International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-12 Issue-2, December 2022

Prototype Implementation of a Smart Locker

Alícia F. S. Luís, Gonçalo M. C. Martins, João M. L. P. Caldeira, Vasco N. G. J. Soares

Abstract: ***Smart lockers are intelligent storage units that are increasingly being adopted to help solving the last mile problem. This paper focuses on the concept of an individual smart locker that can be installed at the entrance of a residential house. First, the operational principle and advantages of this concept are discussed. Then, the design, development and implementation of a functional prototype is demonstrated. The prototype was submitted to tests and can be used as a proof of concept.***

Keywords: Smart Lockers, Parcel Delivery, Prototype, Implementation, IoT.

I. INTRODUCTION

In 2022, e-commerce is being estimated to represent 22% of sales globally, according to research by Morgan Stanley [1]. This number represents a 7% growth in just three years. As it is known, this growth is due to the COVID-19 pandemic, but now that the world is starting to recover some questions are being raised. One of the biggest questions is: will e-commerce continue to grow? According to [1], the answer is yes. Moreover, it predicts a 5% increase in retail sales by 2026. However, it is almost impossible to talk about the growth of e-commerce without mentioning the last mile problem [2]. This is a long-standing problem in the delivery industry and consists of the last part of the process of delivering a product. There are several issues that make delivering the order to the end customer complicated and expensive. These reasons include difficulties in responding to sales peaks, uncontrollable problems such as traffic, and unforeseeable issues on the customer's side (e.g., absence, wrong address, among others). One of the solutions to this problem are smart lockers [3]. These are storage units that notify the customer when a parcel is delivered and keep it safe until pickup. Besides that, smart lockers are particularly useful for deliveries that do not require the customer to be present. As discussed in a previous work [4], smart lockers have several application areas. These areas include health, banking, condominiums, offices, education, and commerce. These solutions, while similar, have subtle specifications and uses that distinguish them from each other. These differences include for example the authentication and opening methods that range from numerical codes [5], tracking numbers [6], customer credentials [7], barcodes [8], QR (Quick Response) codes [9], NFC (Near-Field Communication) and RFID (Radio-Frequency Identification) [10], biometric data [11], and even mobile applications [7]. Although smart lockers are a great solution, the customer still must travel to the smart locker bank to pick up their parcel. While this ensures that the parcel is delivered on time, it can be uncomfortable for the customer. Especially if the solution requires the customer to pay for the time of use. If we have mailboxes for letters, why not have that concept for packages? To solve this problem an individual smart locker placed at the customer's door is proposed in this paper. One of the many questions this idea may raise is: what do the customers think? Is it useful for them? To understand this and other issues, a questionnaire was developed, not shown here due to page limitation. A total of 58 responses were obtained, with most respondents being between the ages of 18-25. As expected, 87.9% have shopped online, buying more frequently between 2 and 6 times a year. Another not surprising fact is the preference for home delivery: 96.1% choose this delivery method. With these shopping patterns analysed, it is important to understand if customers are satisfied with the delivery of their orders. The collected data shows that more than half of the respondents have experienced problems with delivery. One of the most frequently mentioned problems is delivery times and dates. Either because the companies are late in delivering, or because they were not at home at the time of delivery. The questionnaire then moves on to present and ascertain the interest on the concept of an individual smart locker. 86.2% of those questioned say they would like to have this technology, and 69% think it would be useful to them. Furthermore, 79.4% find this method more efficient than existing delivery methods. The results of this questionnaire show the importance and the good acceptance of the concept of a smart locker installed on a residential house. Furthermore, they support the development of an innovative customer-focused solution. Nevertheless, this concept poses questions such as: How can it be assured that only a courier can open and place the package in the smart locker? How can it be guaranteed that the delivery has been made and to the correct smart locker? What is the most secure authentication method? Points arising from these questions will be discussed later in the paper. The remainder of this paper is organized as follows. Section 2 presents the prototype architecture of the smart locker. Its implementation and results from experiments are described in Section 3. Finally, Section 4 presents the conclusions and future work.

Manuscript received on 13 October 2022 | Revised Manuscript received on 28 October 2022 | Manuscript Accepted on 15 December 2022 | Manuscript published on 30 December 2022.
***** Correspondence Author (s)
Alícia F. S. Luís, Instituto Politécnico de Castelo Branco, Portugal. Email: alicia.luis@ipcbcampus.pt
Gonçalo M. C. Martins, Instituto Politécnico de Castelo Branco, Portugal. Email: goncalo.martins1@ipcbcampus.pt
João M. L. P. Caldeira, Instituto Politécnico de Castelo Branco, Instituto de Telecomunicações, Portugal. Email: jcaldeira@ipcb.pt
Vasco N. G. J. Soares *, Instituto Politécnico de Castelo Branco, Instituto de Telecomunicações, Portugal. Email: vasco.g.soares@ipcb.pt

II. PROTOTYPE ARCHITECTURE

This section presents the proposed prototype architecture. It describes how the different components of the system communicate, the technologies used and the hardware and software components. To successfully implement a prototype that represents an individual smart locker installed at the entrance of a residential house, it is necessary to consider concepts of databases, Internet of Things (IoT), web and mobile app development, and security. Therefore, the proposed solution is based on an architecture composed by several elements. Fig. 1 represents a scheme of this system and the technologies used. The prototype will use Wi-Fi technology for Internet connection. To store all the system data the MariaDB [12] database system will be used. For viewing and changing data in the database a website will be developed with a login system and encryption of passwords based on SHA256 (Secure Hash Algorithm) [13] with the use of *salt*. For the issues related with the delivery of orders, the couriers will have a mobile application that will be connected to the database and the microcontroller through an API (Application Programming Interface) developed through PHP scripts.

A. Hardware Component

The Wemos D1 Mini Pro (v3.1.0) [14] is the microcontroller used for the smart locker prototype. It is a microcontroller from the ESP8266 family [15], which allows establishing an Internet connection via Wi-Fi. It is equipped with 11 digital input/output pins. The RFID Reader [16] is one of the ways that can be used to open the smart locker using a small keyring (i.e, a tag) with a Unique Identification (UID). The RFID readers are a two-way radio transmitter-receiver that are responsible to send a signal to the tag and to read its response. A keyring was chosen for the convenience of the customer. A 16x2 display [17] is used in the prototype to show the messages to users to facilitate the interaction with them. The display will mainly be used to inform the user when the door is closed or opened. The implementation of the hardware prototype results in the system shown in Fig. 2. It contains the microcontroller, the 16X2 display, the RFID Reader, and an RFID tag associated to the reader. In terms of physical appearance, the prototype is based on the previously studied smart locker compartments. Fig. 3 shows a three-dimensional (3D) model of what this individual smart locker would look like and where the sensors and electronic equipment would be located. Fig. 3 (A) of the figure shows a view of the outside of the locker and Fig. 3 (B) a view of the inside. The construction of this 3D model is outside the scope of this project.

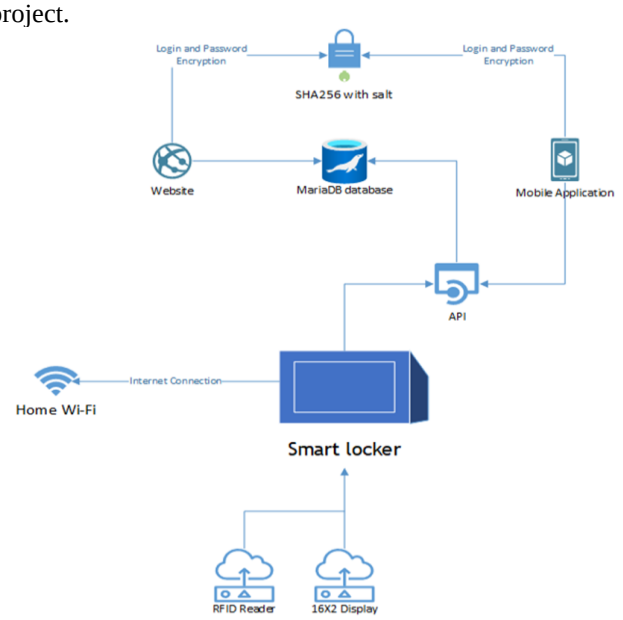

Fig. 1. Prototype architecture.

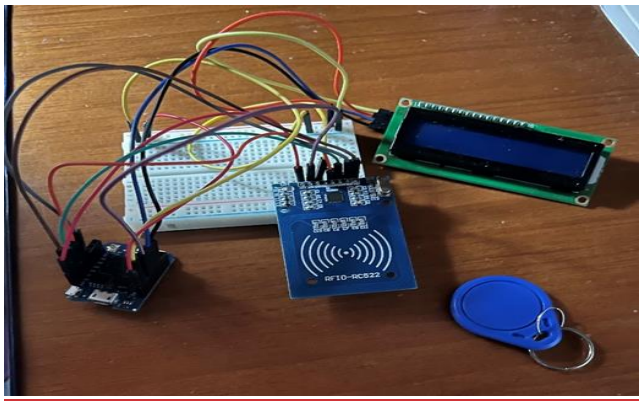

Fig. 2. Hardware used on the prototype.

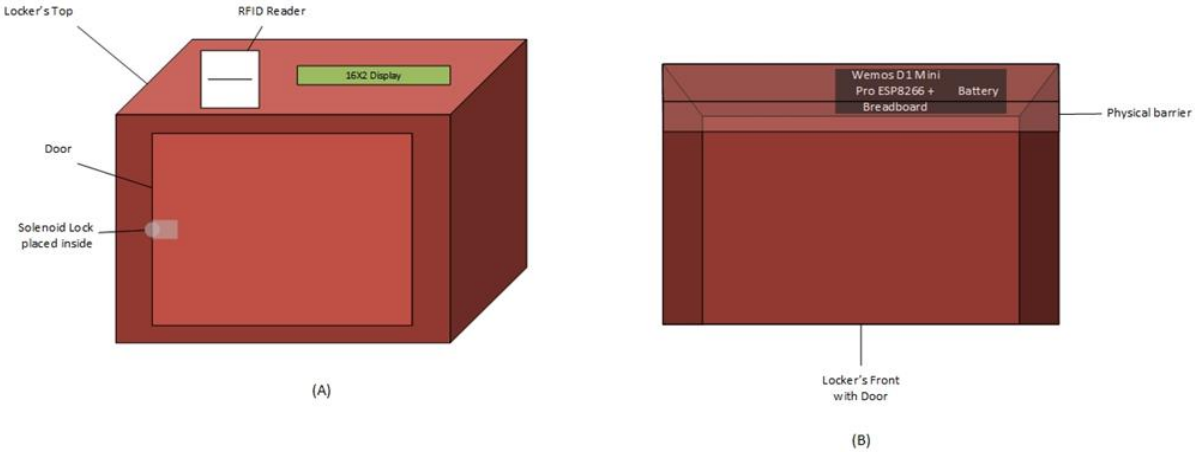

Fig. 3.Prototype 3D Model.

Retrieval Number: 100.1/ijeat.B38771212222
DOI: 10.35940/ijeat.B3877.1212222
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

B. Software Component

Specific software needs to be developed for the prototype. A website is required so that customers and couriers can view and manage their orders. For this purpose, XAMPP [18] will be used as a web server. This solution is cross-platform, allowing the website to be developed using the PHP programming language. Furthermore, it includes the MariaDB [19] database system, where all the information will be stored. At the moment when the courier prepares to deliver the parcel to the smart locker, it is necessary to have a mobile application to check which parcels are ready to be delivered and the addresses of the smart lockers to which they should be delivered. For the development of this application the Android Studio Integrated Development Environment (IDE) [20] was used. This IDE allows to develop programs in different languages such as Java [21] or Kotlin [22] and also allows to simulate the code execution in a simulator within the IDE. APIs are also required. They are composed by PHP scripts that receive and send data. Thus, POST variables [23] and the HTTP POST method [24] are used. Similar to what is implemented in the microcontroller, this method makes POST requests, according to HTTP Content-Type of application/x-www-form-urlencoded [25]. The API is used for the connection establishment between the microcontroller and the Android application with the developed database.

The microcontroller was programmed using the Arduino IDE [26]. This IDE uses C and C++ functions, and allows to easily compile and upload the program to the microcontroller.

III. PROTOTYPE IMPLEMENTATION AND RESULTS

In this section the implementation and testing of the functional prototype are discussed. Two types of users are considered for the prototype: customers (i.e., clients) and couriers. The diagram in Fig. 4 shows the flow of actions for the smart locker on the client side. The normal state of the locker is 'Door Closed!' on the display. If a correct RFID keyring is presented, the locker state changes to 'Door Open!'. In this state, if the client closes the door, the locker returns to its normal state. If an incorrect RFID keyring is presented, the locker goes to the status 'Access Denied!' and after a while it returns to its normal status. If no RFID keyring is shown, the smart locker never exits the 'Door Closed!' status.

The diagram in Fig. 5 shows the flow of actions for the smart locker on the courier side. Here, the normal state of the smart locker is also 'Door Closed!'. If the courier presses the 'Connect to Locker' button in his application, the code is shown on the smart locker display. In this state there are two possible actions: the courier enters a correct code, and the door opens, returning to the normal state of the locker after it is closed; or the courier enters a wrong code, showing the wrong number of times on the display. In the latter state, a new code is generated and shown again on the smart locker. If the courier misses the code three times, the smart locker returns to the 'Door Closed!' state.

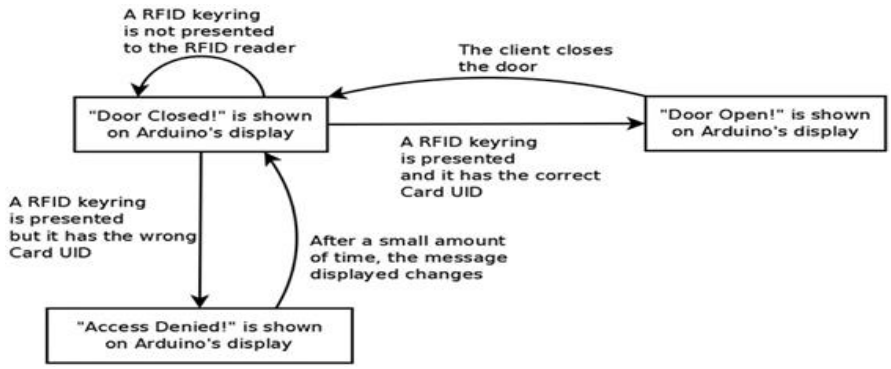

Fig. 4. Client-side flowchart.

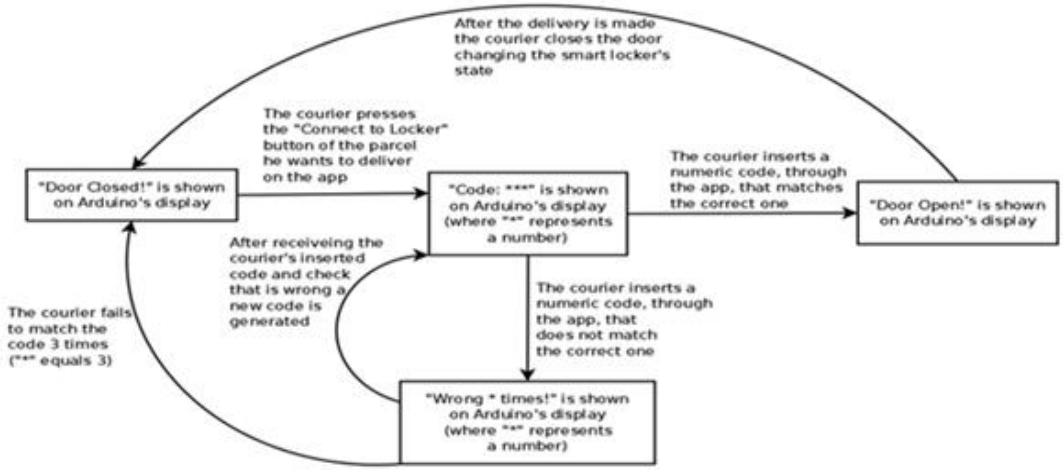

Fig. 5. Courier-side flowchart.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

A.Smart Locker

The implementation of the hardware component of the prototype began by establishing a successful connection to Wi-Fi. This requires the ESP8266WiFi [27] library and the definition of two variables with the Wi-Fi credentials: one for the network SSID (Service Set Identifier) and one for the password. Another required library is the ESP8266HTTPClient [28], which allows the microcontroller to do HTTP requests.

Next, all the hardware components were added to the system, namely the display and the RFID Reader. For the display to work properly the Wire [29] and LiquidCrystal_I2C [30] libraries were used. For the RFID, the SPI [31] and MFRC522 [32] libraries were added too. In physical terms, this implementation used eight pins of the microcontroller. Table 1 shows the function of each of the pins.

Table 1: Pin connection.

Pin	Connection
D1	LCD – SCL
D2	LCD – SDA
D3	RFID – RST
D5	RFID – SCK
D6	RFID – MISO
D7	RFID – MOSI
D8	RFID - SDA

There are two possible ways to open the smart locker. One is using an RFID registered keyring from a client. The other is performed by the courier, when a parcel is to be delivered, and requires inserting a correct code through an application.

The following figures demonstrate how the microcontroller reacts to the different actions that can be performed on it. In its normal state, presented in Fig. 6, it has the door closed with the message "Door Closed!" being shown on the display. It periodically checks if there is any order ready to be delivered by a courier. For this, a POST request is sent to the API, containing the smart locker's id value to check if there is any order in the Code table directed to that smart locker.

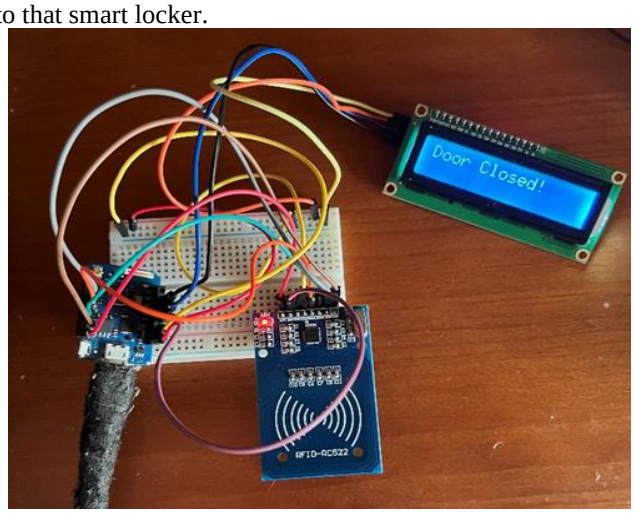

Fig. 6. Microcontroller: Smart Locker's initial state.

If there is not an order ready to be delivered at the time, the microcontroller receives an information from the API containing as payload "Nothing" as shown in the Fig. 7 and the door remains closed.

```
[HTTP] begin... CheckOrder 1
[HTTP] POST...
[HTTP] POST... code: 200
received payload:
<<
Nothing
>>
```

Fig. 7. Microcontroller: POST request, HTTP Code and Received Payload when there is not an order.

If there is an order ready to be delivered, it means that a courier wants to deliver a parcel. Then, the microcontroller generates a random code that is shown on the display so that the courier can see it on the spot and insert that code in the application. These can be seen in Figs. 8 and 9.

```
[HTTP] begin... CheckOrder 1
[HTTP] POST...
[HTTP] POST... code: 200
received payload:
<<
ID:63946
Code:000
>>
```

Fig. 8. Microcontroller: POST request, HTTP Code and Received Payload when there is an order to be delivered.

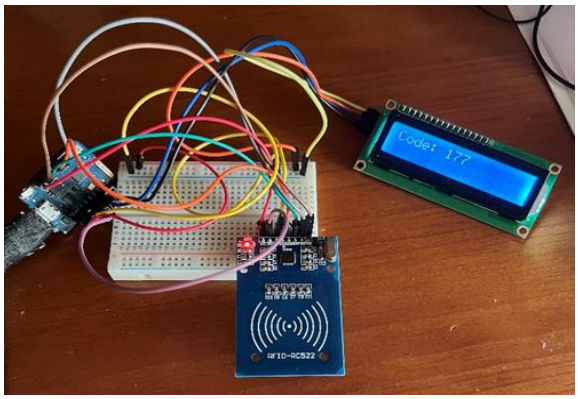

Fig. 9. Microcontroller: Code generation for parcel delivery.

If this code is a match, the door is opened and the message "Door Open!" is shown on the display and the tracking number of the order is sent through a POST request to the API. Then, the API changes the order status to 'Delivered' and removes the order from the Code table, returning to the microcontroller the payload of "Delivered!", as shown in Fig. 10.

```
[HTTP] begin... Delivery
[HTTP] POST...
[HTTP] POST... code: 200
received payload:
<<
Delivered!!
>>
```

Fig. 10. Microcontroller: POST request, HTTP code and received payload, when the order is delivered.

If the code entered is incorrect, the message "Wrong 1 time!" is displayed showing this information to the courier. If the code is entered wrong again, the message is updated for the number of times the code was entered incorrectly. This can be seen in Fig. 11. If the code is entered wrong 3 times or no code is entered at all within a certain amount of time, the delivery of the parcel is cancelled, and it is removed from the Code table. For this, the microcontroller sends a POST request to the API with the order's id and a variable with the value "three" indicating that the courier did not enter any code in time, or he missed the code 3 times. As a response, the API sends the value "Cancelled!", as it can be seen in Fig. 12.

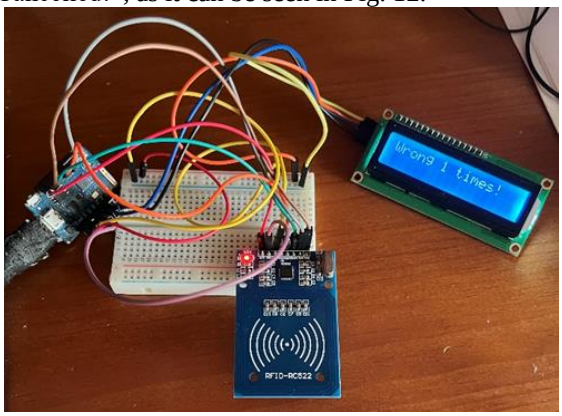

Fig. 11. Microcontroller: The wrong code was inserted on courier's app

```
[HTTP] begin... Cancel Delivery
[HTTP] POST...
[HTTP] POST... code: 200
received payload:
<<
Cancelled!
>>
```

Fig. 12. Microcontroller: POST request, HTTP code and received payload, when the order is cancelled.

In addition, the client can open his smart locker by means of an RFID tag, in the form of a keyring. If the tag is associated with the smart locker, it opens it, as shown in Fig. 13. Otherwise, an 'Access Denied!' message is displayed and the door remains closed, as shown in Fig. 14.

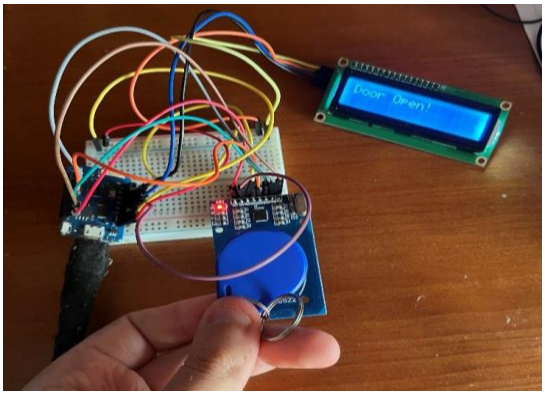

Fig. 13. Microcontroller: Using client's RFID keyring to open the door.

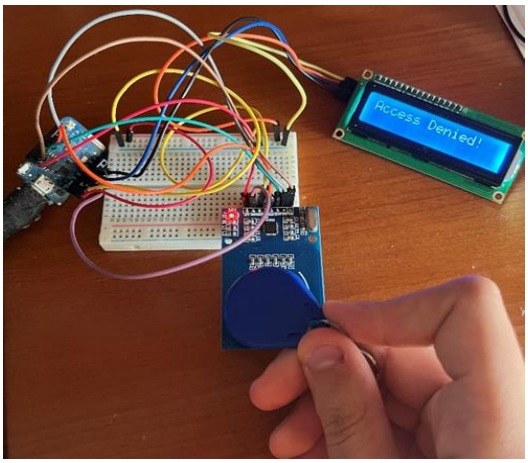

Fig. 14. Microcontroller: Using a non-client's RFID keyring to open the door.

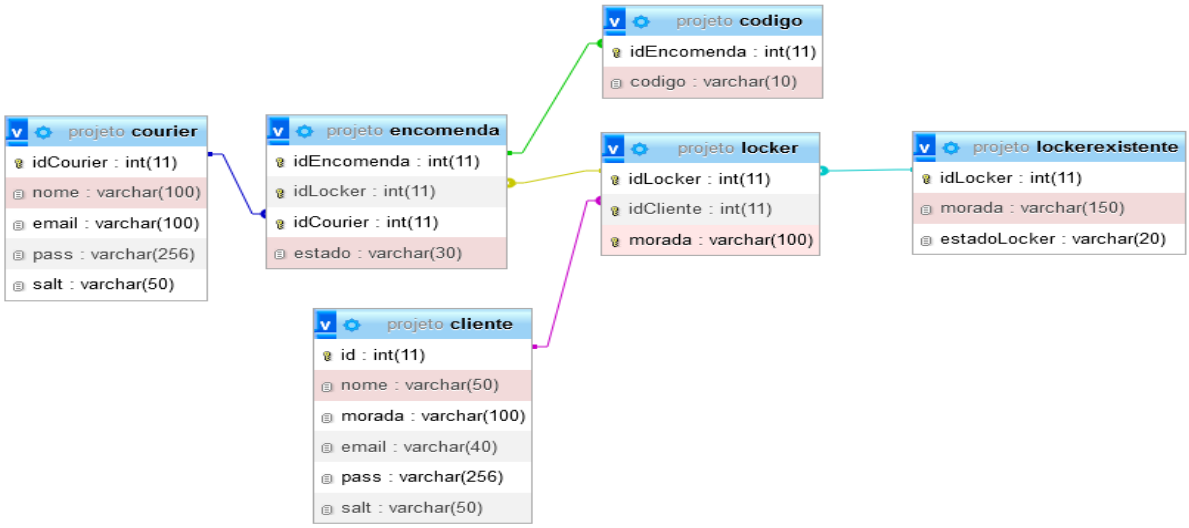

Fig. 15. Relational model of the database.

B. Database

The relational model of the database is illustrated in Fig. 15. The Client table (‘cliente’) stores the client information, such as id, name (‘nome’), address (‘morada’), email, and the salt generated for the encryption of the password.

The Courier table stores information similar to the Client’s table, apart from the address.

The Existing Lockers table ('lockerexistente') has all the information about the existing smart lockers, such as their id (‘idLocker’), address (‘morada’) and also their status (‘estadoLocker’): if they are ‘Registered’ or ‘Available’.

The Locker table stores all the information related to the lockers registered by clients: their id (‘idLocker’) (through a Foreign Key for the table ExistingLockers), the id of the client (‘idCliente’) to which they are associated (through a Foreign Key for the table Client) and their address (‘morada’). The Order table (‘encomenda’) stores the information related to each order: its id ('idEncomenda') that represents its tracking number, the id of the locker where it will be delivered ('idLocker') (through a Foreign Key to the Locker table), the id of the courier responsible for the delivery ('idCourier') (through a Foreign Key to the Courier table) and, the order status (‘estado’) that can be set as ‘To be delivered’, ‘Ready for Delivery’ and ‘Delivered’. The Code table (‘codigo’) stores the information when a parcel is ready to be delivered by a courier to a smart locker and codes are generated for the opening of that smart locker. To do so, the id of the order to be delivered ('idEncomenda') is stored (through a Foreign Key to the Order table) and the code inserted by the courier ('codigo').

C. Website

It is necessary to have a system that allows for example an existing smart locker to be associated with a particular client, so that he can place orders and pick them up there. Therefore, it was necessary to develop a website with a login system using the SHA256 password encryption method with the use of *salt* [33]. SHA256 [13] is a cryptographic hashing function that generates a 256-bit hash. It is part of the group of algorithms belonging to SHA 2 [34], a family consisting of six algorithms such as SHA256. When a new client or courier is registered, a new *salt* is generated. For that, a function was developed, so that, given a certain length through a parameter, a random string is obtained with that same length. Then, the actual password of the client/courier is concatenated with the salt and that output is hashed according to the SHA256 algorithm, with the use of the function *hash()* [35]. The website layout, contents and navigation are shown in Figs. 16 to 24. The website has a homepage that allows the user to choose which role to assume client or courier. After choosing his role, the user is sent to a login page where he must insert his user e-mail and password or, if the user does not have an account yet, he can create one using the registration page (Figs. 18 and 20).

Client Role

Once authenticated (Fig. 17), clients are redirected to a welcome page shown in Fig. 21. The page allows seeing the smart lockers associated with the account, and which orders were already received or placed. Additionally, if an order has the status 'To be delivered', it can be cancelled by the client. The options button allows to: 1) log out and return to the home page; 2) place an order as shown in Fig. 22 (i.e., the purchase of an item is simulated and added to the client’s account by associating the smart locker’s address as the delivery address; although the client can place multiple orders, a smart locker can only have one order associated with it); 3) add a smart locker from the list of available lockers to the client's account as shown in Fig. 23.

D. Courier Role

Once authenticated (Fig. 19), couriers are redirected to a welcome page shown in Fig. 24. The page allows seeing which orders are associated with the account and whose status is 'To be delivered' and ‘Delivered’. In addition, the courier may also give an order to prepare the delivery if that order is already in his possession. If this option is selected, the order status changes from 'To be delivered' to 'Ready for Delivery'. Delivered orders only appear in the table for information purposes.

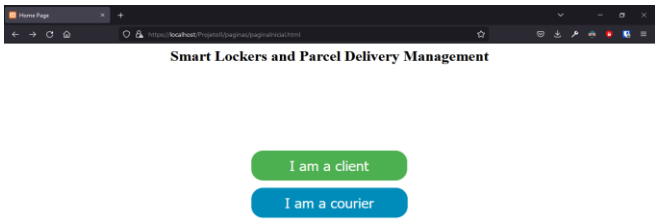

Fig. 16. Website: Home page.

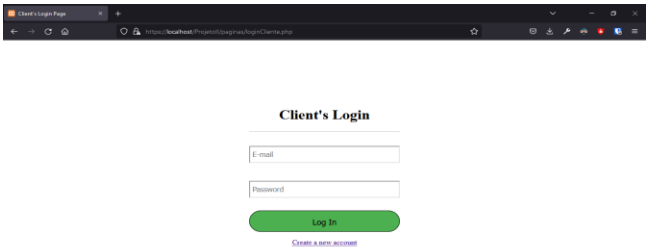

Fig. 17. Website: Client’s login page.

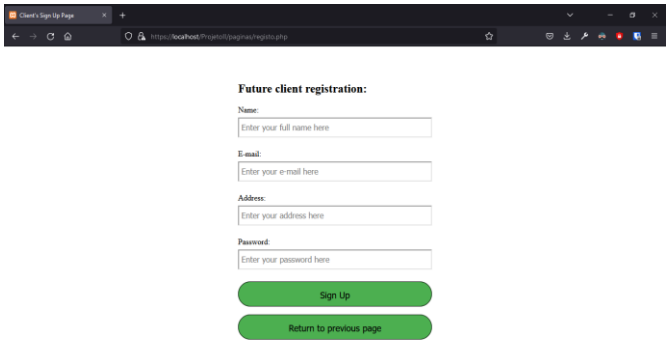

Fig. 18. Website: Client’s registration page.

Retrieval Number: 100.1/ijeat.B38771212222
DOI: 10.35940/ijeat.B3877.1212222
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

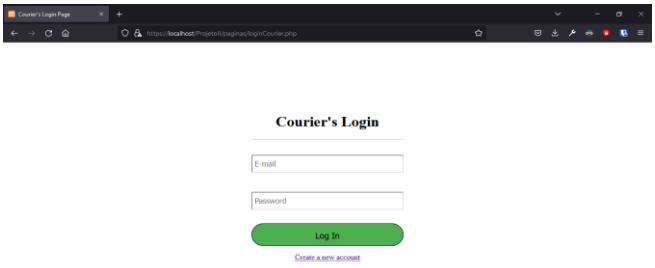

Fig. 19. Website: Courier's login page.

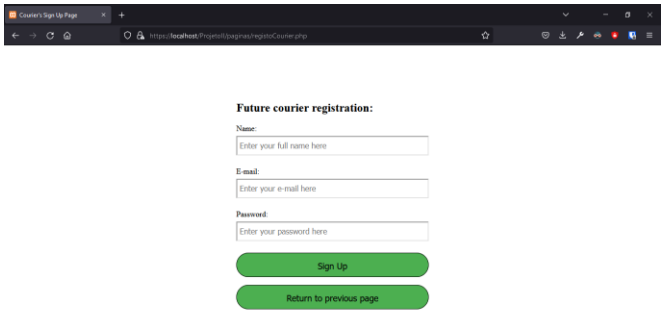

Fig. 20. Website: Courier's registration page.

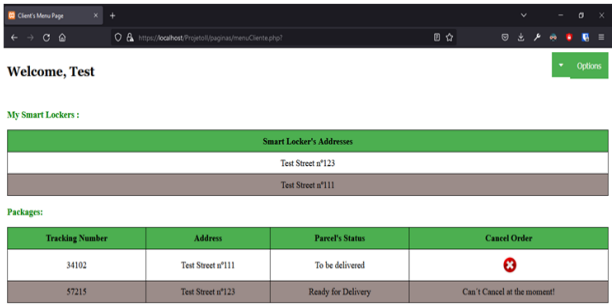

Fig. 21. Website: Client's menu.

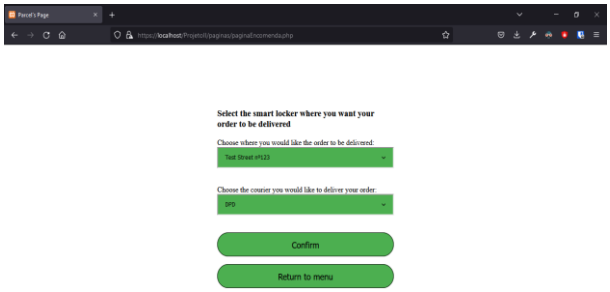

Fig. 22. Website: Make a new order.

Fig. 23. Website: Associate a new smart locker to the account.

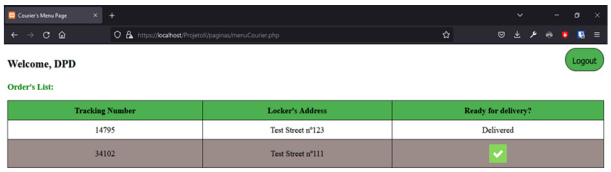

Fig. 24. Website: Courier's Menu.

E. Android Application

A mobile application is required to allow the courier to authenticate and open the smart locker to deliver a package. Details about this app are discussed in this subsection.

Fig. 25 shows the login screen where the courier authenticates to access its account. After successful login, a new screen will come where all orders associated with the courier's account are displayed in a table, as can be seen in Fig. 26. On the last cell of each table line there is a button ('Connect to Locker') that allows the courier to send an information to a smart locker that is ready to deliver that order. To perform the login operation and improve efficiency, the library Advanced-Http URL Connection [36] was used. The Put Data class was used, passing as input parameters the URL, the method to use (GET or POST), and the data to send, in this case the courier's e-mail and password.

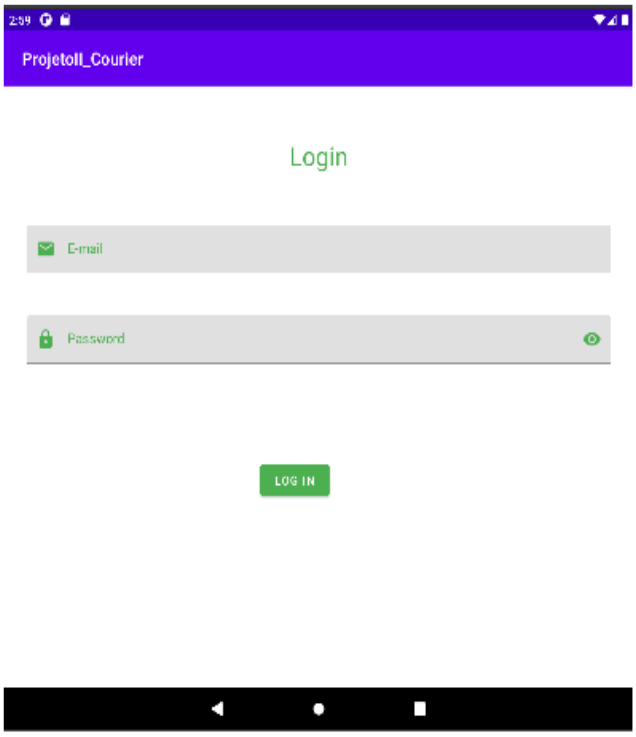

Fig. 25. Mobile app: Login screen.

The application communicates with the API for connecting to the database, to obtain a list of all the orders that are associated with the account of a courier. To do so the application sends a GET request containing the e-mail of the logged-on courier to the API that encodes that list in JSON (JavaScript Object Notation) [37]. The GET request and the response are presented in Fig. 27.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The application obtains that list by using the Volley library [39], an HTTP library that allows the use of a Json Object Request [40] class, which through a URL obtains an array encoded in JSON.

After receiving the response, the JSON Array [41] class allows using a loop to go through the results and get all the objects, through the JSON Object [42] class, to split the JSON attributes by the table shown in the Fig. 16.

After the table is formed with the data coming from the Json Object Request, the courier presses the "Connect to Locker" button and sends the tracking number of that order through the POST method to the API which, through a MySQL query, inserts that id in the Code table with the code '000'. Then, the courier is sent to the screen shown in Fig. 28 where he needs to insert the same code generated by the smart locker. If the courier misses the code three times or presses the "Cancel" button or the back button, the order is not delivered and the API receives, through the POST method, a variable indicating that the code was missed or the delivery was cancelled, thus performing a DELETE operation on the Code table for that parcel.

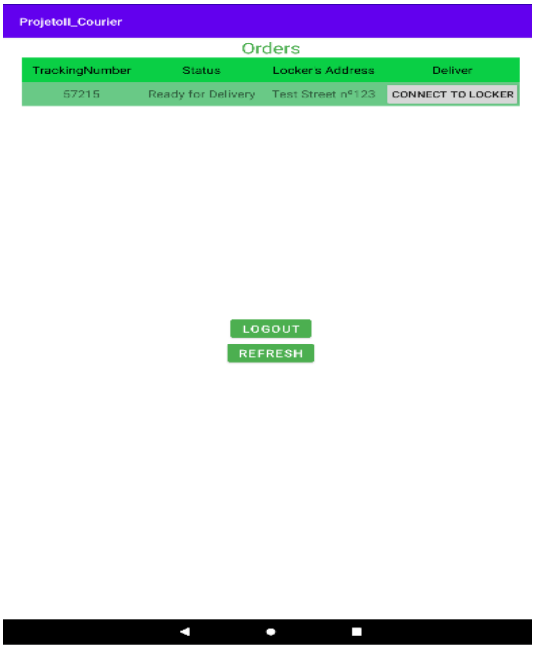

Fig. 26. Mobile app: Main screen.

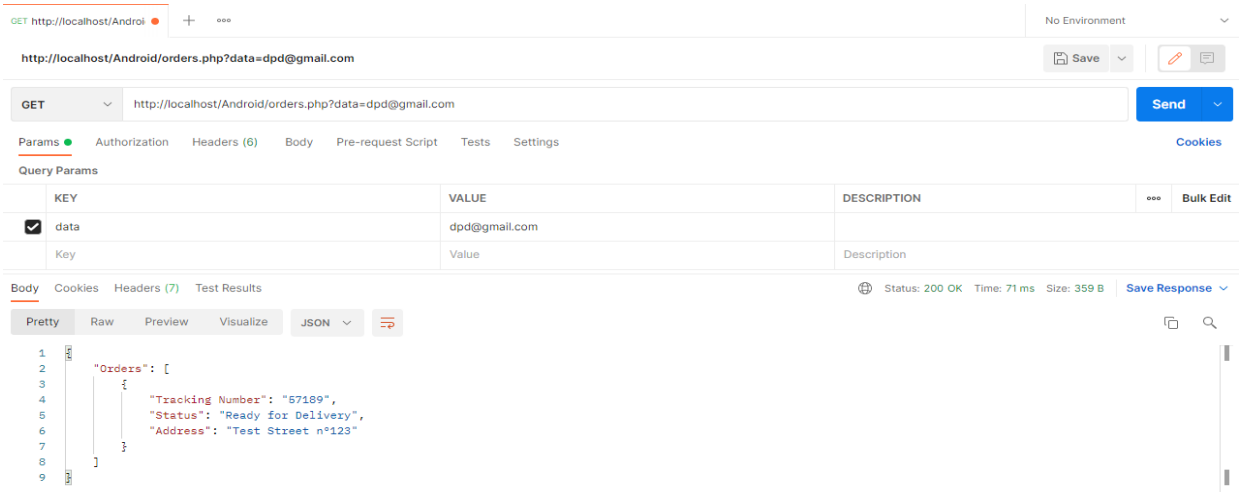

Fig. 27. API: GET request and JSON encoded response testing using Postman [38].

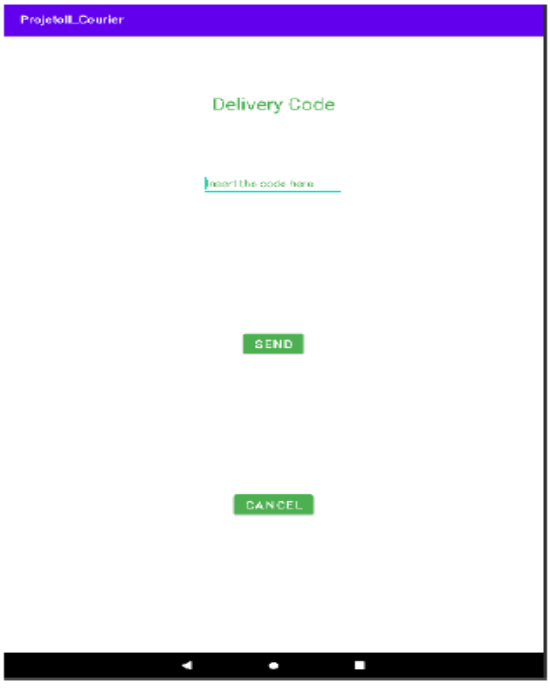

Fig. 28. Mobile app: Delivery screen.

IV. CONCLUSIONS

Although smart locker technology is not recent, their usage for customer and consumer pickup is attracting increasing interest, as they are a convenient way to receive orders without being present. This work focused on the concept of deploying smart lockers to facilitate deliveries to residential home addresses.

To demonstrate the functionality and benefits of this approach, this paper described the design, development, and implementation of a functional smart locker prototype integrating hardware and software components. It serves as proof-of-concept and as a testbed for future use case scenarios and for testing new services, technologies, and approaches. Plans for future work include using blockchain [43] for ensuring security and privacy. It would be interesting to have a system with different private blockchains - one for each client [44].

Retrieval Number: 100.1/ijeat.B38771212222
DOI: 10.35940/ijeat.B3877.1212222
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

This way it would be possible to protect the client's parcel information. Multi Chain Private Blockchain [45] may be used. This blockchain should only be accessible to the client and to the company for technical support issues. The blockchain could store information from one or more smart lockers, depending on the number of these devices owned by the client.

Another improvement could be to ensure that the courier does not steal any packages from the smart locker. In this prototype, the problem was solved by limiting to one the number of orders per locker. However, this would not be realistic or convenient, because a customer may receive more than one order. A weight sensor could be used to detect any change in the locker and alert the customer.

REFERENCES

1. "Global Ecommerce Growth Forecast 2022 | Morgan Stanley," Jun. 14, 2022. https://www.morganstanley.com/ideas/global-ecommerce-growth-forecast-2022 (accessed Sep. 26, 2022).
2. ARTI - Autonomous Robot Technology GmbH, "The famous 'last mile' problem explained – More Than Digital," Jan. 26, 2021. https://morethandigital.info/en/the-famous-last-mile-problem-explained/ (accessed Sep. 26, 2022).
3. M. Tarpey, "4 Ways Smart Lockers Improve Last Mile Delivery | Exela," Feb. 04, 2021. https://www.exelatech.com/blog/4-ways-smart-lockers-improve-last-mile-delivery?language_content_entity=en (accessed Sep. 27, 2022).
4. "A. F. S. Luís, G. M. C. Martins, J. M. L. P. Caldeira, and V. N. G. J. Soares, "Smart Lockers: Approaches, Challenges and Opportunities," Int J Eng Adv Technol, vol. 11, no. 3, pp. 141–149, Feb. 2022, doi: 10.35940/IJEAT.C3374.0211322. [CrossRef]
5. H. J. Tang, "How does a smart locker system work?," Dec. 07, 2021. https://yellowbox.app/articles/how-does-a-smart-locker-system-work (accessed Oct. 10, 2022).
6. "Universal Parcel Tracking - Global Package Tracking." https://parcelsapp.com/en/tracking (accessed Oct. 10, 2022).
7. "How Smart Locker Systems Work [+Why You Need One]." https://elocker.com/how-smart-locker-systems-work/ (accessed Oct. 10, 2022).
8. "Meridian Kiosks Automated Smart Locker System," 2019. https://www.meridiankiosks.com/wp-content/uploads/2019/05/2019-Meridian-Kiosks-Automated-Smart-Locker-System.pdf (accessed Oct. 10, 2022).
9. "How to Use Smart Lockers | QR Code - StorageX Philippines." https://storagex.com.ph/qr-code/ (accessed Oct. 10, 2022).
10. "RFID Makes Smart Lockers Even Smarter." https://www.elatec-rfid.com/en-us/case-study-detail/rfid-makes-smart-locker-systems-even-smarter (accessed Oct. 10, 2022).
11. "What is a Biometric smart lock? What are its advantages?" https://www.nordencommunication.com/en/blog/what-is-biometric-smart-lock (accessed Oct. 10, 2022).
12. "MariaDB Foundation - MariaDB.org." https://mariadb.org/ (accessed Oct. 06, 2022).
13. "What Is SHA-256 Algorithm: How it Works and Applications," Jun. 11, 2022. https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm (accessed Oct. 06, 2022).
14. "LOLIN D1 mini v3.1.0 — WEMOS documentation." https://www.wemos.cc/en/latest/d1/d1_mini_3.1.0.html (accessed Oct. 10, 2022).
15. J. Morais, "O que é o ESP8266? -A Família ESP e o NodeMCU | Portal Vida de Silício," Jun. 09, 2017. https://portal.vidadesilicio.com.br/o-que-esp8266-nodemcu/ (accessed Oct. 06, 2022).
16. "Security Access Using RFID Reader - Arduino Project Hub." https://create.arduino.cc/projecthub/Aritro/security-access-using-rfid-reader-f7c746 (accessed Apr. 14, 2022).
17. A. Thomsen, "Controlando um LCD 16x2 com Arduino – Filipe Flop," Sep. 11, 2011. https://www.filipeflop.com/blog/controlando-um-lcd-16x2-com-arduino/ (accessed Apr. 14, 2022).
18. P. Pinto, "XAMPP: Ambiente de desenvolvimento de sites em PHP na sua máquina," May 08, 2021. https://pplware.sapo.pt/tutoriais/xampp-desenvolva-sites-em-php-na-sua-maquina/ (accessed Oct. 04, 2022).
19. "New XAMPP with MariaDB." https://www.apachefriends.org/blog/new_xampp_20151019.html (accessed Oct. 06, 2022).
20. "Download Android Studio & App Tools - Android Developers." https://developer.android.com/studio (accessed Oct. 04, 2022).
21. "Introduction to Java." https://www.w3schools.com/java/java_intro.asp (accessed Oct. 06, 2022).
22. "Kotlin Introduction." https://www.w3schools.com/KOTLIN/kotlin_intro.php (accessed Oct. 06, 2022).
23. "PHP: $_POST - Manual." https://www.php.net/manual/en/reserved.variables.post.php (accessed Oct. 04, 2022).
24. "POST - HTTP | MDN," Sep. 09, 2022. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST (accessed Oct. 06, 2022).
25. "Difference Between form-data, x-www-form-urlencoded and raw in Postman | Baeldung," Jul. 12, 2022. https://www.baeldung.com/postman-form-data-raw-x-www-form-urlencoded (accessed Oct. 06, 2022).
26. "Software | Arduino." https://www.arduino.cc/en/software (accessed Jun. 27, 2022).
27. I. Grokhotkov, "ESP8266WiFi library — ESP8266 Arduino Core documentation," 2017. https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html (accessed Oct. 04, 2022).
28. M. Sattler, "ESP8266HTTPClient.h · GitHub," 2015. https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266HTTPClient/src/ESP8266HTTPClient.h (accessed Oct. 04, 2022).
29. "Wire - Arduino Reference." https://www.arduino.cc/reference/en/language/functions/communication/wire/ (accessed Oct. 04, 2022).
30. "Liquid Crystal I2C - Arduino Reference." https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/ (accessed Oct. 04, 2022).
31. "SPI - Arduino Reference." https://www.arduino.cc/reference/en/language/functions/communication/spi/ (accessed Oct. 04, 2022).
32. "MFRC522 - Arduino Reference." https://www.arduino.cc/reference/en/libraries/mfrc522/ (accessed Oct. 04, 2022).
33. J. Nicol, "Introduction to Salted-Hashed Passwords - Medium," Feb. 25, 2020. https://medium.com/swlh/introduction-to-salted-hashed-passwords-d19bd6f92480 (accessed Oct. 04, 2022).
34. J. Lake, "What is SHA-2 and how does it work?," Feb. 17, 2022. https://www.comparitech.com/blog/information-security/what-is-sha-2-how-does-it-work/ (accessed Oct. 06, 2022).
35. "PHP: hash - Manual." https://www.php.net/manual/en/function.hash.php (accessed Oct. 10, 2022).
36. V. Sivadas, "GitHub - Advanced-Http URL Connection: Making Http URL Connection easy and secure," Apr. 02, 2022. https://github.com/VishnuSivadasVS/Advanced-HttpURLConnection (accessed Oct. 04, 2022).
37. "JSON Introduction." https://www.w3schools.com/js/js_json_intro.asp (accessed Oct. 04, 2022).
38. "Postman API Platform | Sign Up for Free." https://www.postman.com/ (accessed Oct. 10, 2022).
39. "Volley overview | Volley." https://google.github.io/volley/ (accessed Oct. 04, 2022).
40. "Make a standard request Volley." https://google.github.io/volley/request.html (accessed Oct. 04, 2022).
41. "JsonArray (Java(TM) EE 7 Specification APIs)." https://docs.oracle.com/javaee/7/api/javax/json/JsonArray.html (accessed Oct. 04, 2022).
42. "Json Object (Java(TM) EE 7 Specification APIs)." https://docs.oracle.com/javaee/7/api/javax/json/JsonObject.html (accessed Oct. 04, 2022).
43. V. Tabora, "Databases and Blockchains, The Difference Is In Their Purpose And Design | Hacker Noon," Aug. 04, 2018.

44. https://hackernoon.com/databases-and-blockchains-the-difference-is-in-their-purpose-and-design-56ba6335778b (accessed Oct. 05, 2022).
45. "Multichain - Cross-Chain Router Protocol." https://multichain.org/ (accessed Oct. 03, 2022).

AUTHORS PROFILE

Alícia F. S. Luís is an undergraduate informatics engineering student at the Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal. She has an interest in networking and software development.

Gonçalo M. C. Martins is an undergraduate informatics engineering student at the Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal. He has an interest in networking and software development.

João M. L. P. Caldeira is a Professor at the Polytechnic Institute of Castelo Branco, Portugal, and a Researcher at the Instituto de Telecomunicações, Portugal. He is also Co-Founder of the start-up Inspiring's, Lda. He received the Ph.D. degree in Computer Science and Engineering from the University of Beira Interior, Covilhã, Portugal and from the University of Haute Alsace, Colmar, France. He received the B.S. degree (Hons. (5-year)) in Electrical and Computer Engineering from the University of Coimbra, Portugal, and the M.Sc. degree in Information Systems and Technologies from the University of Beira Interior, Covilhã, Portugal. He has authored or co-authored more than 40 papers in refereed book chapters, journals, and conferences. He has served as a TPC member and reviewer for many international conferences and journals. His current research interests include mobility support for wireless sensor networks, Internet of Things, and smart cities.

Vasco N. G. J. Soares is a Professor and Cisco Instructor at the Polytechnic Institute of Castelo Branco, Portugal, and a Researcher at the Instituto de Telecomunicações, Portugal. He is a Co-Founder of the start-up InspiringSci, Lda. He received a five-year B.Sc. degree (licentiate) in Informatics Engineering from the University of Coimbra, Portugal and the Ph.D. degree in Computer Science and Engineering from the University of Beira Interior, Covilhã, Portugal. He has authored or co-authored more than 80 papers in refereed book chapters, journals, and conferences, and 2 patents. He has served as a TPC member and reviewer for many international conferences and journals, and as a publicity chair and web chair for several international conferences. He is a member of the editorial board of international journals. Also, he has participated in several national and European research projects related to vehicular delay-tolerant networks and industrial internet of things. He is an IEEE Senior Member. His current research interests include vehicular networks, delay/disruption tolerant networks, Internet of everything, smart cities, and technological solutions for the agro-industrial sector.