Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра комп'ютерних систем та технологій

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Розробка системи "розумні двері" з віддаленим керуванням»

«Development of a "smart door" system with remote control»

Виконав: здобувач денної форми навчання
спеціальності 123 – Комп'ютерна інженерія
Освітня програма «Комп'ютерна інженерія»

Поталєй Кирило Олегович
Керівник_____ст.викл.Берков Ю.М_____
(науковий ступінь, вчене звання, прізвище та ініціали, підпис)
Рецензент_____доц. Шугайло Ю.Б._____
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від _____2023 р.

Завідувач кафедри

_____ Юрій ГУНЧЕНКО
(підпис) (прізвище, ім'я)

Захищено на засіданні ЕК №_____
протокол № ____ від _____ р.
Оцінка_____/___ ___/_____
(за національною шкалою, шкалою ECTS, бали)
Голова ЕК

_____ Алла КОБОЗЄВА
(підпис) (прізвище, ім'я)

Одеса – 2023

АНОТАЦІЯ

У дипломній роботі розробляється тема «Розробка системи "розумні двері" з віддаленим керуванням».

Мета роботи – розробка проекту "розумні двері" у концепції інтернету речей, який має використовувати віддалений доступ. Розробка апаратної частини у вигляді мікроконтролеру з підтримкою підключення до бездротових мереж, а також розробка додатку за допомогою якого буде виконуватися взаємодія з мікроконтролером та його модулями.

В результаті проведеної роботи була розроблена система яка включає в себе апаратну частину у вигляді розробки проекту для мікроконтролеру з підтримкою зміни даних о мережі в реальному часі та взаємодією за додатком за допомогою HTTP запитів, у програмній частині був розроблений додаток , за допомогою якого користувачі матимуть змогу керувати модулями мікроконтролеру з віддалених пристроїв, також була розроблена невелика база даних яка зберігає в собі дані для функціонування веб додатку.

ABSTRACT

The diploma thesis focuses on the development of a "smart door" system with remote control.

The objective of this work is to develop a project for "smart doors" in the concept of the Internet of Things, utilizing remote access. The hardware part of the project involves the development of a microcontroller with wireless network connectivity support, as well as the creation of an application for interacting with the microcontroller and its modules.

As a result of the conducted work, a system has been developed, which includes the hardware part in the form of a project for the microcontroller with real-time data network updates and interaction with the application through HTTP requests. In the software part, an application has been developed that allows users to control the microcontroller modules from remote devices. Additionally, a small database has been created to store data for the functioning of the web application.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

HTTP - протокол передачі даних

JSON – JavaScript Objective Notation.

JWT – JSON Web Token.

REST (Representational State Transfer) – передача “самоописуваного” стану.

API (Application Programming Interface) – програмний інтерфейс додатку.

ВСТУП

Сучасний розвиток технологій і наукових досягнень змінює наше повсякденне життя, а також виробництво і навчання. На сьогоднішній день, з огляду на зростаючі вимоги до безпеки та зручності, існує зростаючий попит на системи дистанційного контролю доступу до різноманітних приміщень.

Системи дистанційного контролю дверми вже успішно використовуються в різних країнах світу. Такі системи дозволяють власникам контролювати доступ до своїх будинків через мобільні додатки, бездротові пульти або біометричні ідентифікатори, такі як відбитки пальців або розпізнавання обличчя. Такі рішення дозволяють забезпечити високий рівень безпеки і зручності, зменшити ризик зламу дверей або крадіжки ключів, а також забезпечити зручний контроль доступу для користувачів. Подібні системи можуть стати необхідною складовою частиною сучасних будинків, офісів та промислових будівель. Розробка і впровадження нових технологій в цій сфері сприятиме підвищенню безпеки життя і майна, а також досягненню енергоефективності в будівництві.

Метою розробки системи "розумні двері" з віддаленим керуванням є створення функціональної системи, яка забезпечує віддалений доступ та керування доступом через веб-додаток.

Для досягнення цієї мети необхідно вирішити ряд задач:

- розробка апаратної частини, включаючи мікроконтролер та модулі;
- розробка програмного забезпечення для зберігання і керування доступом, а також для взаємодії з веб-додатком;
- інтеграція системи з мережею інтернет та забезпечення безпеки передачі даних;
- тестування та валідація системи для забезпечення її надійності та функціональності;
- документування розробленої системи та створення інструкцій для користувачів.

1 ОГЛЯД АНАЛОГІЧНИХ СИСТЕМ

1.1 Опис предметного середовища

Розробка системи "розумні двері" з віддаленим керуванням спрямована на створення системи керування доступом до будівель або приміщень. Ця система має на меті поліпшити зручність та безпеку використання дверей, дозволяючи користувачам віддалено контролювати доступ через смартфон або веб-додаток.

Розумні двері з віддаленим керуванням є актуальним рішенням в галузі "інтернету речей" (IoT) і "розумного будинку". Застосування таких систем може бути корисними у різних сферах, включаючи житлові будинки, офісні приміщення, готелі, апартаменти, комерційні будівлі та інші об'єкти з обмеженим доступом.

Розробка системи "розумні двері" з віддаленим керуванням є актуальною через зростання популярності розумних пристроїв та розумних будинків. Вона забезпечує зручність користувачам, дозволяючи їм керувати доступом до своїх приміщень з будь-якого місця та в будь-який час. Така система може забезпечити високий рівень безпеки, адміністрування та моніторингу доступу.

1.2 Огляд наявних аналогів

У цьому підрозділі розглянемо опис наявних аналогів.

1) August Smart Lock Pro: Цей аналог пропонує систему "розумних дверей" з віддаленим керуванням через мобільний додаток. Він має можливості автоматичного розблокування дверей при наближенні, можливість надання обмеженого доступу іншим користувачам, а також інтеграцію з голосовими асистентами [1].

Переваги:

– легка установка;

– можливість віддаленого керування через мобільний додаток;

– можливість інтеграції з іншими розумними пристроями.

Недоліки:

– висока вартість;

– обмежені можливості налаштування доступу;

– можуть виникати проблеми з підключенням до мережі.

2) Yale Assure Lock SL: Ця система розумних дверей також пропонує віддалене керування через смартфон. Вона має механізм автоматичного замикання дверей, можливість створення тимчасових кодів доступу для гостей та інтеграцію з популярними платформами "розумних будинків" [2].

Переваги:

– елегантний дизайн;

– можливість віддаленого керування через мобільний додаток;

– сумісність зі стандартними дверними замками.

Недоліки:

– вимагає заміни існуючого дверного замка;

– можливі проблеми зі сумісністю з деякими дверними конструкціями.

3) Schlage Connect Smart Deadbolt: Цей аналог системи "розумні двері" також дозволяє віддалене керування за допомогою мобільного додатка. Він має можливості контролю доступу для сім'ї та гостей, відстеження історії входів і виходів, а також можливість інтеграції з іншими "розумними" пристроями [3].

Переваги:

– сильний механізм блокування;

– можливість віддаленого керування через мобільний додаток;

– можливість інтеграції з системами "розумного будинку".

Недоліки:

– вимагає заміни існуючого дверного замка;

– деякі проблеми зі сумісністю зі старішими моделями.

Ці аналоги є прикладами комерційних рішень, які вже доступні на ринку. Вони надають різні функції та можливості віддаленого керування системою "розумні двері". Однак, розробка власної системи може дозволити більшу гнучкість, зменшення ціни та налаштованість під потреби конкретного користувача.

1.3 Основні задачі системи

До основних задач системи можна віднести наступні:

– засоби взаємодії з дверми;

– надання переліку доступних модулів керування;

– зміна даних про мережу в реальному часі.

1.3.1 Типи користувачів системи

У даній темі «Система "розумні двері" з віддаленим керуванням» виділяються такі категорії користувачів:

1) Клієнт – має право на перегляд інформації про доступні йому плати, взаємодію з дверми через додаток;

2) Адміністратор – має право на перегляд інформації про усі плати, надання прав доступу до плат користувачам, обмеження прав доступа для користувачів, перегляд статистики користувачів.

1.3.2 Список задач користувачів

Пропонована система керування дверима передбачає декілька категорій користувачів: адміністратор та клієнт. Кожна з них має відповідні задачі які перелічені у таблиці 1.1.

Таблиця 1.1 – Список задач користувачів «Система "розумні двері" з віддаленим керуванням»

Номер	Задача	Вхідні дані	Вихідні дані
	Адміністратор		
А1	Перегляд плат	Відсутня	Доступні плати
А2	Надання доступу користувачеві	Користувач ,серійний номер плати	Відсутня
А3	Зміна прав доступу для користувачів	Користувач , нові права доступу	Відсутня
А4	Перегляд статистики	Номер плати	Статистика про дії користувачів
Клієнт			
К1	Реєстрація	Логін, пароль	Новий обліковий запис
К2	Перегляд доступних йому плат	Відсутня	Доступні плати
К3	Взаємодія з дверми	Номер двері	Відкриття або закриття обраної двері

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Вибір апаратної компонентів

Для виконання мети дипломної роботи потрібно було обрати мікроконтролер з підтримкою віддаленого доступу та модулі які потрібні для взаємодії з різноманітними замками.

Для виконання цих потреб було обрано мікроконтролер NodeMCU v3 (рис. 2.1) [4]:

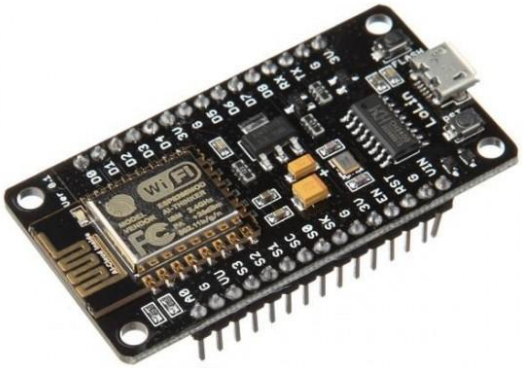

Рисунок 2.1 – Мікроконтролер NodeMCU v3

Мікроконтролер – це мозок апаратної частини проекту , через який ми будемо взаємодіяти з модулями. На ньому буде запускатися веб сервер , який буде приймати HTTP запити з веб серверу додатку.

Переваги:

– оснований на мікроконтролері esp8266, який має вбудований wi-fi модуль, що дозволяє бездротове підключення до мережі інтернет;

– підтримує популярні мови програмування, такі як arduino c і мову lua;

– має вбудовану пам'ять для зберігання програмного коду і даних;

– має широкий спектр введення/виведення (gpio) для підключення до різних сенсорів та пристроїв;

– легкий у використанні та програмуванні.

Аналоги:

1) Arduino Uno: Популярна платформа розробки з широким спектром введення/виведення та великою спільнотою користувачів;

2) Raspberry Pi: Більш потужний одноплатний комп'ютер з підтримкою ОС Linux, має багато можливостей для розробки IoT-проектів.

Для відображення поточної інформації та налаштувань застосовано LCD-дисплей 16x2 з I^2C шиною (рис. 2.2)[5]:

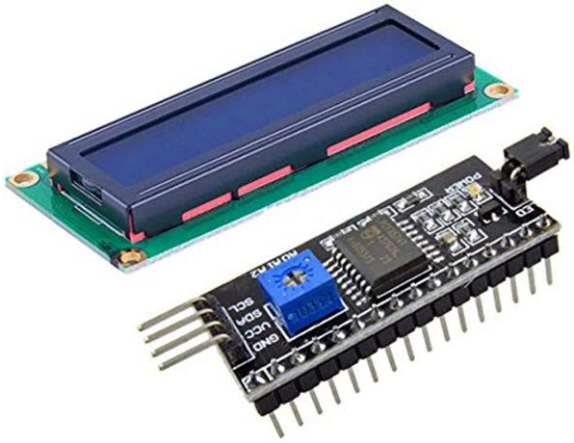

Рисунок 2.2 – Дисплей та шина I^2C

Дисплей – це модуль який буде виступати у ролі фізичної панелі інформації, яка буде відображати статус мікроконтролеру.

Переваги:

– має два рядки по 16 символів для відображення тексту або символів;

– використовує I^2C шину для комунікації, що дозволяє зменшити кількість пінив, необхідних для підключення;

– забезпечує простий інтерфейс для відображення інформації, такої як стан системи, значення сенсорів тощо.

Аналоги:

1) HD44780 LCD: Класичний дисплей LCD зі стандартним 16x2 розташуванням символів, вимагає більшої кількості пінив для підключення;

2) OLED дисплеї: Менші, компактні дисплеї з OLED технологією, які забезпечують більшу яскравість та кольорову гаму.

Два модулі реле високого рівня (рис. 2.3) [6]:

Рисунок 2.3 – Модуль реле

Реле – це модуль який буде керуватися мікроконтролером через веб додаток. Реле використовується для вмикання та вимикання дверей або замків.

Переваги:

– забезпечують високий рівень комутації електричного струму, що дозволяє керувати потужними пристроями.

– легкі у використанні та підключенні до мікроконтролерів.

– мають низьку вартість та доступні для багатьох проектів.

Аналоги:

1) Модулі реле низького рівня: Забезпечують комутацію меншого електричного струму, підходять для менш потужних пристроїв;

2) Solid State Relays (SSR): Використовують напівпровідникові компоненти замість механічних реле, що забезпечує швидку комутацію та більшу надійність.

2.2 Вибір архітектури та шаблону програмування

При проектуванні архітектури проекту була обрана триланкова архітектура клієнт-сервер та архітектурний стиль REST.

REST – це стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, який зазвичай використовується для побудови веб-служб. Системи, що підтримують REST, називаються RESTful-системами (рис. 2.4) [7].

У загальному випадку REST є дуже простим інтерфейсом управління інформацією без використання якихось додаткових внутрішніх шарів. Кожна одиниця інформації однозначно визначається глобальним ідентифікатором, таким, як URL. Кожна URL, у свою чергу, має строго заданий формат. В ідеальному випадку одна програма-сервер може виконувати запити від безлічі програм-клієнтів, тому її слід розміщувати на спеціально виділеній обчислювальній машині, налаштованій особливим чином, як правило, спільно з іншими програмами-серверами, тому продуктивність цієї машини максимізується.

Через особливу роль такої машини в мережі, специфіки її обладнання та програмного забезпечення, її також називають сервером, а машини, які виконують клієнтські програми, відповідно, клієнтами.

Клієнт (рівень клієнта) – зазвичай це комп’ютер чи додаток, який звертається до серверу та запитує необхідні для роботи користувача дані, зазвичай, має графічний інтерфейс, який допомагає користувачеві зручно та легко працювати з отриманою інформацією. Окрім цього клієнт не тільки отримує інформацію, але і оброблює її – наприклад, проводить обчислення та перетворення даних.

Цей рівень не повинен мати прямих зав’язків з базою даних (за вимогами безпеки і масштабованості), та зберігати стан додатку (за вимогами надійності).

Сервер (рівень даних) надає функцію або послугу одному або кільком клієнтам, які ініціюють запити. Сервери класифікуються по послугам, які вони надають. Наприклад, веб-сервер обслуговує веб-сторінки, файловий сервер обслуговує комп’ютерні файли, сервер бази даних надає доступ до баз даних.

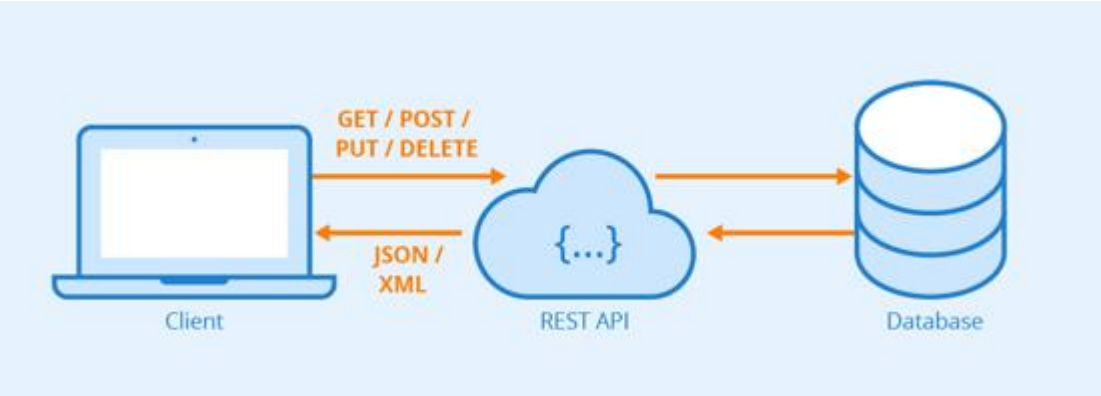

Рисунок 2.4 – Схема архітектури REST

Ґрунтуючись на архітектурі, необхідно також обрати шаблон проектування додатку. Підходящим можна назвати шаблон MVC (рис. 2.5). Model-View-Controller – схема поділу даних програми, призначених для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, представлення і контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно [8].

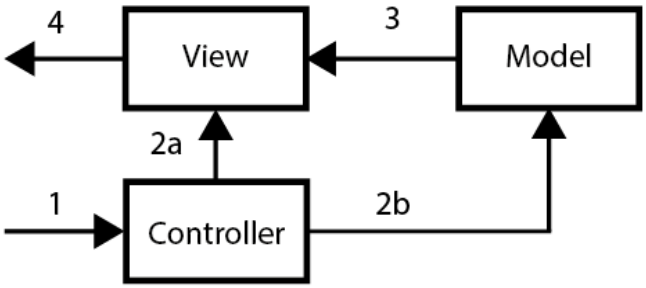

Рисунок 2.5 – Схема шаблону MVC

Модель (Model) надає дані і методи роботи з ними: запити до бази даних, перевірка на коректність. Модель не залежить від представлення і контролера, просто надаючи доступ до даних і управління ними. За рахунок незалежності від візуального представлення, може мати кілька різних представлень для однієї «моделі».

Представлення (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі. Отримує необхідні дані з моделі і відправляє їх користувачеві. Представлення не виконує обчислень з даними користувача.

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін. Цей елемент забезпечує «зв’язок» між користувачем і системою. Контролює і направляє дані від користувача до системи і навпаки. Використовує модель і представлення для реалізації необхідного дії.

Покроковий принцип роботи програми при використанні шаблону MVC:

1) Контролер отримує наступний запит від користувача;

2) Далі, залежно від внутрішньої логіки:

а) Формується уявлення якоїсь сторінки;

б) Або викликаються методи моделі;

3) Модель повідомляє про зміни;

4) Представлення оновлюється (якщо в ланцюжку була задіяна модель) та відображається користувачеві.

2.3 Інформаційна модель системи

В базі даних системи "розумні двері" з віддаленим керуванням слід виділити наступні сутності:

– плати – містить інформацію про плати, та їх пін-коди;

– користувачі – містить інформацію про логін, та дату останньої авторизація та взаємодії з дверми.

– зв’язки – містить інформацію про доступ користувачів до плат та обмеження доступу.

Детальніше сутності та їх властивості з описом обмежень, що потрібні для розв’язання поставлених задач, наведено в таблиці у додатку А.

Між сутностями у базі даних наявні один тип зав’язків-«багато-до-багатьох». Зв'язок «багато-до-багатьох» наявний між платами та користувачами :

– користувачі та плати. Кожному користувачу може відповідати декілька плат (або жоден), та кожній платі може відповідати декілька користувачів(або жоден).Для формалізації зв'язку у таблиці relations є зовнішній ключ, який посилається на первинний ключ таблиці boards , та є інший зовнішній ключ який в свою чергу посилається на первинний ключ таблиці users.

2.4 Модель програмного застосунку

Користувачі та адміністратори користуються єдиним додатком. При авторизації у ролі одного з користувачів, додаток завантажує необхідні елементи інтерфейсу, в залежності від ролі користувача.

Підхід розміщення функціоналу залежно від ролі лише на одній сторінці зменшує об’єм написаного коду, лаконічно розміщує інформацію на екрані (без зайвого пустого простору) та спрощує навігацію для рядового користувача .

Дотримання шаблону MVC в даному випадку можливе без створювання додаткових класів. Система гармонічно розподілена на директорії та файли, кожен з яких виконує лише одну певну дію. Клієнт з UI посилає запит контролеру (який збережений в однойменному файлі директорії Controllers), а контролер передає інформацію наступній ланці системи, де вже виконуються операції із БД (Model).

2.5 Вибір програмного забезпечення

Додаток розроблен у вигляді веб-додатку. Для роботи з базою даних обрано СУБД PostgreSQL, для розробки програмного додатку

використовується платформа Node.js із фреймворком Express.js (серверна частина) та бібліотека React.js із використанням готових бібліотечних компонентів react-bootstrap (клієнтська частина) та для розробки програмного коду для мікроконтролера була обрана мова Arduino C.

Сьогодні існує багато систем керування базами даних (MySQL, Oracle, MariaDB, PostgreSQL тощо) [9]. Серед них у проекті обрано PostgreSQL з наступних причин:

– реалізує реляційну модель даних, яка є зрозумілою для кінцевого користувача;

– має гнучкий механізм управління правами користувачів БД (ролі);

– є можливість створювати схеми для відокремлення даних користувачів;

– підтримує мову plpgSQL як розширення стандарту SQL для створення ефективних збережених процедур;

– є безкоштовним ПЗ із відкритим кодом.

Node.js або Node – програмна платформа, заснована на движку V8, Node.js додає можливість передавати дані з серверу на клієнт, та с клієнта на сервер інформаційної системи.

Express.js – фреймворк web-додатків для Node.js, реалізований як вільне та відкрите програмне забезпечення під ліцензією MIT. Він спроектований для створення веб-додатків та API [10]. Цей фреймворк використаний для проміжної обробки запитів для Node.js.

Зв’язка Node.js+React нині є одною найпопулярніших для створення сучасних крупних web-додатків. Про переваги саме цієї комбінації говорить статистика. JavaScript (у контексті Node.js) станом на 2022 рік [11].

2.6 Модель системи

Основним принципом взаємодії апаратної частини та додатку є трьохланкова архітектура клієнт-сервер (рис. 2.6).

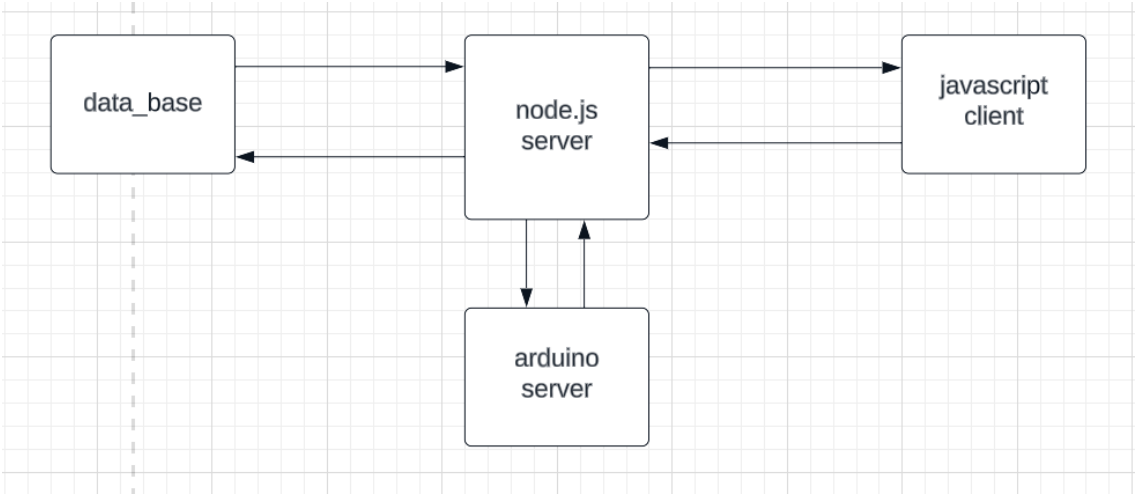

Рисунок 2.6 – Діаграма архітектури системи

Клієнт – це графічний інтерфейс, який звертається до серверу веб-додатка та запитує необхідні для роботи користувача дані. Коли дані для взаємодії отримані, запит з переліку доступних - надсилається до серверу веб-додатку.

Сервер(node.js) – надає функцію або послугу одному або кільком клієнтам, та надсилає запит до бази даних або HTTP запит до серверу мікроконтролеру.

Сервер(мікроконтролер) – приймає HTTP запити від сервера веб-додатку на взаємодію з модулями та за межами наведеної вище системи дозволяє змінювати дані о мережі у реальному часі.

3 СТВОРЕННЯ ПРОТОТИПУ СИСТЕМИ

3.1 Реалізація функціонування апаратної частини

У цьому підрозділі описано створення та функціонування апаратної частини системи.

Була створена фізична схема (рис. 3.1), та був написаний програмний код.

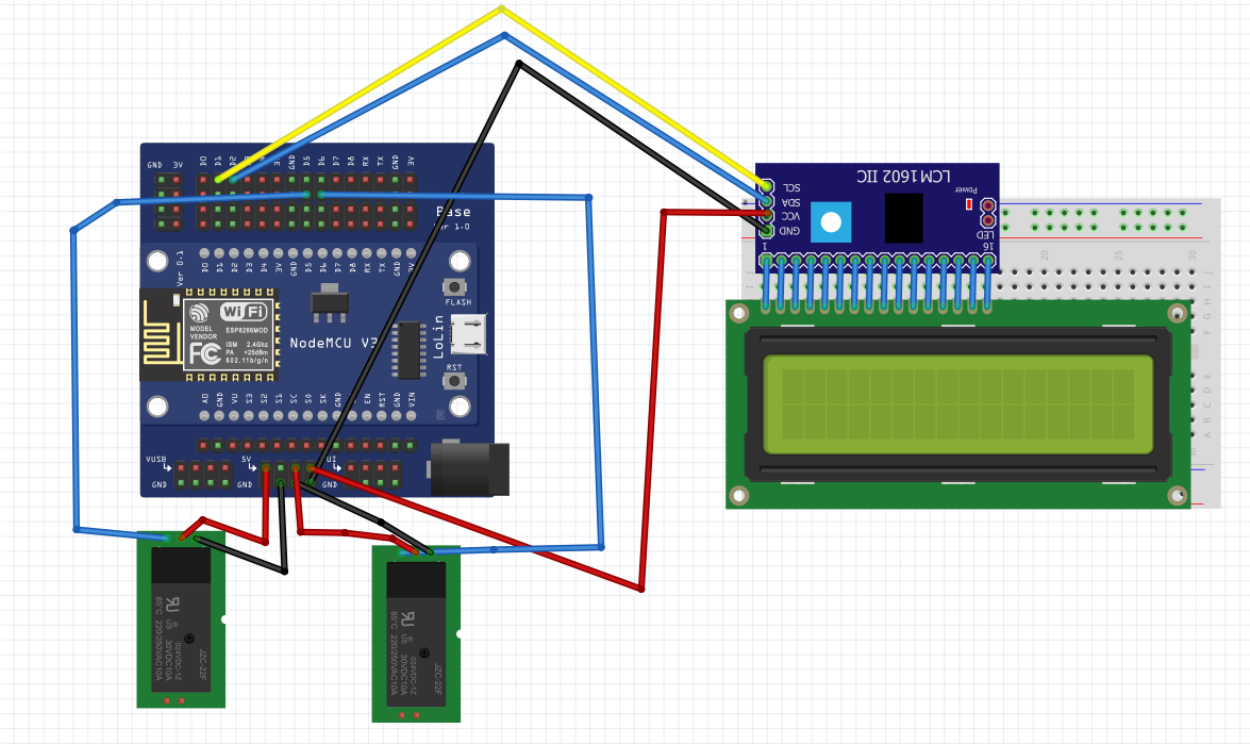

Рисунок - 3.1 – Електрична схема

Основною проблемою в вирішувані задач , була зміна даних о мережі в реальному часі. За замовчуванням у всіх мікроконтролерах або у модулях що підтримують віддалений доступ, інформація о мережі прописується у програмному коді. Для зміни цих даних потрібно перезавантажувати програмний код у контролер зі зміненими даними о мережі. Для системи

«Система "розумні двері" з віддаленим керуванням» це є неприпустимим. Тому потрібно розглянути це питання детальніше.

Рішенням цієї проблеми став запис даних о мережі в EEPROM (Electrically Erasable Programmable Read-Only Memory) – енергонезалежну пам’ять мікроконтролеру. Програмний код який записує дані о мережі в перший раз , або при зміні даних перезаписує дані о мережі. При відсутності підключення до мережі запускається точка доступу на якій запускається веб-сервер , до якої повинен підключитися користувач та ввести нові дані о мережі (лістинг 3.1).

```
  if (request->hasArg("ssid") && request->hasArg("pass")) {
    String qsid = request->arg("ssid");
    String qpass = request->arg("pass");

    Serial.println("clearing eeprom");
    for (int i = 0; i < 96; ++i) {
      EEPROM.write(i, 0);
    }
    for (int i = 0; i < qsid.length(); ++i)
    {
      EEPROM.write(i, qsid[i]);
    }
    Serial.println("writing eeprom pass:");
    for (int i = 0; i < qpass.length(); ++i)
    {
      EEPROM.write(32 + i, qpass[i]);
    }
    EEPROM.commit();
String content = "{\"Success\":\"saved to eeprom... reset to
boot into new wifi\"}";
    int statusCode = 200;
    ESP.reset();
    request->send(statusCode, "application/json", content);
  } else {
    String content = "{\"Error\":\"404 not found\"}";
    int statusCode = 404;
    Serial.println("Sending 404");
    request->send(statusCode, "application/json", content);
  }
    });
```

Лістинг 3.1 – Функція запису даних о мережі

Одразу після виконання запису нових даних о мережі , виконується зчитування з енергонезалежної пам’яті(лістинг 3.2). Та мікроконтролер підключається до мережі.

```
for (int i = 0; i < 32; ++i) {
    esid += char(EEPROM.read(i));
  }
  Serial.println();
  Serial.print("SSID: ");
  Serial.println(esid);

  Serial.println("Reading EEPROM password");
  String epass = "";
  for (int i = 32; i < 96; ++i) {
    epass += char(EEPROM.read(i));
  }
```

Лістинг 3.2 – Функція зчитування даних о мережі

3.2 Реалізація бази даних

У цьому підрозділі описано створення основних об’єктів бази даних. Як приклад наведено програмний код створення таблиці users (лістинг 3.3), таблиці relations (лістинг 3.4) та таблиці boards (лістинг 3.5).

У таблиці users зберігаються дані про користувачів, relations зберігає дані про плати які доступні для деякого користувача, boards зберігає дані про серійні номера плат.

```
CREATE TABLE users (
    client_id serial PRIMARY KEY,
     login varchar (15) unique not null,
     password varchar (300) not null,
     last_auth_date timestamp,
     last_action_date timestamp
);
```

Лістинг 3.3 – Код створення таблиці users

```
CREATE TABLE relations (
    relation_id serial PRIMARY KEY,
     client_id int not null,
     board_id int not null,
     relays varchar (5) CHECK (relays IN ('0','1','12')),
     counter smallint NOT NULL DEFAULT 0,
     CONSTRAINT fk_board
    FOREIGN KEY(board_id)
     REFERENCES boards(board_id),
     CONSTRAINT fk_client
    FOREIGN KEY(client_id)
     REFERENCES users(client_id),
     Constraint composite_key_2 UNIQUE (client_id, board_id)
);
```

Лістинг 3.4 – Код створення таблиці relations

```
CREATE TABLE boards (
    board_id serial PRIMARY KEY,
     serial_number varchar (10) unique not null check
(serial_number ~ '^[A-Z]{4}\d{3}$')
);
```

Лістинг 3.5 – Код створення таблиці boards

Розглянемо детальніше створення таблиці зав’язків у базі даних. Створення таблиці відбувається за допомогою команди CREATE TABLE, далі вказується ім'я таблиці. Кожному полю таблиці зіставляються тип даних обраної СУБД (PostgreSQL) і обмеження цілісності. Поле relation_id є первинним ключем (PRIMARY KEY) і має тип smallserial (цей тип не є справжнім типом даних, а являє собою зручний спосіб створення унікального ідентифікатора шляхом збільшення попереднього значення в даному стовпці на вказане значення, за замовчуванням на 1). Поле client_id є зовнішнім ключем (FOREIGN KEY). NOT NULL при визначенні полів таблиці означає, що вони не можуть містити порожні значення. Запис DEFAULT означає, що дані у відповідному полі за замовчуванням приймають наступні після слова DEFAULT значення. CHECK перевіряє чи виконані умови введених значень. SIMILAR TO дозволяє формувати «шаблон», який у поєднанні з CHECK перевіряє, чи відповідають значення до шаблонних.

Далі наведено схему бази даних, що ілюструє сутності та зв’язки між ними (рис. 3.2).

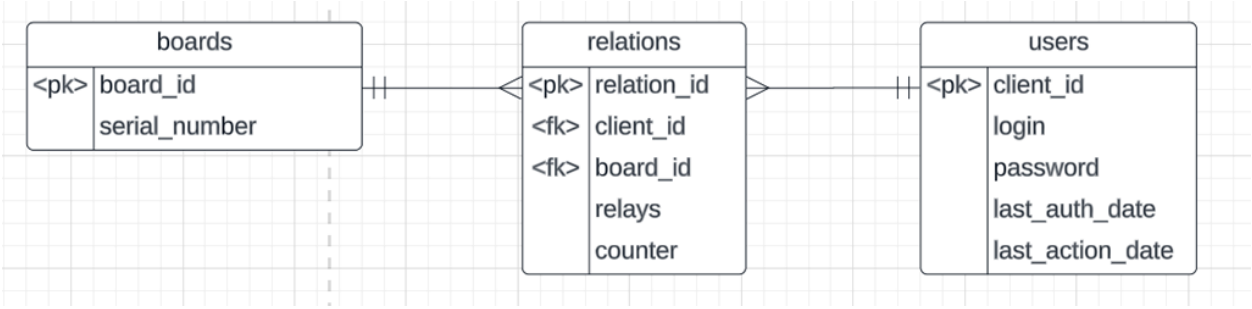

Рисунок 3.2 - ER-діаграма бази даних

Далі представлені запити до бази даних, які виконуються у додатку для отримання чи редагування таблиць. Необхідно зазначити, що усі ідентифікатори сховані від користувачів та підставляються системою автоматично.

Запити для задач адміністратора:

Для задачі A1 використовується запит

```
(`select serial_number from boards`)
```

Лістинг – 3.6

Отримаються дані про всі плати.

Для задачі A2 використовується запит

```
(`insert into relations values (default,$1,$2,'12')
`,[client_id,board])
```

Лістинг – 3.7

Додаються дані про доступ деякого користувача до деякої плати, до сутності relations.

Для задачі А3 використовується запит

```
`UPDATE relations SET relays = $1 FROM boards WHERE
relations.board_id = boards.board_id and serial_number=$3 and
client_id=$2 `,[relays,client_id,board_id])
```

Лістинг – 3.8

Отримаються дані про плату, права доступу, унікальний ідентифікатор користувача, та змінюється поле у сутності relations.

Для задачі А4 використовується запит

```
(`select users.client_id,login,counter from  boards inner
join relations on boards.board_id=relations.board_id join users
on relations.client_id=users.client_id where
 boards.serial_number=$1 and relations.client_id =$2
`,[serialNumber,user_id])
```

Лістинг – 3.9

Отримаються дані про плату, та кількість використань для кожного користувача плати з сутності relations.

Запити для задач користувача:

Для задачі К1 використовується запит

```
(`insert into users values
(default,$1,$2,current_timestamp,null)`, [login, hashPassword])
```

Лістинг – 3.10

Отримаються дані користувача, та вони додаються до сутності users.

Для задачі К2 використовується запит

```
(`select boards.board_id ,serial_number,relays from  boards
inner join relations on boards.board_id=relations.board_id join
users on relations.client_id=users.client_id where users.login =
$1 `,[login]))
```

Лістинг – 3.11

Отримаються дані про доступні користувачеві плати.

Для задачі КЗ використовується запит

```
"GET", "http://192.168.1.239:80/"+sr+"/1/"+ (ledState ? "off" : "on"), true
```

Лістинг – 3.12

Надсилається запит до плати.

3.3 Програмна реалізацію інтерфейсу

Щоб відповідати шаблону MVC, для основних сутностей у системі є свої окремі набори файлів, які належать до моделі, контролеру та класів графічного інтерфейсу, що беруть дані з відповідної таблиці. Примітка: файли в реалізації додатку називаються невідповідно до їхнього функціоналу – тут Controls - контролери, а Controllers – як моделі шаблону MVC.

Кожна модель реагує на команди контролерів в яких йде звернення до цієї моделі. Так, як кожен користувач системи має свій окремий простір, контролери поділено між ролями. Список контролерів:

– GuestAPI. Забезпечує зв’язок із GuestController, де виконується реєстрація та аутентифікація;

– AdminAPI. Забезпечує зв’язок із AdminController, де виконуються задачі адміністраторів;

– UserAPI. Забезпечує зв’язок із UserController, де виконуються задачі користувачів;

Рівень View шаблону MVC представлено у вигляді сторінок чи так званих панелей, які звертаються до контролера для завантаження чи зміни даних.

Як приклад, наведемо процес відображення на екран списку усіх плат із бази даних:

1) рівень Model (лістинг 3.13)

```
async board_for_admin (req, res) {
  const boards = (await admin.query(`select serial_number
from boards`)).rows
  res.json(boards)
}
```

Лістинг 3.13 – Запит до бази даних

2) 2) рівень Controller (лістинг 3.14)

```
export const board_for_admin = async () => {
const{data}=await$authHost.get('api/admin/boards_for_admin')
    return data;}
```

Лістинг 3.14 – Отримання даних на клієнтську частину

3) рівень View (лістинг 3.15)

```
const boards_func = async () => {
  setBoards(await board_for_admin());
};
```

Лістинг 3.15 – Отримання даних на сторінку

3.4 Безпека системи

Безпека на рівні БД забезпечується шляхом створення ролей і наділення їх відповідними привілеями. В системі реалізовано три ролі: Адміністратор, Користувач та Гість. У таблиці 3.2 наведені привілеї ролей на таблиці та елементи бази даних. При цьому використані наступні позначення:

– I (Insert)– додавання нової інформації до таблиці;
– S (Select) – перегляд вмісту таблиці;
– U (Update) – редагування існуючої інформації;
– D (Delete) – видалення інформації з таблиці;
– (Execute) – право на виклик збереженої процедури.

Як можна побачити с позначень, жодна роль не може створювати чи видаляти самі таблиці, а лише взаємодіяти з даними в них. Таким чином жоден з користувачів не може змінити структуру бази даних та вплинути на її працездатність.

Таблиця 3.2 – Права доступу користувачів до бази даних

Об’єкти БД	Ролі		
	Адміністратор	Користувач	Гість
Таблиці			
users	S		I
boards		S	
relations	ISU	S	
Представлення			
Users_view			S

Створення ролей і надання їм привілеїв відповідно до наведеної вище таблиці здійснюється за допомогою наступних SQL-запитів:

1. Користувач

```
CREATE ROLE client WITH LOGIN PASSWORD client;
GRANT USAGE ON SCHEMA public TO client;
GRANT SELECT ON boards,relations TO client;
```

Лістинг 3.16 – Надання привілеїв користувачу

2. Гість

```
CREATE ROLE guest WITH LOGIN PASSWORD ‘guest’;
GRANT USAGE ON SCHEMA public TO guest;
GRANT INSERT ON users TO guest;
GRANT SELECT ON users_view TO guest;
```

Лістинг 3.17 – Надання привілеїв гостю

3. Адміністратор

```
CREATE ROLE admin WITH LOGIN PASSWORD ‘admin’;
GRANT USAGE ON SCHEMA public TO admin;
GRANT SELECT ON users TO admin;
GRANT INSERT, SELECT, UPDATE ON relatioins TO admin;
```

Лістинг 3.18 – Надання привілеїв адміністратору

Безпека на рівні клієнта додатку забезпечується хешування паролів, коли користувач надсилає запит на реєстрацію його пароль хешується алгоритмом sha256 та записується у відповідну до типу користувача.

```
Const hashPassword = await bcrypt.hash(password, 5)
```

Лістинг 3.19 – Функція хешування паролю

Коли користувач надсилає запит на авторизацію, функція перевіряє правильність введеного паролю, якщо він правильний користувач авторизується.

```
PassCorrect =bcrypt.compareSync(password,dbpassword)
```

Лістинг 3.20 – Функція порівняння паролю з хешованим паролем

Коли користувач успішно авторизувався генерується JWT терміном дії 24 години. Параметрами JWT є id, login.

```
const generateJwt = (id, login) => {
    return jwt.sign(
        {id, login, role},
        process.env.SECRET_KEY,
        {expiresIn: '24h'})}
```

Лістинг 3.21 – Функція генерації токену

Безпека на рівні мікроконтролеру забезпечується фільтруванням IP-адрес(лістинг 3.22). Якщо запит на взаємодію приходить мінуючи веб-додаток , в доступі буде відмовлено. Запити будуть виконані лише якщо запит відправляється з серверу веб-додатку.

```
IPAddress clientIP = request->client()->remoteIP();
  if (clientIP == serverIP ) {
    request->send(200, "text/plain", "Hello from server 1");
    digitalWrite(relPin_2, HIGH);
  } else {
    request->send(403, "text/plain", "REFUSED");
  }
```

Лістинг 3.22 – Функція отримання даних о адреси з якої надсилається запит

3.5 Інструкція користувача

3.5.1 Підключення пристрою до бездротової мережі

Після першого підключення мікроконтролера та модулів до мережі на дисплеї або відсутній wifi мережі до якої раніше відбувалося підключення, користувач побачить що немає підключення до мережі wifi , та точка доступу та веб сервер запущено (рис. 3.3).

Рисунок 3.3 – Відображення статусів на дисплеї

Далі потрібно перейти до налаштувань wifi на пристрої що підтримує підключення до wifi мереж, увімкнути wifi та підключитися до точки доступу (рис. 3.4-3.5).

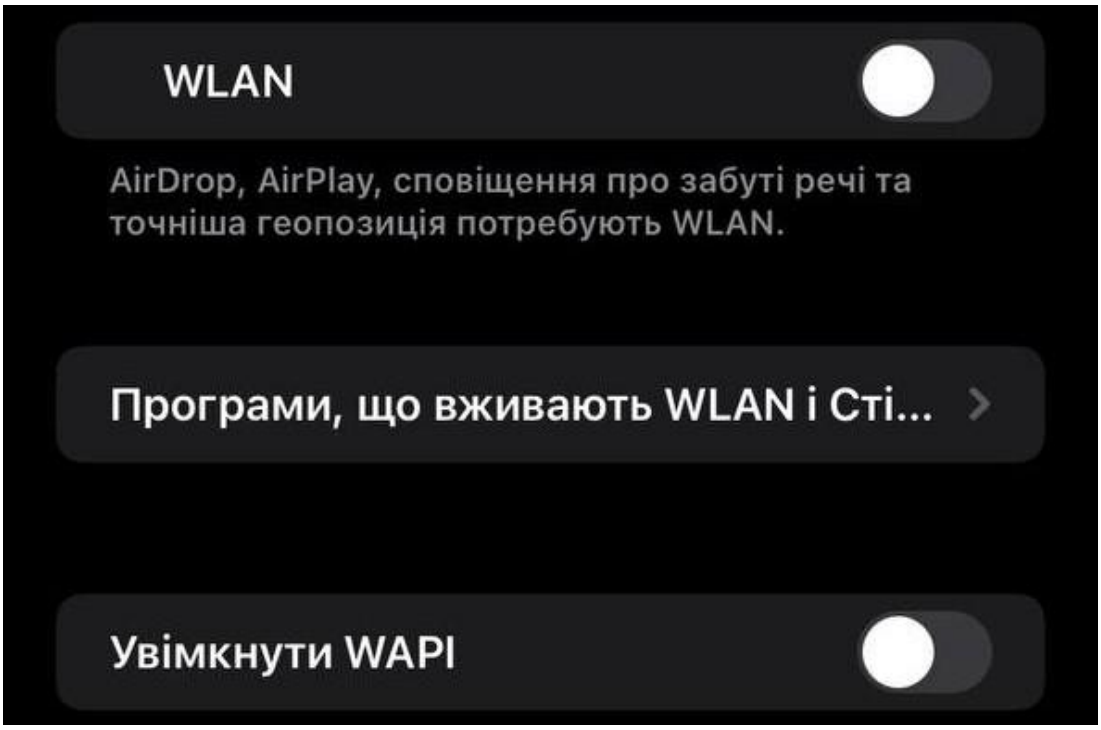

Рисунок 3.4 – Налаштування wifi на телефоні(вимкнено)

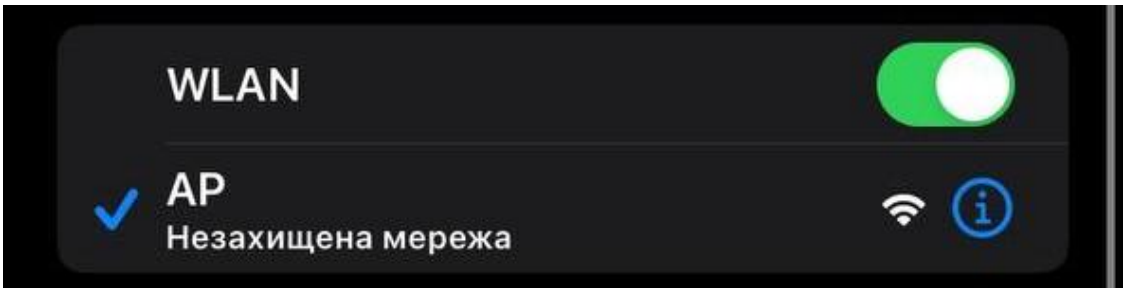

Рисунок 3.5 – Підключення до точки доступу

Далі перейти у браузер та набрати адресу веб-серверу – 192.168.4.1:81. Перед користувачем з’явиться сторінка на якій потрібно буде ввести дані про доступну мережу (рис. 3.6).

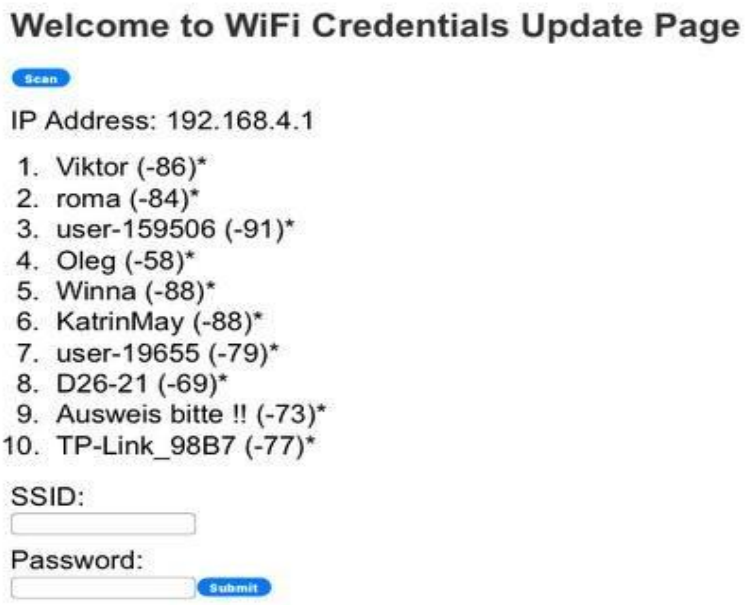

Рисунок 3.6 – Інтерфейс зміни даних о мережі при підключені до точки доступа

У поля SSID та Password потрібно ввести назву та пароль до мережі wifi. Після натискання кнопки «submit», дані про мережу запишуться до EEPROM, та мікроконтролер її зчитає. Після успішного підключення до

мережі дані на дисплеї зміняться – є доступ до wifi мережі , точка доступу та веб сервер відключені (рис. 3.7).

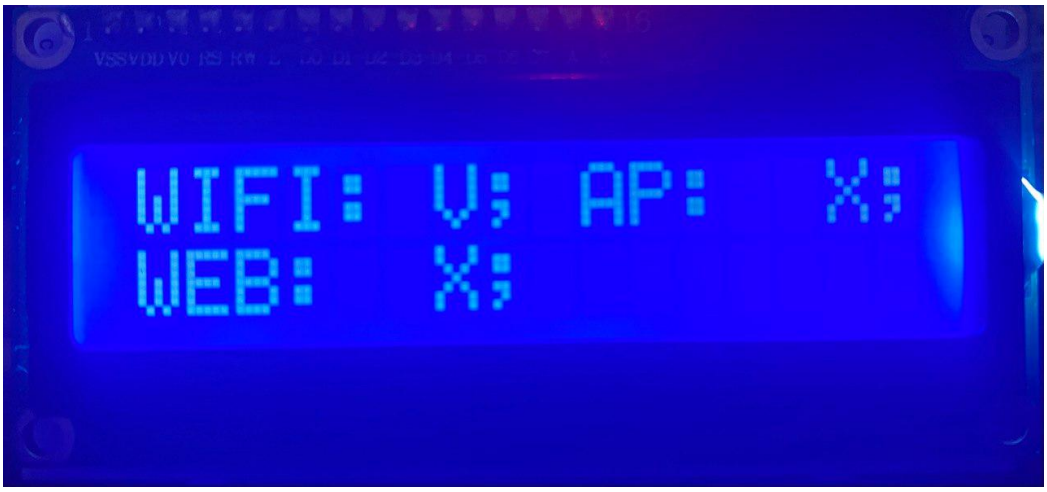

Рисунок 3.7 – Відображення статусів на дисплеї

3.5.2 Користування веб додатком для клієнта з комп'ютеру

Після підключення до веб-адреси додатку - http://212.178.5.97:3000, користувач бачить сторінку реєстрації (рис. 3.8).

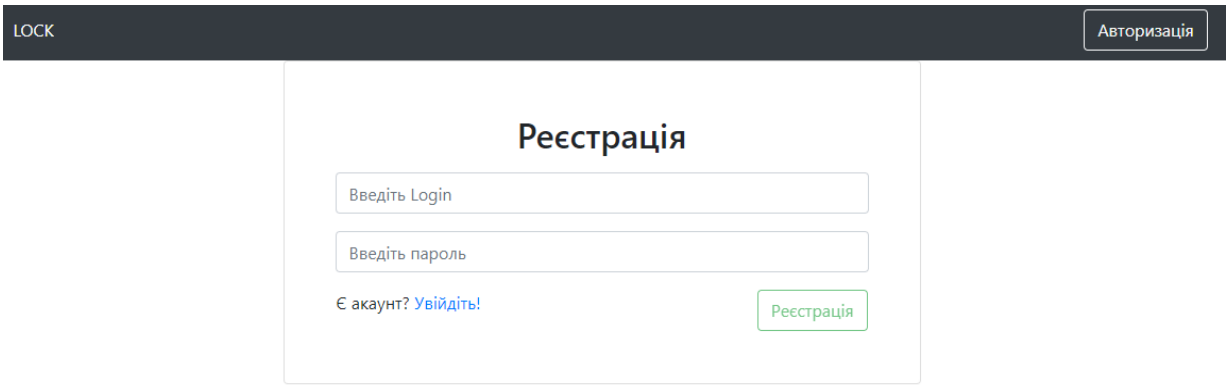

Рисунок 3.8 – Вікно реєстрації

При введені некоректний даних інтерфейс повідомить користувача про помилки (рис. 3.9).

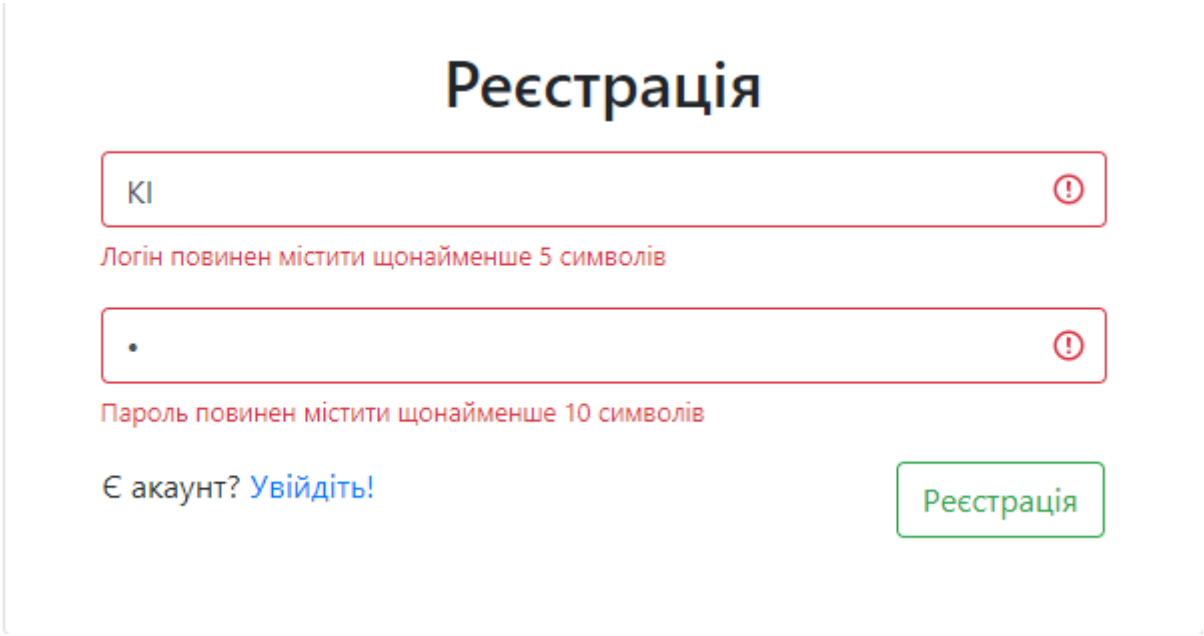

Рисунок 3.9 – Вікно реєстрації при помилках у вводі

Після вводу підходящих даних користувача перенаправить на сторінку авторизації (рис. 3.10 – 3.11). Після чого потрібно натиснути кнопку «Увійти».

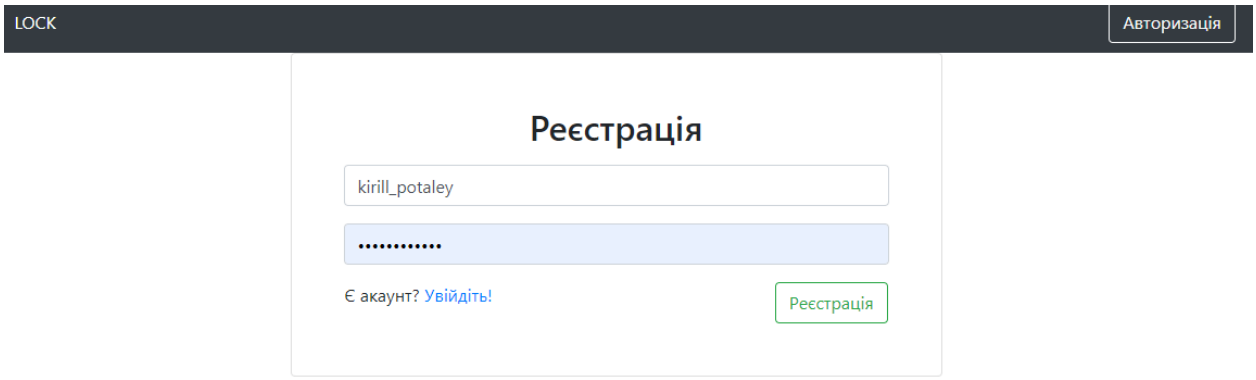

Рисунок 3.10 – Вікно реєстрації при відсутності помилок у вводі

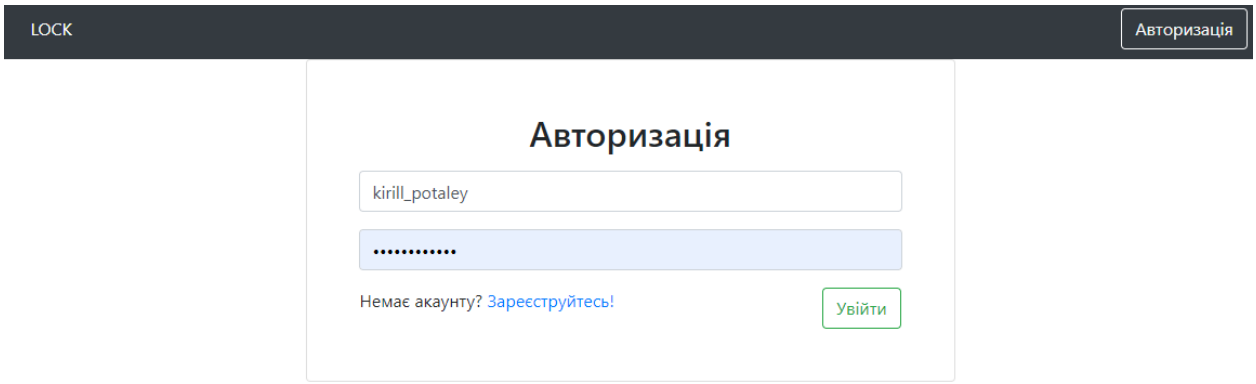

Рисунок 3.11 – Вікно авторизації при відсутності помилок у вводі

Після успішної авторизації користувач бачить пусту сторінку, адміністратору потрібно надати користувачеві права доступу до плати (рис. 3.12).

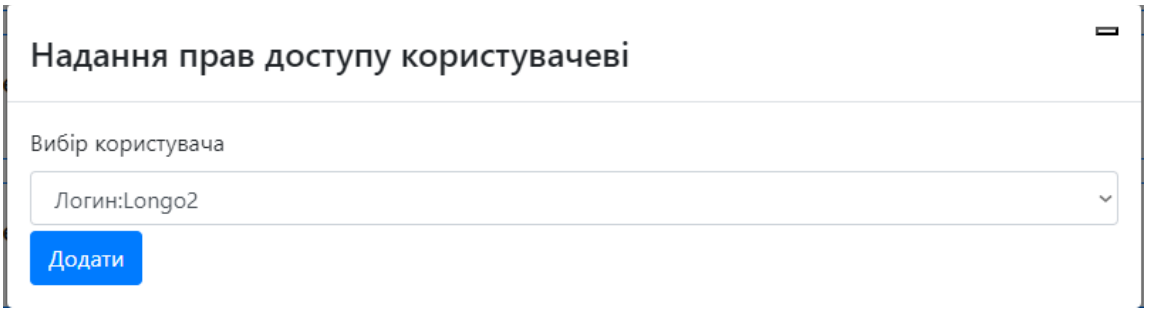

Рисунок 3.12 – Вікно надання прав доступу обраному користувачу

Коли дані були введені , у користувача з’явиться плата (Рисунок 3.13).

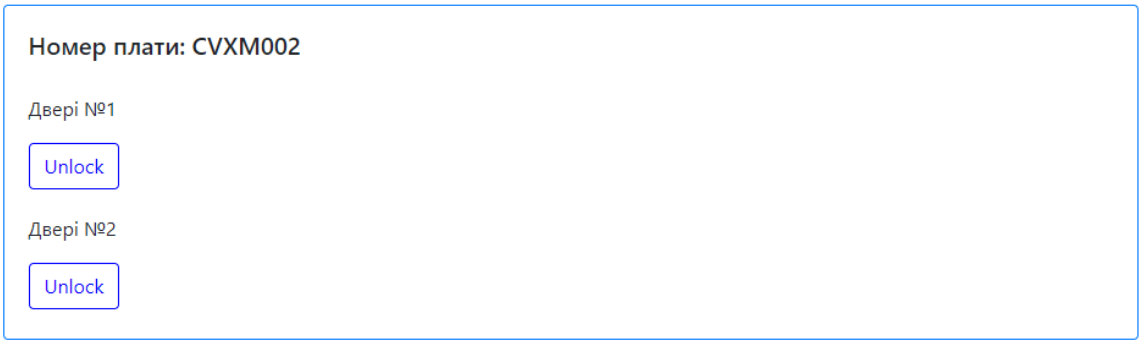

Рисунок 3.13 – Інтерфейс взаємодії з платою

Для взаємодії з дверми ,користувачу потрібно натиснути на кнопку бажаної двері , після чого робота кнопки буде затримана на 15 секунд, а інша кнопка для взаємодії стане не активною також на 15 секунд (рис. 3.14).

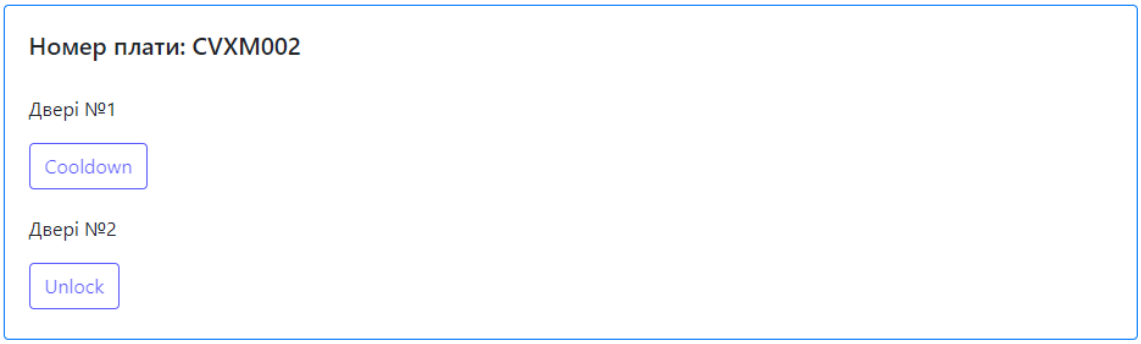

Рисунок 3.14 – Змінений стан кнопок

Для змінення прав доступу користувачів цієї плати потрібно натиснути на «Змінити права доступу», у модальному вікні обрати користувача та варіант зміни прав та натиснути «Ввести» (рис. 3.15 – 3.16).

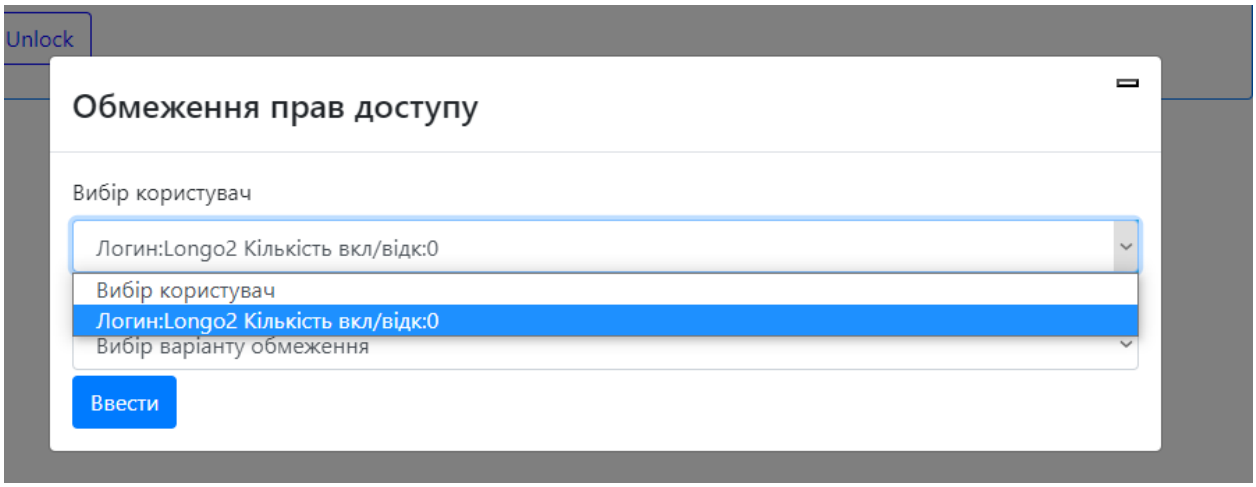

Рисунок 3.15 – Вибір користувача

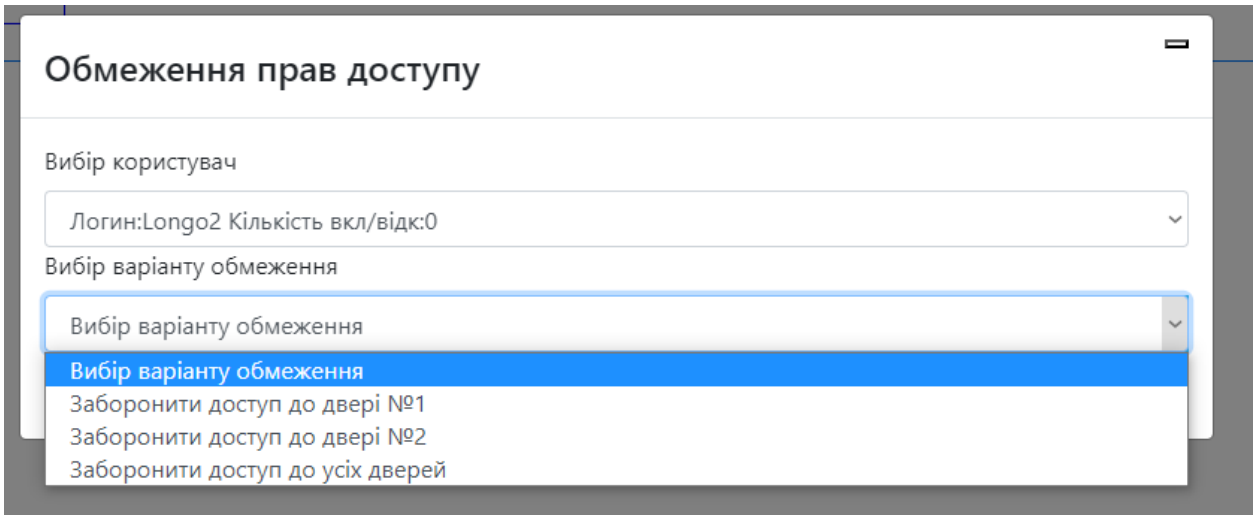

Рисунок 3.16 – Вибір обмеження для обраного користувача

Після чого будуть зміненні права доступу для обраного користувача (рис. 3.17).

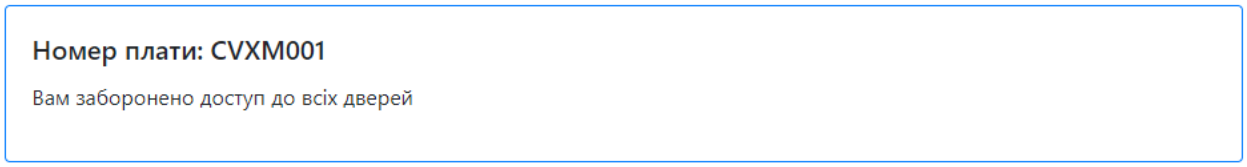

Рисунок 3.17 – Інтерфейс взаємодії з платою при відсутності прав доступу

3.5.3 Користування веб додатком з віддаленого пристрою

Після підключення до веб-адреси додатку - http://212.178.5.97:3000, (в даному випадку через мобільний інтернет) користувач бачить сторінку реєстрації (рис. 3.18), усі функції зберігаються , тому з рисунків 3.18 – 3.24 буде лише демонстрація інтерфейсу.

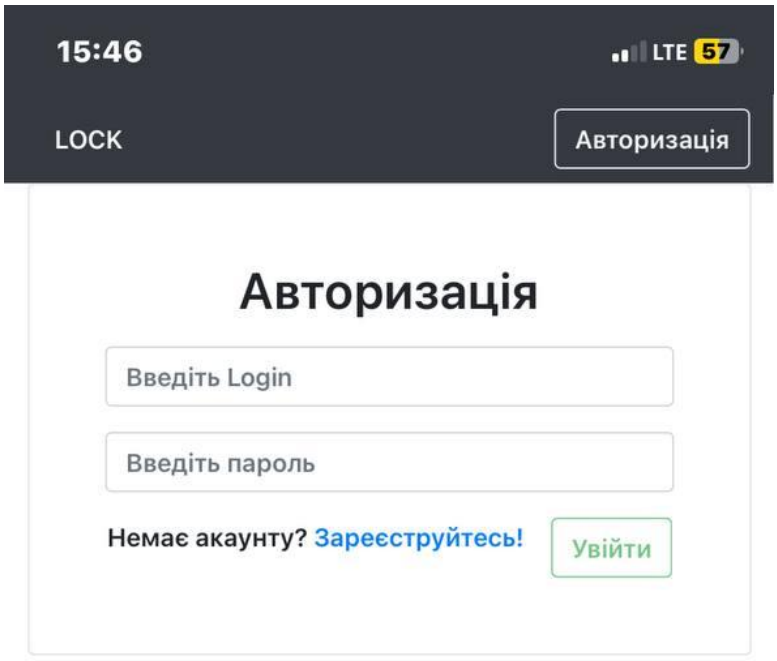

Рисунок 3.18 – Вікно авторизації при відсутності помилок у вводі(телефон)

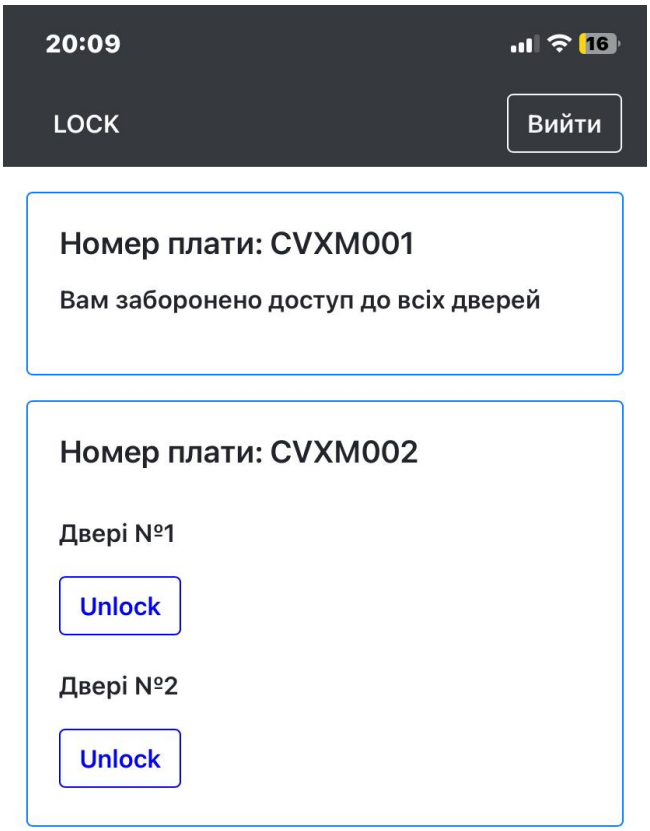

Рисунок 3.19 – Інтерфейс взаємодії з платою

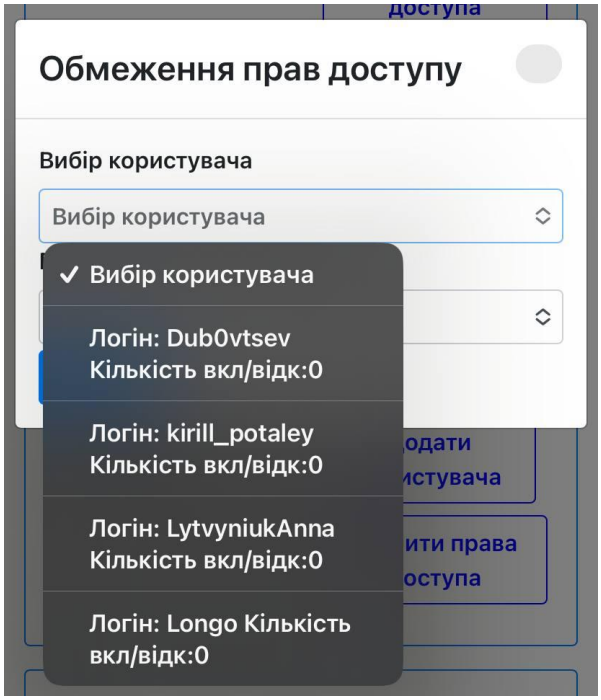

Рисунок 3.20 – Вибір користувача(телефон)

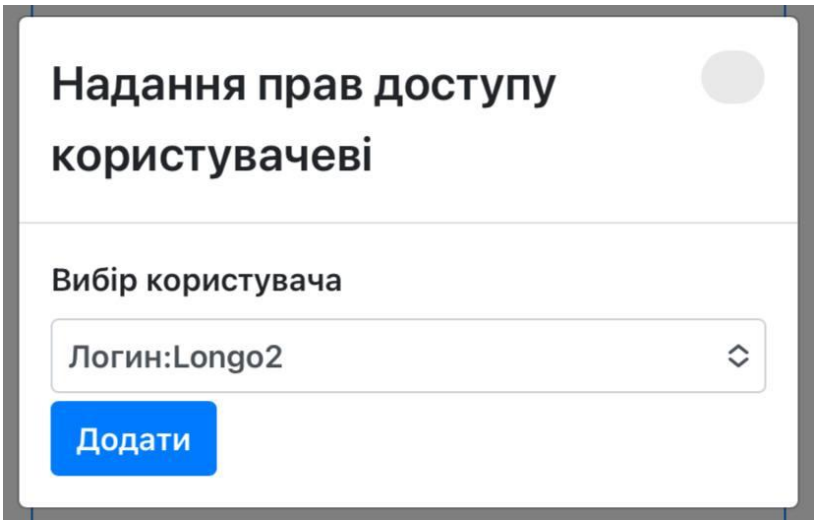

Рисунок 3.21 – Вікно надання прав доступу обраному користувачу(телефон)

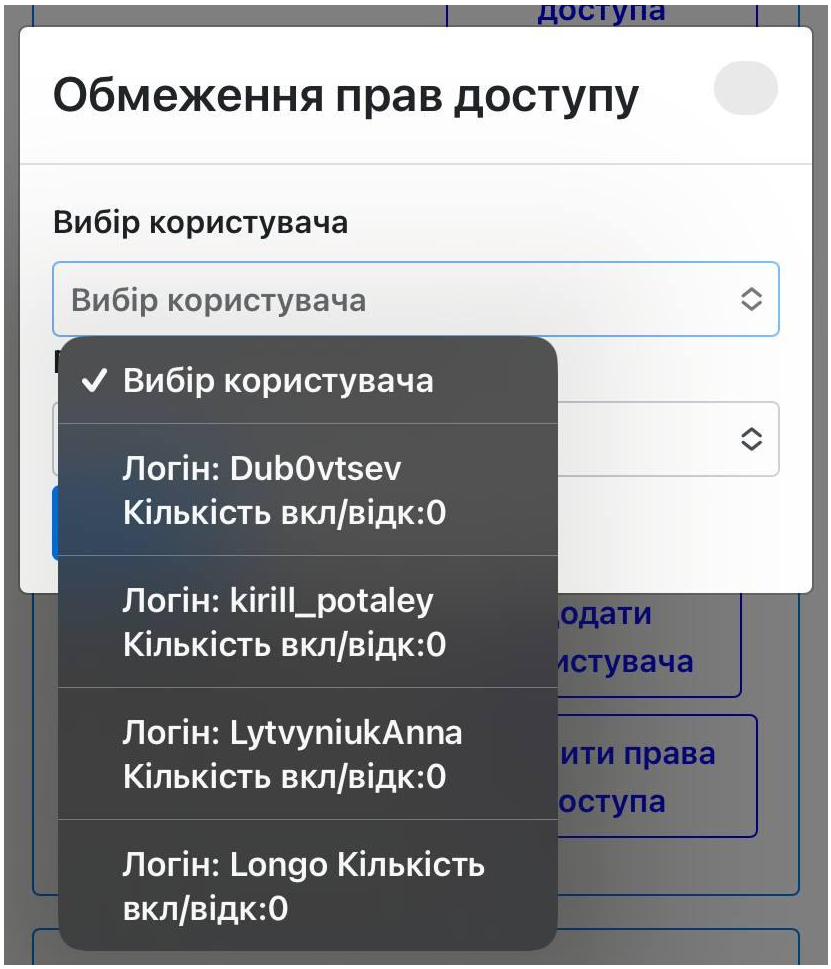

Рисунок 3.22 – Вибір обмеження для обраного користувача(телефон)

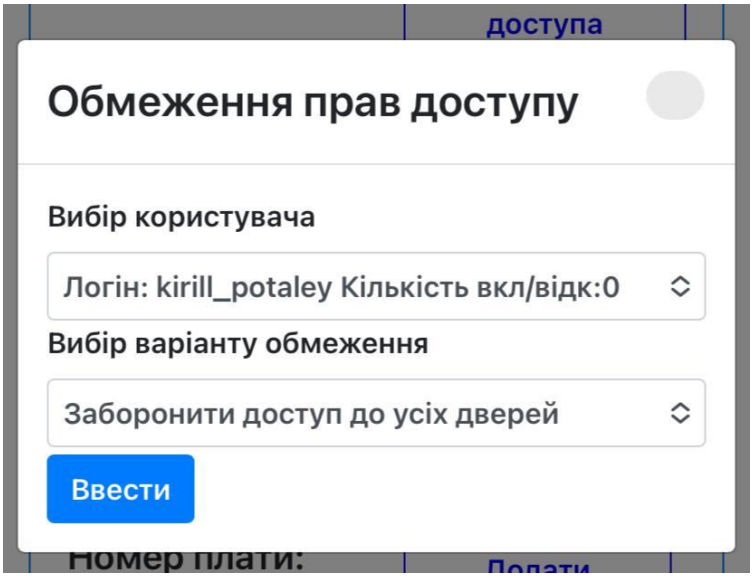

Рисунок 3.23 – Обмеження прав обраного користувача(телефон)

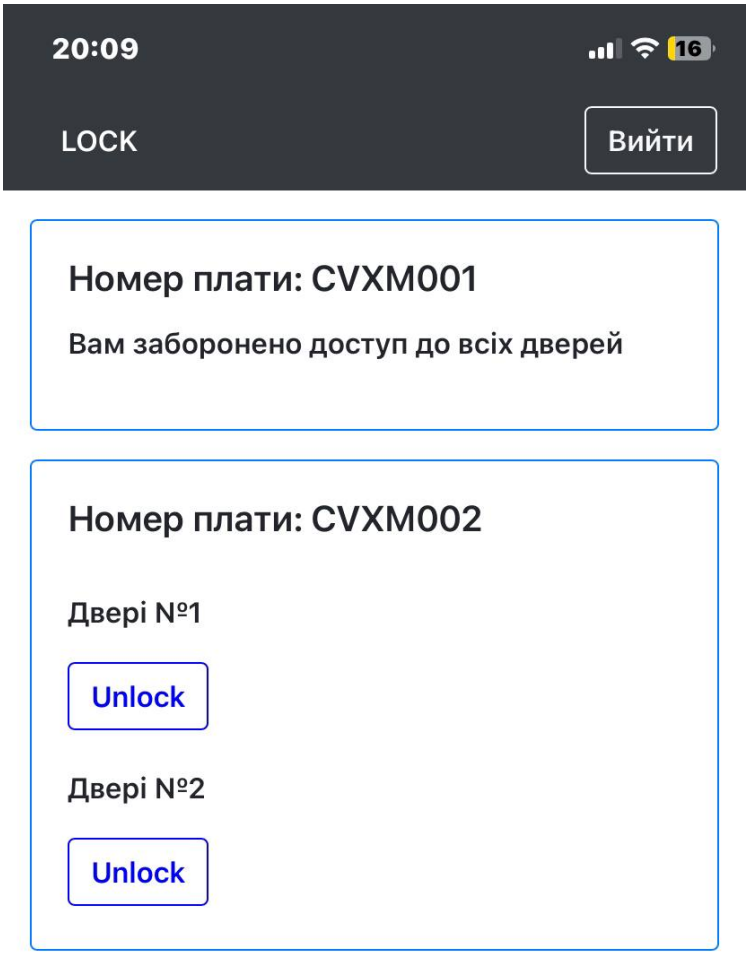

Рисунок 3.24 – Інтерфейс користувача , якому обмежили доступ(телефон)

ВИСНОВКИ

У процесі розробки системи "розумні двері" було розглянуто різні аспекти, такі як апаратна та програмна частини, протоколи комунікації, а також питання безпеки. Основними компонентами системи є мережне обладнання, замки, мікроконтролери, сервер та мобільний додаток для віддаленого керування.

Були обрані оптимальні технології та методи розробки, такі як використання мікроконтролера NodeMCU v3 з вбудованим Wi-Fi-модулем та засоби мобільної розробки для створення інтуїтивно зрозумілого та функціонального інтерфейсу. Була розроблена база даних для обліку користувачів та надання їм відповідних прав доступу.

Розроблена система успішно пройшла тестування та демонструє надійну роботу в умовах реальної експлуатації. Вона дозволяє віддалено контролювати доступ до дверей та отримувати інформацію про події, такі як відкриття або закриття дверей будь якими користувачами.

Впровадження системи "розумні двері" з віддаленим керуванням може принести значні переваги, включаючи підвищення безпеки житла, зручність використання та можливість контролю доступу до будинку в будь-який час та з будь-якого місця.

Подальший розвиток системи може включати розробку додаткових функцій, таких як розпізнавання осіб, інтеграція з іншими системами "розумного дому" або використання біометричних методів ідентифікації.

В цілому, розробка системи "розумні двері" з віддаленим керуванням є успішним і корисним завданням, що сприяє розвитку технологій розумного будинку та підвищенню рівня безпеки та комфорту для користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Компанія August [Електронний ресурс] – Режим доступу: https://august.com

2. Компанія Yalehome [Електронний ресурс] – Режим доступу: https://www.yalehome.com/au/en

3. Компанія Schlage [Електронний ресурс] – Режим доступу: https://www.schlage.com/en/home/smart-locks/connect-zwave.html

4. Nodemcu. Документація [Електронний ресурс] – Режим доступу: https://nodemcu.readthedocs.io/en/release/

5. I2C LCD. Документація [Електронний ресурс] – Режим доступу: http://wiki.sunfounder.cc/index.php?title=I²C_LCD1602

6. Relay. Документація [Електронний ресурс] – Режим доступу: https://docs.arduino.cc/tutorials/4-relays-shield/4-relay-shield-basics

7. Архітектура REST [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/38730/

8. Патерни для новачків: MVC vs MVP vs MVVM. [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/215605/

9. Малахов Є.В. Організація баз даних: конспект. – О.: ОНУ, 2015.

10. Node.js, Express та MongoDB: API за півгодини. [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/ruvds/blog/321104/

11.Рейтинг мов програмування 2022 [Електронний ресурс] – Режим доступу: https://dou.ua/lenta/articles/language-rating-2022/

ДОДАТОК

Опис сутностей та їх властивостей

Таблиця А.1 – Опис сутностей та їх властивостей

Властивість	Опис	Обмеження
Об’єкт «boards»		
board_id	Ідентифікатор плати	Первинний ключ, не порожнє
Serial_number	Серійний номер	Не порожнє, Шаблон '^[A-Z]{4}\d{3}$', <= 10 символів
Об’єкт «relations»		
relation_id	Ідентифікатор зв’язку	Первинний ключ, не порожнє
client_id	Ідентифікатор користувача	Не порожнє, зовнішній ключ для зв’язку з сутністю users(client_id)
board_id	Ідентифікатор плати	Не порожнє, зовнішній ключ для зв’язку з сутністю boards(board_id)
relays	Доступні реле	Не порожнє, Шаблон = `1 ` або `2` або `12 `
counter	Лічильник	Не порожнє, за замовчуванням - 0

Продовження таблиці А.1

Властивість	Опис	Обмеження
Об’єкт «users»		
client_id	Ідентифікатор користувача	Первинний ключ, не порожнє
login	Логін користувача	Не порожнє, <= 15 символів
password	Пароль	Не порожнє, <= 300 символів
Last_auth_date	Дата останньої авторизації	Немає
Last_action_date	Дата останньої взаємодії з дверми	Немає