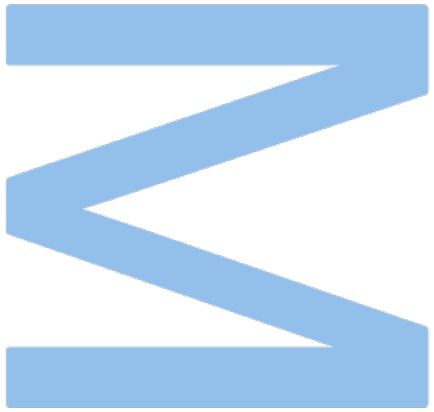
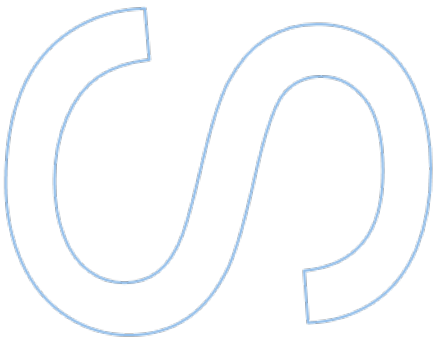
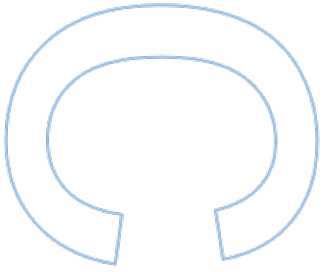
Design and Implementation of a Smart-City Digital Twin for Supporting Autonomous Vehicles







Francisco Rodrigues Gonçalves
Mestrado em Ciência de Computadores
Departamento de Ciência de Computadores
2023

Supervisor
João Paulo da Conceição Soares, Doutorado Equiparado a Investigador Principal, Faculdade de Ciências da Universidade do Porto

Co-supervisor
Rolando da Silva Martins, Professor Auxiliar, Faculdade de Ciências da Universidade do Porto


U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Sworn Statement

I, Francisco Rodrigues Gonçalves, enrolled in the Master Degree Ciência de Computadores at the Faculty of Sciences of the University of Porto hereby declare, in accordance with the provisions of paragraph a) of Article 14 of the Code of Ethical Conduct of the University of Porto, that the content of this dissertation reflects perspectives, research work and my own interpretations at the time of its submission.

By submitting this dissertation, I also declare that it contains the results of my own research work and contributions that have not been previously submitted to this or any other institution.

I further declare that all references to other authors fully comply with the rules of attribution and are referenced in the text by citation and identified in the bibliographic references section. This dissertation does not include any content whose reproduction is protected by copyright laws.

I am aware that the practice of plagiarism and self-plagiarism constitute a form of academic offense.

Francisco Rodrigues Gonçalves

30/06/2023

Abstract

This thesis project delves into the realms of Internet of Things (IoT), Smart-City Digital Twin (SCDT), Autonomous Driving (AD) and the FIWARE framework. It encompasses a comprehensive study of these domains, exploring their interconnections and potential for transforming urban environments, specially over the context of deploying autonomous vehicles in real-world scenarios without physical test and experimentation. This whole research encompasses an analysis of the existing literature about SCDT projects, both those leveraging FIWARE components in their structure and those using alternative approaches; to be able to understand their strengths and weaknesses, and enlighten the technical vision with the best and worst practices in the development of a robust and flexible solution capable of meeting the required goals.

Drawing upon this knowledge, this dissertation presents a meticulously designed system architecture that fulfills the project's objectives in context management, context simulation and context analysis and visualization. There is a focus on leveraging the FIWARE ecosystem, including the developed abstraction layer service and the context management web interface, that facilitates an efficient configuration and deployment of the solution. The system is containerized to ensure scalability and portability, which contributes to its deployment and configuration process.

The validation of the proposed solution is conducted to ensure its correctness and effectiveness. This process involves rigorous testing and verification of the system's functionalities, with a focus on context management, simulation and analysis. The validation results confirm the solution capability in meeting the expected standards and requirements.

Concluding the research, this thesis offers insights and reflections on the findings and outcomes of the project and also highlights the potential for future optimization and evolution, outlining areas for further improvement and development. The study conclusions contribute to the advancement of SCDT solutions and emerging AD technologies, fostering sustainable and resilient transportation systems capable of addressing both existing and upcoming challenges.

Resumo

Este projeto de tese aborda os domínios de Internet of Things (IoT), Smart-City Digital Twin (SCDT), Autonomous Driving (AD) e o ecossistema FIWARE. É feito um estudo abrangente nestes domínios, tendo sido focadas as suas interligações, dependências e potencial para transformar ambientes urbanos, especialmente no contexto da implementação de veículos autónomos em cenários do mundo real, sem recorrer a testes ou a modelos de experimentação física. Toda esta pesquisa engloba uma análise da literatura existente sobre projetos relacionados com o conceito de SCDT, havendo uma subdivisão entre aqueles que recorrem a componentes de FIWARE na sua estrutura de base, e aqueles que adotam abordagens alternativas; com a finalidade de compreender os pontos fortes e fracos de cada um e aprimorar a visão técnica sobre as melhores e piores práticas no desenvolvimento de uma solução robusta e flexível capaz de atender aos objetivos requeridos da solução idealmente proposta.

Com base neste conhecimento, esta dissertação propõe um modelo de arquitetura cuidadosamente projetado para atender aos objetivos propostos, em termos de gestão de contexto de entidades, simulação contextual, como também em termos da análise e visualização dos dados. É dado foco à utilização dos *generic enablers* do FIWARE como camada de suporte, incluindo ainda o serviço da camada de abstração e a interface web de gestão de contexto desenvolvidos de raiz, que facilitam a configuração e a implementação eficiente da solução. O sistema é ainda containerizado, o que permite garantir uma maior escalabilidade e portabilidade da solução, facilitando o processo de instalação e devida utilização.

A validação da solução proposta é ainda conduzida numa fase final do documento para garantir que esta é correta e eficaz, e atende aos requisitos propostos. Este processo envolve testes de integração e validação rigorosos das várias funcionalidades do sistema. Os resultados da validação confirmam as capacidades da solução em corresponder aos padrões e requisitos esperados.

No final deste documento, são apresentadas reflexões sobre as aprendizagens e resultados do desenvolvimento. É também ainda destacado o potencial de otimização e evolução futura da solução, onde são delineadas áreas de possível melhoria e adaptação ao próprio desenvolvimento. As conclusões deste estudo certamente contribuem para o avanço dos desenvolvimentos em SCDT e das tecnologias emergentes de AD, o que promove sistemas de transporte sustentáveis e resilientes capazes de enfrentar desafios não só existentes, como também possibilidades futuras.

Acknowledgments

This is where I feel the urge to thank my life pillars, without whom I wouldn't be able to reach this day. To you mother, father and sister, I hope that you will come to know the significance of this thesis and recognize that its accomplishments are also a testament to your own presence, for the care and support you always put in me through all these academic years. To my closest friends, who have stood by my side through thick and thin, you know who you are. I want you to understand the profound impact you have on me and on my motivation to be better. To my grandmother and godfather who have watched over me in the same way as a second set of parents since I was little. I hope you both are now proud, because as you always encouraged me to, we made it. To you cousin, for growing with me and letting me grow with you. It all started with us lying side by side on the bed at our grandmother's house in that `GameMaker` tutorial of yours, I will never forget that and I am sure you won't either. To you, uncle, for being my social and professional role model when I was younger, whose steps I followed without ever doubting that it would be the right path for me. You are still and will always be a reference for me.

This achievement is also very yours. We made it.

Contents

List of Tables

List of Figures

Listings

Acronyms

AD Autonomous Driving

AI Artificial Intelligence

AMQP Advanced Message Queuing Protocol

CB Context Broker

CORS Cross-Origin Resource Sharing

CQL Context Query Language

CV Computational Vision

DT Digital Twin

EC European Commission

EU European Union

GDPR General Data Protection Regulation

GE Generic Enabler

GUI Graphical User Interface

ICT Information and Communication Technology

IoT Internet of Things

JS JavaScript

KPI Key Performance Indicator

LoRaWAN Long Range Wide Area Network

M2M Machine-to-Machine

MaaS Mobility-as-a-Service

ML Machine Learning

MQTT Message Queuing Telemetry Transport

NGSI Next Generation Service Interface

OS Operating System

SCDT Smart-City Digital Twin

URI Uniform Resource Identifier

VM Virtual Machine

VR Virtual Reality

UN United Nations

Chapter 1

Introduction

The convergence of smart-cities and autonomous vehicles harbors boundless potential for elevating urban mobility and the overall quality of life. However, their integration faces many challenges. This research project aims to address these by designing and implementing a Smart-City Digital Twin (SCDT) platform based on the FIWARE standards. This system enables the comprehensive testing of Autonomous Driving (AD) models, facilitates system integration within the smart-city ecosystem, is capable of handling large volumes of data, provides real-time insights and offers an intuitive interface for visualizing and interacting with the context state. This research outcomes will certainly contribute to the advancement of the current autonomous vehicle technology and smart-city development, fostering sustainable and resilient transportation systems.

1.1 Motivation

Driven by rapid technological advancements and the growing emphasis on sustainable urban development, the concept of smart-cities has emerged as a promising solution. Smart-cities leverage information and communication technologies to enhance the general quality of life, optimize resource utilization and improve infrastructure efficiency. In this progressive journey, autonomous vehicles have shown to be a key enabler to the general optimization of urban mobility which directly translates to enhancements in a city proper function and citizens well-being, which revolutionizes not only the landscape of human transportation but also of city planning and management.

Alongside progress, new risks have surfaced, with the COVID-19 pandemic being a recent prominent example, along with more traditional risks like flooding, air pollution, natural disasters or even the daily traffic congestion. The integration of technology enables cities to monitor real-time situations, offering innovative services and experiences to citizens and businesses, making more informed decisions [35].

This raises the question of how digital technologies, such as the Digital Twin (DT) concept, can contribute to predicting, preparing and preventing future risks. The design and implementation

of DT in an urban context, allow cities to create comprehensive and realistic representations of their environments, which can aid in assessing and mitigating potential hazards.

The concept of the SCDT holds a significant importance and value across multiple aspects of any urban environment. It offers a transformative solution that can enhance the efficiency, sustainability and resilience of cities. By integrating data from various sources and creating a virtual replica of the physical world, it provides a comprehensive and real-time representation of the metropolitan context, including:

- **Urban planning and design**, as it enables urban planners and designers to simulate and visualize different scenarios and configurations, from which they can assess the impact of proposed changes;
- **Infrastructure management**, facilitates the monitoring, maintenance and management of critical infrastructure systems within a city;
- **Energy efficiency and sustainability**, playing a crucial role in optimizing energy consumption and promoting sustainability in cities by simulating energy flows, analyzing patterns and identifying inefficiencies, a SCDT can help optimize energy distribution, grid management and resource allocation;
- **Citizen engagement and participation**, it can foster citizen engagement and participation in urban decision-making processes by providing accessible and interactive platforms, as it enables citizens to visualize proposed projects, share feedback and actively contribute to the metropolitan planning.

One crucial aspect of integrating autonomous vehicles into digital twins is the ability to accurately simulate and test these vehicles in real-world urban environments without actually placing them in the field. This is a scenario where the concept of the SCDT becomes invaluable as it allows for the comprehensive testing of various AD models under different configurations and scenarios, providing a dynamic platform to evaluate their performance in a virtual representation. By simulating and testing these in diverse and realistic conditions, the SCDT facilitates the development and refinement of these emerging AD technologies, ensuring their effectiveness and adaptability to the complex challenges of variable urban settings.

The motivation behind the research in this area lies in recognizing the tremendous potential of AD to transform urban mobility. This new driving paradigm has the ability to revolutionize transportation efficiency by reducing traffic congestion, minimizing accidents and decreasing carbon emissions [43]. However, their safe and efficient operation depends on real-time and precise information about their surroundings, including road conditions, traffic patterns, infrastructure and the behavior of other road entities [45].

The DT, a virtual replica of the a physical entity, which in the context of smart-cities is a representation of the real world, offers a transformative solution to overcome these challenges.

By integrating data from diverse sources such as sensors, Internet of Things (IoT) devices and urban infrastructure, the digital instance creates a comprehensive and real-time representation of the contextual environment. This powerful tool enables the testing and virtual deployment of multiple AD models without the need for extensive real-world testing. Through it, various scenarios and configurations can be simulated, allowing for rigorous evaluation and optimization of autonomous systems. This not only saves time and resources but also enhances safety by identifying potential issues and fine-tuning the performance of these vehicles before their physical implementation. By leveraging the technology, municipalities can accelerate the development and deployment of AD technologies, leading to a lot more efficient and reliable transportation systems.

By exploring the potential of a SCDT in the context autonomous vehicles, this research aims to contribute to the advancement of urban mobility, fostering sustainable and resilient transportation systems that can effectively address both existing and emerging risks.

1.2 Main Goals

The main goals lying behind this thesis project are the design and implementation of a SCDT specifically tailored and based in FIWARE, capable of supporting autonomous vehicles and real-time simulated environments that can help cities better understanding and better studying their urban transportation fluxes. All the research here involved aims to create a robust but also scalable ecosystem that effectively reads, processes and visualizes timely data from diverse input sources within the urban environment. By achieving these proposed goals, this research strives to make significant contributions to the study field of AD technology and to the smart-city development world. Each one of the outcomes of this extensive study will not only enhance the understanding of DT solutions but also provide valuable insights into the practical implementation of autonomous vehicles in real-world urban environments and real-time simulated environments.

Through the design and implementation of a SCDT, this project seeks to:

1. Address the current challenges associated with autonomous vehicle deployment;
2. Provide seamless integration and interoperability among diverse systems, devices and platforms within the smart-city ecosystem;
3. Build a robust and scalable architecture that can handle large volumes of generated context data within the smart-city;
4. Enable real-time data ingestion processing to provide up-to-date and easily accessible insights;
5. Systematize an efficient process for persistent context update storage which enables historical and conditional context analytics;

6. Design a repetition system that permits the rerun of simulations based on time-specific context data snapshots;

7. Develop an intuitive and user-friendly web interface for visualizing and interacting with the contextual data.

Addressing these core objectives, the developed solution can provide a robust and flexible foundation for the implementation of a SCDT that will effectively support not only the development of innovative smart-city applications and services but also successful AD scenarios.

1.3 Dissertation Layout

This dissertation is divided into seven main parts.

This **Introduction (1)** chapter sets the stage for the thesis project by providing a comprehensive overview of the whole research topic. It presents the problem set to be addressed, outlines the main motivations while giving a clear roadmap of what to expect before reading.

In the **Background (2)**, the fundamental concepts and relevant frameworks are explored in detail. This section establishes a solid foundation for the proposed architecture by examining key concepts from the multiple knowledge pillars composing this study context. The main goal is to provide a comprehensive understanding of the theoretical underpinnings that inform the research.

Following, the **State of the Art (3)** demonstrates a deep understanding of the existing work done in the area related to the research topic. It serves as a comprehensive review of the current knowledge scenario, emphasizing the gaps this project aims to fill. Previous studies, methodologies that have been employed to address similar problems are critically analyzed, giving special attention to their strengths and limitations when designing and developing a SCDT for supporting AD.

The **Proposed Architecture (4)** forms the crux of the dissertation, where the core architecture designed to address the research problem is introduced for the first time. This section provides a detailed description of the components, modules and interactions within the system skeleton. The choices made during the design process are justified, highlighting how they may contribute to achieving the previously set objectives.

As proposed and described in the previous chapter, the **Configuration and Deployment (5)** outlines the steps taken to implement the proposed architecture in the best technical way possible. It discusses the methodologies, tools and techniques employed during the development phase, as well as the principal challenges faced and how they got beaten.

In order to present the testing process and properly cover all the components correctness, the **Validation (6)** chapter focuses on validating the final solution going through all the possible

use case scenarios by simulating them against the system.

Finally, conducting this research outcomes, in the **Conclusions (7)** chapter, the main findings of the research are summarized and their implications are discussed. The motivation goals are restated, emphasizing how the proposed architecture contributes to addressing the starting problem and additionally, it explores potential directions for further development and improvement of the solution, suggesting areas of focus for future research and highlighting opportunities to enhance the already reached capabilities.

Chapter 2

Background

This conceptual review provides an in-depth exploration of some key concepts and theoretical frameworks that form the foundation for understanding and investigating the research topic of the Smart-City Digital Twin (SCDT) and its relationship with Autonomous Driving (AD). It aims to present a comprehensive overview over the surrounding theoretical landscape, laying the groundwork for the subsequent chapters and contributing to a better understanding of the study field.

2.1 IoT, the Internet of Things

The Internet of Things (IoT) refers to a network of interconnected physical objects, devices and systems that are embedded with sensors, software and network connectivity, allowing them to collect and exchange data. IoT enables these objects to interact with each other and with the digital world, creating a seamless and pervasive network that enhances communication, automation and data-driven decision-making [38].

The foundation of IoT lies in the integration of physical objects with digital technologies. Everyday objects, ranging from household appliances and vehicles to industrial machinery and infrastructure [2], are equipped with sensors and actuators that enable them to monitor and respond to their environmental context. These sensors gather data about the object's state, behavior and performance, while actuators enable control and manipulation of the object based on the received and processed information.

The collected data from IoT devices is transmitted over networks, such as the internet and it is processed and analyzed in real-time or near real-time. This data analysis often involves complex algorithms, Machine Learning (ML) and Artificial Intelligence (AI) techniques to derive valuable insights and enable intelligent actions [38]. These insights can be used to optimize processes, improve efficiency and enable new services and applications across various domains, such as healthcare, agriculture, transportation, manufacturing, all present in the smart-city world.

IoT networks and platforms provide the infrastructure and necessary tools for connecting and managing a large number of devices and data streams. These networks can be wireless like Wi-Fi or Bluetooth or wired such as Ethernet [38], and they enable seamless communication and data exchange between systems. Platforms in this area offer capabilities for device management, data ingestion, storage, analysis and even visualization, providing a unified framework for developing and deploying IoT applications.

The impact of this extends beyond individual devices and systems. The ability of IoT to connect and integrate various objects and systems results in a network effect, where the collective intelligence and capabilities of interconnected devices surpass the capabilities of individual devices. This makes the development of more sophisticated and interconnected applications possible, such as in smart homes, smart grids, intelligent transportation systems or even precision agriculture, and crucially helps to achieve higher levels of efficiency, automation and responsiveness in these fields [2].

However, as IoT expands, there are challenges to address, such as security, interoperability and scalability. Protecting data privacy and ensuring the security of devices and networks is crucial, as these interconnected systems can handle sensitive information and can be potential targets for cyber-attacks. Interoperability standards and protocols are essential for a seamless integration and communication among different IoT devices and platforms. Additionally, a scalable infrastructure is needed to handle the vast amount of data generated and to support the continuously growing ecosystem.

As IoT continues to evolve, it is growing the potential to transform industries, improve quality of life and drive the advancement of a more connected and intelligent world.

2.2 Smart-City Digital Twin

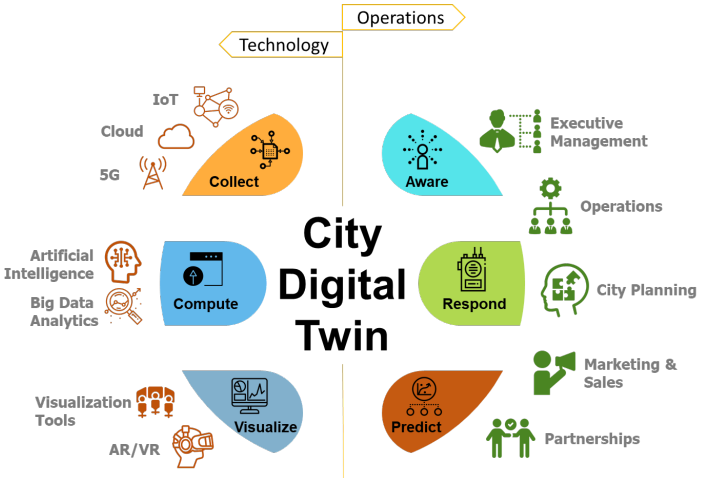


Figure 2.1: The concept of a smart-city digital twin

2.2.1 Smart-City

A smart-city is a concept that encompasses the integration of advanced information and communication technologies with urban infrastructure and services to enhance the quality of life, sustainability and overall efficiency of a metropolis as represented in the above figure 2.1. It represents a holistic approach to the urban development by leveraging data-driven insights, automation and connectivity in order to improve various aspects of urban living, including transportation, which is this the greater focus in this research, but also energy, governance, public safety, healthcare and environmental sustainability.

At its core, a smart-city is built upon the foundation of a robust Information and Communication Technology (ICT) infrastructure that serves as the nervous system, enabling the collection, analysis and exchange of data from various input sources [26]. These sources may include sensors, IoT devices, social media, mobile applications and other digital platforms. The collected data is then processed and analyzed in real-time, providing valuable insights to decision-makers, city planners and general citizens.

A key characteristic of a smart-city is its ability to use the generated insights to optimize resource allocation and improve the delivery of services [40]. For instance, in the transportation domain, real-time data on traffic patterns, road conditions or parking availability can be taken into consideration to optimize traffic flow, reducing overall congestion while enhancing public transportation systems. Similarly, in the energy consumption sector, smart grids and energy management systems can smartly monitor and adjust electrical powered vehicles consumption based on the demand patterns which may lead to a more efficient use of resources while also reducing their environmental impact.

Furthermore, a smart-city emphasizes sustainability and environmental stewardship. By leveraging ICT, cities are capable of monitor and reduce energy consumption, optimize waste management, promote green transportation options and mitigate the environmental risks associated with nowadays urban mobility. This focus on sustainability not only helps to reduce the ecological footprint of the city but also enhances the quality of life for its residents.

2.2.2 Digital Twin

A Digital Twin (DT) is a virtual replica or representation of a physical object, system, or environment that is created and maintained in real-time [22]. It is a digital counterpart that mimics the characteristics, behaviors, and functionalities of its physical counterpart, enabling a deeper understanding and analysis of the physical entity throughout its lifecycle.

A DT is constructed through the integration of various data sources, including sensors, IoT devices, simulations and historical context data. These sources capture and continuously update information about the physical entity, such as its geometric structure, operational parameters, performance metrics and environmental conditions. This real-time data is then properly processed,

analyzed and visualized, providing an accurate and dynamic representation of the physical entity and its interactions with the surrounding environment.

The primary purpose of a DT is to enable enhanced decision-making, optimization and predictive capabilities. By simulating and analyzing different scenarios within the DT, stakeholders can gain insights into the performance, maintenance requirements and potential improvements of the physical entity. For instance, in manufacturing, a production line DT can be used to identify bottlenecks, optimize processes and even predict maintenance needs, which naturally leads to an increased efficiency and reduced downtime.

Moreover, a DT allows for the monitoring and control of physical entities remotely. By integrating this concept with control systems and analytics platforms, operators can monitor real-time data, detect anomalies and remotely adjust operational parameters [21]. This capability is particularly beneficial in complex and hazardous situations where direct human intervention may be limited or risky.

The advantages extend beyond the operational phase. It can also be used during the design and development stages to simulate and test different configurations, evaluate performance and optimize designs before the real physical implementation. Furthermore, a DT can support predictive maintenance by analyzing real-time data and generating alerts or recommendations for maintenance activities, thereby reducing also costs and extending the lifespan of base assets.

By bridging the gap between the physical and digital realms [21], this technological concept enables enhanced decision-making, improves operational efficiency, reduces risks and facilitates innovation across a variety of domains, including manufacturing, infrastructure, energy, healthcare and human transportation.

2.3 Autonomous Driving

Also referred to as self-driving or driverless technology, is the capability of a vehicle to operate and navigate without any human intervention. This represents a significant advancement in transportation, harnessing cutting-edge technologies to enable vehicles to perceive their surroundings, make safe and efficient decisions while executing maneuvers autonomously.

At the core of AD lies a complex system of sensors, cameras and advanced algorithms which work in harmony to interpret the environment around [23]. These sensors, including *LiDAR*[1] radars and cameras, provide a continuous stream of data, capturing information about the vehicle's surroundings [5] such as the position of other vehicles, pedestrians, traffic signals and road conditions which represent a great source of context data in a DT environment, as it is possible to visualize below in figure 2.2.

[1]Stands for Light Detection and Ranging and it is a remote sensing technology that uses laser light to measure distances and create detailed three-dimensional maps of the physical surroundings.

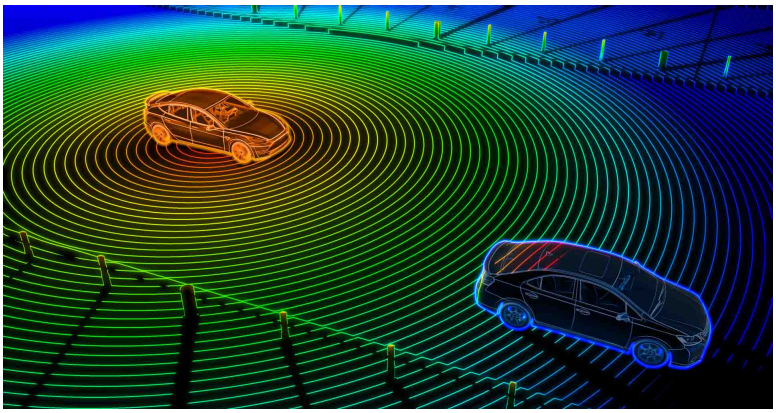


Figure 2.2: LiDAR surroundings detection process representation

Utilizing this wealth of data, sophisticated software algorithms process and analyze the incoming information in real-time. By applying AI, ML and Computational Vision (CV) techniques, the vehicle can understand and interpret its surroundings, identifying obstacles, recognizing road signs and predicting the behavior of other road users [5].

Based on this comprehensive understanding of the surroundings, the autonomous vehicle's decision-making algorithms come into play. These algorithms evaluate multiple factors, including traffic conditions, navigation routes and safety considerations, to make informed decisions about acceleration, braking, lane changes and other driving base maneuvers [5]. The main goal is to ensure a safe and efficient transportation while adhering to traffic regulations. To translate these decisions into action, autonomous vehicles are equipped with actuators and control systems that physically control the mechanical movements. These systems precisely execute the determined actions with a high level of precision and responsiveness. This technology holds the promise of revolutionizing transportation in numerous ways. It has the potential to enhance road safety by significantly reducing human error, which is responsible for the majority of traffic accidents [28]. Autonomous vehicles can also improve traffic flow, optimize fuel efficiency and also reduce general congestion through better coordination and communication between vehicles. This radically transforms the concept of commuting, allowing occupants to use their travel time more productively or enjoy a more relaxing and enjoyable experience. Furthermore, AD has the potential to enhance mobility for individuals who are unable to drive, such as the elderly or people with disabilities, offering them newfound independence and access to transportation services.

While it has made significant strides, there are still many challenges to overcome before widespread adoption becomes a common reality. Issues related to legal and regulatory frameworks, public acceptance, cybersecurity and the integration of autonomous road entities with existing transportation infrastructure are areas that require further attention and development in the next few years. Nonetheless, AD represents a huge transformative shift in the way we perceive and interact with vehicles as it holds the potential to reshape our cities making them smarter, improve safety and revolutionize urban transportation as we know it till these days, paving the

way for a future where vehicles can navigate the roads independently, efficiently and with the most reduced environmental impact possible.

2.4 FIWARE

FIWARE is an open-source platform that enables the development of smart solutions for various domains such as smart-cities, smart-agriculture, smart-manufacturing, and much more into the smarter-world solutions. It provides a set of reusable components and tools that simplify the development of services by abstracting complexity and promoting interoperability.

One of the main advantages of FIWARE is its modularity. It consists of several building blocks, referred to as Generic Enabler (GE) that can be assembled and combined to create a customized modular solution. These building blocks include components for data management, context awareness, access control and visualization, among others. This is key when making scalable solutions able to adapt to different scenarios and use cases. It also promotes the use of open standards and APIs, which makes it easier to integrate the ecosystem with other systems and services. This enables developers to leverage existing resources and data sources and build upon them, rather than having to start from scratch [7]. Another key feature of FIWARE is its focus on data and context management. The platform provides a standard data model and a context management system that allows applications to retrieve and process data in a timely way. This is very useful for creating intelligent solutions able to make decisions based on real-time information such as the ones in AD scenarios.

Overall, this is a powerful toolkit that empowers developers to create innovative solutions for a variety of domains. Its modularity, open standards and focus on data and context management make it an ideal platform for building solutions that can make a positive impact on society.

2.4.1 Architecture

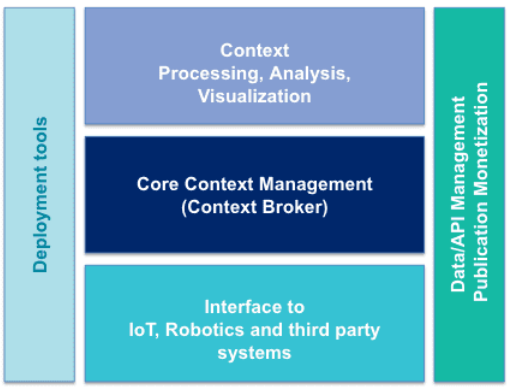


Figure 2.3: The architecture of the FIWARE ecosystem

The functional architecture of FIWARE is designed to facilitate the integration and management of multiple heterogeneous data sources, devices and services in the context of smart-city applications.

As the figure 2.3 suggests, the Context Broker (CB) is the centerpiece of the FIWARE architecture. Responsible for managing and providing access to real-time context data, it acts as the central repository for context, enabling the efficient storage and retrieval of data associated with the stored entities. The CB employs the Next Generation Service Interface (NGSI) which will be described in more detail in the following sections.

IoT Agents facilitate the integration of IoT devices and systems into the FIWARE ecosystem [7]. They provide a standardized interface for communication with a large diversity of these devices, abstracting the underlying protocols and possible data formats.

FIWARE also offers a range of data visualization and user interface tools to enable developers and end-users to interact with the contextual data. These tools provide intuitive interfaces to visualize and analyze real-time and historical data, which allow users to monitor and manage the smart-city systems effectively. These tools enhance decision-making capabilities by presenting context data in a meaningful and actionable manner.

2.4.2 Context Management

In a FIWARE-based system, context refers to any information that characterizes the state, conditions or environment of a specific entity or set of entities. It encompasses various data types, such as sensor readings, location information, weather conditions or traffic flows updates. Context data provides valuable insights into the current and historical state of the smart-city environment, enabling applications to make informed decisions and adapt to changing conditions.

This way, context management is a fundamental concept not only in the FIWARE framework but also within a SCDT, serving as a crucial component for capturing, storing and also providing access to real-time and historical context data.

- **Real-time context data** captures the current state of entities and systems, allowing applications to react promptly to changing conditions and events;
- On the other hand, **historical context data** provides a valuable resource for analysis, trend identification and long-term planning.

By managing both types, FIWARE enables a holistic understanding of the smart-city ecosystem and empowers applications with comprehensive insights over its entities.

As stated above and in the following figure 2.4, the CB is the brain central component responsible for all the context management within FIWARE. It provides a standardized interface, based on the NGSI standard, to request specific context information and receive updates through publish/subscribe mechanisms.

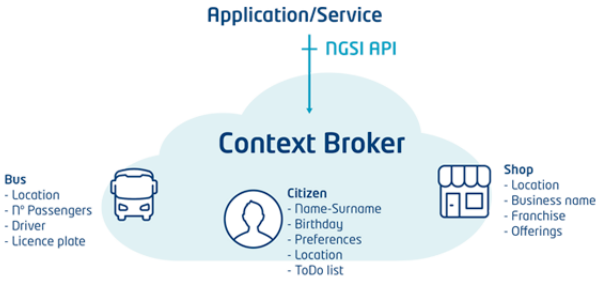


Figure 2.4: Context Broker framework placement

However, the CB itself does not provide built-in historical data storage capabilities. This is where FIWARE Cygnus comes into play. It acts as a connector between the CB and external databases or data storage systems, enabling the long-term persistence of entity data. Cygnus is responsible for retrieving updates on the context data and storing them in the chosen data storage solution, which could be either a relational database, a NoSQL database or a distributed file system. Cygnus provides various configuration options and adaptors to support different data storage solutions, such as:

- MySQL;
- PostgreSQL;
- MongoDB;
- HDFS;
- CKAN.

This ensures that historical data is captured, organized and stored in a way that allows for efficient querying and analysis, while also being flexible to different development necessities or requirements. It is relevant to note that FIWARE Cygnus is just one of the components within the ecosystem that facilitates historical data persistence. Depending on the specific requirements of a smart-city project, other FIWARE components such as QuantumLeap or Draco may also be used to address historical data storage and analysis needs.

2.4.3 Data Integration and Access

These are essential aspects of the FIWARE framework, enabling the seamless integration of diverse data sources and providing efficient mechanisms for accessing and querying context information. It facilitates the integration of heterogeneous data origins that may differ in terms of formats, protocols and semantics. By providing a flexible and scalable architecture, FIWARE can

accommodate a wide range of information providers, enabling the integration and harmonization of data from diverse places.

A Context Query Language (CQL) is provided to enable developers and applications to query and retrieve context information from the platform. CQL offers a powerful and expressive syntax for specifying filters, conditions and projections which refines the data retrieval process [1]. Also, as seen, FIWARE offers a set of standardized APIs that enable applications to access and interact with context information and services. These APIs provide a uniform and consistent interface for retrieving and manipulating context data, enabling developers to build applications that seamlessly integrate with the FIWARE platform. These APIs support various operations ranged from data retrieval, to update and even to real-time context update subscriptions.

NGSI is a standardized specification and set of APIs that facilitate the development and interoperability of smart-city solutions. It designed to enable seamless communication, data exchange and integration among the different components within a SCDT. At its core, NGSI defines a data model and a set of operations for managing and manipulating context information. The standard data representation format used by FIWARE provides a structured model for context information, defining standardized attributes and relationships that describe the properties, location, behavior and other relevant aspects of entities. This common data model once more ensures consistency and interoperability among different systems and applications that exchange and process context information. Its APIs can provide a standardized way to access, update and subscribe to context since they define a set of operations, such as querying entities based on specific criteria, updating entity attributes and subscribing to real-time notifications when changes occur. They use widely adopted protocols, such as *HTTP*[2] and *MQTT*[3], for communication, making it easier to integrate and interact with various systems and platforms.

2.4.4 Privacy and Security

Designing smart-city systems that are private and secure is a critical consideration when working on this field. In the context of FIWARE, ensuring the privacy of personal data and maintaining a high level of security are fundamental principles for safeguarding the integrity, confidentiality and availability of sensitive information.

Regarding personal data protection and compliance, the framework emphasizes this with relevant privacy regulations, such as the General Data Protection Regulation (GDPR). This is a comprehensive data protection law implemented by the European Union (EU) to protect the personal data and privacy of individuals within the EU. These regulations came into effect on the 25th of May, 2018 and apply to organizations that process personal data of EU residents, regardless of where the organization is located [42]. The main objectives of the GDPR are to give

[2]Stands for Hypertext Transfer Protocol and it is a protocol that governs communication between web clients and web servers.

[3]Stands for Message Queuing Telemetry Transport and it is a lightweight messaging protocol designed for efficient communication between devices in IoT and Machine-to-Machine (M2M) applications.

individuals greater control over their personal data as it introduces several rights and principles that organizations must comply with when processing personal data, including:

- **Consent**, so organizations must obtain clear and explicit consent from individuals before collecting and processing their personal data;
- **Transparency**, as organizations must provide individuals with clear and understandable information about how their personal data will be processed;
- **Data breach notification**, since they must notify the relevant supervisory authority and affected individuals in the event of a data breach that poses a risk to individuals' rights and freedoms;
- **Right to access**, so individuals have the right to access their personal data held by organizations and request information about how it is being processed;
- **Right to erasure**, as individuals have the right to request the deletion of their personal data under certain circumstances;
- **Data protection impact assessment**, which forces organizations to assess and mitigate the hazards associated with processing personal data, especially when using new technologies or engaging in high-risk activities.

This provides mechanisms to handle personal data in a privacy-preserving manner, ensuring that sensitive information is collected, processed and stored in accordance with all the legal defined requirements [42].

FIWARE encourages the principles of data minimization and purpose limitation to protect privacy. Data minimization involves collecting and retaining only the necessary data required for specific purposes, while purpose limitation ensures that collected data is used solely for the stated purposes and not for other unrelated activities [34]. By adhering to these principles, it is possible to promote privacy by minimizing the collection and usage of personal data to the extent necessary for smart-city applications.

2.4.5 Use Cases

The FIWARE framework has a lot of use case scenarios within the context of smart-cities, including:

1. **Smart energy management**: the FIWARE ecosystem can be used to develop solutions for optimizing energy consumption in a city. It can integrate data from smart-meters, weather forecasts and building automation systems to enable real-time monitoring and control of energy usage. This allows for efficient management of the energy resources, reduction of waste and the implementation of demand-response strategies.

2. **Intelligent transportation systems**: it is possible to apply in order to create intelligent transportation systems in smart-cities. By integrating data from traffic sensors, public transportation networks and weather conditions, it enables the development of applications that provide real-time traffic information, optimize traffic signal timings, facilitate smart parking management and also enable efficient routing and navigation.

3. **Waste management and recycling**: able to support smart-waste-management solutions as it can integrate data from smart-bins equipped with fill-level sensors, location tracking and route optimization algorithms. This enables efficient waste collection, reducing unnecessary trips and optimizing collection schedules. Additionally, FIWARE can facilitate recycling initiatives by providing information on recycling points and incentives for citizens.

4. **Urban governance and citizen engagement**: enhance citizen engagement and participation in urban governance by providing open data and APIs, it enables the development of applications that deliver information on public services, events and facilities. Citizens can provide feedback, report issues and interact with local authorities, fostering a collaborative and transparent relationship between the city administration and its residents.

5. **Public safety and security**: it can be utilized to build smart, safety and security solutions. It enables the integration of data from surveillance cameras, sensors and emergency response systems which allows for the timely monitoring of public spaces, early detection of security threats and efficient emergency response coordination.

2.5 Summary

This conceptual review introduces key concepts and theoretical frameworks related to the research topic of SCDT, how it relates to AD.

Internet of Things

- Refers to a network of interconnected physical objects, devices and systems embedded with sensors, software and network connectivity;
- Enables the collection and exchange of data among objects and the digital world, facilitating communication, automation and data-driven decision-making;
- Integrates physical objects with digital technologies through sensors and actuators, allowing monitoring, control and data analysis.

Smart-City Digital Twin

- Integrates advanced information and communication technologies with urban infrastructure and services to enhance the quality of life, sustainability and efficiency;

- Is a virtual replica or representation of a physical city, created and maintained in real-time;
- Makes use of IoT to leverage data-driven insights, automation and connectivity to improve multiple aspects of urban living, such as transportation, energy, governance, public safety, healthcare and environmental sustainability.

Autonomous Driving

- Refers to the capability of a vehicle to operate and navigate without human intervention;
- Relies on sensors, cameras and advanced algorithms to perceive the vehicle's surroundings and make safe and efficient decisions;
- Has the potential to revolutionize transportation by improving road safety, traffic flow, fuel efficiency and accessibility for individuals who are not capable to drive.

FIWARE

- Open-source platform that enables the development of smart solutions for various domains, which includes the smart-city one;
- Provides a modular and interoperable framework with reusable components and tools;
- Promotes open standards for context management and integration with other systems and services;
- The architecture of FIWARE includes components for context management such as the CB, IoT agents and data visualization tools;
- Context management is a fundamental concept, involving the collection, storage and retrieval of real-time and historical context data, enabling applications to make informed decisions and adapt to changing context conditions.

Chapter 3 presents a comprehensive review of the state of the art initiatives in designing and implementing a SCDT. To facilitate a systematic analysis and provide a clearer understanding over the topic, this review categorizes the projects based on their utilization of the FIWARE ecosystem. This categorization allows to explore and relate the diverse approaches and solutions employed in this study field.

The first category focuses on projects that adopt alternative approaches and technologies instead of relying directly on the open-source components. These initiatives tackle the challenges of designing and implementing an SCDT to address a variety of urban challenges. By analyzing the architectures, solutions and added values of these projects, valuable insights can be gained into the different approaches employed in the absence of FIWARE.

In contrast, the second category examines projects that have embraced it as their foundational framework and leverage the benefits of integrating with FIWARE, utilizing its robust architecture,

standardized interfaces and interoperability capabilities. By examining the architectural designs, solution implementations and the added values provided by these FIWARE-based projects, it is possible to better understand and appreciate their contributions to the research purpose.

Chapter 3

State of the Art

Smart-City Digital Twin (SCDT) projects are emerging as valuable tools in improving the quality of life in urban spaces and enhancing city daily operations. By harnessing advanced technology and data-driven insights, these projects enable cities to optimize their daily flows and enhance overall management efficiency. Through the utilization of these projects, cities are empowered to transform their urban transportation systems while simultaneously creating a more livable and sustainable environment for their residents.

Throughout the next few sections are presented and reviewed notable examples and trends in this field of study. There is a clear categorization between non-FIWARE and FIWARE-based projects which gives a better overview of the key aspects, advantages and disadvantages of using different context management foundations. As a note, each one of these projects is carefully studied with the objective of finding patterns and approaches that enhance this thesis focus and core motivations. This way it was adopted a checklist system where it is possible to relate goal coverage on each approach by extracting valuable insights from all of them.

3.1 No-FIWARE Based Solutions

3.1.1 DUET

The DUET project [32], acronym for Digital Urban European Twins delivers a combination of several value propositions that makes its offering unique in the world of the SCDT technologies. From a societal point of view, DUET is distinctive in how it tests the Digital Twin (DT) concept of cooperation between policy domains in a quadruple helix setting in three entirely different locations in Europe:

- The **Athens** metropolitan area
- The belgian **Flanders** region of interconnected cities

- The medium-sized city in Czechia, **Pilsen**

This research was mainly motivated by the lack of unified models and frameworks for data fusions capable of linking the physical and virtual data exchange. With the goal of tackling this, DUET developed the T-Cell framework that acts as a container for models, data and simulations that interact dynamically in a common environment providing useful insights for smart-city decision makers. A dynamic correspondence that links the architecture with entity models and data turns possible the monitoring and synchronization of the state and behavior of the DT with its physical environment being mirrored in real-time. Individual models are integrated through APIs to form a cloud of models that can be called upon to perform various what-if urban analyses related to traffic, air quality and noise pollution.

Architecture

DUET's T-Cell framework relies on a central entity data broker to integrate different data models by means of suitable APIs. They can initiate or terminate a variety of tasks such as data exchange, calibration, validation and simulation; and can be accessible through the Apache Kafka platform embedded in the T-Cell architecture. Kafka is a platform developed by the Apache for real time environment usage, that takes advantage of distributed public-subscribed messaging system for handling a large volume of data and enables users to send messages at end-point [20].

Note: smart-city models then need to integrate this connection in order to communicate with the T-Cell.

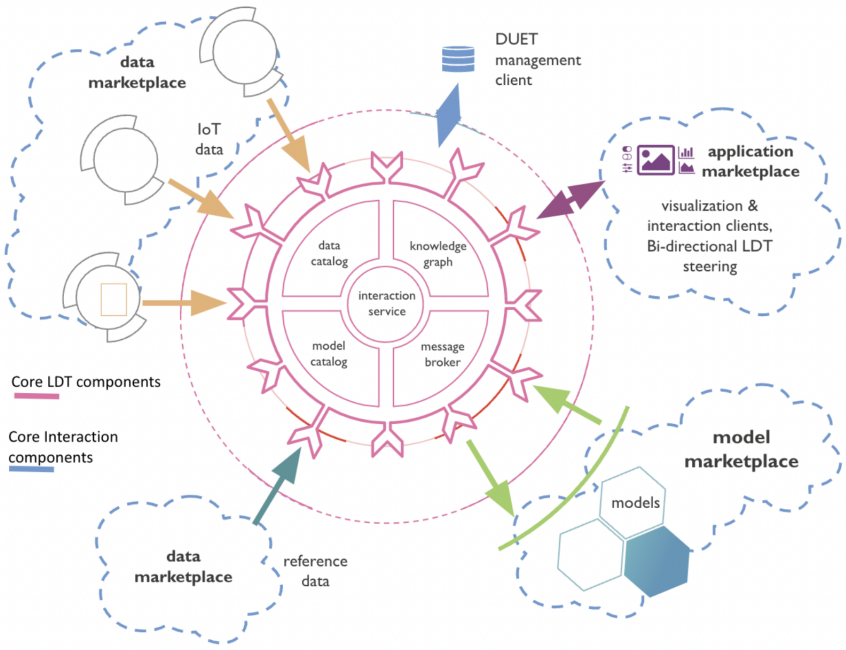


Figure 3.1: DUET's T-Cell architecture

The DUET solution architecture combines a number of guiding principles as shown above in figure 3.1. The main principle can be described as the following:

> **"Decouple components with different functions to maximise their interchangeability."**

Every component should be replaceable by a different component that fulfils the same role. A clear separation of concerns is required for this. This principle is especially important for the use of data sources and their semantics, simulation models and algorithms and the orchestration between both.

Solution

Individual models are made available through Docker containers [32], forming a cloud of models. To start, stop or retrieve the status of a particular model or a group of models, it is used the *Docker Orchestrator*[1] that automates the process of managing and scheduling the work of individual containers for applications based on micro-services within multiple clusters.

Other components of the T-Cell include a data register for administering different sources of information like Internet of Things (IoT) event sources or time series data, a data catalogue for storing descriptions of data sources in the form of schemas and metadata, an event module for handling IoT interaction and simulations and a security module for tenant configuration, user authentication and access management.

In order for DUET to offer useful insights to city planners, their system is configured to normalize all incoming data into a common data model. This turns the compatibility of datasets a lot easier and facilitates the integration into a working solution. The structuring of metadata has a strong effect on the ease of carrying out data searches.

Data search may sound like a trivial task, however, efficient data search is a prerequisite for the effective use of a DT. The ability to carry out this type of search efficiently underpins the ability to merge datasets that are gathered at different points in time and space but relate to the same object such as roads or buildings. Metadata also plays a key role in the use of long-term persistent data. To improve the semantic interoperability, DUET relies on linked data instead of static data models. Common interoperable standards, such as *NGSIv2*[2] and *NGSI-LD*[3], as well as some *OGC*[4] [31] standards, such as *CityGML* [5], are supported in the T-Cell by default.

[1] A tool used to manage and automate the deployment, scaling and management of containerized applications.

[2] An API specification which defines a standard way to interact with context data in a machine-readable format.

[3] An extension of NGSIv2 that leverages the concepts of linked data to provide a more semantically rich and interoperable representation of context information.

[4] Stands for Open Geospatial Consortium and is an international organization that develops and promotes standards for geospatial and location-based technologies.

[5] A standard that defines a conceptual model and exchange format for the representation, storage and exchange of virtual 3D city models which facilitates the integration of urban geodata for a variety of applications for SCDT.

Added value

The project delivers a unique combination of value propositions that make it relevant and valuable in the realm of SCDT. It stands out in its approach to testing the concept of cooperation between policy domains in a quadruple helix setting across the proposed european locations.

One of the key aspects that sets DUET apart is its focus on addressing the lack of unified models and frameworks for data fusion, specifically those capable of linking physical and virtual data exchanges. To overcome this challenge, DUET with the development of the T-Cell framework, acting as a container for models, data and simulations that interact dynamically in a shared environment [32]. This framework enables smart-city decision makers to gain valuable insights by integrating various models through open-source APIs and performing what-if analyses related to traffic, air quality and noise pollution.

The architecture of DUET's T-Cell framework embodies several guiding principles, with a primary emphasis on decoupling components to maximize interchangeability. This principle allows for the replacement of components with alternative ones fulfilling the same role, promoting flexibility and separation of concerns. Furthermore, the integration of Apache Kafka, facilitates efficient data exchange and communication between smart-city models and the T-Cell.

The proposed solution offers added even more value through its containerization process, which house individual models that form a cloud of models. These containers can be managed using the Docker Orchestrator [32], streamlining the process of starting, stopping and retrieving the status of specific models or groups of models. Additionally, the T-Cell incorporates various components such as a data register, data catalog, event module and security module, providing comprehensive support for administering different data sources, handling IoT interactions and simulations while ensuring a secure access and configuration [41].

In order to provide practical insights to city planners, the DUET project adopts a data normalization approach, where incoming data is transformed into a common data model. This normalization process enhances compatibility and facilitates seamless integration into the overall solution. Metadata structuring plays a pivotal role in enabling efficient data searches and merging datasets collected at different times and locations but pertaining to the same urban objects. To achieve semantic interoperability and effective data integration, DUET leverages linked data and industry-standard interoperable frameworks.

This comprehensive approach, implemented through its T-Cell framework, demonstrates its relevance in the domain of SCDT. The project addresses challenges related to data fusion, model interoperability and scenario analysis and its value proposition stems from the decoupled architecture of its components, the utilization of container-based model clouds, the capability for data normalization and support for semantic interoperability standards. By empowering decision makers with these powerful tools, DUET enables them to tackle urban challenges and make informed, evidence-based decisions for creating sustainable and livable cities.

Use cases

Pilot 1: Athens

Athens is a metropolitan area that suffers from a lot of traffic congestion and large surges of tourist inflows. The current challenge for the city is to create an interactive pool of urban data that will be dynamically updated, open, robust and usable for evidence-based decision-making, enhancing the capital's attraction for locals and visitors. Athens, therefore, has a need for an integrated DT with the capacity to merge all of the city data and make it easily accessible and useful for dealing with traffic and air pollution.

Pilot 2: Pilsen

Pilsen is a mid-sized city in the western part of the Czech Republic. Being a hub for commuters, retail, entertainment and tourism, Pilsen is facing many challenges in terms of transport planning and urban development. Using both traditional and new data sources (e.g., crowdsourced data from mobile phones) to feed its DT, the czech city wants to simulate different urban design scenarios and measure their impact on the quality of life over an extended period of time.

Pilot 3: Flanders

Flanders is an urbanized, densely populated region with a very busy road network. The slightest issue or the smallest accident during the rush hour can trigger very long tailbacks. No wonder the region is also a hot spot for air pollution. The DUET solution for Flanders helps designing, implementing and evaluating new policy measures. The ultimate goal is to effect a change that protects the environment and reduces negative impacts on human health.

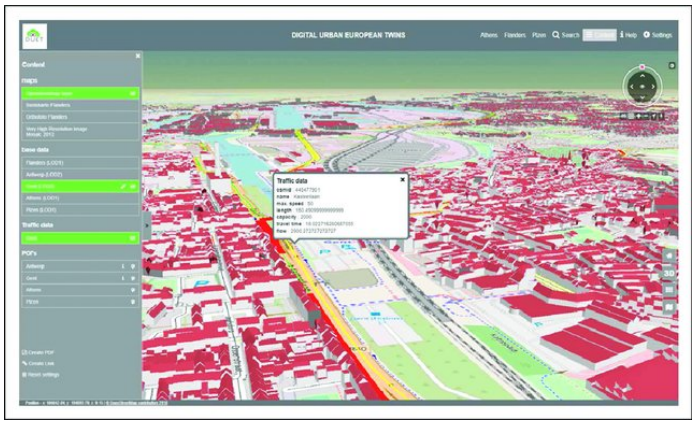


Figure 3.2: DUET alpha version for the three pilots

Leveraging traffic, air and noise models in a DT environment, DUET pilots explore, via the 3D interface, shown in figure 3.2, a range of valuable what-if scenarios.

The three pilots mentioned above share common issues related to traffic, air quality and high levels of noise in their respective urban environments. By applying the three proposed smart-city models on these pilots, DUET aims to address and improve these specific areas, ultimately contributing to the evolution of urban environment quality.

- **Traffic models**

  DUET utilizes three types of traffic models, a static, a dynamic and a local mobility model called Cityflows, which is a descriptive model used for data fusion and extrapolations to future scenarios [41]. Static and dynamic traffic assignment models offer a comprehensive understanding of traffic flows and enable predictions based on alternative road network configurations and varying demand patterns like the instance given in 3.3. These models are valuable in simulating a wide range of scenarios using DT technology. For instance, they can be used to assess the effects of new developments on traffic, predict the impact of tolls on route choices or simulate the consequences of lane closures due to construction activities.

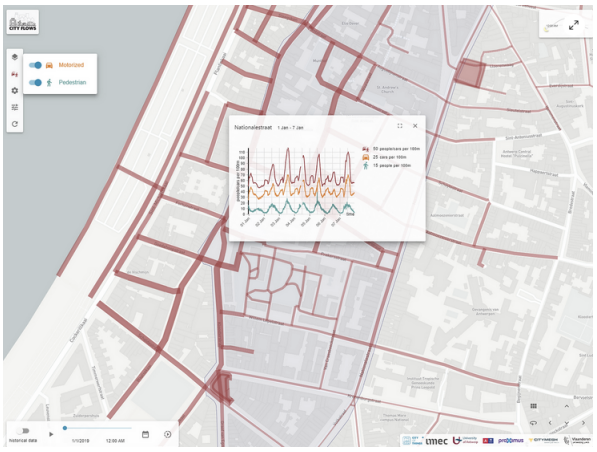


Figure 3.3: Example of Cityflows output visualization

- **Air-quality models**

  These models employ quantitative methods to simulate the physical and chemical processes that impact the dispersion and reaction of air-pollutants in the atmosphere. In DUET's architecture, the air quality model calculates emissions based on traffic volume and road network data, taking into account variables like the wind speed and direction. Subsequently, the model calculates the spread of air pollution resulting from traffic for a grid of specific calculation points. The outcomes are then transformed into map images using interpolation and heat-mapping techniques as presented in 3.4.

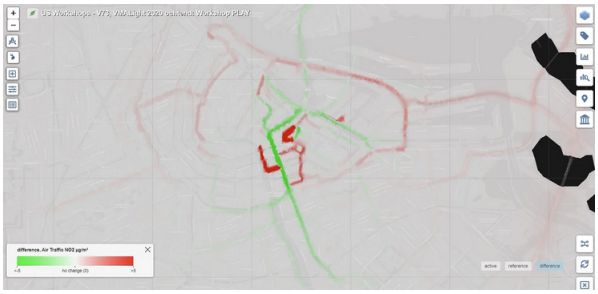


Figure 3.4: Example of a difference plot for NO2 showing the impact of a road closure

- **Noise emission Models**

  Urban areas experience environmental noise primarily due to industrial activities and various forms of transportation. Within the DUET project, the aim is to leverage readily available open-source libraries like NoiseModelling, which can generate noise maps of cities and is accessible for both research and professional use [41]. Additionally, tools such as the Urban Strategy Noise Module are utilized to generate noise level maps and provide data for deriving noise exposure distributions [41] like the example shown in figure 3.4. These resources contribute to the analysis and understanding of noise pollution in urban environments.

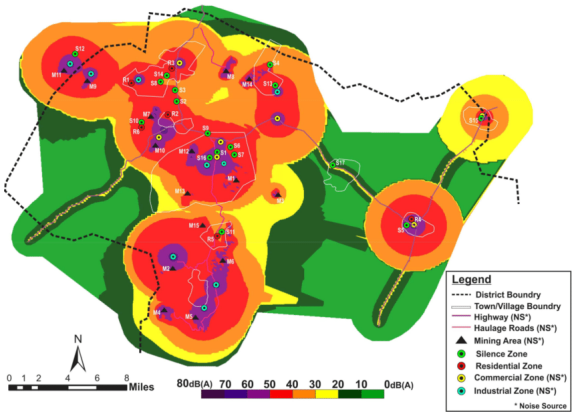


Figure 3.5: Example of an output noise-level map

3.1.2 Herrenberg Digital Twin

Digital technologies and their applications play a crucial role in facilitating smarter and more sustainable governance of cities. They help bridge formal and rational knowledge, which enables a more effective urban planning processes that involve the experiences and input of citizens. Cities are intricate systems influenced by economic, ecological and demographic factors, and they undergo constant change. Additionally, cities are characterized by diverse perspectives and interests among citizens and stakeholders. This research focuses on a specific case study on the town of Herrenberg. The case study develops a three-dimensional city model using the geographic data provided by regional authorities. Resultant findings emphasize that digital tools

have a lot of potential to support citizen participation; however, it is essential to integrate them within meaningful dialogues to ensure their effectiveness and relevance [29].

Architecture

This research involves 3D interactive simulations in Virtual Reality (VR) in combination with qualitative and quantitative data, like the figure 3.6 demonstrates below. The modelling approach includes a mixed method of three-dimensional models, simulations, 3D mapping and a street network model that utilizes space syntax which is detailed and explained in the following solution section. For fostering citizen participation, the DT was embedded in the collaborative visualisation and simulation environment *COVISE*[6]. This DT for Herrenberg was exposed to approximately 1000 citizens using both a mobile and stationary virtual reality environment; and for its verification and consolidation, a public survey was carried out. The results demonstrated that the use of this method and technology could significantly aid in participatory and collaborative processes.

The architecture around COVISE provides a flexible framework for developers to enhance its existing functionality by integrating new modular plugins and also including the render module *OpenCOVER*[7]. Additionally to this, the framework offers support modules for reading results from simulation codes like *OpenFOAM*[8], which extracts geometry details and generates visualization features such as isosurfaces and streamlines which both stand as the two main visualization techniques to represent and analyze data within the SCDT.

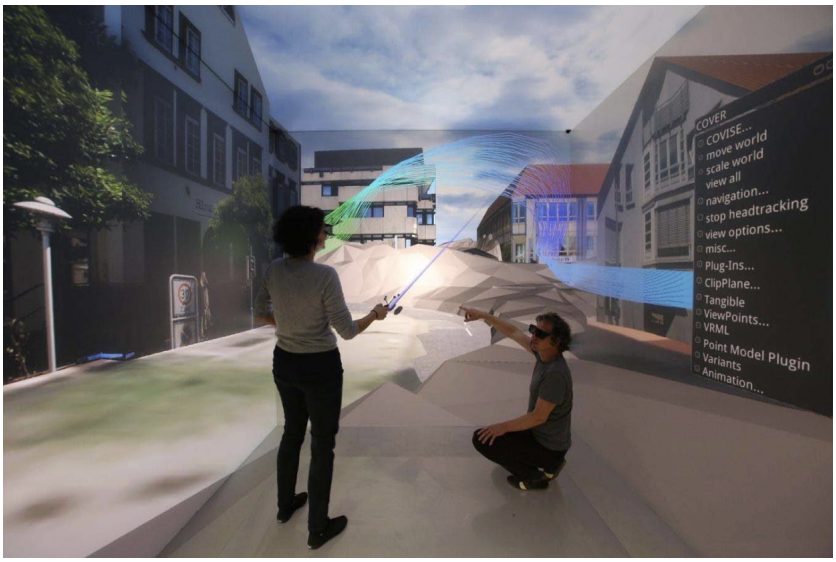


Figure 3.6: Virtual simulation visualized in the CAVE

[6]Stands for Collaborative Visualization and Simulation Environment and it is a software framework and platform for developing and deploying interactive visualizations and simulations.

[7]A software framework for interactive three-dimensional visualization and rendering, specifically designed for virtual and augmented reality environments.

[8]Stands for Open Field Operation and Manipulation and it is a free and open-source software package that includes pre-processing utils for creating computational grids, defining boundary conditions, and setting up simulation cases.

Note: CAVE is a stationary five-sided cube for VR back-projection run by a 22-node cluster, five 3D-projectors, a tracking system and active shutter glasses for the participants.

Solution

All the solution development focused on an empirical and computational method using a mixed approach. The SCDT is set up as follows:

1. **The environment is built over a 3D morphological model**

   The SCDT proper functionality relies on a well-established three-dimensional representation of the city itself, which is created using geographic data like the digital elevation and digital building models provided by regional authorities. These data sources serve as the foundation for constructing the DT.

   The Herrenberg model is a combination of detailed representations of specific neighborhoods. The level of detail in the model is carefuly determined based on the area being analyzed, where accurate information about the potential environmental impact on the neighborhood is necessary. To capture the three-dimensional scan, a *Laser Scanner Focus 3D S 120*[9] is utilized. These scanners offer a quick and accurate method for capturing the intricate geometries of objects and entities within a real city scenario for an efficient digital representation. The scanner's built-in camera captures high-quality images in natural colors, even under challenging lighting conditions and its mobility allows for efficient scanning of large areas within a short time-frame.

   Additionally, these building information models are integrated to include building data beyond just geometry. The integration extends to include data from *HVAC*[10] technology, as well as information from electricity, water and sewage networks, if available, capable of enhancing the general capabilities of the SCDT.

2. **The street network model is constructed using space syntax**

   The application of space syntax theory and methodology, which makes use of a graph-based approach, helps analyzing the movement potentials for cars and pedestrians within a road network [4]. It measures the normalized angular choice, which considers the shortest angular routes from all nodes to each other in a network, to understand the centrality of street segments in relation to others in the urban system web which indicates the potential usage or traffic levels of the streets [30].

   To model the street network, a hybrid approach is used which geo-referenced road-center lines with *axial lines*[11]. The space syntax method is applied to gain a deeper understanding

[9] A specific model of laser scanner manufactured by Faro Technologies, which is also a portable and high-precision device used for capturing three-dimensional data of objects and environments.

[10] Stands for Heating, Ventilation and Air Conditioning which refers to the technology used to control the indoor environment of buildings, vehicles and other enclosed spaces.

[11] Sightlines representing movement paths.

over the spatial configuration of the street network, which refers to the underlying logic of the urban system's publicly accessible places. This analysis goes beyond visualizing traffic census data and focuses on the spatial arrangement of the streets. By understanding spatial configuration the development and testing of scenarios within the entire system are unlocked and facilitated.

The street network graph is computed using the open-source software called *depthmapX*[12]. To integrate the space syntax model into the SCDT, the street network is transformed from two-dimensional to three-dimensional data for its efficient visualization in the virtual reality VR environment. New modules for COVISE and OpenCOVER software are developed to automate and support this two into three-dimensional representation processing of space syntax data. This enables the immersive exploration of space syntax analysis within virtual realities.

3. **There is a urban mobility simulation based in *SUMO*[13] and airflow simulation**

   To gain insights into traffic patterns, the model was enhanced with a traffic simulation system using SUMO. This simulation allowed for the visualization of several different vehicles, including cars, trucks, buses, bicycles and also pedestrians, moving within the virtual city model. A new plugin was also introduced to the SCDT which enables real-time simulation and visualization of changes in modal split and travel volume. This feature makes easier to test and demonstrate different scenarios and strategies for addressing traffic congestion and reducing exhaust emissions.

   In addition, the SCDT incorporates an airflow simulation which considers official weather and climate data from environmental sensors. This simulation combines data on gas emissions with traffic volume and wind patterns, as well as factors like temperature and humidity. This integration enables the analysis of how emissions are distributed and affected by environmental conditions.

   Both integrations provide a comprehensive understanding of traffic behavior and its impact on air quality. It allows for the exploration of different scenarios and the evaluation of strategies to reduce congestion and gas emissions.

4. **Gathered qualitative and quantitative data from people's movement routes, social data and photographic impression**

   In order to collect empirical data using volunteered geographic information, a mobile application called Reallabor Tracker is specifically designed and developed for this purpose. The application, built upon the *OSM Tracker*[14], enables users to track their daily routes, assess the quality of public spaces, document stationary activities, capture images and sound samples, and even add comments or notes for community insight sharing. Users can

[12] An open-source software tool used for spatial network analysis, specifically in the field of space syntax.

[13] Stands for Simulation of Urban Mobility and it is an open-source microscopic traffic simulation software that allows for the modeling and simulation of traffic scenarios in urban environments.

[14] A mobile application that is based on OpenStreetMap, which is an open-source map of the world.

record their routes, including information on speed, which allows for the analysis of the distribution of transportation modes and stationary activities in public spaces.

5. **Pollution simulation based on an empirical data set resultant from a preconfigured sensor network**

   To collect empirical data from the urban environment, a set of linked sensors must be deployed. These sensors measure various parameters such as particulate matter, temperature and humidity. The research team provides pre-built sensors to dedicated citizens who install them in suitable locations, including outdoor spaces like balconies and rooftops. The selection of households and sensor placement tooks into account different traffic levels and frequented areas to ensure comprehensive data coverage.

   The collected data from this sensor network is then correlated with the space syntax model. The model calculates values for different route segments and provides insights into the spatial configuration of the urban environment. Additionally, data originated from the traffic simulation is integrated into the analysis. The simulation takes into account official weather and climate data which enables researchers to study the relationship between emissions and factors like wind direction, temperature and humidity. This approach is not limited to specific emissions but can be extended to simulate other pollutants such as NO_2 and CO_2 and to analyze climate-related phenomena like floods, urban heat islands and air flow patterns.

Added value

The Herrenberg Digital Twin project brings several valuable contributions to the field of SCDT. Over the details proposed in the architecture and solution descriptions, these contributions can be summarized as follows:

- **Enhanced citizen participation**

  The project demonstrates the potential of digital tools in supporting citizen participation in multiple urban planning processes. By integrating the DT within a collaborative visualization and simulation environment, citizens are empowered to actively engage with the virtual representation of their city. The use of mobile and stationary VR environments allows multiple citizens to interact with the DT at the same time, providing their input and feedback in a more efficient way. This approach fosters a much more inclusive and participatory decision-making process, as the figure 3.7 suggests, which ensures that the experiences and perspectives of citizens are considered in the urban planning and development processes.

Figure 3.7: Herrenberg participatory and collaborative citizen involvement

- **Integrated data analysis**

  The initiative integrates various data sources and analysis methods to provide a comprehensive understanding of the city environment. By combining both qualitative and quantitative data, including geographic data, space syntax modeling, urban mobility simulation and pollution simulation outputs, the DT instance enables researchers to analyze complex urban systems from multiple perspectives. This integration of diverse data sources allows for a holistic view of the city and facilitates informed decision-making.

- **Real-Time visualization and simulation**

  The SCDT leverages real-time visualization and simulation capabilities to support dynamic analysis and scenario testing. With the use of technologies like VR and software focused on traffic simulations such as SUMO, the solution enables the visualization of real-time traffic patterns, modal split changes and even the distribution of emissions considering factors like wind direction and climate conditions. This capability empowers urban planners and decision-makers to assess the potential impact of different interventions and strategies in real-time, and once more supporting evidence-based decision-making processes.

- **Urban Sustainability and Environmental Analysis**

  The Herrenberg digital intervention emphasizes the importance of addressing sustainability and environmental considerations in urban planning. By integrating pollution sensors and correlating their data with the DT, the project enables the analysis of air quality, gas emission distribution and the impact of traffic on some environmental issues. This provides valuable insights for designing strategies to reduce congestion, improve air quality and possibly mitigate the environmental impact of these activities.

- **Flexibility and Scalability**

  The architecture proposed for this study case, which was designed around the COVISE framework premise, offers the flexibility and scalability needed for future developments

and integration of new functionalities. The modular design allows for the incorporation of additional plugins and tools, which makes easier the expansion of the SCDT capabilities over time. This aspects ensure that the digital representation of the city can adapt to evolving urban challenges and incorporate emerging technologies and data sources.

Use cases

Herrenberg direct application

Herrenberg, located south of Stuttgart, faced a natural urban growth and automobile traffic levels raise which consequently led to an amplification on the environmental pollution levels and increased overall noise. To address this challenge, the integrated mobility plan *IMEP 2030*[15] was developed with the support of civic engagement. The Herrenberg DT project can directly apply the proposed solution in this context. By integrating the DT within a collaborative visualization and simulation environment, it enables participatory and collaborative planning.

Participatory and collaborative planning with SCDT

Digital tools have the potential to enhance the overall citizen participation, particularly through the visualization of SCDT in virtual and augmented realities. This capability can greatly support communication and decision-making processes between city-planners, responsible administrative staff, and citizens who may not have technical expertise in the field. By providing these virtual representations of the city easily accessible, the communication gap between different stakeholders is efficiently bridged. Simulations and visualizations offer by these tools to simplify complex information and present results in a more accessible manner. As a result, participation naturally becomes more inclusive.

3.2 FIWARE Based Solutions

3.2.1 CO2-Mute

Due to the worrying climate changes and pollution levels rises, the European Commission (EC) has set the ambitious target of at least 55% net reduction in greenhouse gas emissions by 2030 in the European Union (EU) by 2030 [12]. If member states are responsible for defining strategies and targets, it is the responsibility of cities and urban regions to put these policies into practice adapting them to their own contexts.

The urban mobility sector is one of the biggest contributors to the negative environmental impacts due to the high levels of carbon dioxide emission. To be able to reduce the traffic congestion and its related environmental impact, local governments must invest in Mobility-

[15] A comprehensive strategic plan developed for the city of Herrenberg to address mobility challenges and promote sustainable transportation solutions.

as-a-Service (MaaS) strategies and infrastructures to encourage the use of public transport or bicycles.

The main motivation behind this project is to be able to give support to local governments in their efforts to deploy policies for a more sustainable mobility system and urban green infrastructure as part of their digital transition strategy. Additionally, CO_2-Mute [15] prepares citizens for their participation in the implementation of these policies to change their mobility habits. As a natural way to motivate and gather people to join the cause, it is made an effort into gamifying the alternative mobility usage and enhancing the role of green spaces in pollution mitigation by proposing collective and individual achievements for commuters and workers. These achievements can be used to evaluate impacts on costs, traffic and the overall environment.

Architecture

From a technical perspective, all partners implement their own instances of the *KeyRock*[16] and *Kong*[17]. The information flow in CO_2-Mute mainly consists of two phases:

- **Phase 1**: Data collection/analysis during the period of local mobility challenges where mobility and environmental information is collected and analyzed to understand the local situation;
- **Phase 2**: Impact and results dissemination after the end of each local mobility challenge.

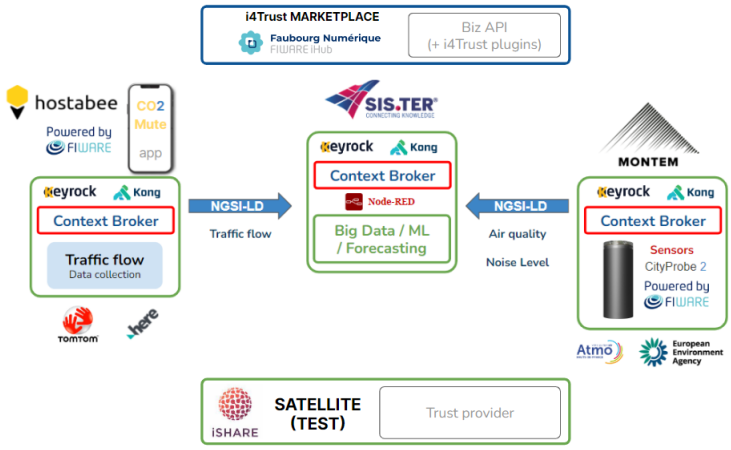


Figure 3.8: CO2-Mute architecture diagram

Data collection, sharing and analysis during the mobility challenges

All of the data collected can be accessible to the data science service provider via the NGSI-LD API. This provider can then remotely analyze the relevant data using Big Data and Machine

[16] An identity and access management component within the FIWARE platform.

[17] An open-source API gateway and service mesh platform that acts as an intermediary between clients and services, providing features such as routing, authentication, rate limiting, load balancing and logging.

Learning (ML) techniques through the Context Broker (CB) and specific *Node-RED*[18] processors that make the data accessible for tools like *PySpark*[19] as it is possible to see above in figure 3.8.

Impact and results dissemination

The results of the analysis conducted in the project are shared in the marketplace using a suitable FIWARE data model. The Key Performance Indicator (KPI) was selected as the most appropriate for the project's goals which model allows for a wide range of results and has keys aligning with the project's main motivations. The analysed data is then submitted to the marketplace, which submits the policies to the relevant data owner's authorization registries for access by the application provider.

Identification, authentication and authorisations

As all stakeholders are involved in the project as data providers, they are required to make their services available on the marketplace instance. The data flows in the project involve only Machine-to-Machine (M2M) interactions between the stakeholders' systems and to ensure this exchange is mainly secure, all partners are required to obtain certificates from *iSHARE*[20] and authenticate their systems through the use of *JSON Web Tokens*[21]. This validation process involving iSHARE satellite is also used to validate access tokens when they are requested.

Solution

CO_2-Mute makes use of a variety of data sources, such as:

- Traffic data from prominent companies in the field of location-based services and mapping technologies APIs (Tom-Tom and HERE);
- Data from five air quality and noise sensors provided by MONTEM A/S;
- Data obtained from local weather stations about the environmental status and time range.

All this data is correlated and fed to an algorithm that calculates the estimated impact of traffic reduction on the air-pollutant concentrations. The results of the analysis are then made available to the local government, and after evaluated, are integrated into the urban planning decision processes. At each stage of the value proposition, seamless access to heterogeneous data from various sources and data providers is crucial to deliver relevant services and reliable analysis.

[18] An open-source visual programming tool by the JS Foundation that simplifies app development by connecting nodes, each one of them representing a specific function or operation, and the connections between them defining data flows.

[19] The Python library for Apache Spark, knowing that Apache Spark is designed for Big Data processing and analytics, providing high-speed data processing capabilities and support for various data sources.

[20] A framework for secure and controlled data sharing among organizations which is designed to facilitate trusted and standardized data exchange between parties, enabling seamless collaboration across different systems.

[21] A compact, URL-safe way of representing claims between two parties.

CO_2-Mute has a high level of flexibility in its configuration and a strong ability to deal with various types of data thanks to:

- Interoperability by design approach;
- The use of a DT platform implementation based on the NGSI-LD standard.

Note: this ability to customize and correlate data based on local situations is crucial for measuring impacts.

Added value

From a technical standpoint, CO_2-Mute employs a robust architecture, leveraging advanced technologies such as data analytics, Big Data and ML.

The biggest added value lies in the system's ability to provide customizable solutions tailored to local contexts. By integrating various data sources, including traffic data, air quality, noise sensors and weather information, CO_2-Mute offers a comprehensive analysis on the impact of traffic reduction on air-pollutant concentrations. This data-driven approach enables local governments to evaluate the effectiveness of their policies and make informed decisions regarding urban planning and mobility strategies.

Its relevance core lies in a holistic and data-centric approach to sustainable urban mobility, combining gamification, green infrastructure and advanced data analysis. By empowering local governments and citizens, the project actively engages them in creating a more sustainable future and addressing the pressing challenges of climate change and pollution.

Use cases

Once implemented, the project, used by local governments, presents several compelling use cases that leverage its data-driven approach to tackle the worrying carbon dioxide emissions and promote sustainable urban mobility. It can be very game-changing in terms of:

Traffic Congestion and Pollution Mitigation

CO_2-Mute addresses the pressing issue of traffic congestion and its adverse environmental impacts. By collecting real-time traffic data from sources such as the one coming from location-based and mapping APIs, the whole system can analyze traffic patterns and calculate the estimated impact of traffic reduction on air-pollutant concentrations. This information enables local governments to make informed decisions regarding urban planning, infrastructure development and traffic management strategies, ultimately leading to a proper reduction in CO_2 emissions and generally improved air quality, which was one of the main motivations of it.

Gamification of Sustainable Mobility

To encourage citizens participation and engagement in sustainable mobility practices, CO_2-Mute incorporates some gamification elements. The project aims to motivate commuters and workers to adopt alternative means of transportation such as public transport or bicycles. Through this gamified approach, individuals can earn collective and individual achievements based on their usage of green mobility options. By transforming sustainable mobility into a rewarding and interactive experience its is possible to better incentivize behavior changes and promote the transition towards a more sustainable urban mobility ecosystem.

Impact Evaluation and Policy Development

This implementation facilitates the evaluation of the impact on mobility-related policies and measures implemented by local governments. By collecting and analyzing mobility and environmental data during specific periods of local mobility challenges, the project generates valuable insights into the effectiveness of possible different strategies. The results on these analyses, shared through the suitable FIWARE data model, are able to inform policymakers about the impacts on costs, traffic congestion and the overall environment.

Collaboration and Knowledge Sharing

Collaboration is promoted among various stakeholders, including local governments, data science service providers and mobility solution providers. The project establishes a marketplace where the results of the analysis process are shared. This collaborative flux facilitates knowledge sharing, best practices exchange and the dissemination of impactful solutions. By leveraging the expertise and resources of diverse participants, CO_2-Mute fosters innovation, accelerates the adoption of sustainable mobility practices and contributes to the development of a vibrant stakeholders ecosystem working towards a common goal.

3.2.2 Snap4City

The motivations behind the Snap4City project [3, 19] align with the ones behind this thesis project. Cities are undergoing rapid transformations in order to meet the diverse societal, environmental, and economic demands of today. In response to this, there is a shift from relying on narrowly focused smart solutions to adopting more comprehensive solutions that can leverage a wide range of data channels. The goal is to enable cities to effectively address the multifaceted challenges they face and provide a broader array of services to their residents.

Currently, each urban area has its unique set of requirements, making it necessary to have a highly adaptable solution. A flexible and dynamic platform is crucial for empowering cities to have full control over their operational objectives. Traditional approaches that rely on limited data sources are being replaced by a more comprehensive understanding of data channels, where information and actions flow in multiple directions. The implementation of a city Operating System (OS) requires the ability to adjust and respond to the daily challenges faced by urban

regimes. This adaptability is essential for cities to become smarter, driven not only by internal motivations but also by external regulations imposed by international bodies like the EC.

Architecture

As mentioned earlier, Snap4City has the ability to interact with various types of data channels and integrate them with multiple FIWARE solutions. Once these data channels are established, city entities can be represented in a knowledge base. This enables semantic queries that exploit spatial, temporal, and relational relationships between different entities. This system also facilitates the connection and management of multiple brokers and devices through an IoT directory. Additionally, remote IoT edge devices can be maintained, allowing for control logic updates and data processing using Node-RED. Snap4City offers the capability to handle both open and private data for different domains and organizations, ensuring a more secure and customized data management. This schema can be better analyzed below, with figure 3.9.

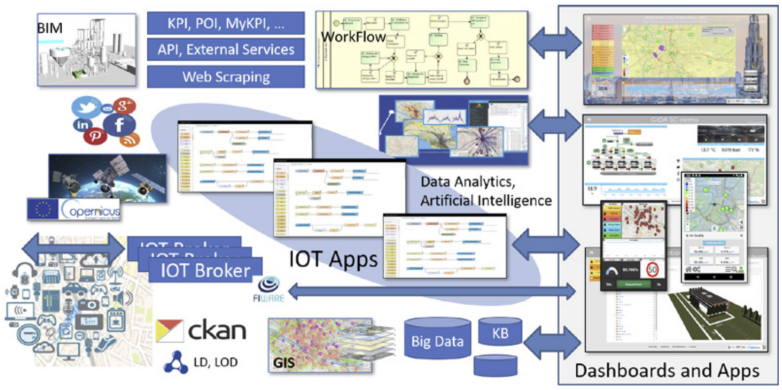


Figure 3.9: Snap4City component communication architecture

The Dashboard Builder offers a wide range of graphic rendering tools allowing users to visualize and interact with complex data in various formats such as:

- Maps;
- Orthmaps;
- Tables;
- Trends;
- Heatmaps;
- Traffic flows.

By making use of a *wizard*[22], users can easily create vertical solutions or build complex applications with multiple dashboards and tools. This Dashboard Builder enables event-driven solutions capable of operating in real-time, providing interactive web tools and mobile apps for both operators and end-users. It supports a rich set of interactive features, including:

- DT representation;
- Context navigation;
- Integration with workflow management systems for ticketing and management;
- Synoptics for industrial monitoring.

Additionally, the Dashboard Builder prioritizes data protection and compliance with the General Data Protection Regulation (GDPR) regulations, ensuring that user privacy and data security are maintained throughout the platform.

Snap4City seamlessly integrates with powerful data processing tools like Node-RED. As the following figure 3.10 suggests, the project offers an extensive library of micro-services within the tool, enabling the creation of data adapters, business logic and data transformations. These services are highly versatile and can be utilized to compute various indicators and KPI required by cities. This includes important metrics like EC indicators for pollutants, ISO_{37120} and ISO_{37122} standards, city indexes and more. With these capabilities, Snap4City empowers cities to effectively analyze and monitor their data, facilitating informed decision-making and sustainable urban development.

In addition, this platform fully supports the development of real-time data analytic processes through ML, Artificial Intelligence (AI) and statistical languages such as:

- Python;
- Java;
- R-Studio.

and also exploits data analysis libraries like *Tensor Flow*[23] and *Keras*[24].

[22] A user-friendly feature in a software application that guides users through a series of steps to complete a specific task or configuration.

[23] An open-source ML framework developed by Google and designed to facilitate the development and deployment of ML models, particularly deep learning models.

[24] An open-source high-level neural networks API written in Python and designed to provide a user-friendly interface for building and training deep learning models.

Figure 3.10: Snap4City system overview

Snap4City actively develops and shares a range of open-source data analytics tools and algorithms which cover various domains such as heatmap generation, anomaly detection and general prediction.

Solution

As proposed by its architecture, Snap4City is a comprehensive smart-city platform which combines data integration, analysis, visualization and application development capabilities. It gives an answer to some core features such as the ones this thesis seeks in the aftermath and it is now managing data, with on-cloud and on-premise solutions, of currently more than 40 urban areas in european countries such as Italy, Spain, France, Finland, Belgium, Greece, Croatia and Sweden.

The primary objective of Snap4City is to assist cities and businesses in enhancing performance and reducing costs by connecting various management and control domains. The platform leverages the FIWARE framework and places at its core the Orion CB, which supports secure communications and multiple broker connections through both Next Generation Service Interface (NGSI)v1 and v2 protocols. This interoperability, combined with the system flexibility and modularity, enables the development of applications across a wide range of scenarios, including those mentioned earlier. Snap4City facilitates the establishment of federations of smart-cities through its smart-city API and NGSI-based solutions. All involved APIs are well-documented and easily accessible for developers, allowing for a proper contribution and customization. By integrating this with the multi-tenant CB, the system can efficiently collect and process data.

Note: Snap4City promotes the concept of living-lab development and provides a supportive environment for developers.

The entire solution has successfully passed *PEN tests*[25] and vulnerability tests, ensuring a high level of security and compliance with GDPR regulations, which is essential for SCDT projects. With Snap4City, users can develop end-to-end event-driven applications that prioritize security. This includes seamless connectivity from devices to dashboards, incorporating data processing, storage, and analytics. All the development process was made open source, covering all layers of the application, as well as aspects of multi-tenancy, assessment and auditing. It is important to note that Snap4City can be installed on both private and public clouds, making use of an open source Virtual Machine (VM) or a `docker compose` setup, with no need for any licensing requirement. It offers a range of configurations suitable for various scales, accommodating small standalone installations to large-scale deployments.

Added value

The usage of Snap4City has brought many improvements and has been a great benefit to a wide range of situations where it has been implemented. It can be freely installed on premise with its full solution components. Thanks to FIWARE's openness, interoperability and spread, it enables for a faster integration and exploitation of the IoT aspects in the SCDT. The first usage of FIWARE technology by Snap4City was in the fields of smart-industry and smart-cities.

The project has received recognition by having its research published in reputable journals, specifically focusing on topics like:

- Smart-parking;
- Smart-biking;
- Traffic flow analysis and prediction;
- NO_X and NO_2 levels prediction;
- People flow analysis;
- Public transportation analysis;
- Routing.

The integration of data analytics extends to predictive analysis tools, offering valuable insights for control rooms and operators. This comprehensive approach enhances the project's ability to support urban management and decision-making processes effectively. Also the possibility of establishing bi-directional connections with data channels in these domains, enables higher levels of integration and exploitation, thus allowing the generation of unexpected hints and deduction.

[25]Security assessments that mimic real-world attacks to identify vulnerabilities in computer systems, networks or applications.

Use cases

- **Mobility and transport**: reducing people congestion, traffic congestion, monitoring and controlling traffic flow, simulating and analysing mobility and transport, smart parking;
- **Environment**: predicting NO_X and long term NO_2, monitoring pollutants of any kind and alerting, informing city users;
- **Energy**: recharging stations monitoring, smart light control;
- **Strategic planning**: performing what-if analysis with respect to critical conditions, planning production, system thinking on smart decision support systems;
- **City management**: predicting maintenance interventions, multichannel alerting, anomaly detection as early warning, etc.;
- **People flow**: monitoring and alerting on critical cases.

3.2.3 Orchestra Cities

According to a 2006 report on the urbanization in developing countries [9] , it is projected that by the year of 2030, approximately 61% of the global population will reside in urban areas. This rapid urbanization process poses various challenges such as:

- Limited space;
- Outdated infrastructure;
- Strained public services;
- Environmental sustainability issues;

This also has some negative impacts on the environment and people's general well-being. To address these challenges, cities worldwide are leveraging Information and Communication Technology (ICT) solutions. However, there are still obstacles to overcome, such as compatibility issues and lack of interoperability among these technological platforms. It is crucial to have open-source systems that are not only independent but also sufficiently scalable to meet any future demand. The focus should not be solely on optimizing individual products but rather on optimizing the entire ecosystem, ensuring that technologies can connect with each other in a seamless way, including future technologies.

Ecosystems, particularly in complex socio-technical systems like a SCDT, should be cultivated and nurtured rather than constructed. This entails adopting an open approach that transcends rigid system boundaries. Various parties such as city authorities, citizens, service providers like urban mobility, waste management, street lighting and businesses should be empowered by the

platform to actively participate in the city community and contribute to its growth and success. Additionally, the current climate crisis and the urgent need for circular economy practices, as well as the demand for sustainable solutions throughout the entire lifecycle, have introduced a whole new aspect to the future landscape of the smart-cities.

Architecture

From a technical standpoint, the components composing Orchestra Cities [17, 27], which include:

- Data acquisition;
- Data transformation;
- Data visualization;
- Platform management.

are implemented as a collection of numerous services. These services are distributed across multiple nodes within a cloud infrastructure and can dynamically adjust the number of instances based on the current computational workload. This design follows the principles of micro-service architecture, which promotes the development of small, independent services and employs elastic resource management techniques to optimize performance in a cloud-native software environment [24] like this one.

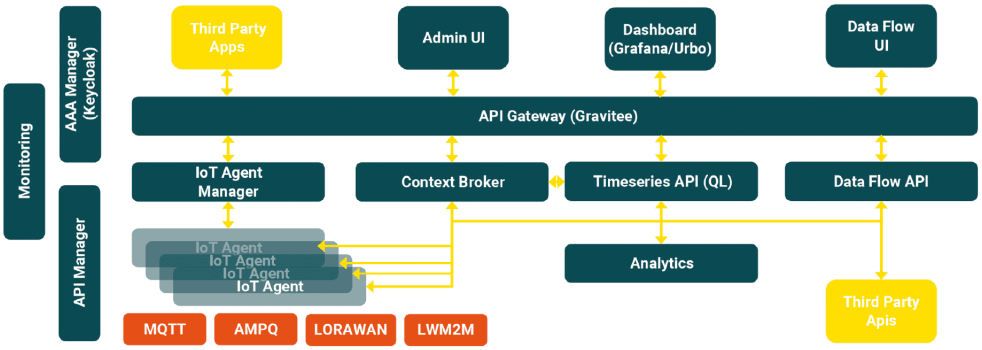


Figure 3.11: Orchestra Cities architecture diagram

The Orchestra Cities concept builds upon the principles established by FIWARE and aims to expand upon them, which can be deducted from the above figure 3.11. This approach influences the selection of software technologies used in the project, all of which fall under the umbrella of open-source software. These technologies can be categorized into three main distinct groups:

- FIWARE components originally developed by *Martel*[26], who now lead a community of Open Source developers and users;
- FIWARE components that Martel uses and contributes to as a community member, but is not the original contributor and development leader;
- Open-source components not originating within the FIWARE space, but considered a battle-tried and globally well-supported solution.

The set of services that belong to the FIWARE framework cover two main areas:

- **Context Management**, handling real-time and historical context, which includes the Orion CB and time series APIs;
- **IoT Management**, handling IoT devices and IoT data transportat.

The CB API in Orchestra Cities serves as a data bus that facilitates the management of contextual data. It allows for the seamless exchange and handling of contextual information within the platform.

To store the historical progression of contextual data, Orchestra Cities utilizes the time series API, which is built upon QuantumLeap. QuantumLeap, initially developed by Martel and now officially certified as a FIWARE Generic Enabler (GE), is responsible for storing and managing historical data. It leverages a time-series database, which associates the data with a specific time index and location. This makes possible to users retrieve and query historical data using dedicated APIs within the platform.

The IoT management APIs in the architecture schema encompass various components such as agents that support different protocols and are responsible for transforming incoming data into relevant contextual information. The IoT Agent Manager serves as a centralized configuration hub that allows users to manage and configure multiple agents from a single point of control.

For data transport, the exchange of data relies on the Advanced Message Queuing Protocol (AMQP) and Message Queuing Telemetry Transport (MQTT) protocols by utilizing the MQTT/AMQP broker. This ensures a reliable and efficient communication between all the devices and the platform.

Additionally to this, Orchestra Cities provides integration for Long Range Wide Area Network (LoRaWAN) devices through the LoRaWAN application server. This integration enables seamless connectivity between these devices and the platform by utilizing the specific LoRaWAN IoT agent designed for this purpose.

The open-source components that have been specifically selected from outside the FIWARE core are labelled above in figure 3.11 and include:

[26] An european innovation company that specializes in providing research and innovation services in various domains, including smart cities, digital transformation and ICT.

- **Keycloak**, for identity and access management;
- **Gravitee**, as the API gateway for the northbound interface;
- **Grafana**, for interactive context visualization dashboards.

Through years of experience, Orchestra Cities has gained valuable knowledge about the highly diverse landscape of IoT sensors. These sensors vary greatly in terms of integration capabilities, often requiring reliance on the manufacturer's proprietary platforms. This has emphasized the significance of having integration and configuration tools that are flexible and user-friendly. Furthermore, the development team has recognized the importance of a scalable and fault-tolerant architecture. This is more than crucial to ensure the platform can handle sustained high levels of activity and large volumes of data over extended periods of time without compromising performance or reliability. These insights have informed the development process, emphasizing the need for robust and adaptable solutions in the face of evolving IoT challenges.

Solution

The Orchestra Cities platform presents itself as an open-source, cloud-native, enterprise-strength solution that leverages and extends the FIWARE component baseline by adding, among other things:

- **Multi-tenancy**, to support groups of different cities and towns working together for their common benefit;
- **Advanced service orchestration and infrastructure-as-code**, using Kubernetes runtime and toolchain;
- **Data model standardisation and data transformation** with geographical and time-series information to build real-time historical records.

The developed platform was tested with several cities, including Antwerp, Helsinki, Mexico City and Wolfsburg before the current *EKZ*[27] adoption in the canton of Zurich.

The platform serves as a practical demonstration of harnessing the potential of IoT for the sustainable digital transformation of any society and its economy. It enables the measurement and achievement of a more environmentally friendly and sustainable world, aligning with not only the United Nations (UN) *Sustainable Development Goals*[28] but also other international agreements.

[27] Stands for Elektrizitätswerke des Kantons Zürich and it is a swiss electric utility company based in the canton of Zurich, responsible for the generation, transmission and distribution of electricity in the whole region.

[28] Also known as the Global Goals, is a set of 17 interconnected objectives established by the United Nations in 2015 that aim for fostering prosperity, protecting the planet and ensuring well-being for all, guiding efforts towards sustainable development by the year 2030.

At the core of Orchestra Cities are guiding principles that foster collaboration among different urban environments. The platform provides a space where these entities can come together and share relevant portions of their resources. By operating within a cloud-native and multi-tenant environment, each city has the autonomy to decide which data to keep restricted to their own tenancy and which one to exchange with neighboring cities. This allows for flexible and tailored data sharing arrangements based on the specific needs and preferences of each city.

Furthermore, Orchestra Cities supports the integration of external data sources alongside sensor data. By combining and integrating these diverse data sources, the system offers valuable insights and information to various stakeholders within the city. This integration enables a comprehensive understanding of the urban environment, supporting evidence-based decision-making and empowering city stakeholders to take proactive measures towards a more sustainable development. The following figure 3.12 provides a better summary of the solution environment.

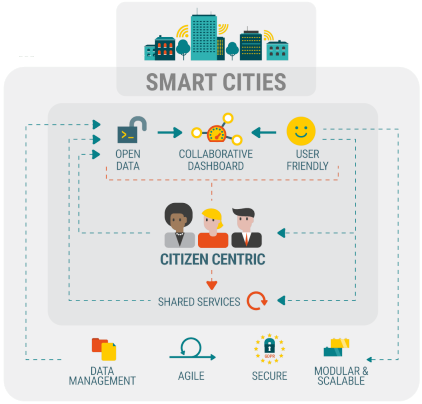


Figure 3.12: The conceptual environment behind Orchestra Cities

The key concept of openness is articulated in three major dimensions:

- **Open APIs**, where all platform features are made available to third parties through suitable gateways, based on open and well documented concepts and data models as the ones from FIWARE;
- **Open data**, as it is possible to combine data streams from deployed sensors with third-party data sources. Likewise, time series and other data generated by the platform can be published to integrators and consumers as open data if desired;
- **Open standards**, throughout the software stack, the superior quality and transparency of open standards support high performance levels while not limiting the future option of customers and new third parties joining the platform ecosystem.

The development approach on Orchestra Cities is built upon the foundation of open-source software and agile software development methodologies. Martel takes a lead role in developing

specific open-source systems and components within the FIWARE ecosystem. They actively contribute to the enhancement and evolution of various other software projects in parallel.

Orchestra Cities provides users with a comprehensive set of integrated features that encompass the overall functionality of the platform, regardless of specific scenarios or use cases. One key aspect is the ability to connect different types of IoT sensors to the platform, enabling the ingestion of relevant data streams. The platform management tools facilitate the mapping of individual sensors or groups of sensors, and this allows the configuration of common settings and the transformation of sensor-specific data formats into the NGSI standard shared by FIWARE components.

To enable more advanced and dynamic integration capabilities, the platform offers graphical low-code tools. These tools empower administrative users to define workflows for data extraction and transformation without requiring additional software development. This enables the seamless integration of external sensors, systems and APIs, which provides a quick and efficient mean of incorporating diverse data sources into the system.

Secondly, for data visualization, the platform provides an array of real-time dashboards covering data aggregation, drill-down and navigation with a variety of pre-built views to suit the most common use case scenarios, as represented below in figure 3.13. The transparent geographical and temporal data augmentation results in a great ease of use and definition of complex widgets such as interactive maps, multi-resolution time series and heat maps.

Figure 3.13: EKZ real-time dashboard

Last, but definitely not least, the platform and application deployment follows the most modern approaches for cloud-native software orchestration. Multi-tenancy and access management combine into a flexible and secure data sharing environment that can be run on the infrastructure from several major cloud service providers while enabling elastic scalability as well as effective and safe data and cost sharing across tenants [44].

Added value

One of the key differentiating factors of Orchestra Cities is its highly efficient and effective approach to achieve openness and sustainability, with a particular focus on supporting multiple cities on just a single platform. This approach brings several advantages, including cost-effectiveness, scalability and modularity, making it an important aspect of the partnership with EKZ.

As a trusted energy provider, EKZ, plays a crucial role as a system integrator and solution provider for individual swiss urban places as well as clusters of municipalities. This positioning aligns well with the swiss market and enables the extension of IoT and smart-city benefits to a much broader and diverse population beyond just the residents of major cities. By leveraging the Orchestra Cities platform, EKZ can bring the advantages of IoT and smart-city solutions to a wider range of communities which include both urban and rural areas.

This approach fosters greater inclusivity and ensures that the benefits of digital transformation and smart-city technologies reach a larger segment of the population. By supporting multiple cities and collaborating with EKZ, Orchestra Cities aims to create a more sustainable and socially impactful environment, trying to empower municipalities and clusters to leverage more IoT solutions for their unique needs and contribute to the overall advancement of smart-city initiatives across all Switzerland.

With its flexible integration capabilities and elastically scalable cloud-native deployment, Orchestra Cities allowed continuous platform execution for a period of two years, during which, data from more than a hundred different sensors was collected and analysed without impacting on the system execution performance. The gathered results and insights allow EKZ to sharpen their offer to municipalities in terms of available sensor sets and dimensioning of the cloud-based service provisioning, and overall cost profile and business model shaping.

Some key outcomes arising from the adoption of Orchestra Cities are:

- Supports migration from vertical data silos to unified data spaces over the city;
- Creates a collaborative space where different cities can share data and services, while retaining control of their own data;
- Enables flexible service provisioning, where each city can acquire just the needed services and resource quotas;
- Reduces ownership costs by sharing the platform among different cities;
- Leverages open standards and open source, thus building on the work of a large European and global community;
- Allows citizens and businesses to take part in the city services co-creation process.

Martel has been a member of the FIWARE Foundation from its inception and immediately recognised the multiple ways that it can be a critical enabler for advanced IoT and smart-city

services. A first very important aspect relates to interoperability and standards, as the Orchestra Cities relies on the open-source framework assets that have also been recognized. Another clear added value is the integrated governance of the several open-source projects that deliver FIWARE approved software components, thus ensuring reliable and consistent ecosystems for platforms and applications.

From a technological standpoint, the platform offers significant value to its users by leveraging the capabilities of FIWARE which by itself is a robust and flexible platform. The data models are well-defined and provide a solid foundation while remaining open and adaptable to facilitate the integration of diverse sensors and sources of data. This allows Orchestra Cities to quickly and seamlessly incorporate various types of data into the platform.

A critical component of the solution is its collection of IoT Agents, which support multiple protocols and data formats. These agents play a vital role in connecting and interacting with IoT devices, enabling seamless communication and data exchange between them. This versatility ensures that Orchestra Cities can accommodate a wide range of devices and technologies, enhancing its overall functionality and effectiveness.

The openness and inclusivity that this implementation brings to cities and citizens stem from the use of FIWARE's open-source components. These components not only contribute to the platform's interoperability but also enable its reliability, elastic scalability and efficient distribution of services. This service-based architecture, incorporating publish/subscribe and REST/JSON messaging, forms the foundation for a loosely coupled system and allows for dynamic deployment in cloud-native environments, providing the required flexibility for alternative implementations beyond the FIWARE baseline, such as for the API gateway.

These technological choices made by Orchestra Cities, based on the capabilities of FIWARE, offer key benefits to its end-users. The platform can efficiently handle the integration of diverse sensors and data sources, while also ensuring reliable and scalable performance. Moreover, the flexibility in technology selection and deployment options provides customers with choices that align with their specific requirements and preferences, enhancing their overall experience with the platform.

Use cases

The two main success stories behind Orchestra Cities are EKZ and Stadtwerke Wolfsburg AG. EKZ, in collaboration with Martel Innovate, has implemented a comprehensive deployment of various sensors from different manufacturers as part of their Smart-City Lab initiative. These sensors cover a range of use cases such as traffic management, waste management and environmental monitoring. To support and deliver the required smart-city services associated with these sensors, EKZ has chosen the Orchestra Cities platform [12].

Through this partnership, EKZ is able to provide an integrated solution to the municipalities it serves. With Orchestra Cities, each municipality can customize their chosen scenarios and

adjust the service levels and data loads accordingly. This flexibility allows for the implementation of a diverse range of smart-city initiatives and ensures that the platform can support different municipalities with varying needs and capacities.

Overall, through this collaboration, EKZ is able to deliver an integrated solution that combines sensor data, shared information and open data sources, empowering municipalities to make informed decisions and provide more efficient and tailored services to their communities.

This covers several scenarios and includes important climate-relevant aspects:

- Data stream from air quality sensors, measuring ozone, SO_2 and particulate matter of different sizes;
- Integration with open-data stream for weather measurements and forecast;
- Pervasive time series and geo-tagging on data streams, that creates high-quality historical information bases;
- Analytics and dashboards like the heat map of the combined air quality for area and time.

In one of the adjacent countries of the swiss land, there is Stadtwerke Wolfsburg AG and its daughter companies who are the infrastructure service provider for Wolfsburg and the surrounding region in Germany. This Wolfburg's SCDT combines strong and reliable data collection, processing and visualization flux with top class and accessible services for the citizens. A recurring need is the integration of sensor data with geographical information systems and intelligent route planning. This is where Orchestra Cities shows up giving resources up on multiple scenarios such as providing citizens with a smart advisory system for domestic waste management. Users can plan the most effective route for waste disposal, considering container type, fill level and user destination. A further smart-route planning scenario manages electrical vehicles and their charging stations, with their occupancy and supported vehicles.

3.2.4 Snifferbike

By the year 2030, over 80% of the population in the Netherlands is projected to reside in urban areas [9, 16]. Given this trend, it becomes crucial to focus on developing and maintaining a healthy living environment that promotes the well-being and health of the dutch people. A healthy urban living environment not only addresses the health concerns of individuals but also aligns with important societal issues like the energy transition and a sustainable and secure food supply chain.

The state of humans health is significantly influenced by various environmental factors. These factors include where people live and work, dietary choices, the level of exposure to air pollution, social interactions and even the lifestyle decisions people make. It is estimated that around

70% of chronic diseases, such as cardiovascular issues and cancer, can be attributed to these environmental factors [8].

To gain a better understanding of these environmental influences, it is crucial to rely on data-driven knowledge based on evidence. This necessitates adopting new collaborative approaches among governments, businesses, academic institutions and also residents. Together, they can work towards developing practical and scientifically grounded products and services that positively impact the public health.

The main goal of this project is to encourage collaborations and utilize a diverse range of disciplines to address health issues in urban areas. By utilizing data and insights based on evidence, the aim is to make informed decisions and develop practical solutions that improve the well-being of individuals living in cities.

One particular aspect that significantly affects the health of urban dwellers is the combination of mobility and air pollution. Air pollution poses a significant and escalating problem that harms both the environment and human health over time. Currently, air quality is primarily measured through stationary monitoring stations, which provide limited information on the presence of harmful pollutants in the surroundings.

In the Netherlands, cycling is one of the most popular mode of transportation, however, there is a lack of valuable and usable data specifically focused on cycling compared to the abundance of data available on car traffic. As a result, cycling is often overlooked in mobility policies. Nevertheless, it is known that bicycles can be an appealing alternative for many individuals, offering personal benefits. The increasing attention given to cycling, particularly during the COVID-19 pandemic the humanity faced off, can contribute to addressing various societal issues beyond just mobility, including air quality, public health, participation and overall well-being. However, the emergence of electric bicycles, higher speeds, bicycle delivery services and also the use of mobile technology while cycling present new challenges for the existing infrastructure and safety measures.

The province of Utrecht aims to become a front-runner in terms of cycling knowledge, data and tools. Their objective is to foster the development of a positive cycling culture throughout all the Netherlands. In order to achieve this goal, Utrecht initiated the Snifferbike [16, 33] experiment. By combining innovative technology, open data and active user engagement, this experiment promotes an healthy behavior, helps shift transportation patterns and contributes a lot to a healthier living environment.

Architecture

Snifferbike is a joint solution by *SODAQ*[29], *Civity*[30] and the *RIVM*[31]. SODAQ is responsible for the design and construction of the physical IoT device, while Civity is responsible for the whole process of collecting, storing and provisioning context data.

Snifferbike sensors measure air quality and GPS-location every 10 seconds. These sensors have a SIM card installed and transmit their data via *LTE-M*[32] every two minutes from bike sensors to the FIWARE platform.

The diagram provided below in figure 3.14 illustrates the various stages involved in transferring data from the sensor to the dashboard. SODAQ, the entity responsible for the mobile sensors, collaborates with Civity, a provider of data infrastructure and dashboard solutions, to facilitate this process.

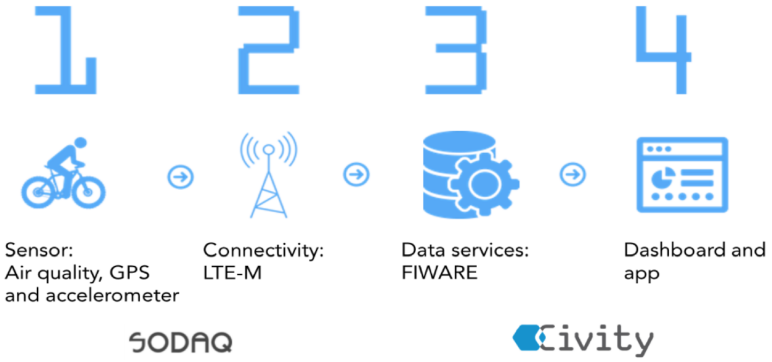


Figure 3.14: Snifferbike process flow diagram

The core element of the Civity platform, as in any FIWARE-based system is the CB. It serves as the central component responsible for aggregating data from different devices, specifically the Snifferbike sensors. The CB organizes and contextualizes this data to generate meaningful information and operates based on the globally recognized NGSIv2 API specifications.

Another FIWARE GE that is used in the Snifferbike solution, is Cygnus for managing the historical context data, which is connected to *CKAN*[33] to make the open data available for consultation and proper reuse.

Prior to being consolidated by the CB, the context data collected by sensors undergoes a

[29]Stands for Solar Powered Data Acquisition and it is a company that designs and manufactures open-source hardware and software solutions for IoT and sustainable development.

[30]A consulting and technology company that specializes in providing solutions for smart cities and urban development.

[31]The Dutch National Institute for Public Health and the Environment.

[32]Stands for Long-Term Evolution for Machines and it is a low-power wide area network technology designed to enable efficient and cost-effective communication for IoT devices.

[33]An open data management platform, and also part of the FIWARE reference architecture, for the publication, management and consumption of open data.

transformation process using a custom NGSI connector developed by the Civity team. This connector adapts data to the most appropriate data model. Once transformed, the CB receives the data, verifies its values and disseminates updated information to all the subscribed parties. In addition to being processed by the CB, the Snifferbike data is also transmitted to RIVM where the data is compared against readings from official static monitoring stations and calibrated accordingly. This ensures the accuracy and reliability of the Snifferbike data in relation to established measurement standards. This architecture schema can be better analyzed with the below figure 3.15.

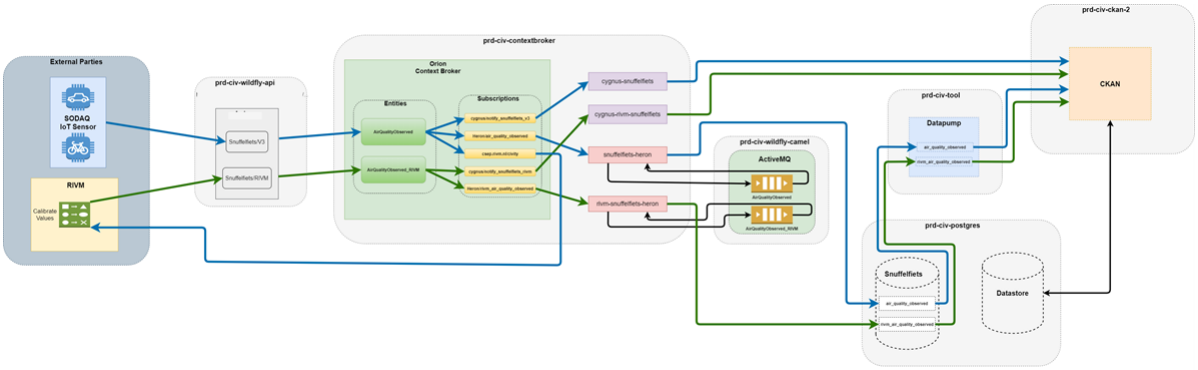


Figure 3.15: Snifferbike adopted architecture

Solution

The Snifferbike project was initiated in 2018 through a small-scale pilot. During this pilot phase, cyclists were equipped with sensors attached to their bikes, which gathers data regarding air pollution levels and cycle routes. These sensors collected anonymous data on cyclists' movements to identify patterns and assess areas where cycling infrastructure could be improved based on traffic flow.

The pilot proved to be highly successful, involving approximately 60 volunteers from the municipality of Zeist located in Utrecht. Encouraged by positive results, the project expanded in the summer of 2019, incorporating nearly 700 newly developed sensors. The device evolution process is described in figure 3.16. Notably, the provinces of Gelderland and cities including Sittard-Geleen, Zwolle, Den Bosch and Eersel also joined the initiative. All participants were volunteers who actively contributed to the project's success.

During the duration of 12 months, from the summer of 2019 to the next year's summer, the committed volunteers using the Snifferbike device generated an impressive dataset consisting of more than 35 million data points. Apart from measuring particulate matters, the sensors collected various other information including GPS coordinates, temperature, air pressure, humidity and detected road irregularities using the accelerometer feature.

Figure 3.16: Snifferbike sensor device evolution

In order to empower citizens, a mobile application was developed, enabling cyclists to track air quality and make informed decisions about choosing healthier routes. In parallel, a management dashboard was created to offer policymakers and environmental experts but a local research agencies vital data on the present environmental conditions. This comprehensive information plays a pivotal role in addressing environmental and mobility challenges, as well as in shaping a healthier urban environment accessible to all individuals.

Added value

At the present, there are nearly 700 Snifferbike sensors actively being used. These sensors have the flexibility to be shared among users, which means that the number of participants may go beyond the actual count of sensors. A community platform is established to ensure that all members involved in the project remain well-informed and connected.

Within the user community, there is a diverse range of individuals, including elderly people, students, wheelchair users, couriers and more. Each user group may derive distinct advantages and benefits from their involvement in the Snifferbike project. The project aims to cater to the specific needs and requirements of these different user categories, which fosters inclusivity and enhances their overall experience.

The benefits of using these cycling-oriented sensors are:

- Empowers cyclists to choose healthier routes;
- Helps individuals suffering from lung diseases to avoid certain levels of air pollution, as a well-known trigger;
- Supports local and provincial policy makers and urban planners in making educated decisions based on new insights;
- Allows research agencies and national monitoring institutions to collecting additional data and better understand air pollution;
- Supports climate neutrality.

The primary aim is to expand the deployment of Snifferbikes and leverage the collected data to develop innovative solutions. In 2020, numerous Dutch government agencies demonstrated their commitment to improving the air quality by signing the Clean Air agreement. This agreement strives to bring about a lasting enhancement of air quality throughout the Netherlands, with active involvement from all the stakeholders. A key motivation for this agreement is to achieve a 50% improvement in public health by 2030, compared to the faced in 2016, by addressing domestic sources of air pollution. By achieving this goal, the dutch population may enjoy longer and healthier lives, ultimately enhancing their overall well-being.

Use cases

The Snifferbike project plays a crucial role in providing valuable insights and supporting cities in efficiently achieving their objectives while minimizing capital investments. To further enhance the capabilities of this project, ongoing developments are taking place through the establishment of the *Healthy Urban Living Data and Knowledge Hub*[34]. By leveraging the data collected through this initiative, monitoring efforts can be enhanced, leading to more informed decision-making processes.

Globe[35], has created educational materials tailored for secondary schools. These materials focus on the project, creating a platform for increased engagement and awareness among young students, nurturing a deeper understanding of nature and environmental issues.

Until August 2020, the project had generated an extensive dataset, comprising almost 35 million data points. Participants had collectively traveled nearly 500.000 kilometers and spent around 35.000 hours cycling. The COVID-19 pandemic and subsequent lockdown measures in March 2020 resulted in a notable decline in car, bus and train commuting, while there was a rise in longer recreational bike rides [6].

What initially started as an experimental initiative has clearly demonstrated its value over the years. It has attracted a growing community of participants and sparked numerous innovative ideas for expanding the solution, incorporating additional data sources, catering to diverse user groups, and promoting inventive solutions. The project has highlighted the significance of an open urban platform, underscoring the success and sustainability that arise from embracing openness, collaboration, and standardization.

Civity, an organization that has been utilizing FIWARE components since 2016, played a pivotal role in establishing the Dutch FIWARE Lab in 2017 [25]. This lab aims to engage a broader audience, including urban planners, developers, researchers and civil servants, in collective efforts to drive progress in the city development.

[34]A platform focused on collecting, analyzing and sharing data and knowledge related to promoting healthy urban living.

[35]An organization that partners with scientists to provide nature and environmental lessons in schools.

3.2.5 SmartData Wien

In the complex realm of smart-cities, one of the major challenges faced by projects is the aggregation of relevant context data from diverse sources, each one of them with its own unique approach and format. Often, this data is gathered in a non-interoperable and non-standardized manner, which makes it difficult to consolidate and harmonize. Such context data can originate from legacy systems, individual users through mobile apps, IoT devices, static datasets and APIs, which lack a standardized approach.

The process of converting this diverse context data into harmonized data models and meaningful information can be time-consuming and may take over a month or more for the engineering team to map and understand the value of each context provided by the different suppliers. However, the effort involved is much more complex as it requires a non-intrusive approach to integrate contextual data with existing and also emerging systems, minimizing the need for special adaptations. This allows data providers to reduce the impact on their architectures and achieve cost savings in creating and maintaining context information models.

The SmartData.Wien [18] initiative focuses on promoting open data principles and fostering data-driven innovation within the city of Vienna, capital of Austria. It involves the collection, integration and provision of a wide range of data from different domains such as transportation, environment, energy and demographics [36]. The project aims to enable citizens, businesses and public entities to access and leverage this data for research, development and for the creation of innovative applications.

Architecture

When working with API responses, it is very common to have entities that are associated with other entities derived from them, to what is often called relationships. According to the *FIWARE Data Modelling Guidelines*[36] for linked data, when an entity's attribute is used as a reference to other entities, it is recommended to prefix the attribute name with `ref` followed by the name of the linked entity type and assigning the attribute type to `Relationship` which again is purely a naming convention with no real semantic weight. Using this convention helps establish the proper connection between entities [39].

The final data model, represented below in figure 3.17, may consist of multiple API responses, depending on the specific context provider and the way data is published. By following these guidelines and creating the appropriate links between entities within the system, a comprehensive and interconnected data model can be constructed.

[36] A published set of recommendations and best practices for designing data models within the FIWARE ecosystem.

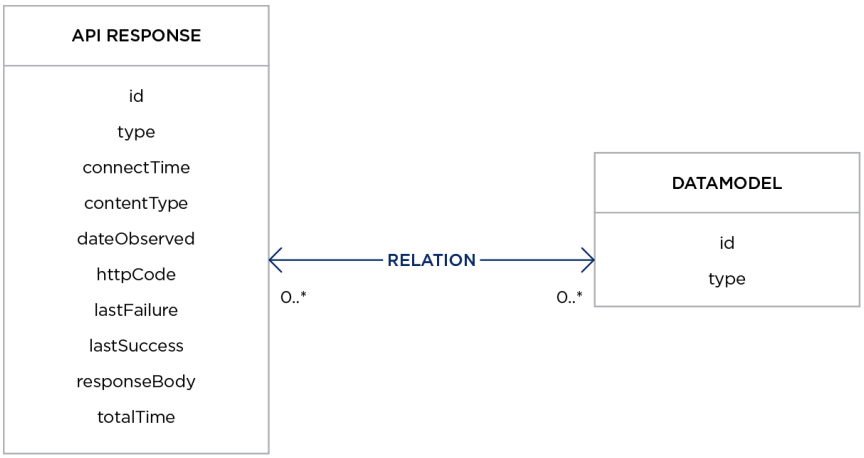


Figure 3.17: SmartData Wien adopted data model

The method proposed involves utilizing a consumer, which is a service that sends HTTP requests to the API and captures data from their defined response data models. Each request, with its specific properties, parameters and query strings, represents a unique request identified by a singular identification. This request is then transformed into an NGSI data model to generate an entity object, which is used to either create a new entity or update an existing one in the responsible CB. The consumer can also share this information with other entities based on the objectives of the project.

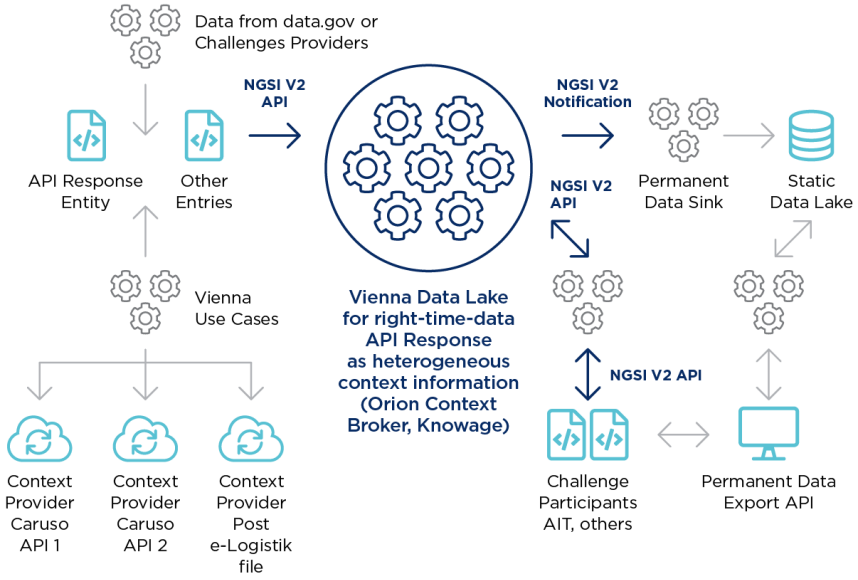


Figure 3.18: SmartData Wien representation of information flow

As the above information flow 3.18 states, Vienna has opted to make use of the latest NGSIv2 data model for its personal API, which offers enhanced support for the creation of meaningful context for the data which makes it easier to make associations between entities. To ensure this data security, a secure layer is implemented to restrict access to authorized stakeholders only.

After more than a year of adaptation and customization, the project now handles a wide range of data requests and leverages innovative visualization capabilities through *KnowAge*[37]

By adopting this communication model, the capital of Austria can leverage additional advantages provided by the CB, such as the NGSIv2 notifications for creating historical data storage or developing real-time applications like dashboards and monitoring panels. This approach ensures that solution remains future-proof and incorporates the recommended topology and design features.

Solution

To overcome these challenges, the platform makes use of the Orion CB, which serves as a standardized and scalable solution for managing contextual information. The CB acts as a centric hub that facilitates the exchange of data between various services. For instance, it enables notifications to be sent automatically to a historical database, referred to as the "data lake", whenever any changes to the context occur.

The Smarter Together project, funded by the EC, aims to foster innovation and enhance urban mobility through the implementation of cutting-edge solutions for smart-cities. This initiative has yielded notable outcomes, including:

- Sharing of open urban data;
- Increased citizen engagement;
- Improved knowledge management;
- Enhanced monitoring of data.

Data providers gather information from their sensors and other sources, which is then transmitted to the FIWARE platform using secure protocols such as *HTTPS*[38] and predefined APIs, represented in the diagram 3.19. To ensure data privacy and access control, a security layer is implemented, allowing only authorized individuals or entities to view and interact with the data.

[37]The FIWARE's complementary open-source business intelligence component for modern business analytics over traditional sources and big data systems.

[38]Stands for Hypertext Transfer Protocol Secure and it is the secure version of HTTP.

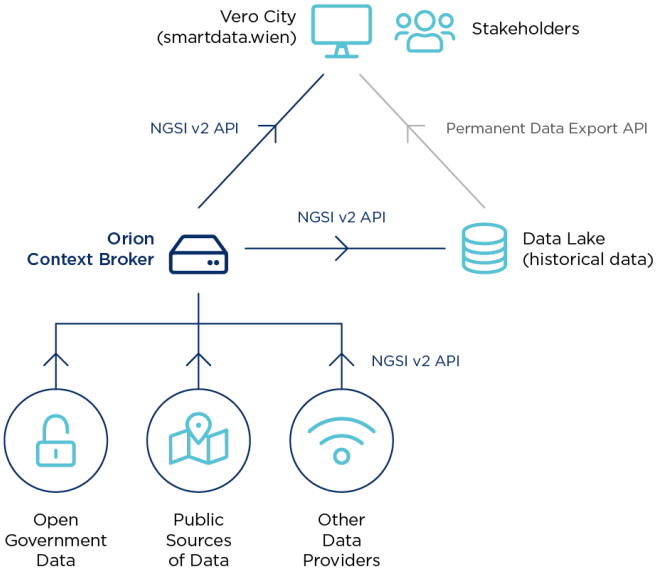


Figure 3.19: The diagram of data exchange inside SmartData Wien

Added value

The platform offers web services that benefit both citizens and city officials, so they can streamline their daily activities. It provides easily accessible and visualized information, eliminating the need to work with raw data. This process enhances transparency in monitoring and benchmarking across various policy areas, while encouraging citizen engagement in shaping any city's future.

From a software developer's point of view, the CB offers valuable key aspects that save a lot of time and resources. Its APIs allow seamless integration with existing and future systems, facilitate data exchange across different sectors and simplify the creation and maintenance of contextual data models. As an open-source solution, it eliminates the risk of vendor lock-in and fostering innovation. Additionally, FIWARE provides a comprehensive suite of complementary solutions for data modeling, visualization, security, user management and more, empowering developers to create better and more complex smart-city solutions.

- **Open data platform**: it is established an open data platform capable of providing access to a diverse range of datasets related to the city of Vienna. These datasets are made available to the public in a machine-readable and easily accessible format, promoting transparency, innovation and collaboration.
- **Data integration**: the whole project focuses on integrating data from multiple sources and domains, ensuring data consistency, quality and interoperability. By consolidating and harmonizing these datasets, SmartData Wien aims to provide a comprehensive view of the city's data landscape, which enables a more accurate analysis and way better informed insights.
- **Collaboration and co-creation**: it encourages collaboration among various stakeholders,

which include government agencies, research institutions, businesses and regular citizens. It facilitates the exchange of ideas, expertise and resources to foster co-creation and innovation in the development of smart-city solutions.

- **Smart-City applications**: the availability of open data serves as a foundation for the development of smart-city applications. The platform enables developers to make use of the data to create innovative solutions that address urban challenges and enhance the general quality of life in Vienna. These applications can span various domains, such as transportation optimization, environmental monitoring, energy efficiency and citizen engagement.
- **Data privacy and security**: data privacy and security are prioritized. The project ensures that appropriate measures are in place to protect sensitive data and comply with relevant privacy regulations. Data access controls and anonymization techniques may be implemented to safeguard individual privacy while enabling data analysis and utilization.

Use cases

As part of this project, three specific use cases have been designed and developed:

- **Facility management**: the main goal while managing facilities is the collection of energy, heating and water consumption data from schools before and after the proposed renovation program. Logically the data sources are energy, heating and water meters and the visualization is their consumption over time.
- **Monitoring mobility**: the project can be used in urban mobility analysis regarding changes in the choice of transportation modes. Data is given by ride sharing apps and e-vehicles across the city and the overall platform performance can be visualized through previously set KPIs and energy consumption evolution, as represented below in figure 3.20.
- **Harmonization of facility and energy information**: by giving these different data sources unique identifiers, controlling and labeling these sources regarding facility and energy information. This can be visualized using consolidated information from different sources in the geological data system.

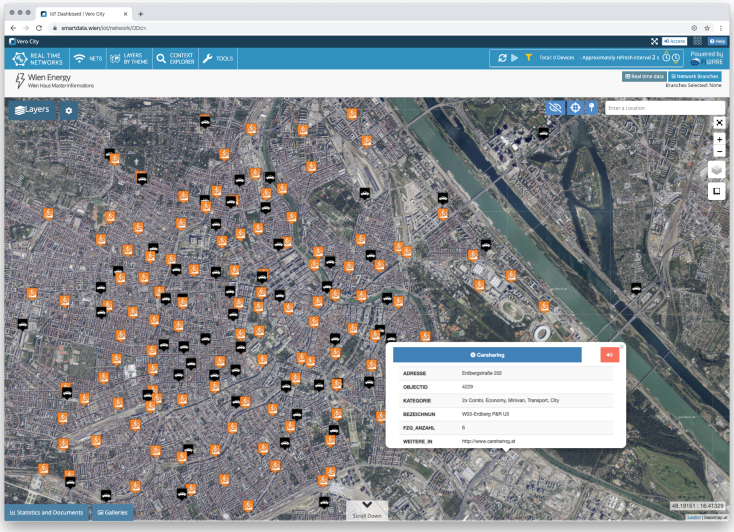


Figure 3.20: SmartData Wien data visualization for car/bike sharing

3.3 Summary

Through a thorough examination of successful past initiatives, valuable insights can be gleaned that can significantly enhance the design and development process of this intended solution.

Regarding the solutions not leveraging the FIWARE framework in their foundations,

DUET, Digital Urban European Twins

- Focuses on creating a unified framework for data fusion in the context of smart-cities;
- Incorporates IoT devices and data sources to enhance the collection and analysis of real-time data for urban modeling and decision-making;
- Uses the T-Cell framework, which acts as a container for models, data and simulations, which enables a dynamic interaction and provides valuable insights for smart-city decision-makers;
- Adopts a quadruple helix approach, involving collaboration between policy domains, academia, industry and citizens, which fosters innovation and address urban challenges in a more effectively way:
- The project architecture decouples components to maximize interchangeability and integrates Apache Kafka for efficient data exchange;
- Leverages Docker containers for managing individual models and incorporates various modules for data administration, event handling and general security;

- Emphasizes data normalization and semantic interoperability;
- Enables real-time monitoring of the urban environment and supports scenario analysis by integrating various models, simulations and data sources to provide decision-makers with valuable insights and evidence-based recommendations.

Herrenberg Digital Twin

- Focuses on using digital technologies to support a smarter and more sustainable local Herrenberg governance;
- Involves the development of a 3D city model using geographic data, incorporating qualitative and quantitative data, simulations and mapping;
- Embedded in a collaborative visualization and simulation environment called COVISE, which enables and fosters citizen participation and also offers flexibility and scalability for future enhancements;
- The proposed architecture includes modules for data integration, street network modeling using space syntax, urban mobility simulation using SUMO and pollution simulation based on sensor data;
- Utilizes a mobile application called Reallabor Tracker that collects empirical data from citizens, including their movement routes, assessments of public spaces and multimedia documentation;
- Integrates environmental sensors deployed in the urban environment which allows for the analysis of air quality and the distribution of gas emissions;
- Demonstrates the value of real-time visualization and simulation in analyzing traffic patterns, modal split changes and the impact of interventions in a VR environment;
- Enhances citizen participation, integrates diverse data sources, enables real-time visualization and simulation, addresses urban sustainability, and offers flexibility for future developments.

Going through the solutions counting on the FIWARE framework for their core functionality,

CO2-Mute

- Aims to support local governments in implementing sustainable mobility policies and promoting urban green infrastructure;
- Focuses on reducing carbon dioxide emissions and addressing the environmental impacts of urban mobility;

- Employs gamification and data analysis to encourage the use of public transport and bicycles, while also emphasizing the role of green spaces in mitigating pollution;
- Emphasizes the importance of data interoperability and makes use of the NGSI-LD standard to enable seamless access to heterogeneous data from various sources;
- Its architecture involves data collection and analysis during local mobility challenges, followed by the dissemination of impact and results;
- Data sources, such as traffic data from Tom-Tom and HERE APIs, air quality and noise sensors, and weather information are used to calculate the estimated impact of traffic reduction on air-pollutant concentrations;
- Ability to provide customizable solutions tailored to local contexts.

Snap4City

- Addresses the challenges faced by cities in meeting societal, environmental, and economic demands;
- Offers a flexible and dynamic architecture, integrating various data channels and enabling semantic queries;
- Includes a dashboard builder component with interactive visualization tools and supports real-time event-driven solutions;
- Seamlessly integrates with Node-RED for data processing and analysis, including ML and AI technologies;
- Supports the development of applications across multiple domains and complies with GDPR regulations;
- Ability to provide customizable solutions tailored to local contexts;
- Promotes the concept of living lab development and provides a supportive environment for developers;
- Successfully deployed in over 40 urban areas and offers added value in terms of improved performance and reduced costs knowing it can be installed on both private and public clouds, accommodating various scales of deployments;
- Used in various use cases such as mobility, environment, energy, strategic planning, city management and people flow analysis.

Orchestra Cities

- Leverages ICT solutions to optimize urban environments and overcome limitations such as limited space, outdated infrastructure, strained public services and environmental sustainability issues;

- Follows a micro-service architecture and is implemented as a collection of distributed services within a cloud infrastructure;
- Offers advanced service orchestration and infrastructure-as-code using Kubernetes runtime and toolchain;
- Incorporates open-source components such as Keycloak for identity and access management, Gravitee as the API gateway and Grafana for interactive context visualization dashboards;
- Promotes collaboration among different urban environments, supports data sharing, and enables the integration of external data sources;
- Offers features for connecting IoT sensors, data visualization through real-time dashboards and cloud-native deployment;
- Facilitates the migration from data silos to unified data spaces, enables flexible service provisioning and reduces ownership costs;
- Fosters citizen and business participation in the co-creation of city services.

Snifferbike

- Combines innovative technology, open data and active user engagement to address health issues in urban areas;
- Focuses on the combination of mobility and air pollution, recognizing that air quality significantly impacts the health of urban dwellers;
- Involves equipping cyclists with sensors attached to their bikes to measure air pollution levels and gather data on cycle routes;
- Data is collected, processed and disseminated through a FIWARE-based platform, which facilitates informed decision-making and the development of practical solutions;
- Demonstrated success through pilot phases and has expanded its participant base;
- Collected data has been used to empower cyclists in choosing healthier routes, support policymakers in making informed decisions, aid research agencies in understanding air pollution and contribute to climate neutrality;
- Added value by fostering inclusivity, supporting public health improvement goals, and providing insights for efficient urban planning;
- Facilitated collaborations and educational initiatives, emphasizing the importance of openness and collaboration in urban development.

SmartData Wien

- Promotes open data and data-driven innovation in smart cities;
- Addresses the challenge of aggregating diverse context data by using standardized approaches;
- Collects and integrates data from various domains and provides access for research and smart city applications;
- Uses a consumer service to capture data from API responses and create interconnected data models;
- Leverages the FIWARE CB for managing contextual information and enables automatic notifications to a historical database;
- Ensure data privacy and security through secure protocols and access controls;
- Added value by fostering inclusivity, supporting public health improvement goals, and providing insights for efficient urban planning;
- Facilitated collaborations and educational initiatives, emphasizing the importance of openness and collaboration in urban development.

Literature	**Motivations**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
DUET		✓	✓	✓			
Herrenberg Digital Twin		✓		✓			✓
CO2-Mute		✓	✓				
Snap4City	✓	✓	✓	✓			✓
Orchestra Cities		✓	✓	✓	✓		✓
Snifferbike			✓	✓	✓		✓
SmartData Wien		✓	✓	✓	✓		✓

Table 3.1: Literature motivation checklist

Legend	**Description**
1	Addresses the current challenges associated with autonomous vehicle deployment
2	Provides seamless integration and interoperability among diverse systems, devices and platforms within the smart-city ecosystem
3	Builds a robust and scalable architecture that can handle large volumes of generated context data within the smart-city
4	Enables real-time data ingestion processing to provide up-to-date and easily accessible insights
5	Systematizes an efficient process for persistent context update storage which enables historical and conditional context analytics
6	Designed for simulation and experimentation, i.e. a repetition system that permits the rerun of simulations based on time-specific context data snapshots
7	Develops an intuitive and user-friendly web interface for visualizing and interacting with the context data

After the comprehensive study over the state of the art, it is possible to state from the checklist 3.1 on the set of motivations and goals, defined by us, the above solutions were able to reach, that none of the reviewed projects explicitly addressed the need for a repetition system capable of enabling the rerun of simulations based on time-specific context data snapshots. This realization presents a unique opportunity and motivates the focus of this thesis on contributing to a pioneering contribution to the realm of SCDT. By developing such a system, there is the the potential to fill a crucial gap in the existing literature. This contribution not only fills an existing gap but also opens up new avenues for research and development, paving a way for enhanced simulation capabilities, improved urban planning strategies and more effective decision-making in the context of smarter cities.

Chapter 4 outlines the system's proposed architecture and discusses its design process, including all the challenges faced during this operation. The chapter delves into an in-depth analysis of the project's requirements and considerations, highlighting how the proposed architecture can address them and provide an improved solution to overcome the predefined challenges.

Chapter 4

Proposed Architecture

In this architecture proposal, we provide a comprehensive description of the core structure and design of the system developed as part of this project. The architecture serves as a fundamental framework that greatly influences the overall functionality and design of any system, so by establishing a robust foundation, it becomes possible to ensure the successful implementation of the proposed solution. Therefore, a meticulous and detailed explanation of the creative process behind the development scene is crucial in setting the stage for a thorough understanding of its significance and practical impact.

4.1 Design Process

In the process of conceptualizing and designing what would be the ideal architecture for system solution, a creative and abstract approach was employed to envision a result that fulfills the predefined specific goals. The architecture aims to serve a range of features, each contributing to the overall proposed functionality of the system. By revisiting the underlying motivations that gave rise to this thesis project, listed below, we can delve deeper into analyzing and defining the most suitable architectural design for the required necessities.

1. Address the current challenges associated with autonomous vehicle deployment;
2. Provide seamless integration and interoperability among diverse systems, devices and platforms within the smart-city ecosystem;
3. Build a robust and scalable architecture that can handle large volumes of generated context data within the smart-city;
4. Enable real-time data ingestion processing to provide up-to-date and easily accessible insights;
5. Systematize an efficient process for persistent context update storage which enables historical and conditional context analytics;

6. Design a repetition system that permits the rerun of simulations based on time-specific context data snapshots;

7. Develop an intuitive and user-friendly web interface for visualizing and interacting with the contextual data.

4.1.1 Integration and interoperability

In the context of integration and interoperability, the proposed architecture should aim to foster a seamless and cohesive environment by effectively accepting and integrating diverse systems, devices and platforms within the Smart-City Digital Twin (SCDT). This integration is crucial to enable smooth communication, data exchange and collaboration among the various components of the ecosystem.

To achieve this, the architecture must adopt standardized protocols and interfaces that facilitate interoperability between possibly different technologies and systems. It needs to establish a common language and communication framework, allowing diverse entities to interact and share data effortlessly. By promoting interoperability, the architecture can enable the collaboration of multiple stakeholders, including autonomous vehicles, urban infrastructure, sensor networks, Internet of Things (IoT) devices and data management systems. This way the flow of data is a lot easier which promotes collaboration between different entities and enhances the overall solution efficiency and effectiveness.

4.1.2 Robustness and scalability

The architecture places a strong emphasis on robustness and scalability to effectively handle the large volumes of context data possibly generated within the smart-city environment.

- **Robustness** refers to the architecture's ability to maintain its performance and functionality even under challenging or adverse conditions. In the context of a SCDT, this means that the architecture can handle unexpected events, fluctuations in data volume and potential failures of individual components without compromising the overall system's integrity and performance. It incorporates mechanisms such as fault tolerance, error handling and redundancy to ensure continuous operation and availability of data.

- **Scalability**, on the other hand, addresses the capacity to accommodate the growing demands and increasing data volumes within the smart-city ecosystem. As the volume of generated context data continues to expand, the architecture is designed to scale up its resources and capabilities to handle the increased load effectively. This scalability can be achieved through techniques such as horizontal or vertical scaling, distributed processing and efficient resource allocation [2].

To ensure these robustness and scalability key aspects, the architecture needs to leverage modern technologies and frameworks that support distributed processing and parallelization. It must adopt scalable data storage solutions, such as distributed databases, to handle the large volumes of both current context data and historical context updates efficiently. By prioritizing this, this architecture can effectively cope with the ever-growing data demands within any environment. It enables the seamless processing, storage and analysis of large volumes of context data, contributing to the overall solution's efficiency, reliability and sustainability.

4.1.3 Timely data ingestion processing

Real-time data processing is also a crucial aspect of this architecture, as it enables the ingestion and processing of data in near real-time, ensuring that the insights and information derived from the environment context are up-to-date and readily available.

In the context of a SCDT, where data is continuously generated from various sources, real-time data processing plays a vital role in capturing and analyzing this data in a timely manner. By processing it right after ingesting it, the architecture can provide immediate insights, enabling quick decision-making and response to dynamic conditions. Overall, this fast data processing empowers the system to harness the value of data as it is generated, providing a dynamic and responsive environment for decision-making, planning and operational optimization within the current context.

Being capable of properly storing context updates is a crucial functionality the architecture must focus on. This process enables historical and conditional context analytics which allows for deeper insights and analysis of the collected data [10]. In a setting like the one this thesis involves, various factors can contribute to the context dynamic change, such as traffic conditions, environmental data, infrastructure status and user behavior. These contextual updates are continuously generated and provide valuable information for understanding the state of entities and making informed decisions.

4.1.4 Persistent context update storage

The architecture must incorporate a systematic approach to store and manage these context updates. This involves designing an efficient storage mechanism isolated from the short-term storage that can handle the high volume and variety of historical updates generated by simulations running on the SCDT.

In addition to historical analysis, the architecture should enable conditional context analytics. This means that the stored context updates can be queried and analyzed based on specific conditions or criteria. Conditional analysis can provide valuable insights into the behavior of the running instance under different circumstances, which can help to understand the cause-effect relationships and optimize some decision-making processes.

4.1.5 Context repetition system

The repetition system is one of the most important components of the set of motivations, so the development skeleton should focus on facilitating the rerun of simulations based on time-specific context data snapshots. This feature allows for testing and validation of various scenarios in a controlled and reproducible manner.

In a smart-city environment, it is crucial to assess the impact of different factors, such as changes in infrastructure or traffic patterns on the overall system performance. Conducting real-world experiments to evaluate these scenarios can be costly, time-consuming and may disrupt the normal functioning of the city. Then, to overcome these physical challenges, the architecture must incorporate a repetition system that leverages time-specific context data snapshots. These snapshots are captures of the state of the smart-city instance at specific points in time. By using these snapshots, simulations can be rerun over the current context, past dates or even indexed simulation repetitions.

4.1.6 Context management interface

The architecture should accomodate the seamless integration of a Graphical User Interface (GUI) to enhance the general user experience by providing an intuitive and accessible way to visualize and interact with the contextual data. To address this challenge, the architecture must incorporate a layer serving as a visualization and interaction tool for the underlying management set of components. The interface should be designed with a focus on user-friendliness, ensuring that users, such as urban-planners, policymakers and researchers, can easily navigate and interact with system without requiring an extensive technical expertise.

Having a customized interface like so also fosters the collaboration and knowledge sharing among different stakeholders. It may provide features for data sharing, collaboration and annotation, which allows users to exchange information, discuss findings and collaborate on research projects initiatives.

4.2 Component Schema

The component schema is the result of many long sessions of brainstorming and a careful solution design process. It is built upon the FIWARE framework, which as reviewed previously, provides a comprehensive set of open-source components and standards for developing smart-city solutions. By leveraging this open-source ecosystem, the architecture offers a range of advantages and aligns with the underlying motivations of the project.

The integration and interoperability capabilities provided by the chosen framework play a central role in the architecture. It enables seamless integration among diverse systems, devices, and platforms, promoting interoperability and creating a cohesive environment. The

architecture adopts the standardized protocols and interfaces supported by FIWARE, facilitating communication and data exchange between different technologies and systems.

Since robustness and scalability are crucial aspects of the desired architecture, the utilization of ready-to-use generic enablers contribute to this achievement. FIWARE provides a scalable and distributed architecture, allowing the system to handle large volumes of context data generated within the SCDT. By leveraging modern technologies like horizontal scaling and distributed databases, the proposed component schema can effectively accommodate the growing demands and increasing data volumes while ensuring the system's robustness and scalability.

Real-time data processing, is also facilitated by this integration. The FIWARE ecosystem offers real-time data ingestion and processing capabilities, which enables the architecture to capture and analyze data in near real-time. This empowers the system to provide up-to-date insights, facilitating the desired quick decision-making and response to dynamic conditions.

The data management components of FIWARE also play a significant role in the architecture's design. It incorporates the Context Broker (CB), a core Generic Enabler (GE), which enables the most efficient and flexible context data management possible [10]. Additionally, the Next Generation Service Interface (NGSI) specification, which is a key part of the ecosystem, facilitates data abstraction and enhances the architecture's ability to handle and manage contextual data.

Cygnus also plays a crucial role in the proposed architecture, specifically in terms of historical persistency and the repetition system goal. Its inclusion addresses the need for efficient storage and retrieval of historical context updates [37], enabling deep analysis and the rerun of simulations based on time-specific context data snapshots.

One of the key functionalities of Cygnus is to bridge the CB and a long-term data sink [37], so by acting as an intermediary, Cygnus subscribes to contextual updates and captures them after being received from the CB, ensuring their proper formatting and persistence in the sink. This enables the main storage system to maintain a comprehensive record of all context changes over time.

The decision to construct the architecture using a foundation based on FIWARE brings numerous advantages. By utilizing the existing features and customizable components offered by the framework, not only does it streamline the development process, but also establishes a solid and effective solution architecture from the outset. This choice not only makes development significantly smoother but also ensures a robust and efficient foundation for the final solution.

The system architecture consists of several interdependent components that work together to enable an efficient data communication and storage. At the core of this architecture is the NGSI Service, which acts as a direct interface for the web application and also an actual abstraction layer for all the requests made to the CB via the NGSI API. As proposed, the NGSI Service is directly linked to the long-term data persistency which enables direct historical data retrieval using the API. To facilitate the long-term data retrieval and persistence, the long-term persistency unit is assigned to Cygnus as the main historical storage as Cygnus acts as the linking bridge

between the contextual updates from CB and the sink, while persisting these updates. This ensures that every change to any entity is formatted and properly persisted.

To further enhance short-term data persistency and availability, the CB is set to be directly linked to a NoSQL database as this integration ensures that recent context updates and current state of entities are readily available for a fast retrieval and analysis.

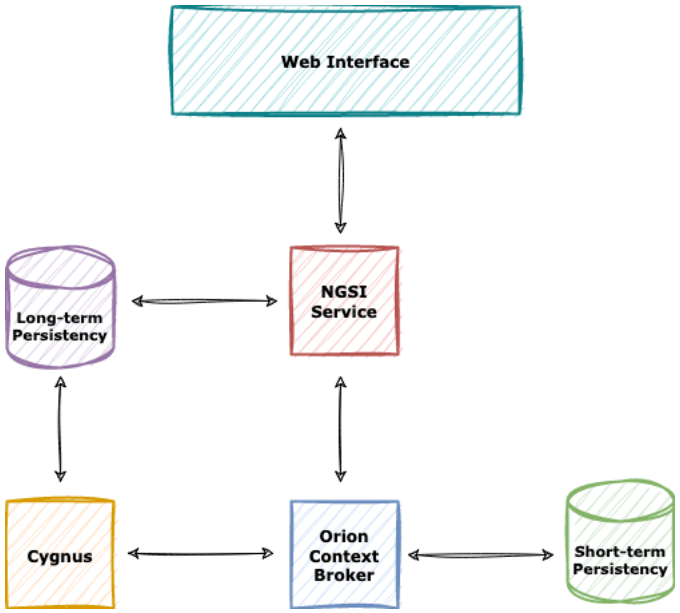


Figure 4.1: Proposed system architecture sketch

In summary, the system architecture combines the power of the NGSI Service, the short and long-term data sinks, Cygnus and the CB to create a robust and efficient data communication and storage system. The web interface makes use of the NGSI Service abstraction, which manages the data flow between the various components. The long-term data storage unit, Cygnus and the CB work in conjunction to store and retrieve data for long-term persistency, while the short-term data sink in conjunction with the CB provide short-term persistency for recent context updates. This way it is possible to reach a scalable, flexible and reliable solution for all the context management, visualization and simulation needs. This understanding is pivotal for the subsequent chapters where there are insights into the real solution's implementation, testing and evaluation details and the research that was built upon the framework here presented.

4.3 Context Management

4.3.1 Entity modelling

The NGSI is a standard data model and API specification developed by the FIWARE community for managing and exchanging data in the context of smart applications and IoT systems. One of

its key aspects is entity modelling, which provides a structured and standardized approach to representing real-world entities and their attributes in a digital form.

Since this project is developed based upon the FIWARE standards, it seemed natural to work and store all the context data following the proposed data model. In a NGSI data model like this, entities are the core building blocks representing physical objects, such as vehicles, persons, sensors, buildings or even abstract concepts like weather conditions or traffic congestion. Each entity is uniquely identified by its unique Uniform Resource Identifier (URI) and belongs to a particular entity type that defines its shared characteristics and behavior.

Inspired by the NGSI entity data model, the adopted structure for the entity instance is explained in this list of fields and exemplified in the following listing 4.1.

- **Identifier**, being an unique URI that follows the pattern `<type>:<name>:<idx>`
- **Type**, which represents the entity data type for a better categorization
- **Attributes**, the rest of the attributes hold a JSON structure with a specific `type` and `value`

```
1  {
2      "id": "vehicle:Motorbike:1"
3      "type": "Vehicle"
4      "speed": {"type": "Float", "value": 76.4},
5      "weight": {"type": "Float", "value": 193.2},
6      "color": {"type": "String", "value": "#EEEFFF"}
7  }
```

Listing 4.1: Example of the entity data model

Entities are this way modeled using a set of attributes that describe various aspects of them. Attributes can represent both static and dynamic properties, such as location, speed or any other relevant information regarding their actual context. These attributes can have different data types, including `string`, `number`, `boolean`, `date` or even more complex data structures. This design was thought to adhere to a flexible and extensible data model, which allowed the addition of new attributes or entity types without actually breaking any compatibility. This enables the representation of diverse and evolving data requirements in a fluid and dynamic smart-city. Lastly, the system global data model is shown in the following diagram 4.2.

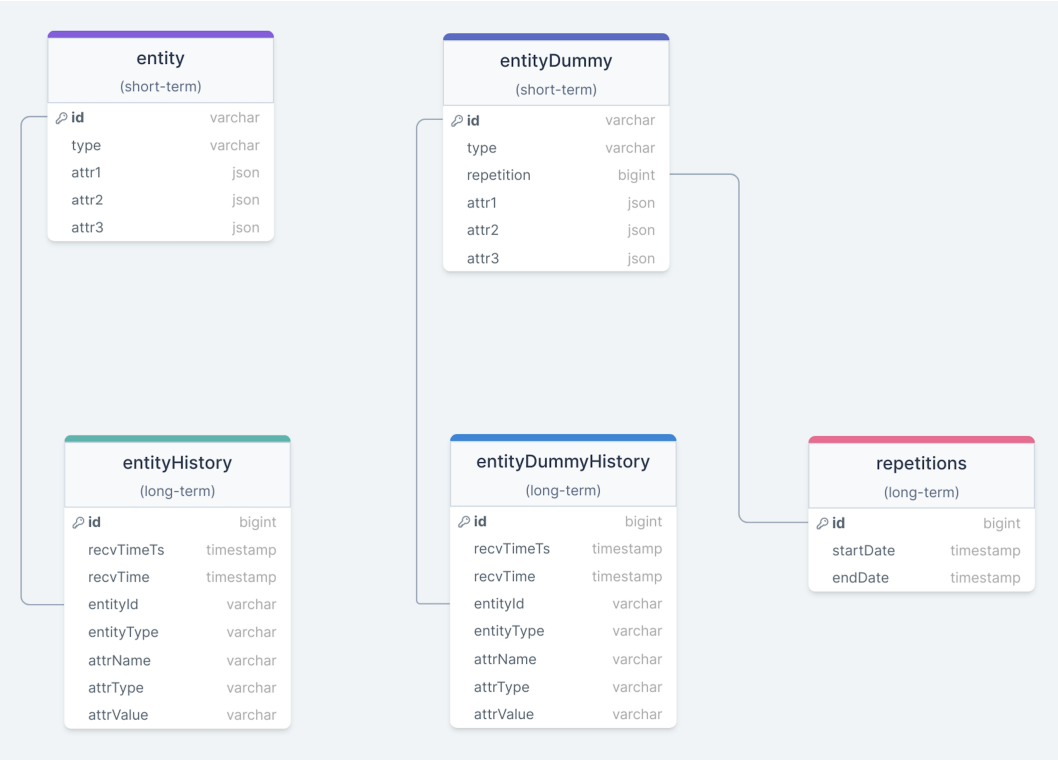


Figure 4.2: Entity data model

4.3.2 Short-term persistency

The CB is the key component of the FIWARE platform and it is responsible for managing and providing access to real-time context information. It supports the publish/subscribe model, where data producers publish their context information and consumers subscribe to specific types of data they are interested in.

In this architecture, the CB is configured to store the received context information directly into a NoSQL database based on data collections. These collections can be thought of as equivalent to tables in a relational database and each collection contains multiple documents, analogous to rows in a table. One positive aspect this approach offers by using a NoSQL database is its flexible schema design. These models provide a dynamic data storage, which means that each document in a collection can have different fields and structures. This flexibility is particularly useful in a dynamic environment like in a SCDT oriented to Autonomous Driving (AD) where the data produced by different sources and vehicles may have varying and evolving attributes over time.

As the CB receives new context information, it can create or update documents in the appropriate collections based on the type of data being stored. For instance, if the context information pertains to environmental sensors, it can be stored in a `SensorData` collection. Similarly, data from different sources or entities can be stored in separate collections to ensure data organization and easy retrieval.

The use of this model provides several other benefits in the short-term persistency field. The document-based storage allows for fast read and write operations, making it suitable for real-time data streams. It also provides horizontal scalability of data, which allows any system the project is running on to handle increasing data volumes by distributing the data across multiple instances or replica sets. This is crucial in situations like a real-time driving scenario where there is a massive data production from sensors of different types.

Benefits of using the document collection model in short-term persistency

1. Flexible schema design;
2. High performance and scalability;
3. Real-time data processing;
4. Horizontal scalability;
5. Dynamic data model;
6. Rich query capabilities;
7. Easy integration with FIWARE.

It's worth noting that while a NoSQL database is a tremendous option for short-term data storage and retrieval, it may not be that optimal for long-term storage or historical analysis.

4.3.3 Long-term persistency

Ensuring the long-term persistence of contextual data is mandatory in situations where the core use-case relies on entity history analysis for decision-making support. One of the tools that makes this possible is FIWARE Cygnus, a component designed to persist context data streams.

"**History will be kind to me for I intend to write it**" (Churchill W., 1940)

Cygnus takes it's place in this architecture schema as a bridge between the CB that manages and provides real-time context information and persistent storage systems, including a range of many popular databases such as PostgreSQL, MySQL or AWS DynamoDB.

Upon receiving a context update, Cygnus extracts the relevant information, such as the involved entities' `id`, `type` and other attributes value. It then maps this information to the corresponding columns where each row corresponds to a specific context update, capturing, per column, the changes made to the observed entities' attributes over time.

Note: by default, Cygnus uses *row-like storing mode*[1].

[1] A configuration setup where contextual data is stored in a row-based format which allows for efficient storage and retrieval of individual updates, preserving original data integrity and structure.

With this configuration, as represented in the table schema 4.1, all the data captured from the CB can be efficiently stored into the appropriate tables in a structured manner, making historical querying easier. The sink acts as a target for Cygnus to deliver that data, ensuring its durability and accessibility over an extended period of time [13].

Attribute Name	Data Type	Description
recvTimeTs	TIMESTAMP	expressed in miliseconds
recvTime	TIMESTAMP	expressed in human-readable format
fiwareServicePath	INT	notified fiwareServicePath
entityId	VARCHAR(100)	notified entity identifier
entityType	VARCHAR(100)	notified entity type
attrName	VARCHAR(100)	notified attribute name
attrType	VARCHAR(50)	notified attribute type
attrValue	VARCHAR(50)	notified attribute value
attrMd	VARCHAR(100)	string serialization of the metadata array

Table 4.1: Cygnus long-term database table schema

Note: it is relevant to state that, different from the way short-term data sink is utilized exceptionally by the CB in a concealead layer, the NGSIJS service directly connects to the historical storage itself. This way, the system architecture requires the ability to execute direct manual queries on this database.

Based on these assumptions, it makes sense to follow a relational data model, where tables can be related to each other through defined relationships. The schema design for a database like this involves identifying the entities, attributes and relationships between them based on the requirements of the FIWARE configuration. This ensures the most efficient data organization.

One of the main system goals is to be able to retrieve historical context information about specific entities. By using a relational database to store this, it is possible to make SQL queries such as filtering, sorting and aggregation, which grants flexibility and expressiveness in the data retrieval process.

Benefits of using a relational database in long-term persistency

1. Data integrity and consistency;
2. Advanced querying capabilities;
3. Data indexing and optimization;
4. High availability and durability;
5. Transaction support.

On top of all the previous positive aspects of using a relational data sink for the persistence of historical data, my familiarity with databases like MySQL influenced the decision a lot. Having

experience with a particular database system allows a smoother development process, setup, configuration, data modeling and query optimization. This familiarity significantly reduces the learning curve and minimizes potential pitfalls that could arise when adopting a new database technology. Overall, this choice enables the design and development of a robust and well-supported long-term persistency storage in the FIWARE-based ecosystem.

4.4 Context Simulation

As it was described previously, Cygnus is configured with the necessary information to connect to both the CB and the long-term storage system. Cygnus establishes a subscription to the CB to receive notifications about entity updates. This subscription system ensures that Cygnus receives real-time updates whenever changes occur in the registered entities which contains information about the updated entity, including its identifier, type, attributes, values and the date when it occurs. Because of this, and since one of the motivations that led this project to take place was creating a context management system resistant to the idea of repetitions and experiments where simulations can take place without losing the original context, it was necessary to come up with a configuration model capable of empowering FIWARE to not only persist entity history but also their contextual history over possible repetitions indexed by that repetition identifier.

4.4.1 Repetition tracking

The system is repetition-aware which means each repetition is stored in a dedicated table in the long-term persistent database. This table serves as a repository for tracking the registered repetitions and consists of three essential columns: an auto-incremented index, a start and an end datetime as described in more detail in table 4.2.

Column	Type	Description
`id`	PK AUTO_INCREMENT	unique identifier
`startDate`	TIMESTAMP	start captured moment
`endDate`	TIMESTAMP	end captured moment

Table 4.2: Repetitions table schema

The repetition's `id` attribute assigns a unique identifier to each repetition, allowing for easy differentiation and referencing. This serves as a crucial reference point for retrieving specific repetitions and analyzing associated data, while linking each entity context with the current repetition identifier.

The `startDate` attribute captures the precise date and time when a repetition begins. It provides temporal context to the stored data, facilitating chronological organization and analysis of temporal patterns or trends.

The `endDate` attribute marks the conclusion of a repetition, representing the point when it ends or is terminated. As well as for the `startDate`, storing the finish datetime accurately captures the duration of each repetition, enabling relevant duration-based queries or event-based analysis.

4.4.2 Repetition history tracking

To overcome this repetition aware system challenge, the concept of "dummy" is introduced. A dummy is nothing more than an entity replica. Every time an entity gets registered by the CB, its dummy is created along, as Cygnus subscribes to both separately. This way it is possible to separate the original entity space from the so named dummy experimental instance which ensures data integrity and consistency.

- **Identifier**, an additional `:dummy` suffix is added to the entity original id in order to differentiate the replica from the original one;
- **Repetition**, the dummy entity holds an extra attribute `repetition` containing the value of the current repetition index it participated, respecting the NGSI entity attribute model of being a JSON containing its `value` and `type`;
- **Attributes**, being a replica, the dummy mirrors every other attribute from its original entity so all its information gets properly preserved.

This way, for a new entity like this in the following listing 4.2:

```
{
  "id": "sensor:MultipleSensor:1",
  "type": "Sensor",
  "temperature": {"type": "Float", "value": 27.3},
  "humidity": {"type": "Float", "value": 13.2},
  "noise": {"type": "Integer", "value": 72}
}
```

Listing 4.2: Example of a new entity instance

its dummy replica should be created as the below listing 4.3:

```
{
  "id": "sensor:MultipleSensor:1:dummy",
  "type": "Sensor",
  "repetition": {"type": "Integer", "value": X},
  "temperature": {"type": "Float", "value": 27.3},
  "humidity": {"type": "Float", "value": 13.2},
  "noise": {"type": "Integer", "value": 72}
}
```

Listing 4.3: Example of a new entity dummy instance

Note: the X value represents an arbitrary value of the current repetition index.

This process of creating a dummy replica for each entity instance brings some key aspects to the table such as:

1. **Historical context preserving**, since dummies enable the retention of historical context data, allowing for a comprehensive analysis of past states. This is invaluable for data-driven decision-making processes;
2. **Incremental updates**, by updating the dummy instead of the original entity, it is possible to ensure that modifications related to repetitions are stored separately. This approach promotes data traceability and allows for a clear distinction between original data and subsequent repetition changes;
3. **Long-term persistence**, as storing historical context data in dummies facilitates long-term persistence mitigating the risk of data loss or corruption. This is particularly crucial when dealing with critical systems like driving-scenarios where the ability to track and analyze changes over time is essential;
4. **Parallel revisions**, where it is possible to track multiple revisions of an entity simultaneously. This becomes advantageous when different repetitions or versions of an entity need to coexist, allowing for side-by-side comparison and analysis.

Although the presence of dummies adds complexity to the development process in terms of the data access and retrieval operations, it is still the most effective way of reaching the desired feature, while respecting the natural flow of data within FIWARE and preserving the scalability model this system requires.

4.5 Context Analysis and Visualization

For an efficient management of complex environments is crucial to deliver their end-users an intuitive and easy-to-use point of access. An ecosystem such as the one developed during this project, naturally suggests the need for a user-friendly interface to interact with service as one.

In an early stage of the development process, data flow was controlled via an instance of *Grafana*[2] as the Orchestra Cities project did for its solution context visualization. As the service reached a more evolved form, it seemed more natural to build a custom-designed solution-oriented interface. The following motivations, led the interface to be raised:

[2]Open-source analytics and visualization platform that allows to query, analyze and monitor data from various sources in real-time by providing a rich set of features and a user-friendly interface for creating interactive dashboards, charts and graphs.

User-Friendly interaction

A web interface is capable of providing a way for users to interact with the context management capabilities of FIWARE, allowing users to easily access and manage context data without requiring deep technical knowledge or needing a direct API integration.

Visualization and monitoring

A solution like this enables the visualization of context data, making it easier to understand and analyze the current state of entities and their attributes. It allows users to monitor real-time updates and events in a more visual format, providing valuable insights and facilitating decision-making.

Context data exploration

With the proper GUI, users can explore the available context data by searching, filtering and navigating through entities and their attributes. It allows them to drill down into specific data points, understand relationships and gain a more comprehensive view of the context information. This capability aids in data exploration, troubleshooting and analysis.

Context data manipulation

It provides the ability to create, update and delete contextual entries. Users can modify context information, set thresholds or rules and trigger actions based on specific conditions. This flexibility allows for a more dynamic management of the context and the adaptation of the system to changing requirements.

Configuration and administration

The interface is designed with a section where is possible to manage FIWARE subscriptions. This administrative view simplifies the management of the publish/subscribe infrastructure.

Accessibility and collaboration

A centralized platform like this can be accessible from different devices and locations. This accessibility promotes collaboration among stakeholders, which allows multiple users to work together, share insights and collaborate on context management tasks. This way it serves as a vital tool for leveraging the capabilities of FIWARE and maximizing the benefits of context-aware applications and services.

4.5.1 Entity management

This web interface, as it purpose indicates, provides a highly intuitive and efficient way to manage all the entities within the ecosystem stored and managed by the CB. Users can easily create or remove entities from the sink, allowing a seamless entity control. The platform grants an interactive and simple design, as presented in the figures 4.3 and 4.4, which simplifies the process of working with an arbitrary number of entity structures.

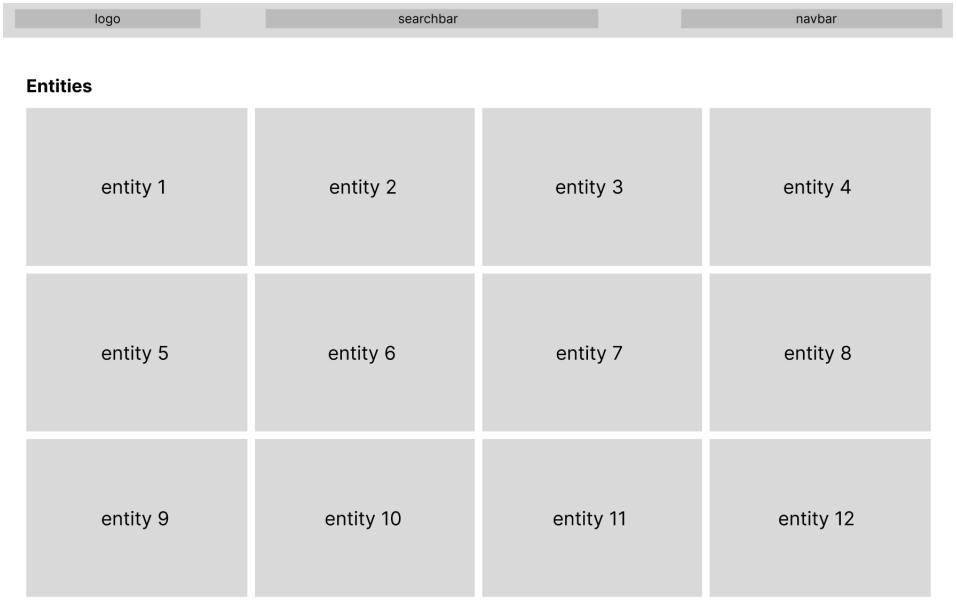


Figure 4.3: Entity management wireframe

Figure 4.4: Entity details wireframe

4.5.2 Repetition control

Managing repetitions involving tracking changes in entities attributes over time is a critical aspect of the ecosystem management. The interface empowers users to effortlessly monitor and control all the repetitions that took place in the simulation study by listing the repetition tracking like designed below in figure 4.5. This level of repetition control allows for improved decision-making, efficient troubleshooting and the implementation of data-driven strategies.

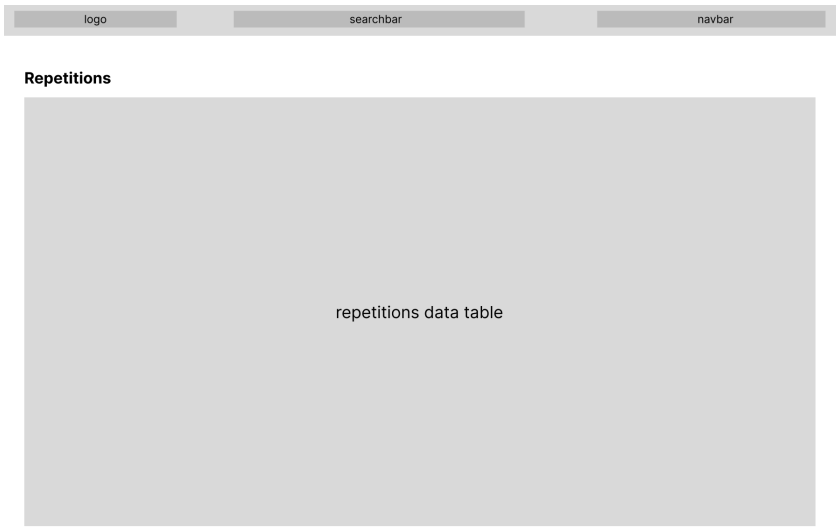


Figure 4.5: Repetition management wireframe

4.5.3 Entity history comparison

The ability to compare entities history is a powerful feature of the web interface and its conceptual wireframe is shown in figure 4.6. It enables users to track changes made to entities over time and compare different versions side by side. This functionality proves invaluable when investigating issues, debugging, or analyzing data inconsistencies. With the ability to navigate through entity revisions, users gain deeper insights into the evolution of their data, facilitating better understanding and more effective management of the FIWARE ecosystem.

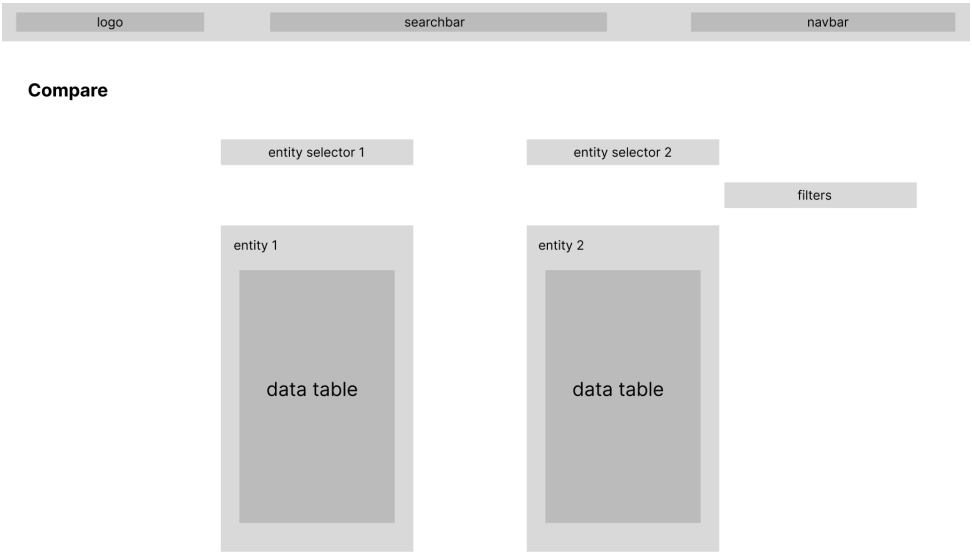


Figure 4.6: Context comparison wireframe

4.5.4 Subscription management

Subscriptions are fundamental to the real-time interaction between FIWARE services and external applications. The React web interface offers a comprehensive suite of tools for managing subscriptions effortlessly. Users can create, modify, and delete subscriptions with ease, ensuring the continuous flow of data from the FIWARE ecosystem to external systems. The interface provides clear visualizations of subscription status, enabling users to monitor and troubleshoot subscriptions effectively as proposed below in figures 4.7 and 4.8. This streamlined subscription management contributes to the overall stability and reliability of FIWARE-based applications.

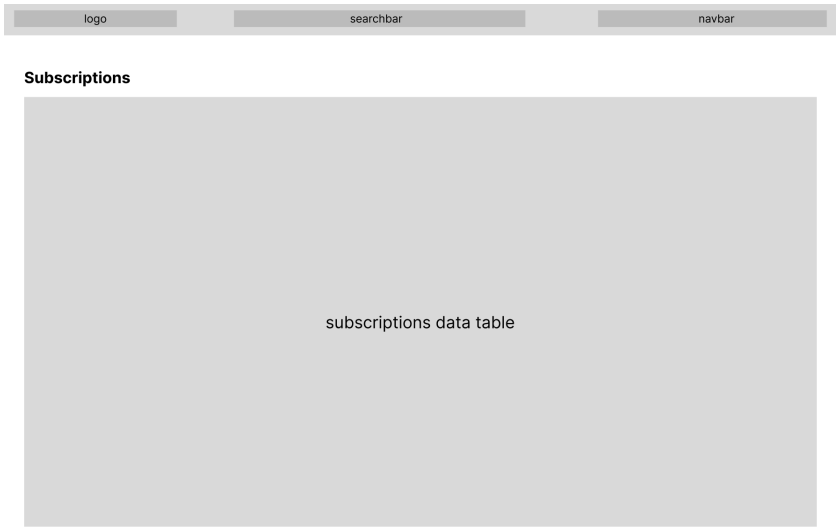


Figure 4.7: Subscription management wireframe

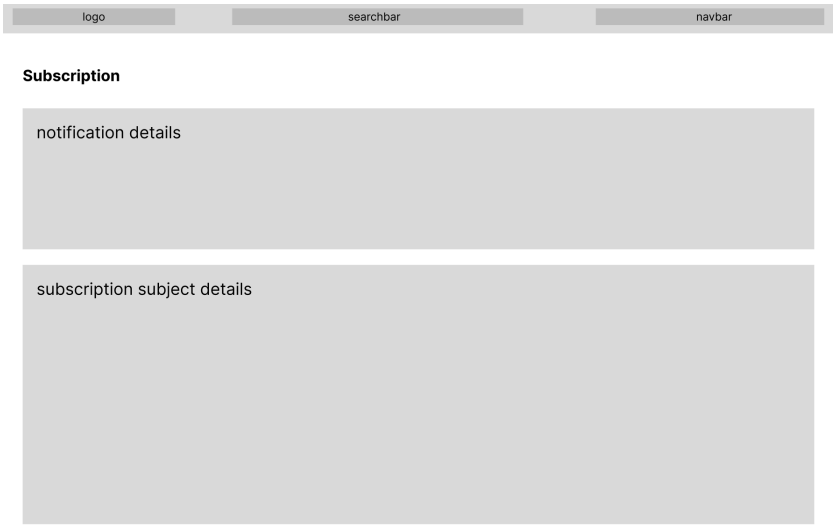


Figure 4.8: Subscription details wireframe

4.6 Summary

In this chapter it is discussed the design process and the skeleton components of the optimal architecture for this system. It is given an overview of the solution core, explained how data is set to be acquired and processed, and described the communication infrastructure.

Context Management

- **Entity modelling**
  - The NGSI is a standard data model and API specification developed by FIWARE for managing and exchanging data in smart-city applications and IoT systems;
  - Entity modelling is a key aspect of NGSI which provides a structured approach to representing real-world entities and their attributes contextual data;
  - Entities in NGSI are the core building blocks representing physical objects or abstract concepts, uniquely identified by an URI and belonging to an entity aggregation type;
  - The adopted structure for entity instances includes an identifier following the pattern `[type]:[name]:[idx]`, a type and other related attributes as JSONs containing a type and a value;
  - Attributes can represent both static and dynamic properties with different data types, which grants the data model flexibility and extensibility;
  - This design allows for the addition of new attributes or entity types without breaking compatibility, accommodating diverse and evolving fluid data requirements in a smart-city context.
- **Short-term persistency**
  - The CB is the component responsible for managing and providing access to real-time context information;
  - It follows a publish/subscribe model, with data producers publishing context information and consumers like the CB and Cygnus subscribing to specific types of data;
  - The selected data model for short-term storage in the architecture is the document collection model;
  - The NoSQL flexible schema design allows for different fields and structures within each document in a collection which is relevant in a SCDT scenario;
  - Context information received by the CB is stored in appropriate collections based on data type;
  - The document collection model offers benefits such as fast read and write operations, real-time data processing and horizontal scalability which contribute for the best possible performance in terms of short-term persistency of context.

- **Long-term persistency**
  - The long-term persistency is crucial for analyzing entity history in decision-making support;
  - FIWARE Cygnus acts as a bridge between the CB and persistent storage systems;
  - Cygnus maps context updates to tables in a row-based format for efficient storage and retrieval;
  - The relational database model is adopted for historical context persistency;
  - The table schema includes columns for timestamp, service path, entity identifier, entity type, attribute name, value and metadata;
  - The system architecture requires that direct manual queries can be performed on the database for historical data retrieval from the NGSIJS service;
  - A relational data model offers data integrity, advanced querying capabilities, indexing, high availability, durability and transaction support;
  - The familiarity with relational databases like MySQL influenced the decision to use it as the main historical data sink supporting the service.

Context Simulation

Cygnus is configured to connect to the CB and the long-term storage system. It establishes subscriptions to each entity registered to receive real-time notifications about their updates, including their identifiers, types, other attribute values and these snapshot dates. The project focuses on creating a context management system that can handle repetitions and simulations while preserving the original context of entities. A configuration model is implemented to enable the persistence of contextual history across repetitions, indexed by a repetition identifier.

- **Repetition tracking**
  - The system is designed to be repetition-aware, with each repetition being stored in a dedicated table in the long-term persistence database;
  - The repetitions table consists of three columns: an auto-incremented index, a start datetime and an end datetime;
  - The `id` attribute assigns a unique identifier to each repetition for easy referencing;
  - The `startDate` attribute captures the precise date and time when a repetition begins;
  - The `endDate` attribute marks the conclusion of a repetition, representing when it ends or is terminated;
  - This repetitions table allows for easy differentiation and chronological organization;
  - The current repetition is defined by the last row in the repetitions table.

- **Repetition history tracking**
  - The concept of dummy is introduced to overcome the challenge of managing experimental repetitions while preserving data integrity and consistency;
  - A dummy is an entity replica, created once the original is registered by the CB differentiated by adding a `:dummy` suffix to its identifier;
  - The dummy includes an additional attribute `repetition` that contains the value of the current repetition index it is taking part of;
  - This dummy mirrors the other attributes from the original entity to preserve all its information;
  - Dummies enable the retention of historical context data, allowing for comprehensive analysis of past states;
  - By updating a dummy instead of the original entity, modifications and updates related to repetitions are stored separately, promoting data traceability;
  - Storing historical context data in dummies facilitates long-term persistence and mitigates the risk of data loss or corruption;
  - The dummy system allow for tracking multiple revisions of an entity simultaneously, enabling side-by-side comparison and analysis;
  - Although this system add complexity to the development of data access and retrieval operations, it is the most effective way to achieve the desired features within the FIWARE framework nature.

Context Analysis and Visualization

- **Entity management**
  - Main view for managing entities stored and managed by the CB;
  - Contain an intuitive and efficient design for creating or removing entities;
  - Reduces the effort of working with an arbitrary number of entity structures.
- **Repetition control**
  - Enables monitoring and control of registered repetitions;
  - Tools and visualizations for comparing historical data points;
  - Facilitates decision-making, troubleshooting and data-driven strategies.
- **Entity history comparison**
  - Powerful feature for tracking changes made to entities over time;
  - Allows side-by-side comparison of different entities;
  - Valuable for investigating issues and analyzing data patterns and trends.

- **Subscription management**
  - Comprehensive tool for managing subscriptions seamlessly;
  - Clear administrative visualizations of subscription status for monitoring and troubleshooting;
  - Contributes to the stability and reliability of entity updates notifications.

In the upcoming chapter, 5, an in-depth exploration of the configuration and deployment of the solution is presented. The focus of this chapter is to provide a comprehensive overview of the more technical aspects involved in the development process and it is structured in a way that corresponds to the different architectural components mentioned through this architecture chapter. This examination aims to shed light on the intricacies of implementing the proposed solution, providing valuable insights into the development journey.

Chapter 5

Configuration and Deployment

As proposed in the previous chapter 4, the system development follows the architecture designed. The system components responsible for creating a robust data communication and storage unit are in the bottom layer of the layout which includes the the MySQL sink, Cygnus, Context Broker (CB) and the MongoDB sink. The web interface on the top side of the layout makes use the middle layer service for managing data flow between south and north components. With this configuration as presented by figure 5.1, all the components form a scalable and reliable unit capable of addressing the context management, visualization and simulation requirements. Their interdependencies and communication ports are listed in the table 5.1 below.

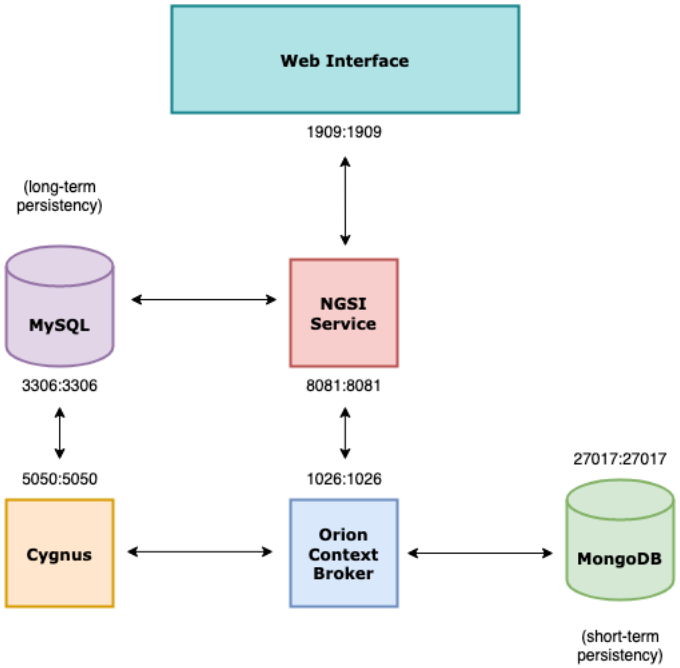


Figure 5.1: System component architectural layout

Note: Cygnus, by default, exposes the port 5050 for receiving notifications of contextual data updates and the port 5080 for operational purposes [14].

Component	Dependencies	Listening Ports
Context Broker	Short-term Storage	1026
Cygnus	Long-term Storage	5050:5080
Short-term Storage	-	27017
Long-term Storage	-	3306
NGJSI Service	Context Broker, Cygnus	8081
Web Interface	NGSIJS Service	1909

Table 5.1: System components dependencies and network configuration

5.1 FIWARE Ecosystem

At the configuration level of FIWARE components, the environment variables for Cygnus are set using the following configuration, as shown in table 5.2. Additionally, the access configuration for the long-term persistency sink using MySQL is set up as illustrated in the following table 5.3.

Variable	Value	Description
CYGNUS_MYSQL_SERVICE_PORT	5050	notification port Cygnus listens to when subscribing to context data updates
CYGNUS_MYSQL_HOST	mysql-db	MySQL server hostname
CYGNUS_MYSQL_PORT	3306	port used by MySQL server to listen to commands
CYGNUS_MYSQL_USER	root	username for MySQL database user
CYGNUS_MYSQL_PASS	123	password for MySQL database user
CYGNUS_SERVICE_PORT	5050	port on which Cygnus is listening for subscriptions
CYGNUS_API_PORT	5080	port on which Cygnus is listening for operations
CYGNUS_LOG_LEVEL	DEBUG	logging level for Cygnus

Table 5.2: Cygnus environment variables

Variable	Value	Description
MYSQL_ROOT_HOST	%	MySQL server root host
MYSQL_ROOT_PASSWORD	123	MySQL server root password

Table 5.3: MySQL sink environment variables

5.2 NGSIJS Service

A huge fraction of this project focused on designing and building a centralized service dedicated to the context management. This service aimed to simplify the complexities involved in directly managing FIWARE using the Next Generation Service Interface (NGSI) API, by providing a

custom *RESTful*[1] API capable of acting as a proxy layer between the user and the FIWARE context management components such as Orion CB and Cygnus and also the long-term persistency. Thus, it's motivations are being able to let users configure and monitor the various components within the ecosystem. Being a centralized interface, easier-to-use, users can harness the full potential of the FIWARE side, streamlining their operations, thereby reducing the effort required into the module control.

5.2.1 Dependencies

The service was build using NodeJS and upon the following dependencies in table 5.4:

Dependency	Version
`express`	4.18.2
`ngsijs`	1.5.0-rc.0
`mysql`	2.18.1
`moment`	2.29.4
`cors`	2.8.5

Table 5.4: Service npm-dependencies

`express` was used to simplify the process of building the API. Perfectly suitable to use with NodeJS, it abstracts away many low-level details, such as HTTP request handling, routing or middleware management, which allowed the focus on the core functionality design of the service. https://www.npmjs.com/package/express

`ngsijs` is a JavaScript (JS) library that provides a client-side implementation of NGSI and serves as a toolset for developers to interact with NGSI-comptabile servers from their applications. This way it was possible to mask the NGSI API with a custom one. https://www.npmjs.com/package/ngsijs

`mysql` is a package which provides a driver for connecting to and interacting with MySQL databases. Since the historical-persistency sink associated with Cygnus is running on MySQL, it is possible to establish a connection with the sink and perform historical query operations directly. https://www.npmjs.com/package/mysql

`moment` is a library that provides a straightforward way to work with dates and times in JS applications. It simplifies tasks related to parsing, manipulating, formatting and displaying them, which was truly valuable due to the nature of data entries on entity management and context updates. https://www.npmjs.com/package/moment

`cors` is a package is a middleware for handling Cross-Origin Resource Sharing (CORS) in NodeJS applications. CORS is a security mechanism implemented in web browsers which

[1] An architectural style for designing networked applications, based on the principles of the REST architectural pattern, which emphasizes a stateless, client-server communication model.

restricts cross-origin HTTP requests, i.e. requests that are made from one domain to another. By default, web browsers enforce the same-origin policy, which allows requests only from the same domain as the one serving the web page. Given the need to make cross-origin requests from the service to NGSI. CORS provides a way for servers to indicate which cross-origin requests should be allowed and which should be blocked. This package simplified this implementation process. https://www.npmjs.com/package/cors

5.2.2 API Documentation

For a better endpoint categorization, the API can be divided into three main sectors regarding their purpose nature:

- **Entity-based**, grouping all the endpoints related to entity management;
- **History-based**, holds all the endpoints for long-term persistency;
- **Subscription-based**, for all the subscription processing.

5.2.2.1 Entity-based

GET /entity/list

Allows to retrieve the list of entities registered by the CB in the short-term persistency storage. The response body contains the list of entities and the total number of entries found. It is also possible to filter this query by the set of attributes in focus, by an entity identifier matching pattern and even choosing to include or not dummy replicas in the response, as described below in table 5.5.

Query Param	Description
attrs	list of attributes to select
idPattern	filters entities whose identifiers match this pattern
noDummies	states entity replica inclusion

Table 5.5: Entity listing API documentation

GET /entity/:id

Retrieves an entity from the data sink given its identifier value as in the table 5.6. The successful response contains the retrieved entity object. Although, if the entity is not found, a 404 status code will be returned along with the not found error message.

Path Variable	Description
id	requested entity identifier

Table 5.6: Entity retrieval API documentation

POST /entity/create

This is the core entity creation endpoint. It performs additional sub-operations such as creating the dummy entity replica right after the original entity gets registered, building and registering subscriptions for both. Finally, the server responds with the result of these operations.

The request body must be populated with the entity model, like the example in listing 5.1:

```
{
    "id": "sensor:EnvSensor:1",
    "type": "Sensor",
    "airQuality": {"type": "Integer", "value": 12},
    "humidity": {"type": "Float", "value": 96.5},
    "pressure": {"type": "Float", "value": 1012.3},
    "temperature": {"type": "Float", "value": 13.2},
    "cloudcover": {"type": "Integer", "value": 50}
}
```

Listing 5.1: Example of the API response for an entity creation

POST /entity/update

Makes possible to update specific attributes of an existing entity by providing the actual changes in the request body. The server then performs the update operation responds with its result status along with a success or error message depending on the case scenario.

The request body must be populated with the entity identifier, followed by the attributes to change, like this example 5.2:

```
{
    "id": "sensor:EnvSensor:1",
    "humidity": {"type": "Float", "value": 30.1},
    "temperature": {"type": "Float", "value": 23.9},
}
```

Listing 5.2: Example of the API response for an entity update

DELETE /entity/:id/delete

Allows to delete entities from the record by passing an `id` identifying the entity desired to be deleted, like illustrated in the table 5.7. The server then performs the deletion operation, returning the result status along with the operation result message.

Path Variable	Description
id	Requested entity identifier

Table 5.7: Entity deletion API documentation

5.2.2.2 History-based

GET /history/entity/:id

Enables the entity history data retrieval given its identifier, described in table 5.8. The server then fetches the entity's history based on the filters provided via query parameters and sends the response back to the client, containing the found history data entries.

Through this call it is possible to filter the history data using the following list of query parameters which are then described in the table 5.9:

- `attrName`, sets the attribute to focus on, ignoring all the other entity attributes (note that this parameter can be used multiple times in the same query for multiple attribute focus);
- `startDate`, sets the search lower date boundary;
- `endDate`, sets the search higher date boundary;
- `limit`, limits the response length.

Path Variable	Description
id	requested entity identifier

Table 5.8: Entity history retrieval API documentation I

Query Param	Description
attrName	attribute to view
startDate	data records starting from given date
endDate	data records until given date
limit	result size limit

Table 5.9: Entity history retrieval API documentation II

POST /history/repetition

Used to notify the CB and Cygnus about a new contextual repetition. The server then performs the necessary operations to update all the entities context and propagates the event to both sinks. There are three ways of starting a new contextual repetition and for each one of them, the required request body structure is exemplified below in listings 5.3, 5.4 and 5.5:

- From current context:

```
{
    "entitiesModified": [
        {"id": "sensor:EnvSensor:1", "temperature": {"value": 12}}
    ],
}
```

Listing 5.3: Example of the API request body for repetition start from the current context

- From the context on the repetition that started the closest to the given date:

```
{
    "entitiesModified": [
        {"id": "sensor:EnvSensor:1", "temperature": {"value": 12}}
    ],
    "startDate": "2023-04-12T12:32:45.00Z"
}
```

Listing 5.4: Example of the API request body for repetition start from past date

- From the context on the repetition with given index:

```
{
    "entitiesModified": [
        {"id": "sensor:EnvSensor:1", "temperature": {"value": 12}}
    ],
    "fromRepetition": 5
}
```

Listing 5.5: Example of the API request body for repetition start from a given repetition index

GET /history/repetition/list

Allows to retrieve the list of all the registered repetitions along with their indexes and starting and ending dates.

PATCH /history/repetition/end

With this, it is possible to notify the CB and Cygnus about the ending of a repetition. The server tries to mark the repetition as ended in the historical database and sends a response back to the client indicating the result of this operation.

The request body must contain the repetition index, like the following listing 5.6:

```
{
    "repetitionId": 5
}
```

Listing 5.6: Example of the API request body for ending a repetition

5.2.2.3 Subscription-based

GET /subscription/list

Makes possible to retrieve the list of all the subscriptions registered by the CB. After fetching the results, the server sends the response, containing the plain list along with their entry details, back to the client.

POST /subscription/create

This is the API call intended to create a new subscription to the CB server by providing the subscription details in the request body. The service then registers the subscription and sends the result status as the response back to the client.

The request body must be populated with the subscription description, subject, notification and expiration date, like the example 5.7 below:

```
{
    "description": "Subscription to "sensor:EnvSensor:1" airQuality updates.",
    "subject": {
        "entities": [{"id": "sensor:EnvSensor:1", "type": "Sensor"}],
        "condition": {"attrs": ["airQuality"]}
    },
    "notification": {
        "http": {"url": "[HOST]:[PORT]/notify"},
        "attrs": ["airQuality"]
    },
    "expires": "2040-01-01T00:00:00.00Z"
}
```

Listing 5.7: Example of the API request body for creating a subscription

DELETE /subscription/:id/delete

Enables the subscription removal, where `id` identifies the subscription to be deleted, as described in table 5.10. The server then tries to delete the subscription from its short-term entry and sends the operation result back to the client.

Path Variable	Description
id	requested entity identifier

Table 5.10: Subscription deletion API documentation

POST /subscription/update

Used to update an already existing subscription by including the required updates on the subscription and its identifier in the request body. The server then applies the changes and sends the response back to the client, indicating the success or failure of the update operation.

The request body must be populated with the subscription new attribute values, like it is properly exemplified in the listing 5.8:

```
{
    "id": "64205c5666a9a2761c071a2b",
    "description": "Subscription to "sensor:EnvSensor:1" cloudcover updates.",
    "subject": {
        "entities": [{"id": "sensor:EnvSensor:1", "type": "Sensor"}],
        "condition": {"attrs": ["cloudcover"]}
    },
    "notification": {
        "http": {"url": "[NEW_HOST]:[NEW_PORT]/notify"},
        "attrs": ["cloudcover"]
    },
    "expires": "2050-01-01T00:00:00.00Z"
}
```

Listing 5.8: Example of the API request body for updating a subscription

This API documentation provides an overview of the NGSI service features, and its details are summarized in the table 5.11 below.

Endpoint	Description
/entity/list	lists all entities registered in the sink
/entity/:id	retrieves an entity from the sink
/entity/create	creates a new entity along with its dummy
/entity/update	update an existing entity
/entity/:id/delete	deletes an entity from the sink
/subscription/create	creates a new subscription
/subscription/:id/delete	deletes a subscription from the sink
/subscription/update	updates an existing subscription
/subscription/list	lists all subscriptions registered in the sink
/history/entity/:id	retrieves the historical data of an entity
/history/repetition	creates a new contextual repetition
/history/repetition/end	ends a contextual repetition

Table 5.11: NGSIJS Service API description

5.2.3 Utils

All the development process was mainly focused on the organization, reusability, maintainability and modularity of core components, either services or interfaces. This way, the system main logic and relevant helpers are kept decentralized and distributed across multiple utils, which follows the modular architecture previously proposed.

- **Context Broker util**, containing the toolkit for entity and subscription building. Overall, this toolkit enhances the usability of working with the CB by providing convenient functions to parse input data, construct entity listing query options, modify entity structures and

generate subscriptions. These helpers are presented and described in the table 5.12 below. They save development time and effort by offering reusable and purpose-specific functionalities for common tasks related to the NGSIJS service and CB communication.

Method	Description
`parseStringToBoolean`	parses 'true' and 'false' to their proper boolean values
`buildEntityListOptions`	receives an entity list request query object and maps it into the appropriate NGSIv2 entity list options object
`buildEntity`	receives an entity and pushes a new `subscriptions` attribute to it
`buildEntityDummy`	receives an entity and builds up its corresponding dummy replica for repetition experiments
`buildCygnusSubscription`	receives an entity and propagates its information to a Cygnus environment subscription

Table 5.12: Context Broker utilities description

- **Cygnus-MySQL util**, containing the toolkit for Cygnus-sink data pipes. This toolkit facilitates the data interchangeability between Cygnus and the long-term data sink by providing utility functions for string manipulation, entity identifier processing, MySQL interaction, date conversion, attribute value parsing and context handling. These auxiliar functions, presented in the table 5.13, save a lot of development time by improving the overall code readability and offering a lot of convenience in common tasks related to database interactions and consequent data transformations.

Method	Description
`replaceAll`	given a string, a substring and a replacement, returns the original string with all the substring ocurrences replaced.
`getEntityType`	gets the entity type given its identifier (using NGSI entity adopted structure `[type]:[name]:[idx]`)
`matchMySQLTableName`	translates the NGSI entity identifier adopted structure into MySQL tables name convention `[type]_[name]_[idNum]_[Type]`
`runQuery`	given a `mysqlConnection` instance and a SQL query returns a promise with the query results
`dateToSQLDateTime`	given a date, parses it to match the SQL datetime format
`parseMySQLAttrValue`	since attributes values come in string format from MySQL, this parses them regarding the attribute type
`getContextOnRepetition`	gets the list of all the dummy entities present in the repetition

Table 5.13: Cygnus-MySQL utilities description

- **Query util**, containing all the defined MySQL driver query calls. The utility includes a variety of functions, described in table 5.14, that allow the retrieval of historical data from the MySQL database based on different criteria such as entity identifiers, attributes names and date ranges. It provides a more flexible query system that enhances the historical data retrieval.

Method	Description
`getEntityHistory`	gets an entity history given its identifier
`getEntityHistoryFromAttribute`	gets an entity attribute history given its name and its entity identifier
`getEntityHistoryFromLimit`	gets an entity history given its identifier limiting the number of entries to be returned
`getEntityHistoryFromDateRanges`	gets an entity history given its identifier and date ranges
`getEntityHistoryFromAttribute-AndLimit`	gets an entity attribute history given its name and entity identifier limiting the number of entries to be returned
`getEntityHistoryFromAttribute-AndDateRanges`	gets an entity attribute history given its name and its entity identifier given a date range
`getClosestRecvTime`	gets an entity closest `recvTime` value given a date value
`getRepetitionList`	gets the list of registered repetitions
`getRepetitionStartDate`	gets the `startDate` value of the corresponding repetition given its index
`getNextRepetitionIndex`	gets the index to be used in the next repetition by incrementing the current one's
`startRepetition`	starts a repetition by propagating this event to the repetitions table
`endRepetition`	ends a repetition given its index by updating its `endDate` attribute on the repetitions table

Table 5.14: Query utilities description

5.3 Context Management Interface

The web interface was developed using React 18.2.0 and NodeJS 18.14.2. By employing React, the interface benefits from its component-based approach, which allows for a modular development and reusability. On the middleware side, NodeJS enables efficient handling of HTTP requests coming from the NGSIJS server and provides a robust foundation for the client-server communication. This implementation process took in consideration some key development concepts:

1. **Modular architecture**, as all the application can be broken down into smaller components, easier to manage, reuse and maintain which helps promoting code reusability, scalability and improves general code review efficiency;
2. **Responsive design**, where it has been made an effort into ensuring that the layout adjust and adapt seamlessly to provide an optimal user experience;
3. **Test-driven development**, where for every feature or view developed, it was properly tested against real context data, which helped identifying potential issues early, ensuring better code coverage and improving overall code quality;
4. **Version control**, where *Git*[2] was used to track development changes.

[2]A distributed version control system widely used for tracking changes in source code during software development.

5.3.1 Components

The structure of a React component can vary based on its complexity and specific requirements. However, a common practice among the frontend developers community is to organize them into separate directories and files for better maintainability and scalability like this:

```
Component/
└── index.jsx
```

In this structure, the `Component` directory contains the `index.jsx` file, which serves as the entry point for the component, including its implementation and any necessary imports. This is helpful because inside other components or pages it is possible to make cleaner imports, which keeps them more organized like:

```
1 import Component from '../Component';
```

Below, in the descriptions table 5.15, are presented and briefly described the main components that build up the web interface views.

Component	Description
ActionFloatButton	float button for opening action modals
AttributeTypeTag	attribute type tags display
CompareCard	card element for the comparing area
CompareEntitySearch	component for searching and comparing entities
EntityAttribute	entity attribute display
EntityCard	card format entity information display
EntityHistoryTable	table with entity historical data
EntityList	list of entities
Navbar	main navigation bar
NewEntityModal	modal for entity creation
NewRepetitionForm	form for repetition building
NewRepetitionModal	modal for repetition creation
OnCreateEntityModal	modal for entity creation result
OnStartRepetitionModal	modal for repetition start result
RepetitionTable	list of repetitions
SearchBar	entity search bar
SubscriptionTable	table with subscription/notification information

Table 5.15: Interface components description

5.3.2 Pages

The different feature panels of the application are described below in table 5.16, which provides a brief overview of their role in the usability system.

Page	Description
Compare	entity historical data comparison
Entity	entity current and historical contexts
EntityList	entity listing control
Home	application entry view
RepetitionList	repetition listing control
Subscription	subscription information details
SubscriptionList	subscription listing control

Table 5.16: Interface pages description

5.3.2.1 Entity management

From this view, illustrated in figure 5.2, it is possible to see the list of entities registered in the current context storage. Each entity is presented in a card-format for a better data display, where is possible to see the identifier, type tag and four principal attributes and their data types.

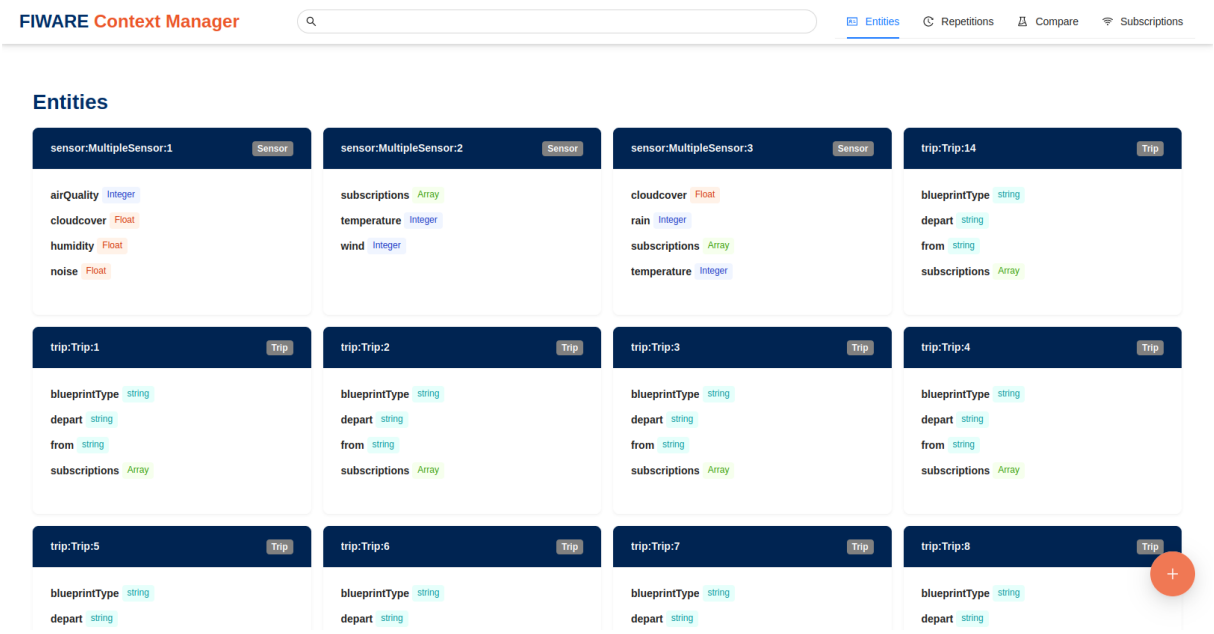


Figure 5.2: Entity list page

By clicking on one of the cards, the application will navigate to the corresponding entity view where it is possible to look up its context in more detail as it is presented in the figure 5.3 below.

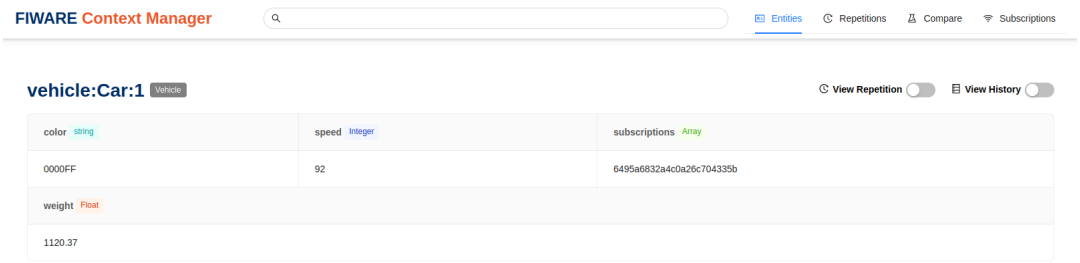


Figure 5.3: Entity details page

As shown in figure 5.4, by activating the repetition view filter it is possible to visualize the current dummy details without leaving the entity context visualization page.

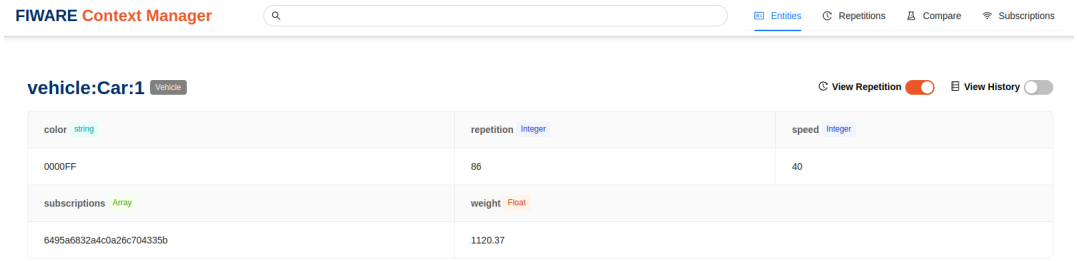


Figure 5.4: Entity dummy details page

Toggling the entity history view filter it is possible to visualize in detail the persisted context data over time, as shown here with figure 5.5.

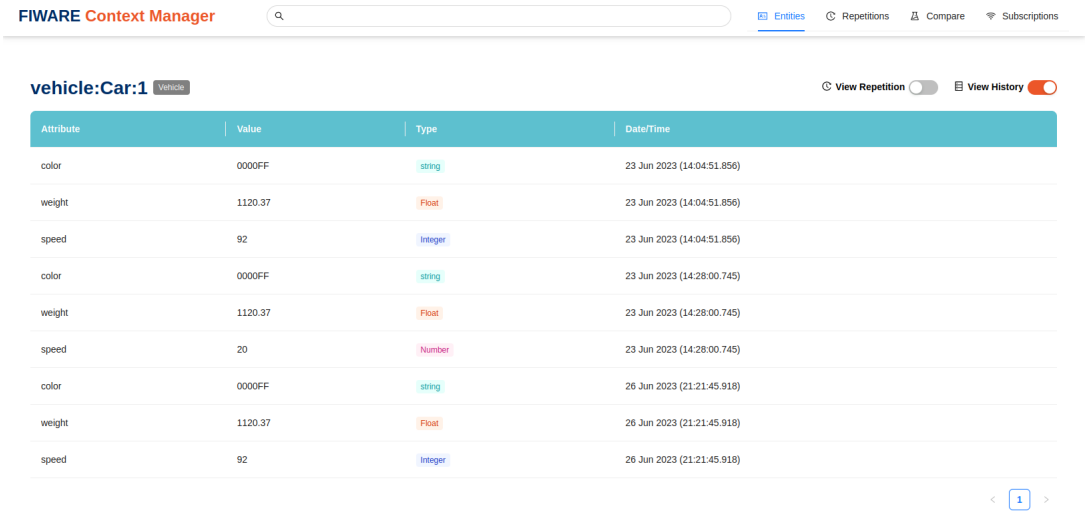


Figure 5.5: Entity history page

If both filters are activated, it becomes possible to go through the persisted dummy replica historical updates over time, like the illustration 5.6 demonstrates.

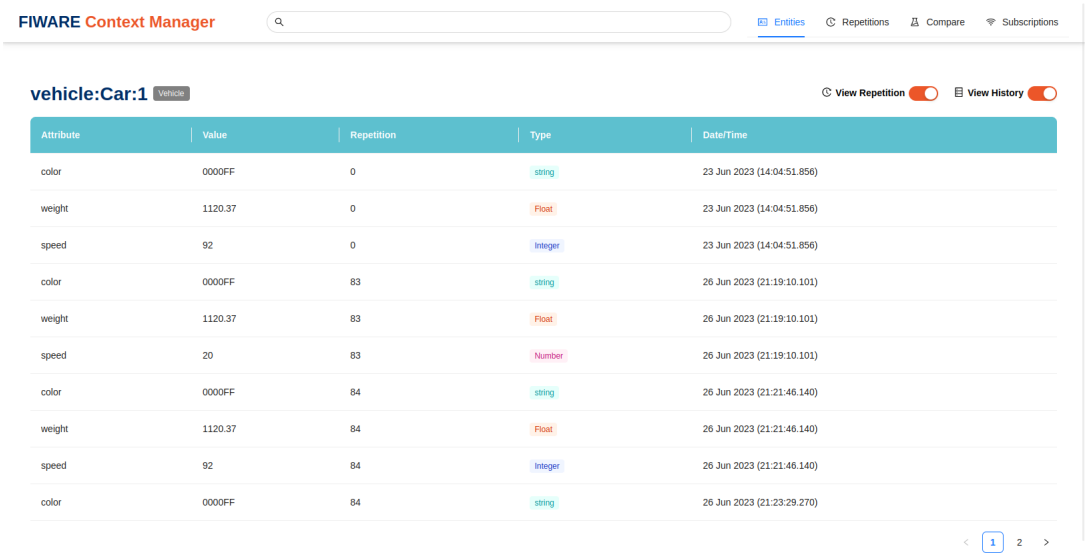


Figure 5.6: Entity dummy history page

5.3.2.2 Repetition control

The repetition control panel lists all the repetitions started and registered within the system, indexed by their identifier, displaying the date they started and their end date, if they are already terminated. This view can be analyzed here in figure 5.7.

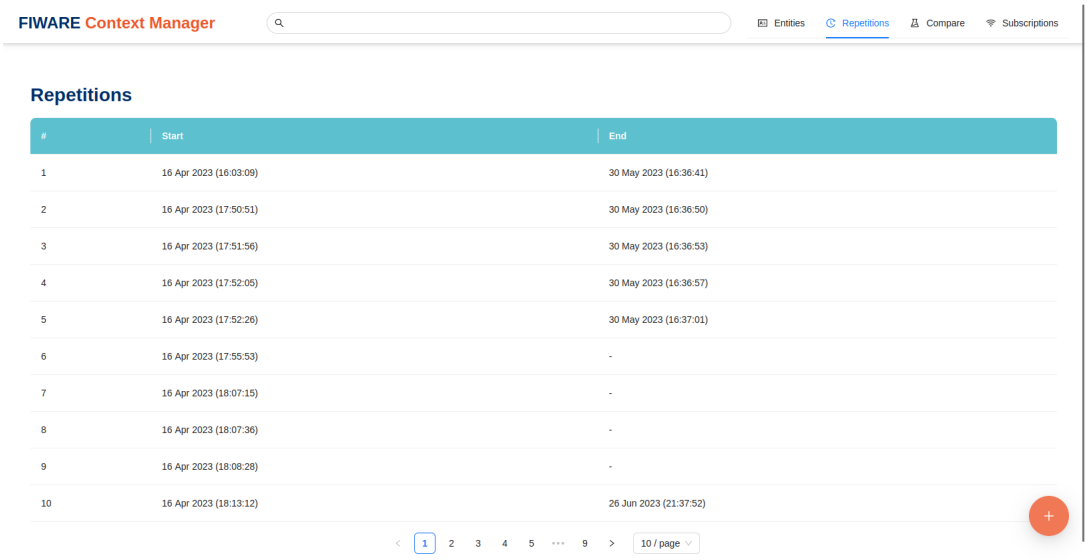


Figure 5.7: Repetition list page

5.3.2.3 Entity history comparison

The entity history comparison view enables users to compare the historical context evolution of two entities side by side, as the figure 5.8 illustrates. This feature allows to easily analyze and contrast the past data and attributes of these entities. By visually presenting the historical changes and patterns in a convenient side-by-side view, users can effectively evaluate their similarities, differences and trends over time.

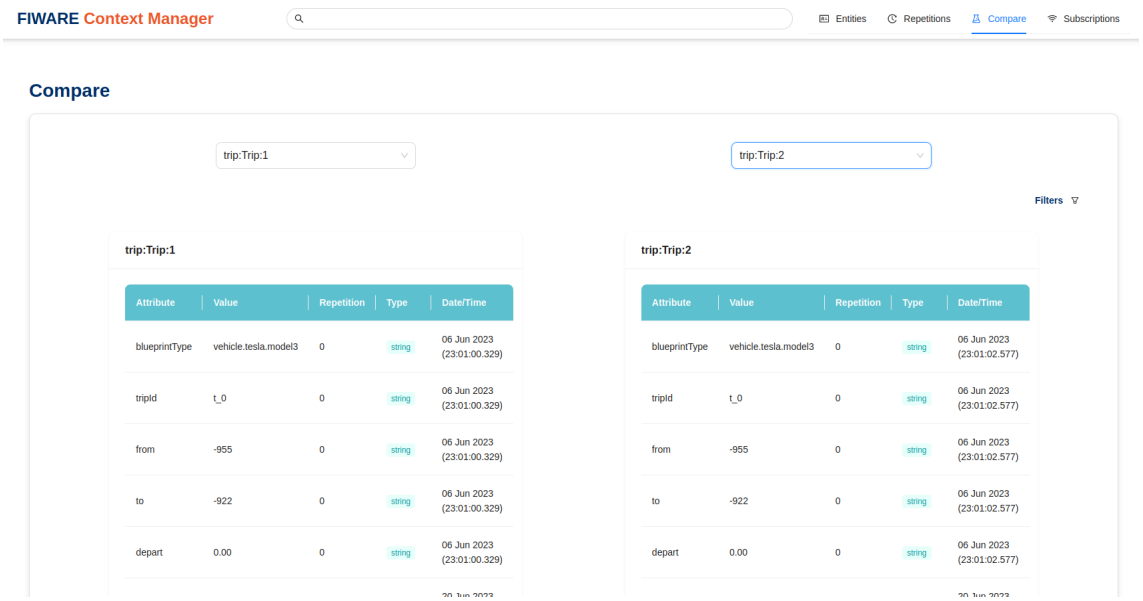


Figure 5.8: Entity history comparison page

5.3.2.4 Subscription management

The subscription management panel provides users with the list of contextual subscriptions registered in the CB. This feature allows users to conveniently access and manage all their subscriptions in one centralized location. This functionality enables to easily monitor and control data subscriptions, ensuring they stay up-to-date with relevant information and enabling efficient management of their subscribed data sources. This feature is properly illustrated in the following figure 5.9.

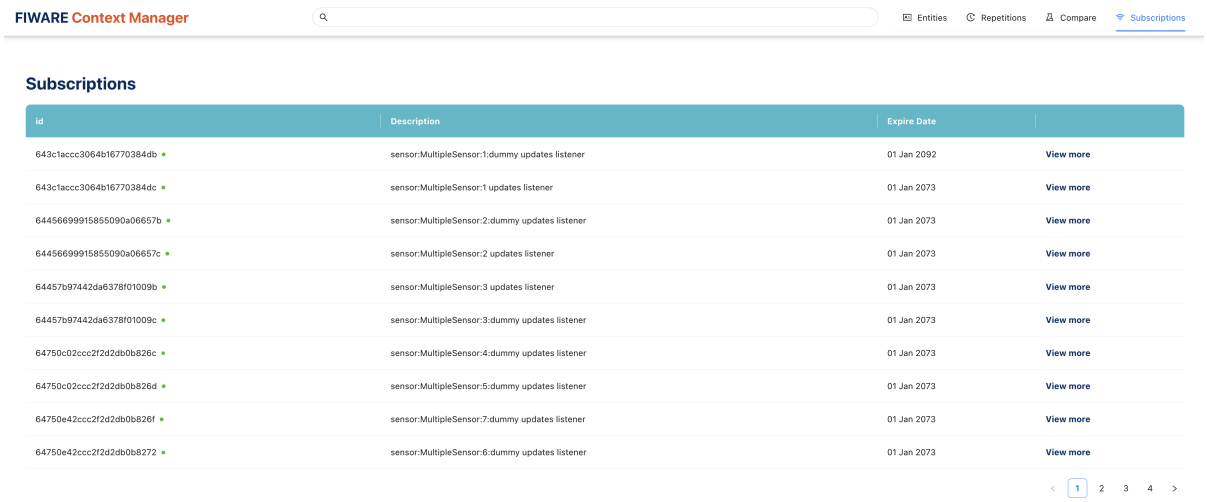


Figure 5.9: Subscription list page

By clicking in the “View more“ button, a detailed view over that entry’s subscription opens up, as shown in the figure 5.10.

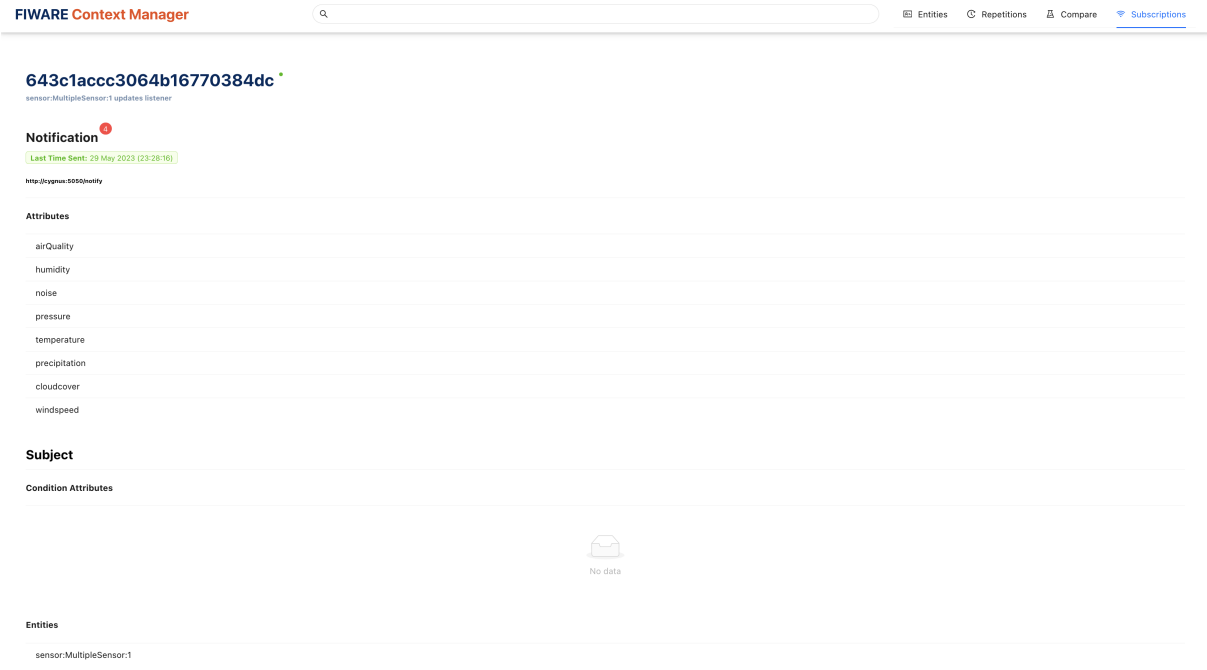


Figure 5.10: Subscription details page

5.4 System Containerization

Since this project is composed by several dependent components, it seemed natural to keep each one of them in containers building up a stable virtual environment.

Containerization, illustrated below in figure 5.11, revolutionizes the way that applications are packaged and deployed, providing huge benefits that address the challenges of modern software development and deployment. At its core, it is the process of encapsulating an application and its dependencies into a lightweight, self-contained unit known as a container [11]. This leverages the concept of operating system-level virtualization, allowing developers to build isolated and portable environments that can run consistently across different computing environments. Docker was the platform of choice for this project containerization.

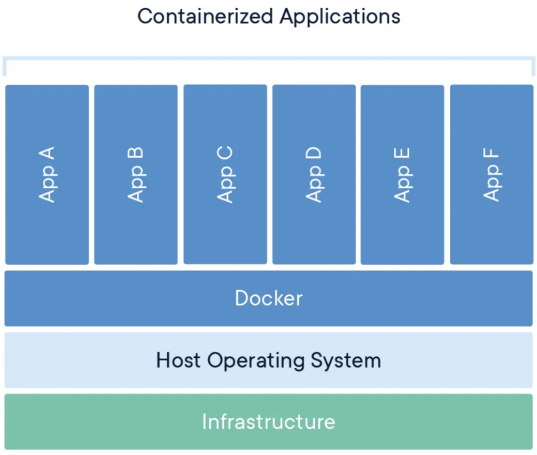


Figure 5.11: Containerization with Docker

5.4.1 Motivation

1. **Portability**, by encapsulating all the services within containers, it is possible achieve a remarkable level of portability. Each container includes all the code, dependencies and configurations needed to run a service. This approach ensures that all the services run consistently across different environments, such as development, testing or production which eliminates potential issues caused by variations in host operating systems, libraries and configurations. This was key during all the development process, because of the need to swap between Windows 11, macOS Ventura and Linux Ubuntu due to equipment availability.

2. **Isolation**, providing essential process-level isolation for any project component. Each container operates as an isolated process, having its own file-system, network stack and resource limits. This isolation mitigates conflicts between different services or dependencies. Moreover, it is possible to run multiple instances of the same service on a single host without any interference. This level of isolation enhances the stability and security of any service.

3. **Dependency Management**, it enables developers to define service's dependencies explicitly. This includes specifying the required libraries, packages and tools for a proper

function. By containerizing services, it is possible to eliminate the need for manual installation and configuration of dependencies on each host. This process ensures a consistent and reproducible environment, reducing the risk of compatibility issues and simplifying the management of dependencies across different services and environments.

4. **Scalability**, as containers are designed for easy horizontal scaling. This enables effortless scaling of services through the deploy of multiple instances of the same container. Containers can also be orchestrated using tools like Docker Swarm or Kubernetes, which automates scaling, load balancing and fault tolerance.

5. **Resource Efficiency**, since containers are lightweight and leverage the host operating system's kernel, this results in a minimal resource consumption compared to a traditional Virtual Machine (VM). Containers have faster startup times, require less memory and impose minimal overhead. This efficiency leads a project to run more containers on the same hardware infrastructure, optimizing resource utilization and reducing overall costs.

6. **Version Control and Reproducibility**, the image-based approach allows to package and distribute any service in a version-controlled manner. Each image represents a specific version of the service which enables the track of changes, the rollback to previous versions and the insurance of reproducibility. Containerization facilitate the creation, tagging and sharing of images across teams and environments, enhancing collaboration and ensuring consistent deployments.

All the code involved in this development process can be found at https://github.com/franciscorg7/fiware-scdt

5.5 Summary

This chapter delves into a comprehensive and technical exploration of the development process.

- **FIWARE ecosystem**

  In this section are provided all the environment variables for the Cygnus component within the FIWARE ecosystem. Cygnus is the data management tool that subscribes to context data updates and stores them in the MySQL database. This content is organized into technical tables that provide details about the environment values required for configuring Cygnus and the long-term persistency sink. These environment variables are used to configure and connect Cygnus with the MySQL server, allowing it to store the subscribed context data in the database.

- **NGSI service**

  The NGSI service is a centralized platform designed to simplify the management of the FIWARE context using the NGSI API. It acts as an intermediary between the user and core

components like the CB and Cygnus. The service provides a totally customized API that allows users to configure and monitor various components within the ecosystem, making it easier to use and fully utilize the capabilities of FIWARE. This was a major advantage since the need to have a tailored web interface capable of communicating with the underlying system.

The service is directly supported by a custom toolkit containing three utility files:

- `context-broker.toolkit.js`, that makes easier the interaction with the CB;
- `cygnus-mysql.toolkit.js`, for integrating Cygnus with the historical data sink;
- `queries.js`, containing a set of predefined queries for historical data retrieval and manipulation.

The centralized layer proper functionality depends on five different npm packages:

- `express`, version 4.18.2
- `ngsijs`, version 1.5.0-rc.0
- `mysql`, version 2.18.1
- `moment`, version 2.29.4
- `cors`, version 2.8.5

The API can be divided into three main sections based on the endpoints nature:

- Entity-based, for entity management purposes;
- History-based, used for long-term persistency operations;
- Subscription-based, for subscription processing.

- **Context management interface**

  The web interface is developed using React 18.2.0 and NodeJS 18.14.2, and across the chapter there is an overview of the atomic components and feature panels composing the application, with real illustrations references.

- **System containerization**

  The decision was made to utilize Docker for containerization in order to bundle and deploy all the different parts of the project. Containerization involves packaging an application and its dependencies into standalone and lightweight entities known as containers. Docker serves as a platform that facilitates the creation and administration of these containers and this approach brings forth several advantages including enhanced mobility, isolation, effective management of dependencies, scalability, efficient utilization of resources and the ability to control versions. By adopting containerization, significant time savings can be achieved when deploying the entire system as a unified instance in any environment.

The system validation process is a crucial step in the development lifecycle of any software project as it involves testing and evaluating the system to ensure that it meets the specified requirements, functions as intended and satisfies the needs of its use cases scenarios. The next chapter 6 focuses on the various aspects of the system validation process to make sure it delivers a reliable and high-quality Smart-City Digital Twin (SCDT) environment.

Chapter 6

Validation

This validation process aims to ensure the correctness and effectiveness of the system as a unit from the user interaction perspective. The validation is divided into a list of use cases that need to succeed, covering various aspects of the system's functionality. This process employs two main testing approaches using *Postman*[1] for the API validation and the developed web interface for all the usage scenarios.

Note: for some API responses, there was the need to present the actual truncated JSONs, so it become possible to visualize the validation evidences, since screenshotting these from Postman would not be possible due to the lengthy nature of their response bodies.

6.1 EUC: Entity Use Cases

6.1.1 Entity creation

Description: validate the ability to create entities within the system.

The creation of an entity process can be validated either via an API call or by direct use of the web interface. Using Postman to call `/entity/create` with the entity instance inside the request body it is possible to validate if it is properly created and stored. Not only the creation of an entity is required but also the synchronous storage of its replica. Along with these two processes, the respective subscriptions should be set to make the Context Broker (CB) and Cygnus aware of any context update. Below, in figure 6.1 is the entity creation request made to the API for this validation step and after it, in the listing 6.1 can be validated the proper response.

[1] A user-friendly powerful tool for interacting with APIs, allowing developers to test API endpoints, customize requests, organize them into collections and automate workflows.

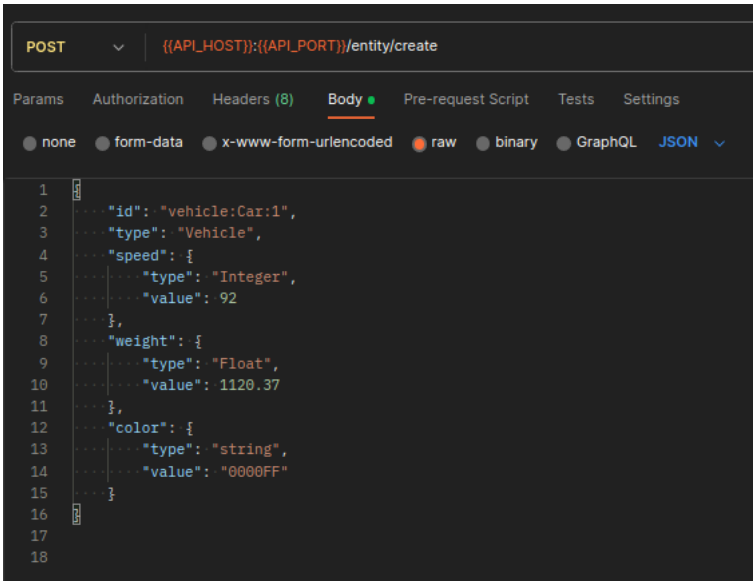


Figure 6.1: EUC entity creation validation via API request

```
{
    "entityResult": {
    "correlator": "ec1de4d2-11ce-11ee-a164-0242ac140005",
    "entity": {
        "id": "vehicle:Car:1",
        "type": "Vehicle",
        "speed": {"type": "Integer", "value": 92},
        ...
    },
    "location": "/v2/entities/vehicle:Car:1?type=Vehicle"
    },
    "entitySubscriptionResult": {
        "correlator": "ec1f8292-11ce-11ee-9e1f-0242ac140005",
        "subscription": {
            "description": "vehicle:Car:1 updates listener",
            "subject": {...},
            "notification": {...},
            "expires": "2073-01-01T00:00:00.000Z",
            "id": "6495a6832a4c0a26c704335b"
        },
        "location": "/v2/subscriptions/6495a6832a4c0a26c704335b"
    },
    "dummyResult": {
    "correlator": "ec1eb9d4-11ce-11ee-9ac4-0242ac140005",
    "entity": {
        "id": "vehicle:Car:1:dummy",
        "type": "Vehicle",
        "speed": {"type": "Integer", "value": 92},
        ...
        },
        "location": "/v2/entities/vehicle:Car:1:dummy?type=Vehicle"
    },
```

```
    "dummySubscriptionResult": {
        "correlator": "ec1f8300-11ce-11ee-87df-0242ac140005",
        "subscription": {
            "description": "vehicle:Car:1:dummy updates listener",
            "subject": {...},
            "notification": {...},
            "expires": "2073-01-01T00:00:00.000Z",
            "id": "6495a6832a4c0a26c704335c"
        },
        "location": "/v2/subscriptions/6495a6832a4c0a26c704335c"
    },
    "message": "Both entity and its repetition dummy were successfully created.
        Cygnus is now listening to their changes."
}
```

Listing 6.1: EUC entity creation validation via API response

After performing the service-side validation, it is feasible to verify in figure 6.2 that the entity `vehicle:Car:1` appears in the interface list of entities as a consequence of this previous API call, which validates the use case scenario.

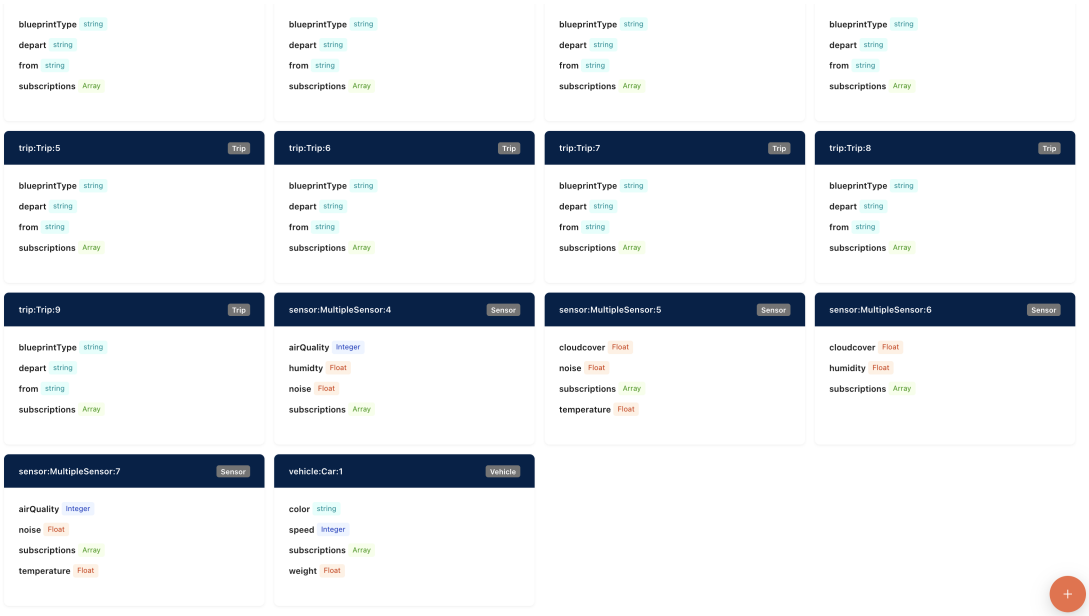


Figure 6.2: EUC entity creation validation via API request aftermath

Upon clicking on the action button situated in the bottom right corner of the screen, a modal window is expected to appear. This modal serves a form, represented in figure 6.3, where the user can input all the necessary details for creating a new entity instance. By filling out the form and completing the creation process, a confirmation modal is then displayed to indicate the outcome of this creation process which explicitly states whether it was successful or unsuccessful. The validation evidence can be analyzed below in figure 6.4.

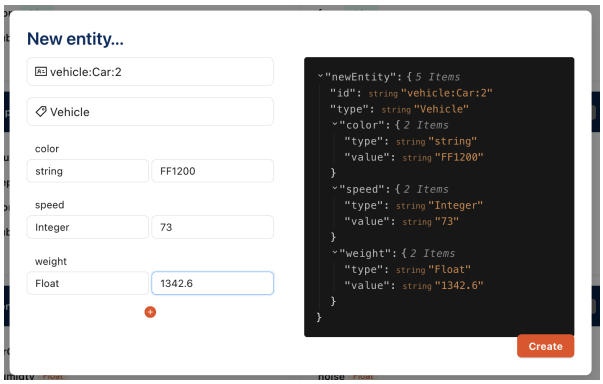


Figure 6.3: EUC entity creation validation via interface form

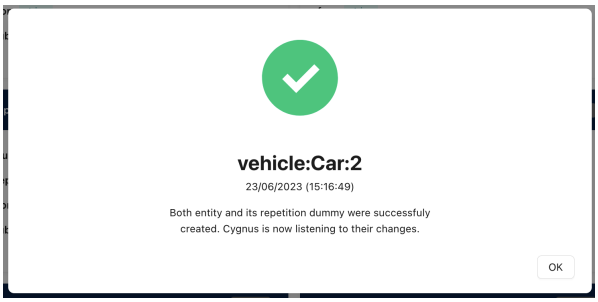


Figure 6.4: EUC entity creation validation via interface success modal

After presenting the success confirmation modal, the interface entity list shows the recently created entity as it is possible to see in figure 6.5, which validates the proper entity creation.

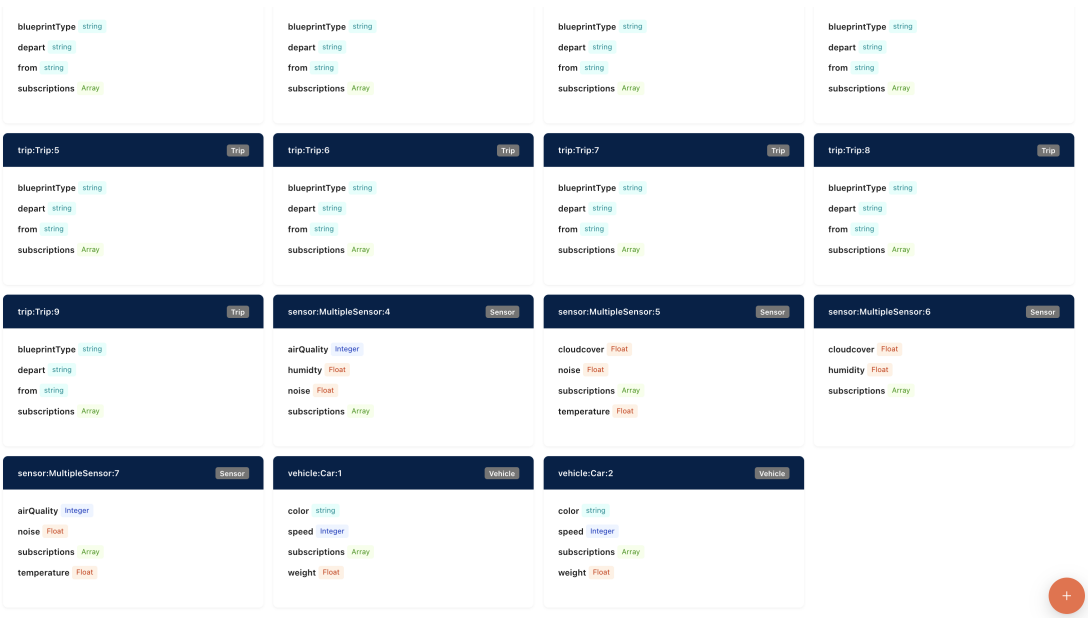


Figure 6.5: EUC entity creation validation via interface aftermath

6.1.2 Entity deletion

Description: verify the capability to delete a specific entity.

The deletion of an entity can only be accomplished through an API call. By making a request to `/entity/delete`, it is possible to validate if the entity is successfully removed from the system. Similar to the creation process, it is essential to ensure the synchronous deletion of the entity's replica, checking these deletions correlators. The request made to the API during this step is presented below in figure 6.6, its expected response can be seen in figure 6.7 and at last the deletion consequence in the interface entity list is validated in figure 6.8.

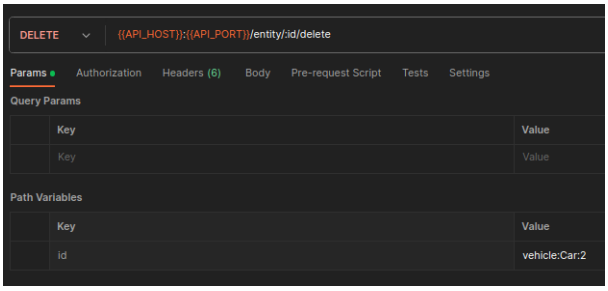


Figure 6.6: EUC entity deletion validation via API request

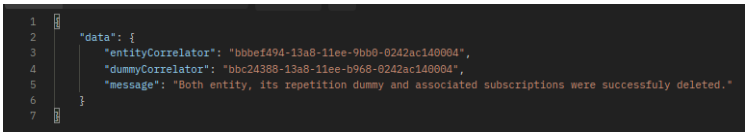


Figure 6.7: EUC entity deletion validation via API response

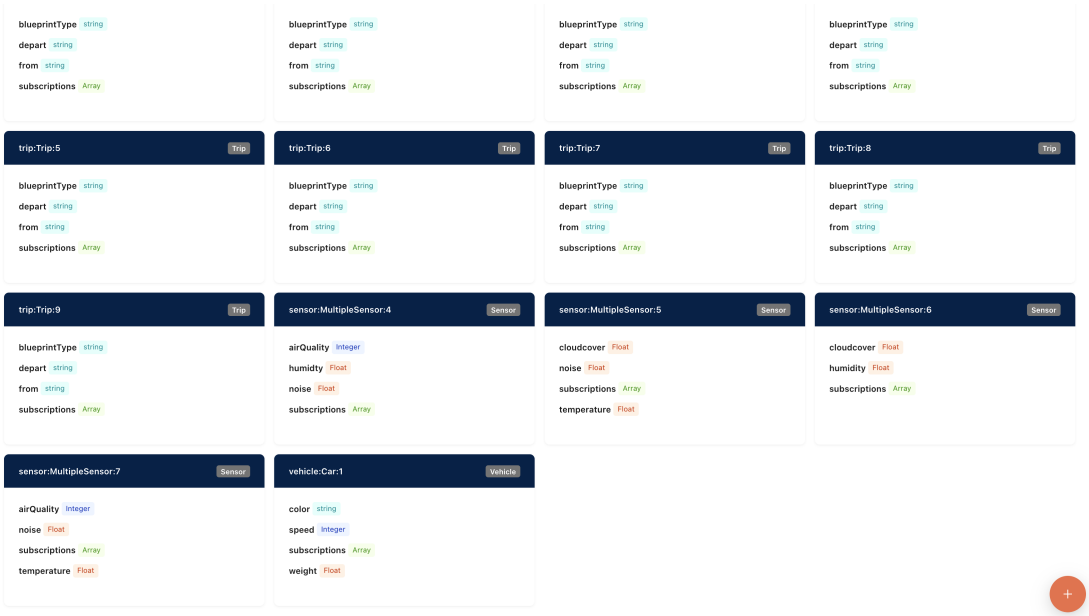


Figure 6.8: EUC entity deletion validation via interface aftermath

6.1.3 Entity retrieval

Description: validate the retrieval of a specific entity.

To ensure the successful retrieval of an entity based on its identifier, there are two methods that can be utilized, either Postman for direct API verification or using the web interface itself. By making a request to the endpoint `/entity/id`, where `id` represents the specific entity identifier, it becomes possible to validate whether the returned data aligns with the expected results. For instance, if we consider the entity identifier `vehicle:Car:1` and incorporate it as the path variable for the identifier, as shown in figure 6.9, the expectation is that the endpoint will return the current contextual data associated with that particular entity, which is corroborated by the evidence in 6.10.

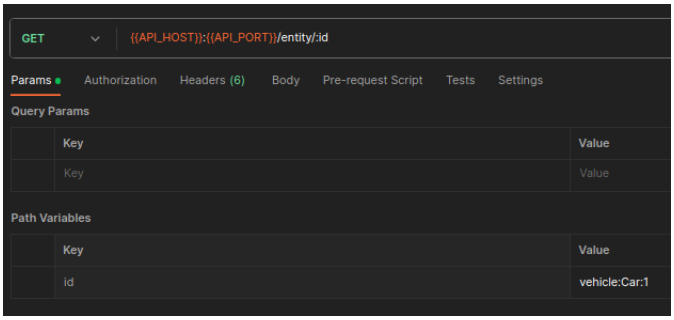


Figure 6.9: EUC entity retrieval validation via API request

Figure 6.10: EUC entity retrieval validation via API response

By utilizing the context visualization interface, it is feasible to validate the entity retrieval process through routing. This can be accomplished by passing the entity identifier, like the following figure 6.11 demonstrates and just as it is done when invoking the entity retrieval endpoint. As a result, the web application will redirect the user to the corresponding entity instance page, where he can conveniently visualize the contextual data associated with the entity, as it is represented in figure 6.12.

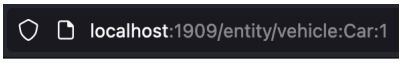


Figure 6.11: EUC entity retrieval validation via interface routing

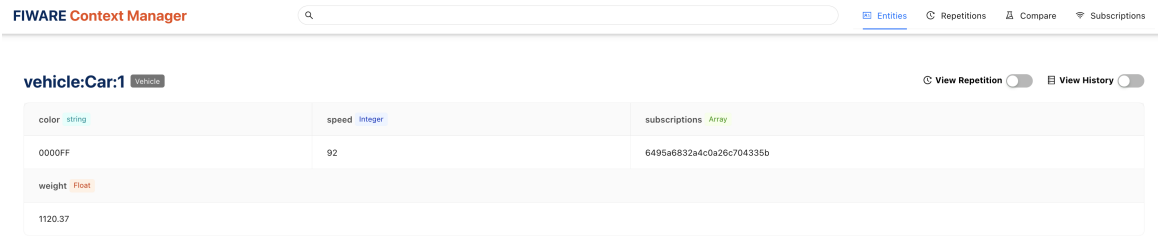


Figure 6.12: EUC entity retrieval validation via interface page

6.1.4 Entity listing

Description: check the ability to list all entities within the system.

To confirm the accurate listing of entities within the system, two approaches can be employed either using Postman for direct API verification or the web interface itself. By sending a request to the `/entity/list` endpoint like presented in below figures 6.13, 6.14, 6.15 and 6.16, it becomes feasible to validate that the returned data corresponds to the expected results as shown respectively in the listings 6.2, 6.3, 6.4 and 6.5.

It's worth noting that the listing feature offers four filter variants that allow users to customize their desired selection:

- **List all**, which displays all entities in the system without any filtering;
- **List all except for dummy replicas**, so entities that are considered dummy replicas should be excluded from the listing;
- **List the ones matching an id pattern**, allows users to specify a pattern or criteria for entity identification providing a more filtered view;
- **List all considering specific attributes**, lets users to specify certain attributes that the listed entities must possess.

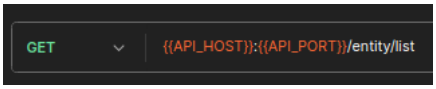


Figure 6.13: EUC entity listing validation via API request


```
{
    "results": [
        {
            "id": "sensor:MultipleSensor:1",
            "type": "Sensor",
            "subscriptions": {
                "type": "Array",
                "value": ["643c1acccc3064b16770384dc"],
            },
            "temperature": {"type": "Float", "value": 13.4},
            "windspeed": {"type": "Float", "value": 2.2}
        },
        {
            "id": "sensor:MultipleSensor:2",
            "type": "Sensor",
            "subscriptions": {
                "type": "Array",
                "value": ["64456699915855090a06657c"],
            },
            "temperature": {"type": "Float", "value": 30.1},
            "windspeed": {"type": "Integer", "value": 12.33}
        },
        ...
        {
            "id": "trip:Trip:1:dummy",
            "type": "Trip"
            "subscriptions": {
                "type": "Array",
                "value": ["82356699915855090a03332a"],
            },
            "depart": {"type": "Float", "value": 0.00},
            "from": {"type": "Float", "value": -255.2},
            "to": {"type": "Float", "value": -105.7}
        }
    ],
    "numOfEntitiesFound": 36
}
```


Listing 6.2: EUC entity listing validation via API response

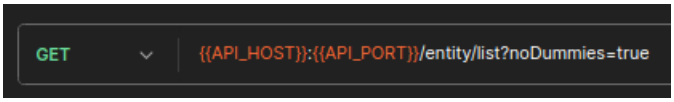


Figure 6.14: EUC entity listing with no dummies validation via API request

```
{
    "results": [
        {
            "id": "sensor:MultipleSensor:1",
            "type": "Sensor",
            "subscriptions": {
                "type": "Array",
                "value": ["643c1accc3064b16770384dc"],
            },
            "temperature": {"type": "Float", "value": 13.4},
            "windspeed": {"type": "Float", "value": 2.2}
        },
        {
            "id": "sensor:MultipleSensor:2",
            "type": "Sensor",
            "subscriptions": {
                "type": "Array",
                "value": ["64456699915855090a06657c"],
            },
            "temperature": {"type": "Float", "value": 30.1},
            "windspeed": {"type": "Integer", "value": 12.33}
        },
        ...
        {
            "id": "trip:Trip:1",
            "type": "Trip"
            "subscriptions": {
                "type": "Array",
                "value": ["82356699915855090a03332a"],
            },
            "depart": {"type": "Float", "value": 0.00},
            "from": {"type": "Float", "value": -255.2},
            "to": {"type": "Float", "value": -105.7}
        },
    ],
    "numOfEntitiesFound": 18
}
```

Listing 6.3: EUC entity listing with no dummies validation via API response

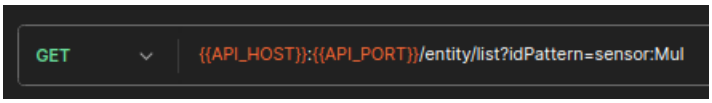


Figure 6.15: EUC entity listing with id pattern matching validation via API request


```
{
    "results": [
        {
            "id": "sensor:MultipleSensor:1",
            "type": "Sensor",
            "subscriptions": {
                "type": "Array",
                    "value": ["643c1accc3064b16770384dc"]
                },
            "temperature": {"type": "Float", "value": 13.4},
            "windspeed": {"type": "Float", "value": 2.2}
        },
        {
            "id": "sensor:MultipleSensor:1:dummy",
            "type": "Sensor",
            "subscriptions": {
                "type": "Array",
                "value": ["64456699915855090a06657c"],
                "metadata": {}
            },
            "temperature": {"type": "Float", "value": 13.4,},
            "windspeed": {"type": "Integer", "value": 2.2}
        },
        ...
        {
            "id": "sensor:MultipleSensor:7:dummy",
            "type": "Sensor",
            "subscriptions": {
                "type": "Array",
                "value": ["64856699915866090a088773"],
                "metadata": {}
            },
            "temperature": {"type": "Float", "value": 17.34},
            "windspeed": {"type": "Integer", "value": 32.9}
        }
    ],
    "numOfEntitiesFound": 14
}
```


Listing 6.4: EUC entity listing with id pattern matching validation via API response

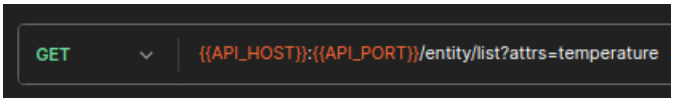


Figure 6.16: EUC entity listing with attribute filter validation via API request


```
{
    "results": [
        {
            "id": "sensor:MultipleSensor:1",
            "type": "Sensor",
            "subscriptions": {
                "type": "Array",
                "value": ["643c1accc3064b16770384dc"]
            },
            "temperature": {"type": "Float", "value": 13.4},
        },
        {
            "id": "sensor:MultipleSensor:1:dummy",
            "type": "Sensor",
            "subscriptions": {
                "type": "Array",
                "value": ["64456699915855090a06657c"]
            },
            "temperature": {"type": "Float", "value": 13.4},
        },
        ...
        {
            "id": "trip:Trip:1",
            "type": "Trip"
        },
        {
            "id": "trip:Trip:1:dummy",
            "type": "Trip"
        }
    ],
    "numOfEntitiesFound": 36
}
```


Listing 6.5: EUC entity listing with attribute filter validation via API response

By navigating to the entity list view in the web interface, users can access the list of system entities obtained by invoking the NGSI Service with the filter `noDummies=true`. This parameter ensures that only the original entities within the system are presented, excluding any simulation replica. The below figure 6.17 is the evidence of this visual validation process.

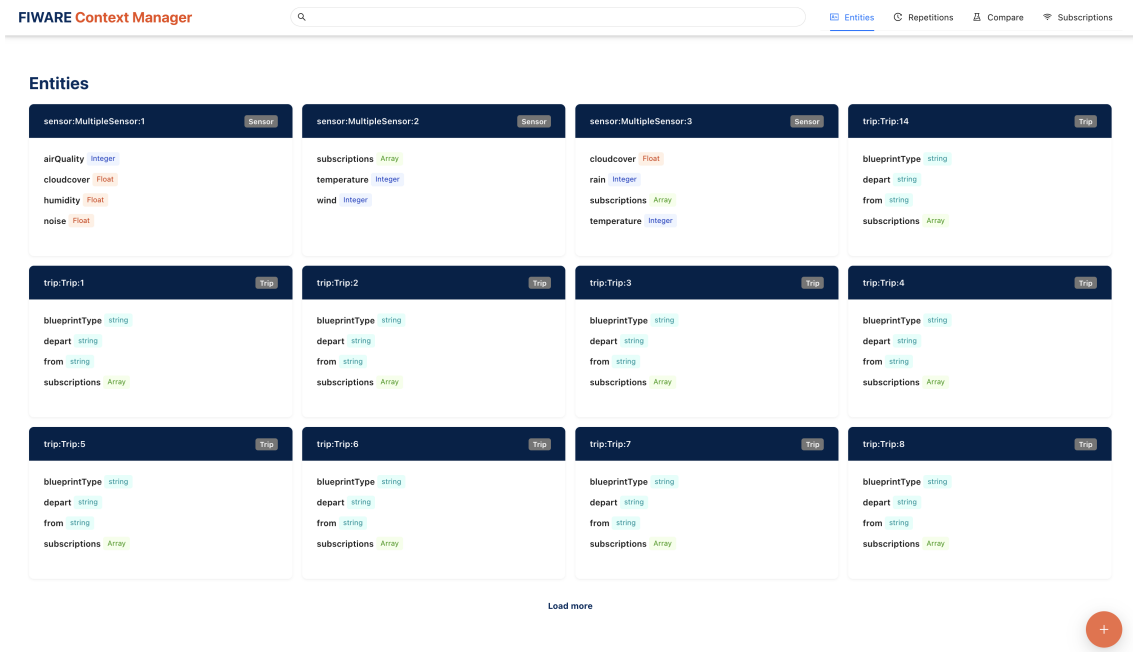


Figure 6.17: EUC entity listing validation via interface

6.1.5 Entity context update

Description: verify the ability to update the context of a specific entity.

To verify the functionality of the entity update feature, one can directly call the API with a new context. By making a request to the `/entity/update` endpoint and providing the entity's id reference along with the desired changes to be applied inside the request body, as shown below in figure 6.18, it becomes possible to validate that the returned data aligns with the expected results as shown in figure 6.19.

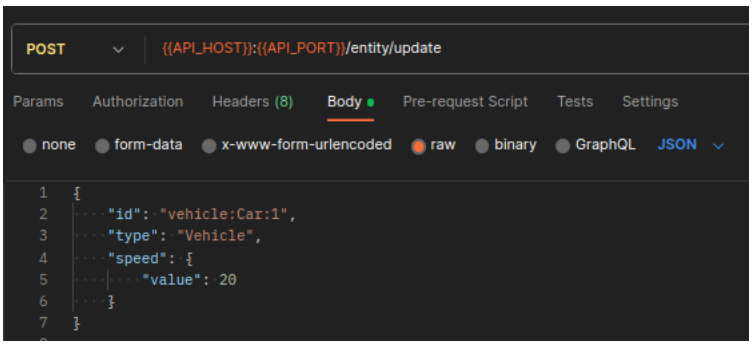


Figure 6.18: EUC entity update validation via API request

```
{
    "data": {
        "correlator": "2804e844-11d2-11ee-b005-0242ac140005",
        "changes": {
            "speed": {
                "value": 20
            }
        },
        "message": "Entity successfully updated."
    }
}
```


Figure 6.19: EUC entity update validation via API response

After this process, if we send a request for the same entity instance, it is possible to check in figure 6.20, that its speed property value got updated as expected, which validates the use case.


```
{
    "correlator": "3cf97f76-11d2-11ee-877c-0242ac140005",
    "entity": {
        "id": "vehicle:Car:1",
        "type": "Vehicle",
        "color": {
            "type": "string",
            "value": "0000FF",
            "metadata": {}
        },
        "speed": {
            "type": "Number",
            "value": 20,
            "metadata": {}
        },
        "subscriptions": {
            "type": "Array",
            "value": [
                "6495a6832a4c0a26c704335b"
            ],
            "metadata": {}
        },
        "weight": {
            "type": "Float",
            "value": 1120.37,
            "metadata": {}
        }
    }
}
```


Figure 6.20: EUC entity update validation via API aftermath

6.2 HUC: History Use Cases

6.2.1 Entity history retrieval

Description: validate the retrieval of historical context data of a specific entity.

To ensure the accurate retrieval of persisted data for an entity, there are two validation methods available, by using Postman to call the API directly or checking the historical view mode in the entity view of the context visualization interface.

For the API validation approach, it is possible to utilize Postman to send a request to the `/history/entity/id` endpoint, where the entity's identification is provided in the path

variable `id` like the following figures 6.21, 6.22, 6.23 and 6.24 suggest. By doing so, the response data from the call can be examined to verify in listings 6.6, 6.7, 6.8 and 6.9, it aligns with the expected result. This method allows for a direct and programmatic validation of the history data retrieval.

It is important to highlight that the history retrieval feature provides users with four distinct filter options to customize the most wanted selection:

- **Retrieve all recorded history**, allows users to retrieve the complete recorded history for an entity;
- **Retrieve recorded history between dates**, gives the flexibility to specify a specific date range between which to look for the history data;
- **Retrieve recorded history of specific attributes**, enables users to narrow down the history retrieval based on specific attributes associated with the entity;
- **Retrieve recorded history limiting the number of results**, control the number of historical data records returned by specifying a maximum limit.

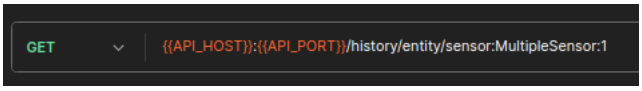


Figure 6.21: HUC entity history validation via API request

```
{
    "results": [
        {
            "attrName": "airQuality",
            "attrValue": "12",
            "attrType": "Integer",
            "recvTime": "2023-04-16 15:57:00.703"
        },
        {
            "attrName": "precipitation",
            "attrValue": "70.22",
            "attrType": "Float",
            "recvTime": "2023-04-16 15:57:00.703"
        },
        {
            "attrName": "noise",
            "attrValue": "90.13",
            "attrType": "Float",
            "recvTime": "2023-04-16 15:57:00.703"
        },
        ...
        {
        "attrName": "cloudcover",
```

```
        "attrValue": "50.555",
        "attrType": "Float",
        "recvTime": "2023-05-29 22:28:16.967"
        }
    ]
}
```

Listing 6.6: HUC entity history validation via API response

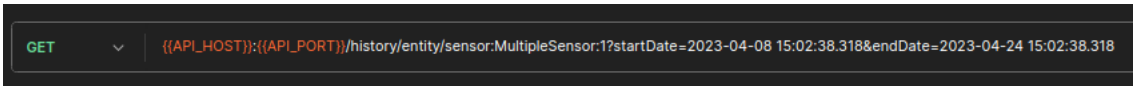


Figure 6.22: HUC entity history within date interval validation via API request

```
{
    "results": [
        {
            "attrName": "airQuality",
            "attrValue": "12",
            "attrType": "Integer",
            "recvTime": "2023-04-16 15:57:00.703"
        },
        {
            "attrName": "precipitation",
            "attrValue": "70.22",
            "attrType": "Float",
            "recvTime": "2023-04-16 15:57:00.703"
        },
        ...
        {
            "attrName": "cloudcover",
            "attrValue": "50.555",
            "attrType": "Float",
            "recvTime": "2023-04-16 18:12:57.49"
        }
    ]
}
```

Listing 6.7: HUC entity history within date interval validation via API response

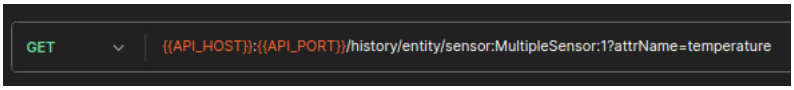


Figure 6.23: HUC entity history with attribute filter validation via API request


```
{
    "results": [
        {
            "attrName": "temperature",
            "attrValue": "13.4",
            "attrType": "Float",
            "recvTime": "2023-04-16 15:57:00.703"
        },
        {
            "attrName": "temperature",
            "attrValue": "10.2",
            "attrType": "Float",
            "recvTime": "2023-04-16 18:12:57.49"
        },
        {
            "attrName": "temperature",
            "attrValue": "25.8",
            "attrType": "Float",
            "recvTime": "2023-05-09 20:39:32.990"
        },
        {
            "attrName": "temperature",
            "attrValue": "13.4",
            "attrType": "Float",
            "recvTime": "2023-05-29 22:28:16.967"
        }
    ]
}
```


Listing 6.8: HUC entity history with attribute filter validation via API response

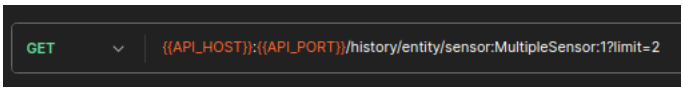


Figure 6.24: HUC entity history with entries limit validation via API request


```
{
    "results": [
        {
            "attrName": "airQuality",
            "attrValue": "12",
            "attrType": "Integer",
```

```
            "recvTime": "2023-04-16 15:57:00.703"
        },
        {
            "attrName": "precipitation",
            "attrValue": "70.22",
            "attrType": "Float",
            "recvTime": "2023-04-16 15:57:00.703"
        }
    ]
}
```

Listing 6.9: HUC entity history with entries limit validation via API response

6.2.2 Repetition Start

This particular feature holds significant importance within the overall solution as it involves recreating a simulation context with the ability to make modifications to it. Ensuring the precise management and processing of this repetition start is crucial, and to do so, there are two distinct ways available to facilitate the process, either making use of Postman to directly invoke the API or accessing the repetition start modal window within the web application.

There are three options available as starting points when initiating a repetition:

- **From the system current context**, allows users to begin a new repetition from the present context;
- **From the context of a previous repetition**, allows users to start the repetition from the context of a previous one;
- **From the context on a past date**, makes possible to initiate the repetition from a specific context that took place on a past date.

Note: new repetitions identifiers are sequenced as below because the validation process was made for each starting point option for the Postman calls first and only then for each one of them using the web application.

6.2.2.1 From current context

Description: validate the starting point of a context simulation based on the current context.

By calling directly the `/history/repetition` endpoint, passing the context changes via the request body, like presented below in figure 6.25, it is possible to validate the success status from the server operation as shown after in figure 6.26.

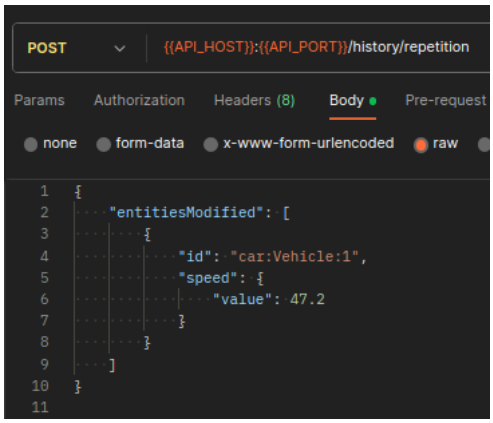


Figure 6.25: HUC repetition start from current context validation via API request

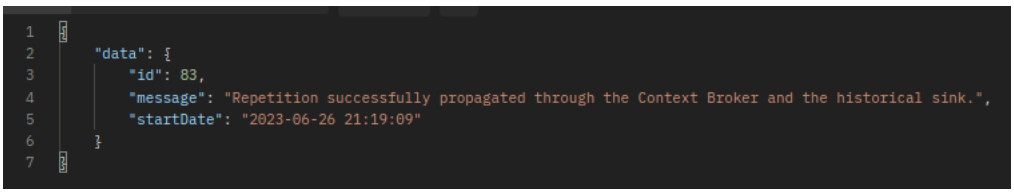


Figure 6.26: HUC repetition start from current context validation via API response

When the user clicks on the action button located in the bottom right corner of the repetition listing screen, it triggers the display of a modal window represented in figure 6.27. This modal serves as a form where is possible to enter all the required information to initiate a new repetition instance. Additionally, the form includes a field for pre-visualization of the request body. The success modal evidence of this validation step is shown below in figure 6.28.

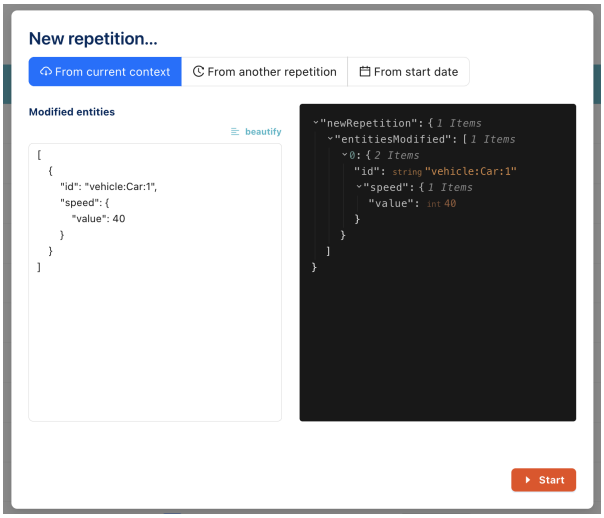


Figure 6.27: HUC repetition start from current context validation via interface form

Figure 6.28: HUC repetition start from current context validation via interface success modal

6.2.2.2 From past repetition

Description: validate the starting point of a context simulation based on the configuration of a past one.

Similarly, by following a similar approach as described above, users can validate the expected success status of the server operation, as in figure 6.30, by making the same API call but including the reference past repetition index within the request body as below in figure 6.29.

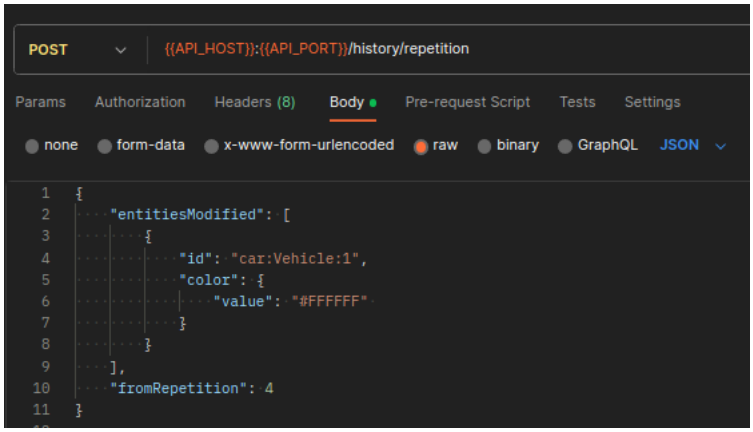


Figure 6.29: HUC repetition start from past repetition validation via API request

Figure 6.30: HUC repetition start from past repetition validation via API response

In the web application flow, a similar process can be applied with a slight difference. Instead

of initiating it from the current context tab, there is the option to select a different starting point. Specifically, it is possible to choose to start the repetition from another previously created repetition, identified by its index as in figure 6.31. The success modal evidence can be affirmed after it with figure 6.32.

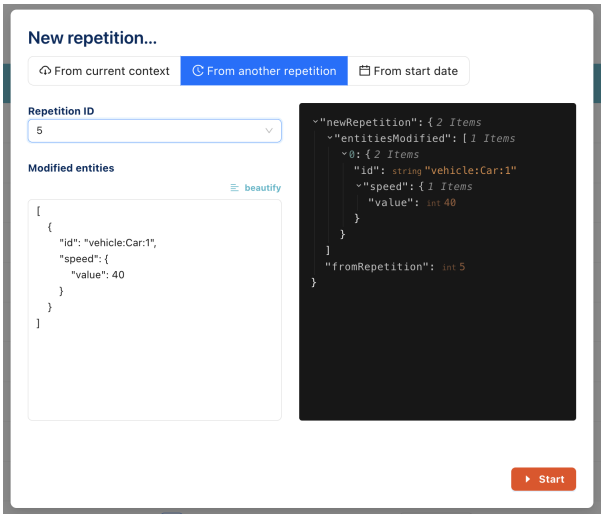


Figure 6.31: HUC repetition start from past repetition validation via interface form

Figure 6.32: HUC repetition start from past repetition validation via interface success modal

6.2.2.3 From start date

Description: validate the starting point of a context simulation based on a starting date.

By calling directly the /history/repetition endpoint, passing the context changes and the past date starting point in `startDate` via the request body as in figure 6.33, it is possible to validate the expected success status from the server operation as shown below in figure 6.34.

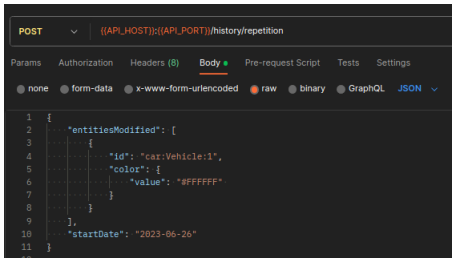


Figure 6.33: HUC repetition start from past date validation via API request

Figure 6.34: HUC repetition start from past date validation via API response

Within the previously outlined flow of the web application, users are granted the flexibility to initiate a repetition from a distinct starting point by utilizing the last option tab. Rather than initiating the repetition based on the default options presented in the first two tabs, it is possible to select a specific starting date which allows them to establish the context of their new repetition based on the data available at the chosen date as suggested below in figure 6.35. The next figure 6.36 shows the validation evidence that proves the flux correctness.

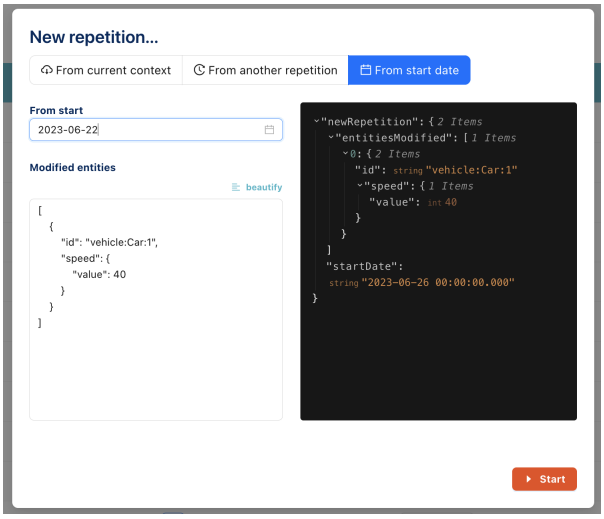


Figure 6.35: HUC repetition start from past date validation via interface form

Figure 6.36: HUC repetition start from past date validation via interface success modal

6.2.3 Repetition end

Description: validate the termination of a specific repetition.

To verify the effective completion of a given repetition, it is possible to directly call the API. By using the `/history/repetition/end` endpoint and providing the repetition index inside the request body, users can verify the repetition context is successfully terminated and updated, like the figure 6.37 suggests.

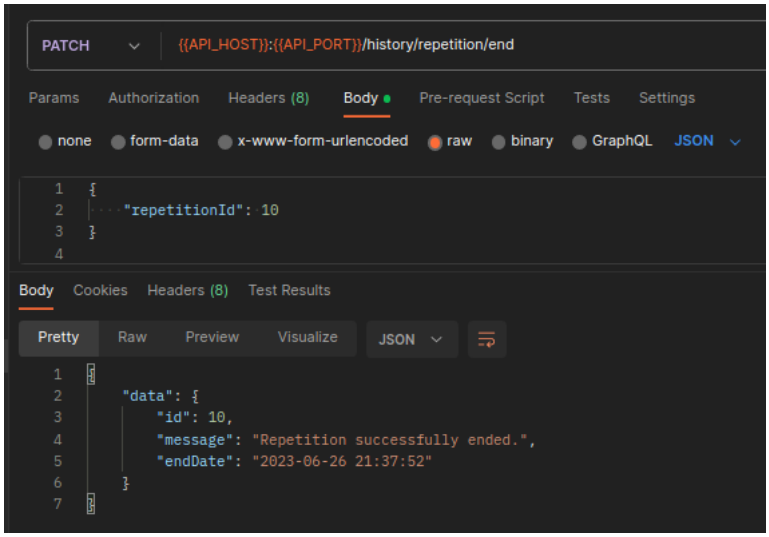


Figure 6.37: HUC repetition end validation via API

6.2.4 Repetition listing

Description: validate the listing of all the repetitive simulations within the system.

To ensure the accurate listing of initiated repetitions within the system, users have two options for verification. The first method involves using Postman to call the API directly, while the second method entails checking the repetition listing view within the context visualization interface.

By making a request to the `/history/repetition/list` endpoint, like represented below in figure 6.38, users can easily verify that the persisted repetitions are correctly listed. This API call retrieves the necessary data and provides a response that includes the list of repetitions stored in the system as shown in the listing 6.10.

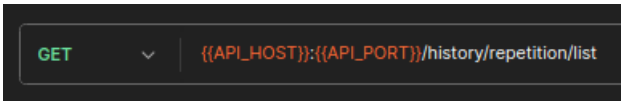


Figure 6.38: HUC repetition listing validation via API request

```
{
    "results": [
        {
            "id": 1,
            "startDate": "2023-04-16T15:03:09.000Z",
            "endDate": "2023-05-30T15:36:41.000Z"
        },
        {
            "id": 2,
            "startDate": "2023-04-16T16:50:51.000Z",
            "endDate": "2023-05-30T15:36:50.000Z"
        },
        {
            "id": 3,
            "startDate": "2023-04-16T16:51:56.000Z",
            "endDate": "2023-05-30T15:36:53.000Z"
        },
        {
            "id": 4,
            "startDate": "2023-04-16T16:52:05.000Z",
            "endDate": "2023-05-30T15:36:57.000Z"
        },
        {
            "id": 5,
            "startDate": "2023-04-16T16:52:26.000Z",
            "endDate": "2023-05-30T15:37:01.000Z"
        },
        {
            "id": 6,
            "startDate": "2023-04-16T16:55:53.000Z",
            "endDate": null
        },
        ...
    ]
}
```

Listing 6.10: HUC repetition listing validation via API response

When accessing the repetitions list view within the web interface, the displayed content should mirror the data returned by the API. The view is designed to present the list of repetitions in a user-friendly format, providing the same information that is obtained above. This is properly reflected in the figure 6.39 below.

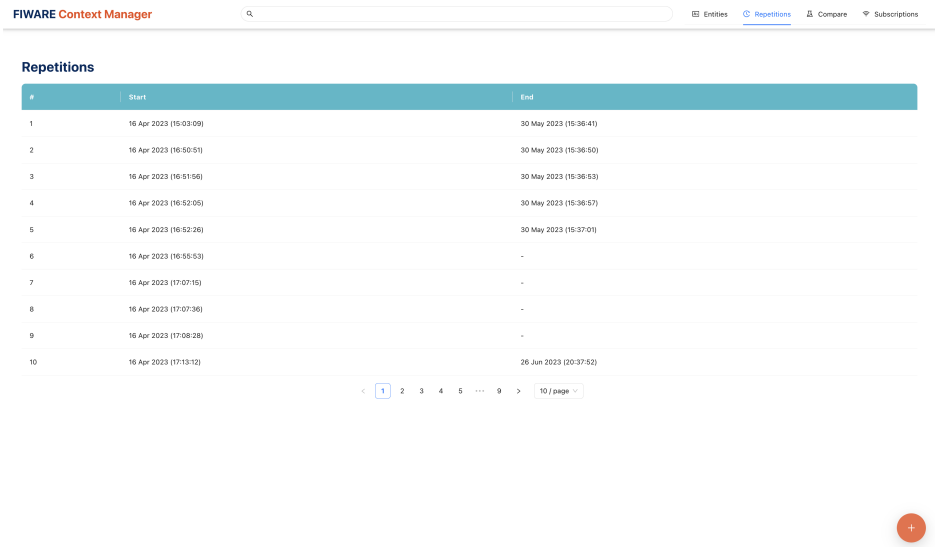


Figure 6.39: HUC repetition listing validation via interface

6.3 SUC: Subscription Use Cases

6.3.1 Subscription creation

Description: validate the creation of a context based subscription.

To ensure the accurate and proper functionality of the subscription creation, it is possible to perform a direct API call. By sending a request to the `/subscription/create` endpoint and providing the necessary subscription subject and notification details within the request body like suggested in figure 6.40, one can validate the expected resulting status of the creation process as in the listing 6.11.

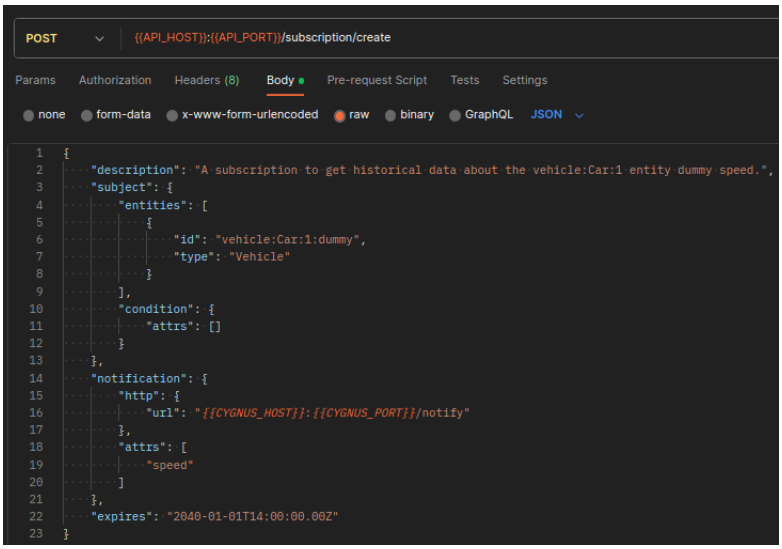


Figure 6.40: SUC subscription creation validation via API request

```
{
    "data": {
        "correlator": "b5b429ee-146f-11ee-b584-0242ac140005",
        "subscription": {
            "description": "A subscription to get historical data about the vehicle:
                Car:1 entity dummy speed.",
            "subject": {
                "entities": [
                    {
                        "id": "vehicle:Car:1:dummy",
                        "type": "Vehicle"
                    }
                ],
                "condition": {"attrs": []}
            },
            "notification": {
                "http": {"url": "http://cygnus:5050/notify"},
                "attrs": ["speed"]
            },
            "expires": "2040-01-01T14:00:00.00Z",
            "id": "649a0f432a4142f68b05180b"
        },
        "location": "/v2/subscriptions/649a0f432a4142f68b05180b",
        "message": "Subscription created successfully."
    }
}
```

Listing 6.11: SUC subscription creation validation via API response

6.3.2 Subscription deletion

Description: check the ability to delete a specific subscription.

The ability to delete a subscription is also a functionality that is limited to the server-side. To validate this feature, users can make use of the API by sending a request to the `/subscription/id/delete` endpoint, where the path variable `id` corresponds to the specific subscription that needs to be deleted. Upon successful deletion, the service will provide an operation correlator as a response, which confirms the deletion, as represented below in figure 6.41. In the event of any errors or failures during the deletion process, the service would return an error message instead.

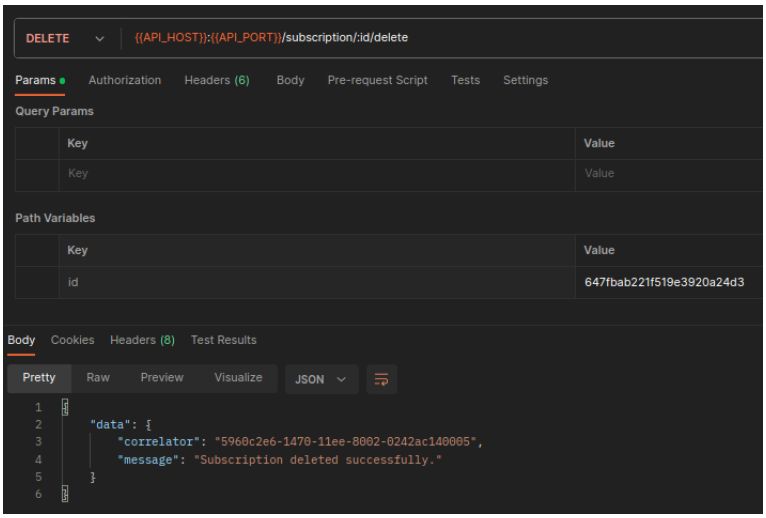


Figure 6.41: SUC subscription deletion validation via API

6.3.3 Subscription update

Description: verify the ability to update details of a specific subscription.

Similarly to how users can manually update entities, the system also offers the capability to update subscriptions. To validate this functionality, users can utilize the `/subscription/update` endpoint by sending a request with the subscription identification reference and the desired context updates. If the update is successful, the service will respond with an operation correlator, like suggested in figure 6.42, confirming the updated details of the subscription. However, if any issues arise during the update process, the service will return an error message instead.

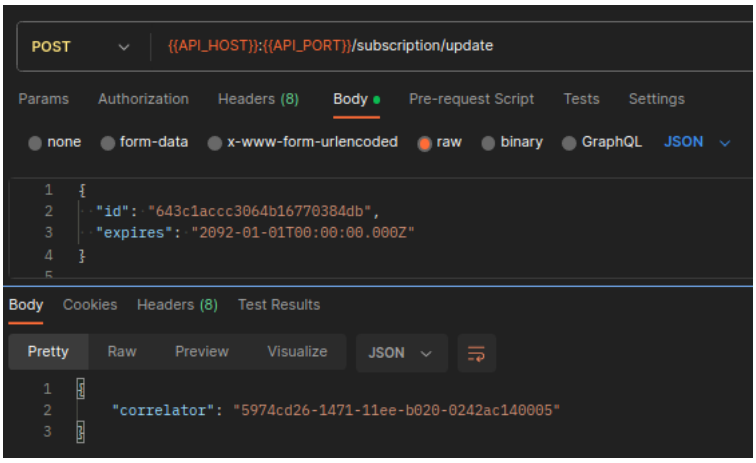


Figure 6.42: SUC subscription update validation via API

6.3.4 Subscription listing

Description: validate the listing of all the subscriptions within the system.

To validate the correct retrieval of registered subscriptions, users have two options at their disposal: direct API testing and utilizing the web application interface.

By making a request to the `/subscription/list` endpoint, as shown in figure 6.43, users can examine the resulting response data to ensure its alignment with the expected subscription results like presented in the listing 6.12. This approach provides a straightforward mean of validating the retrieval of subscription data including both original and dummy replica update listeners.

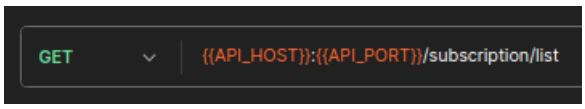


Figure 6.43: SUC subscription listing validation via API request

```
{
    "results": [
        {
            "id": "643c1accc3064b16770384db",
            "description": "sensor:MultipleSensor:1:dummy updates listener",
            "expires": "2092-01-01T00:00:00.000Z",
            "status": "active",
            "subject": {
                "entities": [
                    {
                        "id": "sensor:MultipleSensor:1:dummy",
                        "type": "Sensor"
                    }
                ],
```

```
                "condition": {"attrs": []}
            },
            "notification": {
                "timesSent": 89,
                "lastNotification": "2023-06-26T21:30:04.000Z",
                "attrs": [
                    "airQuality",
                    "humidity",
                    ...
                    "repetition"
                ],
                "onlyChangedAttrs": false,
                "attrsFormat": "normalized",
                "http": {"url": "http://cygnus:5050/notify"},
                "lastSuccess": "2023-06-26T21:30:04.000Z",
                "lastSuccessCode": 200,
                "covered": false
            }
        },
        ...
        {
            "id": "649a0f432a4142f68b05180b",
            "description": "A subscription to get historical data about the vehicle:
                Car:1 entity dummy speed.",
            "expires": "2040-01-01T14:00:00.000Z",
            "status": "active",
            "subject": {
                "entities": [
                    {"id": "vehicle:Car:1:dummy", "type": "Vehicle"}
                ],
            "condition": {"attrs": []}
            },
            "notification": {
                "attrs": ["speed"],
                "onlyChangedAttrs": false,
                "attrsFormat": "normalized",
                "http": {"url": "http://cygnus:5050/notify"},
                "covered": false
            }
        }
    ]
}
```


Listing 6.12: SUC subscription listing validation via API response

Upon accessing the subscriptions listing view in the interface, users can expect to see the content that accurately reflects the data returned by the API, as the figure 6.42 presents. This view is specifically designed to display the list of subscriptions registered within the system, providing concise descriptions of their purpose and indicating their respective expiration dates.

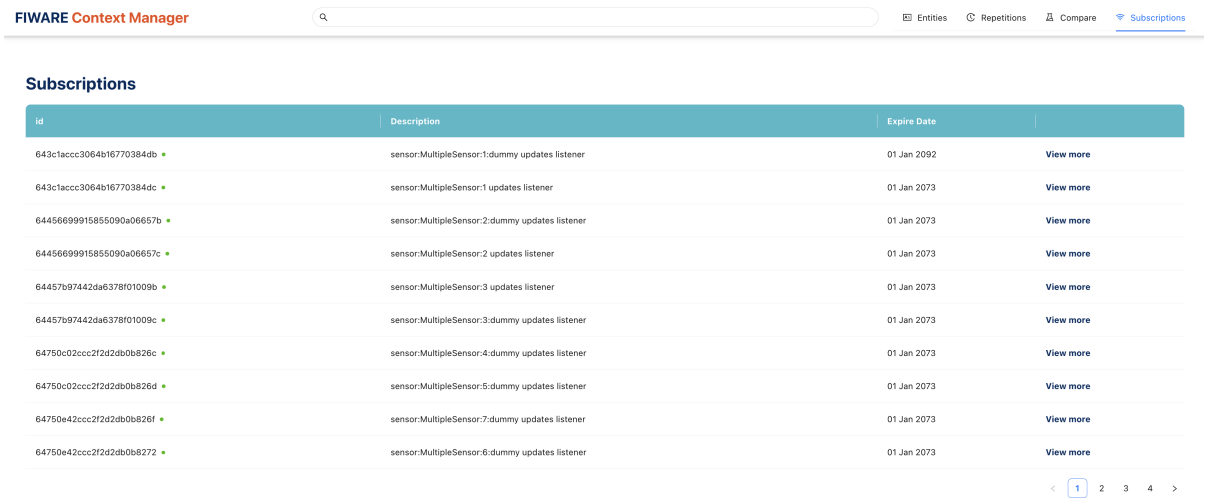


Figure 6.44: SUC subscription listing validation via interface

Upon clicking the “View more“ button on a subscription table entry, the web application redirects the user to that subscription details view, illustrated in the following figure 6.42. This section functions as an administrative view where users can control all relevant information regarding the subscription and notification relationship. Its primary purpose is to ensure that the communication of contextual updates is effectively and accurately executed.

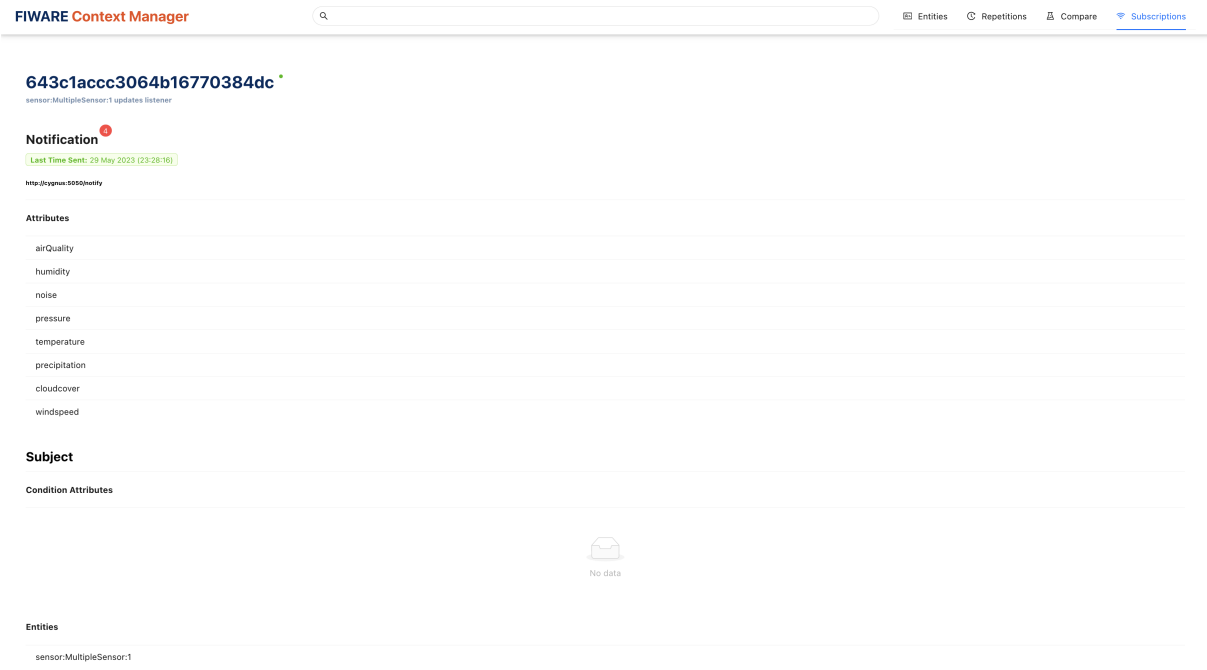


Figure 6.45: SUC subscription details validation via interface

6.4 Summary

Based on the thorough review and validation of the three key use cases, it can be confidently stated that the proposed solution is correct and effective. Through a comprehensive evaluation, each use case has been carefully examined not only during its development state but also after the solution being ready, with a higher focus on real integration testing, which provides a substantial evidence of the solution's ability to meet the specified and proposed requirements.

The entity-based use cases underwent meticulous assessment, demonstrating proficiency in creating, retrieving, updating and deleting entity instances. This validation encompassed both API calls and the direct utilization of the web interface, ensuring the accuracy and reliability of entity management within the whole system as a unit.

Likewise, the history-based use cases were subjected to a rigorous validation, confirming the capacity to not only retrieve and present historical data of entities and repetitions but also managing them.

In addition, the subscription-based use cases were carefully checked, reaffirming the system's effectiveness in managing real-time data updates. The implementation of a robust subscription mechanism, coupled with seamless integration between the system and external data sources, equips users with a dependable and efficient mean of staying abreast of the dynamic state of entities.

To summarize, the reviewed and validated use cases collectively attest to the solution's robustness, usability and adherence to the specified requirements. This ends up the development discussion by reinforcing the confidence in the system's ability to provide reliable and efficient context data management, visualization and experimentation.

The last chapter 7 concludes this thesis project by highlighting the successful design and implementation of a solution capable of tackling the challenges of deploying autonomous vehicles in a smart-city environment. It is given focus to the achievements in creating a resilient and scalable architecture, efficient data storage and processing mechanisms. Additionally, it is outlined future work opportunities, including Machine Learning (ML) and enhanced data analytics, integration with emerging technologies, expansion of simulation capabilities, integration with external data sources, usability improvements and real-world deployment and validation.

Chapter 7

Conclusions

After these long months of intensive work, a comprehensive system has been successfully designed and implemented to tackle the challenges associated with deploying autonomous vehicles in a smart-city environment. The primary objective was to create a resilient and scalable architecture capable of handling large volumes of contextual data generated within the urban space. Through careful design and implementation, this objective has been firmly accomplished, ensuring that the system can efficiently ingest and process real-time data, which provides up-to-date accessible insights to users.

A significant contribution to the project, and also to the work done until this day, has been the development of an efficient process for persistently storing context updates. This enables historical analysis and condition-based contextual analytics, empowering smart-city stakeholders to derive valuable insights from the vast amount of contextual data that can be accumulated over time.

Moreover, the implementation of the repetition system stands out as one of the core and most complex features among the final solution. This functionality allows simulations to be rerun based on specific snapshots of contextual data and empowers researchers, city-planners and decision-makers to evaluate scenarios, test hypotheses and optimize smart-city operations based on different contextual conditions and associated outcomes.

To facilitate the interaction and visualization of the contextual data, a user-friendly web interface has been fully developed from scratch. This interface enables users to explore and manipulate the rich contextual information available within the service system, providing an intuitive mean of controlling registered entities.

Throughout the completion of this thesis project, valuable insights and knowledge have been gained in several different domains. A deeper understanding of the challenges associated with deploying autonomous vehicles in a smart-city context has been acquired and moreover, expertise has been developed in building scalable architectures, implementing efficient data storages and processing mechanisms based on the FIWARE technology, which formed the core of the designed Smart-City Digital Twin (SCDT) instance and of its consequent development. Additionally,

web development skills have been also strengthened through the creation of the dynamic web interface that seamlessly communicates with the NGSIJS service, acting as the middleware and abstraction layer between the Graphical User Interface (GUI) and the context management and data storage components, integrating two separate fractions into one fully functional system.

Overall, the work invested in this project represents a significant advancement towards realizing a SCDT that effectively supports the development of innovative applications and services. By addressing the identified challenges and achieving the predefined goals, a robust and flexible foundation has been established for further advancements, not only in the field of smart-city technologies but also in the realm of Autonomous Driving (AD).

7.1 Future Work

The completion of this thesis project marks a significant milestone in the involved study field. However, as in all technological sectors, there are always several areas that offer opportunities for future work and further advancements.

7.1.1 ML and enhanced data analytics

While the developed system provides great insights through historical and condition-based contextual analytics, there is room for further improvement in what data analysis and Machine Learning (ML) have to offer. Future work could focus on incorporating advanced data analytics techniques and algorithms to extract deeper insights from the accumulated contextual data. This could involve exploring predictive analytics models, anomaly detection algorithms and real-time pattern recognition methods to enable a more proactive and intelligent decision-making within the smart-city ecosystem and the AD scenario.

7.1.2 Integration with emerging technologies

As technology continues to evolve, it is super important to keep any developed system aligned with emerging trends and advancements. Future work could also involve integrating the system with emerging technologies such as edge computing, real Internet of Things (IoT) devices and 5G networks. By leveraging these technologies, it would be possible to enhance the real-time data processing capabilities, reducing latency and supporting more diverse and dynamic applications.

7.1.3 Expansion of the simulation capabilities

The repetition system implemented in this project allows for simulations based on time-specific context data snapshots. However, there is a lot of potential for expanding and refining these capabilities. Future developments could focus on incorporating more complex simulation models,

integrating additional factors such as traffic patterns, weather conditions and sensor data. This would enable more realistic and comprehensive simulations, providing more valuable insights for urban planning, traffic management and autonomous vehicle deployment in a real-world scenario.

7.1.4 Integration with external data sources

To enrich the contextual data available within the system, future work could also focus on integrating external data sources. This could include leveraging open-data initiatives, smart-city shared APIs and data feeds from multiple sources such as weather services, public transportation systems and social media platforms. By integrating these diverse data sources, the developed system would provide a more comprehensive and holistic view of the smart-city ecosystem, enabling a more accurate and better informed decision-making.

7.1.5 Usability improvements

While the developed web interface empowers users to interact and visualize the contextual data, there is always room for improving the experience and usability of components and views. To ensure a seamless and intuitive user experience, it is important to adopt user-centric design principles, conduct user studies and gather feedback to enhance the intuitiveness, responsiveness and overall user satisfaction over the application. This iterative process will help identify pain points, streamline workflows and optimize the solution based on real user needs and preferences.

Additionally, considering the increasing prominence of smart-phone devices in our daily lives, it is crucial to explore the possibility of developing a mobile application alongside the web interface. The importance of mobile-friendly technology cannot be understated in the modern society. With the widespread adoption of smartphones, tablets and smart watches, users expect seamless access to information and services on the go. Developing a mobile application would enable users to conveniently access and interact with the system from their mobile devices, providing a more flexible and versatile user experience.

By developing a mobile application, users would have the freedom to access and utilize the system's features from anywhere, at any time which would enhance the system's usability and accessibility, allowing users to stay connected to the smart-city ecosystem even when they are away from their desktop computers. Moreover, a mobile application could leverage mobile-specific capabilities, such as location services and push notifications, to further enhance the context-aware functionalities.

Incorporating responsive design principles would ensure that the user interface adapts seamlessly to different screen sizes and resolutions, providing a consistent and optimized experience across various devices. Mobile compatibility would not only cater to the evolving needs and preferences of users but also align with the modern trend of mobile-centric computing.

7.1.6 Real-world deployment and validation

Lastly, some future work could involve the real-world deployment and validation of this developed system in collaboration with stakeholders, urban planners and expert autonomous vehicle industry partners. This would provide an opportunity to assess the system's performance, scalability and effectiveness in a live environment. Conducting pilot projects and gathering feedback from real users and stakeholders would obviously contribute to refining the solution and validating its impact on improving smart-city operations and autonomous vehicle deployment.

In conclusion, the future work outlines potential avenues for further development and advancement in the field of autonomous vehicle deployment in a smart-city context. By addressing these areas, the developed SCDT solution can continue to evolve, enhance its capabilities and serve as a valuable resource for researchers, industry professionals and city planners striving for the most efficient and sustainable urban environments possible.

Bibliography

[1] A. Arasu, S. Babu, and J. Widom. Cql: A language for continuous queries over streams and relations. In G. Lausen and D. Suciu, editors, *Database Programming Languages*, pages 1–19, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[2] V. Araujo, K. Mitra, S. Saguna, and C. Åhlund. Performance evaluation of fiware: A cloud-based iot platform for smart cities. *Journal of Parallel and Distributed Computing*, 132:250–261, 2019.

[3] C. Badii, E. Belay, P. Bellini, d. Cenni, M. Marazzini, M. Mesiti, P. Nesi, G. Pantaleo, M. Paolucci, S. Valtolina, M. Soderi, and I. Zaza. Snap4city: A scalable iot/ioe platform for developing smart city applications. In *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 2109–2116, 2018.

[4] S. Bafna. Space syntax: A brief introduction to its logic and analytical techniques. *Environment and Behavior*, 35(1):17–29, 2003.

[5] D. Bastos, P. P. Monteiro, A. S. R. Oliveira, and M. V. Drummond. An overview of lidar requirements and techniques for autonomous driving. In *2021 Telecoms Conference (ConfTELE)*, pages 1–6, 2021.

[6] R. Buehler and J. Pucher. Covid-19 impacts on cycling, 2019–2020. *Transport Reviews*, 41(4):393–400, 2021.

[7] F. Cirillo, G. Solmaz, E. L. Berz, M. Bauer, B. Cheng, and E. Kovacs. A standard-based open source iot platform: Fiware. *IEEE Internet of Things Magazine*, 2(3):12–18, 2019.

[8] W. C. Cockerham, B. W. Hamby, and G. R. Oates. The social determinants of chronic disease. *American Journal of Preventive Medicine*, 52(1, Supplement 1):S5–S12, 2017. Social Determinants of Health: An Approach to Health Disparities Research.

[9] B. Cohen. Urbanization in developing countries: Current trends, future projections, and key challenges for sustainability. *Technology in Society*, 28(1):63–80, 2006. Sustainable Cities.

[10] J. Conde, A. Munoz-Arcentales, A. Alonso, S. López-Pernas, and J. Salvachúa. Modeling digital twin data and architecture: A building guide with fiware as enabling technology. *IEEE Internet Computing*, 26(3):7–14, 2022.

[11] Use containers to build, share and run your applications. https://www.docker.com/resources/what-container. Accessed on March 2023.

[12] European commission 2030 climate target plan. https://climate.ec.europa.eu/eu-action/european-green-deal/2030-climate-target-plan_en. Accessed on May 2023.

[13] Ngsimysqlsink. FIWARE-CYGNUS, https://fiware-cygnus.readthedocs.io/en/1.18.3/cygnus-ngsi/flume_extensions_catalogue/ngsi_mysql_sink/index.html. Accessed on May 2023.

[14] Running cygnus. FIWARE-CYGNUS, https://fiware-cygnus.readthedocs.io/en/latest/cygnus-ngsi/integration/orion_cygnus_kafka.html#running-cygnus. Accessed on April 2023.

[15] Co2-mute: Fighting co2 emission with data space. FIWARE Impact Stories, https://www.fiware.org/2023/03/06/co2-mute-fighting-co2-emission-with-data-space. Accessed on April 2023.

[16] Cycle routes and air quality monitoring with real citizen's engagement. FIWARE Impact Stories, https://www.fiware.org/2020/10/01/cycle-routes-and-air-quality-monitoring-with-real-citizens-engagement. Accessed on April 2023.

[17] Orchestra cities, a one-stop-shop multi level iot platform for cities. FIWARE Impact Stories, https://www.fiware.org/2021/12/02/orchestra-cities-a-one-stop-shop-multi-level-iot-platform-for-cities-built-using-fiware. Accessed on April 2023.

[18] A platform facilitating citizens' urban experience. FIWARE Impact Stories, https://www.fiware.org/2020/04/02/a-platform-facilitating-citizens-urban-experience. Accessed on April 2023.

[19] Snap4city: Fiware powered smart app builder for sentient cities. FIWARE Impact Stories, https://www.fiware.org/2021/04/29/snap4city-fiware-powered-smart-app-builder-for-sentient-cities. Accessed on April 2023.

[20] B. R. Hiraman, C. Viresh M., and K. Abhijeet C. A study of apache kafka in big data stream processing. In *2018 International Conference on Information , Communication, Engineering and Technology (ICICET)*, pages 1–3, 2018.

[21] B. Ketzler, V. Naserentin, F. Latino, C. Zangelidis, L. Thuvander, and A. Logg. Digital twins for cities: A state of the art review. *Built Environment*, 46:547–573, 12 2020.

[22] V. V. Lehtola, M. Koeva, S. O. Elberink, P. Raposo, J.-P. Virtanen, F. Vahdatikhaki, and S. Borsci. Digital twin of a city: Review of technology serving city needs. *International Journal of Applied Earth Observation and Geoinformation*, 114:102915, 2022.

[23] Y. Li and J. Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61, 2020.

[24] H. Lv, T. Zhang, Z. Zhao, J. Xu, and T. He. The development of real-time large data processing platform based on reactive micro-service architecture. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 2003–2006, 2020.

[25] G. M., B. J., V. C. L.J., V. H., and K. R. Smart emission - building a spatial data infrastructure for an environmental citizen sensor network. Technical report, CEUR-WS, 2016.

[26] A. Manfreda, K. Ljubi, and A. Groznik. Autonomous vehicles in the smart city era: An empirical study of adoption factors important for millennials. *International Journal of Information Management*, 58:102050, 2021.

[27] Orchestra cities. https://www.orchestracities.com. Accessed on April 2023.

[28] D. W. Michele Bertoncello. Ten ways autonomous driving could redefine the automotive world. Technical report, McKinsey&Company, 2015.

[29] S. ming Wang and L. H. Vu. The integration of digital twin and serious game framework for new normal virtual urban exploration and social interaction. *Journal of Urban Management*, 12(2):168–181, 2023.

[30] I. Omer and B. Jiang. Can cognitive inferences be made from aggregate traffic flow data? *Computers, Environment and Urban Systems*, 54:219–229, 2015.

[31] Citygml. Open Geospatial Consortium, https://www.ogc.org/standard/citygml. Accessed on June 2023.

[32] L. Raes, P. Michiels, T. Adolphi, C. Tampere, A. Dalianis, S. McAleer, and P. Kogut. Duet: A framework for building interoperable and trusted digital twins of smart cities. *IEEE Internet Computing*, 26(3):43–50, 2022.

[33] J. Schering, C. Janßen, R. Kessler, V. Dmitriyev, J. Stüven, J. Marx Gómez, E. van Dijk, W. Brouwer, A. Kamermans, L. Verweij, and G. Janssen. *ECOSense and Sniffer Bike: European Bike Sensor Applications and Its Potential to Support the Decision-Making Process in Cycling Promotion*, pages 157–182. Springer International Publishing, Cham, 2022.

[34] D. Shanmugam, F. Diaz, S. Shabanian, M. Finck, and A. Biega. Learning to limit data collection via scaling laws: A computational interpretation for the legal principle of data minimization. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 839–849, New York, NY, USA, 2022. Association for Computing Machinery.

[35] A. Sharifi, A. R. Khavarian-Garmsir, and R. K. R. Kummitha. Contributions of smart city solutions and technologies to resilience against the covid-19 pandemic: A literature review. *Sustainability*, 13(14), 2021.

[36] L. M. C. d. Silva and J. Uhlmann. Contributing factors for the underutilization of mobility stations: the case of the "wien mobil station" in vienna. *Revista Produção e Desenvolvimento*, 7, Feb. 2021.

[37] A. Sinanovic, E. Meskovic, A. Mujcic, and N. Suljanovic. Smart city use case development based on fiware technology. In *2022 30th Telecommunications Forum (TELFOR)*, pages 1–4, 2022.

[38] S. T. Somayya Madakam, R. Ramaswamy. Internet of things (iot): A literature review. Technical report, Journal of Computer and Communications, 2015. Accessed on March 2023.

[39] Adding linked data concepts to fiware data entities. STEP-BY-STEP FOR NGSI-V2, https://fiware-tutorials.readthedocs.io/en/stable/linked-data/index.html. Accessed on May 2023.

[40] K. Su, J. Li, and H. Fu. Smart city and the applications. In *2011 International Conference on Electronics, Communications and Control (ICECC)*, pages 1028–1031, 2011.

[41] C. Tampère. Digital urban european twins. Technical report, KUL, 2021. Accessed on June 2023.

[42] C. Tikkinen-Piri, A. Rohunen, and J. Markkula. Eu general data protection regulation: Changes and implications for personal data collecting companies. *Computer Law & Security Review*, 34(1):134–153, 2018.

[43] Z. Wadud, D. MacKenzie, and P. Leiby. Help or hindrance? the travel, energy and carbon impacts of highly automated vehicles. *Transportation Research Part A: Policy and Practice*, 86:1–18, 2016.

[44] S.-J. Yang, P.-C. Lai, and J. Lin. Design role-based multi-tenancy access control scheme for cloud services. In *2013 International Symposium on Biometrics and Security Technologies*, pages 273–279, 2013.

[45] I. Yaqoob, L. U. Khan, S. M. A. Kazmi, M. Imran, N. Guizani, and C. S. Hong. Autonomous driving cars in smart cities: Recent advances, requirements, and challenges. *IEEE Network*, 34(1):174–181, 2020.