Inno-EAV: An open-source framework for high performance charging of electric vehicles

Ziyad Elbanna, Manuel Mazzara, Ilya Afanasyev, Luiz J.P. Araújo[1], Rasheed Hussain, Mansur Khazeev, and Dragos Strugar

Innopolis University, Innopolis 420500, Russia
[1] Corresponding author: l.araujo@innopolis.university

Abstract. Electric Autonomous Vehicles (EAVs) have gained increasing attention of industry, governments and scientific communities concerned about issues related to classic transportation including accidents and casualties, gas emissions and air pollution, intensive traffic and city viability. One of the aspects, however, that prevent a broader adoption of this technology is the need for human interference to charge EAVs, which is still mostly manual and time-wasting. This study approaches such a problem by introducing the Inno-EAV, an open-source charging framework for EAVs that employs machine-to-machine (M2M) communication. The idea behind M2M is to have networked devices that can interact, exchange information and perform actions without any manual assistance of humans. The advantages of the Inno-EAV include the automation of charging processes and the collection of relevant data that can support better decision making in the spheres of energy distribution and the prospect of adoption of this technology to mention a few. In this paper, we present the concepts involved in the introduced framework, the discussion about the back-end database that is used to store car owners, cars, and charging stations. The authors expect that the use of Inno-EAV will contribute to more efficient charging process while it enables more precise decision making.

Keywords: Autonomous cars; Electric cars; Machine-to-machine economy; Software engineering; Data science

1 Introduction

Electric Autonomous Vehicles (EAV) have been ubiquitously replacing standard vehicles since their invention in the 20^{th} century [2]. Research has shown the technology as a practical approach to solve transportation issues such as accidents [3], traffic congestion, and harmful gas emissions [10, 11]. Autonomous cars are already pervading our roads in the form of pilot execution from different stakeholders and are speculated to be on the verge of commercialisation [8]. It is also worth mentioning that autonomous cars are poised to have profound social and economic impacts on our lives [7]. Nevertheless, the predictions indicate that autonomous vehicles will have a positive effect on the environment.

The EAV technology is now emerging as an enabler to establish the foundation for the Machine-to-Machine (M2M) economy, i.e. networked devices that can interact, exchange information and perform actions without any manual assistance of humans. For this to happen, the market should be able to offer appropriate charging services without involving humans [12, 13].

In the recent years, wireless M2M communication [14, 15] has extended the features of wired communication machines to include onboard Global Positioning System (GPS), flexible surface mounting of land grid array, embedded M2M optimised smart cards (e.g. phone Subscriber Identity Modules (SIMs) known as M2M Identification Modules (MIMs). To enable such a vast number of possible applications, IoT application developers often gravitate toward the embedded Java as their programming language of choice due to its portability, flexibility and versatility. M2M systems often use public networks and access methods (e.g. cellular or Ethernet) to make it more cost-effective. The main components of M2M systems include sensors, Radio Frequency Identification (RFID), Wireless - Fidelity (WiFi) or cellular communications link and autonomic computing software programmed to aid the network device to interpret data and make decisions. Such M2M applications translate the data, which can trigger pre-programmed automated actions.

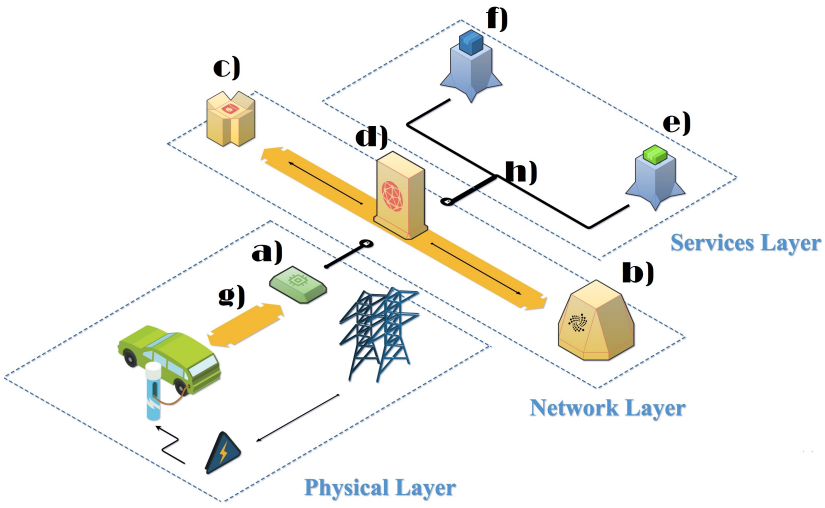

Fig. 1. Inno-EAV Architecture. Currently Inno-EAV consists of 3 layers dedicated to high performance as explained and depicted in [12], with labels refering to: a)Main Controller b)IOTA's tangle, c)Local database, d)Bidirectional Communication, e)EAV's app, f)SP application, g) Component connection in one layer and h) Secured connection between two layers respectively.

In this paper, we present a versatile open-source framework developed for improving and automating the charging process of electric vehicles and establishing the foundation of M2M economy using WiFi networks. Inno-EAV [1] is available as a set of open-source modules which can be used independently or concomitantly. The following layers constitute the framework (Fig. 1):

- **Physical layer**. It encapsulates all the hardware components embedded in the charging station (CS) used for sensing and gathering information about the charging process, as well as communicating with external entities.
- **Network Layer**. Since our scenario is composed of autonomous, dynamic decision-making CSs and vehicles, we suggest the communication between the entities in the system be entirely peer-to-peer (P2P) with distributed ledger technologies (DLTs) as a possible solution to store both data and value transactions [4, 5]. We also consider Message Queuing Telemetry Transport (MQTT) connectivity protocol to be the most appropriate choice for inter-device communication. In this way, the network layer would be able to transfer sensor data from the Physical Layer to the Services Layer rapidly and securely [12, 13].
- **Services Layer**. On top of the architecture sits the Services Layer, which makes use of the two layers described above to deliver services to both consumers and service providers. Such services include:
  - Charging services for EAVs;
  - Data Analysis for Service Providers.

The Inno-EAV framework is proposed to solve problems common to multiple application scenarios, and it is compatible with a multitude of environmental goals and application protocols.

2 The Inno-EAV framework

Inno-EAV is installed as a standard set of modules, and it is fully Java-based open-source and user-friendly (Fig. 2). Its routines can be easily integrated within larger hardware tools used to complete the charging process. Each of its modules possesses its separate manual and documentation. To date, it encompasses two additional Java Archive (JAR) packages (Apache, JSON) alongside with the existing original packages in the Java JDK. The modular nature of Inno-EAV allows its components to be updated independently and the framework to be easily extended by appending new packages.

The normal flow of Inno-EAV user use case is as follows:

- The car owner connects to the same network of the charging station, and send the JSON file to it. In Inno-EAV, we used JSON files as its a light, easy to read way of transmitting data over networks as well as being a native to javascript which makes it easier to extract data.
- The charging station verifies the car details by sending the file to the database system.

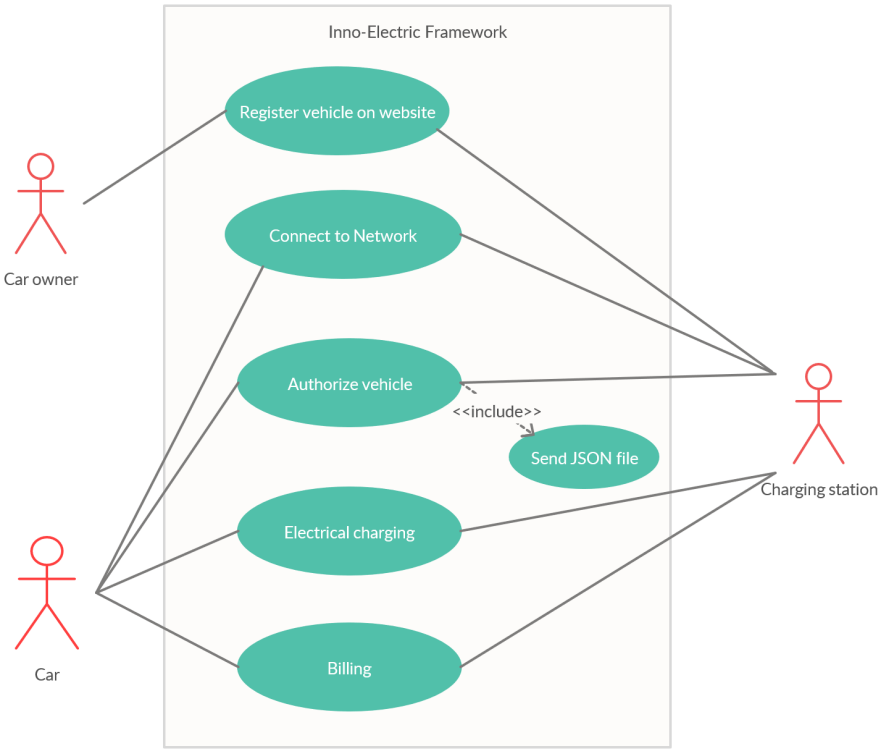

Fig. 2. UML Use case diagram explanation for Inno-EAV Framework. Inno-EAV enables the automatic charging of the electric car. Thes use case begins when an EAV owner connects to the same network as the charging station which it is registered to, and to send the JSON file to the charging station. The system sends the JSON file to the database system for verification. If the database system approves the car details, the server asks for the amount of electric charge and issues a receipt. The transactions are recorded in the database.

- The charging station then asks for the electricity amount needed.
- The car enters the amount.
- The bill is sent to the car.
- The car pays the bill to the charging station.

3 Network Communication

M2M communication commonly occurs during the acquisition of the same network to the server and the client, often as a result of the client connecting to the same network where the server is connected. We assume that the network IP address of the charging station is the server with a static IP address. While

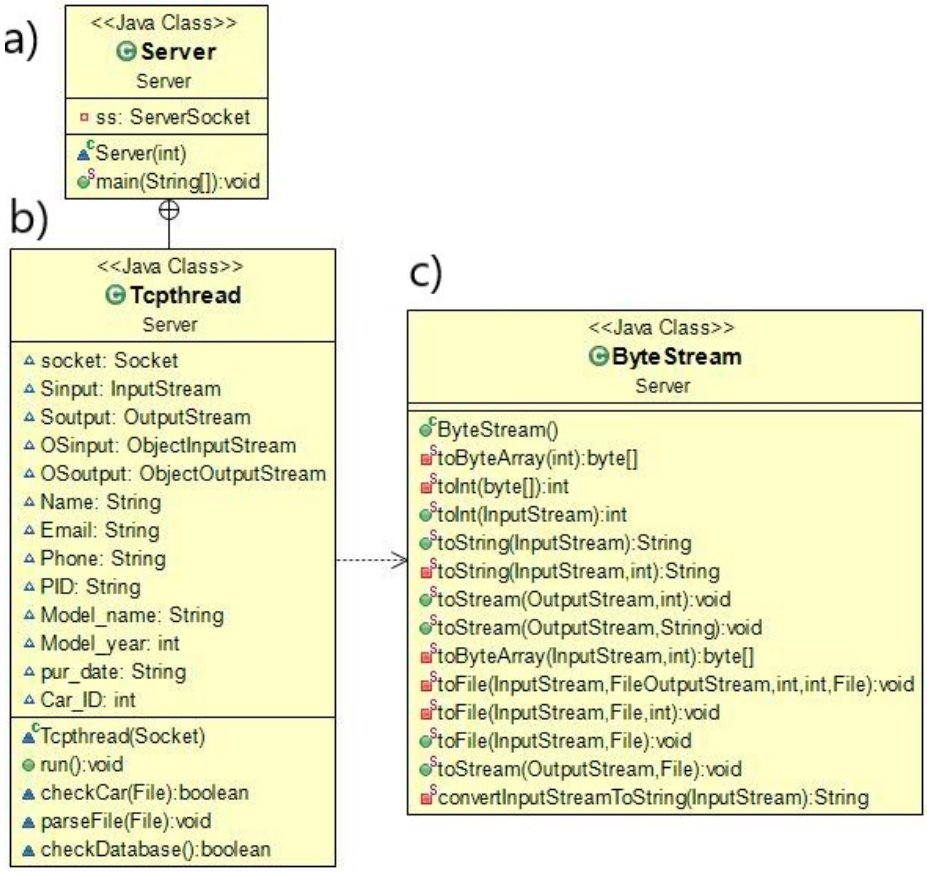

Fig. 3. UML class diagram for Server package. a) Image of Server class diagram with two main functions: Main, and Server. The latter class is used to specify the port number of connection. **b)** TCP thread class which extends the Thread class and is run inside Server routine shown in a). **c)** Byte Stream class used to send and get output streams to and from the client

different clients trying to charge from this server will have a different IP address each time they connect to the wifi network, the charging stations will still be prone to have different IP address each time they connect to the network (Fig. 3). However, in our case where the charging station is considered to be the server, we assume that it uses the same IP address each time it connects to the network and sets up a fixed static IP address for it.

Inno-EAV breaks the task of network communication into two distinct parts: Server socket creation from the server, followed by socket creation from the client.

Server - As a first step, Inno-EAV-Core charging station creates the server socket which represents it and waits for a connection from the client on a specific

port number, which is the listening port number (Fig. 3b). The number of the port is previously agreed on by both stakeholders in the connection that is later achieved as the server socket accepts the socket trying to connect to the listening port using socket programming in Java. Once the connection is established, the data of the client can be directly retrieved by getting the input stream using an instance of the ObjectInputStream class shown in (Fig. 3b). The data retrieving process will, however, change the current input stream of the listening port, which can have an influence on further stream writing required to exchange information between the client and the server.

For the specific case of our project with Object output stream, there will only be a few writes to the stream of the listening port, as there is only a few information to be exchanged between stakeholders. The first input stream to the server will be the file name of the client, such as 'test.json' for instance. As an option Inno-EAV-Core can store the incoming input lines in the stream within a new file in the sent documents, thus converting the input stream to a new file containing the client's information.

Client - Client package in Inno-EAV differs slightly from the server package, as it creates the socket with the specific port number and the fixed IP address of the server that we have set in previous steps. This allows the connection to be systematic once the client enters the region of the network to which the server is connected. Furthermore, Inno-EAV-Server (as described in the previous section) can accept the socket creation (Fig.4a) created by Inno-EAV-Client and use this connection directly to retrieve the data from the client.

As we discussed later, The sequence of Inno-EAV begins when the car connects to the same network that the charging station is connected to, as explained in and (Fig.4) shows the sequence diagram of Inno-EAV.

4 Charging Station Registration

In M2M communication between server and client, a file is required to be sent to the server at which is often unique for every car. This file has the following attributes: (Name, Email, Phone and Personal-ID) of the car owner and (Model-name, Model-year, and Date purchased) of the car. Each ID is unique, and the combination of the car owner ID and his car is also unique, likewise the ID of charging stations. However, after the file is sent to the server, it checks the combination of the car ID and owner ID against the database for authorization. Charging station registration is therefore essential for assigning the car of each owner to a specific charging station at which it can be charged from in the future. Our team created a web application aiming to ease the registration process for car owners to register their car to a particular station, and thus be able to charge from this station in the future. Therefore, Inno-EAV-Core Charging station registration offers a unique tool to perform this registration quickly and robustly on a wide variety of datasets.

The owner will be asked to enter his personal information through form filling, along with his car information, the information about the charging station

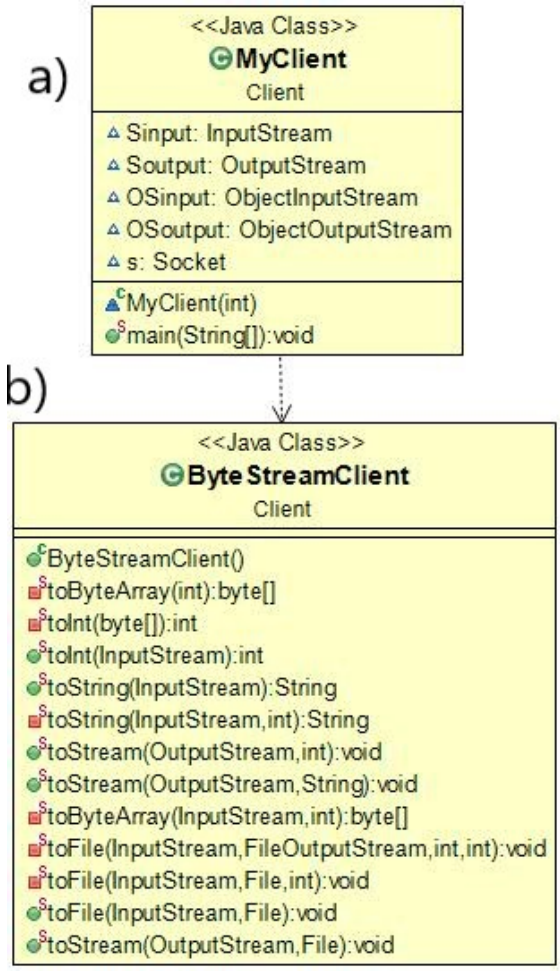

Fig. 4. UML class diagram for Client package. a) Image of MyClient class diagram with two main functions: Main, and MyClient. The latter class is used to specify the port number of connection. **b)** Byte Stream class used to output and get output streams to and from the client

where the car is to be registered. In order to complete the authorization step successfully, it is necessary to enter the same details of the vehicle which will charge from the station.

The sequence diagram begins after the EAV connects to the same WIFI network as the charging station. It has the same steps as refered in section 2, but the additional step happens when the car details are invalid/ car is not registered in the network. In that case, the car is not allowed to charge from this station and the session is closed as shown in Fig 5.

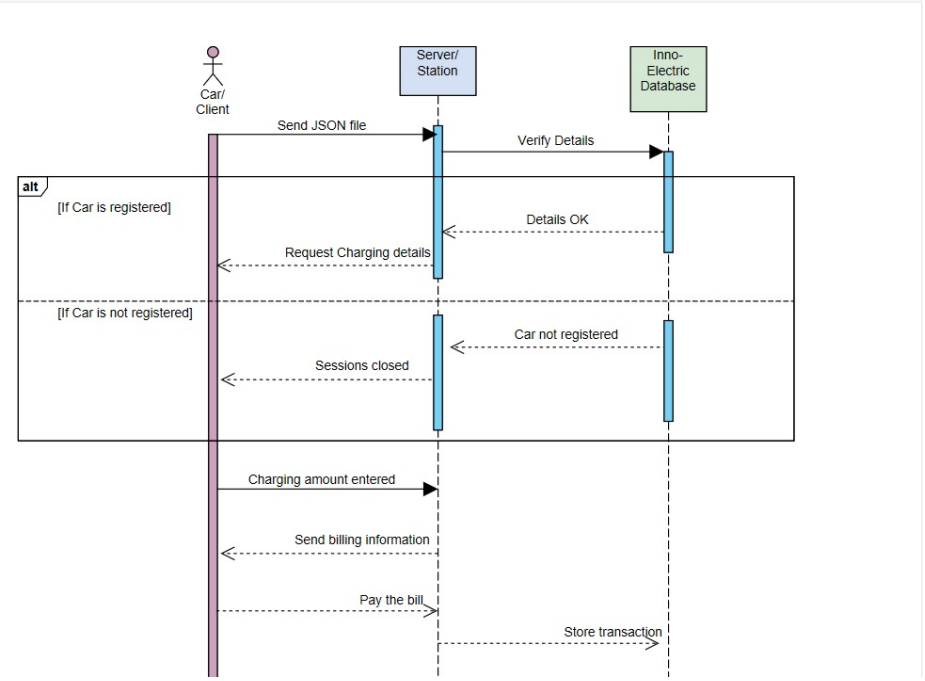

Fig. 5. Inno-EAV sequence diagram explanation. Allows any electric car owner to charge their electric car automatically. This sequence begins when an EAV owner connects to the same network as the charging station which it is registered to, it sends the JSON file to the charging station. The system sends the JSON file to the database system for verification. If the database system approves the car details, the server asks for the amount of electric charge and issues a receipt, afterwards, the user pays the bill, and the transaction is stored in the database. Otherwise, the session is closed, and the server waits for another client on the same port.

5 Database System Design

Inno-EAV-Core performs the charging station registration by creating a new tuple representing each relation entry, where the attributes values for each relation entry corresponds to the value from which the user entered at the registration form. For cases in which these entries violate the key constraints, an error message is used to warn the user, and the registration is not completed successfully. Because the violation of key constraints can happen with different keys in the same entry, the query execution is done after obtaining all the attribute values from the user for each of the car, the owner and the charging station relations, so the violation can also quickly checked and the user asked to correct his entry and enter a valid/unique key.

To generate these key constraints, Inno-EAV-Core first executes 'Owner' relation query, (Fig. 6). This is achieved by storing the owner's information as

a new tuple in the 'Owner' relation, the ID attribute is considered a primary key for the 'Owner' table, as shown in Fig. 6. Then the next step is to execute the 'Car' relation query, which also has a primary key 'ID' representing the car's unique ID (Fig. 6b). For the given schema, each car entry has a foreign key representing the car's owner. Finally, Inno-EAV-Core can then execute the 'Charging station' query to enter the tuple representing the charging station information that the user can charge from in the future (See Fig. 6).

Furthermore, the entries should contain non-repeated key entries uniquely representing each owner, car or charging station. A typical entry for this framework is three tuples with valid key constraints. Following the tuples creation of this entry, Inno-EAV-Core will create another tuple for 'charging station-has-car' relation which represents a many-to-many relationship as described in the following section, This can be used to assign each car to a charging station depending on its unique (Owner-ID, Car-ID) combination. Given that all the entries are valid, therefore, four new tuples will be added to each relation. The relations are: Owner, Car, Charging station, and Charging station-has-car, Inno-EAV-Core can then search the database for the car asking to charge from the station, which is known as authorization step, the car will be accepted or refused depending on whether its registered in the charging station or not (Fig. 6).

5.1 Entity Relationship Diagram

As part of the Inno-EAV framework, we include our database entity-relationship diagram (ERD) designed to match our system's needs, which is required to explain the relationships between different relations in the schema. As shown in Figure 6, the database has been constructed to keep track of the car owners, cars and charging station of a city. A charging station has several cars. It is desired to keep track of the owners owning each vehicle for each charging station, their name, ID, email, and phone the game. Each owner can have more than one car, so the relationship between car and owner relation is one-to-many as can be seen in the diagram. A charging station has more than one car, and a car can be registered in more than one charging station, which yields a many-to-many relationship.

6 Conclusion and Future Work

In this paper, we proposed a high performance and open-source framework for electrically charging online electric vehicles. The communication is done without the interference of human, wirelessly through WIFI network. In our proposed scheme, the cars are considered clients and the servers are charging stations. Clients are authenticated with the charging station after they send the file containing the car information to the charging station. Car registration is done using our web application, at which the owners fill in their information and then be assigned to a specific charging station meanwhile, the electric power service provider bills the vehicles on per charging palate basis. Our proposed scheme

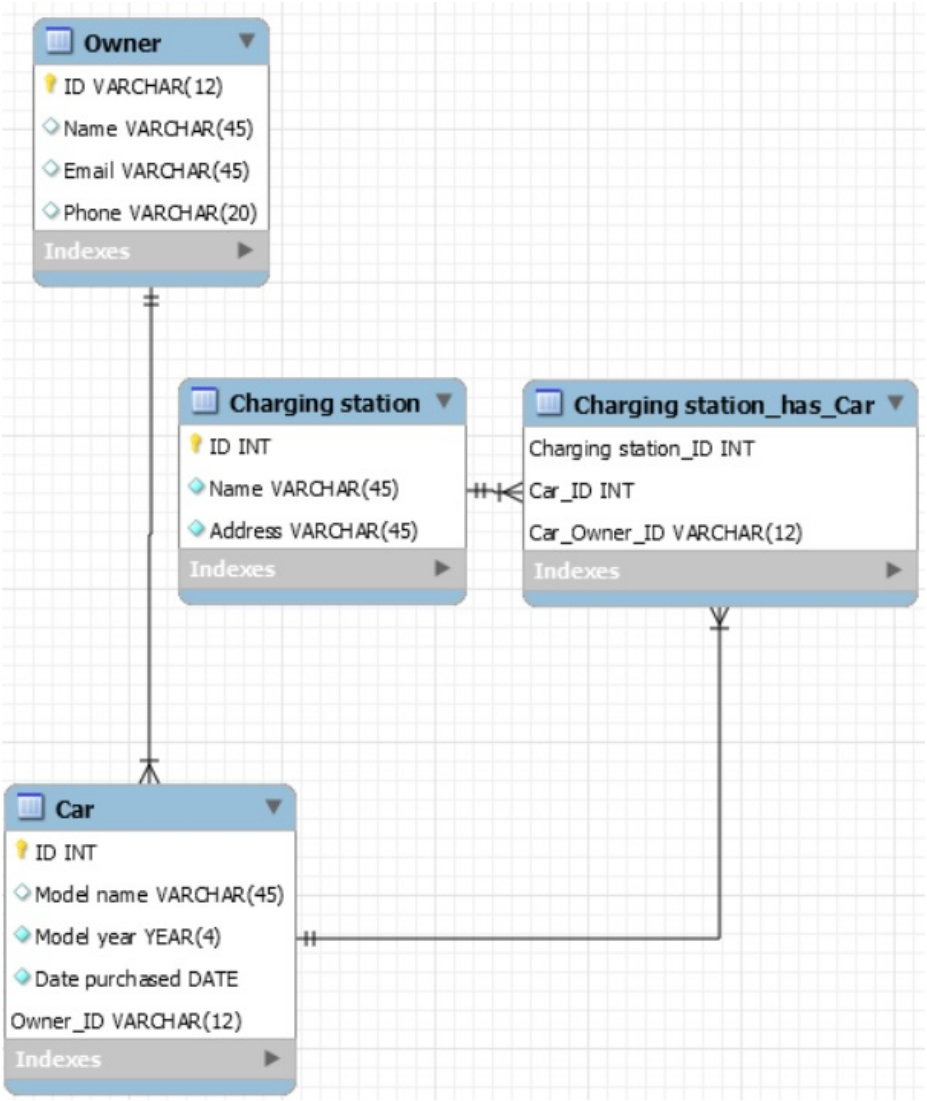

Fig. 6. Registration Database System Entity Relationship Diagram with Inno-EAV-Core. a) Owner of Electric-car table representation consisting of four main attributes which are: ID, name, Email and Phone **b)** Car table representation with four main attributes: ID, Model name, Model year, and Date-purchased and one inherited attribute: Owner-ID **c)** Charging station table representation with three main attributes: ID, Name, and Address **d)** Charging station-has-car table representation with three attributes: Charging-station-id, Car-id, Car-owner-ID

provides a secure and privacy-aware bidirectional audit, where the billing pro-

cess is valid by both parties [9]. Moreover, we also present the game-theoretic approach to validate the bidirectional audit.

In the future, we aim to implement the system to have a more in-depth insight into the performance issues. Moreover, we also aim to develop the Inno-EAV system to work on a larger scale. We will also focus on a more robust mechanism where the vehicles will have a choice to buy the charge according to their convenience. We will address the complexity involved in such robust charging and its billing mechanism and can thus be used to optimise acquisition workflows [9]. Furthermore, the current framework is fundamentally monolithic-based. To cope with scalability issues, we may in future refine it into a microservice-based one [6].

References

1. Electric autonomous vehicles. https://eav.innopolis.university/, accessed: 2019-23-8
2. Sit back, relax, and enjoy a ride through the history of self-driving cars. https://www.digitaltrends.com/cars/history-of-self-driving-cars-milestones/, accessed: 2019-23-8
3. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Tech. rep., U.S. Department of Transportation, 1200 New Jersey Ave, SE, Washington, DC 20590, USA (February 2015), https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115
4. Benčić, F.M., Žarko, I.P.: Distributed ledger technology: Blockchain compared to directed acyclic graph. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). pp. 1569–1570. IEEE (2018)
5. Burkhardt, D., Werling, M., Lasi, H.: Distributed ledger. In: 2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC). pp. 1–9 (June 2018)
6. Dragoni, N., Lanese, I., Larsen, S.T., Mazzara, M., Mustafin, R., Safina, L.: Microservices: How to make your application scale. In: Perspectives of System Informatics - 11th International Andrei P. Ershov Informatics Conference, PSI 2017, Moscow, Russia, June 27-29, 2017, Revised Selected Papers. pp. 95–104 (2017)
7. Hussain, R., Lee, J., Zeadally, S.: Autonomous cars: Social and economic implications. IT Professional **20**(6), 70–77 (Nov 2018)
8. Hussain, R., Zeadally, S.: Autonomous cars: Research results, issues, and future challenges. IEEE Communications Surveys Tutorials **21**(2), 1275–1313 (Secondquarter 2019)
9. Hussain, R., Son, J., Kim, D., Nogueira, M., Oh, H., Tokuta, A.O., Seo6, J.: Pbf: A new privacy-aware billing framework for online electric vehicles with bidirectional auditability (2017)
10. Kyriakidis, M., Happee, R., de Winter, J.: Public opinion on automated driving: Results of an international questionnaire among 5000 respondents. Transportation Research Part F: Traffic Psychology and Behaviour **32**, 127 – 140 (2015)
11. Rouse, M.: What is machine to machine. https://internetofthingsagenda.techtarget.com (2018)
12. Strugar, D., Hussain, R., Mazzara, M., Rivera, V., Afanasyev, I., Lee, J.: An architecture for distributed ledger-based M2M auditing for electric autonomous vehicles. In: Web, Artificial Intelligence and Network Applications - Proceedings of the

Workshops of the 33rd International Conference on Advanced Information Networking and Applications, AINA Workshops 2019, Matsue, Japan, March 27-29, 2019. pp. 116–128 (2019)
13. Strugar, D., Hussain, R., Mazzara, M., Rivera, V., Lee, J.Y., Mustafin, R.: On m2m micropayments: a case study of electric autonomous vehicles. In: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). pp. 1697–1700. IEEE (2018)
14. Tesla, N.: Experiments with alternate currents of very high frequency and their application to methods of artificial illumination. ELECTRICAL REVIEW (1891)
15. Williams, J.: Transport innovation of the week: electric charging lanes. https://makewealthhistory.org/2017/01/23/transport-innovation-of-the-week-electric-charging-lanes/ (2017)