

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Towards a Smart City Digital Twin Methodology

Bernardo Dinis Guimarães Pinto



Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Rosaldo Rossetti

Second Supervisor: Prof. Ana Paula Rocha

Third Supervisor: Iohan Soares

July 30, 2026

Towards a Smart City Digital Twin Methodology

Bernardo Dinis Guimarães Pinto

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. João Correia Lopes

Examiner: Prof. Rui Gomes

Supervisor: Prof. Rosaldo Rossetti

July 30, 2026

Resumo

Os *Digital Twins* (representações virtuais de objetos, sistemas ou processos) podem ser aplicados ao contexto urbano para apoiar o planeamento de serviços como o trânsito, a energia, a saúde ou a segurança. As soluções atuais tendem, no entanto, a focar-se em serviços isolados, não sendo concebidas para integrar múltiplos domínios numa única plataforma, o que limita o potencial do conceito quando aplicado a cidades inteligentes.

Nesta dissertação propomos uma arquitetura que assegura a interoperabilidade entre módulos de simulação distintos e uma interface web unificada, permitindo configurar cenários, executar simulações e analisar resultados a partir de um único ponto de entrada. Sobre esta arquitetura foram integrados dois serviços: um módulo de trânsito, que permite simular o impacto de cortes de vias na rede viária, e um módulo de veículos elétricos, que otimiza a localização e configuração de postos de carregamento através de um algoritmo multi-objetivo, considerando o custo de infraestrutura e o tempo médio de espera.

Os resultados obtidos no Porto mostram que a arquitetura permite avaliar de forma integrada o impacto de intervenções na rede de trânsito (incluindo o efeito em vias alternativas e no transporte público) e identificar configurações eficientes de infraestrutura de carregamento em cenários com múltiplos objetivos. A replicação da simulação de trânsito em Lisboa, através da simples substituição dos inputs, confirmou que a solução é adaptável a diferentes cidades sem necessidade de alterar os módulos de simulação. Estes resultados validam a arquitetura proposta como uma abordagem viável para a integração de múltiplos serviços urbanos num único *Digital Twin*, contrariando a tendência de soluções isoladas identificada na literatura, e abrem caminho à extensão da plataforma a outros domínios urbanos.

Palavras-chave: Gémeos Digitais • Cidades Inteligentes • Trânsito • Veículos Elétricos • FIWARE

Abstract

Digital Twins (virtual representations of objects, systems, or processes) can be applied to the urban context to support the planning of services such as traffic, energy, healthcare, or public safety. Current solutions, however, tend to focus on isolated services and are not designed to integrate multiple domains within a single platform, which limits the potential of the concept when applied to smart cities.

In this dissertation, we propose an architecture that ensures interoperability between distinct simulation modules and a unified web interface, allowing users to configure scenarios, execute simulations, and analyse results from a single entry point. On top of this architecture, two services were integrated: a traffic module, which enables the simulation of the impact of road closures on the road network, and an electric vehicles module, which optimises the location and configuration of charging stations through a multi-objective algorithm, considering infrastructure cost and average waiting time.

The results obtained in Porto show that the architecture makes it possible to jointly assess the impact of interventions on the traffic network (including their effect on alternative roads and public transport) and to identify efficient charging-infrastructure configurations in multi-objective scenarios. The replication of the traffic simulation in Lisbon, through the simple replacement of the inputs, confirmed that the solution is adaptable to different cities without requiring changes to the simulation modules. These results validate the proposed architecture as a viable approach for integrating multiple urban services within a single Digital Twin, countering the trend of isolated solutions identified in the literature, and pave the way for extending the platform to other urban domains.

Keywords: Digital Twins • Smart Cities • Traffic • Electric Vehicles • FIWARE

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for achieving a better and more sustainable future for all. This dissertation contributes to these goals by presenting a platform capable of integrating multiple urban domains, helping stakeholders to better plan and assess city infrastructure through simulation-based analysis, while also supporting future extension to additional urban services and different city contexts.

This work contributes to the following UN Sustainable Development Goals:

SDG 11 Make cities and human settlements inclusive, safe, resilient and sustainable

SDG 7 Ensure access to affordable, reliable, sustainable and modern energy for all

SDG	Target	Contribution	Performance Indicators and Metrics
11	11.2	Supports the evaluation of sustainable urban mobility scenarios, including road closures and lane restrictions, and their effects on network performance.	Simulated traffic volume. Average speed. Network congestion index over time.
7	7.2	Supports the integration of electric vehicles in the city by enabling the evaluation of alternative charging-infrastructure configurations.	Total infrastructure cost. Average waiting time. Charging sessions total.

Acknowledgements

At the end of this long journey, I would like to thank Professor Rosaldo Rosetti for accompanying and guiding me throughout this project, as well as Professor Ana Paula Rocha and Iohan for your feedback over these past months, which helped me complete this work.

I would also like to thank my entire family for helping me become the man I am today. I will always be grateful to you for supporting and standing by me at every stage of my life, from the moment I was born to the day I complete this chapter.

To my great friends from the course, Eduardo, Máximo, and Francisco, my heartfelt thanks for always being by my side throughout these five years, during which you helped me overcome many obstacles along the way.

Finally, I would like to thank all my friends, from those I met during the course to those whose friendship has lasted for more than a decade, for all the moments we have shared and those we will share in the future.

Bernardo Pinto

Declaração de Honra

Declaro que a presente dissertação é de minha autoria e não foi previamente apresentada e avaliada nesta ou em outra instituição. As referências a outros autores (afirmações, ideias, pensamentos), assim como a utilização de trabalhos publicados respeitam escrupulosamente as regras da atribuição, e encontram-se devidamente indicadas no texto e nas referências bibliográficas, de acordo com as normas de referência. Tenho consciência de que a prática de plágio ou de autoplágio constitui ilícitos académicos, conforme estabelecido no Código de Ética da U.Porto.

Declaro, ainda, que utilizei ferramentas de Inteligência Artificial apenas como apoio à revisão e melhoria da escrita da presente dissertação. A definição dos conteúdos, argumentos, metodologia, análise e conclusões é da minha autoria. A utilização destas ferramentas e os respetivos resultados foram devidamente verificados, de modo a assegurar a correção, coerência e fidelidade ao trabalho desenvolvido.

Bernardo Dinis Guimarães Pinto



Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	1
1.3	Problem	2
1.4	Research Questions	2
1.5	Document Structure	2
2	Literature Review	4
2.1	Review Protocol	4
2.1.1	Search Queries	4
2.1.2	First Results	5
2.1.3	Filtering Results	6
2.1.4	Search Results	6
2.2	Related Work	9
2.2.1	Conceptual Frameworks	9
2.2.2	Digital-Twins for traffic	11
2.2.3	Digital-Twins for Healthcare	15
2.2.4	Digital-Twins for Public Safety	16
2.2.5	Digital-Twins for Energy	19
2.2.6	Digital-Twins for Water Management	21
2.2.7	Digital-Twins for Pollution	22
2.2.8	Digital-Twins as a Multi-Agent System	23
2.2.9	Adapting Game Models into a Digital-Twin	23
2.3	Main Findings	24
3	Methodology	27
3.1	The Proposed Architecture	27
3.2	Assessment Method	28
3.3	Practical Implementation	28
3.3.1	FIWARE	28
3.3.2	Transit Module	32
3.3.3	Electric Vehicles Module	35
3.4	Web-Visualization Platform	39
3.5	Chapter Summary	40
4	Results and Evaluation	41
4.1	Experimental Setup and Objectives	41
4.2	Testing Transit Module	42

4.3	Testing EV Module	48
4.4	Testing the Platform in Another City	54
4.5	Results Summary	54
4.6	Towards a Theoretical Methodology	56
4.7	Chapter Summary	57
5	Conclusion	58
5.1	Dissertation's Contributions	58
5.2	Limitations	59
5.3	Future Work	59
	References	61

List of Figures

2.1	Literature review process	8
2.2	Traffic management architecture	14
2.3	Smart Health architecture	17
3.1	Proposed architecture	29
3.2	Simulation process	34
3.3	EV optimization process	38
3.4	Home page	39
4.1	Normal porto traffic	43
4.2	Blocked roads in the dashboard	44
4.3	Comparison of traffic conditions at peak hour for the baseline scenario (top) and for the scenario with the three selected links closed (bottom)	45
4.4	Comparison of traffic volume with 20% of the full population (top) and full population (bottom)	47
4.5	Live traffic	48
4.6	Galp real locations	49
4.7	Galp locations at the iterative map	50
4.8	Configuration used	50
4.9	Solution example	52
4.10	EV station with some vehicles	53
4.11	EV station with queued vehicles	53
4.12	First results, normal simulation	55
4.13	Results with the blocked road	55

List of Tables

2.1	Inclusion criteria in the search for all queries	5
2.2	Search results in different platforms	6
2.3	Inclusion criteria for the first query	6
2.4	Exclusion criteria for the first query	7
2.5	Inclusion criteria for the second query	7
2.6	Exclusion criteria for the second query	7
2.7	Inclusion criteria for the third query	7
2.8	Exclusion criteria for the third query	8
2.9	Summary of related studies	25
4.1	Top three solutions obtained in the first experiment	51
4.2	Top three solutions obtained in the second experiment	51

List of Acronyms

AI	Artificial Intelligence
CLI	Command Line Interface
DL	Deep-Learning
DT	Digital Twin
ICT	Information and Communication Technologies
IoT	Internet of Things
LPR	License Plate Reader
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAS	Multi-Agent System
ML	Machine Learning
R^2	R-Squared
VCI	Via de Cintura Interna
VR	Virtual-Reality

Chapter 1

Introduction

In this chapter it is presented the context and motivation of this project, alongside the problem definition and research questions.

1.1 Context

City planners have a difficult task in planning a new city capable of meeting the requirements of the population. There are many variables that city planners must take into account so that the new planned city is sustainable, efficient, and improves the quality of life of its population. Relying only on traditional methods, such as calculations to estimate the city's needs for services like transportation and healthcare, makes it challenging to accurately predict future demands.

To address these limitations, the concept of smart cities has emerged, which can be defined as a city that is integrated with Information and Communication Technologies (ICT) to coordinate and enhance traditional infrastructures through digital technologies [29]. Data collected from various urban sources can be used to analyze, predict, and optimize city operations. Real-time data can be collected with sensors, leveraging the Internet of Things (IoT), and integrating existing urban data platforms, making it possible to create a digital representation of the city, known as Digital Twin (DT). This virtual model continuously receives real-time data and enables the simulation of different scenarios, allowing city planners to make more informed and data-driven decisions. In addition, it can be used to simulate unexpected events, such as earthquakes, in order to see the response from the city services in such extreme events.

1.2 Motivation

The use of DT can be a giant step towards the efficiency of a city, saving potential costs and time infrastructure planning and development. By simulating different scenarios, city planners can identify potential issues and better allocate their budget towards city services, in order to improve the city sustainability, efficiency and quality of life.

Despite this ongoing evolution on city planning, current approaches on this theme lack scalability and are focused on isolated systems [13]; therefore, there is no widely adopted or standardized methodology that is used when creating a digital twin. To build this methodology, it is essential to identify the truly critical services that are going to be integrated in the DT, so that the resulting model accurately represents the city's most important services, providing valuable information and, consequently, helping city planners to make better informed decisions.

Furthermore, implementing DTs can bring many other benefits besides operational efficiency. It enables real-time monitoring of a city, since the data is synchronized between the physical and virtual environment, allowing systems to be optimized dynamically. This synchronization enables the early identification of potential failures, allowing maintenance teams to take preventive actions towards possible issues, improving the system's reliability [31].

1.3 Problem

Since the concept of smart cities is relatively new, currently there is no adopted methodology/framework when making a new DT of a specific city. With no standard frameworks, when a stakeholder wants to create a DT of a city, they face a lack of guidance, resulting in redundant development efforts. Consequently, developers frequently 'reinvent the wheel' by building custom architectures that are costly to maintain and difficult to replicate in other cities, since each city has his own characteristics and infrastructure.

Therefore, there is a need for a methodology that guides DT developers, supporting scalability, interoperability, and the integration of heterogeneous data sources. This methodology should also account for multiple urban domains and services (e.g., transportation, energy) and remain adaptable to the city's evolution over time.

1.4 Research Questions

Based on the challenges identified in the Motivation and Problem sections, the following questions can be formulate to guide this research:

1. What are the examples of methodologies to design DT of smart cities, that integrate scalability, sustainability and interoperability?
2. Which urban services are critical to smart cities that should be included in the DT?
3. How can data (either historical or real-time) obtained from different sources (e.g., sensors, IoT, existing platforms) be effectively integrated in the DT framework developed?

1.5 Document Structure

In addition to this introduction, the remainder of this document is formed by other four chapters, as follows. Chapter 2 presents the literature review, covering the main concepts and related work

relevant to this dissertation. Chapter 3 describes the proposed solution, including its overall design, main components, and implementation approach. Chapter 4 presents the experimental evaluation, describing the tested scenarios and analysing the results obtained. Finally, Chapter 5 concludes the dissertation by summarising the main contributions, discussing the limitations of the work, and identifying possible directions for future development.

Chapter 2

Literature Review

2.1 Review Protocol

In this section, it will be presented how the review was done, from the search strings that were used and why, to the selection of papers and its respective reason.

2.1.1 Search Queries

For this process, our goal is to extract and define the key concepts presented in each research question, in order to retrieve the maximum number of relevant documents. Although the research questions have a common topic, which is digital twins, they each differ in the aspects targeted inside the theme digital twins, so the goal is to make a search query for each research question presented.

In the first query, the target is to find methodologies that already exist in the context of digital twins or smart cities, that can integrate scalability, sustainability and/or interoperability.

```
("smart cities" OR "smart city")  
AND ("digital twin" OR "digital twins")  
AND (methodology OR method OR methods)  
AND (scalability OR sustainability OR interoperability  
OR integrat*)
```

The term "framework" was initially added to the query and a lot more results were retrieved, however, this term included makes the search more broad, since some papers refer framework as "smart cities framework", rather than specifically focusing on methodologies directly related to the implementation or integration of digital twins in smart cities.

For the second query, we focused on identifying which urban services are critical for smart cities and should be incorporated into the digital twin.

```
("smart city" OR "smart cities")  
AND ("digital twin" OR "digital twins")
```

Table 2.1: Inclusion criteria in the search for all queries

#	Inclusion Criteria
1	Published from 2020 to 2025
2	Full articles
3	It is a conference paper or in a journal
4	Text published in english

```
AND ("urban service?" OR "city service?" OR transport*
OR energy OR health* OR mobility OR water OR emergenc*)
AND (critical OR essential OR "key service?")
```

In this search query, it was included the most comprehensive services, since if it was only ("urban service?" OR "city service?") instead of ("urban service?" OR "city service?" OR transport* OR energy OR health* OR mobility OR water OR emergenc*), much fewer results would be retrieved.

For the third query, the goal is to know how can data (either historical or real-time) obtained from different sources (e.g., sensors, IoT, existing platforms) be effectively integrated in the DT framework developed. For this one, the key terms were already really specified in the question, leading to the following query:

```
("smart city" OR "smart cities")
AND ("digital twin" OR "digital twins")
AND ("historical data" OR "real-time data")
AND (sensor* OR IoT OR "urban platform?" OR "existing platform?")
AND integrat* AND (framework OR methodology)
```

2.1.2 First Results

To apply this search queries, we selected 3 databases: IEEE Xplore ¹, Scopus ², Web of Science ³ and ACM digital library ⁴. Before performing this queries on the selected databases, we first thought about which documents that can be considered relevant, regardless of the query content. We then defined the following inclusion criteria, as shown in table 2.1, to guarantee, at a first instance, that all documents retrieved meet the basic standards to be included in the study.

After applying all search queries in the 4 databases, a total of 1318 documents were retrieved, with the first query with a total of 652 documents, the second with 504 and the third with 142, as it is visible in table 2.2.

¹<https://ieeexplore.ieee.org/>

²<https://www.scopus.com>

³<https://www.webofscience.com>

⁴<https://dl.acm.org>

Table 2.2: Search results in different platforms

Search Query	IEEE Xplore	Web of Science	Scopus	ACM DL
QS1	133	96	178	245
QS2	96	58	93	257
QS3	18	9	7	108

2.1.3 Filtering Results

Before applying any filter to the retrieved documents, Zotero⁵ was used to support in the screening process and manage the collected records. The first step in the filtering process was to eliminate duplicate records. After this, the total number of documents retrieved for each query was reduced: the first query from 652 to 495, the second from 504 to 379, and the third from 142 to 136.

Next, we needed to manually screen each entry to find if it was relevant or not. In this manual process, we first analyzed the abstract; if it was classified as not relevant, we discarded it, otherwise, the document went to the next processing step, where both the introduction and conclusion were examined. If in this next step the document was not relevant, we discarded it, otherwise it advanced to a full-text analysis. For each query, a set of inclusion and exclusion criteria was defined, to help us define what a relevant document is, presented in tables 2.3, 2.4, 2.5, 2.6, 2.7 and 2.8.

2.1.4 Search Results

Here is a summary of the results from the literature review, where 136 documents were retrieved after analyzing the abstract with the inclusion and exclusion criteria. In the second phase, where the introduction and conclusion were analyzed, the number of documents was reduced from 136 to 54. Finally, the full-text was analyzed and the final total number of documents was reduced from 54 to 23, as it is visible in figure 2.1

⁵<https://www.zotero.org>

Table 2.3: Inclusion criteria for the first query

#	Inclusion Criteria
1	Proposes and describes a methodology for implementing digital twins
2	Considers in their method sustainability, scalability and/or interoperability
3	Describes how urban services were considered in their methodology
4	The method was implemented in a case study

Table 2.4: Exclusion criteria for the first query

#	Exclusion Criteria
1	Not related to smart cities
2	Describes digital twin applications without addressing methodology
3	Methodology only focus on non-urban domains (e.g. industrial sectors)
4	Duplicated entry

Table 2.5: Inclusion criteria for the second query

#	Inclusion Criteria
1	Identify or analyze urban services relevant to smart city development
2	Presents a classification of how important are the urban services identified
3	Discuss the integration of these critical services in a DT context
4	Provide evidence or case studies to support the findings

Table 2.6: Exclusion criteria for the second query

#	Exclusion Criteria
1	Duplicated
2	Not related to smart cities
3	No identification of critical services in DTs
4	Lack of details on why these services are critical

Table 2.7: Inclusion criteria for the third query

#	Inclusion Criteria
1	Address data integration within digital twin frameworks
2	Uses real-time or historical data from sensors, IoT and/or city platforms in the framework
3	Proposes a method to continuously integrate data in DTs
4	Discusses how to integrate data from multiple sources at the same time
5	Applied in a practical case or case study

Table 2.8: Exclusion criteria for the third query

#	Exclusion Criteria
1	Duplicated
2	Not related to smart cities
3	Focus solely on data collection without integrating in a DT
4	Discuss conceptual DTs without practical or technical integration methods

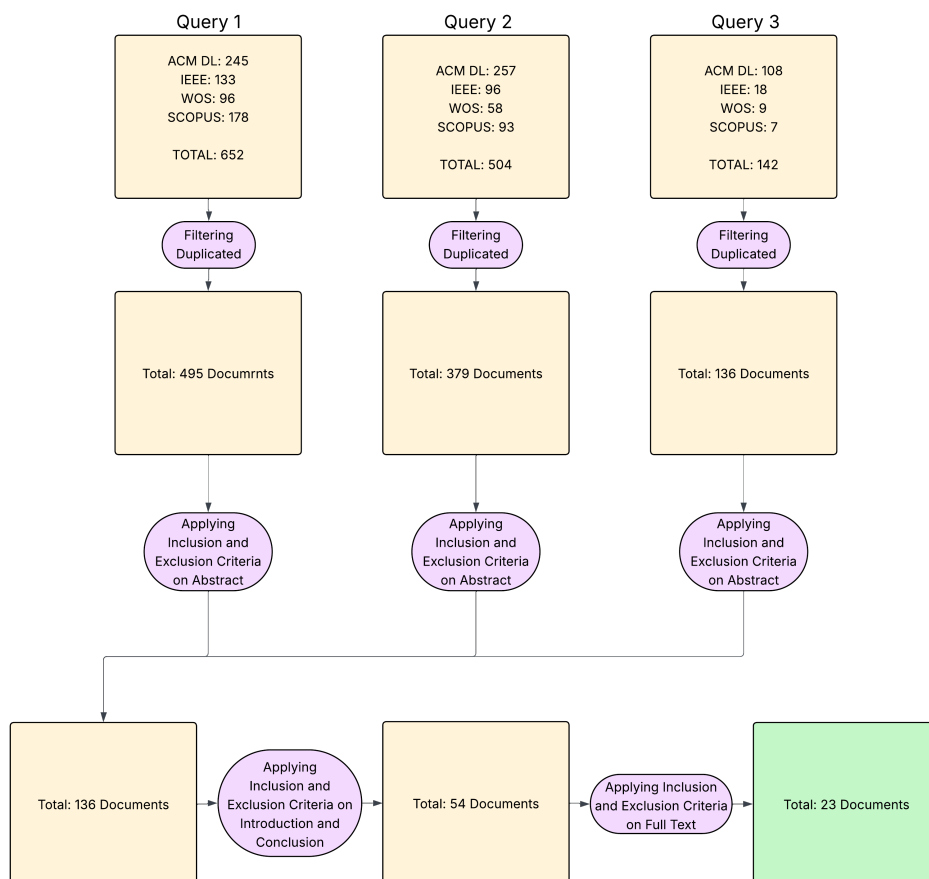


Figure 2.1: Literature review process

2.2 Related Work

In this section, it will be presented the main findings in the selected papers, regarding the research questions.

2.2.1 Conceptual Frameworks

To build a Digital-Twin of a specific city, it is necessary to define the areas of intervention and the corresponding digital ecosystem, ranging from the structural dimensions that characterize the twin to the agile frameworks required to sustain its life-cycle.

2.2.1.1 Structural Dimension

On the structural dimension, Xiong, Li, and Gao [32] propose a six-dimensional framework, that covers both conceptual aspects, which define the strategic scope and structure, and implementation aspects, which address the technical execution. This allows for a standardized characterization of Digital Twins, across several domains, such as agriculture, industry and smart cities. The dimensions proposed by the authors are detailed as follows:

1. Conceptual Dimensions

- **Twinning Objects:** This dimension classifies the physical target based on its hierarchical complexity. The authors distinguish between *Unit-level* (single assets), *System-level* (inter-connected assets like a production line), and *System-of-Systems* (e.g., *Smart Cities*).
- **Purposes:** The framework evaluates the intent of the Digital-Twin by intersecting two vectors: the *Level of Intelligence* (ranging from passive monitoring and diagnostics to predictive capabilities and autonomous control) and the *Value Drivers* (distinguishing between performance-centric goals, such as efficiency, and human-centric goals, such as sustainability and social well-being).
- **System Architectures:** This dimension analyzes the structural design of the Digital-Twin, focusing on architectural patterns (e.g., layered or service-oriented architectures) and critical quality attributes.

2. Implementation Dimensions

- **Data:** This dimension encompasses the entire data life-cycle, which is critical for maintaining the fidelity of the virtual model. It includes *Data Acquisition* (via IoT sensors, crowd-sourcing, or remote sensing), *Transmission* (utilizing protocols such as MQTT or 5G), and *Data Management*.
- **Modelling:** Identified by the authors as a core implementation dimension, this aspect covers the computational and mathematical representations required to mirror the physical entity.

In the context of Smart Cities, this typically involves integrating geometric models (such as BIM and GIS) with dynamic simulation engines to capture the complexity of the urban environment.

- **Services:** Finally, this dimension maps the functional output of the Digital-Twin against the *DIKW hierarchy* (Data, Information, Knowledge, Wisdom). Services are classified by their delivery mode: *Machine-to-Human (M2H)* services, which support decision-making through dashboards and visualization, and *Machine-to-Machine (M2M)* services, which enable closed-loop automation and autonomous actuation within the physical environment.

The authors argue that current architectural approaches have led to "fragmented, domain-isolated implementations" where sectors unnecessarily "reinvent the wheel". To address these "digital silos," they propose a transition towards a **Digital Twin Platform-as-a-Service (DT-PaaS)** model, which centralizes core capabilities on a shared cloud platform to facilitate cross-domain interoperability.

2.2.1.2 Digital Twin Continual Improvement

In Almeida, Muse, and Abreu [10], it is proposed a replacement for traditional methods of governance, with two complementary frameworks: the **4U-Cycle** (inspired by the PDCA cycle: Plan, Do, Check, Act [5]) and the **Agile Model for Urban Continual Improvement** (adapted from the Scrum methodology [28]). These frameworks function as the strategic and operational engines for city management, respectively, and are structured as follows:

1. The 4U-Cycle (Strategic Loop)

- **Urban Planning (PLAN):** Corresponds to the strategic definition phase, focusing on recognizing opportunities and establishing Public Policies and the Master Plan.
- **Urban Management (DO):** Represents the execution phase, involving the implementation of pilot projects and the daily management of city dynamics.
- **Urban Monitoring (CHECK):** Involves the observation and measurement of results using standardized Key Performance Indicators.
- **Urban Intervention (ACT):** Focuses on the actuation based on data, aiming to scale successful initiatives and correct failures identified during monitoring.

2. Agile Model for Urban Continual Improvement (Operational Loop)

- **Product Backlog:** Mapped to the Master Plan, representing the long-term goals and the strategic vision of the city.
- **Sprint Backlog:** Mapped to the Budget Plan, defining the specific resources allocated for interventions within a given period.

- **Sprints:** Mapped to Urban Management cycles (e.g., specific mandates or fiscal years), during which defined "Urban Intervention Packages" are executed.
- **Review/Retrospective:** Mapped to Urban Monitoring, providing the feedback loop necessary to adjust the Master Plan based on real-world performance.

The combination of these two frameworks reduces the complexity of cities in governance, allowing more efficiency and greater agility, while at the same time reduces the risk of urban interventions. For this methodology to be effective, it is necessary to use technologies that facilitate information management, requiring a centralized data systematization infrastructure to support decision-making.

2.2.2 Digital-Twins for traffic

Traffic and mobility is one of the most common domains present in Smart Cities DTs. According to Ferré-Bigorra, Casals, and Gangolells [11], the prevalence of Digital Twins in the mobility sector is largely driven by the high availability of IoT data streams. Unlike other urban domains that lack established sensing infrastructure, transportation networks are already densely monitored, facilitating the real-time synchronization required for effective digital twinning.

2.2.2.1 Hybrid Optimization of Traffic

External tools such as Google Maps already offer a comprehensive view of real-time traffic conditions by leveraging crowdsourced data. With this tools, we can create a methodology that uses this data and make a DT that can be used for diverse reasons, such as travel efficiency or fuel consumption, as presented by Ramal and Anbalagan [24].

The authors presented a methodology that involves three distinct phases:

1. **Data Collection and Preprocessing:** In this phase, the system obtains data from Google Maps SDKs and APIs, including Traffic Layer for density and Street View API for physical road attributes (width, lanes).
2. **Hybrid Optimization:** Two algorithms were used in this phase. The first one was the **Particle Swarm Optimization** (PSO), that was used for traffic signal optimization, where the objective was to minimize the total vehicle waiting time. This function can be represented as:

$$f(x) = \sum_{i=1}^N W_i \cdot T_i \quad (2.1)$$

where W_i is the waiting time for vehicle i , and T_i is the time spent by vehicle i at the inter-subsection. The **Artificial Bee Colony** (ABC) algorithm was used for route planning, with the goal of finding optimal paths. The fitness function can be defined as:

$$F = \alpha \cdot \text{Travel Time} + \beta \cdot \text{Fuel Consumption} \quad (2.2)$$

Where F is the overall fitness score, α and β are weights that balance the importance of travel time and fuel consumption in the optimization process.

3. **Machine Learning (Transformer Networks):** The framework uses a Transformer model (Encoder-Decoder architecture) for traffic flow prediction due to its ability to keep long range of knowledge in sequential data. It also uses a self-attention mechanism to weigh the importance of different time steps in the historical data. Since transformers process data in parallel, positional encodings are added to retain the order of the time-series traffic data.

As a result of this methodology, the average vehicle waiting time reduced from 120 seconds (Traditional Method) to 75 seconds (Proposed Model), the average travel time reduced from 300 to 225 seconds and the fuel consumption decreased from 50 to 35 liters. The transformer model achieved an accuracy of 97.82 %, outperforming commonly used architectures such as LSTM and RNN, which yielded accuracies of 95.88 % and 95.15 %, respectively. Although the authors propose a real-time architecture capable of ingesting live data via the Google Maps SDK, the experimental validation presented relies on a pre-collected "real-world dataset" derived from several weeks of observation. Consequently, while the system demonstrates the computational efficiency required for real-time operations (with a processing time of 450 ms), the reported performance metrics reflect an offline evaluation rather than a live, closed-loop implementation where the Digital Twin actively controls physical traffic signals in real-time.

2.2.2.2 AI-Driven Digital-Twin of Madrid Traffic

Capitalizing on this established sensing infrastructure, several metropolitan areas have moved towards operational implementations. A prime example is Madrid, where a Digital Twin was developed to predict traffic flow by leveraging the extensive network of IoT devices distributed across the Spanish capital [14]. This study introduces a systematic and scalable methodology for designing Smart City DTs, specifically focused on real-time traffic management. The proposed architecture is composed of six interconnected subsystems, as illustrated in Figure 2.2:

1. **Edge Subsystem:** Connects IoT sources. In this case study, this involved 60 permanent traffic stations in Madrid recording vehicles per hour. Data was obtained from Madrid's Open Data Platform.
2. **Data Management Subsystem:** This subsystem can be divided in 4 sub-modules:
 - **Preprocessor:** Examines the time-series data and verifies whether the data contains enough information for conversion and scales data using Min-Max scaling.
 - **Converter:** Transforms the scaled data into NGSI-LD format, where the data model used by the converter is defined by the digital twin subsystem.
 - **Injector:** Pushes standardized entities into the Context Broker. It updates the entities created in the context broker for each traffic station using its REST API.

- **Context-Broker:** In this module, Stellio[12] is used because it follows linked-data principles and complies to the NGSI-LD standard, providing a standardized API that supports data retrieval, entity querying, and subscription mechanisms.
3. **Digital Twin Subsystem:** Uses the data in the context broker for correlation analysis, training the Deep-Learning (DL) models, and real-time prediction. This is the subsystem that communicates with the external users. It can be divided in 3 sub-modules:
 - **Correlation Analyzer:** Calculates similarity between data streams to identify "clusters" of sensors that behave similarly, with the goal of reducing the need for training separate AI models for each traffic station, thus minimizing both time and cost.
 - **Predictor:** In this module, five well-known DL models were used: RNN, Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP), GRU, and Autoencoder (AE). These models were then tested on the similar traffic stations identified through the correlation analysis.
 - **Visualization and API:** Uses FastAPI to expose results to external users.
 4. **Event Management Subsystem:** Orchestrates data flow, ensuring that when a prediction is requested for a specific road, the data from its highly correlated "parent" station is routed correctly.
 5. **System Management Subsystem:** Handles overall security via Keycloak⁶ for authentication/authorization.
 6. **Resource Management Subsystem:** Uses lightweight kubernetes to manage containerized components across core servers and edge devices.

This architecture is an extension of previous work made by the authors [15]. With the introduction of correlation analysis, this architecture integrates scalability, because it is possible to verify which traffic stations are highly correlated, so with this methodology we can train a single model on one station and successfully apply it to multiple correlated stations, significantly reducing the number of AI models required for a city-wide deployment. After training the models, the authors found that the simplest model (Multilayer Perceptron - MLP) was the top performer, achieving the highest average R-Squared (R^2) score of 0.939 and the lowest Mean Absolute Error (MAE) across test stations, outperforming more complex recurrent networks (RNN, LSTM). According to the paper, this occurs due to the stability of the traffic patterns and limited dataset size. This suggests the proposed advanced "AI-driven" aspect might be over-engineered for stable flows unless larger datasets are used. Also the validation of this study was conducted using data from a relatively short period (January and February 2024), which may not capture seasonal traffic variations or long-term structural changes in urban mobility. Overall, with the proper data, this methodology can be applied not only to traffic, but also to other domains, like energy grids, where

⁶[https://stellio.readthedocs.io/en/latest/admin/keycloak integration.html](https://stellio.readthedocs.io/en/latest/admin/keycloak%20integration.html)

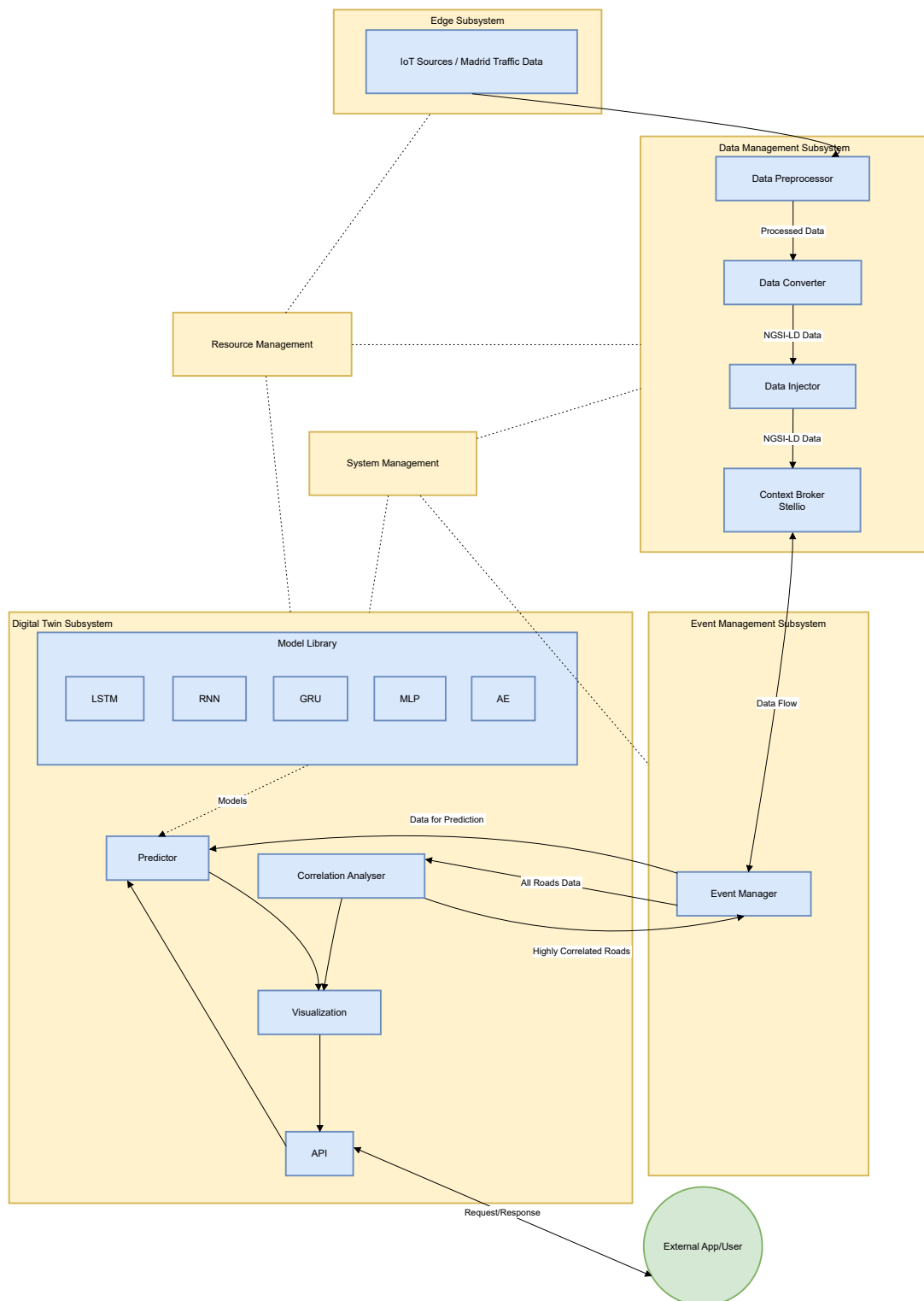


Figure 2.2: Traffic management architecture

the sheer density of smart meters creates a massive computational bottleneck, applying this correlation technique. This approach can solve the scalability challenge, shifting the paradigm from brute-force processing to resource-efficient, modular modeling.

2.2.3 Digital-Twins for Healthcare

Healthcare is one of the domains that can be significantly impacted and improved by Digital-Twins. With the increasing availability of historical datasets from real-world resources, Artificial Intelligence (AI) can be used to predict future outcomes for specific patients in the form of an AI-generated DT. Such technologies can be useful in potential diagnoses, disease progression and treatment responses, allowing for high-level medical interventions and therapies [23].

At a city level, the deployment of such AI-driven frameworks is instrumental in assessing the "urban pulse". By continuously monitoring livability metrics, like air quality, these systems help authorities to mitigate health risks proactively. Even with wearable devices, city officials can monitor residents in terms of their heart rate, respiratory rate, etc, and act accordingly to the data obtained in real-time. To operationalize this, Jayachandran et al. [16] propose a layered architecture implemented via the *Digital Twin Realisation for Smart City* (DTRSC) tool. The authors validate the approach through specific use cases, such as the "Healthy Route" for pollution avoidance and real-time vital signs monitoring, effectively bridging the gap between city-wide environmental data and individual health outcomes. This architecture can be divided in four different modules, and it is represented in 2.3:

1. **Data Emulation Module:** Contains emulated devices designed to replicate real-world IoT nodes, with each device being able to host several sensors of different categories. These sensors rely on pre-trained ML models, that were built from realistic datasets. When real-time sensor data are unavailable or difficult to obtain, the system uses those models to produce a time series of synthetic correlated data in real time. Like physical IoT devices, the emulated devices communicate with the data transfer module via the MQTT protocol ⁷.
2. **Data Transfer Module:** Encapsulates generated data into MQTT packets and transmits them to a broker. It uses PyShark to capture network traffic, providing a unified interface for both real and emulated data.
3. **Big Data Handling Module:** Streams the captured data into Kafka topics ⁸. This ensures the system can handle the volume of data required for a city-scale deployment.
4. **Data Processing and Analytics Module:** In the processing phase, the system performs several data preparation steps to make the information suitable for Machine Learning (ML) analysis, addressing aspects such as data encoding, feature scaling and missing values. After this preparation, the models are used to generate predictions and deliver insights that can

⁷<https://mqtt.org/>

⁸<https://kafka.apache.org/41/design/protocol/>

support decision-making for city authorities or tool users. This users, through the Command Line Interface (CLI) integrated in the system, can choose specific ML algorithms such as long short-term memory (LSTM), linear regression, random forest, gradient boosting or neural networks, to perform regression and classification tasks.

To validate this architecture, three specific use cases were implemented through the Digital Twin Realisation for Smart City (DTRSC) tool to demonstrate its capability to handle environment and eHealth-related scenarios. The first one was applied to **Air Quality Forecasting**, where it was used a LSTM model to forecast Air Quality Index (AQI) based on pollutants ($PM_{2.5}$, NO_2 , etc.). The model achieved an R^2 score of 0.88 and low error rates (MAE: 0.7043), demonstrating effective short-term environmental forecasting.

In the second case study was **monitoring respiratory conditions**, with the help of wearable devices, with the goal of early detection of potential health risks, such as those related to lung function. In this case, it was used classification models (Logistic Regression, Random Forest, Gradient Boosting, ANN) to detect respiratory risks from wearable data. All four models demonstrated high accuracy levels, ranging from 93.19 % to 94.75 %, with Gradient Boosting achieving the best performance.

In the third case study the goal was to combine this two case studies into a new one, to demonstrate. In this concrete case, combining environmental data (from case 1) and personal health data (from case 2), this case study offers individuals tailored health alerts whenever their physiological metrics and surrounding environmental conditions reach critical levels. Here, the data from these two previous cases is emulated and transferred via the emulation and transfer modules and then streamed into the big data handling module. Model topics are then created for each case study and aligned with timestamps. Finally, the 4 classification models mentioned above process these combined topics effectively, generating timely warnings (alert) or no alert status based on the integrated data considering the respiratory rate and AQI index. Once again, the accuracy of each one is excellent, ranging from 95.88 % to 97.63 %, with Gradient Boosting once again achieving the best performance.

With these three case studies, we can see the architecture's ability to facilitate cross-domain intelligence, triggering alerts only when both environmental conditions and personal health metrics indicate a risk, where users can easily use their own ML models and tune each model with hyper-parameters. However, the current version has some limitations in handling a very large number of devices due to support for only a single MQTT broker and a lack of remote management capabilities for MQTT clients.

2.2.4 Digital-Twins for Public Safety

In the context of Smart Cities, the domain of public safety encompasses both the prevention of criminal activity and the mitigation of traffic-related risks. DTs can be a useful tool to address this domain, offering a unified vision that integrates predictive analysis with real-time monitoring.

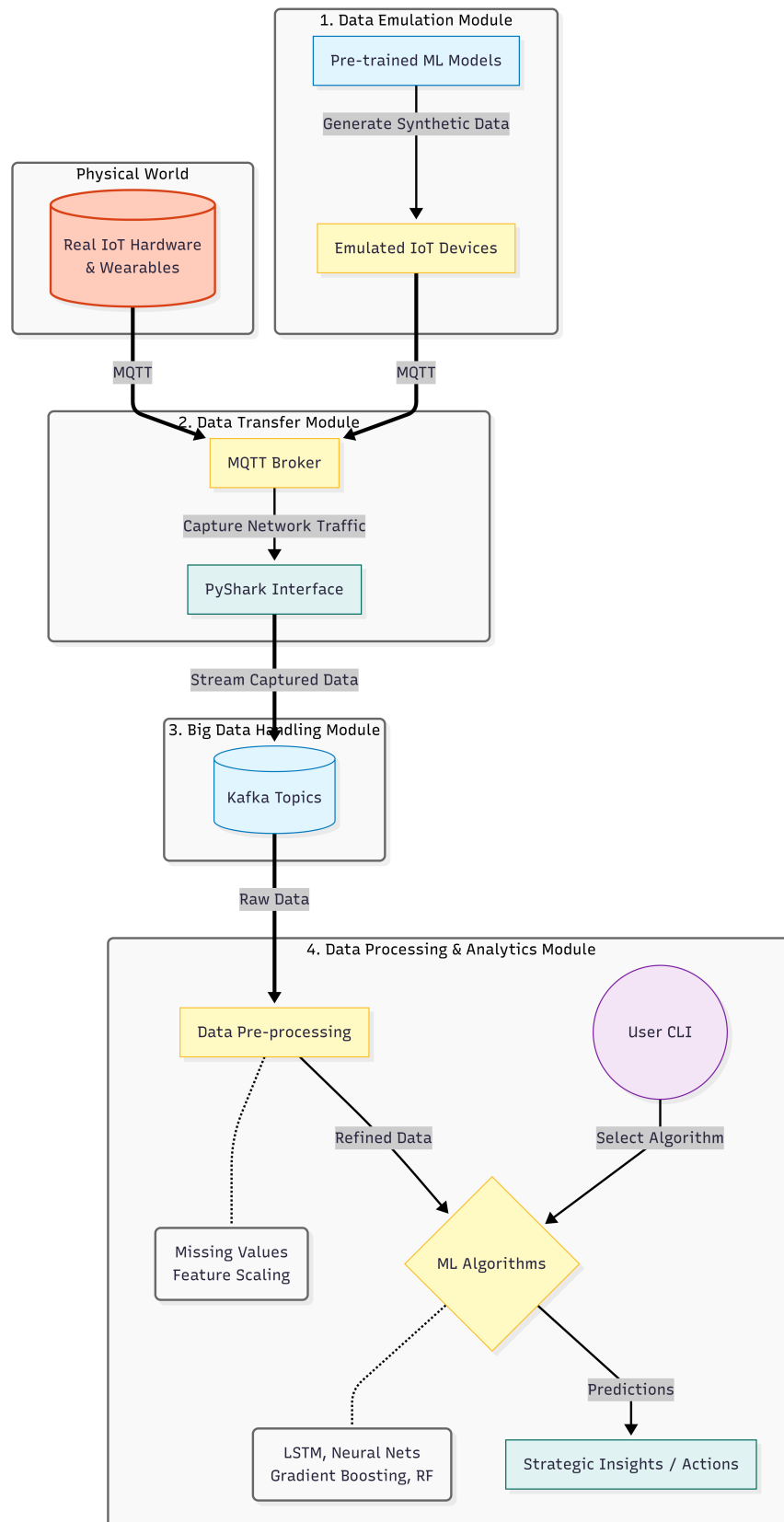


Figure 2.3: Smart Health architecture

2.2.4.1 Identifying Criminal Routes

License Plate Reader (**LPR**) is a scanning technology that can read license plates of cars in real-time and immediately check across databases and give alerts when such plate is linked to a car or person of interest. With this technology, local authorities can detect stolen cars or help in other specific criminal cases. However, static sensor placement is inefficient because crime patterns are spatiotemporally dynamic, according to Pan, Mohammadi, and Taylor [22]. With the creation of a **DT**, it is possible to simulate crime patterns and dynamically place the **LPRs** to enhance crime detection and deterrence.

The authors propose the following two-stage methodological framework:

1. **Crime Prediction:** In this stage, the city road network was divided into Thiessen Polygons, where each polygon contains points closest to a specific road segment. Then, Crime records were aggregated by polygon and by week to create a spatiotemporal dataset. Finally, a **ConvLSTM** was used, which combines a **Convolutional Neural Network**, that captures local spatial patterns, with a **LSTM**, that captures temporal patterns, to predict which polygons will likely contain crime in the next time step.
2. **Sensor Placement:** Based on the predicted crime locations, the system initiates a traffic simulation. Suspects are modeled as agents departing from the crime scene, moving through the road network and choosing directions randomly at intersections. Then, the system calculates which road segments and directions (North/South/East/West) have the highest probability of "capturing" these agents. Finally, **LPR** sensors are virtually placed at these high-probability bottlenecks. The simulation iterates, removing captured agents and recalculating to find the next best location until all sensors are placed.

With an average precision of 0.90, the framework demonstrated high efficacy in forecasting crime locations. However, performance was constrained by the assumption that fleeing vehicles instantaneously transition to major roadways, potentially oversimplifying escape routes that involve traveling on minor streets. One of the main findings of this study was that the optimal sensor location does not strictly correlate with high-crime density zones. The simulation showed that one sensor was outside of the crime hotspot, at a location where multiple escape routes converge. Not only this method determined the best location to place the sensors, but also its specific orientation (e.g., facing East), optimizing resource allocation since many police stations have limited access to this technology (90 % own less than 15 **LPRs**).

2.2.4.2 VALLPASS System

DTs play a pivotal role in ensuring public safety, extending beyond crime prevention to include the mitigation of traffic accidents and risks arising from infrastructure failures. The VALLPASS system is a set of energy-autonomous lampposts that are distributed along urban regions that are not controlled by traffic lights. At each crosswalk, two lampposts are deployed, one at each end,

with sensors that detect vehicles, pedestrians, weather and accidents. Based on the information retrieved, the system autonomously adjusts luminosity levels to enhance the security of each pedestrian crossing [25].

To integrate these physical assets into the wider Smart City ecosystem, the project establishes a virtual counterpart based on the NGSI-LD standard. By mapping the heterogeneous hardware components (e.g. batteries, luminaires, and pedestrian sensors) into standardized semantic data models, the system creates a Digital Twin capable of real-time remote management and seamless interoperability within the FIWARE framework.

To model such project, the physical system (VALLPASS crosswalk) was broken down into granular entities: Pole, Luminaire, Battery, WeatherSensor, VehicleSensor, etc. Subsequently, leveraging the FIWARE Smart Data Models repository, the authors systematically mapped each physical component to a standardized digital equivalent. By aligning the proprietary hardware with these globally accepted schemas, the project ensures that the resulting Digital Twin is not merely a digital copy, but a semantically interoperable system capable of exchanging data with any other FIWARE-compliant platform.

2.2.5 Digital-Twins for Energy

Through DTs, city authorities can manage energy systems, by real-time monitoring consumption patterns and dynamically balancing supply with demand. This granular visibility allows operators to detect load imbalances or voltage anomalies instantaneously, triggering automated demand-response mechanisms to prevent grid overloads, leading to improved energy efficiency, reduced downtime, and lower maintenance costs [2].

2.2.5.1 Digital-Twin based on System Development Life-Cycle

Ozkan, Kok, and Ozdemir [21] propose a cloud-based layered architecture implemented via microservices, which adapts the traditional software engineering SDLC (Planning, Analysis, Design, Development, Operation, and Maintenance). This architecture is composed by six layers:

1. **Physical Component Layer:** Represents the tangible assets of the energy system, including the Micro-grid components (PV panels, battery storage systems, and manageable loads) and the Main Grid connection. These assets are equipped with sensors and actuators to monitor real-time energy flow and market pricing conditions.
2. **Data Collection and Device Control Layer:** Acts as the bi-directional middleware bridging the physical and virtual domains. It is responsible for aggregating raw telemetry data via IoT gateways and transmitting control commands back to physical devices using secure TCP/IP communication protocols.

3. **Digital Twin Layer:** The core of the architecture, hosting the mathematical models and state equations that virtually replicate each physical asset. This layer continuously synchronizes with incoming data to mirror dynamic parameters, such as battery State-of-Charge (SoC) or instantaneous solar generation.
4. **Cloud Management Layer:** Orchestrates the deployment and scalability of the **DT** ecosystem. Utilizing containerization technologies (e.g., Docker) and orchestration platforms (e.g., Kubernetes, OpenStack), this layer ensures that microservices run in isolated, scalable environments.
5. **Digital Twin Management Layer:** Provides advanced intelligence and operational logic atop the virtual models. It encompasses modules for system configuration, anomaly detection, "what-if" scenario simulation, and predictive forecasting using algorithms like ARIMA and Facebook Prophet.
6. **User Interface Layer:** Facilitates Human-Machine Interaction (HMI) through specialized web-based dashboards. It offers dual functionality: a management interface for deploying cloud resources and an operational dashboard for visualizing real-time metrics, historical trends, and forecasting insights.

The study compared ARIMA and Facebook Prophet, with Prophet consistently outperforming ARIMA in this context, as it better captured daily peaks, seasonal trends, and sharp fluctuations typical of renewable energy and consumption patterns. The advantage of a micro-service architecture is that it allows for independent development, deployment, and scaling of individual DT assets, which is highly relevant for complex systems like Smart Cities, however, for this experiment, the results showed an upward trend in latency as data transmission requests increased, primarily because the cloud platform was geographically distant and used a public network.

2.2.5.2 Digital Energy Twin of Regensburg

Thelen et al. [30] propose a **DT** of Regensburg, focused on energy, that facilitates visualization of buildings that are using more energy, through a red-yellow-green color scale. This platform was built as a web platform, with a frontend composed of Solid.js and CesiumJS, a backend with Python FastAPI and a data storage composed of 2 databases: one with MongoDB, that stores static building information and the other with InfluxDB, that stores time series data like consumption data. Live data (from sensors) is transferred to InfluxDB via a live data module, which is implemented as a message-broker architecture based on Apache Kafka. Topics are organized hierarchically (Room > Apartment > Building > Block > Block group > Sub-district > District), in which workers automatically propagate data up the hierarchy, allowing for seamless aggregation of energy usage at the district level.

The city provided a shapefile with the sub-districts, however, they did not have the precise borders of the block groups and blocks. To address this challenge, they used LOD2 building

coordinates to query the city's API, determining which block a building belongs to. Then, with the alpha shapes algorithm, they generated fitting polygons around the point sets of buildings belonging to the same block, visually reconstructing the administrative districts. The authors found another challenge when calculating heat consumption since walls bordering adjacent buildings lose less heat than walls bordering the outside. To overcome this, they used an adapted algorithm to detect overlapping walls between adjacent buildings, using the bounding box of each wall to detect overlapping. If walls overlap, heat loss is assumed to be lower.

To measure electricity consumption, smart meters were used to record active and reactive energy, current, and voltage at a frequency of 1 Hz. They developed a energy detection algorithm, to identify events (e.g. turning a device on/off) by detecting significant changes in energy absorption and, based on public datasets, they used a threshold list to infer which device caused the event. The system assigns generic IDs to each device, but users can use the web platform and name each id to the corresponding device.

This platform also offers the possibility to users to transform a specific building into a BIM model, which can be useful for further development of discipline-specific technical models, however, this transformation is not bidirectional (from BIM to the Twin), which would significantly enrich the model.

2.2.6 Digital-Twins for Water Management

With new technologies, governments can efficiently simulate catastrophes related to weather (e.g. floodings) and better prepare their resources to when something like that happens in the real-world. DTs can be used for this purpose, allowing government officials to visualize in a virtual representation such phenomenons and to be more prepared when these events happen.

For such type of DT, Alexandridis et al. [1] conducted two case studies following a spatially-explicit framework, consisting of three layers [26]: **Data and Integration Layer**, where data is obtained from IoT devices, whether it is static (e.g. terrain) or dynamic (e.g. river water levels). Then the data is converted to a desired format (e.g. GeoJson, Shapefiles, ...); **Spatial Analysis and Modelling Layer**, where GeoAI-generated data can be combined with the real data for predictive analysis of extreme events and for research analysis; **Visualization Layer**, where the results are displayed, for city planners to monitor. Game Engines like Unity or Unreal Engine can be used for immersive representation and better help the users.

The first case study was conducted in Orange County, California, where it was implemented an IoT-enabled real-time dashboard for Hydstra stations. The system monitors flood zones and integrates with camera feeds to provide situational awareness for decision-makers, through a dashboard, providing accurate and real-time data for the condition of the water availability and weather conditions throughout the County and also emergency services in case on catastrophic water crises, and other water-based natural resource emergency management situations. The second case was applied in Australia, where a real-time water-sensitive tool was developed based on this architecture. It allowed planners to run "what-if" scenarios, such as changing road widths or building footprints, and immediately calculates the impact on storm water capacity and flood resilience.

These case studies demonstrate that **DTs** can be a powerful instrument for mitigating such catastrophes. However, the proposed framework lacks specific guidelines regarding data formats, which presents a significant interoperability challenge. Furthermore, the reliance on game engines for visualization entails relatively higher computational and financial costs compared to traditional dashboard solutions.

2.2.7 Digital-Twins for Pollution

DTs can be applied not only to specific services, like healthcare, but also to achieve sustainable and social goals. With numerous cities pursuing carbon neutrality, **DTs** provide the necessary framework to monitor environmental parameters in real-time, allowing urban planners to simulate the environmental impact of urban policies prior to implementation. Kljaić et al. [17] propose a methodology that combines environmental, social and governance goals, to be aligned with EU standards to achieve carbon neutrality by 2050, focused on transportation and energy.

Pollution might be environmental, but it can also be visual. Campo Jiménez et al. [6] affirms that a precise and realistic graphical rendering of lighting is a crucial aspect for assessing its potential effects on human and wildlife activities, specifically for reducing light pollution. By utilizing **DTs** integrated with Virtual-Reality (**VR**), city planners can test various scenarios, such as different lighting schedules or bulb types, to analyze their impact on the urban environment without disrupting real-world operations.

In big cities, the noise produced by cars and crowds can dramatically lower the quality of life of the population. Ascari et al. [4] proposes a **DT** specifically for dynamic Road Traffic Noise (RTN) mapping. The authors propose a shift from traditional, static noise maps, which rely on expensive, infrequent measurement campaigns, to a dynamic approach driven by crowdsourced data streams. This architecture is composed by three phases:

1. **Data Collection and Processing:** Uses a data lake to store structured and unstructured data at scale, integrating four distinct types of data: Real-time, Near-time (Cloud Services), Social media data (e.g. posts regarding noise complaints) and legacy, from open datasets. Data is cleaned before going into the next phase.
2. **Road Traffic Noise Estimation:** Instead of measuring noise directly everywhere (which is expensive), the system calculates noise based on traffic flow. It uses "Link Delay Functions" to estimate traffic volumes from the travel time data collected in Stage 1. The model is calibrated in a test area (Pisa) using real traffic counters to fine-tune the equations before being deployed to the target area (Brindisi).
3. **Digital Twin Development:** Built on Unreal Engine 5. The 3D model of the city is imported via Datasmith and live data is fetched via RESTful APIs. An additional layer is added to represent visually the areas with more noise, allowing policymakers to establish priorities in an easier and faster way, enhancing the implementation of effective mitigation action.

Relying on crowdsourced data significantly reduces costs on implementing such solution, however, there is a dependency on third-party APIs and the data quality might be not good, mainly from social media posts.

2.2.8 Digital-Twins as a Multi-Agent System

Extending the concepts explored previously, this subsection introduces a distributed architecture. When modeled as a Multi-Agent System (**MAS**), the city is no longer treated as a monolithic entity, but as a composition of active agents. Unlike the previous centralized models, these agents possess autonomy, allowing them to pursue either collective goals or their own specific objectives. Consequently, **MASs** provides a more faithful representation of urban dynamics, capable of capturing the inherent stochasticity and non-linear patterns that emerge from the complex interactions between independent actors, thereby outperforming traditional representations of more complex systems [18].

When modeling according to this paradigm, **DT** can be represented as a quintuple, constituted by **Agents**, which are the active components of the system (people), **Entities**, which are the passive components that can be used by agents, **Layers**, where agent perform their actions, **Sensors**, which are physical components that can communicate to the real world, and **Mapping**, that defines the logical connection between the virtual components (Agents, Entities and Layers) and the physical sensors, as described by Clemen et al. [8].

In this same article, it is proposed an architecture for bicycle renting in Hamburg, based on this paradigm, with a methodology that uses the MARS framework with a Temporal Vector Layer. In this specific case, agents are the cyclists within the city, the entities are the bicycles and the rental stations, the layers represent the map of Hamburg and the logic inside renting a bicycle (e.g. bicycle renting time), the sensors measure the bicycles available at each station, and the mapping connects the real-world with the virtual one, through each station id, looking for bicycles and updating them accordingly, integrating real-time data via a Temporal Vector layer. This study implements a practical "mixture of soft and hard correction" strategy: updating static resources (bikes at stations) while leaving dynamic agents (bikes en route) untouched to avoid disrupting ongoing logic. The authors noted that handling real-time data presented significant challenges due to outliers, latency, and intermittent service availability, requiring complex cleaning and validation procedures. However, this correction is indispensable: without it, the variance between simulated and real states increases continuously, whereas synchronization effectively resets this accumulated error.

2.2.9 Adapting Game Models into a Digital-Twin

As visible in previous sections, some architectures use game engines like Unreal Engine or Unity to improve the visualization of **DTs** for the end user. These engines offer near-photorealistic rendering capabilities that enhance the immersive experience. By facilitating the integration of

vast geographical and environmental datasets, they enable the creation of highly accurate and comprehensive virtual replicas of cities [7].

Costagliola, Montanari, and Bellavista [9] propose a methodology based on Glassbox framework, which is a game engine for simulation games like SimCity. While players can never "rewind" to see what different decisions would have brought, this model allows for users to try different decisions on the city and see the outcome of each one, with the possibility to rewind and take another decision. It can be divided in 4 layers:

1. **Presentation Layer:** Interfaces with the external systems and users, ensuring that data are both ingested and accessed in a standardized and meaningful way.
2. **Application Layer:** Handles ETL (Extraction, Transformation, Load) and workflow orchestration using tools like CKAN.
3. **Data Layer:** Manages raw storage. Its main function is to ensure that information is organized in a scalable, secure and efficient manner, supporting analysis, simulation and visualization operations.
4. **Integration Layer:** The core simulation engine that applies rules and processes cross-domain logic.

This model was applied to a mobility use case, extending the original framework to encompass specific urban elements. In this context, Units (interacting entities) are defined as roadside sensors and charging stations, while Resources (measurable assets) comprise vehicles and electricity. Furthermore, the Networks representing the connection between units are identified as the road infrastructure and the electric grid, visualized through specific Maps such as air quality and noise distribution heatmaps.

This bottom-up approach creates limited additional work/overhead compared to rigid top-down ontological models, facilitating faster adoption and scalability for different urban contexts. It also promotes interoperability between different DTs, as each network can be represented by a single point, simplifying communication between diversified sub-systems. However, the integration of data and metrics for this adaptation adds complexity to the model itself, making the management and maintenance of simulations more challenging

2.3 Main Findings

Table 2.9 summarizes the reviewed literature, categorizing studies by domain, middleware used, and methodology ("Yes" if it is new and "No" if it is an extension of an existing one). It also details the specific data sources and formats employed, alongside their real-world deployment locations, if any.

From this review, we can see that it is possible to combine historical data with real-time data to create a better model to represent a DT. Even when data is scarce, we can use artificial intelligence to produce more data to better train the model. Combining data from different sources

Table 2.9: Summary of related studies

Article	Domain	Middleware/Framework	Methodology	Input Data	Data Format	City or Country Applied
AI-driven DT for Traffic management [14]	Traffic	Stellio Context-Broker (FIWARE)	No	IoT Sensors, Cameras	NGSI-LD	Madrid
Slim DT [30]	Energy	Shapefile Provided by City Administration	Yes	Connector Modules, IoT Devices	GeoJson	Regensburg
An Architecture for e-health [16]	Health	No	Yes	physical and virtual IoT trained with ML	Kafka Topics	N/A
Hybrid Optimization for DT [24]	Traffic	Google Maps API	Yes	Datasets accross Google Maps APIs	Processed datasets	N/A
DT for Smart Water Management [1] [26]	Water	Geo-IoT	Yes	GeoAI-generated data, IoT devices	Various, like GeoJson, Shapefiles, BIM/IFC	California Australia
VALLPASS [25]	Safety	FIWARE	Yes	IoT devices	NGSI-LD	Northeast Portugal
Interconnecting Urban Networks [9]	Mobility	OpenMetadata and CKAN (For description of data)	No	Not specified	Dataset (csv)	N/A
OUTFIT Crowdsourc Noise [4]	Traffic	Google Directions API	Yes	People's devices (social media), open datasets, IoT devices	JSON	Brindisi
Multi-agent system DT [8]	Mobility	MARS	Yes	IoT devices	Data Objects	Hamburg
Public Safety DT [22]	Safety	No	Yes	Historical data	Dataset	Warner Robins
SDLC for DT [21]	Energy	Openstack, Kubernetes	No	IoT devices	Not specified	Helsinki

can be challenging; to address that, the different formats should be converted into a standardized format to ensure semantic interoperability and facilitate seamless integration across heterogeneous systems. Moreover, acquiring data from third-party sources is significantly less expensive than installing city-wide sensor infrastructure, however, it usually entails a more rigorous preprocessing workflow, primarily due to inconsistent data formats and a higher prevalence of outliers, which can significantly compromise the system's calibration.

From the analysed literature, we can also observe that a set of requirements should be met for a **DT** to be useful to stakeholders. First, it should support the integration of multiple data domains, ranging from real-time **IoT** streams to static or AI-generated data. Second, it should support the execution of what-if scenarios, allowing urban planners to create alternative scenarios and assess their impact before applying them in the real world. Third, it should provide performance indicators, such as traffic conditions, pollution levels, or noise levels, so that performance of the smart city can be evaluated appropriately. Finally, it should include adequate visualization of results and appropriate interaction mechanisms, facilitating the interpretation of outputs by city officials and planners. These four requirements suggest that data interoperability, what-if scenario analysis, performance evaluation based on appropriate indicators, and decision support through visualization should be treated as essential elements of a **DT** platform.

DTs can be implemented in several ways, with each architecture having its own specifications and layers. However, from what we analysed, they usually comprise these three layers: a data and ingestion layer, where data is obtained from different sources and transformed into a standardized

format, a model layer where calibration of the digital model of the city is done, regarding the specified service, and a visualization layer where the end user or system communicates with the model, visualizing the results and make modifications to improve the model performance. Scalability can be offered through the decoupling of these layers and the use of containerization technologies, ensuring that the expansion of one component (e.g., adding more sensors) does not compromise the performance of the entire system. A significant limitation observed in the literature is that current **DTs** often lack multi-service support; most architectures are designed to address a single, isolated domain rather than integrating multiple urban services.

This limitation motivates the architecture proposed in this dissertation, which is designed to support multiple urban services within the same Digital Twin environment through a shared communication layer and a unified user interface that can accommodate different simulation modules. Implementing such solution would provide a centralized interface for various stakeholders (e.g., traffic analysts, energy managers), allowing them to monitor their respective sectors in real-time and execute targeted interventions based on comprehensive data. Furthermore, it could enable the identification of cascading effects across critical infrastructures, such as understanding how a power grid failure might instantly compromise traffic signalling systems.

Chapter 3

Methodology

In this chapter, we present the methodology adopted in this dissertation to address the main limitation identified in the literature review, namely the lack of support for multiple urban services within the same **DT** environment. To this end, the chapter describes the proposed architecture, from the implementation plan to how each element of this architecture was implemented.

3.1 The Proposed Architecture

Based on the literature review, the proposed architecture was designed to address the main limitation identified in existing **DTs**, namely the lack of support for multiple urban services within the same platform. In addition, the architecture was guided by four main requirements derived from the analysed studies, which are integrated into the proposed solution: support for heterogeneous data sources, support for what-if scenario analysis, evaluation through relevant performance indicators, and adequate visualization and interaction mechanisms for decision support.

A visual representation of the proposed architecture can be seen in Figure 3.1. This architecture is composed by 4 modules:

1. **Real World & Sources Layer:** This layer gathers the different sources of information required by the platform. These may include datasets, **IoT** or sensor inputs, and synthetic data. It represents the origin of the data later consumed by the services integrated in the **DT** environment.
2. **Context Management Layer:** This layer is responsible for managing the contextual information exchanged across the platform. In the proposed architecture, this role is supported by FIWARE, which acts as the intermediary component between the visualization layer and the service modules, allowing entities, parameters, and state updates to be exchanged in a consistent way.
3. **DT Services Layer:** This layer contains the domain-specific services of the platform, such as the Transit Module and the EV Module, used in this dissertation. Each service integrates its own API and simulation engine. The API is responsible for retrieving the required input

data, interacting with FIWARE, preparing the execution, and triggering the corresponding simulation engine. The simulation component then processes the configured scenario and produces the outputs associated with that service.

4. **Visualization and Interaction Layer:** This layer corresponds to the unified city dashboard through which the user interacts with the platform. It allows the configuration of service parameters, the triggering of simulations, and the observation of the resulting outputs, providing a single entry point for the different services integrated into the DT.

3.2 Assessment Method

The proposed architecture is assessed through the implementation and evaluation of different service modules within the same DT platform. More specifically, the assessment focuses on verifying whether the architecture is capable of supporting multiple services, integrating their respective data and simulation workflows, enabling what-if scenario analysis, and presenting the resulting outputs through a unified visualization dashboard.

For the implementation of our architecture, we build upon previous work on traffic modelling in Porto, where a synthetic population was generated and used to support the creation of a traffic DT of the city [27]. Based on this foundation, we add a Mobility Service for electric vehicles and charging stations, integrating both modules in a unified visualization platform and allowing different stakeholders to modify each model and run what-if scenarios. MATSim is used as the simulation engine, since it was already adopted in the traffic implementation and its agent-based nature allows the precise tracking of individual vehicle states, such as battery levels. The visualization component is implemented as a web platform through which users can interact with the system, analyse results, and trigger new simulations with different parameters, such as the number of vehicles.

3.3 Practical Implementation

This section describes the practical implementation of the proposed solution and the main steps followed to make it operational.

3.3.1 FIWARE

In the proposed architecture, FIWARE is adopted as the middleware solution responsible for context management and data integration. Its main role is to provide a common environment for representing relevant entities, storing their state, and supporting the exchange of information between the visualization platform and the respective simulation module. By relying on this intermediate layer, the system avoids direct and rigid point-to-point communication between modules, thereby promoting interoperability, modularity, and extensibility.

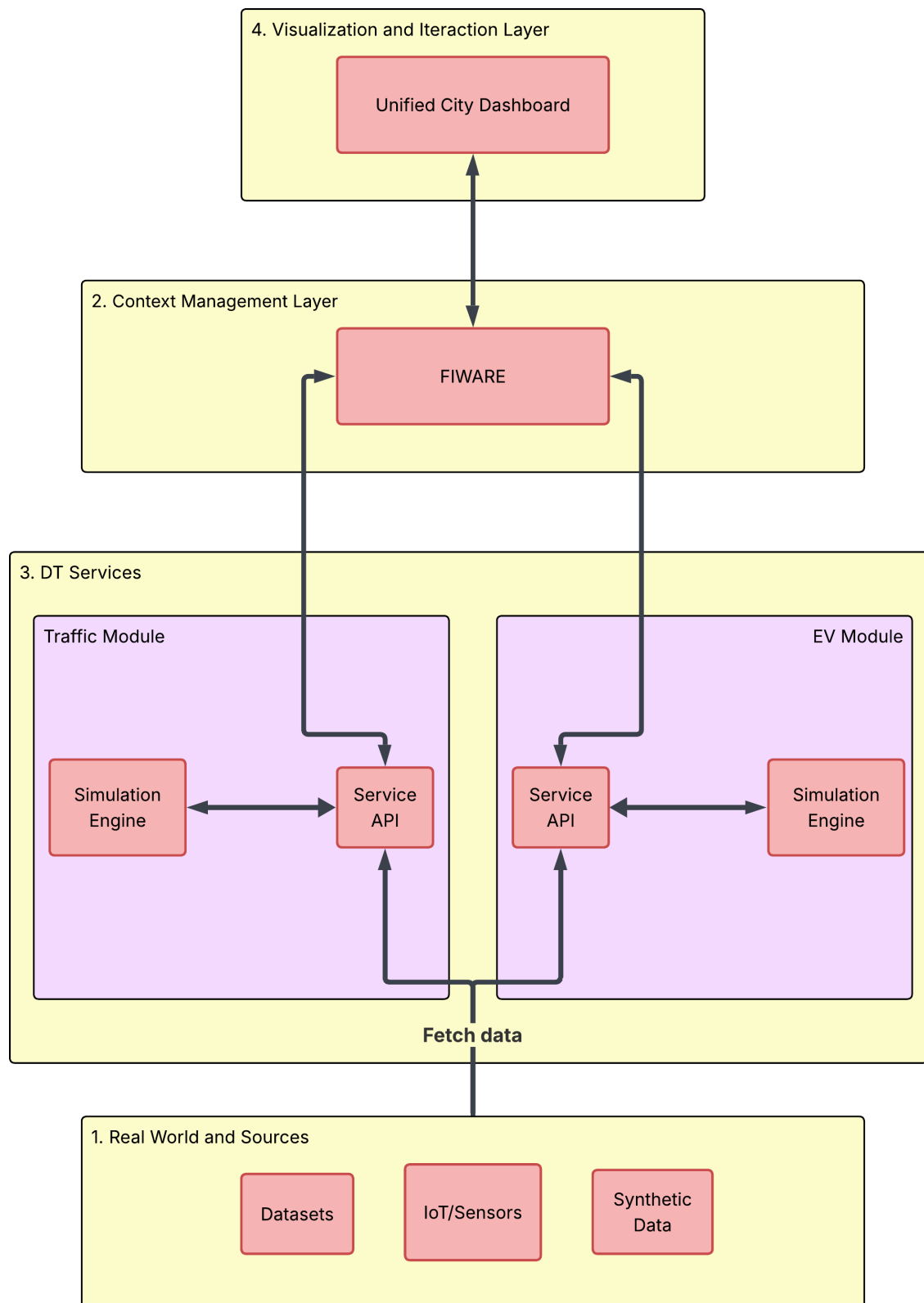


Figure 3.1: Proposed architecture

In practical terms, each simulation module is associated with a control entity that governs its execution through three operational states:

- **STOPPED:** In this state, the simulation module is inactive. It corresponds to the configuration stage, in which the user may define parameters, adjust scenario conditions, and prepare the execution of the module.
- **STARTED:** This state indicates that the simulation is running. During this phase, the module uses the previously prepared synthetic input data together with the selected configuration in order to execute the corresponding simulation process.
- **FINISHED:** In this state, the execution has been completed and the simulation outputs become available for analysis and visualization. These outputs may then be used to assess the simulated scenario, support what-if analyses, and prepare further runs with modified parameters.

Although the current implementation relies on previously prepared synthetic datasets, the adopted architecture is also designed to support the future integration of real-time data through FIWARE. This possibility emerges from the role of the middleware layer as a common context management environment, capable of receiving and updating contextual information from external sources according to a standardized representation. Under such an approach, data originating from sensors, external APIs, or other operational systems could be incorporated into the platform and made available to the corresponding simulation and visualization components.

3.3.1.1 NGSI-LD

An important aspect of this middleware layer is the adoption of the NGSI-LD information model, which provides a standardized way of representing context entities and their properties. By relying on this model, the platform is able to organize contextual information in a more structured and semantically consistent manner, which is particularly relevant in systems composed of multiple modules with distinct responsibilities.

The use of NGSI-LD also introduces important semantic and architectural advantages. First, it enables the adoption of standardized vocabularies for the representation of entities and attributes, reducing ambiguity in the interpretation of contextual information. Second, it facilitates the reuse of predefined data models, which is especially useful in application domains where common urban concepts, such as road segments, charging stations, or control entities, must be represented consistently across services. Third, it allows the definition of new vocabularies and entity descriptions whenever the existing models are not sufficient to represent the specific concepts required by the platform. For these reasons, the adoption of NGSI-LD strengthens the consistency and reusability of the proposed architecture.

Despite these advantages, the use of shared semantic models across multiple modules also introduces design challenges. In particular, when equivalent entities are used in different simulation domains, concurrent updates on the same entity may lead to semantic conflicts or unintended interference between modules. In the present implementation, this issue emerged with the `RoadSegment` entity, which was used in both the Transit Module and the Electric Vehicles Module.

To address this limitation, the representation of the road segment itself was separated from the representation of its module-specific state. In this way, `RoadSegment` continues to describe the road segment as a common urban entity, while the dynamic information associated with a specific module is stored independently. This makes it possible to keep a shared representation of the road infrastructure without allowing the internal state of one module to interfere directly with that of another.

3.3.1.2 Connecting the Dashboard with the Simulation Modules

This subsection explains how the interaction between the dashboard and the simulation modules is established within the proposed architecture. Rather than relying on direct communication between the user interface and the simulation engines, the connection is mediated through the context management layer, where user actions are reflected as updates to the corresponding entities.

For each simulation module included in the proposed architecture, FIWARE establishes the connection through a subscription-based mechanism, in which a subscription payload is used to trigger the corresponding simulation process. A simplified representation of this subscription structure is shown below:

```
{
  "description": "Trigger simulation module",
  "type": "Subscription",
  "entities": [{"type": "<SimulationControlEntity>"}],
  "watchedAttributes": ["simulationStatus"],
  "q": "simulationStatus==\"STARTED\"",
  "notification": {
    "endpoint": {
      "uri": "<module-endpoint>",
      "accept": "application/json"
    }
  },
  "@context": ["<context-url>"]
}
```

In this structure, the `description` field provides a human-readable explanation of the purpose of the subscription. The `type` field identifies the object as a subscription within the FIWARE

environment. The `entities` field specifies the type of entity that the subscription monitors, which in this case corresponds to the simulation control entity associated with the module.

The `watchedAttributes` field indicates which attribute changes should be observed, namely the `simulationStatus`. The `q` field defines an additional condition over that attribute, so that the subscription is only triggered when the status becomes `STARTED`.

The `notification` block defines how FIWARE should react once the condition is met. More specifically, the `endpoint` indicates the URI to which the notification is sent, corresponding to the service responsible for starting the simulation module. Finally, the `@context` field specifies the semantic context used to interpret the entity and its attributes according to the adopted NGSI-LD representation.

The same subscription logic is used in both simulation modules, with differences limited to the entity names, semantic context, and notification endpoint associated with each specific module.

3.3.2 Transit Module

The Transit Module was developed to simulate and analyse traffic behaviour under different operational scenarios. Within the proposed platform, it supports both an analytical mode, aimed at evaluating predefined configurations, and a live mode, intended to reflect dynamic changes introduced during execution. In this way, the module allows the assessment of how modifications in the road network, such as road closures or capacity reductions, affect traffic conditions and route adaptation.

3.3.2.1 Inputs

For MATSim to execute a scenario, five input files are required:

1. **config.xml:** Defines the global configuration of the simulation. This file specifies the paths to the main scenario inputs and determines the execution settings of the model, including scoring, routing, replanning, controller parameters, and output generation.
2. **network.xml:** File that defines the physical structure of the transport network through its nodes and connecting links. Each link is described by attributes such as its identifier, number of lanes, permitted modes, capacity, and free-flow speed. The permitted modes may include, for example, `car` and `bus` among others. In this way, `network.xml` provides the spatial and operational representation of the network over which agents travel during the simulation.
3. **plans.xml:** This file represents the daily activity schedule of each agent. Each plan is composed of a sequence of activities and travel legs, where activities are associated with a type, a location, and an `end_time`, while legs specify the transport mode used between consecutive activities. The starting time of an activity does not need to be explicitly included, as it is inferred from the completion of the previous leg. To simulate the retrieval of real-time data snapshots, this file was first converted into JSON format. At the start of each simulation, the

corresponding snapshot is fetched from an external database and converted back into XML, ensuring compatibility with MATSim.

4. **schedule.xml**: Defines the public transport service structure of the scenario. This file contains the transport lines, their routes, the associated stops, and the planned departures, thereby specifying how public transport operates over the network.
5. **vehicles.xml**: Defines the fleet used by the public transport services described in `schedule.xml`. This file specifies the available vehicle types and their main characteristics, such as capacity and operational properties, allowing MATSim to simulate public transport movements consistently with the schedule.

3.3.2.2 Analysis Mode

In this mode, users can configure a traffic scenario before execution and evaluate its effects after the simulation is completed. The main purpose of this mode is to support what-if analysis, allowing the user to introduce predefined changes, such as road closures or capacity reductions, which are applied before the simulation starts, remain fixed throughout the run, and influence the behaviour of the simulated transport system. In the current implementation, road closures are represented through a drastic reduction in link capacity rather than by completely removing the affected link from the network, since activities may still be located on the affected segment. This allows the population to continue completing their daily plans without creating infeasible routing situations.

Once the simulation is triggered, MATSim executes the scenario for a fixed number of iterations, allowing agents to repeatedly experience the transport network and adjust their routes accordingly. This adaptation process is guided by a scoring mechanism, through which each agent evaluates the utility of its daily plan. In the present implementation, the score is mainly influenced by the utility of performing activities (`performing = +6`) and by the timing of the realized daily plan, with a penalty parameter for late arrival (`lateArrival = -18`). Although activity plans are typically defined through activity sequences and `end_time` values rather than explicit `start_time` values, the actual start of each activity is inferred from the completion of the previous leg. As a result, delays still affect the realized timing and duration of activities, which in turn influences the score associated with the plan. In addition, the scoring function is also shaped by the typical duration associated with each activity type, such as `home`, `work`, `groceries`, or `school`, and by a monetary distance rate defined for public transport. Over successive iterations, agents tend to favour plans that achieve better scores, making it possible to observe how traffic patterns evolve in response to the configured scenario. This cycle is represented in 3.2.

After the execution, users may either restart the simulation with a different road configuration, such as alternative road closures or lane restrictions, or explore the results through SimWrapper¹, a web-based visualization tool that provides insights into traffic conditions, network usage, and the most frequently used roads.

¹<https://simwrapper.app/>

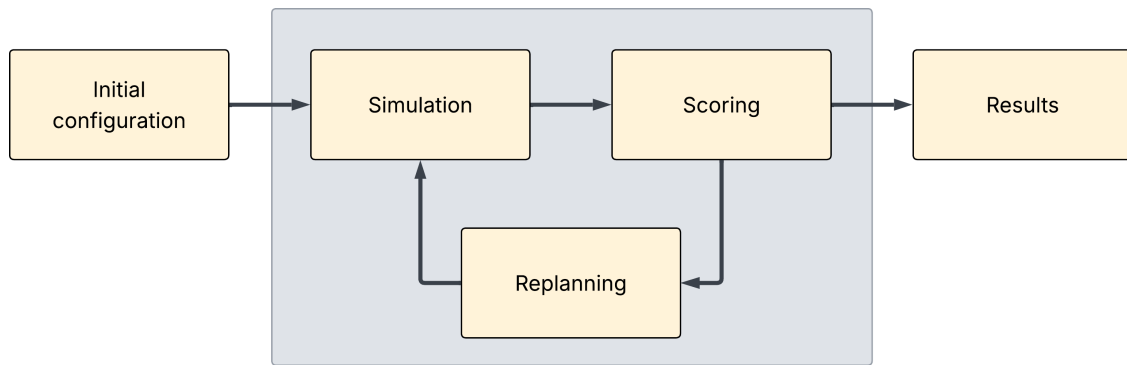


Figure 3.2: Simulation process

For public transportation, the analysis mode also incorporates the effect of road interventions on bus routes. Because these changes are processed before the simulation begins, the affected routes can be adjusted at the schedule level rather than during execution. More specifically, when a restricted road belongs to a bus route, the system attempts to generate a detour between the relevant links while preserving the planned stops. If no valid detour can be found, or if a stop is directly located on a blocked road segment, the corresponding route is excluded from that scenario. This allows the analysis to capture the effect of road interventions on both private traffic and public transport operation.

3.3.2.3 Live Mode

In Live Mode, users interact with the simulation through the map interface provided in the dashboard, where they can modify road availability during execution by closing roads or restricting the number of available lanes. This is made possible through within-day replanning, which allows agents to adapt their routes during the simulation rather than only between iterations. As a result, the effects of dynamic interventions can be observed in real time, making it possible to assess how travellers respond to sudden changes in network conditions. The within-day replanning mechanism used in this module was implemented with reference to the official MATSim withinday code examples, which provide practical examples of how agents can be rerouted during an ongoing simulation [19].

This run mode requires a continuous communication flow between the browser and the simulation module, since changes in the network must be transmitted while the scenario is being executed. To reduce the latency associated with these updates, live information is delivered through WebSockets, which provides a bidirectional communication channel for the transmission of live updates, while FIWARE is used at the beginning of the process to coordinate the simulation trigger and establish the initial communication workflow.

Through this live communication channel, the simulation periodically transmits traffic-related updates to the dashboard. More specifically, the information sent includes the current simulation time and the traffic condition associated with the network links being monitored. In the present implementation, these updates include a speed value for each link, computed as the ratio between

the current simulated speed and the free-flow speed of that link. This allows the dashboard to represent traffic conditions in real time and to show how road closures or lane restrictions influence congestion patterns across the network. To avoid overloading the communication channel with all network changes, only links whose speed changes beyond a small threshold are transmitted to the dashboard.

Within this mechanism, whenever a road is modified, the corresponding network links are updated by changing their effective capacity and number of available lanes according to the restriction defined by the user. Before each simulation step, these changes are incorporated into the routing process by updating travel times and route costs according to the new road conditions. Under this updated routing process, more restrictive links are assigned a higher travel cost and become increasingly unattractive for route choice. In the case of a road closure, the link is not removed completely from the network; instead, its effective capacity is reduced to a very small value so that network connectivity is preserved.

The simulation then periodically inspects the agents that are currently travelling and identifies those whose remaining routes still cross the affected links. For these agents, a new least-cost path is computed from their current position to their destination, replacing the previous route during execution. Furthermore, re-routing is applied probabilistically according to the severity of the intervention, so that more restrictive changes result in a higher likelihood of route adaptation. For example, if the user closes 3 lanes out of 4 on a road, the corresponding reduction in available capacity makes that link significantly less attractive, increasing the probability that agents whose remaining routes still cross it will be rerouted. In contrast, if only 1 lane out of 4 is closed, the intervention is less severe, and the likelihood of route adaptation is correspondingly lower. The probability of replanning can be represented as:

$$p_{\text{replan}} = 1 - \min_{i \in A}(f_i) \quad (3.1)$$

where A denotes the set of affected links that are still included in the remaining route of the agent, and f_i represents the capacity factor associated with link i .

However, while public transport routes can be modified in Analysis Mode, since the simulation has not yet started and the corresponding vehicle routes and stops can still be adjusted beforehand, this is not feasible in Live Mode. Once the simulation is already running, public transport operations are more constrained, and dynamic changes of this kind are not handled by the current implementation. For this reason, within-day replanning is applied only to road users with cars in Live Mode, whereas public transport adjustments are limited to the pre-execution stage.

3.3.3 Electric Vehicles Module

With the growing adoption of electric vehicles, charging stations have become a necessary element in cities seeking to support a more sustainable and electrified mobility system. In this context, the Electric Vehicles Module was developed to study the behaviour of charging infrastructure

under different scenario configurations, allowing the assessment of factors such as station location, charging capacity, waiting conditions, and infrastructure usage.

3.3.3.1 Inputs

For this module, the input files are almost the same as those used in the Transit Module, since the Electric Vehicles Module is built on top of the same MATSim scenario structure. In particular, the simulation still relies on the general configuration file, the transport network, the daily plans of the agents, and the public transport schedule and fleet definitions.

The main additional input required in this case is the **chargers.xml** file, which defines the charging infrastructure available in the city. More specifically, this file specifies, for each charger, its identifier, the link corresponding to the road segment where it is located, the plug type, the number of available plugs, and the plug power. In the present implementation, only default plug types were considered in order to simplify the simulation setup.

This file is generated when the simulation is triggered by converting the `EVChargingStation` entities stored in FIWARE into a MATSim-compatible `chargers.xml` representation. This allows the charging infrastructure configured through the web platform to be automatically incorporated into the simulation scenario at execution time.

3.3.3.2 Analysis Mode

This analysis mode was inspired by previous work on evolutionary bi-objective optimization for electric-vehicle charging infrastructure planning, where simulation is combined with evolutionary algorithms to evaluate alternative charging-station layouts according to multiple objectives [3]. Similarly, in the proposed platform, the simulation is used as part of an optimization process aimed at identifying suitable charging infrastructure configurations. The analysis evaluates alternative solutions according to two main criteria, namely total infrastructure cost and average waiting time, making it possible to determine which combinations of station allocation, number of ports, and plug power provide better overall performance.

In the dashboard, the user can define the percentage of the population that will use electric vehicles instead of conventional vehicles, place and remove candidate charging-station locations on the road segments shown on the map, and specify how many of those candidate locations must be selected by the optimization algorithm. The user can also define the minimum and maximum number of ports per station, the available plug power values, and the monetary cost associated with each station, each port, and each kilowatt of installed power. A higher plug power reduces charging time, but also increases the total cost of the solution.

From an implementation perspective, the analysis mode follows a complete execution flow that starts in the main simulation runner and ends with the evaluation of candidate charging configurations. Once the user triggers the process, the runner loads the scenario and the analysis parameters defined in the dashboard, and then initializes the optimization routine. Each candidate solution is translated into a concrete charging-station configuration, applied to the MATSim scenario, and

evaluated through simulation after adapting the population to include electric-vehicle users and charging activities. During execution, charging-related waiting times are monitored, and together with the infrastructure cost they form the two objective values used to assess the solution. By repeating this cycle over multiple alternatives, the optimization process is able to identify the most suitable configurations according to the selected criteria.

At the beginning of this process, the system reads the scenario configuration together with the optimization parameters defined by the user in the dashboard. Based on these inputs, the optimization routine starts exploring alternative configurations. Each candidate solution is represented by a specific combination of decision variables associated with the charging infrastructure. These variables determine which candidate locations are activated, how many charging ports are assigned to each selected station, and which plug power is used in each case. In this way, every solution corresponds to a concrete charging-network configuration that can be instantiated in the simulation scenario and evaluated.

Before each evaluation, the scenario is adapted so that a predefined share of the population is represented as electric-vehicle users. At this stage, the corresponding vehicle attributes are introduced and charging activities are incorporated into the daily plans of the selected agents. In the current implementation, this charging activity is inserted into the last car trip identified in the selected daily plan, so that each user is guaranteed one charging opportunity within the simulated day. This allows the simulation to explicitly reproduce charging demand under the infrastructure configuration currently being tested. Although, in a real-world scenario, not every electric-vehicle user would necessarily need to charge on a given day, we assumed that each agent assigned to the electric-vehicle fleet and performing at least one car trip during the daily plan requires one charging opportunity. If an agent does not perform any car trip during the day, we don't add this charging activity to his plan. Also, it is important to note that each vehicle starts with a randomly assigned state of charge. Furthermore, since charging sessions do not always correspond to full battery recharges, the maximum duration of each charging activity was limited to 20 minutes in order to represent a more realistic intermediate charging stop.

Once the scenario has been prepared, the simulation is executed and the charging performance of the candidate solution is assessed. In particular, the evaluation focuses on two objective functions: the total infrastructure cost and the average waiting time experienced by electric vehicles when accessing charging stations. By repeating this procedure for multiple candidate solutions, the optimization process is able to identify the configurations that provide the most suitable trade-offs between economic cost and operational performance. At the end of the process, the analysis does not return a single solution, but rather a set of alternative configurations (maximum 3) with different trade-offs between infrastructure cost and charging performance, providing the user with multiple planning possibilities from which a final decision can be made based on the most relevant criteria. A representation of this cycle can be viewed in Figure 3.3.

Since there are too many possible combinations of charging-station locations, port numbers, and plug power values, testing every scenario would be impractical. Therefore, an optimization algorithm is used to search for good solutions more efficiently, focusing on the best trade-offs

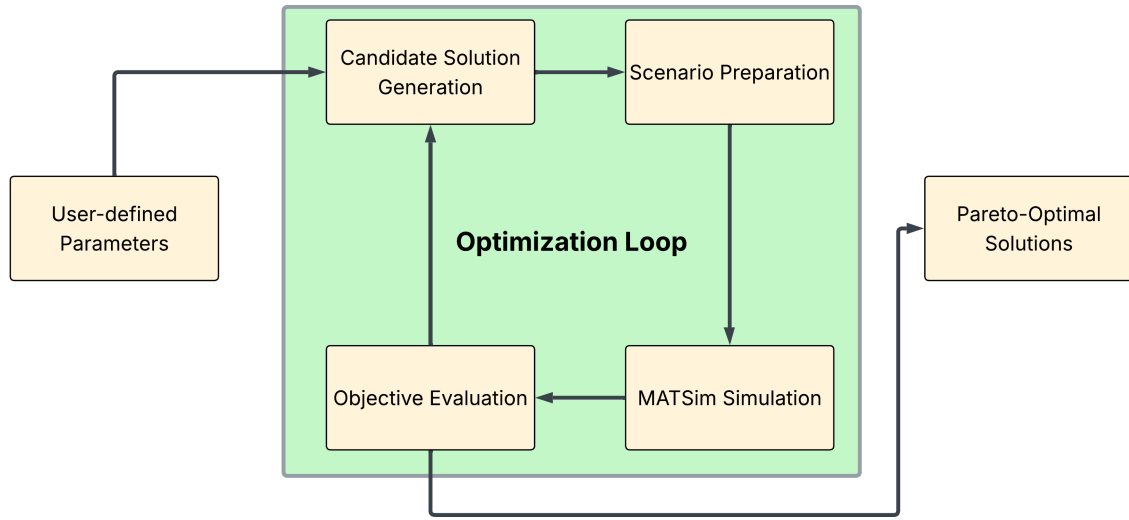


Figure 3.3: EV optimization process

between infrastructure cost and waiting time. In this work, the selected optimization method is **NSGA-II**, a multi-objective evolutionary algorithm suited to problems in which more than one objective must be optimized simultaneously, by performing the following equation:

$$\text{selectionScore} = \sqrt{(\text{totalCost})^2 + (\text{waitingTime})^2} \quad (3.2)$$

The average waiting time is computed by tracking each charging session individually. For every electric vehicle that needs to charge, the waiting time corresponds to the time elapsed between the moment the vehicle joins the queue at a charging station and the moment charging effectively starts. The final metric is obtained as the average of these waiting times across all charging sessions observed during the simulation. In this case, the analysis aims to minimize both total infrastructure cost and average waiting time, which means that no single solution is necessarily optimal in every aspect. Instead, some solutions may be cheaper, while others may provide better charging performance. These alternative solutions form the **Pareto front**, that is, the set of non-dominated solutions for which no objective can be improved without worsening the other. This allows the user to compare different configurations and choose the one that best fits the intended planning criteria. For these scenarios, we considered that both the cost and the time have the same importance, however, for other case studies, one factor might be more important than the other, therefore the **selectionScore** equation must be adjusted accordingly.

3.3.3.3 Live Mode

In Live Mode, the user can test custom charging-infrastructure configurations by defining charging-station locations and their associated parameters directly through the platform. In contrast with the analysis mode, which searches automatically for efficient solutions, this mode is intended for interactive what-if exploration, allowing the user to assess the operational behaviour of a user-defined scenario.

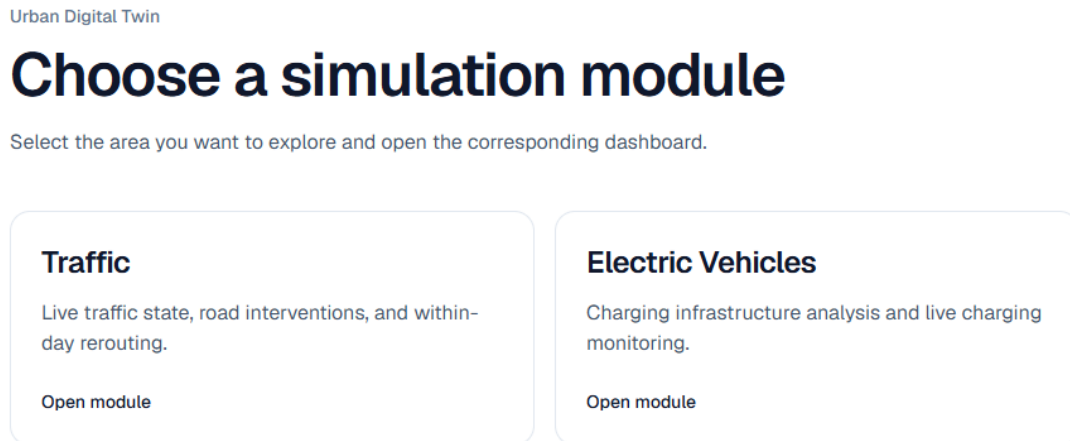


Figure 3.4: Home page

Before execution, the user can place charging stations on the road segments displayed on the map and define the main parameters of each station, namely the number of charging ports and the plug power. These stations are stored in FIWARE as `EVChargingStation` entities and, when the simulation is triggered, they are converted into a MATSim-compatible `chargers.xml` file.

Once the simulation starts, the behaviour of this system can be monitored in real time. More specifically, the module provides live information about the state of each charging station, including its occupancy, the number of vehicles currently charging, the number of vehicles waiting in queue, and whether the station is effectively full. This allows the user to observe how the configured infrastructure performs under the simulated demand conditions.

As in the Transit Module, this run mode requires a continuous communication flow between the browser and the simulation module, since charging-state information must be transmitted while the scenario is being executed. To support this interaction effectively, live updates are delivered through WebSockets, enabling the dashboard to receive real-time information about station occupancy and queue conditions. FIWARE, in this case, is mainly used to coordinate the simulation trigger and to provide the charging-station entities required to build the simulation inputs before execution.

3.4 Web-Visualization Platform

The web-visualization platform is the point of interaction between the user and the proposed system. Through this interface, users can configure scenarios, trigger simulations, and inspect the outputs generated by each module in an intuitive and interactive way. At the home page, the user can choose the module, and, clicking on one of the options, he will be redirected to the module page, as shown in Figure 3.4.

In both modules, the interface provides dedicated controls to execute the simulation either in Analysis Mode or in Live Mode. Each module page also includes an interactive map and a set

of configurable parameters, allowing the user to define the scenario directly through the platform before triggering the simulation. In the Transit Module, the user can interact with the road network to define traffic-related scenarios, whereas in the Electric Vehicles Module it is possible to select charging-station locations and configure parameters such as the EV population percentage, the number of ports, and the charging power. After the configuration is completed, the simulation can be launched from the interface and the corresponding outputs are presented visually through the platform.

The map visualization is supported by GeoJSON files stored in the public resources of the web application. These files contain the road network representation of the selected city and are used to render the spatial layout shown in the interface, enabling direct interaction with the corresponding road segments.

3.5 Chapter Summary

In this chapter, we presented a multi-service Digital Twin methodology designed to overcome the limitations identified in the literature review. The proposed architecture relies on a multi-layered design that integrates FIWARE as an NGSI-LD context management broker, decoupling system components and ensuring semantic interoperability. Through the implementation of the Transit and Electric Vehicles modules, supporting both analysis and live execution, we demonstrated how heterogeneous urban services can co-exist within a single platform supported by a unified web dashboard.

Chapter 4

Results and Evaluation

This chapter presents the results obtained from the experimental evaluation of the proposed platform. The assessment focuses on the two simulation modules developed in this work, namely the Transit Module and the Electric Vehicles Module, considering both their functional behaviour and their usefulness in supporting scenario analysis.

4.1 Experimental Setup and Objectives

The experimental evaluation was conducted to assess the behaviour, applicability, and flexibility of the proposed platform in the context of urban mobility analysis. In particular, the tests focused on the two main modules developed: the Transit Module, designed to analyse traffic conditions and road network performance, and the Electric Vehicles Module, intended to support the study of charging infrastructure configurations.

The experiments were carried out using the real transport infrastructure of Porto, including the road network of the metropolitan area and the actual public transport routes. Since it was not possible to access real-time individual mobility data, travel demand was represented through a previously generated realistic synthetic population of the city.

Both modules were executed using the same base input data. In particular, the original plans file, representing a realistic synthetic population of 231,799 inhabitants, was adopted as the common demand input. To make the experiments more manageable in terms of execution time and computational cost, a random 20% sample of this population was selected for the test scenarios.

To maintain consistency between travel demand and network supply, the MATSim configuration was adjusted accordingly. More specifically, the parameter `flowCapacityFactor` was set to 0.2, scaling the effective flow capacity of the network links, that is, the number of vehicles that can pass through a link over time. Similarly, `storageCapacityFactor` was also set to 0.2, reducing the effective storage capacity of each link, namely the number of vehicles that can simultaneously occupy it.

Later, after the initial experiments had been completed, additional tests were carried out for the Traffic Module using the full synthetic population. These tests were executed on a more powerful

remote machine, which made it possible to assess the behaviour of the platform under a more computationally demanding scenario and to compare the results obtained with the reduced 20% sample against those produced with the complete population.

To start the experimental workflow, the FIWARE was first deployed using Docker, thereby providing the context management and communication layer required by the platform. After that, each simulation module was launched in a separate terminal, with each one registering its own controller entity in FIWARE. This setup made it possible to emulate a distributed architecture, in which different modules operate independently while remaining coordinated through the same middleware layer.

From an experimental perspective, this approach was important not only to execute the simulations, but also to validate the interoperability of the platform components. In particular, it allowed testing how different services, potentially running on different machines, can exchange state information, receive execution parameters, and publish their results through FIWARE to be seen in the visualization platform.

With these experiments, we aim to evaluate the platform from both a functional and an analytical perspective. First, the goal is to verify whether the proposed architecture is capable of coordinating independent simulation modules through FIWARE, ensuring correct communication between the dashboard, the middleware, and the simulation services. Second, the experiments aim to assess whether the Transit Module and the Electric Vehicles Module are able to produce meaningful and interpretable outputs with different user inputs. Finally, the evaluation seeks to determine whether the platform can be adapted to a different city by changing only the required input data, without requiring structural modifications to the overall system.

4.2 Testing Transit Module

To evaluate the Transit Module, we started with the Analysis Mode, where the user can define road-network disruption scenarios and analyse their impact on traffic conditions, with the final results being presented through a set of traffic statistics and visual indicators describing the behaviour of the simulated network.

For the case study considered in this evaluation, the analysis began with a baseline simulation in which no roads were blocked. After it finishes, with SimWrapper, we can analyse several aspects, ranging from technical indicators to more functional ones, including, for example, the total execution time required to run the scenario. Looking at the traffic volume in the metropolitan area (Fig. 4.1), the links with higher traffic volume are shown in red. As expected, the Via de Cintura Interna (VCI), indicated by the blue arrow in the figure and described in [20] as one of the busiest roads in the country, appears coloured in red. Moreover, the A3, indicated by the green arrow in the figure, is also among the roads with the highest traffic volume, with values as high as 7104 cars passing in this segment during the day. Based on this observation, a what-if scenario was considered in which this road is closed in the direction from Braga towards Porto, that is, in the direction approaching the VCI.



Figure 4.1: Normal porto traffic

To perform this experiment, in the dashboard, three roads were blocked: the two access links from the A4 to the A3, together with the main A3 segment immediately before the junction with A4 in the direction from Braga to Porto (Fig. 4.2). After the execution, it is possible to compare the two scenarios and verify that the link connecting to the **VCI** now carries only 416 vehicles over the course of the day. This represents a substantial reduction in traffic flow, showing that the closure of the three selected links was effective in strongly discouraging the use of that corridor. A small residual flow is still observed because, as discussed in the methodology chapter, road closures are implemented in the model as a drastic capacity reduction rather than as a complete removal of the link from the network, which means that the affected segment remains technically usable in a very limited number of cases, so that activities can still happen within the blocked link.

We can now compare these two scenarios and draw some conclusions. When comparing the utilization percentage of the links at each scenario (Fig. 4.3), which represents the degree to which each link is used relative to its available capacity during the peak hour, we can observe that the A3 links went from red to blue, which is expected since the accesses to that road are closed (black links in the image). However, the road segment located to the right of the A3, running parallel to it (pointed by the purple arrow), show a clear increase in utilization, with the whole segment becoming red in that scenario. This indicates that vehicles were forced to use alternative roads to reach their destinations, increasing the traffic pressure on neighbouring links, as would happen in a real world scenario.

When analysing the score of the two scenarios, the baseline scenario achieves a higher average score (68.99) than the modified scenario (62.09), which is expected. Since the road intervention introduces additional restrictions in the network, agents are more likely to experience longer or less convenient routes, which negatively affects the utility of their daily plans. A similar pattern can be observed when analysing the number of stuck agents, that is, agents who were unable to complete their daily plans within the simulation time. In the baseline scenario, 1011 agents did not finish their plans, whereas in the modified scenario this number increased to 1624. This result is consistent with the previous indicators, since the road intervention introduced additional constraints in the network, increasing travel difficulties and making it harder for a larger number of agents to complete their planned activities in time. Overall, the obtained results show that the closure of the



Figure 4.2: Blocked roads in the dashboard



Figure 4.3: Comparison of traffic conditions at peak hour for the baseline scenario (top) and for the scenario with the three selected links closed (bottom)

selected links had a clear impact on network performance. Although the intervention successfully reduced traffic flow along the targeted corridor, it also increased pressure on surrounding roads, lowered the average score of the population, and raised the number of stuck agents.

For the final phase of the analysis mode testing, the simulation was executed using the full synthetic population. The main objective of this experiment was to verify whether the reduced 20% population used in the previous tests was still able to provide a realistic representation of traffic behaviour in Porto. For that purpose, the results obtained with the reduced sample were compared with those produced by the full-demand scenario, allowing us to evaluate whether the main traffic patterns, congestion areas, and performance indicators remained consistent when increasing the scale of the simulation. Comparing the traffic volume of both scenarios (Fig. 4.4), it is possible to observe that the overall spatial patterns are very similar, particularly along the VCI (blue arrow) and the A3 (green arrow). Some local differences can still be observed, with certain links presenting different colour classes, which is expected due to the random sampling process and the lower number of simulated agents in the reduced scenario. However, the main high-volume corridors remain consistent, suggesting that the 20% population sample is able to represent the major traffic patterns of the full-demand scenario with reasonable accuracy.

Looking at the score, there is a clear difference between the two scenarios: the reduced population achieved an average score of 68.99, whereas the full population achieved an average score of 92.97. This difference can mainly be explained by the fact that the two simulations are not fully equivalent in terms of network capacity. In the reduced scenario, the population was sampled to 20%, and the network capacity was also scaled down using the corresponding MATSim capacity factors. Although this keeps demand and supply proportional at a global level, it can still introduce local effects, since some links may become more restrictive and the sampled population may not preserve all trip distributions perfectly. As a result, some agents may experience less efficient routes or additional delays, reducing the utility of their executed plans. In contrast, the full-population scenario was executed with the complete network capacity, which provides a more stable representation of demand and supply and leads to fewer artificial effects caused by sampling. For this reason, the score values should not be interpreted as a direct indication that one scenario represents better traffic conditions than the other, but rather as a consequence of running the same mobility demand at different levels of aggregation and network scaling.

Overall, this comparison indicates that the reduced 20% population is suitable for exploratory experiments, since it preserves the main spatial traffic patterns while significantly reducing execution time. However, absolute performance indicators, such as the score, should be interpreted with caution when comparing reduced and full-population scenarios, as they are affected by the sampling process and by the corresponding network scaling assumptions.

Besides the analysis mode, the Traffic Module also provides a live mode, where users can interact with the road network while the simulation is running. As shown in Fig. 4.5, the road segments are displayed in green, indicating that they are active and currently present low traffic flow according to the live simulation output. Selecting a road segment opens a details panel with information about the street name, identifier, current status, and available lane capacity. From

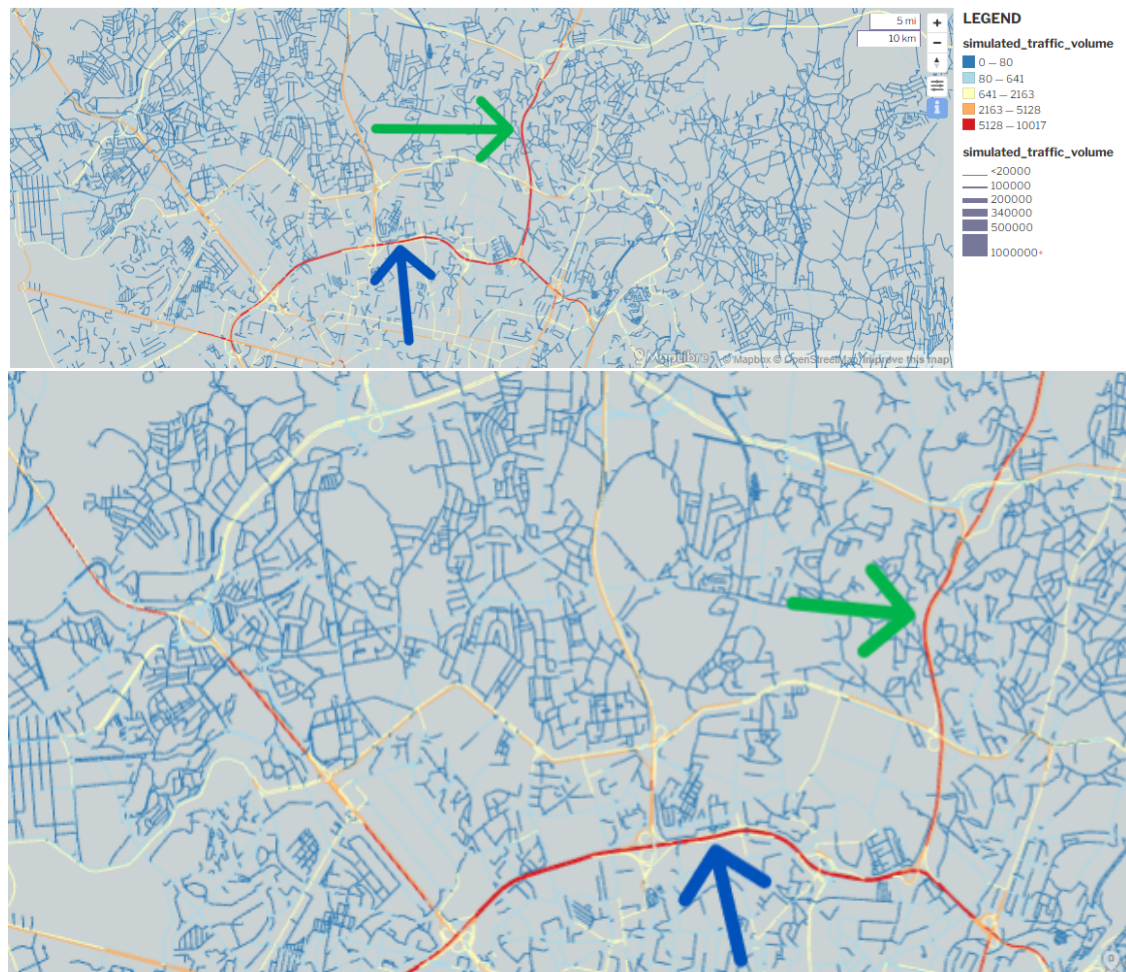


Figure 4.4: Comparison of traffic volume with 20% of the full population (top) and full population (bottom)



Figure 4.5: Live traffic

this panel, the user can reduce the number of available lanes or fully close the selected road. These changes are then sent to the simulation during execution, allowing the platform to represent dynamic traffic interventions and observe how the network reacts in real time.

4.3 Testing EV Module

The evaluation of the Electric Vehicles Module started with the Analysis Mode, in which the system searches for the best charging-station configuration among the alternatives defined by the user in the dashboard. In this mode, the user selects the candidate station locations and defines the main configuration parameters, such as the number of stations to activate, the number of charging ports, the available charging powers, and the associated infrastructure costs. Based on these inputs, the system tests different combinations and evaluates them according to total cost and average waiting time.

For the case study considered in this experiment, the candidate locations were defined based on 10 GALP stations in the city of Porto (Fig. 4.6). From these 10 possible sites, the objective was to select 5 locations to host electric-vehicle charging infrastructure. So we selected all of them in the interactive map (Fig. 4.7) (The green segments indicate the locations where these stations are), and then we use the following configuration (which can be seen in Fig. 4.8):

- **EV population percentage:** 20%
- **Stations to select:** 5
- **Minimum number of ports:** 2
- **Maximum number of ports:** 10
- **Available power levels:** 20kW, 50kW, 150kW and 250kW
- **Base cost / station:** 50000€



Figure 4.6: Galp real locations

- **Cost / port:** 8000€
- **Cost / kW:** 200€

To analyse the behaviour of the optimization process under different search depths, the experiment was divided into two slightly different sub-experiments. In both cases, the Electric Vehicles Module relied on the NSGA-II algorithm, configured with a population size of 8 individuals. This means that, at each generation, the algorithm maintains 8 candidate solutions, which are then combined and modified in order to explore new alternatives while preserving the most promising ones according to the Pareto dominance principle, in which a solution is considered better if it improves one objective without worsening the other, or if it performs better in at least one objective while being no worse in the remaining ones. The lower mutation rate was used to introduce only small random changes, helping to prevent premature convergence to a solution that may be good, but not globally optimal, while also avoiding an unstable search process.

In the experiments, the algorithm used a HUX crossover rate of 0.9 and a bit-flip mutation rate of 0.02. This high value of crossover was chosen to encourage the combination of characteristics from different solutions, therefore intensifying the search around promising regions of the solution space.

The two sub-experiments differed in the number of evaluations allowed to the algorithm. In this context, one evaluation corresponds to the assessment of one candidate configuration, meaning that a complete MATSim simulation is executed for that solution in order to obtain its total cost and average waiting time. Since the population size was set to 8, the first 8 evaluations correspond to the initial population, while the subsequent evaluations correspond to new candidate solutions generated by the evolutionary process. In the first sub-experiment, the process was limited to 40 evaluations, resulting in a relatively short search. In the second, the limit was increased to



Figure 4.7: Galp locations at the iterative map

SIMULATION

EV POPULATION (%) 20% STATIONS TO SELECT (MAX 10)

PORTS PER STATION

MINIMUM PORTS MAXIMUM PORTS

PLUG POWER OPTIONS

SELECT AVAILABLE POWER LEVELS (KW)

Higher power = faster charging but higher cost

COST PARAMETERS (€)

BASE COST / STATION COST / PORT COST / KW

Example: 1 station × 8 ports × 50 kW → €124 000

Figure 4.8: Configuration used

Table 4.1: Top three solutions obtained in the first experiment

Solution	Total Cost (€)	Average Waiting Time (min)	Charging Sessions
First solution	1050899	14.26	7421
Second solution	811715	14.83	7585
Third solution	1026899	14.63	7417

80 evaluations, allowing a broader exploration of the solution space and a more representative approximation of the Pareto-optimal front, at the expense of a higher computational cost.

During this testing phase, it was also observed that increasing the number of evaluations could lead to solutions with fewer charging sessions (For instance, initial tests with 80 evaluations yielded solutions with around 3500 charging sessions, whereas 40 evaluations resulted in up to 6500). This happened because the optimization process was initially based only on the two main objectives, namely total infrastructure cost and average waiting time. As a result, the algorithm could favour configurations with low cost and low average waiting time even when a considerable number of vehicles did not effectively start charging before the end of the simulation. This could produce misleading results, since the average waiting time was calculated only for vehicles that actually initiated a charging session. To avoid this behaviour, an additional constraint was introduced: only solutions in which at least 85% of the planned charging activities resulted in an effective charging session were accepted. This ensured that the selected solutions were not only efficient in terms of cost and waiting time, but also capable of serving a sufficiently large share of the charging demand. For the first experiment, with 40 evaluations, 3 solutions were provided, with the results presented in table 4.1. For the second, with 80 evaluations, 3 other solutions were obtained (see table 4.2.)

It is important to note that the number of solutions presented at the end of the optimization process does not necessarily correspond to the number of tested configurations. The final result only includes the feasible non-dominated solutions identified by the algorithm. Therefore, even if 40 or 80 candidate configurations are evaluated, only a smaller subset may remain in the final Pareto front. In some cases, this could result in only two or even one final solution being provided. However, in the experiments conducted here, three feasible non-dominated solutions were obtained in each case.

As can be observed, increasing the number of evaluations from 40 to 80 led to a clear improvement in the obtained solutions. The solutions from the second experiment present substantially

Table 4.2: Top three solutions obtained in the second experiment

Solution	Total Cost (€)	Average Waiting Time (min)	Charging Sessions
First solution	805205	14.69	7927
Second solution	797205	14.85	7925
Third solution	789205	14.92	7925

Analysis Results

Browse the top-ranked charging configurations generated by the optimisation process.

< Solution 1 / 2 >

TOTAL COST
€656 150

AVERAGE WAITING TIME
14.44 min

CHARGING SESSIONS
6903



Figure 4.9: Solution example

lower infrastructure costs, while maintaining average waiting times within a very similar range. The number of charging sessions is also slightly higher in the 80-evaluation experiment. However, this should be interpreted as a result specific to the tested solutions rather than as a direct consequence of increasing the number of evaluations, since the number of completed charging sessions may vary according to the selected station locations, port allocation and charging powers. Therefore, the main conclusion is that the longer optimization process allowed the algorithm to identify cheaper configurations without substantially worsening the average waiting time. The average waiting time obtained in these experiments can be considered relatively high, as it indicates that an electric-vehicle user may spend around 14 minutes waiting before charging can begin, in addition to the charging stop itself, which may last up to 20 minutes. However, this value should be interpreted in the context of a demanding scenario, since the experiment assumes that every electric-vehicle user with at least one car trip requires a charging session during the day. Under this assumption, relatively high waiting times are expected.

After the analysis process is finished, the user can see in the dashboard these solutions and the respective configuration. The map will be presented with the selected links, and correspondent configuration, with green colour, and the links not selected with red, as we can see in Fig. 4.9. Right next to the map, it is listed the configuration of the selected stations, where the user can see the number of ports and the charging power of each one.

For the manual configuration, where the user can configure the parameters of each station, we used the same locations of the previous experiment. In this case, 10 stations were used, all with 7 ports and a charging power of 50 kW. After the simulation is running, the user can see the charging infrastructure as a whole, with a total of 70 spaces to be used, as well as the total vehicles waiting

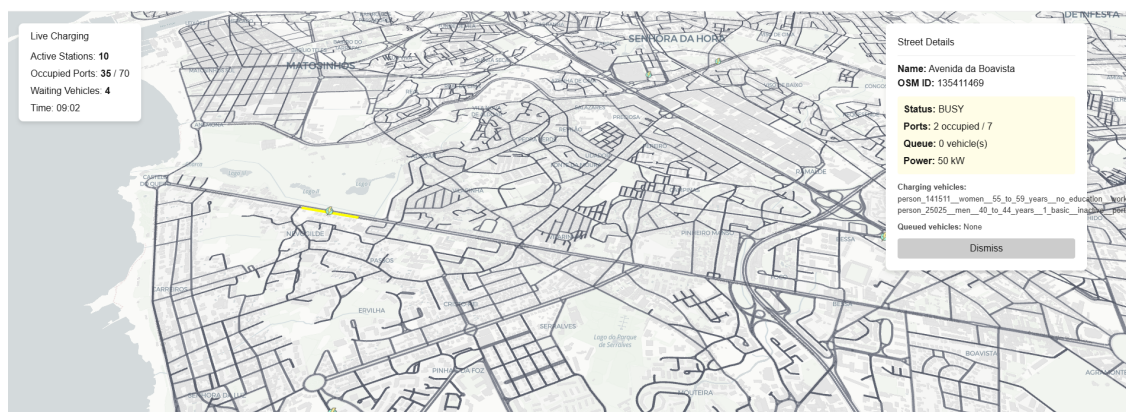


Figure 4.10: EV station with some vehicles

at that time of the day. After clicking in one of the stations, it is possible to see the number of occupied ports of the station, and if there is any queue of vehicles waiting for charging. (Fig. 4.10 and Fig. 4.11)

After the simulation is completed, the dashboard presents the final results of the tested configuration, namely the total infrastructure cost, the average waiting time, and the total number of charging sessions. For this experiment, the defined configuration resulted in a total cost of 1,383,364€, an average waiting time of 12.47 minutes, and 8234 charging sessions. As expected, using all 10 candidate locations with 7 ports per station and a charging power of 50 kW led to a relatively high infrastructure cost, but it also provided a broad charging capacity across the network. Consequently, because this configuration offers a larger charging capacity than the previously tested mode, it leads to a lower average waiting time and a higher number of charging sessions. In particular, while the manual configuration uses 10 stations with 7 ports each, resulting in a total of 70 available charging ports, the previous experiment allowed the selection of only 5 candidate locations, with a maximum of 10 ports per station, meaning that at most 50 ports could be allocated.

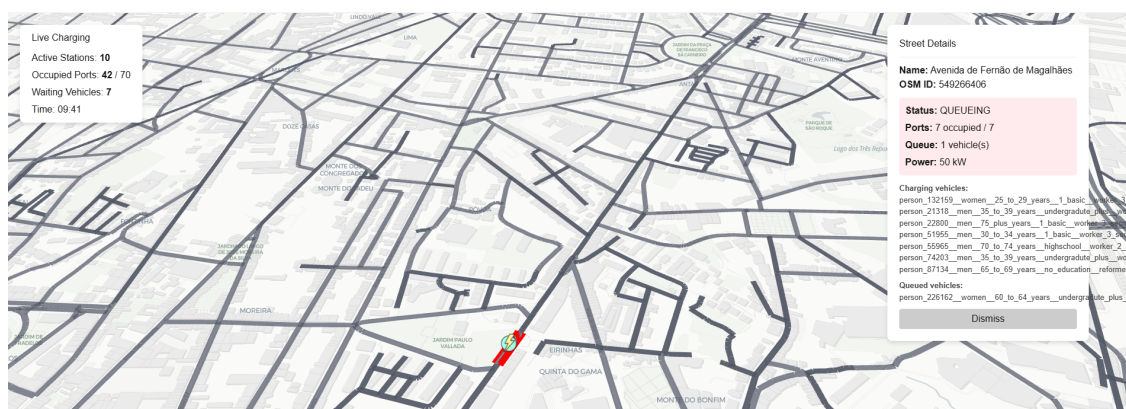


Figure 4.11: EV station with queued vehicles

4.4 Testing the Platform in Another City

For this experiment, we replicated the traffic simulations previously conducted for the city of Porto using the city of Lisbon instead. To achieve this, we extracted Lisbon’s network from the Portugal OSM file, which contains the road network for the entire country. From this dataset, we isolated the Lisbon metropolitan area in order to generate the network file. For the plans file, we used artificial intelligence to generate 100 synthetic plans for demonstration purposes. To run the simulation, we set the flow capacity factor and storage capacity factor to 0.2 and 0.1, respectively. In addition, since we were unable to obtain the public transport schedules for Lisbon, we set `useTransit` to false, meaning that agents could not use public transportation to travel between activities. Finally, from Lisbon’s network, we generated the corresponding GeoJSON file so that users could select and manipulate the desired roads, just like in the experiments conducted for the city of Porto.

With the new inputs prepared, switching the platform from Porto to Lisbon simply required replacing the active simulation input files with the Lisbon ones. More specifically, we just need to replace the 5 files required by MATSim previously mentioned in section 3.3.2.1. It is also necessary to replace the `link_dict.json` file, which stores the mapping between network links and their associated road information, as well as the `network.geojson` file used by the web interface. Lastly, the environment configuration file must also be updated so that, instead of pointing to Porto’s plans snapshot, it references the Lisbon snapshot stored in the external database. All of these steps can be carried out through a simple script that moves the required files to the appropriate locations, without requiring any changes to the simulation code itself.

After switching cities, a simple experiment was conducted to verify whether the simulation modules continued to operate correctly in a different urban context. To do so, a simulation was first executed in Analysis Mode within the Transit Module, and the resulting traffic patterns were inspected in order to identify the most heavily used links. As shown in Fig. 4.12, the blue arrow highlights the road segment with the highest observed traffic volume. Using the dashboard adapted to this urban context, we selected this road and blocked its traffic flow. After running a second time with this modification, we can observe in Fig. 4.13 that the link is now in orange colour, which indicates a clear reduction in traffic volume on that link. However, it does not turn blue because the road closure is represented in the model as a severe capacity reduction rather than as a complete removal of the link, allowing a small residual flow to remain, since activities might still happen in the closed link. It is also possible to observe that other links changed from blue to orange, indicating that they were used as alternative routes by agents in response to the road closure.

4.5 Results Summary

Based on the experiments carried out, it can be concluded that both the Transit Module and the EV Module operate as expected, for the purposes that they were designed. In the Transit Module, the results showed that the platform is capable of representing the impact of road interventions



Figure 4.12: First results, normal simulation

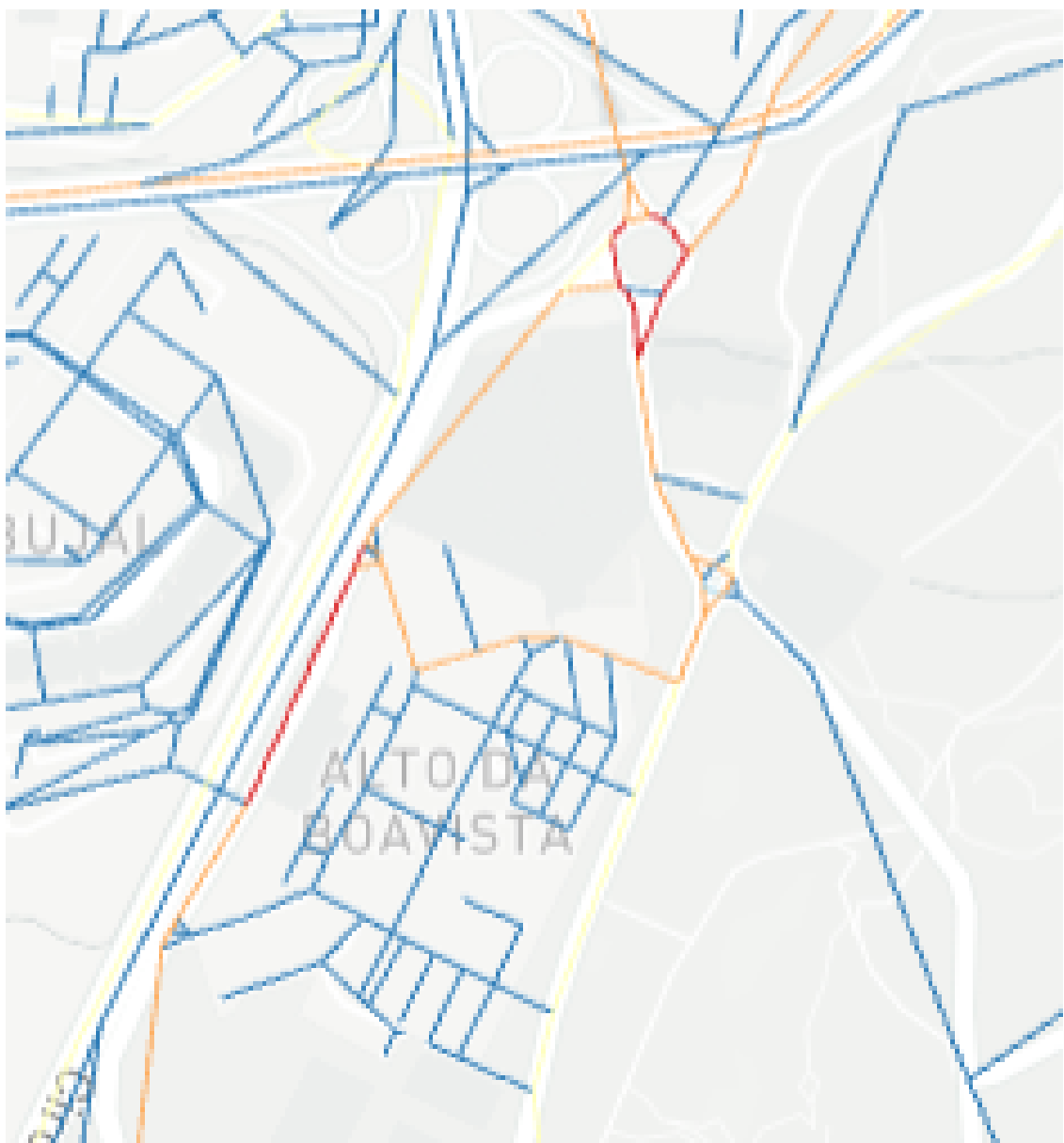


Figure 4.13: Results with the blocked road

on network performance, making it possible to observe both the reduction of flow in the affected corridors and the redistribution of traffic to alternative routes. These changes were also reflected in other simulation indicators, such as the average score of the population and the number of stuck agents, confirming that the imposed disruptions affected the overall quality of travel conditions. The EV Module demonstrated its ability to support charging-infrastructure planning by identifying efficient configurations through optimization, improving the considered objectives, namely total infrastructure cost and average waiting time. In addition, it allows the user to define manual configurations and observe their behaviour live during simulation execution.

At the same time, it can be observed that both modules operate, as expected, as independent simulation components, while remaining integrated within the same platform architecture through FIWARE, which supports the exchange of information with the frontend. When real-time communication is required, particularly for transmitting a large number of data packets within short time intervals, FIWARE is used to coordinate the initial interaction between the web platform and the simulation module, establishing the execution workflow and enabling the live communication process. After this initial coordination step, the continuous exchange of high-frequency updates is handled through WebSockets, which are better suited for low-latency communication.

4.6 Towards a Theoretical Methodology

With the wide vocabulary offered by FIWARE across multiple domains, ranging from agriculture to health, the platform can more easily support the creation of new services from scratch, since data models and communication structures are already available for many different application contexts. This facilitates the extension of the system beyond the modules developed in this dissertation, making it suitable for the integration of other smart city applications.

To integrate a new module into this platform, the first step would be to identify the relevant entities and relationships of the target domain and map them to existing FIWARE data models whenever possible. For example, if we want to integrate a water management module in Porto, such as the one described in 2.2.6, the Data and Integration Layer would first require the representation of the main domain elements, such as water-monitoring stations, river segments or flood-prone areas, as FIWARE entities, if feasible, otherwise, new terms could be introduced into the vocabulary. These data could then be collected from IoT devices and converted into formats suitable for simulation and visualization. The second step would correspond to the Spatial Analysis and Modelling Layer, in which a simulation or analytical module would process the available information in order to support predictive analysis or what-if scenarios, similarly to the modules developed in this dissertation. Finally, the results would be transmitted from the simulation module to FIWARE, from which the Visualization Layer could retrieve the relevant information and present it to the user. This illustrates how the same platform architecture can be reused to support additional urban domains while keeping each module independent and not interfering with each one.

4.7 Chapter Summary

In this chapter, we presented the experimental evaluation of the proposed platform across different urban scenarios and service domains. The results demonstrated that the Transit Module effectively models the impact of road network disruptions, capturing traffic flow reductions, route adaptations, and shifts in network-wide performance metrics. Similarly, the Electric Vehicles Module successfully optimized charging station placement and capacity using the NSGA-II algorithm, balancing total infrastructure cost against user waiting times. Furthermore, replicating the traffic simulation for the city of Lisbon confirmed the platform's adaptability to different geographic contexts by merely swapping simulation input datasets. Overall, these experiments validate the functional correctness, cross-domain interoperability, and practical utility of the developed architecture in supporting data-driven urban planning.

Chapter 5

Conclusion

This chapter presents the main contributions of this dissertation, highlighting the most relevant findings and work done, the identified limitations, and possible directions for future work.

5.1 Dissertation's Contributions

This dissertation proposes a platform capable of aggregating multiple urban services within the same environment, thereby reflecting more closely the concept of a **DT** for a smart city. In particular, this work addresses a limitation identified in the literature review, namely the fact that many Digital Twins focus on isolated domains rather than being designed to support scalability and the integration of multiple urban services within the same environment. To address this issue, a modular platform was developed based on interoperable communication through FIWARE, allowing different services to be integrated into the same system while remaining accessible through a common web interface.

To demonstrate this methodology, two different services were integrated into the platform: a Transit Module and an Electric Vehicles Module. The first supports the analysis of road-traffic disruptions, which can be observed either after the simulation is completed or during execution in live mode, while the second allows users to plan the placement of charging infrastructure across the city. The experimental results showed that both services operated as expected, providing useful insights to the user and enabling more informed decisions before such changes are applied in the real world.

With the final experiment, it was possible to demonstrate that the proposed platform is not only extensible to other service domains, but can also be adapted to different cities simply by replacing the simulation inputs, without requiring changes to the underlying code. This highlights both the flexibility of the architecture and its potential for reuse in different urban contexts.

In summary, this platform represents an alternative step towards the development of Digital Twins for smart cities, promoting service scalability, interoperability across different domains, the integration of multiple data sources within a single platform, and a more unified approach to the simulation and analysis of urban systems.

5.2 Limitations

This approach also presents some limitations. The platform depends heavily on FIWARE as the central context-management component, which introduces a potential single point of failure in the overall architecture. Also, the proposed architecture was validated only with two service domains, so its applicability to a broader range of urban services remains theoretical. The realism of the obtained results depends heavily on the quality of the inputs provided to the simulation engines. Since part of the data used in the experiments is synthetic, although designed to closely represent the city of Porto, it does not correspond exactly to real-world conditions, which may limit the representativeness of the results when compared with real-time or fully observed data.

In the Transit Module, road closures are modelled through severe capacity reductions rather than complete link removal. Although this approach allows only a very small share of the population to continue using the affected links, typically in situations where no feasible alternative exists, it does not fully reproduce the effect of a completely inaccessible road. A further limitation concerns public transport handling. In real-world operation, if a bus stop becomes unavailable due to a road closure, the service may be diverted and the stop temporarily relocated to a nearby street. In the current implementation, however, stops are not reassigned to alternative locations. If a bus route can be detoured while preserving its existing stops, the route is adjusted accordingly, however, when a stop is directly located on a blocked link, the corresponding route is removed from the scenario rather than having the stop reassigned to a nearby street. As a result, this representation does not fully reflect what happens in a real-world scenario.

In the EV Module, agents charge either until their battery reaches full capacity or until a duration of 20 minutes is reached. This may not fully represent real-world behaviour, since EV users do not always aim to fully charge their vehicles, or may charge for more time if they want to. Another limitation has to do with the insertion of charging activities into the daily plan of electric-vehicle users. In the current implementation, the charging stop is always inserted into the last car trip identified in the selected plan, rather than being determined dynamically according to the user's travel context during the day. As a result, the model may fail to capture situations in which an agent could decide to charge earlier in the day after passing near a charging station.

5.3 Future Work

As future work, additional urban services may be integrated into the platform in order to enrich its scope and further demonstrate its ability to support a wider range of services. Although the current services are integrated within the same platform, they still operate independently, meaning that the execution of one module does not directly influence the other. A possible direction for future work would be to allow the different services to interact with each other, for example by testing charging-infrastructure configurations under road-network disruption scenarios.

Currently, the developed modules rely on synthetic data, so an important direction for future work would be to integrate data from official sources, such as the municipality of Porto, and

convert it into formats compatible with MATSim. This would improve the realism of the simulated scenarios and increase the credibility of the obtained results, allowing the platform to better reflect real urban conditions and better inform urban planners.

Finally, the robustness of the platform may be improved by reducing its dependence on FI-WARE as a single central component, since a failure at this level may compromise the overall architecture. Fault tolerance and recovery mechanisms should be added to this system so the platform can still operate when the central component fails.

References

- [1] K. Alexandridis et al. “Designing and Implementing a Robust, Modular and Interoperable Digital Twin Smart City Framework for Critical Water Spatial Infrastructure”. In: *Proceedings of the Annual Hawaii International Conference on System Sciences*. Type: Conference paper. 2024, pp. 7479–7488. DOI: [10.1109/SMARTCOMP55677.2022.00064](https://doi.org/10.1109/SMARTCOMP55677.2022.00064).
- [2] J.V.S. do Amaral et al. “Energy Digital Twin applications: A review”. In: *Renewable and Sustainable Energy Reviews* 188 (2023), p. 113891. ISSN: 1364-0321. DOI: [10.1016/j.rser.2023.113891](https://doi.org/10.1016/j.rser.2023.113891).
- [3] Rolando Armas, Hernan Aguirre, and Daniel Orellana. “Evolutionary bi-objective optimization for the electric vehicle charging stand infrastructure problem”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO ’22. Boston, Massachusetts: Association for Computing Machinery, 2022, pp. 1139–1146. ISBN: 9781450392372. DOI: [10.1145/3512290.3528859](https://doi.org/10.1145/3512290.3528859).
- [4] Elena Ascari et al. “OUTFIT: Crowdsourced Data Feeding Noise Maps in Digital Twins”. In: *Proceedings of the 4th Workshop on Flexible Resource and Application Management on the Edge*. FRAME ’24. event-place: Pisa, Italy. New York, NY, USA: Association for Computing Machinery, 2024, pp. 39–46. ISBN: 979-8-4007-0641-7. DOI: [10.1145/3659994.3660312](https://doi.org/10.1145/3659994.3660312).
- [5] ASQ. *PDCA Cycle - What is the plan-do-check-act cycle?* 2026. URL: <https://asq.org/quality-resources/pdca-cycle> (visited on 06/30/2026).
- [6] Guillermo del Campo Jiménez et al. “Digital Twins for Street Lighting: Challenges for a Virtual Reality solution based on Internet-of-Things Devices and Photometry Rendering”. In: (2023). Type: Conference paper. DOI: [10.1109/LS1858153.2023.10170533](https://doi.org/10.1109/LS1858153.2023.10170533).
- [7] CIBSE. “Twin cities: harnessing gaming technology for immersive digital twins”. In: *CIBSE Journal* (Mar. 2024). URL: <https://www.cibsejournal.com/technical/twin-cities-harnessing-gaming-technology-for-immersive-digital-twins/>.
- [8] Thomas Clemen et al. “Multi-Agent Systems and Digital Twins for Smarter Cities”. In: *Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. SIGSIM-PADS ’21. event-place: Virtual Event, USA. New York, NY, USA: Association for Computing Machinery, 2021, pp. 45–55. ISBN: 978-1-4503-8296-0. DOI: [10.1145/3437959.3459254](https://doi.org/10.1145/3437959.3459254).
- [9] Andrea Roberta Costagliola, Marco Montanari, and Paolo Bellavista. “Interconnecting Urban Networks: A Novel Approach to Digital Twins Through GlassBox Adaptation”. In: *International Conference on Smart Cities and Green ICT Systems, SMARTGREENS - Proceedings*. Type: Conference paper. 2025, pp. 193–204. DOI: [10.5220/0013495100003953](https://doi.org/10.5220/0013495100003953).

- [10] Priscila Couto de Almeida, Larissa Paredes Muse, and Patrícia Araújo de Abreu. “Proposal of a Methodology for Urban Continual Improvement with Smart Cities Integrated Solutions & Technologies”. In: *2020 IEEE International Smart Cities Conference (ISC2)*. ISSN: 2687-8860. Sept. 2020, pp. 1–8. DOI: [10.1109/ISC251055.2020.9239094](https://doi.org/10.1109/ISC251055.2020.9239094).
- [11] Jaume Ferré-Bigorra, Miquel Casals, and Marta Gangolells. “The adoption of urban digital twins”. In: *Cities* 131 (2022), p. 103905. DOI: [10.1016/j.cities.2022.103905](https://doi.org/10.1016/j.cities.2022.103905).
- [12] FIWARE. *Stellio Context Broker*. 2026. URL: <https://stellio.readthedocs.io/en/latest/> (visited on 06/30/2026).
- [13] Jose Sanchez Gracias et al. “Smart Cities—A Structured Literature Review”. In: *Smart Cities* 6.4 (2023), pp. 1719–1743. ISSN: 2624-6511. DOI: [10.3390/smartcities6040080](https://doi.org/10.3390/smartcities6040080).
- [14] Manoj Herath et al. “An AI-Driven, Scalable, and Modular Digital Twin Framework for Traffic Management”. In: *2025 IEEE Wireless Communications and Networking Conference (WCNC)*. ISSN: 1558-2612. Mar. 2025, pp. 01–06. DOI: [10.1109/WCNC61545.2025.10978260](https://doi.org/10.1109/WCNC61545.2025.10978260).
- [15] Manoj Herath et al. “Smart City Digital Twins: A Modular and Adaptive Architecture for Real-Time Data-Driven Urban Management”. In: *2024 20th International Conference on Network and Service Management (CNSM)*. 2024, pp. 1–7. DOI: [10.23919/CNSM62983.2024.10814627](https://doi.org/10.23919/CNSM62983.2024.10814627).
- [16] Nandana Jayachandran et al. “An Architecture and Realisation of a Smart City Digital Twin With eHealth Case Studies”. In: *IET Smart Cities* 7.1 (2025). Type: Article. DOI: [10.1049/smc2.70006](https://doi.org/10.1049/smc2.70006).
- [17] Zdenko Kljaić et al. “Digital Twin Models with ESG Methodology as a Tool for the Transformation of Cities in the Area of Transport and Energy”. In: *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*. ISSN: 2623-8764. May 2024, pp. 1937–1942. DOI: [10.1109/MIPRO60963.2024.10569833](https://doi.org/10.1109/MIPRO60963.2024.10569833).
- [18] Hui Min Lee, Ruhollah Jamali, and Sanja Lazarova-Molnar. “A Conceptual Framework for Digital Twins of Multi-Agent Systems”. In: *Procedia Computer Science* 257 (2025), pp. 321–328. ISSN: 1877-0509. DOI: [10.1016/j.procs.2025.03.043](https://doi.org/10.1016/j.procs.2025.03.043).
- [19] matsim-org. *matsim-code-examples*. 2026. URL: <https://github.com/matsim-org/matsim-code-examples/tree/dev.x/src/main/java/org/matsim/codeexamples/withinday> (visited on 06/30/2026).
- [20] Observador. *Tráfego automóvel aumenta 5% e bate recorde*. 2025. URL: <https://observador.pt/2025/01/05/trafego-automovel-nas-autoestradas-aumenta-5-e-bate-recorde-ic19-em-lisboa-e-vci-no-porto-sao-as-mais-movimentadas/> (visited on 06/30/2026).
- [21] Erol Ozkan, Ibrahim Kok, and Suat Ozdemir. “System Development Life-Cycle Assisted Digital Twin Development Model for Smart Micro-grids”. In: *INTERNET OF THINGS* 31 (May 2025). Type: Article. ISSN: 2543-1536. DOI: [10.1016/j.iot.2025.101580](https://doi.org/10.1016/j.iot.2025.101580).
- [22] Xiyu Pan, Neda Mohammadi, and John E. Taylor. “Smart City Digital Twins for Public Safety: A Deep Learning and Simulation Based Method for Dynamic Sensing and Decision-Making”. In: *Proceedings of the Winter Simulation Conference. WSC ’22*. Singapore, Singapore: IEEE Press, 2023, pp. 808–818. DOI: [10.1109/WSC57314.2022.10015421](https://doi.org/10.1109/WSC57314.2022.10015421).
- [23] K. Papachristou et al. “Digital Twins’ Advancements and Applications in Healthcare, Towards Precision Medicine”. In: *Journal of Personalized Medicine* 14.11 (Nov. 2024), p. 1101. DOI: [10.3390/jpm14111101](https://doi.org/10.3390/jpm14111101).

- [24] P.Janaki Ramal and E. Anbalagan. “A Hybrid Optimization and Machine Learning Framework for Urban Traffic Management Using Cyber-Physical Digital Twin Architecture”. In: *2024 IEEE 11th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*. ISSN: 2687-7767. Nov. 2024, pp. 1–6. DOI: [10.1109/UPCON62832.2024.10983125](https://doi.org/10.1109/UPCON62832.2024.10983125).
- [25] Tiago Ribeiro et al. “Modelling with NGSI-LD: the VALLPASS project case study”. In: *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*. Vol. 2023-July. ISSN: 2378-363X. July 2023, pp. 1–8. DOI: [10.1109/INDIN51400.2023.10218110](https://doi.org/10.1109/INDIN51400.2023.10218110).
- [26] Soheil Sabri et al. “Designing a Spatially-explicit Urban Digital Twin Framework for Smart Water Infrastructure and Flood Management”. In: *2023 IEEE 3rd International Conference on Digital Twins and Parallel Intelligence (DTPI)*. Nov. 2023, pp. 1–9. DOI: [10.1109/DTPI59677.2023.10365478](https://doi.org/10.1109/DTPI59677.2023.10365478).
- [27] Iohan Sardinha et al. “Towards Dynamic Population Synthesis Through the Use of Machine Learning”. In: *Advances in Social Simulation*. Ed. by Marcin Czupryna, Bogumił Kamiński, and Harko Verhagen. Cham: Springer Nature Switzerland, 2025, pp. 439–449. ISBN: 978-3-031-91782-0. DOI: [10.1007/978-3-031-91782-0_32](https://doi.org/10.1007/978-3-031-91782-0_32).
- [28] Scrum.org. *What is Scrum?* Scrum.org. 2026. URL: <https://www.scrum.org/resources/what-scrum-module> (visited on 06/30/2026).
- [29] F. Tao and H. Zhang. “Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0”. In: *EPJ Special Topics* (2012). DOI: [10.1140/epjst/e2012-01703-3](https://doi.org/10.1140/epjst/e2012-01703-3).
- [30] Simon Thelen et al. “A Slim Digital Twin for a Smart City and Its Residents”. In: *Proceedings of the 12th International Symposium on Information and Communication Technology*. SOICT ’23. event-place: Ho Chi Minh, Vietnam. New York, NY, USA: Association for Computing Machinery, 2023, pp. 8–15. ISBN: 979-8-4007-0891-6. DOI: [10.1145/3628797.3628936](https://doi.org/10.1145/3628797.3628936).
- [31] J. Wang, C. Xu, and D. Zhang. “Digital Twin-Based Smart Production Management for Complex Manufacturing”. In: *Procedia CIRP* 83 (2019), pp. 163–168. DOI: [10.1016/j.procir.2019.04.033](https://doi.org/10.1016/j.procir.2019.04.033).
- [32] Guanyu Xiong, Haijiang Li, and Yan Gao. “Cross-domain comparative analysis of digital twins and universalised solutions”. In: *ADVANCED ENGINEERING INFORMATICS* 68.B (Nov. 2025). ISSN: 1474-0346. DOI: [10.1016/j.aei.2025.103726](https://doi.org/10.1016/j.aei.2025.103726).