ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

BACHELOR THESIS

«OpenSCN: Bridging Smart Campus and Smart City Concepts through an IoT Platform for Enhanced Educational Environments»

Student
Georgios Apostolopoulos
Student ID: 175012

Supervisor
Prof. Periklis Chatzimisios

September 10, 2023

Thesis Title: OpenSCN: Bridging Smart Campus and Smart City Concepts through an IoT Platform for Enhanced Educational Environments
Thesis Code: 22217
Student's Full Name: Georgios Apostolopoulos
Supervisor's Full Name: Prof. Periklis Chatzimisios
Date of Thesis Initiation: 07/04/2022
Date of Thesis Completion: 10/09/2023

I certify that I am the author of this thesis and that any assistance I received in preparing it is fully acknowledged and referenced in the thesis. I have also documented any sources from which I have used data, ideas, images, and text, whether they are quoted or paraphrased. Furthermore, I certify that this thesis was prepared by me personally, specifically as a bachelor's thesis, at the Department of Computer Engineering and Electronic Systems of the International University of Greece.

«Στην οικογένειά μου για τη στήριξή τους καθόλη τη φοιτητική μου πορεία.»

Περίληψη

Το OpenSCN, μία πλατφόρμα IoT ανοιχτού κώδικα, επεκτείνει την έννοια των Έξυπνων Πόλεων στον ακαδημαϊκό τομέα, κάνοντας δυνατή τη δημιουργία ευφυών, οδηγούμενων απο δεδομένα πανεπιστημιακών χώρων. Το εν λόγω ολιστικό πλαίσιο στοχεύει στο να συνδυάσει έξυπνες υποδομές, δίκτυα σενσόρων, διαχείριση και ανάλυση δεδομένων, βοηθώντας στη λήψη πληροφοριών και αποφάσεων. Η παρούσα Διπλωματική Εργασία εξερευνά τα θεμέλια της ενοποίησης του IoT στις πολιτικές υποδομές και την επίδραση της στη βιωσιμότητα, αποδοτικότητα και ποιότητα ζωής. Η αρχιτεκτονική του OpenSCN, η οποία δίνει έμφαση στην αποθήκευση time series δεδομένων, στη δυνατότητα δικτύωσης μέσω MQTT, στον σχεδιασμό γύρω από γεγονότα, τη δυνατότητα αξιοποίησης υποδομών νέφους και στην ασφάλεια συσκεύων, ξεχωρίζει την πλατφόρμα ως μία πλήρη λύση φιλική προς τους χρήστες, κατάλληλη για ακαδημαϊκούς οργανισμούς. Η σύγκριση με υπόλοιπες λύσεις τοποθετεί το OpenSCN ως μία μοναδική εναλλακτική ανοιχτού κώδικα που εξισσοροπεί την προσβασιμότητα με την λειτουργικότητα. Η Διπλωματική Εργασία επίσης εξετάζει τα ειδικά θέματα του εσωτερικού σχεδιασμού της πλατφόρμας καθώς και τις μελλοντικές βελτιώσεις που μπορούν να επιτευχθούν.

«OpenSCN: Bridging Smart Campus and Smart City Concepts through an IoT Platform for Enhanced Educational Environments»

«Georgios Apostolopoulos»

Abstract

OpenSCN, an open-source IoT platform, extends the concept of Smart Cities into academia, enabling the creation of intelligent, data-driven Smart Campuses. This holistic framework aims to combine Smart Infrastructure, IoT Sensor Networks, Data Management, and Analytics, facilitating informed decision-making. The current Diploma Thesis explores the foundations of IoT integration into civilian infrastructure and its impact on sustainability, efficiency, and quality of life. OpenSCN's architecture, emphasizing time series data storage, MQTT networking, event-driven design, cloud deployability, and robust device security, distinguishes it as a user-friendly, all-in-one solution tailored for educational institutions. The performed comparative analysis positions OpenSCN as a unique, open-source alternative, balancing accessibility with functionality. The current Diploma Thesis also delves into the intricacies of the internal design of the platform as well as the future work that can be considered to achieve further improvements.

Acknowledgments

Firstly, I would like to express my gratitude to my supervisor Prof. Chatzimisios for his trust to undertake and complete this thesis. I appreciate the time he dedicated and his guidance, willingness, and support throughout the writing process. Additionally, I would like to thank my family for their unwavering support throughout my studies.

Contents

List of Figures

Chapter 1: Introduction

1.1 Internet of Things

The Internet of Things (IoT) [2] refers to a paradigm where objects, devices, and sensors are seamlessly connected through Internet, enabling them to collect, exchange, and process data autonomously. These interconnected entities interact and cooperate to gather real-time information, facilitate data-driven decisions, and enhance functionalities across various domains.

The term ”Internet of Things” gained prominence in the late 1990s and early 2000s [3] since the technological world recognized the transformative potential of connecting a vast array of objects and devices to Internet. Over the past two decades, the IoT landscape has experienced exponential growth, reshaping industries and domains across the globe. What was once a theoretical concept has matured into a tangible reality, with billions of interconnected devices spanning from wearable fitness trackers to industrial machinery.

This period of extraordinary growth and adoption of IoT can be attributed to several key technological advancements. Namely, the miniaturization of devices and sensors, the evolution of energy efficient wireless communication protocols and the rise of edge computing and energy harvesting solutions are some of the achievements which have collectively driven the IoT boom by enabling seamless connectivity, efficient data exchange, intelligent processing, and new opportunities for innovation across various industries.

The Internet of Things has enabled transformative shifts across diverse sectors, revolutionizing industries by harnessing the power of connectivity and data-driven insights. In smart cities, IoT-enabled systems optimize urban infrastructure, enhancing energy efficiency, waste management, and public services. In healthcare, IoT devices monitor patients remotely, enabling timely interventions and personalized treatment plans. Agriculture leverages IoT to enhance crop management, precision agriculture, and resource conservation. The automotive sector adopts IoT for smart navigation, vehicle diagnostics, and safety enhancements. Likewise, logistics and supply chain management benefit from real-time tracking, inventory optimization, and streamlined operations. As IoT permeates industries, its potential to optimize resource utilization, enhance decision-making, and elevate user experiences is driving a profound revolution across sectors.

Despite the fact that the traction gained these past years for the technology of IoT has been tremendous, the future is looking even brighter. The consistent growth of the total number of interconnected smart devices is not projected to come to a halt, but rather maintain a steady increasing pace [Figure 1.1], thanks to the ever advancing technological landscape. In particular, the advent and widespread adoption of the newer wireless network generations of 5G and 6G will usher in an era of massive IoT deployment with unprecedented capabilities [4]. Anticipated to deliver ultra-high data rates, exceptionally low latency, and enhanced device density, 6G will enable seamless connectivity for a multitude of IoT devices. Additionally, the recent history-defining breakthroughs in artificial intelligence will further drive the need for data collection from smart device networks and will turn industries to leveraging the power of data-driven decisions. The synergy between Artificial Intelligence (AI) and IoT will amplify the capabilities of

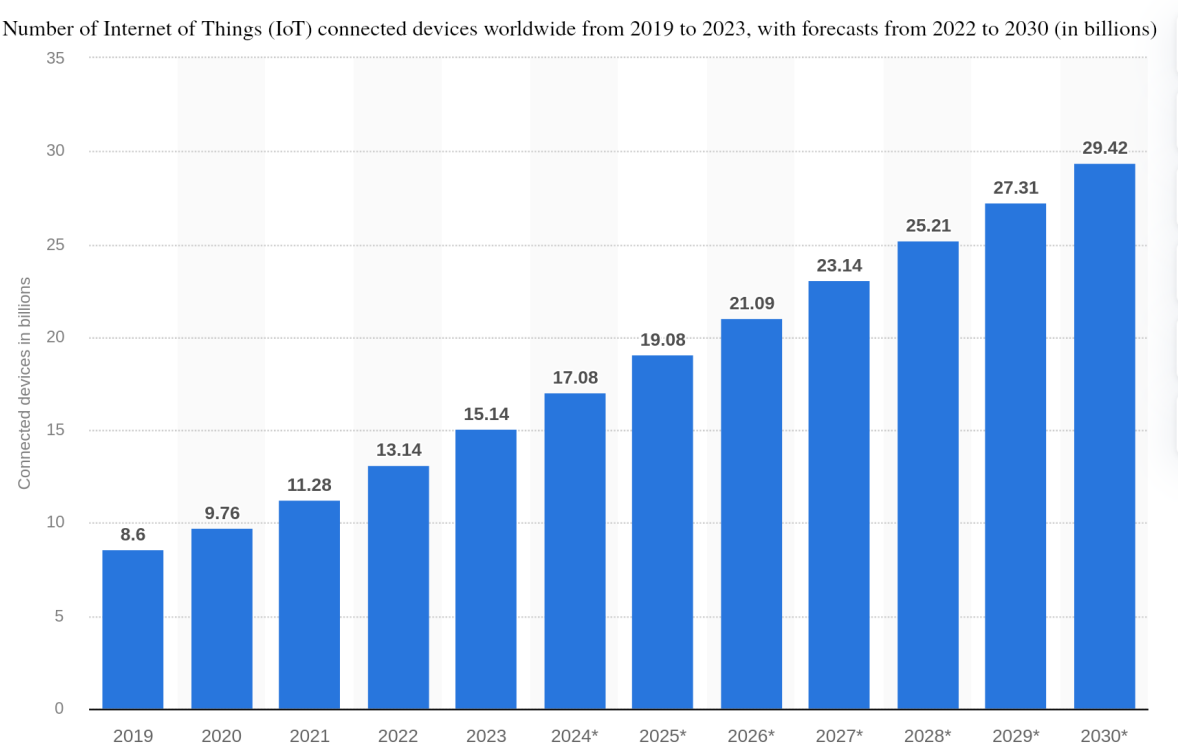

Figure 1.1: Number of connected IoT devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030 in billions (source: Statista)

connected systems, driving innovation across industries and transforming how data is utilized to enhance efficiency and effectiveness.

1.2 Motivation

The motivation behind the current Thesis lies in the pursuit of creating an open-source alternative to conventional smart campus solutions that is both performance-driven and scalable, while fostering collaboration among student developers, educators, and administrators. Traditional proprietary solutions often come with substantial financial barriers and limitations that hinder innovation and customization. By developing an open-source IoT platform, we aim to break down these barriers and provide a platform that encourages participation from students, teachers, and developers alike, enabling them to contribute, customize, and iterate upon a shared framework. This collaborative ecosystem is envisioned to foster creativity within our university, as we aspire to provide our co-students with the ability to seamlessly create and integrate their own projects into our platform, building up our campus' smart infrastructure block by block.

Furthermore, our motivation extends to leveraging the advantages of IoT technology within the educational domain. An open-source smart campus platform offers the potential to optimize resource utilization, enhance security measures, and provide real-time insights for campus management. With capabilities ranging from smart energy management and environmental monitoring to seamless campus navigation, the IoT platform aims to empower educational institutions to harness the benefits of connectivity, data analytics, and automation.

Lastly, this endeavor is driven by a profound desire to explore and learn about emerging technologies in the realm of IoT, cloud computing, data analytics, and beyond. By embarking on this project, we embrace the opportunity to delve into the intricacies of platform architecture, data handling, and user experience design. The pursuit of knowledge and the challenge of translating theoretical concepts into practical solutions are at the core of this motivation.

In summary, this thesis embarks on the journey to develop an open-source IoT platform for smart campus solutions with the overarching motivation of democratizing IoT technology, enabling collaboration, enhancing campus operations, and engaging with emerging technologies. Through this endeavor, we aspire to contribute to the educational and technological landscape by providing a dynamic platform that empowers students, educators, and institutions to explore the potential of IoT in an inclusive and innovative manner.

1.3 Thesis Organization

This thesis is organized into several chapters and sections, each designed to comprehensively present the development, architecture, and applications of OpenSCN, our IoT platform for smart campus solutions. Chapter 1 already served as an introduction to the core concepts of IoT as well as an overview of our motivation behind this endeavor.

Chapter 2 delves into the broader landscape of IoT within civilian infrastructure. We introduce the concept of smart cities, outlining their key components, benefits, challenges, and real-world case studies. This background establishes the context within which our development of a smart campus solution is situated, showcasing the transferability of IoT principles from urban environments to educational settings.

Building upon the insights gained from the previous chapter, Chapter 3 specifically focuses on the extension of smart city techniques to educational environments. We outline the rationale behind the creation of OpenSCN, highlighting the motivation to foster collaboration among student developers and educators while facilitating the utilization of IoT to enhance campus operations. This chapter further examines the components, applications, benefits, integration with educational goals, challenges, and exemplary solutions within the realm of smart campus initiatives.

The subsequent Chapter 4 offers an in-depth analysis of the technological infrastructure that underpins OpenSCN. Each subsection within this chapter examines crucial aspects of the platform, including time series data storage and analysis, MQTT networking, event-driven architecture, cloud deployability, containerization, and device security. This comprehensive exploration sheds light on the intricacies of building a robust, scalable, and secure IoT platform tailored to the unique needs of a smart campus.

In Chapter 5, we provide a detailed overview of the OpenSCN platform's architecture and functionalities. Subsections delve into various components of the platform, such as data modeling, web APIs, time series databases, analytics, and communication mechanisms. Additionally, we explore the user-facing web client, elucidating its role in empowering users to interact with the IoT platform and extract meaningful insights from the collected data.

The concluding Chapter 6 synthesizes the findings and contributions made throughout the thesis. We

reflect on the achievement of our objectives, the implications of our work for educational environments, and the potential impact on the broader IoT landscape. Furthermore, we acknowledge the limitations of the present study and propose avenues for future research and development, underscoring the ongoing evolution of IoT platforms and their integration within educational contexts.

Chapter 2: IoT in the Context of Civilian Infrastructure

2.1 Introduction to Smart Cities

In an era characterized by rapid urbanization and technological advancement, the concept of smart cities has emerged as a pivotal solution to address the complex challenges of modern urban living. A smart city can be defined as an urban environment that harnesses innovative technologies and data-driven strategies to enhance the quality of life for its residents, optimize resource utilization, and foster sustainable economic development [5]. At its core, a smart city represents a transformative approach to urban planning, where connectivity, information, and intelligence converge.

Smart city initiatives are driven by a set of primary objectives that collectively aim to redefine urban living and elevate the quality of life for residents [6]. One fundamental objective is the pursuit of enhanced operational efficiency across various urban systems. This encompasses optimizing the management of transportation networks [7], waste disposal [8], energy distribution [9], and public services [10]. By harnessing technology and data-driven solutions, smart cities seek to streamline processes, reduce resource wastage, and minimize inefficiencies, leading to a more sustainable and cost-effective urban ecosystem. In an era of ambitious carbon reduction goals for the globe's largest cities, a pivotal objective revolves around sustainability. Smart cities prioritize the responsible use of resources, with a focus on reducing carbon emissions, conserving energy, and minimizing the environmental footprint [11]. By integrating sustainable practices into urban planning and infrastructure development, these initiatives contribute to a greener, healthier, and more resilient urban environment with the aim of benefitting both current and future generations.

Furthermore, the quest for an improved quality of life stands at the forefront of smart city endeavors. Through innovative approaches, smart cities aim to create urban spaces that offer seamless access to amenities, services, and opportunities [12]. This objective encompasses the use of convenient transportation options, efficient healthcare services, advanced educational facilities, and vibrant cultural and recreational spaces. By leveraging technology to enhance citizen well-being and satisfaction, smart cities aspire to foster a sense of community, belonging, and engagement among residents. Moreover, the pursuit of economic prosperity constitutes another vital dimension of smart city initiatives. By fostering an environment that welcomes innovation, entrepreneurship, and sustainable economic growth, smart cities aim to attract investments, create job opportunities, and nurture a vibrant economic ecosystem. These objectives collectively reflect the transformative potential of smart city initiatives to reshape urban living, empower residents, and promote holistic development across economic, social, and environmental dimensions.

The integration of technology and data within urban environments has emerged as a cornerstone of modern urban development, reshaping the way cities function and interact with their inhabitants. This integration holds immense importance as it enables cities to harness the power of real-time information and insights to enhance decision-making, resource management, and overall urban efficiency. By embedding sensors, IoT devices, and data analytics platforms into various infrastructural elements, cities gain the ability to monitor and collect data on diverse aspects such as traffic patterns, air quality, energy consumption, and waste management. This data-driven approach empowers city administrators to make

informed decisions, predict trends, and proactively address urban challenges.

The process of integrating technology and data within urban environments follows a strategic trajectory involving several key steps. It begins with the deployment of a network of sensors and IoT devices across the city's physical infrastructure, ranging from roads and buildings to utilities and public spaces. These devices collect real-time data on various parameters, generating vast streams of information. Subsequently, data analytics and machine learning algorithms come into play, processing and interpreting the collected data to extract meaningful insights. This analysis provides valuable information about urban trends, patterns, and potential issues, allowing city administrators to respond proactively. The data is often visualized through dashboards and user interfaces, enabling stakeholders to understand complex urban dynamics at a glance. Importantly, the integration of technology and data isn't limited to administrative applications; it extends to citizen engagement as well. Through open data initiatives such as the case study of Helsinki [13] and interactive platforms, residents are empowered to access and contribute data, fostering a participatory approach to urban planning and governance. Ultimately, the integration of technology and data drives innovation, efficiency, and sustainability in urban environments, enhancing quality of life and promoting more connected, intelligent, and responsive cities.

2.2 Key Components of Smart Cities

2.2.1 Infrastructure

At the heart of every successful smart city lies a robust and well-connected infrastructure that forms the backbone of its functionality. The importance of advanced infrastructure, spanning communication networks, energy systems, transportation, and more, cannot be overstated in the context of smart cities [14]. These components collectively facilitate the seamless flow of information, services, and resources, laying the foundation for efficient urban operations and enhanced quality of life for residents.

Communication Networks for Real-Time Connectivity

Advanced communication networks constitute a fundamental component of modern smart cities. The newest generations of mobile networking such as 5G and 6G enable real-time connectivity between devices, sensors, and citizens. An illustrative example of a smart city network driven by the newest communication architectures can be seen in Figure 2.1 [15]. This connectivity is crucial for the exchange of data and information that drives smart city applications. Citizens can access services, monitor resources, and engage with urban platforms in real time, enhancing convenience and accessibility. Moreover, robust communication networks support the establishment of smart grids and efficient energy distribution systems, facilitating real-time monitoring and control of energy consumption.

Energy Systems for Sustainability

Sustainable energy systems play a pivotal role in the development of smart cities. Efficient energy generation, distribution, and consumption are essential for reducing carbon footprints and ensuring the

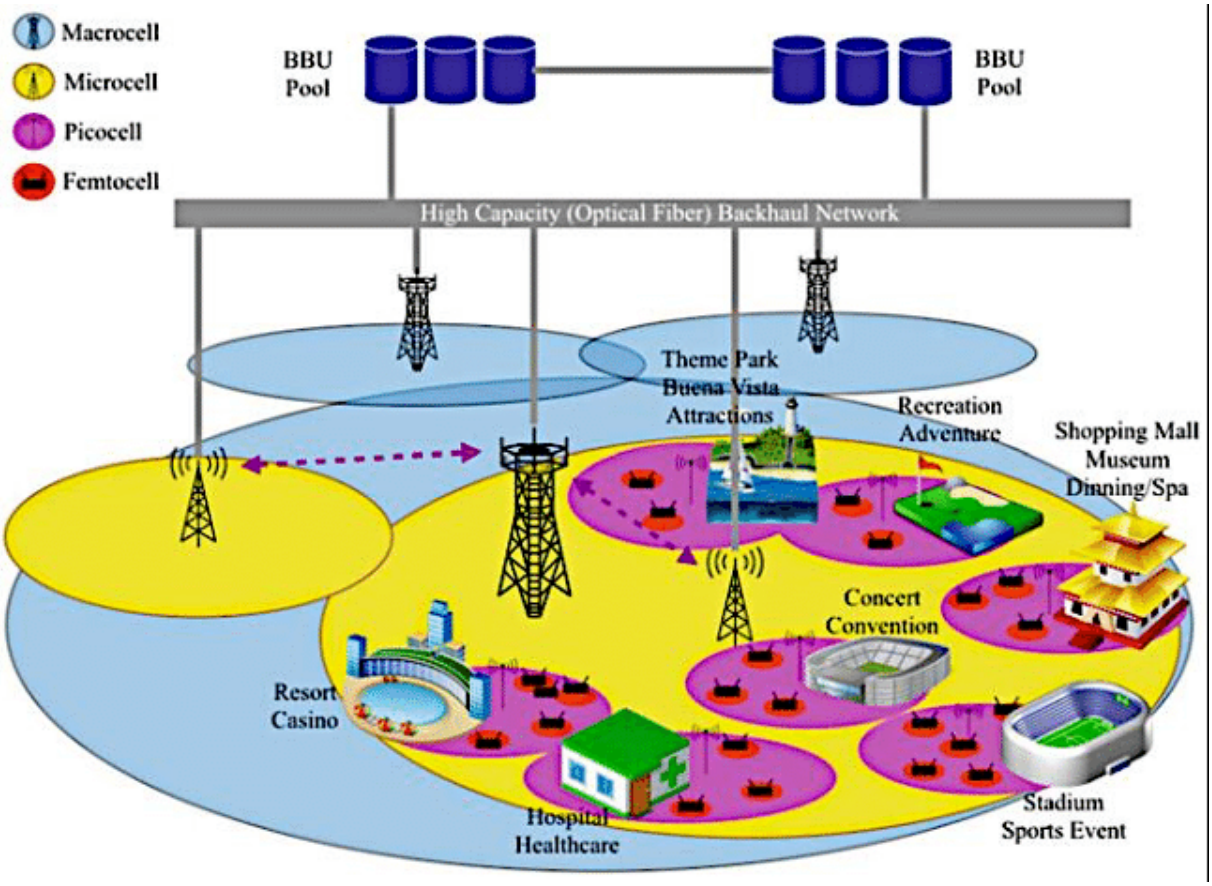

Figure 2.1: Smart City Communication Infrastructure powered by 5G, Source: [15]

long-term viability of urban areas. Smart cities leverage renewable energy sources, microgrids, and energy-efficient technologies to optimize energy consumption and minimize waste (Figure 2.2, [16]).

By integrating smart meters and sensors, cities can monitor energy usage patterns and implement demand-response strategies to balance supply and demand. This not only reduces energy costs but also contributes to environmental conservation and resilience against energy shortages.

Transportation Infrastructure for Mobility

Advanced transportation systems are integral to smart cities, as they enhance mobility, reduce congestion, and lower pollution levels. Smart transportation solutions, such as intelligent traffic management, real-time transit information, and shared mobility services, promote efficient urban mobility. By utilizing data from sensors and GPS devices and interconnecting vehicles together along with the urban network (Figure 2.3 [17]), cities can analyze traffic patterns and optimize traffic flow, leading to reduced travel times and improved air quality. Furthermore, the integration of electric vehicles, smart parking systems, and bike-sharing initiatives aligns with sustainability objectives, contributing to a cleaner and more accessible urban environment.

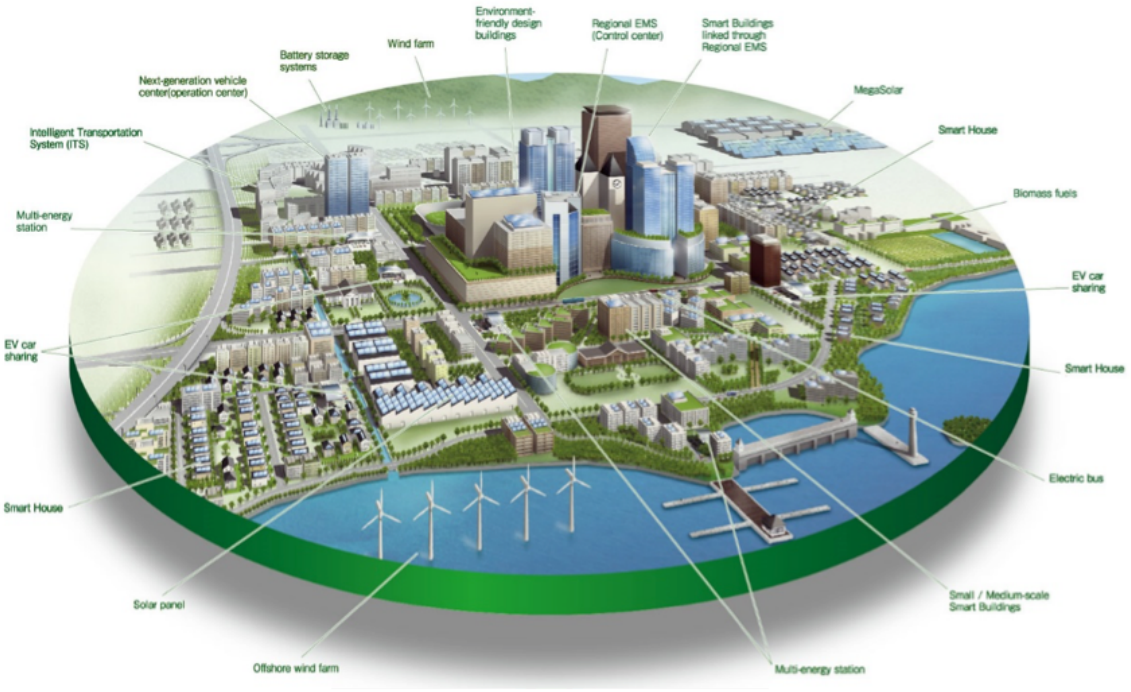

Figure 2.2: Energy Infrastructure for Smart Cities, Source: [16]

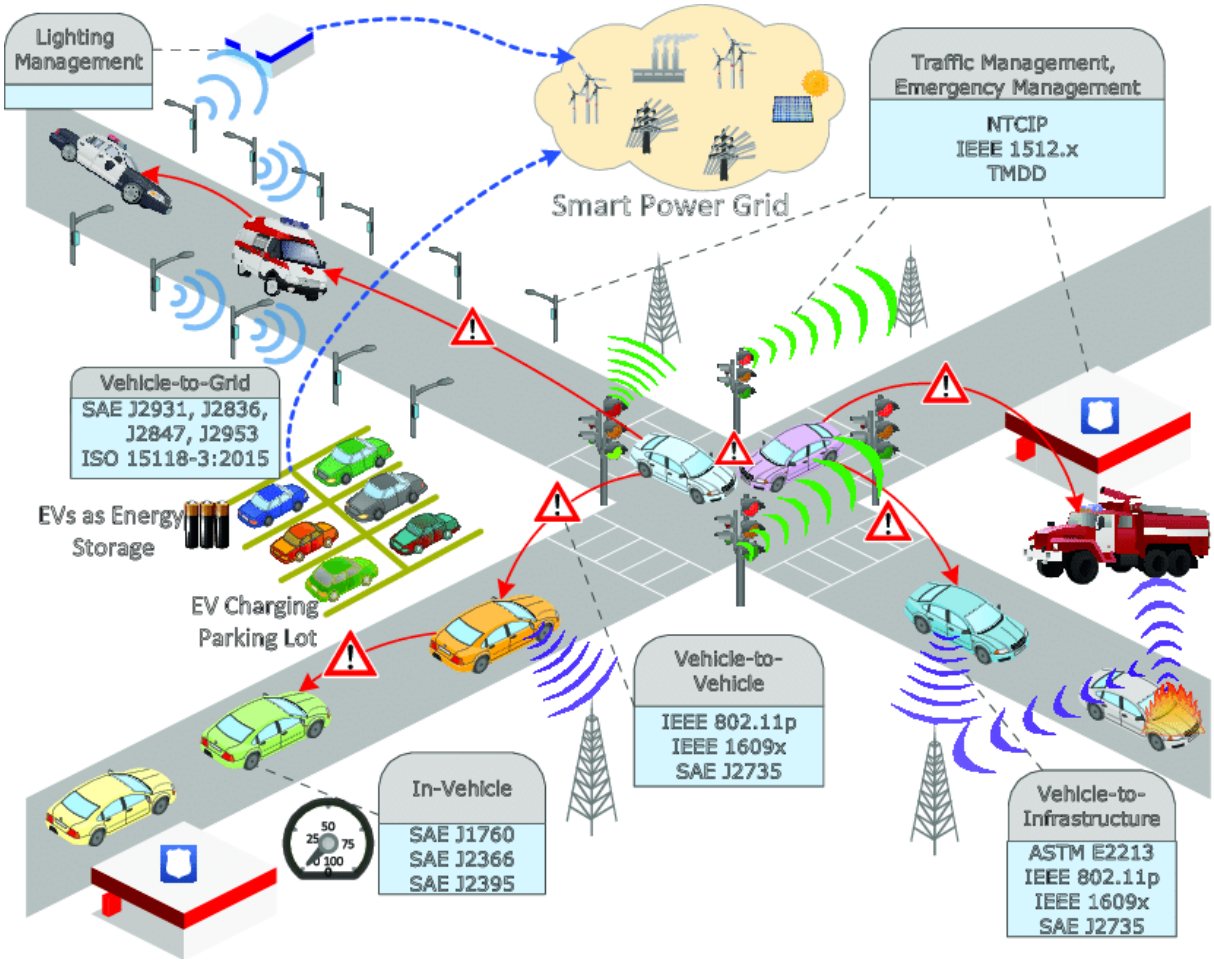

Figure 2.3: Smart City Transportation, Source: [17]

2.2.2 Data Collection

In the dynamic landscape of smart cities, data collection emerges as a pivotal component that empowers informed decision-making and drives the efficiency of urban systems. At the heart of this process are sensors and IoT devices strategically embedded throughout the city's infrastructure. These devices serve as the eyes and ears of the city, continuously gathering real-time data from various sources to provide a comprehensive understanding of urban dynamics.

Sensors

Sensors play a transformative role in the data collection ecosystem of smart cities. These small and intelligent devices are deployed across urban environments, ranging from streetlights and waste bins to buildings and public spaces. They capture a multitude of data types, including environmental variables like temperature, humidity, air quality, and noise levels. Furthermore, sensors monitor pedestrian and vehicular movement, traffic flow, and energy consumption. This wealth of real-time data enables city officials to respond promptly to events, allocate resources efficiently, and enhance the overall urban experience.

IoT Devices

IoT devices extend the capabilities of sensors by enabling interconnected intelligence and a higher level of data processing. These devices are equipped with embedded processors and communication modules that allow them to collect, transmit, and receive data. They create a network of interconnected elements, enabling the seamless exchange of information across diverse applications. For instance, in the context of transportation, IoT-enabled vehicles and infrastructure components can communicate with traffic management systems to optimize traffic flow and mitigate congestion. Similarly, in waste management, IoT-equipped bins can signal when they are full, optimizing waste collection routes and reducing operational costs.

Data Collection Mechanisms

Data collected from sensors and IoT devices are transmitted through a variety of mechanisms. Cloud computing and edge computing play crucial roles in managing and processing this data. Cloud platforms provide storage and computational capabilities, allowing city officials to aggregate and analyze large datasets for strategic decision-making. Edge computing, on the other hand, involves processing data closer to the source, reducing latency and enabling real-time responsiveness. This is particularly valuable in scenarios where immediate action is required, such as traffic signal optimization or emergency response systems.

2.3 Benefits of Smart Cities

2.3.1 Sustainaibility

One of the primary benefits of smart cities is their ability to optimize resource consumption through data-driven insights and real-time monitoring. By deploying sensors and IoT devices throughout the urban landscape, cities gain unprecedented visibility into resource flows, such as energy usage, water consumption, and waste generation. This data is then analyzed to identify inefficiencies and areas for improvement. For instance, smart grids enable the efficient distribution of electricity by adjusting power delivery based on real-time demand, reducing energy wastage. Similarly, smart water management systems detect leaks and regulate water usage, minimizing water loss and conserving a precious resource.

The advantages of technology and sustainability in smart cities extends beyond resource management to encompass broader environmental concerns. By optimizing transportation systems, including public transit and shared mobility, smart cities promote reduced reliance on personal vehicles, mitigating air pollution and greenhouse gas emissions. Additionally, data-driven waste management enables efficient collection and disposal, minimizing landfill usage and promoting recycling. These collective efforts lead to a reduced environmental footprint, paving the way for cleaner air, healthier ecosystems, and a more resilient urban environment.

2.3.2 Efficiency

At the core of the smart city paradigm lies a profound commitment to efficiency. Efficiency in smart cities extends beyond the realm of technology; it encompasses a strategic approach to resource allocation. Through data-driven insights and predictive analytics, urban planners can allocate resources in a targeted manner, responding to actual demand rather than relying on historical patterns. For instance, smart traffic management systems dynamically adjust signal timings based on real-time traffic conditions, reducing congestion and minimizing travel time. Similarly, energy consumption can be optimized by adjusting lighting as well as Heating, Ventilation, and Air Conditioning (HVAC) systems based on occupancy patterns detected by sensors, leading to significant energy savings.

The advent of intelligent systems enables urban services to operate with unprecedented precision and responsiveness. Public transportation systems equipped with real-time tracking and predictive modeling optimize routes, minimize waiting times, and enhance the overall commuting experience. Waste collection is also revolutionized as sensors detect fill levels in bins, enabling optimized collection routes and reducing unnecessary pickups. In healthcare, telemedicine and smart clinics facilitate remote patient monitoring, early disease detection, and efficient allocation of medical resources.

2.3.3 Quality of life

The concept of quality of life takes center stage in smart city initiatives, with a holistic approach that aims to enhance various aspects of urban life, from accessible services to improved urban planning.

One of the hallmarks of smart cities is their ability to deliver enhanced public services that cater to the diverse needs of their residents. By leveraging technology, data, and connectivity, smart cities create

a responsive ecosystem that fosters convenience and accessibility. For instance, smart transportation systems offer real-time updates on bus and train schedules, reducing waiting times and making commutes more efficient. Access to e-governance platforms simplifies administrative processes, allowing citizens to access government services, pay bills, and submit applications online. Moreover, smart healthcare systems offer telemedicine services, ensuring timely medical consultations and reducing the strain on physical healthcare facilities.

Smart cities also prioritize urban planning strategies that cater to the well-being of their residents. By leveraging data analytics and simulations, urban planners can make informed decisions that optimize land use, green spaces, and public amenities. Real-time monitoring of air quality, noise levels, and environmental factors enables prompt interventions to maintain a healthier urban environment. Inclusive urban planning considers the needs of all citizens, ensuring that public spaces are accessible, safe, and conducive to social interaction.

2.3.4 Economic Growth

Smart cities not only enhance the quality of life for their citizens but also hold significant potential for fostering economic growth and development. The integration of technology, data-driven decision-making, and sustainable practices creates an environment conducive to innovation, entrepreneurship, and the attraction of investments. The economic benefits of smart cities extend beyond local markets, positioning them as hubs for technology-driven economic growth on a global scale.

Startups and tech companies find smart cities attractive due to the availability of infrastructure, data resources, and a skilled workforce. The collaboration between academia, research institutions, and industries further accelerates technological advancements, resulting in a thriving innovation ecosystem.

Additionally, deployment of smart technologies and infrastructure in cities has a direct impact on attracting investments from both domestic and international sources. Investors recognize the potential for returns on investments in areas such as smart energy grids, intelligent transportation systems, and digital infrastructure. The implementation of smart solutions not only enhances the efficiency of urban operations but also opens up new avenues for revenue generation. Public-Private Partnerships (PPPs) often play a crucial role in funding and executing smart city projects, creating a synergy between government initiatives and private sector expertise.

Smart cities position themselves as globally competitive hubs that can attract talent, companies, and investment. The creation of innovation districts, technology parks, and incubators fosters an ecosystem where startups and established companies can thrive. As these cities embrace sustainable practices, efficient resource management, and advanced technological solutions, they become models for urban development that other cities aspire to replicate. The reputation of being a smart city can lead to economic collaborations, partnerships, and exchanges with other cities, further fueling economic growth.

2.4 Challenges and Considerations

The transition towards smarter urban environments presents opportunities to enhance efficiency, sustainability, and quality of life for citizens. However, this transformation is not without its complexities, requiring a comprehensive understanding of the potential hurdles that need to be navigated. This section delves into the multifaceted challenges that smart cities encounter, ranging from data privacy and security concerns to the intricacies of managing urban infrastructure. By exploring these challenges, it becomes evident that building and sustaining smart cities necessitates a holistic approach that balances technological advancements with ethical, legal, and societal dimensions.

2.4.1 Data Privacy and Security

As smart cities harness the power of data to enhance urban life, a critical challenge arises in ensuring the privacy and security of the vast amount of data collected from citizens and various urban systems. The seamless integration of technology and data in smart cities raises concerns about the potential misuse, unauthorized access, and potential breaches of sensitive information [18]. The collection of data from sensors, IoT devices, and urban infrastructure has the potential to reveal intricate details of citizens' daily lives, leading to significant privacy implications.

One of the foremost challenges in smart cities is striking a balance between the utility of data for improving urban services and safeguarding citizens' privacy rights. The extensive collection of data, including location information, personal habits, and preferences, can create profiles that might be exploited for targeted advertising, surveillance, or other unauthorized purposes. Ensuring that data is collected with consent, anonymized where possible, and used only for legitimate purposes is vital to prevent the erosion of trust between citizens and the city administration.

The interconnected nature of smart city systems exposes them to cybersecurity risks that can have far-reaching consequences [19]. A breach in a single component of the urban infrastructure can potentially compromise the entire ecosystem. Smart city initiatives involve a wide array of data sources, from traffic cameras and environmental sensors to health records and financial transactions. Ensuring the security of this diverse data landscape requires robust encryption, intrusion detection systems, and continuous monitoring to detect and mitigate potential cyber threats.

2.4.2 Interoperability

Smart cities encompass a diverse array of systems, ranging from transportation networks and energy grids to public safety and healthcare services. These systems are often developed and maintained by different vendors, public and private entities, and governmental bodies. The challenge arises when these systems need to collaborate seamlessly to deliver holistic benefits to citizens [20]. Interoperability gaps can result in inefficiencies, data silos, and missed opportunities to optimize urban operations.

To address the interoperability challenge, smart cities must strive for standardized communication protocols, data formats, and Application Programming Interfaces (APIs). Achieving interoperability requires collaborative efforts from various stakeholders, including technology providers, policymakers,

and industry consortia. By adopting open standards, cities can ensure that different systems can communicate effectively, enabling data sharing and integration. Thus, the creation of methodologies to converge to this goal is of vital importance, and efforts such as the SEG 3.0 methodology proposition are paramount to the success of interconnected urban undertakings [21]. Standardization not only enhances the technical compatibility of systems but also fosters innovation by enabling developers to build upon existing platforms and solutions.

Interoperability is also closely tied to data integration, where information from various sources is aggregated for meaningful insights. For instance, integrating data from traffic sensors, weather forecasts, and public transportation schedules can facilitate efficient urban mobility solutions. Data integration paves the way for advanced analytics, predictive modeling, and real-time decision-making, ultimately contributing to the effective management of urban resources and services.

2.4.3 Governance and Regulation

Smart city governance must encompass policies that prioritize ethical use and respect for individual rights. This includes guidelines for data collection, storage, and sharing, as well as mechanisms for transparency and informed consent. Additionally, a well-defined regulatory framework can empower citizens to participate in decision-making processes and hold stakeholders accountable. When citizens have a voice in the policies that shape their city, it fosters a sense of ownership and participation in the smart city journey.

Given the complex nature of smart cities, governance and regulation require collaboration across various domains and expertise, as well as new breakthroughs in urban policy management [22]. Policymakers, urban planners, technologists, legal experts, and citizens must work together to formulate comprehensive policies that address the multifaceted challenges posed by emerging technologies. A multidisciplinary approach ensures that regulations are adaptable, forward-looking, and capable of keeping pace with the rapid evolution of the smart city landscape.

2.4.4 Inclusivity

Smart city initiatives have the potential to amplify existing social and economic disparities if not approached with a lens of inclusivity. The digital divide can manifest in multiple ways: limited access to high-speed internet, lack of digital skills, and unequal distribution of technology resources across neighborhoods. For a smart city to be truly smart, it must strive to bridge these gaps, ensuring that technology benefits are not limited to certain segments of the population [23].

Inclusive smart cities prioritize equitable access to information, services, and opportunities. This encompasses efforts to provide affordable and widespread internet connectivity, digital literacy programs, and technology infrastructure that reaches underserved communities. By ensuring that citizens from all walks of life can access essential services, engage in digital transactions, and participate in the smart city ecosystem, cities can create a more cohesive and resilient urban environment.

Moreover,in order to address the challenge of inclusivity, smart cities require collaborative efforts involving governments, technology providers, community organizations, and educational institutions.

Initiatives that focus on digital education and skills training can empower citizens to navigate the digital landscape, enabling them to make informed decisions and leverage the benefits of smart technologies. Additionally, co-designing solutions with the input of diverse communities ensures that smart city technologies reflect the needs and aspirations of the entire population.

Chapter 3: Smart Campus: Extending Smart City Concepts to Educational Environments

3.1 Introduction

In the pursuit of harnessing technological advancements to optimize urban living, the concept of the smart city has garnered significant attention. As cities transform into interconnected ecosystems powered by data and technology, there arises a natural extension of this idea: the smart campus. A smart campus embodies the principles of smart cities but tailors them to the unique environment of educational institutions, bringing forth a new era of enhanced learning experiences, efficient resource management, and streamlined campus operations.

A smart campus can be understood as an educational environment that embraces the core principles of smart cities, adapting them to the specific needs and dynamics of academic institutions. Just as smart cities integrate technology and data to improve the quality of urban life, a smart campus leverages the power of connectivity, automation, and intelligent systems to elevate the educational experience for students, faculty, and staff. This concept represents a harmonious convergence of innovative technologies such as IoT and educational approaches, creating an environment that fosters learning, innovation, and sustainability.

3.2 Components of a Smart Campus

3.2.1 Smart Infrastructure

The concept of a smart campus relies heavily on the establishment of a sophisticated and interconnected infrastructure that encompasses advanced communication networks, energy systems, and modern transportation solutions. This smart infrastructure lays the groundwork for seamless connectivity, efficient resource utilization, and the integration of cutting-edge technologies. This section will enumerate the key elements which comprise a successful smart infrastructure.

Advanced Communication Networks

One of the cornerstone components of a smart campus's infrastructure is the establishment of advanced communication networks. These networks connect various devices, systems, and stakeholders across the campus, fostering seamless communication, data exchange, and collaboration. Unlike conventional networks, which focused primarily on basic connectivity, advanced communication networks within a smart campus are designed to handle the diverse and growing demands of the modern educational landscape.

These communication networks play a pivotal role in facilitating the integration of IoT devices and sensors across the campus. Smart devices, ranging from sensors that monitor environmental conditions to wearables that track students' well-being, rely on specific communication protocols designed for IoT applications.

The range of communication protocols which can be chosen is vast and heaviliy depends on the needs of each deployment. For instance, Low Power Wide Area Network (LPWAN) [24] technologies like LoRaWAN enable long-range communication with low power consumption [25], ideal for transmitting data from remote sensors across a smart campus. These protocols ensure that even battery-operated sensors can transmit data reliably over long distance, contributing to a comprehensive data collection.

Moreover, within the realm of IoT communications, the publish-subscribe (pub/sub) architecture emerges as particularly well-suited for the intricate web of interactions within a smart campus. This architecture allows devices and sensors to communicate without direct, point-to-point connections, enabling one-to-muany and many-to-one communication patterns. This flexibility is integral to the smart campus's dynamic environment, where various devices need to share data with multiple endpoints and respond to real-time events. MQTT (Message Queuing Telemetry Transport), an exemplar of the pub/sub model, exemplifies this efficiency by enabling devices to publish data to specific topics while other devices subscribe to these topics, ensuring that information is relayed in a streamlined manner. This architectural approach facilitates data sharing among devices, supports event-driven applications, and enhances the responsiveness of the entire smart campus ecosystem. Figure 3.1 showcases an example of a proposed smart campus networking architecture combining the MQTT and LoRa-WAN protocols [26].

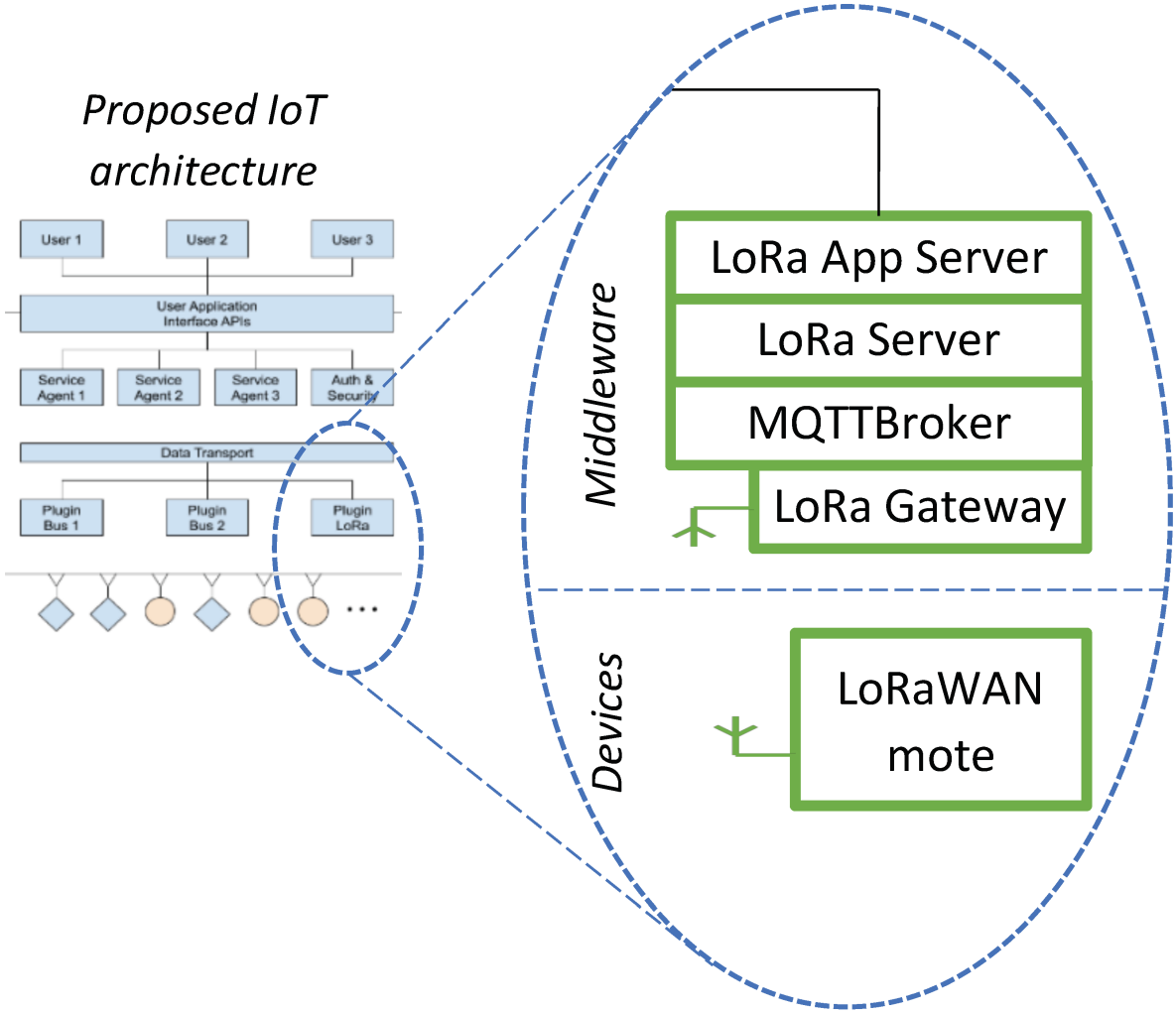

Figure 3.1: Proposed LoRaWAN - MQTT IoT architecture, Source: [26]

Furthermore, protocols like Zigbee [27] and Bluetooth Low Energy (BLE) [28] enable connectivity and communication among IoT devices in proximity, contributing to indoor navigation systems, interactive displays, and even personal health monitoring applications. The integration of these specialized communication protocols into the advanced networks of a smart campus extends beyond traditional connectivity, enabling the seamless interaction of smart devices and enhancing the overall campus experience.

Lastly, as the landscape of communication technologies continues to evolve, the advent of 5G and the emerging promise of 6G bring new dimensions to the connectivity of smart campuses. 5G's high data rates, ultra-low latency, and massive device connectivity potential offer unprecedented opportunities for real-time data exchange, enabling intricate applications such as augmented reality-enhanced learning experiences and remote-controlled IoT devices. The anticipated advancements of 6G, building upon 5G's foundation, have the potential to unlock even more possibilities, including enhanced network resilience, higher energy efficiency, and ubiquitous connectivity. The seamless integration of 5G and 6G technologies with established communication protocols such as MQTT, Zigbee, and LoRaWAN can create a robust communication framework that caters to the diverse and evolving needs of a smart campus, ensuring an interconnected and responsive environment that empowers both learning and innovation.

Energy Systems for Sustainability and Efficiency

Similarly to smart cities, a smart campus incorporates intelligent energy systems that optimize resource consumption and enhance sustainability. Energy-efficient technologies such as smart lighting and HVAC systems, use sensors and automation to adjust usage based on occupancy and environmental conditions. These systems not only reduce energy waste but also contribute to cost savings and a lower environmental footprint.

Smart lighting solutions play a pivotal role in energy efficiency and environmental sustainability. By integrating occupancy sensors and daylight harvesting technology, lighting systems can intelligently adjust illumination levels according to the presence of people and available natural light. Furthermore, these systems can be remotely managed and scheduled to minimize energy consumption during non-peak hours. An example of such a system is the use of motion sensors to activate lighting only when required, ensuring that energy is not wasted in unoccupied spaces.

Intelligent HVAC systems are another critical component of energy-efficient campuses [29]. Smart HVAC systems utilize data from occupancy sensors, temperature sensors, and weather forecasts to dynamically regulate indoor climate conditions. These systems can automatically adjust heating, cooling, and ventilation based on real-time demand, leading to reduced energy consumption while maintaining a comfortable environment for occupants. Predictive analytics also optimize energy usage by learning from historical data and patterns. Renewable energy integration is embraced by smart campuses to supplement their energy needs [30]. Renewable sources like solar panels and wind turbines contribute to sustainability and potential energy cost savings. Advanced energy management systems monitor renewable energy generation and consumption patterns, ensuring efficient utilization and optimal distribution of clean energy across campus facilities. An illustration of this concept can be viewed in Figure 3.2.

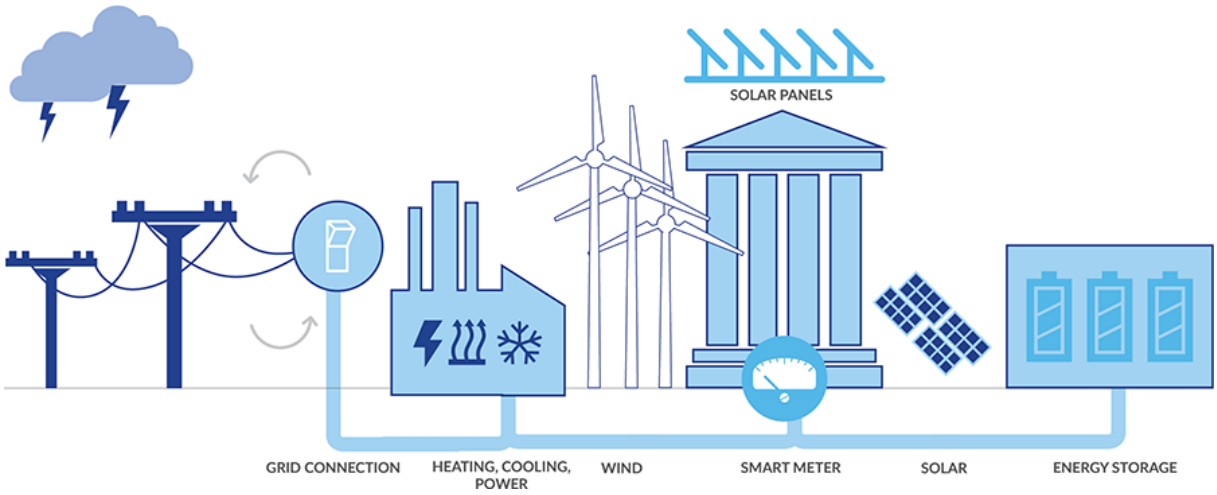

Figure 3.2: Smart Campus Energy Framework (source: Anbaric)

Energy monitoring and analytics offer data-driven insights essential for effective energy management. Integration of energy monitoring devices and analytics platforms provides real-time visibility into energy consumption patterns, identifying inefficiencies and implementing targeted strategies for improvement. Dashboards and reports provide stakeholders valuable information to make informed decisions about energy usage and conservation initiatives.

In conclusion, the integration of intelligent energy systems within a smart campus is instrumental in achieving sustainability goals and optimizing resource utilization. The combination of smart lighting, HVAC systems, renewable energy sources, and energy monitoring mechanisms collectively contributes to reduced energy waste, cost savings, and an environment aligned with environmental stewardship principles. Such systems underscore the commitment of educational institutions to address energy challenges while providing an enhanced learning experience for students and staff.

Intelligent Building Management Systems

In the realm of smart campuses, the implementation of intelligent building management systems stands as a pivotal component that not only streamlines operations but also contributes to significant enhancements in overall efficiency. These systems empower campuses to monitor and regulate crucial factors like lighting, temperature, and occupancy, thus revolutionizing the way campus environments are managed.

The essence of intelligent building management systems lies in their ability to respond to real-time data inputs and orchestrate actions that align with the evolving needs of the campus. There is a plethora of examples of this type of solutions in the smart campus corpus [31] [32], These include temperature control systems which can intelligently regulate HVAC units, adapting to occupancy patterns and external weather conditions. This dynamic responsiveness translates into tangible benefits – ranging from energy conservation to cost reduction – while simultaneously fostering a conducive atmosphere for learning, work, and collaboration.

Furthermore, the integration of occupancy sensors and other IoT devices into building management

systems enables granular insights into space utilization. This data-driven approach not only aids in optimizing room allocations but also assists in the design of future campus expansions and facility improvements. By harnessing the power of data analytics, campuses can identify usage trends, peak hours, and underutilized spaces, enabling informed decisions that lead to a more efficient allocation of resources.

Intelligent building management systems hold the promise of enhancing occupant experiences as well. Through continuous monitoring and proactive adjustments, these systems ensure that spaces are maintained at optimal conditions, fostering a productive and comfortable environment for students, faculty, and staff. The dynamic nature of these systems also facilitates predictive maintenance, preemptively identifying issues before they escalate into disruptions, thereby minimizing downtime and service interruptions.

In summary, the deployment of intelligent building management systems within smart campuses offers multifaceted advantages. These systems epitomize the synergy between technological innovation and operational efficiency, transforming conventional campuses into agile and responsive environments. By integrating real-time data analytics, automation, and sensor-driven insights, campuses can achieve sustainable resource utilization, energy savings, and a heightened quality of campus life, ultimately enriching the educational experience for all stakeholders involved.

3.2.2 IoT and Sensor Networks

At the heart of the smart campus lies the intricate web of IoT devices, each acting as a data point contributing to a comprehensive understanding of the campus environment. These devices encompass a diverse range of sensors, from motion and occupancy detectors to environmental monitors, enabling real-time data collection across various facets of campus life. By interconnecting these devices, a sophisticated network is established, facilitating seamless communication and enabling comprehensive insights into campus operations.

Sensor networks stand as the backbone of a smart campus, propelling the collection and transmission of valuable data streams. These networks encompass clusters of interconnected sensors strategically placed throughout the campus landscape. For instance, environmental sensors track factors such as temperature, humidity, and air quality, providing critical data for maintaining optimal learning and working conditions. Occupancy sensors contribute insights into space utilization, enabling informed decisions for resource allocation and optimization.

Complementing the role of sensors, actuators play a crucial part in the functionality of a smart campus. Actuators are devices capable of performing physical actions based on the data received from sensors and the decisions made by the system. For example, actuators can adjust lighting levels in response to occupancy data, optimize heating or cooling systems for energy efficiency, and control access to different areas of the campus. By translating data into tangible actions, actuators bring the concept of a responsive and adaptable campus environment to life.

3.2.3 Data Management and Analytics

In the realm of smart campuses, the incredible amount of data generated by sensors, devices, and interconnected systems serves as a treasure trove of insights waiting to be unearthed. The section on Data Management and Analytics explores how the strategic management, processing, and analysis of this data drive the intelligence of a smart campus, facilitating informed decision-making and optimizing various aspects of campus life.

Central to the data management strategy of a smart campus is a robust infrastructure for data storage and organization. Data repositories, often in the form of databases or cloud-based solutions, house the vast array of collected information. By structuring data in a coherent and accessible manner, campus administrators and stakeholders can readily access historical and real-time data streams. This organized repository serves as a home for analytics, fostering a data-driven culture within the campus environment. The true value of data emerges when it is harnessed to uncover patterns, trends, and correlations that would otherwise remain concealed. Advanced analytics tools, including machine learning algorithms and data visualization platforms, sift through data to generate actionable insights. For instance, analytics can identify energy consumption patterns to optimize usage and reduce waste, predict maintenance needs for campus facilities, and highlight student engagement trends for tailored educational approaches.

Smart campuses leverage real-time data monitoring and analysis to empower stakeholders with up-to-the-minute information. Facilities managers can oversee energy consumption and environmental conditions, responding swiftly to anomalies or inefficiencies. Educators can track student engagement metrics, adjusting teaching strategies as needed. Students themselves can access information about available resources, class schedules, and campus events, enhancing their overall experience.

In summary, the Data Management and Analytics component forms the cognitive engine of a smart campus, transforming raw data into actionable insights. By strategically collecting, organizing, and analyzing data, campuses can make informed decisions, optimize resources, and enhance the overall campus experience. Through a synergy of technology and intelligent decision-making, data management and analytics pave the way for an agile and responsive campus environment that adapts to the evolving needs of its community.

3.3 Applications and Benefits

This section delves into the diverse array of smart campus applications and their associated benefits. Figure 3.3 illustrates an overview of the wide spectrum of intelligent campus solutions. From energy management systems that promote sustainability to bolstered security measures ensuring a safe environment, personalized learning experiences driven by data analytics, and streamlined facility management processes optimizing resource utilization, we explore how these innovations are reshaping the modern educational landscape, aligning institutions with the digital era's demands, and ultimately enriching the educational journey for students and educators alike.

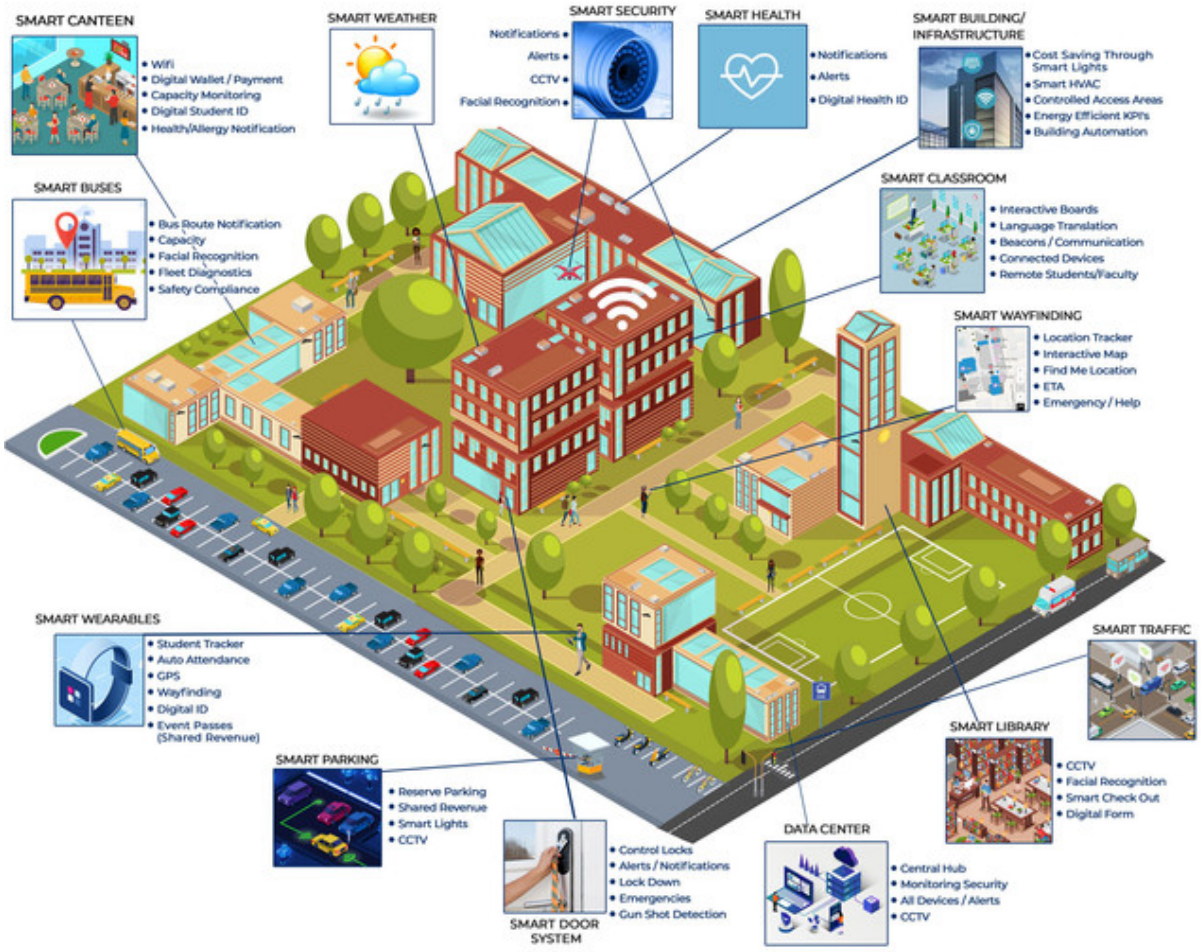

Figure 3.3: Smart Campus Applications (source: clovity.com)

3.3.1 Smart Energy Management

The implementation of smart energy management strategies within a smart campus represents a significant leap towards achieving environmental sustainability and operational efficiency. Smart campuses harness data-driven insights to intelligently monitor, control, and optimize energy consumption across campus facilities, minimizing waste and reducing the institution's carbon footprint.

Smart energy management systems leverage real-time data to implement demand response strategies. During periods of peak energy demand or grid instability, the system can automatically adjust energy-intensive processes, such as HVAC systems or lighting, to reduce consumption and support grid stability. Load shifting further optimizes energy usage by redistributing consumption across time periods, taking advantage of off-peak rates and minimizing strain on the grid.

Automation plays a pivotal role in reducing energy waste by ensuring that systems operate at optimal efficiency levels. Smart lighting systems equipped with occupancy sensors adjust illumination based on room occupancy, significantly reducing energy consumption in unoccupied areas. Likewise, HVAC systems adapt heating and cooling based on occupancy and environmental conditions, maintaining comfort while minimizing unnecessary energy use.

Predictive analytics offer a proactive approach to maintenance, enabling the identification of equipment anomalies and inefficiencies before they escalate. By continuously monitoring the performance of energy-intensive equipment, such as boilers and chillers, predictive algorithms can forecast potential malfunctions. This enables timely maintenance interventions, preventing costly breakdowns and optimizing equipment lifespan.

Smart energy management extends beyond technology to encompass the engagement of the campus community. Real-time energy usage data can be visualized and shared with occupants through user-friendly interfaces and applications. By fostering awareness of individual energy consumption and its impact, users are motivated to adopt energy-conscious behaviors, contributing to overall energy reduction efforts.

The adoption of smart energy management systems offers multifaceted benefits to a smart campus. These include significant reductions in energy consumption, leading to cost savings on utility bills. Additionally, optimized energy usage contributes to a smaller environmental footprint, aligning with the institution's sustainability goals and demonstrating social responsibility.

Furthermore, the implementation of these systems enhances the reliability of campus energy infrastructure by preventing equipment failures through predictive maintenance. This minimizes disruptions to academic and administrative operations and extends the lifespan of critical assets.

3.3.2 Enhanced Security

In the dynamic environment of a smart campus, ensuring the safety and security of students, faculty, and facilities is of paramount importance. The integrated platform plays a pivotal role in enhancing campus security through advanced surveillance, access control mechanisms, and incident detection. By harnessing the power of IoT devices, data analytics, and real-time monitoring, the smart campus platform provides a comprehensive security solution that goes beyond traditional methods.

Surveillance for Proactive Monitoring

The platform's surveillance capabilities enable the deployment of a network of smart cameras and sensors strategically placed across the campus [33]. These devices continuously monitor various areas, providing real-time video feeds and data streams. Through intelligent algorithms and pattern recognition, the system can automatically detect unusual activities, unauthorized access, and potential security threats. This proactive monitoring approach not only aids in preventing security breaches but also allows security personnel to respond swiftly to emerging situations.

Access Control

Access control is a fundamental aspect of campus security, and the smart platform takes it to a new level of sophistication. With IoT-enabled access points, such as smart card readers and biometric scanners, access to different areas of the campus can be tightly regulated [34]. The platform's centralized management

system grants administrators the ability to define access permissions based on roles, time schedules, and other factors. This granular control not only enhances security but also enables seamless integration with other campus functions, such as attendance tracking and resource utilization.

Incident Detection and Response

In the event of an incident, whether it's a security breach or an emergency situation, the platform's real-time data analytics and alerts system come into play. Anomalous patterns detected by sensors, combined with data from other sources, can trigger instant alerts to security personnel and relevant authorities. This timely information empowers rapid response, enabling the campus to mitigate risks and address situations swiftly. Moreover, historical data collected by the platform allows for post-incident analysis and refinement of security protocols.

3.3.3 Personalized Learning

In the realm of modern education, the concept of personalized learning has gained significant traction as institutions seek to provide tailored educational experiences that cater to the unique needs and preferences of each student. The smart campus platform serves as a catalyst for realizing this vision by harnessing the power of data analytics, behavioral insights, and intelligent recommendations. Through the analysis of student behaviors and preferences, the platform enables the creation of highly personalized learning pathways that enhance engagement, retention, and overall educational outcomes.

Central to the platform's ability to facilitate personalized learning is its capacity to collect and analyze data related to student behaviors. IoT sensors deployed throughout campus facilities gather information on factors such as student attendance patterns, activity participation, and resource utilization. By applying data analytics and machine learning algorithms to this wealth of information, the platform generates insights into individual learning styles, achievement predicition, study habits, and engagement levels [35]. This holistic understanding of student behavior forms the basis for tailoring educational approaches.

The platform leverages the behavioral insights obtained to create customized learning pathways for each student. Through intelligent recommendation engines, students are guided towards resources, courses, and learning materials that align with their learning preferences and strengths. For instance, a student who demonstrates a preference for visual learning might be directed towards multimedia-rich content, while a student inclined towards collaborative work could be suggested group projects and activities. This individualized approach not only enhances learning efficacy but also fosters a sense of ownership and empowerment in students' educational journeys.

One of the platform's unique strengths lies in its real-time feedback loop. As students engage with educational content, their interaction is continuously monitored and analyzed. This allows educators gaining instant insights into students' progress, challenges, and areas of mastery. Instructors can then adjust their teaching methods and interventions to cater to individual needs promptly. The platform's dynamic nature ensures that personalized learning experiences are adaptive and responsive to students' changing requirements.

3.3.4 Facility Management

Effective management of campus facilities is a cornerstone of maintaining a conducive and productive learning environment. The smart campus platform, with its robust data collection and analysis capabilities, revolutionizes facility management by providing actionable insights that optimize maintenance operations, cleaning routines, and space utilization strategies. Through real-time monitoring and informed decision-making, the platform enhances the overall quality of campus infrastructure while fostering a seamless experience for both students and staff.

Traditional facility management often involves reactive maintenance practices that address issues only after they become apparent. The smart campus platform shifts this paradigm by utilizing IoT sensors to continuously monitor the condition of various infrastructure components such as HVAC systems [29] and lighting [31]. By analyzing sensor data, the platform can predict maintenance needs well in advance, enabling maintenance teams to proactively address potential issues before they escalate. This approach minimizes disruptions, extends the lifespan of equipment, and reduces overall maintenance costs.

Maintaining a clean and hygienic campus environment is essential for the well-being of the campus community. The platform enhances cleaning operations by offering real-time insights into foot traffic and occupancy patterns within different areas. This data-driven approach enables cleaning staff to focus their efforts on high-traffic zones and adjust their schedules to align with peak activity times [36]. As a result, campus spaces remain clean and welcoming without unnecessary resource allocation.

Effective utilization of campus spaces is vital for creating an environment that maximizes functionality and accommodates various activities. The platform's data analytics capabilities enable accurate tracking of space utilization trends, including room occupancy, usage patterns, and peak demand periods. With this information, administrators can optimize the allocation of rooms, labs, and study spaces, minimizing conflicts and ensuring efficient use of available resources [37]. Additionally, data-driven insights can guide decisions about facility expansions or redesigns to align with actual usage.

3.4 Integration with Educational Goals

In our pursuit of smart campus excellence, it's essential to underscore how our platform, OpenSCN, not only aligns with educational goals but also actively encourages student participation and innovation. While we've already explored the significant impact of smart campus initiatives on personalized learning experiences in a prior section, here we delve into the unique aspects that make OpenSCN an ideal platform for students to contribute their IoT expertise.

A distinguishing feature of OpenSCN is its commitment to fostering a culture of student innovation and collaboration. Our platform is not merely a closed system but a vibrant ecosystem open to student developers. It actively encourages students to contribute their IoT hardware solutions, sensors, and innovative projects.

First and foremost, OpenSCN's open-source nature allows students to engage in real-world technology development, welcoming code contributions and improvements. This allows both the platform itself to grow according to the institution's need and allows the students themselves to strengthen their practical

skills and nurture a spirit of creativity and innovation.

Furthermore, OpenSCN's primary goal is to provide an open gateway for students to seamlessly integrate new IoT devices, sensors, and projects into the smart campus environment. This integration empowers students to explore and implement their IoT ideas within a real-world setting, contributing to the continuous expansion and improvement of our smart campus ecosystem.

3.5 Smart Campus Deployment Process and Challenges

The journey towards building a smart campus is rife with complexities and challenges. It demands a well-thought-out deployment process to navigate the intricate landscape of technology integration, data management, and stakeholder engagement. We believe that a structured approach is indispensable to harness the full potential of smart campus solutions while mitigating the challenges that can arise during deployment.

3.5.1 Challenges and Considerations

Deploying a smart campus solution entails a multitude of challenges and considerations. These challenges are not only technical but also encompass organizational, logistical, and societal aspects. Here, we enumerate some of the prominent challenges that institutions may encounter on the path to realizing a smart campus, much to the similarity of the aforementioned challenges of deploying a smart city solution.

Integration Complexity

Smart campus solutions often involve the integration of diverse technologies, including IoT devices, data analytics platforms, and legacy systems. Ensuring seamless communication and interoperability among these components is a formidable task.

Data Security and Privacy

With the proliferation of data collection and analysis, safeguarding sensitive information becomes paramount. Balancing data accessibility with security and privacy concerns is a continuous challenge.

Resource Allocation

The financial and human resource requirements for smart campus deployment can be substantial. Institutions must allocate resources judiciously to prevent budget overruns and ensure sustainable growth.

Stakeholder Engagement

Engaging students, faculty, staff, and administration in the smart campus journey is vital. Resistance to change and varying levels of tech-savviness among stakeholders can impede progress.

3.5.2 A Framework for Smart Campus Deployment

To navigate these challenges effectively, we endorse a structured deployment process illustrated in Figure 3.4 [1]. This methodological framework is aligned with the broader scope of our study and is deemed suitable for our software platform and university facilities. The process consists of three distinct stages:

Stage 1: Sample Building Selection and Framework Establishment

In this initial phase, a representative building from the university campus is chosen, such as the one housing the Information Technology facilities. A comprehensive framework is developed, focusing on domains such as energy preservation, classroom administration, management and access control. The required hardware and networking infrastructure is deployed and integrated into our platform and the existing digital backbone of the premises, turning the academic grounds into an invaluable testbed for establishing an initial IoT ecosystem.

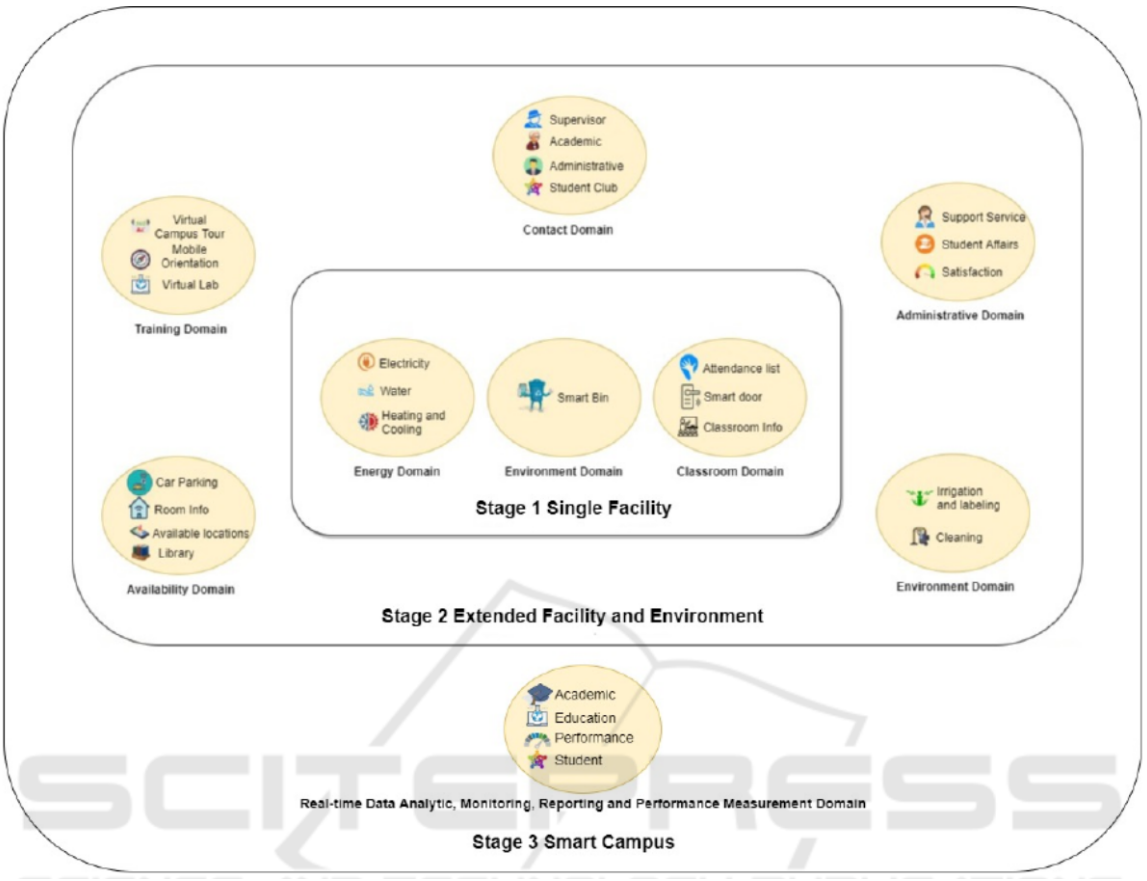

Figure 3.4: Smart Campus Deployment Framework [1]

Stage 2: Framework Expansion Across the Campus

Building upon the domains established in Stage 1, the framework is extended to cover more areas, with the ambition being to expand to the entire campus. Additional domains are identified and incorporated into the framework.

Stage 3: Data Integration and Advanced Analytics

Data generated by the systems in Stages 1 and 2 are integrated with existing university data. At this stage, predictive, prescriptive, diagnostic, and descriptive analytics can be performed using the consolidated data. This is enabled by OpenSCN's open data nature which makes all the information gathered available via secure APIs for reserach and model creation. Furthermore, analytical models can be easily integrated into the platform itself, with our ambition being to transform an OpenSCN powered campus into an ever growing source of valuable information and insights.

This methodological framework provides a clear path for the deployment of our software platform within the smart campus context. It allows us to systematically address challenges, maximize the utility of smart campus solutions, and create an ecosystem that enhances the overall educational experience. It is our hope that by following the roadmap and through an iterative embetterment process driven by our fellow student developers, the goals of a truly self-sufficient and intelligent academic site can be met.

3.6 Literature Review of Smart Campus Software Solutions

In the ever-evolving landscape of smart campus solutions, a variety of approaches have emerged to address the unique challenges and opportunities presented by educational institutions. In this section, we embark on a comparative journey, exploring different types of smart campus solutions found in existing literature. These solutions range from small-scale, task-specific scripts and basic web servers to sophisticated cloud-based platforms and renowned open-source frameworks like FIWARE. Each of these approaches serves its purpose, offering distinct advantages and trade-offs. At the heart of this exploration is OpenSCN, an open-source, academia-focused platform designed with ease of use in mind. We aim to unravel the strengths and weaknesses of various solutions, shedding light on their suitability for academic institutions and highlighting what sets OpenSCN apart in the pursuit of smarter campuses.

3.6.1 Small-Scale Solutions

One class of smart campus solutions that has emerged includes small-scale scripts and rudimentary web servers designed to address particular issues within the educational setting. These solutions are often created to tackle specific problems, such as classroom air quality monitoring or occupancy tracking. While these applications excel in their specialized tasks, they come with inherent limitations when considering broader smart campus implementation.

One indicative example of this type of solution found in the smart campus corpus is the paper describing an air monitoring solution in the university of Doha in Qatar [38], where platform specific firmware is

developed and a basic Apache server along with a Python script handle the workload and business logic of the solution. Another such example is a classroom attendance measurement and prediction platform developed in the University of South Wales in Sydney, Australia [37]. In this case, a similar problem-specific architecture has been developed, where a simple REST API handles the incoming measurements and posts them in a proprietary relational database through a message broker. An overview of this approach is illustrated in figure 3.5. Perhaps the most representative example of this categorization of intelligent campus solutions is the case of the University of Lisbon [39], where a separate client side application for each smart campus domain (e.g. HVAC control, lighting control) has been developed.

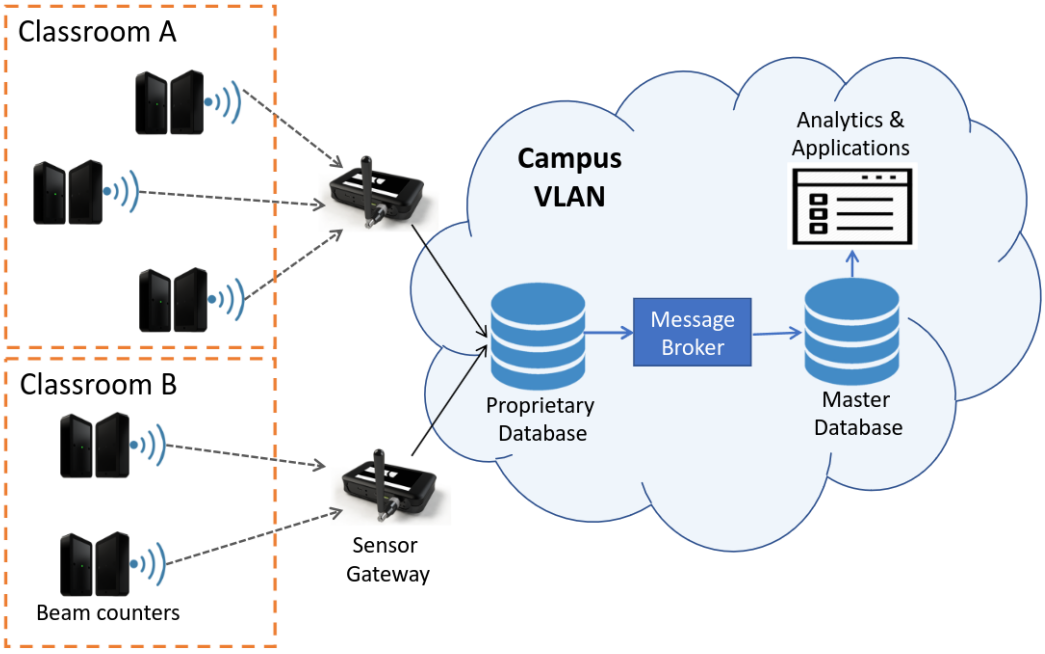

Figure 3.5: Occupancy Measurement Application Architecture [37]

In contrast to such solutions, OpenSCN offers a more comprehensive and integrated approach to smart campus management. The advantages and disadvantages of each approach warrant closer examination.

Pros of Small-Scale Solutions:

- Task-Specific Excellence: Small-scale scripts and web servers excel at their designated tasks. They provide precise solutions for individual challenges, such as monitoring air quality or tracking classroom occupancy.

- Simplicity: These solutions are typically straightforward to develop and deploy, making them accessible to institutions with limited resources or technical expertise.

Cons of Small-Scale Solutions:

- Lack of Scalability: The primary limitation of small-scale solutions lies in their scalability. While effective for specific issues, they struggle to adapt to a broader range of smart campus needs, hindering their usefulness in larger academic environments.
- Fragmentation: Deploying multiple small-scale solutions for various challenges can result in a fragmented smart campus ecosystem. Each solution may operate independently, making it challenging to achieve a cohesive, integrated infrastructure.

3.6.2 Utilizing Proprietary Cloud Platforms

Another category of smart campus solutions leverages cloud platforms, such as Amazon Web Services (AWS), Grafana, Prometheus, and others, to fulfill their data storage, web server, analysis and observability requirements. A plethora of these types of examples exist in the available literature. For instance, the case of the Sacramento State University [40] which uses AWS instances to host and deploy a predictive policing application on campus grounds. Another case of using existing cloud solutions is the University of Malaga [41] which uses the Grafana platform to create dashboards which provide insights into the campus' green space sensor measurements. The example which better encompasses this category however is the case study of the academic grounds in the University of Modena and Reggio Emilia in Italy [42]. Here, an entire stack of AWS solutions, which include cloud functions, a database, hosting and log monitoring are employed to create a smart campus environment which can be interacted with via a conversational interface. All of the aforementioned cloud-based services are known for their scalability and reliability. While they offer robust capabilities, they present distinct challenges and considerations when compared to OpenSCN's approach.

Pros of Proprietary Cloud Platforms

- Scalability and Reliability: Cloud platforms like AWS provide unparalleled scalability and reliability, which are crucial for handling large datasets and maintaining consistent service uptime.
- Rich Ecosystem: Proprietary cloud ecosystems often offer a wealth of tools and services, including data storage, analytics, and visualization, streamlining the development of smart campus applications.

Cons of Proprietary Cloud Platforms:

- Technical Expertise Required: Implementing and fine-tuning solutions on proprietary cloud platforms can be complex and typically requires a high level of technical expertise. This can pose challenges for educational institutions with limited IT resources.
- Vendor Lock-In: Reliance on proprietary cloud services may result in vendor lock-in, making it challenging to transition to alternative platforms or migrate data and services in the future.

- Cost: While cloud platforms offer scalability, they can also incur significant costs, particularly as smart campus deployments grow in scale and complexity.

3.6.3 FIWARE Based Solutions

Finally, there is FIWARE, which we would regard as one of the best alternatives to OpenSCN. FIWARE is a comprehensive smart solution known for its versatility and extensive capabilities. It provides a large number of software components which can address a broad spectrum of smart campus requirements, making it suitable for a wide range of applications. Examples of FIWARE powered platforms include the efforts of the Polytechnic Institute of Viana do Castelo to create an intelligent ecosystem combining six academic facilities into a single managerial entity [43].The authors use a multitude of FIWARE components to serve as an intermediary enabler of the data flow between the edge IoT devices and the client applications. Another successful implementation of a FIWARE powered smart campus is the example of Federal University of Rio Grande do Norte in Brazil [44]. In this case study, the authors use a context broker and a data persistence component provide by FIWARE in order to create a resilient architecture powering their academic ecosystem as seen in Figure 3.6.

However, there are critical distinctions between OpenSCN and FIWARE that stem from their target audience, ease of use, and deployment requirements.

Pros of FIWARE:

- Holistic Smart Solution: FIWARE is a holistic smart solution designed to cater to various industry needs. Its wide array of features can accommodate diverse smart campus applications, offering a high degree of flexibility.
- Interoperability: FIWARE is designed to ensure compatibility and interoperability between various smart technologies. This can be advantageous when integrating multiple systems or devices within a smart campus environment.
- Scalability: FIWARE's architecture allows for scalability, making it suitable for both small- and large-scale smart campus implementations. This adaptability is beneficial for institutions with varying sizes and resource capacities.

Cons of FIWARE:

- Complex Deployment and Maintenance: Deploying and maintaining FIWARE in a smart campus environment can be a complex task. Its comprehensive nature and extensive capabilities may require a dedicated team of engineers, contributing to high operational costs.
- Resource-Intensive: FIWARE's holistic approach, while offering a wide range of features, can be resource-intensive. This means that academic institutions, in particular, may find it challenging to allocate the necessary resources, both in terms of manpower and finances, to effectively implement and sustain FIWARE-based solutions

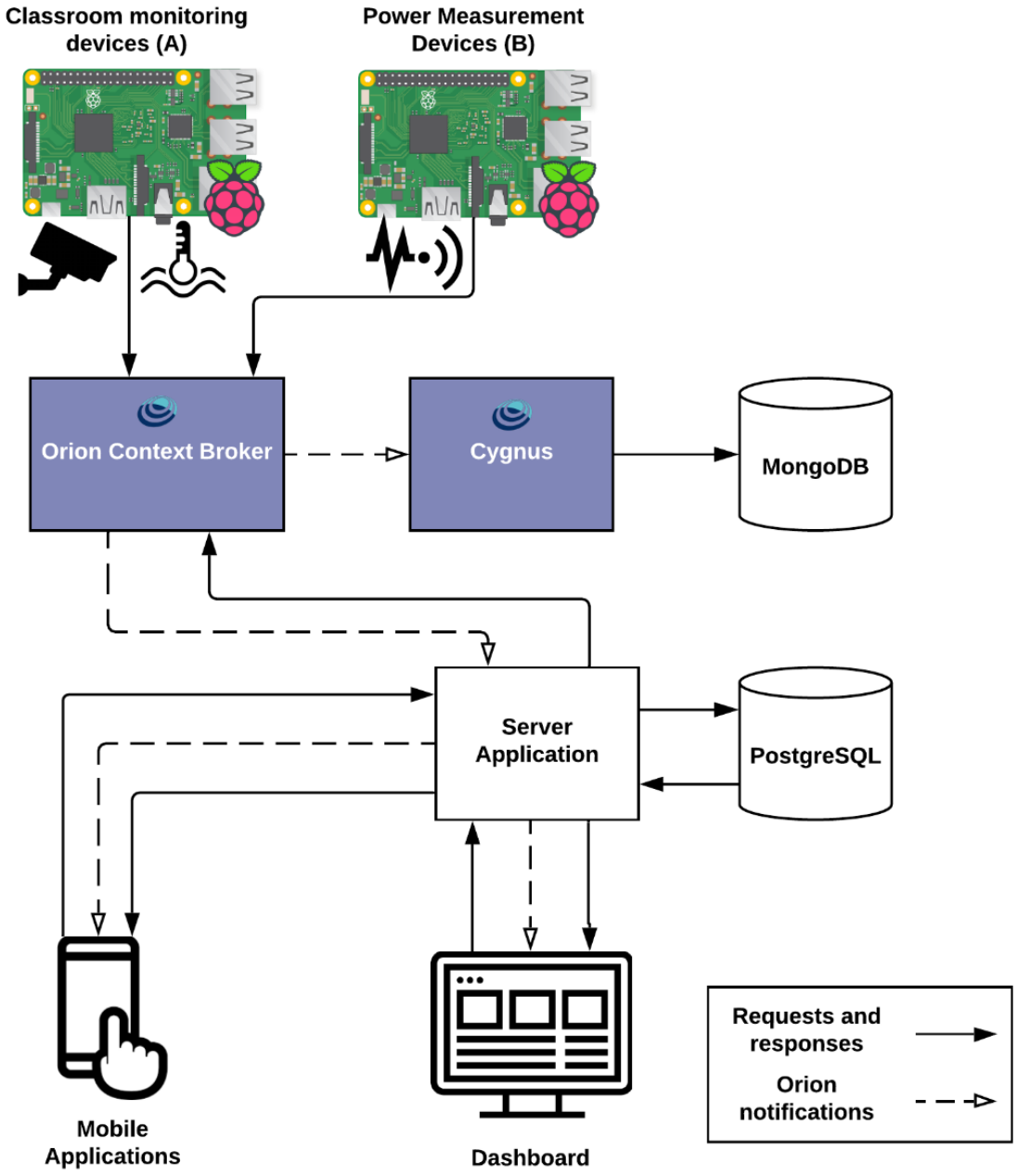

Figure 3.6: Architecture of Smart Campus in Brazil using FIWARE components [44]

On the other hand, OpenSCN distinguishes itself by providing an open-source, all-inclusive solution. While it may offer fewer features compared to FIWARE, OpenSCN prioritizes ease of use and accessibility, making it a more user-friendly choice for educational institutions and academia.

3.6.4 Summary

In conclusion, among the diverse landscape of smart campus solutions, we believe OpenSCN emerges as a promising and distinctive contender. While small-scale scripts serve specific needs efficiently, their lack of scalability and generality limits their broader application. Proprietary cloud platforms offer

robust capabilities but demand significant technical expertise and incur operational costs. FIWARE, the holistic industry-focused solution, though versatile, requires substantial resources both in manpower and operational costs. In contrast, OpenSCN presents a unique proposition – an open-source, user-friendly, and low-resource oriented platform. Its straightforward deployment, self-hosted nature, scalability and streamlined features make it an accessible choice for educational institutions. We hope to bridge the gap between simplicity and functionality, offering a cost-effective and easy-to-implement smart campus solution.

Chapter 4: Foundations of Smart Campus Software

4.1 Introduction

Creating a robust and effective IoT-driven smart campus platform involves addressing various challenges associated with data management, network communication, security, and scalability. In this section, we delve into the technological infrastructure that underpins our platform, OpenSCN, and discuss key architectural decisions made to tackle these challenges. We explore five fundamental components that we have identified as pivotal in shaping the platform's capabilities along with the problems they attempt to solve: Time Series Data Storage and Analysis, MQTT Networking, Event-Driven Architecture and Background Workers, Cloud Deployability and Containerization, and Device Security. By understanding the theoretical foundations and technologies behind our platform, readers will gain insight into how we hope to harness IoT to enhance campus functionality while maintaining security, scalability, and efficiency.

4.2 Time Series Data Storage and Analysis

4.2.1 Definition of Time Series Data

Time series data is a distinct form of data that comprises a sequential collection of observations, measurements, or values recorded at equally spaced time intervals or timestamps. Unlike conventional datasets, which typically consist of independent and unrelated data points, time series data introduces a temporal dimension. Each data point in a time series is associated with a specific moment in time, making it a valuable resource for understanding how phenomena evolve over time [45].

4.2.2 Treating Time Series Data Differently

Time series data demands specialized handling due to its inherent characteristics. Unlike conventional data, which can often be treated as independent and identically distributed (i.i.d.) observations, time series data exhibits temporal dependencies. The ordering of data points is crucial, as the sequence of observations is meaningful. For instance, in financial markets, the historical prices of a stock are interconnected, and the current price is influenced by past prices [46]. Similarly, in environmental monitoring, which is directly applicable to IoT, temperature readings at different times are linked by the laws of physics [47]. Therefore, analyzing time series data necessitates techniques that capture these dependencies and patterns, which are typically not applicable to conventional datasets.

4.2.3 Time Series Databases: Purpose and Advantages

To accommodate the unique characteristics of time series data, dedicated Time Series Databases (TSDBs) have emerged [48]. These databases are engineered to efficiently store, manage, and query time-ordered data. TSDBs are designed with several key purposes and advantages:

- Efficient Storage: TSDBs optimize data storage by employing compression techniques tailored for

time series data. This reduces storage requirements and enables the retention of extensive historical data.

- High Throughput: Time series data often streams in at high rates, such as sensor measurements or financial market ticks. TSDBs are built to handle such high data ingestion rates without sacrificing performance.
- Specialized Querying: TSDBs provide specialized querying capabilities for time-based data, allowing users to retrieve data within specific time intervals or apply time-based aggregation functions efficiently.
- Data Retention Policies: TSDBs enable organizations to define data retention policies, automatically purging or archiving older data, ensuring that storage remains manageable.
- Scalability: TSDBs can scale horizontally to handle growing volumes of time series data, making them suitable for large-scale applications.

In the realm of IoT, the importance of time series data and databases cannot be overstated. IoT ecosystems consist of numerous interconnected devices, sensors, and actuators that continuously generate data. This data is inherently temporal, as it captures the real-time behavior of devices and the physical world. Time series data from IoT devices provide a dynamic view of changing environmental conditions, equipment states, and user interactions.

TSDBs are the backbone of IoT data management because they are engineered to handle the high volume, velocity, and variety of data generated by IoT devices. By efficiently storing and querying time series data, IoT applications can make real-time decisions, detect anomalies, predict maintenance needs, and optimize resource allocation. For instance, in a smart campus scenario, TSDBs can seamlessly manage data from environmental sensors, occupancy detectors, and energy meters. This data can then be analyzed to ensure efficient building operations, enhance security, and deliver personalized services to students and staff. Without the specialized capabilities of TSDBs, IoT applications would struggle to extract valuable insights from the continuous stream of time-sensitive data, limiting their potential to transform various industries, including education.

4.2.4 Mining Insights from Time Series Data

The potential for time series analysis on such data is vast. Time series data allows us to uncover underlying patterns [49], seasonality [50], and anomalies [51] that might be hidden within conventional datasets. This information is invaluable in various domains, including finance, healthcare, environmental monitoring, and industrial maintenance.

For instance, in the context of smart campuses, time series data collected from various sensors and IoT devices can be leveraged to optimize energy consumption patterns [52], predict equipment failures [53], enhance security through anomaly detection [54], and provide personalized learning experiences. By delving into the temporal aspects of data, time series analysis empowers organizations to make data-driven decisions that maximize efficiency, reduce costs, and improve overall operations.

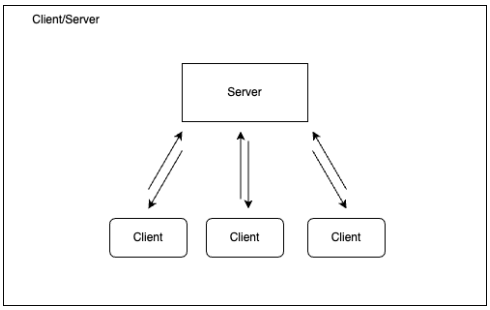

Figure 4.1: Client/Server diagram

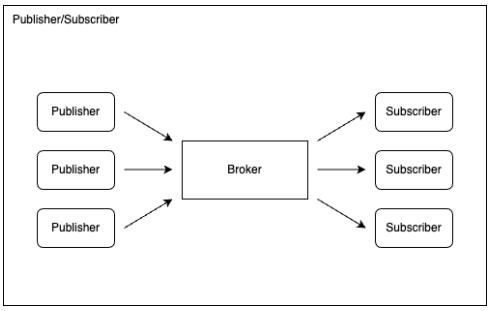

Figure 4.2: Publisher/Subscriber diagram

4.3 MQTT Networking

4.3.1 Introduction to MQTT

MQTT implements a publisher-subscriber paradigm, wherein devices, referred to as clients, communicate with a central broker [Figure 4.2]. This broker serves as a mediator, effectively decoupling the sender from the receiver. MQTT's minimal overhead and simplicity make it a suitable choice for IoT deployments, particularly in resource-constrained environments. Moreover, its asynchronous messaging model ensures that messages are delivered efficiently and reliably, even when devices intermittently connect and disconnect from the network. MQTT's adaptability extends to various Quality-of-Service (QoS) levels, allowing developers to fine-tune the balance between data reliability and network efficiency according to specific application requirements.

4.3.2 MQTT Architecture

MQTT architecture lies in the concept of a publish-subscribe messaging pattern. This pattern diverges from the traditional client-server model, offering greater flexibility and scalability in IoT ecosystems. In this paradigm, MQTT clients act as both publishers and subscribers, collectively forming a network of intercommunicating devices. Central to this architecture is the MQTT broker, an intermediary that serves as a message hub. Clients, be they sensors, actuators, or other IoT devices, publish messages to specific topics hosted by the broker. Subscribing clients express their interest in specific topics, and the broker ensures the delivery of relevant messages to these subscribers. This decoupled approach allows for efficient, one-to-many communication, reducing the complexity associated with point-to-point connections and enabling real-time data dissemination.

4.3.3 Quality of Service (QoS)

MQTT supports three kinds of quality of service. 0, 1 and 2 [55]. At the lowest level, QoS 0 in MQTT offers a best-effort delivery mechanism where the sender does not expect an acknowledgement or guarantee of message delivery. In QoS 1 of MQTT, the focus is on ensuring message delivery at least once to the receiver. When a message is published with QoS 1, the sender keeps a copy of the message until it receives a PUBACK packet from the receiver, confirming the successful receipt. If the sender doesn't receive the PUBACK packet, it re-transmits the message to ensure its delivery. QoS 2 offers the highest level of service in MQTT, ensuring that each message is delivered exactly once to the intended recipients. To achieve this, QoS 2 involves a four-part handshake between the sender and receiver [Figure 4.3].

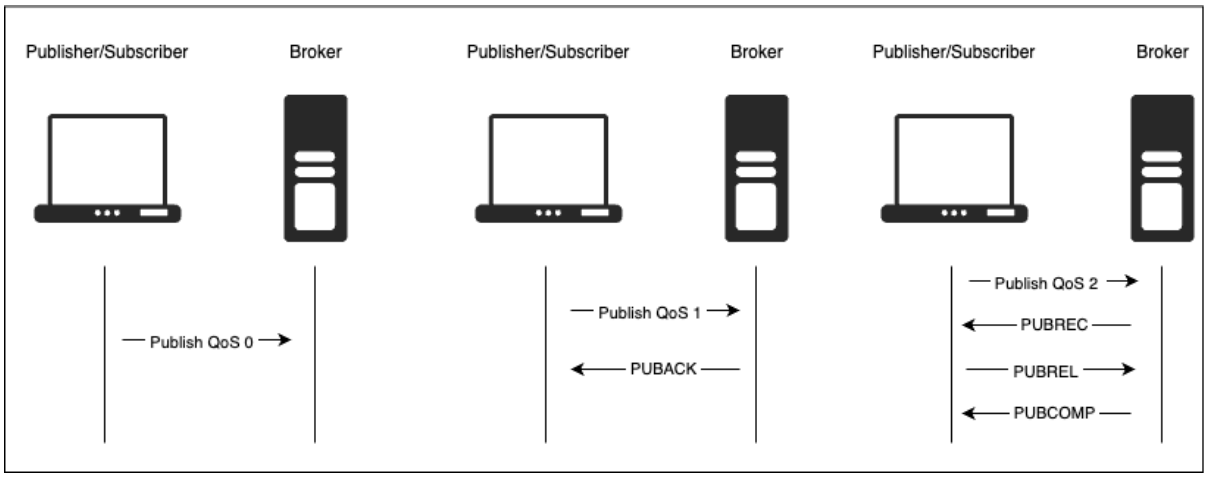

Figure 4.3: Packet transmission in MQTT per QoS level

4.4 Event Driven Architecture and Background Workers

4.4.1 Introduction to Event-Driven Architecture

In the digital age of interconnected systems, Event-Driven Architecture (EDA) has emerged as a pivotal paradigm for efficiently managing the flow of data and orchestrating actions across various components in real-time. EDA is fundamentally centered around the concept of events, which are significant occurrences or updates within a system. These events trigger the execution of predefined actions or workflows, enabling systems to react swiftly to changing conditions as is illustrated in Figure 4.4. In the context of IoT applications, where an abundance of data is continuously generated by sensors, devices, and users, EDA becomes paramount for processing, analyzing, and responding to data in a timely and efficient manner.

4.4.2 Background Workers: Processing Data Asynchronously

One of the key components of an event-driven architecture is the utilization of background workers. Background workers are specialized processes or services designed to perform tasks asynchronously, independent of the main application thread. In the context of IoT applications, background workers play a pivotal role in handling data-intensive operations without causing delays or bottlenecks in the core

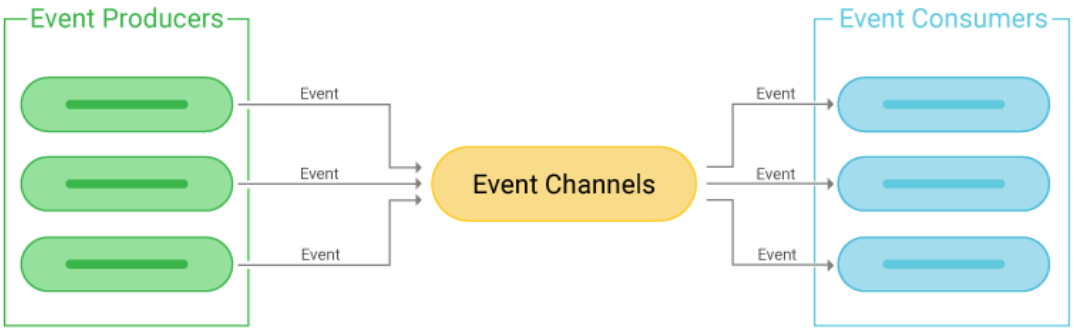

Figure 4.4: Event Driven Architecture Overview (source: ScyllaDB)

application logic. They can perform resource-intensive computations, data transformations, and database operations in the background, ensuring that the main application thread remains responsive to incoming events.

4.4.3 Scalability and Handling Data Intensity in IoT Applications

Scalability is a critical consideration in IoT applications, particularly when dealing with large-scale deployments in smart campus environments. Traditional monolithic architectures may struggle to cope with the increasing data intensity and complexity. Event-driven architecture, combined with background workers, offers an elegant solution to scalability challenges. By distributing tasks across multiple workers that can run in parallel, IoT applications can efficiently process and analyze vast amounts of data. This approach not only ensures that the system remains responsive but also allows for easy scalability by adding more workers as the data load grows.

In data-intensive IoT applications, the need for real-time responsiveness, scalability, and efficient data processing is paramount. EDA, with its ability to trigger actions in response to events, aligns perfectly with the dynamic nature of IoT data. Background workers complement EDA by enabling asynchronous data processing, preventing data bottlenecks, and ensuring that critical tasks are executed without hindering the overall system performance. Together, EDA and background workers empower IoT applications to harness the full potential of the data they collect, allowing for intelligent decision-making, predictive analytics, and timely responses to events. This architecture is especially relevant in the context of smart campus solutions, where data-driven insights and automation enhance campus operations, security, and the overall learning experience.

4.5 Cloud Deployability and Containerization

4.5.1 Defining Cloud Deployability and Containerization

Cloud Deployability

The term in the context of software development and deployment, refers to the ability of an application or system to be seamlessly hosted and operated within cloud computing environments. It involves the adaptation of software components and infrastructure to leverage cloud resources effectively, including computing power, storage, databases, and networking capabilities. Cloud Deployability provides numerous advantages, such as scalability, flexibility, and cost-efficiency. It enables software applications to harness the on-demand provisioning of resources offered by cloud service providers, thereby ensuring optimal performance and resource utilization.

Containerization

Containerization is a foundational technology within the realm of cloud deployability and represents a method of packaging, distributing, and managing software applications and their dependencies in a lightweight, consistent, and efficient manner [56]. At its core, containerization involves encapsulating an application and all its required libraries, runtime, and configurations into a single, standalone package known as a container. Containers are designed to operate consistently across various environments, from development and testing to production, irrespective of differences in underlying infrastructure. This consistency is achieved through the utilization of container engines, such as Docker [57], which provide an isolated runtime environment for containers on top of the host Operating System (OS) (Figure 4.5). Containerization abstracts the application from the underlying infrastructure while maintaining compatibility and portability, allowing developers to focus on application logic rather than system-specific configurations.

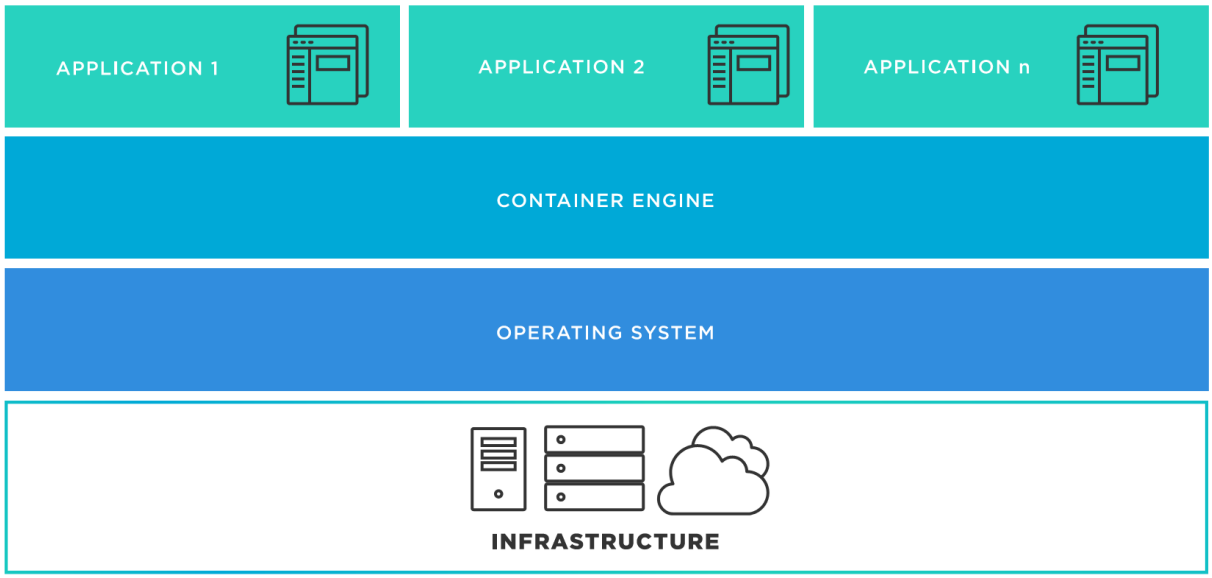

Figure 4.5: Software Containerization Diagram (source: TIBCO Software)

In essence, containers serve as a lightweight, portable unit that contains everything needed for an application to run, including code, runtime, system tools, and libraries. These containers can be rapidly deployed and executed on any cloud server or container orchestration platform, making them highly suitable for modern, distributed architectures. Containerization ensures that applications and their dependencies are consistently reproducible across different development, staging, and production environments, which is crucial in IoT platforms like ours, where uniformity and predictability are paramount for robust and scalable operations.

4.5.2 Importance of Cloud Deployability and Containerization in IoT Platforms

In the realm of IoT platforms, the importance of Cloud Deployability and Containerization cannot be overstated. IoT systems inherently involve a multitude of devices and sensors, generating copious amounts of data. Traditional monolithic architectures often struggle to efficiently manage this influx of data and the dynamic nature of IoT workloads.

4.5.3 Containerization and Distributed Architectures

One of the key advantages of containerization is its seamless integration with distributed architectures, such as the one embraced by our platform. In a distributed IoT environment, various components need to scale independently to accommodate fluctuations in demand. Containerization allows these components to be packaged individually, with their specific dependencies and configurations, while maintaining compatibility and consistency across the entire system.

For example, in a Smart Campus scenario, different components, such as data ingestion, analytics engines, or user interfaces, may experience varying levels of demand. Containerization enables these components to be independently scaled up or down, responding to real-time requirements without affecting the stability of other system elements. This granularity in scaling ensures efficient resource utilization and optimal performance.

4.5.4 Container Orchestration Potential

Containerization also opens the door to container orchestration, a critical aspect of managing containerized applications in dynamic environments. Container orchestration platforms like Kubernetes [58] and Docker Swarm provide tools for automating the deployment, scaling, and management of containers. This automation not only simplifies the operational aspects of IoT platforms but also enhances their resilience and fault tolerance.

In the context of Smart Campuses, container orchestration can play a vital role in ensuring the reliability and availability of services [59]. For instance, if a specific IoT application experiences a surge in usage, orchestration tools can detect this change and automatically spawn additional containers to handle the increased load, all without manual intervention. Conversely, when demand diminishes, excess containers can be gracefully decommissioned to optimize resource utilization.

4.6 Device Security

4.6.1 Importance of Device Security in IoT Applications

In the rapidly evolving landscape of IoT applications, device security stands as an unequivocal imperative. This significance extends profoundly into the realm of smart campuses, where an intricate web of devices continuously gathers, processes, and transmits data to enhance the educational and operational experience. The foundational principle driving the criticality of device security in IoT applications is the interconnectedness of these devices. In a smart campus ecosystem, everything from environmental sensors and access control systems to classroom devices and student wearables operates in unison.

Security breaches within this ecosystem can lead to catastrophic consequences. Unauthorized access to devices or data can compromise the integrity of campus operations and student safety [60]. For instance, an unsecured device could potentially allow a malicious actor to manipulate environmental controls, disrupt educational services, or access sensitive student records. In more extreme scenarios, an unsecured device could become a gateway for broader attacks on the campus network, putting the entire infrastructure at risk.

Moreover, the ramifications of inadequate device security extend beyond operational disruptions. In an environment where personal information and sensitive data are routinely exchanged, protecting user privacy becomes paramount. A breach in device security can result in the unauthorized access and exposure of personal data, violating the trust of students, faculty, and staff. This not only leads to potential legal consequences but also erodes the reputation and credibility of the institution.

To mitigate these multifaceted risks, comprehensive device security measures are essential. This entails the implementation of stringent access controls, authentication mechanisms, encryption protocols, and continuous monitoring to detect and respond to security incidents promptly. In the context of smart campuses, where the stakes are high and the attack surface vast, device security is not merely a choice but a foundational necessity. Its importance lies not only in safeguarding operations and data but also in upholding the trust and safety of the campus community.

4.6.2 Authentication Mechanisms for Device Identity Verification

A common authentication method used in IoT applications is the Public Key Infrastructure (PKI) [61]. PKI relies on cryptographic principles to establish the authenticity of devices. Each device is assigned a unique pair of cryptographic keys: a public key, which is shared openly, and a private key, which is securely stored and known only to the device. When a device attempts to connect to the network, it presents its public key for verification. The network, in turn, uses the corresponding private key to confirm the device's identity. This method ensures that only devices possessing the correct private key can gain access.

An alternative approach gaining traction in IoT security is the use of authentication tokens [62], [63]. Authentication tokens are unique, secure, and often time-limited codes that devices use to prove their identity. These tokens offer a pragmatic solution for IoT applications, where the management of cryptographic keys for a large number of devices can be complex and resource-intensive.

Authentication tokens can be generated, securely distributed, and validated within the IoT ecosystem. An example of device authentication using MQTT and security tokens is illustrated in Figure 4.6 [64]. When a device seeks access, it presents its authentication token to the system. The platform verifies the token's authenticity, granting access if the token is valid. One of the primary advantages of this method is its simplicity, making it an efficient and user-friendly way to establish device identity in large-scale IoT deployments.

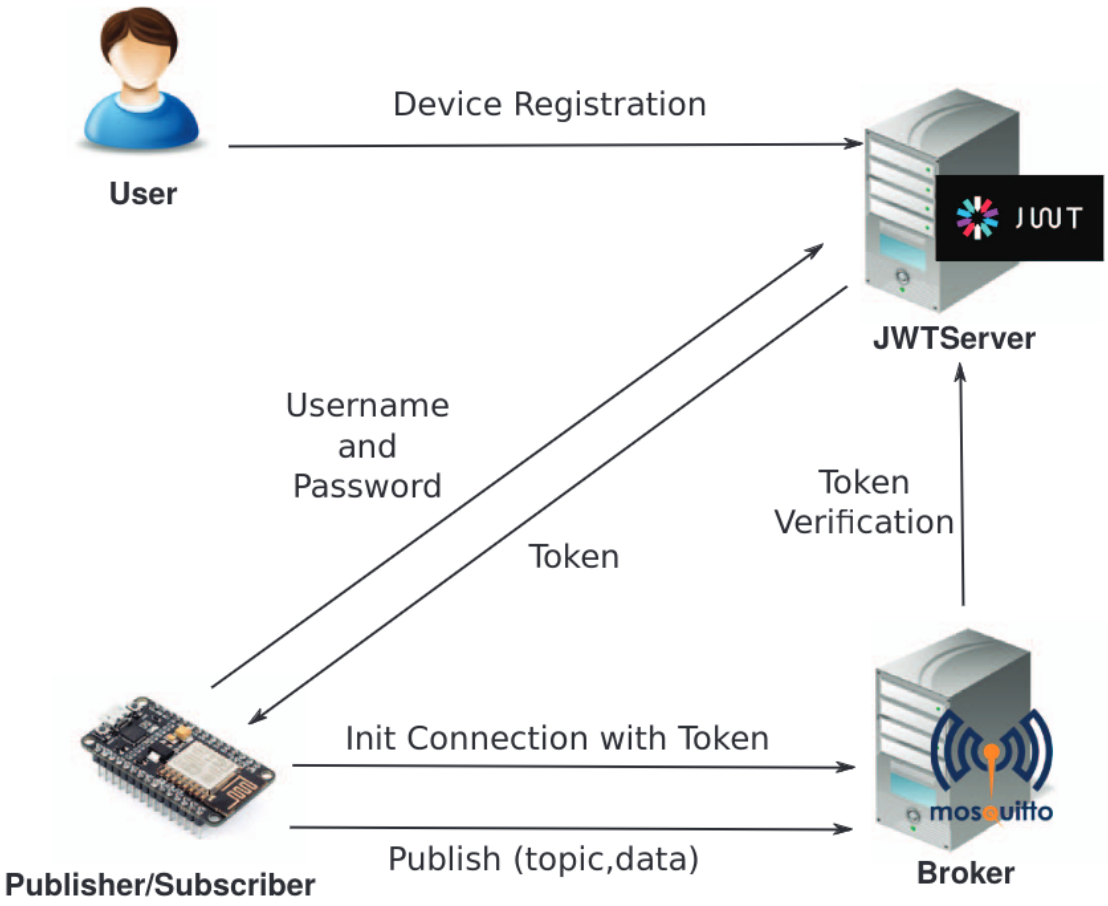

Figure 4.6: IoT Device Authentication with MQTT and Tokens [64]

Chapter 5: The OpenSCN Platform

5.1 Architecture

5.1.1 Introduction

In the current section, a holistic analysis of all of OpenSCN's building blocks will be presented and meticulously explained following a top-down approach. Starting by showcasing the workflow of the application and how all services are orchestrated, the following subsections will focus on each of the main software components along with the technologies used to realize them and the authors' thought process behind these architectural decisions.

It is reminded that OpenSCN takes pride in being fully open source and all of the application's source code can be found in Appendix A for further review and collaboration.

5.1.2 Overview

OpenSCN consists of multiple components working in sync to provide all of the application's suite of IoT data collection, monitoring, analysis, alerting and actuation services while simultaneously working towards minimizing performance hindering when the underlying smart infrastructure grows in size.

Figure 5.1 features a high-level illustration of the application's main components. A simplified view of the data and communication flow within OpenSCN is the following: The main producers and consumers of data are sensors and actuators respectively, which communicate via their parent devices with the app by reaching the main web OpenSCN server. They can achieve communication either via regular HTTP requests, or by using the application's MQTT networking capabilities. The collected IoT measurements are stored in a time series database, while all of the rest of the data needed by the application is stored in a traditional PostgreSQL relational database.

A separate engine responsible for doing any desired analysis on the collected data and producing alerts whenever necessary is always running simultaneously to the main server. These two components communicate with each other via a shared events queue. The end user is offered a simple dashboard to interact with the app and monitor its activity.

5.1.3 Data Model framework

The schema of OpenSCN uses the standard models defined in schema.org [65]. This resource is a collaborative project that aims to create a standardized vocabulary or schema for structured data [66].

5.1.4 Web application data model

OpenSCN is designed in a modular fashion. Each of the application modules provides its own data models.

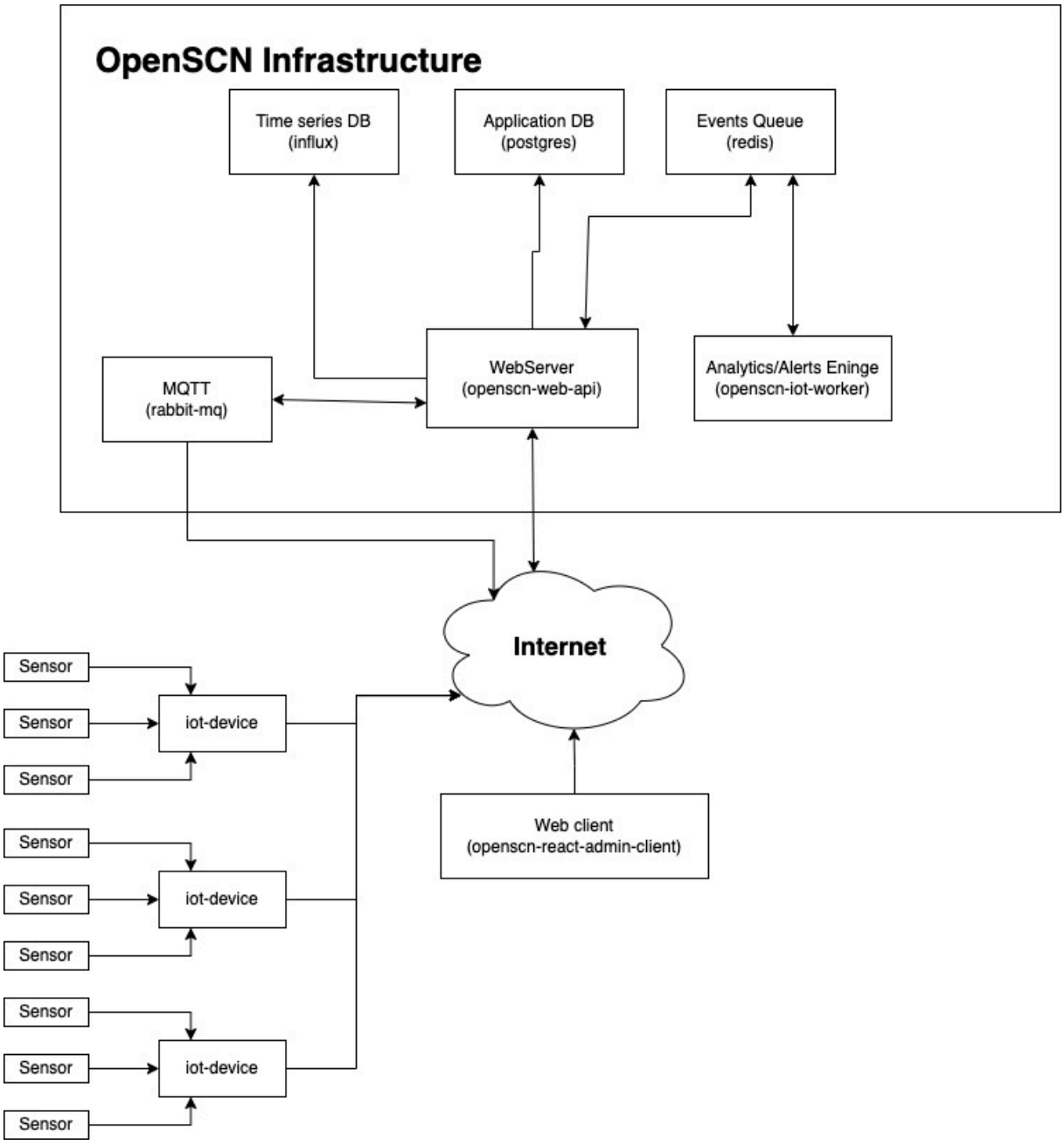

Figure 5.1: Architecture Diagram

Common module

The common module contains commonly used functionality and models. This module is expected to be shared across many modules of the application. The models it provides are some core entities that should not be instantiated, namely the ”Thing”, ”Person” and ”EntityBase” models. Other modules are expected to extend those entities and use their functionality and fields. Apart from these, the common module provides the ”ApplicationEvent” model. This model describes an event in the system. It is used for loose communication between modules (for modules that make sense to be independent services) and system status replication (event sourcing [67]).

User module

The user module provides the User as an entity. The User is a human actor and is characterized by their identifier and their privileges level. There are two levels of privileges, the ”admin” and ”non-admin” users. By default, a user does not have admin privileges. The user model is used to store other information related to the client such as the preferred user email and preferred display name [Figure 5.2].

Auth module

The entities of the Auth module describe user accounts. At this point, OpenSCN supports only local accounts with email and password, but abstraction is present to decouple the user account from its credentials. In detail, the ”UserAccount” model represents an access method for a user and it is identified by its email property. A user can be related to multiple accounts. This relation is a loose reference for keeping the modules decoupled. The ”UserAccount” model has the ”user_id” property as a reference to a User. A ”LocalAccount” store the encrypted credentials. A ”LocalAccount” instance is coupled with a ”UserAccount” with a one-to-one relation [Figure 5.2].

In addition to the ”UserAccount”, the Auth module provides the ”AuthToken” entity. This entity stores a unique token with random symbols and can be used for either password reset or registration. The ”additionalType” property is being used to distinguish the target functionality (as defined in schema.org). This model defines other metadata such as whether the token has been used and its expiration date [Figure 5.3]

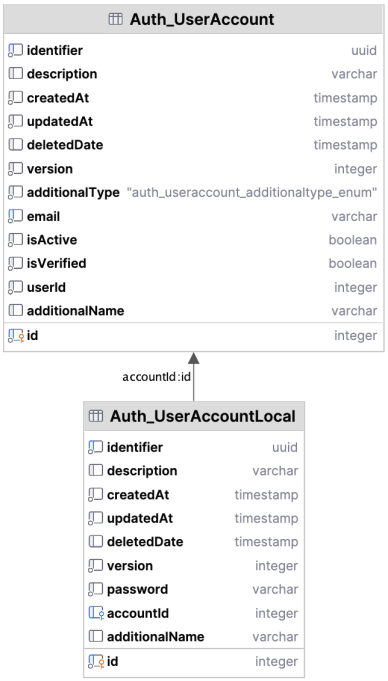

Figure 5.2: ”UserAccount” model definition. ”UserAccount” has a relationship with ”UserAccountLocal” and uses the ”userId” property for loosely referencing ”User” entities by their id

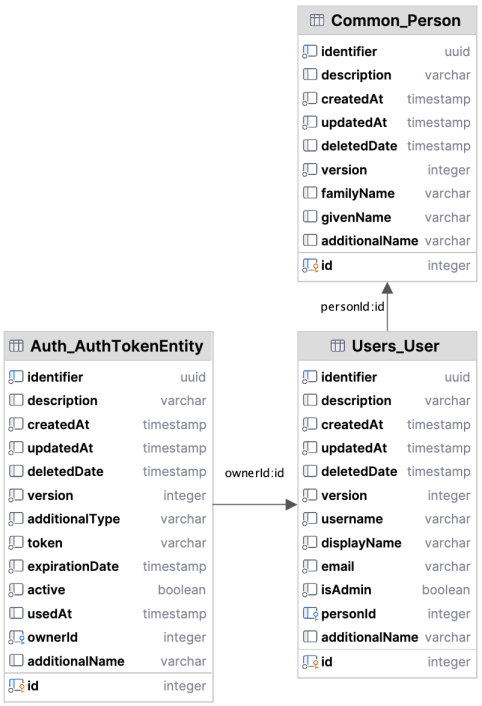

Figure 5.3: User model definition in the context of the Auth module. User entity has references to Person and AuthToken entities

IoTDomain

The IoT domain module contains most of the application logic. It is designed to provide all the functionality of the entities in the IoT context of the application that users interact with. Such models are ”IoTDevice”, ”IoTDeviceAccessToken”, ”Sensor”, ”Actuator”, ”Label”, ”Measurements” (data collections), ”Alerts” etc.

IoTDevice represents an IoT gateway device with web connectivity. ”IoTDeviceAccessToken” represents access tokens that devices can use for authentication purposes. Sensors represent physical sensors that can be characterized by their value type and measurement unit. ”Actuator” entities represent physical devices that can have effects on the real world. Sensors and actuators have a parent ”IoTDevice” that they belong to and use for authentication in the system. The ”Alert” entity a model that represents an interesting event in regard to the IoT domain that the user (or the system) should react to. These are the core IoT Models [Figure 5.4]

The other models are complementary models that help users organize data and configure the analytics engine of OpenSCN. Such models are ”IoTAlertConfiguration”, ”IoTActuatorTrigger”, ”SensorMeasurement” and ”SensorLabel”. The ”IoTAlertConfiguration” and ”IoTActuatorTriggerConfiguration” models describe the configuration for autonomous actions of the system for user alerting and automatic actuator triggering respectively. The ”SensorMeasurement” model is being used to divide the data collection in time and to mark a data collection as testing or production use. The absence of an active ”SensorMeasurement” is understood by the system as the sensor does not accept values. Labels are used to organize sensors and actuators in convenient dashboards.

Task

The task module encapsulates the background task configuration. The models defined in this module are the generic configurations that tasks require such as the target sensor and owner of the task and the other models represent the details of each task category. For instance, the configuration models are the ”IoTtask” and ”IoTTaskConfiguration” models and the models that encapsulate the details are the ”IoTTaskCheck”, ”IoTTasckCheckThreshold” and ”IoTTaskConfigurationDetails” models. At this point, the threshold task is fully developed.

Client

The Client module is being used for functionality that is specific to the user interface and has a single model for the client notifications. These are the notifications that the user sees on the web client. It also stores information about whether the user has interacted with the notification.

5.1.5 IoT Data models

The modeling the the IoTData is dictated by our technical infrastructure. We use InfluxDB (more on section 5.1.7), so the values are represented in the InfluxDB format. We use tags to further mark IoT data. The tags the OpenSCN uses are ”type” (”sensor” or ”actuator”), ”targetId” (the public identifier of the sensor or actuator) and ”measurement” (the public identifier of the data collection that the measurement belongs to).

5.1.6 Web API

At the heart of our application sits the Web API. This component handles the entirety of OpenSCN’s business logic and is the main communication hub of all its intermediate components.

We have chosen to build the API using NestJS. It is a TypeScript based open source web framework which was chosen specifically to serve the two main pillars of our vision for the application; scalability and extensibility.

The former is achieved due to the asynchronous nature of the programming language and the NestJS framework. During a regular IoT web application workflow, the bulk of the application’s delays 7comes from I/O driven operations. The constant stream of incoming and outgoing data which results in a great number of databases reads and writes, as well as the need of continuous network requests in order to communicate with the rest of OpenSCN’s components, means that the overall performance and scalability of this type of decoupled architecture heavily relies on minimizing I/O driven delays.

JavaScript and by extension TypeScript, is incredibly well equipped to handle these types of scenarios. At the core of the programming language sits the event loop. JavaScript’s engine enables concurrency by removing tasks which involve I/O delays from the main thread’s call stack and placing them in a queue. The event loop constantly checks these tasks’ progress and when they are ready to be processed,

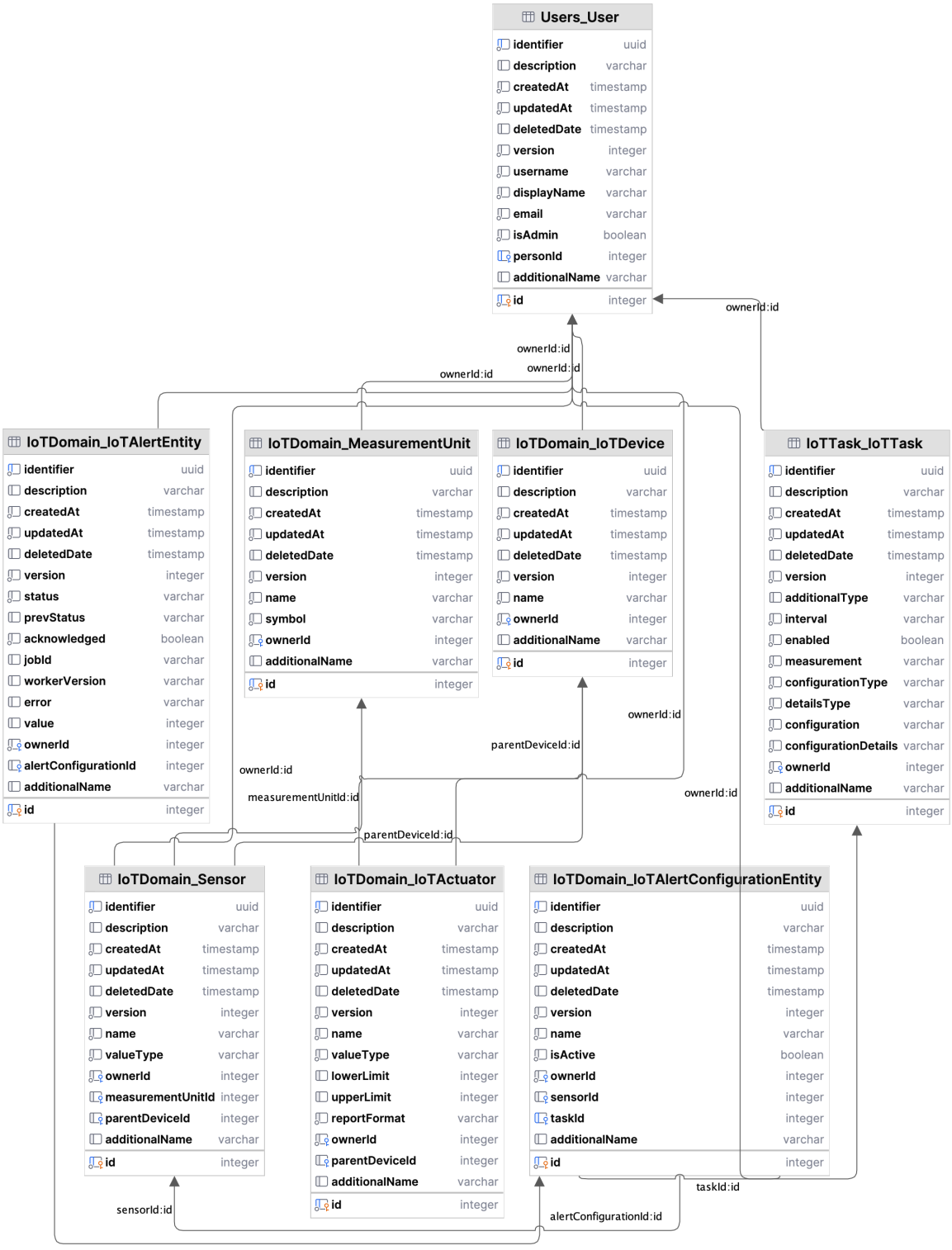

Figure 5.4: Database schema of the core IoT Entities

it places them back into the main thread's calls stack. This mechanism ensures that the main thread is always busy and no time is wasted waiting for long I/O operations to complete. In practice, this type of concurrency management has been proven to outperform traditional multithreaded approaches [68] and ensures optimal computational resource utilization for our application.

The NestJS framework also offers us the advantage of a well defined and extensible internal architecture. It is based on battle tested software development principles over decades, such as the SOLID guidelines [69] and ensures that OpenSCN's source code is future proof, adheres to industry standards, is inviting for new collaborators' contributions and is not havily coupled with underlying dependencies, allowing freedom of evolution.

5.1.7 Time Series Database

As explained in section 4.2, choosing a Database Management System (DBMS) specifically tailored for the needs of time series data is a crucial optimization which pays dividends in an IoT focused app's performance. Our Time Series Database Management System (TS-DBMS) of choice is InfluxDB. It is a fully open source DBMS which has become an industry standard in time-critical applications. By observing figure 5.5, one can draw the conclusion that not only are TS-DBMSs steadily rising in popularity due to the meteoric ascent of IoT and smart infrastructure, but also that InfluxDB sits comfortably at the top of the popularity charts.

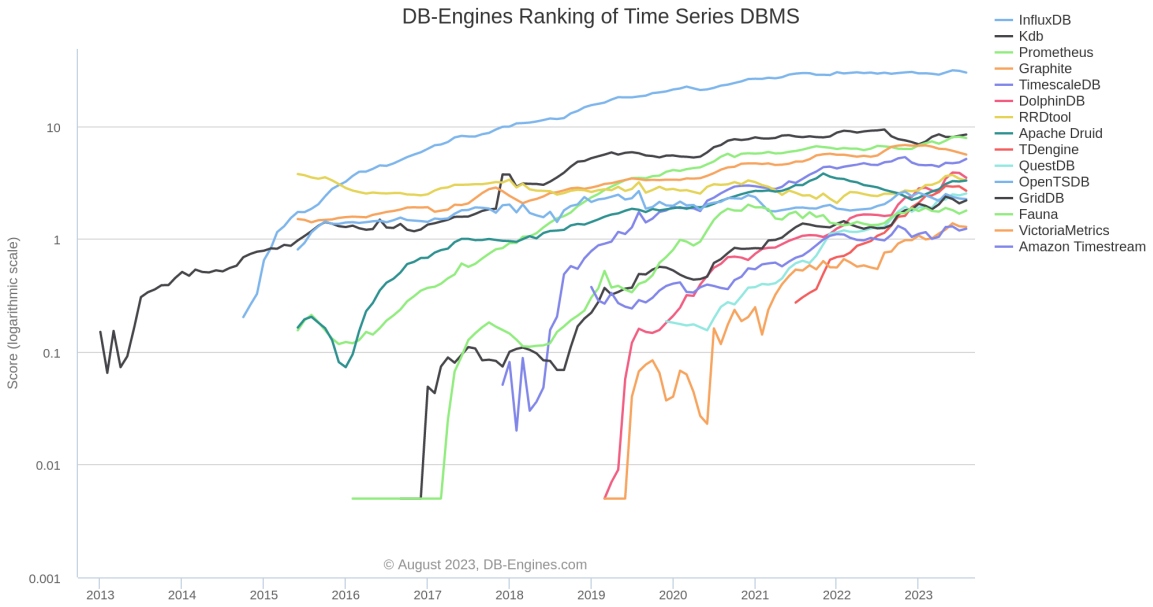

Figure 5.5: Popularity chart of Time Series Databases (source: https://db-engines.com)

InfluxDB features a flexible data structure tailored to the needs of timestamped collected data. In particular, data in in Influx is grouped into measurements, which are user-defined collections that share the same metadata. Each record in a collection is a sensor or actuator reading which contains a timestamp which is always used as the sorting key, a key-value pair of what is being measured and the stored value called "fields", along with metadata called "tags". Tags can be removed or added on command for each point, providing a flexible schema. Each record is called a data point, but since single data points do not offer valuable insight by themselves, the main analytical focus is on groups of continuous points, called "series". Figure 5.6 showcases an overview of how a series of data points would look like in InfluxDB. In this example, the series of records all belong to the "weather" measurement, the unit is measured is temperature and "city" and "country" are tags added to enrich the collected information. This is the general schema that all OpenSCN data follows.

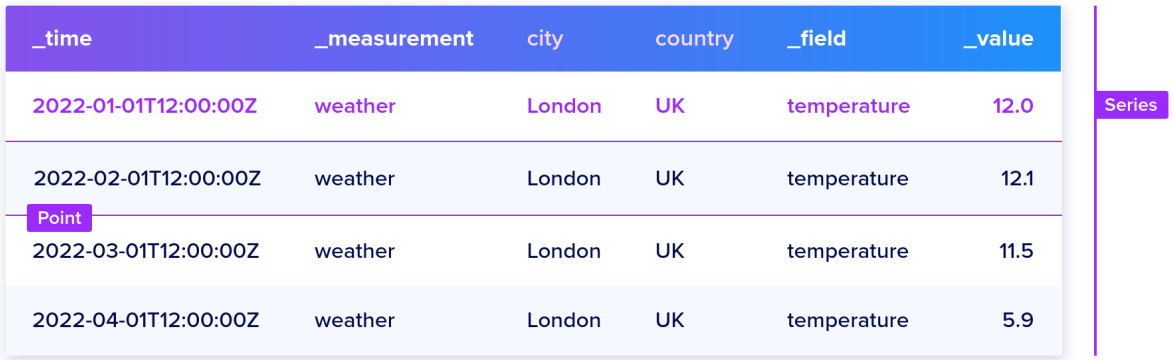

Figure 5.6: Example of data points in InfluxDB (source: https://docs.influxdata.com/influxdb/v2)

The internal engine and design of InfluxDB prioritises performance of information storage and retrieval, driven by the special use cases of IoT data which makes it an ideal choice for our application. In particular, it always handles read and write queries first, returning results of queried data first and subsequently processing any transaction related information, as well as putting tight restrictions on update and delete operations.

Furthermore, driven by the assumption that in the broader context a single IoT measurement is insignificant compared to the whole of the collected data set, Influx offers built in tools and optimization to aggregate data and handle larger measurement collections.

5.1.8 Analytics and Alert Engine

In Section 4.4 the advantages of an event driven architecture were analyzed. In OpenSCN, these advantages are leveraged by our analytics and alerts engine. In essence, this engine is a collection of background workers that collect task requests from the main web API, perform the execution separately and concurrently and finally send the execution results back to the server. An illustration of this workflow is provided in Figure 5.7.

The analytics and alerts engine is built using BullMQ, an open source library built on top of JavaScript which is designed to support performant queue based systems. It was chosen due to its ease of use and the fact that it is designed to tackle the most prominent issues with distributed systems such as ours, prioritizing performance and scalability. In particular, BullMQ makes horizontal scaling a trivial matter, as it is allows to simply add worker processes on demand when the processing load increases. In addition, each worker can be configured to execute multiple jobs concurrently, thus exponentially increasing the amount of processing that can be executed in parallel.

This setup also puts a heavy focus into consistency. A common issue encountered with this type of decoupled architecture is that as the main API and the worker engine do not communicate directly, but through an event queue, any failed or problematic job requires careful handling and precise updates to the web API. In addition, issues such as events being accidentally put into the queue more than once can cause major inconsistencies. BullMQ attempts to handle all the problems by providing out of the box features such as exactly once queue semantics, meaning that all messages between parties are delivered no more than once, automatic retries of failed jobs, as well automatic recovery from process crashes.

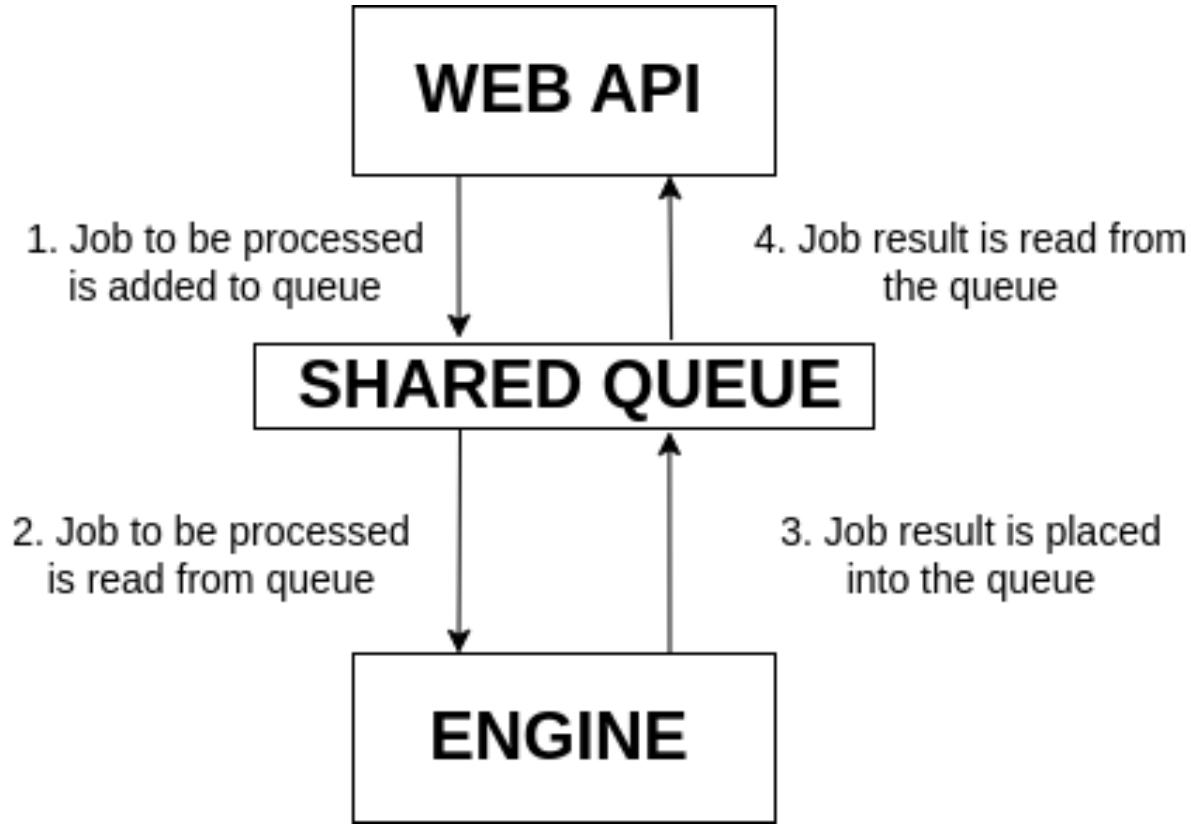

Figure 5.7: Worker Engine Workflow

Regarding the types of jobs can be set up for execution, once again we have attempted to set up the OpenSCN platform with extensibility in mind. All of the tasks are simple JavaScript scipts. We provide a few basic ones out of the box such as IoT value threshold checks, however users can easily add their own ones to fit their data processing needs.

5.1.9 MQTT with RabbitMQ

The MQTT protocol is a lightweight and efficient messaging protocol designed for reliable communication between devices in a network, particularly in scenarios where bandwidth, power, and network resources are constrained [70]. It operates on the client-server model, where devices or clients communicate with a central server, often referred to as the broker.

MQTT is built upon the publish-subscribe messaging pattern [Figure 4.2], which allows devices to publish messages to specific topics and subscribe to topics of interest. Topics are essentially named channels that messages are associated with. Clients can publish messages to a topic, and the broker forwards these messages to all subscribers of that topic, ensuring that data is efficiently distributed to interested parties.

RabbitMQ is a widely used open-source message broker that facilitates communication between various applications and systems. While RabbitMQ is not inherently an MQTT broker, it can be configured to support the MQTT protocol through plugins.

Integration strategy

There are many proprietary and open-source MQTT solutions with various support for MQTT features. OpenSCN uses RabbitMQ as the MQTT broker. This selection was based on the key feature of RabbitMQ that can unify HTTP and MQTT traffic under the same stream. This is achieved based on the Advanced Message Queuing Protocol (AMQP), that RabbitMQ uses internally, which supports both client/server [Figure 4.1] and publisher/subscriber models. RabbitMQ is primarily a message broker and provides MQTT support using the MQTT plugin.

RabbitMQ is designed for federated cluster deployment, but OpenSCN in its current form uses it as a single node in a single deployment. OpenSCN creates a virtual host and an administrative user as a platform representative. Each device is assigned a user and uses a username and password as authentication. The device's public identifier is selected for the user name and the active authentication token is selected as the password.

MQTT Topics and Subscriptions

In the current implementation of OpenSCN sensors are logically organized under a single node. This node can be either a device or a label. This topology dictates the MQTT topics, which have two levels, the parent entity and the target entity. A parent entity can be an IoT edge device or a label. Target entities are sensors and actuators. For all entities, their public identifiers are used in topics. An example topic can be ‘”/<device_identifier>/<sensor_identifier>”‘ (e.g. ‘”/5c1d92a7-aabf-433a-94da-098b980b4e8d/91468981-8a14-46eb-bca6-4f71ff5ded17”‘).

5.1.10 Containerization with Docker

In this section, we provide a detailed overview of how Docker containerization was strategically employed for each critical component of OpenSCN. By leveraging Docker, we gained numerous advantages in terms of deployment, scalability, and management as described in Section 4.5, while also addressing specific challenges effectively.

Component Identification

We embraced Docker containerization for several pivotal components within our IoT platform, including our Web API, client application, and the analytics and Alerts engine. Each of these components represents a vital cog in our platform's operation, and Dockerization ensured that they functioned optimally in a consistent, isolated environment.

Docker Image Creation

Creating a Docker image involves encapsulating an application and its dependencies into a portable package. For our components, this process initiated with defining a Dockerfile for each component, which precisely outlines the required configuration, libraries, and dependencies. Once these Dockerfiles were established, Docker automatically built images based on these specifications. These images could then be deployed across various environments, ensuring that each instance of our components was uniform and reproducible.

Docker Compose for Streamlined Deployment

further enhance the ease of deployment and orchestration of our Dockerized components, we employed Docker Compose. Docker Compose simplifies the process of managing multi-container applications by allowing us to define, configure, and run all the services that compose our IoT platform in a single, easy-to-read docker-compose.yml file.

With Docker Compose, deploying our platform along with associated services like PostgreSQL, InfluxDB, background workers, and Redis becomes an effortless task. The docker-compose.yml file specifies the services, their configurations, and the networks that connect them, all in a declarative format. This simplifies not only the deployment process but also the scaling and management of our platform across different environments.

Benefits of Docker Containerization

- Scalability and Replication: Docker containers have been instrumental in enhancing the scalability of individual components. By enabling containers to scale based on demand, our platform as a whole achieves a greater level of scalability and responsiveness.
- Resource Efficiency: Docker containerization optimizes resource utilization. Components run in isolated containers, preventing resource contention and ensuring efficient utilization of available resources.
- Deployment Consistency: Docker's containerization ensures deployment consistency across various environments, including development, testing, and production. This consistency minimizes discrepancies and reduces deployment-related issues.
- Updating and Rollbacks: Docker facilitates seamless updates by deploying new container versions. In case of issues, Docker allows for easy rollbacks to previous container versions, maintaining platform stability and minimizing disruptions.
- Isolation and Security: Containerization enhances security by providing isolation between components. It reduces the attack surface and prevents cross-component interference, making the platform more secure.

- Development Workflow: Docker containers have improved our development workflow significantly. Developers work in identical environments, reducing the common problem of ”it works on my machine.” Docker Compose also simplifies environment setup, making it easy to replicate environments.
- Operational Efficiency: Docker containers streamline operational tasks such as monitoring, logging, and scaling. This operational efficiency results in effective and efficient platform management.

5.2 Architectural Comparison with Commercial IoT cloud

5.2.1 Azure IoT

This section will compare OpenSCN with a large-scale commercial IoT cloud solution to demonstrate the position of our solution to an industry-grade one. Microsoft has made publicly available the details of their IoT cloud platform [71], Azure IoT. Azure IoT is an end-to-end IoT solution which manages all aspects from the IoT devices provision to analytics. We will compare these two solutions on the principles that they are built on and not on the actual features and capabilities, providing an insight into how our own platform compares to an industry standard and an architectural overview of which we were heavily influenced.

Microsoft splits their infrastructure into three categories. “Things”, “Insights” and “Actions” [Figure 5.8]. We will use these terms to describe our system as well. OpenSCN does not implement all of these categories but has many similarities with Azure on a smaller scale.

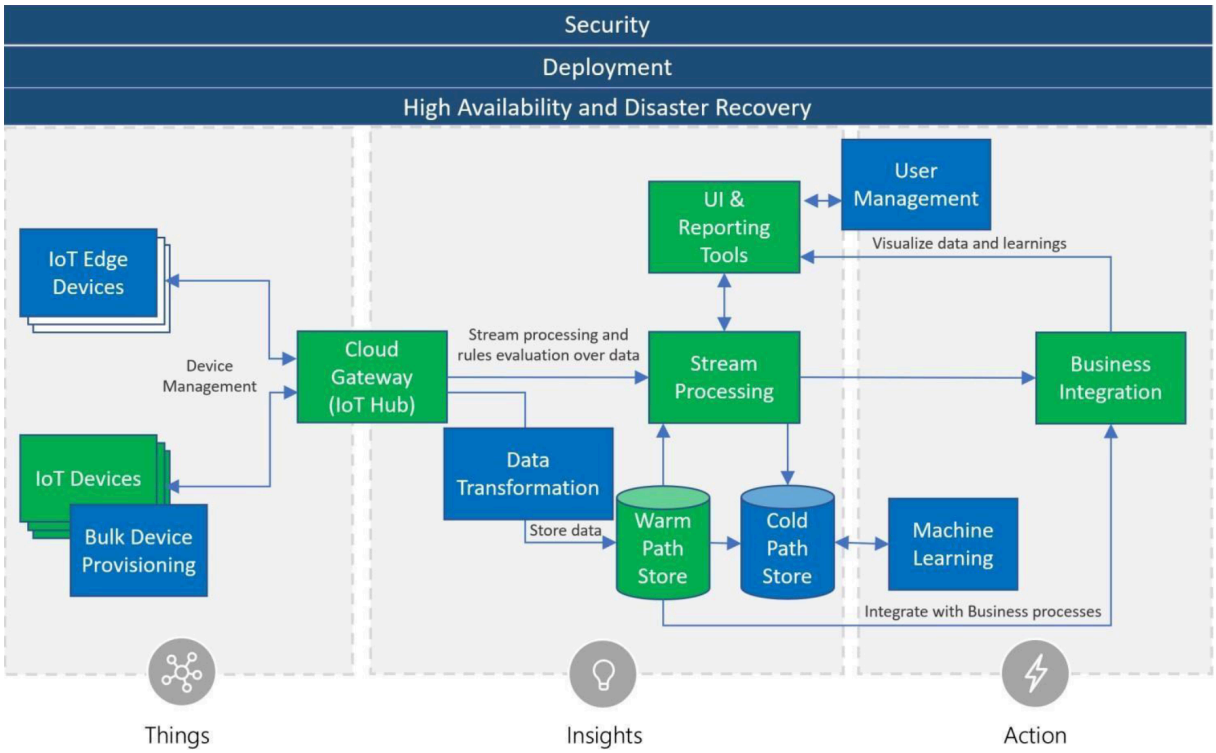

Figure 5.8: Azure IoT high-level architecture (source: `https://azure.microsoft.com/mediahandler/files/resourcefiles/microsoft-azure-iot-reference-architecture/Microsoft_Azure_IoT_Reference_Architecture_2_1_1_update.pdf`)

5.2.2 Things

Starting from the network edge, the "Things" category which groups IoT-enabled devices and edge devices, Microsoft has the ability to provision those devices in bulk. The provisioning procedure registers the devices in the system with identifiers and device details in great numbers. In contrast, OpenSCN requires a human to register devices manually in the system. This difference comes from the scale of operation for each solution. Azure is an industry-grade product that is expected to be used with commercially available IoT devices from well-known manufacturers. On the other hand, OpenSCN is designed to meet the needs of smaller-scale requirements in an academic environment, in which the IoT devices can be either commercial products or custom solutions. Therefore the complexities of solving the automatic provision of devices isn't considered as significant a need for the system at its current status.

5.2.3 Insights

The "Insights" category is the core of the IoT system and describes the procedures of the data ingress and storage with the "Cloud gateway" component as the connecting node. OpenSCN implements most of the "cloud gateway" features like authentication and authorization of devices and protocol translation, but each feature is less capable than its Azure counterpart. For example, our solution doesn't support COAP as protocol (it supports only HTTP and MQTT) or PKI as an authentication option (it supports devices authentication tokens). The rest of the system is comprised of data transformation, data stream, storage and monitoring. OpenSCN implements the data transformation to a uniform format that contains extra data and labels, the stream processing processes new IoT values for potential actions that should be taken and implement a UI and reporting user interface. Another difference is that Azure differentiates "Warm" and "Cold" storage of data. Microsoft refers to "Cold" storage as stale data which are not expected to be used actively by the system. It is an aggregated format of past data. There isn't a similar concept within OpenSCN. In contrast, "Warm" storage is data that should be easily and quickly available for processing. OpenSCN uses only that kind of storage in a time series database.

5.2.4 Action

The third category of Azure architecture is the "Action" category. Azure uses that layer to add functionality that can not be standardised like business rules and user access management along with more generic actions like machine learning model training for specific use cases. OpenSCN has a limited capability in this category and as in the "Devices" category, pushes these responsibilities to human action. User management is being done in the application as the reporting and the business rules are encoded by users using alert configurations and automatic actuator triggering. OpenSCN can't train machine learning models at this moment.

5.3 User facing Web client

5.3.1 Introduction

This section of the Thesis will showcase the application's client-side environment. We have built a dashboard based on the popular React framework focusing on users' ease of use and management, as

well as a clear display of important information from sensor metrics. All of the features offered by our web client application are outlined in the following subsections, accompanied by screenshots of the user interface components we used to make them available.

5.3.2 User identity

OpenSCN implements its own user identity in order to minimize dependencies on other software, but its modular design and the usage of common protocols allow integration with third-party services without much effort. OpenSCN implements a role-based authentication mechanism and relies on JSON web token (JWT) [72] for the user authorization, as per the examples given in Section 4.6.

Authentication is implemented using email and password pairs. Authorization with JWT tokens is a stateless way to share data among two or more parties. User identity is shared in JSON [73] format and encrypted using JSON web encryption [74]. Apart from the JWT standard fields, user identity is described with a set of custom fields that are called claims. These claims should be enough to uniquely identify a user (either using a public identifier or a reference key). In our case, we opt into sharing an identifier.

5.3.3 Authentication and Registration

Authentication is achieved by using an email and password check. The web client interfaces that are used for the user authentication workflow are the familiar ”Register”, ”Login” and ”Reset password” screens. The registration and reset password screens require an email-bound, single-use unique key generated by the OpenSCN service in order to be accessible. All the authentication pages follow the same visual pattern [Figure 5.9]. This key is delivered to the specified email.

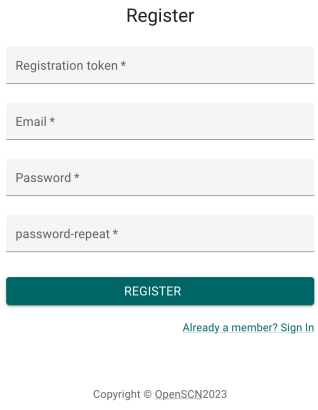

Figure 5.9: The registration screen. All the authentication pages are in a similar visual form. The user sees a form at the centre of the screen.

Registration to the OpenSCN platform is designed to be an invitation-only process. This allows for a more fine-grained access control process since OpenSCN is meant to be used for building specific situations

with a controlled subset of users. User invitations can be handed out by the admin user. In our current implementation, a user can be given admin rights only by the system administrators. Administrators of the platform can monitor active invitations by navigating to the ”User Invitations” page [Figure 5.10]. From there, they can obtain information about their creation and their current status. The admin user can choose to resend an invitation [Figure 5.10] and subsequently, the receiving party will receive their invitation token by email.

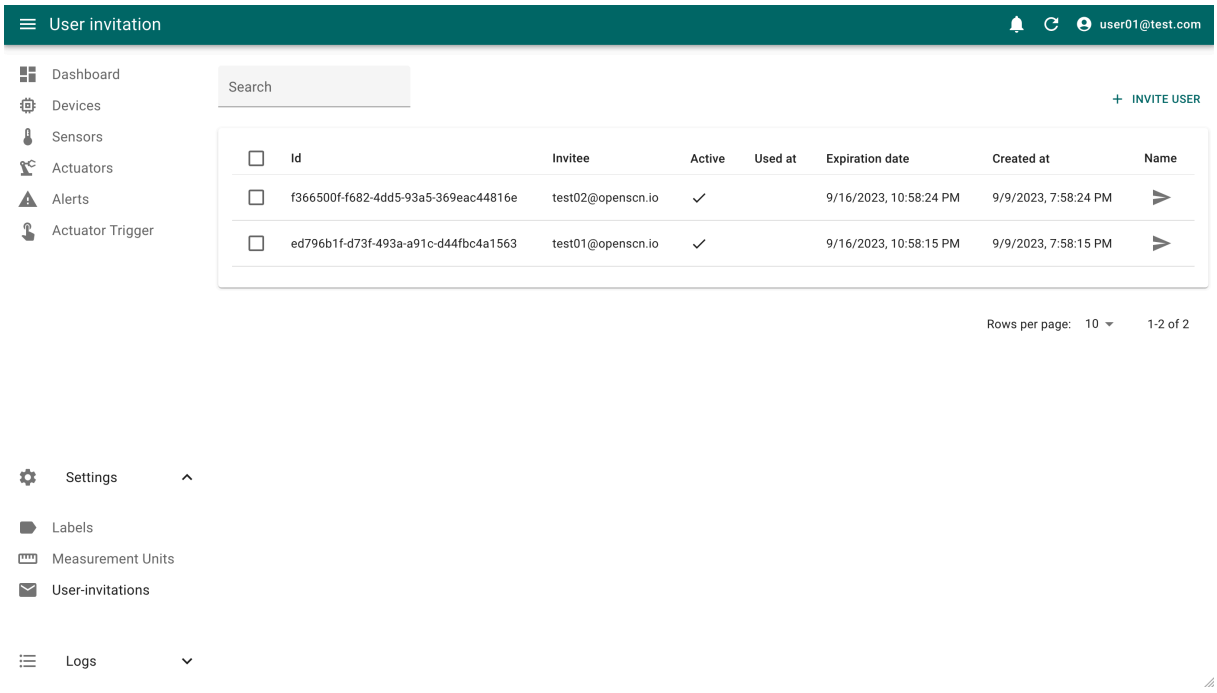

Figure 5.10: The user invitation screen. The main table is the list of user invitations with information about the invitation status. The user can create an invitation by using the ”Invite user” button at the top and right part of the screen. The send icon at the rightmost part of a table row can be used to send the invitation email again.

5.3.4 Dashboard and Data Monitoring

Onboarding dashboard

When a user hasn’t yet registered at least one IoT device and at least one sensor the main dashboard shows an onboarding [Figure 5.11] screen that prompts the user to register a device and a sensor. The onboarding screen has a welcome message with two prompts in the form of cards. The first is considered the one on the left of the screen and the second is the one on the right. The first describes to the user what an IoT device means to the system and provides a button that opens a drawer to create an IoT device. The second does the same for a sensor. After each step, the corresponding card changes its visual appearance to indicate that the step is completed [Figure 5.12]. The completion appearance makes the card more transparent, the action button is marked as “disabled” and a green check-mark icon is added at the right side of the button. After all steps are completed, the main dashboard changes its appearance and displays the label dashboard view.

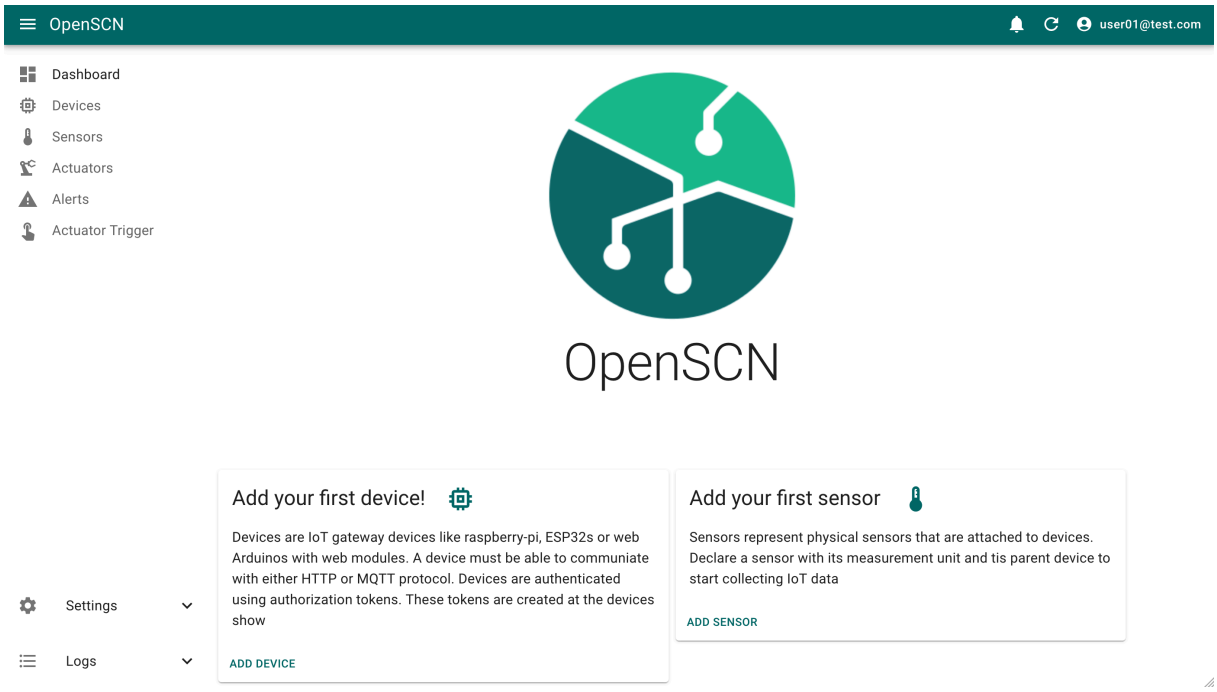

Figure 5.11: Dashboard with onboarding appearance. The user interface shows two cards with onboarding actions

Main Dashboard

The core of our user interface is the dashboard view (Figure 5.13), which is the first page presented to an authenticated user. Users can set up their dashboards - as will be described below- to view each of their desired sensor metrics and actuator status at a glance.

In order to present sensor measurements, we created custom graph components which allow users to zoom in and out in time and measurements. In particular, users can choose to be presented with metrics in a number of different time frames as well as hover over a particular point in time to focus on a specific measurement.

The dashboard is organized in a tabbed view, each tab containing sensors and actuators which are tagged with one or more labels. This was a choice with flexibility in mind, given that users can create any number of labels they desire and tag an actuator or sensor with any number of labels they want. This approach ensures there are no restrictions to the amount of different views a user can create to suit their needs. Our example in Figure 5.13 highlights a dashboard organized by different rooms in a hypothetical campus, each including that rooms' environment sensors and actuator controls. However, different types of tabs, such as grouping by sensor type, where for example all the campus' temperature measurements can be viewed at a glance, can be just as easily created.

5.3.5 Device Management

At the top of OpenSCN's data hierarchy sit "IoT Devices". We consider an IoT device that holds one or more sensors or actuators. In most practical scenarios, these are microcontrollers such as Arduinos/ESPs.

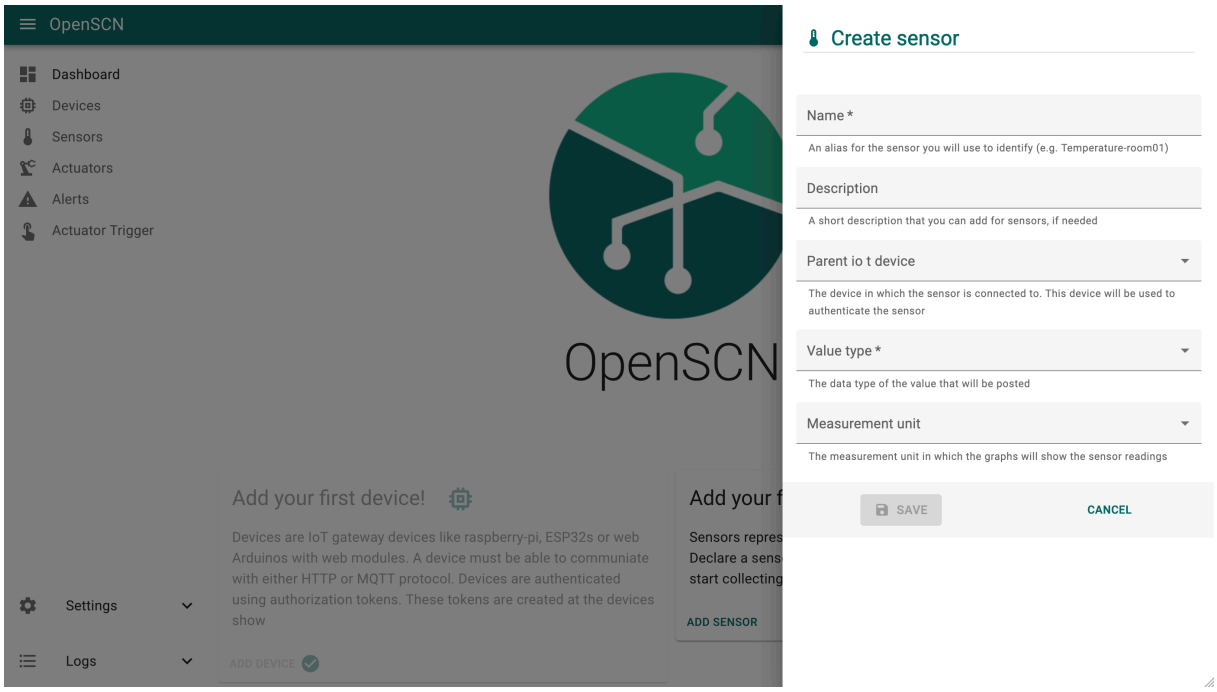

Figure 5.12: Dashboard with an onboarding step completed and the form for completing the next step open

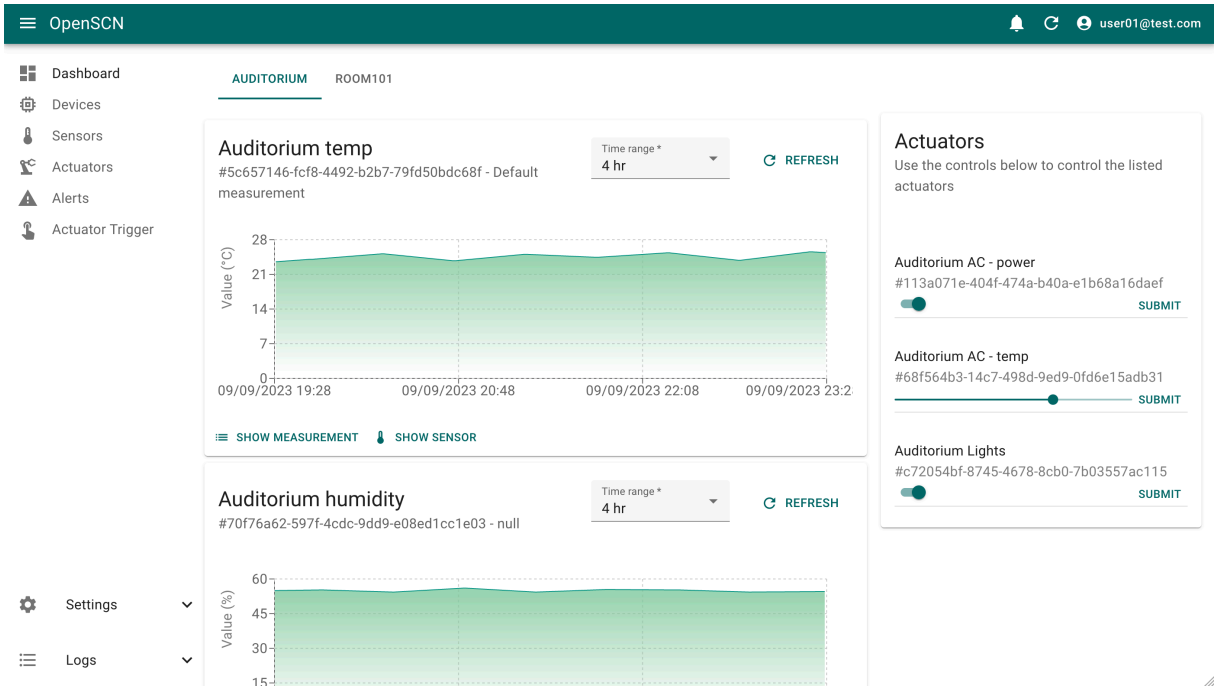

Figure 5.13: Dashboard View

Even though it is the sensors and actuators performing the data collection and submission in our flow, their parent devices are required to be registered in the application to provide the information needed in order to allow correct machine-to-machine identification and secure data transmission, as detailed in Section 4.6.

Clicking on the ”Devices” option from the left menu will navigate the user to the device list view (Figure

5.14), where all the devices along with the number of their attached sensors and actuators are outlined. A search field is provided to allow quick filtering of the list, as well as a button to swiftly create a new device registration. From this view, users can use quick access buttons to edit device information through a simple form or navigate to each device's detailed view.

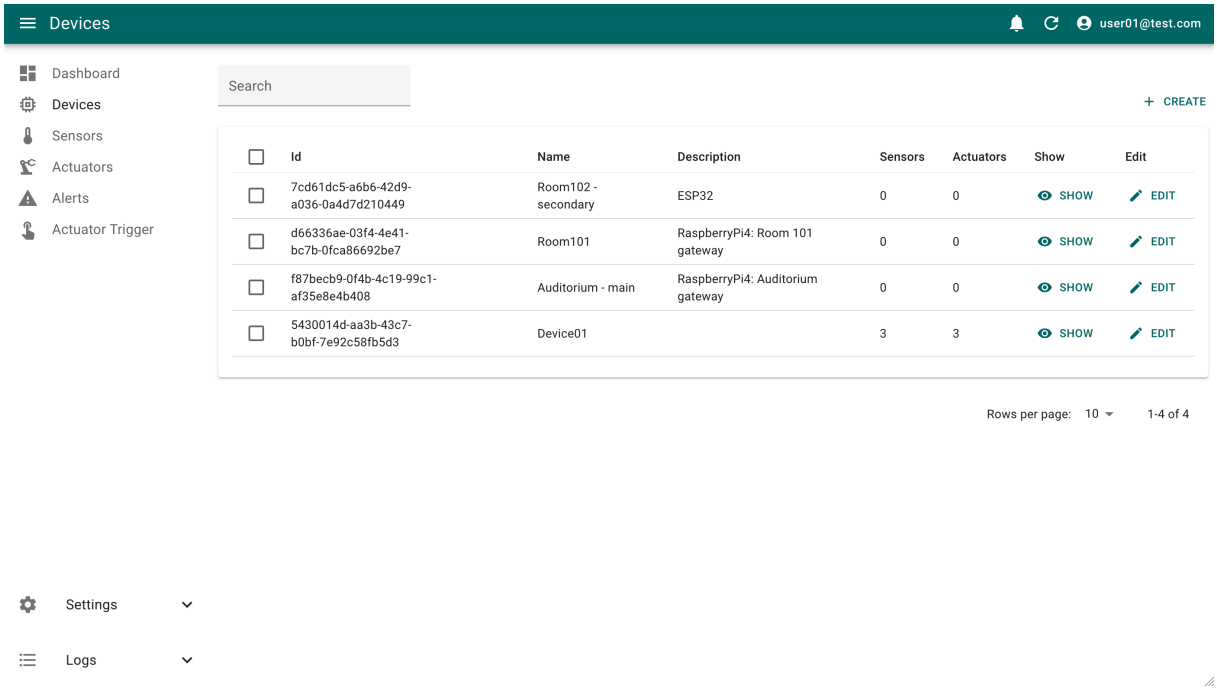

Figure 5.14: Device List View

The device details view (Figure 5.15) features a top bar of quick actions, allowing the users to easily register new sensors and actuators and automatically attach them to the selected device without leaving the page. Furthermore, apart from the deletion and edit options, a quick action to generate a new security token is also provided.

In the main view, a list of all the device details is highlighted, along with quick access to all the attached sensors and actuators. Lastly, users can view graphs of the attached sensors' data and manually post values on the device's associated actuators without leaving the device page.

5.3.6 Sensor Management

In the "Sensors" (Figure 5.16) page, users are presented with a list of all the sensors they have registered in OpenSCN together with their basic information. Additional filters are available, allowing users to search within the list by a sensor's name, description, parent device, value type or attached label. Quick actions for sensor editing and creation are provided along with a "Show" button which redirects users to the sensor's detailed view.

In order to create a new sensor registration, users are required to provide the device it will be attached to, along with information about the values it will be posting, meaning the type and the measurement unit they will be tagged with (Figure 5.17).

The sensor detail view (Figure 5.18) lists all of the sensor's information.such as their labels, their assigned

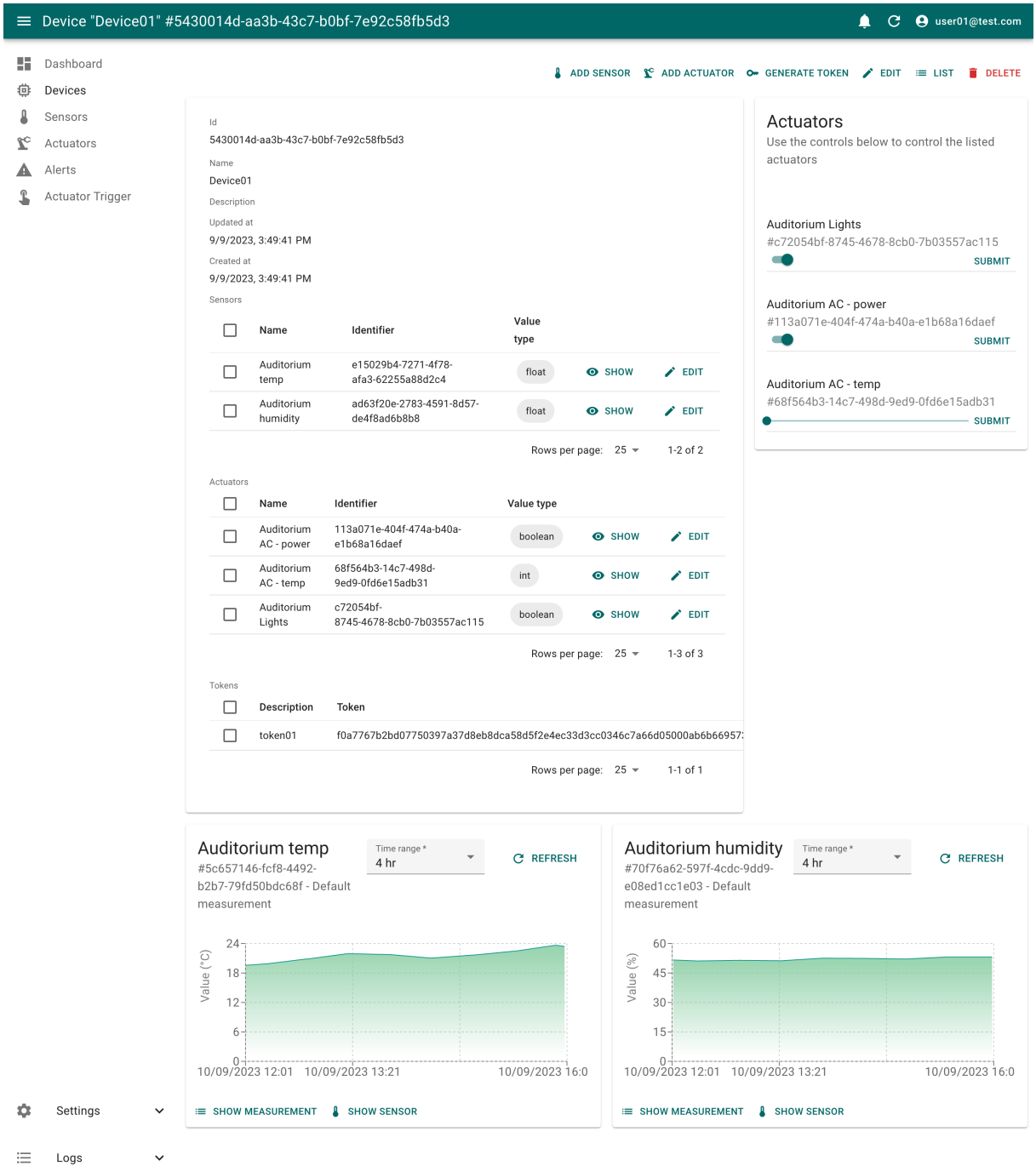

Figure 5.15: Device Details View

measurement units which are used to enrich the app's graphs with information and parent devices.

A piece of sensor information that requires its own reference is its list of "data collections". A data collection can be thought of as a bucket of collected metrics, where only one bucket can be active at a time per sensor. Having the option of multiple data collections can allow the user to create different environments for a specific sensor, which can be tremendously useful when it comes to sensor calibration and configuration testing. These procedures generate outlier data points which one would want to discard or ignore when it comes to data presentation and analysis, and having separate data collections can allow exactly that.

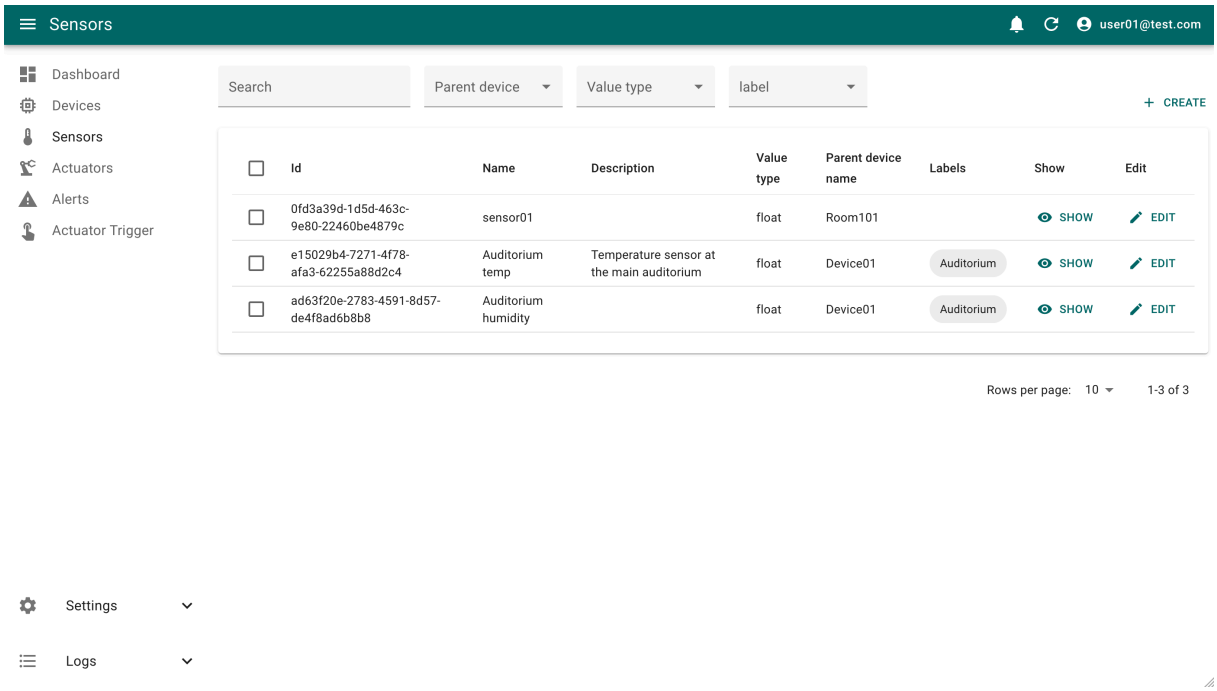

Figure 5.16: Sensor List View

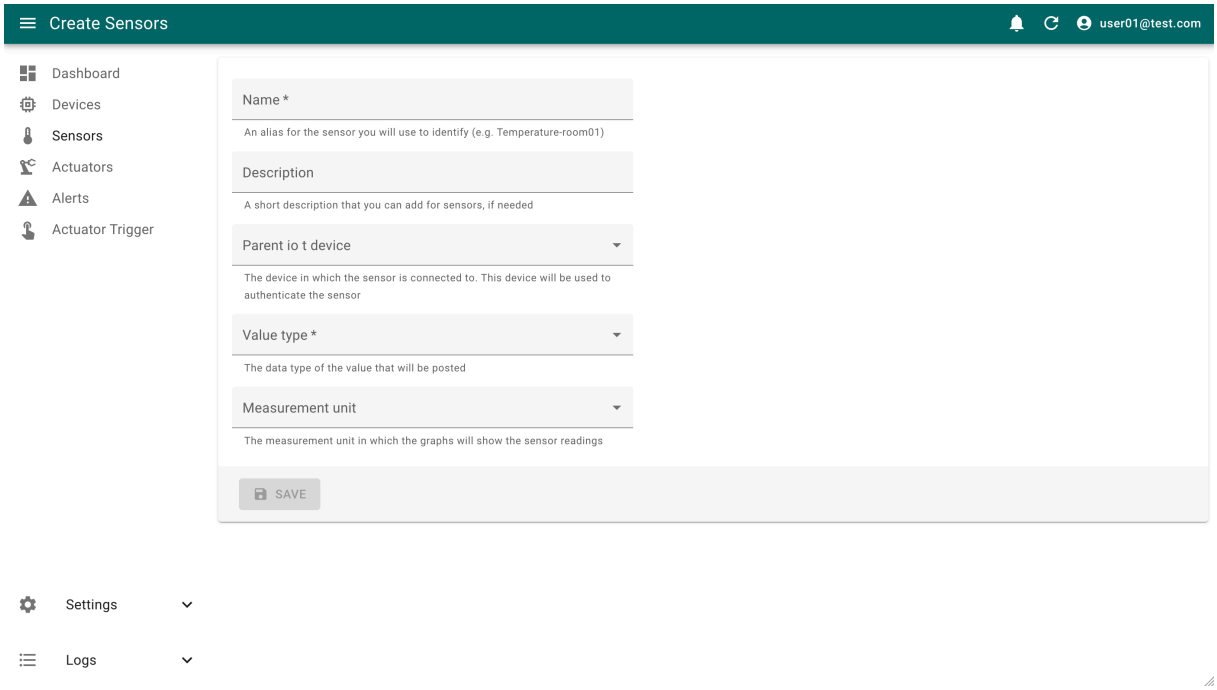

Figure 5.17: Sensor Create View

Another advantage of data collections is that they give the ability to separate measurements in different time periods for analytic purposes.

Additionally, another central part of sensor functionality that is presented to the user on this page is alert configurations. These are user-specified conditions that trigger notifications and will be analyzed further below in Section 5.3.8. The sensor detail view also features a quick action to create new ones.

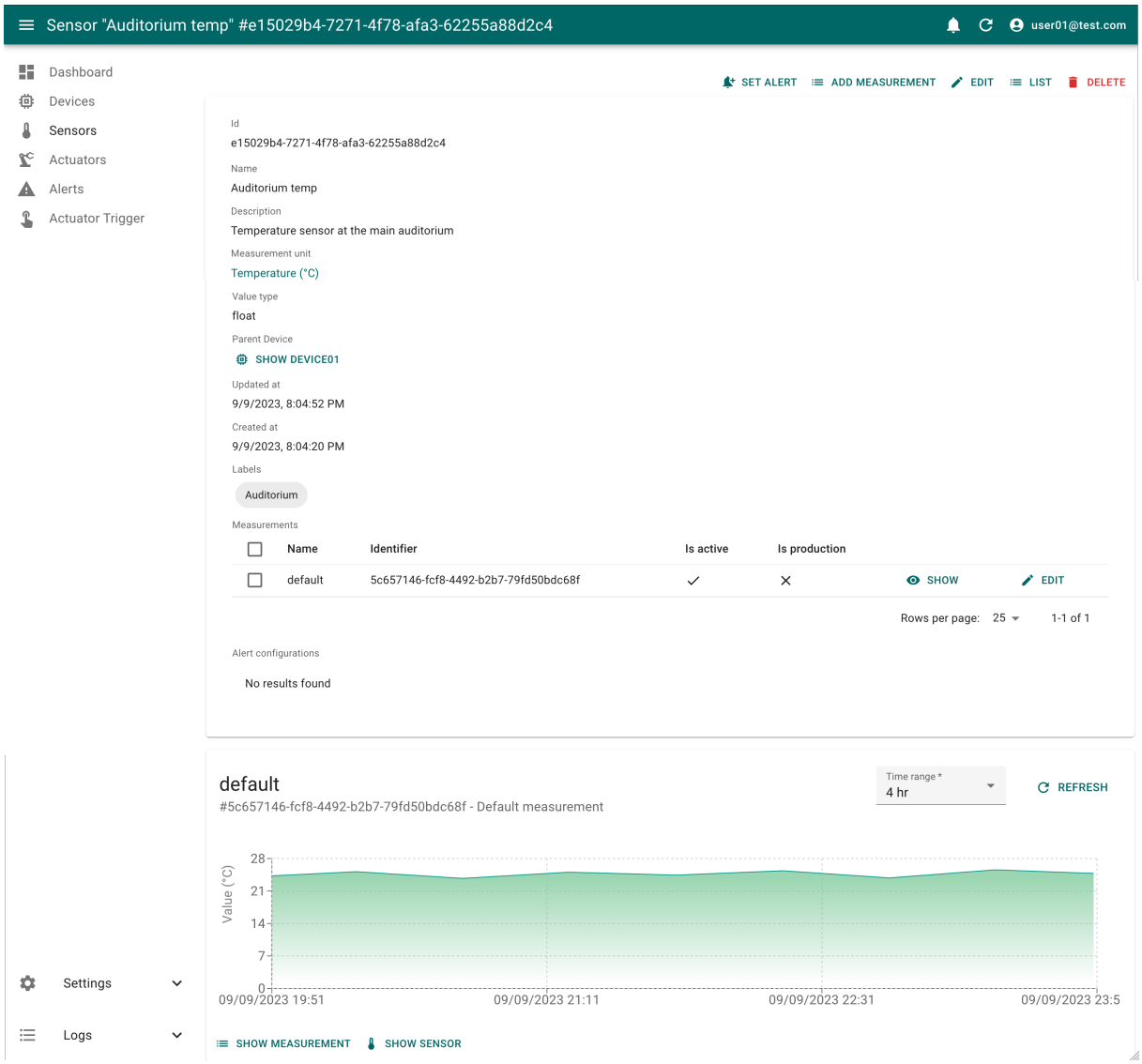

Figure 5.18: Sensor Details View

Finally, a graph of the sensor's measurements is presented to the user to provide an overview of its data collection.

5.3.7 Actuator Management

By clicking on the "Actuators" section of the main menu, users can navigate to the actuator list view (Figure 5.19). They are met with the familiar list UI along with filters to search by actuator name, description or parent device.

The "Create" quick action allows users to create new actuator registrations using a simple creation form (Figure 5.20). Some notable required information is a value type, a boundary inside which the actuation values will be posted, and the device it belongs to.

The actuator details view [Figure 5.21] features a top bar with quick actions that allow the user to navigate back to the main list view and quickly edit, delete or create a new data collection to store the posted values.

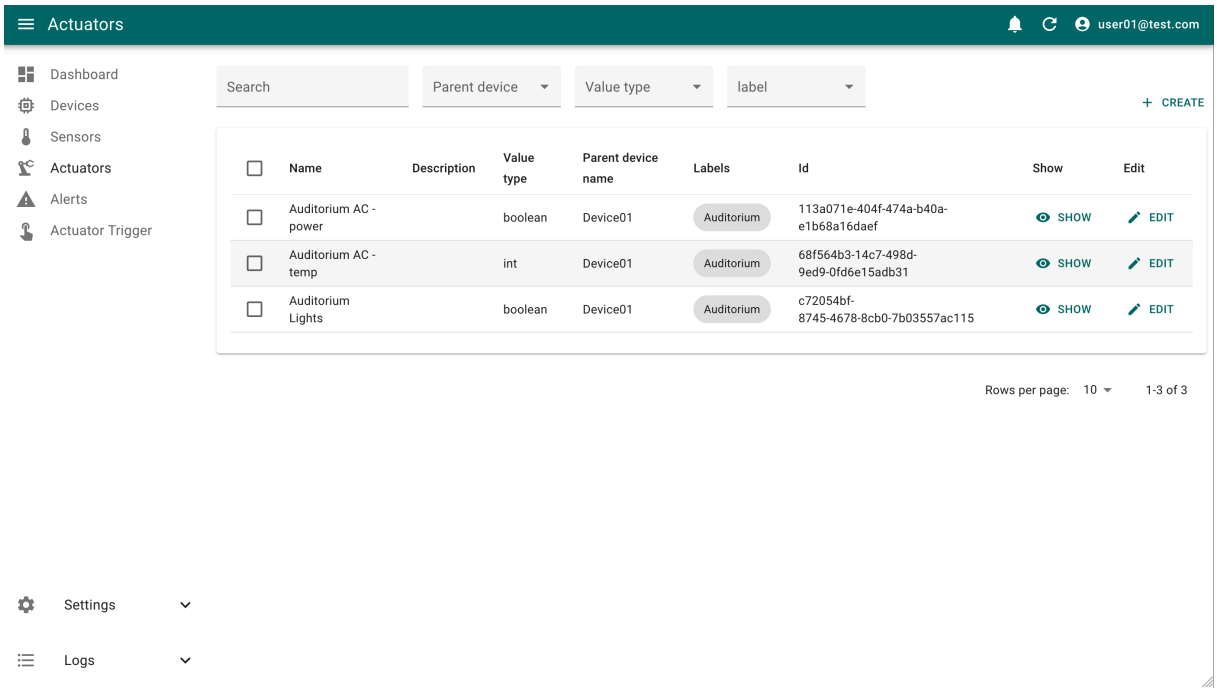

Figure 5.19: Actuator List View

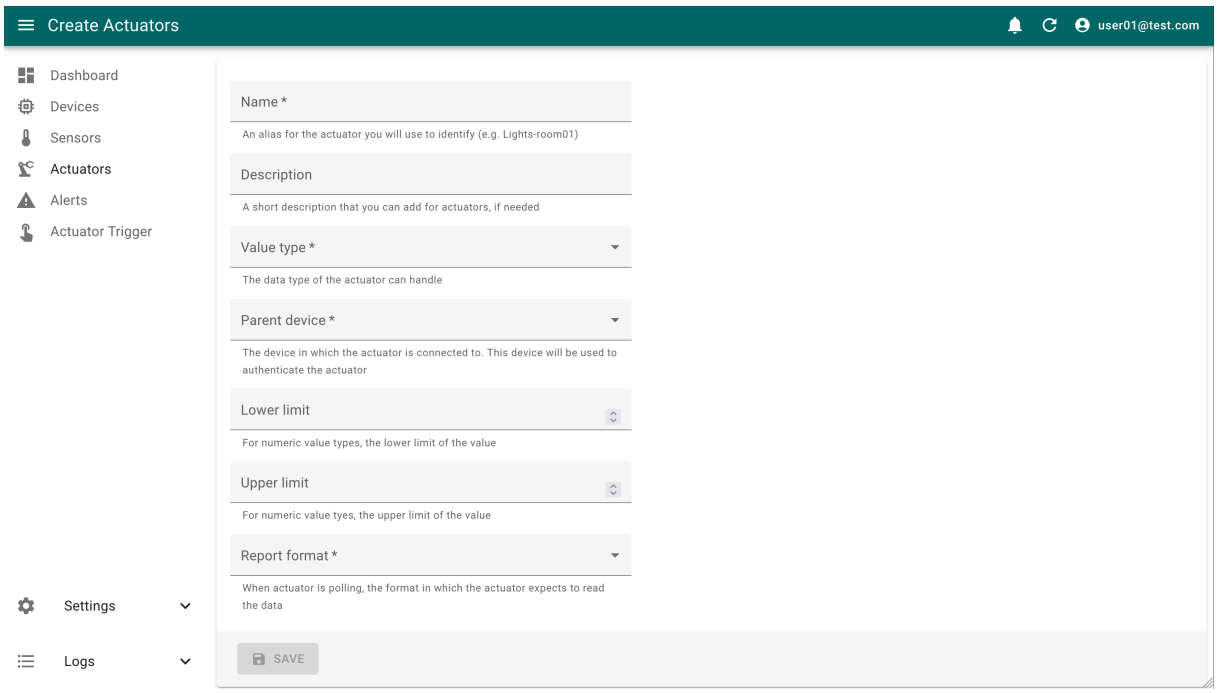

Figure 5.20: Actuator Create View

The main details components features a detailed overview of all the actuator's associated information. Furthermore, a control which matches the actuator's value type, meaning a switch for boolean values and a slider for numeric values is provided to manually post new ones.

Finally, a graph of all the previously posted values is presented.

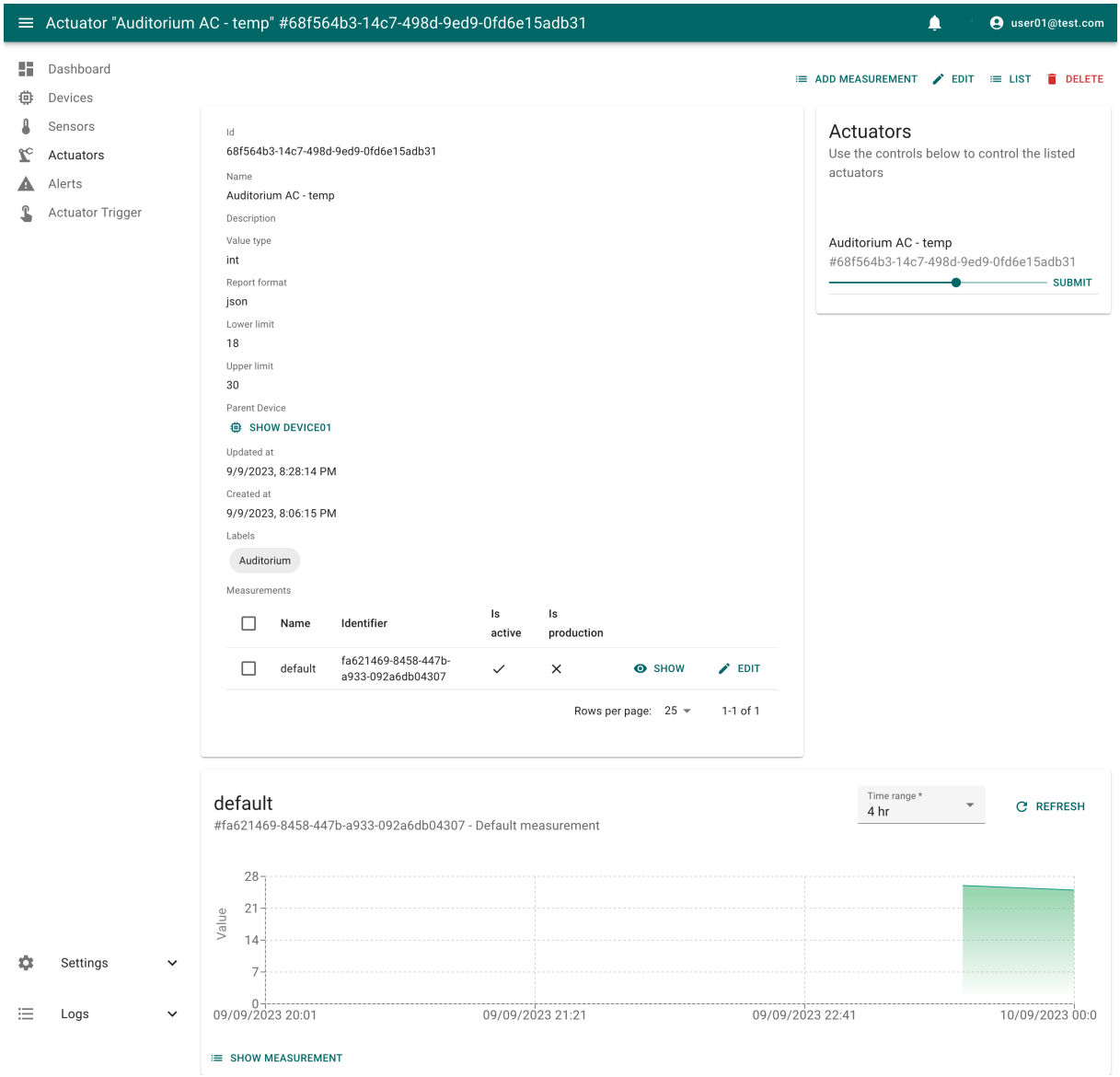

Figure 5.21: Actuator Details View

5.3.8 Alerts and Notifications

An integral part of any IoT data collection and monitoring application is being able to be notified in real time of critical states, and OpenSCN is no exception. To achieve this functionality, we use ”Alert Configurations”.

By creating an alert configuration, a user can define a ”Critical” and a ”Warning” status for each of their registered sensors. When the sensor registers measurements which are classified under these categories as defined by the users, they are sent notifications to be made aware.

The notifications panel (Figure 5.22) is accessed through the always available bell icon on the top right, which changes appearance when a new notification is ready, to further grasp the user’s attention.

As analyzed in Section 5.1.8, the user experience will not be slowed down no matter the amount of alert configurations, due to this load being assigned to our analytics and alert engine.

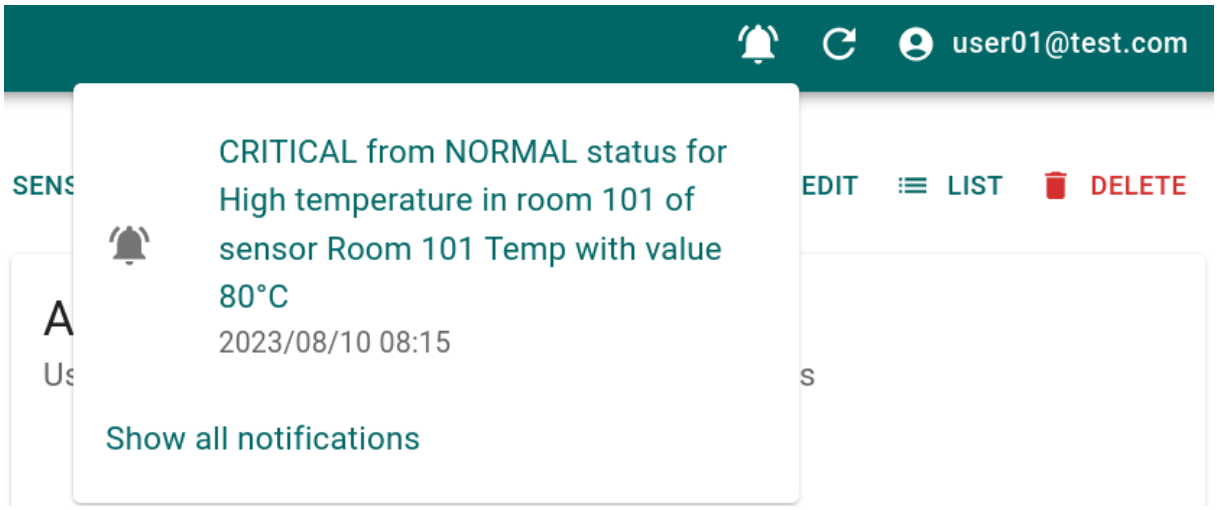

Figure 5.22: Notification Panel

In the main ”Alerts” page (Figure 5.23), the user can see a list of all registered configurations and filter by name, associated sensor or active status.

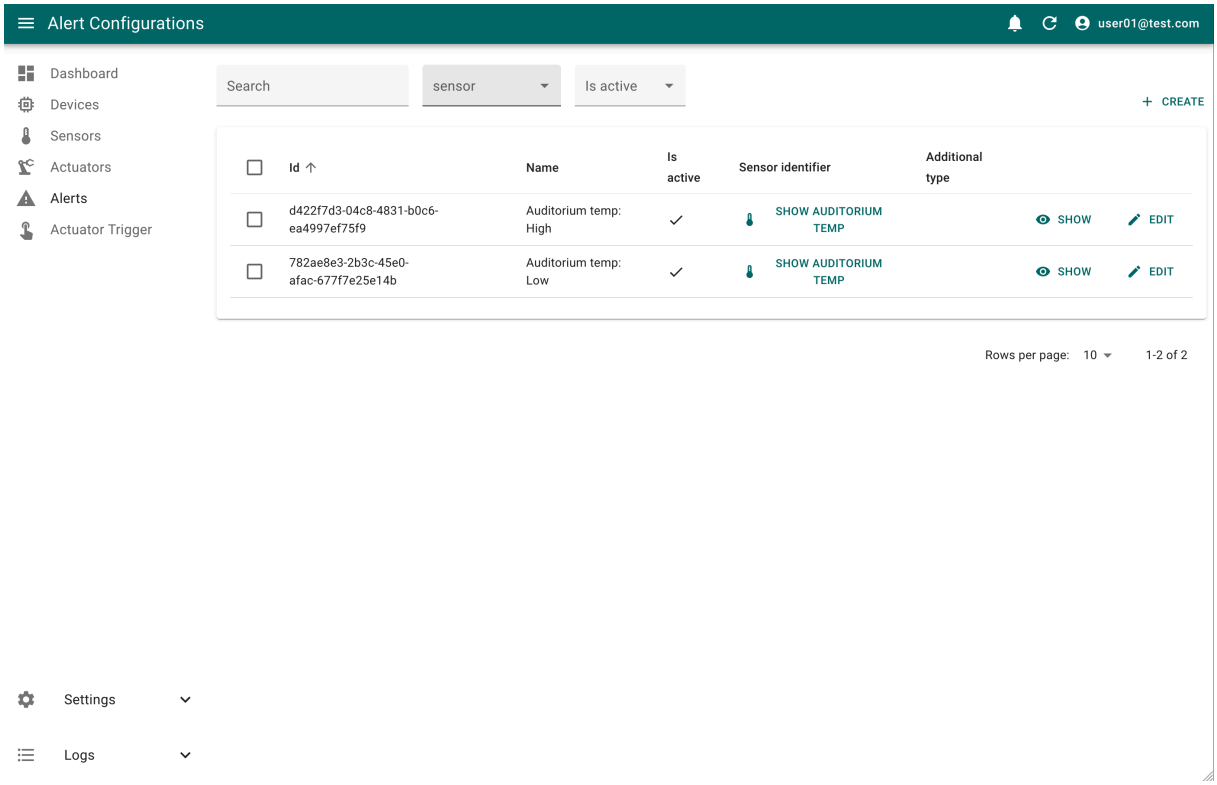

Figure 5.23: Alerts View

A quick action to create a new configuration is provided and the creation process can be seen in Figure 5.24. The user is asked to select the sensor to be alerted about. Furthermore, they provide a warning and critical measurement value and specify whether these values constitute upper or lower thresholds. The option to disable or enable an alert configuration is also available at any point in time.

Figure 5.25 showcases the detailed alert view, with quick actions to edit or delete the specified configuration.

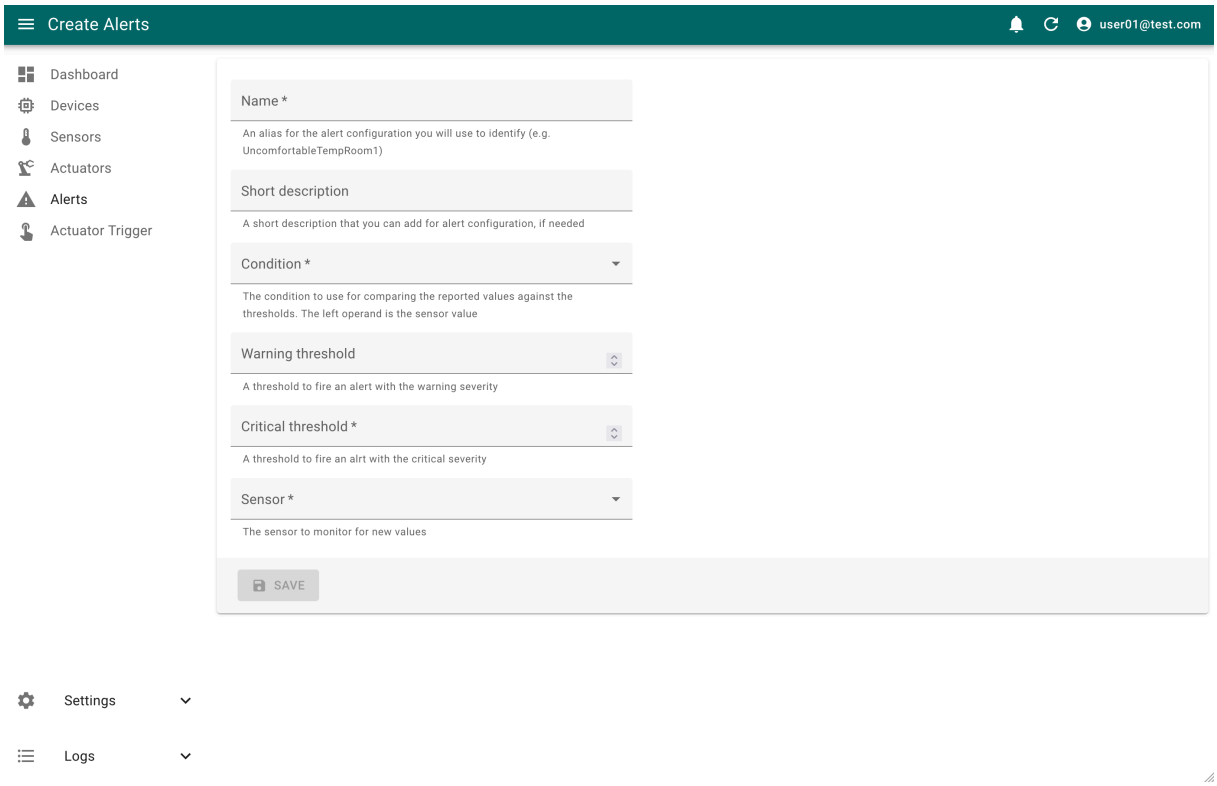

Figure 5.24: Create Alert View

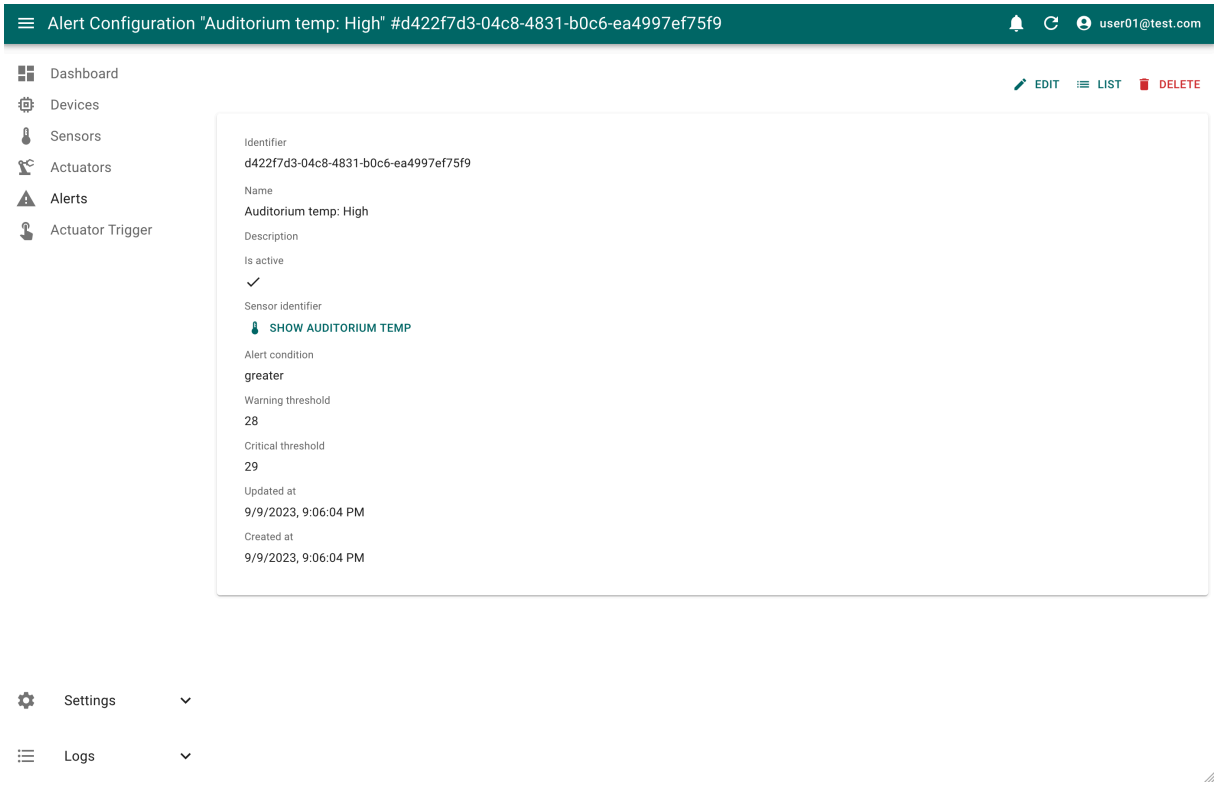

Figure 5.25: Show Alert View

5.3.9 Actuation Triggers

All of the previously mentioned actuation examples throughout this overview featured manual input from the user. Even though the offer of a dashboard readily available to control a plethora of smart devices in

a campus is useful by itself, the real value of an IoT infrastructure is having the ability to automate the aforementioned devices and react dynamically to changes in the campus environment.

In OpenSCN, this is made possible with the use of ”Actuation Triggers”. Actuation triggers allow the users to react to alerts produced by sensor measurements by posting specific actuator values as a response.

By selecting the ”Actuator Triggers” entry of the app’s main menu, the user will navigate to the actuator trigger list view (Figure 5.26). This view includes a list of all the previously created actuator trigger configurations. A search field is included as usual along with a filter to search configurations for selected actuators.

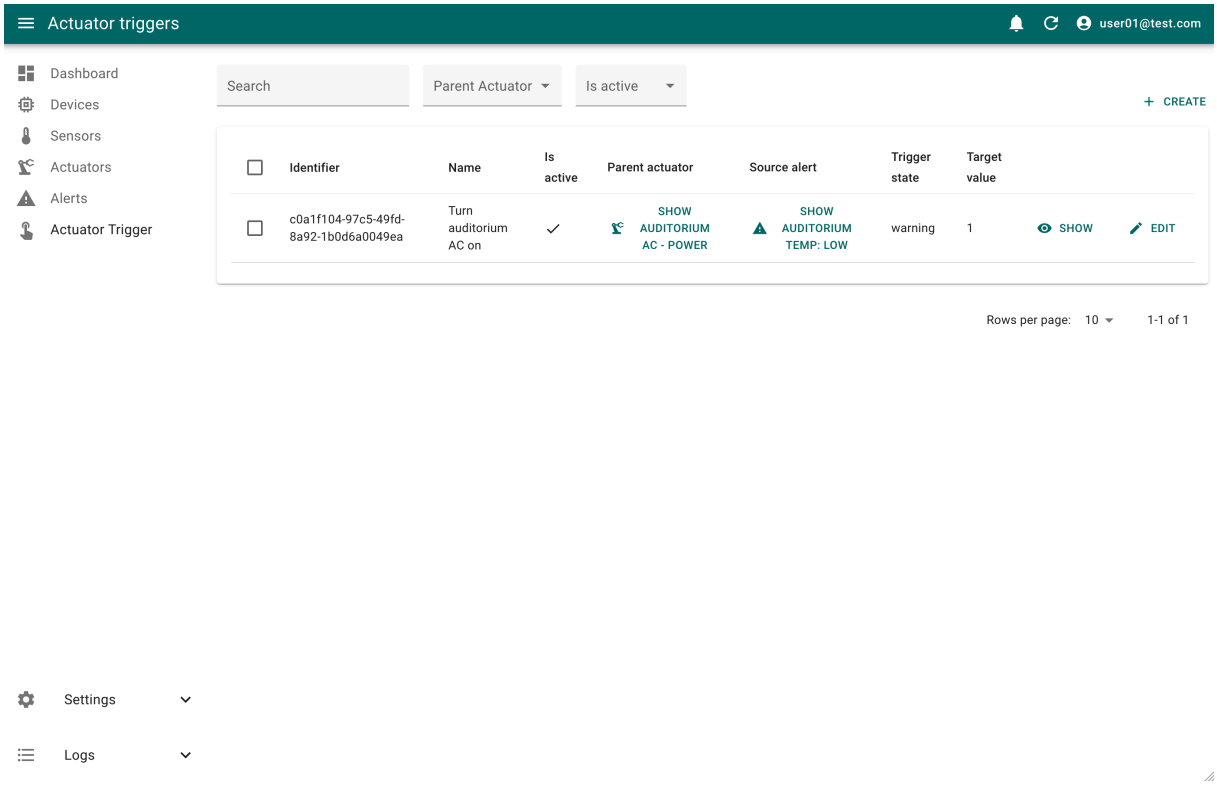

Figure 5.26: Actuator Trigger List View

Configuring new actuator triggers is made available by the ”Create” button. The trigger creation form (Figure 5.27) prompts the user to provide the alert configuration and the alert state to which the trigger will react, as well as the target actuator and the value that will be posted in response.

Clicking on the ”Show” button for the trigger list view will navigate the user to the actuator show view (Figure 5.28) where all the trigger configuration’s details are displayed.

5.3.10 Configuration and Logging

Further configuration options are provided in the ”Settings” sub-menu of OpenSCN’s main menu. In this section, users can opt into creating, editing or deleting labels (Figure 5.29), as well as configure all the measurement units being used for the app’s data tagging (Figure 5.30).

Lastly, a logs section for bookkeeping and monitoring purposes is provided. Users can choose to view and

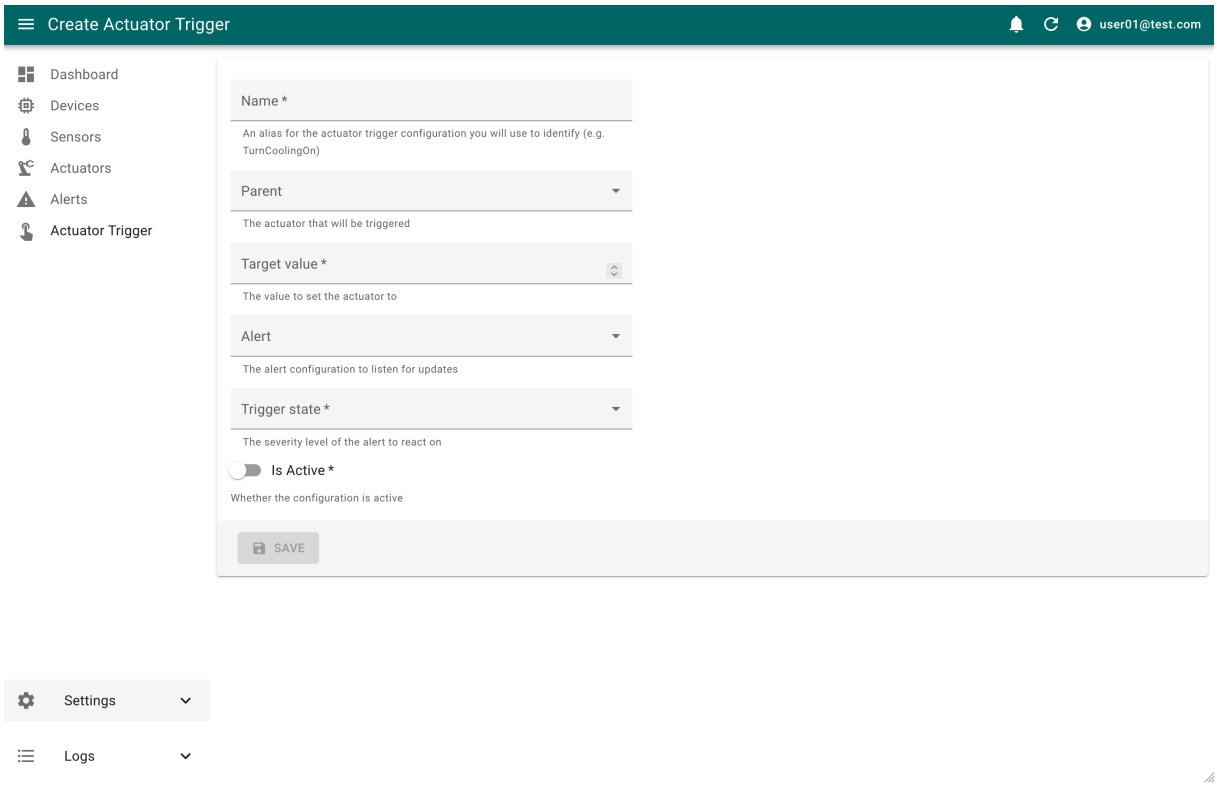

Figure 5.27: Actuator Trigger Create View

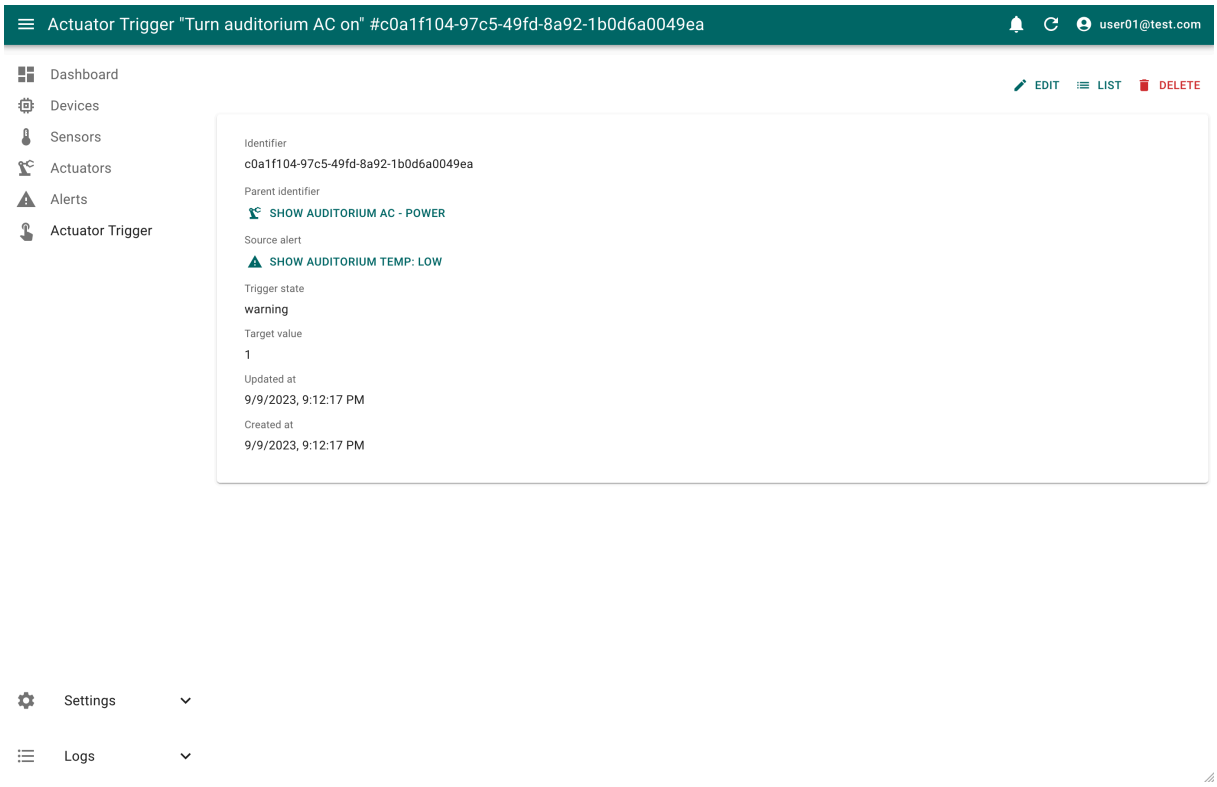

Figure 5.28: Actuator Trigger Show View

filter through all the app's recorded alerts (Figure 5.31), data collections, actuator triggers and notifications.

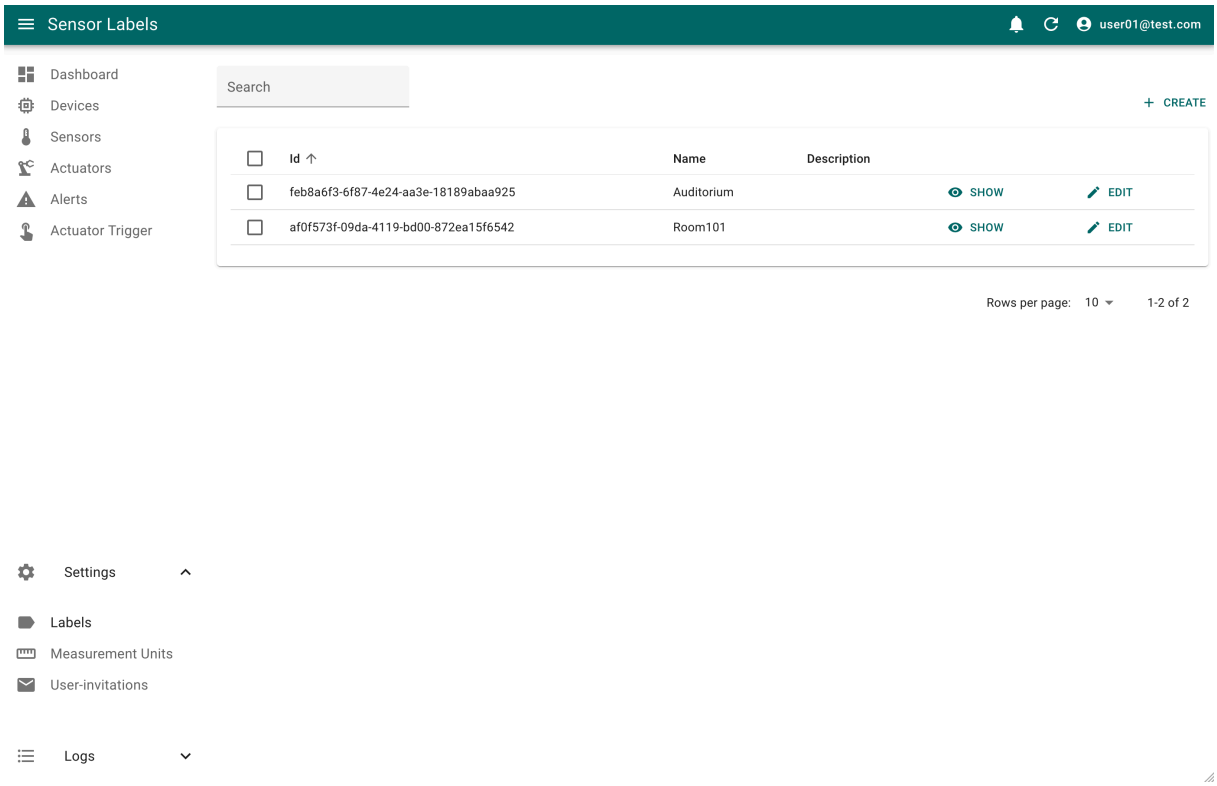

Figure 5.29: Labels View

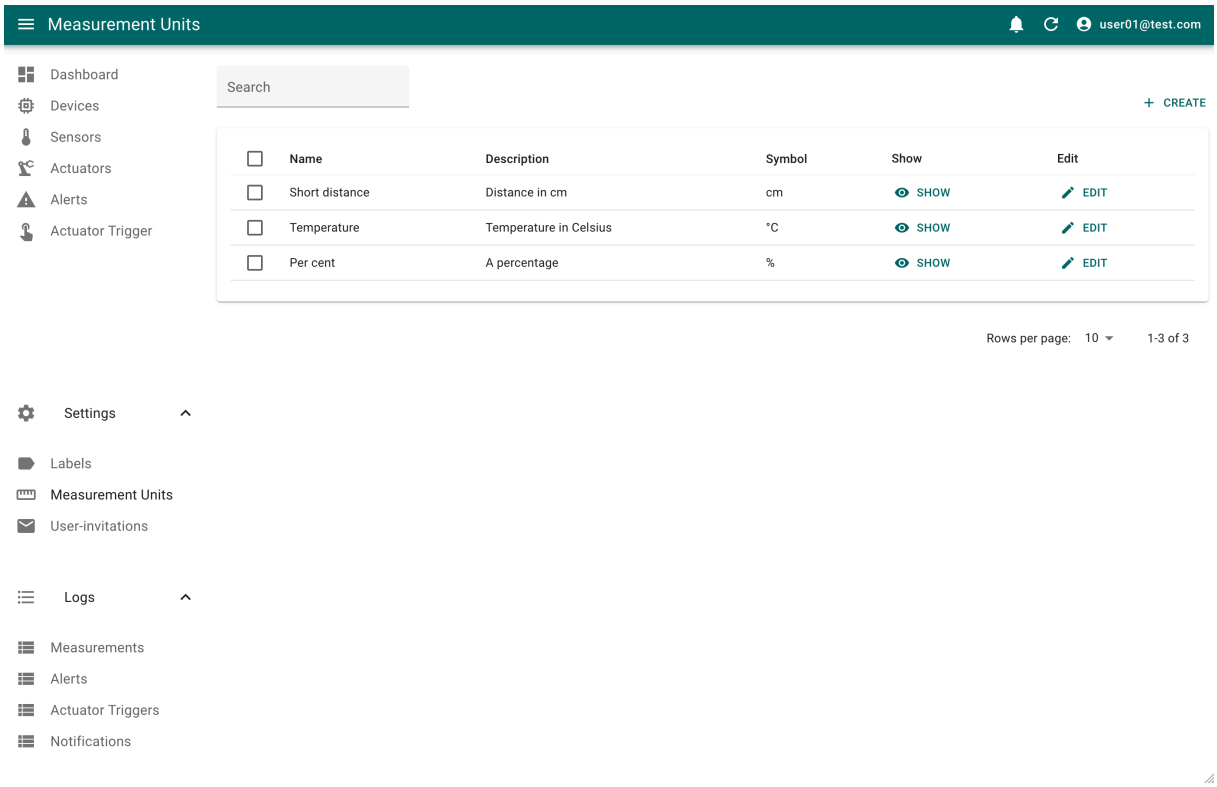

Figure 5.30: Measurement Units View

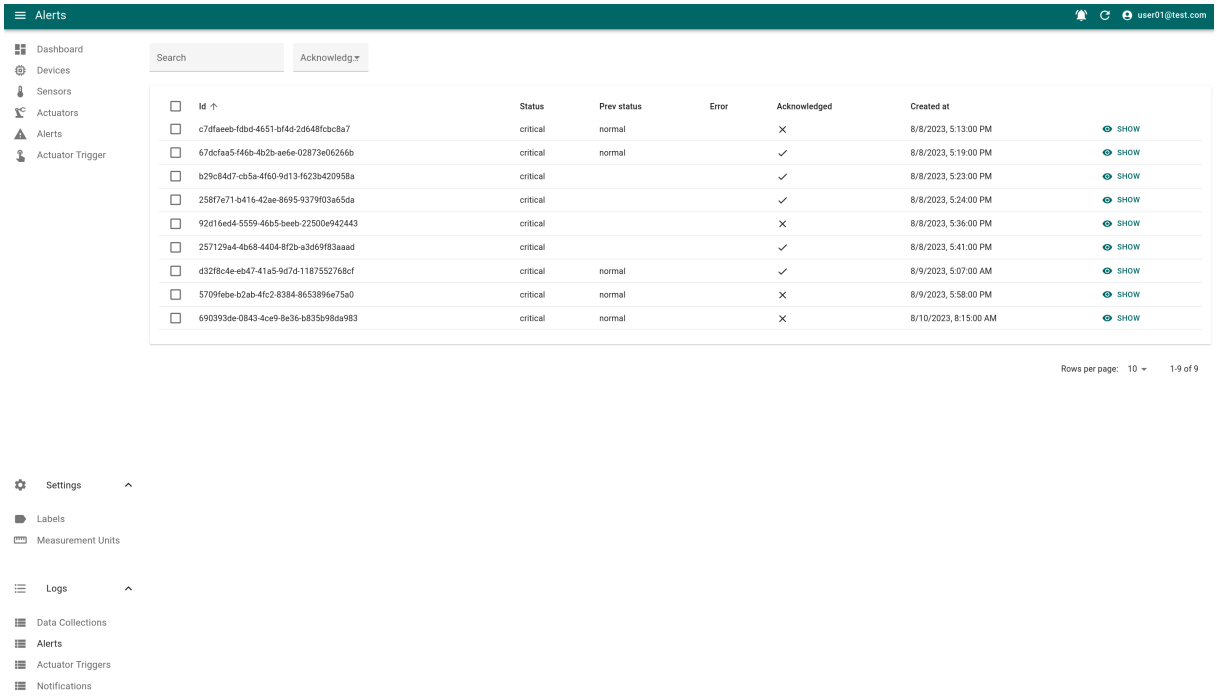

Figure 5.31: Alert Logs

Chapter 6: Conclusions and Future Work

6.1 Conclusions

In this Thesis, we embarked on a journey through the realm of IoT in the context of civilian infrastructure, exploring the fascinating domain of smart cities and their integration into educational environments, epitomized by smart campuses. Our exploration has revealed the numerous benefits that such smart implementations can bring, including sustainability, efficiency, improved quality of life, and economic growth. However, we also delved into the challenges and considerations that need to be addressed, such as data privacy and security, interoperability, governance, and inclusivity.

Through the lens of the smart campus concept, we discussed the critical components that constitute these intelligent educational environments. From smart infrastructure and IoT sensor networks to data management and analytics, we highlighted how these elements come together to deliver a diverse array of smart campus applications, ranging from energy management and enhanced security to personalized learning and facility management.

One of the pivotal contributions of the Thesis is the elucidation of the smart campus deployment process, along with the challenges and considerations that accompany it. We discussed the importance of a well-structured framework and introduced a robust methodological approach that could serve as a blueprint for future smart campus implementations.

Our journey also involved a comprehensive literature review of existing smart campus software solutions. We categorized these solutions into three primary types: small-scale solutions, those utilizing proprietary cloud platforms, and solutions based on platforms like FIWARE. While each of these approaches has its merits and drawbacks, we introduced OpenSCN, our very own open-source and user-friendly software platform tailored to the academic sector. OpenSCN aims to bridge the gap between complexity and accessibility, offering ease of use and flexibility for educational institutions.

In the foundational aspects of our smart campus software, we explored critical technologies and how we have utilized them within our platforms' architecture to enable a scalable and modern software solution.

In summary, the current Thesis has provided a holistic understanding of smart cities and smart campuses, introduced OpenSCN as a compelling solution tailored for academia, and presented an in-depth exploration of the technologies underpinning this platform. The challenges, benefits, and complexities of implementing smart campus solutions have been thoroughly examined, and we believe that OpenSCN, with its accessible approach and open-source nature, has a promising place in the ever-evolving landscape of educational technology. We hope that our work can provide a platform which will enable academic institutions to aid a transition of their facilities into smart ecosystems and encourage student collaboration on the way towards this task.

6.2 Future Work

While the current iteration of OpenSCN provides a robust foundation for smart campus solutions, we have identified some key areas which could be improved and be expanded upon to enhance the platform's

capabilities, expand its use cases, and make it even more accessible to a broader audience.

6.2.1 Time Series Analysis and Forecasting

One promising avenue for future work involves enhancing the capabilities of time series data within the OpenSCN platform. Currently, we have established robust time series data storage, which serves as a solid foundation. To further empower users and optimize decision-making processes, integrating time series analysis and forecasting tools could be valuable. This would enable users to gain insights from historical data, identify trends, and make data-driven predictions, thus enhancing the platform's predictive capabilities.

6.2.2 Composite Sensors and Data Aggregation

Expanding the platform's sensor capabilities is another area ripe for exploration. While the current system excels at handling data from individual sensors, many real-world scenarios require the aggregation of data from multiple sensors. By developing support for composite sensors and data aggregation, OpenSCN could provide users with a more comprehensive view of their environments. This enhancement would enable the platform to capture complex interactions and relationships among various data sources.

6.2.3 Smart Context Integration

The concept of "smart context" holds immense potential in augmenting the decision-making processes of IoT applications. Smart context encompasses a set of values representing the non-technical environment in which the application operates. It includes external measurements (e.g., weather forecasts), hard-to-measure facts (e.g., sun position), standardized knowledge (e.g., academic calendars), or even human decisions and procedures. Integrating smart context into OpenSCN can elevate the platform's decision accuracy by considering factors beyond raw sensor data. This could lead to more informed and context-aware decision-making.

6.2.4 Helm Charts Repository for Seamless Deployments

To simplify deployment processes and enhance user experience, establishing a Helm charts repository could be beneficial. Helm is a package manager for Kubernetes that streamlines the installation and management of applications. By providing pre-configured Helm charts for various components of the OpenSCN platform, users can deploy and scale their instances with greater ease. This repository would foster a more seamless adoption process and reduce the complexities associated with initial setup.

6.2.5 Machine Learning Model Training at IoT Workers

Expanding the capabilities of IoT workers to include machine learning model training is a significant step toward enhancing the intelligence of the platform. By integrating machine learning into the worker nodes, OpenSCN can continuously train and update models based on incoming data. This enables the platform to adapt, learn from changing conditions, and offer more advanced predictive and prescriptive

analytics capabilities. This extension would mark a substantial stride toward creating a self-learning and self-improving ecosystem.

These future work directions underscore OpenSCN's commitment to ongoing innovation and improvement. By pursuing these avenues, the platform aims to further empower users, increase its predictive prowess, and provide a more seamless and intelligent IoT experience.

BIBLIOGRAPHY

[1] Z. Kostepen, E. Akkol, O. Dogan, S. Bitim, and A. Hiziroglu, "A framework for sustainable and data-driven smart campus," pp. 746–753, 01 2020.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] N. Sharma, M. Shamkuwar, and I. Singh, "The history, present and future with iot," *Internet of things and big data analytics for smart generation*, pp. 27–51, 2019.

[4] F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. Leung, "Enabling massive iot toward 6g: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 11891–11915, 2021.

[5] V. Albino, U. Berardi, and R. M. Dangelico, "Smart cities: Definitions, dimensions, performance, and initiatives," *Journal of urban technology*, vol. 22, no. 1, pp. 3–21, 2015.

[6] M. Angelidou, A. Psaltoglou, N. Komninos, C. Kakderi, P. Tsarchopoulos, and A. Panori, "Enhancing sustainable urban development through smart city applications," *Journal of Science and Technology Policy Management*, vol. 9, no. 2, pp. 146–169, 2018.

[7] Z. Xiong, H. Sheng, W. Rong, and D. E. Cooper, "Intelligent transportation systems for smart cities: a progress review," *Science China Information Sciences*, vol. 55, pp. 2908–2914, 2012.

[8] T. Anagnostopoulos, A. Zaslavsky, K. Kolomvatsos, A. Medvedev, P. Amirian, J. Morley, and S. Hadjieftymiades, "Challenges and opportunities of waste management in iot-enabled smart cities: a survey," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 3, pp. 275–289, 2017.

[9] C. F. Calvillo, A. Sánchez-Miralles, and J. Villar, "Energy management and planning in smart cities," *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 273–287, 2016.

[10] D. Mutiara, S. Yuniarti, and B. Pratama, "Smart governance for smart city," in *IOP Conference Series: Earth and Environmental Science*, vol. 126, p. 012073, IOP Publishing, 2018.

[11] S. Giest, "Big data analytics for mitigating carbon emissions in smart cities: opportunities and challenges," *European Planning Studies*, vol. 25, no. 6, pp. 941–957, 2017.

[12] J. M. Shapiro, "Smart cities: quality of life, productivity, and the growth effects of human capital," *The review of economics and statistics*, vol. 88, no. 2, pp. 324–335, 2006.

[13] H. Hielkema and P. Hongisto, "Developing the helsinki smart city: The role of competitions for open data applications," *Journal of the Knowledge Economy*, vol. 4, pp. 190–204, 2013.

[14] A. P. P. Kasznar, A. W. A. Hammad, M. Najjar, E. Linhares Qualharini, K. Figueiredo, C. A. P. Soares, and A. N. Haddad, "Multiple dimensions of smart cities' infrastructure: A review," *Buildings*, vol. 11, no. 2, 2021.

[15] C. Yang, P. Liang, L. Fu, G. Cui, F. Huang, F. Teng, and Y. Bangash, "Using 5g in smart cities: A systematic mapping study," vol. 14, p. 200065, 02 2022.

[16] F. Calabrese, "Wikicity: Real-time location-sensitive tools for the city," in *Handbook of research on urban informatics: The practice and promise of the real-time city*, pp. 390–413, IGI global, 2009.

[17] Y. Yilmaz, S. Uludag, E. Dilek, and Y. Ayozen, "A preliminary work on predicting travel times and optimal routes using istanbul's real traffic data," 12 2016.

[18] L. van Zoonen, "Privacy concerns in smart cities," *Government Information Quarterly*, vol. 33, no. 3, pp. 472–480, 2016. Open and Smart Governments: Strategies, Tools, and Experiences.

[19] A. S. Elmaghraby and M. M. Losavio, "Cyber security challenges in smart cities: Safety, security and privacy," *Journal of Advanced Research*, vol. 5, no. 4, pp. 491–497, 2014. Cyber Security.

[20] K. Chaturvedi and T. H. Kolbe, "Towards establishing cross-platform interoperability for sensors in smart cities," *Sensors*, vol. 19, no. 3, p. 562, 2019.

[21] A. Gyrard and M. Serrano, "Connected smart cities: Interoperability with seg 3.0 for the internet of things," in *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 796–802, 2016.

[22] T. Nam and T. A. Pardo, "Smart city as urban innovation: Focusing on management, policy, and context," in *Proceedings of the 5th international conference on theory and practice of electronic governance*, pp. 185–194, 2011.

[23] C. H. Wang, E. Steinfeld, J. L. Maisel, and B. Kang, "Is your smart city inclusive? evaluating proposals from the us department of transportation's smart city challenge," *Sustainable Cities and Society*, vol. 74, p. 103148, 2021.

[24] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *ieee communications surveys & tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[25] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of lorawan for iot: From technology to application," *Sensors*, vol. 18, no. 11, p. 3995, 2018.

[26] M. Pasetti, P. Ferrari, D. R. C. Silva, I. Silva, and E. Sisinni, "On the use of lorawan for the monitoring and control of distributed energy resources in a smart campus," *Applied Sciences*, vol. 10, no. 1, 2020.

[27] A. I. Ali, S. Z. Partal, S. Kepke, and H. P. Partal, "Zigbee and lora based wireless sensors for smart environment and iot applications," in *2019 1st Global Power, Energy and Communication Conference (GPECOM)*, pp. 19–23, IEEE, 2019.

[28] K. E. Jeon, J. She, P. Soonsawad, and P. C. Ng, "Ble beacons for internet of things applications: Survey, challenges, and opportunities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 811–828, 2018.

[29] C. Petrie, S. Gupta, V. Rao, and B. Nutter, "Energy efficient control methods of hvac systems for smart campus," in *2018 IEEE Green Technologies Conference (GreenTech)*, pp. 133–136, IEEE, 2018.

[30] A. M. Eltamaly, M. A. Alotaibi, A. I. Alolah, and M. A. Ahmed, "Iot-based hybrid renewable energy system for smart campus," *Sustainability*, vol. 13, no. 15, p. 8555, 2021.

[31] W. Muhamad, N. B. Kurniawan, S. Yazid, *et al.*, "Smart campus features, technologies, and applications: A systematic literature review," in *2017 International conference on information technology systems and innovation (ICITSI)*, pp. 384–391, IEEE, 2017.

[32] A. Abuarqoub, H. Abusaimeh, M. Hammoudeh, D. Uliyan, M. A. Abu-Hashem, S. Murad, M. Al-Jarrah, and F. Al-Fayez, "A survey on internet of things enabled smart campus applications," in *Proceedings of the International Conference on Future Networks and Distributed Systems*, pp. 1–7, 2017.

[33] T. Anagnostopoulos, P. Kostakos, A. Zaslavsky, I. Kantzavelou, N. Tsotsolas, I. Salmon, J. Morley, and R. Harle, "Challenges and solutions of surveillance systems in iot-enabled smart campus: a survey," *IEEE Access*, vol. 9, pp. 131926–131954, 2021.

[34] M. A. Bouazzouni, E. Conchon, F. Peyrard, and P.-F. Bonnefoi, "Trusted access control system for smart campus," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pp. 1006–1012, IEEE, 2016.

[35] S. Qu, K. Li, S. Zhang, and Y. Wang, "Predicting achievement of students in smart campus," *IEEE Access*, vol. 6, pp. 60264–60273, 2018.

[36] Y. Zhang, C. Yip, E. Lu, and Z. Y. Dong, "A systematic review on technologies and applications in smart campus: A human-centered case study," *IEEE Access*, vol. 10, pp. 16134–16149, 2022.

[37] T. Sutjarittham, H. H. Gharakheili, S. S. Kanhere, and V. Sivaraman, "Experiences with iot and ai in a smart campus for optimizing classroom usage," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7595–7607, 2019.

[38] M. Benammar, A. Abdaoui, S. H. Ahmad, F. Touati, and A. Kadri, "A modular iot platform for real-time indoor air quality monitoring," *Sensors*, vol. 18, no. 2, p. 581, 2018.

[39] R. Gomes, H. Pombeiro, C. Silva, P. Carreira, M. Carvalho, G. Almeida, P. Domingues, and P. Ferrão, "Towards a smart campus: building-user learning interaction for energy efficiency, the lisbon case study," *Handbook of Theory and Practice of Sustainable Development in Higher Education: Volume 1*, pp. 381–398, 2017.

[40] S. Shrikhande, *Sac State Smart Campus Police: An application to provide predictive policing service to the Sac State campus*. PhD thesis, California State University, Sacramento, 2021.

[41] S. Fortes, N. Hidalgo-Triana, J.-M. Sánchez-la Chica, M.-L. García-Ceballos, J. Cantizani-Estepa, A.-V. Pérez-Latorre, E. Baena, A. Pineda, J. Barrios-Corpa, and A. García-Marín, "Smart tree: An architectural, greening and ict multidisciplinary approach to smart campus environments," *Sensors*, vol. 21, no. 21, p. 7202, 2021.

[42] M. Bortoli, M. Furini, S. Mirri, M. Montangero, and C. Prandi, "Conversational interfaces for a smart campus: A case study," in *Proceedings of the International Conference on Advanced Visual Interfaces*, pp. 1–5, 2020.

[43] P. Martins, S. I. Lopes, and A. Curado, "Designing a fiware-based smart campus with iot edge-enabled intelligence," in *Trends and Applications in Information Systems and Technologies* (Á. Rocha, H. Adeli, G. Dzemyda, F. Moreira, and A. M. Ramalho Correia, eds.), (Cham), pp. 557–569, Springer International Publishing, 2021.

[44] Í. França, N. Araújo, A. Gomes, N. Cacho, F. Lopes, J. Lima, and E. Adachi, "Sigoc: A smart campus platform to improve public safety," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pp. 1–6, IEEE, 2020.

[45] D. R. Brillinger, *Time series: data analysis and theory*. SIAM, 2001.

[46] R. S. Tsay, *Analysis of financial time series*. John wiley & sons, 2005.

[47] B. D. Santer, T. Wigley, J. Boyle, D. J. Gaffen, J. Hnilo, D. Nychka, D. Parker, and K. Taylor, "Statistical significance of trends and trend differences in layer-average atmospheric temperature time series," *Journal of Geophysical Research: Atmospheres*, vol. 105, no. D6, pp. 7337–7356, 2000.

[48] S. N. Z. Naqvi, S. Yfantidou, and E. Zimányi, "Time series databases and influxdb," *Studienarbeit, Université Libre de Bruxelles*, vol. 12, 2017.

[49] J. Lin, S. Williamson, K. Borne, and D. DeBarr, "Pattern recognition in time series," *Advances in Machine Learning and Data Mining for Astronomy*, vol. 1, no. 617-645, p. 3, 2012.

[50] P. H. Franses, "Seasonality, non-stationarity and the forecasting of monthly time series," *International Journal of forecasting*, vol. 7, no. 2, pp. 199–208, 1991.

[51] H.-S. Wu, "A survey of research on anomaly detection for time series," in *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 426–431, IEEE, 2016.

[52] K. Chooruang and K. Meekul, "Design of an iot energy monitoring system," in *2018 16th International Conference on ICT and Knowledge Engineering (ICT&KE)*, pp. 1–4, IEEE, 2018.

[53] B. Chen, Y. Liu, C. Zhang, and Z. Wang, "Time series data for equipment reliability analysis with deep learning," *IEEE Access*, vol. 8, pp. 105484–105493, 2020.

[54] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2019.

[55] S. Lee, H. Kim, D.-k. Hong, and H. Ju, "Correlation analysis of mqtt loss and delay according to qos level," in *The International Conference on Information Networking 2013 (ICOIN)*, pp. 714–717, 2013.

[56] M. J. Scheepers, "Virtualization and containerization of application infrastructure: A comparison," in *21st twente student conference on IT*, vol. 21, pp. 1–7, 2014.

[57] I. Docker, "Docker," *lınea]. [Junio de 2017]. Disponible en: https://www. docker. com/what-docker*, 2020.

[58] T. Kubernetes, "Kubernetes," *Kubernetes. Retrieved May*, vol. 24, p. 2019, 2019.

[59] J. Rufino, M. Alam, J. Ferreira, A. Rehman, and K. F. Tsang, "Orchestration of containerized microservices for iiot using docker," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1532–1536, IEEE, 2017.

[60] G. Ikrissi and T. Mazri, "A study of smart campus environment and its security attacks," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 44, pp. 255–261, 2020.

[61] C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou, and G. Vassilacopoulos, "Enabling data protection through pki encryption in iot m-health devices," in *2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE)*, pp. 25–29, IEEE, 2012.

[62] M. Dammak, O. R. M. Boudia, M. A. Messous, S. M. Senouci, and C. Gransart, "Token-based lightweight authentication to secure iot networks," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–4, IEEE, 2019.

[63] A. Bhawiyuga, M. Data, and A. Warda, "Architectural design of token based authentication of mqtt protocol in constrained iot device," in *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, pp. 1–4, IEEE, 2017.

[64] A. Bhawiyuga, M. Data, and A. Warda, "Architectural design of token based authentication of mqtt protocol in constrained iot device," *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, pp. 1–4, 2017.

[65] Schema.org, "Schema.org vocabulary," n.d.

[66] R. V. Guha, D. Brickley, and S. Macbeth, "Schema.org: Evolution of structured data on the web," *Commun. ACM*, vol. 59, p. 44–51, jan 2016.

[67] B. Erb, G. Habiger, and F. J. Hauck, "On the potential of event sourcing for retroactive actor-based programming," in *First Workshop on Programming Models and Languages for Distributed Computing*, PMLDC '16, (New York, NY, USA), Association for Computing Machinery, 2016.

[68] S. Tilkov and S. Vinoski, "Node.js: Using javascript to build high-performance network programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, 2010.

[69] G. K. Arora, "Solid principles succinctly," 2017.

[70] L. Nastase, "Security in the internet of things: A survey on application layer protocols," in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, pp. 659–666, 2017.

[71] Microsoft, "Microsoft azure iot reference architecture 2.1.1," 2023. `https://azure.microsoft.com/mediahandler/files/resourcefiles/microsoft-azure-iot-reference-architecture/Microsoft_Azure_IoT_Reference_Architecture_2_1_1_update.pdf`.

[72] Y. Sheffer, D. Hardt, M. Jones, “Json web token best current practices.” RFC 8725, 2020.

[73] T. Bray, “The JavaScript Object Notation (JSON) Data Interchange Format.” RFC 8259, Dec. 2017.

[74] M. B. Jones and J. Hildebrand, “JSON Web Encryption (JWE).” RFC 7516, May 2015.

[75] K. Srihari, V. Sakthivel, G. V. K. Reddy, S. Subhasree, P. Sankavi, and E. Udayakumar, “Implementation of alexa-based intelligent voice response system for smart campus,” in *Innovations in Electrical and Electronics Engineering: Proceedings of the 4th ICIEEE 2019*, pp. 849–855, Springer, 2020.

[76] J. L. Spudich and B. H. Satir, *Sensory Receptors and Signal Transduction.* New York: Wiley-Liss, 2001.

[77] A. S. A. E. Ouafa Bentaleb, Adam S. Z. Belloum, “Containerization technologies: taxonomies, applications and challenges,” *The Journal of Supercomputing*, vol. 78, no. 6, pp. 1144–1181, 2021.

[78] C. Barna, H. Khazaei, M. Fokaefs, and M. Litoiu, “Delivering elastic containerized cloud applications to enable devops,” in *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 65–75, 2017.

[79] J. Gao, P. Pattabhiraman, X. Bai, and W. T. Tsai, “Saas performance and scalability evaluation in clouds,” in *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE)*, pp. 61–71, 2011.

[80] “Gitlab public host.” `https://gitlab.com`.

[81] “Git web page.” `https://git-scm.com`.

Appendix A

The source code of the application is hosted publicly on GitLab [80] and is split into multiple Git [81] repositories under the OpenSCN group. The following list contains the links to the code that was developed and used for the current Thesis.

- GitLab group: `https://gitlab.com/openscn`
- Web API: `https://gitlab.com/openscn/openscn-web-api`
- IoT Worker: `https://gitlab.com/openscn/openscn-iot-worker`
- Web client: `https://gitlab.com/openscn/openscn-react-admin-client`
- Docker compose configuration: `https://gitlab.com/openscn/openscn-web-api/-/blob/master/docker-compose.yml`