electronics

MDPI

Article

Distributed Platform for Offline and Online EV Charging Simulation

Joaquim Perez [1,†], Filipe Quintal [1,2,*,†] and Lucas Pereira [2,3]

[1] Faculdade de Ciências Exatas e da Engenharia, Universidade da Madeira, 9020-105 Madeira, Portugal; 2029015@student.uma.pt
[2] Interactive Technologies Institute, LARSyS, 1049-001 Lisbon, Portugal; lucas.pereira@tecnico.ulisboa.pt
[3] Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisbon, Portugal
* Correspondence: filipe.quintal@staff.uma.pt
† These authors contributed equally to this work.

Abstract: Efforts to enhance electric vehicle (EV) charging processes have spurred the emergence of smart charging algorithms. However, these studies are intricate and costly, necessitating preliminary simulations to assess EV integration into power grids. Existing solutions to this issue tend to be limited to academia and proprietary systems. To address this, we propose a user-friendly and intuitive simulation tool employing a decoupled and flexible architecture. This architecture, achieved through open design and containerized microservices, streamlines maintenance, extension, and scalability. We substantiated the validity of our solution by simulating the charging infrastructure from an H2020 Research Project. Furthermore, we integrated our solution with an external system that executes smart charging algorithms. The proposed system yielded the desired results, enabling the project team to evaluate both the integration and algorithms, even amidst the COVID-19 lockdown.

Keywords: smart charging; simulation; electric vehicle; case study

Citation: Perez, J.; Quintal, F.; Pereira, L. Distributed Platform for Offline and Online EV Charging Simulation. *Electronics* **2023**, *12*, 4401. https://doi.org/10.3390/electronics12214401

Academic Editor: Fabio Corti

Received: 20 September 2023
Revised: 17 October 2023
Accepted: 22 October 2023
Published: 25 October 2023

1. Introduction

Global warming is one of the most significant challenges facing humanity today. To address this issue, a wide variety of measures has been implemented worldwide, encompassing various areas, notably public campaigns and the development of greener technologies. This work concentrates on two of the most crucial areas: energy and transportation.

One of the most prominent factors contributing to global warming is the continuous emission of CO_2 by the majority of vehicles used in our daily routines. One approach to mitigate these emissions is to transition the transportation sector toward cleaner alternatives, such as the adoption of electric vehicles (EVs) [1]. This shift has prompted the rise of EVs, which are currently being produced and offered by manufacturers. In alignment with this trend, governments worldwide are actively promoting the adoption of these vehicles, offering incentives such as tax breaks or free parking.

On the other hand, despite the growth of electrical networks over the years [2], they still struggle to keep pace with the rapid evolution of EVs [1] and are often unable to meet the increasing demand for electric vehicles. The surging adoption of EVs places a considerable strain on existing electrical networks, many of which are outdated and ill-prepared for this new demand [3]. For instance, according to an OFGEM report, it is estimated that approximately one-third of the electrical networks in the UK would require upgrades if around 40.

Consequently, it is hypothesized that the grid could revert back to fossil fuels to produce enough energy to address this increasing demand [4]. This obstruction takes us back to our original problem presented in the first place (our CO_2 footprint).

Traditionally, the charging process involves the plug-and-charge method, which simply charges the vehicle in an uncontrolled and uncoordinated manner whenever it needs

charging. This approach leads to theoretical issues concerning the electric network's ability to meet the demand and charge EVs without overloading the network, as described in the previous paragraph. To tackle this issue, the field of smart charging [5] introduces intelligent and coordinated charging methods that take into account various factors, including energy cost and electricity network availability. Employing a diverse range of algorithms, these coordinated charging methods lead to a more balanced distribution of the load in electrical networks and reduced peak loads [6,7].

1.1. Problem Statement

From a practical standpoint, studies on smart charging algorithms are feasible but challenging; expensive; and, in some situations, potentially hazardous. This is due to the complex management and coordination required in terms of electric power and road transport systems [8]. Thus, to successfully analyze and predict the integration of EVs in electrical networks, these studies are usually first conducted using simulations [8].

Data shortages pose a challenge in assessing algorithms and machine learning in general [9]. Simulations can play a crucial role in expanding datasets and enhancing decision making and predictions in such systems [10].

Therefore, as is common in the area of smart charging, a lack of data (or a lack of non-noisy data) leads up to testing being performed based on the simulation of factors such as the usage of EVs, the grid's load, the cost of energy, and the charging of batteries in general (for example, [11–14]).

1.2. Simulating Smart Charging Scenarios

There are simulation solutions available that fit smart charging parameters. Still, they largely consist of either academic or proprietary solutions or limited/specific solutions for a particular context that require a significant amount of customization effort from researchers/practitioners for their intended context (even if the required changes are minimal) [15,16]. Furthermore, the literature review carried out in this paper revealed that in terms of architecture, the existing solutions are usually developed in a monolithic way, containing a single code base, resulting in solutions that are rigid and challenging to adjust. Besides that, technologically speaking, there are not many available frameworks that integrate this kind of simulation while being open-source, extensible, and easy to use [15,16], which obstructs potential reusability or customization in other contexts.

To overcome this, we propose a simulation platform that allows researchers to simulate any set of smart charging algorithms under different conditions with enough abstraction to fit any charging simulation context, using an open design both in terms of technologies and architecture.

1.3. Contributions and Paper Organization

The proposed system comprises an open-source platform designed to simulate smart charging algorithms in a versatile manner that can adapt to various contexts and accommodate a variable array of data models. In simpler terms, it employs modularity in both simulation algorithms (smart charging algorithms) and simulation components (which are defined by users and can include items such as batteries, chargers, and charging time). Additionally, the proposed simulation system includes two other essential components: a data server (for exposition of simulation data via a web server) and a Web client (to provide an interactive dashboard displaying the simulation status).The solution is technically open and easily extensible to any energy simulation context.

The remainder of this paper is organized as follows. In the next section, we present several simulation platforms that motivated our approach. In this review, we mainly focus on the technologies used for each of the reviewed platforms. After this review, the proposed solution is presented in Section 3, where the solution attempts to fill the gaps and extend the reviewed work, especially considering how open-source tools could be used to develop it. Section 4 presents a case study in which the proposed system was evaluated within

a smart charging pilot that was halted due to the COVID-19 pandemic. The results of the simulation are presented and discussed in Section 5. Finally, we present the main conclusions in the last section.

2. Related Work

The challenge of developing simulation platforms for transportation and charging has inspired a significant body of work. In this section, we introduce the reviewed platforms designed to simulate entire or partial smart charging scenarios. For each tool, we provide a brief overview of its scope, strengths, weaknesses, and how it has influenced our approach.

When it comes to battery energy storage systems (BESS), the paramount objectives are to reduce energy costs and enhance efficiency. Given the multitude of adjustable parameters associated with storage system design, the University of Applied Sciences in Berlin developed a system called PerMod in MATLAB [17] that enables the end user to simulate and analyze the performance of a particular energy storage system.

Also in the context of evaluating BESS, SimSES [18] was developed. This simulation system allows for analysis not only from the technical perspective but also from the economic standpoint. In its initial phase, this tool was developed in MATLAB but was later ported to open-source Python software [19], resulting in a technically modern and flexible solution.

Focusing on the precise context of BESS and electric vehicles and with the goal of economically evaluating EVs, the National Renewable Energy Laboratory (NREL) developed the BLAST system [20] to predict battery responses according to battery properties (such as degradation and thermal performance), as well as usage and historic climate data. However, with BLAST, researchers only have access to its installer (and its binary files). The same entity, under the more specific context of renewable energy, developed and distributed SAM, a simulation platform [21]. It comprises a desktop application that empowers end users to simulate renewable energy projects and analyze technoeconomic aspects, including performance, financial metrics, and available incentives for such projects. This open-source platform was built using C/C++. Within the same field of renewable energy, which is characterized by the unpredictability of its sources (such as solar and wind), the German Aerospace Center developed FreeGreenius [22]. Based on the given input (meteorological and economic data), the tool is capable of simulating renewable power plants from an economic point of view, taking into account investment costs and financing costs and considering a certain period. In order to simulate power electronic converters and motor drives in general, the PSIM proprietary software (https://powersimtech.com/products/psim, version 1.0) was developed. This tool allows for the development/modeling of custom module boxes [23]. These custom developments enable this tool to concretely simulate and study conventional vehicles, EVs, and hybrid electric vehicles (HEVs).

In the same context (automotive systems), the JANUS [24] simulation package was developed by the Engineering Department of Durham University. It consists of a program that allows the end user to evaluate the design, performance, and efficiency of vehicles. The program was written to allow for possible extensions. To that end, its structural approach consists of separating each vehicle component into a separate subroutine. Also contributing to the subject of automotive systems, Texas A&M University developed a MatLab/Simulink simulation/modeling package, V-Elph [25], to ease the analysis and comparison of EV and HEV setups and energy management strategies. This tool allows the end user to perform simulations based on the selected component model (including component models designed for general and user-defined/customized use cases). G. H. Cole proposed SIMPLEV [26] as an alternative to systems targeting vehicle simulations. It consists of a program written in BASIC that provides the end user with the possibility of performing parametric studies of EVs. Considering it was developed in BASIC, SIMPLEV turns is a significantly restricted and bounded solution due to the minimal capabilities of the adopted programming language to build it. ADVISOR [27] is another tool with similar objectives that emerged from NREL. This tool was made publicly available, encouraging

continuous development in terms of flexibility. As ADVISOR was built in MATLAB, its portability and versatility are hindered compared with other work reviewed in this section.

Similarly, for the analysis of hybrid electric vehicle (HEV) systems, Argonne National Laboratory (ANL) developed the MARVEL program [28]. In the context of simulation, MARVEL requires input data from the end user regarding the driving cycle and the battery. However, due to its core being built in FORTRAN, it shares the same technological limitations as the JANUS tool, making it an outdated and restrictive tool in terms of programming capabilities.

Architecturally speaking, we have observed from the analysis presented above that the reviewed works have adopted more closed and rigid architectures, making them difficult to adapt to other contexts and affecting their extensibility and scalability.

Therefore, we argue that a monolithic approach is not ideal for this kind of simulator. Instead, we believe a microservice [29] approach is more suitable, since it results in higher levels of scalability, maintainability, and versatility (in terms of programming languages used in the several microservices).

3. Proposed Solution

As a result of the thorough analysis of the state of the art, several points were taken into account for the functional and technical aspects of the proposed solution. It is imperative to clarify that the proposed solution does not attempt to make advancements in the current state of the art concerning the enhancement of models, algorithms, use cases, or general machine learning methodologies utilized within the simulation domain. Instead, the principal aim of this proposal lies in the expansion of the field, with a specific focus on augmenting attributes such as openness, configurability, extensibility, and ease of installation. The ultimate objective is to develop a versatile tool capable of accommodating and implementing the diverse range of simulation approaches that have been reviewed.

Many of the reviewed tools follow strict and closed design patterns. This inspired our project to not only embrace open design patterns and approaches but also to be developed as a fully open-source project.

Moreover, a significant number of existing tools rely on MATLAB, either partially or exclusively, leading to proprietary and inflexible solutions. In contrast, tools like SIMses were built in Python, resulting in more open and extensible implementations. You can find a detailed description of our implementation in [30].

Following this approach, we chose to write our proposal in Python, providing an open development platform built upon more versatile programming languages and libraries. Similar to SAM, this choice enabled the creation of custom enhancements in a range of compatible programming languages, enhancing the system's extensibility. Additionally, we adopted a structural approach similar to that of JANUS, where each vehicle component is separated into distinct subroutines. This approach makes our code base easier to maintain, read, and extend, as different researchers can work on various modules independently. It also allows for the usage of user-defined or customized models, as demonstrated by V-Elph.

To build upon these choices, in addition to selecting Python as our programming language, we adopted a microservice architecture. This decision brings about higher scalability; supports easier deployments; and, most importantly, ensures versatility and technical agnosticism regarding programming languages. Each component can be developed independently in the language that suits it best.

Lastly, one of our core considerations was the observation that most of the reviewed work consists of desktop applications. These types of applications typically demand more time for setup and maintenance, along with specific software and hardware requirements for execution. In contrast, by opting for a Web-based approach, our proposed solution ensures that researchers can easily access it from any device, with minimal to no setup required. This not only reduces development costs but also guarantees consistency across various systems

3.1. Architecture

The architecture of the developed simulator consists of four main modules (simulator, gateway, data, and Web client) separated into Docker containers. Each module runs independently and can be rebuilt in any programming language as long as the communication protocol is respected. An overview of the architecture is provided in Figure 1.

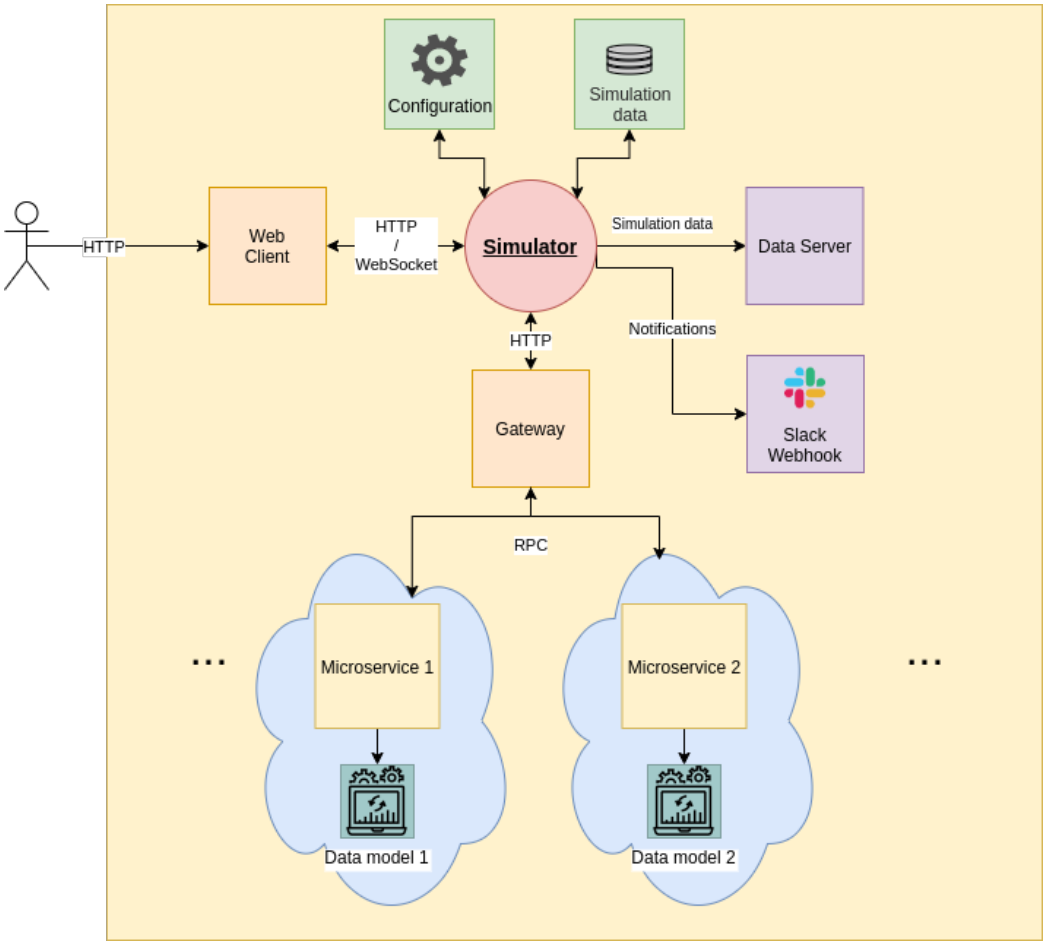

Figure 1. Functional diagram of the simulator showing its modules.

3.1.1. Simulator

The simulator's core serves as a wrapper class for the main functionalities of this system, such as initiation and stopping of the simulations, the WS message receiving/sending process, and database export.

The core also contains other functionalities, such as exposing the external REST API endpoints, handling the configuration of the simulation, unifying the logging process, notifying the research team through a Slack (https://slack.com/) channel, and providing persistent data handling through a database, among other functionalities needed for the setup, as well as debugging of the simulation.

3.1.2. Gateway

The gateway is used to centralize the communication between the simulator and the data models. It is implemented as a microservice encapsulated in a Docker container that serves a set of HTTP end points, which, when called, delegate its logic to the respective data model (via the respective RPC proxy). Afterward, the corresponding data model handles the request and returns a response according to its business logic.

3.1.3. Data Models

In the current version, six data models were modeled after the data collected in the real-world deployment. Each of the models is briefly described next.

- Travel-Affluence: This model is intended to serve as a proxy for the number of trips to be simulated each day. Regarding the travel affluence, the model consists on the calculation of the travel affluence according to a certain hour of day as an input.
- Travel—Distance: This model acts as a proxy for trip distance in km. It was modeled by selecting a random travel distance from the set of real travel distances recorded during the real-world deployment of the monitoring platform.
- Travel—Duration: In the same way, the logic involved in the duration model plainly revolves around a random selection of a travel duration from the dataset of previously collected real travels.
- Travel—Final Battery Level: The model concerning the final battery level of a car at the end of travel consists of a model that, based on the initial battery level and the travel distance as input, calculates the final battery level for the car based on a linear regression calculation.
- Charging—Duration: For the charging model, it was assumed that, if possible, vehicles charge up to 100%.
  The charging duration was modeled using a random charging duration collected from a set of previously acquired complete charging sessions. The instantaneous power consumption depends on the duration of the charging session and the battery state of charge.
- Charging—Energy: The model of total energy consumed, based on progress during the charging period (in percentage), calculates the energy used during charging.

All data models are encapsulated in microservices within individual Docker containers. As it happens for the gateway, their respective Docker images are started, and they connect to the same RabbitMQ message broker through the Nameko framework. With the connection set up, the data models are ready to be called up from the gateway.

3.1.4. Web Client

A Web-based client application was developed to provide researchers with an accessible means of interacting with the simulation. Furthermore, the Web client facilitates real-time visualization of the status of each simulated component. This tool enables various members of the research team to remotely configure and analyze the simulation's status without requiring direct configuration of the distinct containers and without necessitating any specialized skill sets. Within this Web client, researchers can configure various parameters of the simulation, including but not limited to the number of vehicles, simulation steps, and time intervals between steps. Configuration is accomplished by editing a config file (see Listing 1), where all those parameters are defined. In practice, this allows researchers to define, for example, the duration of the simulation or the number of points generated per cycle, or define URLs needed for external integration. The Web client then interacts with the simulator and consumes its data. Like all the components described above, the Web client is also contained withing a independent Docker image. The configuration used for our case study is presented below. The user can also visualize past simulation days, as presented in Figure 2. It is also possible to visualize the state of different variables of the simulation such as the SoC and consumption of the scooters (see Figure 3).

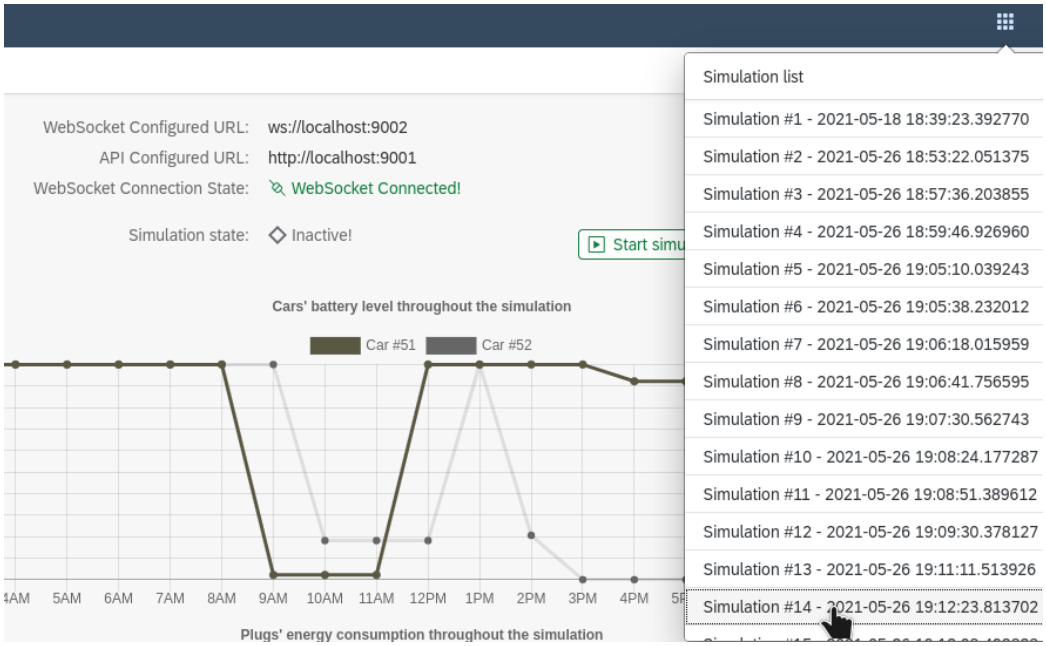

Figure 2. Simulation visualization in the Web client, showing a list with the different performed simulations.

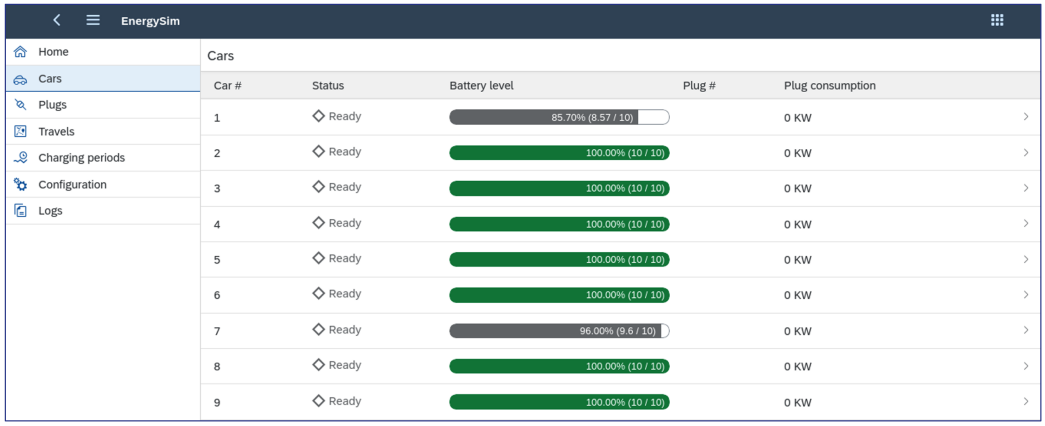

Figure 3. Simulation visualization in the Web client, showing a list with the status of simulated scooters. A similar view is also available to analyze the status of each charging point.

Listing 1. Configuration file of the Simulator as used in the case study.

```
{
  "number_of_cars": 10,
  "number_of_charging_plugs": 4,
  "sim_sampling_rate": 900000,
  "travel_affluence_multiplier": 1,
  "minutes_per_sim_step": 15,
  "number_of_steps": 96,
  "gateway_request_base_url": "http://cont_energysim_gateway:8000/{}",
  "enable_debug_mode": false,
  "webhook_url": "https://hooks.slack.com/services/XXXXXXXXX"
}
```

3.2. Development Tools

In terms of the development itself, as mentioned before, its core and main modules were be built in Python (https://www.python.org), as its containerization was ac-

complished using Docker (https://www.docker.com) and the whole scripting side was developed using Makefiles (https://www.gnu.org/software/make).

Furthermore, SQLObjects (http://www.sqlobject.org) were be used to handle the connection between the Python objects and the database through object-relational mapping classes. For the implementation of the models themselves, the following frameworks were used: TensorFlow (https://www.tensorflow.org) for their development and training, Nameko (https://nameko.readthedocs.io) for the construction of the microservices, and RabbitMQ (https://www.rabbitmq.com) as a message broker for the microservices.

The WebSockets (https://websockets.readthedocs.io) library was used to serve as a WS between the simulator and its web client. In addition, the solution uses Flask (https://flask.palletsprojects.com) for setup of the external REST API, serving as the Web client's static files. Regarding the web client, OpenUI5 (https://openui5.org/) was used as the front-end UI framework.

4. Evaluation and Analysis

As a case study for the simulation platform, we present how this platform was used to simulate the data from a smart charging pilot within the scope of a Horizon 2020 project. An EV pilot was carried out by a company that provides tourist tours on electric scooters (commonly known as Tuks). The goal of this pilot was to implement smart charging techniques in small vehicles with low energy requirements (e.g., scooters). A detailed presentation of the pilot scope and hardware is provided in [31]. However, the pilot was completely halted due to the COVID-19 pandemic. When the lockdown happened, most of the monitoring hardware had been running for approximately 2 years, and the smart charging intervention had just started. This left the team without an adequate approach to evaluate the whole pilot, especially the charging algorithms and communication with the third party responsible for implementing them. As an alternative evaluation, it was decided to model the entire pilot within the proposed platform. This case study allowed us to evaluate the proposed system considering several aspects. First, the installability of the simulator was assessed by installing and configuring the system on a server. Secondly, it allowed us to test the adaptability of the simulator to a specific use case. Finally, we were also able to evaluate the integration of the simulator with third-party services. This evaluation is crucial, since it can also help to identify whether, when used in real-life scenarios, the implemented solution fulfills the objectives set out in its proposal. The configuration used during this period is presented in Listing 1.

4.1. Integration with a Third-Party Route Management Algorithm

In this particular use case, the platform had to interact with a company that, within the scope of this paper, is referred to as Smart-Charging-Company (SMC). SMC offers a series of route management services. Within the scope of this research project, SMC was responsible for generating smart charging schedules using a proprietary algorithm that attempts to maximize renewables usage at the whole-island level. SMC's algorithm was modeled using a service that provides real-time local energy production by quota (e.g., thermal, wind, and hydro). This service is updated every 15 min, and the algorithm also uses real-time weather forecasts gathered from an online source. The output from the smart charging service produces commands with a granularity of 15 min.This partner originally had access to real data gathered from the pilot. However, once the COVID-19 restrictions were put in place, there were no data for SMC to use. A proposed approach involved the partner's route optimization planning production system consuming the REST API routes available in the simulator. Afterward, SMC could also act upon the simulation through the REST API (see Figure 4). A sample request is presented in Listing 2.

In the original implementation, SMC was expecting to act upon the charging loads in real time. With this in mind, the simulator was set up to run in simulated real time. Each cycle of simulation lasts 24 h, and the simulation runs in steps of 15 min (see Listing 1 for the full configuration used in the case study).

It is imperative to state that the objective of this study was not the development or evaluation of smart charging algorithms. While SMC algorithms utilize data from the site and the simulation, it is important to note that the algorithm was entirely developed in house. This case study serves to illustrate how the simulation can interact with a completely separate algorithm, thanks to its open design.

Listing 2. Sample smart-charging command received by the simulator in this example plug 4 gets disabled.

```
05/Sep/2021 10:45:00 "GET /plugs/4/disabled HTTP/1.10"
```

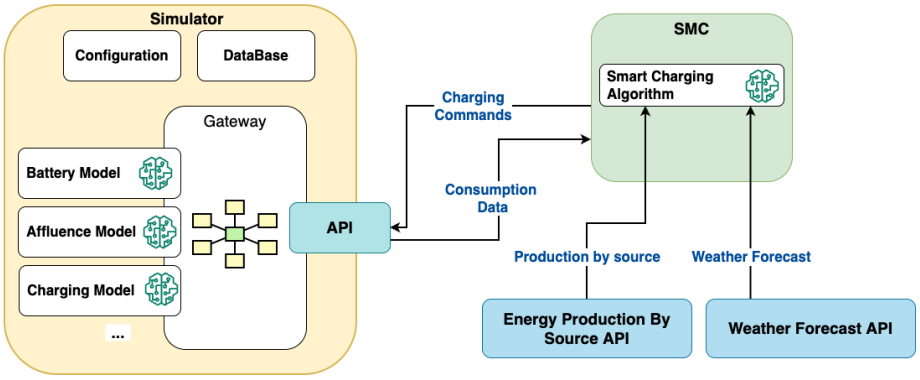

Figure 4. Overall representation of the communication between the Simulator and the SMC for the case study.

4.2. Modeling the Solution

Although our contribution is not centered on a specific case study, the process outlined below demonstrates how to integrate data from various sources into the proposed platform.

The models presented below were primarily built using a sample of baseline data, including real consumption, travel patterns, and affluence data collected over a two-year period prior to this study. In practical terms, the process is outlined as follows:

1. Extraction of the data from the data source;
2. Processing of the data;
3. Creation of the formulas of the mathematical models.

For the modeled case study, the the following entities were considered:

- Travel;
- Battery consumption;
- Charging;
- Affluence.

The first data model encompasses information related to the distances traveled during scooter routes on tours. This model was constructed using all previous travel data collected in the pilot study. It calculates the average travel distance and the standard deviation of the distance. Each generated travel is calculated using the average, with random values added within the standard deviation.

For the Battery consumption model, the average and standard deviation of scooter battery consumption per kilometer were calculated based on baseline data measured before the intervention. This model calculates battery consumption for a given travel by first generating a value of consumption per kilometer, which is the average consumption per kilometer plus a random value within its standard deviation. This value is then multiplied by the distance (generated from the travel model), providing the battery consumption for a given travel. The final battery state of charge (SoC) is determined by subtracting this consumption from the initial SoC value.

The charging duration model follows a similar approach. The average and standard deviation of charging duration and charging peak values were calculated. Using this information, the model generates the charging duration by adding the average value to a random value within its standard deviation. The generation of charging peak values follows the same method.

The data models presented here serve as a basic demonstration of the process involved in integrating and modeling data from external sources into the proposed platform. This illustrates the flexibility and versatility of our solution, which allows for the seamless incorporation of data from various sources.

Simulation Report

During the evaluation, the simulator completed 61 full cycles of days (31 days for August and 30 for September). Figure 5 displays various samples of affluence and trip distance. Concerning affluence, it is evident that the simulation tends to generate higher demand for scooters in the morning. In the right-hand graph, we can observe the wide range of trips that the simulator can generate. The simulation followed the models briefly presented earlier. For example, Figure 6 illustrates the relationship between distance traveled and battery consumption for fifteen random trips generated during the simulation. Regarding the integration with SMC, the simulation received 1208 charging commands and generated more than 58,000 consumption points. After one week of testing, the simulation ran continuously without any issues. The simulation started on the 1st of August 2021 and ended on the 30th of September. During the month of August, the simulation did not receive any smart charging commands from SMC. Smart charging was introduced for the whole month of September.

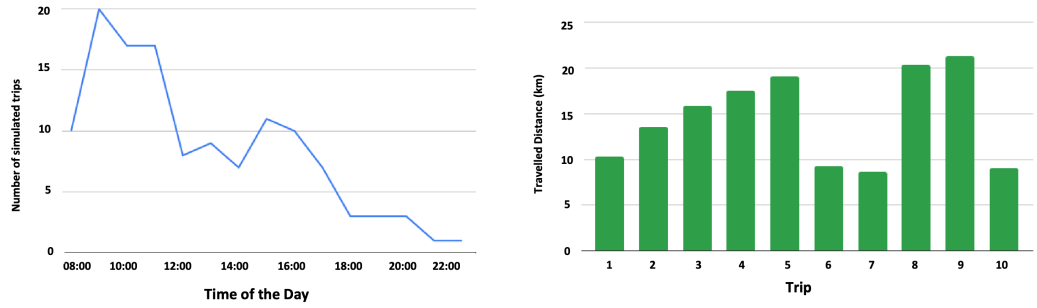

Figure 5. Affluence and travels samples generated during some of the simulation runs.

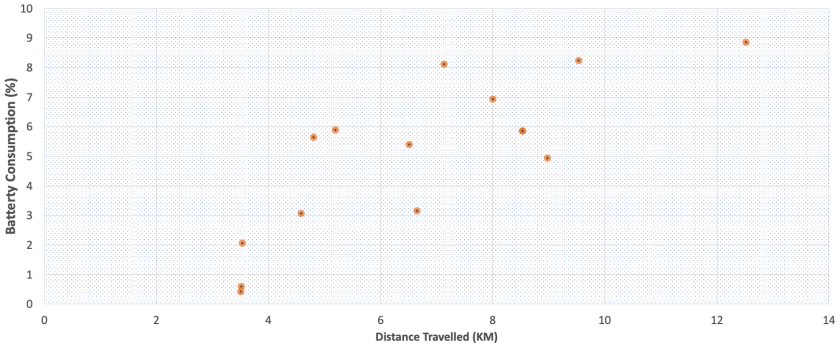

Figure 6. Representation of the relation between the distance and battery consumption of fifteen random trips generated during the simulation.

As mentioned above, during the simulation, there were 1208 instances of smart charging events. These are, simply put, ON or OFF commands directed at specific charging points forwarded by SMC based on the output of their algorithm. Listing 2 shows an example of a received command from SMC.

In simpler terms, there were more than 300 charging hours that were shifted according to the weather forecast and other inputs used by SMC. The commands significantly affected how charging occurred during the period of the simulation.

Figure 7 presents how the battery SOC of two scooters evolved during a run of the simulation with smart charging. The same comparison is presented in Figure 8; again, the difference between the charging pattern at the charging point is quite visible. Consumption occurred in shorter bursts.

In general, the data show that there were fewer charging periods when smart charging was active: 612 compared to 655 without smart-charging. On the other hand, charging lasted, on average, 10 min longer under smart charging. This observation can be explained by the fact that actuation cut the power, which may have resulted in a longer charging period during smart charging. These were periods during which the algorithm decided to turn off the power from the charging points, delaying or postponing charging according to its inputs, again with a focus on renewable energy availability. However, in general, the smart charging intervention did not affect the total energy usage, nor did it affect the travels simulated by the platform. During the month of August, there was a total of 1543 simulated travels, and for September, the platform simulated 1433 travels. The travels had an average length of 12.3 km (*SD* = 3.5 km) in August and 12.5 km (*SD* = 3.4 km) in September. Regarding energy consumption, each travel consumed an average of 4.6 kWh of energy in both the months of August (*SD* = 3.5) and September (*SD* = 3.4). Lastly, on average, each simulated day consumed an 91.6 kWh (*SD* = 14.2kWh) in August and 85.8 kWh (*SD* = 5.4) in September.

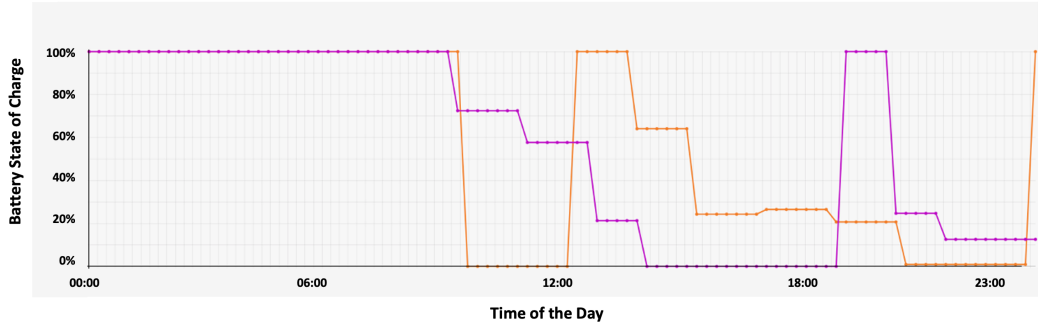

Figure 7. Battery state of charge for two scooters during a simulation run. Each line represents a different scooter present in the simulation.

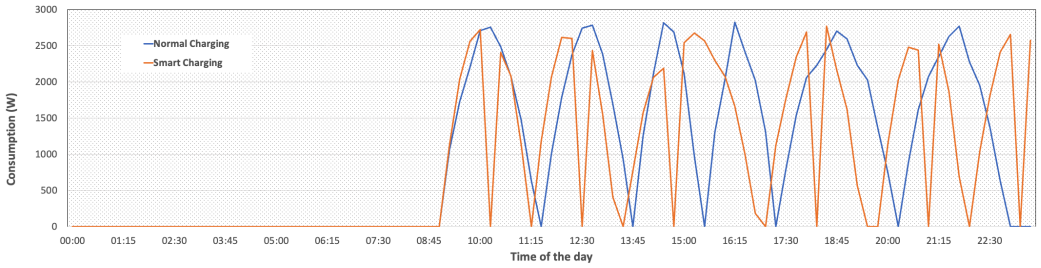

Figure 8. Consumption of one of the charging points during a simulation run with and without smart charging.

5. Discussion

The proposed solution entails an open-source and easily extendable simulator for EV charging data, which relies on models of different factors impacting consumption, such as battery capacity and usage patterns. The research team identified a gap in the state of the art for a user-friendly system that could be conveniently extended, developed using open-source tools, and accessible via a web client. In summary, we believe that the system successfully fulfilled its intended purpose. It enabled the modeling of a specific case study,

allowing researchers to continue experimenting with smart charging algorithms, even during periods when data were scarce.

Development of a Smart Charging Simulator

Unlike existing tools, such as PerMod [17], SA [21], JANUS [24], V-Elph [25], SIMPLEV [26], ADVISOR [27], and MARVEL [28] (and similarly to SIMSes [19]), this solution is built on Python, a more developer-friendly programming language that results in code that is easier to read, simpler, and flexible.

In the same way, in contrast to tools like PerMod [17], FreeGreenius [22], PSIM [23], JANUS [24], V-Elph [25], BLAST [20], SIMPLEV [26], and MARVEL [28] (and as adopted in SIMSes [19] and ADVISOR [27]), this solution was made to be open-source and available to the public community. Motivated by SAM [21], PSIM [23], JANUS [24], V-Elph [25], and ADVISOR [27], this simulator adopts an open architecture and used open design approaches, making it more favorable for future enhancements/modifications. The addition of microservices also accomplishes this, providing increased scalability of the solution, making the deployment process easier, largely reducing downtime, and making it simpler to maintain/extend. This enables the implementation of each microservice in any programming language. In situations in which the simulator needs another data model, the following are needed: the creation of a microservice and its implementation (with the model), including the respective RPC proxy in the gateway, as well as a corresponding HTTP call in the simulator. Consequently, this results in a decoupled and flexible system for maintenance and development. The User interface of the proposed system was highly motivated by SIMSes [19], SAM [21], FreeGreenius [22], PSIM [23], JANUS [24], V-Elph [25], BLAT [20], SIMPLEV [26], and ADVISOR [27] (while not present in PerMod [17] and MARVEL [28]). It also provides a visual representation of the simulated data in order to provide an easy and hands-on way to analyze the same data.

However, as opposed to all the related work reviewed herein, the proposed solution includes a Web client. This client provides researchers with a solution within their reach, with no setup involved and no device specificity, bringing more practicality and convenience. In addition, with this Web approach, the simulation results can be consumed by other researchers. Moreover, the UI allows the end users to start/stop simulations, configure the simulator, and browse through the simulation data in real time. This feature also allowed all the researchers from the SMILE research project to monitor the state of the simulation in real time and to visualize passed simulated days.

6. Conclusions

In summary, owing to the necessity of improving the charging process of EVs, the field of smart charging has emerged. However, the studies revolving around this field are complex and could be expensive and risky. Consequently, this leads to a need for prior simulations to analyze/predict the integration of EVs in electrical networks. Some simulators are already available, yet they consist of either academic, proprietary, or limited/rigid solutions. Therefore, we have presented a solution that provides a handy and intuitive tool for researchers to simulate scenarios backed up by a simulation system with a decoupled architecture. Such an architecture is materialized by adopting open design approaches and the concept of containerized microservices, making the process of maintaining/extending easier, in addition to providing a high level of scalability. When evaluating this solution, we can argue that it fulfilled its objectives of solving the problem the research team faced, in addition to contributing to the field with an open platform to test smart charging algorithms. Furthermore, the solution ran on a production system while also being consumed externally by a third-party partner, which corroborates the solution's relevance. Given the size of the baseline sample, more advanced models could have been built; however, this was not the purpose of this study. Furthermore, considering the solution's flexibility, more advanced models could have been easily integrated.

The developed platform is available in various Git repositories [32].

Future Work

There are plenty of improvements that could be implemented in complex platforms such as that proposed herein, for instance, the development of a travel route renderer and additional data models and integration of the simulator with its gateway and Web client. Another potential improvement is the refinement of the data models to incorporate the full potential of TensorFlow models, utilizing fully self-learning models to advance the simulation process. By incorporating the full potential of TensorFlow, these data models can be sufficiently advanced to be used for other purposes besides this simulator.

However, given the current state of the platform, a more exhaustive evaluation is required further test its flexibility, including new models, interaction with new data sources, or the inclusion of real hardware in the loop, since the simulator can be configured to run in real time.

Author Contributions: Conceptualization, F.Q. and L.P.; methodology, J.P., F.Q. and L.P.; software, J.P.; validation, J.P. and F.Q.; formal analysis, F.Q.; investigation, J.P.; data curation, F.Q.; writing—original draft preparation, F.Q. and L.P.; writing—review and editing, F.Q. and L.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Union Horizon 2020 research and innovation programme under grant agreement number 731249 and by the Fundação para a Ciência e Tecnologia under grant number UID/EEA/50009/2019.

Data Availability Statement: All the data generated during the simulation are available upon request.

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

Abbreviations

The following abbreviations are used in this manuscript:

COVID-19	Coronavirus Disease 2019
OFGEM	Office of Gas and Electricity Markets
EV	Electric vehicle
BESS	Battery energy storage system
REST	Representational state transfer
API	Application programming interface
RPC	Remote procedure call
WS	Web Service
SMC	Smart Charging Company
SoC	State of Charge
SD	Linear dichroism

References

1. Wu, M.; Chen, W. Forecast of Electric Vehicle Sales in the World and China Based on PCA-GRNN. *Sustainability* **2022**, *14*, 2206. [CrossRef]
2. Rutter, P.; Keirstead, J. A brief history and the possible future of urban energy systems. *Energy Policy* **2012**, *50*, 72–80. [CrossRef]
3. Directorate-General for Energy (European Commission); Gallego Amores, S. *E-Mobility Deployment and Impact on Grids: Impact of EV and Charging Infrastructure on European T&D grids: Innovation Needs*; Publications Office of the European Union: Luxembourg, 2022.
4. Martins, M. Evaluation of Energy and Environmental Impacts of Electric Vehicles in Different Countries. Ph.D. Thesis, Instituto Superior Técnico, Lisbon, Portugal, 2015.
5. Sadeghian, O.; Oshnoei, A.; Mohammadi-Ivatloo, B.; Vahidinasab, V.; Anvari-Moghaddam, A. A comprehensive review on electric vehicles smart charging: Solutions, strategies, technologies, and challenges. *J. Energy Storage* **2022**, *54*, 105241. [CrossRef]
6. Mangipinto, A.; Lombardi, F.; Sanvito, F.D.; Pavičević, M.; Quoilin, S.; Colombo, E. Impact of mass-scale deployment of electric vehicles and benefits of smart charging across all European countries. *Appl. Energy* **2022**, *312*, 118676. [CrossRef]
7. Fachrizal, R.; Ramadhani, U.H.; Munkhammar, J.; Widén, J. Combined PV–EV hosting capacity assessment for a residential LV distribution grid with smart EV charging and PV curtailment. *Sustain. Energy, Grids Netw.* **2021**, *26*, 100445. [CrossRef]

8. Marmaras, C.; Xydas, E.; Cipcigan, L. Simulation of electric vehicle driver behaviour in road transport and electric power networks. *Transp. Res. Part C Emerg. Technol.* **2017**, *80*, 239–256. [CrossRef]
9. Hoffmann, J.; Bar-Sinai, Y.; Lee, L.M.; Andrejevic, J.; Mishra, S.; Rubinstein, S.M.; Rycroft, C.H. Machine learning in a data-limited regime: Augmenting experiments with synthetic data uncovers order in crumpled sheets. *Sci. Adv.* **2019**, *5*, eaau6792. [CrossRef] [PubMed]
10. Barring, M.; Johansson, B.; Flores-Garcia, E.; Bruch, J.; Wahlstrom, M. Challenges of data acquisition for simulation models of production systems in need of standards. In Proceedings of the 2018 Winter Simulation Conference (WSC), Gothenburg, Sweden, 9–12 December 2018; pp. 691–702. [CrossRef]
11. Mousavi G., S.M.; Nikdel, M. Various battery models for various simulation studies and applications. *Renew. Sustain. Energy Rev.* **2014**, *32*, 477–485. [CrossRef]
12. Zhou, F.; Li, Y.; Wang, W.; Pan, C. Integrated energy management of a smart community with electric vehicle charging using scenario based stochastic model predictive control. *Energy Build.* **2022**, *260*, 111916. [CrossRef]
13. Huang, Q.; Jia, Q.S.; Qiu, Z.; Guan, X.; Deconinck, G. Matching EV Charging Load With Uncertain Wind: A Simulation-Based Policy Improvement Approach. *IEEE Trans. Smart Grid* **2015**, *6*, 1425–1433. [CrossRef]
14. Yang, Z.; Li, K.; Foley, A. Computational scheduling methods for integrating plug-in electric vehicles with power systems: A review. *Renew. Sustain. Energy Rev.* **2015**, *51*, 396–416. [CrossRef]
15. Limpens, G.; Moret, S.; Jeanmart, H.; Maréchal, F. EnergyScope TD: A novel open-source model for regional energy systems. *Appl. Energy* **2019**, *255*, 113729. [CrossRef]
16. Hesse, H.; Schimpe, M.; Kucevic, D.; Jossen, A. Lithium-Ion Battery Storage for the Grid—A Review of Stationary Battery Storage System Design Tailored for Applications in Modern Power Grids. *Energies* **2017**, *10*, 2107. [CrossRef]
17. Weniger, J.; Tjaden, T.; Orth, N.; Maier, S. *Performance Simulation Model for PV-Battery Systems (PerMod)*; Technical Report; University of Applied Sciences Berlin (HTW Berlin): Berlin, Germany, 2019.
18. Naumann, M.; Truong, C.N.; Schimpe, M.; Kucevic, D.; Jossen, A.; Hesse, H.C. SimSES: Software for techno-economic Simulation of Stationary Energy Storage Systems. In Proceedings of the International ETG Congress 2017, Bonn, Germany, 28–29 November 2017; pp. 1–6.
19. Möller, M.; Kucevic, D.; Collath, N.; Parlikar, A.; Dotzauer, P.; Tepe, B.; Englberger, S.; Jossen, A.; Hesse, H. SimSES: A holistic simulation framework for modeling and analyzing stationary energy storage systems. *J. Energy Storage* **2022**, *49*, 103743. [CrossRef]
20. Neubauer, J. *Battery Lifetime Analysis and Simulation Tool (BLAST) Documentation*; Technical Report NREL/TP-5400-63246; National Renewable Energy Lab. (NREL): Golden, CO, USA, 2014. [CrossRef]
21. Blair, N.; DiOrio, N.; Freeman, J.; Gilman, P.; Janzou, S.; Neises, T.; Wagner, M. *System Advisor Model (SAM) General Description*, Version 2017.9.5; Technical Report; National Renewable Energy Laboratory: Golden, CO, USA, 2018.
22. Dersch, J.; Dieckmann, S. Techno-Economic Evaluation of Renewable Energy Projects using the Software greenius. *Int. J. Therm. Environ. Eng.* **2015**, *10*, 17–24. [CrossRef]
23. Onoda, S.; Emadi, A. PSIM-based modeling of automotive power systems: Conventional, electric, and hybrid electric vehicles. *IEEE Trans. Veh. Technol.* **2004**, *53*, 390–400. [CrossRef]
24. Bumby, J.R.; Clarke, P.H.; Forster, I. Computer modelling of the automotive energy requirements for internal combustion engine and battery electric-powered vehicles. *IEE Proc. A (Phys. Sci. Meas. Instrum. Manag. Educ. Rev.)* **1985**, *132*, 265–279. [CrossRef]
25. Butler, K.; Ehsani, M.; Kamath, P. A Matlab-based modeling and simulation package for electric and hybrid electric vehicle design. *IEEE Trans. Veh. Technol.* **1999**, *48*, 1770–1778. [CrossRef]
26. Cole, G.H. *SIMPLEV: A Simple Electric Vehicle Simulation Program*, Version 1.0. Technical Report DOE/ID-10293; EG and G Idaho, Inc.: Idaho Falls, ID, USA, 1991. [CrossRef]
27. Wipke, K.; Cuddy, M.; Burch, S. ADVISOR 2.1: A user-friendly advanced powertrain simulation using a combined backward/forward approach. *IEEE Trans. Veh. Technol.* **1999**, *48*, 1751–1761. [CrossRef]
28. Marr, W.W.; He, J. *MARVEL: A PC-Based Interactive Software Package for Life-Cycle Evaluations of Hybrid/Electric Vehicles*; Technical Report ANL/ES/CP-87322; CONF-9510282-1; Argonne National Lab.: Lemont, IL, USA, 1995.
29. Pump, R.; Koschel, A.; Ahlers, V. Applying Microservice Principles to Simulation Tools. In *SERVICE COMPUTATION 2019, The Eleventh International Conference on Advanced Service Computing*; IARIA: Washington, DC, USA, 2019.
30. Perez, M.J.A.S. Development and Analysis of an Open-Source Platform to Simulate Electric Vehicle Charging Needs. Master's Thesis, Universidade da Madeira, Funchal, Portugal, 2021.
31. Quintal, F.; Scuri, S.; Barreto, M.; Pereira, L.; Vasconcelos, D.; Pestana, D. MyTukxi: Low Cost Smart Charging for Small Scale EVs. In Proceedings of the Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA '19), Glasgow, Scotland, UK, 4–9 May 2019; pp. 1–6. [CrossRef]
32. Quintal, F. Simulator Repositories. GitHub Gist: Instantly Share Code, Notes, and Snippets. GitHub, United States. 2023. Available online: https://gist.github.com/Lipegno/2fbd1943a365b25efe7b425a193e9dc3 (accessed on 1 August 2023).