first GNSS measurement initializes the filter. Then, for each position update, the service updates the filter status and returns the estimated measure. For the filter to be effective, the service must save its state, i.e., the Kalman gain and the estimation matrices, and load it on the next iteration.

API.01.03	**Filtered Map Matching**
POST position/*:pp*/road/matching/filtered	
Request Parameters	• pp → path polyline
Request Body	• coordinates confidence list
Response	• matched road coordinates

Table 6.5: Filtered Map Matching API Specification

Regarding road safety, an essential task requires detecting hazardous events. By proposing Wrong Way Driving (6.6), GMIP wants to provide a tool for detecting a specific improper driving events category in urban environments and highways.

API.01.04	**Vehicle Heading Control**
POST position/*:pp*/road/heading	
Request Parameters	• pp → position polyline
Request Body	• vehicle's heading • vehicle's heading confidence
Response	• vehicle and road headings comparison

Table 6.6: Vehicle Heading Control API Specification

As illustrated in algorithm 6.1, the advanced service takes advantage of a map provider for retrieving the topology of the road traveled by the vehicle. However, at the moment, the only one to provide this type of data is OpenStreetMap. To clarify, OSM defines a road as an ordered set of points. By convention, for one-way streets, this order corresponds to the direction of travel. The algorithm embraces the following consideration: the evaluation of the road's heading must explore the surrounding of the vehicle. In particular, since the former can have curved sections, its direction can drastically vary along the path. Moreover, this methodology assumes that the vehicle's position belongs on the road. Its procedure composes of the steps described below. Initially, interaction with OSM retrieves the road's data. From that, the algorithm computes the road's node nearest to the vehicle. It then

Algorithm 6.1: Vehicle's Heading Control Algorithm

```
Input  : vehicle's position, vehicle's heading, vehicle's heading confidence
Output : whether the vehicle moves in the right direction or not
Procedure CheckVehicleHeading()
    initialize variables
    snap vehicle's position to the nearest road
    retrieve road's information from map provider
    for each node of the road do
        compute node's distance from vehicle if node's distance is lower than
         the current minimum then
            save node's distance to the current minimum distance
        end
    end
    retrieve the road's node following the one at minimal distance
    compute road's heading in the surrounding of the vehicle
    compute road's heading range considering vehicle's heading confidence
    if road's heading range includes vehicle's heading then
        OK! The vehicle moves in the right direction
    else
        KO! WRONG WAY DETECTED!
    end
```

calculates the road's direction by retrieving the next node. Finally, it compares the vehicle's heading with the road's heading range. If excluded, it notifies the occurrence of a wrong way driving event.

Finally, GMIP implements the Points of Interest service (6.7) for querying geographic locations by exploiting map providers. This feature wants to provide a search tool for finding specific locations within a certain distance radius.

API.01.05	**Points of Interest**
GET position/poi/*:lt*/*:pp*/*:dr*	
Request Parameters	• lt → location type • pp → position polyline • dr → distance radius
Request Body	none
Response	• surrounding POIs coordinates

Table 6.7: Points of Interest API Specification

Road API

This microservice deals with the querying of information relating to the road network. The technologies introduced by the ADAS systems make it possible to increase the vehicle's awareness of the surrounding environment. GMIP wants to propose a new methodology by exploiting the geographical description of the roads to help the vehicle. In this way, vehicles can rely on a different source of data, thus strengthening the information gathered by its sensors.

The Road Topology 6.8 advanced service leverages geographic information from map providers and Map Extended messages to describe road topology.

API.02.01	**Road Topology**
GET road/*:pp*/topology	
Request Parameters	• pp → position polyline
Request Body	none
Response	• road structure information

Table 6.8: Road Topology API Specification

Alternatively, the Road Information 6.9 resource allows for the retrieval of precise information on the characteristics of the road, e.g., the number of lanes, the speed limit, and other data.

API.02.02	**Road Information**
GET road/*:pp*/info/*:rc*	
Request Parameters	• pp → position polyline • rc → road characteristic
Request Body	none
Response	• road characteristic information

Table 6.9: Road Information API Specification

Route API

Travel route calculation is a crucial context in which the manipulation of geographic data can produce beneficial information for promoting vehicle cooperation. This paragraph illustrates the advanced routing services proposed by the framework to help Connected, Cooperative and Automated Mobility applications.

First, since a route composes of source and destination points, each resource will require this information. In particular, the path's *rp* parameter, i.e., the route's polyline, embeds these coordinates into a string. In addition, the response from each routing service will include a polyline-encoded route description, route distance in meters, and route duration in seconds.

The Fastest Route service (6.10) is the simplest of the features set. It uses map providers to find the fastest route to reach the destination. Moreover, it allows deciding whether to include traffic information in the computation or not.

API.03.01	**Fastest Route**
GET route/*:rp*/fastest/*:tf*	
Request Parameters	• rp → route's polyline • tf → traffic flag
Request Body	none
Response	• route's polyline • route distance • route duration

Table 6.10: Fastest Route API Specification

Enforcing a different solution, the Alternative Routes advanced service (6.11) boosts the vehicle's awareness by enabling event-aware routing. It computes an alternative route by excluding the locations of dangerous events from the path computation. OSM and MapBox allow embedding this information as their service input data. In particular, the service assumes that the caller knows the relevant dangerous events and defines a path parameter, i.e., the hazardous events' polyline, for communicating their coordinates using polyline coding.

API.03.02	**Alternative Routes**
GET route/*:rp*/alternative/events/exclude/*:ep*	
Request Parameters	• rp: route's polyline • ep: hazardous events' polyline
Request Body	none
Response	• route's polyline • route distance • route duration

Table 6.11: Alternative Routes API Specification

By thinking of more environmentally friendly solutions, vehicle routing can play a fundamental role in promoting more sustainable choices.

The first advanced service is Traffic Jam Prevention Route (6.12). It aims to prevent the occurrence of road congestion and limit the passage of vehicles in the most polluted areas. For example, the roads close to schools and universities are usually heavily trafficked during the early morning hours. Therefore, the computation of an alternative route would make those areas safer and drastically reduce pollution.

API.03.03	**Traffic Jam Prevention Route**
GET route/*:rp*/alternative/traffic	
Request Parameters	• rp → route's polyline
Request Body	none
Response	• route polyline • route distance • route duration

Table 6.12: Traffic Jam Prevention Route API Specification

The Sustainability Awareness Route advanced service (6.13) implements another solution. The eco-friendly routing mode in Google Maps [21] aims to raise people's awareness of environmental issues. However, their model is based solely on the proactive collaboration of people. Instead, this service aims to provide an alternative destination by encouraging the user. In particular, the feature exploits the concept of «Social Nudging», already discussed in 4.5. For example, suppose the driver wants to go to a restaurant very far away. In this case, the algorithm tries to suggest a similar but closer alternative, providing a discount coupon as an incentive.

API.03.04	**Sustainability Awareness Route**
GET route/*:rp*/sustainable/poi/*:lt*	
Request Parameters	• rp → route's polyline • lt → location type
Request Body	none
Response	• route polyline • route distance • route duration

Table 6.13: Sustainability Awareness Route API Specification

Compared to the other services discussed, the Path Quadkeys advanced service (6.14) does not calculate road routes. Instead, it aims to calculate the list of tiles at a specific level representing a given route. Specifically, the feature must treat the path from source to destination as a list of tiled boxes rather than a set of lines. Referring to 4.1.2, OBU devices support a long-range communication technology. As already discussed, this strategy adopts a cloud-based *pub-sub* communication channels, e.g., exploiting message brokers. Therefore, in the geographical context, each publisher is responsible for disseminating messages in a specific area corresponding to a tile. With this premise, it is possible to understand why this service can help find the information necessary to connect to local communication channels to receive notifications of relevant dangerous events.

API.03.05	**Path Quadkeys**
GET route/*:rp*/quadkeys/*:tl*	
Request Parameters	• rp → route's polyline • tl → tile's level
Request Body	none
Response	• ordered list of quadkeys

Table 6.14: Path Quadkeys API Specification

The algorithm 6.2 describes the procedure with pseudocode. Below is an explanation. First, after the initialization of the variables at row 2, the algorithm must exploit the input tile level to retrieve the tile's detailed information. Then, it computes the maximum accepted distance between the coordinates, as stated in row 4. This strategy ensures the inclusion of each tile into the output list by thickening the path's coordinates. By recalling table 2.1, each tile represents a square geographic area, which dimension depends on its level. For example, tile level 18 determines a tile width of about $111m$. Suppose the distance between two consecutive coordinates is greater than a specific tile's diagonal. In that case, the algorithm would exclude the tile as it does not include coordinates. Row 5's responsibility is to decode the input polyline into an array of geographic coordinates. Then, the procedure checks if the maximum distance is compliant for each following coordinate. In particular, that loop starts by considering the second coordinate as the current and the first as the previous. The algorithm thickens the segment between the two coordinates if the distance is not compliant. In rows 10 and 11, it first computes the number of points it must generate and then applies a geodetic interpolation. After that, the procedure exploits each new coordinate to compute the related quadkey, i.e., the tile's index, and store it in a map. The outcome of this algorithm is the ordered list of quadkeys related to the input path.

Algorithm 6.2: Path Quadkeys Computation Algorithm

```
Input  : path's polyline, tile level
Output : ordered list of quadkeys
Procedure ComputePathQuadkeys()
    initialize variables
    retrieve tiles information from tile level
    compute coordinates' max distance as half of the tile's diagonal
    decode path's polyline to an array of geographic coordinates
    assign the first coordinate to the previous coordinate variable
    for each coordinate of the path do
        compute distance between the current and the previous
        if distance is greater than max distance allowed then
            compute the number of coordinates to add
            compute the new coordinates by geo-interpolation
            for each new coordinate do
                compute coordinate's quadkey and save it in a map
            end
        end
        compute coordinate's quadkey and save it in a map
        replace the previous coordinate with the current
    end
```

Event API

Promoting safety in the CCAM environment is vital. Chapter 4 discusses multiple use cases related to detecting dangerous events. In particular, the ETSI C-ITS Decentralized Environmental Notification messages notify of the occurrence of such events to the vehicles, providing details about their cause, location, and area of relevance.

API.04.01	**Event Traces Creation**
GET event/*:ep*/traces/*:tl*	
Request Parameters	• ep → event polyline • tl → traces length
Request Body	none
Response	• list of traces towards the event

Table 6.15: Event Traces Creation API Specification

Regarding that, GMIP wants to support the ETSI C-ITS communication stack in sharing event's geographic information. The "Event Traces Creation" advanced service (6.15) proposes a methodology to calculate the traces towards the event. The primitive version of the algorithm involved the creation of a circle whose radius corresponded to the maximum relevant distance. However, this approach frequently generated too long paths incompatible due to the maximum distance constraint. Furthermore, its effectiveness heavily depended on the topology of the road network in the event's proximity, producing unsatisfactory results in most cases.

Algorithm 6.3: Event Traces Generation Algorithm

```
   Input  : event's position, event's maximum relevant distance
   Output : list of event's traces
 1 Procedure GenerateEventTraces()
 2     initialize variables
 3     retrieve the isodistance polygon from map provider
 4     thicken the coordinates of the polygon edges
 5     ask for map matching to map provider
 6     for each coordinate of the polygon's perimeter do
 7         retrieve the information of the street to which it belongs
 8         add the street to a map
 9     end
10     retrieve the distance matrix from map provider
11     calculate the distance threshold as a factor of the maximum distance
12     for each route of the distance matrix do
13         if route distance greater than the distance threshold then
14             add the candidate route to the streets map
15         else
16             discard route
17         end
18     end
19     for each street of the streets map do
20         sort its candidates by descending distance
21         take the longest candidate route
22     end
23     for each candidate do
24         check if the route is contained in a longer one
25         if is contained then
26             discard candidate
27         end
28     end
```

Thus, this feature relies on information supplied by map providers. In particular, the framework exploits isodistance services. An isodistance polygon consists of a shape whose edge points are equally distant from a specific point. For example, by applying this concept to a road network, this polygon represents the feasible paths at equal distances from a specific point, e.g., an hazardous road event. The pseudocode in 6.3 describes the algorithm steps. First, the procedure asks the map provider to compute the polygon equidistant from the event. In line 4, it checks that the distance between the points that define the perimeter of the polygon is sufficiently small; otherwise, it proceeds with the generation of new points. Then, it asks the map provider to match these points to the nearest road. The cycle at line 6 retrieves the information of the road coordinates. Next, it adds the coordinates to a map data structure by exploiting the road information to build the map's key. Subsequently, it asks the map provider to calculate the distance matrix, defining the generated coordinates as sources and the event as the destination. The circle in line 12 reads the distance matrix and filters the road paths. In particular, if the route length is greater than a percentage of the maximum distance, the route is discarded by deleting the source coordinate from the map. Summarizing, the steps carried out up to line 18 allow the identification of the candidate paths to represent the events' traces. The subsequent loop furtherly filters the routes and chooses the longest one for each starting street. Finally, the loop on line 23 rejects candidates whose path is entirely contained within a longer path. It sort the candidate traces by decreasing length. Then, for each trace it projects the start coordinate on the longest candidate path. If the length of the projection segment is shorter than a given threshold, then the candidate is discarded.

6.2.2 Storage

The advanced services of the GMIP framework require a data storage layer. As discussed in 5.2.1, this layer's design can be more or less complex, depending on the characteristics of the information. Therefore, a brief analysis of this GMIP's component can help determine which solution to adopt. The advanced services consist of a microservices architecture. In particular, each microservice requires an appropriate configuration to communicate successfully with the map providers and interact with the other framework components where required. Moreover, for safety reasons, the microservices installed on the On Board Unit may need a specific database for retrieving the vehicle status. For this reason, in the first version of GMIP's advanced services, the storage layer will consist of a non-relational database (NoSQL), i.e., MongoDB [64], for saving the configurations of the various microservices and the OBUs' station status. However, this solution does not fully comply with the modular approach. The correct architecture would involve creating databases for each microservice.

6.3 Visualization tool for C-ITS

To support Traffic Control Centers in monitoring traffic and events, GMIP implements a solution for observing ETSI C-ITS messages in real time. That is the second component of GMIP. Chapter 5.3.1 explains that this component will consist of two applications: the **visualizer** for the frontend and the **handler** for the backend. Below is a top-down analysis of the implementations.

6.3.1 Client side

The visualizer is the application that will show the geographic information of the ETSI C-ITS messages in real-time, possibly applying user-defined filters. By way of example, imagine having to monitor Cooperative Awareness vehicular messages of a geographic area without considering road events, the traffic light network, and other data. The CAM specification [65] establishes that its basic service shall generate the messages with a frequency interval between 1 and $10Hz$. Moreover, compression and encryption algorithms reduce payload size and safely preserve sensitive data. However, front-end applications must convert the data into a more usable format. Considering a JSON format, the size of a default message nearly approximates 1.7MB. Thus, in the worst-case scenario, the application should handle a minimum of 17MB per second for each station. Assuming order of magnitude equal to a hundred vehicles moving in the area, it would mean managing almost 200MB of data per second. This application will therefore have to optimize memory consumption to ensure the HMI's fluid and responsive behavior. Again, it is necessary to compare the frontend web frameworks to identify the technology that can best adapt to the visualizer's needs.

Angular It is a web framework designed by Google and first released in 2016. It is based on TypeScript and adopts a modular approach, promoting using small reusable components to build complex applications. However, its complexity and steep learning curve do not make it popular with the community.

React It is the longest-running web framework among those analyzed; Facebook released it in 2013. It bases on reusable components and promises lightning-fast performance. However, it does not come with good documentation, even though its community is quite large.

Svelte It is the most recent web framework and also the most innovative one. Promote code reuse, allowing component development. Unlike the others, it does not base on the concept of virtual DOM. Instead, it transforms the components into JavaScript code that updates the page on time.

Vue It is a web framework created in 2014 and inspired by AngularJS. It bases its architecture on reusable components and has curated documentation. However, it turns out to be incomplete.

The comparison of the framework performance highlights Svelte as the most efficient, as demonstrated by [66]. However, by extensively examining the modern frontend web frameworks, Angular proved the most comprehensive technology for building a single-page web application. Furthermore, exploiting a component-based architecture allows the design of complex HMI while ensuring simple and modular development. That is extremely important, as the visualizer will consist of multiple web page components that will have to interact with each other. Below is a characterization of the visualizer's HMI, consisting of a set of dashboards.

Dashboards

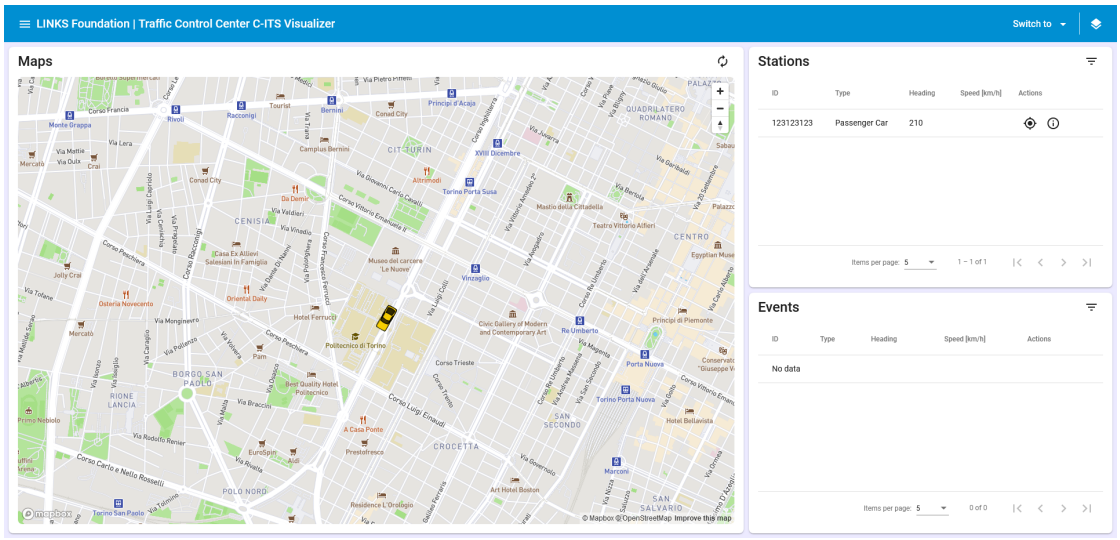


Figure 6.1: Visualization tool for C-ITS - Main Dashboard

Before starting the application description, it is necessary to point out that this paragraph could improperly refer to different pages, which can be confusing. However, the visualizer implements a single-page web application and exploits the routing engine provided by Angular to move between the pages.

First, the landing page supports the integration of an authentication form. However, the system requires a proper configuration to activate it. Then, the homepage design consists of the main dashboard, which splits into three components: the map and the list of stations and events. The stations correspond to any V2X communication unit in the ETSI C-ITS standards. Received messages belong to a

specific station. Each message requires an analysis to update the stored station data and display it on the map. Furthermore, the same operation will be carried out for each event, notifying the event in the component visible at the bottom right corner of figure 6.1. Finally, the first version of the visualization tool offers a detailed dashboard for the individual monitoring of the desired station. In particular, the stations' table of the main dashboard allows the routing to this page. Figure 6.2 shows its layout, which consists of a map to visualize the vehicle's path history and other valuable statistics.

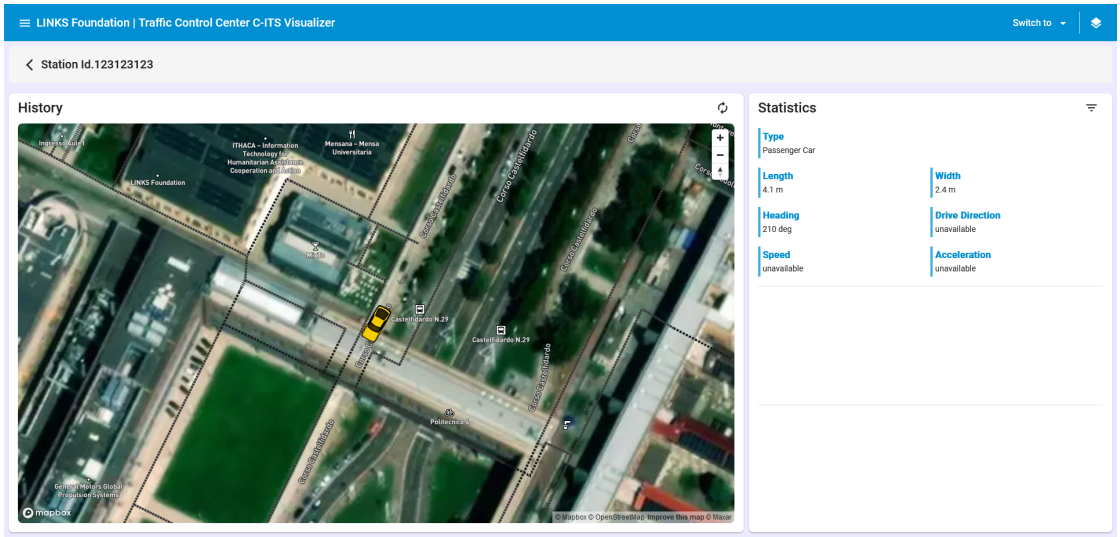


Figure 6.2: Visualization tool for C-ITS - Station Information Dashboard

6.3.2 Server side

The backend application is an interface between the cloud and the message viewer. As described in 5.3, C-ITS messages generated by vehicles and road infrastructure are forwarded to edge servers. Once messages are received, these servers can share data with third-party users through cloud-side message brokers and services. Hence, the handler communicates with these cloud entities to retrieve the messages and forward them, via WebSockets, to the visualizer.

Message Broker API

At start-up, the application loads its configuration into memory to establish connections to the cloud-side entities. In particular, each configured entity will establish a network connection based on its protocol type. For example, message brokers are software modules that exploit the publisher-subscriber messaging pattern to share

data. Therefore, they require a subscription request to a specific communication channel called topic. The first version of the handler will only support connection with Advanced Message Queuing Protocol (AMQP) and Message Queuing Telemetry Transport (MQTT) message brokers.

WebSocket API

The second role of the manager is to forward the messages received from the cloud to the viewer, establishing a unidirectional or bidirectional communication with the latter. Briefly recalling the previous subsection, the frontend application will show the information received on the map and allow the user to create geographic information and ETSI C-ITS messages. However, message forwarding will establish via unidirectional communication, while user interactions will leverage specific REST resources. Four leading technologies are available for implementing one-way communication: WebSocket, HTTP streaming, SSE, and RPC. WebSockets proved to be the best choice for real-time data visualization by briefly analyzing the proposed solutions. Considering that, the handler will register the communication channels for each type of ETSI C-ITS message to expose them to the visualizer.

6.3.3 Storage

The architecture of the visualization tool also provides for the use of a storage layer. In particular, the component requires saving the system configuration and ETSI C-ITS messages. However, these two categories of data require different saving methods. For example, the system configuration can be saved in plain text format, in an encrypted form, or directly to a database, depending on the data security needs and the system's complexity. On the contrary, the generation frequency of vehicular messages implies the inevitable adoption of complex systems for data management. Therefore, the messages will be stored in non-relational databases, i.e., NoSQL, to guarantee information historicization and exploit tools for controlled access to data. The first version of this GMIP tool will store the system configuration in a simple file, adopting the proper precautions to secure sensitive data.

Chapter 7

Results

In general, the research and development of new technologies include phases of validating the characteristics and evaluating the results. Specifically, software development includes defining functional and non-functional requirements, evaluating algorithms' effectiveness, and verifying the implementation's quality. Therefore, the chapter first wants to summarize the services offered by the framework. Then the exposure of the problems encountered will allow for discussing any solutions adopted to address these difficulties during the project's analysis, design, and implementation phases. Furthermore, this chapter aims to analyze the results obtained to evaluate the effectiveness of the Geographic Map Information Provider framework. In particular, the study will focus on the most relevant scenarios in the Connected, Cooperative and Automated Mobility context.
Below is an overview of the chapter's outline:

- Section 7.1 provides an extensive report of the features implemented for the GMIP framework and discusses the problems encountered during its development;
- Section 7.2 evaluates the proposed methodologies' outcomes, considering the most relevant use cases for a detailed analysis.

7.1 Implemented features analysis

The concept behind creating the Geographic Map Information Provider framework stems from the intention to enable the **interaction** between two technological systems, the **Geographic Information Systems** and the ETSI **Cooperative-ITS** vehicular **communication stack**. More precisely, the idea wants to support mobility innovation by **promoting cooperation** between vehicles and road infrastructures by exploiting geographic data.

The development of the GMIP framework involved multiple phases. At first, it was crucial to define the requirements that this framework should have met. In particular, the study analyzed the most relevant use cases in the CCAM context to facilitate the identification of requirements. Next, the design phase began with a high-level discussion of the components that would be part of the framework. This phase ended with a detailed definition of the architecture of the GMIP's components. Finally, based on the requirements previously identified, the implementation phase focused on creating services and tools to support OBU, RSU, and TCC.

The **first version** of the framework wants to focus on offering features mainly supporting **road safety**. Specifically, the advanced services component implements REST APIs for each of the contexts defined in 4.5, i.e., relating to information on location, road, routing, and events. Instead, the Traffic Control Center visualization tool implements vehicle monitoring and dangerous event trace creation. The validation of the tool's features required a simulation environment. This environment consists of the visualizer, the message broker, and the message generator. During testing, the visualizer registered to a broker topic to receive messages. The message generator simulated the Cooperative Awareness messages that a generic vehicle would generate and sent them to the message broker. The latter forwarded messages to the subscribers of a particular topic.

7.1.1 Challenges, problems, and solutions

Chapter 2 introduces some **metrics** (2.1.1) used to evaluate the **quality** of the **data** offered by **GISs**. These systems are responsible for storing a massive amount of geographic data. However, the creation and maintenance of this data can be complex and very expensive in terms of time and effort. Specifically, the information could be inaccurate, vague, outdated, or inconsistent. For instance, a traffic light intersection may have been transformed into a roundabout. A similar problem emerged during the development of GMIP. The validation of the event-aware routing service highlighted, in Google Maps, an uncertainty between the GIS data and reality. Specifically, a section of a motorway interchange incorrectly defined the permitted directions, thus allowing traffic to flow in only one direction, unlike reality. Fortunately, manually editing the map and requesting validation and approval of the contribution solved the problem. On the other hand, OpenStreetMap offers controlled public access to the data for quickly resolving any data inconsistency.

Again concerning the data quality, the development phase faced problems related to the **completeness** of the data, which made it impossible to satisfy some requirements. In particular, the services proposed by the map providers do not include data concerning the topology of the roads. Google Maps only offers a service for recovering the speed limits to be respected along a specific road. Instead, MapBox offers AI-powered services for detecting road signs using vehicle sensors.

As discussed above, OpenStreetMap offers public access to its data to enable project growth through user contributions. However, the only service that provides road data does not offer detailed information on the road topology. Therefore, it was impossible to implement solutions to support scenarios in which detailed knowledge of the road traveled is required. For example, a vehicle approaching an intersection would need information about the permitted directions.

Finally, another critical issue to address concerns the map providers' **services availability**. GMIP adopts a multi-provider approach to ensure greater flexibility in use in geographic areas with limited compatibility. Furthermore, this strengthens the reliability of the services. However, it may happen that a set of map providers do not implement specific features. Therefore, the first version of GMIP uses the **HTTP 501 Not Implemented** status code to notify the user of the unavailability of the service. Future implementations based on automatic provider selection policies may also consider this aspect. Furthermore, referring to OSM, integrating features within the framework can be difficult as it is necessary to exploit services provided by separate open-source projects.

7.2 Use cases validation

Following the in-depth examination of the features provided by the framework, this section wants to introduce the results obtained. Recalling 1.2, this thesis aims at enabling the exploitation of geographic data for supporting connected and automated vehicles in their cooperation. Therefore, the following subsections will analyze the impacts of the implemented functionalities on the framework's effectiveness. Each subsection will deal with a specific use case, selecting it from the list of the most relevant use cases discussed in chapter 4.

Speaking of **methodologies** for validating the results, the adoption of **numerical metrics** facilitates the interpretation and understanding of the examined data. It is noteworthy to report that the preliminary phases of this analysis involved a study of the strategies and metrics to be adopted. For instance, the GNSS enhancement use case assessment might employ **statistical analysis** of real-world scenarios. Therefore, reference geodetic and automotive GNSS traces would be required to represent the ground truth and the noisy signal, respectively. In this way, a conceivable metric could measure the distance between the coordinates of the map match results and the reference track. Instead, evaluating the wrong-way driving detection use case could exploit the **confusion matrix** and the traditional classification metrics to define its **accuracy**, **recall**, **precision**, or **F1 score**. Finally, validating the road event trace generator would involve defining a metric describing the algorithm's accuracy. Specifically, it should refer to the number of correctly generated traces out of the total. However, the numerical validation

of the results is not among the main objectives of this thesis. Furthermore, the definition of valuation methodologies based on personalized metrics may be the subject of future studies. Thus, for this examination, each use case will involve a graphical analysis of the results to facilitate their understanding and highlight the impact of the service for supporting CCAM applications. In particular, the images will consist of the superimposition of the data calculated by GMIP on specific geographical areas of the maps.

7.2.1 GNSS Enhancement

Exploiting a map matching service can help determine the road traveled by the vehicle, thus increasing the accuracy of geographic-related information. Below is a comparison between the standard map matching and the one proposed by GMIP.

Figure 7.1: GNSS Enhancement - Ground truth and noisy traces

Primarily, the figure 7.1 consists of two lines overlapped on a map. The thin black line characterizes the ground truth of the path. At the same time, the thick red line represents the vehicle's simulated GNSS measurements. In particular, the signal consists of the sum of the ground truth and an Additive White Gaussian Noise (AWGN) with zero mean and variance approximated to 8.9 meters. Note that the choice of the variance value fits an urban scenario, as evaluated in [67].

The second image 7.2 shows the map match response based on interacting with a map provider. This strategy uses raw noisy coordinates as input data. For the output, the data consists of a list of matched coordinates and a polyline estimate of the matched road route. Analyzing the results individually, the projection of each coordinate is correct. However, the estimated polyline is incomplete. In particular, it is necessary to highlight the presence of coordinates outside the ground truth

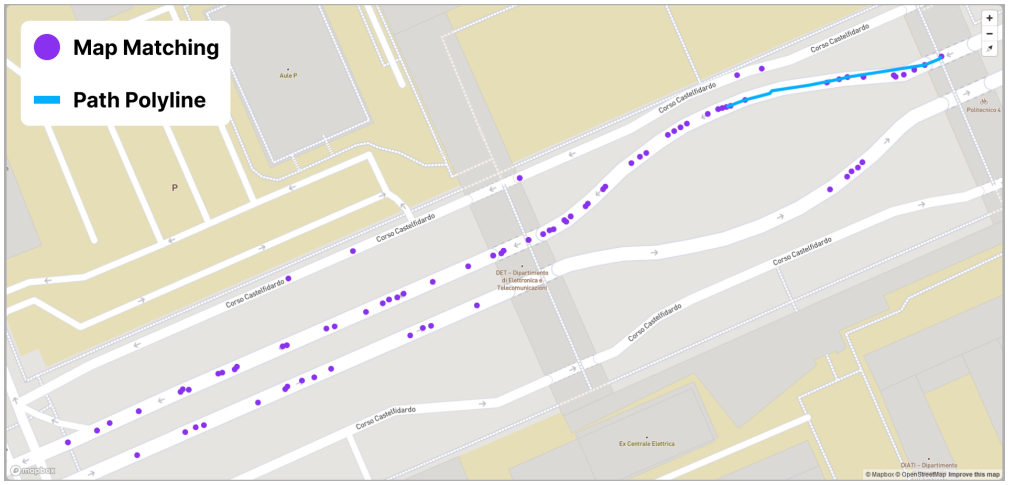


Figure 7.2: GNSS Enhancement - Map Matching

path, where noisy measurements are closer to the nearby roads. Service's behavior suggests that it cannot evaluate road matches by considering the history. Therefore, the matches in the side streets confuse the algorithm, resulting in a low-quality polyline estimate.

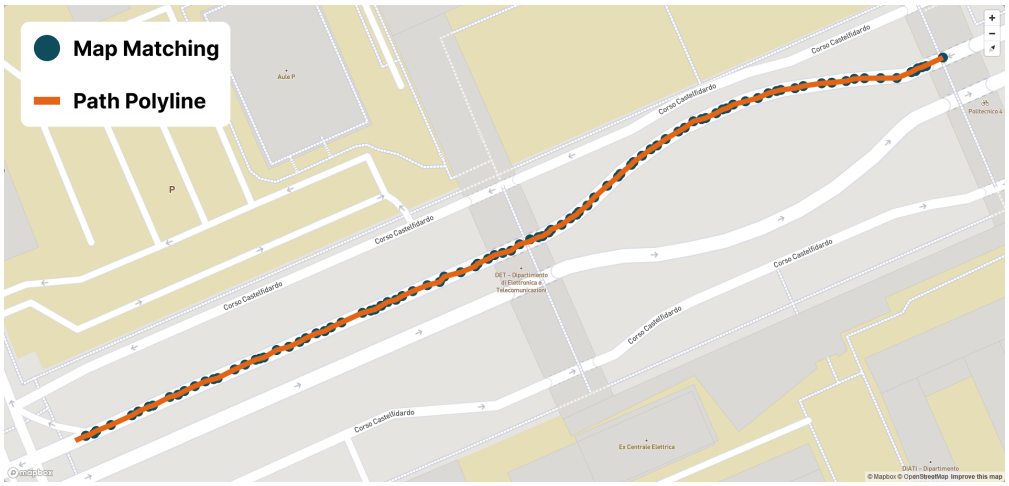


Figure 7.3: GNSS Enhancement - Filter-enhanced Map Matching

Considering the filtered map matching, the results shown in 7.3 demonstrate a substantial improvement in determining the vehicle's traveled road. In particular, GMIP proposes the adoption of a Kalman filter to mitigate the effects of noise on GNSS measurements. The strategy consists in evaluating the vehicle's path through the filter and employing the estimates as input of the map matching service.

7.2.2 Wrong-way Driving Detection

Concerning road safety, it is necessary to pay attention to accidents related to dangerous driving. In 2022, the Italian National Institute of Statistics (ISTAT) and ACI released a report [68] on road accidents in 2021. The analysis considers a total of approximately 200,000 registered accidents. The document shows that more than 2% relates to vehicles driving in the wrong direction.

In this regard, the GMIP framework wants to offer a service for detecting wrong-way driving scenarios. As discussed in the previous chapter, this feature leverages geographic data provided by map providers to calculate road-related information. In particular, GMIP relies on OpenStreetMap, as it is the only map provider to publicly provide detailed information on the topology of the road network. Precisely, it describes the road network employing a huge graph. Each road consists of nodes connected by arcs to define the feasible paths. However, the graph is not oriented. Thus, it does not include the description of one-way and multi-way streets. Consequently, detecting wrong-way driving events on multi-way roads is unfeasible. That is due to the difficulties in determining a priori the instants when the vehicle invades the oncoming lane based solely on GNSS data. The detection procedure suggested by GMIP requires comparing the vehicle's heading with the road's heading. However, for further details, the algorithm 6.1 describes the steps.

Figures 7.4, 7.5, and 7.6 shows the results obtained. Each image consists of a map, which highlights the road's structure and direction and the vehicle's position.

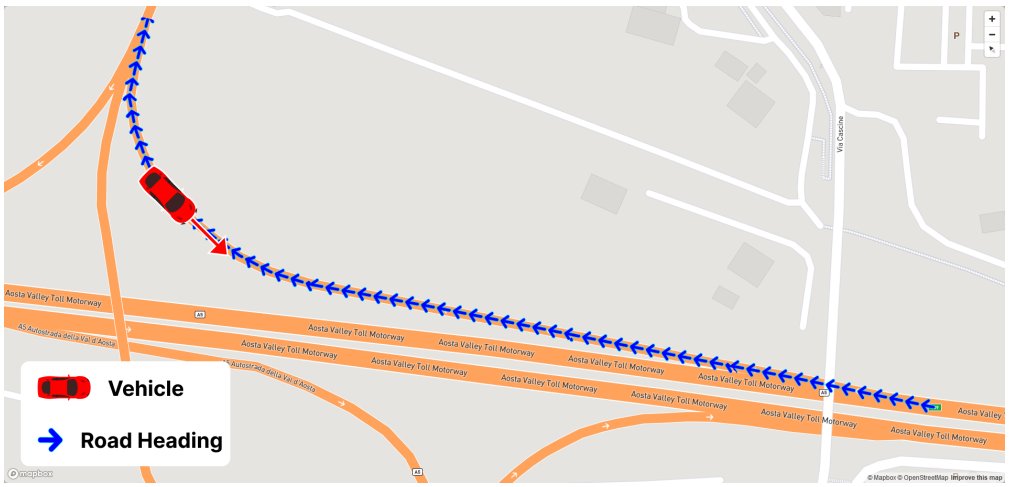


Figure 7.4: Wrong-way Driving Detection - Unsafe highway scenario

Image 7.4 takes into consideration a motorway exit. This road section consists of a single one-way lane from right to left. The simulation considers a vehicle in the opposite direction, that is, it is entering the highway in the wrong direction. The

algorithm recognizes the dangerous situation. The vehicle's heading is opposite to that of the road, considering a small neighborhood of the vehicle.

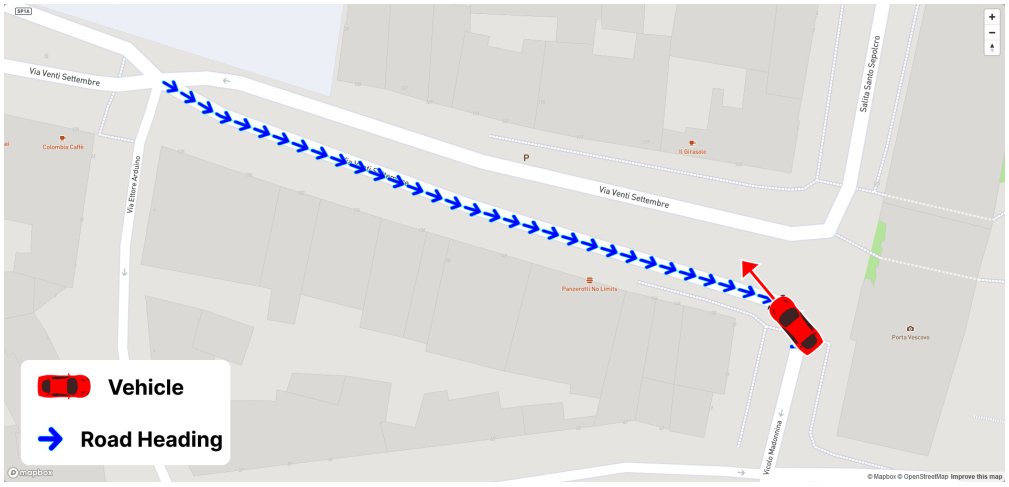


Figure 7.5: Wrong-way Driving Detection - Unsafe urban scenario

Similarly, image 7.5 reproduces a urban driving scenario. The vehicle's heading is opposite the direction of the road. Moreover, the vehicle's position is close to the road end. Thus, it can cause criticalities in data retrieval from the map provider. However, GMIP demonstrates to handle the situation satisfactorily.

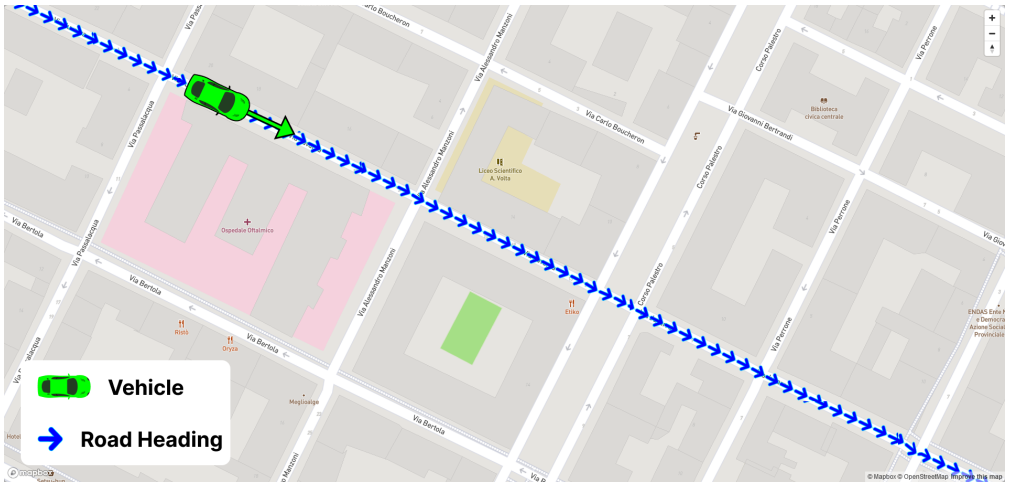


Figure 7.6: Wrong-way Driving Detection - Safe urban scenario

Image 7.6 aims to verify correct operation even in safe driving situations. This last scenario considers an urban context. The vehicle travels on a one-way street in the proper direction, so the algorithm does not detect dangerous events.

7.2.3 Event-aware Routing

Routing algorithms are the major players in digital maps and road navigation. Each proposes different strategies for calculating the optimal route in specific scenarios. GMIP offers a service for calculating the alternative fastest route by considering any relevant events notified through the ETSI C-ITS stack.

The following figures illustrate the results obtained by GMIP in different contexts. Each image shows the original and recalculated routes and the events on a map. In particular, the green marker identifies the starting position, while the red one identifies the finish. Danger triangles identify the locations of reported events.

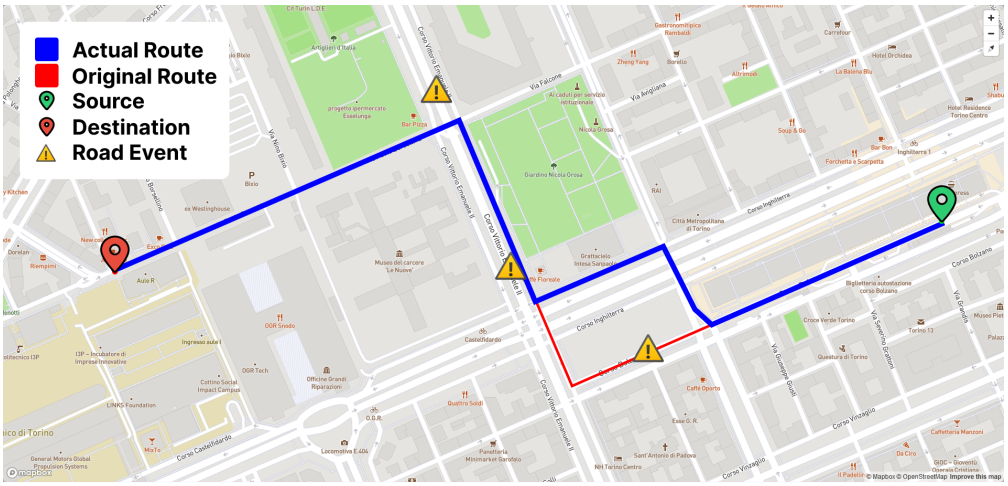


Figure 7.7: Event-aware Routing - Urban scenario

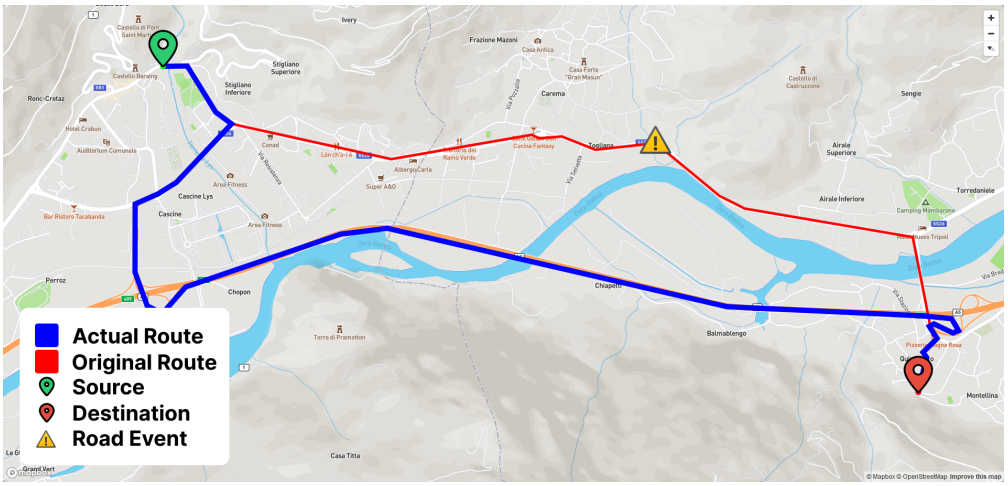


Figure 7.8: Event-aware Routing - Highway access required scenario

First, image 7.7 considers an urban context. The right-most event forces the algorithm to recalculate a route. On the other hand, the second event occurs on the main street. Thus, the process diverts the path along the side street. Image 7.8 depicts a highway scenario. In this simulation, the occurrence of a dangerous event along the only viable primary road forces the use of the highway to arrive at the destination.

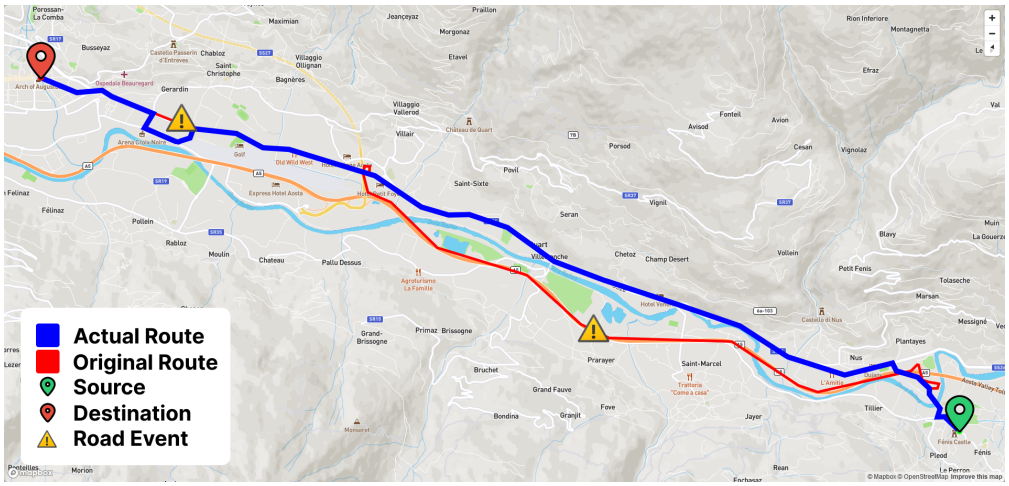


Figure 7.9: Event-aware Routing - Medium-long path scenario

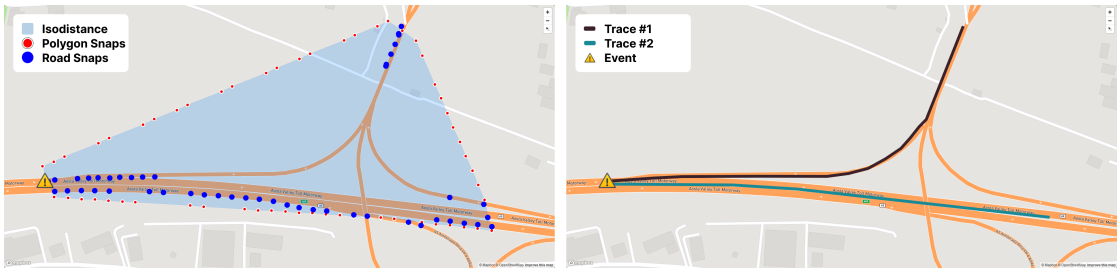
Finally, the events illustrated in image 7.9 aims to validate the effectiveness of the service in calculating an alternative route in a medium-long distance circumstances. The highway event diverts the vehicle's path to the primary road. Instead, the event close to the destination causes the selection of an alternative route.

7.2.4 Road Event Creation

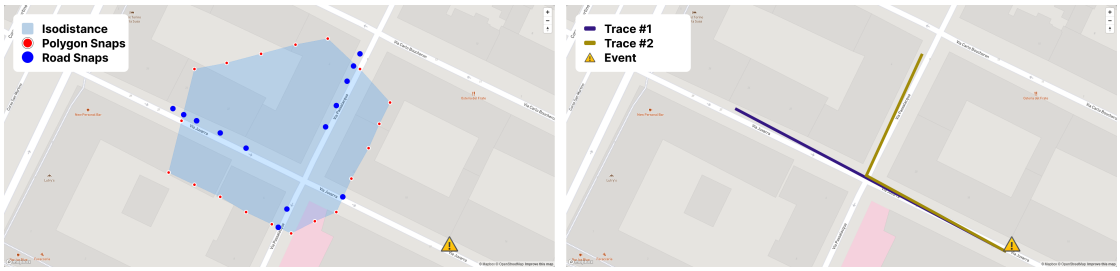
Road safety is undoubtedly one of the most vital objectives for research and technological innovation in the automotive sector. Regarding smart mobility, CCAM promotes safety through the cooperation of vehicles and infrastructure. For this, the ETSI C-ITS communication stack offers services for disseminating Decentralized Environmental Notification messages. These messages include all the details for describing the occurrence of dangerous road events. Specifically, referring to the specification [69], a DEN message can include up to 7 relevant paths leading to the event. Thus, the Event Traces Generator service implemented by GMIP proposes automatically creating these traces. Algorithm 6.3 provides further details about the procedure's steps.

Figure 7.10 presents the results obtained by the generator in different road contexts. Each scenario consists of two images to facilitate an understanding of how

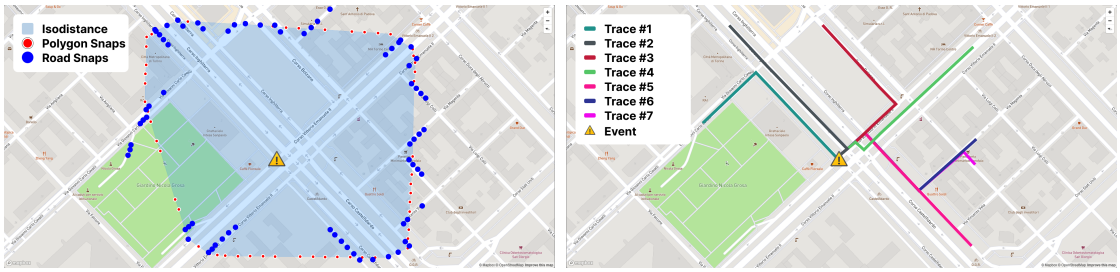
computation occurs. The first image shows the data exploited by the algorithm for identifying the traces. In particular, the semi-transparent blue polygon identifies the road area at a maximum distance. On the other hand, the blue points identify the projections of the perimeter points on the nearest road. The second image shows the traces generated by the procedure, coloring each of them differently.



(a) Highway entrance scenario



(b) Urban scenario



(c) Complex urban scenario

Figure 7.10: Event Traces Generator - Scenarios comparison

Image 7.10a refers to an extra-urban context near a motorway entrance, thus the isodistance refers to a maximum distance of $500m$. The projection of the polygon's perimeter coordinates depicts the opposite lane, the correct lane, the exit, and the motorway entrance as trace candidates. However, the generator successfully computes the traces by excluding the invalid ones. Similarly, image 7.10b demonstrates the correct behavior of the framework in an urban context at a maximum distance of $100m$. Specifically, the two blue points at the bottom

left do not produce traces, as they would need to go around the block. The point closest to the event belongs to longer routes; thus, the algorithm discards it. Finally, image 7.10c simulates a complex urban scenario. In this case, the algorithm could not calculate all the feasible traces. However, this behavior depends on the representation of geographic data. For instance, it is impossible to easily distinguish an avenue from its counter avenue, as the address is identical.

Chapter 8

Conclusion

The significant phenomenon of **urbanization**, which is constantly growing in every city in the world, has led to the study of new architectural and technological solutions aimed at radical change. In this regard, the **Internet-of-Things** concept finds application in cities. A **Smart City** introduces six areas of competence, each responsible for proposing solutions to the criticalities of conventional cities. Among these, **Smart Mobility** will have to deal with the **efficiency** and **safety** of the road **transport** network.

Over the years, the exploitation of geographic data has proved essential to support research and innovation in transportation, especially for road vehicles and infrastructures. The idea of employing geographical data in the transportation context was born from the need to provide new technologies and services to promote the safety and sustainability of road vehicles. Precisely, this study aims to exploit such knowledge for assisting the transition of road mobility toward a connected and cooperative future. Furthermore, the research phase in the scientific literature was instrumental in finding applications within **ITS** to support road infrastructures and promote user cooperation with the help of **Geographic Information Systems**. However, it is necessary to highlight the lack of proposals for the connected and automated cooperation of vehicles and infrastructure, i.e., the **Connected, Cooperative and Automated Mobility** field. This project aims to promote this behavior by proposing a solution for the manipulation and dissemination of geographic knowledge through the **ETSI C-ITS** communication stack.

This thesis proposes a software framework called **Geographic Map Information Provider (GMIP)**. GMIP provides services and tools for exchanging, creating, and monitoring geographic information. For this reason, the framework offers compatibility with the most comprehensive map providers on the market, namely Google, OpenStreetMap, and MapBox. Furthermore, the framework comprises two distinct software components as its use will support different Cooperative-ITS contexts. Specifically, these components are designed to meet the

usage constraints and requirements identified by On Board Units, Road Side Units and Traffic Control Centers.

The first component consists of a set of microservices, organized in Docker containers, for exposing APIs as web resources. The idea is to promote the interaction between vehicles and infrastructures for sharing geographic data among CCAM applications. In particular, it proposes services related to map matching, wrong-way driving detection, intelligent and cooperative routing, generation of road events' traces, and more.

Instead, the second solution arises from the need to provide a tool to facilitate the surveillance of road vehicles and infrastructures. Specifically, a TCC requires the presence of human operators to monitor and manage traffic through the use of multiple technologies. For this reason, the second component of GMIP consists of a single page web application for displaying ETSI C-ITS messages. That allows for real-time monitoring of vehicular messages on a map.

In summary, the GMIP framework implements an efficient, extremely flexible, and environment-independent solution. Also, although it was impossible to comply with some of the requirements defined during the developments, GMIP successfully enables the exploitation of geographic data for supporting OBUs, RSUs, and TCCs.

The evaluation of the framework's effectiveness comprised two different stages. First, concerning the framework's advanced services component, the behavior assessment considered some of the most relevant use cases in CCAM. In particular, using a Kalman filter to mitigate the impact of noise on vehicle GNSS measurements demonstrates the feature's usefulness in improving the robustness of the Map Matching service and providing more reliable results. Then, the exploitation of OpenStreetMap for retrieving information related to the road's composition allowed the implementation of an algorithm for detecting wrong-way driving situations. The results obtained by the algorithm prove the effectiveness of the methodology. However, as explained in the previous chapter 7.1.1, this feature cannot be employed in cases of invasion of the oncoming lane. GMIP wants to help vehicles on their journeys by supporting their cooperation. In this regard, the event-aware routing service can correctly calculate alternative routes to avoid any events that occur along the route in urban and medium-long distance scenarios. Finally, the algorithm designed to generate the traces of the events proves to be robust and effective in most cases. In particular, the evaluations refer to extra-urban and urban contexts, and in both cases, the computed traces are accurate. However, when simulating a more complex scenario, it is necessary to indicate the partial inaccuracy of the procedure. This problem leads back to the inaccuracy of the geographic data provided by the map providers. More precisely, it was not possible to calculate some traces as it is not possible to distinguish an avenue from its counter avenue. On the other hand, the visualization tool evaluation consisted of the simulated generation of ETSI C-ITS Cooperative Awareness messages. In particular, the

visualizer subscribed to a message broker, constantly publishing the messages from the generator. The analysis aimed to verify the resource management of the handler and the visualizer HMI's smoothness.

Thus, validating the framework's effectiveness by simulating real-world scenarios constitutes the achievement of the objectives of the thesis. That demonstrates the project's feasibility and shows the significant impact that geographic information systems can have in interacting with vehicles and infrastructures within CCAM. However, as a final consideration, it is essential to underline the strong **dependence** of the framework's **effectiveness** on the **quality** of the **data** and **services** provided by the map providers.

8.1 Future work

Although the first version of the framework offers advanced solutions for vehicles and road infrastructure, the project is in constant development. As discussed in the previous chapter, some of the requirements defined during the development phase could not be met. In particular, the lack of detailed information on the topology of the roads has prevented the implementation of road knowledge-related services. For this reason, a future solution to overcome this problem would be exploiting ETSI C-ITS Map Extended messages. These messages describe the road intersections, providing detailed information on the characteristics of the road and lanes in a neighborhood of about $200m$. An alternative would be to leverage OpenStreetMap's self-hosting solutions. That would allow manually creating geographical structures to be included in OSM to provide a detailed road description. However, this strategy would require a considerable amount of time and effort.

Another future evolution of the framework concerns map providers. The first version of GMIP supports three providers: Google Maps, OpenStreetMap, and MapBox. Compatibility could therefore improve by adding support to other providers. In addition, automatic map provider selection policies could be implemented based on geographic area and usage restrictions.

Speaking of event trace generator, in the future the algorithm could be strengthened to improve the results in very complex scenarios. A potential approach could employ clustering on the projected coordinates.

Regarding the visualization tool for TCC, it would be worthwhile to implement applications for simulating the dissemination of road events and the state of traffic lights in the future. That could help to validate the satisfaction of the requirements.

Finally, the framework could implement features not yet discussed or not directly related to the exploitation of Geographic Information Systems. For example, GMIP could integrate routing solutions based on the state of level crossings or automate the creation of DEN messages in areas affected by adverse weather conditions.

Bibliography

[1] United Nations - Department of Economic and Social Affairs. *The World's Cities in 2018*. 2018 (cit. on p. 1).

[2] CEDR. *2019 Performance Report*. Tech. rep. Conference of European Directors of Roads, 2019 (cit. on p. 3).

[3] Xuyu Wang, Shiwen Mao, and Michelle X Gong. «An overview of 3GPP cellular vehicle-to-everything standards». In: *GetMobile: Mobile Computing and Communications* 21.3 (2017), pp. 19–25. DOI: 10.1145/3161587.3161593 (cit. on p. 4).

[4] Andreas Festag. «Cooperative intelligent transport systems standards in Europe». In: *Communications Magazine, IEEE* 52 (Dec. 2014), pp. 166–172. DOI: 10.1109/MCOM.2014.6979970 (cit. on p. 4).

[5] PierNext. *Connected Infrastructures*. 2021. URL: https://piernext.portdebarcelona.cat/en/mobility/self-driving-vehicles-objective-2030/ (cit. on p. 5).

[6] Jean-Claude Thill. «Geographic information systems for transportation in perspective». In: *Transportation Research Part C: Emerging Technologies* 8.1 (2000), pp. 3–12. DOI: 10.1016/S0968-090X(00)00029-2 (cit. on p. 5).

[7] Wikipedia contributors. *Software framework — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/w/index.php?title=Software_framework&oldid=1107942821 (cit. on p. 6).

[8] Stephen C Guptill and Joel L Morrison. *Elements of spatial data quality*. Elsevier, 2013 (cit. on p. 10).

[9] Joana María Seguí Pons and Maurice Ruiz Pérez. «Geographic information systems and intelligent transport systems: technologies used to form new communication networks». In: *NETCOM: Réseaux, communication et territoires/Networks and communication studies* 17.1 (2003), pp. 53–70. DOI: 10.3406/netco.2003.1572 (cit. on p. 10).

[10] OpenStreetMap Wiki. *Zoom levels — OpenStreetMap Wiki*. 2021. URL: https://wiki.openstreetmap.org/w/index.php?title=Zoom_levels&oldid=2167744 (cit. on p. 12).

[11] MapTiler Support. *Raster vs Vector Map Tiles: What Is the Difference Between the Two Data Types?* 2021. URL: https://documentation.maptiler.com/hc/en-us/articles/4411234458385-Raster-vs-Vector-Map-Tiles-What-Is-the-Difference-Between-the-Two-Data-Types- (cit. on p. 13).

[12] Sakiko Nishino, Kimitake Wakayama, Md. Shoaib Bhuiyan, Haruki Kawanaka, and Oguri Koji. «Measurement and Management of the Lane Markings' Stripping Ratio from In-vehicle Camera Image». In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 1755–1760. DOI: 10.1109/ITSC.2015.285 (cit. on p. 14).

[13] Juliet Moso, Ramzi Boutahala, Brice Leblanc, Hacène Fouchal, Cyril De Runz, Stephane Cormier, and John Wandeto. «Anomaly Detection on Roads Using C-ITS Messages». In: Dec. 2020, pp. 25–38. DOI: 10.1007/978-3-030-66030-7_3 (cit. on p. 14).

[14] Ane Dalsnes Storsaeter, Jo Skjermo, Odd Andre Hjelkrem, Jan Erik Hakegard, Petter Arnesen, and Erlend Dahl. «A GIS-Based tool for Optimizing C-ITS Communication Infrastructure». In: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. 2019, pp. 1–6. DOI: 10.1109/VTCFall.2019.8891519 (cit. on p. 15).

[15] Fang Clara Fang and Neftali Torres. «Analysis of hydrogen station network using geographic information systems». In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2011, pp. 834–839. DOI: 10.1109/ITSC.2011.6082933 (cit. on p. 15).

[16] *Car Ownership Report*. 2022. URL: https://www.confused.com/car-insurance/car-ownership-report (cit. on p. 15).

[17] Pavel Kubíček, Dalibor Bartoněk, Jiri Bures, and Otakar Švábenský. «Proposal of Technological GIS Support as Part of Resident Parking in Large Cities–Case Study, City of Brno». In: *Symmetry* 12 (Apr. 2020), p. 542. DOI: 10.3390/sym12040542 (cit. on p. 15).

[18] Christian Kjellström and Håkan Regnér. «The Effects of Geographical Distance on the Decision to Enrol in University Education». In: *Scandinavian Journal of Educational Research* 43.4 (1999), pp. 335–348. DOI: 10.1080/0031383990430401 (cit. on p. 16).

[19] Mofareh Qoradi et al. «Using GIS-based intelligent transportation systems in the enhancement of university campus commuting in a smart city context». In: *Arabian Journal of Geosciences* 14 (May 2021), p. 742. DOI: 10.1007/s12517-021-07098-z (cit. on p. 16).

[20] Luis Leon Ojeda, Alexandre Chasse, and Romain Goussault. «Fuel consumption prediction for heavy-duty vehicles using digital maps». In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 1–7. DOI: 10.1109/ITSC.2017.8317613 (cit. on p. 16).

[21] Google. *Google Maps Eco-Friendly Routing - Accelerating the journey to sustainability for one billion people*. 2021. URL: https://www.gstatic.com/gumdrop/sustainability/google-maps-eco-friendly-routing.pdf (cit. on pp. 16, 77).

[22] Anne-Sophie Puthon, Fawzi Nashashibi, and Benazouz Bradai. «A complete system to determine the speed limit by fusing a GIS and a camera». In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2011, pp. 1686–1691. DOI: 10.1109/ITSC.2011.6082951 (cit. on p. 17).

[23] Yong-Geon Choi, Kyung-Il Lim, and Jung-Ha Kim. «Lane change and path planning of autonomous vehicles using GIS». In: *2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. 2015, pp. 163–166. DOI: 10.1109/URAI.2015.7358855 (cit. on p. 17).

[24] Mengyin Fu, Wenjie Song, Yang Yi, and Meiling Wang. «Path Planning and Decision Making for Autonomous Vehicle in Urban Environment». In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 686–692. DOI: 10.1109/ITSC.2015.117 (cit. on p. 17).

[25] Wikipedia contributors. *Nudge theory — Wikipedia, The Free Encyclopedia*. 2022. URL: https://en.wikipedia.org/w/index.php?title=Nudge_theory&oldid=1109863061 (cit. on p. 17).

[26] Bhupendra Singh and Ankit Gupta. «Recent trends in intelligent transportation systems: a review». In: *Journal of Transport Literature* 9 (Apr. 2015), pp. 30–34. DOI: 10.1590/2238-1031.jtl.v9n2a6 (cit. on p. 17).

[27] Zhong-Ren Peng. «A methodology for design of a GIS-based automatic transit traveler information system». In: *Computers, Environment and Urban Systems* 21.5 (1997), pp. 359–372. DOI: 10.1016/S0198-9715(98)00006-4 (cit. on p. 17).

[28] Jianwei Zhang, Feixiong Liao, Theo Arentze, and Harry Timmermans. «A multimodal transport network model for advanced traveler information systems». In: *Procedia - Social and Behavioral Sciences* 20 (2011), pp. 313–322. DOI: 10.1016/j.sbspro.2011.08.037 (cit. on p. 18).

[29] Jungwoo Cho and Yoonjin Yoon. «GIS-Based Analysis on Vulnerability of Ambulance Response Coverage to Traffic Condition: A Case Study of Seoul». In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 1402–1407. DOI: 10.1109/ITSC.2015.230 (cit. on p. 18).

[30] Long Ke-jun, Liu Yong, and Luo Xiangwu. «Emergency Accident Rescue System in Freeway Based on GIS». In: *2008 International Conference on Intelligent Computation Technology and Automation (ICICTA)* 2 (2008), pp. 247–250. DOI: 10.1109/ICICTA.2008.410 (cit. on p. 18).

[31] Christophe Claramunt, Evtim Peytchev, and Andrzej Bargiela. «Real-time GIS for the analysis of a traffic system». In: vol. 1. Feb. 1999, 15–20 vol.1. DOI: 10.1109/AFRCON.1999.820669 (cit. on p. 18).

[32] Sourabh Jain and S. Jain. «Development of Intelligent Transportation System and Its Applications for an Urban Corridor During COVID-19». In: *Journal of The Institution of Engineers (India): Series B* 102 (May 2021). DOI: 10.1007/s40031-021-00556-y (cit. on p. 18).

[33] Vittorio Astarita, Demetrio Carmine Festa, Pasquale Giofrè, Giuseppe Guido, and Domenico Walter Edvige Mongelli. «Co-operative ITS: ESD a Smartphone Based System for Sustainability and Transportation Safety». In: *Procedia Computer Science* 83 (2016), pp. 449–456. DOI: 10.1016/j.procs.2016.04.208 (cit. on p. 18).

[34] IEEE. «ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering». In: *ISO/IEC/IEEE 29148:2018(E)* (2018), pp. 1–104. DOI: 10.1109/IEEESTD.2018.8559686 (cit. on pp. 34, 35).

[35] LINKS Foundation. *Passion for Innovation*. 2022. URL: https://linksfoundation.com/ (cit. on pp. 36, 37).

[36] Storsaeter et al. «A GIS-Based tool for Optimizing C-ITS Communication Infrastructure». In: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. 2019, pp. 1–6. DOI: 10.1109/VTCFall.2019.8891519 (cit. on p. 36).

[37] M. Fünfrocken et al. *CONVERGE Project - Deliverable D4.3 Architecture of the Car2X systems network*. 2015 (cit. on p. 36).

[38] ICT4CART Partners. *A connected future for automated driving*. 2022. URL: https://www.ict4cart.eu/ (cit. on p. 37).

[39] Swarco. *A system integrator for ITS in Romania*. 2022. URL: https://www.swarco.com/companies/swarco-traffic-romania-srl (cit. on p. 37).

[40] IEEE. «IEEE Recommended Practice for Architectural Description for Software-Intensive Systems». In: *IEEE Std 1471-2000* (2000), pp. 1–30. DOI: 10.1109/IEEESTD.2000.91944 (cit. on p. 53).

[41] S. Newman. *Building Microservices*. 2021. ISBN: 9781492033998 (cit. on p. 57).

[42] M.L. Abbott and M.T. Fisher. *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise*. 2015. ISBN: 9780134032801 (cit. on p. 57).

[43] Bob Reselman (Red Hat). *An architect's guide to APIs: SOAP, REST, GraphQL, and gRPC*. 2020. URL: https://www.redhat.com/architect/apis-soap-rest-graphql-grpc (cit. on p. 58).

[44] Roy Thomas Fielding. «REST: architectural styles and the design of network-based software architectures». In: *Doctoral dissertation, University of California* (2000) (cit. on p. 58).

[45] GraphQL Foundation. *A query language for your API*. 2022. URL: https://graphql.org/ (cit. on p. 58).

[46] gRPC Authors. *A high performance, open source universal RPC framework*. 2022. URL: https://grpc.io/ (cit. on p. 58).

[47] Docker Inc. *Develop faster. Run anywhere.* 2022. URL: https://www.docker.com/ (cit. on p. 60).

[48] C++ Foundation. *C++ Programming Language*. 2022. URL: https://isocpp.org/about (cit. on p. 66).

[49] Google. *Go Programming Language*. 2022. URL: https://go.dev (cit. on p. 66).

[50] Oracle. *Java Programming Language*. 2022. URL: https://dev.java/ (cit. on p. 66).

[51] Kotlin Foundation. *Kotlin Programming Language*. 2022. URL: https://kotlinlang.org/ (cit. on p. 66).

[52] Python Software Foundation. *Python Programming Language*. 2022. URL: https://www.python.org/about/ (cit. on p. 66).

[53] Rust Foundation. *Rust Programming Language*. 2022. URL: https://www.rust-lang.org/ (cit. on p. 66).

[54] GitHub Inc. *Top languages over the years*. 2021. URL: https://octoverse.github.com/#top-languages-over-the-years (cit. on p. 67).

[55] Will Glozer. *wrk - a HTTP benchmarking tool*. 2022. URL: https://github.com/wg/wrk (cit. on p. 68).

[56] Raspberry Pi Foundation. *Raspberry Pi 3 Model B+ Specification*. 2022. URL: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/ (cit. on p. 68).

[57] SWIG developers. *SWIG-4.0 Documentation*. 2022. URL: https://www.swig.org/Doc4.0/SWIGDocumentation.html (cit. on p. 68).

[58] Balena. *What is balenaOS?* 2022. URL: https://www.balena.io/os/docs (cit. on p. 69).

[59] Yocto Project. *About the Yocto Project*. 2022. URL: https://www.yoctoproject.org/about (cit. on p. 69).

[60] Fenny and Contributors. *An Express-inspired web framework written in Go.* 2022. URL: https://gofiber.io/ (cit. on p. 71).

[61] The Linux Foundation. *OpenAPI Initiative*. 2022. URL: https://www.openapis.org/ (cit. on p. 71).

[62] Google. *Encoded Polyline Algorithm Format*. 2022. URL: https://developers.google.com/maps/documentation/utilities/polylinealgorithm (cit. on p. 71).

[63] Greg Welch, Gary Bishop, et al. «An introduction to the Kalman filter». In: (1995). DOI: 10.1.1.117.6808 (cit. on p. 72).

[64] MongoDB Inc. *Build faster. Build smarter.* 2022. URL: https://www.mongodb.com/ (cit. on p. 81).

[65] ETSI. *Specification of Cooperative Awareness Basic Service*. Tech. rep. 2019. URL: https://www.etsi.org/deliver/etsi_EN/302600_302699/03026370 2/01.04.01_30/en_30263702v010401v.pdf (cit. on p. 82).

[66] krausest. *A comparison of the perfomance of a few popular JavaScript frameworks*. 2022. URL: https://github.com/krausest/js-framework-benchmark (cit. on p. 83).

[67] Andrej Štern and Anton Kos. «Positioning performance assessment of geodetic, automotive, and smartphone gnss receivers in standardized road scenarios». In: *IEEE access* 6 (2018), pp. 41410–41428 (cit. on p. 90).

[68] ISTAT. *Incidenti stradali in Italia. Anno 2021*. Tech. rep. Italian National Institute of Statistics, 2022 (cit. on p. 92).

[69] ETSI. *Specification of Decentralized Environmental Notification Basic Service*. Tech. rep. 2019. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.03.01_60/en_30263703v010301p.pdf (cit. on p. 95).