uc3m | Universidad Carlos III de Madrid

University Degree in Data Science and Engineering and
Telecommunication Technologies Engineering
2024 – 2025

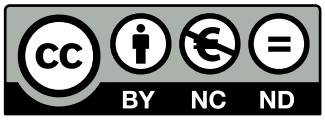
Bachelor thesis

Building a Scalable Parking Management Solution for Multi-Community Deployment

Andrés Navarro Pedregal

Tutor
David Larrabeiti López

Leganés, 2025



ABSTRACT

this is an abstract

Keywords: keyword1, keyword2, keyword3

ACKNOWLEDGMENTS

Thanks

AGRADECIMIENTOS

Gracias

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACROYNMS

AI	Artificial Intelligence.
AWS	Amazon Web Service.
GDPR	General Data Protection Regulation.
IaaS	Infrastructure as a Service.
INCOSE	International Council on Systems Engineering.
IoT	Internet of Things.
KPI	Key Performance Indicators.
LPR	Licence Plate Recognition.
ML	Machine Learning.
NIS	Network and Information Systems.
PaaS	Platform as a Service.
PMS	Parking Management System.
PoE	Power over Ethernet.
SaaS	Software as a Service.

Part I

Introduction

1. MOTIVATION

The rapid growth of urban populations and the corresponding increase in vehicle ownership have significantly intensified the challenges associated with parking management in cities worldwide. For instance, the dramatic rise in car usage in Madrid has substantially contributed to heightened levels of environmental pollution [1]. Despite the number of parking spaces in Spain remaining stable between 2014 and 2020 [2], the escalating demand for parking has resulted in higher costs, prolonged search times, increased traffic congestion, and elevated urban pollution levels.

The inadequacies of traditional, manual Parking Management Systems (PMSs) have further highlighted the need for innovative solutions in this domain. Studies highlight the importance of user-friendly, secure, and reliable systems, with these factors playing a crucial role in influencing parking behavior and preferences [3]. Consequently, the development of effective PMSs remains an essential area of research, particularly as urbanization continues to accelerate.

Advancements in technology provide new opportunities to address these challenges. The proliferation of internet-connected devices, growing by 20% annually [4], has catalyzed the emergence of Internet of Things (IoT)-based PMSs. These systems aim to mitigate traffic congestion and reduce urban pollution while promoting sustainable and efficient urban mobility. By leveraging real-time data and automation, modern PMSs offer transformative potential for urban planning and management.

Integrating these advanced systems into urban environments offers multiple benefits, including decreased traffic congestion, reduced pollution, enhanced security, and an improved quality of life for residents. Additionally, PMSs enable automated parking space management, making them adaptable for a wide range of applications, including urban centers, residential communities, and commercial buildings.

Despite these advancements, the widespread adoption of PMSs in Spain has faced significant obstacles. Many existing systems rely heavily on human intervention, leading to delays, insufficient information dissemination, and limited control over parking space availability. This dependence on manual operations not only impairs system efficiency but also increases operational costs.

Emerging technologies have inspired the development of innovative PMSs, such as RFID-based systems [5], IoT-enabled solutions [6], and intelligent parking systems employing image processing techniques [7]. However, these systems frequently rely on centralized architectures, which are prone to scalability and reliability issues, particularly during service disruptions. Moreover, many current solutions lack the flexibility needed to accommodate the specific requirements of diverse user groups, limiting their broader applicability.

Addressing these limitations, the primary objective of this project is to design and implement a fully distributed and autonomous PMS that overcomes the challenges inherent in existing systems. This new approach emphasizes scalability, reliability, and adaptability, with a particular focus on meeting the demands of next-generation smart cities. Through this work, the project aims to contribute to the development of sustainable, efficient, and user-centric parking solutions for urban environments.

TODO: add a graph or somehting to showcase the importance

2. STATEMENT OF THE PROBLEM

The lack of efficient PMSs is a pervasive problem in modern cities, exacerbated by the growing number of vehicles and the resulting increase in urban population. Traditional manual PMSs have proven inadequate in addressing the complexities of modern urban parking, leading to higher costs, longer search times, traffic congestion, and elevated levels of urban pollution. The ongoing development of PMSs remains a crucial research area, as citizens favor systems that prioritize user-friendliness, security, and reliability [3] [8] [9] [10].

This project aims to address the gap by designing and implementing a fully distributed PMS tailored for the next generation of smart cities. The proposed system will focus on addressing the inefficiencies and challenges inherent in current PMSs through a distributed approach that leverages modern technologies.

Given the broad nature of this problem, the scope of this project will be narrowed to a specific environment and use case, as outlined below:

- **Environment:** The system will be designed for urban environments with high vehicle density and limited parking spaces. The case study will focus on a city center with a high volume of vehicles and limited parking availability, reflecting the challenges faced by many modern cities.
- **Atmospheric Conditions:** The system will be designed to operate under typical urban atmospheric conditions, with no specific constraints on temperature, pressure, or humidity. The system will be designed to function in a variety of weather conditions, including rain, heat, and cold.
- **Operational Parameters:** The system will be designed to manage parking spaces in a city center, with a focus on optimizing space utilization, reducing search times, and minimizing traffic congestion. The system will be scalable to accommodate a large number of parking spaces and users.
- **Hardware:** The system will be designed to leverage modern technologies such as IoT devices, sensors, and cloud computing to provide real-time monitoring and management of parking spaces. The system will be designed to be cost-effective, scalable, and user-friendly.

3. OBJECTIVES

The primary goal of this bachelor thesis is to design and implement a distributed parking management system to address the evolving needs of smart cities. The project aims to overcome the limitations of existing parking management solutions by utilizing scalable and IoT technologies. By doing so, it seeks to enhance performance, security, and user experience, contributing to the development of smart city solutions. The specific objectives of the project are outlined below:

- **Analyze Existing Systems:** Review current parking management systems, identifying inefficiencies and challenges. This includes assessing the limitations of centralized systems and understanding the needs of users (drivers and facility managers) to uncover areas for improvement using a distributed approach.
- **Evaluate Relevant Technologies:** Examine key technologies used in parking management systems, focusing on their suitability for a distributed architecture. This involves evaluating IoT devices, sensors, communication protocols, and data storage methods to determine the most scalable and secure technologies for the proposed system.
- **Design the System Architecture:** Develop the overall design of the distributed parking management system. This includes selecting IoT sensors, determining data storage solutions, and ensuring that the system can handle high data loads while integrating seamlessly with other smart city systems.
- **System Development:** Implement the system using an agile approach, following best practices in software engineering. This includes iterative planning, design, coding, testing, deployment, and maintenance, with an emphasis on modularity and extensibility to accommodate future updates and integration with emerging technologies (e.g., autonomous vehicles, AI-based traffic management).
- **Evaluate System Performance:** Assess the implemented system's performance based on Key Performance Indicatorss (KPIs) such as scalability, reliability, data accuracy, and real-time processing capabilities to ensure the system meets its design objectives.
- **Real-World Testing and Validation:** Deploy the system in a real-world urban environment with limited parking capacity. The goal is to validate the system's effectiveness in managing parking allocation, optimizing resource use, and reducing congestion while testing its robustness, scalability, and performance under actual conditions.

4. DOCUMENT STRUCTURE

TODO: Write this chapter when all the document is finished.

5. METHODOLOGICAL FRAMEWORK

TODO: rewrite to adapt to this project

To address the research objectives outlined in this thesis, a structured methodological framework is adopted to ensure a comprehensive and rigorous development process. This chapter presents the methodological approach employed in this research, which is grounded in the V-model as established by the International Council on Systems Engineering (INCOSE) [11]. The V-model provides a robust framework for project development, facilitating timely and budget-compliant completion through a comprehensive development process.

The methodological framework is composed of seven key stages, each addressing a distinct aspect of the project development process. The stages are designed to ensure clear validation and verification of initial requirements at every stage, guaranteeing that the proposed solution aligns with the project's objectives.

1. **User Requirement Identification**: This stage involves a detailed analysis of the problem statement to identify primary issues and potential solutions. User requirements are defined to ensure that the proposed solution aligns with the project's objectives. This stage is crucial in establishing a clear understanding of the project's scope and requirements.
2. **System Design and Architecture**: Based on the identified user requirements, the system architecture is developed, ensuring that the proposed solution is feasible and aligns with the project's objectives. This stage involves a high-level overview of the system components and their interconnections. Requirements are formulated to satisfy the previously defined solution requirements.
3. **Component Design and Development**: Building upon the high-level architecture of the solution, a more detailed approach is outlined for each component, taking into account their specific power and data transmission needs. This stage culminates in a comprehensive architecture of the solution, providing a detailed overview of the components, their selection rationale, and their interconnections.
4. **Implementation and Manufacturing**: The proposed solution is implemented and manufactured using available tools, integrating the necessary electrical components. This stage includes a detailed description of the implementation process, including the tools and materials used, as well as the integration of electrical components. The development of software and hardware components is also detailed.
5. **Component Testing and Verification**: The functionality of each component is verified in a standalone mode, with detailed information provided regarding the verification process. This stage ensures that each component functions as intended, paving the way for system integration.
6. **System Testing and Integration**: The methodology for conducting system tests and subsequent analyses is elaborated. System integration is performed by assessing communication between module pairs to ensure that data can be transmitted freely and utilized effectively.
7. **Acceptance Testing and Validation**: Validation of the initial requirements is conducted to confirm that all solution requirements have been met. This stage also in-

cludes preparations for potential future enhancements, ensuring the solution's scalability and adaptability.

A graphical representation of the V-model is provided in Figure 5.1 to illustrate the methodology's structure and relationship between the various stages.

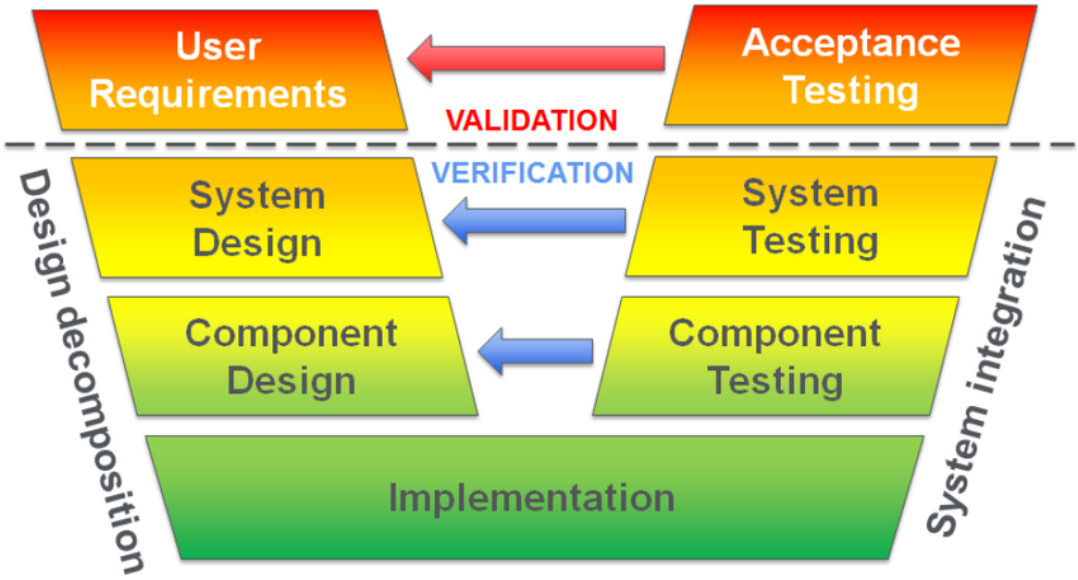


Figure 5.1. *Methodological framework based on the V-model from INCOSE with the different stages of the project development process[12]*

The V-model provides a structured approach to project development, ensuring that all aspects of the project are considered and addressed. By following this methodological framework, this research aims to develop a comprehensive and effective solution for the design and implementation of a fully distributed parking management system tailored for the next generation of smart cities.

Part II

Theoretical Background

6. CLOUD COMPUTING

Cloud computing is a technological paradigm that enables the delivery of computing services, including servers, storage, databases, networking, software, and analytics, over the internet. This model shifts the management of physical infrastructure and resources from on-premises systems to remote data centers managed by cloud service providers. Cloud computing is fundamental to modern digital ecosystems, supporting a wide range of applications from personal use to large-scale enterprise operations.

6.1 Definition and Key Concepts

Cloud computing leverages a network of remote servers hosted on the internet to store, manage, and process data, rather than relying on local computers or private data centers. It offers resources on-demand, enabling users to scale operations according to their needs. The essential characteristics of cloud computing include:

- **On-Demand Self-Service:** Users can access computing resources as needed without requiring human interaction with service providers.
- **Broad Network Access:** Services are accessible over a network, typically the internet, using various devices such as smartphones, tablets, and laptops.
- **Resource Pooling:** Resources are pooled to serve multiple users, ensuring efficiency and scalability.
- **Rapid Elasticity:** Resources can be elastically provisioned and released, enabling scalability according to demand.
- **Measured Service:** Cloud systems automatically control and optimize resource use, charging users based on their consumption.

6.2 Advantages of Cloud Computing

The adoption of cloud computing offers several advantages that have contributed to its widespread popularity. One of the most significant benefits is its scalability and flexibility. Cloud computing enables organizations to adjust their resource usage dynamically based on real-time needs, eliminating the need for over-provisioning and supporting workloads of varying demands effectively. This adaptability ensures that resources are utilized efficiently, minimizing waste and optimizing performance.

Cost efficiency is another critical advantage. By using cloud services, organizations can avoid the significant capital expenditures associated with purchasing and maintaining hardware and software. Instead, they incur operational costs only for the resources they consume, making cloud computing an economically viable solution for businesses of all sizes. Additionally, cloud computing enhances accessibility and collaboration. Since cloud-based applications and services are accessible from anywhere with an internet connection, distributed teams can collaborate seamlessly, a feature that has become indispensable in today's increasingly globalized and remote work environments.

Reliability is also a key strength of cloud computing. Leading cloud providers ensure high levels of availability by deploying redundant systems across geographically dispersed data

centers, thereby minimizing downtime and enhancing disaster recovery capabilities. Furthermore, security is a paramount concern addressed by cloud providers. They invest significantly in advanced measures such as encryption, firewalls, and regular security audits, ensuring robust protection against cyber threats and unauthorized access. These combined advantages make cloud computing a transformative technology, supporting innovation and operational efficiency in diverse sectors.

6.3 Types of Cloud Computing

Cloud computing services are categorized into different types based on their deployment models and the services they provide. These distinctions allow organizations to select a cloud strategy that aligns with their specific requirements.

6.3.1 Deployment Models

- **Public Cloud:** A public cloud is owned and operated by third-party providers, delivering resources over the internet. Examples include Amazon Web Service (AWS), Microsoft Azure, and Google Cloud Platform. It offers cost-effectiveness and scalability but may pose data sovereignty concerns for certain organizations.
- **Private Cloud:** A private cloud is used exclusively by a single organization. It offers greater control over data and infrastructure, making it suitable for industries with strict regulatory requirements.
- **Hybrid Cloud:** A hybrid cloud combines public and private cloud environments, enabling organizations to benefit from both scalability and control. This model is particularly useful for balancing workloads and maintaining sensitive data on-premises while leveraging the public cloud for other operations.
- **Multi-Cloud:** A multi-cloud strategy involves using multiple cloud providers to mitigate risks associated with vendor lock-in and enhance reliability and performance.

6.3.2 Service Models

Cloud computing services are typically offered in the following models:

- **Infrastructure as a Service (IaaS):** Provides virtualized computing resources over the internet, including servers, storage, and networking. Users manage operating systems and applications while the provider handles the infrastructure.
- **Platform as a Service (PaaS):** Offers a platform that includes hardware, software, and development tools to build, test, and deploy applications. It abstracts the complexities of infrastructure management, enabling developers to focus on coding.
- **Software as a Service (SaaS):** Delivers software applications over the internet on a subscription basis. Users can access the software through a browser without worrying about installation or maintenance.

7. INTERNET OF THINGS

The IoT is a paradigm that establishes a network of interconnected devices equipped with sensors, software, and communication technologies. This enables seamless interaction between the physical and digital realms, facilitating the collection, transmission, and processing of data. Through its ability to deliver enhanced functionality and automation, the IoT has numerous applications, particularly in smart city infrastructure. These applications span healthcare, transportation, energy management, and urban planning.

The main point of IoT is its capacity for interconnectivity, enabling devices to autonomously communicate and share data. Advanced sensing technologies augment this connectivity, capturing real-world metrics such as temperature, motion, or occupancy. The processing of the data is achieved through edge computing, which provides localized analysis, or cloud computing for centralized processing, tailored to meet latency and scalability demands. This processed data is subsequently integrated with actuators, enabling automation of tasks such as the management of parking space access without requiring human intervention.

7.1 Architecture of IoT Systems

The architecture of IoT systems is structured across four principal layers. The perception layer constitutes sensors and actuators, which interface directly with the physical environment to gather data and execute specific actions. Data transmission occurs within the network layer, leveraging communication protocols such as WiFi, Bluetooth, or cellular networks. The processing layer transforms raw data into actionable insights through edge or cloud computing methodologies. Finally, the application layer provides interfaces for end-users, ensuring an intuitive and accessible experience tailored to diverse functionalities.

7.2 Benefits of IoT

The adoption of IoT technologies confers significant advantages across multiple sectors, particularly in the realm of parking management systems. Real-time monitoring capabilities, enabled by sensors, deliver continuous updates on parking space availability, reducing search times and alleviating urban congestion. Automation, facilitated by actuators, streamlines operations such as access control and reservation management, minimizing human intervention. Moreover, IoT-driven data analytics empower urban planners with insights to optimize resource allocation and develop informed policies. The energy efficiency inherent in smart systems aligns with sustainability objectives by curbing unnecessary resource consumption.

7.3 Challenges and Limitations

The deployment of IoT systems presents several challenges. Scalability remains a critical concern, particularly in large-scale implementations like city-wide parking systems. The management of sensitive information collected by IoT devices necessitates data security and privacy measures to itigate risks. Interoperability challenges, stemming from

disparate manufacturer standards and communication protocols, complicate system integration.

Part III

State of the art

8. HISTORICAL DEVELOPMENT

TODO: Write this chapter

9. TYPES & TECHNOLOGIES

TODO: Write this chapter

10. MODERN TRENDS

TODO: write this chapter

Part IV

Methodology

11. REQUIREMENTS

The PMS for this project is designed to address the evolving needs of various users, including parking facility managers and drivers. The system's primary objective, detailed in Chapter 3, is to overcome the limitations of traditional parking management systems by improving scalability, reliability, and user adaptability, while aligning with the goals of smart city initiatives.

To achieve this, user and system requirements are identified. These requirements are essential for ensuring the system's effectiveness, efficiency, and alignment with the diverse needs of its users. The system's technical and functional specifications are also established to ensure its successful implementation.

11.1 User Requirements

The user requirements for the distributed PMS are determined through a thorough analysis of the needs and expectations of parking facility managers and drivers. These requirements are critical for designing a system that effectively addresses operational challenges, improves the efficiency of parking space utilization, and integrates seamlessly into existing urban mobility frameworks.

The primary users of the system, parking facility managers and drivers, have specific needs that the system is designed to meet. Their main objectives include improving parking management efficiency, ensuring the safety of the parking environment, and optimizing space utilization across multiple facilities. The system needs to enable them to monitor parking space availability in real-time, manage entry and exit of vehicles, and provide essential data for decision-making and reporting.

The requirements for the users can be broken down into the following key areas:

11.1.1 Real-Time Monitoring and Management

Parking facility managers need a system that provides real-time monitoring of parking spaces. This feature is essential for ensuring that parking spaces are used efficiently and that any underutilized spaces are identified and optimized. Real-time data to track the occupancy status of each parking spot, which is crucial for streamlining operations, reducing congestion, and improving the overall customer experience.

The system also allows parking facility managers to retrieve occupancy patterns and identify peak demand times, which helps adjust pricing strategies, optimize space allocation, and ensure parking spaces are available when needed.

11.1.2 Automation of Parking Access

Automation is another key requirement for the parking management system. Parking facility managers need a system that controls vehicle access to parking areas without requiring human intervention. Automated gates and barriers are essential for improving efficiency, reducing delays, and ensuring that vehicles can enter and exit parking facilities smoothly. These automated systems are integrated with real-time monitoring data to ensure that only authorized vehicles can access specific parking areas.

The system also provides tools for managing access rights based on specific criteria, such as license plate, membership status, or occupancy. Parking facility managers need to configure access control rules and monitor the effectiveness of these rules in real-time.

11.1.3 Security and Incident Management

Ensuring the security of parking facilities is a critical concern for parking facility managers. The system needs to incorporate security features to prevent unauthorized access to parking areas, detect potential security breaches, and provide timely alerts for incidents such as unauthorized parking or other violations. While surveillance systems are removed from the technical specifications, the cameras used for real-time monitoring still play a role in incident detection, as they capture images or video that can be reviewed in the event of an issue.

Real-time alerts are required for events such as access attempts by unauthorized vehicles, security breaches, or potential hazards. These alerts are sent to parking facility managers through the system's notification mechanisms, allowing them to respond quickly and effectively to incidents.

11.1.4 Data Analytics and Reporting

Parking facility managers need access to detailed reports and analytics on parking usage, including occupancy rates, peak demand times, revenue generation, and space utilization patterns. The system provides the ability to download the user data to peform reports, assisting in operational planning, resource allocation, and decision-making.

11.2 System Requirements

To meet the diverse user requirements, the distributed PMS is designed with several key system requirements in mind. These requirements focus on ensuring that the system is scalable, reliable, secure, and user-friendly, while also supporting the goals of smart city initiatives.

The main system requirements are as follows:

11.2.1 Real-Time Monitoring

The system needs to incorporate cameras for monitoring parking space occupancy in real-time. These cameras must be accurate, reliable, and capable of detecting vehicle presence or absence in parking spaces. The system must be able to process data from numerous cameras across multiple locations simultaneously, ensuring that the information provided to parking facility managers is accurate and timely.

Furthermore, the camera-based monitoring system must be robust enough to handle environmental factors such as weather conditions and lighting variations.

11.2.2 Automation and Access Control

Automated gate and barrier systems are a critical component of the system. These systems must be fully integrated with the parking management platform, allowing for real-time updates on space availability and enabling seamless access control. Vehicles must be able to enter and exit parking facilities autonomously based on the available space data, without the need for human intervention.

Access control mechanisms also need to ensure that only authorized vehicles can enter restricted areas. This can include integration with Licence Plate Recognition (LPR) systems, RFID-based solutions, or mobile applications that validate the identity of the vehicle and its access privileges.

11.2.3 Scalability and Flexibility

The system architecture needs to be scalable and flexible to accommodate a growing number of parking facilities, users, and devices. As demand for parking management grows, the system must be able to scale horizontally, adding additional cameras, sensors, or access control systems without impacting performance. Moreover, the system must be modular, allowing for easy updates or integration of new features.

11.2.4 Security and Availability

Given the importance of the system in managing urban mobility and infrastructure, security and availability are paramount. The system must be resilient to cyberattacks and other threats, with multiple layers of security in place to protect sensitive data and ensure continuous availability of services. Redundancy mechanisms, such as backup servers and data replication, are required to ensure high availability and minimize service interruptions. The system must be capable of maintaining operations even in the event of network outages or hardware failures. Interoperability is also a key consideration. The system must be designed to work seamlessly with existing infrastructure and technologies, both within the parking facilities and within the broader smart city ecosystem.

11.2.5 Physical System Requirements

In terms of hardware, the parking management system requires a robust and reliable physical infrastructure to support the real-time monitoring and automation functions. This infrastructure includes:

- **Cameras:** High-resolution cameras capable of detecting vehicle occupancy in parking spaces. These cameras must be weather-resistant, durable, and equipped with night vision or infrared capabilities to ensure reliable performance in various environmental conditions, such as low light or harsh weather.
- **Gate and Barrier Systems:** Automated gates and barriers for controlling vehicle access to parking areas. These systems must be seamlessly integrated with the overall parking management system to allow for real-time updates on space availability and facilitate the automated entry and exit of vehicles.
- **Network Infrastructure:** A robust communication system to transmit data from cameras, sensors, and access control devices to central servers. The network infrastructure must support high-speed data transfer and be resilient to failures.

The physical infrastructure must be designed for scalability and ease of maintenance, allowing for the expansion or upgrade of components as needed.

11.2.6 Data Privacy and Storage

Data privacy and security are paramount concerns in the design of the parking management system. The system must comply with relevant data protection laws, such as the General Data Protection Regulation (GDPR) [13], to safeguard personal information and prevent unauthorized access to sensitive data.

The data collected by the system, including parking history, vehicle entry and exit times, and payment details, must be encrypted both during transmission and while stored in the system's databases. Key data privacy and storage requirements include:

- **Secure Storage:** All personal and parking-related data should be stored in secure, encrypted databases with access restricted to authorized personnel only.
- **Data Retention:** The system must adhere to strict data retention policies, ensuring that data is stored only for the minimum time necessary to fulfill the purpose of the service, in accordance with relevant privacy regulations.
- **User Consent:** The system must obtain explicit consent from users for data collection, explaining the types of data collected and how it will be used, stored, and shared.
- **Access Control:** The system must include role-based access control mechanisms to ensure that only authorized personnel can access sensitive data, and it must provide audit logs to track data access and modifications.

In addition to ensuring compliance with privacy regulations, the system must include robust mechanisms for data breach detection and response, ensuring that any unauthorized access or data compromise is identified and addressed promptly.

12. DESIGN

This chapter outlines the architecture of the system design, detailing the different subsystems and components to meet the requirements detailed in Chapter 11. The system designed is composed of two main components, a server, in charge of manageming the users, authenticating them, and relaying the information form the users, and the terminals, which are the brains of the system and hold the main control for each community. It is worth to note that while the terminal is deployed in every community, the server is only deployed once and can be used for every user in the system. The architecture of the system can be seen in **TODO:** add image with architecture. Note that there are other complementary elementes that they will be later explained.

TODO: add image of architecture and explain

12.1 Terminals

The terminals are the core part of the deployment and is where all the processing is done to manage each community. In order to provide the necesary capabilities for the community such as opening the garage doors and detecting the cars the following components are needed.

12.1.1 Cameras

Two cameras per door to be able to detect the cars when they are close to the garage door. This cameras are the ones in charge of sending live video to be able to detect the cars and later identify the licence plates. The requirements for the cameras are that they are wether proof and can withstand heavy rain, can send live feed and be connected to a central computer, and have support for infrared detection for low light events such as inside garage parkings or night. Different models of cameras where studied but the final camera used for this project was the **TODO:** add camera because of **TODO:** add reason

TODO: add image of camera

Moreover, even thought the camera has infrarred capabilities, the infrarred light of the camera is pretty weak. Therefore an anditional infrared led was used to increase the quality of the image at night. Also the good part of this camera is their relative low price point and that they can be powered with Power over Ethernet (PoE), that is, they can be power and connected to the networking with just one cable.

TODO: add infrared image and description

12.1.2 Computer

For the computer, the main requirements where that it would need to have the necessary computing power to run the detection algorithm to detect the different vehicles and licence plates for each camera and the price point is low enough as there will be one in every community. This restiction to be able to run the detection algorithm limits to have a onboard GPU to run the models. For this, the Raspberry Pi family [14] was discarded due to the limitation of not having an onboard GPU. This leads to only be able to use a product of the NVIDIA Jetson family [15], as this are on board computers designed to be used with Machine Learning (ML) models.

For this project, the NVIDIA Jetson Nano of 4 GBs was used as it offered the best price performance ratio and was sufficient enough to perform the computing requirements.

12.1.3 Internet Connectivity

In order to provide conectivity from the on-board computer to the server, a networking connection is needed to be established. As this deployment must be self sufficient and not depend on the infrastructure of each community, a router with 4G connection is deployed. For this case, a conventional router is used, the **TODO:** add router. Moreover, in order to provide the connectivity for the different cameras and the computer, a switch was used. The switch was the **TODO:** add switch.

TODO: add photo of switch and router

12.1.4 Rele Extension board

In order to trigger the mechanism to activate the opening of the garage door of each community, this is done by means of a rele connected in paralel to the door system. The opening doors function with a line where a pull-down signal is sent whenever the mechanism wants to be activated. In order to perform this operation, a Rele Extension board was used for the NVIDIA Jetson. The reason is that after trials, the on-board GPIO were not strong enought to send a constant signal for some specific communities and the mechanism malfunctioned.

The Rele Extension board used was the **TODO:** add extension due to the low price-point and the easy access to buy.

12.2 Server

The server is the point of contact to connect the users and the different communities. It allows the users to authenticate and connect to the sepcific community. It is based on a scalable archtutecture deployed in AWS due to the fast time-to-market capabilities and the scalability of it. The architecture used for this implementation is based on an Autoscaling Group deployed in a private VPC. An autoscaling group is used mainly to adapt to the load of the incoming users dinamically and be able to recover from errors. In order to allow connections from the different users, an Application Load Balancer is used. This distributes the load into the different instances deployed.

Inside each instance, a web server is isntalled with the content of the website is shown to the users. The framweork used is the NextJS **TODO:** add ref as it is a React framwork used to build full-stack web applications. This allows to embed the website in mobile applications for iOS and Android in an easy and low mantainance manner. Check the **TODO:** add ref for the implementation to see more information.

12.3 Networking

In order to connect the terminals with the server, a ZeroTier **TODO:** add ref is used. This enableds to connect all the different terminals and server in a straighforwards and easy way on a single network. Moreover, it allows to connect remotely to the terminals to do mainenance or updates in the system.

12.4 Monitoring

In order to montior the different terminals and check that they are functioning correctly, an AWS Managed Prometheus is used to gather the data of the different users and a AWS Managed Grafana is used to visualize and make alerts.

12.5 Deployment

Finally, for the deployment, in order to automate the deployment in the multiple terminals two tools are used. Github Actions is in charge of building the packages for the different servers and software used which are later defines, and also Ansible is used to automate the instalation of the new software in the different devices.

13. IMPLEMENTATION

Give the reqruiements and the design in the previous chatpers, refer to Chapter 11 and Chapter 12, the different modules are implemented following the V-model established in Chapter 5.

13.1 Community deployment

In order to deploy a new system in a new community the following elements must be installed

TODO: add architecture

A terminal in charge of the processing of all the incoming data from the different elements and has the algorithm to open the gates when a allowed user enters or exists. The cameras already explained in the Chapter 12, and the networking system that provides connection to the central server and allows users and manager to connect to the specific commuity and administer it.

13.1.1 Terminals

As previously stated, the terminals are the ones in charge of the main functionality of the parking management system, storing the data, detecting the vehicles, and allowing the parking management maangers administer the community. For this the following subsystems are design, a database to store the data locally, a server to expose the service externally, and a detection algorithm to detect the vehicles. Moreover, other subsystems are also implemented, such as a backup system, and monitoring system.

TODO: add architecture of the terminals

database

In order to store the data of each community, a database is used. The options are between SQL or NoSQL database and a local database in the terminal or a centralized database in the cloud. For this case, a local SQL database based on SQLite **TODO:** add ref is used. The main reason behind it was due to the requirements of having the data inside the community for data privacy and security reasons.

Moreover, while it is true that using a centralized database would be easier for maintenance, backups, and upgradability, the requirement of high availability would requier to store a cache database portion in each terminal in order to make the database robust to internet connection losses. Using a local database we ensure that the systems keeps working even if internet connection losses occur. The drawback using a local database is that we have to ensure to backup the database periodically to meet Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO) so no critical data is lost.

The schema used for the dataset is the following.

TODO: add schema image and explain

Terminal Server

To provide the capabilities for the server to communicate to the terminal, a local server is developed. This server will have all the require actions that a user can perfom to administer and use the system. The server used is a ExpressJS **TODO:** add ref server and based on a RestAPI methdology. The reason to use this is that ExpressJS is a widely used framework with a wide adoptions and multiple libraries to perform different tasks such as database management and more.

The RestAPI methodoloy **TODO:** explain

The following different edpoints are used: **TODO:** add endpoints

Note that in order to authenticate the users, a custom JWT token is used that will be later explained in **TODO:** add section explained.

Detection Algorithm

In order to detect the different licence plates of the vehicles, a custom deep learning algorithm is used based on YOLO **TODO:** add ref. The implementation of this model is out of the scope of this project but more information can be found in the following paper https://www.sciencedirect.com/science/article/pii/S0921889023002476?via

In summary the system has two main steps, one step is to detect a licence plate in the image, it creates a bounding box of the licence plate that is later processed by a second algorithm to detect the characters of the licence plate.

TODO: add images

Other subsystems

The cameras are used to send the image captured in the different points of the community, usually places in the gate doors to the terminal. The terminal processes the data with the detection algorithm. The data is sent through an RTSP **TODO:** add ref and gls protocol and connected via Ethernet to the network.

In order to connect the cameras with the terminal and the terminal to the internet, the router with a switch is used. The specific router is a 4G router to povide connectivity to the terminal.

Finally, to ensure the correct functionality of them, a backup system that backups up the database every day to Amazon S3 **TODO:** ref encrypted, and also a monitoring system that sends metrics of the terminal to a centralized server in oder to keep track of the different terminals and detect if a specific terminal is malfunctioning.

13.2 Cloud deployment

To enable the access to the different communities, a web server is deployed in AWS. The main component is a NextJS server that holds the website that the users use to access the system and also the backend functionality to connect to each specific community.

13.2.1 Cloud Architecture

In order to deploy a scalable and robust application the following arhictecture is used.

add cloud architecture and explain

Note that this deployment is based on EC2 instances as the use of ZeroTier requried to have virtualmachines installed as it cannot be install in containers. The main probelm is that the Zerotier docker installation needs special network admin permissions https://github.com/zyclonite/zerotier-docker, which ECS **TODO:** add ref does not allow. Also a budget architecture was needed to keep the cost low.

Moreover, all hte deployment was built and administer using Terraform **TODO:** add ref. The reason was to have a description of the deployment used and be able to make changes.

13.2.2 Website

The website was design to be responsive to be able to be used in mobile devices and computer devices. The website is built using the React framework **TODO:** ref to provide the functionality and responsive of a modern website. The website also uses TailwindCSS **TODO:** ref for a ascethic and mother look. Check out **TODO:** add image.

Moreover, the website is embeded in Progressive Web Apps in Android and iOS to allow users to download an application through the mobile store of choice and be able to use the website in their phones. **TODO:** add image of mobiles.

13.2.3 User Authentication

In order to authenitcate the users and be able for the users to sign in, the requirement was that a phone SMS based system was used to make it easier and more secure for the users to sign in. for this a Firebase Auth **TODO:** add ref was used. This allows the application to send sms to phone numbers of the users to authenticate. Also it sends a JWT token to the server that is later used by the terminal to authenicate the users and check the permissions.

13.3 Deployment

In order to perform updates to the system that is upgradeing and making software changed for the different terminals, an automatic deployment is design. This system is based on Github Actions, to build the NPM packages to be installed on the temrinals. This is done buy building github action workflows that build the package with the necesarry dependencies and are stored in GitHub packages for later used. It is worth to note that the creation of the package is done automaticically when a new version is created using tags in the Git system.

Once the package is build, Ansible is used, **TODO:** explain ansible. This is done by creating a playbook and setting the required actions. Moreover a inventory is used, this holds all the terminals that Ansible will connect and update.

TODO: show the tasks done

It is worth noting that the migration of the database are done manually for the moment, as no straightforward way and a way that ensures the database integrity was design yet, but this is an area of current development.

TODO: extend

13.4 Monitoring

In order to monitor the different terminals in case of errors, The prometheus deployment is used to store the metrics. Custom scripts are done to send information to the system. Once the information is stored in prometheus, a aws managed grafana is used to view and send allerts

TODO: extend

14. TESTING

TODO: write this chapter

add the problem with bad connection and trying different sims

Part V

Results

Part VI

Conclusions

15. CONCLUSIONS

TODO: write this chapter

16. FUTURE WORK

TODO: write this chapter

add that to change the db to a nosql for the users and keep the sql for the records

use prisma for the db

17. SOCIO-ECONOMIC ENVIRONMENT

TODO: write this chapter

18. REGULATORY FRAMEWORK

The regulatory framework governing smart cities in Europe is primarily based on the data protection provisions outlined in the GDPR. This is largely because these systems employ surveillance technologies such as cameras that monitor public spaces, and they handle sensitive personal data, including individuals' identification numbers and contact information. Additionally, the integration of technologies such as the IoT, Artificial Intelligence (AI), and data analytics has necessitated robust legal mechanisms to address privacy, security, and ethical considerations.

18.1 Applicable Legislation

The implementation and operation of smart city technologies in the European Union must adhere to a range of legislative measures that regulate data privacy, electronic communications, and emerging technologies. Chief among these is the GDPR, but other complementary regulations and directives are also critical in shaping the regulatory landscape.

18.1.1 General Data Protection Regulation (GDPR)

The GDPR [13], also known as the Regulation (EU) 2016/679 and enacted in 2016, provides a comprehensive framework for the protection of personal data within the European Union. Its principles of transparency, accountability, and data minimization serve as the cornerstone of privacy and data protection.

Under GDPR, smart city systems are required to obtain explicit consent from individuals before collecting or processing their personal data. Furthermore, data controllers must implement privacy-by-design and privacy-by-default principles, ensuring that data protection is an integral aspect of technological development. Smart cities must also conduct Data Protection Impact Assessments for projects involving high-risk data processing, such as large-scale surveillance or facial recognition.

18.1.2 Data Retention Policies

Smart city technologies often rely on continuous data collection through sensors, cameras, and connected devices. The GDPR mandates that data should only be retained for as long as necessary to fulfill its intended purpose. To comply, smart city operators must establish robust data retention and deletion policies. Automated systems are frequently employed to enforce these policies, ensuring that unnecessary data is securely deleted to minimize privacy risks and reduce storage costs.

18.1.3 ePrivacy Directive

The ePrivacy Directive (Directive 2002/58/EC) [16], commonly known as the "Cookie Law," governs the confidentiality of communications and the processing of electronic communications data. While initially designed for traditional telecommunications, its principles are increasingly relevant to smart city applications, such as real-time traffic monitoring and public Wi-Fi networks.

Under this directive, consent is required for the use of tracking technologies and location-based services, ensuring that users are aware of how their data is being utilized. The on-

going transition to the ePrivacy Regulation aims to enhance these protections and provide greater harmonization across member states.

18.1.4 Directive on Security of Network and Information Systems (NIS 2 Directive)

The Network and Information Systems (NIS) 2 Directive [17], adopted in 2016, focuses on the security of critical infrastructure and essential services, including those used in smart cities. It requires operators of essential services to implement risk management measures and report significant cybersecurity incidents to relevant authorities. For smart cities, this includes safeguarding critical systems such as transportation networks, energy grids, and communication systems against cyber threats.

18.1.5 Artificial Intelligence Act

The Artificial Intelligence Act [18], which came into force on 1 August 2024, establishes a common regulatory and legal framework for AI across the European Union. Its provisions will be implemented gradually over the following 6 to 36 months. The Act classifies AI systems into categories based on risk, ranging from minimal to unacceptable, and imposes specific requirements on high-risk AI applications.

For smart cities, the Act directly impacts technologies such as AI-driven surveillance systems, autonomous transportation, and automated decision-making tools. These applications must adhere to stringent requirements, including transparency, accountability, and mandatory human oversight. The Act ensures that AI technologies in smart cities are deployed ethically and align with EU values, particularly with respect to fundamental rights and safety.

18.2 Regulatory Challenges and Implications

The evolving nature of smart city technologies presents significant challenges for regulatory compliance. These challenges include the integration of multiple technologies such as IoT, AI, and data analytics, which often involve cross-border data flows and raise questions about jurisdiction and enforcement. Furthermore, the pace of technological innovation frequently outstrips the development of regulatory frameworks, creating gaps that may expose users to privacy and security risks.

Ensuring compliance with a wide array of legislative measures requires substantial investment in training and capacity-building for smart city administrators. Policymakers and industry stakeholders must collaborate to establish clear guidelines that address emerging issues while balancing innovation and the protection of fundamental rights.

BIBLIOGRAPHY

[1] Irene Lebrusán and Jamal Toutouh. "Assessing the Environmental Impact of Car Restrictions Policies: Madrid Central Case". In: *Smart Cities*. Ed. by Sergio Nesmachnow and Luis Hernández Callejo. Cham: Springer International Publishing, 2020, pp. 9–24. ISBN: 978-3-030-38889-8.

[2] Raquel Fernández-González et al. "Urban mobility trends and climate change: sustainability policies in the parking industry". In: *Environmental Science and Pollution Research* 30 (May 2023), pp. 1–14. DOI: 10.1007/s11356-023-26925-2.

[3] A. Ibeas et al. "Modelling parking choices considering user heterogeneity". In: *Transportation Research Part A: Policy and Practice* 70 (2014), pp. 41–49. ISSN: 0965-8564. DOI: https://doi.org/10.1016/j.tra.2014.10.001. URL: https://www.sciencedirect.com/science/article/pii/S0965856414002341.

[4] Fredrik Dahlqvist et al. "Growing opportunities in the Internet of Things". In: *McKinsey & Company* 22 (2019).

[5] Symeon Papavassiliou Eirini Eleni Tsiropoulou John S. Baras and Surbhit Sinha. "RFID-based smart parking management system". In: *Cyber-Physical Systems* 3.1-4 (2017), pp. 22–41. DOI: 10.1080/23335777.2017.1358765. eprint: https://doi.org/10.1080/23335777.2017.1358765. URL: https://doi.org/10.1080/23335777.2017.1358765.

[6] M. Venkata Sudhakar et al. "Development of smart parking management system". In: *Materials Today: Proceedings* 80 (2023). SI:5 NANO 2021, pp. 2794–2798. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2021.07.040. URL: https://www.sciencedirect.com/science/article/pii/S2214785321048926.

[7] H. Al-Kharusi and I. Al-Bahadly. "Intelligent Parking Management System Based on Image Processing". In: *World Journal of Engineering and Technology* 2 (2014), pp. 55–67. DOI: 10.4236/wjet.2014.22006.

[8] Amira A. Elsonbaty and Mahmoud Shams. "The Smart Parking Management System". In: *International Journal of Computer Science and Information Technology* 12.4 (Aug. 2020), pp. 55–66. ISSN: 0975-4660. DOI: 10.5121/ijcsit.2020.12405. URL: http://dx.doi.org/10.5121/ijcsit.2020.12405.

[9] Adel Mounir Said, Ahmed E. Kamal, and Hossam Afifi. "An intelligent parking sharing system for green and smart cities based IoT". In: *Computer Communications* 172 (2021), pp. 10–18. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2021.02.017. URL: https://www.sciencedirect.com/science/article/pii/S0140366421000864.

[10] Paul Melnyk, Soufiene Djahel, and Farid Nait-Abdesselam. "Towards a Smart Parking Management System for Smart Cities". In: *2019 IEEE International Smart Cities Conference (ISC2)*. 2019, pp. 542–546. DOI: 10.1109/ISC246665.2019.9071740.

[11] International Council on Systems Engineering. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. 4th. Hoboken, NJ: Wiley, 2015. URL: https://www.incose.org/products-and-publications/se-handbook.

[12] Alastair Ruddle. *Simplified System Engineering 'V' Model*. https://www.researchgate.net/figure/Simplified-system-engineering-V-model_fig1_338672391. Accessed: 2024-10-15. 2020.

[13] General Data Protection Regulation GDPR. "General data protection regulation". In: *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC* (2016).

[14] *Raspberry Pi*. https://www.raspberrypi.com/. Accessed: 2024-11-20.

[15] *Jetson Modules, Support, Ecosystem*. https://developer.nvidia.com/embedded/jetson-modules. Accessed: 2024-11-20.

[16] Council of European Union. *Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector*. http://data.europa.eu/eli/reg/2002/58/oj. 2002.

[17] Council of European Union. *Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union*. http://data.europa.eu/eli/reg/2016/1148/oj. 2016.

[18] Council of European Union. *Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence*. http://data.europa.eu/eli/reg/2024/11689/oj. 2024.