

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Secure and Scalable Data Architecture for the Integration of Heterogeneous Urban Data in Smart Cities

João Pedro Moreira Costa



Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Bruno Lima

Second Supervisor: Prof. Rosaldo Rossetti

July 16, 2026

A Secure and Scalable Data Architecture for the Integration of Heterogeneous Urban Data in Smart Cities

João Pedro Moreira Costa

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. João Bispo

Examiner: Prof. José Duarte

Member: Prof. Bruno Lima

July 16, 2026

Resumo

A transição para cidades inteligentes gera volumes de dados heterogêneos cuja gestão expõe as limitações das arquiteturas monolíticas, frequentemente associadas a data silos e constrangimentos de desempenho. Neste contexto, o projeto URBAN-NET visa reconfigurar a mobilidade urbana através de uma plataforma inteligente de otimização multimodal rumo à neutralidade carbónica. Para responder a esta complexidade técnica, a presente dissertação propõe uma arquitetura de dados híbrida, segura e escalável.

A solução proposta implementa uma arquitetura de dados híbrida bifurcada SQL/NoSQL integrada com o ecossistema FIWARE. A gestão de contexto em tempo real é centralizada através do Orion-LD Context Broker, suportado pelo MongoDB, o que garante representações de estado de baixa latência. Simultaneamente, os dados históricos e as análises complexas são geridos pelo QuantumLeap, que encaminha a telemetria para o TimescaleDB, separando eficazmente as cargas de trabalho analíticas das operações em tempo real.

Para superar os desafios de interoperabilidade, o sistema utiliza pipelines de transformação personalizados que harmonizam diversos formatos de ficheiros (CSV, GTFS e GeoJSON) no formato NGSI-LD padronizado. Além disso, em relação à segurança, o controlo de acesso é garantido através da integração do Keycloak como fornecedor centralizado de Gestão de Identidades e Acessos, assegurando proteção contra o acesso não autorizado.

Concebido como a espinha dorsal tecnológica das cidades inteligentes, o UrbanData Nexus centraliza e gere informação urbana heterogênea, desde sensores IoT em tempo real a dados de mobilidade. Validada através do projeto URBAN-NET na região do Grande Porto, esta infraestrutura unificada garante o armazenamento seguro e o acesso imediato aos dados, suportando a análise e a tomada de decisão em ecossistemas urbanos complexos.

Avaliações empíricas demonstram a eficiência computacional e a escalabilidade linear da arquitetura na gestão de telemetria de IoT de alta velocidade. O sistema mitiga eficazmente os silos de dados urbanos, centralizando diferentes formatos de dados e assegurando uma camada de segurança independente e transparente. Em última análise, a solução UrbanData Nexus estabelece uma base resiliente e preparada para o futuro que transforma a gestão reativa do tráfego num ecossistema de mobilidade urbana proativo e sustentável.

Palavras-chave: Smart Cities • Hybrid Data Architecture • Big Data • Internet of Things • SQL • NoSQL • FIWARE • NGSI-LD

Abstract

The transition to smart cities generates heterogeneous data volumes, the management of which exposes the limitations of monolithic architectures, often associated with data silos and performance constraints. In this context, the URBAN-NET project aims to reshape urban mobility through an intelligent multimodal optimization platform geared toward carbon neutrality. To address this technical complexity, this dissertation proposes a hybrid, secure, and scalable data architecture.

The proposed solution implements a bifurcated SQL/NoSQL hybrid data architecture integrated with the FIWARE ecosystem. Real-time context management is centralized via the Orion-LD Context Broker, powered by MongoDB, which ensures low-latency state representations. Simultaneously, historical data and complex analytics are managed by QuantumLeap, which routes telemetry to TimescaleDB, effectively separating analytical workloads from real-time operations.

To overcome interoperability challenges, the system uses custom transformation pipelines that harmonize various file formats (CSV, GTFS, and GeoJSON) into the standardized NGSI-LD format. Furthermore, regarding security, access control is ensured through the integration of Keycloak as a centralized Identity and Access Management provider, ensuring protection against unauthorized access.

Designed as the technological backbone of smart cities, UrbanData Nexus centralizes and manages heterogeneous urban information, ranging from real-time IoT sensors to mobility data. Validated through the URBAN-NET project in the Greater Porto region, this unified infrastructure ensures secure storage and immediate access to data, supporting analysis and decision-making in complex urban ecosystems.

Empirical evaluations demonstrate the architecture's computational efficiency and linear scalability in managing high-speed IoT telemetry. The system effectively mitigates urban data silos by centralizing different data formats and ensuring an independent and transparent security layer. Ultimately, the UrbanData Nexus solution establishes a resilient, future-ready foundation that transforms reactive traffic management into a proactive and sustainable urban mobility ecosystem.

Keywords: Smart Cities • Hybrid Data Architecture • Big Data • Internet of Things • SQL • NoSQL • FIWARE • NGSI-LD

The United Nations Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for achieving a better and more sustainable future for all. It includes 17 goals (<https://sdgs.un.org/goals>) to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

This dissertation directly contributes to these global objectives by developing a secure and scalable data architecture tailored for smart cities. By effectively integrating heterogeneous urban data, particularly in the context of intelligent mobility and IoT infrastructure within the URBAN-NET project, the platform provides a robust foundation for data-driven decision-making to foster sustainable, efficient, and resilient urban environments.

The specific Sustainable Development Goals mentioned have the following names:

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation

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

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.1	The platform's hybrid SQL/NoSQL architecture provides a reliable, scalable foundation for managing real-time IoT data.	System uptime, ingestion latency, and data throughput.
	9.4	Modernizes urban data pipelines by standardizing diverse formats (CSV, GTFS, GeoJSON) into a unified NGSI-LD model.	Number of formats harmonized and API response times.
11	11.2	By merging public transit and traffic data, the system directly supports intelligent mobility and better public transport management.	Real-time tracking of transit entities and historical query performance.
	11.6	Secure, centralized storage of spatial-temporal IoT data enables better environmental monitoring and sustainable urban planning.	Storage efficiency and the successful integration of IoT sensors.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors and teachers, Bruno Lima and Rosaldo Rossetti. Their invaluable guidance, continuous encouragement, and immense patience throughout this journey have been instrumental in the completion of this work. Their expertise and insightful feedback challenged me to grow both academically and professionally, and for that, I am truly grateful.

I would also like to extend my sincere thanks to all the people working in the URBAN-NET project. The collaborative and inspiring environment provided by the team greatly enriched my research. The discussions, shared knowledge, and support from everyone involved were essential to the success of this project.

Gratitude is also extended to the participants who generously contributed their time and insights to the case study component of this thesis.

Sincerely thanks to all.

João Pedro Costa

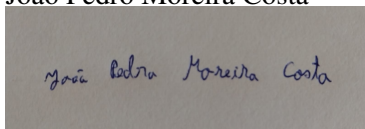
Declaration of honour

I declare that this dissertation is my own work and has not previously been submitted or assessed at this or any other institution. References to other authors (statements, ideas, thoughts), as well as the use of published works, strictly comply with the rules of attribution and are duly identified in the text and in the bibliographic references, in accordance with the applicable referencing standards. I am aware that the practice of plagiarism or self-plagiarism constitutes an academic offence, as established in the U.Porto Code of Ethics.

I further declare that I have included text(s) published in co-authorship, and that I have specified, with transparency and rigor, the contribution of each co-author, as well as stated whether there exists any other dissertation in which the same text(s) may also have been included. The authorization from the journal in which the work is published is also included, together with the express agreement signed by all co-authors authorizing the inclusion of the article.

I also declare that I have used Artificial Intelligence tools to complete part(s) of this dissertation, and that this use and its results are duly noted and have been carefully checked for errors, inaccuracies, unfounded attributions or other forms of bias in content and arguments, as well as determining authorship.

João Pedro Moreira Costa

A rectangular box containing a handwritten signature in blue ink. The signature is written in a cursive style and reads "João Pedro Moreira Costa".

“A challenge for those who go. A dream for those who stay behind. ”

Thierry Sabine

Contents

1	Introduction	1
1.1	Context & Motivation	1
1.2	Problem	1
1.3	Research questions	2
1.4	Hypothesis	2
1.5	Dissertation Structure	3
2	Background and State of The Art	4
2.1	Literature Review	4
2.1.1	Systematic Review Process	5
2.1.2	Information Sources	5
2.1.3	Search Strategy	5
2.1.4	Selection Process	6
2.2	The Challenge of Urban Data and Intelligent Mobility Systems	7
2.2.1	Performance and Scalability in IoT Systems	7
2.3	Relational and NoSQL for Big Data	8
2.3.1	The Strengths of SQL Databases	8
2.3.2	The Limitations of SQL Databases	9
2.3.3	The Strengths of NoSQL Databases	10
2.3.4	The Limitations of NoSQL Databases	10
2.3.5	Performance Evaluation of Data Storage Systems	11
2.3.6	Comparative Analysis of Relational and Non-Relational Models	12
2.4	Standards and Interoperability via FIWARE and NGSI-LD	13
2.4.1	The FIWARE Ecosystem and Context Management	13
2.4.2	FIWARE Context Broker Performance and Scalability	14
2.4.3	The NGSI-LD Standard	14
2.5	Security, Authentication, and Access Control	17
2.5.1	Security Challenges and Compliance	17
2.5.2	Access Control and Authentication	18
2.5.3	Secure RESTful APIs	19
2.6	Hybrid Solutions and the NewSQL Paradigm	21
2.6.1	The NewSQL Paradigm: Synthesizing Consistency and Scalability	21
2.6.2	Architectural Integration and Practical Implementations	22
2.6.3	Comparative Analysis of Solutions for UrbanData Nexus	25
2.7	Summary	27

3	UrbanData Nexus	29
3.1	Hybrid Data Architecture Design	29
3.1.1	Real-Time Context Management	31
3.1.2	Historical Data & Analytics	32
3.2	Data Transformation and NGSI-LD Conversion Methodology	34
3.2.1	CSV Data Processing for Traffic Observations	34
3.2.2	Relational Transit Data Processing (GTFS)	36
3.2.3	Spatial Data Processing (GeoJSON)	37
3.3	Application Data Flow Pipeline	39
3.4	Visualization & User Interface Layer	40
3.4.1	Log In & Register	40
3.4.2	Streamlit Codebase and Operational Flow	41
3.4.3	Advanced Dashboarding with Grafana	46
3.5	Security & Access Management	47
3.5.1	Centralized Identity & Network Isolation	47
3.5.2	Streamlit Access Control Implementation	47
3.6	IoT Ingestion & Device Connectivity Layer	48
3.6.1	Core Architecture and Protocol Translation	48
3.6.2	Interactive Pipeline Validation via Streamlit	49
3.7	Summary	51
4	Validation	52
4.1	System Validation	52
4.1.1	CSV data validation	53
4.1.2	GTFS data validation	55
4.1.3	GeoJson data validation	57
4.1.4	Validation of the IoT Tester Module	59
4.2	Use Case	61
4.2.1	Scenario Setup and Objectives	62
4.2.2	Results	64
4.2.3	Discussion	66
4.3	Answer to the Research Questions	67
4.3.1	RQ1 - How can a hybrid software architecture be designed to effectively ingest, process, and store heterogeneous and high-velocity data streams within an urban environment?	68
4.3.2	RQ2 - How can security mechanisms be optimally designed and integrated to simultaneously guarantee regulatory privacy compliance and robust access control?	68
4.3.3	RQ3 - Which performance evaluation frameworks and scalability metrics are most effective for quantifying the efficiency of urban mobility management systems?	68
4.4	Summary	69
5	Conclusion and Future Work	70
5.1	Conclusion	70
5.2	Future Work	71
	References	72

A	Validation	78
A.1	User Survey	78

List of Figures

2.1	PRISMA Flow Diagram	7
2.2	Taxonomy of Big Data Management in IoT	8
2.3	Size of the 350 MB data set in different databases	12
2.4	Context broker ingestion time	15
2.5	Query execution in NGSI-LD context brokers	15
2.6	Taxonomy of RESTful API vulnerabilities	20
2.7	Overview of the VoltDB architecture	22
2.8	Architecture for hybrid SQL/NoSQL database integration	23
2.9	System Architecture Design for Android applications	23
2.10	Hybrid database Architecture in Smart City ecosystems	24
3.1	Layered architecture of the UrbanData Nexus	30
3.2	Hybrid Data Architecture Diagram	30
3.3	Data pipeline for the UrbanData Nexus	39
3.4	Keycloak login interface	41
3.5	Keycloak user registration interface	42
3.6	Interface layout of the UrbanData Nexus frontend	42
3.7	Interface layout of the UrbanData Nexus frontend for a Standard User	43
3.8	Graphical user interface of the data ingestion tab	43
3.9	Graphical user interface of the CSV to NGSI-LD Converter	44
3.10	Graphical user interface of the Dataset to NGSI-LD Converter	44
3.11	Graphical user interface of the GeoJSON to NGSI-LD Converter	45
3.12	Graphical user interface of the Data Visualization module	46
3.13	Graphical user interface of the Data Models module	46
3.14	IoT Ingestion and Connectivity Architecture	49
3.15	Dashboard Interface for the Pipeline Health Check	50
3.16	Operational workflow of the IoT Tester interface	50
4.1	Web interface for converting raw CSV traffic data into the NGSI-LD	53
4.2	Data ingestion dashboard for the converted data	53
4.3	Interface for fetching and validating raw historical JSON data from TimescaleDB	54
4.4	Time-series of traffic intensity, occupancy, and average speed	54
4.5	Dashboard panel showing aggregated statistics and sparklines for key traffic metrics	55
4.6	Dataset to NGSI-LD Converter interface showing the successful conversion of a GTFS ZIP archive using the UrbanMobility context	55
4.7	QuantumLeap subscription configuration for converted GTFS stops	56
4.8	Raw NGSI-LD JSON payload and coordinates for a GTFS stop	56
4.9	Map of ingested GTFS bus stops in Porto	57

4.10	Interface for converting GeoJSON data into NGSI-LD	57
4.11	Successful QuantumLeap subscription creation	58
4.12	Raw NGSI-LD JSON data for a parking entity	58
4.13	Geospatial distribution of off-street parking facilities in Porto	59
4.14	Proportional distribution of parking typologies in Porto	60
4.15	Step 1: Successful connection establishment with the target IoT device	60
4.16	Step 2: Configuration of the validation parameters	60
4.17	Step 3: Test suite execution and real-time monitoring	61
4.18	Step 4: Comprehensive dashboard displaying the final validation results	62
4.19	Distribution of participants by age group	64
4.20	Average perceived difficulty per task	65
4.21	Average execution time required by participants to complete each task	65
4.22	Average participant scores for the System Usability Scale (SUS) questionnaire	66
A.1	User survey questionnaire: Page 1	79
A.2	User survey questionnaire: Page 2	80
A.3	User survey questionnaire: Page 3	81
A.4	User survey questionnaire: Page 4	82
A.5	User survey questionnaire: Page 5	83
A.6	User survey questionnaire: Page 6	84
A.7	User survey questionnaire: Page 7	85
A.8	User survey questionnaire: Page 8	86
A.9	User survey questionnaire: Page 9	87
A.10	User survey questionnaire: Page 10	88
A.11	User survey questionnaire: Page 11	89
A.12	User survey questionnaire: Page 12	90
A.13	User survey questionnaire: Page 13	91
A.14	User survey questionnaire: Page 14	92
A.15	User survey questionnaire: Page 15	93
A.16	User survey questionnaire: Page 16	94
A.17	User survey questionnaire: Page 17	95

List of Tables

2.1	Number of results per source for each query.	6
2.2	Storage Footprint Comparison for a 350 MB Dataset	12
2.3	Comparative analysis of key characteristics between Relational and Non-Relational databases.	13
2.4	Comparison of hybrid and database architectures for the UrbanData Nexus platform.	26

List of Acronyms

ABAC	Attribute-Based Access Control
ACID	Atomicity, Consistency, Isolation, and Durability
API	Application Programming Interface
BASE	Basically Available, Soft State, Eventually Consistent
BOLA	Broken Object Level Authorization
CAP	Consistency, Availability and Partition Tolerance
CDC	Change Data Capture
CPS	Cyberphysical Systems
CSV	Comma Separated Values
DTLS	Datagram Transport Layer Security
ETSI	European Telecommunications Standardization Institute
EV	Electric Vehicles
FGAC	Fine-Grained Access Control
GDPR	General Data Protection Regulation
GE	Generic Enabler
GeoJSON	Geographic JavaScript Object Notation
GTFS	General Transit Feed Specification
IAM	Identity and Access Management
IDOR	Insecure Direct Object Reference
IoT	Internet of Things
JWT	JSON Web Tokens
LDAP	Lightweight Directory Access Protocol
LSM	Log-Structured Merge
MQTT	Message Queuing Telemetry Transport

NGSI	Next Generation Service Interfaces
NGSI-LD	Next Generation Service Interface - Linked Data
NoSQL	Not only SQL
NoSQLi	Not only SQL Injection
OMA	Open Mobile Alliance
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
RBAC	Role-Based Access Control
RDBMS	Relational Database Management Systems
REST API	Representational State Transfer Application Programming Interface
SQL	Structured Query Language
SQLi	Structured Query Language Injection
SUS	System Usability Scale
TLS	Transport Layer Security
TSDB	Time-Series Database
UI	User Interface
URN	Uniform Resource Name
URI	Uniform Resource Identifier

Chapter 1

Introduction

1.1 Context & Motivation

The increasing amount of data from smart city initiatives [36] presents a significant challenge. These data come from various sources, such as Internet of Things (IoT) sensors, transportation systems, and public utilities, creating a complex and diverse data landscape [7].

A primary motivation for this work is addressing the critical challenges associated with managing the escalating volume and variety of urban data. Current systems frequently lack the scalability and interoperability required to support real-time smart city applications [27]. Integrating these disparate data streams into a unified, actionable system remains a significant hurdle [1].

Furthermore, because this information often includes sensitive data spanning personal and public infrastructure, ensuring robust security and privacy is paramount [3]. Consequently, this effort focuses on engineering a secure, efficient, and accessible data backbone, which is essential for unlocking the full potential of smart city initiatives.

To validate the proposed solution, this research uses the URBAN-NET project [23] as its primary use case. This initiative tackles the complex challenge of sustainable urban mobility through the development of a smart platform. The dissertation contributes directly to that overarching objective by developing the secure and scalable data architecture intended to serve as the platform's foundation. Specifically, the design of a hybrid architecture leveraging both Structured Query Language (SQL) and Not only SQL (NoSQL) databases, seamlessly integrated with the FIWARE Context Broker. By validating our approach within this ecosystem, this work aims to ensure that the underlying data remains standardized, interoperable, and efficiently shared across all constituent systems.

1.2 Problem

Smart cities face significant hurdles in data management, starting with data silos, where different departments operate in isolation, making it difficult to share information and gain a unified

understanding of urban systems [28]. This is compounded by performance bottlenecks, as traditional databases struggle to handle the massive, real-time data streams from IoT sensors [42]. Security vulnerabilities pose another critical risk, as sensitive data is constantly at risk of being compromised without robust protection [32]. Finally, a pervasive lack of interoperability, due to the absence of a standardized data model and Application Programming Interface (API), prevents a cohesive ecosystem of applications from being developed, hindering the creation of a truly integrated and efficient smart city [8, 27].

1.3 Research questions

With the objective of designing a robust and secure data architecture capable of managing the complex, high-velocity, and heterogeneous data streams characteristic of smart city environments, and in order to guide the investigation and correctly evaluate the results, the following Research Questions (RQs) were formulated:

- **RQ1** — How can a hybrid software architecture be designed to effectively ingest, process, and store heterogeneous and high-velocity data streams within an urban environment?
- **RQ2** — How can security mechanisms be optimally designed and integrated to simultaneously guarantee regulatory privacy compliance and robust access control?
- **RQ3** — Which performance evaluation frameworks and scalability metrics are most effective for quantifying the efficiency of urban mobility management systems?

1.4 Hypothesis

The complexity and dynamic nature of modern urban environments require a paradigm shift away from conventional monolithic data management systems [54]. To address the inherent challenges posed by the heterogeneity, velocity, and volume of smart city data, which frequently requires integrating diverse datasets originating from Comma Separated Values (CSV), General Transit Feed Specification (GTFS), and Geographic JavaScript Object Notation (GeoJSON) files, a novel architectural approach is required.

This dissertation hypothesizes that a hybrid SQL/NoSQL data architecture, securely integrated with the FIWARE Context Broker via the Next Generation Service Interface - Linked Data (NGSI-LD) standard, demonstrates superior scalability, security, and interoperability compared to traditional monolithic storage paradigms [26].

Furthermore, the incorporation of Keycloak as a centralized Identity and Access Management (IAM) solution is hypothesized to enhance the security posture of the architecture by providing robust authentication, authorization, and role-based access control mechanisms across distributed services and data sources.

Crucially, to ensure comprehensive semantic interoperability across these varied sources, all models utilized within the database adhere to the official Smart Data Models¹ guidelines. Specifically, this research evaluates the architecture’s efficacy in managing these standardized models alongside heterogeneous, high-velocity data streams for real-time smart city applications within the URBAN-NET platform and analogous distributed environments.

1.5 Dissertation Structure

Beyond this introductory chapter, the remainder of this dissertation is logically organized into four core chapters, each designed to systematically guide the reader through the theoretical background, technical implementation, and empirical evaluation of the proposed architecture.

Chapter 2 provides a comprehensive review of the state of the art, examining the existing literature, current data management paradigms, and underlying technologies. By critically analyzing related work, this chapter establishes the necessary academic context and identifies the specific technological gaps that this research aims to resolve.

Following the review of the state of the art, Chapter 3 details the developed solution, the UrbanData Nexus. This chapter details the core architectural design choices, technical configurations, and data integration strategies, illustrating how the platform effectively manages diverse urban datasets to achieve the project’s objectives.

Subsequently, Chapter 4 describes the methodological framework utilized to validate the proposed solution. Specifically, it outlines the experimental setup, testing scenarios, and evaluation metrics used to rigorously assess the architecture’s performance, scalability, and reliability under dynamic data conditions.

Finally, Chapter 5 concludes the dissertation by summarizing the core findings and highlighting the primary contributions of the project. Reflects on the broader implications of the research results and suggests strategic avenues for future work and ongoing system enhancements.

¹<https://smartdatamodels.org/>

Chapter 2

Background and State of The Art

This chapter establishes the theoretical and technological foundation for developing intelligent, scalable urban mobility systems, specifically contextualizing the architecture required for the UrbanData Nexus project. Systematically explores the core technologies, standards, and challenges inherent in modern smart city data infrastructures.

To provide a comprehensive overview, the chapter begins with a systematic review of the literature using a PRISMA-based methodology to outline the search strategy and selection process used to aggregate state-of-the-art research. It then transitions into an analysis of the challenge of urban data, detailing the constraints imposed by the volume, velocity, and variety of Big Data in the IoT and smart mobility contexts, along with the critical need for horizontal scalability. Following this, a comparative evaluation of data storage paradigms examines the strengths and limitations of Relational (SQL), Non-Relational (NoSQL), and Time-Series Databases (TSDBs) in handling complex urban datasets. The discussion subsequently shifts to standards and interoperability, exploring the FIWARE ecosystem and the NGSI-LD standard to highlight the role of context brokers and semantic interoperability via Digital Twins.

Given the sensitivity of distributed urban data, the chapter also reviews essential security and access control mechanisms necessary to protect this information, encompassing API security, encryption, and GDPR compliance requirements. Finally, an assessment of hybrid solutions and the NewSQL paradigm identifies current research gaps, justifying the need for a novel, highly optimized, and secure approach. By synthesizing these domains, this chapter defines the current technological landscape and demonstrates the necessity of integrating hybrid storage solutions with standardized semantic protocols to effectively manage real-time mobility data.

2.1 Literature Review

This section outlines the rigorous methodology employed to conduct the foundational literature review for this research. To ensure transparency and reproducibility, the entire procedure was structured in accordance with the PRISMA guidelines. The following subsections detail the systematic approach used to gather, filter, and analyze state-of-the-art research published between

2020 and 2025. This encompasses the selection of premier academic databases, the formulation of targeted Boolean search strategies, and the comprehensive multi-phase screening process designed to isolate the most relevant studies concerning hybrid database architectures, the FIWARE ecosystem, and secure smart city data infrastructures.

2.1.1 Systematic Review Process

The systematic review was structured according to the Guidelines for preferred reporting items for systematic reviews and Meta-Analyses (Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA)) [48].

To ensure currency, this analysis restricts its scope to studies published between 2020 and 2025, reflecting the significant and rapid evolution across several key domains: hybrid database architectures (SQL/NoSQL), advanced security mechanisms (including encryption, authentication, and access control), the adoption of the FIWARE Context Broker leveraging the NGSI-LD standard, and modern smart city implementations.

2.1.2 Information Sources

The research team used the following databases to gather a wide spectrum of research articles:

- ACM Digital Library [4]: A primary database for research in computing and information technology.
- IEEE Xplore [31]: Chosen for its comprehensive coverage of engineering, electronics, and technology articles and standards.
- Science Direct [21]: Selected for its extensive collection of scientific, technical, and medical (STM) research, including journals and books published by Elsevier.

2.1.3 Search Strategy

A systematic search strategy was formulated to identify and aggregate relevant academic and technical literature pertaining to hybrid data architectures within smart city ecosystems. Particular emphasis was placed on implementations leveraging the FIWARE framework [25] and the NGSI-LD data exchange standard [22].

To ensure comprehensive coverage of the intersection between relational/non-relational storage and urban data models, the following Boolean search strings were employed:

- **Query 1** — hybrid AND ("SQL" AND "NoSQL") AND architecture
- **Query 2** — ("FIWARE" AND "Context Broker" AND "NGSI-LD" AND (integration OR architecture))
- **Query 3** — ("data storage" OR "data management" OR "data infrastructure") AND ("smart city" OR "smart cities") AND ("SQL" AND "NoSQL")

- **Query 4** — ("FIWARE" OR "NGSI-LD") AND ("data model" OR "information model" OR "data schema") AND ("urban mobility" OR "smart mobility" OR "intelligent transportation")
- **Query 5** — ("SQL" AND "NoSQL" AND hybrid AND "RESTful API") OR ("SQL/NoSQL" AND "REST API")

The quantitative results yielded by each specific search string are summarized in Table 2.1.

Query	ACM	IEEE	ScienceDirect
1	111	4	200
2	12	0	25
3	16	4	100
4	5	2	21
5	5	0	20

Table 2.1: Number of results per source for each query.

2.1.4 Selection Process

Following the successful retrieval of 431 articles from three distinct databases, the methodology began with an initial screening phase. This first step was restricted to the evaluation of the title and abstract of each publication.

The primary objective of this process was to identify articles relevant to a range of critical research themes, specifically: hybrid architectures combining **SQL** and **NoSQL** databases, the application of databases within the context of smart cities, the deployment of the FIWARE Context Broker utilizing the **NGSI-LD** standard, the implementation of Representational State Transfer Application Programming Interface (**REST API**)s for both **SQL** and **NoSQL** systems, and various security measures across different phases of system development.

The subsequent and more intensive second screening involved the remaining 81 pre-selected articles. This phase needed a closer, full-text examination, primarily focusing on the introduction, conclusion, and supporting visual elements (figures and tables). Furthermore, a systematic keyword search for the terms **SQL**, **NoSQL**, FIWARE, Next Generation Service Interfaces (**NGSI**), and **API** was executed. Crucially, the process extended beyond the mere confirmation of keyword presence; the context in which these terms were situated was analyzed to ascertain whether the article presented content that was potentially useful or applicable to research areas beyond the initial inclusion criteria.

Figure 2.1 provides an overview of the flow of information through the different phases of the review.

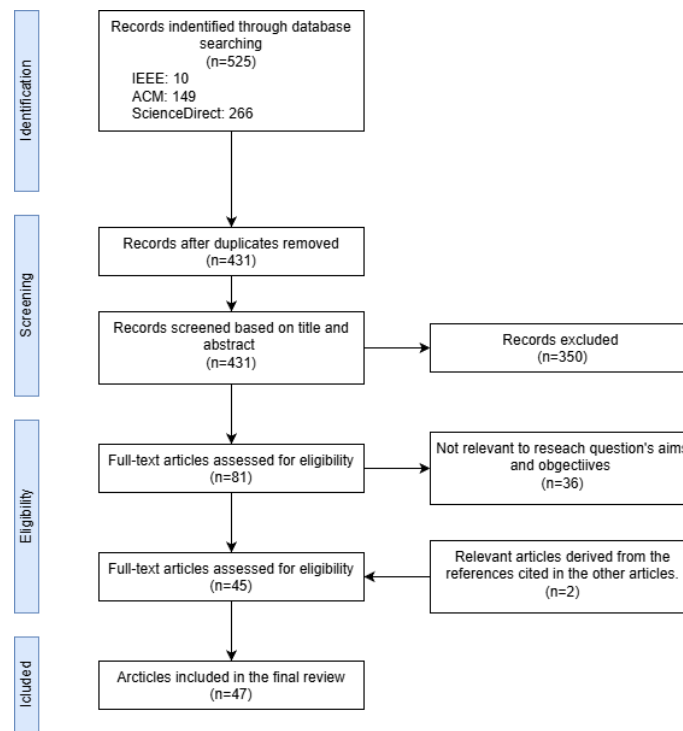


Figure 2.1: PRISMA Flow Diagram illustrating the selection process for studies included in the systematic review

2.2 The Challenge of Urban Data and Intelligent Mobility Systems

The continuous development of technologies such as the Internet of Things and big data has enabled the feasibility of smart city initiatives, including smart transportation systems [12, 33, 53]. Urban data, sourced from sensors, mobility systems, and public infrastructure, is characterized by its massive scale (Volume), diverse formats (Variety—including structured, semi-structured, and unstructured data), and high-speed generation (Velocity) [5, 13, 41, 50, 58]. These attributes are commonly known as the "Vs" of Big Data [13, 50, 58].

Handling the volume and heterogeneity of urban data imposes significant constraints on analysis, processing, and storage [50, 61]. Legacy systems relying on manual input are inefficient and ill-equipped to meet modern scalability demands [33]. In contrast, intelligent urban mobility systems require real-time processing capabilities, making a resilient data infrastructure critical [43, 60]. Therefore, secure and interoperable frameworks are urgently needed to integrate diverse sensor data [61]. Figure 2.2 depicts the proposed taxonomy for Big Data Management in IoT [6], highlighting the Data Storage branch. This work primarily addresses Hybrid Storage solutions, specifically the convergence of SQL and NoSQL databases.

2.2.1 Performance and Scalability in IoT Systems

Evaluating performance and scalability is critical for large-scale urban platforms and the IoT systems that power them. To handle the massive influx of connected devices and data—which can

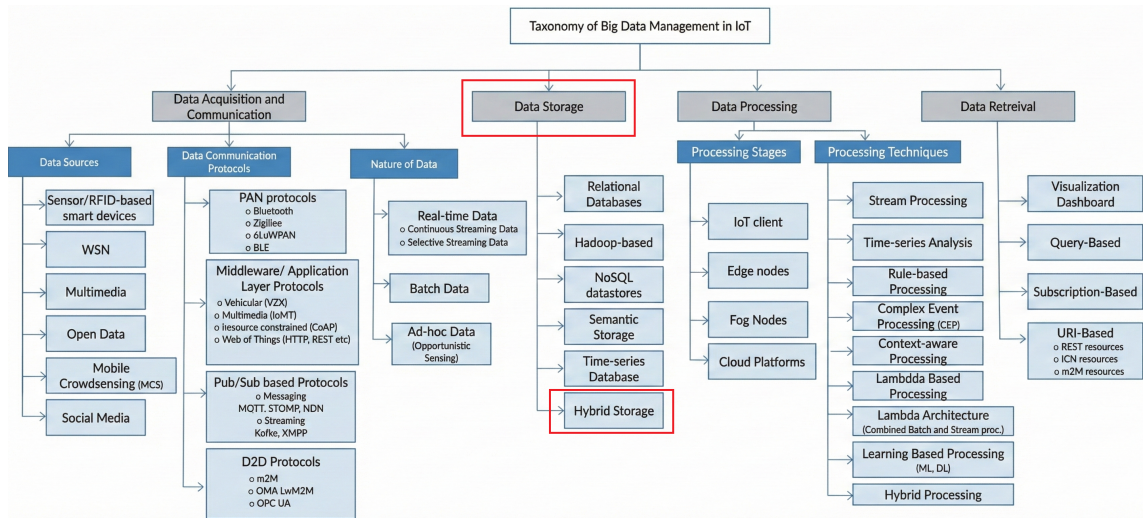


Figure 2.2: Taxonomy of Big Data Management in IoT [6].

reach exabytes—these systems must avoid performance degradation. This is best achieved through horizontal scaling (scaling out), which involves adding more machines to the network to manage increased workloads, rather than relying on the limited and costly method of upgrading a single node’s resources [20, 55, 56].

Consequently, modern architectures increasingly favor non-relational databases (**NoSQL**) over traditional relational databases (Relational Database Management Systems (**RDBMS**)). While **RDBMS** typically rely on vertical scaling (scaling up), **NoSQL** databases are inherently designed for horizontal scaling through data partitioning or sharding. This capability allows them to store and process huge volumes of data efficiently, making them the superior choice for maintaining service quality in Big Data applications [29, 50, 53].

2.3 Relational and NoSQL for Big Data

In the contemporary landscape, the strategic imperatives of forward-thinking organizations (particularly those navigating the intricacies of urban-scale, real-time analytics) demand the seamless harmonization of inherently conflicting data paradigms. It is no longer sufficient to merely collect data; success now depends on the ability to combine divergent requirements into a cohesive operational framework [9, 16].

2.3.1 The Strengths of SQL Databases

One of the primary advantages of **SQL** databases is their unyielding focus on Data Integrity and Consistency. These systems are logically designed to manage data that has a fixed or rarely changing structure, making them highly efficient when consistency is a top priority [9]. This reliability is powered by Atomicity, Consistency, Isolation, and Durability (**ACID**) properties, which guarantee that transactions are processed securely [52, 53, 59]. Whether a transaction completes fully or

fails entirely, the database ensures the data remains in a valid state, making **SQL** systems superior for handling critical transactional queries like financial records or inventory management [29, 52].

To further optimize data management, **SQL** systems utilize normalization and referential integrity. Normalization involves splitting data into smaller, logical tables to eliminate redundancy and optimize storage space [9].

Beyond internal architecture, the standardization of the **SQL** itself is a significant strength. Because **SQL** is a widespread and standardized language, it offers users a high degree of familiarity and convenience [59]. This standardization promotes interoperability across different environments, meaning that skills and queries are often transferable between systems. This significantly reduces the risk of vendor lock-in, allowing organizations to switch between different database providers without having to completely relearn how to interact with their data [24, 29].

Finally, relational databases excel at Complex Query Efficiency. They are specifically engineered to handle intricate operations involving a high number of joins between tables. While non-relational systems may struggle with connecting data points across various structures, **SQL** databases perform these tasks with great speed and precision. For applications that require straightforward data retrieval or need to traverse multiple relationships to generate reports, relational systems consistently outperform their non-relational counterparts [9, 29].

2.3.2 The Limitations of SQL Databases

RDBMS, relying on **SQL**, present critical limitations in the face of modern data challenges, primarily revolving around structural inflexibility and constrained horizontal scalability. The foundational requirement for a static, predefined schema introduces significant overhead during development and maintenance, leading to an undeniable schema evolution cost when structural changes are necessary. This rigidity fundamentally restricts the **RDBMS**'s ability to natively handle data variety, making it cumbersome for storing and querying the unstructured or semi-structured data streams—such as documents and sensor outputs—common in today's Big Data applications. Furthermore, the inherent prioritization of **ACID** properties introduces performance trade-offs, making the mandatory verification of writes comparatively less time-efficient than Basically Available, Soft State, Eventually Consistent (**BASE**)-compliant **NoSQL** systems in high-throughput write environments [15, 51, 53].

A second critical constraint is the challenge of horizontal scalability, or scaling out across a cluster of commodity machines. While **SQL** architectures are optimized for vertical scaling (scaling up resources on a single server), their strong consistency model complicates seamless data distribution and partitioning. The difficulty in maintaining absolute consistency, availability, and partition tolerance simultaneously is encapsulated by the Consistency, Availability and Partition Tolerance (**CAP**) theorem, where **RDBMS** generally prioritize consistency, thereby limiting their partition tolerance and restricting their capacity to manage the explosive data volumes generated by smart cities and **IoT** systems [9, 19, 34].

While **SQL** databases ensure strong consistency through **ACID** checks, this rigorous validation can sometimes lead to lower data write efficiency compared to **BASE**-compliant **NoSQL** systems.

Furthermore, navigating deeply nested or complex data relationships within the relational model frequently requires computationally expensive multiple JOIN operations, which results in queries that are verbose, complex, and less performant than those in specialized alternatives like graph databases [29].

2.3.3 The Strengths of NoSQL Databases

NoSQL databases are fundamentally designed for scalability and high availability, a crucial requirement for managing massive, rapidly growing datasets in systems like **IoT**. Their architecture supports linear horizontal scalability (scaling out) by distributing data and workload across many machines, typically utilizing sharding to partition the database [9, 53, 59]. This distributed model inherently provides high data availability and ensures low latency and service reliability, making **NoSQL** systems highly efficient at handling concurrent, high-throughput I/O operations (reads and writes) compared to vertically scaling relational databases [11, 12, 34].

A key strength of **NoSQL** lies in its flexible schema or schemaless design, which effectively addresses the challenge of data variety. This flexibility allows the system to easily accommodate unstructured or semi-structured data and integrate new sources, such as diverse sensors and actuators, without the rigid requirement for complex, system-wide schema modifications characteristic of **SQL** databases. For instance, Document databases like MongoDB exemplify this flexibility, making them ideal for managing heterogeneous data in real-time applications where data structures evolve frequently [40, 50, 59].

Finally, **NoSQL** provides a significant advantage through its diverse ecosystem of specialized data models, tailored to specific application needs, including key-value, column-oriented, document, and graph databases [29]. This specialization allows developers to choose the optimal structure for the task; for example, graph databases (like Neo4j) are exceptionally proficient for datasets involving complex relationships where efficient traversal is paramount, offering performance benefits that generic relational models cannot match [11, 29, 41].

2.3.4 The Limitations of NoSQL Databases

NoSQL databases present a trade-off where their scalability and flexibility come at the cost of certain features traditionally guaranteed by relational systems. The primary limitation is their weaker consistency model. **NoSQL** systems typically adhere to the **BASE** paradigm, which prioritizes continuous operation and high data availability over the immediate consistency guaranteed by **ACID**, compliant **SQL** systems [51, 53]. This makes them less suitable for enterprise-grade transactional systems where strong data integrity is paramount. Furthermore, the absence of a strictly enforced schema means they generally lack tight control over referential integrity rules, necessitating additional, complex application-level coding to ensure data validity and coherence [9, 59].

Another significant challenge is the lack of a standardized query language. Unlike **SQL**, which provides a universal interface for relational databases, **NoSQL** systems rely on proprietary,

manufacturer-specific query languages (e.g., MQL, CQL, Cypher). This fragmentation restricts cross-system compatibility for development teams and significantly raises the risk of vendor lock-in [29, 30]. Coupled with this is the limitation in data aggregation: many NoSQL databases do not support the standard JOIN operation. While they offer compensatory mechanisms, such as nesting documents or using specific operators like *lookup*, these alternatives can be overly complex and have performance limitations when attempting to efficiently read and integrate large amounts of data from different documents or collections simultaneously [9, 53].

Finally, NoSQL systems historically exhibit security gaps and lower initial maturity compared to established relational databases. These systems were not initially designed with robust security and privacy features deeply integrated into their core architecture. Consequently, certain NoSQL implementations may offer limited support for essential enterprise-level features, including comprehensive authentication, fine-grained access control, and encryption mechanisms. This often forces organizations to implement custom security solutions at the application layer, which adds complexity, increases development effort, and potentially introduces new security vulnerabilities [53, 59].

2.3.5 Performance Evaluation of Data Storage Systems

Evaluations of data storage for high-volume smart city and IoT initiatives highlight a shift away from traditional relational databases toward more specialized technologies like NoSQL and Time-Series Database (TSDB)s. These alternatives address the need for efficient management of massive, dynamic, and often time-stamped data [9, 61].

2.3.5.1 NoSQL Performance Advantages

NoSQL databases are distinguished by their ability to provide low latency and efficient execution of read and write operations. They have emerged as a viable solution for diverse datasets—such as healthcare records—that traditional SQL systems struggle to manage [12].

Speed and architecture in direct comparisons, such as between MongoDB and Oracle, NoSQL solutions frequently demonstrate superior performance in high-volume settings. Research indicates that MongoDB takes the lead in execution speed once data insertion surpasses 1,000 records. This advantage in expensive data insert operations is primarily due to the architectural flexibility of document-based databases, which bypass the rigorous checks and verifications associated with the ACID-based transactional modeling found in relational systems [9, 49].

2.3.5.2 Time-Series Databases (TSDBs) Efficiency

TSDBs, like InfluxDB, are specialized specifically for handling the time-stamped data that characterizes IoT sensor readings. These databases are engineered to deliver high write throughput and efficient querying for specific time ranges [20].

In smart city platform case studies, InfluxDB has demonstrated performance levels similar to MongoDB while significantly outperforming traditional relational databases. However, the most

distinct advantage of **TSDBs** lies in storage optimization. They offer a successful balance between high performance and minimal storage footprint, making them highly attractive for large-scale installations [34].

2.3.5.3 Comparative Storage Footprint

The table 2.2 below illustrates the significant efficiency of **TSDBs** using a specific case study of a 350 MB dataset. Note how InfluxDB compresses data far more effectively than its counterparts [34]. A comparison of the space required to store both datasets can be seen in Figure 2.3.

Technology Class	Database System	Storage Required	Efficiency Rating
TSDB	InfluxDB	69 MB	Excellent
NoSQL	MongoDB (Uncompressed)	265 MB	Moderate
SQL	PostgreSQL	787 MB	Poor

Table 2.2: Storage Footprint Comparison for a 350 MB Dataset

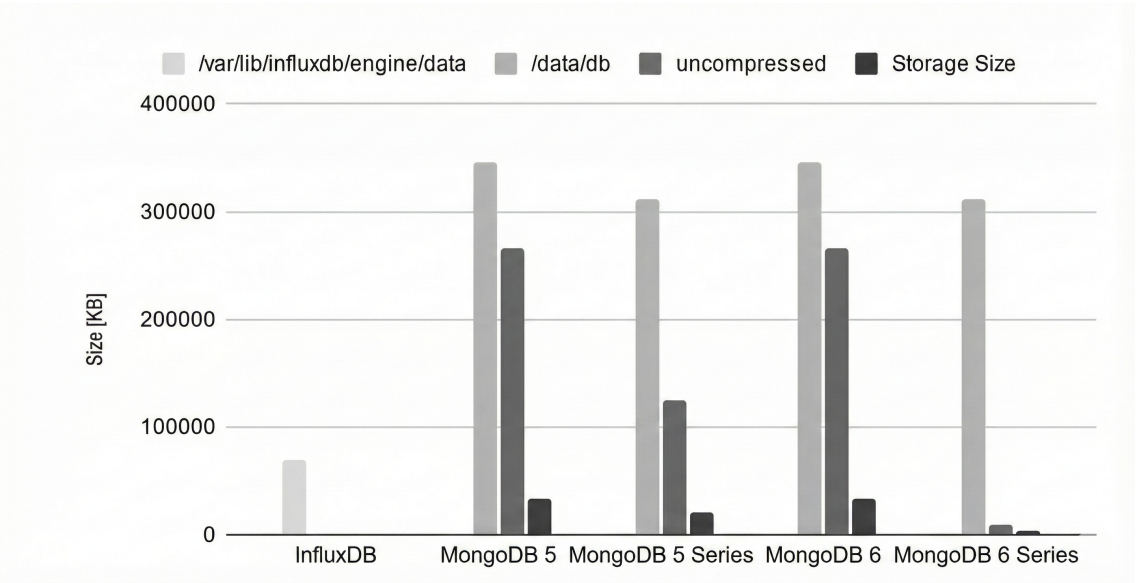


Figure 2.3: Size of the 350 MB data set in different databases [34].

2.3.6 Comparative Analysis of Relational and Non-Relational Models

The growing necessity to harmonize the conflicting demands of data management—preserving **SQL**’s data integrity and complex query capabilities while embracing **NoSQL**’s scalability and flexibility—is the primary driver behind the widespread adoption of hybrid data architectures. This approach involves the parallel use of both **SQL** and **NoSQL** databases, utilizing their specific strengths. For instance, structured metadata might be stored in a relational database (**SQL**), while large volumes of highly variable, real-time sensor data are stored in a non-relational system (**NoSQL**) [9, 14, 34].

Table 2.3 provides a comparative overview of Relational and Non-Relational databases. The comparison highlights the trade-off between the strict consistency and data integrity offered by **RDBMS**, and the horizontal scalability and schema flexibility inherent in **NoSQL** systems.

Feature	Relational (SQL/RDBMS)	Databases	Non-Relational (NoSQL)	Databases
Data Model/Schema	Structured, rigid schema defined by tables and columns [33]		Flexible, dynamic, or schemaless; supports key-value, document, graph, and column family models [41]	
Consistency Model	Strong Consistency (ACID compliant); prioritizes consistency and integrity [59]		Eventual Consistency (BASE compliant); prioritizes availability and partition tolerance [59]	
Scalability	Vertical Scaling (scaling up); faces limitations in horizontal scaling for big data [61]		Horizontal Scaling (scaling out) via distribution/sharding; ideal for massive, growing data volumes [53]	
Query Language	Standardized SQL ; highly efficient for complex queries and joins [9]		Varies by vendor (e.g., MQL, Cypher); generally lacks standardized language; limited/no native JOIN operations [29]	
Security/Integrity	Strong support for integrity rules (foreign keys) and robust security mechanisms [59]		Limited initial security features; often relies on application-level integrity checks [59]	
Best Suited For	Transactional systems, structured data, scenarios requiring complex joins, and data integrity (e.g., banking, ERP) [59]		Large-scale IoT /sensor data, unstructured/semi-structured data, high velocity data ingestion, and rapid growth [50]	

Table 2.3: Comparative analysis of key characteristics between Relational and Non-Relational databases.

2.4 Standards and Interoperability via FIWARE and NGSI-LD

To achieve system flexibility and promote cross-platform interoperability—a crucial goal for smart city solutions—the adoption of standardized data models and open platforms is paramount [58].

2.4.1 The FIWARE Ecosystem and Context Management

The FIWARE initiative is a freely available framework established as part of a European initiative involving collaboration between the public and private sectors to shape the future Internet. The goal of these initiatives is the creation of an open, sustainable ecosystem around public, royalty-free, and implementation-driven software platform standards, easing the development of smart

solutions and supporting organizations in their transition into smart organizations. FIWARE operates on a modular structure underpinned by a range of software components known as Generic Enabler (GE)s. GEs provide functionalities and standards designed to make the development of smart applications easier. They simplify the interaction with IoT systems, contextual data, API management, and information processing and analysis [40, 47].

At the core of any FIWARE-based solution is the Context Broker, which is an essential and mandatory component. The Context Broker manages the entire lifecycle of context information, including updates, queries, registrations, and subscriptions. It serves as a central interoperability hub that links all components of the system [35]. Context data managed by the broker is represented through values assigned to attributes that characterize entities. An entity acts as a Digital Twin, digitally representing a real-world physical asset (e.g., a bus in a city, a milling machine in a factory) or a concept (e.g., a weather forecast) [47].

Context brokers manage the context information using a decentralized and scalable approach and mediate the interactions applied to data entities, making them available through REST APIs. Examples of Context Brokers available in the FIWARE catalogue include Orion (based on NGSI-V2), Orion Context Broker with Linked Data Extensions (Orion-LD), Scorpio, and Stellio [40, 47].

2.4.2 FIWARE Context Broker Performance and Scalability

Research on urban platform components reveals distinct performance trade-offs among FIWARE Context Brokers. Newer microservice-based architectures, such as Stellio and Scorpio, utilize asynchronous query execution to demonstrate superior scalability under high loads. In contrast, the traditional Orion-LD architecture suffers from significant latency due to its reliance on threads and blocking I/O [43].

However, as illustrated in Figure 2.4, Orion-LD maintains a significant advantage in data ingestion speed. While execution times naturally increase for all brokers as batch sizes grow, Orion-LD consistently outperforms its counterparts. This advantage stems from its use of MongoDB, which employs write-optimized Log-Structured Merge (LSM) Trees. LSM Trees write data to fast in-memory structures before merging to disk, minimizing write amplification [43].

Conversely, Stellio and Scorpio rely on PostGIS, which uses B+ Tree structures. While B+ Trees are efficient for read-heavy operations, they incur overhead during high write loads due to the necessity of maintaining a balanced structure, resulting in slower batch processing. Despite these specific strengths, all three brokers face challenges regarding operational costs and bottlenecks when processing temporal queries and managing large volumes of historical time-series data [43].

2.4.3 The NGSI-LD Standard

The underlying API for the FIWARE ecosystem relies on the NGSI standard, which evolved from specifications originally issued by the Open Mobile Alliance (OMA) [47]. The OMA NGSI

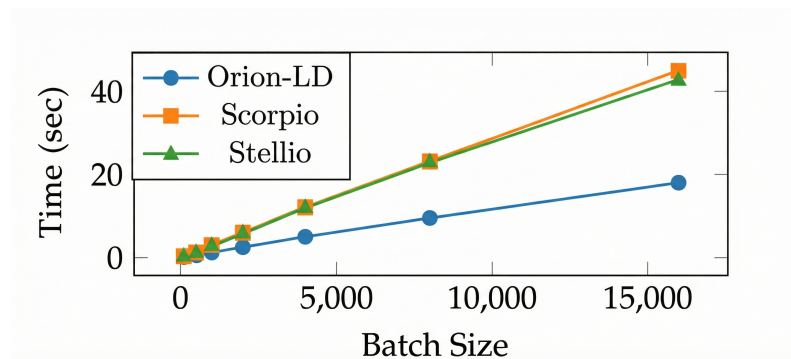


Figure 2.4: Context broker ingestion time [43].

data model offers a common approach for modeling and exchanging data in **IoT** environments, fostering consistency and compatibility between different systems and devices [40].

NGSI-LD (Next Generation Service Interface - Linked Data) extends this concept and is promoted as the current standard, replacing the maintenance-mode **NGSI-V2**. **NGSI-LD** serves as the standard protocol for developing context-aware applications, particularly within smart cities initiatives, encompassing sectors such as healthcare and transportation [43]. A key feature of **NGSI-LD** is its ability to support linked data and semantic interoperability, facilitating seamless information sharing and utilization across systems and organizations [35].

The process illustrated in Figure 2.5 begins when an **NGSI-LD** request is sent to the context broker. These queries are subsequently translated into database queries and executed to locate the necessary information. Once the context-aware brokers retrieve the raw data, it is transformed back into the standard **NGSI-LD** format. Finally, the data is delivered to the user or application as usable contextual information [43].

2.4.3.1 The NGSI-LD Data Model

The **NGSI-LD** specification is a regularly updated standard published by the European Telecommunications Standardization Institute (**ETSI**). Built upon **JSON-LD**, it functions as both a unified

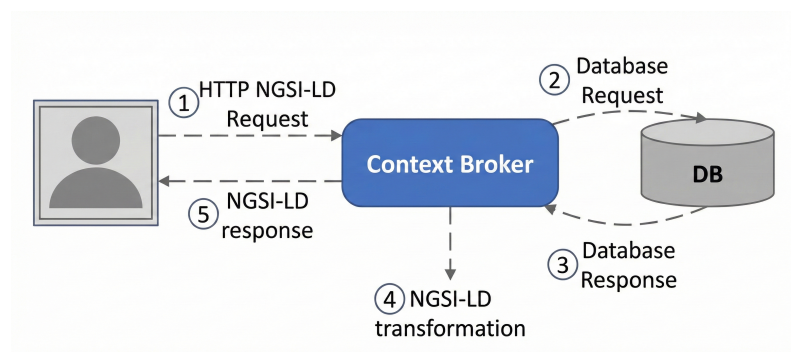


Figure 2.5: Query execution in NGSI-LD context brokers [43].

data model and an **API** designed to exchange and manage contextual data with structured interoperability [35, 40].

The core of the model relies on Entities (Digital Twins), Attributes, and Metadata. The key characteristics of these entities include:

- **Universal Identification:** Every entity is uniquely identified using a Uniform Resource Identifier (**URI**) [47].
- **Typing:** Entities are categorized by specific types (e.g., a Room or Bus), which are also defined by URIs [47].
- **Characterization:** Entities are defined by attributes, which are distinguished as either Properties (containing data values, such as speed) or Relationships (linking different entities together) [47].

To ensure consistency, these definitions leverage Smart Data Models from the FIWARE Foundation, creating a harmonized semantic representation for applications. Additionally, the **NGSI-LD** standard includes a Temporal **API**, enabling the comprehensive management of historical data alongside current context [47].

2.4.3.2 Persistence and Application Examples in FIWARE

FIWARE ensures data persistence through a combination of Context Brokers and specialized **GEs**. Brokers vary in their approach: Orion-LD stores only the current state of Digital Twins using MongoDB, while Stellio manages both current and historical data within PostgreSQL or TimescaleDB. To enhance resilience and decouple communication, architectures often incorporate Apache Kafka [10, 20, 43].

When a broker lacks native historical management, external **GEs** bridge the gap. Cygnus acts as middleware to route data streams to various storage sinks (like **SQL** or Hadoop), while Draco handles real-time data transformation, often for machine learning integration. Additionally, Mintaka and QuantumLeap provide specialized **APIs** for retrieving temporal and spatial-temporal data, leveraging databases such as TimescaleDB and CrateDB respectively [17, 35, 47].

2.4.3.3 Smart Mobility Applications

FIWARE components and the **NGSI-LD** standard are pivotal for advancing smart mobility solutions. A prime example is the GreenMov project in Nice, which integrates heterogeneous data—such as air quality, traffic, and public transport—to provide recommendations that reduce environmental impact. This implementation relies on Orion-LD and Mintaka, using TimescaleDB for robust data persistence [18].

To address data heterogeneity in transportation, the ecosystem employs **IoT** Agents to translate native device protocols (like Message Queuing Telemetry Transport (**MQTT**)) into the standard **NGSI-LD** format [10, 40]. This allows for the effective modeling of assets like buses and stations,

proving that the FIWARE ecosystem is a viable solution for complex Cyberphysical Systems (CPS), including those utilizing TinyML and Edge Computing [17].

2.5 Security, Authentication, and Access Control

Given the sensitivity of urban data, especially personal mobility information, implementing robust security and compliance (such as with General Data Protection Regulation (GDPR)) is a core necessity [2, 33].

Regulatory compliance has become crucial in IoT, requiring the development of data governance tools and methodologies for enterprises and cloud data centers to verify the lawful collection, usage, and sharing of data. Specifically, laws such as the European Union's GDPR have been framed to govern the lawful processing of personal data [6]. Personal data, defined as any information relating to an identified or identifiable natural person, includes explicit identifiers, quasi identifiers (like ZIP code or date of birth), and sensitive information (like health data) [2].

2.5.1 Security Challenges and Compliance

Security and privacy represent critical requirements and one of the most difficult challenges faced by databases, especially NoSQL databases, due to their reliance on sharding (data distributed over multiple servers) [53].

2.5.1.1 Key Challenges in Distributed Environments

In distributed environments, particularly within the IoT, security presents significant challenges centered on data vulnerability, integrity, and overall system complexity. The issue of insecure data is magnified in IoT contexts due to factors such as weakly secured clients, the use of heterogeneous protocols, and the necessity for data to pass through multiple places. This vulnerability is further compounded by the outsourcing of processing to edge or fog nodes, which expands the potential attack surface [6].

Simultaneously, ensuring data integrity and confidentiality is crucial to prevent unauthorized tampering or eavesdropping. However, in cloud contexts where processing is offloaded to external servers, data owners effectively lose control, creating risks where personal data may be exposed to malicious entities [37]. Ultimately, securing cloud-based applications is inherently more complex than securing centralized systems due to the vast number of interacting components, the intricacies of their communication, and the dispersed physical locations of stored data [19]. Furthermore, the implementation of robust security measures in these distributed systems frequently results in unavoidable performance degradation [53].

2.5.1.2 Encryption Mechanisms for Confidentiality and Integrity

Encryption techniques are usually adopted to deal with integrity and confidentiality issues. Data encryption plays a central role in data protection and helps guarantee privacy for sensitive information. Mechanisms supporting end-to-end encryption are preferable, especially in **IoT**-related and fog scenarios where information is stored distributively and transmitted via hop-by-hop communications [53].

2.5.2 Access Control and Authentication

In the realm of big data governance, access control systems serve as the central mechanism for balancing the critical, often competing needs of data protection and data sharing. These systems are essential for maintaining security while ensuring that data remains accessible to authorized users within dynamic environments [2].

2.5.2.1 Key Requirements for Big Data Access Control

Access control solutions must be sophisticated enough to navigate the complexities of big data ecosystems, prioritizing efficiency even in collaborative settings. A primary requirement is the enforcement of security measures across the entire data lifecycle, protecting information whether it is at rest, in use, or in motion. To ensure this, some systematic approaches advocate for executing access control policies at the exact moment of data ingestion. Furthermore, these systems must possess high efficiency and scalability to handle the increasing amount of data and the rapid rate of access requests. This is particularly challenging in distributed environments like **NoSQL** databases, where security functionalities can potentially lead to performance degradation [2].

Finally, Fine-Grained Access Control (**FGAC**) is essential, allowing for policies that specify access conditions based on context and specific data attributes. This enables precise control over both structured and unstructured data elements [53].

2.5.2.2 Access Control Models

To achieve the necessary level of granular control over database resources, various paradigms have been implemented, with Attribute-Based Access Control (**ABAC**) and Role-Based Access Control (**RBAC**) being the most prominent [53].

ABAC is widely identified as a promising solution, particularly for the dynamic environment of the **IoT**. It allows data owners to encrypt information based on specific access criteria, ensuring that only users with the necessary permissions can decrypt and view the data. This method couples access permissions directly with encryption, adding an intrinsic layer of security. **ABAC** supports context-sensitive conditions, regulating access to granular units such as a single cell, a column, or a specific portion of an unstructured file. However, a notable drawback is that updating policies in distributed systems can result in high communication overload, as data may need to be re-encrypted to align with new policies [53].

Conversely, **RBAC** simplifies authorization management by regulating permissions based on user roles rather than individual users. For instance, MongoDB integrates **RBAC** mechanisms and Neo4j provides a simple implementation, but many **NoSQL** systems offer only limited support compared to conventional standards [53].

2.5.2.3 Authentication and Authorization

In big data environments, access control primarily emphasizes authorization to regulate the release of information, while authentication is typically managed through separate, integrated modules to support federation. To ensure secure communication, protocols like Transport Layer Security (**TLS**) and Datagram Transport Layer Security (**DTLS**) are employed for end-to-end encryption, safeguarding data integrity and confidentiality during transit as seen in systems like Neo4j [53].

Furthermore, OAuth2 serves as a critical standard, widely used in **IoT** platforms to provide a security layer with proxy functionality that facilitates collaboration between security and development teams [57]. Beyond these protocols, specific databases offer distinct integration capabilities; for instance, MongoDB aligns with external mechanisms such as Kerberos, Lightweight Directory Access Protocol (**LDAP**), and X.509 certificates, whereas Cassandra relies on standard username and password credentials [53].

2.5.3 Secure RESTful APIs

APIs play an indispensable role in distributed, scalable environments like **IoT** and smart cities, facilitating the requisite ingestion, processing, and retrieval of extensive datasets [43]. To classify **REST API** vulnerabilities systematically, a hierarchical taxonomy is proposed and visually represented in Figure 2.6. This classification system segments security threats into three overarching groups: Authentication & Authorization Issues, Data Validation Flaws, and Configuration & Deployment Issues [57].

2.5.3.1 API Role and Architecture in Smart Systems

REST APIs act as the backbone of modern smart city and **IoT** infrastructures, utilizing stateless HTTP protocols to facilitate seamless, standardized communication between cloud services and clients. This architectural approach supports horizontal scalability through microservices, effectively isolating business logic from diverse data layers, including both **SQL** and **NoSQL** databases [39]. Furthermore, these **APIs** serve a critical role in standardized ecosystems like FIWARE, where context brokers leverage compliant protocols (such as **NGSI-LD**) to ensure secure interoperability and data exchange within a secure architecture across complex urban environments [18].

2.5.3.2 The Necessity of Security and Associated Threats

While **REST APIs** provide significant functional benefits, their widespread adoption and accessibility via the Internet have made them prime targets for cyberattacks. Recent data indicates a

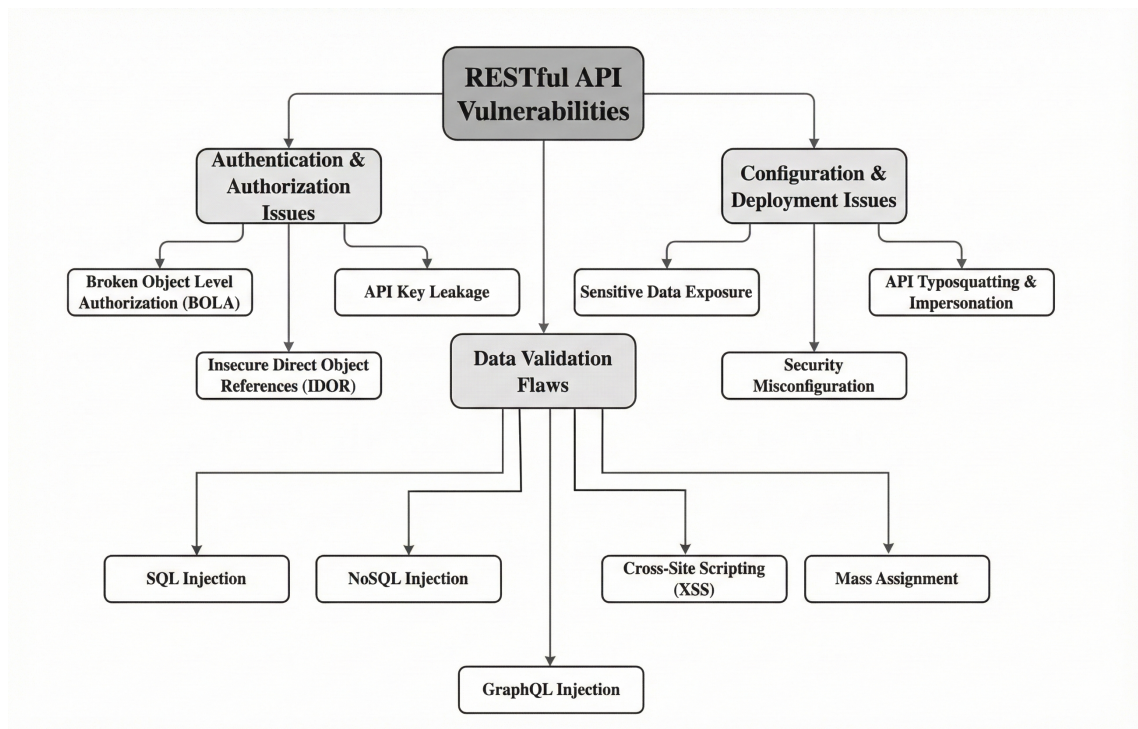


Figure 2.6: Taxonomy of RESTful API vulnerabilities [57].

surge in threats, with 60% of organizations reporting **API**-related breaches in a two-year period. The stateless nature of these **APIs** requires clients to send complete information with every request, creating opportunities for sensitive data leakage if not properly secured [57].

Based on industry standards like the OWASP [45] **API Security Top 10** (2023), the critical risks fall into two main categories: "Authentication & Authorization Vulnerabilities", and "Data Validation, Injection & Configuration Flaws" [57].

Firstly, Authentication & Authorization Vulnerabilities are fundamental security failures regarding who can access what data.

- **Broken Object Level Authorization (BOLA)**: Failures in authorization checks allow users to access resources (objects) they do not own, leading to privilege escalation [57].
- **Insecure Direct Object Reference (IDOR)**: Attackers manipulate object references to access unauthorized data [57].
- **Broken Authentication**: Flaws in authentication mechanisms (e.g., **API** key leakage) allow attackers to bypass security and gain unauthorized access [57].

Secondly, Data Validation, Injection & Configuration Flaws risks stem from how data is processed and how the environment is set up.

- **Injection Flaws (Structured Query Language Injection (SQLi) / Not only SQL Injection (NoSQLi))**: Attackers exploit unsanitized inputs to execute malicious commands and manipulate sensitive data [57].

- **Mass Assignment:** User-provided data is automatically mapped to internal models without filtering, allowing users to modify restricted fields (e.g., permissions) [57].
- **Configuration Issues:** Vulnerabilities such as Excessive Data Exposure (returning more data than the client needs) and Security Misconfiguration (insecure default settings) significantly increase the attack surface [57].

2.5.3.3 Safeguarding the Data Layer through Implementation

Authentication and robust communication security begins with OAuth2 for standardized access control and TLS / DTLS for encrypted communication, ensuring data integrity during transit. To strictly verify identities, the system employs JSON Web Tokens (JWT) to manage user roles and X.509 certificates to authenticate physical IoT devices, preventing unauthorized data transmission [40].

Vulnerability Mitigation and Secure Design Defending against exploits requires enforcing object-level authorization and using indirect references to prevent IDOR and BOLA attacks. Developers must stop injection flaws by using parameterized queries and ORM libraries, while preventing data exposure through data minimization and strictly whitelisting response fields [44, 57].

2.6 Hybrid Solutions and the NewSQL Paradigm

This section explores the convergence of relational and non-relational databases through hybrid architectures and the NewSQL paradigm. It first examines how NewSQL balances strict ACID compliance with horizontal scalability for high-throughput environments. Next, it analyzes practical deployments of polyglot persistence across unified abstraction layers, mobile ecosystems, and Smart City IoT infrastructures. Finally, a comparative analysis evaluates these architectures against the operational constraints of the UrbanData Nexus platform, identifying key technological advantages alongside critical limitations in performance optimization and granular GDPR compliance.

2.6.1 The NewSQL Paradigm: Synthesizing Consistency and Scalability

The NewSQL paradigm represents a significant evolution in database architecture, designed to synthesize the strict ACID guarantees and SQL compatibility of traditional RDBMS with the high availability and horizontal elasticity of NoSQL systems. This hybrid approach is particularly advantageous for high-throughput environments requiring strong consistency, such as e-commerce platforms, financial transaction systems, and real-time analytics [51, 52, 59].

VoltDB serves as a prominent example of this architecture, as illustrated in Figure 2.7. The cluster architecture (depicted on the right) consists of server nodes housing individual processors, referred to as sites, which manage distinct VoltDB partitions. A detailed schematic of an individual

site (depicted on the left) reveals its internal components: a dedicated Work Queue, a complete instance of the Execution Engine, and the localized table and index data [52].

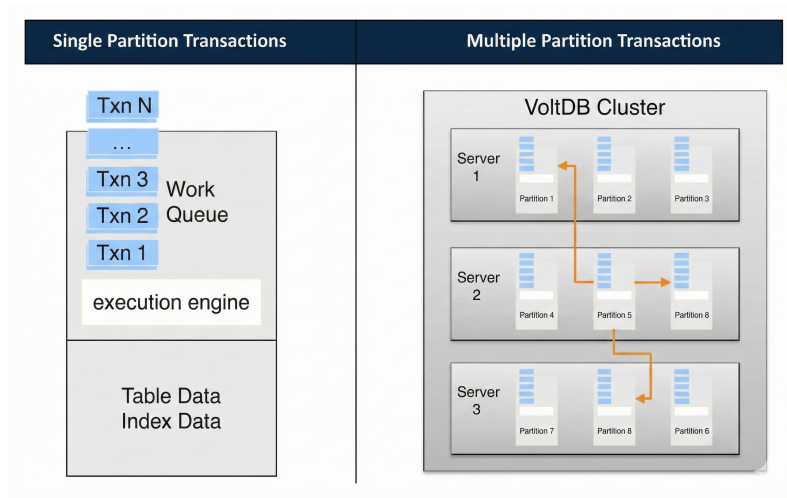


Figure 2.7: Overview of the VoltDB architecture [52].

2.6.2 Architectural Integration and Practical Implementations

While the relational (**SQL**) and non-relational (**NoSQL**) paradigms operate on divergent theoretical frameworks, contemporary engineering efforts focus primarily on facilitating their interoperability within singular computational ecosystems. Such architectural strategies, formally designated as polyglot persistence, demand sophisticated integration mechanisms to reconcile data heterogeneity. The following analysis elucidates the structural taxonomy of these hybrid environments, spanning generalized abstraction models to specialized deployments in mobile computing and Smart City **IoT** infrastructures.

To conceptualize the interoperability of these paradigms within a unified framework, Figure 2.8 delineates the underlying architecture designed to bridge relational and non-relational components.

Drawing upon the abstraction model proposed by [9], this design mitigates the impedance mismatch between structured and semi-structured data sources. By establishing a uniform interface layer, the architecture effectively abstracts the underlying structural divergences, thereby fostering a seamless hybrid **SQL/NoSQL** environment that decouples application logic from the intricacies of data persistence.

Figure 2.9 demonstrates a practical application of this hybrid methodology within the resource-constrained context of mobile environments. As articulated by [49], the proposed architecture creates a synchronized data pipeline by integrating a local **SQL** server for offline capability with a **NoSQL** Firebase Firestore database for cloud-based scalability. Crucially, this implementation utilizes direct query access alongside Change Data Capture (**CDC**) mechanisms to ensure bidirectional consistency and minimize synchronization latency. The efficacy of this approach

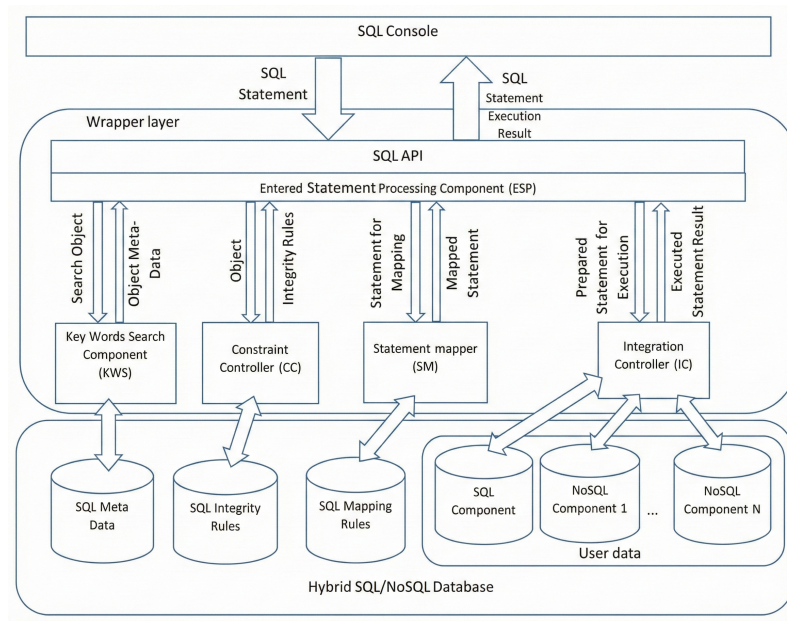


Figure 2.8: Display of the architecture for integration and uniform use of components of hybrid SQL/NoSQL database [9].

was empirically validated through the Flutter development framework, utilizing the Repeato automated testing suite to benchmark the system's throughput and latency performance under hybrid constraints.

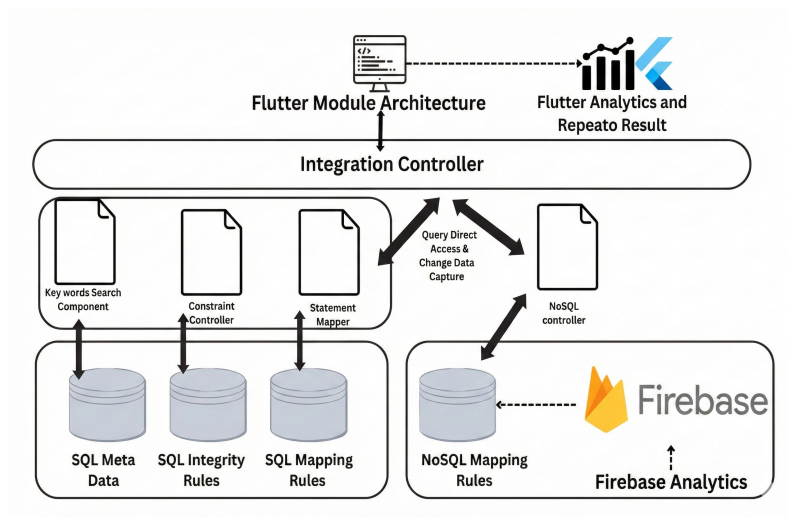


Figure 2.9: System Architecture Design for Android applications [49].

Furthermore, the complexity of hybrid database architectures is exemplified within Smart City ecosystems, where data volume and velocity require robust middleware solutions. [47] presents a comprehensive framework wherein the "Context Broker" functions as the central pivot for integration and interoperability.

As illustrated in Figure 2.10, the backend persistence technology is contingent upon the specific implementation of the Context Broker. Specifically, the architecture reveals a divergent technological stack: the Orion and Orion-LD brokers leverage the document-oriented flexibility of MongoDB (NoSQL), whereas the Scorpio broker utilizes the relational rigidity of PostgreSQL (SQL), and Stellio operates on TimescaleDB (SQL) to optimize for time-series data ingestion.

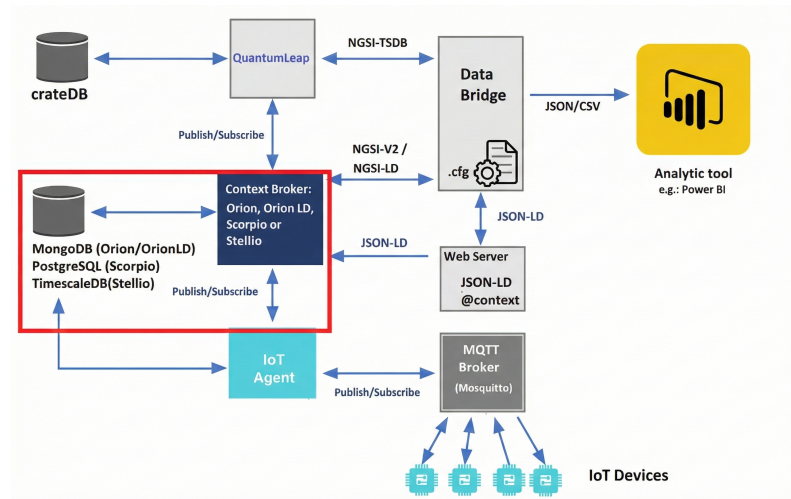


Figure 2.10: Hybrid database Architecture in Smart City ecosystems [47].

2.6.3 Comparative Analysis of Solutions for UrbanData Nexus

Solution / Architecture	Positive Points for UrbanData Nexus	Negative Points for UrbanData Nexus
Unified Hybrid Framework (Wrapper Layer) [9]	Architected a seamless hybrid database strategy, unifying SQL and NoSQL components into a single logical data store to streamline system architecture. Engineered a custom Constraint Controller to enforce referential integrity across heterogeneous urban datasets, including IoT sensors and physical infrastructure. Developed a unified API layer that abstracts underlying data types, enabling the creation of RESTful services using standard SQL commands.	Performance Bottlenecks: The processing and mapping components create overhead that may prevent the system from handling high-volume mobility data in real-time. Scalability & Maintenance: The architecture has high implementation complexity; adding new urban data sources requires labor-intensive updates to metadata dictionaries and mapping rules.
IoT Data Bridge (FIWARE Middleware) [47]	Interoperability: The middleware achieves full compliance with project objectives by natively integrating with the FIWARE ecosystem and NGSI-LD standards. Data Transformation: It solves the "impedance mismatch" technical challenge by converting hierarchical JSON data (NGSI-LD) into the tabular formats necessary for urban analytics tools. Security: It establishes a security baseline by utilizing existing authentication protocols at both the service and tenant levels.	Scalability Issues: Performance degrades significantly with large datasets, making city-scale deployments risky. Compliance Gaps: Security is limited to the tenant level, lacking the fine-grained, entity-level control required for GDPR compliance. Workload Constraints: The system is optimized for read-only analytics and is ill-suited for heavy transactional processing.

Solution / Architecture	Positive Points for UrbanData Nexus	Negative Points for UrbanData Nexus
NewSQL Architectures (VoltDB, MemSQL) [52]	High Scalability & Throughput: Optimizes the system for ingesting massive streams of real-time sensor data. Strict Data Consistency: Full ACID guarantees ensure reliability for critical operations (e.g., traffic management). Ultra-Low Latency: In-memory processing enables immediate response times for real-time decision platforms.	High Cost & Resource Usage: In-memory designs are too expensive and resource-intensive for storing large volumes of historical data. Performance Issues: Some NewSQL solutions suffer from high latency and low throughput, failing to meet real-time operational constraints. Poor Integration: These systems operate as standalone engines, requiring cumbersome data migration rather than integrating seamlessly into existing hybrid architectures.
Mobile Hybrid Architecture (QDA & CDC) [49]	Engineered a Change Data Capture pipeline to synchronize local and cloud databases in real-time, ensuring digital twin accuracy and automated SQL/NoSQL consistency. Automated cross-database consistency (SQL/NoSQL) and digital twin alignment by implementing a real-time Change Data Capture mechanism. Enabled real-time sensor monitoring by deploying CDC synchronization between local and cloud environments, removing the need for manual data reconciliation.	Tech Stack Mismatch: It relies on mobile-centric technologies (Android, SQLite, Firebase) rather than robust server-side solutions necessary for a city-scale platform. Unacceptable Latency: The CDC mechanism creates significant logging and transmission delays, making real-time processing impossible. Critical Reliability Risks: The system's performance is too dependent on network stability, posing a danger to critical infrastructure during connectivity drops.

Table 2.4: Comparison of hybrid and database architectures for the UrbanData Nexus platform.

As detailed in Table 2.4, while various existing solutions offer specific advantages for the UrbanData Nexus, none provide a fully feasible implementation.

The most aligned approach is the **IoT Data Bridge** developed by [47], which successfully addresses the interoperability of **NGSI-LD**. However, this middleware presents significant limi-

tations regarding the UrbanData Nexus objectives. Specifically, the system lacks caching mechanisms and code optimization, leading to performance bottlenecks when handling large datasets.

Furthermore, its current security model operates only at the tenant level without granular access control. This creates a critical compliance gap regarding **GDPR** requirements, making it unsuitable for the secure hybrid architecture envisioned for UrbanData Nexus platform.

2.7 Summary

This chapter has provided a comprehensive review of the background and state-of-the-art technologies requisite for the development of intelligent, scalable urban mobility systems [30, 38]. The analysis began by defining the constraints imposed by modern Urban Data, characterized by the "Vs" of Big Data—Volume, Variety, and Velocity [5, 13, 41, 50, 58]. It was established that legacy systems are insufficient for these demands [33], necessitating a shift toward robust, distributed infrastructure capable of real-time processing and horizontal scaling [51, 53].

A significant portion of the chapter was dedicated to the comparative analysis of data storage paradigms. The text examined the trade-offs between Relational Database Management Systems (**SQL**), which offer **ACID** compliance and data integrity, and Non-Relational (**NoSQL**) databases [50, 59], which prioritize horizontal scalability and schema flexibility through the **BASE** consistency model [29, 52].

Furthermore, the critical role of **TSDBs** was highlighted, with evidence showing that specialized engines like InfluxDB offer vastly superior storage efficiency (compressing data significantly more than traditional **RDBMS**) and performance for timestamped **IoT** sensor data [20, 34].

Within the domain of interoperability, this analysis examined the FIWARE ecosystem and the associated **NGSI-LD** standard. Such open specifications serve as fundamental mechanisms for circumventing vendor lock-in and establishing semantic interoperability via Digital Twins. The comparative performance assessment of FIWARE Context Brokers revealed distinct structural synergies: Orion-LD, capitalizing on MongoDB's **LSM**-tree architecture, yields superior throughput in write-heavy ingestion workloads. In contrast, relationally backed brokers like Stellio and Scorpio, utilizing PostGIS and B+ trees, provide robust capabilities for spatial query resolution [6, 17, 43].

The challenges of securing distributed **IoT** environments were explored, emphasizing compliance with regulations such as **GDPR**. The analysis of **REST API** security identified critical vulnerabilities, ranging from injection flaws to **BOLA**, and outlined necessary mitigation strategies, including the implementation of OAuth2 [57], **JWT** for stateless authentication, and granular **ABAC** [2, 53].

Finally, the comparative analysis of existing solutions, particularly within the context of the UrbanData Nexus project, identifies a clear research gap. While current middleware solutions like the **IoT** Data Bridge successfully achieve **NGSI-LD** interoperability, they fail to meet the strict requirements for high-performance caching and code optimization necessary for city-scale deployments. Moreover, these existing frameworks lack the granular, entity-level security mechanisms

required for rigorous regulatory compliance. Therefore, the state of the art confirms the need for a novel architectural approach that integrates the interoperability of FIWARE with a highly optimized, secure hybrid storage engine capable of handling real-time mobility data without compromising privacy or performance.

In conclusion, current research indicates that an effective intelligent mobility platform requires a composite architecture. It must integrate hybrid storage solutions and standardized semantic protocols, all underpinned by robust security to safeguard data both in transit and at rest [16, 18].

Chapter 3

UrbanData Nexus

This chapter presents the comprehensive architectural and technical solution for the UrbanData Nexus. Designed to address the inherent complexities of smart city data management, the proposed framework establishes a robust, highly scalable ecosystem for both real-time contextual awareness and longitudinal analytics.

The following sections detail the core operational components of this platform, beginning with the bifurcated hybrid data architecture that separates ephemeral state representations from historical data repositories. Subsequent sections explore the data transformation pipelines responsible for harmonizing heterogeneous sources into the **NGSI-LD** standard, followed by an analysis of the interactive visualization and user interface layers. Furthermore, the chapter outlines the stringent identity and access management protocols enforced across the system, concluding with a detailed examination of the **IoT** ingestion and device connectivity layer. Together, these modular components orchestrate a secure, interoperable, and highly performant data infrastructure.

To support these operations, the system relies on a well-defined structural foundation. The high-level layered architecture of the proposed platform, which integrates the aforementioned components, is illustrated in Figure 3.1.

3.1 Hybrid Data Architecture Design

The proposed architectural framework¹ implements a strictly bifurcated data management paradigm within the FIWARE ecosystem, deliberately decoupling ephemeral state representations from longitudinal data repositories. As illustrated in Figure 3.2, the architecture handles ingestion from **REST APIs** and FIWARE **IoT** Agents, routing the telemetry through distinct pathways based on its temporal requirements. To mitigate the risk of computationally intensive analytical processes degrading the low-latency constraints of live operations, the management of real-time context is centralized via the Orion-LD Context Broker.

¹The source code and deployment configurations for this architecture are available at: <https://github.com/CADX03/URBAN-NET-Database>

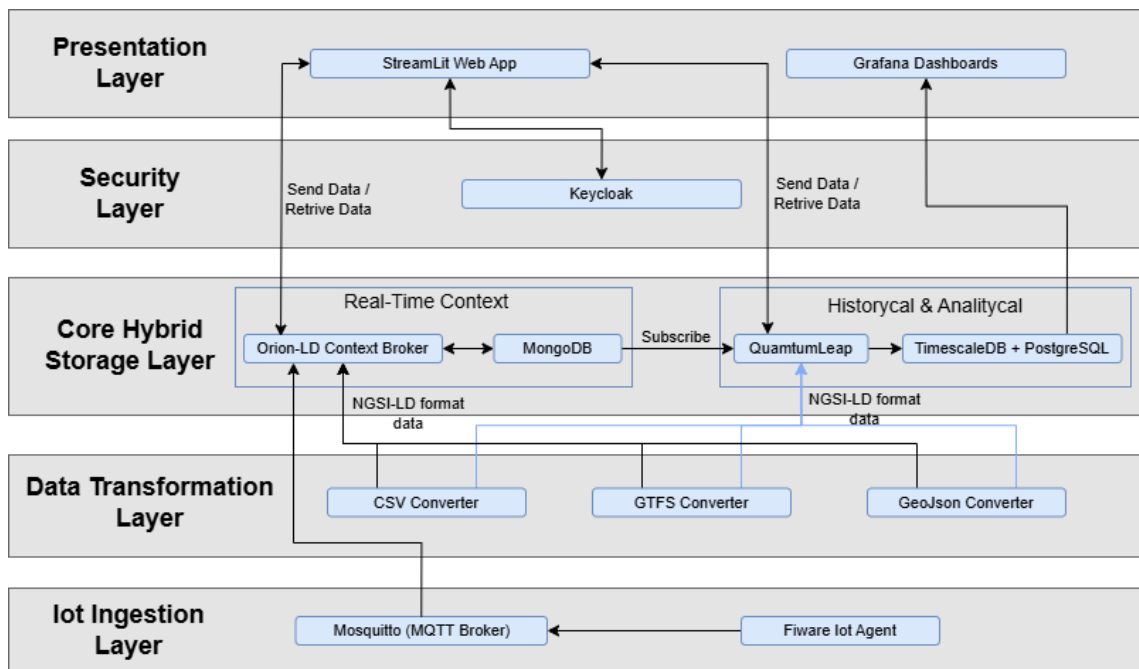


Figure 3.1: Layered architecture of the UrbanData Nexus.

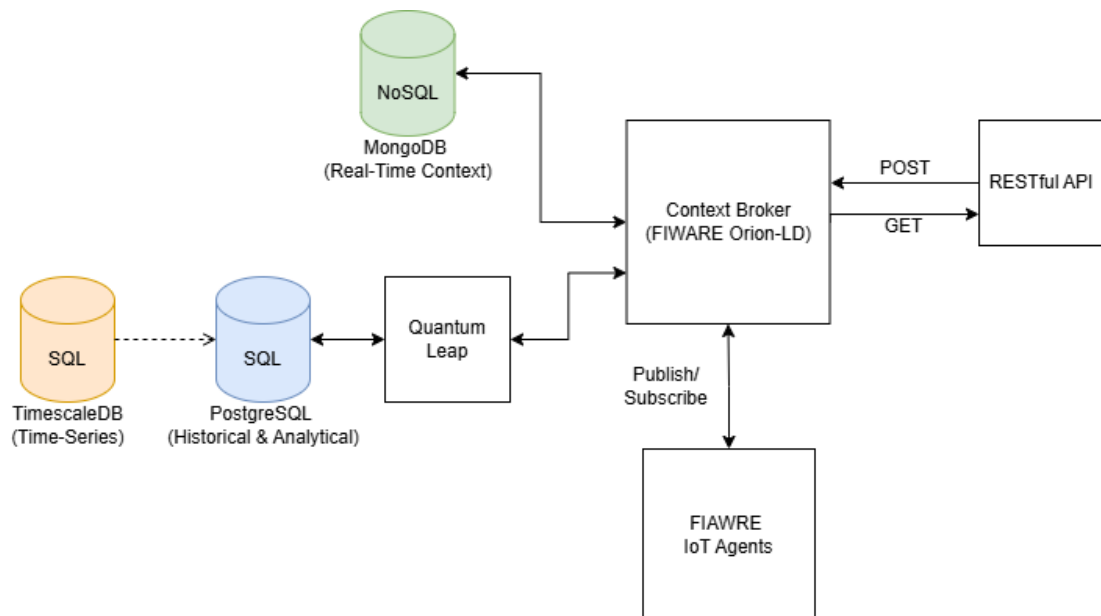


Figure 3.2: Hybrid Data Architecture Diagram.

3.1.1 Real-Time Context Management

Serving as the operational core of the architecture, this layer functions as a centralized broker for the contemporaneous state of all physical and virtual entities within the ecosystem. Within this framework, the Orion-LD Context Broker operates as the primary architectural nexus, systematically ingesting **NGSI-LD** telemetry updates from the **IoT** Agent while processing asynchronous queries from the presentation layer. Through this mechanism, it maintains the authoritative single source of truth regarding immediate systemic and environmental conditions.

Directly supporting this broker is MongoDB, which functions as the foundational **NoSQL** datastore to maintain the current contextual state alongside **IoT** device provisioning registries. By utilizing this configuration, the architecture leverages MongoDB's structural optimization to guaranty high-throughput, low-latency read and write operations across dynamic, document-oriented data structures.

To facilitate the insertion of data into this persistence layer, external systems interface directly with the Orion-LD Context Broker rather than executing direct database writes to MongoDB. Data ingestion occurs via the broker's **REST API**, specifically utilizing endpoints such as *entity-Operations/upsert*, ensuring all incoming telemetry is properly validated through the architectural nexus. Incoming JSON payloads are processed using the appropriate HTTP content types (*application/ld+json* or *application/json*, depending on the presence of an *@context* directive) to maintain strict **NGSI-LD** compliance (see Listing 3.1). Whether handling single-entity updates or multi-status batch operations, this **API-driven** approach reliably pipelines external telemetry to the broker, which subsequently executes the optimized write operations to the underlying MongoDB datastore.

Listing 3.1: Python implementation for dynamic NGSI-LD payload ingestion and batch response handling via the Orion-LD REST API.

```

1 try:
2     with open(file_path, 'r') as file:
3         raw_payload = json.load(file)
4
5     # 1. Determine if the payload contains @context
6     # We check the first item in the list (assuming it's a batch array)
7     has_context = False
8     if isinstance(raw_payload, list) and len(raw_payload) > 0:
9         has_context = '@context' in raw_payload[0]
10    elif isinstance(raw_payload, dict):
11        has_context = '@context' in raw_payload
12
13    # 2. Dynamically assign the correct Content-Type header
14    headers = {}
15
16    if has_context:
17        headers['Content-Type'] = 'application/ld+json'

```

```

18     else:
19         headers['Content-Type'] = 'application/json'
20
21         # 3. Send the request
22         response = requests.post(url, headers=headers, json=raw_payload)
23
24         if response.status_code == 201:
25             return "Entity created successfully."
26         elif response.status_code == 207:
27             return f"Batch operation returned 207 (Multi-Status).\nBroker Response:\n{
                json.dumps(response.json(), indent=2)}"
28         else:
29             raise Exception(f"Broker rejected request. Status Code: {response.
                status_code}. Response: {response.text}")

```

Conversely, to access the persisted telemetry and contextual state, downstream services interact with the same architectural nexus rather than querying MongoDB directly. Data retrieval is managed exclusively through the Orion-LD Context Broker's **REST API**, specifically utilizing the standard entities endpoint via HTTP GET operations. This **API-centric** abstraction layer allows for flexible data extraction, supporting both highly granular queries for specific devices via safely encoded unique Entity IDs, as well as broader aggregations that filter data by entity type using standardized query parameters. By funneling all read operations through the broker, the architecture guarantees that consumers consistently receive the most current, strictly validated **NGSI-LD** representations of the stored documents, thereby preserving the integrity and encapsulation of the underlying **NoSQL** persistence layer.

3.1.2 Historical Data & Analytics

The implementation of a rigorously bifurcated hybrid architecture necessitates the intentional decoupling of longitudinal datasets from the real-time context broker. This separation ensures that computationally intensive analytical workloads do not compromise the low-latency requirements of live state management. Within this paradigm, QuantumLeap serves as a critical interoperability layer, functioning as a data-routing bridge that subscribes to state transitions within the Orion-LD broker.

Upon the detection of telemetry updates, QuantumLeap intercepts and parses these transitions, systematically transforming **NGSI-LD** spatial-temporal data into a normalized schema optimized for relational time-series ingestion.

The persistence layer for this historical data is subsequently anchored by TimescaleDB, a robust analytical backend built upon PostgreSQL. By augmenting standard PostgreSQL architecture with specialized time-series extensions, TimescaleDB facilitates the highly efficient storage and indexing of massive historical datasets. Consequently, this repository empowers downstream visualization interfaces and analytical workflows to execute complex aggregations, statistical evaluations, and multidimensional temporal queries without compromising the system's operational

core.

To facilitate the bulk ingestion or initial population of large-scale datasets, data is programmatically injected into QuantumLeap via its `/v2/notify` **API** endpoint. Rather than transmitting monolithic, resource-intensive payloads, the ingestion pipeline employs a systematic batching mechanism (see Listing 3.2) that segments extensive JSON arrays into discrete, manageable chunks. This approach, which incorporates calculated micro-delays between payload transmissions, proactively mitigates pipeline congestion and prevents computational exhaustion within both QuantumLeap and the underlying TimescaleDB architecture. Consequently, high-volume data streams are reliably ingested into the time-series repository while maintaining strict system stability and data integrity.

Listing 3.2: Python implementation of the throttled batch ingestion pipeline for QuantumLeap.

```

1  # Chunk the data and send in batches
2  for i in range(0, total_entities, batch_size):
3      batch = raw_entities[i:i + batch_size]
4
5      payload = {
6          "data": batch
7      }
8
9      response = requests.post(url, headers=headers, json=payload)
10
11     if response.status_code in [200, 201, 204]:
12         batches_sent += 1
13     else:
14         # If a batch fails, stop everything and raise an error to the frontend
15         print(f"Failed at batch starting at index {i}. Status: {response.
16             status_code}. Response: {response.text}")
17         raise Exception(f"Failed at batch starting at index {i}. Status: {response.
18             status_code}. Response: {response.text}")
19
20     # Brief pause to let QuantumLeap and TimescaleDB breathe
21     time.sleep(0.1)
22
23 # If the loop finishes without raising an exception, return a success summary
24 print(f"All batches sent successfully! Total entities: {total_entities}, Total
25     batches: {batches_sent}")
26 return f"Successfully sent {total_entities} entities across {batches_sent} batches
27     to QuantumLeap."

```

The operationalization of analytical workflows is mediated by QuantumLeap's **REST API**, with particular reliance on the `/v2/entities` endpoint. Through this architectural interface, downstream services can dynamically access the TimescaleDB persistence layer via parameterized HTTP requests, thereby aligning data retrieval with the precise scope of the analysis. This retrieval

mechanism supports both highly granular, entity-specific temporal trajectories, utilizing safely encoded Uniform Resource Name (URN), and broad, system-wide aggregations.

3.2 Data Transformation and NGSI-LD Conversion Methodology

The integration pipeline is designed to ingest heterogeneous data formats—specifically **CSV**, **GeoJSON**, and **GTFS** archives, and harmonize them into the **NGSI-LD** standard. This transformation ensures interoperability, standardized geospatial representation, and linked data compliance across different Smart Data Models, available here <https://smartdatamodels.org/index.php/smart-cities-domain-at-smart-data-models/>.

3.2.1 CSV Data Processing for Traffic Observations

To process tabular **CSV** data representing traffic observations, the system implements a memory-efficient streaming approach for handling large datasets. During ingestion, individual records are systematically mapped to predefined conceptual data models, specifically the *TrafficFlowObserved* ontology.

During the ingestion phase, string variables extracted from the **CSV** file are subjected to automated type inference and casting. To maintain data integrity within the target JSON payload, these values are parsed and converted to their corresponding native numerical types (e.g., casting `TOTAL_VOLUME` as an integer and `AVG_SPEED_ARITHMETIC` as a floating-point number). Additionally, the architecture performs temporal normalization on datetime fields, primarily aggregation periods. For example, the `AGG_PERIOD_START` attribute is parsed, standardized to the ISO 8601 format with a Coordinated Universal Time (UTC) designator (Z), and instantiated as an **NGSI-LD** `DateTime` property (e.g., `2015-01-01T11:55:00Z`).

Beyond primitive data types, the system facilitates the parsing of complex attributes. Custom string formats embedded within the raw dataset—such as serialized axle class volumes (e.g., `{E2:48}`)—are deserialized into native JSON objects. These objects are subsequently encapsulated within standard **NGSI-LD** Property wrappers.

Core attributes including `id`, `type`, `dateObserved`, `equipmentId`, `laneDirection`, `intensity`, `averageVehicleSpeed`, and `occupancy` are successfully mapped and instantiated. These parameters constitute mandatory prerequisites within the designated Smart Data Models context to guarantee strict compliance with the **NGSI-LD** specification.

To operationalize this transformation logic, Listing 3.3 delineates the `map_traffic_flow_observed` function, which serves as the primary mapping engine for individual **CSV** records. This routine processes a raw data row and systematically constructs the foundational **NGSI-LD** entity architecture. It dynamically synthesizes a unique **URN** for the entity's `id` attribute based on the `AGGREGATE_BY_LANE_BUNDLEID` field, reverting to a sequentially generated identifier in the absence of this data. Following the integration of the requisite `@context` URIs, the function iterates through a predefined array of selected columns. Throughout this traversal, it executes the aforementioned temporal normalization and numerical casting protocols, explicitly aligning

domain-specific attributes with their standardized Smart Data Models equivalents. Furthermore, the function is designed with extensibility in mind; it dynamically accommodates additional configured columns by securely encapsulating them as generic **NGSI-LD** Properties.

Listing 3.3: Python implementation mapping raw CSV traffic observations to the NGSI-LD TrafficFlowObserved data model.

```

1 def map_traffic_flow_observed(row, selected_columns, index):
2     """Maps CSV rows specifically to the TrafficFlowObserved data model."""
3     entity_id = row.get('AGGREGATE_BY_LANE_BUNDLEID', f"unknown_{index}")
4
5     entity = {
6         "id": f"urn:ngsi-ld:TrafficFlowObserved:{entity_id}",
7         "type": "TrafficFlowObserved",
8         "@context": [
9             "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld",
10            "https://raw.githubusercontent.com/smart-data-models/dataModel.
11                Transportation/master/context.jsonld"
12        ]
13    }
14
15    for col in selected_columns:
16        val = row.get(col)
17        if val is None or val == "":
18            continue
19
20        if col == "AGG_PERIOD_START":
21            try:
22                date_obj = datetime.strptime(val, "%Y-%m-%d %H:%M:%S")
23                iso_date = date_obj.isoformat() + "Z"
24            except ValueError:
25                iso_date = val
26            entity["dateObserved"] = {"type": "Property", "value": {"@type": "
27                DateTime", "@value": iso_date}}
28        elif col == "EQUIPMENTID":
29            entity["equipmentId"] = {"type": "Property", "value": val}
30        elif col == "LANE_BUNDLE_DIRECTION":
31            entity["laneDirection"] = {"type": "Property", "value": val}
32        elif col == "TOTAL_VOLUME":
33            entity["intensity"] = {"type": "Property", "value": convert_to_number(
34                val)}
35        elif col == "AVG_SPEED_ARITHMETIC":
36            entity["averageVehicleSpeed"] = {"type": "Property", "value":
37                convert_to_number(val)}
38        elif col == "OCCUPANCY":
39            entity["occupancy"] = {"type": "Property", "value": convert_to_number(
40                val)}
41        elif col == "AXLE_CLASS_VOLUMES":

```

```

37     entity["axleClassVolumes"] = {"type": "Property", "value":
38         parse_axle_volumes(val)}
39     elif col == "AGGREGATE_BY_LANE_BUNDLEID":
40         pass # Handled in the entity ID
41     else:
42         entity[col] = {"type": "Property", "value": convert_to_number(val)}
43
44     return entity

```

3.2.2 Relational Transit Data Processing (GTFS)

GTFS data introduces relational complexities and highly structured route information. The system processes the zipped text files using *pandas* for chunked dataframes, mapping the relational database into a linked data graph. This process relies on three primary techniques: linked relationships, geospatial aggregation, and memory optimization.

To establish *linked relationships*, **GTFS** foreign keys (like `route_id` or `shape_id` within a trip) are converted into **NGSI-LD** Relationship objects. For example, a `GtfsTrip` entity utilizes `hasRoute` and `hasShape` attributes to point directly to the URNs of the corresponding route and shape entities. *Geospatial aggregation* is handled distinctly from simple points; **GTFS** shapes are parsed by grouping individual coordinate rows by their shape ID, sorting them by sequence, and compiling them into complex `LineString GeoProperties`.

Finally, for *memory optimization*, the system addresses the exceptionally large size of **GTFS** datasets by utilizing an orchestration function. This function reads the relational files in strict chunks (e.g., 100,000 rows) and subsequently outputs the **NGSI-LD** entities into partitioned JSON files. This prevents out-of-memory errors and ensures smooth downstream ingestion.

To illustrate this transformation process, Listing 3.4 demonstrates the Python implementation that handles context generation, chunked file orchestration, and entity mapping for **GTFS** stops. In this conversion example, the `id`, the `type`, the `name` and `location` attributes are mandatory for the context <https://raw.githubusercontent.com/smart-data-models/data-model-urban-mobility/master/context.jsonld>.

Listing 3.4: Python script for orchestrating GTFS archive processing and converting tabular data into NGSI-LD entities.

```

1 def process_gtfs_zip(uploaded_zip_bytes, selected_domain):
2     """
3     Takes a ZIP file and a list of Smart Data Model domains.
4     Generates the dynamic context and converts the data into chunked files.
5     """
6     # 1. Build the dynamic context based on user selection
7     if (selected_domain != "Generic"):
8         context_list = [

```

```

9         f"https://raw.githubusercontent.com/smart-data-models/dataModel.{
            selected_domain}/master/context.jsonld"
10     ]
11     else:
12         context_list = []
13         # Always include the core NGSi-LD context at the end
14         context_list.append("https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.
            jsonld")
15
16     out_zip_buffer = io.BytesIO()
17
18     # Target ~80MB per JSON file. This is a balance between file size and number of
        files, but can be adjusted as needed.
19     CHUNK_SIZE = 100000
20
21     with tempfile.TemporaryDirectory() as tmp_dir:
22         ...
23
24         file_map = {
25             'agency.txt': ('agencies.json', convert_agency),
26             'stops.txt': ('stops.json', convert_stops),
27             'routes.txt': ('routes.json', convert_routes),
28             'trips.txt': ('trips.json', convert_trips),
29             'stop_times.txt': ('stop_times.json', convert_stop_times),
30             'calendar.txt': ('calendar.json', convert_calendar),
31             'calendar_dates.txt': ('calendar_dates.json', convert_calendar_dates),
32             'shapes.txt': ('shapes.json', convert_shapes),
33             'transfers.txt': ('transfers.json', convert_transfers)
34         }
35         ...
36
37 def convert_stops(df, context_list):
38     return [{
39         "id": f"urn:ngsi-ld:GtfsStop:Porto:{row['stop_id']}",
40         "type": "GtfsStop",
41         "name": {"type": "Property", "value": str(row['stop_name'])},
42         "location": {
43             "type": "GeoProperty",
44             "value": {"type": "Point", "coordinates": [float(row['stop_lon']),
                float(row['stop_lat'])]}
45         },
46         "@context": context_list
47     } for _, row in df.iterrows()]

```

3.2.3 Spatial Data Processing (GeoJSON)

GeoJSON features intrinsically comprise spatial boundaries and associated metadata, facilitating an efficient mapping onto **NGSi-LD** spatial constructs. This conversion process necessitates two

primary operations: geometry translation and property sanitization. Specifically, the geometry object of a given **GeoJSON** feature is directly transposed into an **NGSI-LD** `GeoProperty`, which is subsequently bound to the `location` attribute. Concurrently, the feature's intrinsic properties are iteratively encapsulated within standard **NGSI-LD** `Property` objects.

To preserve structural integrity and mitigate namespace collisions within the linked data graph, the system executes a systematic sanitization of the property keys. This procedure entails substituting whitespace characters with underscores and reallocating reserved JSON-LD keywords. For example, the native `id` and `type` fields are programmatically aliased to `original_id` and `original_type`, respectively.

The programmatic implementation of these transformations is detailed in Listing 3.5. The function outlines the core logic required to convert a standard **GeoJSON** feature, such as an off-street parking facility, into a compliant **NGSI-LD** entity. As enforced by the transformation algorithm, the foundational attributes `id`, `type`, and `location` are strictly required to align with the target Smart Data Models context (<https://raw.githubusercontent.com/smart-data-models/dataModel.Parking/master/context.jsonld>).

Listing 3.5: Python implementation detailing the programmatic conversion of a standard GeoJSON feature into an NGSI-LD compliant entity.

```

1 def create_ngsild_entity(feature, entity_type):
2     """Converts a single GeoJSON feature dictionary into an NGSI-LD entity
3         dictionary."""
4
5     raw_id = feature.get("id")
6     if not raw_id:
7         raw_id = str(uuid.uuid4())
8     entity_id = f"urn:ngsi-ld:{entity_type}:{raw_id}"
9
10    ngsild_entity = {
11        "id": entity_id,
12        "type": entity_type
13    }
14
15    geometry = feature.get("geometry")
16    if geometry:
17        ngsild_entity["location"] = {
18            "type": "GeoProperty",
19            "value": geometry
20        }
21
22    properties = feature.get("properties", {})
23    for key, value in properties.items():
24        if value is None or value == "":
25            continue

```

```

26     safe_key = key.replace(" ", "_")
27     if safe_key == "id":
28         safe_key = "original_id"
29     elif safe_key == "type":
30         safe_key = "original_type"
31
32     ngsild_entity[safe_key] = {
33         "type": "Property",
34         "value": value
35     }
36
37     return ngsild_entity

```

3.3 Application Data Flow Pipeline

This section describes the high-level data processing and storage architecture, an overview of which is provided in Figure 3.3.

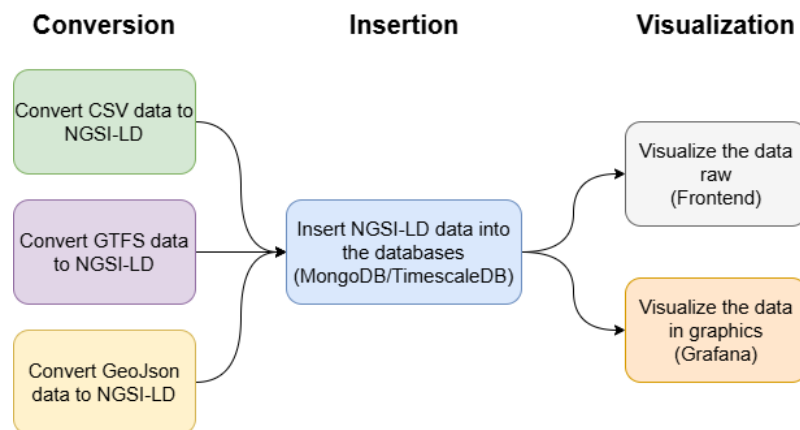


Figure 3.3: Data pipeline for the UrbanData Nexus.

For example, the standard operational workflow of the application begins with the ingestion of datasets, typically in **CSV**, **GTFS**, or **GeoJSON** formats, through their respective interface modules for conversion into the standardized format **NGSI-LD**. Subsequently, the formatted **NGSI-LD** data is ingested into a targeted database management system. Depending on the temporal requirements of the data, MongoDB is utilized to process real-time data streams, while TimescaleDB is used to store and analyze historical datasets. Finally, to facilitate data presentation, the system offers two distinct visualization modalities: the direct display of raw data within the native frontend, or advanced graphical representation facilitated by Grafana.

The application's data flow is structured into a sequential, multi-stage pipeline. The initial phase, Data Conversion, involves the ingestion and processing of heterogeneous data sources. As shown, various input formats, specifically **CSV**, **GTFS**, and **GeoJSON**, are converted into a

standardized **NGSI-LD** representation. This step ensures data harmony and semantic consistency across the system.

Following standardization, the Data Insertion phase persists the normalized **NGSI-LD** data within the target storage systems. For this architecture, the target includes MongoDB, a document-oriented database tasked with handling real-time data and the current operational state of the city's entities, alongside TimescaleDB, which is optimized for time-series data storage and historical analysis.

The third stage, Data Visualization, offers diverse mechanisms for accessing and consuming the processed information. Users can either access a raw JSON data feed via a custom frontend application, facilitating direct data inspection, or employ a comprehensive graphical visualization and analytical dashboard powered by Grafana.

Finally, to handle the immense volumes of data generated continuously by the smart city ecosystem and prevent long-term performance degradation, the architecture implements rigorous Data Lifecycle Management. Although recent and highly queried data remains immediately available in the primary databases to ensure rapid dashboard performance, historical data is systematically tied. TimescaleDB, for example, utilizes automated retention policies and native columnar compression to manage time-series growth. Older, infrequently accessed records are periodically offloaded to cost-effective cold storage, ensuring the core system remains highly performant without sacrificing the availability of historical data for auditing and long-term analytics.

3.4 Visualization & User Interface Layer

The presentation layer provisions the essential interfaces required for user interaction, system configuration, and longitudinal trend analysis. Serving as the primary user portal, a interactive web application developed utilizing the Streamlit framework facilitates active engagement with the system. This frontend orchestrates bidirectional integration with both the Orion-LD Context Broker, enabling real-time state observation and command execution, and the QuantumLeap and TimescaleDB pipeline for the extraction of historical analytics.

3.4.1 Log In & Register

The security architecture of the system relies on a centralized **IAM** framework to handle user authentication and authorization. To achieve this, Keycloak—a robust, open-source **IAM** solution—is employed to securely manage the user lifecycle and safeguard sensitive credentials. By decoupling the authentication processes from the core application logic, the architecture adheres to modern security paradigms, ensuring stringent protection against unauthorized access while facilitating secure, token-based session management.

The authentication workflow is designed such that unauthenticated users attempting to access protected resources are securely redirected to the Keycloak-hosted login portal. As depicted in Figure 3.4, this login interface provides a standardized mechanism for existing users to assert their identity. The authentication protocol requires the submission of valid credentials, specifically

a registered email address or unique username paired with a secure password. Upon successful cryptographic verification of these credentials, Keycloak issues an access token and securely routes the authenticated user back to the application with the appropriate operational privileges.

The image shows the Keycloak login interface for the 'FIWARE-REALM'. At the top, it says 'Sign in to your account'. Below this are two input fields: 'Username or email' and 'Password'. The 'Password' field has a toggle icon (an eye) to the right of it. Below the password field is a link that says 'Forgot Password?'. At the bottom of the form is a blue button labeled 'Sign In'. Below the button is a link that says 'New user? Register'.

Figure 3.4: Keycloak login interface.

For the onboarding of new users, individuals without pre-existing accounts must navigate to the dedicated registration interface, illustrated in Figure 3.5. This provisioning portal is engineered to capture the foundational demographic and credential data required to establish a new user profile within the Keycloak realm.

Prospective users are required to supply a unique username, a robust password that complies with the system's security policies, a valid email address, and their full name. Once this information is submitted, Keycloak securely processes the data, utilizing industry, standard hashing algorithms for password retention, and initializes the new account, thereby granting the user immediate and secure access to the platform's authenticated features.

3.4.2 Streamlit Codebase and Operational Flow

To operationalize this architecture, the Streamlit application relies on a secure, modular codebase designed to handle complex data processing tasks while enforcing strict authorization protocols. The operational logic is structured into three key functional components, each serving a distinct role in the data lifecycle.

As we can see in the figure 3.6, the left-hand sidebar manages the current user session, confirming successful authentication and displaying the active "Admin" role, which grants full access to the system's management tools. The primary navigation menu across the top illustrates the platform's modular architecture through a series of specialized tabs. These include, the "Send Data (Ingestion)" tab, which serves as the primary interface for sending data to the system's two different databases: Orion-LD and QuantumLeap, data transformation utilities "Convert CSV",

FIWARE-REALM

* Required fields

Register

Username *

Password *

Confirm password *

Email *

First name *

Last name *

[← Back to Login](#)

Register

Figure 3.5: Keycloak user registration interface.

"Convert GTFS", "Convert GeoJSON", system validation tools "IoT Agent Tester", analytical interfaces "Get Data (Visualization)", and schema management "Data Models".

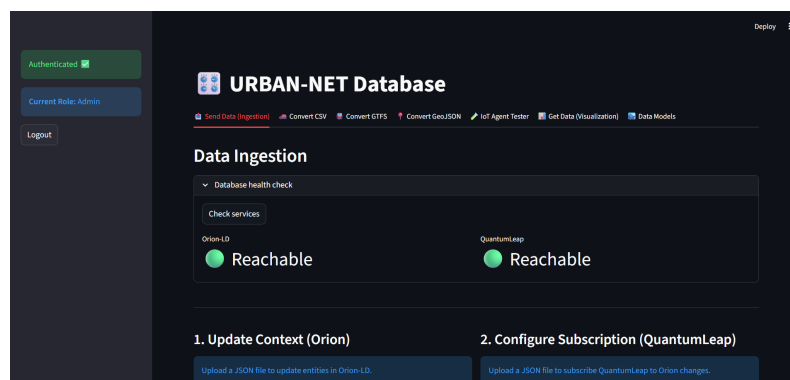


Figure 3.6: Interface layout of the UrbanData Nexus frontend.

In contrast to the comprehensive management dashboard, Figure 3.7 illustrates the interface tailored for a "Standard User". The platform implements **RBAC** to dynamically restrict or grant functionalities based on the authenticated profile.

While the Admin role possesses full privileges, including data ingestion, format conversion, and system health checks, the Standard User is limited strictly to read-only and analytical features. As shown in the streamlined top navigation menu, a Standard User only has access to the "Get Data (Visualization)" and "Data Models" tabs. This strict separation of privileges ensures database

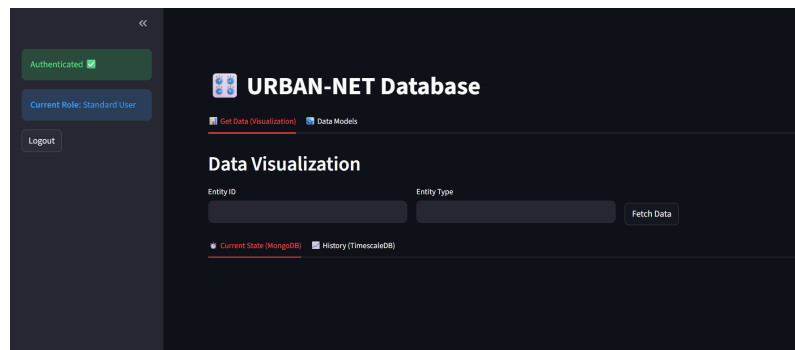


Figure 3.7: Interface layout of the UrbanData Nexus frontend for a Standard User.

integrity by preventing unauthorized data modifications, while still empowering regular users to query, explore, and visualize the available urban data.

3.4.2.1 Data Ingestion and System Validation

The administrative interface provides a dedicated "Data Ingestion" environment for direct interaction with the core system components, ensuring database stability and allowing manual updates to system brokers. As illustrated in Figure 3.8, the frontend application provides a specialized interface for data ingestion and system validation by facilitating several core functions.

First, it enables system health monitoring through an integrated health check utility that allows administrators to query and display the real-time status of various database services, ensuring the underlying infrastructure is operational before data is ingested.

Second, it streamlines broker management by allowing administrators to upload JSON payloads to directly update context entities within the Orion-LD broker, which makes immediate, manual state modifications possible.

Finally, the interface provides tools for subscription configuration, allowing users to configure subscriptions and trigger batch data notifications to QuantumLeap via JSON file uploads, effectively linking real-time entity changes to historical data storage.

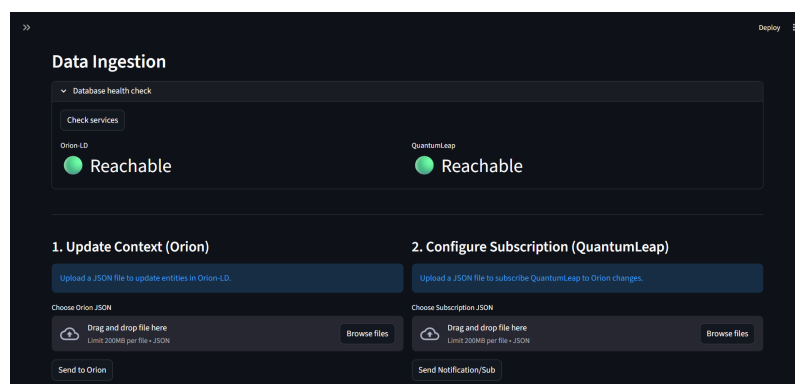


Figure 3.8: Graphical user interface of the data ingestion tab.

3.4.2.2 Format Conversion Engines

Facilitating data interoperability among diverse external sources necessitates robust, memory-efficient transformation mechanisms. The application leverages these engines to translate heterogeneous data into the compliant **NGSI-LD** schema through three principal conversion pipelines.

First, the Streaming **CSV** Converter is designed for large-scale datasets; this tool partitions data streams and archives outputs to prevent memory overflow during processing. Second, a Transit Data Parser serves as a dedicated pipeline for extracting and converting public transit data packaged in **GTFS** ZIP archives. Finally, the Geospatial Processor provides specialized utilities for mapping complex geographic structures provided in **GeoJSON** formats into the **NGSI-LD** schema.

Figure 3.9 presents the graphical user interface corresponding to the **CSV** to **NGSI-LD** conversion module. This component serves as an abstraction layer, translating raw tabular datasets into standardized, interoperable smart city data structures. Users utilize this interface to define an applicable Smart Data Model context, such as **TrafficFlowObserved**, and supply the corresponding source archive for processing.

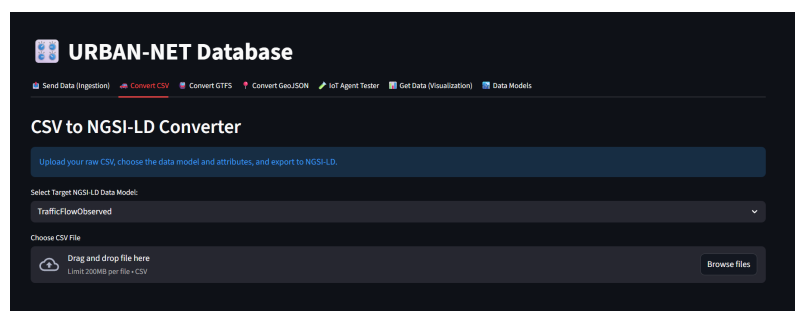


Figure 3.9: Graphical user interface of the CSV to NGSI-LD Converter.

Figure 3.10 illustrates the "Convert GTFS" module, a specialized utility for transforming complex transit datasets (uploaded as ZIP archives) into the standardized **NGSI-LD** format. Users select an appropriate Smart Data Model context, such as **UrbanMobility**, and upload their source archive.

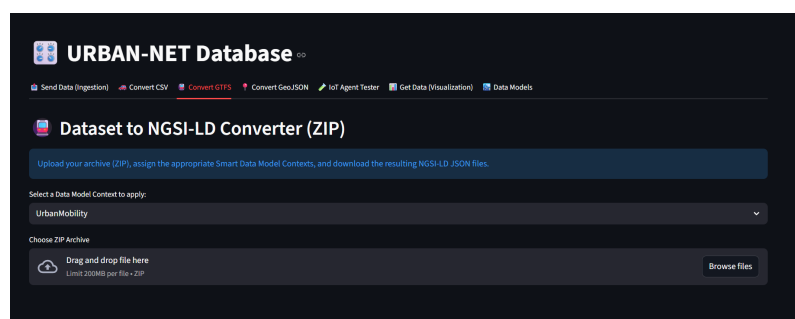


Figure 3.10: Graphical user interface of the Dataset to NGSI-LD Converter.

Figure 3.11 illustrates the "Convert GeoJSON" module, a specialized utility designed to transform geospatial data files into the standardized **NGSI-LD** format. Users select an appropriate **NGSI-LD** Entity Type, such as **OffStreetParking**, and upload their source **GeoJSON** file.

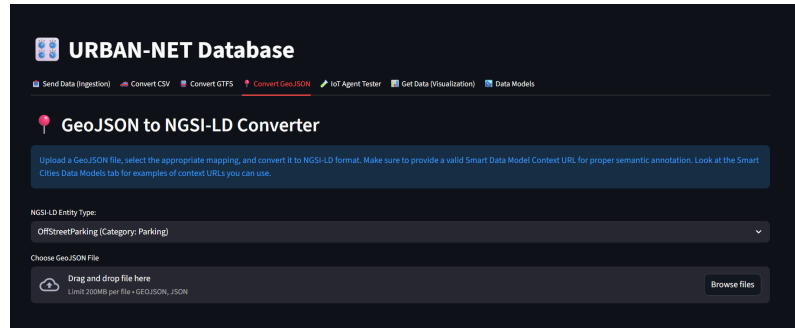


Figure 3.11: Graphical user interface of the GeoJSON to NGSI-LD Converter.

3.4.2.3 Data Retrieval and State Observation

Serving as the primary query interface, this module enables authorized users to fetch, analyze, and extract operational data efficiently.

Through targeted querying, users can retrieve specific datasets by filtering through target Entity IDs and Entity Types. Additionally, the application features dual-backend routing, which programmatically directs these requests based on temporal needs. Specifically, real-time data representing current operational states is retrieved directly from the Orion-LD and MongoDB backend, whereas longitudinal historical arrays and time-series data are extracted from the TimescaleDB repository.

Finally, the system handles data packaging by taking the retrieved information, aggregating the responses, and presenting them via the User Interface (**UI**) for immediate visual analysis or preparing them for direct user download.

Figure 3.12 presents the "Get Data (Visualization)" module, an interface designed for querying and inspecting specific entities within the database. Users can retrieve precise records by inputting a unique Entity ID. The system offers two distinct viewing modes: the "Current State," which fetches the latest entity context from MongoDB, and the "History" view, which retrieves temporal data from TimescaleDB. As depicted, a successful query displays the raw JSON payload of the requested entity directly within the browser, alongside options to copy the code or download it as a standalone file for external analysis.

3.4.2.4 Data Model Reference and Standardization

To facilitate system-wide interoperability and architectural coherence, the platform incorporates an embedded reference module for **NGSI-LD** compliant Smart Data Models. Functioning as an interactive documentation layer, it enables the *in situ* examination of representative JSON payloads across critical urban domains, namely transportation, urban mobility, and spatial planning.

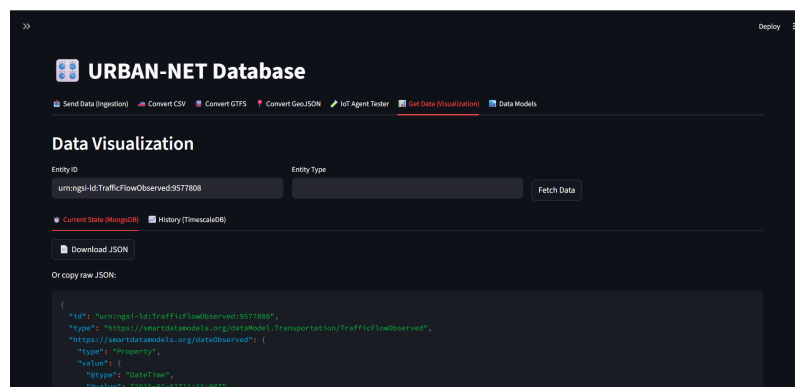


Figure 3.12: Graphical user interface of the Data Visualization module.

By coupling dynamic schema visualization with direct linkages to authoritative global repositories, the module allows stakeholders to verify structural dependencies and ensure strict adherence to standardized data protocols without disrupting the operational workflow.

Figure 3.13 displays the "Data Models" module within the frontend application. This section serves as a centralized reference guide for the harmonized Smart Data Models (NGSI-LD) utilized throughout the system to ensure data interoperability. Users can follow direct links to full external specifications or explore the structural requirements of specific standard domain models—such as Traffic Flow Observed, Public Transport Stop GTFS, and Off-Street Parking—directly through the expandable interface panels.

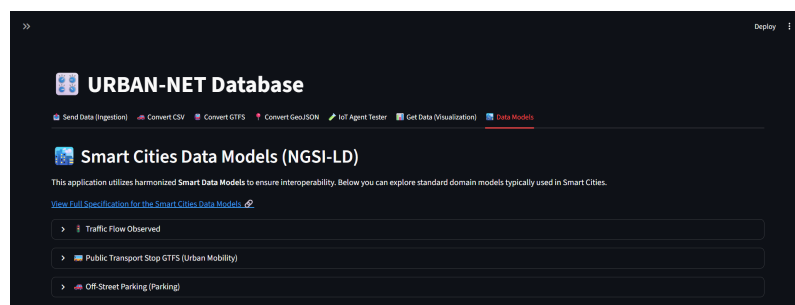


Figure 3.13: Graphical user interface of the Data Models module.

3.4.3 Advanced Dashboarding with Grafana

As a supplement to the primary Streamlit frontend, Grafana is implemented to provide comprehensive monitoring and sophisticated visual dashboards. It generates highly adaptable analytics for time-series data through a direct query link to TimescaleDB. This intentional architectural decoupling effectively offloads demanding visualization tasks, protecting the core Streamlit application from performance degradation caused by excess computational strain.

3.5 Security & Access Management

This section details the comprehensive security and access management framework implemented across the system architecture. To safeguard data integrity and prevent unauthorized access, the system employs a multi-layered, defense-in-depth strategy. The following subsections outline this approach, beginning with the foundational infrastructure, encompassing centralized identity management and strict network isolation, before detailing the practical enforcement of these security policies within the application's frontend interface.

3.5.1 Centralized Identity & Network Isolation

To enforce robust security protocols and strict access controls across the architecture, Keycloak is deployed as the centralized **IAM** provider. This component safeguards the broader ecosystem by orchestrating critical security lifecycle events, including rigorous user authentication, cryptographic token issuance, and systematic session revocation.

By acting as the definitive security gateway, Keycloak guarantees that interaction privileges, whether navigating the primary Streamlit interface, or executing programmatic requests against the underlying **APIs** are strictly restricted to properly authenticated and explicitly authorized personnel. This centralized approach to access management mitigates unauthorized intrusion while maintaining a unified, highly resilient security posture across all system endpoints and presentation interfaces.

Complementing this application-level security, the infrastructure minimizes its external attack surface through strict network isolation. By leveraging Docker's internal network, external port bindings for backend components, MongoDB, TimescaleDB, Orion, and QuantumLeap, have been entirely eliminated. These microservices now communicate securely via internal DNS, restricting public network exposure solely to essential ingress points: the Frontend, Grafana, and Keycloak.

3.5.2 Streamlit Access Control Implementation

The enforcement of Keycloak's security policies within the Streamlit frontend is handled through a sequence of automated OAuth 2.0 workflows and dynamic rendering logic.

First, secure configuration and initialization are achieved by dynamically loading critical Keycloak endpoint URLs and the Client ID from environment variables, ensuring that sensitive routing data is decoupled from the application's source code. Next, for authentication routing, the application maintains a session state to track active tokens; unauthenticated users are presented with an interactive gateway that halts application execution and redirects them to Keycloak's secure login portal.

Upon successful login, token decoding and **RBAC** enforcement occur as the application decodes the resulting **JWT** to extract user roles. The authorization logic checks both *realm_access* and *resource_access* payloads to verify if the user possesses the admin role.

Following this, dynamic **UI** provisioning employs conditional rendering based on the extracted **RBAC** profile to expose functionality. Specifically, administrative personnel are granted access to write-heavy and administrative tabs (e.g., Data Ingestion, **IoT** Testing, File Conversions), whereas standard users are restricted entirely to read-only tabs (Data Visualization and Data Models).

Finally, session lifecycle management is handled by a dedicated logout protocol that clears the local Streamlit session state and triggers Keycloak's native browser logout URL, effectively neutralizing the token and terminating the session across the architecture.

3.6 IoT Ingestion & Device Connectivity Layer

This section explicates the foundational layer responsible for the acquisition of raw telemetry from distributed edge devices and its systematic integration into the core operational platform.

The ensuing discussion is structured around two principal domains: first, an analysis of the core architecture and protocol translation mechanisms utilized to normalize unstructured data into the standardized **NGSI-LD** specification; and second, the presentation of an interactive validation pipeline implemented via the Streamlit framework.

This pipeline facilitates real-time system monitoring, streamlined device provisioning, and rigorous end-to-end validation of the data ingestion workflows.

3.6.1 Core Architecture and Protocol Translation

This foundational layer is tasked with the continuous acquisition of raw telemetry from edge devices and its subsequent normalization into a standardized architectural format. As illustrated in the expanded architectural overview in Figure 3.14, this ingestion pipeline, specifically highlighted within the red bounding box, is driven by the continuous interactions between edge infrastructure, a central **MQTT** broker, and the FIWARE **IoT** Agents.

Within this subsystem, the Mosquitto broker serves as the initial data ingress point, enabling simulated **IoT** agents to publish telemetry via **MQTT** or WebSockets. FIWARE **IoT** Agents then retrieve this unstructured data directly from the broker using a Publish/Subscribe pattern.

It actively subscribes to the **MQTT** broker to intercept the incoming raw JSON payloads, systematically parsing and transforming this data to comply with the standardized **NGSI-LD** specification. Upon successful syntactic and semantic normalization, the **IoT** Agents efficiently routes this structured contextual data downstream to the Context Broker, thereby ensuring seamless integration between the physical edge infrastructure and the core operational platform.

To operationalize and manage this connectivity layer, a programmatic Python administration module is utilized. This module interfaces directly with the **IoT** Agent's **REST APIs** to automate the provisioning of service groups and individual edge devices, explicitly defining the attribute mappings required for **NGSI-LD** translation. Additionally, it leverages the `paho-mqtt` library to simulate edge sensor behavior by publishing telemetry payloads to device-specific **MQTT** topics.

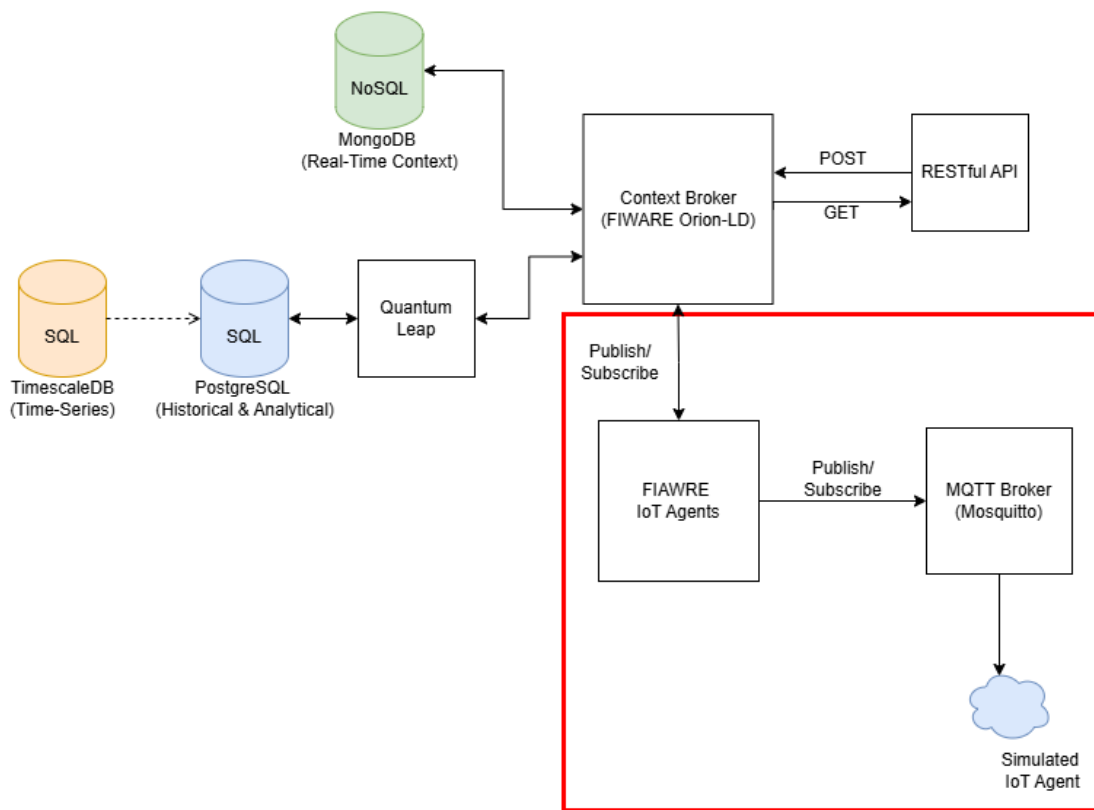


Figure 3.14: IoT Ingestion and Connectivity Architecture.

Finally, by integrating direct querying capabilities against the Orion Context Broker, the module enables end-to-end validation of the ingestion pipeline, ensuring that raw **MQTT** transmissions are successfully captured, translated, and queried as standardized context entities.

3.6.2 Interactive Pipeline Validation via Streamlit

To facilitate user interaction and provide immediate visual validation of the ingestion layer, a graphical dashboard was developed using the Streamlit framework. This interface abstracts the underlying Python administration module, guiding the user through a step-by-step end-to-end test of the data pipeline.

Before initiating any data transactions, the dashboard performs an automated system health check (see Figure 3.15). This ensures that both the FIWARE **IoT** Agent and the Orion-LD Context Broker are active and reachable, verifying the operational readiness of the foundational infrastructure.

The first active step in the validation sequence, shown in Figure 3.16 (first panel), involves the provisioning of a service group. This configuration defines the baseline parameters for a fleet of edge devices, establishing the shared **API** key, the resource endpoint, and the target Context Broker routing URL necessary for downstream communication.

Following the setup of the service, the interface manages the provisioning of individual devices (Figure 3.16, second panel). This critical step maps the physical device ID to its standardized

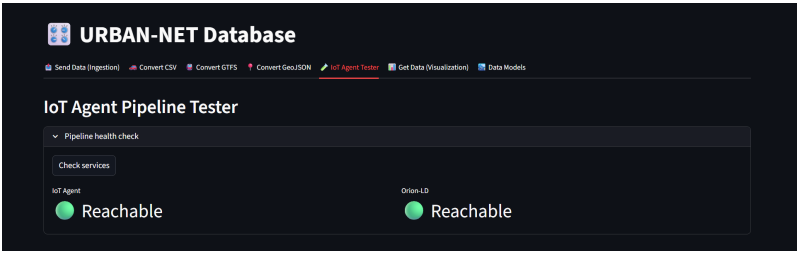


Figure 3.15: Dashboard Interface for the Pipeline Health Check.

NGSI-LD entity **URN** and establishes the attribute translation rules, ensuring that raw incoming keys (such as "t") are correctly parsed as specific context properties (such as "temperature").

To simulate real-world edge device behavior, the dashboard includes an **MQTT** publication interface, illustrated in Figure 3.16 (third panel). Users can inject a raw, unformatted JSON payload directly into the Mosquitto broker via the device-specific topic, triggering the **IoT Agent**'s interception and translation mechanisms.

The final validation step (Figure 3.16, fourth panel) queries the Orion-LD Context Broker directly using the mapped entity ID. The successful retrieval of a fully formatted **NGSI-LD** JSON document—reflecting the previously injected payload—confirms that the pipeline successfully captured, translated, and stored the raw telemetry as standardized contextual data.

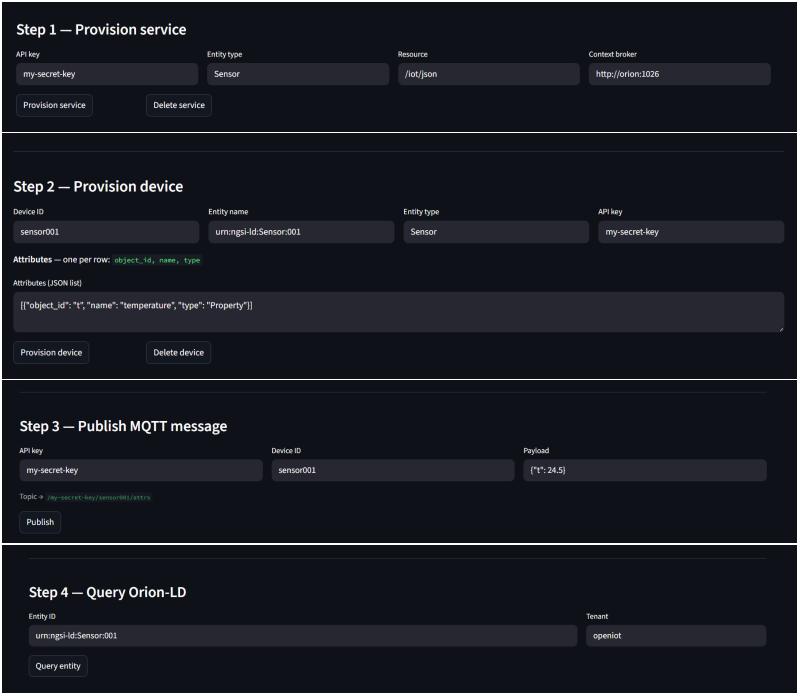


Figure 3.16: Operational workflow of the IoT Tester interface

3.7 Summary

This chapter presented the comprehensive architectural framework and technical execution of the UrbanData Nexus. By implementing a bifurcated data management strategy, the platform effectively isolates real-time operations from historical analytics. The Orion-LD Context Broker and MongoDB handle the low-latency requirements of live state observation, while QuantumLeap and TimescaleDB manage the computational load of longitudinal data analysis.

To guarantee system-wide interoperability, the architecture features robust transformation pipelines that standardize diverse heterogeneous data formats, specifically **CSV**, **GTFS**, and **GeoJSON**, into the **NGSI-LD** schema. This commitment to standardization extends to the **IoT** connectivity layer, which intercepts and translates raw **MQTT** edge telemetry into structured, queryable context entities.

User interaction and system management are facilitated by a modular presentation layer built on Streamlit and Grafana. This layer equips administrators and users with intuitive tools for data ingestion, end-to-end pipeline validation, and advanced visual analytics. Safeguarding this entire ecosystem is Keycloak, which enforces strict, centralized identity and access management.

Ultimately, the synthesis of these integrated components results in a scalable, secure, and highly performant data infrastructure designed explicitly to solve the complex data-routing challenges of modern smart city administration.

Chapter 4

Validation

This chapter presents a rigorous empirical assessment of the UrbanData Nexus platform, formulated to evaluate both its computational robustness and end-user efficacy. To ensure a comprehensive validation of the proposed framework, the methodological approach is bifurcated into two primary analytical domains: System Validation (Section 4.1) and User Experience (Section 4.2).

The first section empirically tests the end-to-end technical capabilities of the system, specifically scrutinizing its ability to seamlessly ingest, standardize, and visualize heterogeneous data streams sourced from the URBAN-NET project, which encompass high-volume **CSV** traffic logs, **GTFS** transit schedules, **GeoJSON** spatial structures, and simulated real-time **IoT** telemetry.

The second section shifts the focus to human-in-the-loop operational dynamics, analyzing user interaction metrics, task execution efficiency, and overall platform usability through a structured testing protocol.

Together, these complementary evaluations contextualize the empirical findings to conclusively address the core research questions and substantiate the platform’s viability as a modern urban data management solution.

4.1 System Validation

This section presents a comprehensive validation of the proposed system architecture, evaluating its end-to-end capability to ingest, transform, store, and visualize heterogeneous urban data streams. This validation process was conducted utilizing real-world datasets and operational parameters provided by the URBAN-NET project. To verify the robustness and interoperability of the data pipeline, the validation procedures systematically assess multiple structured data formats derived from the URBAN-NET initiative alongside the platform’s operational modules.

The testing protocol traces the data lifecycle, from initial upload and standardization into the **NGSI-LD** format using Smart Data Models, through to historical storage in TimescaleDB, and ultimately to its macroscopic visualization on interactive dashboards. Specifically, this evaluation is divided into four key operational areas: the processing of high-volume **CSV** traffic flows, the

spatial mapping of **GTFS** schedules, the structural parsing of **GeoJSON** parking facility datasets, and the real-time telemetry simulated by the custom **IoT** Tester module.

4.1.1 CSV data validation

The validation phase of the traffic data pipeline begins with an initial data transformation step. Specifically, raw **CSV** data are uploaded and restructured in the **NGSI-LD** format, where the required columns are assigned to the `TrafficFlowObserved` data model (see Figure 4.1). Performance evaluations demonstrated that transforming a dataset of approximately 160 MB was completed in 174.80 seconds, ultimately dividing the processed data into 27 separate files.

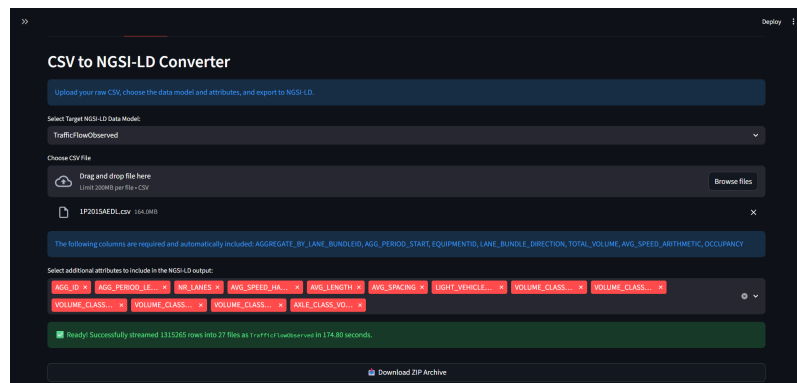


Figure 4.1: Web interface for converting raw CSV traffic data into the NGSI-LD.

Following the completion of the conversion process, the generated JSON files advance to the data ingestion phase. As illustrated in Figure 4.2, this stage involves configuring subscriptions to route the batched data to QuantumLeap for historical storage. Performance measurements indicate that transmitting an approximately 80 MB file required 119.58 seconds, successfully processing 50,000 entities distributed across 100 discrete batches.

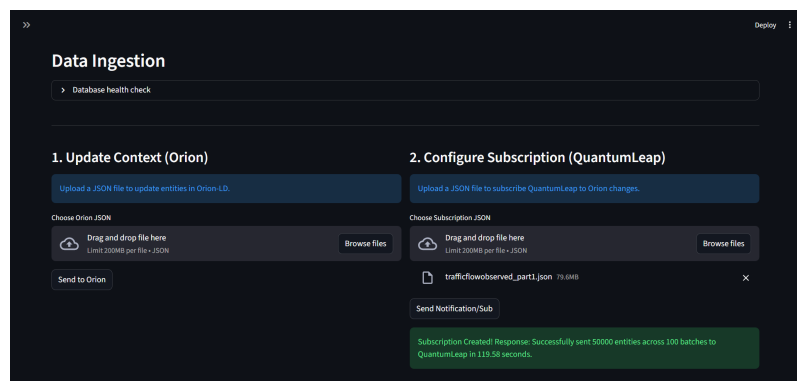


Figure 4.2: Data ingestion dashboard for the converted data.

To verify that the data were processed and stored correctly, the system allows for direct data retrieval. As depicted in Figure 4.3, specific entities can be queried by their ID to inspect the raw historical JSON data stored within TimescaleDB.

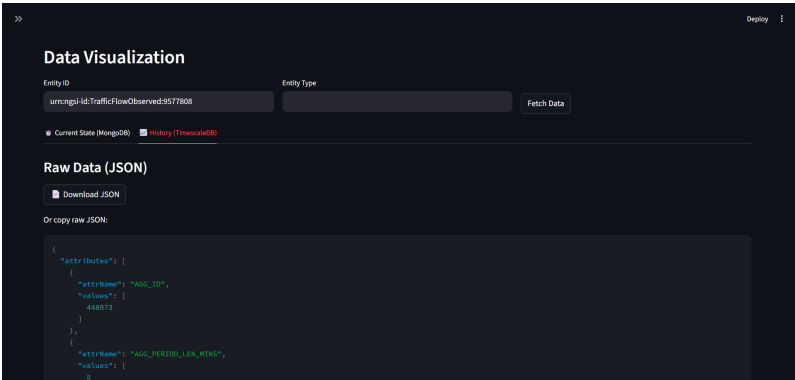


Figure 4.3: Interface for fetching and validating raw historical JSON data from TimescaleDB.

Ultimately, the validated dataset is integrated into monitoring dashboards to facilitate comprehensive analytical review. Figure 4.4 presents a robust time-series visualization detailing the fluctuations in average vehicular speed, spatial occupancy, and traffic intensity across consecutive days. This representation serves to elucidate peak traffic events and affirm the underlying consistency of the data.

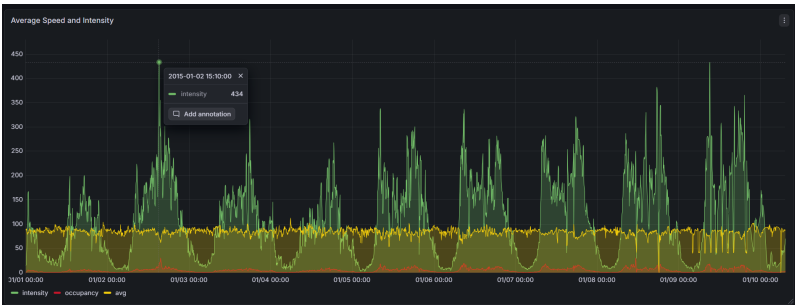


Figure 4.4: Time-series graph displaying traffic intensity, occupancy, and average speed over a multi-day period.

Listing 4.1: SQL query utilized for the daily aggregation of temporal traffic metrics

```
1 SELECT
2     AVG(intensity) as intensity,
3     AVG(occupancy) as occupancy,
4     AVG(averageVehicleSpeed),
5     dateobserved
6 FROM ettrafficflowobserved
7 GROUP BY dateobserved;
```

To generate these macro-level insights, the raw observational data must be systematically aggregated. Listing 4.1 details the SQL query utilized to extract the daily mean values for traffic intensity, occupancy, and average vehicle speed from the `ettrafficflowobserved` database

table. By grouping the high-frequency temporal data by the `dateobserved` parameter, the query effectively normalizes the dataset, rendering it suitable for longitudinal trend analysis.

Complementing this granular longitudinal analysis, Figure 4.5 delineates the aggregated high-level statistics alongside sparkline trend indicators for the corresponding metrics. This macroscopic perspective empirically substantiates the successful end-to-end ingestion, processing, and visualization of the traffic flow dataset.

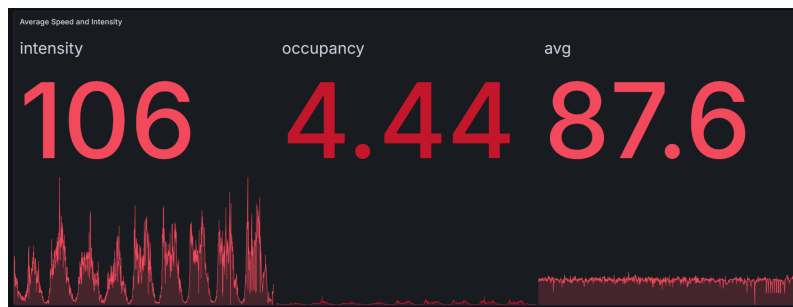


Figure 4.5: Dashboard panel showing aggregated statistics and sparklines for key traffic metrics.

4.1.2 GTFS data validation

The validation process for **GTFS** data ensures that raw transit schedules and geographic information are accurately transformed, stored, and visualized within the system. First, the raw **GTFS** dataset (typically provided as a ZIP archive) is uploaded and converted into standard **NGSI-LD** entities. During this step, the appropriate Smart Data Model Context, such as *UrbanMobility*, is applied to standardize the data structure, as shown in Figure 4.6.

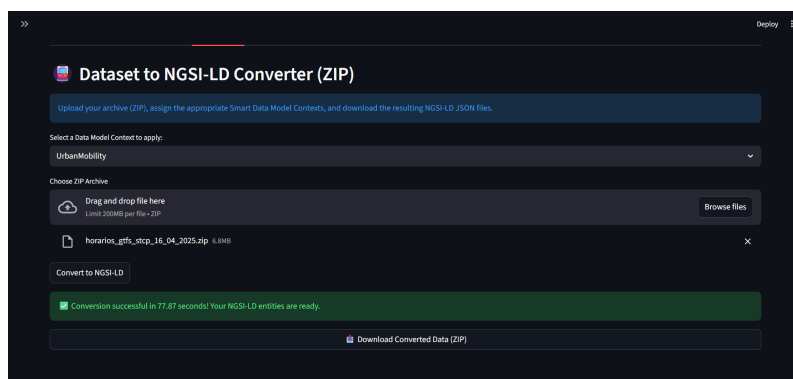


Figure 4.6: Dataset to NGSI-LD Converter interface showing the successful conversion of a GTFS ZIP archive using the *UrbanMobility* context.

Following a successful conversion, the resulting JSON files are ingested into the platform. Subscriptions are configured in components like *QuantumLeap* to track changes and store historical data. Figure 4.7 demonstrates the successful creation of a subscription, confirming that the converted entities are correctly registered and routed.

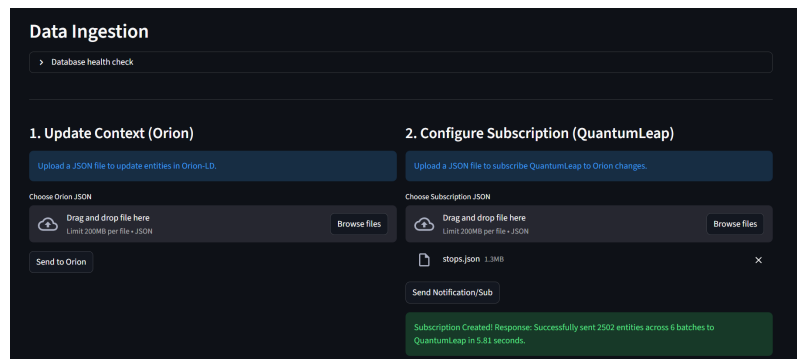


Figure 4.7: Data Ingestion dashboard displaying the successful configuration of a QuantumLeap subscription for the converted GTFS stops.

To verify the integrity of individual entities, the Data Visualization dashboard is used to inspect the raw **NGSI-LD** JSON payloads. This step allows developers to validate that specific attributes, such as geographical coordinates (longitude and latitude), are correctly formatted as **GeoJSON** points (Figure 4.8).

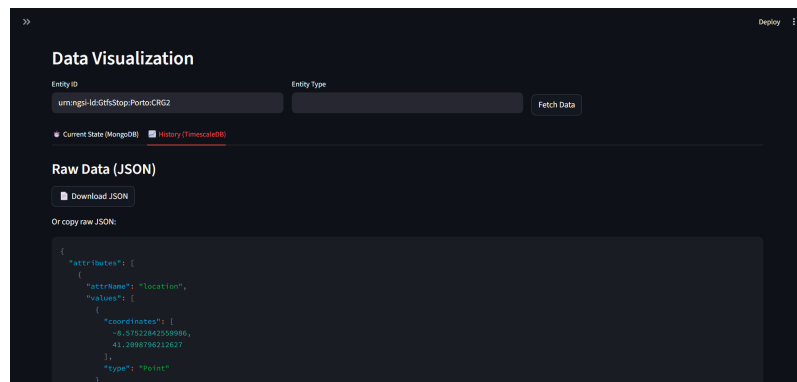


Figure 4.8: Data Visualization interface revealing the raw NGSI-LD JSON payload for a specific GTFS stop, including its point coordinates.

Listing 4.2: SQL query used to extract the designation, longitude, and latitude of GTFS stops for map visualization.

```

1 SELECT
2     name AS designacao,
3     ST_X(location) AS longitude,
4     ST_Y(location) AS latitude
5 FROM etgtfsstop;

```

Finally, a macroscopic spatial validation is performed by rendering the coordinates on a geographical map. To achieve this, the required spatial attributes, specifically the stop designations and their coordinates, are retrieved from the database using the query presented in Listing 4.2. As

illustrated in Figure 4.9, the mapped bus stops accurately reflect the transit network across the city of Porto, confirming that the entire pipeline has processed the spatial data accurately.

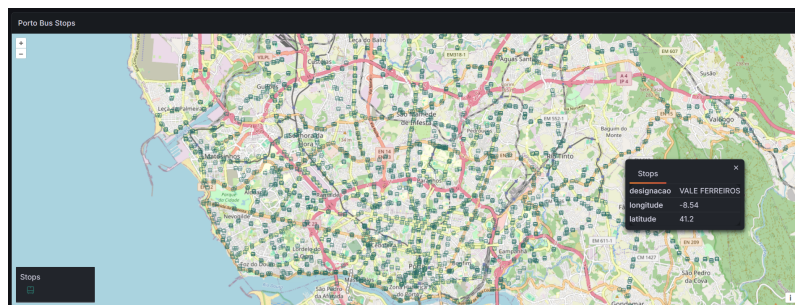


Figure 4.9: Geographical visualization of the ingested GTFS bus stops accurately mapped across the city of Porto.

4.1.3 GeoJson data validation

The initial phase of the procedure involves transforming the dataset into the **NGSI-LD** format. As illustrated in Figure 4.10, the source **GeoJSON** file, which encapsulates the raw parking data, is ingested by the conversion utility. During this process, the data attributes are systematically mapped to the `OffStreetParking` entity type, guaranteeing that the resulting dataset strictly adheres to the semantic specifications mandated by the Smart Data Models framework. Furthermore, the computational overhead of this operation is negligible, with the conversion executing almost instantaneously (yielding a recorded processing time of 0.00 seconds).

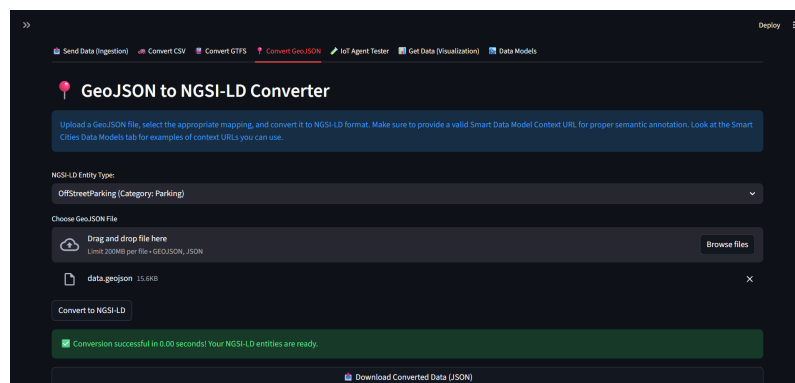


Figure 4.10: Interface for converting GeoJSON data into NGSI-LD format.

Subsequent to the data conversion phase, the resultant **NGSI-LD** file is ingested into the platform's architectural framework. As depicted in Figure 4.11, the data ingestion interface facilitates the upload of the transformed JSON document to initialize a data subscription mechanism via the QuantumLeap service. Furthermore, the system validates the successful transmission and processing of the spatial parking entities, completing the integration process with a recorded execution latency of 0.23 seconds.

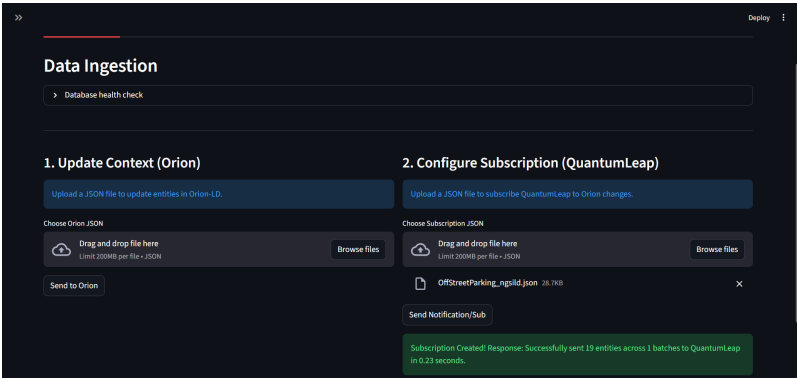


Figure 4.11: Data ingestion dashboard showing the successful creation of a subscription in QuantumLeap.

To substantiate the structural and semantic integrity of the ingested dataset, the platform’s Data Visualization interface is employed. Figure 4.12 illustrates the retrieval of the current state of a discrete entity from the underlying database. Examination of the raw JSON output verifies that intricate attribute hierarchies, including concession specifications and administrative designations, have been accurately parsed and persistently stored within the system.

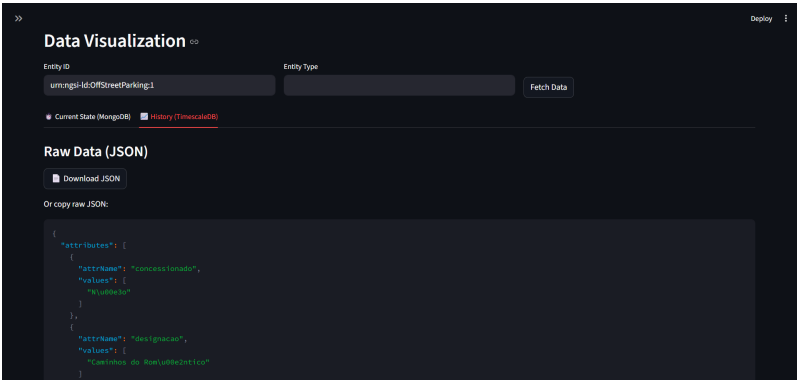


Figure 4.12: Data visualization interface displaying the raw NGS-LD JSON data for a specific parking entity.

Listing 4.3: SQL query extracting the geospatial coordinates and designations of off-street parking facilities.

```
1 SELECT
2   designacao AS Name,
3   ST_X(location) AS longitude,
4   ST_Y(location) AS latitude
5 FROM etoffstreetparking;
```

Subsequently, the validated spatial and statistical datasets are integrated into analytical dashboards. To facilitate this visualization, the underlying spatial data must be systematically queried.

Listing 4.3 details the **SQL** statement utilized to extract the nomenclature and the precise geospatial coordinates (longitude and latitude) of each facility. The results of this extraction are rendered in Figure 4.13, which presents a cartographic projection detailing the geospatial distribution of the off-street parking infrastructure across the municipality of Porto.

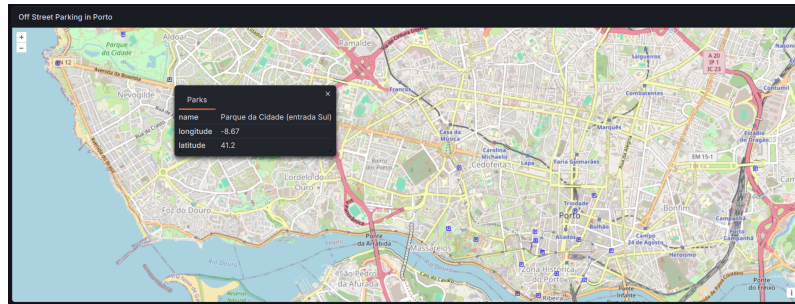


Figure 4.13: Geospatial distribution of off-street parking facilities within the municipality of Porto.

Complementing this geospatial analysis, a quantitative statistical summary is required to fully assess the infrastructure’s capacity. Listing 4.4 outlines the **SQL** aggregation query executed to compute the cumulative capacity of various parking typologies. This query uses **NGSI-LD** data to count spaces for light and heavy vehicles, Electric Vehicles (**EV**) charging, reduced mobility, motorcycles, and bikes. The output of this aggregation is visually delineated in Figure 4.14, which illustrates the proportional distribution of these distinct parking typologies across all monitored municipal facilities.

Listing 4.4: **SQL** query computing the aggregated capacity of distinct parking typologies across monitored facilities.

```

1 SELECT
2   SUM(n_lugares_ligeiros) AS n_lugares_ligeiros,
3   SUM(n_lugares_mobilidade_reduzida) AS n_lugares_mobilidade_reduzida,
4   SUM(n_lugares_motociclo_ciclomotores) AS n_lugares_motociclo_ciclomotores,
5   SUM(n_lugares_velocipedes) AS n_lugares_velocipedes,
6   SUM(n_lugares_pesados) AS n_lugares_pesados,
7   SUM(n_lugares_veiculos_eletricos) AS n_lugares_veiculos_eletricos
8 FROM etoffstreetparking;
```

4.1.4 Validation of the IoT Tester Module

This subsection details the systematic procedure employed to validate the functionality and accuracy of the **IoT** Tester within the application environment. To ensure robust performance and reliable data processing, the validation workflow is executed across four sequential stages.

This step represents the creation of the service. Once created, the validation process begins with the initialization of the testing environment. As depicted in Figure 4.15, the application

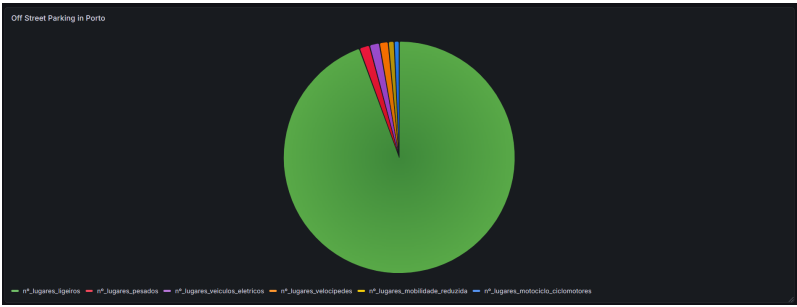


Figure 4.14: Proportional distribution of cumulative parking capacity by vehicle typology across monitored facilities in Porto.

first establishes a connection with the target **IoT** device, verifying that the basic communication channels are active and ready for testing.

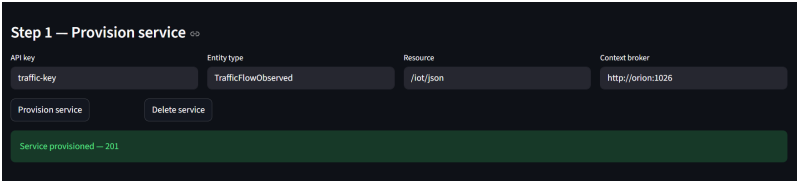


Figure 4.15: Step 1: Successful connection establishment with the target IoT device.

To complete the creation of the device following a successful connection, the next phase involves defining the specific test parameters. Figure 4.16 illustrates the configuration interface, where the user can select appropriate communication protocols, define payload specifications, and set the testing intervals required to simulate real-world conditions.

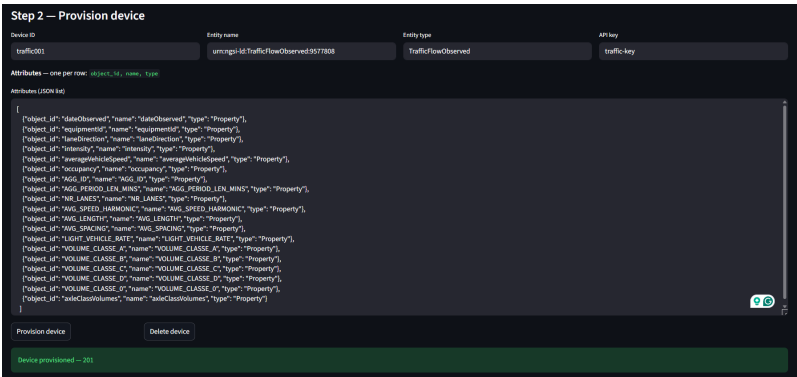


Figure 4.16: Step 2: Configuration of the validation parameters.

Once the parameters are securely set, the active testing phase is initiated by publishing the data via an **MQTT** message. As shown in Figure 4.17, the application monitors this transmission in real-time. This step is crucial for observing the live telemetry data being published and verifying that the system logs accurately capture the outbound messaging traffic. An example of the **MQTT** telemetry payload published during this step is detailed in Listing 4.5.

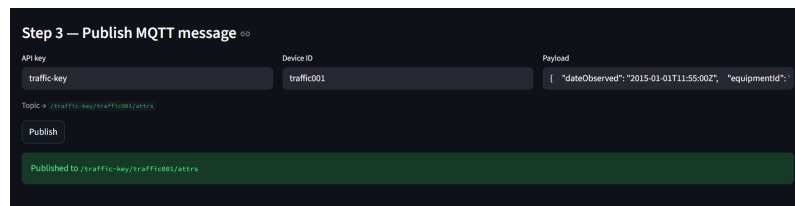


Figure 4.17: Step 3: Active execution of the test suite, demonstrating real-time data telemetry and system log monitoring.

Listing 4.5: Example JSON payload representing telemetry data exchanged during the active testing phase.

```

1 {
2   "dateObserved": "2015-01-01T11:55:00Z",
3   "equipmentId": "121729",
4   "laneDirection": "D",
5   "intensity": 48,
6   "averageVehicleSpeed": 97.77,
7   "occupancy": 1.54,
8   "AGG_ID": 448873,
9   "AGG_PERIOD_LEN_MINS": 5,
10  "NR_LANES": 3,
11  "AVG_SPEED_HARMONIC": 94.9,
12  "AVG_LENGTH": 443.75,
13  "AVG_SPACING": 283.4,
14  "LIGHT_VEHICLE_RATE": 100,
15  "VOLUME_CLASSE_A": 0,
16  "VOLUME_CLASSE_B": 48,
17  "VOLUME_CLASSE_C": 0,
18  "VOLUME_CLASSE_D": 0,
19  "VOLUME_CLASSE_0": 0,
20  "axleClassVolumes": { "E2": 48 }
21 }
```

Finally, upon completion of the test cycle, the application compiles the gathered data to evaluate the system's overall performance. Figure 4.18 presents the final validation results dashboard, which highlights key metrics such as response latency, success rates, and any recorded errors, thus confirming the operational readiness of the **IoT** tester.

4.2 Use Case

This section presents a comprehensive use case designed to validate the practical application and end-to-end functionality of the platform. Moving beyond isolated component testing, the evaluation examines the system's efficacy through the lens of a simulated, real-world user journey.

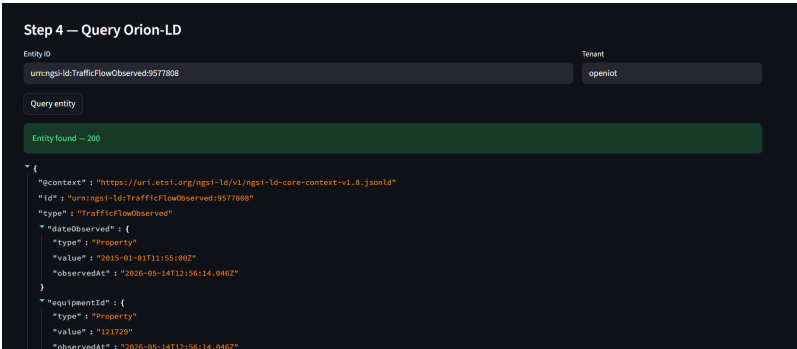


Figure 4.18: Step 4: Comprehensive dashboard displaying the final validation results.

The following subsections detail the experimental methodology, outline the specific operational workflows—spanning from secure authentication to complex urban data ingestion and visualization—and provide a thorough quantitative and qualitative analysis of the platform’s overarching usability, operational efficiency, and user experience.

4.2.1 Scenario Setup and Objectives

To demonstrate the platform’s practical application and validate its end-to-end functionality, a comprehensive use case scenario was designed. Rather than conducting isolated integration tests, this phase evaluates the system through the lens of a real-world user journey.

The objective is to track a user’s complete workflow: beginning with secure system access via Keycloak, progressing through the ingestion and conversion of urban data files (CSV and GTFS formats) using the Orion-LD and Quantum Leap components, and concluding with the successful visualization and analysis of that data on the Streamlit interface.

Before starting the scenario, the foundational infrastructure, specifically the Orion Context Broker and TimescaleDB, was deployed to simulate a fully operational environment. To accurately mirror a typical user’s access flow, a standardized user persona was established within Keycloak, providing the participants with a secure set of credentials (testuser / 1234).

4.2.1.1 User Journey and Evaluation Protocol

To systematically guide participants through this use case and capture their experiential feedback, they were provided with a structured scenario instrument titled the "URBAN-NET Database Validation Test," administered via Google Forms, the full form is in the Annex A.1. Designed as a 15-minute guided exercise, this protocol strictly anonymized all user interaction metrics. The instrument served a dual purpose: it acted as an instructional walkthrough for the user journey, and as a comprehensive data collection tool.

Structurally, the scenario was divided into three distinct procedural phases, allowing for targeted evaluation of both the platform’s technical reliability and the overall user experience at each step of the pipeline.

4.2.1.2 Phase I: Pre-Test Profiling

Prior to system interaction, participants provided baseline data to facilitate the categorization of their technological proficiency. This preliminary section gathered fundamental demographic variables, including age cohort and gender, alongside their primary academic discipline or professional domain (e.g., Management, Information Technologies, Engineering).

Furthermore, the survey established a baseline assessment of participants' self-reported comfort levels and self-efficacy regarding the operation of web applications and the execution of complex computational tasks.

4.2.1.3 Phase II: Guided Test Cases

To ensure uniform testing conditions and mitigate network latency, participants accessed the application interface exclusively via a localized host environment. Following a preliminary briefing on the navigation of the system, the users sequentially executed five key operational scenarios. This progressive workflow, which goes from authentication to data visualization, was designed to simulate a comprehensive real-world data processing pipeline.

Test Case 1 assessed the system's authentication mechanisms via Keycloak, necessitating that users authenticate utilizing pre-configured test credentials.

Test Case 2 evaluated data parsing and real-time ingestion capabilities utilizing Orion-LD. Within this phase, participants uploaded a standardized **CSV** dataset, *tests_csv.csv*; executed its structural transformation into the `TrafficFlowObserved` schema; retrieved the synthesized payload; and initiated its injection into the real-time database.

Test Case 3 examined **GTFS** format conversion and historical data integration employing QuantumLeap. Participants processed a centralized urban mobility dataset, *tests_gtfs.zip*; validated the extraction of specific structural entities—namely, `stop_times`, `stops`, and `trips`, and subsequently transmitted these discrete entities to the historical database.

Test Case 4 focused on real-time data retrieval and visualization, wherein participants queried the Context Broker for an active entity utilizing a specified Uniform Resource Name identifier, *urn:ngsi-ld:TrafficFlowObserved:9577808*.

Finally, Test Case 5 investigated historical data visualization capabilities. Participants were prompted to query time-series data via QuantumLeap for the `GtfsStop` entity type to facilitate the analysis of historical tabular records and underlying temporal trends.

Following the execution of each experimental scenario, task completion times were recorded for all participants. Subsequently, task load and perceived difficulty were empirically quantified utilizing a 7-point Likert scale. Notably, the reliance on manual chronometry introduces a minor systemic variance, specifically, human-operated stopwatches yield a documented reaction-time error margin of 0.061 seconds [46].

4.2.1.4 Phase III: System Usability and Qualitative Feedback

Subsequent to the practical execution of the data pipeline, the assessment instrument measured the platform's overarching usability through the standardized System Usability Scale (SUS). The participants evaluated a series of statements regarding systemic complexity, the anticipated requisite for technical support, and architectural consistency.

The protocol concluded with the elicitation of qualitative feedback, where users were prompted to synthesize their overall user case into a single statement, identify the primary operational pain point assuming continuous daily usage, and propose one actionable interface optimization.

4.2.2 Results

This section presents the findings from the usability evaluation of the UrbanData Nexus hybrid database application. The primary objective of this assessment was to evaluate the user case and system efficiency regarding the mechanisms for converting CSV and GTFS files, alongside their injection and querying capabilities within real-time and historical databases.

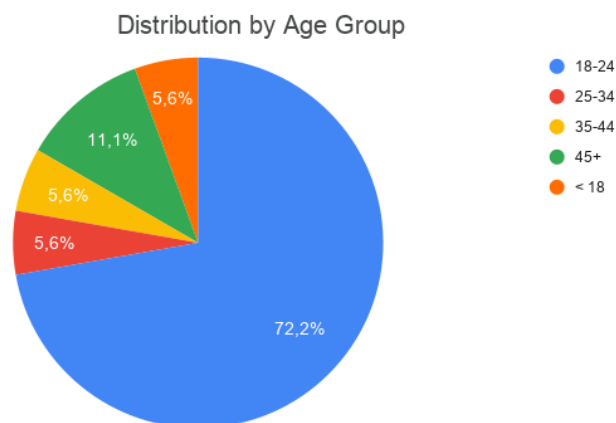


Figure 4.19: Distribution of participants by age group.

The participant sample consisted predominantly of young adults, aged between 18 and 24 years (see Figure 4.19), with a strong background in Information Technologies as either students or active professionals. A preliminary demographic and behavioral analysis revealed a highly technologically proficient cohort.

Participants reported an exceptional level of comfort with web applications in their daily lives, consistently rating their familiarity between 6 and 7, on a 7-point scale, and indicated frequent engagement with advanced computational tasks beyond standard web browsing.

The quantitative analysis of the usability test focused on two primary metrics: the average execution time per task, reported in minutes, seconds, and milliseconds according to the mm:ss.sss format (see Figure 4.21), and the perceived difficulty, rated on a 7-point Likert scale where higher values indicate greater ease of completion (see Figure 4.20). The empirical data demonstrated a

strong inverse correlation between execution time and perceived ease of use. Basic operational tasks yielded highly efficient outcomes.

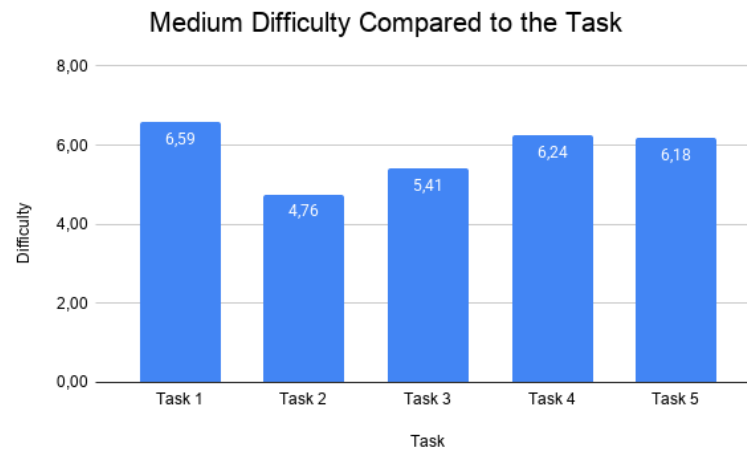


Figure 4.20: Average perceived difficulty per task.

Throughout the session, the mean final time for all users was 06:59.267. However, this metric decreased significantly to 05:39.847 for participants whose professional role was within Information Technologies or Engineering.

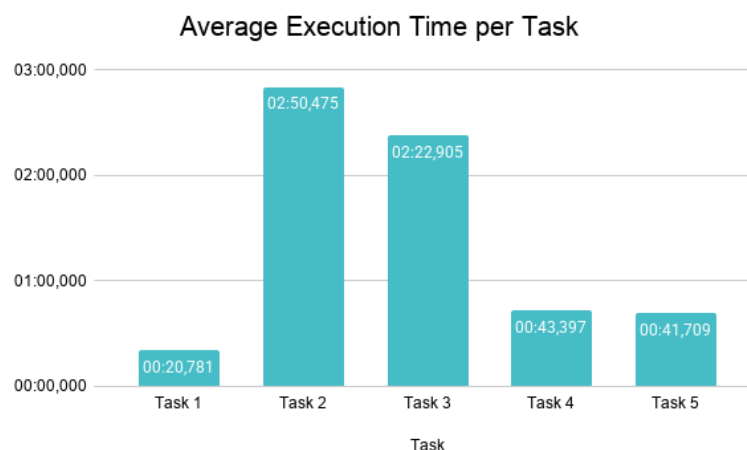


Figure 4.21: Average execution time required by participants to complete each task.

Task 1 was the most rapidly completed, requiring an average of merely 21 seconds, and was correspondingly rated as the easiest, with a mean score of 6.59. Similarly, Task 4 and Task 5 were executed swiftly, taking approximately 43 and 42 seconds respectively, and received high ease-of-use ratings of 6.24 and 6.18. Conversely, Tasks 2 and 3 introduced a noticeable increase in operational complexity.

Task 2 proved to be the most demanding, requiring an average execution time of 2 minutes and 50 seconds and yielding the lowest ease-of-use score at 4.76. Task 3 followed a similar trend,

demanding 2 minutes and 23 seconds for completion with an ease-of-use rating of 5.41.

These findings suggest that while the foundational features of the platform are highly intuitive, specific procedures within Tasks 2 and 3 impose a steeper learning curve or require a higher cognitive load from the user.

Complementing the quantitative metrics, qualitative feedback and the SUS results provided valuable insights into the user interface and overall experience (see Figure 4.22). The SUS questionnaire items were evaluated on a standard 5-point Likert scale, where a score of 1 represents "Strongly Disagree" and 5 signifies "Strongly Agree". Participant responses largely corroborated the high usability scores observed in the simpler tasks.

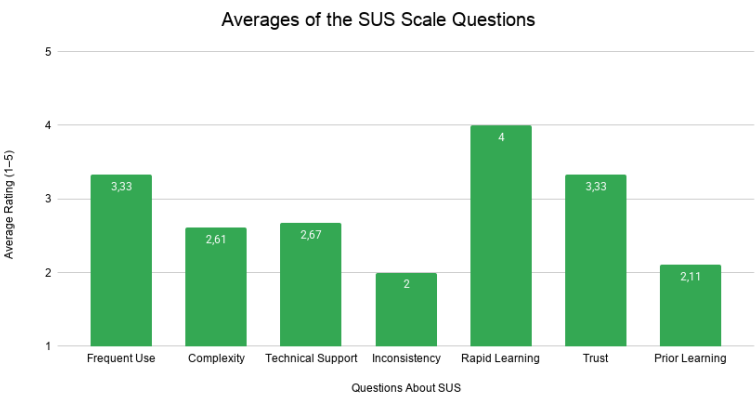


Figure 4.22: Average participant scores for the System Usability Scale (SUS) questionnaire.

Users explicitly praised the system’s interface, noting that the layout was easy to follow and the required actions felt highly intuitive. In addition to positive appraisals, the cohort provided constructive criticism aimed at refining the system’s interactive elements.

A specific recommendation highlighted the "Get Data (Visualization)" tab, where participants suggested that implementing real-time updates within the search bar would significantly enhance the user interface experience and streamline data retrieval processes.

4.2.3 Discussion

Despite the robust operational capacity demonstrated by the backend infrastructure, the user case elucidates a pronounced trade-off between systemic capability and manual operational complexity. Analysis of the SUS metrics and corresponding task execution logs reveals a significant dichotomy between basic administrative actions and advanced workflow configurations.

4.2.3.1 Operational Efficiency and Cognitive Load

Fundamental processes were perceived as highly intuitive by the test cohort. For instance, user authentication in Task 1 (averaging 21 seconds) and dashboard telemetry retrieval in Tasks 4 and 5 (averaging 43 and 42 seconds, respectively) yielded high ease-of-use scores above 6.0. Conversely,

the subsequent data conversion and subscription synthesis phases introduced substantial cognitive load.

Task 2, which required **CSV** standardization and injection, proved the most demanding with an average execution time of 2 minutes and 50 seconds and the lowest ease-of-use rating (4.76). Task 3, involving **GTFS** conversion, similarly extended execution durations beyond the 2-minute threshold (2 minutes and 23 seconds).

4.2.3.2 The Role of Technical Proficiency

All participants completed the task well within the 15 minute limit, with the longest sessions taking only 10 minutes. Because the platform is designed specifically for data and engineering professionals, this target demographic performed exceptionally well. Their average completion time was 05:39, significantly faster than the overall average of 06:59, reflecting their inherent familiarity with relational databases and complex data manipulation.

Importantly, users outside this specialized demographic also successfully executed complex workflows autonomously. Even without domain-specific training, a cohort largely made up of technologically literate young adults navigated the platform with ease. This ability of non-experts to seamlessly adopt the tool serves as a robust indicator of the system's strong foundational usability and broad accessibility.

Ultimately, the difference in completion times between these user profiles highlights the success of the interface design. The system is intuitive enough to guide beginners, yet highly optimized to accelerate the workflows of technical professionals. Any minor points of friction observed during testing do not indicate systemic flaws, but rather clear opportunities for future refinements to make the platform even more accessible.

4.3 Answer to the Research Questions

This section synthesizes the empirical findings and experimental validations conducted throughout the study to address the three core research questions formulated at the outset of this research. By correlating the platform's architectural capabilities, its security implementations, and the quantitative performance metrics derived from stress testing, the following subsections provide a comprehensive evaluation of the proposed urban mobility management system.

Specifically, the discussion details the efficacy of the hybrid data ingestion pipeline in resolving urban data heterogeneity (RQ1), the robustness of the decoupled access control mechanisms in ensuring regulatory compliance (RQ2), and the empirical scalability of the system under high velocity spatial and time-series workloads (RQ3).

4.3.1 RQ1 - How can a hybrid software architecture be designed to effectively ingest, process, and store heterogeneous and high-velocity data streams within an urban environment?

The empirical validation demonstrates that the homogenization of disparate data typologies into the unified **NGSI-LD** format via standardized Smart Data Models constitutes a highly effective methodology for mitigating urban data heterogeneity. The developed platform successfully ingested, contextualized, and mapped multi-structural variations ranging from tabular traffic records (**CSV**) and public transit schedules (**GTFS**) to complex spatial geometries (**GeoJSON**).

Furthermore, the hybrid backend topology proved indispensable for reconciling the competing demands of data velocity and structural persistence. The Orion-LD context broker efficiently managed the real-time state of urban entities, while the QuantumLeap service seamlessly offloaded high-frequency, time-series state updates to TimescaleDB for historical preservation. Continuous validation via the **IoT** Tester module substantiated this capability, confirming that the architecture maintains strict structural integrity and prevents semantic degradation or data attrition even when subjected to prolonged, high-throughput simulated telemetry feeds.

4.3.2 RQ2 - How can security mechanisms be optimally designed and integrated to simultaneously guarantee regulatory privacy compliance and robust access control?

Enforcing robust security protocols and strict regulatory compliance (e.g., **GDPR**) within a shared, multi-tenant municipal data environment represents a paramount architectural challenge. Within this framework, this imperative was addressed through the strategic integration of Keycloak to govern **IAM**.

The operational efficacy of this security layer was verified during Phase II of the user experience evaluation (Task 1). Participants authenticated into the platform via pre-configured security profiles without introducing operational friction, thereby validating that robust access control mechanisms can be cleanly decoupled from core data processing pipelines. This decoupled, layered security paradigm ensures that sensitive urban datasets, including high-resolution spatial trajectories and operational parking telemetry, are restricted exclusively to authorized principals and clients, establishing a verifiable baseline for data privacy and regulatory compliance without compromising systemic latency.

4.3.3 RQ3 - Which performance evaluation frameworks and scalability metrics are most effective for quantifying the efficiency of urban mobility management systems?

The quantitative performance metrics obtained during the stress testing of the system provide an empirical baseline to assess the scalability of the proposed hybrid architecture. The conversion of a massive 160 MB raw traffic dataset into compliant **NGSI-LD** entities necessitated 174.80 seconds,

whereas the subsequent batch ingestion of an 80 MB payload (comprising 50,000 discrete entities distributed across 100 batches) was executed within 119.58 seconds. Conversely, for lower-density spatial data layers, such as parking facility **GeoJSON** matrices, the computational overhead of conversion was mathematically negligible (0.00 seconds), and subscription routing latency was limited to a mere 0.23 seconds.

These behavioral metrics indicate that the platform exhibits predictable linear scalability relative to dataset volume. The query execution benchmarks, specifically the relational aggregations and PostGIS spatial operations (`ST_X`, `ST_Y`) executed to map Porto's public transit topology and parking infrastructure, confirm that consolidating time-series data within an optimized relational extension like TimescaleDB facilitates high-throughput, macro-level analytical querying.

Consequently, this architecture successfully circumvents the performance degradation characteristic of conventional relational databases under high-velocity JSON workloads, while simultaneously avoiding the spatial aggregation limitations inherent to purely **NoSQL** alternatives.

4.4 Summary

This chapter provided a rigorous evaluation of the developed hybrid urban computing platform, validating both its backend architectural integrity and its user-centric operational efficiency. System tests proved the platform can successfully break down urban data silos. It easily ingests, harmonizes, and visualizes diverse data, like transit schedules, maps, and high-speed **IoT**, by converting it all into a unified **NGSI-LD** standard.

Furthermore, performance benchmarks confirmed the architecture's computational efficiency and linear scalability. The strategic integration of the Orion-LD context broker for real-time state management, coupled with QuantumLeap and TimescaleDB for historical preservation, facilitated rapid data retrieval and high-throughput spatial aggregation without semantic degradation.

Concurrently, the user experience evaluation yielded critical insights into the platform's usability nuances. Although foundational tasks, such as secure authentication through Keycloak and macroscopic data visualization, were executed quickly and rated highly intuitive, quantitative data revealed a pronounced cognitive load associated with advanced data conversion and ingestion workflows. This operational friction, particularly evident among users outside the Information Technology domain, highlights a critical area for future systemic abstraction to better accommodate non-technical municipal planners.

Ultimately, synthesizing the systemic performance metrics with the usability feedback definitively addresses the study's research questions. The empirical findings substantiate that the proposed hybrid architecture, fortified by decoupled Identity and Access Management protocols, offers a scalable, secure, and highly capable infrastructure for modern urban mobility management.

Chapter 5

Conclusion and Future Work

This chapter concludes the dissertation by summarizing the key findings and contributions of the proposed UrbanData Nexus architecture. Additionally, it outlines potential avenues for future research and system optimization based on the empirical evaluations conducted.

5.1 Conclusion

The core objective of this dissertation was the design and implementation of a secure, scalable, and hybrid data architecture to facilitate the integration of heterogeneous urban datasets within smart city frameworks. To empirically validate the proposed architecture, the URBAN-NET project was utilized as a practical case study. Subsequent evaluation established that harmonizing disparate data typologies—namely, tabular traffic records (**CSV**), public transit schedules (**GTFS**), and spatial geometries (**GeoJSON**)—into a unified **NGSI-LD** standard via Smart Data Models constitutes a robust methodological approach for overcoming urban data silos.

The implemented hybrid backend topology successfully reconciled the competing demands of data velocity and structural persistence. By integrating the Orion-LD context broker for real-time state management with QuantumLeap and TimescaleDB for historical data preservation, the architecture facilitated rapid data retrieval and high-throughput spatial aggregation without semantic degradation. Performance benchmarks confirmed the computational efficiency and linear scalability of the platform, even when processing massive data volumes and high-velocity **IoT** telemetry.

Furthermore, the integration of Keycloak as a centralized **IAM** provider successfully enforced robust security protocols and regulatory compliance. This layered security paradigm validated that robust access control mechanisms can be cleanly decoupled from core data processing pipelines without introducing operational friction or compromising systemic latency.

In synthesis, the proposed architecture delivers a future-proof foundation tailored to the complex demands of modern urban mobility. It achieves unparalleled scalability by leveraging cloud-native technologies to seamlessly process massive volumes of transit data in real time, ultimately transforming reactive traffic management into a proactive, efficient, and sustainable transit ecosystem.

5.2 Future Work

While the developed platform demonstrated robust operational capacity, the user experience evaluation highlighted specific areas for future enhancement. The most prominent opportunity for future work lies in the systemic abstraction of the data ingestion and conversion frameworks. The usability testing revealed that advanced workflows, specifically data conversion and subscription synthesis, introduced substantial cognitive load for users outside the Information Technology and Engineering domains. To ensure sustainable deployment among non-technical municipal planners, future iterations of the platform must minimize manual intervention and streamline the ingestion pipeline.

Additionally, qualitative feedback from the testing cohort underscored the need for a comprehensive frontend redesign to establish a more intuitive overall user experience. As part of this broader interface overhaul, users identified specific optimizations. For instance, they noted that the "Send Data" feature currently restricts imports to a single file at a time. While a minor issue in the short term, this could create workflow inefficiencies during repeated use over time. Furthermore, future developments should focus on refining the "Get Data (Visualization)" tab by implementing real-time updates within the search bar, which would significantly enhance data retrieval processes and overall interface responsiveness.

Finally, future research could explore further integration of predictive analytics and machine learning modules directly into the data pipeline, leveraging standardized historical time-series data in TimescaleDB to anticipate urban mobility patterns and automate decision-making for smart city environments.

References

- [1] A. Akanbi and M. Masinde. “A Distributed Stream Processing Middleware Framework for Real-Time Analysis of Heterogeneous Data on Big Data Platform: Case of Environmental Monitoring”. In: *Sensors (Basel, Switzerland)* 20 (2020). URL: <https://api.semanticscholar.org/CorpusId:219399238>.
- [2] Marco Anisetti et al. “Dynamic and Scalable Enforcement of Access Control Policies for Big Data”. In: *Proceedings of the 13th International Conference on Management of Digital EcoSystems*. Association for Computing Machinery, 2021, pp. 71–78. ISBN: 9781450383141. DOI: [10.1145/3444757.3485107](https://doi.org/10.1145/3444757.3485107). URL: <https://doi.org/10.1145/3444757.3485107>.
- [3] Mudassar Aslam et al. “Getting Smarter about Smart Cities: Improving Data Security and Privacy through Compliance”. In: *Sensors (Basel, Switzerland)* 22 (2022). URL: <https://api.semanticscholar.org/CorpusId:254131035>.
- [4] Association for Computing Machinery. *ACM Digital Library*. <https://dl.acm.org/>. Accessed: 2025-10-31. 2025.
- [5] Hanen Balti et al. “Multidimensional architecture using a massive and heterogeneous data: Application to drought monitoring”. In: *Future Generation Computer Systems* 136 (2022), pp. 1–14. ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2022.05.010>. URL: <https://www.sciencedirect.com/science/article/pii/S0167739X22001753>.
- [6] Maggi Bansal, Inderveer Chana, and Siobhán Clarke. “A Survey on IoT Big Data: Current Status, 13 V’s Challenges, and Future Directions”. In: *ACM Comput. Surv.* 53 (6 Dec. 2020). ISSN: 0360-0300. DOI: [10.1145/3419634](https://doi.org/10.1145/3419634). URL: <https://doi.org/10.1145/3419634>.
- [7] Michael Batty. “Big data, smart cities and city planning”. In: *Dialogues in human geography* 3.3 (2013), pp. 274–279.
- [8] Simon Elias Bibri et al. “The synergistic interplay of artificial intelligence and digital twin in environmentally planning sustainable smart cities: A comprehensive systematic review”. In: *Environmental science and ecotechnology* 20 (2024), p. 100433.
- [9] Srdja Bjeladinovic, Zoran Marjanovic, and Sladjan Babarogic. “A proposal of architecture for integration and uniform use of hybrid SQL/NoSQL database components”. In: *Journal of Systems and Software* 168 (2020), p. 110633. ISSN: 0164-1212. DOI: <https://doi.org/10.1016/j.jss.2020.110633>. URL: <https://www.sciencedirect.com/science/article/pii/S0164121220301060>.

- [10] Georgios Bouloukakis et al. “Enabling IoT-enhanced Transportation Systems using the NGSI Protocol”. In: *Proceedings of the 12th International Conference on the Internet of Things*. Association for Computing Machinery, 2023, pp. 33–40. ISBN: 9781450396653. DOI: [10.1145/3567445.3567460](https://doi.org/10.1145/3567445.3567460). URL: <https://doi.org/10.1145/3567445.3567460>.
- [11] Rogério Luís de C Costa et al. “A Survey on Data-driven Performance Tuning for Big Data Analytics Platforms”. In: *Big Data Research* 25 (2021), p. 100206. ISSN: 2214-5796. DOI: <https://doi.org/10.1016/j.bdr.2021.100206>. URL: <https://www.sciencedirect.com/science/article/pii/S221457962100023X>.
- [12] Antonio Celesti et al. “Information management in IoT cloud-based tele-rehabilitation as a service for smart cities: Comparison of NoSQL approaches”. In: *Measurement* 151 (2020), p. 107218. ISSN: 0263-2241. DOI: <https://doi.org/10.1016/j.measurement.2019.107218>. URL: <https://www.sciencedirect.com/science/article/pii/S026322411931084X>.
- [13] Antonio Celesti et al. “Information management in IoT cloud-based tele-rehabilitation as a service for smart cities: Comparison of NoSQL approaches”. In: *Measurement* 151 (2020), p. 107218. ISSN: 0263-2241. DOI: <https://doi.org/10.1016/j.measurement.2019.107218>. URL: <https://www.sciencedirect.com/science/article/pii/S026322411931084X>.
- [14] C C Cerbulescu, M Marian, and E Ganea. “IoT Big Data Management for Improved Response Time”. In: *2021 22nd International Carpathian Control Conference (ICCC)*. 2021, pp. 1–6. DOI: [10.1109/ICCC51557.2021.9454644](https://doi.org/10.1109/ICCC51557.2021.9454644). URL: <https://ieeexplore.ieee.org/document/9454644>.
- [15] C C Cerbulescu, M Marian, and E Ganea. “IoT Data Management Architectures to Detect Critical Data Evolution”. In: *2024 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*. 2024, pp. 1–6. DOI: [10.1109/INISTA62901.2024.10683827](https://doi.org/10.1109/INISTA62901.2024.10683827). URL: <https://ieeexplore.ieee.org/abstract/document/10683827>.
- [16] C C Cerbulescu et al. “Optimize Critical Data Pattern Detection In Systems With Real Time Decisions”. In: *2022 23rd International Carpathian Control Conference (ICCC)*. 2022, pp. 324–330. DOI: [10.1109/ICCC54292.2022.9805932](https://doi.org/10.1109/ICCC54292.2022.9805932). URL: <https://ieeexplore.ieee.org/abstract/document/9805932>.
- [17] Javier Conde et al. “Enhanced FIWARE-Based Architecture for Cyberphysical Systems With Tiny Machine Learning and Machine Learning Operations: A Case Study on Urban Mobility Systems”. In: *IT Professional* 26 (5 2024), pp. 55–61. DOI: [10.1109/MITP.2024.3421968](https://doi.org/10.1109/MITP.2024.3421968).
- [18] Benoit Couraud et al. “GreenMov: A Fiware Based Interoperable Solution to Reduce the Environmental Impact of Mobility”. In: *Internet of Things*. Ed. by Gaëtan Rey, Jean-Yves Tigli, and Erwin Franquet. Springer Nature Switzerland, 2025, pp. 261–278. ISBN: 978-3-031-81900-1.
- [19] Ricardo Dintén, Sebastián García, and Marta Zorrilla. “Fleet management systems in Logistics 4.0 era: a real time distributed and scalable architectural proposal”. In: *Procedia Computer Science* 217 (2023), pp. 806–815. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2022.12.277>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050922023559>.

- [20] Juergen Doellner et al. “Towards Concepts for Climate and Energy-Oriented Digital Twins for Buildings”. In: *Proceedings of the 28th International ACM Conference on 3D Web Technology*. Association for Computing Machinery, 2023. ISBN: 9798400703249. DOI: [10.1145/3611314.3616066](https://doi.org/10.1145/3611314.3616066). URL: <https://doi.org/10.1145/3611314.3616066>.
- [21] Elsevier B.V. *ScienceDirect*. <https://www.sciencedirect.com/>. Accessed: 2025-10-31. 2025.
- [22] ETSI ISG CIM / NGSI-LD Initiative. *NGSI-LD — Next Generation Service Interfaces Linked Data*. <https://ngsi-ld.org/>. Accessed: 2025-10-31. 2025.
- [23] Faculdade de Engenharia da Universidade do Porto. *Faculdade de Engenharia da Universidade do Porto (FEUP)*. <https://www.up.pt/feup/pt/>. Accessed: 2025-10-31. 2025.
- [24] Nuno Faria et al. “Aida-db: A data management architecture for the edge and cloud continuum”. In: *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2022, pp. 1–6.
- [25] FIWARE. *FIWARE Context Broker: The engine for future energy systems*. <https://www.fiware.org/>. Accessed: 2025-10-31. 2020.
- [26] Daniel González-Sánchez et al. “Toward Building a Digital Twin for Network Operations and Management”. In: *IEEE Open Journal of the Communications Society* 6 (2025), pp. 2583–2598. URL: <https://api.semanticscholar.org/CorpusId:276920993>.
- [27] Christos Goumopoulos. “Smart city middleware: A survey and a conceptual framework”. In: *IEEE Access* 12 (2024), pp. 4015–4047.
- [28] Jelena Große-Bley and Genia Kostka. “Big data dreams and reality in Shenzhen: An investigation of smart city implementation in China”. In: *Big Data & Society* 8.2 (2021), p. 20539517211045171.
- [29] Eyosias Dawit Guyo and Timo Hartmann. “Evaluating the efficiency and performance of data persistent systems in managing building and environmental Data: A comparative study”. In: *Advanced Engineering Informatics* 62 (2024), p. 102582. ISSN: 1474-0346. DOI: <https://doi.org/10.1016/j.aei.2024.102582>. URL: <https://www.sciencedirect.com/science/article/pii/S1474034624002301>.
- [30] Yosra Hajjaji et al. “Big data and IoT-based applications in smart environments: A systematic review”. In: *Computer Science Review* 39 (2021), p. 100318. ISSN: 1574-0137. DOI: <https://doi.org/10.1016/j.cosrev.2020.100318>. URL: <https://www.sciencedirect.com/science/article/pii/S1574013720304184>.
- [31] Institute of Electrical and Electronics Engineers. *IEEE Xplore Digital Library*. <https://ieeexplore.ieee.org/>. Accessed: 2025-10-31. 2025.
- [32] Francisco J Jaime et al. “Strengthening privacy and data security in biomedical microelectromechanical systems by IoT communication security and protection in smart healthcare”. In: *Sensors* 23.21 (2023), p. 8944.
- [33] Jingjing Jiang et al. “Logistics industry monitoring system based on wireless sensor network platform”. In: *Computer Communications* 155 (2020), pp. 58–65. ISSN: 0140-3664. DOI: <https://doi.org/10.1016/j.comcom.2020.03.016>. URL: <https://www.sciencedirect.com/science/article/pii/S014036642030219X>.

- [34] P John et al. “Application of Time Series Database for IoT Smart City Platform”. In: *2023 Smart City Symposium Prague (SCSP)*. 2023, pp. 1–6. DOI: [10.1109/SCSP58044.2023.10146237](https://doi.org/10.1109/SCSP58044.2023.10146237). URL: <https://ieeexplore.ieee.org/abstract/document/10146237>.
- [35] Michiel Kallenberg et al. “Interoperable agricultural digital twins with reinforcement learning intelligence”. In: *Smart Agricultural Technology* 12 (2025), p. 101412. ISSN: 2772-3755. DOI: <https://doi.org/10.1016/j.atech.2025.101412>. URL: <https://www.sciencedirect.com/science/article/pii/S2772375525006434>.
- [36] JungHoon Kim. “Smart city trends: A focus on 5 countries and 15 companies”. In: *Cities* 123 (2022), p. 103551. ISSN: 0264-2751. DOI: <https://doi.org/10.1016/j.cities.2021.103551>. URL: <https://www.sciencedirect.com/science/article/pii/S0264275121004509>.
- [37] Aparna Kumari and Sudeep Tanwar. “Secure data analytics for smart grid systems in a sustainable smart city: Challenges, solutions, and future directions”. In: *Sustainable Computing: Informatics and Systems* 28 (2020), p. 100427. ISSN: 2210-5379. DOI: <https://doi.org/10.1016/j.suscom.2020.100427>.
- [38] Francisco Márquez-Saldaña, Gonzalo A. Aranda-Corral, and Joaquín Borrego-Díaz. “Bike sharing systems data interoperability by a unified station status concept and big data solutions”. In: *Journal of Traffic and Transportation Engineering (English Edition)* 12.2 (2025), pp. 420–433. ISSN: 2095-7564. DOI: <https://doi.org/10.1016/j.jtte.2024.06.003>. URL: <https://www.sciencedirect.com/science/article/pii/S2095756425000509>.
- [39] Agya Java Maulidin, Faiza Renaldi, and Fajri Rakhmat Umbara. “Online Integration of SQL and No-SQL Databases using RestAPIs: A Case on 2 furniture e-Commerce Sites”. In: *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*. 2020, pp. 261–266. DOI: [10.1109/IC2IE50715.2020.9274613](https://doi.org/10.1109/IC2IE50715.2020.9274613).
- [40] M Muñoz et al. “An Internet of Things platform for heterogeneous data integration: Methodology and application examples”. In: *Journal of Network and Computer Applications* 240 (2025), p. 104197. ISSN: 1084-8045. DOI: <https://doi.org/10.1016/j.jnca.2025.104197>. URL: <https://www.sciencedirect.com/science/article/pii/S1084804525000943>.
- [41] Edjola Naka and Vassil Guliashki. “Optimization Techniques in Data Management: A Survey”. In: *Proceedings of the 2021 7th International Conference on Computing and Data Engineering*. Association for Computing Machinery, 2021, pp. 8–13. ISBN: 9781450388450. DOI: [10.1145/3456172.3456214](https://doi.org/10.1145/3456172.3456214). URL: <https://doi.org/10.1145/3456172.3456214>.
- [42] Mohammad Nasar and Mohammad Abu Kausar. “Suitability of influxdb database for iot applications”. In: *International Journal of Innovative Technology and Exploring Engineering* 8 (10 Aug. 2019), pp. 1850–1857. ISSN: 22783075. DOI: [10.35940/ijitee.J9225.0881019](https://doi.org/10.35940/ijitee.J9225.0881019).
- [43] Efstratios Ntallaris, Georgios Bouloukakakis, and Kostas Magoutis. “Benchmarking Context-Aware Services for IoT-driven Transportation Systems”. In: *Proceedings of the 14th International Conference on the Internet of Things*. Association for Computing Machinery, 2025, pp. 99–107. ISBN: 9798400712852. DOI: [10.1145/3703790.3703802](https://doi.org/10.1145/3703790.3703802). URL: <https://doi.org/10.1145/3703790.3703802>.

- [44] Christoforos Ntantogian et al. “NodeXP: NDe.js server-side JavaScript injection vulnerability DEtection and eXPloitation”. In: *Journal of Information Security and Applications* 58 (2021), p. 102752. ISSN: 2214-2126. DOI: <https://doi.org/10.1016/j.jisa.2021.102752>. URL: <https://www.sciencedirect.com/science/article/pii/S221421262100003X>.
- [45] Open Worldwide Application Security Project. *OWASP Foundation*. <https://owasp.org/>. Accessed: 2025-11-24. 2025.
- [46] WK Perangin-Angin et al. “Mini robot for calibration of stopwatch/timer”. In: *Journal of Physics: Conference Series*. Vol. 1230. 1. 2019, p. 012029. DOI: [10.1088/1742-6596/1230/1/012029](https://doi.org/10.1088/1742-6596/1230/1/012029). URL: <https://doi.org/10.1088/1742-6596/1230/1/012029>.
- [47] Rui Humberto Pereira. “Lightweight data bridge for connecting self-service end-user analytic tools to NGSI-based IoT systems”. In: *Internet of Things* 25 (2024), p. 101125. ISSN: 2542-6605. DOI: <https://doi.org/10.1016/j.iot.2024.101125>. URL: <https://www.sciencedirect.com/science/article/pii/S2542660524000672>.
- [48] PRISMA Executive. *PRISMA Statement — Preferred Reporting Items for Systematic reviews and Meta-Analyses*. <https://www.prisma-statement.org/>. Accessed: 2025-10-31. 2025.
- [49] M A Rifqi, A Y Husodo, and H Wijayanto. “Development of SQL and NoSQL Database Integration Using Query Direct Access and Change Data Capture Approaches in an Android Application”. In: *2024 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*. 2024, pp. 671–679. DOI: [10.1109/COMNETSAT63286.2024.10862087](https://doi.org/10.1109/COMNETSAT63286.2024.10862087). URL: <https://ieeexplore.ieee.org/abstract/document/10862087>.
- [50] Samer Sawalha and Ghazi Al-Naymat. “Towards an efficient big data management schema for IoT”. In: *Journal of King Saud University - Computer and Information Sciences* 34 (9 2022), pp. 7803–7818. ISSN: 1319-1578. DOI: <https://doi.org/10.1016/j.jksuci.2021.09.013>. URL: <https://www.sciencedirect.com/science/article/pii/S1319157821002640>.
- [51] Geomar A Schreiner et al. “A Hybrid Partitioning Strategy for NewSQL Databases: The VoltDB Case”. In: *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services*. Association for Computing Machinery, 2020, pp. 353–360. ISBN: 9781450371797. DOI: [10.1145/3366030.3366062](https://doi.org/10.1145/3366030.3366062). URL: <https://doi.org/10.1145/3366030.3366062>.
- [52] Geomar A Schreiner et al. “NewSQL Through the Looking Glass”. In: *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services*. Association for Computing Machinery, 2020, pp. 361–369. ISBN: 9781450371797. DOI: [10.1145/3366030.3366080](https://doi.org/10.1145/3366030.3366080). URL: <https://doi.org/10.1145/3366030.3366080>.
- [53] Sabrina Sicari, Alessandra Rizzardi, and Alberto Coen-Porisini. “Security&privacy issues and challenges in NoSQL databases”. In: *Computer Networks* 206 (2022), p. 108828. ISSN: 1389-1286. DOI: <https://doi.org/10.1016/j.comnet.2022.108828>.
- [54] Bhagya Nathali Silva et al. “Urban Planning and Smart City Decision Management Empowered by Real-Time Data Processing Using Big Data Analytics”. In: *Sensors (Basel, Switzerland)* 18 (2018). URL: <https://api.semanticscholar.org/CorpusId:52187370>.

- [55] V Srinivasan et al. “Techniques and Efficiencies from Building a Real-Time DBMS”. In: *Proc. VLDB Endow.* 16 (12 Aug. 2023), pp. 3676–3688. ISSN: 2150-8097. DOI: [10.14778/3611540.3611556](https://doi.org/10.14778/3611540.3611556). URL: <https://doi.org/10.14778/3611540.3611556>.
- [56] Artem A Sukhobokov et al. “The concept of an intelligent data lake management system: machine consciousness and a universal data model”. In: *Procedia Computer Science* 213 (2022), pp. 407–414. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2022.11.085>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050922017768>.
- [57] Fatima Tanveer et al. “Towards Secure APIs: A Survey on RESTful API Vulnerability Detection”. In: *Computers, Materials and Continua* 84 (3 2025), pp. 4223–4257. ISSN: 1546-2218. DOI: <https://doi.org/10.32604/cmc.2025.067536>. URL: <https://www.sciencedirect.com/science/article/pii/S1546221825007040>.
- [58] Denis Ushakov et al. “Big data analytics in smart cities’ transportation infrastructure modernization”. In: *Transportation Research Procedia* 63 (2022), pp. 2385–2391. ISSN: 2352-1465. DOI: <https://doi.org/10.1016/j.trpro.2022.06.274>. URL: <https://www.sciencedirect.com/science/article/pii/S2352146522005294>.
- [59] Sathishkumar Veerappampalayam et al. “Next-Generation Databases: A Comparative Insight into NoSQL and NewSQL Systems”. In: *2025 International Conference on Emerging Technologies in Engineering Applications (ICETEA)*. 2025, pp. 1–8. DOI: [10.1109/ICETEA64585.2025.11099997](https://doi.org/10.1109/ICETEA64585.2025.11099997).
- [60] Michael Wilbur et al. “Efficient Data Management for Intelligent Urban Mobility Systems”. In: *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*. Association for Computing Machinery, 2021, pp. 22–26. ISBN: 9781450384452. DOI: [10.1145/3459609.3460527](https://doi.org/10.1145/3459609.3460527). URL: <https://doi.org/10.1145/3459609.3460527>.
- [61] Fei Yang et al. “A survey on multisource heterogeneous urban sensor access and data management technologies”. In: *Measurement: Sensors* 19 (2022), p. 100061. ISSN: 2665-9174. DOI: <https://doi.org/10.1016/j.measen.2021.100061>. URL: <https://www.sciencedirect.com/science/article/pii/S2665917421000246>.

Appendix A

Validation

A.1 User Survey

This annexe presents the complete questionnaire utilized for the user survey.

URBAN-NET Database Validation Test

O presente formulário foi desenvolvido por João Costa, estudante do Mestrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto (FEUP). Tem como principal objetivo a recolha de informação com vista à documentação e validação da execução dos testes de integração da plataforma.

O fluxo em análise contempla a validação do mecanismo de autenticação (Keycloak), os processos de conversão e ingestão de dados nos formatos CSV e GTFS (recorrendo aos componentes Orion-LD e QuantumLeap), bem como a correta representação e visualização da informação na interface Streamlit.

English version

This form was developed by João Costa, a master's student in Informatics and Computing Engineering at the Faculty of Engineering of the University of Porto (FEUP). Its primary objective is to collect data to document and validate the execution of the platform's integration tests.

The testing workflow encompasses the validation of the user authentication mechanism (Keycloak), the conversion and ingestion processes of data files in CSV and GTFS formats (utilizing the Orion-LD and QuantumLeap components), as well as the accurate representation and visualization of the data on the Streamlit interface.

** Indica uma pergunta obrigatória*

Figure A.1: User survey questionnaire: Page 1.

Obrigado por participar neste estudo. Estamos a testar uma base de dados híbrida, com mecanismos de conversão de ficheiros CSV e GTFS, e na sua injeção e consulta em bases de dados de tempo real e histórico. O teste demorará aproximadamente 10 a 15 minutos. Os seus dados e resultados de interação permanecerão anónimos. *

English version

Thank you for participating in this study. We are testing a hybrid database that includes mechanisms for converting CSV and GTFS files, as well as for loading and querying real-time and historical databases. The test will take approximately 10 to 15 minutes. Your data and interaction results will remain anonymous.

☐ Compreendo o procedimento e concordo em participar neste estudo. (I understand the procedure and agree to participate in this study.)

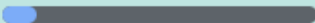
Seguinte  Página 1 de 9 [Limpar formulário](#)

Figure A.2: User survey questionnaire: Page 2.

The image shows a web form titled "URBAN-NET Database Validation Test". Below the title is a red asterisk and the text "* Indica uma pergunta obrigatória". The form is divided into sections by teal-colored headers. The first section is titled "Pre-Test Questionnaire" and contains two paragraphs: "Por favor, responda a algumas perguntas sobre o seu perfil. Isto ajuda-me a categorizar os participantes." and "English version Please answer a few questions about your background. This helps me categorize participants." The second section is titled "Age Group *" and contains five radio button options: "< 18", "18-24", "25-34", "35-44", and "45+".

URBAN-NET Database Validation Test

* Indica uma pergunta obrigatória

Pre-Test Questionnaire

Por favor, responda a algumas perguntas sobre o seu perfil. Isto ajuda-me a categorizar os participantes.

English version
Please answer a few questions about your background. This helps me categorize participants.

Age Group *

☐ < 18

☐ 18-24

☐ 25-34

☐ 35-44

☐ 45+

Figure A.3: User survey questionnaire: Page 3.

Género (Gender) *

☐ Homem (Male)

☐ Mulher (Female)

☐ Prefiro não dizer (Prefer not to say)

☐ Outra: _____

Qual é a sua área principal de formação ou a sua função profissional atual? *

English version

What is your main area of study or your current professional role?

☐ Gestão (Management)

☐ Tecnologias de Informação (Information Technologies)

☐ Engenharia (Engineering)

☐ Transportes (Transportation)

☐ Estudante (Student)

☐ Outra: _____

Figure A.4: User survey questionnaire: Page 4.

Quão confortável se sente a utilizar aplicações web no seu dia a dia? *

English version
How comfortable do you feel using web applications in your daily life?

1 2 3 4 5 6 7

Nada Confortável (Not Comfortable at All) ☐ ☐ ☐ ☐ ☐ ☐ ☐ Muito confortável (Very comfortable)

Com que frequência utiliza o computador para tarefas além da navegação básica *
(ex: gestão de pastas zipadas, instalação de software, organização complexa de ficheiros)?

English version
How often do you use your computer for tasks beyond basic browsing (e.g., managing zipped folders, installing software, complex file organization)?

☐ Diariamente (Daily)

☐ Semanalmente (Weekly)

☐ Raramente (Rarely)

☐ Nunca (Never)

[Anterior](#) [Seguinte](#)[Limpar formulário](#) Página 2 de 9

Figure A.5: User survey questionnaire: Page 5.

URBAN-NET Database Validation Test

* Indica uma pergunta obrigatória

Test Case 1: Autenticação e Acesso à Aplicação (Keycloak)

Objetivo: Validar o acesso à aplicação frontend e garantir que o fluxo de autenticação externo funciona corretamente.

1. **Aceder à Aplicação:** Abrir o browser e ir a <http://localhost:8501>.
2. **Iniciar Sessão:** Clicar em login, inserir as credenciais da conta fornecida no ecrã do Keycloak e submeter.

- User: testuser
- Password: 1234

English version

Objective: Validate access to the frontend application and ensure that the external authentication flow works correctly.

1. **Accessing the Application:** Open your browser and go to <http://localhost:8501>.
2. **Logging In:** Click "Login," enter the account credentials provided on the Keycloak screen, and submit.

- User: testuser
- Password: 1234

Figure A.6: User survey questionnaire: Page 6.

No geral, foi difícil ou fácil realizar a tarefa? *

English version
Overall, how difficult or easy was to complete the task?

1 2 3 4 5 6 7

Difícil (Difficult) ☐ ☐ ☐ ☐ ☐ ☐ ☐ Fácil (Easy)

Anterior Seguinte

 Página 3 de 9

Limpar formulário

Figure A.7: User survey questionnaire: Page 7.

Test Case 2: Conversão de CSV e Carga em Real-Time (Orion-LD)

Objetivo: Validar a conversão de ficheiros CSV pela aplicação e a respetiva injeção de dados na base de dados de tempo real (Orion Context Broker).

1. **Navegação:** Na aplicação, clicar na aba chamada " **Convert CSV**".
2. **Configuração do Upload:** Selecionar o tipo de dados correto no seletor da interface para corresponder ao ficheiro. Neste caso, *TrafficFlowObserved*.
3. **Upload:** Fazer upload do ficheiro CSV de teste, "tests_csv.csv" que se encontra na pasta "testes tese" no Ambiente de Trabalho, para a aplicação.
4. **Download:** Clicar no botão para descarregar a pasta zipada gerada pela aplicação com os dados convertidos.
5. **Extração e Validação Local:** No sistema operativo local, fazer *unzip* da pasta descarregada, "csv_converted_trafficflowobserved_data.zip", e verificar a existência do ficheiro, "trafficflowobserved_part1".
6. **Carga de Dados:** Voltar à aplicação, clicar na aba " **Send Data (Ingestion)**" e executar a ação de "load"/"enviar" para a base de dados *real-time* (Orion-LD), nome do ficheiro: "trafficflowobserved_part1".

English version

Objective: Validate the application's conversion of CSV files and the subsequent injection of data into the real-time database (Orion Context Broker).

1. **Navigation:** In the application, click the tab labeled " **Convert CSV**".
2. **Upload Configuration:** Select the correct data type from the interface dropdown to match the file. In this case, *TrafficFlowObserved*.
3. **Upload:** Upload the test CSV file, "tests_csv.csv," located in the "testes tese" folder on your desktop, to the application.
4. **Download:** Click the button to download the zipped folder generated by the application containing the converted data.
5. **Local Extraction and Validation:** On the local operating system, unzip the downloaded folder, "csv_converted_trafficflowobserved_data.zip", and check if the file exists, "trafficflowobserved_part1".
6. **Data Loading:** Return to the application, click the " **Send Data (Ingestion)**" tab, and execute the "load"/"send" action to the real-time database (Orion-LD), file name: "trafficflowobserved_part1".

Figure A.8: User survey questionnaire: Page 8.

No geral, foi difícil ou fácil realizar a tarefa? *

English version
Overall, how difficult or easy was to complete the task?

1 2 3 4 5 6 7

Difícil (Difficult) ☐ ☐ ☐ ☐ ☐ ☐ ☐ Fácil (Easy)

Anterior Seguinte

Página 4 de 9

Limpar formulário

Figure A.9: User survey questionnaire: Page 9.

Test Case 3: Conversão de GTFS e Carga de Dados Históricos (QuantumLeap)

Objetivo: Validar o processamento de ficheiros de transportes (GTFS) e a persistência de séries temporais na base de dados histórica (TimescaleDB via QuantumLeap).

1. **Navegação:** Na aplicação Streamlit, clicar na aba "🗺️ **Convert GTFS**".
2. **Configuração do Upload:** Selecionar o tipo de estrutura/dados correspondente ao ficheiro GTFS a inserir. Neste caso, *UrbanMobility*.
3. **Upload:** Fazer upload do ficheiro GTFS, "tests_gtfs.zip" que se encontra na pasta "testes tese" no Ambiente de Trabalho.
4. **Download:** Clicar no botão para converter e depois descarregar a nova pasta zipada gerada.
5. **Extração:** Fazer *unzip* localmente, "gtfs_converted_urbanmobility_data", e validar a integridade dos dados gerados, "stop_times", "stops", "trips".
6. **Carga de Dados:** Na aplicação Streamlit, clicar na aba "📤 **Send Data (Ingestion)**". Executar a ação de "load"/"enviar" para a base de dados histórica (QuantumLeap), nome dos ficheiros: "stop_times", "stops", "trips". *Nota:* Os ficheiros tem de ser enviados individualmente.

English version

Objective: To validate the processing of transportation files (GTFS) and the persistence of time series in the historical database (TimescaleDB via QuantumLeap).

1. **Navigation:** In the Streamlit application, click the "🗺️ **Convert GTFS**" tab.
2. **Upload Configuration:** Select the structure/data type corresponding to the GTFS file to be uploaded. In this case, *UrbanMobility*.
3. **Upload:** Upload the GTFS file, "tests_gtfs.zip," located in the "testes tese" folder on your desktop.
4. **Download:** Click the button to convert and then download the newly generated zipped folder.
5. **Extraction:** Unzip locally, "gtfs_converted_urbanmobility_data", and verify the integrity of the generated data, "stop_times", "stops", "trips".
6. **Data Loading:** In the Streamlit application, click the "📤 **Send Data (Ingestion)**" tab. Execute the "load"/"send" action to the historical database (QuantumLeap), files names: "stop_times", "stops", "trips". *Note:* Files must be submitted individually.

Figure A.10: User survey questionnaire: Page 10.

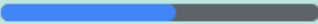
No geral, foi difícil ou fácil realizar a tarefa? *

English version
Overall, how difficult or easy was to complete the task?

1 2 3 4 5 6 7

Difícil (Difficult) ☐ ☐ ☐ ☐ ☐ ☐ ☐ Fácil (Easy)

Anterior Seguinte

 Página 5 de 9

Limpar formulário

Figure A.11: User survey questionnaire: Page 11.

URBAN-NET Database Validation Test

* Indica uma pergunta obrigatória

Test Case 4: Visualização de Dados em Real-Time

Objetivo: Garantir que a aplicação frontend consegue consultar e apresentar os dados atuais guardados no Context Broker.

- Navegação:** Clicar na aba "📊 Get Data (Visualization)".
- Filtro/Seleção:** Escolher a opção para visualizar dados *real-time* e procurar pelo ID `urn:ngsi-Id:TrafficFlowObserved:9577808`.
- Visualização:** Analisar a interface gerada (tabelas, mapas ou métricas).

English version

Objective: To ensure that the frontend application can retrieve and display the current data stored in the Context Broker.

- Navigation:** Click the "📊 Get Data (Visualization)" tab.
- Filter/Selection:** Select the option to view real-time data and search for the ID `urn:ngsi-Id:TrafficFlowObserved:9577808`.
- Visualization:** Review the generated interface (tables, maps, or metrics).

No geral, foi difícil ou fácil realizar a tarefa? *

English version

Overall, how difficult or easy was to complete the task?

	1	2	3	4	5	6	7	
Difícil (Difficult)	☐	☐	☐	☐	☐	☐	☐	Fácil (Easy)

Figure A.12: User survey questionnaire: Page 12.

URBAN-NET Database Validation Test

* Indica uma pergunta obrigatória

Test Case 5: Visualização de Dados Históricos

Objetivo: Confirmar que a aplicação consegue consultar séries temporais através do QuantumLeap e apresentar a evolução dos dados.

1. **Navegação:** Permanecer na aba "📊 Get Data (Visualization)".
2. **Filtro/Seleção:** Apague os conteúdos do "Campo de texto" do ID e depois procura o tipo **GtfsStop**.
3. **Visualização:** Tabelas históricas apresentadas no ecrã.

English version

Objective: To verify that the application can retrieve time series data via QuantumLeap and display the data trends.

1. **Navigation:** Stay on the "📊 Get Data (Visualization)" tab.
2. **Filter/Selection:** Delete the contents of the "Text field" for the ID, and then search for the type **GtfsStop**.
3. **Visualization:** Historical tables displayed on the screen.

No geral, foi difícil ou fácil realizar a tarefa? *

English version

Overall, how difficult or easy was to complete the task?

1

2

3

4

5

6

7

Difícil (Difficult)

☐
☐
☐
☐
☐
☐
☐

Fácil (Easy)

Figure A.13: User survey questionnaire: Page 13.

System Usability Scale (SUS)

Avalie as seguintes afirmações sobre o sistema que acabou de utilizar.

English version
Please rate the following statements about the system you just used.

	Discordo totalmente (Strongly Disagree)	Discordo (Disagree)	Neutro (Neutral)	Concordo (Agree)	Concordo totalmente (Strongly Agree)
Acho que gostaria de utilizar este sistema frequentemente. (I think I'd like to use this system often.)	☐	☐	☐	☐	☐
Achei o sistema desnecessariamente complexo. (I found the system unnecessarily complex.)	☐	☐	☐	☐	☐
Acho que precisaria de apoio técnico para conseguir utilizar este sistema. (I think I would need technical support to be able to use this system.)	☐	☐	☐	☐	☐

Figure A.14: User survey questionnaire: Page 14.

The image shows a user survey questionnaire interface. It consists of four rows, each with a statement in Portuguese and English, followed by a five-point Likert scale represented by five empty circles. The statements are:

- Achei que havia demasiada inconsistência neste sistema. (I thought there were too many inconsistencies in this system.)
- Imagino que a maioria das pessoas aprenderia a utilizar este sistema muito rapidamente. (I imagine most people would learn how to use this system very quickly.)
- Senti-me muito confiante ao utilizar o sistema. (I felt very confident using the system.)
- Tive de aprender muitas coisas antes de conseguir lidar com este sistema. (I had to learn a lot before I could figure out how to use this system.)

At the bottom of the form, there are two buttons: "Anterior" (Previous) and "Seguinte" (Next). To the right of these buttons is a green progress bar. Further right, it says "Página 8 de 9" (Page 8 of 9). On the far right, there is a link that says "Limpar formulário" (Clear form).

Figure A.15: User survey questionnaire: Page 15.

URBAN-NET Database Validation Test

Feedback qualitativo (Qualitative Feedback)

Por favor, partilhe as suas considerações finais.

English version
Please share your final thoughts.

Numa frase, como descreveria a sua experiência com esta ferramenta de fluxo de dados?

English version
In one sentence, how would you describe your experience using this data pipeline tool?

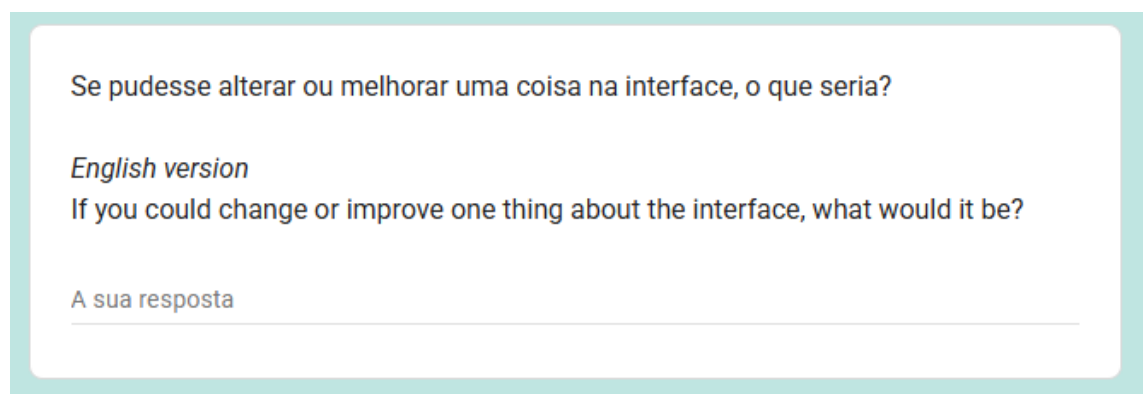
A sua resposta

Se tivesse de utilizar esta ferramenta diariamente, qual seria o seu maior incómodo ou ponto fraco?

English version
If you had to use this tool on a daily basis, what would be your biggest annoyance or pain point?

A sua resposta

Figure A.16: User survey questionnaire: Page 16.

A screenshot of a user survey questionnaire page. The page has a light blue border. Inside, there is a white rounded rectangle containing the text. The text is in Portuguese, English, and Portuguese again, with a line for the answer.

Se pudesse alterar ou melhorar uma coisa na interface, o que seria?

English version
If you could change or improve one thing about the interface, what would it be?

A sua resposta

Figure A.17: User survey questionnaire: Page 17.