Universidade da Beira Interior
Departamento de Informática

Nº 1 - 2023: ***Developing an IoT System Utilizing Raspberry Pi for Efficient Class Attendance Management***

Developed by:

Guilherme Gonçalves Teixeira - a45662

Supervisor:

Professor Dr. Bruno M. C. Silva

June 25, 2023

Acknowledgements

I would like to express my heartfelt gratitude to everyone who played a significant role in the successful completion of my final project for the Computer Engineering graduation. Your unwavering support, dedication, and expertise were instrumental in bringing this endeavor to fruition, and I am truly grateful for your contributions.

First and foremost, I would like to thank my project advisor, Professor Dr. Bruno M. C. Silva, for his exceptional mentorship, advice and feedback, which were essential for the development and execution of the entire project. With his invaluable contributions, I was able to navigate challenges, grow as a student, and ultimately achieve success.

Additionally, I would like to express my sincere gratitude to the "SINS Lab" group for their generosity in allowing me to utilize their Raspberry Pi kit. Their support and provision of necessary resources were instrumental in the successful completion of this project.

Furthermore, it is important to mention my colleague Henrique Ferreira, who was responsible for the digital data analysis platform and the for the database. Teamwork was crucial for the success of this project, and our collaboration and communication were essential in achieving the established goals.

I cannot fail to acknowledge the fundamental role of all members of the Department of Informatics at University of Beira Interior (UBI), who provided us with quality education, and all the professors who helped us acquire the necessary knowledge throughout the course to carry out this project.

I also express my gratitude to all the close people who supported and encouraged me throughout this process, including family, girlfriend, friends, and classmates. The emotional support and encouragement I received were crucial in maintaining my determination and perseverance throughout the project. Thank you very much to everyone for the unconditional support and constant encouragement.

In summary, this project was an enriching and rewarding experience, and it was only possible thanks to the hard work and dedication of all those involved. Once again, I thank everyone for making this project a reality.

Contents

List of Figures

List of Tables

Code snippet list

Acronyms

UBI University of Beira Interior
IoT Internet of Things
RFID Radio Frequency Identification
URL Uniform Resource Locator
WAP Wireless Access Point
CPU Central Processing Unit
JS JavaScript
HTML Hypertext Markup Language
CSS Cascading Style Sheets
EJS Embedded JavaScript
ID Identification
JSON JavaScript Object Notation
HTTP Hypertext Transfer Protocol
GUI Graphical User Interface
IP Internet Protocol
QR Quick Response
SQL Structured Query Language
MySQL My Structured Query Language
WPA2 Wi-Fi Protected Access 2
AI Artificial Intelligence
NFC Near Field Communication
UHF Ultra-High Frequency
WWW World Wide Web
SAMS Student Attendance Management System
LAMS Lecture Attendance Management System
UUID Universally Unique Identifier

API Application Programming Interface

FR Functional Requirement

NFR Non-Functional Requirement

Chapter

1

Introduction

1.1 Background

This report presents my final project for the Bachelor's Degree in Computer Engineering at University of Beira Interior (UBI), focusing on the development of an Internet of Things (IoT) solution for tracking classroom attendance.

This project is part of the UBI - Learning HUB project being developed by the University of Beira Interior, which involves the construction of a module called Learning Analytics. This module aims to collect various data from a classroom, including attendance control, entry and exit times, and the seating location of students. Therefore, in the development of this project, I worked closely with Henrique Ferreira, a Master's student in Computer Engineering and responsible for the development of Learning Analytics, that is also under the supervision of Professor Bruno Silva.

In summary, the main objective was to create a framework that acts as an intermediary between students and teachers, utilizing IoT technology to gather and process data related to student attendance in the classroom. The system then transmits this information to the data analysis platform, offering valuable insights to teachers regarding student performance. This approach ensures accurate and efficient attendance tracking while also monitoring student punctuality, thereby optimizing the teaching-learning process and reducing fraudulent cases.

1.2 Motivation

Education is a dynamic field that constantly evolves, and technology plays a pivotal role in driving this progress. As electronic devices become more prevalent in classrooms, the development of technology-driven solutions becomes increasingly vital to enhance the quality of teaching. In this context, the utilization of IoT has emerged as a valuable approach for collecting real-time data, offering significant contributions to the teaching and learning process.

The selection of this project is justified by its relevance in improving education through the development of technological solutions, particularly by harnessing the power of IoT for real-time data collection. Moreover, this solution holds immense potential to deliver substantial benefits to both students and teachers, optimizing the teaching and learning process and fostering student success.

1.3 Goals

The primary objective of this project is to develop an IoT system that collects student attendance information for subsequent analysis on a data analytics platform. The system aims to serve as an intermediary between students and the data analytics platform, facilitating direct interaction with both parties.

To accomplish this, the project intends to deploy a Raspberry Pi device in the classroom, acting as a hotspot for students to connect to the network. Students will then scan a Quick Response (QR) code located on their tables and enter their student number on the website, hosted on the Raspberry Pi web server, to register their attendance. The system will subsequently transmit the attendance data to the shared database linked to the digital platform.

Moreover, the project will incorporate anti-fraud mechanisms to prevent students from marking the attendance of absent classmates or absent students from falsely marking their attendance remotely. Additionally, the system will address logistical errors, such as verifying table availability, confirming student enrollment in the specific class, and checking if the class is currently in session.

By implementing these measures, the project aims to establish an efficient and dependable system for tracking student attendance, utilizing IoT technology while ensuring security and addressing logistical challenges. This solution will yield benefits for both students and the educational institution, enhancing attendance data management and subsequently improving the efficiency and effectiveness of the teaching and learning process.

1.4 Document Organization

To demonstrate the stages of the process carried out, this report is organized as follows:

1. The first chapter – **Introduction** – presents the project, the reasons that led to its choice, the context in which it operates, the objectives to be achieved and its structure.

2. The second chapter – **State of the Art** – discusses how IoT can improve classroom attendance monitoring and also what are the most commonly used solutions to combat this problem and their differences when compared with the developed solution.

3. The third chapter – **Technologies Used** – highlights the main concepts related to this project , as well as the technologies used throughout the application development cycle.

4. The fourth chapter – **Software Engineering** – provides a comprehensive overview of software engineering principles, system architecture, requirements analysis, use cases, and activity diagrams, all of which are crucial for the successful development of the software.

5. The fifth chapter – **Implementation** – focuses on web application development and Raspberry Pi configuration. It begins with an introduction and then delves into the details, covering topics such as webpages, server configuration, database queries, and communication between the website and the server. Subsections within communication discuss the verification of QR code data, user phone Identification (ID), and student registration. The chapter also includes information on configuring Raspberry Pi, setting up an access point, and running the server at startup. A user manual provides instructions for Raspberry Pi installation in the classroom, working with QR codes, and student usage. Tests are conducted, and the chapter concludes with overall conclusions drawn from the discussed topics.

6. The sixth chapter – **Conclusions and Future Work** – provides a comprehensive overview of the main conclusions derived from the project. The chapter summarizes the key findings and outcomes, emphasizing the significant insights and implications that have surfaced throughout the project. Additionally, it explores future work by identifying potential avenues for further investigation and development, taking into

account the project's limitations and areas that demand additional research. This section aims to inspire and guide future researchers by proposing potential directions to extend the current project and advance the field.

7. The seventh chapter – **Bibliography** – the emphasis is placed on providing the referenced content throughout the entirety of the project.

Chapter

2

State of the Art

2.1 Introduction

This chapter serves as an introduction to the application of IoT in the field of education, with a specific focus on monitoring student attendance in the classroom. Furthermore, it examines various solutions commonly employed to address the challenge of recording attendance and compares them with the chosen approach.

2.2 IoT Potential in Education

The IoT represents an innovative technology that has the potential to connect various devices, objects, and sensors to the Internet, enabling seamless information exchange and remote control capabilities. This transformative technology holds immense promise in revolutionizing different aspects of our modern lives, including the monitoring of individuals' presence in specific locations.

In the context of classroom settings, one area where the application of IoT holds significant interest is attendance monitoring. Leveraging the capabilities of IoT, it becomes possible to gather real-time data on student attendance within a particular class. Various monitoring technologies can be employed to achieve this goal, including QR codes, motion sensors, cameras, and attendance card readers.

The data collected through these IoT-enabled attendance monitoring systems holds immense value. It provides valuable insights into students' attendance patterns and behaviors within the classroom environment. This information encompasses metrics such as attendance frequency, average time

spent in class, frequently visited areas, and other pertinent factors. By leveraging this data, educational institutions can enhance student safety, optimize classroom space utilization, streamline operational efficiency, and ultimately deliver an improved learning experience for students.

Furthermore, IoT-based classroom attendance monitoring empowers teachers and educational institutions to identify potential issues related to student attendance or behavior. By promptly recognizing these challenges, appropriate measures can be taken to effectively address them, fostering an environment conducive to enhanced academic performance among students.

In summary, the integration of IoT into classroom attendance monitoring enables the collection of real-time data, providing valuable insights into student attendance patterns and behaviors. This information empowers educational institutions to optimize resources, enhance safety measures, and proactively address attendance-related concerns. As the IoT revolutionizes attendance monitoring, the educational landscape stands to benefit from increased efficiency and an enriched learning environment.

2.3 IoT Revolutionizing Classroom Attendance: Top Trends

In this chapter, we will explore some of the most widely used and successful approaches to combat the problem of classroom attendance marking. These solutions harness the potential of IoT devices, machine learning algorithms, and data analytics to revolutionize attendance tracking and improve overall classroom management. Let us delve into the realm of these innovative solutions and discover their transformative impact on the educational landscape.

- IoT Based Secured Online Attendance Management System [1] - This document presents a web application that uses Wi-Fi technology to track and manage employee attendance. It discusses design options like IoT sensors, cloud storage, and Wi-Fi positioning. The application offers features such as automated attendance, location tracking, face detection, real-time cloud database and chat-box. It includes admin and staff modules, with admin having extra rights to check employee location and attendance summaries. The goal is to create a reliable, cost-effective, and secure system for employee monitoring.

- Machine learning classifier model for attendance management system [2] - This document explores facial recognition methods and introduces a new system for recording and storing faces to track attendance. The

system includes registration and face detection, overseen by an administrator. It captures multiple images of the user's face and identifies facial features like eyes, eyebrows, nose, mouth, and jawline. The system compares the captured face with the database to update attendance records automatically. To accomplish this, the proposed system employs the Haar cascade classifier for face detection, local binary pattern for training images, and Python for attendance processing.

- Automated Attendance System in the Classroom Using Artificial Intelligence and Internet of Things Technology [3] - The document outlines a facial recognition system designed for automated attendance in smart classrooms. It is integrated into an embedded device and utilizes Artificial Intelligence (AI) model and IoT technology for real-time monitoring and updating of attendance records. The system boasts low operating costs and can function independently of a network connection. The document covers the system's technology and deployment, including the utilization of a web server and training of AI models. The system achieves high accuracy and quick recognition within a range of 4-15 meters.

- Student Attendance Management System (SAMS): An IoT Solution for Attendance Management in Universities [4] - The document presents the development and deployment of SAMS, a student attendance management system for universities. SAMS utilizes fingerprint verification and wireless communication. SAMS is characterized as user-friendly, dependable, and affordable, with the potential to meet crucial attendance management requirements in universities.

- A Novel Framework for Smart Classroom Lecture Attendance Management System (LAMS) using IoT [5] - The document presents a proposed system architecture for a smart classroom lecture attendance management system that employs fingerprint verification. The system consists of several components, including a power supply unit, buzzer system, microcontroller unit with display, keypad unit, and relay switch. The objective of the proposed system is to offer a smart and secure biometric attendance solution for lecturers by utilizing fingerprint verification.

- Enhanced Biometric Attendance Management System using IoT & Cloud [6] - The document examines two distinct systems. The first system is an information service management system that leverages real-time database and cloud functions to develop IoT applications. The second system is an advanced biometric attendance management system that

employs IoT and cloud technology to capture attendance data using fingerprint or facial recognition modules. The attendance records are securely stored on the cloud for backup and security purposes and can be accessed via a mobile app.

- IoT Based Identification And Attendance Monitoring System Using Design Thinking Framework [7] - The document introduces a proposed ID and attendance monitoring system that combines Radio Frequency Identification (RFID) tags and face detection models to monitor the entry and exit of students and employees on campuses. The system's goals include reducing proxy attendance, detecting unauthorized individuals, and maintaining comprehensive digital attendance records. It is presented as a valuable tool that can replace previous methods of attendance and dress code monitoring. The document also outlines future plans to enhance the project, such as incorporating Near Field Communication (NFC) cards to monitor attendance and behavioral patterns of individuals.

- Development of Attendance Monitoring System using IoT Technologies [8] - The document introduces a proposed student attendance monitoring system that utilizes RFID technology and an IoT hardware platform for automated attendance tracking. The system offers students the ability to view their attendance rate and timetable, while instructors can access their own attendance records as well as their students' attendance rate. Instructors can also modify attendance status if needed and send notifications to students. Additionally, the system includes an Auto Notifier feature that alerts instructors and students when the absence rate falls below a certain percentage.

- Cloud Based Smart Attendance System for Educational Institutions [9] - The document explores the integration of cloud computing, IoT, and fingerprint technology for automating attendance management in educational institutions. The traditional manual attendance system, prone to errors and impersonation, is identified as a challenge. The proposed system aims to overcome these challenges by leveraging cloud storage and real-time access to attendance data. Fingerprint scanning is utilized to ensure accurate identification and prevent impersonation. Storing attendance data in the cloud addresses storage limitations. The system currently relies on an active internet connection, but future enhancements may include an offline mode and course-specific attendance sheets.

- Log System for Mass Crowd and Live Tracking Using Radio Frequency Based on IoT [10] - The document discusses the problems with traditional attendance management systems and proposes a new RFID-based attendance system that uses passive tags and Ultra-High Frequency (UHF) RFID readers for marking attendance and live location mapping of individuals within an organization. The proposed system is evaluated against traditional methods and barcode readers, and the results show that it increases the simplicity and feasibility of attendance management by up to 90%. The system also sends a confirmation message to authorized personnel for attendance confirmation and can provide information about daily job sheets of individuals.

2.4 Advantages of Our Innovative Solution

For the development of this project, we consciously chose to deviate from the conventional approaches for attendance tracking, as shown in table 2.1. Instead, we opted to implement QR codes assigned to each table, containing information such as the Uniform Resource Locator (URL) of the website where the student will mark its attendance, the room, and the table where the student is seated. This alternative solution not only fulfills the essential requirement of tracking attendance but also provides the added advantage of accessing information about the seating arrangement of the students, which was a crucial aspect of our project.

A QR code is a matrix barcode consisting of black squares arranged on a white background in a square grid pattern. QR codes can store a large amount of data compared to traditional barcodes. They can encode alphanumeric characters, binary data, or even URL's, making them versatile and flexible for various applications.

By employing QR codes, we eliminate the need for students to carry any form of identifying tag. They can simply use their mobile phones to scan the QR code and access the designated website to register their presence seamlessly. The necessary verifications are carried out discreetly in the background, ensuring a streamlined and hassle-free experience for the students.

Furthermore, adopting QR codes proves to be a cost-effective solution compared to RFID tags, which would require each student to possess an individual tag. When compared to solutions that leverage facial recognition, it also becomes much more cost-effective, as no type of camera is required. The use of QR codes significantly reduces expenses since it relies on the prevalence of smartphones among students, eliminating the need for additional hardware (sensors, cameras, etc) or specialized tags.

In summary, our decision to implement QR codes for attendance tracking not only efficiently fulfills the project's requirements but also offers enhanced functionalities such as seamless access to seating arrangement details. Additionally, the cost-effectiveness and simplicity of using QR codes make them the optimal choice, as they only require a smartphone to facilitate attendance registration for students.

Ref	Technology used	Our solution
[1]	Wi-Fi, IoT Sensors, Cloud-Based Storage, Wi-Fi Positioning Algorithms	QR codes, Web Appplication, Raspberry Pi , Wireless Communication
[2]	Facial Recognition, Haar Cascade Classifier, Local Binary Pattern	
[3]	Facial Recognition, Embedded Device, IoT Connectivity, Web Server	
[4]	Fingerprint Verification, Wireless Communication	
[5]	Fingerprint Verification, IoT devices or sensors	
[6]	IoT devices and sensors, Cloud functions, Mobile app	
[7]	RFID, Face detection models, NFC cards	
[8]	RFID, Web App	
[9]	Cloud Computing, IoT devices or sensors, Fingerprint Verification	
[10]	RFID	

Table 2.1: Comparing our solution to the existing alternatives

2.5 Conclusions

In conclusion, this project was undertaken with the goal of creating an innovative, practical, and cost-effective solution for real-time and reliable classroom attendance tracking.

Chapter

3

Technologies and Tools Used

3.1 Introduction

This chapter summarizes the technologies used in the development of the project, specifically in the creation of the web server, web application and project report.

3.2 Raspberry Pi 4 Model B 1.5GHz 8GB

We utilized a Raspberry Pi 4 Model B as a versatile Wireless Access Point (WAP) and web server. The Raspberry Pi 4 Model B boasts a built-in Wi-Fi adapter that supports WAP mode, enabling the device to function as an access point. Moreover, its Gigabit Ethernet port enhances networking capabilities, surpassing those of its predecessors, resulting in swifter data transfer between the server and clients. As a consequence, latency is reduced, and the overall performance of the web server is greatly enhanced. The inclusion of 8GB RAM empowers the web server to handle a greater number of simultaneous requests and process data more efficiently. Additionally, the improved 1.5 GHz Central Processing Unit (CPU) furnishes heightened processing power, thereby offering superior performance for web server tasks.

3.3 Node.js

Node.js is an open-source JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript code on the server-side, making it an excellent choice for building servers and network applications. Node.js

offers an event-driven, non-blocking I/O model, which enables handling multiple connections simultaneously without getting blocked. Its extensive package ecosystem, facilitated by the npm registry, provides a wide range of ready-to-use modules that can be easily integrated into server applications. With Node.js, we can leverage the power of JavaScript to create efficient and scalable servers, making it a popular choice for Raspberry Pi projects and beyond. We also employed Express.js, a web application framework built on top of Node.js, to streamline the development of our web server and effectively handle HTTP requests. Express.js provides a robust set of features and utilities, making it a valuable tool for building web applications.

3.4 MySQL

My Structured Query Language (MySQL) is a popular and widely used relational database management system that offers a range of advantages for project databases. With its intuitive interface and Structured Query Language (SQL)-based query language, MySQL provides developers with a user-friendly environment to efficiently manage their data. Its robust performance capabilities enable it to handle large volumes of information with ease, ensuring quick response times and efficient data retrieval. MySQL's compatibility with various platforms and programming languages further enhances its versatility, making it adaptable to different project requirements. Additionally, MySQL's reliability, security features, and active community support contribute to its reputation as a trusted and dependable choice for database management. In the context of this project, it was employed to execute queries to the database (5.3.3) from the web server.

3.5 JavaScript

JavaScript (JS) is a widely used programming language that has become a fundamental tool for web development. It enables developers to add interactivity and dynamic features to websites, making them more engaging and user-friendly. With its versatility and extensive libraries, JS empowers developers to create everything from simple interactive forms to complex web applications.

3.6 EJS

Embedded JavaScript (EJS) is a versatile templating engine that simplifies the process of generating dynamic HTML content. It seamlessly blends JavaScript code with HTML markup, enabling developers to build dynamic web pages with ease.

3.7 HTML

Hypertext Markup Language (HTML) is a fundamental building block of the modern web. It serves as the backbone of web pages, enabling the structure and presentation of content on the internet. With its simple yet powerful syntax, HTML allows developers to create interactive and visually appealing websites that are accessible to a wide range of users.

3.8 CSS

Cascading Style Sheets (CSS) plays a crucial role in web development by allowing developers to control the presentation and visual styling of HTML elements. It provides a powerful set of rules and properties that define how elements should be displayed, making it an essential tool for creating aesthetically pleasing and user-friendly websites.

3.9 LaTeX

LaTeXis a powerful typesetting system widely used in academia and scientific research for creating professional-looking documents. Its unique feature lies in its ability to handle complex mathematical equations, symbols, and scientific notations with ease. By employing a markup language, LaTeX allows users to focus on the content and structure of their documents while automating the formatting process.

3.10 GitHub

GitHub is an online hosting platform for source code and files, with version control, powered by the Git system. Version control allows programmers to manage all the changes made to the code of a given project through the concept of branching. This concept enables a developer to write code in a copy of

the main branch without risking potential damage to the original code. All of this has facilitated the application's development and allowed for its gradual development with detailed descriptions of each change made.

Link for the project repository: here.

3.11 Conclusion

In summary, this chapter provides an overview of the technologies and tools utilized throughout the project's development, showcasing their remarkable benefits and advantages.

Chapter

4

Software Engineering

4.1 Introduction

This chapter aims to detail the decisions made regarding the architecture and design of the system during its planning phase. The first section presents functional and non-functional requirements are surveyed, and several use case and activity diagrams are provided. Next, we show the overall system architecture, explaining how the attendance marking component interacts with the digital platform for teachers, along with the underlying database model used.

4.2 Requirements Analysis

4.2.1 Functional Requirements

The following Functional Requirement (FR) have been defined taking into account the needs of operators and system:

- **FR 01**: Handle the GET request from the QR code in the webserver.

FR 01	Handle the GET request from the QR code in the webserver
Function	Handle the GET request "http://10.42.0.1:3333/classroom/table" in the webserver.
Inputs	Classroom name, table number.
Outputs	Form page or feedback page.
Action	It is necessary to verify if the classroom actually exists, if there is any ongoing class in that room, if the table exists, and if it is available. If all the previously mentioned conditions are successfully verified, then the main page with the form for entering the student number should be returned to the student. Otherwise, a page with feedback on what went wrong should be returned.
Pre-conditions	Connection to the Wi-Fi network of the classroom, shared by the Raspberry Pi. Stable connection to the database. Stable connection to the web server hosted on the Raspberry Pi.
Side effects	N/A
Layout	N/A

Table 4.1: Specification table for FR 01

- **FR 02**: Clear the list of students' numbers and phone ID's whenever a class changes.

FR 02	Clear the list of students' numbers and phone ID's whenever a class changes.
Function	Clear the list of students' numbers and phone ID's whenever a class changes.
Inputs	Current class ID, Last class ID, phone ID's list, student numbers list.
Outputs	Form page or feedback page.
Action	If current class ID and last class ID are different, then clear phone ID's and student number lists.
Pre-conditions	This function should run inside of the GET request talked in FR 01 (4.2.1).
Side effects	N/A
Layout	N/A

Table 4.2: Specification table for FR 02

- **FR 03**: Obtain the student's phone ID and check if they have already marked their attendance.

FR 03	**Obtain the student's phone ID and check if they have already marked their attendance.**
Function	Obtain the student's phone ID and check if they have already marked their attendance.
Inputs	Student phone ID, phone ID's list.
Outputs	Form page or feedback page.
Action	When the form page is being loaded, the student's phone ID should be obtained in order to then make a Hypertext Transfer Protocol (HTTP) POST request to the server to check if the phone ID's list contains that student phone ID. If the student phone ID isn't in the list, then the main page with the form for entering the student number should be returned to the student. Otherwise, a page with feedback on what went wrong should be returned.
Pre-conditions	FR 01 (4.2.1) executed successfully. Connection to the Wi-Fi network of the classroom, shared by the Raspberry Pi. Stable connection to the database. Stable connection to the web server hosted on the Raspberry Pi.
Side effects	N/A
Layout	N/A

Table 4.3: Specification table for FR 03

- **FR 04**: Register student attendance.

FR 04	Register student attendance.
Function	Register student attendance.
Inputs	Student number.
Outputs	Feedback page.
Action	Upon submission of the form with the student's number for attendance registration, the first step is to verify if it is in the correct format (LETTER+NUMBER). Then, a HTTP POST request is made to the server to check if that student number is already registered for that class, if the class that was active when scanning the QR code is still the same, if the student is enrolled in the specific course and if the table is still available. If all the verifications are successful, then the attendance is recorded and a positive feedback page is shown. Otherwise, a page is displayed with information about what went wrong.
Pre-conditions	FR 03 (4.2.1) executed successfully. Connection to the Wi-Fi network of the classroom, shared by the Raspberry Pi. Stable connection to the database. Stable connection to the web server hosted on the Raspberry Pi.
Side effects	N/A
Layout	Simple page with a form that is submited by clicking on a button.

Table 4.4: Specification table for FR 04

4.2.2 Non-Functional Requirements

In this section, for a better understanding, Non-Functional Requirement (NFR) have been separated into following categories:

4.2.2.1 Usability

- **NFR 01**: The QR code should have a link that contains the classroom name, the table number, the Internet Protocol (IP) and the port of the webserver.
- **NFR 02**: The Raspberry Pi should function as an WAP as soon as it is connected to power.
- **NFR 03**: The Raspberry Pi should function as a web server as soon as it is connected to power, listening on port 3333.

- **NFR 04**: The shared Wi-Fi network name should be the name of the room where the Raspberry Pi is installed.
- **NFR 05**: The website should work on any of the following browsers: Google Chrome, Mozilla Firefox, and Microsoft Edge.
- **NFR 06**: The website should allow users with basic knowledge of web technologies to use the application to its full extent.
- **NFR 07**: The student must be connected to the room's wifi (provided by the Raspberry Pi), in order to be able to access the website

4.2.2.2 Availability

- **NFR 07**: The application should be able to communicate with the database at any time of the day.

4.2.2.3 Efficiency

- **NFR 08**: Requests between the server and the database should not delay the service.

4.3 Diagrams

4.3.1 Use Case Diagram

In order to illustrate the functioning of the system, a use case diagram was created that reflects the interactions between the system and the student. This diagram can be observed in Figure 4.1.

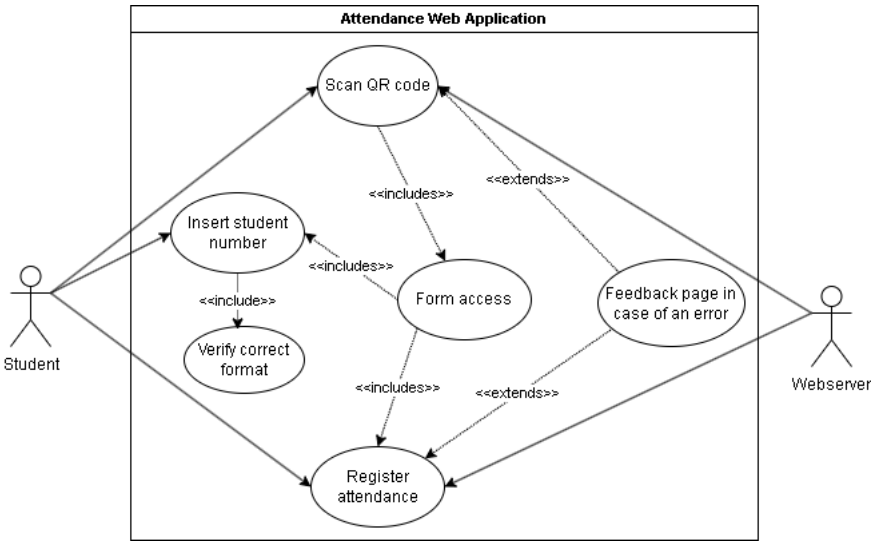

Figure 4.1: Web Application - Use Case Diagram

Name	**Register attendance process**
Actor	Student
Pre-conditions	Connection to the Wi-Fi network of the classroom, shared by the Raspberry Pi. Stable connection to the database. Stable connection to the web server hosted on the Raspberry Pi.
Data	Student number, Student phone ID, table number, classroom name.
Description	The student begins by interacting with the system when it scans the QR code on their table. Upon scanning, the system makes a request to the server, which performs several checks involving the student's table and classroom. If any of these checks fail, a feedback page is displayed. Otherwise, the user gains access to a page with a form where they can enter their student number. The number will only be accepted if it is in a valid format (letter + number). Finally, the student can click on the registration button, which sends a request to the server to validate their attendance. If any validation fails, a feedback page is displayed.

Table 4.5: Case 1 - Register attendance process

4.3.2 Activity Diagrams

Activity diagrams are valuable tools for illustrating the coordination between various activities within a system. In the context of this web application, the following activity diagrams will effectively showcase the system's operations and the subsequent server requests.

4.3.2.1 Website operation

This section explains the different stages involved in the student's attendance marking process on the website and outlines the actions taken during each stage. There are three primary stages that involve making requests to the web server: verifying the QR code link, obtaining and validating the student's phone ID, and completing the student registration. In case any of these stages encounter an issue, a feedback page is displayed; otherwise, the process proceeds seamlessly.

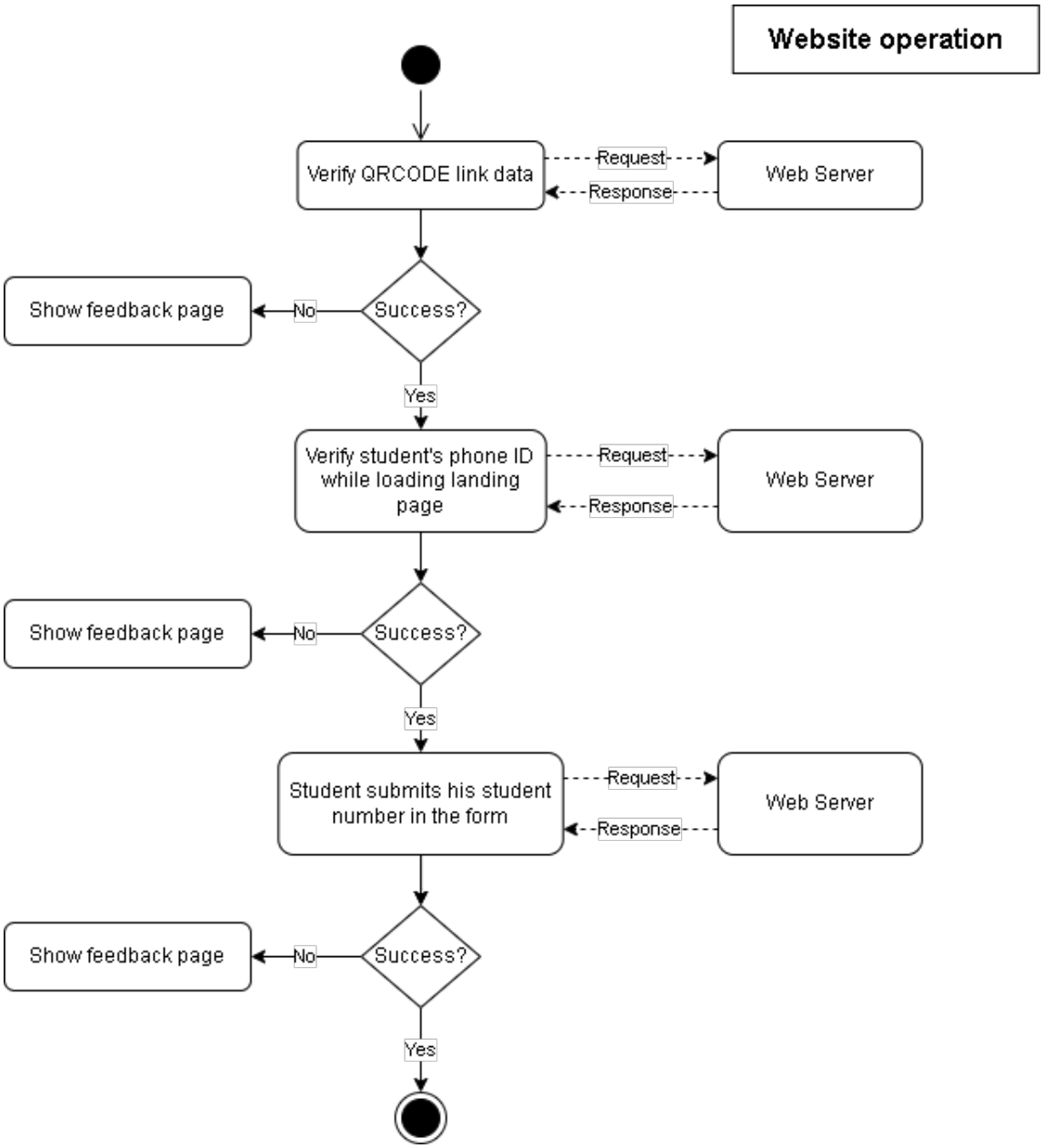

Figure 4.2: Website operation - Activity Diagram

4.3.2.2 Verify QR code data - Anti-fraude measure

In this section, we illustrate the process of verifying the content of the QR code scanned by the student through the server. The QR code follows the format "http://10.42.0.1:3333/classroom/table". The user will only receive a response from the server if they can access it, and to do so, they must be connected to the classroom's Wi-Fi (provided by the Raspberry Pi). This ensures that only physically present students can register their attendance on the website.

Then, the server's role is to ensure the existence of both the classroom and the corresponding table, as well as retrieve the currently active class in that particular classroom. To achieve this, the server begins by retrieving the ID of the currently active class and comparing it with the last recorded ID. If the ID's differ, it indicates that a new class is in progress, and consequently, the lists of student numbers and phone ID's are cleared. Furthermore, the server checks if the specified table in the QR code is already occupied by another student.

In case any of these verification steps fail, a feedback page is displayed to provide information on what went wrong. On the other hand, if all verifications are successful, the main page of the website is displayed, enabling the student to register their attendance.

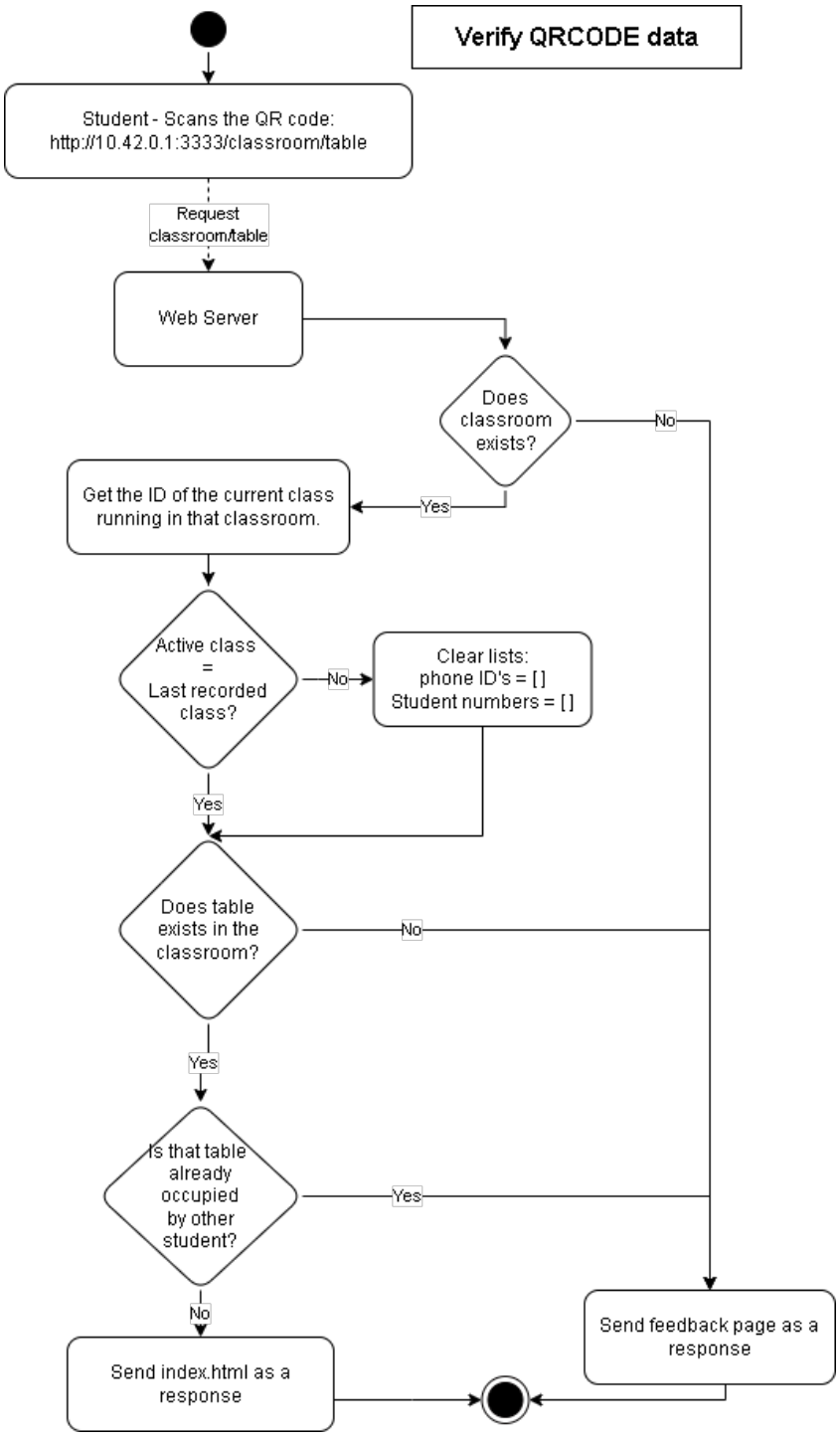

Figure 4.3: Verify QR code data - Activity Diagram

4.3.2.3 Verify user's phone ID - Anti-fraude measure

In this section, we demonstrate how the verification of the user's phone ID, who is attempting to load the HTML page, is performed. In other words, it checks whether their phone ID has already attended the class. If yes, then the user receives a page with feedback on what went wrong. Otherwise, the index.html page is loaded in its entirety. This system prevents a student do mark the attendance from an absent colleague.

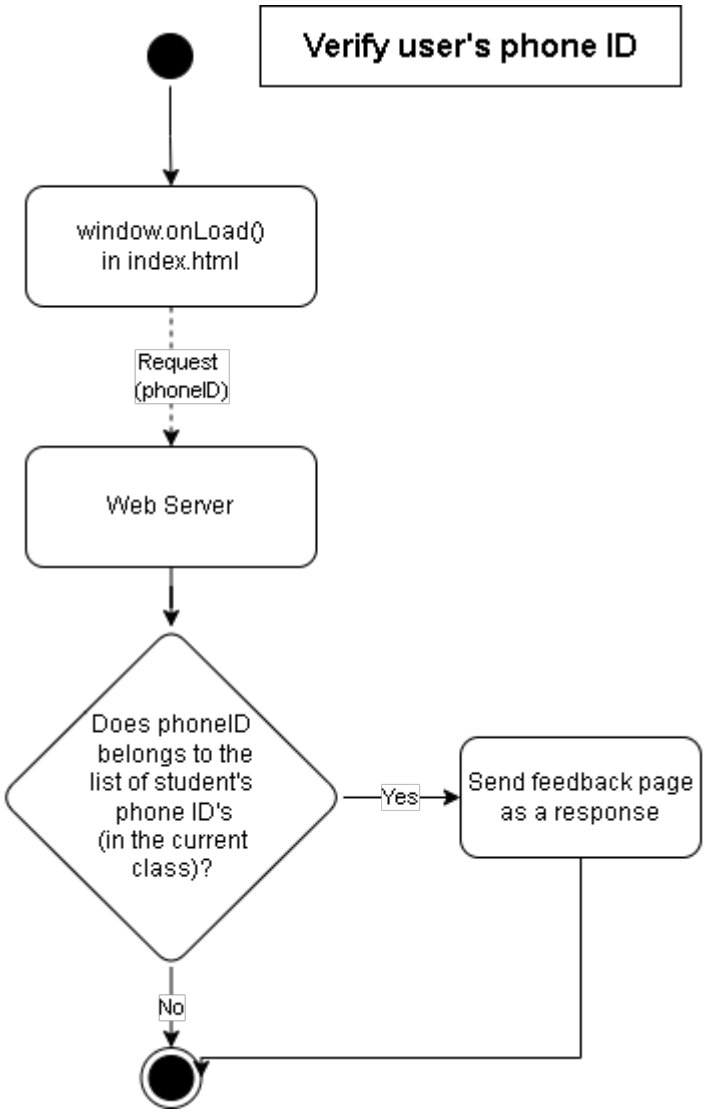

Figure 4.4: Verify user's phone ID - Activity Diagram

4.3.2.4 Register student

In this section, we will demonstrate how the student registration process is carried out. Firstly, when the student clicks on the registration button, we verify if the input for the student number is in the expected format (i.e., a letter followed by the student number). If it's not in the correct format, an alert is displayed. If the format is correct, a request is sent to the server, including the student number and the acquired information so far (phone ID, table name, table ID, room ID, initial class ID).

On the server side, we begin by checking if the student number is already registered for that class (i.e., if it already belongs to the list of student numbers). We also verify if the class is still the same as when the student scanned the QR code. Next, we check if the student is registered for the course associated with the ongoing class, and then we confirm if the table is still available. If any of these checks fail, a page with the respective feedback is displayed. If all checks are successful, the attendance registration process proceeds. We add the entry ['student_id', 'class_id', 'room_table'] to the student_logs table and update the status of the corresponding table to 'occupied' in the room_tables table.

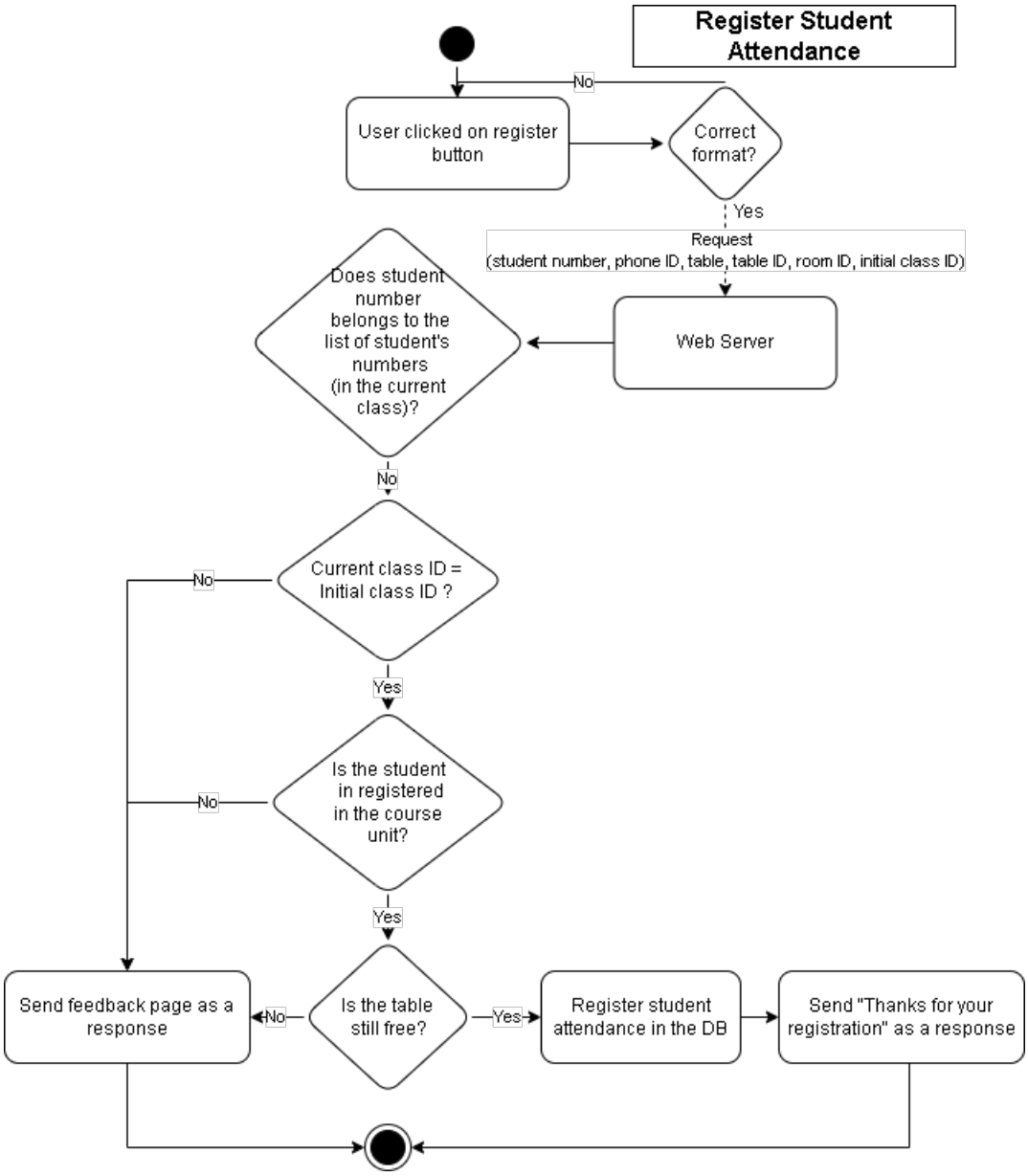

Figure 4.5: Register Student Attendance - Activity Diagram

4.3.3 System Architecture Diagram

In this section and in the figure 4.6, we present the architecture of the system as a whole. We establish the relationship between the classroom attendance marking system, developed by me (green background), and the database and digital platform for teachers, both developed by my colleague Henrique Ferreira (blue background).

Within the classroom, a Raspberry Pi will provide two services: a WAP and a web server. Students can connect to the Raspberry Pi's network and scan a QR code on their table to access the hosted website (Node.js server). On this website, they enter their student number, and the web server handles confirmations for accurate attendance without fraud attempts. The student receives feedback on the attendance record.

The classroom attendance system and the digital platform for teachers share a database with information on classrooms, tables, students, etc.

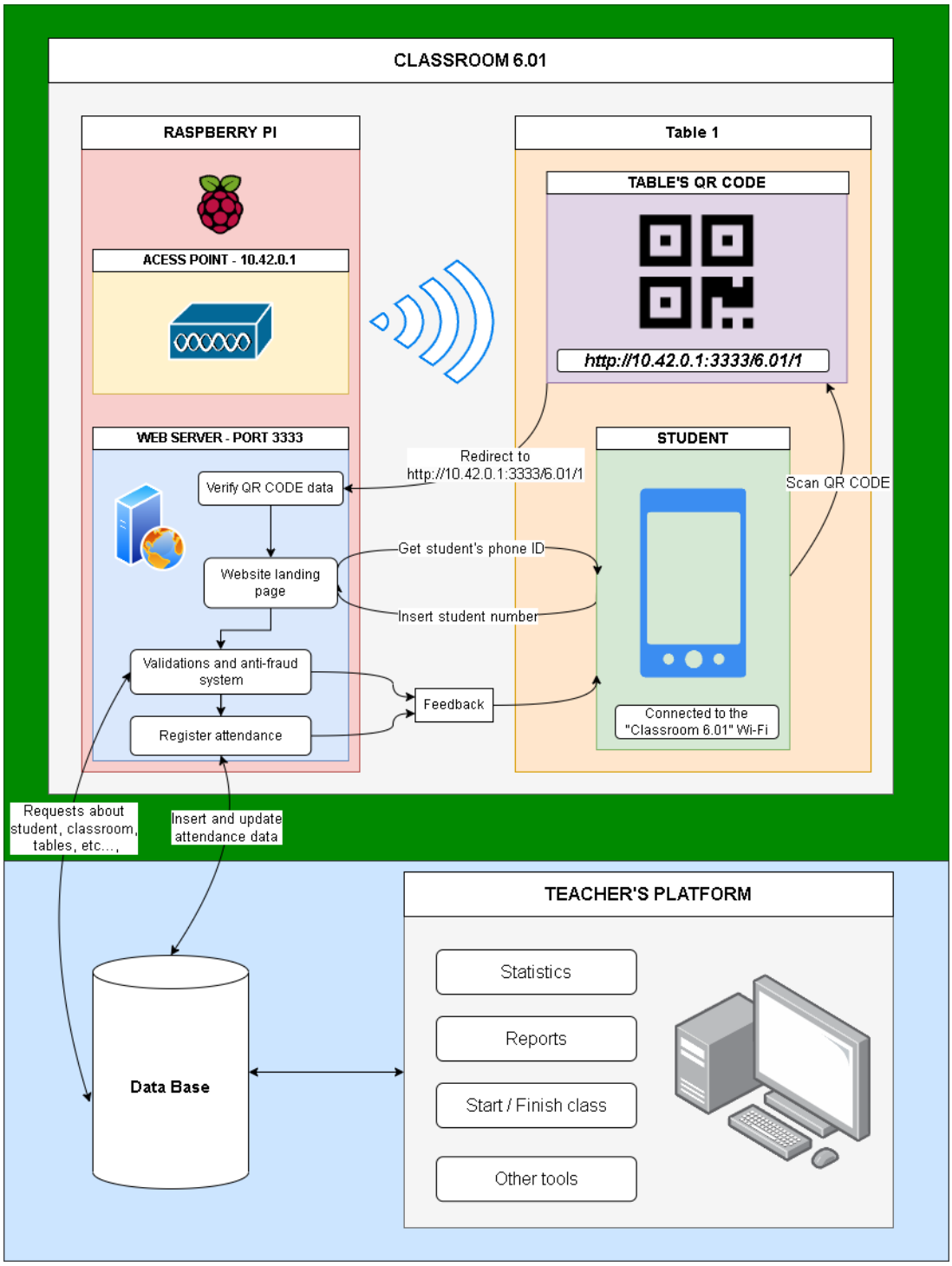

Figure 4.6: Conceptual Design of the System Architecture

4.4 Database

Regarding the database, it was developed and implemented in MySQL by my colleague Henrique Ferreira as part of the web application for teacher statistics. In this project, which I developed, I used the database only to retrieve relevant information for validating attendance records and to insert and update student attendance information.

To confirm the existence of specific classrooms and obtain their ID's, I utilized the "rooms" table, which consists of the columns "room_id" and "room_name".

To determine the active class in a particular classroom and subsequently discover the ID of the corresponding academic unit, I employed the "classes" table, which includes the columns "class_id", "class_name", "class_room", "class_status", "id_uc", "class_day", and "class_time".

To verify the existence of a specific table within a particular classroom and select its ID, I used the "room_tables" table, comprising the columns "room_table_id", "room_id", "table_number" and "tablet_status". I also utilized this table to check if the table was unoccupied and updated its status upon recording any attendance.

In order to ensure that the student attempting to mark attendance for a specific class is actually enrolled in that class and to retrieve their ID, I relied on the "students" table, which contains the columns "student_id", "student_name", "student_number", "id_uc" and "student_status".

Finally, I added a new row with the student's attendance data for the class to the "student_logs" table, which includes the columns "student_logs_id", "student_id", "class_id", "time_arrival" and "room_table".

4.5 Conclusions

In conclusion, this chapter enhances the comprehension of the application's operations, particularly regarding the website's interaction with the web server and the reasons behind it. It showcases diagrams that were integral to the project planning phase, illustrating these interactions.

Chapter

5

Implementation

5.1 Introduction

In this chapter, the description of the web application is provided, which was developed to enable students to mark their attendance in class. The concepts covered in the previous chapters are further explored, and finally, the user manual is presented, demonstrating the practical use of the application.

5.2 IoT System

The IoT aspect of this system includes a student scanning a QR code on their table, that contains some information, namely related to the classroom and table. . This QR code will direct the user to an endpoint in the format "http://10.42.0.1:3333/classroom/table," which will send a request to the Application Programming Interface (API) running on the server with the IP address 10.42.0.1 (Raspberry Pi) on port 3333. To ensure the success of this operation, the student must be connected to the Wi-Fi network shared by the Raspberry Pi, otherwise he will not be able to connect to the server.

5.2.1 QR codes

To ensure proper functioning of the system, there should be one QR code per table, each containing a link that will initiate a GET request to the web server, including the room and table information where it is located. Therefore, the QR code should contain a link in the format: "http://10.42.0.1:3333/classroom/table". Next, I will provide a more detailed explanation of the information present in the link:

- HTTP - is a protocol or set of rules that allows communication between web servers and web browsers. It is the foundation of data communication for the World Wide Web (WWW). HTTP governs how data is formatted and transmitted between a client (usually a web browser) and a server (where the website or web resource is hosted).

- 10.42.0.1 - RaspberryPi's (webserver) IP in the local network.

- 3333 - RaspberryPi (webserver) port where the server is running.

- classroom - Name of the classroom where the QR code is located.

- table - table number where the QR code is placed.

Therefore, after the user is redirected to this link, the web server analyzes the values "classroom" and "table" and returns a response that can be one of two pages, as will be discussed in 5.3.4.1.

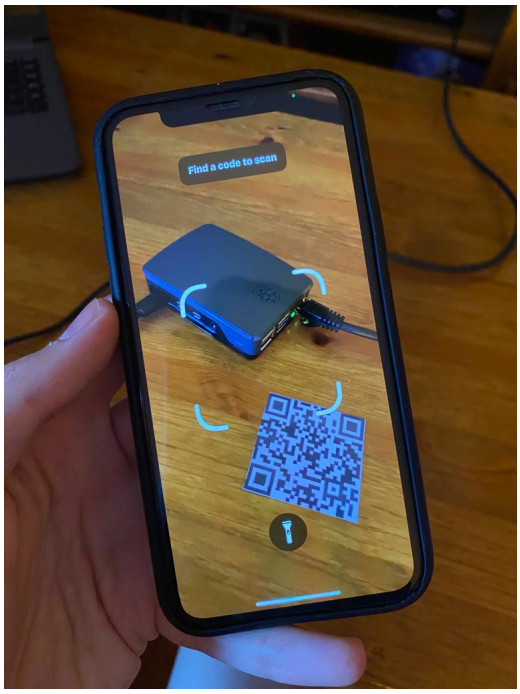

Figure 5.1: Reading the table's QR code

5.2.2 Raspberry Pi configuration

5.2.2.1 WAP

Raspberry Pi is a versatile single-board computer that can be used for various projects and applications. Setting up a wireless hotspot on the Raspberry Pi can enable it to act as a WAP, allowing other devices to connect to it and access the internet. In the following text, I will walk through the steps to configure the Raspberry Pi as a wireless hotspot using the Network Manager utility, according to [11].

1. Connect the Raspberry Pi to an Ethernet connection.
2. Open a terminal window on the Pi and update the device with the commands: "`sudo apt update`" and "`sudo apt upgrade -y`"
3. Use "`sudo raspi-config`" to edit the configuration of the Raspberry Pi via Graphical User Interface (GUI).
4. Using the cursor keys, navigate to Advanced Options and pressed Enter.
5. Navigate to Network Config and pressed Enter.
6. Select Network Manager and then clicked "OK".
7. Click on "OK" again.
8. Click on "Finish".
9. Select "Yes" to reboot the device.
10. After restart, left click on the Network icon, select "Advanced Options" and then "Create Wireless Hotspot".
11. Set the Network name of the access point, Wi-Fi security to Wi-Fi Protected Access 2 (WPA2), and then set the password for the WAP. Click create to save.
12. Reboot the Raspberry Pi again.
13. Click on the Network icon to check that the WAP is active.

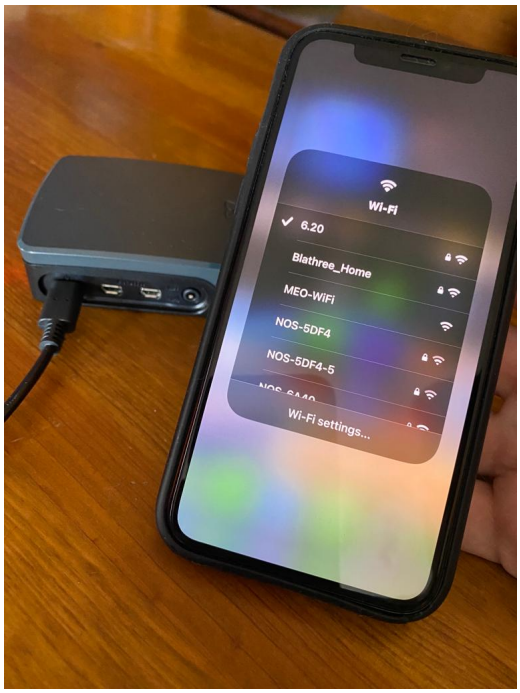

Figure 5.2: Connecting to the classroom (6.20) Wi-Fi

5.2.2.2 Server running at startup

Systemd is a powerful system and service manager for Linux that offers efficient management of various processes and services. One of its notable features is the ability to start and control services automatically during system startup. This capability is particularly useful when you want to ensure your Node.js server starts on boot without manual intervention.

Next, I will show the steps followed according to [12]:

1. Created a service unit file for the systemd on your system. For this I used the command "`sudo nano /etc/systemd/system/project.service`"

2. Inside that file I put the following code:

```
[Unit]
Description=IoT Project Service
Afetr=network.target

[Service]
WorkingDirectory=/home/guilherme/tabletop/IoT_Attendance_Project
ExecStart=/usr/bin/npm start
restart=always
User=guilherme

[Install]
WantedBy=multi-user.target
```

Code snippet 5.1: project.service file

3. After restarting the device, the service automatically started.

5.3 Web Application

Regarding the web application, the aim was to create a means of communication between the user and the server, for which an API was developed on the server side. During the "regular" use of the system by a student, there are three key moments where, depending on the information present on the server and the result of the queries it makes to the database, the student progresses in the attendance registration process or is blocked on a feedback page. These moments are as follows:

- A request with information about the room and table, where a series of checks will be performed, which will be discussed further in section 5.3.4.1. If any of these checks fail, the user will receive a "`response.ejs`" page, mentioned in section 5.3.1, as feedback. Otherwise, the user will receive the main page with the attendance registration form, "`index.ejs`", discussed in section 5.3.1.

- A request to validate the student's phone ID when they load the "`index.ejs`" page. In other words, it is checked whether it is the first time that the student with that phone ID is attempting to mark attendance for that class, which prevents fraudulent situations, such as a present student trying to mark the attendance of an absent student after marking their own attendance. If a fraudulent situation is detected, a feedback page is displayed; otherwise, the "`index.ejs`" page (5.3.1) is fully loaded. This point is addressed in section 5.3.4.2.

- A request to validate information related to the student number upon form submission. In this step, it is verified, for example, if the student number is indeed registered and if the table is still available, among other checks. If everything is successfully validated, the attendance is recorded. Otherwise, the student receives a feedback page explaining what happened. This point is addressed in Section 5.3.4.3.

5.3.1 Webpages

Regarding the web pages of the application, there are two with different objectives:

- index.ejs - This page will display the form for entering the student number. It is an HTML page with EJS templates, as it will automatically store some useful information during each user's registration process, handled by the server. Additionally, there are imported JS files that will handle requests to the server, which we will discuss later. Also, a JS library is imported to retrieve the user's mobile ID, which will be explained further. The HTML code of this webpage can be found in A.

- response.ejs - This page will display a feedback message in case something goes wrong (e.g., if the table where the user is trying to mark attendance is already occupied or if the student is not enrolled in the class). It is an HTML page with EJS templates, as the feedback message will be automatically inserted by the server. The HTML code of this webpage can be found in A.

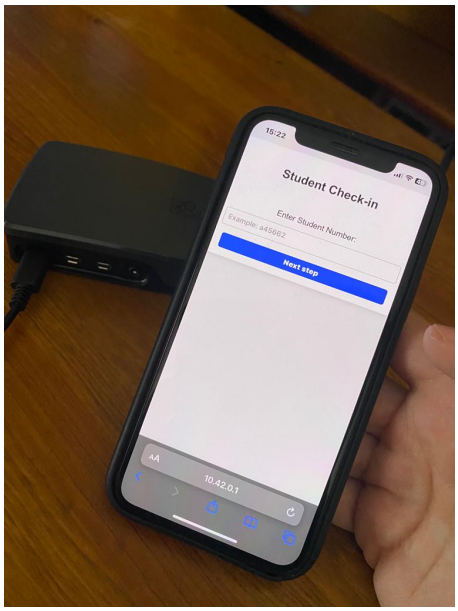

Figure 5.3: index.ejs shown after scanning the QR code

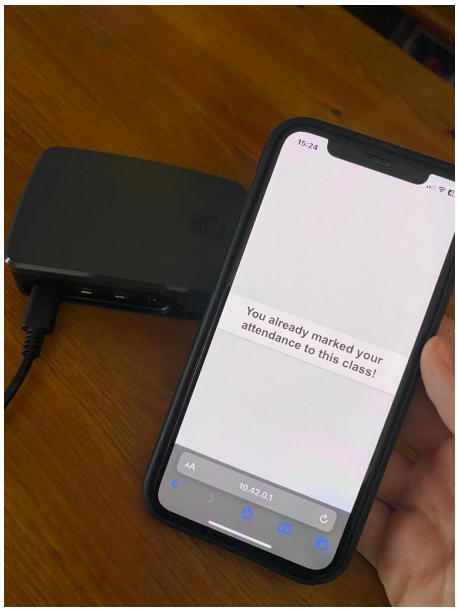

Figure 5.4: response.ejs displayed when a fraudulent activity is detected

5.3.2 Webserver configuration

The code presented in A sets up a server using the Express framework in Node.js. It imports necessary modules such as 'express', 'http', 'fs', 'mime', 'path', and 'mysql'. The Express application is created and configured with middleware for JavaScript Object Notation (JSON) parsing and template engine setup (EJS). Environment variables are loaded from a .env file. A connection to a MySQL database is established using the provided credentials. Static files are served using the express.static middleware, with proper Content-Type headers based on file extensions. An HTTP server is created and configured to listen on port 3333.

5.3.3 Database Queries

When it comes to queries for the database, the following functions have been developed, with each one accessing the database to insert or retrieve the necessary data for the attendance marking process.

- Does the classroom really exist? - To verify the existence of a classroom name in the database, a function has been constructed where the server executes a query to the database, requesting the ID of the classroom with the name provided as an argument to the function. If the room name does not exist, it calls the provided callback function with null. If it exists, it calls the provided callback function with the ID of that classroom. If there is an error, the callback function is called with the error as the first argument. The code for this function can be found in A.

- Get active class ID and course unit ID - To retrieve the current active class ID and the corresponding course unit ID in a specific classroom, a function has been implemented. This function utilizes a server-side query to the database, requesting the active class ID and course unit ID for the given classroom name. If there is no active class in the specified classroom, it calls the provided callback function with null. However, if an active class exists, it calls the provided callback function with both the active class ID and its associated course unit ID. If there is an error, the callback function is called with the error as the first argument. The code for this function can be found in A.

- Does the table really exist in the classroom? - To verify the existence of a table in the classroom, a function has been constructed where the server executes a query to the database, requesting the ID of the table with the classroom ID provided as an argument to the function. If the

table does not exist, it calls the provided callback function with null. If it exists, it calls the provided callback function with the ID of that table. If there is an error, the callback function is called with the error as the first argument. The code for this function can be found in A.

- Is the table occupied by other student? - This function checks the status of a table using its ID. It queries the database to retrieve the `tablet_status` of the table. If the status is "occupied," it calls the provided callback function with true. Otherwise, it calls the callback function with false. If there is an error, the callback function is called with the error. The code for this function can be found in A.

- Is the student registered in the course unit? - This function searches for a student's ID in the "students" table using their course ID and student number. It executes a query to retrieve the `student_id` based on the provided parameters. If a result is found, the callback function is called with the `student_id` as the second argument. If no result is found, the callback function is called with null as both arguments. If there is an error, the callback function is called with the error as the first argument. The code for this function can be found in A.

- Register student attendance log - The function inserts a new row into the "`student_logs`" table in the database. It takes three parameters: `columnNames`, representing the column names in the table, values, representing the corresponding values to be inserted, and callback, a function to be executed once the insertion is completed. The code for this function can be found in A.

- Update table status - This function updates the status of a table in the "`room_tables`" table of a database. It takes four parameters: `room_id`, `room_table`, `status`, and callback. The function constructs a SQL query to update the "`table_status`" column in the "`room_tables`" table with the provided status, based on the matching `room_id` and `room_table` values. The code for this function can be found in A.

5.3.4 Communication Between the website and the server

One of the main objectives of this project is to implement a communication protocol that allows the exchange of information between the web server and the website (user). Therefore, it was necessary to implement HTTP requests to request information and send data to be stored in the database.

In the context of this application, two different methods were implemented:

- GET - This method is used to retrieve data from the server, according to [13]. In this application, it was used for verifying the QR code content.
- POST - This method is used for sending data to a server to create or update a resource, according to [13]. In this application, it was used, for example, to submit a HTML form with the student number in 5.3.4.3.

Below, you will find code snippets with examples of using these types of requests, previously presented in section 4.3.2. The implementation on the server side, as well as its invocation on the client side, will be shown.

5.3.4.1 Verify QR code data - Anti-fraude measure

This section showcases the code snippets shown in A that involve the HTTP GET request from the client side and its handler in the web server. The handler function, `handleGetRequest`, accepts two parameters: a request (req) object, which contains the QR code content ("http://10.42.0.1:3333**/classroom/table**"), and a response (res) object.

Within the handler function, the code extracts the values of the `classroom` and `table` parameters from the URL using req.params. It then calls the `doesRoomNameExist` function (5.3.3) to verify the existence of the specified `roomName`. If the room does not exist, the function renders a response using the response template, displaying the message indicating that the room does not exist.

The code proceeds by invoking the `getActiveClassAndUCID` function (5.3.3) to retrieve the active class ID and course unit ID (UCID) associated with the `roomName`. If there is no active class, the function renders a response using the response template, displaying the message indicating that no class is currently active in the room. If there is one, it checks if it is different from the previous one (the class changed) and if it is, function clears the list of student numbers and phone ID's and updates the value of `classLastID`.

Subsequently, the code utilizes the `doesTableExist` function (5.3.3) to check if the specified table exists within the identified `roomID`. If the table does not exist, the function renders a response using the response template, displaying the message indicating that the table does not exist in the room.

Finally, the code employs the `isTableOccupied` (5.3.3) function to determine if the table is already occupied by another student. If the table is occupied, it sends a response page to the client with a message indicating that the table is already occupied. Otherwise, it renders the 'index' view, providing the room ID, room table, table ID, and active class ID as data.

In summary, this function handles a GET request in a web application, retrieving information about a specific room and table. It performs various

checks and provides appropriate responses based on the encountered conditions.

5.3.4.2 Verify user's phone ID - Anti-fraude measure

The code snippet presented in A runs in the load of index.ejs and initializes a variable called `phoneID` and assigns it the value retrieved from the browser's local storage using the key '`phoneID`'. If the value is not present in the local storage, it generates a new Universally Unique Identifier (UUID) using the `uuidv4()` function and saves it in the local storage with the key '`phoneID`'.

Then, it makes a POST request to the '/verify-phoneID' endpoint with the `phoneID` as the payload. The request includes the 'Content-Type' header set to 'application/json' and the body containing the `phoneID` in JSON format.

The response from the server is then converted to text and assigned to the HTML variable. Finally, the HTML content of the current document is replaced with the received HTML content.

The code snippet shown in A presents the function `handleVerifyPhoneID` that is associated with the route /verify-phoneID in the website. When a POST request is made to this route, the function is executed.

The function extracts the `phoneID` from the request body and checks if it exists in the `phoneIds` list. If the `phoneID` is found in the list, it means that the attendance for this class has already been marked by the student. In this case, the function renders a response using the response template, displaying the message "You already marked your attendance to this class!".

Overall, these functions are responsible for verifying if a particular `phoneID` has already marked attendance for a class and providing an appropriate response.

5.3.4.3 Register student

The `handleSubmit` function shown in A is designed to handle the submission of a form. When the function is called, it prevents the default form submission behavior.

Next, it retrieves the value entered in an input field with the ID "studentnumber" and assigns it to the variable`studentNumber`.

Several other elements with specific ID's are also retrieved and assigned to corresponding variables: `roomTable`, `roomID`, `tableID`, and `classActiveStart`.

If the `studentNumber` value matches the pattern '/$\hat{}$[a-zA-Z]+$/', which represents a letter followed by one or more digits, it proceeds with the following steps:

- It formats the `studentNumber` by capitalizing the first letter and concatenating it with the rest of the string.
- It initiates a fetch request to the '/register-studentNumber' endpoint using the POST method. The request includes a JSON payload containing the formatted `studentNumber`, as well as the values of `phoneID`, `roomTable`, `roomID`, `tableID`, and `classActive`.
- It handles the response from the server by converting it to text and replacing the HTML of the current document with the received HTML content.

If the `studentNumber` value does not match the specified pattern, a window alert is displayed, indicating that the student number must start with a letter followed by one or more digits.

The function presented in A is a handler for a POST request to register a student number. It expects certain data in the request body, including the student number, phone ID, room table, room ID, table ID, and active class ID.

The function performs several checks and actions based on the provided data. Here's a breakdown of the function:

- It first checks if the student number is already present in the `studentNumbers` array. If so, it renders a response indicating that the student has already marked attendance to the class.
- Next, it compares the `classActiveStart` with `classActiveID` to determine if the class is still active. If they are not equal, it renders a response indicating that the class is no longer active.
- If the class is still active, it proceeds to find the student's ID in the course unit using the `findStudentIDinUC` function (5.3.3). If the student ID is not found, it renders a response indicating that the student is not registered in the course unit.
- Assuming the student ID is found, it checks if the table specified by `tableID` is already occupied using the `isTableOccupied` function (5.3.3). If the table is occupied, it renders a response indicating that another student is already occupying that table.
- If the table is available, it adds a new row to the table with the student's ID, class ID, and room table information using the `addRowToTable` (5.3.3) function. It also updates the status of the table to "occupied" using the `updateTableStatus` function (5.3.3).

- Additionally, it adds the phone ID and student number to their respective lists (`phoneIds` and `studentNumbers`).

- Finally, it renders a response indicating successful registration.

Overall, this function handles the registration of a student number, performs various checks, updates the database, and renders appropriate responses based on the conditions met.

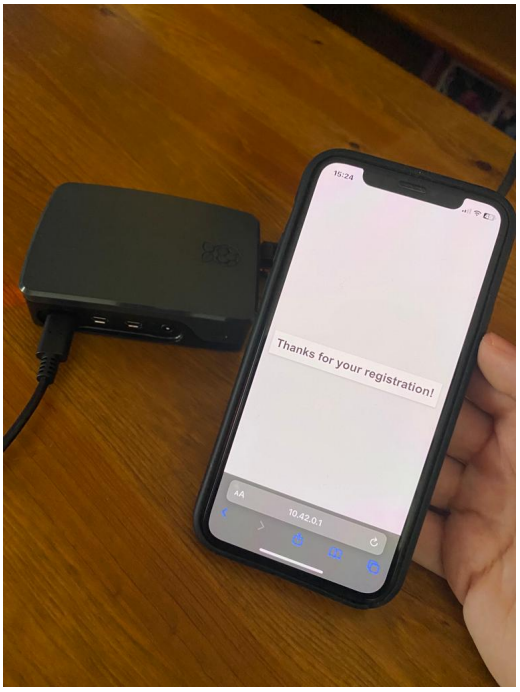

Figure 5.5: Successful registration of student attendance

5.4 User manual

5.4.1 Raspberry Pi installation in the classroom

In the following steps, we will outline the process of installing the application and configuring the Raspberry Pi in the classroom:

1. First, you need to download the project code to the Raspberry Pi. You can download it: here.

2. Next, navigate to the project's main directory, install Node.js, install npm, and run the command "`npm install`" to install the necessary dependencies.

3. Then, proceed with the configuration mentioned in 5.2.2, where the WAP's network name and password will be directly related to the room where the installation is being done. In the unit file, set the value of the "WorkingDirectory" variable to the path of the root folder downloaded in the first step.

Figure 5.6: Raspberry Pi used in this project

5.4.2 QR CODES

Regarding the content of the QR codes to be displayed for each table in the classroom, each one should contain the following link, which includes information about the specific room and table, as well as the IP address of the Raspberry Pi and the port where the web server is running.

For example, for the Raspberry Pi with IP address 10.42.0.1 and the server running on port 3333, and for table 1 in room 6.01, a QR code should be generated with the following link: "http://10.42.0.1:3333/6.01/1".

As many QR codes should be generated as there are tables in the respective room.

5.4.3 Student's usage

On the student's side, they simply need to connect to the Raspberry Pi network and then scan the QR code on their table. If everything goes smoothly, they only need to enter their student number and register their attendance by clicking the button in the webpage.

5.5 Tests

Regarding the testing phase, several simulations of classroom scenarios were conducted to explore all possible user pathways. Additionally, specific tests were conducted on the anti-fraud system. It was determined that the system functions as intended and is ready for implementation in a classroom setting.

5.6 Conclusions

In summary, this chapter extensively explained the complete implementation of the application, including the configuration used on the Raspberry Pi. Additionally, we have prepared a user manual for individuals interested in utilizing the system within a classroom setting.

Chapter

6

Conclusions and Future Work

6.1 Main conclusions

Upon concluding this project, I can say that it has allowed me to learn and solidify knowledge in topics, programming languages, tools, and technologies that I was previously unaware of or had limited exposure to. It has enabled my growth both as a student and as an individual.

6.2 Future work

This project will be put into practice during the first semester of the 2023/2024 academic year in a first-year class at the Department of Informatics, UBI.

The goals set for the project were successfully accomplished: the construction of a system that serves as an intermediary between students and teachers, enabling attendance marking in the classroom through IoT technologies.

Regarding future work, there are several improvements we could make. One of them would be to enhance the application's interface, making it more intuitive and user-friendly. Additionally, we could provide students with more information about the ongoing class on the page where they enter their student number. This would allow them to access relevant details about the course, such as schedule, location and more.

To further enhance the security of the system, it would be beneficial to notify the teacher whenever a student attempts to manipulate the attendance system. This notification could include information about the student, such as their name, student number, and details of the fraudulent attempt. This would help ensure the integrity and reliability of the system.

Furthermore, it's worth mentioning that this system could be extended to other cases that require attendance marking. For example, it could be used to track employee attendance in a factory, monitor gym members' presence, or even organize seating arrangements at events. The potential applications are diverse, and adapting the system to meet these different needs would be a promising step for the future.

Appendix

A

Appendix

```
<!DOCTYPE html>
<html>
    <head>
      <title>Student Check-in</title>
      <script src="/phoneID.js" type="text/javascript"></script>
      <script src="/handleRegister.js" type="text/javascript"></script>
      <link rel="stylesheet" href="/index-style.css">
      <script src="/uuidv4.min.js"></script>
      <meta name="viewport" content="width=device-width, initial-scale
          =1.0">
    </head>
    <body>
      <div class="container">
        <h1 id="title">Student Check-in</h1>
        <form id="form-student-number" onsubmit="handleSubmit(event)">
          <label for="student-number">Enter Student Number:</label>
          <input type="text" id="student-number" name="student-number"
              placeholder="Example: a45662" required>
          <button type="submit">Next step</button>
        </form>
      </div>
      <div class="info">
        <h1 id="roomID"><%= roomID %></h1>
        <h1 id="roomTable"><%= roomTable %></h1>
        <h1 id="tableID"><%= tableID %></h1>
        <h1 id="classActiveID"><%= classActiveID %></h1>
      </div>
    </body>
</html>
```

Code snippet A.1: index.ejs page.

```
<!DOCTYPE html>
```

```
<html>
    <head>
        <title>Student Attendance</title>
        <meta name="viewport" content="width=device-width, initial-scale
            =1.0">
        <link rel="stylesheet" href="/response-style.css">
    </head>
    <body>
        <h1><%= msg %></h1>
    </body>
</html>
```

Code snippet A.2: response.ejs page.

```
const express = require('express');
const http = require('http');
const fs = require('fs');
const mime = require('mime');
const path = require('path');
const mysql = require('mysql');

const app = express();
app.use(express.json());
app.set('view engine', 'ejs');

const dotenv = require('dotenv');
const envPath = path.join('/home/guilherme/Desktop/
    IoT_Attendance_Project/src', '..', '.env');
dotenv.config({ path: envPath });

const connection = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME
});

app.use(express.static(__dirname + '/website-student/', {
  index: false,
  setHeaders: (res, path) => {
    if (path.endsWith('.css')) {
      res.setHeader('Content-Type', 'text/css');
    } else if (path.endsWith('.js')) {
      res.setHeader('Content-Type', 'application/javascript');
    }
  }
}));

http.createServer(app).listen(3333, () => {
  console.log('Student server is running on port 3333.');
```

```
});
```

Code snippet A.3: Node.js server configuration

```
function doesRoomNameExist(roomName, callback) {
  const query = `SELECT room_id FROM rooms WHERE room_name = ?`;
  const values = [roomName];

  connection.query(query, values, function(error, results, fields) {
    if (error) {
      return callback(error, null);
    }

    if (results.length > 0) {
      const roomID = results[0].room_id;
      callback(null, roomID);
    } else {
      callback(null, null);
    }
  });
}
```

Code snippet A.4: Query to check if a classroom name really exists and get his ID

```
function getActiveClassAndUCID(roomName, callback) {
  const query = `SELECT class_id, id_uc FROM classes WHERE class_room =
      ? AND class_status = 'ativo'`;
  const values = [roomName];

  connection.query(query, values, function(error, results, fields) {
    if (error) {
      return callback(error);
    }

    if (results.length > 0) {
      const activeClassID = results[0].class_id;
      const UCID = results[0].id_uc;
      callback(null, activeClassID, UCID);
    } else {
      callback(null, null);
    }
  });
}
```

Code snippet A.5: Query to verify the active class in the classroom and retrieve both their ID and the class unit ID

```
function doesTableExists(roomID, tableNumber, callback) {
  const query = `SELECT room_table_id FROM room_tables WHERE room_id = ?
       AND table_number = ?`;
```

```
  connection.query(query, [roomID, tableNumber], function(error, results
      , fields) {
    if (error) {
      return callback(error);
    }

    if (results.length > 0) {
      const tableID = results[0].room_table_id;
      callback(null, tableID);
    } else {
      callback(null, null);
    }
  });
}
```

Code snippet A.6: Query to verify if the table really exists in a certain classroom

```
function isTableOccupied(tableID, callback) {
  const query = `SELECT tablet_status FROM room_tables WHERE
      room_table_id = ?`;

  connection.query(query, [tableID], function(error, results, fields) {
    if (error) {
      return callback(error);
    }

    if (results.length > 0) {
      const tableStatus = results[0].tablet_status;
      callback(null, tableStatus === 'occupied');
    } else {
      callback(null, false);
    }
  });
}
```

Code snippet A.7: Query to verify if a given table is occupied

```
function findStudentIDinUC(UCID, studentNumber, callback) {
  const query = `SELECT student_id FROM students WHERE id_UC = '${UCID}'
       AND student_number = '${studentNumber}'`;

  connection.query(query, function(error, results, fields) {
    if (error) {
      return callback(error);
    }
    if (results && results.length > 0) {
      const studentID = results[0].student_id;
      callback(null, studentID);
```

```
    } else {
      callback(null, null);
    }
  });
}
```

Code snippet A.8: Query to verify if the student is registered in the course unit

```
function addRowToTable (columnNames, values , callback ) {
    const sanitizedValues = values .map( value => connection . escape (
        valu ));

    const query =  INSERT  INTO student_logs (${columnNames.join(    ,
          ) ) VALUES (${sanitizedValues.map(v =>    ?    ).join(    ,
         )} )   ;
    connection.query(query, sanitizedValues, function(error, results,
        fields) {
        if (error) {
            return callback (error);
        }

        callback (null , results);
    });
}
```

Code snippet A.9: Query add a row to the student_logs table

```
function updateTableStatus(room_id, room_table, status, callback) {
  const query = 'UPDATE room_tables SET tablet_status = ? WHERE room_id
      = ? AND table_number = ?';
  const values = [status, room_id, room_table];

  connection.query(query, values, function(error, results, fields) {
    if (error) {
      return callback(error);
    }

    callback(null, results);
  });
}
```

Code snippet A.10: Query update a row in the *table_status* table

```
app.get('/:roomName/:table', handleGetRequest);

function handleGetRequest(req, res) {
  const roomName = req.params.roomName;
  const roomTable = req.params.table;

  doesRoomNameExist(roomName, (error, roomID) => {
    if (!roomID) {
```

```
      return res.render('response', { msg: 'This room does not exist!'
          });
    }
    getActiveClassAndUCID(roomName, (error, classID, UCID) => {
      classActiveID = classID;
      classUCID = UCID;

      if (classActiveID != classLastID) {
        handleClassChange();
      }
      if (!classActiveID) {
        return res.render('response', { msg: 'No class active in room '
            + roomName });
      }

      doesTableExists(roomID, roomTable, (error, tableID) => {
        if (!tableID) {
          return res.render('response', { msg: 'Seems like this table
              does not exist in this room!' });
        }

        isTableOccupied(tableID, (error, tableOccupied) => {
          if (tableOccupied) {
            res.render('response', { msg: 'It looks like this desk is
                already being occupied by another student!' });
          } else {
            res.render('index', { roomID, roomTable, tableID,
                classActiveID });
          }
        });
      });
    });
  });

function handleClassChange() {
    phoneIds = [];
    studentNumbers = [];
    classLastID = classActiveID;
}
```

Code snippet A.11: Function that handles the QR code verification request in the server side

```
let phoneID;

window.onload = function() {
  phoneID = localStorage.getItem('phoneID');
  if (!phoneID) {
    phoneID = uuidv4();
```

```
    localStorage.setItem('phoneID', phoneID);
  }

  fetch('/verify-phoneID', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({ phoneID })
  })
  .then(response => {
      response.text().then(html => {
        document.documentElement.innerHTML = html;
      });
    })
    .catch(error => {
      console.error('Error:', error);
    });
}
```

Code snippet A.12: Phone ID verification request by the client side

```
app.post('/verify-phoneID', handleVerifyPhoneID);

function handleVerifyPhoneID(req, res) {
  const { phoneID } = req.body;
  if (phoneIds.includes(phoneID)) {
    res.render('response', { msg: 'You already marked your attendance to
        this class!' });
  }
}
```

Code snippet A.13: Function that handle the phone ID verification request

```
function handleSubmit(event) {
  event.preventDefault();

  const studentNumberInput = document.getElementById('student-number');
  const studentNumber = studentNumberInput.value;
  const roomTable = document.getElementById('roomTable').innerHTML;
  const roomID = document.getElementById('roomID').innerHTML;
  const tableID = document.getElementById('tableID').innerHTML;
  const classActiveStart = document.getElementById('classActiveID').
      innerHTML;

  if (/^[a-zA-Z]\d+$/.test(studentNumber)) {
    const formattedStudentNumber = studentNumber.charAt(0).toUpperCase()
        + studentNumber.slice(1);

    fetch('/register-studentNumber', {
```

```
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({ studentNumber: formattedStudentNumber,
          phoneID, roomTable, roomID, tableID, classActiveStart })
    })
    .then(response => {
      response.text().then(html => {
        document.documentElement.innerHTML = html;
      });
    })
    .catch(error => {
      console.error('Error:', error);
    });
  } else {
    window.alert('The student number must start with a letter followed
        by one or more digits.');
  }
}
```

Code snippet A.14: Function that handle form submission and register student number request

```
app.post('/register-studentNumber', handleRegisterStudentNumber);

function handleRegisterStudentNumber(req, res) {
  const { studentNumber, phoneID, roomTable, roomID, tableID,
      classActiveStart } = req.body;

  if (studentNumbers.includes(studentNumber)) {
    return res.render('response', { msg: 'You already marked your
        attendance to this class!' });
  }

  if (classActiveStart != classActiveID) {
    return res.render('response', { msg: 'The class is not active
        anymore! You took too long!' });
  }

  findStudentIDinUC(classUCID, studentNumber, (error, studentID) => {
    if (!studentID) {
      return res.render('response', { msg: 'Seems like you are not
          registered in the UC!' });
    }

    isTableOccupied(tableID, (error, tableOccupied) => {
      if (tableOccupied) {
        return res.render('response', { msg: 'It looks like this desk is
```

```
                already being occupied by another student!' });
        }

        addRowToTable(['student_logs_id', 'student_id', 'class_id', '
            room_table'], [null, parseInt(studentID, 10), classActiveID,
            parseInt(roomTable, 10)], (error, results) => { });

        updateTableStatus(roomID, roomTable, 'occupied', (error, results)
            => { });

        phoneIds.push(phoneID);
        studentNumbers.push(studentNumber);

        res.render('response', { msg: 'Thanks for your registration!' });
      });
    });
}
```

Code snippet A.15: Function that handles the attendance registration in the serve side

Bibliography

[1] Purvaja Pradip Godbole, Anubhav Tomar, Savitri Bevinakoppa, Prazna Kundula, Arjun Aryal, and Bhanu Bhakta Bhusal. Iot based secured online attendance management system. In *2020 IEEE 7th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–6, 2020.

[2] M. Gopila and D. Prasad. Machine learning classifier model for attendance management system. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 1034–1039, 2020.

[3] Duy Dieu Nguyen, Xuan Huy Nguyen, The Tung Than, and Minh Son Nguyen. Automated attendance system in the classroom using artificial intelligence and internet of things technology. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 531–536, 2021.

[4] Gopinath Sittampalam and Nagulan Ratnarajah. Sams: An iot solution for attendance management in universities. In *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pages 251–256, 2019.

[5] M. Mythili, R. Sathya, N. Gayathri, M. Yogeshwaran, and S. Madhanbabu. A novel framework for smart classroom lecture attendance management system (lams) using iot. In *2022 3rd International Conference on Smart Electronics and Communication (ICOSEC)*, pages 519–525, 2022.

[6] Y. Keerthi, A. Sneha, P. Vani Manasa, and N. Neelima. Enhanced biometric attendance management system using iot cloud. In *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, pages 1–7, 2022.

[7] K Sangeetha, M Shobana, V S Nagul Pranav, S Darunya, K P Madhumitha, and M Nidharshini. Iot based identification and attendance monitoring system using design thinking framework. In *2023 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6, 2023.

[8] Zhumaniyaz Mamatnabiyev. Development of attendance monitoring system using iot technologies. In *2021 16th International Conference on Electronics Computer and Computation (ICECCO)*, pages 1–6, 2021.

[9] Vikas Yadav and G. P. Bhole. Cloud based smart attendance system for educational institutions. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pages 97–102, 2019.

[10] Nivetha A. S, Ramkumar S, S. Rajalakshmi, K. Aanandha Saravanan, and M. Revathy. Log system for mass crowd and live tracking using radio frequency based on iot. In *2022 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, pages 1–5, 2022.

[11] Les Pounder. How to Turn a Raspberry Pi Into a Wi-Fi Access Point, 2022. [Online] `https://www.tomshardware.com/how-to/raspberry-pi-access-point`.

[12] Tibbo Technology. Node.js Application as a Service. [Online] `https://tibbo.com/linux/nodejs/service-file.html`.

[13] baeldung. What Is the Difference Between GET and POST Methods? [Online] `https://www.baeldung.com/cs/http-get-vs-post`.