energies

MDPI

Article

A Monitoring System for Electric Vehicle Charging Stations: A Prototype in the Amazon

Elen Lobato [1,*], Lucas Prazeres [1], Iago Medeiros [1], Felipe Araújo [1], Denis Rosário [1], Eduardo Cerqueira [1], Maria Tostes [1], Ubiratan Bezerra [1], Wellington Fonseca [1] and Andréia Antloga [2]

[1] Graduate Program in Electrical Engineering (PPGEE), Institute of Technology (ITEC), Federal University of Pará (UFPA), Campus Guamá, Augusto Corrêa, 01-Guamá, Belém 66075-110, PA, Brazil
[2] Norte Energia S.A, Brasília 70390-025, DF, Brazil
* Correspondence: elen.lobato@itec.ufpa.br

Abstract: Among the main problems faced in the context of electric mobility today, the management and monitoring of electric vehicle charging stations, the integration between the diverse types of technologies that make up its architecture, and its low scalability stand out. Therefore, we will present the implementation and complete integration of an electric vehicle charging system in an electric mobility pilot project being executed in the Amazon region in Brazil. Therefore, a literature review of related works will be presented, and its entire implementation will be addressed, from the charging infrastructure, through its back-end system and its Internet of things platform, to its front-end web system for monitoring charging stations. In addition, a complete prototype is created with a real testbed to verify the scalability of the implemented physical system. Based on the testbed evaluations performed, we observe that the implemented system performs well in receiving and sending data from up to 160 electric vehicle charging stations, achieving an average consumption of 26% for CPU and 95% for memory. In addition, it is important to mention that the deployed system supports horizontal scalability, enabling the connection of more charging stations and making it ideal for other integrated systems similar to ours. Based upon the main results obtained with the implemented system, the possibility of carrying out the management and monitoring of charging stations stands out; the integration of different technologies, from the back end and IoT middleware to its front end; a system that supports scalability, enabling the connection of more charging stations; and a reference architecture for charging station management and monitoring systems for the Amazon region.

Keywords: electric vehicles; open charge point protocol; IoT platform; Kubernetes

check for updates

Citation: Lobato, E.; Prazeres, L.; Medeiros, I.; Araújo, F.; Rosário, D.; Cerqueira, E.; Tostes, M.; Bezerra, U.; Fonseca, W.; Antloga, A. A Monitoring System for Electric Vehicle Charging Stations: A Prototype in the Amazon. *Energies* **2023**, *16*, 152. https://doi.org/10.3390/en16010152

Academic Editors: Muhammad Zeeshan Shakir and Islam Safak Bayram

Received: 19 October 2022
Revised: 20 November 2022
Accepted: 28 November 2022
Published: 23 December 2022

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Electric vehicles (EVs) can impact future transport systems' performance and are a promising option to contribute to energy diversification and reduction of greenhouse gas emissions [1,2]. EVs could consider information and communication technologies (ICT) to provide a new set of services and applications, changing urban mobility to smart electric mobility. In this way, ICT enhances the capabilities of EV systems, charging stations, monitoring and managing electric stations, energy storage systems, and photovoltaic systems, which are examples of smart electric mobility [3]. For instance, an EV system consists of EV charging stations and a central management system, enabling monitoring and managing the physical infrastructure. Hence, EVs and ICT are paving the way for a smart electric mobility era [4]. This migration from conventional urban mobility to smart electric mobility brings many sustainable and smart mobility benefits.

As the EV market grows, the demand for charging stations increases, which requires a new structure to manage these electric charging demands [5]. Most EV charging stations are decentralized and have a very complex structure, making managing and maintaining these structures a hard process [6]. In this sense, understanding the real-time status of a charging

station can provide valuable information to users and the system administrator, such as availability, reservations, and the time to arrive at a particular station [7]. For instance, EV charging stations must be online continuously since vehicle drivers/users typically charge their cars through an Internet-connected app. However, the central management system must be informed as soon as a charging station goes offline since understanding the real-time status of the EV charging stations can provide valuable information. In this context, ICT, such as the Internet of things (IoT) and cloud-based systems, enable the collection, storage, and analysis of data from EV infrastructure. Thus it makes EV charging more and more efficient by concentrating and analyzing data provided by the EV physical infrastructure [8,9].

With the use of IoT in the EV context, charging stations become smart, connected, and thus easily accessible for remote support and maintenance [10,11]. For instance, the data from EV charging infrastructure are collected through communication protocols [12], such as the Open Charge Point Protocol (OCPP) [13]. Specifically, the OCPP enables the collection of charging information, authenticates and authorizes users and validates the charging of EVs [14]. Therefore, the OCPP provides means to collect information for the management of EV charging stations, such as the status (online/offline) and charging variables (current, voltage, electric consumption, etc.). This information, in turn, needs to be sent and stored in a cloud-based system, to be later displayed by a software application for monitoring and management of the EVs charging stations.

In this context, the IoT platform must be deployed in the cloud to receive data from IoT devices deployed to collect EV physical infrastructure data [15]. Specifically, the IoT platform has emerged as the central point service for future IoT applications since it combines the benefits of IoT and cloud-based systems technologies [16,17]. At the operations center, analytic applications consume data from the IoT platform to process the measurement data for different purposes. These IoT platforms, such as Dojot [18], provide processing and storage resources for large volumes of data in data centers [19]. For instance, the IoT platform enables accessibility of information, which facilitates the optimization of charging processes or specific analysis of charging stations. Hence, the generated information from the charging stations can be easily managed and made available through several connectivity and processing possibilities that can be applied in the data collection and for the prediction of daily routine situations.

An efficient and scalable cloud-based system structure can generate several benefits for an EV system through greater value aggregation of the collected data. The benefits significantly impact the CAPital EXpenditure (CAPEX) and OPerational EXpenditure (OPEX) of EV applications. For instance, the amount of collected data is expected to increase since the electric mobility infrastructure, composed of EVs and charging stations, also tend to become more popular and economically viable.

Therefore, an IoT platform must scale in relation to the growing number of users, data, and computing areas, thus handling many services to customers in an essential virtual or physical space. Such IoT platforms must also perform data detection, aggregation, and persistence [20]. In this context, an application is said to be scalable as soon as it can resize its resources (CPU, memory, storage space, etc.) to meet a given workload that varies over time [21]. Among the possible scalability strategies for IoT platforms, we can cite vertical scaling and horizontal scaling [22]. The vertical scaling (or scaling up strategy) corresponds to adding more resources to an existing machine, such as more CPU or RAM. On the other hand, the horizontal scaling (or scaling out strategy) corresponds to adding more machines as more resources are needed [23,24]. For instance, the IoT platform could be deployed on a Kubernetes cluster, which relies on horizontal scalability with the addition of more worker machines for resource sharing to meet the increased demand [25]. In addition, Kubernetes enables the distribution of computing resources and the management of containers, as well as reduces the number of manual processes, among other advantages [26–28].

In this context, an EV central management system must consider an efficient communication protocol to collect data from the EV's physical infrastructure [6,14,29–32], have a

scalable IoT platform to handle the large volume of data collected by the physical infrastructure of the EV [16,21,33,34] and present the data in a front end, where it is possible to manage and monitor the charging stations [35,36]. However, integrating all these technologies is still a challenge, as the architecture of a central management system for EV charging stations may vary according to the scenario where the system is deployed [14,35–37].

To efficiently tackle the challenges described above related to the efficient integration of IoT communication and IoT platforms for monitoring and managing highly dense EV systems, we introduce the prototype of a real case study from charging to monitoring EV, called SIMA. The SIMA system consists of charging stations and electric buses, an OCPP back-end system, an IoT platform, and a web application (front end). Briefly, during SIMA operation, the charging station sends data via OCPP to a central management system, where the IoT platform is deployed in a scalable way to receive and store these data. We also introduce a web application that consumes the data from the IoT platform for monitoring all the charging stations. In addition, we analyze the scalability issue in the IoT platform (in terms of CPU and memory usage) by performing a real testbed evaluation involving EV loading data. Hence, the main innovation points of the implemented system consist of its scalability and end-to-end integration of all technologies used by the system, which is deployed in the Amazon region and is the first electric mobility system in this region.

Based on the testbed results, we conclude that implementing the IoT platform in a scalable way ensures that the entire SIMA system remains online and responds well, both for the back end and for the front-end application, regardless of the number of charging stations connected to it. Hence, we summarize the main obtained results with the implementation of the SIMA as: (i) the complete integration between the EV charging stations with the OCPP management system, the IoT platform, and a front-end web application; (ii) the deployment of a scalable IoT platform; (iii) a reference architecture for management and monitoring systems for charging stations for the Amazon region.

The main contributions of this article are stated as follows: (*i*) implementation and complete integration of OCPP to provide and access charging station and EVs communication protocol, an scalable IoT platform to provide an efficient way to store the generated data across multiple devices, and a dashboard website in the front end to exhibit the full readings of the distinct devices; (*ii*) the prototype of SIMA using the physical infrastructure for electric mobility located at the Federal University of Pará (UFPA), which enabled us to fully deploy and test our proposal. This infrastructure comprises electric buses, charging stations, photovoltaic systems, and energy storage systems, among others, within the context of a Research & Development Project for multimodal electric mobility funded by Norte Energia SA.

The rest of this article is structured as follows: Section 2 describes the main related works and topics associated with this article. Section 3 describes the proposed architecture of this work. Section 4 shows the main prototype and testbed of operations, and Section 5 presents and discusses the main results. Finally, Section 6 concludes the work and introduces future works.

2. Related Works

We divided the main related works into three categories: some works involving the OCPP use and EV charging station situations; other works that used IoT platforms for their proposals; and finally, some works that present integrated services or architectures related to EV process from charging to monitoring and also electric mobility pilot projects worldwide.

2.1. OCPP

Pruthvi et al. [6] reviewed the functionality that OCPP offers and how it can be used in the EV charging infrastructure. In this context, the authors demonstrated the implementation of a system based on OCPP, listing the main functionalities and associated messages typically implemented to provide basic functionalities to a charging station.

Alcaraz et al. [29] studied the main security properties of OCPP, e.g., communication between the charging stations and the energy management system. According to the authors, the use of protocol subversions, as well as communication with malicious entities, can lead to the destabilization of energy networks.

Garofalaki et al. [38] also addressed the security and privacy issues of an OCPP-based EV charging system. According to the authors, among the main OCPP security requirements in this EV loading context, one can cite integrity, authenticity, confidentiality, and availability in the context of EV driver information, EV data about the state of charge (SOC), the microgrid's energy, and the billing service process.

Antoun et al. [30] carried out a detailed safety assessment of the EV charging ecosystem. According to the authors, different entities are involved in the charging process depending on the location (i.e., public or private), which can compromise the system's security. In this context, they highlighted and categorized potential threats according to the vulnerabilities of each scenario.

Devendra et al. [31] designed and manufactured their own charging stations, based on the OCPP, for two-wheeled equipment (e-scooter). The authors discussed product design based on manufacturing, accessibility, and mass manufacturing capability. During the construction of these electric charging stations, they considered the security prerequisites for system administrators, installers, consumers, government agencies, and others.

Răboacă et al. [14] analyzed the OCPP specifications with an emphasis on application design based on the current state-of-the-art progress. Therefore, the authors provided an overview of hybrid and electric vehicles, a classification of electric vehicle charging station topologies, a classification of OCPPs, and considered future research directions.

Ruzmetov et al. [32] identified that the lack of information regarding available EV charging points in the streets might negatively impact the adoption of electromobility nowadays. The authors proposed a platform to ensure ongoing collaboration between the various entities involved: energy suppliers, charging stations, EV, and EV users. With this platform, the authors used such entities and proposed optimizing EVs' scheduling and allocation to the charging stations. The driver's chosen destination and the EV battery level are considered when their algorithm suggests some charging stations along the way and tries to ensure that the drivers are not distracted from their route.

2.2. IoT Platforms

Today there are several IoT platforms available for use, such as Apache Kafka, Amazon Web Services, Dojot, FIWARE, Google Cloud Platform, Konker, Sentilo, and ThingsBoard, among others, and they can be applied in several areas [21,39]. These IoT platforms share similarities when they use protocols, offer certain resources, and have analogous working principles. However, some of them are open sources, while others are proprietary. Proprietary IoT platforms are limited concerning their use, where some functionality and even performance are compromised [39]. Therefore, we focus on open-source IoT platforms applied in different areas.

Ottolini et al. [21] performed a qualitative and quantitative analysis of the interoperability and scalability of three IoT platforms: FIWARE, Konker, and ThingsBoard. These platforms were applied to two emulated IoT environments, one for smart city and the other for smart e-health. Each platform was deployed on two virtual machines with separate Amazon Web Services (AWS) infrastructures to assess how each platform manages its system resources. Another factor analyzed was the performance of each scenario with different workloads. According to the authors [21], FIWARE had the worst overall performance and crashed under high workloads. Konker presented the lowest degree of integration with other IoT-based devices and applications. However, the lack of integration did not reflect better performance. ThingsBoard, in turn, had the best overall results in terms of processing time and resource usage (i.e., CPU and memory). In addition, it also enabled better integration with different IoT protocols and third-party platforms. Based on the tests

and analysis performed, the authors concluded that the interoperability resources offered by each platform are not firmly related to their performance, and scalability [21].

Sinaeepourfard et al. [33] presented two case studies using Sentilo, another open-source IoT platform [40]. The first case study is described in the smart city of Barcelona, where almost 1800 sensors were installed throughout the city to collect different types of data. The measured data are sent to the Sentilo cloud platform, where they are categorized into energy, noise, urban, garbage collection, and parking lots. In the second case study, Sentilo is used to collect data from eight different pilot projects in Norway. In both case studies, a high amount of data is generated daily, requiring that Sentilo be a highly scalable platform. Due to this, Sinaeepourfard et al. [33] propose a centralized and distributed data management architecture that supports scalability.

Santos et al. [16] presented an IoT-to-Edge-to-Cloud platform, where their smart meter provided a prototype and testbed for smart grid collection. The authors developed their prototype and used it on two distinct university campuses, where they established an energy consumption analysis using their smart meters integrated into cloud analysis. Therefore, they proposed a low-cost way to obtain real-time energy consumption data using an IoT platform since such platforms are essential elements in storing and processing a large content of real-time data.

Silva et al. [34] described a smart campus project in Brazil that aims to transform the University of Campinas into a "living laboratory" and be a replicable model for other sustainable campuses. The work proposes a new methodology of ICT and various software, which are united into a platform. Such a platform improves the management of the campus through an innovative energy management tool based on IoT. Within this project, Dojot plays the role of an IoT platform between the sensors/collectors and the database. This makes it possible to export data for the full analysis integrated into the system or for offline use, according to the software's profile.

2.3. Integrated Services and Pilot Projects

There are still not many works in the literature describing the complete integration of an EV charging system, from the back end to the front end, similar to the one proposed in this article. Among the works found is that of Răboacă et al. [14], who analyzed the OCPP specifications with an emphasis on application design based on the state-of-the-art progress. Therefore, the authors provided an overview of hybrid and electric vehicles, a classification of electric vehicle charging station topologies, a classification of OCPPs, and considered future research directions. They proposed integrated services involving all the charging processes; however, they did not analyze the scalability of their work.

Ravindran et al. [35] presented a proposal for hardware and software, following Indian traffic patterns. Regarding the hardware part, the authors created what they call Electric Vehicle Supply Equipment (EVSE), based on the OCPP, which can be inserted into charging stations. Regarding the software part, the authors developed an application for EV users and a charging station management software. The work as a whole shows a scenario very similar to ours, but it is focused on Indian standards, and, therefore, is not replicable in other scenarios.

Devendra et al. [36] proposed an integrated architecture that mixes the vertical and horizontal flow of information. For this, the authors used an already existing platform, oneM2M, which will act as a middleware to receive charging requests and handle them. They present three different situations where it would be used: authentication process (to confirm the user identity before using the charging), billing process (to price the charging correctly), and status update (in the cloud). This proposal would be a way to improve the flow of information to the client.

In relation to electric mobility pilot projects developed around the world, similar to SIMA, some works are found in the literature [37,41,42]. Dornberg et al. [41] presented preliminary results of several e-mobility projects supported by the German government in terms of a field test of communication between a vehicle and the network during a charging

session. In addition, this work also presented an ICT solution called Energy Name Service (ENS) and a business model generated from the test performed. This work introduced a use case of an intelligent reservation system that allows communication between the e-mobility service consumer, the service provider, and the EV itself. Preliminary field test results indicated that ICTs enable EV communication with the network and increase consumer acceptance of electric mobility services [41].

To the best of our knowledge, Zero Emission Mobility for All (ZEM 2 ALL) [37] is one of the world's largest electric mobility projects ever developed, deployed in Malaga, Spain, for four years and ended in 2016. The ZEM 2 ALL consisted of an infrastructure of 200 vehicles, 220 conventional charging points, and 23 fast chargers. All these infrastructures were integrated into an ICT platform, where an application was developed that integrated navigation, reservation of fast chargers, demand response programs, records and eco-comparisons, a project intranet, dedicated email, and a web portal, among others.

Fabbri et al. [42] reported about the Bonifica 2.0 project developed in Pontine, Italy, which results from the Integrated Territorial System of Sustainable Mobility and Micro Smart Grids developed by the Sustainable Mobility Pole (POMOS) and its partners. This work utilized a module with several I/O interfaces, such as RS232 and CAN (controller area network) protocols, which are used to manage various charging systems (fast and slow), and also introduced a prototype of an electric boat navigating in shallow waters.

2.4. Concluding Remarks

Based on the analyzed works, IoT platforms are mandatory to collect, store, and send data in smart electric mobility scenarios. These data can be accessed/used through APIs by various types of applications (i.e., front end) that, for the most part, are intended for monitoring the physical infrastructure. In addition, it was also observed that due to the large amount of data that many sensors can generate, it is essential that these platforms be scalable.

Table 1 summarizes our analysis of the state of the art with the main approach used by their authors and the drawback regarding their proposals to work on a smart electric mobility scenario. To the best of our knowledge, as shown in our analysis presented in Table 1, none of the previous work integrated an efficient communication protocol to collect data from the EV physical infrastructure, together with an efficient and scalable IoT platform to deal with the large volume of data collected by the EV physical infrastructure, and an efficient data visualization on the front-end, while only SIMA combines every critical feature previously mentioned not provided by existing solutions. SIMA also stands out as a pioneer pilot project of electric mobility in the Amazon region, which is a key region that must address the use of eco-friendly energy use and enable sustainable projects. Thus, highlighting the innovation of this work through the use and integration of the technologies utilized will be described in the rest of this work.

Table 1. Summary of related works.

Work	Techniques	Drawback
Pruthvi et al. [6]	Full presentation of the main functionalities of OCPP.	It is only a presentation of features of the OCPP.
Alcaraz et al. [29]	Explored threat scenarios and vulnerabilities of the OCPP.	It did not present a real testbed scenario involving multiple entities.
Garofalaki et al. [38]	A Survey on the Security Issues and Challenges of the OCPP.	It only focused on security.
Antoun et al. [30]	Explored security assessment, such as cyber threats, in the EV infrastructure.	It did not present a real testbed scenario involving multiple entities.
Devendra et al. [31]	Designed and projected an electric charger for e-scooters.	It only designed and discussed a single entity in the EV infrastructure.
Ruzmetov et al. [32]	Presented a scheduler of EVs to the closest available charging stations.	It depended on new technology involving communication among EVs and charging stations.
Ottolini et al. [21]	A full comparison regarding distinct IoT platforms.	It only compared IoT platform alternatives.
Sinaeepourfard et al. [33]	Showed a centralized and decentralized approach to data management.	It did not explicitly focus on EV data.
Santos et al. [16]	Proposed a testbed involving IoT devices, such as smart meters.	It did not present a real testbed scenario involving EVs.
Silva et al. [34]	Proposed a sustainable plan for university campus management.	It did not involve the use of EVs.
Răboacă et al. [14]	Presented an overview and classification of OCPP features.	It lacked scalability tests.
Ravindran et al. [35]	Integrated services use case in India	It is too specific to a certain country.
Devendra et al. [36]	Integrated an existing platform for charging processes.	It lacked scalability tests.
Dornberg et al. [41]	The ICT solution, ENS, identified and provided EV and charging station information to the network.	It only presented the results of mobility services in the enterprise environment.
Del Rio et al. [37]	The infrastructure was integrated through an ICT platform and deployed an Energy and EV Management Center.	All project participants paid for the infrastructure and for the services offered.
Fabbri et al. [42]	Featured a system to manage the EV fleet and charging infrastructure.	It only presented the pilot project and illustrated the proposed system.

3. From Charging to Monitoring EVs

This section presents the architecture of SIMA for managing EV charging stations via an efficient IoT and integration and a cloud-based system SIMA, which has two main steps: (1) IoT communication between the charging station and the central system through the OCPP; and (2) cloud-scalable IoT platform that receives EV charging information via Message Queue Telemetry Transport protocol (MQTT) [43], and store such information for further processing. The details of the architecture and the real testbed prototype will be described in the following.

3.1. Main Architecture

The architecture of SIMA is composed of a hardware layer and a software layer, as shown in Figure 1. The hardware layer is composed of EVs and charging stations. The software layer is cloud-based, composed of OCPP back end, IoT middleware, and web application (front end), which follows a typical IoT architecture [16,21,33,34,36]. The EVs can be battery EVs, plug-in hybrid electric vehicles (PHEV), or hybrid electric vehicles (HEV). All the different types are just referenced in this work as EV since the type does not influence our system. The charging stations (also called charge points) have one or more electrical connectors (also called plug-ins), and each electrical connector is used to charge an EV. Finally, the server deploys the cloud-based system in terms of OCPP back end, IoT platform (Dojot), and front end.

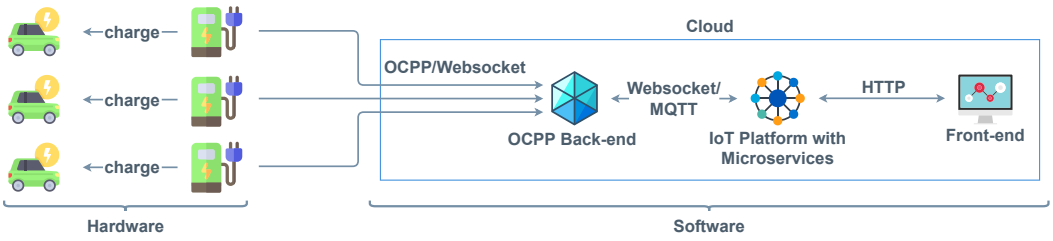

Figure 1. SIMA architecture.

The OCPP back end manages the EV charging stations, obtains information for user authorization, and monitors the changes in the state of the charging stations (i.e., indicate charging phases or failure phases). In this sense, the OCPP back end communicates with the EV charging stations via the OCPP, standardizing data communication between them and enabling better load management. The OCPP back end is responsible for validating all the actions of the charging stations, e.g., user authentication and authorization. In addition, the OCPP back end is responsible for collecting information for managing EV charging sessions, such as status conditions (online/offline) and EV charging variables (current, voltage, electric consumption, etc.).

At the operation's back end, some agents and distinct actions occur. There are communications involving the charging station with the OCPP back-end module in all the charging transactions. This communication can also use Websockets, and it is a full-duplex connection. The OCPP Websockets implementation provides messages in a request and response format, which will be later explained as the `CALL`, `CALLRESULT`, and `CALLERROR` messages.

After collecting the EV charging session data, the OCPP back end considers the MQTT protocol to send the data to the IoT platform. Specifically, MQTT became the standard for IoT communications due to its flexibility to support diverse application scenarios, IoT devices, and services. It is designed as an extremely lightweight message transport, ideal for connecting multiple remote devices with minimal network bandwidth. The protocol allows message deliveries from device to cloud and cloud to device, facilitating the message transmissions to a group of applications.

On the other side, the IoT platform with microservices has an IoT agent to receive data via MQTT. The IoT agent will handle any requisitions/demands from its IoT devices. In other words, the IoT platform is responsible for dealing with the EV and OCPP back-

end-generated data. We considered Dojot [18] as an IoT platform to receive EV charging information and store such information for further processing. Among the main reasons for its use is the fact that it is open source, scalable, and uses the MQTT protocol aimed at IoT applications. This facilitates its integration with the OCPP back end, front end, and other service-based systems [18]. Furthermore, the stored data can be used by an external application to visualize data in real-time using a web application (front end) through Hypertext Transfer Protocol (HTTP) as a communication protocol. Therefore, it is possible to manage data from different charging stations and heterogeneous sensors through the IoT platform.

3.2. IoT Communication between EV Charging Station and OCPP Back-End

The charging stations become easily accessible for remote support and maintenance via the OCPP communication protocol, enabling the collection of information for the management of EV charging stations. Specifically, OCPP is an open protocol standardized by the Open Charge Alliance (OCA) for communication between a charging station and the OCPP back end. In this work, we considered the OCPP version 1.6, which has the following features: JavaScript Object Notation (JSON) messages, Websockets technology, better diagnostics possibilities, more charging status, and the use of trigger messages. The JSON format is used to exchange data between the OCPP back end and the EV charging stations due to its smaller message size. In addition, the protocol supports smart charging for load balancing and the use of distinct user profiles during the charging moments. It is important to state that although we used a specific OCPP version, our proposal could also be used with newer versions [38].

OCPP features and associated messages are grouped in different profiles, namely: (i) the Core is the main profile responsible for the basic functionality of an EV charging station; (ii) the Firmware Management is responsible for managing firmware updates and diagnostic log file download; (iii) the Local Auth List is responsible for managing the local authorization list at the charging station; (iv) the Smart Charging is responsible for basic smart charging, for instance, using the control pilot. It is worth mentioning that an OCPP implementation must contain at least the Core profile because it is the profile with the main features and operations for the correct functioning of the charging stations.

The Core profile enables authorization of clients to use the charging station; displays information and availability of its devices; starts notification about the charging stations; enables configuration of the charging stations; performs clearance of the authorization cache; transfers data among IoT agents; sets up configurations; obtains and sends information about the electrical sensors; enables and disables remote and presential charging (the start and stop of a charging transaction). The transaction is the term used by the OCPP to mention a charging of an EV, which is started after validating all the actions of the charging stations, e.g., user authentication and authorization. For user authorization, the OCPP could consider an authorization cache to store the list of authorized clients/users locally in each charging station for faster authorization.

In this context, the Core profile enables sending a message with 16 different operations, which are: (i) Authorize: enables user authorization/authentication; (ii) BootNotification: allows the notification about the charging station (e.g., model, vendor, firmware version) every time it starts or restarts; (iii) ChangeAvailability: changes the availability status of the charging station, such as inoperative or operative; (iv) ChangeConfiguration: changes the charging station configuration parameters, such as the type of value returned from a *MeterValues* operation and the sampling interval of it; (v) ClearCache: enables cleaning the authorization cache; (vi) DataTransfer: allows sending messages not supported by the OCPP, such as custom warning messages; (vii) GetConfiguration: collects information about a particular charging station configuration; (viii) Heartbeat: enables sending a notification to the OCPP back end to inform that the connection is still active; (ix) MeterValues: allows sending information about the hardware of the charging station's electrical sensors, such as energy transferred from/to the EV: power, voltage, and current measured during the

charging session; (x) RemoteStartTransaction: requests to start the transaction remotely; (xi) RemoteStopTransaction: requests to stop the transaction remotely; (xii) Reset: restarts the charging station; (xiii) Start transaction: requests to start the charging transaction; (xiv) Status notification: enables sending a message by the charging station to the OCPP back end in order to inform about possible errors and status changes; (xv) Stop transaction: requests to stop the charging transaction; (xvi) UnlockConnector: requests to unlock the electrical connector.

The OCPP defines three message types to enable the 16 different operations of the Core profile, namely, `CALL`, `CALLRESULT`, and `CALLERROR`. The first one refers to a request message to perform a specific operation or to inform about some attribute (e.g., charging station status). As soon as the request call can be handled correctly, the response will be a `CALLRESULT` message. Otherwise, the response will be a `CALLERROR` message, informing us that some error has occurred. In other words, every `CALL` message is followed by response type `CALLRESULT` or `CALLERROR` message. Although most request messages are sent from the charging station to the OCPP back end, some procedures are initiated by the OCPP back end.

The `CALL` message follows the JSON and is composed of four attribute fields: (i) `MessageTypeId` refers to the message type identifier. For instance, the value 2 means a `CALL` message; (ii) `MessageId` means an unique message identifier; (iii) `Action` denotes any of the operations performed by both the OCPP back end and the charging station, such as *BootNotification*, *Authorize*, *StartTransaction*, and so on; (iv) `Payload` contains the relevant arguments for the given operation. Listing 1 shows the main structure of a CALL message explained, i.e., it contains the `MessageTypeId`, `MessageId`, `Action`, and its corresponding `Payload`, which is specific to the type of `Action` performed.

Listing 1. CALL message format file (as JSON) for a *StartTransaction* operation

```
[
   2,
   "2959:49",
   "StartTransaction",
   {
   "connectorId": 1,
   "idTag": "NOA",
   "meterStart": 2087100,
   "timestamp": "2022-03-04T13:37:18.616Z"
   }
]
```

The `CALLRESULT` message follows a similar format, and it is composed of only three fields: (i) `MessageTypeId` means the message type identifier. For instance, the value 3 means a `CALLRESULT` message; (ii) `MessageId` means an unique message identifier; (iii) `Payload` contains the relevant arguments for the given operation. The response message identifier must be the same as the request message to identify to which request the response belongs. Additionally, `Payload` has the response content.

If an error occurs during or after the sending of the request, the `CALLERROR` message must be transmitted, which contains five attribute fields: (i) `MessageTypeId` refers to the message type identifier. For instance, the value 4 means a `CALLERROR` message; (ii) `MessageId` means an unique message identifier; (iii) `ErrorCode` is a unique identifier of the error (e.g., ProtocolError); (iv) `ErrorDescription` is an optional brief description of the reported error (e.g., "Payload for an action is incomplete"); (v) `ErrorDetails` is an optional and custom JSON object containing detailed information about the error.

For example, the charging station must send a *BootNotification* request to the OCPP back end to inform its configuration (e.g., version, model, supplier, etc.), which is included on the message payload, as soon as a charging station is turned on or restarted. If an error does not occur after sending the request message, the OCPP back end sends a `CALLRESULT` message to the charging station containing the following fields in the `Payload`: (i) `CurrentTime` to inform the current day and time; (ii) `Interval` to define how often the EV charging station must notify the OCPP back end that the connection is still active (i.e., *Heartbeat*

operation); (iii) `Status` to indicate the current condition of the charging station registered in the OCPP back end (i.e., `ACCEPTED`, `REJECTED`, or `PENDING`). If an error occurs after or during the sending of the request, the central system returns the error message `CALLERROR`.

Therefore, all charging information is captured by the OCPP back end via the OCPP and then forwarded to the IoT platform via the MQTT protocol. The MQTT protocol allows sending messages between the OCPP back end and IoT platform with minimal network bandwidth consumption. In this sense, we create an MQTT message with the attributes already defined in OCPP to send charging data to the IoT platform through the MQTT protocol.

3.3. IoT Platform to Handle the Large Volume of Data Collected by the EV Physical Infrastructure

The IoT platform enables storing the data in a database and provides the data to other external applications, such as the visualization in real-time on a web interface (front end) of the received data. In this sense, we considered Dojot version 0.7 [18] as an IoT platform since it is designed to collect and store large volumes of data from different IoT devices efficiently and by supporting horizontal scaling. Specifically, Dojot is a Brazilian open-source IoT platform that emerged to develop and demonstrate technologies to facilitate the development of IoT solutions [18].

The Dojot Platform has several modules, where each module represents a Dojot service and has a specific function. The user can configure a template or device or send data into Dojot using the graphical user interface (GUI) via HTTP requests. IoT devices at Dojot are virtual representations of real devices or entities, such as charging stations. In this sense, for each charging station connected to the OCPP back end, a virtual device is also created in Dojot through the Device Manager module, which uses Kafka. This device receives and stores the corresponding data in the MongoDB database through the Persister module. In addition, Dojot could have several IoT agents and a service specialized in dealing with a specific protocol, such as MQTT/JSON and HTTP/JSON. After configuring the IoT devices, the IoT agent will be able to handle data received from a specific IoT device via MQTT. The IoT agent must have connectivity to other services at Dojot to use the received data, e.g., Dojot enables it to notify when a specific attribute reaches a given threshold or to save the generated data in a database. The IoT agents must verify if a connection is valid (or not) and enable (or not) the message processing based on the device status.

In our context, a charging station, which is a physical device, sends data to the OCCP back end, which process and forwards the data to Dojot through the MQTT/JSON protocol represented by an IoT Agent. Upon arriving at Dojot, Apache Kafka makes the data available in real-time via a WebSocket connection, and the Persister service stores the data in the Mongo Db database. These data can be viewed through Dojot's own GUI or requested from the history service through REST API to be presented in our external monitoring application (front end). Figure 2 exhibits these examples.

In this proposal, we deploy Dojot in a cluster Kubernetes with two workers and a load balancer (Nginx). Dojot services can be executed simultaneously on both worker machines. When there is a request for Dojot services and data, the balancer distributes these requests between worker machines not to overload them. Additionally, suppose the resources of the two existing worker machines are not enough to meet the existing demand. In that case, the cluster Kubernetes can increase the cluster by adding more worker machines (horizontal scalability). Another point to note is that if any machine in the cluster goes offline, Dojot's services do not stop because there is more than one worker machine online, thus ensuring its availability. Additionally, we can see the available machines and set which worker each service will run on.

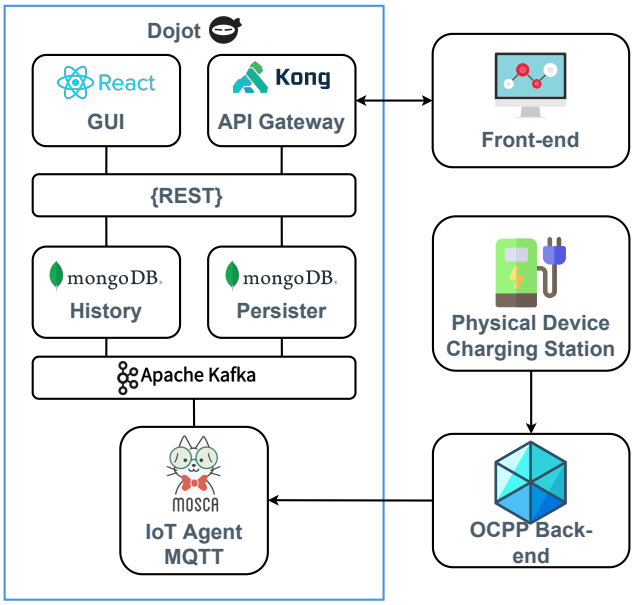

Figure 2. Dojot (IoT platform) service architecture.

In addition to enabling horizontal scalability to meet increased demand, Kubernetes has other important advantages. According to the official Kubernetes documentation [25] and other works found in the literature [26–28], among its main advantages, we can mention: (*i*) increased productivity: it reduces the number of manual processes needed to deploy, update and scale applications; (*ii*) optimization of resource usage: it knows the number of computing resources, such as memory and storage, that each application needs and allocates them according to demand; (*iii*) management: it makes managing containers easier.

Figure 3 presents the architecture of the created cluster Kubernetes, which consists of the following nodes: (i) a master node to administrate and manage the cluster Kubernetes; (ii) two worker nodes to run Dojot services. For an environment where there is a large load of devices (e.g., several EV charging stations), it is recommended to consider at least two worker nodes in the cluster Kubernetes; (iii) a load balancer node to receive all requests (i.e., MQTT and HTTP) and perform load balancing between the workers' nodes of the Kubernetes cluster.

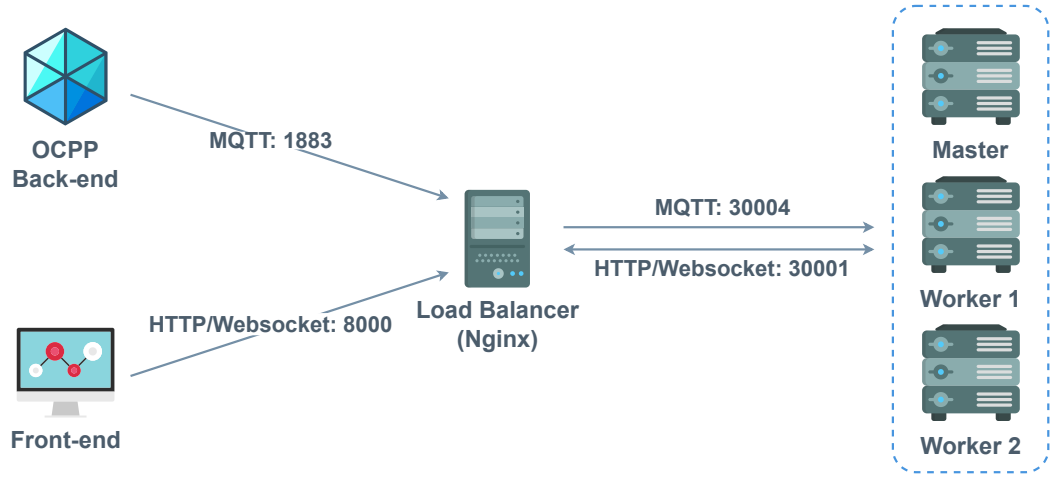

Figure 3. HTTP and MQTT requests.

For instance, in an MQTT connection for sending data, the load balancer (Nginx) receives a connection from the OCCP back end on port 1883 and internally redirects it to one of the two worker machines of the Kubernetes cluster on port 30004. On the other hand,

to provide a connection to consume the data and display it on the front end, the balancer receives a request from the front end on HTTP port 8000 and redirects it to one of the two worker machines on port 30001. Figure 3 illustrates these two types of requests.

3.4. Front End

The front end needs to exhibit in full detail and with great scalability all the content required by the EV operator, whether for a simple conference or a deeper analysis. For instance, Figure 4 shows the use case diagram for the front-end service. All use cases illustrate features that involve fetching data from the IoT platform. The "list available CPs" starts when the user accesses the home page of our front end, which triggers an API call that returns a list of manually registered CPs, including information such as location, software version, and serial number. "Show CP charging history" is activated when the user navigates to the history page, passing the CP Id as a route parameter and returning a detailed list of the transactions handled by the respective CP. "Show CP status history" is triggered when the user clicks on a particular CP on the home page, showing a timeline chart illustrating all CP statuses in the past 24 h. "Send configuration message to CP" is our control screen, where an administrator can configure data sampling parameters of the charger, such as the sampling interval and the type of data retrieved on every sample. Finally, "Show charging data in charts" and "Data analysis" are both about reading the measurements made during the charging processes. The second allows the user to choose the presented information, and the first organizes it into a visual representation. Both use cases are activated when the user selects a particular charging session on the history page.

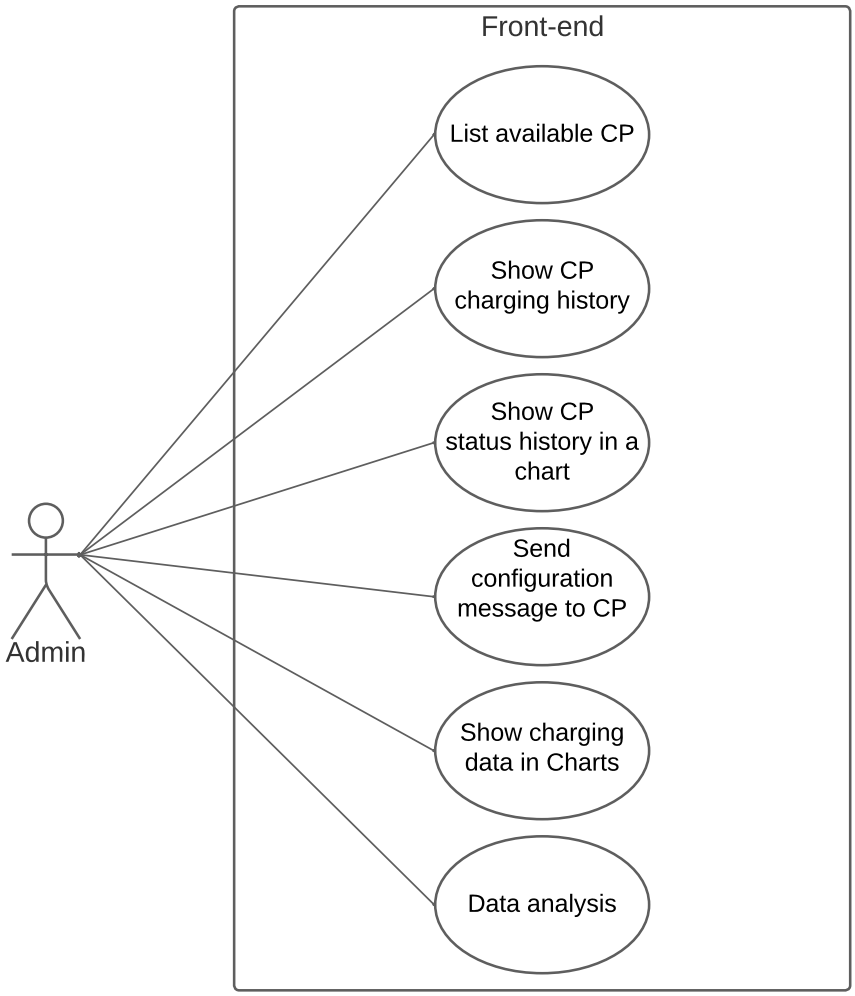

Figure 4. Use case diagram for front-end service.

We can use a scalable framework for the front-end website for visualization and analysis. It can show all the distinct collected variables and data recorded in the charging sessions. Therefore, the IoT platform and front end are essential for real-time visualization and post-analysis if required. Moreover, such analysis can be performed using machine learning techniques, statistic algorithms, information theory quantifiers, deep learning techniques, or any other data processing tools or techniques. In this sense, we implemented a front-end service as a dashboard using Next.js, i.e., a framework built using the React library [44]. This library was created to implement single-page applications (SPAs), a pattern based on having the whole app loaded on the first request instead of retrieving each page from the server when navigation occurs. Since the structure is JavaScript-based, bare React is recommended for dynamic web applications with high user interactivity, such as dashboards or text and image editing tools, without needing to load the content as fast as possible.

This framework takes advantage of the flexibility of React code and enhances performance by introducing the concept of server-side rendering (SSR). The SSR engine is responsible for mounting the interface and sending an optimized page to the client's browser. In terms of user experience, this means faster rendering when accessing the page for the first time, also keeping React's fast navigation. Therefore, such fast rendering is very important to display constant data from the EVs collected at every charging session. Moreover, Next.js offers internal caching mechanisms responsible for making the websites scalable since a few adjustments can reduce the number of requests sent to the server.

In addition, some control commands can also be sent by the front end via OCPP messages to configure the EV chargers or control sessions remotely. For instance, the "change configuration" screen is responsible for configuring the charging station execution parameters, such as the heartbeat interval. By setting the heartbeat interval using a number input and pressing *confirm*, a WebSocket message is sent to the OCPP back end, which is forwarded to the charging station to be configured.

Figure 5 displays the complete sequence diagram to exhibit a history page in the front-end service. The actors are the user (or EV operator), the dashboard server, the Nodejs API, and the IoT Platform. It all starts as a requisition from the user towards the dashboard server to get a single history page containing some recent data from the cache. The user can then select certain charger stations to view the ongoing collected data; therefore, the dashboard server contacts the Nodejs and then the IoT Platform for a full reading of the collected data safely stored in the IoT platform. The platform will return the GET requisition in JSON data format, which is transmitted to the Nodejs API to the dashboard and then finally shown to the user.

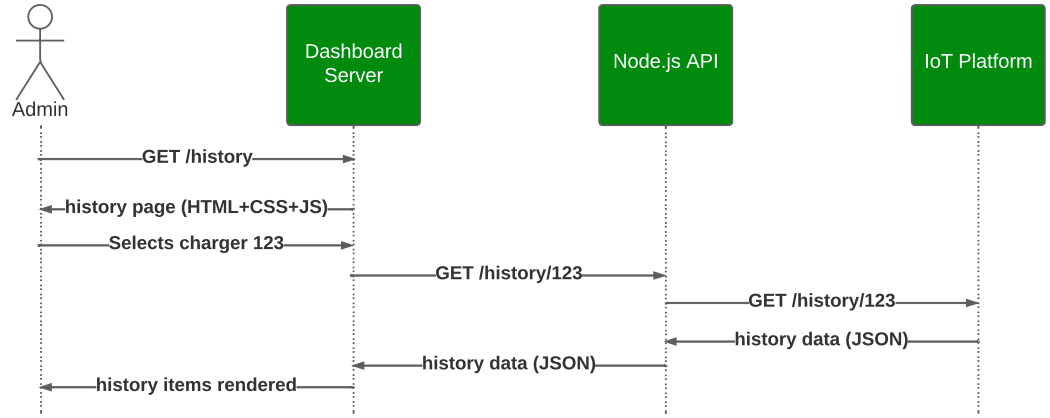

Figure 5. History page sequence diagram.

4. Prototype

We define a prototype within the scope of the ANEEL R&D project entitled "Intelligent System for Efficient Management of Multimodal Electric Mobility", also called the SIMA

project. The prototype comprises EVs, charging stations, and computational infrastructure available at two campuses of UFPA. Specifically, there are two ABB charging stations of the Terra 54 model (T54) with a recharging capacity of 50 kW each, one ABB charging station of the EV Lunic Pro M model with a recharging capacity of 22 kW, and two BYD charging stations of the EVA040 model with a recharging capacity of 40 kW each. The infrastructure also has two electric buses for transporting users/students around the Guamá Campus of UFPA and one electric boat.

For the data collection, the charging station allows communication via IEEE 802.11 interface and data collection via the OCPP. For instance, Figure 6 illustrates one electrical bus (i.e., EV) charging on an ABB charging station at the main campus of UFPA. It is important to mention that the SIMA system allows the addition of any charging station with support to OCPP. However, to collect data from non-OCPP charging stations, the installation of three-phase electric energy meters at these charging stations is required to send data through any network.

Figure 6. EV and charging station deployed at the main campus of UFPA.

In our prototype, some electric buses are traveling around UFPA to serve the students and community that need to move between distinct buildings of the university, such as the main university restaurant, the university hospital, and the computer science building. In addition, there is also an electric bus traveling between two campuses 70 km far away. Therefore, the electric bus needs to charge, from time to time, before going back to the garage, where the bus driver requests to start the vehicle charging.

In terms of computational infrastructure, there is a Power Edge T440 server connected to the local network, which uses Linux Ubuntu Server 18.04 operating system with 64 GB of RAM, an Intel Xeon Silver 4210 processor with 40 cores, and a storage capacity of 5 TB. We deployed the OCPP back end, IoT platform (Dojot), and the front-end application in virtual machines created on the server. Table 2 presents the number of resources allocated to each node. It is important to note that we allocated 700 GB of storage for each worker node since this amount of storage will be enough to allocate telemetry data for up to 100 years on the server, already considering a safety margin of 20%. To obtain this value, we considered the extreme case of 50 charging stations sending data to Dojot every 1 min, where each charging station can send up to 225 bytes every 1 min, achieving a total of 11,250 bytes.

At the EV charging station, there are some charging points located where any driver can charge his/her vehicle. The charging station must evaluate the authorization requisition, validate whether the charging can be performed, and then enable starting the charging transaction. In this context, Figure 7 shows a sequence diagram of operations and messages exchanged via OCPP during a full EV charging session, and also the structure of *StartTransaction* request message operation sent to the OCPP back end during a charging session. Initially, verifying the user's authentication is required to start and finish a charging process.

Thus, the charging station requests authorization from the OCPP back end through the *Authorize* operation, containing the `idTag` identifier. On the other side, the OCPP back end responds by informing the authorization status using the `idTagInfo` attribute field.

Table 2. Number of resources allocated on each machine.

Machine	CPU (Cores)	Memory (GB)	Storage (GB)
Docker-composer	4	16	40
Master	2	4	64
Worker1	4	8	700
Worker2	4	8	700
Nginx	1	2	32
OCPP back end and application front end	8	4	1000

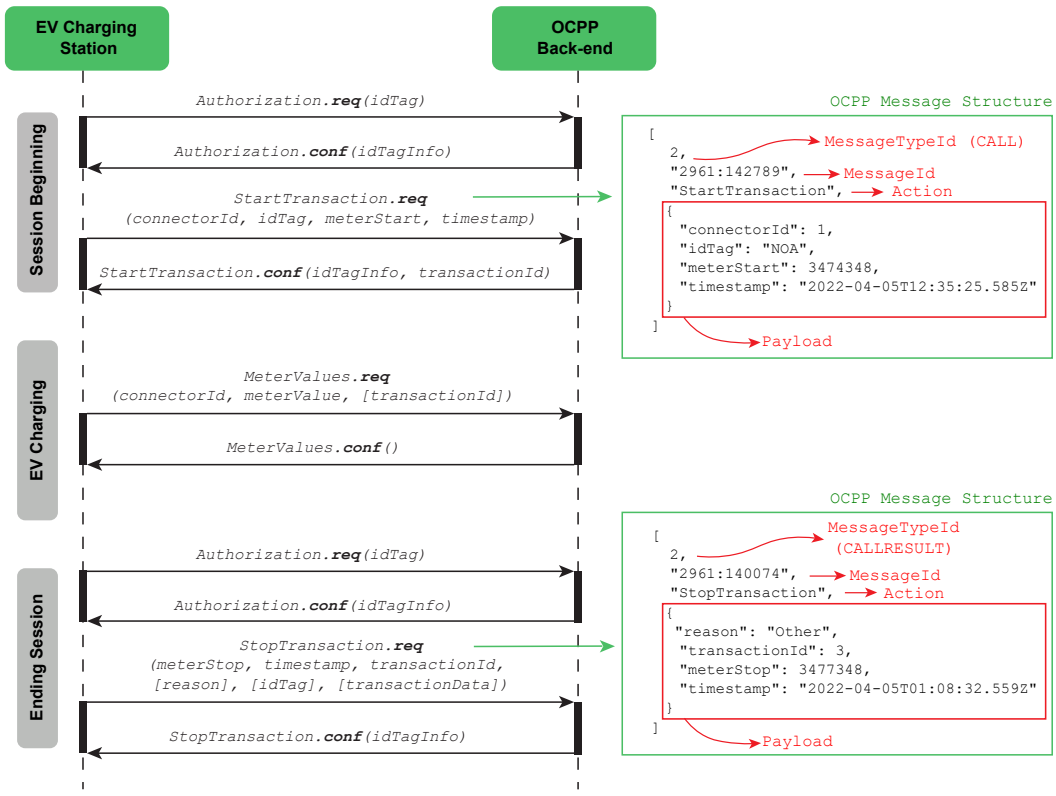

Figure 7. EV charging operation.

If the OCPP back end authorizes the user, the charging station requests the start of the transaction containing the following information: `connectorId` to inform which charging station connector is being used; `idTag` means the authorization identification; `meterStart` contains the meter value in Wh for the connector at the start of the transaction; `timestamp` to inform the day and time the charging session starts. The OCPP back end then sends a reply message containing the authorization status (positive, in this case) and the transaction identifier (`transactionId`).

Naturally, the charging session starts after the user authorization. During the charging session, the charging station sends information (energy, electric current, voltage, etc.) to the OCPP back end through the *MeterValues* operation. We collect the following information: (i) `Current.Import:` instantaneous flow of electric current to the vehicle; (ii) `Energy.Active.Import.Register:` numerical value read (in Wh or kWh) from the electric meter that measures the imported energy (from the electric grid supply); (iii) `Power.Active.Import:` instantaneous active power imported by the EV; (iv) `SoC:` vehicle charging status (in percentage); (v) `Voltage:` instantaneous voltage measured on

the direct current connector's meters during the charging; (vi) `Current.Offered:` maximum current offered to the EV based on the vehicle's maximum capacity provided during the charging session setup; (vii) `Power.Offered:` maximum power offered to the EV based on the maximum capacity of the vehicle provided during the charging session setup; (viii) `Temperature:` temperature inside the charging station.

The information and transmission periods are configured when the charging station is turned on. In this case study, we defined the interval as 10 seconds. Finally, the charging station again performs the *Authorize* operation to finish the charging session, to verify that the user who finished the session is the same user who started the charging process. Once authorized, the charging station sends a *StopTransaction* operation to finish the session (as also seen in Figure 7). This message contains the following mandatory information: `meterStop` that contains the meter value (in Wh) for the connector at the end of the transaction, `timestamp` that records the day and time of the end of charging, `transactionId`, the identifier of the transaction, and other optional information.

After the charging operation, the data from the OCPP back end are sent to an IoT platform. In order to receive the data from the charging stations, we must configure the IoT platform to receive upcoming information from real charging stations, with the following variables: authorization notification and authorization logs, the start of charging transaction, stop of charging transaction, electric current, voltage, among others. After performing this initial configuration, the IoT platform can receive the data sent by the charging stations through the OCPP back end. After preparing the IoT platform and saving the transmission data collected from the charging session into the database, the recorded session from the log can be visualized and analyzed.

Figure 8 illustrates how the interface used on this project is responsible for providing an overview of the EV charges by showing the data stored in the IoT platform and consumed through the application programming interface. This information gives feedback on all the chargers' status and details, such as session history and energy given during these sessions. This information exhibited in the dashboard format using a reliable framework is the ideal structure to support and show dynamic IoT data in a scalable way.

Figure 8. Charge station details screen.

Each of these charging sessions varies between 1 and 7 h and is directly proportional to the total energy supplied for our electric buses. In addition, there is extra information recorded in our log and database that was not illustrated here due to the page limit. The complete prototyping, testbed, and visualization of the charging information were successful and satisfactory and allowed us to continue the implementation and analysis of other variables in the future, whether regarding pricing or more information about the electric vehicle.

5. Evaluation

This section presents the full setup and the evaluation performed in our prototype scenario. In this sense, we perform a scalability evaluation to verify how machines with the IoT platform respond to the increasing number of charging stations (devices) sending data to it. For this, we performed evaluations with the OCPP back-end central system, containing several charging stations, simultaneously sending data to the IoT platform. Regarding the number of charging stations connected to the IoT platform and sending data, it was initially considered 10 charging stations. This amount increased exponentially every 10 min to 20, 40, 80, and 160 charging stations.

We carried out the evaluations in two different environments, namely: (i) we deploy Dojot through Docker-Composer, on a single machine (which will be called "docker-compose"); (ii) we deploy Dojot in a Kubernetes cluster containing two worker machines as introduced before (which will be called "worker 01" and "worker 02"). Each of these two environments was tested separately for a period of 1 h each. The first testbed in the docker-compose environment was carried out on 28 June 2022, from 4 p.m. to 5 p.m. On the other hand, the second testbed in the Cluster Kubernetes environment was carried out on 29 June 2022, from 4 p.m. to 5 p.m. In both environments, the number of charging stations and EV were exponentially increased every 10 min, passing through 10, 20, 40, 80, and 160 devices.

Based on the evaluation performed, we chose to analyze the CPU and memory metrics of the machines that contain the IoT platform. Although average CPU and memory usage are affected by numerous factors, within the context of applications deployed in containers, such metrics are directly affected as application scalability increases, according to the literature [24,45].

Figure 9 shows the average CPU usage on the machines, as the number of charging station devices receiving data increases exponentially from 10 to 20, 40, 80, and 160. Regardless of the number of devices sending data, the machine with the highest CPU consumption was the docker-composer, precisely because all Dojot platform services run on a single machine. When analyzing the worker machines of the Kubernetes cluster, worker machine 02 presented the highest CPU consumption, regardless of the number of devices sending data in the testbed scenario. This occurs because the Kafka services pods, i.e., X.509 and VerneMQ, are running on this machine and need more processing, in line with what is found in the literature [24,45]. Specifically, the average CPU consumption for 10, 20, 40, 80, and 160 charging stations were, respectively, 17%, 17%, 20%, 24%, and 30% on the docker-compose machine; 12%, 13%, 14%, 17%, and 22% on worker machine 01; and 14%, 15%, 17%, 21%, and 26% on worker machine 02. Thus, it was possible to notice that as the number of devices increases exponentially, the consumption of the CPU metric also increases, but in a slight upward trend.

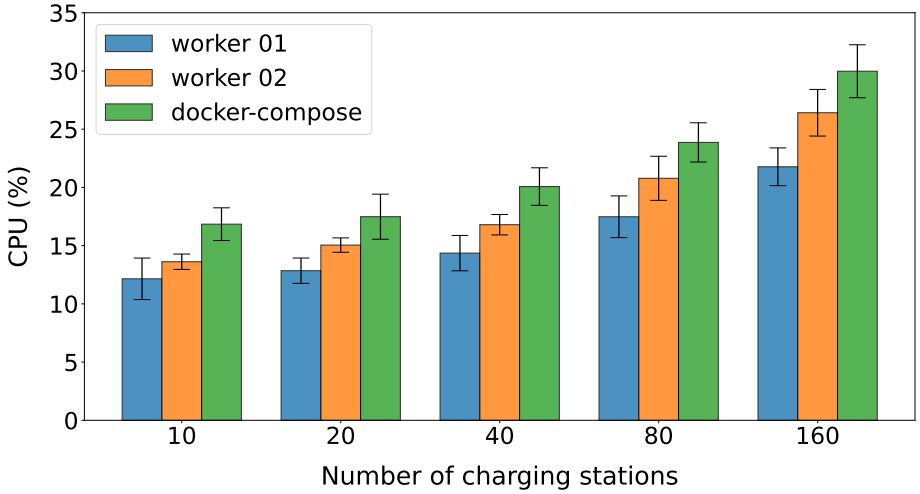

Figure 9. CPU usage in Kubernetes cluster with the IoT platform.

Figure 10 shows the average memory usage on machines as the number of charging station devices receiving data exponentially increases from 10 to 20, 40, 80, and 160. Specifically, the average memory consumed at 10, 20, 40, 80, and 160 charging stations were, respectively: 96%, 95%, 96%, 96%, and 96%, on the docker-compose machine; 94%, 94%, 95%, 95%, and 95%, on worker machine 01; and 96%, 96%, 96%, 95%, and 95%, on worker machine 02. Considering all these three machines, the memory consumption was higher on the docker-compose machine because all Dojot services run on this single machine. Considering Dojot running on worker machines 01 and 02, the average memory consumption decreased because both machines share the execution of the pods. It is also worth noting that worker 01 has the highest memory consumption because most pods of Dojot's services run on it, in line with what is found in the literature [24,45]. Table 3 describes the insights gained from bench testing.

Based on the analysis of testbed results, we can affirm that the SIMA system supports the expansion of charging stations since we deployed a scalable IoT platform. For instance, the system supports up to 160 charging stations with the current amount of resources (CPU and memory). In addition, we observed that the adequate use of CPU and memory resources is essential to guarantee the availability of the IoT platform. Once one of these resources reaches its maximum consumption, the functioning of the IoT platform is compromised: sometimes, it becomes "slow" or even offline due to the forced shutdown of the machines where the platform is installed.

Table 3. Insights.

Machine	CPU	Memory
Docker-composer	Regardless of the number of devices that send data, the machine with the highest CPU consumption was docker-composer because all the services of the Dojot platform run on a single machine.	The docker-compose machine had the highest memory consumption because all Dojot services run on this single machine.
Worker1	Worker machine 01 presented the lowest CPU consumption, compared to worker 02, regardless of the number of devices sending data in the testbed scenario.	Considering Dojot running on workers 01 and 02, the average memory consumption decreased, because both machines share the execution of the pods.
Worker2	Worker machine 02 had the highest CPU consumption, regardless of the number of devices sending data in the testbed scenario. This is because the Kafka services, i.e., x509 and VerneMQ, are running on this machine and need more processing.	When compared to worker 01, worker 02's machine has a 1% lower memory consumption, because only Kafka, x509, and VerneMQ services run on it.

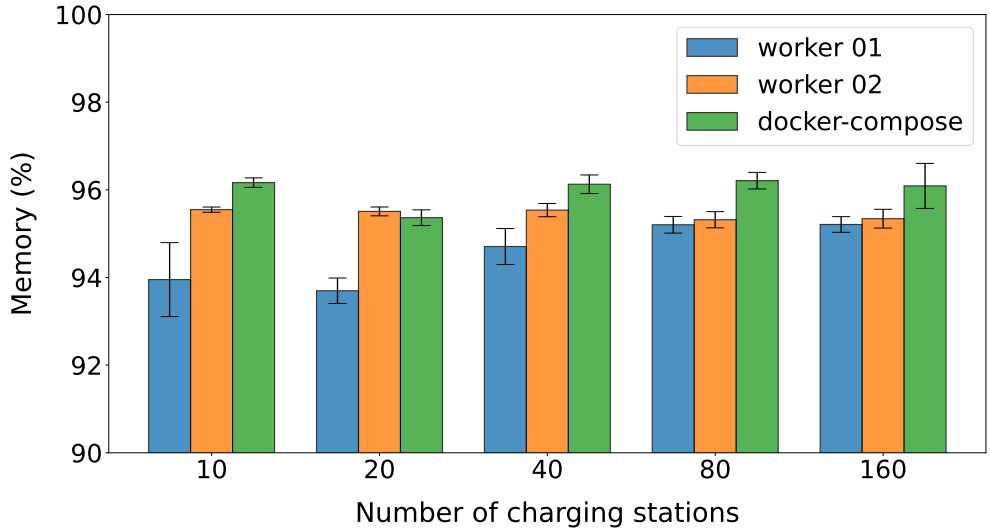

Figure 10. Memory usage in Kubernetes cluster with the IoT platform.

6. Conclusions

This article introduced the prototype of a real case study from charging to monitoring EV, called SIMA. It enables us to verify how to execute the complete integration between EV charging stations with the OCPP management system, the IoT platform, and a front-end web application that constitutes the cloud part version. Regarding the OCPP central system, it is essential for communication and integration with EVs, since it allows the collection of various information for the user and for the system administrator from a simple loading.

Regarding scalability, the IoT platform was deployed to be horizontally scalable, where in addition to the two existing worker machines, it is possible to add more workers to the Kubernetes cluster to increase the number of resources (CPU and memory) available. However, it is noteworthy that the two existing worker machines met the simulated quantity of 160 electrical stations well, sending data simultaneously to the IoT platform. The IoT platform, in turn, received and stored the data and, when requested, sent the data to be displayed on the front end. Concerning the front-end web application, it was possible to use/consume the data stored from the IoT platform and externalize it in a more friendly way for potential users or for the administrator of the EV charging stations.

Based on such efficient integration of IoT communication and IoT platform for monitoring and managing a highly dense EV system, we introduced a prototype of SIMA using the physical infrastructure for electric mobility located at the Federal University of Pará (UFPA), which enabled us to full deploy and test our proposal. This infrastructure comprises electric buses, charging stations, photovoltaic systems, and energy storage systems, among others, within the context of a Research & Development Project for multimodal electric mobility funded by Norte Energia S.A.

Based on the evaluation carried out, it was possible to verify that as the amount of EV charging stations sending data increases exponentially, the IoT platform consumes CPU resources in a slight upward trend. Among the consumed resources, the CPU metric is the one that varies the most. In contrast, the memory metric is the one that remains constant, even though it was the most demanded resource.

One of the main benefits of the implemented system is the creation of a reference architecture for management and monitoring systems for charging stations, in addition to the possibility of generating electric mobility business models specific to the Amazon region. Such models can benefit several electric mobility entities, such as the local energy concessionaire and private companies that have fleets of buses and charging stations. Another benefit for these players is the possibility of managing charging stations through the web application (front end) developed. EV owners, in turn, benefit from the increase in

the number of charging stations available to charge their vehicles and from the possibility of monitoring the charging via the web application.

Another important benefit generated with the implemented system, considering all its physical infrastructure (electric buses, charging stations, photovoltaic systems, and energy storage systems), concerns the environmental benefits for all involved. Once implemented, the system makes the transport system in the Amazon region "cleaner", that is, an eco-friendly approach for such an important and ecological place for the local communities and globally for its importance in biodiversity and global climate impacts. The main highlight is the use of less polluting electric buses and the generation of clean energy to supply them, therefore, in conformity with goals 7, 11, and 13 of the 2030 Agenda for Sustainable Development of the United Nations (UN) [46].

As a future work, we intend to upgrade the system to use version 2.0 of the OCPP and thus guarantee more safety requirements in this context of charging electric vehicles. In addition, we also intend to generate electric mobility business models for the Amazon region considering the system implemented and, therefore, replicate it in other Amazon sub-regions.

Author Contributions: Conceptualization, E.L., L.P., I.M., and F.A.; methodology, E.L., L.P., I.M., and F.A.; software, E.L., L.P., and F.A.; validation, I.M., and D.R.; formal analysis, D.R., E.C., M.T., U.B., W.F., and A.A.; writing—original draft preparation, E.L., L.P., and I.M., writing—review and editing, D.R., E.C., M.T., U.B., W.F., and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Norte Energia S.A. within the scope of the ANEEL R&D project 07427-0319/2019 entitled "Intelligent System of Efficient Management of Multimodal Electric Mobility", carried out through the ANEEL Strategic Call nº22/2018, and also by Dean of Research and Graduate Studies (PROPESP-UFPA), and the Coordination for the Improvement of Higher Education Personnel (CAPES).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lombardi, M.; Panerali, K.; Rousselet, S.; Scalise, J. Electric Vehicles for Smarter Cities: The Future of Energy and Mobility. World Economic Forum, 2018. Available online: https://www3.weforum.org/docs/WEF_2018_%20Electric_For_Smarter_Cities.pdf (accessed on 15 April 2022).
2. Donnellan, P.R. The Future of Mobility—Electric, Autonomous, and Shared Vehicles. *IEEE Eng. Manag. Rev.* **2019**, *46*, 16–18. [CrossRef]
3. Vijayakumar, K. Solar Charging Infrastructure for E-vehicles-A Review. In Proceedings of the 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 4–5 March 2021; pp. 586–588.
4. Mahrez, Z.; Sabir, E.; Badidi, E.; Saad, W.; Sadik, M. Smart Urban Mobility: When Mobility Systems Meet Smart Data. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 6222-6239. [CrossRef]
5. Eisenbarth, M.; Wegener, M.; Scheer, R.; Andert, J.; Buse, D.S.; Klingler, F.; Sommer, C.; Dressler, F.; Reinold, P.; Gries, R. Toward Smart Vehicle-to-Everything-Connected Powertrains: Driving Real Component Test Benches in a Fully Interactive Virtual Smart City. *IEEE Veh. Technol. Mag.* **2021**, *16*, 75–82. [CrossRef]
6. Pruthvi, T.V.; Dutta, N.; Bobba, P.B.; Vasudeva, B.S. Implementation of OCPP Protocol for Electric Vehicle Applications. *E3S Web Conf.* **2019**, *87*, 01008. [CrossRef]
7. Viziteu, A.; Furtună, D.; Robu, A.; Senocico, S.; Cioată, P.; Remus Baltariu, M.; Filote, C.; Răboacă, M.S. Smart Scheduling of Electric Vehicles Based on Reinforcement Learning. *Sensors* **2022**, *22*, 3718. [CrossRef] [PubMed]
8. Mota, R.; Riker, A.; Rosário, D. Adjusting Group Communication in Dense Internet of Things Networks with Heterogeneous Energy Sources. In Proceedings of the 11th Brazilian Symposium on Ubiquitous and Pervasive Computing, Porto Alegre, Brazil, 11–13 September 2019.
9. Hashmi, S.A.; Ali, C.F.; Zafar, S. Internet of things and cloud computing-based energy management system for demand side management in smart grid. *Int. J. Energy Res.* **2021**, *45*, 1007–1022. [CrossRef]
10. Ghasempour, A. Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges. *Inventions* **2019**, *4*, 22. [CrossRef]
11. Khanna, A.; Kaur, S. Internet of things (IoT), applications and challenges: a comprehensive review. *Wirel. Personal Commun.* **2020**, *114*, 1687–1762. [CrossRef]

12. Tightiz, L.; Yang, H. A comprehensive review on IoT protocols' features in smart grid communication. *Energies* **2020**, *13*, 2762. [CrossRef]
13. Open Charge Alliance-Global Platform For Open Protocols. Available online: https://www.openchargealliance.org/ (accessed on 27 May 2022).
14. Răboacă, M.S.; Meheden, M.; Musat, A.; Viziteu, A.; Creanga, A.; Vlad, V.; Filote, C.; Rață, M.; Lavric, A. An overview and performance evaluation of open charge point protocol from an electromobility concept perspective. *Int. J. Energy Res.* **2021**, *46*, 523 – 543. [CrossRef]
15. Saleem, Y.; Crespi, N.; Rehmani, M.H.; Copeland, R. Internet of things-aided smart grid: technologies, architectures, applications, prototypes, and future research directions. *IEEE Access* **2019**, *7*, 62962–63003. [CrossRef]
16. Santos, H.; Eugenio, P.; Marques, L.; Oliveira, H.; Rosário, D.; Nogueira, E.; Neto, A.; Cerqueira, E. Internet of Smart Grid Things (IoSGT): Prototyping a Real Cloud-Edge Testbed. In Proceedings of the 14th Brazilian Symposium on Ubiquitous and Pervasive Computing, Porto Alegre, Brazil, 11–13 July 2022; pp. 1–10.
17. Modesto, W.; Bastos, L.; Venâncio Neto, A.; Rosário, D.; Cerqueira, E. Towards Automating the Integration of Legacy IEDs into Edge-Supported Internet of Smart Grid Things. *J. Internet Serv. Appl.* **2022**, *12*, 33–46. [CrossRef]
18. CPqD. Dojot Documentation. 2020. Available online: https://dojotdocs.readthedocs.io/en/latest/ (accessed on 5 June 2022).
19. Chaqfeh, M.A.; Mohamed, N. Challenges in middleware solutions for the internet of things. In Proceedings of the 2012 International Conference on Collaboration Technologies and Systems (CTS), Denver, CO, EUA, 21–25 May 2012; pp. 21–26.
20. Javed, A.; Malhi, A.; Kinnunen, T.; Främling, K. Scalable IoT Platform for Heterogeneous Devices in Smart Environments. *IEEE Access* **2020**, *8*, 211973–211985. [CrossRef]
21. Ottolini, D.; Zyrianoff, I.; Kamienski, C. Interoperability and Scalability Trade-offs in Open IoT Platforms. In Proceedings of the 2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC), Las Vegas, NV, EUA, 8–11 January 2022; pp. 1–6.
22. Abuabdo, A.; Al-Sharif, Z.A. Virtualization vs. Containerization: Towards a Multithreaded Performance Evaluation Approach. In Proceedings of the IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), Abu Dhabi, Emirados Árabes Unidos, 3–7 November 2019; pp. 1–6.
23. Jutadhamakorn, P.; Pillavas, T.; Visoottiviseth, V.; Takano, R.; Haga, J.; Kobayashi, D. A scalable and low-cost MQTT broker clustering system. In Proceedings of the 2017 2nd International Conference on Information Technology (INCIT), Nakhonpathom, Tailândia, 2–3 November 2017; pp. 1–5.
24. Dewi, L.P.; Noertjahyana, A.; Palit, H.N.; Yedutun, K. Server Scalability Using Kubernetes. In Proceedings of the 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), Bangkok, Tailândia, 11–13 December 2019; pp. 1–4.
25. Foundation, T.L. Kubernetes Documentation. 2022. Available online: https://kubernetes.io/docs/home/ (accessed on 15 April 2022).
26. Moilanen, M. Deploying an application using Docker and Kubernetes. In Proceedings of the Oulu University of Applied Sciences, Oulu, Finland, 23–24 August 2018; pp. 1–51.
27. Xie, X.L.; Wang, P.; Wang, Q. The performance analysis of Docker and rkt based on Kubernetes. In Proceedings of the 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, China, 29–31 July 2017; pp. 2137–2141.
28. Baró Cayetano, L. Creation of a Kubernetes Infrastructure. Bachelor's Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2021.
29. Alcaraz, C.; Lopez, J.; Wolthusen, S. OCPP Protocol: Security Threats and Challenges. *IEEE Trans. Smart Grid* **2017**, *8*, 2452–2459. [CrossRef]
30. Antoun, J.; Kabir, M.E.; Moussa, B.; Atallah, R.; Assi, C. A Detailed Security Assessment of the EV Charging Ecosystem. *IEEE Netw.* **2020**, *34*, 200–207. [CrossRef]
31. Devendra, D.; Malkurthi, S.; Navnit, A.; Hussain, A.M. Compact Electric Vehicle Charging Station using Open Charge Point Protocol (OCPP) for E-Scooters. In Proceedings of the 2021 International Conference on Sustainable Energy and Future Electric Transportation (SEFET), Hyderabad, India, 21–23 January 2021; pp. 1–5.
32. Ruzmetov, A.; Nait-Sidi-Moh, A.; Bakhouya, M.; Gaber, J. Towards an optimal assignment and scheduling for charging electric vehicles. In Proceedings of the 2013 International Renewable and Sustainable Energy Conference (IRSEC), Ouarzazate, Morocco, 7–9 March 2013; pp. 537–541.
33. Sinaeepourfard, A.; Krogstie, J.; Sengupta, S. Distributed-to-Centralized Data Management: A New Sense of Large-Scale ICT Management of Smart City IoT Networks. *IEEE Internet Things Mag.* **2020**, *3*, 76–82. [CrossRef]
34. Silva, L.; Meira, P.; Cypriano, J.; Azzini, H.; Santos, A. Software toolchain to enhance the management and integration of a sustainable campus model. *Energy Inform.* **2021**, *4*, 41. [CrossRef]
35. Ravindran, S.; Amal, S.; Bhavya, Y.V.; Chandrasekar, V. OCPP based Electric Vehicle Supply Equipment and its user interface for AC charging in Indian scenario. In Proceedings of the 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, 10–13 December 2020; pp. 1–6. [CrossRef]

36. Devendra, D.; Mante, S.; Niteesh, D.; Hussain, A.M. Electric Vehicle Charging Station using Open Charge Point Protocol (OCPP) and oneM2M Platform for Enhanced Functionality. In Proceedings of the TENCON 2021—2021 IEEE Region 10 Conference (TENCON), New Delhi, India, 7–10 December 2021; pp. 1–5. [CrossRef]
37. Del Rio, R.; Tamura, H. ZEM 2 All Project (Zero Emission Mobility to All). In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Gran Canaria, Spain, 15–18 September 2015; pp. 2971–2975. [CrossRef]
38. Garofalaki, Z.; Kosmanos, D.; Moschoyiannis, S.; Kallergis, D.; Douligeris, C. Electric Vehicle Charging: A Survey on the Security Issues and Challenges of the Open Charge Point Protocol (OCPP). *IEEE Commun. Surv. Tutor.* **2022**, 1. [CrossRef]
39. Alfalouji, Q.; Schranz, T.; Kümpel, A.; Schraven, M.; Storek, T.; Gross, S.; Monti, A.; Müller, D.; Schweiger, G. IoT Middleware Platforms for Smart Energy Systems: An Empirical Expert Survey. *Buildings* **2022**, *12*, 526. [CrossRef]
40. Licensing-Sentilo. Available online: https://www.sentilo.io/wordpress/sentilo-about-project/licensing/ (accessed on 27 May 2022).
41. Dornberg, J.H.; Lutz, T. Selected findings from a German lighthouse project on electric mobility: A summary of outcomes in the field of vehicle to grid communication, ICT- and business model solutions. In Proceedings of the 2011 11th International Conference on ITS Telecommunications, St. Petersburg, Russia, 2–25 August 2011; pp. 486–491. [CrossRef]
42. Fabbri, G.; Dessì, M.; Mascioli, F.F.; Paschero, M.; Sgreccia, S.; Anniballi, L.; Nardecchia, S. Bonifica 2.0: An Integrated Territorial System of Sustainable Mobility and Micro Smart Grids. In Proceedings of the 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE), St. Petersburg, Russia, 1–4 June 2014; pp. 1622–1627. [CrossRef]
43. MQTT - The Standard for IoT Messaging. Available online: https://mqtt.org/ (accessed on 27 May 2022).
44. Next.js by Vercel—The React Framework for Production. Available online: https://nextjs.org/ (accessed on 27 May 2022).
45. Pereira Ferreira, A.; Sinnott, R. A Performance Evaluation of Containers Running on Managed Kubernetes Services. In Proceedings of the 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Sydney, NSW, Australia, 11–13 December 2019; pp. 199–208.
46. Assembly, G. *Transforming Our World: The 2030 Agenda for Sustainable Development*; Technical Report; United Nations General Assembly: New York, NY, USA, 2015. Available online: https://sustainabledevelopment.un.org/content/documents/21252030%20Agenda%20for%20Sustainable%20Development%20web.pdf (accessed on 11 November 2022).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.