МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

Навчально-науковий інститут електронних та інформаційних технологій

Кафедра інформаційних та комп’ютерних систем

**Допущено до захисту**

Завідувач кафедри

к.е.н., доцент Базилевич В.М.

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2020 р.

ВИПУСКНА КВАЛІФІКЦІЙНА РОБОТА

за освітньо-професійною програмою бакалавра

РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ З ВИКОРИСТАННЯМ ІДЕНТИФІКАЦІЙНИХ КАРТ

Спеціальність 123 – Комп’ютерна інженерія

Галузь знань 12 – Інформаційні технології

Виконавець:

студент гр. КІ–161

Трухан Антон Володимирович

*(підпис)*

Керівник:

|  |  |  |
| --- | --- | --- |
| *доцент* |  | *к.т.н., доцент* |
| **–––––––––––––––––––––––––––––** |  | **–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––** |
| *(посада)* |  | *(науковий ступінь, вчене звання)* |
|  |  |  |

Риндич Євгеній Володимирович

*(підпис)*

Чернігів 2020

**Індивідуальне завдання на випускну кваліфікаційну роботу**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

Навчально-науковий інститут електронних та інформаційних технологій

Кафедра інформаційних та комп’ютерних систем

ЗАТВЕРДЖУЮ:

Зав. кафедри

к.е.н., доцент Базилевич В.М.

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_2020г.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ  
 ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ БАКАЛАВРА

Трухана Антона Володимировича

Тема роботи: РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ З ВИКОРИСТАННЯМ ІДЕНТИФІКАЦІЙНИХ КАРТ

Тему затверджено наказом ректора

від " " 2020р. № с\_

1. **Вхідні дані до роботи:**

Провести аналіз комп’ютерних систем контролю доступу з використання ідентифікаційних карт, спроектувати архітектуру проекту, програмну та апаратну частини.

1. **Зміст розрахунково-пояснювальної записки:**

Кваліфікаційна робота складається з вступу, основної частини та висновків. Основна частина складається з трьох розділів: «Аналіз задачі створення системи», «Проектування системи» та «Реалізація системи».

1. **Демонстраційні матеріали:**

10 слайдів для презентації роботи

**АНОТАЦІЯ**

Кваліфікаційна робота бакалавра, \_\_ с., 48 рис., 7 табл., \_додатки, 21 джерел.

Об'єктом розробки є комп’ютерна система контролю доступу з використанням ідентифікаційних карт.

Метою роботи було реалізація архітектури, структури всієї системи, структури апаратної, серверної та графічної частин системи контролю доступу.

Основним методом проектування було системне та структурне проектування, для побудови взаємодії системи було використано PlantUML діаграми, також були використані інструменти розробки IntelliJ IDEA так VisualStudioCode з використання PlatformIO плагіну.

У ході виконання кваліфікаційної роботи проведено аналіз існуючих систем і було виконано наступні пункти:

– Створення архітектури системи контролю доступу з використання ідентифікаційних карт;

* Створення схеми принципової, вибір апаратних засобів для створення пристрою контролю доступу;
* Проектування і реалізація програмних частин апаратної, серверної та графічної частин системи;

– Розробка структури бази даних ;

Користувач може використовувати систему для налаштування контролю доступу в підприємстві або в приватних цілях на території будь якого масштабу, винятком є те що, на території де повинні працювати пристрої контролю доступу повинна бути WIFI мрежа і всі пристрої повинні знаходитися в одній локальній мережі.

Можливе подальше вдосконалення системи шляхом додавання нового функціоналу: розширення видів зчитувачів, інтеграція з іншими СКДУ.

Робота має практичну цінність. Розрахунок економічної ефективності не проводився.

Typescript, JavaScript, REST, Json, MySql, PlatformIO, Arduino, СКУД, RFID

**АНОТАЦІЯ**

Qualifying work of the bachelor, \_\_ p., 48 fig., 7 table., \_ annexes, 21 sources.

The object of development is a computer access control system using identification cards.

The purpose of the work was to implement the architecture, the structure of the entire system, the structure of the hardware, server and graphical parts of the access control system.

The main design method was system and structural design, PlantUML diagrams were used to build the system interaction, IntelliJ IDEA and VisualStudioCode development tools using the PlatformIO plugin were also used.

During the qualification work the analysis of the existing systems was carried out and the following points were fulfilled:

- Creating the architecture of the access control system using identification cards;

- Creating a schematic diagram, the choice of hardware to create an access control device;

- Design and implementation of software parts of hardware, server and graphics systems;

- Development of database structure;

The user can use the system to set up access control in the enterprise or for private purposes in any area, except that in the area where the access control devices must work, there must be a WIFI network and all devices must be in one local network.

It is possible to further improve the system by adding new functionality: expanding the types of readers, integration with others ACS.

The work has practical value. The calculation of economic efficiency was not performed.

Typescript, JavaScript, REST, Json, MySql, PlatformIO, Arduino, ACS, RFID

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ**

RFID – Radio Frequency Identification

LF – Low frequency

HF – Hight frequency

UHF – Ultra-Hight frequency

NFC – Near field communication

MCV – Model-View-Controller

СУБД – Система управління базами даних

CRUD – Create Read Update Delete

API – Application Programming Interface

HTTP – HyperText Transfer Protocol

JSON – JavaScript Object Notation

**ЗМІСТ**

[ВСТУП 8](#_Toc41215156)

[1 АНАЛІЗ ЗАДАЧІ СТВОРЕННЯ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ З ВИКОРИСТАННЯ ІДЕНТИФІКАЦІЙНИХ КАРТ 9](#_Toc41215157)

[1.1 Аналіз систем контролю та управління доступом 9](#_Toc41215158)

[1.2 Дослідження існуючих аналогів систем контролю та управління доступу 10](#_Toc41215159)

[1.3 Аналіз вимог до системи контролю доступу 17](#_Toc41215160)

[2 ПРОЕКТУВАННЯ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ З ВИКОРИСТАННЯ ІДЕНТИФІКАЦІЙНИХ КАРТ 19](#_Toc41215161)

[2.1 Проектування архітектури комп'ютерної системи контролю доступу з використанням ідентифікаційних карт 19](#_Toc41215162)

[2.1.1 Компонент “Пристрій контрою доступу” 20](#_Toc41215163)

[2.1.2 Компонент “Головний сервер” 21](#_Toc41215164)

[2.1.3 Компонент “Графічний інтерфейс в Web браузері” 22](#_Toc41215165)

[2.1.4 Користувач “ Користувач з ідентифікаційною карткою” 23](#_Toc41215166)

[2.1.5 Користувач Адміністратор” 23](#_Toc41215167)

[2.2 Проектування апаратної частини комп'ютерної системи контролю доступу з використанням ідентифікаційних карт 25](#_Toc41215168)

[2.2.1 Вибір стандарту ідентифікаційних карток 25](#_Toc41215169)

[2.2.2 Вибір технічних засобів побудови апаратної частини системи контролю доступу 27](#_Toc41215170)

[2.2.3 Проектування схеми принципової 32](#_Toc41215171)

[2.3 Проектування програмної частини комп'ютерної системи контролю доступу з використанням ідентифікаційних карт 34](#_Toc41215172)

[2.3.1 Програмні засоби для побудови апаратного застосунку 35](#_Toc41215173)

[2.3.2 Технічні засоби для побудови серверного застосунку 37](#_Toc41215174)

[2.3.3 Технічні засоби для побудови графічного інтерфейсу 39](#_Toc41215175)

[2.3.4 Проектування структури бази даних 39](#_Toc41215176)

[2.3.5 Програмне забезпечення пристрою контролю доступу 41](#_Toc41215177)

[2.3.6 Проектування серверної частини 45](#_Toc41215178)

[2.3.7 Проектування клієнтської частини 50](#_Toc41215179)

[3 РЕАЛІЗАЦІЯ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ З ВИКОРИСТАННЯМ ІДЕНТИФІКАЦІЙНИХ КАРТ 56](#_Toc41215180)

[3.1 Реалізація апаратної підсистеми 56](#_Toc41215181)

[3.1.1 Реалізація програмної підсистеми 56](#_Toc41215182)

[3.1.1.1 Клас Main 56](#_Toc41215183)

[3.1.1.2 Клас ConfigStorage 57](#_Toc41215184)

[3.1.1.3 Клас WebServer 58](#_Toc41215185)

[3.1.1.4 Клас AccessControl 59](#_Toc41215186)

[3.1.2 Реалізація фізичного пристрою контролю доступу 60](#_Toc41215187)

[3.1.3 Перевірка реалізованого пристрою контролю доступу 61](#_Toc41215188)

[3.2 Реалізація серверної підсистеми 63](#_Toc41215189)

[3.2.1 Реалізація бази даних системи контролю доступу 63](#_Toc41215190)

[3.2.2 Реалізація глобальних інтерфейсів та опис глобального класу констант 66](#_Toc41215191)

[3.2.3 Реалізація API контролерів 67](#_Toc41215192)

[3.2.3.1 Контролер AccessControlController 67](#_Toc41215193)

[3.2.3.2 Контролер CardController 68](#_Toc41215194)

[3.2.3.3 Контролер DeviceController 68](#_Toc41215195)

[3.2.3.4 Контролер LogController 69](#_Toc41215196)

[3.2.3.5 Контролер RoleController 69](#_Toc41215197)

[3.2.3.6 Контролер SessionController 69](#_Toc41215198)

[3.2.4 Реалізація сервісів 70](#_Toc41215199)

[3.2.4.1 Сервіс AccessLoggerService 70](#_Toc41215200)

[3.2.4.2 Сервіс DeviceControlService 70](#_Toc41215201)

[3.2.4.3 Сервіс SessionService 71](#_Toc41215202)

[3.2.5 Реалізація моделей для взаємодії з базою даних 71](#_Toc41215203)

[3.3 Реалізація клієнтської підсистеми 72](#_Toc41215204)

[3.4 Взаємодія серверу та клієнта 78](#_Toc41215205)

[3.5 Реалізація автоматизованої збірки кожного модулю проекту та запуск головного серверу з графічним додатком 79](#_Toc41215206)

[ВИСНОВКИ 81](#_Toc41215207)

[ПЕРЕЛІК ПОСИЛАНЬ 83](#_Toc41215208)

**ВСТУП**

В сучасному світі стала дуже важливим питання контролювати доступ до деякий об’єктів або просто до приміщень. Для вирішення цього питання було винайдено різноманітні системи контролю доступу. Великою перевагою сучасних систем перед класичним фізичним контролю де деяка людина знаходиться перед входом до будівлі чи приміщення і контролює доступ, але такий спосіб є не дуже доцільним, до випадків коли контроль повинен бути дуже гнучким і існує багато приміщень де повинен проходити контроль.

Системи контролю доступу та управління повинні вирішити питання контролю доступу до приміщень, будівель та іншого. Також сучасні системи контролю доступу мають широкий спектр використання та інтеграцій з іншими системами, одним із доповнень до системи може бути контроль робочого часу працівників на підприємстві, контроль камер спостереження, відстеження працівників та інших людей на підприємстві.

Системи контролю доступу стали дуже розповсюдженою річчю в світі, їх використовують для різноманітних цілей, для захисту приватних, офісних будівель, контролю доступу до конкретних приміщень, автоматизація для комерційних цілей, коли за пропуск до певної території треба платити і час перебування на території є обмеженим. Більшість таких систем є автоматизованими і керувати ними можна з робочого комп’ютера, але більшість таких інформаційних систем дуже дорогі або для масштабування кількості контролерів для двері потрібно докупати основну систему що дуже сильно підвищує ціну.

Основною метою роботи є створення своєї системи контролю доступу з самим пристроєм контролю доступу, з використанням ідентифікаційних карт. Розроблена система повинна мати зручний інтерфейс для налаштування всієї системи. А також така система повинна бути з можливістю інтеграції з іншими системами або розширення самої її без значних затрат на це.

1. **АНАЛІЗ ЗАДАЧІ СТВОРЕННЯ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ З ВИКОРИСТАННЯ ІДЕНТИФІКАЦІЙНИХ КАРТ**

В даному пункті виконаний аналіз системи контролю доступу, побудовано базову модель системи контролю доступу, а також виконаний аналіз існуючих систем контролю доступу та порівняні можливості кожної з цих систем.

* 1. **Аналіз систем контролю та управління доступом**

Система контролю і управління доступу (СКУД) – це сукупність різних технічних або апаратних видів захисту, яка має мету авторизація, обмеження та реєстрування фізичних або технічних об’єктів на певній території через пункти контролю: двері, ворота, тощо.

Авторизація в системі контролю доступу полягає в перевірці достовірності особи, зазвичай це робиться шляхом зрівняння параметрів особи(це може бути відбиток пальцю, зображення обличчя або структура сітчатки ока), перевірці того що знає особа(наприклад пароль, PIN-код), або перевірка того що є в особи( ідентифікаційної картки тощо).

Головне завдання системи полягає в керуванні доступу до певної території, також в завдання може включати в себе рахування робочого часу, підрахунок заробітної плати в ситуаціях якщо реалізовано інтеграцію з бухгалтерською системою, наступним завданням може бути ведення списку осіб які перетинають територію.

Типове СКУД представляє з себе наступні компоненти:

* Контролер СКУД, пристрій для обробки даних та проведення авторизації осіб, керує технічним засобом для перешкоджанням неавторизованим особам. В ньому зберігається деяка конфігурація системи, список осіб які мають доступ, а також конфігурація про режими роботи пристрою, також контролер може бути налаштований в режим при якому він буде запитувати підтвердження для прийняття рішення у головного комп’ютера;
* Пристрої для зчитування, вони є одним із найголовнішим пристроєм в систему, ці пристрої підключаються до контролеру. Цей пристрої призначені для ідентифікації осіб які хочуть перетнути територію, може представляти з себе радіочастотні зчитувачі, прилади які використовують для ідентифікації машинний зір, дактилоскопічні сканери, які можуть передавати ідентифікаційні данні до контролеру який проведе авторизацію цього ідентифікатора;
* Програмне забезпечення, яке дозволяє створити централізоване управління с віддаленого комп’ютера, вести логування (процес збирання даних о роботі програми або системи) всіх дій в системі;
* Пристрій для підключення апаратних модулів один до одного або з персональним комп’ютером;
* Керований технічний засіб який керується за допомогою реле перешкоджу неавторизованим особам перетинати деяку територію, це може бути двері, турнікети, електромагнітні замки, ворота тощо;
* Допоміжні засоби, блоки живлення, дроти, тощо.

Таким чином були виділені основні поняття та компоненти системи контролю та управління доступом.

* 1. **Дослідження існуючих аналогів систем контролю та управління доступу**

Зазвичай види системи контролю доступу поділяються за способом ідентифікації осіб, шляхом проведення авторизації.

Ідентифікація – це процес під час якого проводиться перевірка особи, щоб упевнитися що це є та особа за яку він з себе представляє. Ідентифікація осіб може бути проведена за допомогою двох способів ідентифікації. Це може бути ідентифікація за допомогою електронних карт і ідентифікація яка використовую підрахунку унікального біометричного або біологічного параметру.

В сучасному світі виділяють наступні види біометричної ідентифікації[1]:

* Ідентифікація по голосу – це є дуже зручний спосіб для користування але недоліком цього виду є низька надійність, тому що голос людини може змінитися в різноманітних ситуаціях, наприклад людина захворіла і в неї зник або змінився голос.
* Ідентифікація по рисам обличчя, це процес аналізу унікальних рис обличчя, для підтвердження особи. Такими рисами є колір шкіри, спеціальна структура точок на обличчі див рис. 1.1. Але недоліком цього способу ідентифікація є те що зображення повинно бути дуже доброї якості, освітлення повинно бути яскравим і такий спосіб не може відрізнити близнюків. Прикладом подібного пристрою може бути термінал контролю доступу з розпізнаванням обличь Hikvision DS-K1T605E[2], пристрій зображено на рис. 1.2.

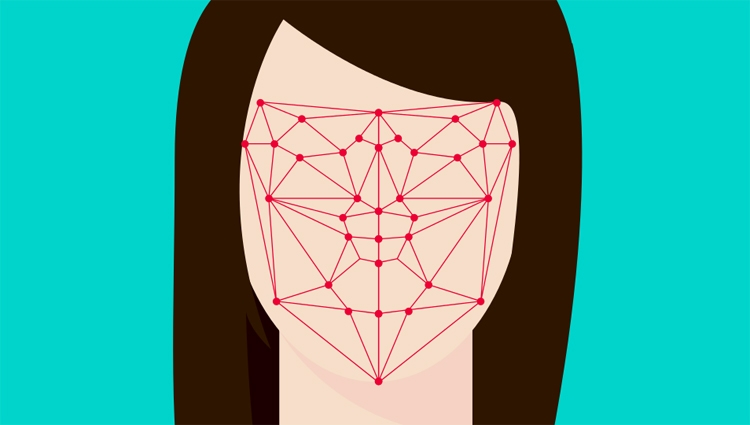


Рисунок 1.1 – Аналіз характерних точок на обличчі людини



Рисунок 1.2 – термінал контролю доступу з розпізнаванням обличь Hikvision DS-K1T605E

* Ідентифікація по відбитку пальцю – сканування є одним із зручних способом ідентифікації. Перевага цього способу є надійність і низька вартість таких пристроїв. Приклад такого пристрою є зчитувач відбитків пальців Hikvision DS-K1201EF [3], пристрою зображено на рис.1.3.



Рисунок 1.3 – Зображення зчитувачу пальців Hikvision DS-K1201EF

* Ідентифікація по геометрії кисті руки або долоні – сканер не перевіряє лінії долоні як це робиться при перевірці відбитку пальцю, а перевіряється геометрія долоні, кисті, довжина пальців і інше. Надійність таких систем є такою самою як і перевірка відбитку, але одним із недоліків є великі габарити зчитувача.
* Ідентифікація по шляхом сканування ока – розрізняють сканування по райдужній оболонці, і сканування сітчатки ока. На рис. 1.4 зображений приклад зчитувача ока. Перший принцип є найпростіший але він не дуже надійний, другий надійним але такі системи є дуже дорогими.



Рисунок 1.4 – Зображення зчитувача Iris Access 3000 Series

Електроні картки поділяються на такі типи:

* Магнітні карти – такі картки мають в собі магнітну полосу з записаній на неї інформацією. На рис. 1.5 зображений приклад магнітних карток. Однією з переваг і в той же час недоліком таких карт є те що інформацію в них можна змінити і такі картки можна з легкістю підробити, іншої перевагою таких карток є те що вони дешеві. Недоліком цих карт є те що вони дуже чутливі до зовнішніх факторів впливу такі як подряпини, забруднення або волога. Також до недоліків можна віднести те що в таких карток є механічний контакт який підноситься до зчитувача, який може бути пошкоджений, іншим схожим недоліком є те що такі картки дуже чутливі до магнітних полів, ці недоліки можуть привести спотворенню даних або видалення всієї інформації з неї.



Рисунок 1.5 – Зображення магнітних карток

* Виганд-карти(Wiegand) – такі карти названі в честь вченого який винайшов спеціальний сплав, який використовується в таких картках. На рис. 1.6 зображений приклад Виганд-карти. В середині таких карток знаходиться металеві дроти які розміщені в певному порядку яка складається в певну комбінацію. Дроти прикріпляються за допомогою спеціального клею, після цього переміщення дротів неможлива. При зчитуванні картки дроти створюють магнітний імпульс який потім зчитується. Особливість таких карток є те що не можливо смінити код картки після її створення. Даний вид карток не схильними до пошкодженням шляхом впливу електромагнітного поля або високої температури. Недоліком таких карток є те що вони можуть бути пошкоджені шляхом згину картки. Прикладом зчитувача для таких карт може слугувати зчитувач Hikvision DS-K1102E, зображення цього зчитувача зображено на рис. 1.7.



Рисунок 1.6 – Зображення Виганд-карти компанії HID Corporation



Рисунок 1.7 – Зображення зчитувача Hikvision DS-K1102E

* Штрих кодові картки – такі картки мають на одній із сторін полоси клір яких відмінним від кольору картки, довжина і відстань між цими полосами представлена як спеціальна кодова комбінація, яка потім зчитується, існує модифікація подібних карток, штрих код знаходиться під спеціальним матеріалом який є прозором для зчитувача і зчитування. На рис. 1.7 зображений приклад штрих кодової картки.



Рисунок 1.7 – Зображення штрих кодові картки компанії Edna

* Електроні ключі “Touch-memory” – представляє з себе брилок всередині яких знаходиться металевий капсуль в якому знаходиться мікросхема. Запис, додавання або стирання даних виконується за допомогою майстер-ключа. На рис. 1.8 зображений приклад такого ключа. Інформація зчитується при доторкуванні ключа з зчитувачем. Всередині ключа знаходиться батарейка яка живить мікросхему. Перевага такого ключа є те що він стійкий до механічних пошкоджень. Недоліком такого ключа є те що батарейка слугує тільки декілька років і після цього потрібно замінити ключ.



Рисунок 1.8 – Зображення touch-memory ключ

* Безконтактні (Proximity) карти – такі картки є найперспективнішими на сьогоднішній час, так як картки працюють на відстані і не потребують чуткого позиціонування на зчитувачі. Зчитувач для таких карток генерую електромагнітне випромінювання заданої частоти. Такі картки поділяються на пасивні і активні. Пасивні картка коли потрапляють в зону дії зчитувача, випромінювання живить чип через антену в карті після отримання достатньої енергію картка відсилає інформацію на зчитувач. Активні мають всередині батарейку яка працює досить довго приблизно до 10 років. Перевагами такої картки є те що картка може бути зчитана через деякі перешкоди, також перевагою є те що зчитувач можна розмістити за не металевою стінкою. Недоліком такої картки є те що вона може не працювати при дуже сильних електромагнітних полів. На рис. 1.9 зображено приклад proximity карток.



Рисунок 1.9 – Зображення proximity карток

Отже, ми розглянули можливі варіанти систем контролю доступу починаючи з біометричних і закінчуючи системами з використанням електронних карт. В сучасному світі системи з біометричною перевіркою є чи досить дорогими , отже для розробки системи ми будемо використовувати систему контролю доступу яка базується на електронних картках, а саме на proximity картках пасивного виду. Вони коштують дешево, не потребують додаткового живлення і можуть працювати доки їх не зламати що зробити досить важко для варіанту в виді брилка.

На даний час в Україні існує дуже велика кількість компаній які розробляють системи контролю доступу які дозволяють давати доступ з використанням ідентифікаційних карт. На підставі аналізу декілька зразків систем контролю доступу компаній Hikvision, Dahua, U-Prox та інших була складена таблиця, яка містить порівняльні функціональні особливості цих систем (таблиця 1.1).

Після проведення аналізу існуючих систем контролю доступу, було виявлено що більшість систем контролю доступу не мають зручного управління системою, в більшості систем є тільки деяка категорія ролей яка дозволяє швидко присвоїти один доступ для деякої груп але цей список невеликий, 1-2 ролі, користувач і адміністратор, деякі системи не мають можливості налаштування через комп’ютер шляхом підключення USB або через локальну мережу. Більшість систем для кожної двері потребують новий контролер який має дуже велику вартість и при масштабуванні вартість системи може вирости дуже сильно. Також велика кількість систем контролю доступу не мають можливості керувати всіма контролерами централізовано з одного комп’ютера.

В зв’язку з аналізом існуючих систем можна зробити вивід що проект який розробляється буде корисним і унікальним.

Таблиця 1.1 – Порівняльна характеристика параметрів систем контролю доступу і управління

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Назва систем контролю доступу | | | | | |
| Fortnet ABC v 12.3e | U-Prox IP400 | ЕМ280-Е комплект | DAHUA DHI-ASA1222E | Hikvision DS-K2604 | Розроблена система |
| Ціна за контролер для однієї двері < 3000 | - | - | + | + | - | + |
| Можливість проводити авторизацію окремим контролером | - | + | - | - | - | + |
| Можливість зберігати > 10000 ключів в системі | + | + | - | - | + | + |
| Можливість керувати контролером з комп’ютера | + | - | - | - | + | + |
| Можливість налаштування пристрою за допомогою протоколів TCP/IP | + | + | - | - | + | + |
| Присуне розмежування користувачів за ролями | - | - | - | + | + | + |
| Можливість масштабування системи шляхом централізованого підключення N-контролерів | + | + | - | - | - | + |

* 1. **Аналіз вимог до системи контролю доступу**

Виходячи з аналізу існуючих систем контролю доступу було зроблено наступні вимоги до системи. Система контролю доступу повинна мати можливість управляти великою кількістю пристрої, приблизно 50000+ пристроїв. Система повинна мати можливість створювати ролі, або групи осіб, за допомогою яких можна буде керувати групами осіб з картками доступу. Система повинна мати можливість проводити авторизацію користувачів за допомогою окремого контролера/комп’ютера. Система має мати гнучкий інтерфейс для керування всіма пристроями та системою взагалі.

Для того щоб зробити можливість керувати системою з комп’ютера, та проводити авторизацію не на боці контролерів, система буде поділена на дві сутності 1) контролери або пристрої контролю доступу задача яких зчитувати інформацію з карток та відсилати на сервер 2) головний сервер, який зберігає інформацію про всі пристрої, проводить авторизацію всіх пристроїв.

Виходячи з вимог до системи пристрої контролю доступу повинен відповідати наступним вимогам:

* Пристрій повинен мати деякий контролер для оброблювання інформації;
* Пристрій повинен мати доступ до локально мережі для можливості відправки запитів за допомогою інтерфейсів TCP/IP;
* Пристрій має мати веб сервер для можливості проводити налаштування пристрою;
* Пристрій повинен мати можливість зберігати інформацію про конфігурацію в флеш-пам’яті;
* Пристрій повинен мати базову авторизацію для внутрішнього веб серверу для того що уникнути керування їм стороннім особам;
* Пристрій повинен мати можливість відправляти запити на головний сервер для проведення авторизації ідентифікаційних карт;
* Пристрій повинен мати можливість зчитувати ідентифікаційну картку формату proximity;
* Пристрій повинен мати індикацію яка буду інформувати про те чи отримав користувач доступ чи ні.

Серверний застосунок повинен відповідати наступним вимогам:

* Застосунок повинен працювати на операційних системах Windows та Linux;
* Розробка застосунка повинна бути на мові програмування яка підтримує ООП модель проектування;
* Застосунок повинен мати веб сервер який надає можливістю керувати всією систему;
* Застосунок повинен мати можливість керувати пристроями контролю доступу;
* Застосунок повинен мати веб інтерфейс який буде користуватися веб сервером для керування всією системою;
* Застосунок повинен мати логування всіх спроб авторизації в систему;
* Застосунок повинен мати авторизацію для захищення системи від сторонніх осіб.

Таким чином створення системи контролю доступу з використанням сучасних мережевих технологій які б дозволили отримувати доступ через різні джерела є актуальною задачею.

1. **ПРОЕКТУВАННЯ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ З ВИКОРИСТАННЯ ІДЕНТИФІКАЦІЙНИХ КАРТ**

Виходячи з виконаного аналізу систем контролю доступу, який був проведений на попередньому етапі, можна приступи до створення всієї архітектури та сформувати базові вимоги до проекту що буде розроблюватись.

* 1. **Проектування архітектури комп'ютерної системи контролю доступу з використанням ідентифікаційних карт**

Архітектура системи побудована на 3-х рівневої моделі клієнт-серверної архітектури або MCV моделі проектування. Всі операції перевірки, створення, отримання даних с СУБД виконуються на стороні серверу. Детальна архітектура системи зображена на рис 2.1.

Модель MCV проектування ділить систему на наступні 3 рівні [4]:

* Model – модель представляє з себе базу даних або будь-яке сховище даних яка виконує функції зберігання інформації, користувач же викликає деякі процедури які виконують операції над моделлю;
* Controller – контролер емулює дії користувача виконуючи операції над моделлю, а також передаючи дані між моделлю і представленням і навпаки;
* View – представлення представляє з себе інтерфейс для взаємодії з системою, задача цього рівню в отриманні даних та представленню користувачу їх.

Розроблювана система поділена на 3 основні компоненти які спроектовані по принципу MCV архітектури та на 2 видів користувачів, див рис 2.1.

1. Компонент: пристрої контролю доступу
2. Компонент: головний сервер
3. Компонент: графічний інтерфейс в Web браузері
4. Користувач “Користувач з ідентифікаційною карткою”
5. Користувач “Адміністратор”
   * 1. **Компонент “Пристрій контрою доступу”**

Пристрій контролю доступу – це пристрій який зчитую інформацію с ідентифікаційних карт(RFID-мітка) відправляє зчитаний ідентифікатор на головний сервер для перевірки, після отримання результату пристрій вирішує давати чи ні живлення на механізм відмикання(відчиняти двері чи ні).

Модуль управління – це головний модуль в пристрої, в цьому модулі знаходиться головний потік виконання. Головна задача цього модулю ініціалізувати інші 3 модулі системи: модуль конфігурації, модуль веб серверу, модуль контролю доступу, також його задача полягає в тому щоб підготувати всі фізичні інтерфейси для роботи з ними.

Модуль конфігурації – це модуль для зчитування і зберігання конфігурації, а також він реалізує підготовку внутрішньої пам’яті для можливості читання. Цей модуль є спільним для модулю контролю доступу та модулю веб серверу так як всередині зберігається інформація яка визначає їх роботу. Модуль записує та зчитує конфігурацію за допомогою бібліотеки FS, яка реалізую основні операції з внутрішньою пам’яттю, детальніше про цю бібліотеку буде описано в розділі 2.2.

Модуль веб серверу реалізує деякі кінцеві точки (англ. endpoints) для керування пристроєм. Головна задача серверу полягає в прийнятті зовнішніх запитів на оновлення конфігурації пристрою та запит на перевірку стану пристрою. Для того щоб запобігти керування пристроєм сторонніми особами було розроблено авторизацію по токену.

Модуль контролю доступу реалізую основні функцію пристрою по зчитування ідентифікаційних карток їх перевірка шляхом відправки запиту на головний сервер так керування модулем живлення механізму замикання і модулем індикації. Модуль реалізує наступні речі: 1) ініціалізація модулів якими він керує, а саме фізичні інтерфейси доступу до Led індикатора SMD-5050 RGB і інтерфейсу керування модулем IRF520, реалізую ініціалізацію модулю RFID-RC520 2) зчитування RFID-міток шляхом перевірки наявності нових карток; 3) замикання або розмикання механізму замикання 4) зміна індикації для сповіщення перевірки доступу. Детальних опис про всі пристрої буде проведено в розділі 2.

* + 1. **Компонент “Головний сервер”**

Головний сервер відповідає за оброблення всіх даних, реалізує функції керування системою такі як CRUD операції над сутностями системи, оновлювання конфігурацій пристрої контролю доступу, а також оброблювання запитів на перевірку доступу до пристрою. Головний сервер поділений на наступні модулі:

* API схеми
* Контролери API
* Сервіси
* Моделі бази даних

API схеми – модуль відповідальний за реалізацію схеми запитів, тобто що кожен запит приймає, що є обов’язковим для запиту, який формат і тип має кожне поле та як поле валідується (чи є поле обов’язковим, яка мінімальна та максимальна довжина поля), також схема описує що метод може повернути. Схеми кросні ще тим що спеціальні бібліотеки які використовують їх перед предачею запиту від користувача, чи відправкою запиту до користувача прибирає поля яких не повинно бути в запиті чи відповіді, що полегшую розробку застосунку.

Контролери API – модуль який реалізую всі можливі запити які клієнт може виконати. Контролери мають всередині методи або так звані хендлери, які описують той чи інший запит. Умовно цей модуль поділений на дві частини, контролери які використовує пристрій контролю доступу, наприклад в середині цього модулю знаходиться метод для перевірки доступу до деякої двері, інша частина контролерів відповідає за реалізацію endpoints для клієнта.

Сервіси – модуль сервісів відповідає за реалізацію утилітну частин проекту, наприклад створення логу про активність користувачів, відправка нового конфігурації до пристрою контролю доступу чи перевірка статусу пристрою.

Моделі бази даних – реалізують схеми баз даний за допомогою який потім з використанням ORM можна буде з легкістю отримувати данні з бази даних.

* + 1. **Компонент “Графічний інтерфейс в Web браузері”**

Ввідповідає за відображення статусу системи а також за відображення основних даних системи таких як існуючі в системі сутності, відображення подій які відбулися за останній час

Головний сервер поділений на наступні модулі:

* Сервіси
* Моделі та стори даних
* Компоненти сторінок

Сервіси – модуль який реалізую основну функції які потрібні для взаємодії з іншими джерелами інформації. В цьому модулі реалізована взаємодія з API головного серверу, яке дозволяє спростити комунікацію. Також в цьому модулі реалізується збереження стану додатку який змінюється кожного разу при переході між сторінками і додатково реалізує деякі інтерфейсні функціональності які використовуються глобально. Цей модуль містить в собі утиліті сервіси які призначені для спрощення зміни деякий станів додатку або реалізацію

Моделі та стори даних – цей модуль відповідає за реалізацію збереження та маніпулювання даними API. Модель – це описання деякої сутності системи. За допомогою моделі можна викликати запит на отримання інформації про конкретний запис, можна оновити запис, видалили його з бази даних або створити. Стори – це сховища для моделі, вона дає можливість викликати заздалегідь відомі списки сутностей.

Компоненти сторінок – цей модуль реалізую всі компоненти графічного додатку, вся бізнес логіка яка потрібка для кожної сторінки також знаходиться тут. Всі компонент поділені на 3 частини: структура/шаблон(англ. template) сторінки; файл стилізації шаблону; бізнес логіка в скриптовому файлі.

* + 1. **Користувач “ Користувач з ідентифікаційною карткою”**

Звичайний користувач який маю ідентифікаційну картку доступу до системи, про яку знаходяться на головному сервері. Користувач користується карткою для взаємодії з пристроєм контролю доступу який потом оброблює інформацію про неї.

* + 1. **Користувач Адміністратор”**

Адміністратор системи відповідає за налаштування системи за допомогою графічного інтерфейсу. Його задача слідкувати за всіма пристроями контролю доступу та змінювати режими роботи при необхідності кожного пристрою.

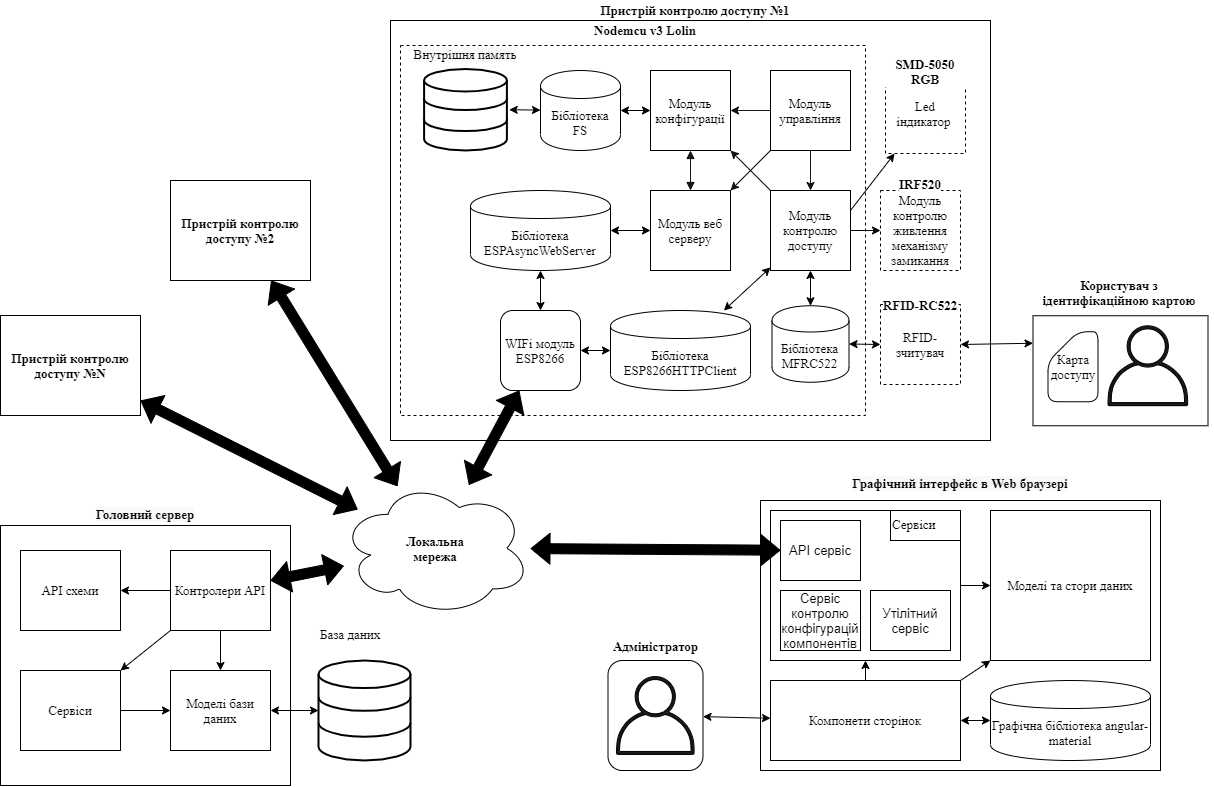


Рисунок 2.1 – Архітектура комп’ютерної системи контролю доступу з використанням ідентифікаційних карт

* 1. **Проектування апаратної частини комп'ютерної системи контролю доступу з використанням ідентифікаційних карт**

Система контролю доступу повинна мати деякі контролера або так званий пристрій контролю доступу. Ці пристрої повинні знаходитись біля дверей і будуть ними керувати, також задача такого пристрою стоїть в тому щоб прочитати ідентифікатор користувача і відправити його на сервер та обробити результат, виходячи з цього відкривати двері.

* + 1. **Вибір стандарту ідентифікаційних карток**

Для того щоб побудувати систему контролю доступу на базі ідентифікаційних карток нам необхідно вбрати стандарт або формат таких карток і визначити тип зчитувача для карток доступу.

Виходячи з аналізу є наступні види карток доступу:

* Магнітні
* Виганд
* Штрих-кодові
* Touch-memory
* Безконтактні (Proximity)

З аналізу було виявлено що на даний час най безпечнішими є Виганд картки, та proximity, інші картки такі як магнітні, Touch-memory та штрих кодові є не досить безпечними та дуже вразливі до зовнішніх факторів таких як механічні пошкоджена та навколишнє середовище, деякі з них дуже легко підробити. Виганд кратки дуже вартісні в розробці і кожна картка коштую дуже багато. Отже було вирішено використовувати proximity картки в користь іх відносної безпечності та дешевизни. Такі катки ще називають RFID картками, системи з цією технологією дозволять зчитувати інформацію які знаходяться поза межами видимості. Ідентифікаційний код зберігається в мітці, що складається з мікро чипу який прикріплений к антені. RFID мітки поділяються на активні і пасивні, різниця між цими мітками стоїть в наявності джерела живлення і передатчика [5]. Вони можуть працювати необмежений час, якщо тільки її не знешкодити механічними засобами, або піднести до дуже сильного електромагнітного поля на довгий час що приведе до знищення даних на картці.

Всі proximity картки поділяють на 4 категорії по частоті роботи [6][7]:

1. Low frequency (LF) – 100 – 150 кГц, всі виробники вибирають середнє значення частот, 125 кГц або 134 кГц, відстань зчитування для таких карток є дуже малим;
2. HF - 13,56 МГц стандартом таких карток є ISO 14443(стандарт, що описує частотний діапазон, метод модуляції і протокол обміну безконтактних пасивних карт [8])
3. UHF – 860 – 960 МГц, мітки з такою частотою мають найбільш дальню дистанцію реєстрації, мітки даного типу є дешевшими за LF та HF але інше обладнання для таких карток є значно дорожчим. В картках такого типу відсутній ідентифікаційний номеру, при створенні карток передбачалось що ідентифікаційним номером буде слугувати EPC (англ. Electronic Product Code) – номер товару, цей номер користувач повинен самостійно записувати в мітку
4. UHF Near-Field (з англ. – ближньої дії) – цей тип міток не є безпосередньо радіо міткою. Мітки використовують магнітні поля антен, це дозволяє вирішити проблеми зчитування при поганих природніх умов. Такий тип міток планується використовувати для фармацевтичних товарів, яким потрібен контроль справжності товари що мітки можуть зробити.

Proximity картки бувають 3-х типів [9]:

1. Em-Marine – всі дані таких карток записуються на заводу виробника, іх можна зчитувати але перезаписати такі картки неможливо. Ці картки є одним із перших зразків безконтактних карток і працюють вони на частоті 125 кГц.
2. Mifare – цей клас карт є дуже розповсюдженим в світі приблизно 90% всього ринку безконтактних смарт-карток, такі картки використовують для проїзних карток в громадському транспорті, також в якості дисконтних карток або як платіжні картки в супермаркетах. Mifare картки поділяються на 3 типи: Ultralight, Standard, Pro/Prox. Перший тип орієнтований для інтеграцію в папір, також в таких картках відсуне шифрування, приладом таких карток може бути одноразові квитки, разові пропуска, інше. Другий тип є стандартним типом який працює на частоті 13.56 та дальністю читання до 10 см., кожен сектор пам’яті захищений двома ключами. Останній тип орієнтований для систем де потрібний дуже великий рівень безпеки, всередині таких карток знаходяться декілька процесорів для шифрування даних за алгоритмом Triple-DES(симетричний блоковий шифр який розширює алгоритм DES, він суттєво підвищує безпеку але продуктивність і кількість використовуваної пам’яті збільшується), цей тип має всередині додатковий крипто процесор що дає можливість використовувати його для банківських карток.
3. HID iClass – цей тип є аналогом до Mifare Standart та Mifare Pro, але винятком для цих карток э те що вони працюють тільки з зчитувачами компанії HID.

Виходячи з існуючих типів безконтактних карток було вибрано картки proximity, а саме RFID картки типу Mifare які перцюють на частоті 13,56 МГц. Цей тип карток поєднує в собі дешеву вартість обладнання(зчитувачу), та низьку вартість самих RFID карток. Перевагою такого вибору є те що на системі з такими картками можна змінити зчитувач і розширити можливості самої системи додавши до пристрої ідентифікації смартфони з підтримкою NFC(бездротова технологія для передачі даний яка працює на невеликій відстані, ця технологія має широке використання в банківських терміналах[10]), так як цей інтерфейс використовую той же стандарт але розширена версія.

* + 1. **Вибір технічних засобів побудови апаратної частини системи контролю доступу**

Відповідно до аналізу вимог до системи було складено наступні вимоги до пристрою контролю доступу.

Пристрою контролю доступу пристрою повинен мати наступні модулі:

* Контролер – головна частина пристрою, за допомогою контролера буде оброблюватися вся інформація;
* Модуль доступу до локальної мережі – модуль який буду давати доступ до локальної мережі що дати можливість користуватися інтерфейсами TCP/IP;
* Модуль зчитування RFID-міток – модуль який відповідає за пристрій який повинен зчитувати інформацію з міток типу Mifare Standart з частотою 13,56 МГц;
* Модуль індикація – пристрій для інформування користувача з карткою про те отримав користувач доступ чи ні, потрібно 3 кольори – червоний, жовтий, зелений(відповідно доступ заборонено, відбувається опрацювання даних, доступ надано);
* Модуль відмикання механізму замикання – пристрій який повинен контролювати живлення для пристрою;
* Модуль окремого живлення для механізму замикання;
* Пристрій повинен живитися від джерела постійної напруги 5V.

Виходячи с таких вимог пристрій контролю доступу може буди спроектований за допомогою наступних пристроїв, модулів:

1. Контролер: Nodemcu v3 Lolin.

Це плата яка дає можливість розробляти проекти на базі чипу ESP8266 [11], який представляє з себе WIFI модуль який не дуже вимогливий до живлення, контролер зображений на рис. 2.2. Головна перевага цієї плата в тому що вона дозволяє спростити розробку для пристроїв які повинні мати доступ до мережі так як в неї вже вистроєний WIFI модуль та є внутрішня пам'ять для зберігання даних в розмірі до 4 Мбайт, а також є можливість підключати плату через USB для перепрограмування. Наступна перевага цієї плати є в тому що вона є повністю сумісною с усіма існуючими бібліотеками для Arduino, що дозволяє з легкістю програмувати цей контролер за допомогою вже існуючих засобів для програмування на Arduino. Отже цей модуль буду виконувати дві функції, оброблення інформаціїі і відправка даних за допомогою WIFI модулю.

Для програмування контролеру ESP8266 було мову програмування C++ так як ця мова дозволяє використовувати ООП для побудови системи, що спрощує розділення програмної частини на окремі рівні відповідальності.

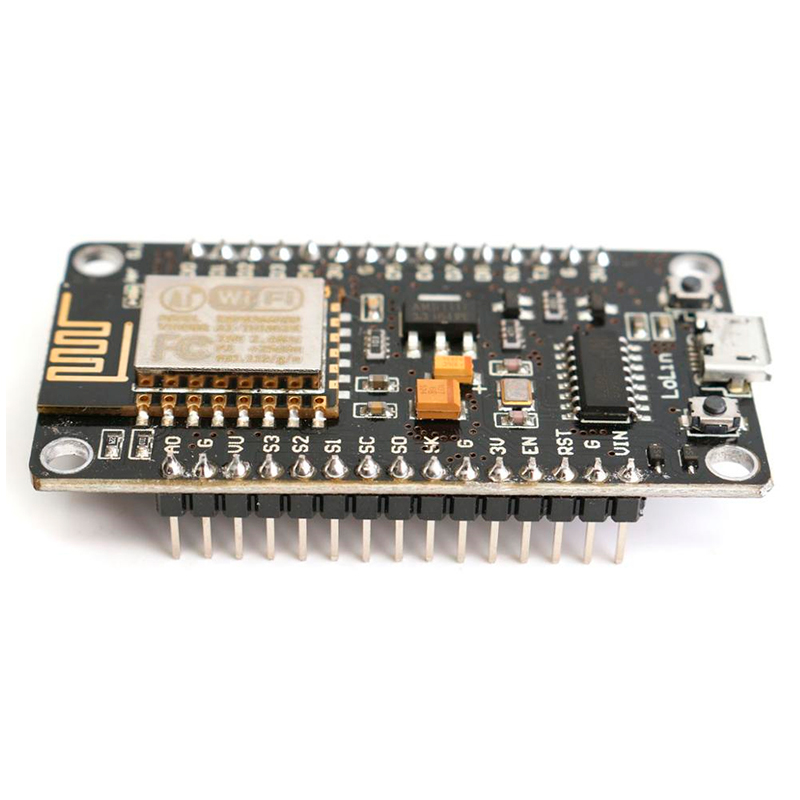


Рисунок 2.2 – Зображення контролер Nodemcu V3 Lolin

1. Модуль зчитування RFID-міток: RFID-RC522

Це модуль який дозволяє з легкістю зчитувати RFID-мітки та передавати данні до контролеру. Модуль випромінює електромагнітні хвилі певної частоти, при зчитуванні мітка відправляє у відповідь інформація про унікальний ідентифікатор, який має кожна мітка, також данні пам’яті та інші данні. Модуль зображений на рис. 2.3.



Рисунок 2.3 – Зображення зчитувача RFID-RC522

1. Модуль індикація: модуль з світлодіодом SMD 5050 RGB з спільним катодом

Цей модуль дозволяє отримати різнокольорове світло яке необхідно для індикації. Перевагами модулю є те що в нього все катоди виводяться в один і для повного з’єднання с контролером необхідно лише 4 сполуки, модуль зображений на рис. 2.4.

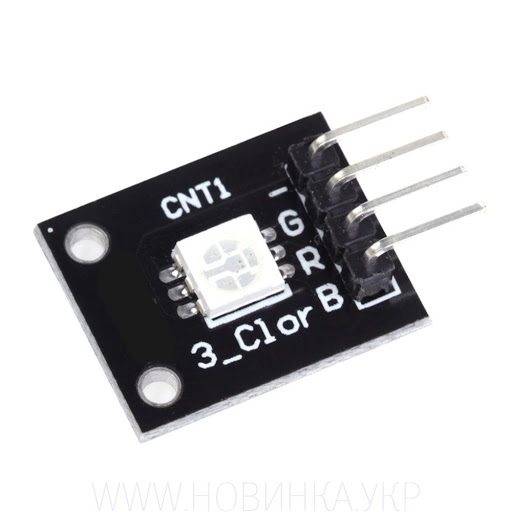


Рисунок 2.4 – Зображення світлодіодного модулю SMD 5050 RGB

1. Модуль відмикання механізму замикання: модуль Mosfet module IRF520

Модуль представляє з себе силовий ключ з польовим транзистором, який дозволяє контролювати живлення для пристроїв з окремим живленням високої напруги, модуль зображений на рис. 2.5.

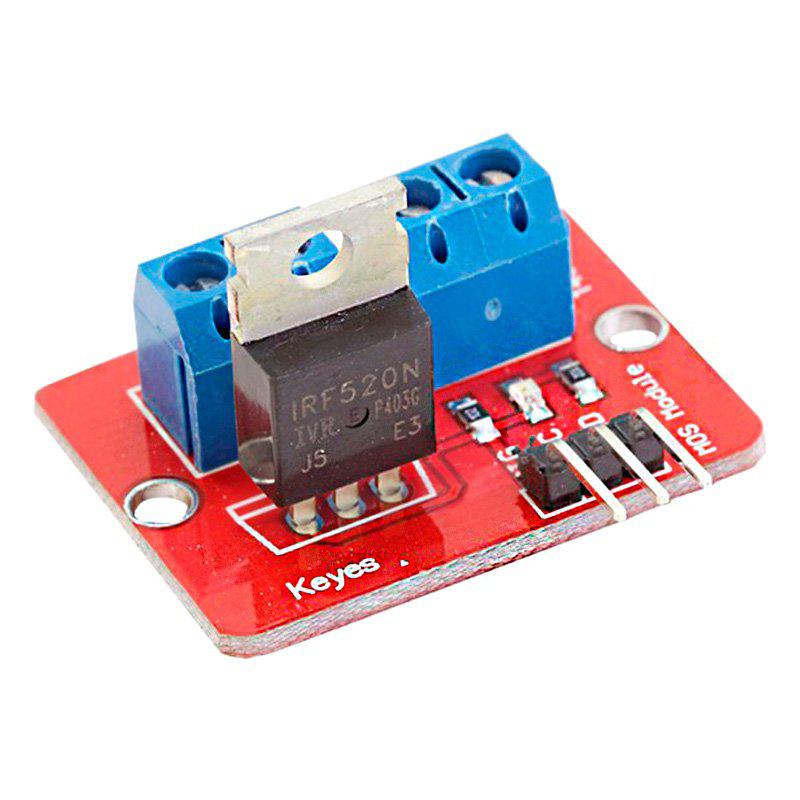


Рисунок 2.5 – Зображення модулю Mosfet module IRF520

1. Модуль окремого живлення для механізму замикання: модуль MT3608

Модуль представляє з себе регульований підвищувач напруги з вихідною напругою до 28В і максимальним струмом до 2А. Вхідна напруга завжди має бути нижче вихідного. Модуль зображений на рис 2.6.



Рисунок 2.6 – Зображення контролер модулю MT3608

* + 1. **Проектування схеми принципової**

Для проектування схеми було використано застосунок EasyEDA, Цей застосунок дає можливість з легкістю проектувати будь які схеми, пристрої які можна потім в схемах використовувати, а також цей застосунок дозволяє створити схему для друкованої плати де програма, якщо користувач захоче, може розмістити всі пристрої компактно.

Пристрій контролю доступу має 5 модулів які потрібно правильно з’єднати для правильної роботи кожного модуля, схема принципова пристрою контролю доступу зображена на рис. 2.7.

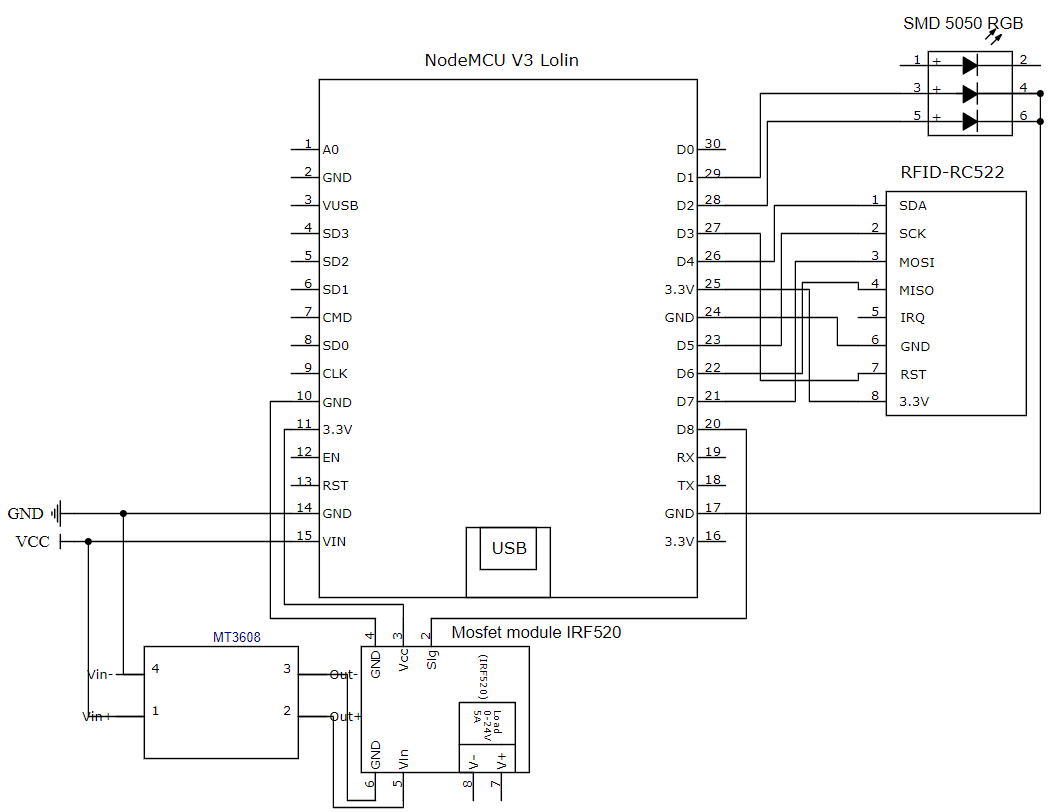


Рисунок 2.7 – Схема принципова пристрою контролю доступу

Пристрій Nodemcu v3 Lolin відповідає за оброблювання всієї інформації в пристрої, і він реалізує всі модулі оброблення інформації. Таблиця розташування використаних виводів та їх з’єднання с іншими пристроями наведено в таб. 2.1.

Виводи які не наведено в таблиці не несуть ніякої користі для проекту тому їх в не наведено.

Таблиця 2.1 – Таблиця розташування та з’єднання всіх виводів Nodemcu v3 Lolin, в таблиці наведено всі виводи які використовуються

|  |  |  |  |
| --- | --- | --- | --- |
| Вивід, пін-символ | Пристрій з який з’єднується | Вхід до якого підключається | Призначення |
| 10 – GND | Mosfet module IRF520 | 4 – GND | Живлення для зовнішнього модулю, 3.3V |
| 11 – 3.3V | Mosfet module IRF520 | 3 – Vcc |
| 14 – GND | Вивід ззовні | GND | Живлення, 5V |
| 15 – VIN | Вивід ззовні | VCC |  |
| 17 – GND | SMD-5050 RGB | – | Земля для катодів діода |
| 20 – D8 | Mosfet module IRF520 | 2 – SIG | Керування пристроєм, посилаючи цифровий сигнал високої чи низької напруги |
| 21 – D7 | RFID-RC522 | 3 – MOSI | Прийом даних від пристрою |
| 22 – D6 | RFID-RC522 | 4 – MISO | Пересилання даних які йдуть від основного пристрою |
| 23 – D5 | RFID-RC522 | 2 – SCK | Керування сигналом синхронізації |
| 24 – GND | RFID-RC522 | 6 – 3.3V | Живлення для зовнішнього модулю, 3.3V |
| 25 – 3.3V | RFID-RC522 | 8 – GND |
| 26 – D4 | RFID-RC522 | 1 – SDA | Контроль вибору керуючого пристрою |
| 27 – D3 | RFID-RC522 | 7 – RST | Керування скидання поточного стану |
| 28 – D2 | SMD-5050 RGB | 5 – G | Керування зеленим кольором світлодіода, з’єднується с анодом |
| 29 – D1 | SMD-5050 RGB | 3 – R | Керування червоним кольором світлодіода, з’єднується с анодом |

Пристрій RFID-RC522 виконую роль зчитувача і його задача зчитувати RFID мітки і передавати данні про мітку до головного контролера. Використання виводів наведено в таб. 2.1., вивід IRQ – відповідає за переривання, цей вивід не потрібен з причини що нам не потрібно керувати перериванням від цього пристрою. Підключення до головного контролера було зроблено згідно документації до вимог бібліотеки MFRC522 яка дає можливість керувати цим пристроєм[12].

Пристрій модуль SMD 5050 RGB є світлодіодом, для пристрою контролю доступу потрібно тільки 3 кольори для індикації, тож нам однозначно потрібен вивід для червоного та зеленого кольорів, в жовтий ми отримаємо шляхом одночасної активації двох кольорів.

Пристрій модуль MT3608 відповідає за підвищення вхідної напруги, і далі передача живлення на механізм живлення дверей через пристрій Mosfet module IRF520 який контролює живлення для дверей. Пристрій виконує підвищення напруги від 5V до 12V.

* 1. **Проектування програмної частини комп'ютерної системи контролю доступу з використанням ідентифікаційних карт**

Програмна підсистема виконаю задачу делегування і курування всією інфраструктурою системи контролю доступу. Програмна система поділена на 3 частини: 1) програмна підсистема апаратної частини; 2) програмна підсистема серверної частини; 3) програмна підсистема графічної частини 4) база даних. Взаємодія всіх елементів програмної підсистеми зображена на рис. 2.8.

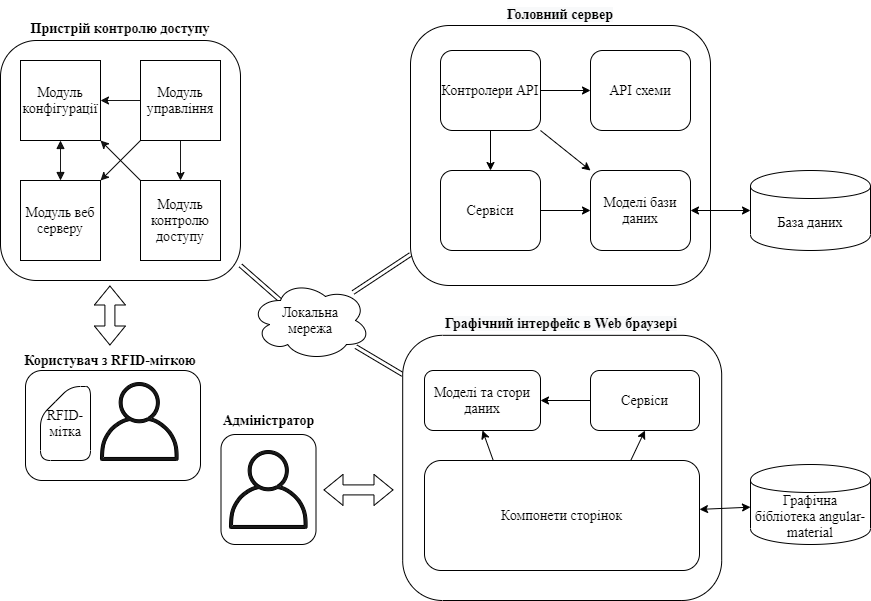


Рисунок 2.8 – Структура програмної частини

* + 1. **Програмні засоби для побудови апаратного застосунку**

Однією із перши завдань перед початком створення програмної частини для апаратної частини є вибір мови програмування та вибір середовища розробки, яке буде зручним в застосуванні. Більшість прошивок(скомпільований машинний код програми) не пишуть самостійно, спочатку його пишуть на мовах високого рівня програмування такі як C, C++ або на скриптових мовах програмування, таких як JavaScript, Python, Lua. Скриптові мови програмування є більш простими але програми на таких мова працюють повільніше ніж програми на мовах програмування C/C++. Отже для написання програмного коду було вирішено використовувати мову програмування C++, так як вона підтримую ООП що дозволить розділити код на більш структуровані модулі.

Розробка програмного застосунку проводилась в текстовому редакторі Visual Studio Code, цей редактор є дуже зручний підтримує безліч мов програмування. Одним із плюсів цього редактору є те що в нього є можливість докачувати та встановлювати додатково в редактор так звані extensions (з англ. розширення), які дозволяють дуже сильно розширити функціональність редактора. Одним із таких розширень є PlatformIO, цей додаток надає можливість створювати програми для контролерів, докачувати відразу в редакторі потрібні бібліотеки через вбудований пошук. Після підключення контролеру до комп’ютера або після налаштування оновлення прошивки через повітря(технологія що не потребую прямого підключення пристрою до комп’ютера, необхідно тільки підключити пристрій до локальної мережі).

Для взаємодії з зчитувачем була використана бібліотека MFRC522, яка реалізує такі функції як перевірки наявності нових карток та зчитування унікального ідентифікатора картки та її внутрішніх даних.

Для побудови локального веб серверу була використана бібліотека ESPAsyncWebServer, ця бібліотека працює разом з іншими бібліотеками, такими як ESPAsyncTCP і ESPAsyncWiFiManager. Додаткові бібліотека ESPAsyncTCP слугує для реалізації базових функцій для створення асинхронних запитів, а ESPAsyncWiFiManager слугує заміною синхронного WiFi менеджеру, так як він не дає можливість працювати асинхронному веб серверу. Ці бібліотеки надають зручний спосіб створення асинхронного веб серверу, який буде працювати паралельно, при появи нового запитів головний потік не буде призупинятися. Також перевага асинхронного серверу в тому що він надає можливість оброблювати декілька клієнтів одночасно.

Для виклику API запитів на головному сервері використовується стандартна бібліотека для мережевих модулів ESP8266 така як ESP8266HTTPClient, вона реалізує можливість створення HTTP клієнтів які дозволяють виконувати GET, POST, PUT та інші виклики до серверу.

Для зчитування внутрішнього конфігу використовується такі бібліотеки як ArduinoJson, FS. Перша бібліотека потрібна для того щоб структурувати всі дані в JSON(текстовий формат для структурування даних) документ і після записати його в файл, відновлення даних проходить також за допомогою бібліотеки ArduinoJson.

Друга бібліотека потрібна для ініціалізації потоку зчитування даних с внутрішньої пам’яті, а також для самого процесу зчитування даних з файлу.

* + 1. **Технічні засоби для побудови серверного застосунку**

Виходячи з вимог для застосунку було вибрано асинхронне Nodejs оточення яке з легкістю дозволяє створювати масштабні застосунки [13]. Екосистема цього середовища дуже велика і це дуже спрощує створення застосунків, так як більшість інструментів вже є в наявності і не потрібно реалізовувати самостійно їх. Nodejs дозволяє створювати застосунки за допомогою мови javascript, але ця мова э не суворо типізованою мовою, а також можна з легкістю допустити помилки при форматуванні коду. Тож для мови написання було вибрано мову програмування typescript. Ця мова програмування є строго типізованою, можна створювати застосунки які базуються на ООП принципах, в порівнянні з javascript в цій мові можна створити біль жорстку архітектуру яка спростить тестування модулів проекту. Ще одна перевага э те що ця мова э тільки настройкою над js це дозволяє застосовувати всі бібліотеки які все є на js.

Після вибору оточення треба вибрати фреймворк для створення веб додатку, було вибрано фреймворк fastify. Цей фреймворк має великі переваги перед іншими, він є дуже простим, а також він є одним із найшвидших фреймворків по швидкості оброблянні запитів і нарешті в нього велика бібліотека додаткових плагінів які можна використовувати разом з ним[14]. Як настройка під цей фреймворк була вибрана бібліотека fastify-decorators яка дозволяє дуже зменшити написання повторюваного коду використовуючи експериментальну функціональність декоратори (спосіб мета-програмування який дозволяє додати деяку функціональність на етапі виконання скриптів до класів або методів [15]) та зосередитися на написанні корисного коду. Ця настройка використовує шаблон проектування “ланцюжок відповідальностей” [16].

Веб сервер для керування системою контролю доступу потребую зберігання інформації про пристрої якими він керує та інформацію про самі ідентифікаційні картки. Для цього було вибрано sql базу даних Mysql 80, це звичайна база даних яка дозволяє зберігати звичайну інформацію та робити зручні вибірки для отримання даних.

Для комунікації серверу з базою даних потрібно створити деякий рівень взаємодії з нею. Щоб спростити реалізацію запитів до бази даних, а також автоматизувати всі дії з нею було вибрано Sequelize ORM (Object-Relational Mapping – з англ. мови перекладається як об'єктно-реляційна проекція, технологія або метод доступу к реляційній базі даних з ООП мови, в якій для того щоб отримати доступ до даних треба створити клас який буду описувати модель таблиці, після цього ми зможемо з легкістю робити всі звичні операції без написання ручних sql запитів), ця бібліотека базується на так званих “Promise” які є одною із основних частин Nodejs середовища та мови програмування javascript, також ця бібліотека має гану підтримку транзакцій, відношень таблиць, пряме завантаження та ледаче завантаження даних і багато іншого [17]. Для кращої взаємодії бібліотеки Sequelize та мови програмування typescript було вибрано використовувати бібліотеку sequelize-typescript, він дозволяє створювати моделі таблиць бази даних описуючи їх в класах за допомогою декораторів [18].

Для розробки серверного додатку була використана середовище розробки IntelliJ IDEA яка розроблена компанією JetBrains, це середовище дуже зручне, дозволяє створювати різноманітні задача, наприклад запустити сервер або інше, які можна викликати з панелі управління. Також це середовище має всередині себе можливість підключитися до бази даних и бачити всі таблиці, та робити вибірки з бази даних. Для typescript/javascript є вистроєні допоміжні плагіні, аналізатори коду, які допомагають при розробці застосунків, також є інтеграції з такими аналізаторами коду як tslint та prettier, ці програми аналізують код та форматують стиль коду згідно правил що користувач налаштує або візьмуть значення за замовчування і будуть форматувати код згідно зашитих в ним стандартів. Під час розробки були використані ці дві програми для форматування коду, і в завершені було отримано кодову базу яка відформатована в одному стилі і яку легко читати.

* + 1. **Технічні засоби для побудови графічного інтерфейсу**

Графічний інтерфейс є одною із головних елементів сучасних інфраструктур і для розробки графічної системи треба вибирати інструменти які дозволять з легкістю масштабувати додаток, підтримувати та тестувати. Для створення WEB додатку будо використано фреймворк Angular(це середовище для розробки та платформа для розробки одно сторінкових додатків [19]). Графічні елементи інтерфейсу такі як кнопки, поля вводу, панель навігації та інші компоненти взяті з бібліотеки графічних компоненті angular-material [20], вона має багато специфікованих компонентів поведінка яких відповідає вимогам Material Design(візуальна мова яка об’єднує класичні принципи доброго дизайну с внесками інноваційних технологій і наук[21]).

Мова програмування для веб інтерфейсу - typescript, вибір цієї мову був зроблений з цілю що серверний додаток буде писатися на цій мові і виходячи з цього підтримка та реалізація буде спрощена тим що мови однакові, а також можна робити модулі які будуть однакові як для серверного так і для графічного застосунків.

Для створення графічної підсистеми як і в випадку серверного додатку було використано середовище розробки IntelliJ IDEA, для спрощення розробки обох застосунків, і це дозволило редагувати та тестувати обидва застосунки одночасно що значно пришвидшило розробку, Для графічної кодової бази було також використано аналізатори коду tslint та prettier так як обидві частини були написані на мові програмування typescript.

* + 1. **Проектування структури бази даних**

Перед початком проектування структури бази даних необхідно визначити що за сутності повинні бути в системі. Система контролю і управління доступу з використання ідентифікаційних карт повинна мати наступні сутності які повинні мати наступну інформацію:

* Інформація про пристрої контролю доступу
* Інформація про карти доступу
* Інформація про ролі доступу
* Інформація про доступу конкретної картки до конкретного пристрою
* Інформація про дії в системі

Система контролю та управління доступу повинна мати захист від сторонніх осіб, тож в системі повинно бути реалізовано авторизації. Також було вирішено створити можливість створювати сесії, яка представлена унікальним ключем, що кожного разу графічному застосунку не потрібно було запитувати у адміністратора логін та пароль, отже для збереження такої інформації потрібно ще дві сутності які містять в собі наступну інформацію:

* Інформація про адміністраторів/користувачів
* Інформація про сесії

Для збереження інформації було вирішено використовувати реляційну базу даних MySQL, яка дозволяю масштабувати збереження даних, ця база даних має вільну ліцензію користування, вона є зручною, з підтримкою реплікацій та багато іншого.

На основі потрібних сутностей було спроектовано наступну базу, схема якої представлена на рис. 2.9.

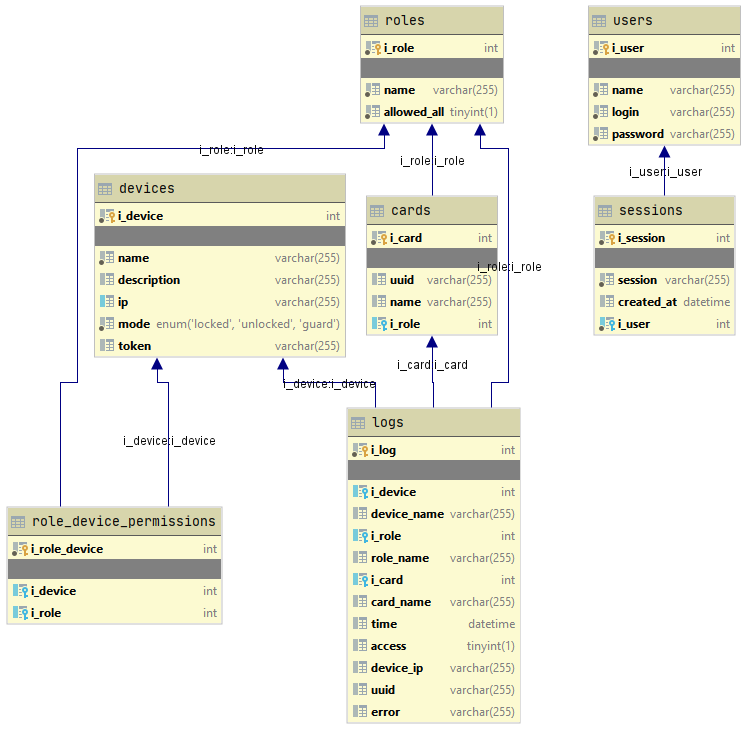


Рисунок 2.9 – Схема бази даних системи контролю та управління доступу з використання ідентифікаційних карт

* + 1. **Програмне забезпечення пристрою контролю доступу**

Основана задача програмного забезпечення апаратного пристрою контролю доступу полягає в обробленні даних з зчитувача, відсилання даних на головний сервер для перевірки, опрацювання HTTP запитів до пристрою, зчитування та запис даних в пам'ять пристрою.

Для зручності проектування програмної частини було реалізовано 4 класи, які повністю відповідають архітектурі пристрою контролю доступу. Для відображення функціоналу класів для відображення залежностей було використано діаграму класів, вона зображена на рис. 2.10.

На цій діаграмі зображено 4 класи:

* Клас контролю доступу, він відповідає модулю контролю доступу
* Клас сховище для конфігурації, він відповідає модулю конфігурації
* Клас веб серверу, він відповідає модулю веб серверу
* Клас основний, він відповідає модулю управління

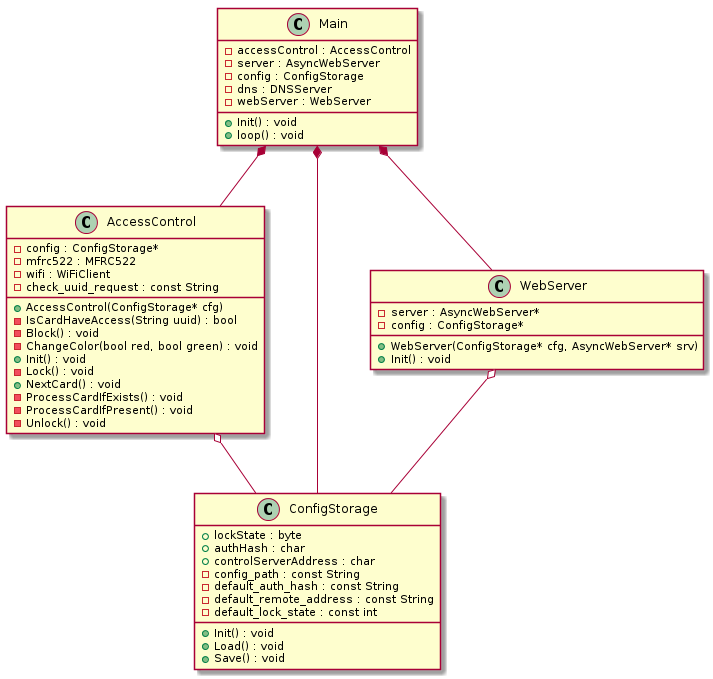


Рисунок 2.10 – Зображення діаграми класів пристрою контролю доступу

Кожний клас має функцію Init яка слугує для ініціалізації інтерфейсів контролера Nodemcu v3 Lolin які потрібні для правильного функціонування цих класів.

Виходячи с того що користувач може отримати доступ чи ні потрібно страктувати алгоритм опрацювання картки і подальшу роботу пристрою. Спроектований алгоритм зображено на рис. 2.11.

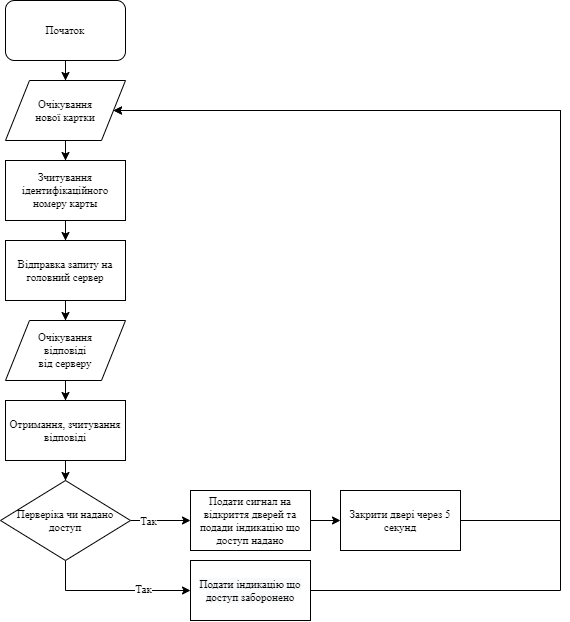


Рисунок 2.11 – Схема алгоритму опрацювання RFID-міток

Пристрій працює в 3-х режимах роботи, коли пристрій працює в звичайному режиму, коли він не повинен опрацьовувати картки і коли він повинен завжди давати доступ, тобто двері будуть завжди відкриті, для позначення цих режимів було вибрано.

* Режим “GUARD” (охорони): режим при якому модуль працює в звичайному режимі опрацюванні карток та відкривання механізму відмикання виходячи з відповіді серверу.
* Режим “LOCKED” (завжди замкнено): режим при якому модуль не буде зчитувати картки, механізм замикання знаходиться в замкненому режиму режимі, тільки ручне відкриття механічним ключем дозволено(якщо механізм замикання підтримує таку можливість)
* Режим “UNLOCKED” (завжди відкрито): режим при якому пристрій постійно подає живлення до механізму замикання, що приводить до постійних відкритих дверей.

Пристрій має конфігураційний файл який зберігається в внутрішній паняті пристрою, в таб. 2.2 наведено всі поля конфігурації, тип цих полів, опис та значення за замовчуванням. Як же було сказано в розділі 2.3.1, файл конфігурації це файл формату JSON, назву для файлу було вибрано config.json, для роботи з бібліотекою ArduinoJson потрібно заздалегідь підрахувати максимальну кількість пам’яті яка може знадобитися для завантаження конфігу в пам'ять. Для цих підрахунків був використаний калькулятор на сайті бібліотеки.

Таблиця 2.2 – Структура даних файлу конфігурації пристрою контролю доступу

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип даних | Значення за замовчуванням | Опис |
| token | char[36] | 00000000-0000-0000-0000-000000000000 | Токен який використовується для авторизації запитів до внутрішнього веб серверу. |
| server\_address | char[255] | 192.168.1.2:3000 | Адрес де знаходиться головний сервер, використовується при перевірці карток доступу |
| mode | byte | 0 | Режим в якому працює пристрій контролю доступу, для економії пам’яті режим зберігається як число, 1 – режим “LOCKED”, 2 – режим “UNLOCKED”, 3 – режим “GUARD”. |

* + 1. **Проектування серверної частини**

Структура серверного додатку розмежована на 6 модулів:

* Контролери API (controllers)
* Моделі бази даних (models)
* Схеми API (schemas)
* Сервіси (services)
* Спільні класи\інтерфейси (shared)
* Сервер (server)

Додаток реалізує клас Server який з’єднує всі модулі з фреймворком fastify і створює всі необхідні залежності для роботи додатку, цей клас повністю відповідає останньому модулю додатку. Діаграма класів проекту зображена на рис. 2.12. На діаграмі зображені всі класи 6-и модулів додатку вона дозволяє побачити взаємодії кожного з цих рівнів (поля та методи кожного класу упущено).

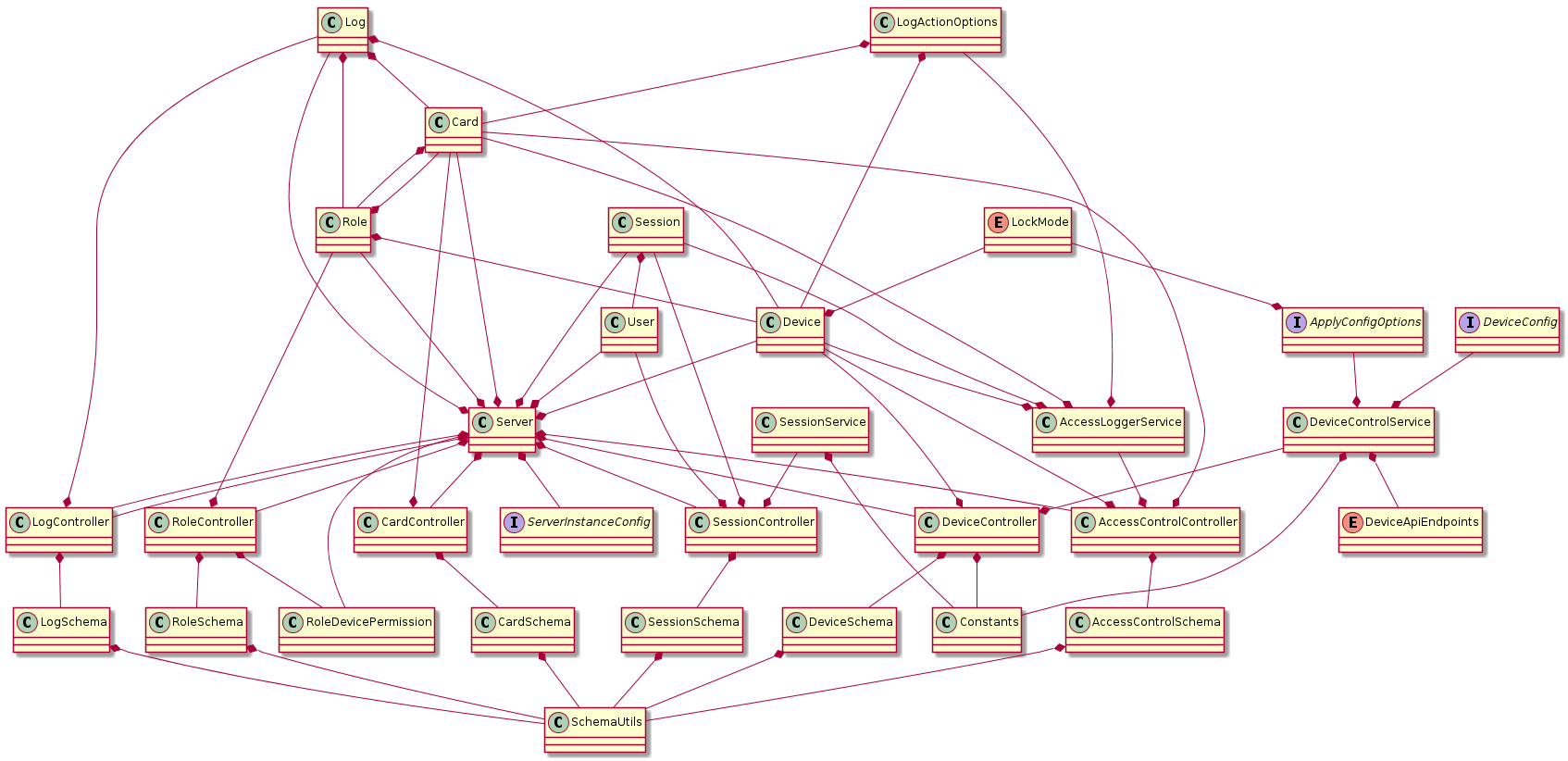


Рисунок 2.11 – Діаграма класів серверного застосунку

Модуль контролери API реалізує класи, які призначені для опрацювання всіх запитів до серверу. Кожен контролер відповідає певній сутності додатку. Наприклад для керування сутності Device слугує контролер DeivceControler. Діаграма класів контролерів та їх взаємодія з іншими класами зображена на рис. 2.12.

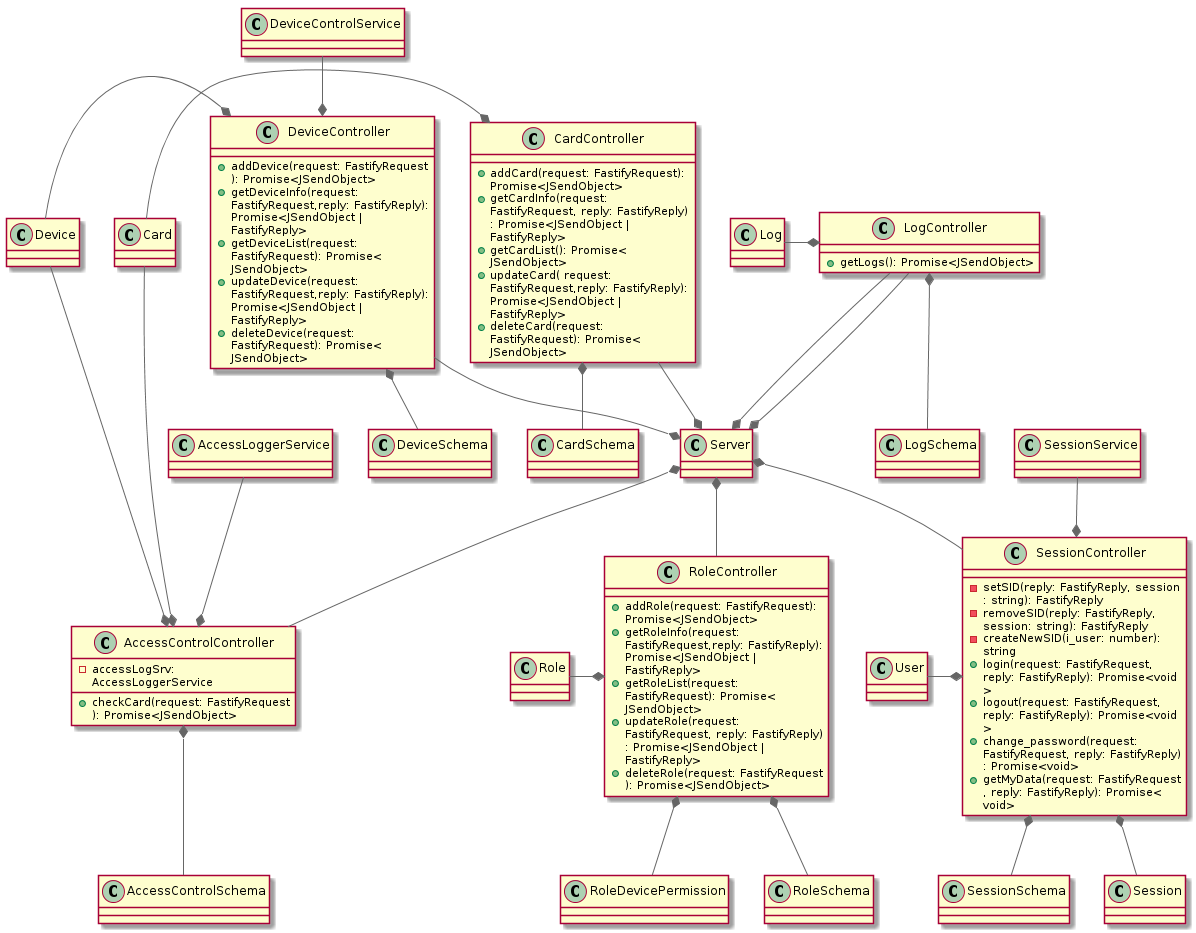


Рисунок 2.12 – Діаграма класів для модуля контролери API

Для обслуговування клієнта, а саме контролери які реалізують CRUD операції над сутностями бази даних, використовуються наступні контролери:

* CardController
* DeviceController
* LogController
* RoleController
* SessionController

Для обслуговування пристроїв контролю доступу існує контролер AccessControlController, який реалізовую лише метод для перевірки доступу до пристрою. Для цієї перевірки необхідно знати данні про пристрій, ці данні представлені в виді IP адресу відправника. Потім цей адрес перевіряється з адресами в систему і якщо його знайдено то ми можемо бути впевненими що запит прислав нам пристрій контролю доступу і ми можемо продовжити операцію перевірки, слідуючи параметр які необхідно дізнатися – це данні про карту доступу, інформацію про роль та доступні їй пристрої, потом перевірити чи має доступ картка до пристрою. На рис. 2.13 зображено алгоритм перевірки картки.

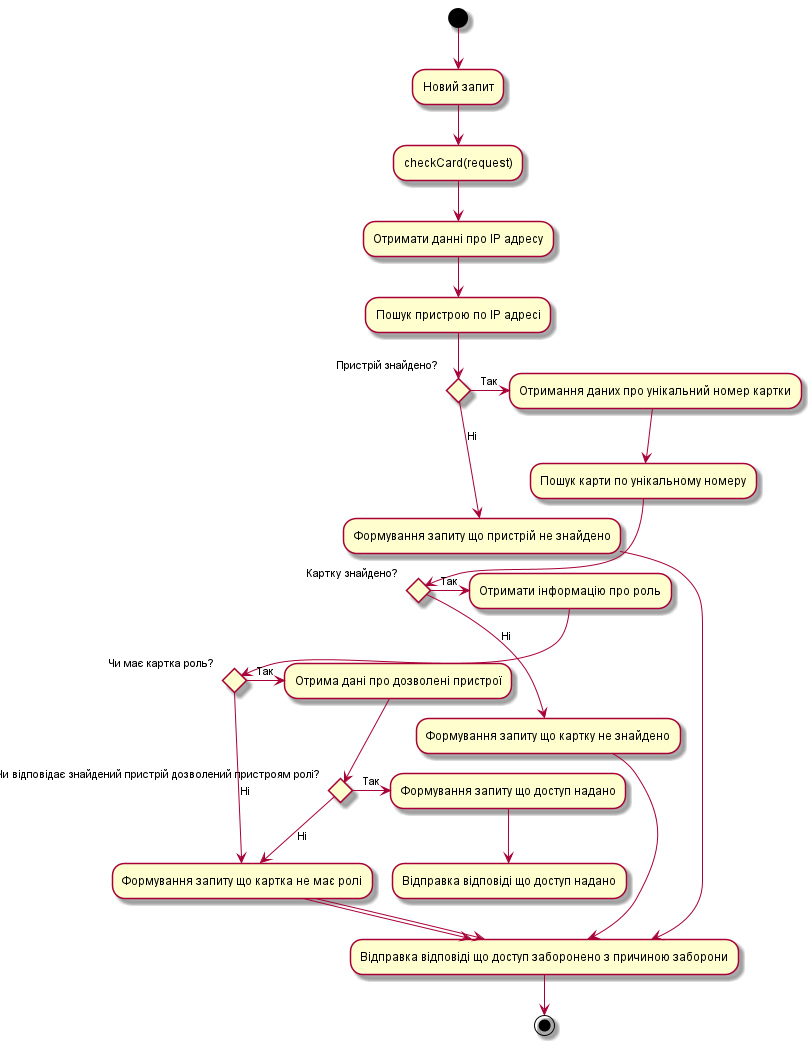


Рисунок 2.13 – Алгоритм перевірки доступу картки до пристрою контролю доступу

Для взаємодії з базою даних слугує модуль “Моделі бази даних”. В ньому реалізовані всі схеми моделей бази, які використовуються для зручної взаємодії базою. Всі моделі реалізують наслідують базовий клас моделі фреймворку Sequalize ORM. На рис. 2.14 зображено діаграму класів для модуля моделі бази даних.

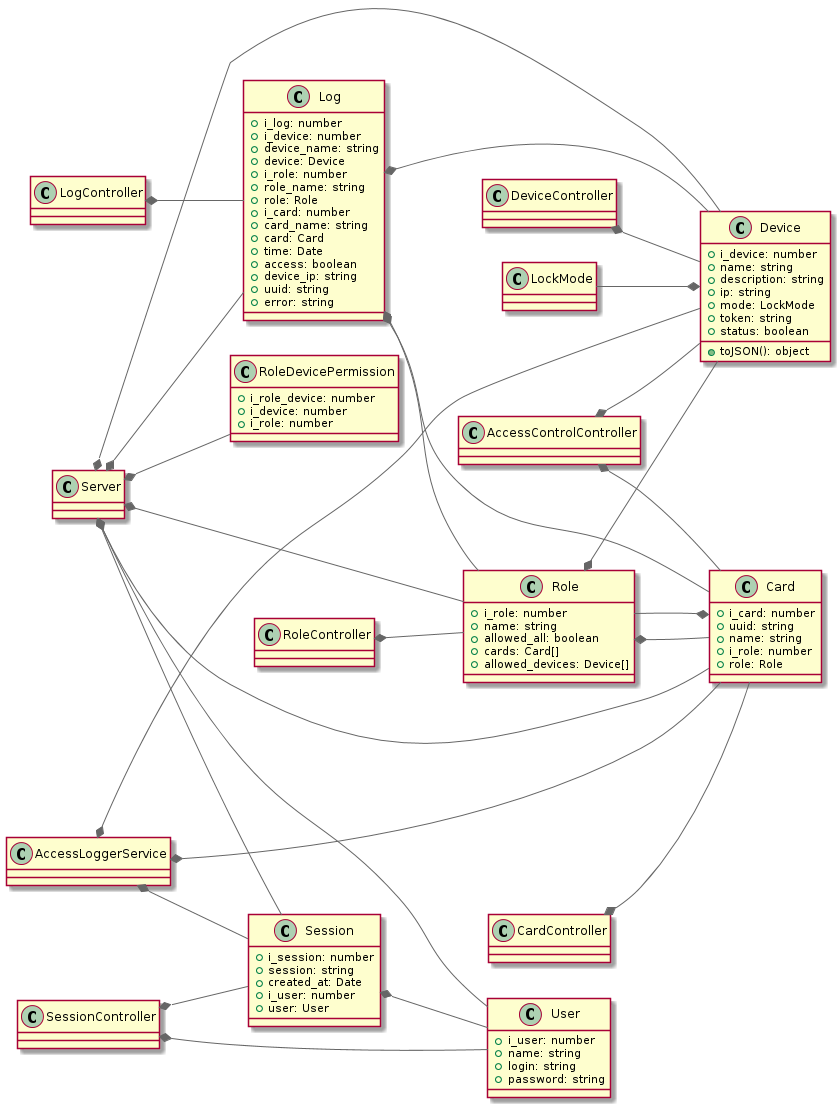


Рисунок 2.14 – Діаграма класів модуля моделі бази даних

Для варіації та для фільтрації даних при взаємодії з клієнтом, використовуються схеми API. Вони описують всі API методи які є в додатку та їх структуру, що приймає метод та що повертає. На рис. 2.15 зображена діаграма взаємодії класів модуля схеми API та інших модулів.

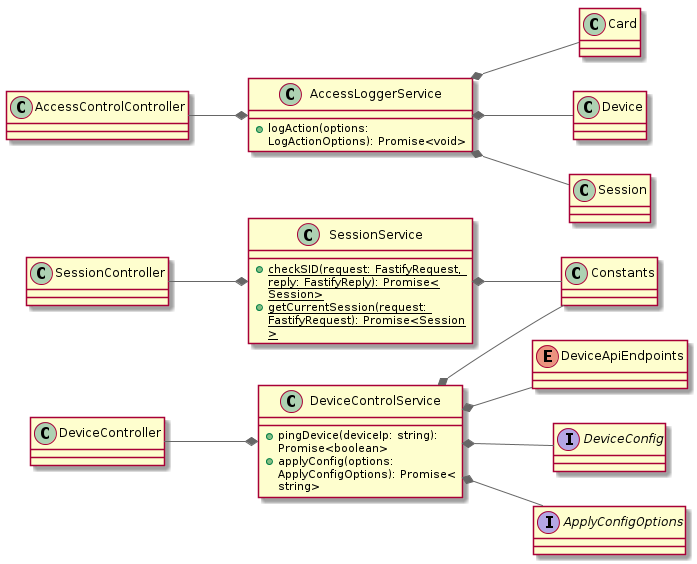


Рисунок 2.15 – Діаграма класів модуля сервіси

Для централізованого збереження спільних типів, інтерфейсів констант слугує модуль Спільні класи\інтерфейси. Діаграма класів цього модуля зображено на рис. 2.16.

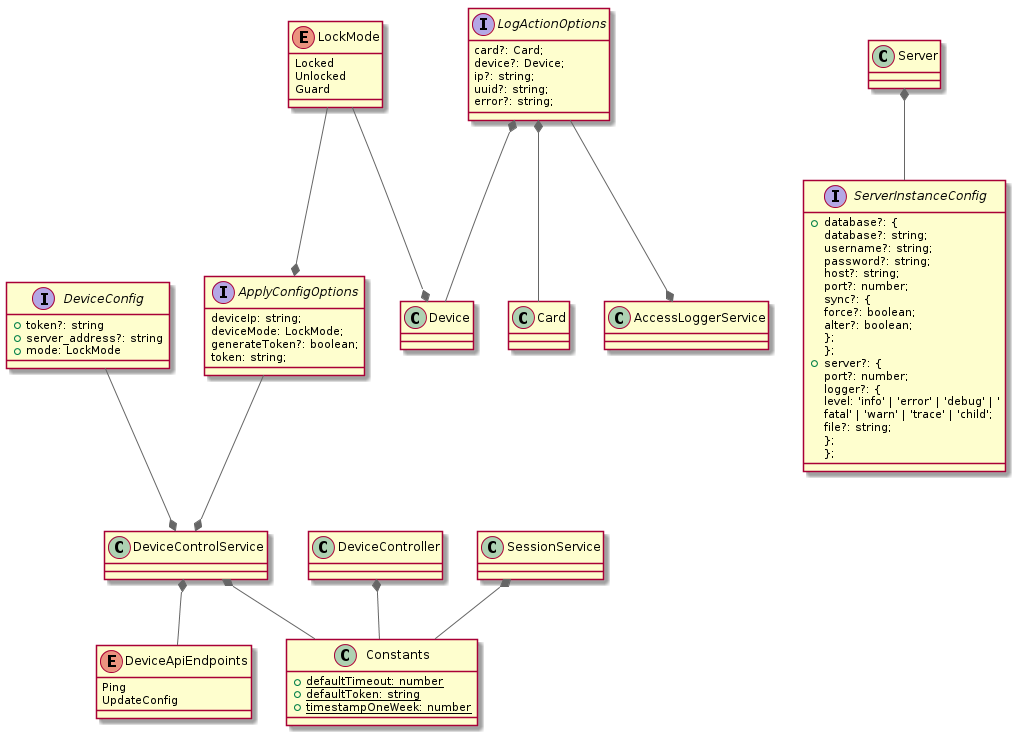


Рисунок 2.16 – Діаграма класів спільного модуля з типами та константами

Для запуску серверу та для підключення всіх необхідних фреймворків, бібліотек, плагінів, контролерів та іншого слугує модуль серверу, який має тільки один клас з реалізацією ініціалізації застосунку та запуску його. На рис. 2.17 зображено діаграму класів модуля серверу.

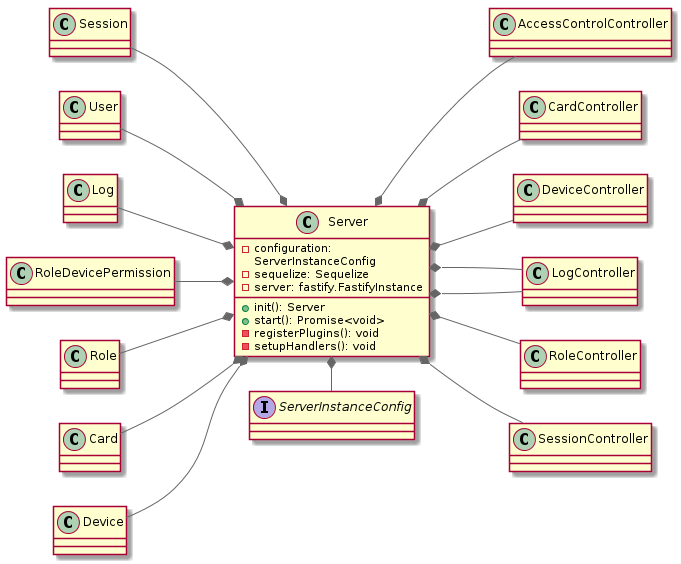


Рисунок 2.17 – Діаграма класів модуля сервер

* + 1. **Проектування клієнтської частини**

Клієнтська частина побудована за стандартними принципами створення графічних додатків за допомогою Angular фреймворку. В додатку ієну головний модуль, який підключає всі компоненти (AppModule) і головний компонент (AppComponent) вони э коренем для додатку. Для маршутиризації використовується ангулярівський Router який допомагає з легкістю створити всю модель додатку і налаштувати так звані Guards які відповідають за охорону сторінок, завдяки їм можна робити деякі перевірки доступу и не давати доступу до сторінок, або перенаправляти користувачів до іншої сторінки. Графічний додаток розмежований на 3 рівні на рис. 2.18 зображена діаграма модулів проекта разом з самими компонентами:

* Головний рівень або рівень програми, в ньому відбувається підключення всіх компонентів і сервісів.
* Рівень компонентів, в цьому рівні знаходяться всі компоненти сторінок їх основана бізнес логіка
* Рівень утилітних класів, на цьому рівні реалізуються всі допоміжні класи, абстрактні та класи моделей та стор.

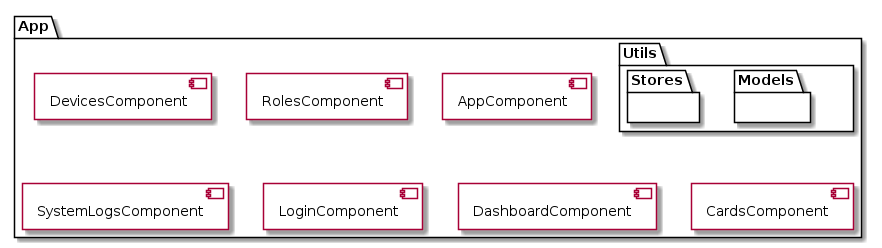


Рисунок 2.18 – Діаграма модулів та компонентів додатку

Діаграма класів для головного рівня зображена на рис. 2.19.

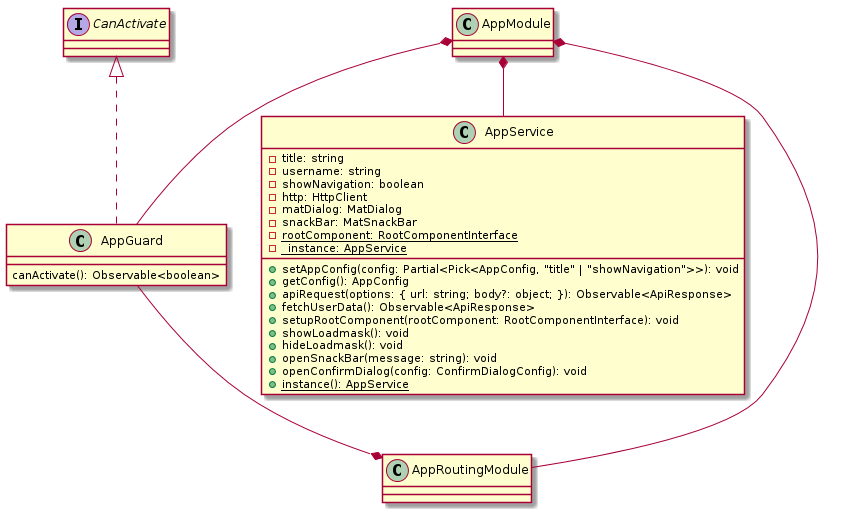


Рисунок 2.19 – Діаграма класі для головного рівня додатку

Клас AppModule слугує для того для підключення всіх модулів, сервісів, компонентів в проекті та підключення сторонніх бібліотек.

Клас AppRoutingModule описує маршутиризацію додатку, який компонент повинен бути відкрити для конкретної адреси, та які дані потрібно передати для кожного компонента якщо це потрібно.

Клас AppGuard необхідний для запобігання відкривання сторінок без авторизації, якщо перевірка була неуспішної, користувач повинен бути перенаправленні на сторінку логіну.

Клас AppService реалізує збереження конфігурації додатка, також в цьому сервісі реалізується метод для взаємодії з сервером та деякі методи для зміни глобального стану додатку.

Діаграма класів компонентів зображена на рис. 2.20. Кожен компонент є незалежний окрім AppComponent, так як він є головним і завжди існує, всередині цього компоненту рендеряться всі інші компоненти.

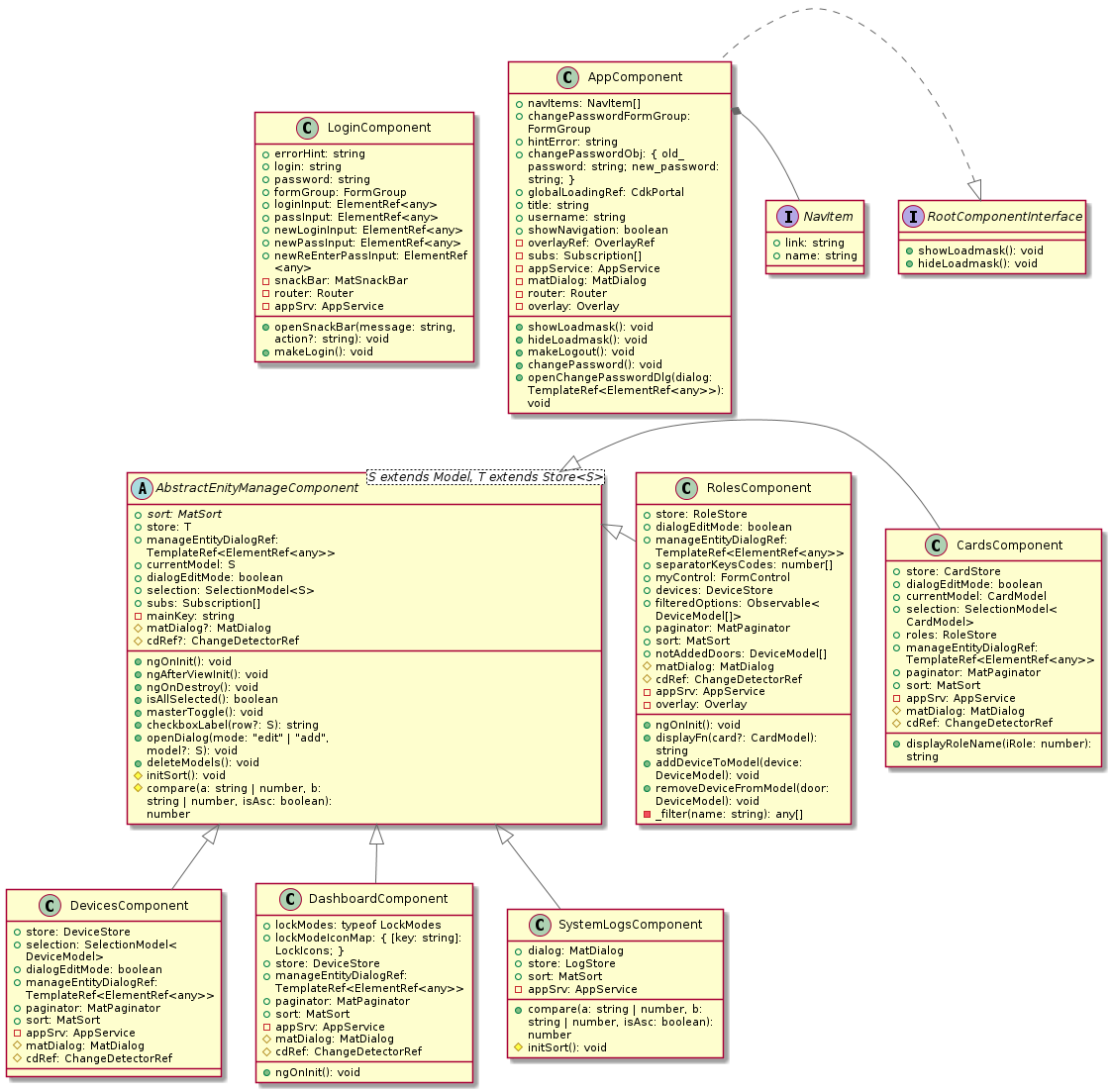


Рисунок 2.20 – Діаграма класів рівня компонентів

Клас AppСomponent – цей компонент відповідає за відображення головної навігації на інтерфейсі, представляє можливість для зміни паролю системи, та можливість вийти з системи.

Клас LoginComponent – реалізує сторінку логіна, на які адмін повинен ввести свій логін та пароль, після успішно вводу паролю, сесія зберігається в браузері і користувач перенаправляється на головну сторінку.

Абстрактний клас AbstractEnityManageComponent – цей клас реалізує основні операції на сутностями, завантаження списків, додавання нової сутності, редагування, видалення сутностей, також цей компонент задає стандартні налаштування для таблиці як є головним компонентом на кожній таблиці.

Клас DashboardComponent цей компонент слугує для відображення статусу системи, цей компонент має таблицю яка відображає всі пристрої в системі та показує їх статус. Також в цьому компоненті реалізована взаємодія з режимом роботи пристрою.

Клас SystemLogsComponent цей компонент призначений для відображення останніх подій в системі, відображення всіх успішних та невдалих спроб авторизації карток доступу.

Класи RolesComponent, DevicesComponent, CardsComponent реалізують управління над головними сутностями системи (Role, Device, Card відповідно), всі класи унаслідуються від абстрактного класу і всередині кожного задаються налаштування для взаємодії з сутністю.

В додатку реалізований модуля утиліт, в якому реалізовані основні інтерфейси, типи, для взаємодією з сутностями сервера реалізований спеціальні класи модель і стора. Задача моделі описати всі можливі поля та описати схему API, та шляхи до методів оновлення, видалення, додавання та отримання інформації. Задача стори полягає в зберіганні масиву моделей, всередині стор описуються шлях до отримання списків даних, після отримання даних кожен елемент перетворюється в повноцінну модель, якою можна керувати. Діаграма класів модулю утиліт зображена на рис. 2.21.

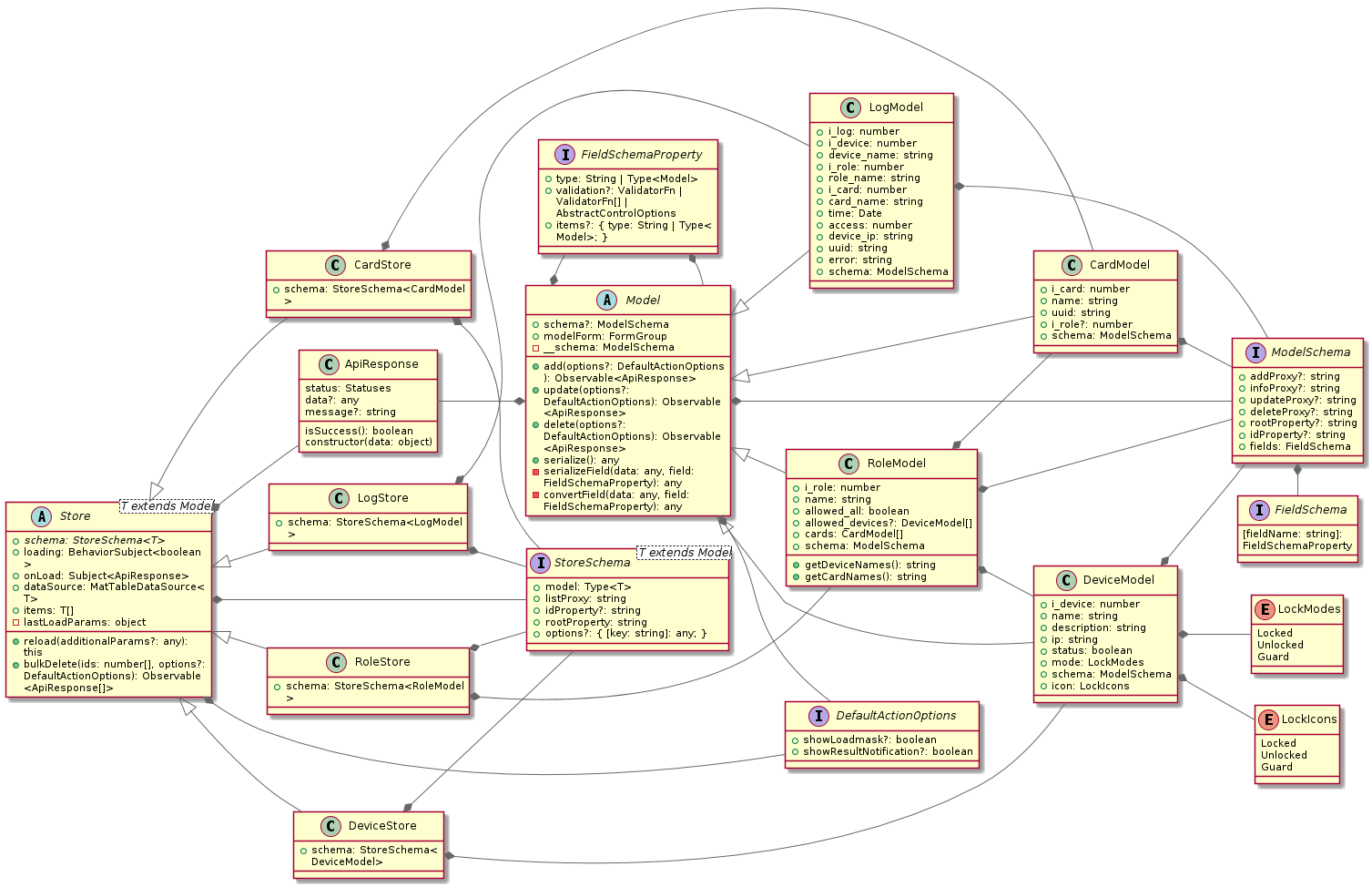


Рисунок 2.21– Діаграма класів утилітного модуля

В розділі 2 було вибрано RFID стандарт proximity карти доступу так як вона э більш-менш захищеною від перепрограмування, і через те що кожна така картка є дешевої і пристрої для зчитування цієї картки є більш дешевими порівняно з іншими пристроями і такі пристрої можна сховати за деякою тонкою стінкою так як зона зчитування через не металеві перешкоди приблизно 3-10см.

Для розробки системи були вибрані мови програмування для кожної частини. Для графічної частини було вибрано мову програмування typesctipt так як для головного графічного фреймворк Angular неохідно використовувати цю мову для кращої взаємодії з ним. Для серверного частини було також вибрано мову програмування typescript щоб спростити створення всієї системи так як графічна частина базується на цій мові, а також для того щоб додати деякої типізації в додаток. Для апаратної програмної частини було вирішено використати мову програмування C++ так як на цій мові дуже бага бібліотек і ця мова підтримую ООП програмування.

Також були вибрані всі бібліотеки, фреймворки так середовища розробки для кожної частини системи.

Була створена базова архітектура додатку яка показує взаємодію всіх модулів в системі і було розроблено діаграми класів і основні алгоритми для всіх програмних частин.

1. **РЕАЛІЗАЦІЯ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ З ВИКОРИСТАННЯМ ІДЕНТИФІКАЦІЙНИХ КАРТ**

В цьому розділі описана реалізація комп’ютерної системи контролю доступу з використанням ідентифікаційних карт, яка базується на зробленому аналізі і проектування що виділило основні вимоги до кожної частини системи.

* 1. **Реалізація апаратної підсистеми**

В підрозділі описана реалізація програмна частина апаратної системи, також представлено результат розробки фізичного пристрою контролю доступу і демонстрації кожного режиму роботи.

* + 1. **Реалізація програмної підсистеми**

Головна відмінність програм для мікроконтролерів типу Arduino в тому що в таких програмах не існує головного класу с методом main замість цього в програмах є два методи для головного main фалу, це setup і loop. Для організації коду апаратної частини було вирішено створити 4 класи, один головний класс Main який буде налаштовувати всю систему і інші допоміжні, такі як клас для керування конфігурацією ConfigStorage, клас для керування внутрішнього вебсерверу WebServer, та клас для контролю доступу AccessControl.

* + - 1. **Клас Main**

Клас має два методи Init і Loop:

* Метод Init який призначений для ініціалізації всієї системи, а саме створення менеджеру WiFi який повинен підключитися до роутера або створити власну WiFi мережу для налаштування доступу до необхідної мережі, також проводиться ініціалізація класів ConfigStorage, AccessControl і WebServer, лістинг методу Init представлено на лістингу 3.1.

Лістинг 3.1 – Метод Main::Init

void Init() {  
 /\*\*  
 \* Init devices  
 \*/  
 Serial.begin(115200);  
  
 /\*\*  
 \* Init Wifi module  
 \*/  
 AsyncWiFiManager wifiManager(&server, &dns);  
 if (!wifiManager.autoConnect(String("AccessControlDevice-" + WiFi.macAddress()).c\_str())) {  
 Serial.println("Failed to connect, setup timeout");  
 delay(3000);  
 /\*\*  
 \* Try again to connect to wifi router   
 \*/  
 ESP.reset();  
 delay(5000);  
 }  
 Serial.println(String("Successfully connected, device IP address is: ") + WiFi.localIP().toString());  
 config.Init();  
  
 accessControl.Init();  
 webServer.Init();  
}

* Loop який виконує дії для циклу роботи програми, а саме запуск перевірки класом контролю доступу наявності нової картки на зчитувачі, код методу наведений в лістингу 3.2.

Лістинг 3.2 – Метод Main::Loop

void Loop() {  
 accessControl.NextCard();  
}

* + - 1. **Клас ConfigStorage**

Цей клас має три методи Init для ініціалізації, метод Load для завантаження конфігурації з внутрішньої пам’яті та метод Save для її збереження. Для завантаження конфігурації використовується LittleFs.open метод який зчитує конфіг, потім створюється динамічний json документ DynamicJsonDocument якому виділяється пам'ять в розмірі JSON\_OBJECT\_SIZE(3) + 360, після цього виконується парсинг файлу и запис параметрів конфігу в класу. Процес збереження схожий до завантаження, але перед збереженням виконується видалення файлу і потім знову створення цього фалу шляхом серіалізації даних в цей файл.

* + - 1. **Клас WebServer**

Клас має один метод який описує методи для комунікації з пристроєм і потім запуск серверу. Для захисту маніпулювання пристроєм сторонніми користувачами пристрій зберігає в конфігурації спеціальний токен який звіряється з заголовком HTTP запиту “TOKEN”, якщо вони співпадають виконання методу продовжується. Приклад проведення авторизації наведено в лістингу 3.3.

Лістинг 3.4 – Приклад проведення авторизації запиту

[this](AsyncWebServerRequest \*request) {  
 Serial.println("WebServer::on(/ping): handled");  
 if (!request->hasHeader("TOKEN") || request->getHeader("TOKEN")->value() != String(config->authHash))  
 {  
 Serial.println("WebServer::on(/ping): Access denied");  
 request->send(403, "application/json", "{\"status\": \"error\", \"message\":\"Access denied\"}");  
 return;  
 }

Веб сервер реалізує два POST методи:

“/ping” – цей метод нічого функціонально не робить, він тільки авторизує запит і повертає відповідь, цей метод необхідний серверу для перевірки правильності налаштованої конфігурації і перевірки статусу.

“/update\_config” – цей метод слугує для оновлення конфігурації. Метод приймає дані в форматі x-form-urlencoded і зчитує 3 параметра:

* mode – режим роботи пристрою
* server\_address – IP адреса головного серверу
* token – новий токен авторизації пристрою

Приклад зчитування одного з параметрів запиту наведено в лістингу 3.4.

Лістинг 3.4 – Приклад зчитування параметру запиту /update\_config та його збереження

if (request->hasParam("mode", true))  
{  
 String newParam = String(request->getParam("mode", true)->value());  
 Serial.println("WebServer::on(/update\_config): param lockState changed: " + String(config->lockState) + " -> " + newParam);  
 config->lockState = newParam == "LOCKED" ? 1 : newParam == "UNLOCKED" ? 2 : 3;  
 Serial.println("WebServer::on(/update\_config): param lockState new state: " + String(config->lockState));  
}

config->Save();

* + - 1. **Клас AccessControl**

Клас відповідає за перевірку каток, відсилання запитів до серверу, подачу сигналу на відкриття дверей і зміні світлодіодної індикації. Реалізація алгоритму який зображений на рис. 2.11 наведений в лістингу 3.5. Процес перевірки розділений на 3 методи, перший NextCard вирішує в якому режиму працює пристрій, якщо він працює в режиму GUARD то перевіряти наявність нових карток шляхом виклику методу ProcessCardIfPresent, якщо в режимі LOCKED нічого не робити і подавати сигнал для закриття дверей, цей процес виконує метод Block, і нарешті якщо UNLOCKED режим то постійно подавати сигнал на відкриття дверей,для цього викликається метод Unlock. Під час перевірки наявності картки спочатку перевіряється наявність картки на зчитувачі, потім при наявності картки, зчитується унікальний ідентифікатор і відсилається до серверу, після отримання результату перевіряється відповідь і далі відкриваються двері шляхом виклику методу Unlock, який також змінює індикацію світлодіода.

Лістинг 3.5 – основні методи класу AccessControl для перевірки доступу

void AccessControl::ProcessCardIfPresent()  
{  
 if (mfrc522.PICC\_IsNewCardPresent() && mfrc522.PICC\_ReadCardSerial())  
 {  
 ChangeColor(true, true);  
 String uuid = "";  
 for (byte i = 0; i < mfrc522.uid.size; i++)  
 {  
 uuid.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : ""));  
 uuid.concat(String(mfrc522.uid.uidByte[i], HEX));  
 }  
 mfrc522.PICC\_HaltA();  
 Serial.println("AccessControl::NextCard: End reading:uid -> " + uuid);  
  
 bool accessGranted = IsCardHaveAccess(uuid);  
  
 if (accessGranted)  
 Unlock();  
 else  
 ChangeColor(true, false);  
  
 Serial.println("AccessControl::NextCard: Access " + accessGranted ? "granted" : "denied");  
  
 delay(2000);  
 Lock();  
 }  
}  
  
bool AccessControl::IsCardHaveAccess(String uuid)

Продовження лістингу 3.5.

{  
  
 HTTPClient http;  
 http.begin("http://" + String(config->controlServerAddress) + check\_uuid\_request);  
 http.addHeader("Content-Type", "application/json");  
 int statusCode = http.POST("{\"uuid\": \"" + uuid + "\"}");  
 String response = http.getString();  
 Serial.println("AccessControl::NextCard::response: " + response);  
 DynamicJsonDocument doc(1024);  
 return statusCode > 0 &&  
 String(deserializeJson(doc, response).c\_str()) == String("Ok") &&  
 String(doc["status"].as<char \*>()) == String("success");  
}  
  
void AccessControl::NextCard()  
{  
 if (config->lockState == 3)  
 {  
 ProcessCardIfPresent();  
 Lock();  
 }  
 else if (config->lockState == 1)  
 Block();  
 else  
 Unlock();  
 delay(500);  
}

* + 1. **Реалізація фізичного пристрою контролю доступу**

Пристрій був зібраний згідно схеми принципової ка була спроектована в 2му розділі. В середині пристрою знаходяться 5 модулів, контролер Nodemcu v3 Lolin з вистроєним WIFI модулем, та інші модулі: Mosfet module IRF520; RFID-RC522; SMD-5050 RGB; MT3608. На рис. 3.1 зображено пристрій в закритому виді та на рисунку 3.2 пристрій зображений в відкритому вигляді.

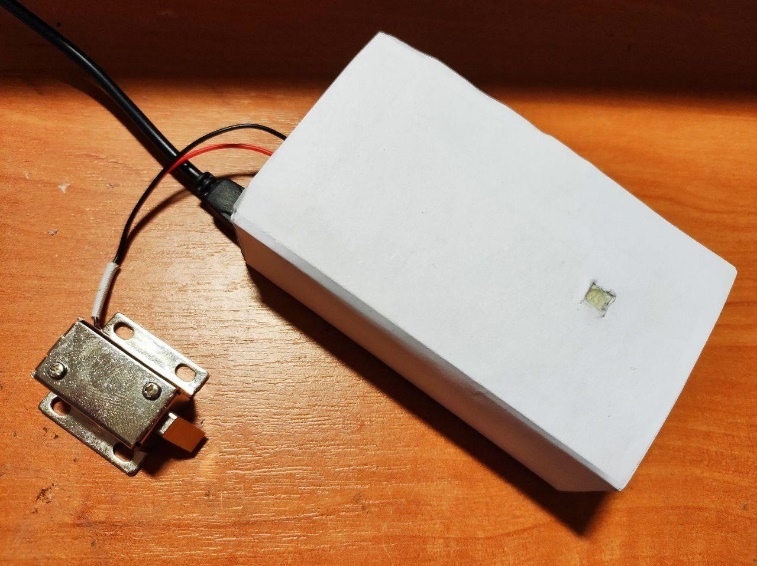


Рисунок 3.1 – Зображення пристрою контролю доступу в закритому вигляді

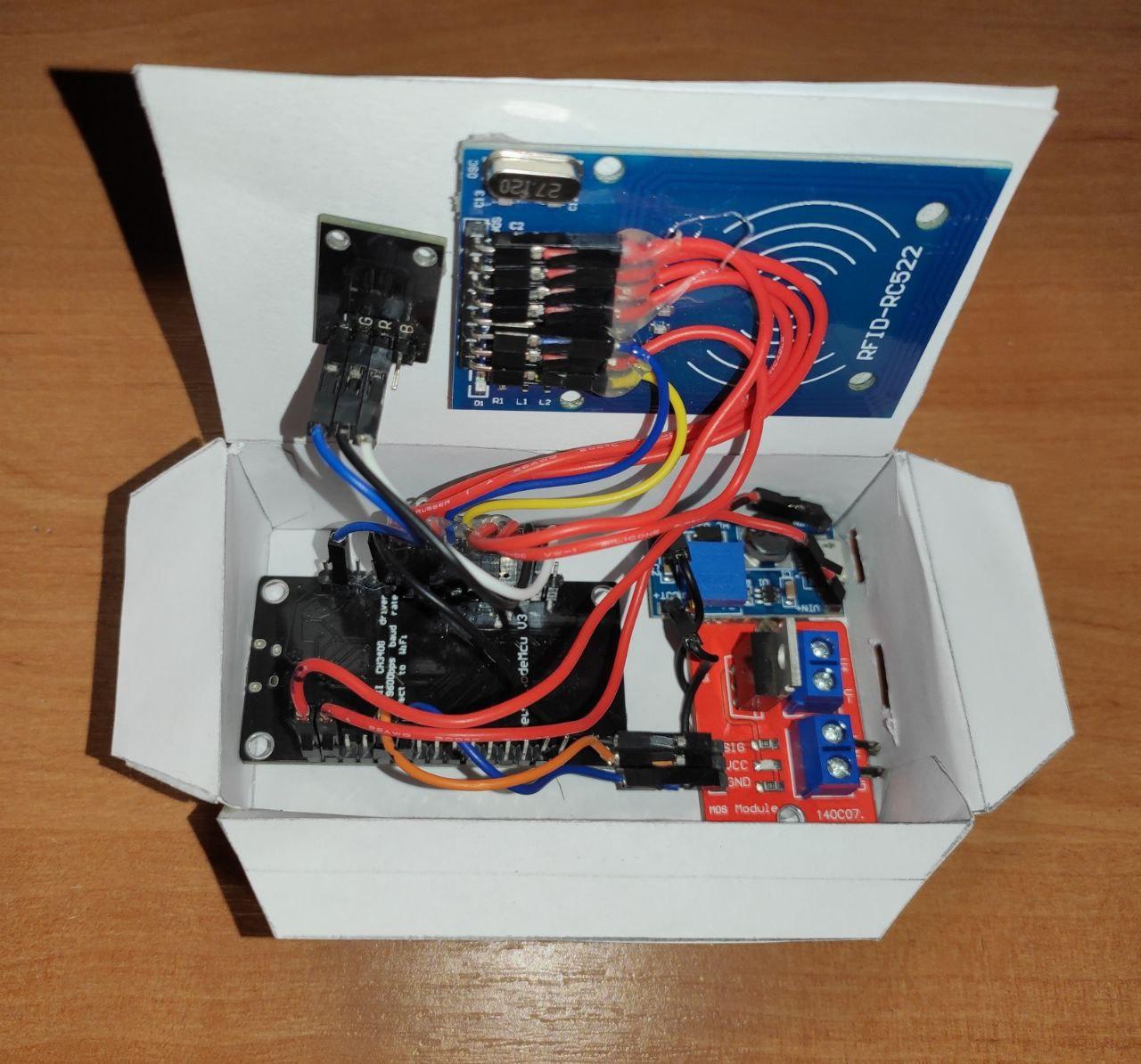


Рисунок 3.2 –Зображення пристрою контролю доступу в відкритому вигляді

* + 1. **Перевірка реалізованого пристрою контролю доступу**

Пристрій контролю доступу має 3 режими роботи, кожен режим роботи супроводжується горінням світлодіоду. В режимі GUARD світлодіод зовсім не світиться, див рис. 3.1, в режимі LOCKED світлодіод світиться червоним кольором див рис. 3.3, в режимі UNLOCKED світлодіод світиться зеленим див рисунок 3.4. Також світлодіод змінює свій колів протягом роботи в режимі GUARD, а саме при обробці даний, зчитування відправка, горить жовтий світлодіод рис 3.5, показуючи що пристрій обробляє дані, потім якщо доступ надано світлодіод світиться зеленим кольором, якщо не надано то червоним.

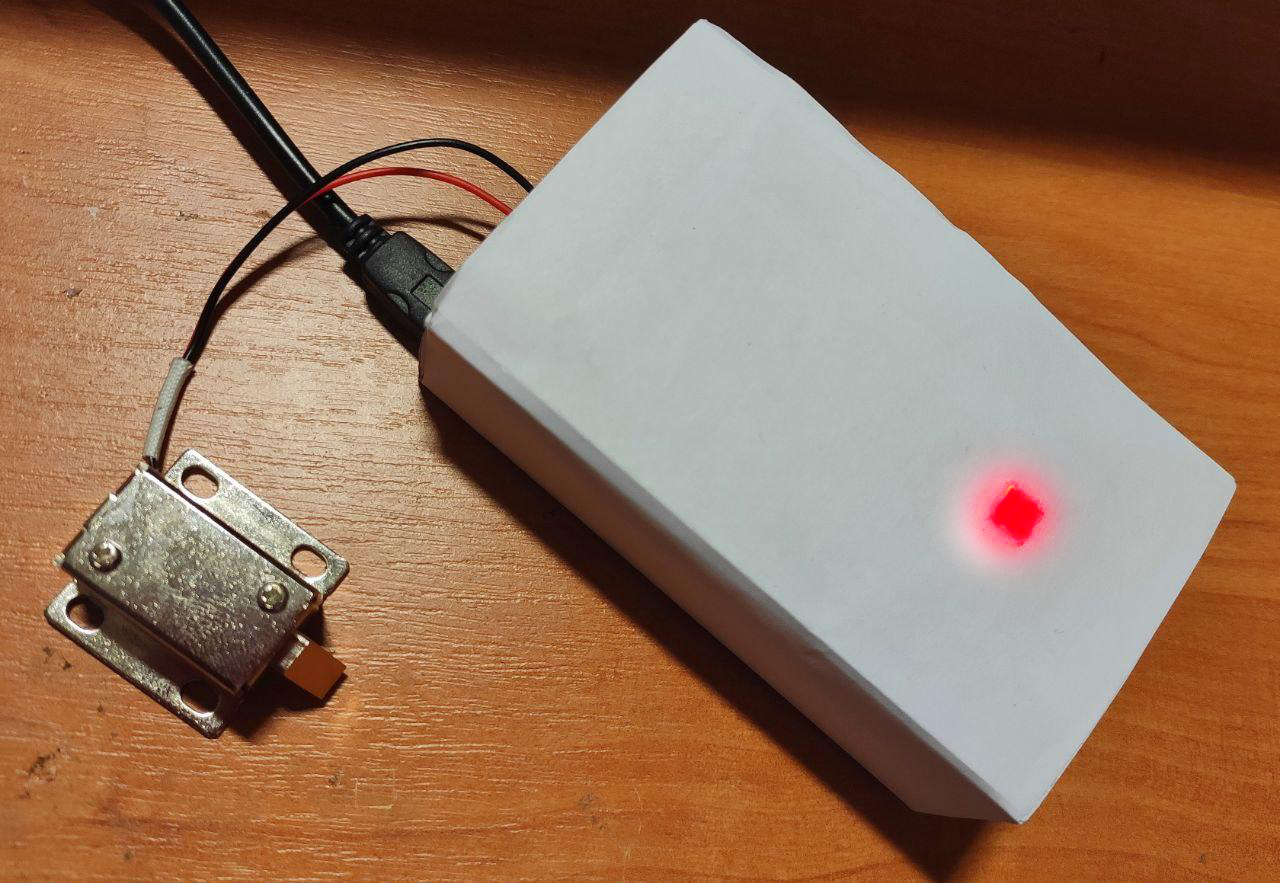


Рисунок 3.3 – Зображення пристрою контролю доступу в режимі LOCKED

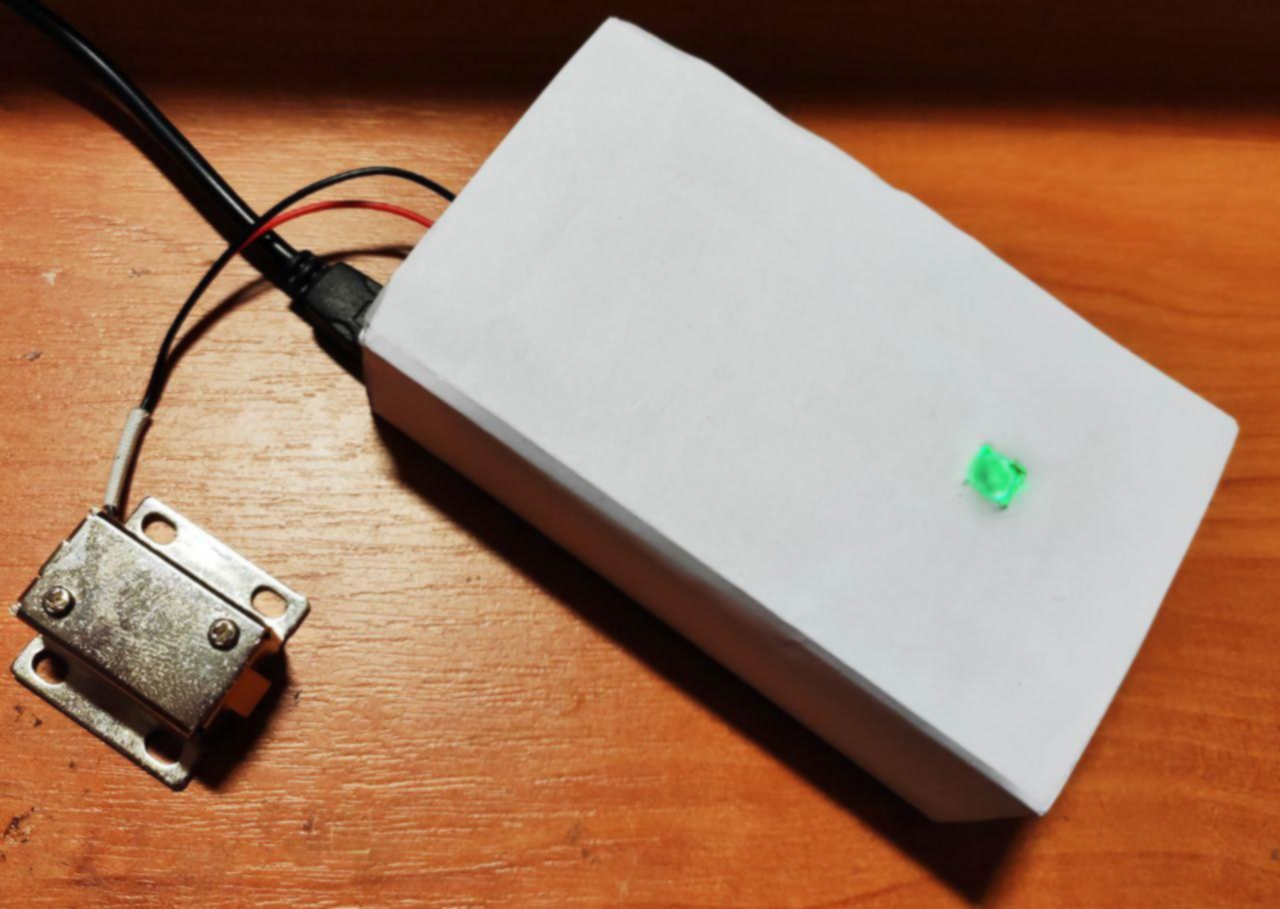


Рисунок 3.4 – Зображення пристрою контролю доступу в режимі UNLOCKED



Рисунок 3.5 – Зображення пристрою контролю доступу в режимі GUARD в момент зчитування даних

* 1. **Реалізація серверної підсистеми**

Реалізація серверної частини складається з двох частин, перша – це створення бази даних і друга – це створення самого серверного додатку.

* + 1. **Реалізація бази даних системи контролю доступу**

База даним комп’ютерної системи контролю доступу з використанням ідентифікаційних карт включає в себе 7 таблиць.

Нижче описані ці таблиці.

Таблиця “Users” відповідає за збереження даних про адмінів системи, в даних час система не має можливості створювати нових адмінів, тож в цій таблиці зберігається тільки один запис, логін і пароль зберігаються в зашифрованому виді і їх значення в новоствореній системі – це логін – admin, та пароль admin, представлений в лістингу 3.6.

Лістинг 3.6 – Дапм для таблиці Users

create table users

(

i\_user int auto\_increment

primary key,

name varchar(255) not null,

login varchar(255) not null,

password varchar(255) not null

);

Таблиця “Sessions“ призначена для зберігання активних сесій які на даний час є в системі, наведено в лістингу 3.7

Лістинг 3.7 – Дапм для таблиці Sessoins

create table sessions

(

i\_session int auto\_increment

primary key,

session varchar(255) not null,

created\_at datetime null,

i\_user int not null,

constraint sessions\_ibfk\_1

foreign key (i\_user) references users (i\_user)

on update cascade

);

create index i\_user

on sessions (i\_user);

Таблиця “Roles” призначена для зберігання інформації про ролі, дамп наведений в лістингу 3.8.

Лістинг 3.8 – Дапм для таблиці Roles

create table roles

(

i\_role int auto\_increment

primary key,

name varchar(255) not null,

allowed\_all tinyint(1) default 0 not null

);

Таблиця “Devices” призначена для зберігання інформації про пристрої контролю доступу, дамп наведений в лістингу 3.9

Лістинг 3.9 – Дапм для таблиці Devices

create table devices

(

i\_device int auto\_increment

primary key,

name varchar(255) not null,

description varchar(255) null,

ip varchar(255) null,

mode enum('LOCKED', 'UNLOCKED', 'GUARD') default 'GUARD' not null,

token varchar(255) null,

constraint ip

unique (ip)

);

Таблиця “Role\_device\_permissions” призначена для зберігання інформації про доступні пристрої для певної ролі, ця таблиця створює зв'язок Many-to-many між таблицями Roles та Devices, дамп цієї таблиці наведено в лістингу 3.10.

Лістинг 3.10 – Дапм для таблиці Role\_device\_permissions

create table role\_device\_permissions

(

i\_role\_device int auto\_increment

primary key,

i\_device int null,

i\_role int null,

constraint role\_device\_permissions\_i\_role\_i\_device\_unique

unique (i\_device, i\_role),

constraint role\_device\_permissions\_ibfk\_3

foreign key (i\_device) references devices (i\_device)

on update cascade on delete cascade,

constraint role\_device\_permissions\_ibfk\_4

foreign key (i\_role) references roles (i\_role)

on update cascade on delete cascade

);

create index i\_role

on role\_device\_permissions (i\_role);

Таблиця Cards зберігає інформацію про карти доступу, дапм наведено в лістингу 3.11.

Лістинг 3.11 – Дапм для таблиці Cards

create table cards

(

i\_card int auto\_increment

primary key,

uuid varchar(255) null,

name varchar(255) null,

i\_role int null,

constraint cards\_ibfk\_1

foreign key (i\_role) references roles (i\_role)

on update cascade on delete set null

);

create index i\_role on cards (i\_role);

Таблиця “Logs” зберігає інформацію про всі події авторизації карток доступу, дамп наведено в лістингу 3.12.

Лістинг 3.12 – Дапм для таблиці Logs

create table logs

(

i\_log int auto\_increment

primary key,

i\_device int null,

device\_name varchar(255) null,

i\_role int null,

role\_name varchar(255) null,

i\_card int null,

card\_name varchar(255) null,

time datetime null,

access tinyint(1) null,

device\_ip varchar(255) null,

uuid varchar(255) null,

error varchar(255) null,

constraint logs\_ibfk\_4

foreign key (i\_device) references devices (i\_device)

on update cascade on delete set null,

constraint logs\_ibfk\_5

foreign key (i\_role) references roles (i\_role)

on update cascade on delete set null,

constraint logs\_ibfk\_6

foreign key (i\_card) references cards (i\_card)

on update cascade on delete set null

);

create index i\_card

on logs (i\_card);

create index i\_device

on logs (i\_device);

create index i\_role

on logs (i\_role);

* + 1. **Реалізація глобальних інтерфейсів та опис глобального класу констант**

За глобальні інтерфейси відповідає файл types.ts в ньому реалізовані наступні інтерфейси ApplyConfigOptions, LogActionOptions, DeviceConfig, ServerInstanceConfig. Константи в проекті знаходяться в файлі constants.ts в якому, реалызовано декілька енамів і один клас з константами

* + 1. **Реалізація API контролерів**

Контролери слугують для створення набору кінцевих точок API які виконують певну бізнес логіку. Клас контролерів використовує декоратор @Controller який специфікує деякий шар роутера, який специфікує деякий набір методів, наприклад є контролер AccessControlController, він специфікується роутом “/access\_control”. Тобто, щоб виконати API метод “/check\_card” потрібно додати до нього в головний роутер контролера “/access\_control/check\_card”.

* + - 1. **Контролер AccessControlController**

Цей контролер відповідає за реалізацію методів для перевікрки карток доступу. Цей контролер реалізує тільки один метод check\_card, який призначений для перевірки доступу карти до деякого пристрою. Код класу наведений в лістингу 3.13.

Лістинг 3.13 – Код класу

@Controller({ route: '/access\_control' })  
export class AccessControlController {  
 @Inject(AccessLoggerService) private accessLogSrv!: AccessLoggerService;  
  
 @POST({ url: '/check\_card', options: { schema: AccessControlSchema.*checkCard* } })  
 async checkCard(request: FastifyRequest): Promise<jsend.JSendObject> {  
 const { ip, body } = request;  
 const { uuid } = body;  
  
 const devices: Device[] = await Device.*findAll*({ where: { ip } });  
 const device: Device = devices[0];  
 const cards: Card[] = await Card.*scope*('extended\_access\_map').*findAll*({ where: { uuid } });  
 const card: Card = cards[0];  
 let error: string = null;  
  
 if (devices.length !== 1) {  
 error = 'Device IP is not recognized';  
 } else if (cards.length !== 1) {  
 error = 'UUID is not registered';  
 } else if (!card.i\_role) {  
 error = 'Role for card not specified';  
 } else if (card.role.allowed\_devices.every(({ i\_device }) => i\_device !== device.i\_device)) {  
 error = 'Device not in scope of allowed for your role';  
 }  
 this.accessLogSrv.logAction({ card, device, ip, error, uuid }).then();  
 if (error) {  
 ***console***.log(`Access denied, device\_ip:${ip}, card:${uuid} reason: ` + error);

Продовження лістингу 3.13.

throw ***Error***('Access denied, reason: ' + error);  
 }  
 ***console***.log(`Access granted, device\_ip:${ip}, card:${uuid}`);  
  
 return jsend.success(null);  
 }  
}

* + - 1. **Контролер CardController**

Цей контролер призначений для отримання інформації про картки доступу, таблиця з API методами зображена в таб. 3.1.

Таблиця 3.1 – Таблиця API POST методів для контролера CardController

|  |  |
| --- | --- |
| Метод | Опис |
| /add\_card | Цей метод відповідає за додавання нових карток доступу до системи |
| /get\_card\_info | Метод повертає дані про конкретній карті доступу |
| /get\_card\_list | Метод повертає інформацію про всі картки доступу |
| /update\_card | Метод відповідає за оновлення інформації про карту доступу |
| /delete\_card | Метод відповідає за видалення інформації про конкретну карту |

* + - 1. **Контролер DeviceController**

Цей контролер призначений для отримання інформації про пристроїв контролю доступу, таблиця з API методами зображена в таб. 3.2.

Таблиця 3.2 – Таблиця API POST методів для контролера DeviceController

|  |  |
| --- | --- |
| Метод | Опис |
| /add\_device | Цей метод відповідає за додавання нових пристроїв контролю доступу |
| /get\_ device \_info | Метод повертає дані про конкретній пристрої контролю доступу |
| /get\_ device \_list | Метод повертає інформацію про всі пристрої контролю доступу |
| /update\_ device | Метод відповідає за оновлення інформації про пристрій контролю доступу |
| /delete\_ device | Метод відповідає за видалення інформації про конкретний пристрій контролю доступу |

* + - 1. **Контролер LogController**

Цей контролер відповідає за процес отамання всіх логів системи. Код класу контролера наведено в лістингу 3.14.

Лістинг 3.14 – Код контролера LogController

@Controller({ route: '/logs' })  
export class LogController {  
 @POST({  
 url: '/get\_system\_logs',  
 options: { schema: LogSchema.*getSystemLogsSchema* }  
 })  
 async getLogs(): Promise<jsend.JSendObject> {  
 return jsend.success({ logs: await Log.*findAll*() });  
 }  
}

* + - 1. **Контролер RoleController**

Цей контролер призначений для отримання інформації про пристроїв контролю доступу, таблиця з API методами зображена в таб. 3.3.

Таблиця 3.3 – Таблиця API POST методів для контролера RoleController

|  |  |
| --- | --- |
| Метод | Опис |
| /add\_role | Цей метод відповідає за додавання нових ролей доступу |
| /get\_ role \_info | Метод повертає данні про конкретну роль |
| /get\_ role \_list | Метод повертає інформацію про всі ролі |
| /update\_ role | Метод відповідає за оновлення інформації про роль |
| /delete\_ role | Метод відповідає за видалення інформації про роль |

* + - 1. **Контролер SessionController**

Контролер SessionController призначений для керування сесіями системи, які використовуються для обмеження доступу стороннім особам на графічному інтерфейсі, таблиця з API методами зображена в таб. 3.4.

Таблиця 3.4 – Таблиця API POST методів для контролера SessionController

|  |  |
| --- | --- |
| Метод | Опис |
| /ping | Цей метод відповідає за пінг системи, цей метод використовується для перевірки доступу до системи або для перевірки сесії |
| /login | Метод для авторизації адмінів системи |

Продовження таблиці 3.14

|  |  |
| --- | --- |
| /logout | Метод для завершення конкретної сесії |
| /change\_password | Метод відповідає за оновлення паролю адміністратора |
| /get\_my\_data | Метод відповідає отримання даних про адміністратора системи |

* + 1. **Реалізація сервісів**
       1. **Сервіс AccessLoggerService**

Сервіс відповідає за збирання даних та записування їх в базу даних, в лістингу 3.15 наведений код цього класу.

Лістинг 3.15 – Код сервісу AccessLoggerService

@Service()  
export class AccessLoggerService {  
 async logAction(options: LogActionOptions): Promise<void> {  
 const log: Log = new Log({  
 device\_ip: options.device?.ip || options.ip,  
 device\_name: options.device?.name || 'N/D',  
 i\_device: options.device?.i\_device,  
 i\_role: options?.card?.i\_role,  
 role\_name: options?.card?.role?.name || 'N/D',  
 access: !options.error,  
 time: new ***Date***(),  
 error: options.error,  
 i\_card: options?.card?.i\_card,  
 card\_name: options?.card?.name || 'N/D',  
 uuid: options?.card?.uuid || options.uuid || 'N/D'  
 });  
 log.save();  
 }  
}

* + - 1. **Сервіс DeviceControlService**

Цей сервіс відповідає за виконання запитів до пристроїв контролю доступу він має метод pingDevice та applyConfig, Код одного з цих методів наведено в лістингу 3.16.

Лістинг 3.16 – Лістинг методу pingDevice класу DeviceControlService

async pingDevice(deviceIp: string): Promise<boolean> {  
 const client: AxiosInstance = ***axios***.create({  
 baseURL: 'http://' + deviceIp,  
 responseType: 'json',  
 timeout: Constants.*defaultTimeout*

Продовження лістингу 3.16.

});  
 try {  
 const pingResult: AxiosResponse = await client.post<{ status: boolean }>(DeviceApiEndpoints.*Ping*);  
 return pingResult?.data?.status === 'success';  
 } catch (e) {  
 return false;  
 }  
}

* + - 1. **Сервіс SessionService**

Цей сервіс відповідає за перевірку та за отримання сесії. Одним із головних методів є метод checkSID, який використовується в системі для перевірки авторизації користувача, тривалість життя сесії становить 2 тижні, після закінчення браузер має видалити сесію власноруч. В лістингу 3.17 наведений код сервісу SessionService.

Лістинг 3.17 – Код сервісу SessionService

@Service()  
export class SessionService {  
 static async *getCurrentSession*(request: FastifyRequest): Promise<Session> {  
 const cookie: string = request.headers['cookie'];  
 if (!cookie) {  
 return new ***Promise***(null);  
 }  
 const SID: string = parseCookie(cookie)['SID'];  
 if (!SID) {  
 return new ***Promise***(null);  
 }  
 return Session.*findOne*({ where: { session: SID } });  
 }  
 static async *checkSID*(request: FastifyRequest, reply: FastifyReply<ServerResponse>): Promise<Session> {  
 if (request.raw.url.match(/\/session\/(login|change\_password)/)) {  
 return;  
 }  
 const session: Session = await SessionService.*getCurrentSession*(request);  
 if (!session) {  
 reply.code(403).send(jsend.error('Access denied: Session is invalid or not present'));  
 return;  
 }  
 if (~~(***Date***.now() / 1000) - ~~(session.createdAt.valueOf() / 1000) > Constants.*timestampOneWeek*) {  
 reply.code(403).send(jsend.error('Access denied: Session is expired'));  
 return;  
 }  
 return session;  
 }  
}

* + 1. **Реалізація моделей для взаємодії з базою даних**

В системі існує 7 таблиць в базі даних, і цім таблицям відповідає 7 моделей. Кожна модель унаслідується від класу Model яркий предоставляяє фреймворк sequelize, в цих моделях повністю описані таблиці. Приклад одної із моделей, одна з таких моделей наведена наведено в лістингу 3.18.

Лістинг 3.18 – Код моделі Card

@DefaultScope(() => ({  
 attributes: ['i\_card', 'uuid', 'name', 'i\_role']  
}))  
@Scopes(() => ({  
 extended\_access\_map: { include: [Role.*scope*('with\_devices')] }  
}))  
@Table({  
 tableName: 'cards',  
 modelName: 'cards'  
})  
export class Card extends Model<Card> {  
 @Column({ primaryKey: true, autoIncrement: true }) i\_card: number;  
 @Column uuid: string;  
 @Column name: string;  
 @ForeignKey(() => Role) @Column({ onDelete: 'SET NULL' }) i\_role: number;  
 @BelongsTo(() => Role) role: Role;  
}

* 1. **Реалізація клієнтської підсистеми**

Розроблений клієнтський веб інтерфейс дозволяє адміністратору системи швидко та зручно маніпулювати системою. Інтерфейс дозволяє створювати всі сутності, дивитися статус кожного пристрою, дивитися логи системи. На головній сторінці можна відразу бачити статус всіх пристрої, ця сторінка зображена на рис. 3.6, також на цьому рисунку зображено 3 пристрої контролю доступу які зараз підключені до системі і знаходяться в активному стані. На цій сторінці також можна змінити режим роботи для кожного пристрою контролю доступу натиснувши на іконку в кінці, після цього відкриється діалог зміни режиму, якщо пристрій не активний режим змінювати заборонено, діалог зміни зображений на рис. 3.7.

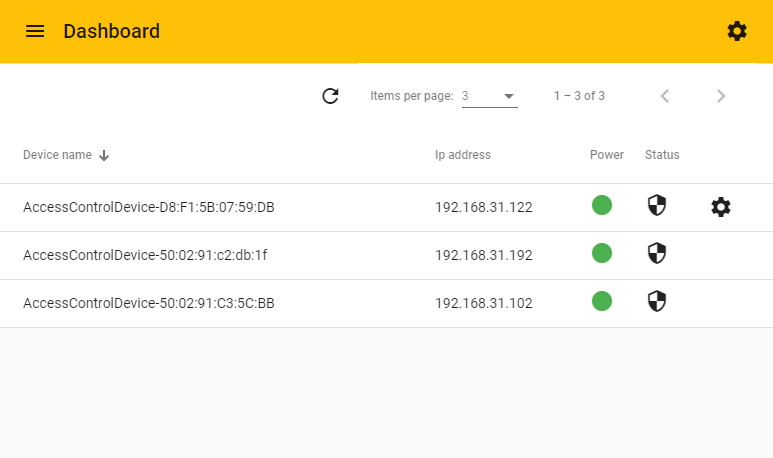


Рисунок 3.6 – Зображення головної сторінки веб інтерфейса

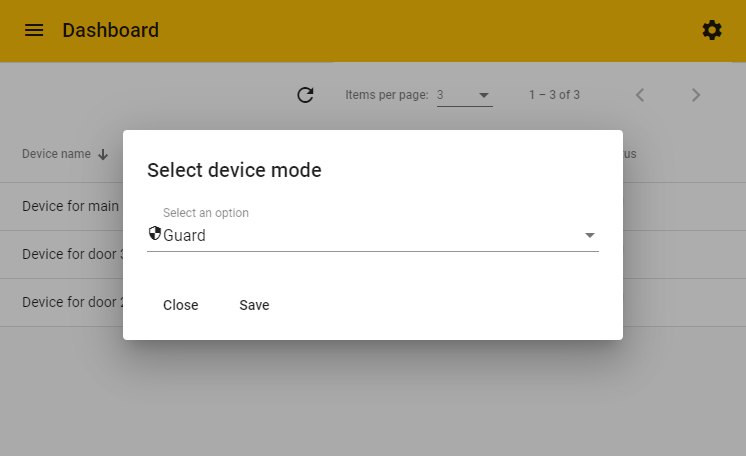


Рисунок 3.7 – Діалог зміни режиму роботи пристрою контролю доступу

На рис. 3.8 зображена сторінка з логами системи, на цьому рисунку можна побачити як система реєструє всі дії в системі, також можна побачити яким користувачам доступ надано, а яким ні, це можна побачити по забарвленню строки або по колонці Access, також в логах можна побачити причину блокування.

Причини блокування можуть бути наступними:

* Пристрій з таким IP адресом не зареєстрований в системі – “Device IP is not recognized”
* Картка не зареєстрована в системі – “UUID is not registered”
* Карта не має ролі – “Role for card not specified”
* Карта не має доступу до двері через те що роль не дає доступ до неї – “Device not in scope of allowed in assigned role”



Рисунок 3.8 – Зображення сторінки логу системи

На рис. 3.9 зображена сторінка з конфігурацією пристроїв контролю доступу. Також на цій сторінці можна додати новий запис, видалити чи відредагувати існуючий. Діалог додавання запису зображений на рис 3.10.

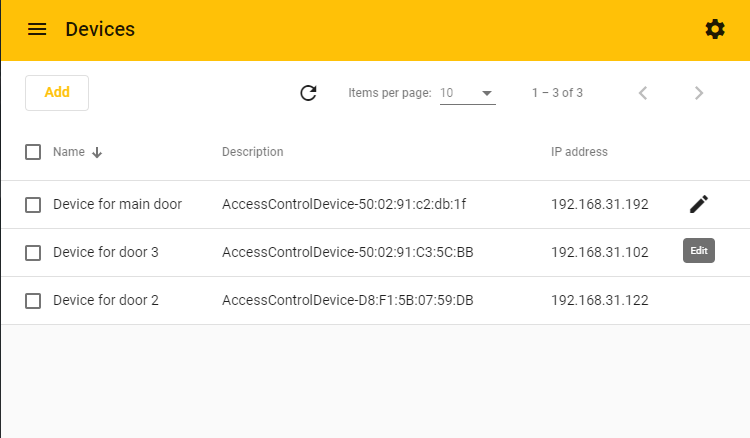


Рисунок 3.9 – Зображення сторінки конфігурації пристроїв контролю доступу

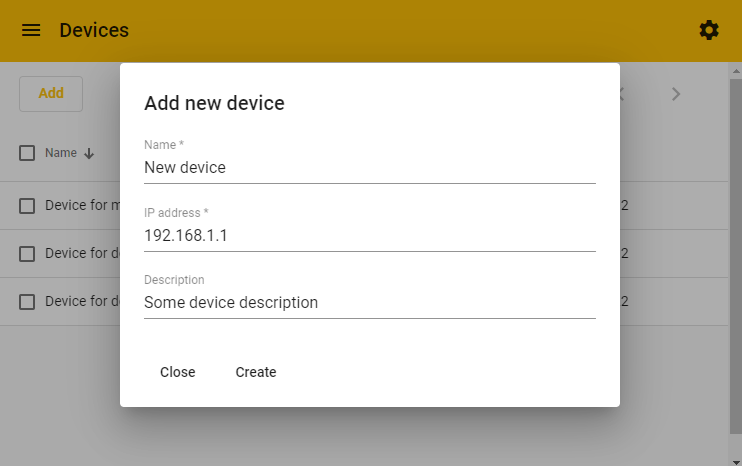


Рисунок 3.10 – Діалог додавання нового пристрою контролю доступу

На рис. 3.11 зображена сторінка конфігурації для ролі. На цій сторінці додатково можна побачити які пристрої доступні для цієї ролі, шляхом наведення мишки на іконку в стовпчику Allowed devices, а для того щоб побачити список користувачів що використовують цю роль треба навести мишку на іконку в колонці Cards with role. Для редагування запису можна натиснути на іконку ручки в кінці рядка і відкриється діалог редагування, на рис. 3.12 зображений цей діалог.

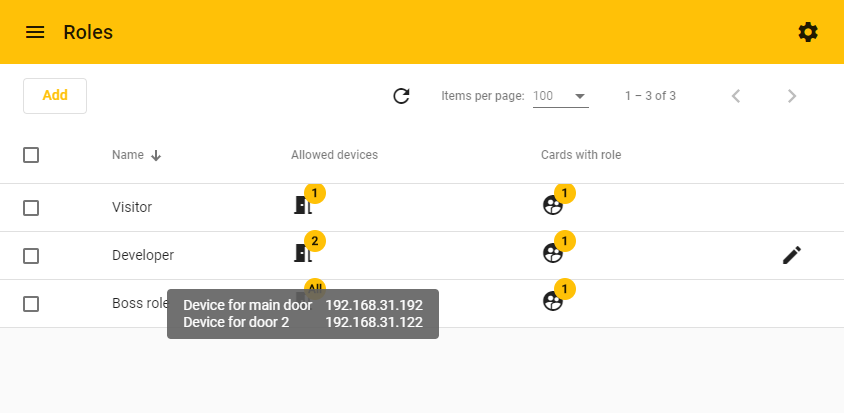


Рисунок 3.11 – Зображення сторінки ролей

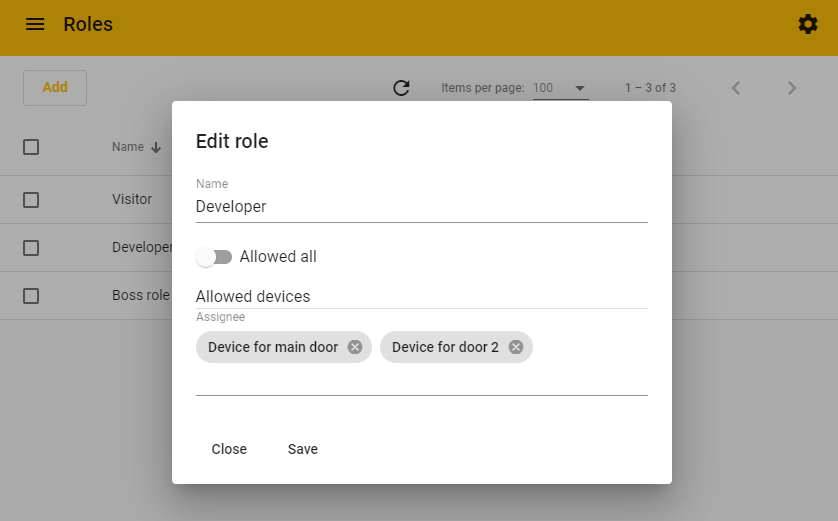


Рисунок 3.12 – Зображення діалогу редагування ролі

На рис. 3.13 можна побачити сторінку з ідентифікаційними картками доступу на якій можна створювати нові картки, видаляти та змінювати на рис 3.14 зображений діалог редагування однієї із карток.

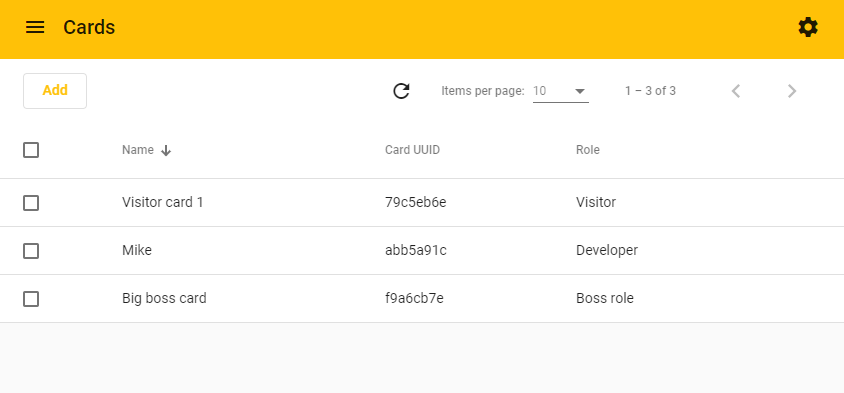


Рисунок 3.13 – Зображення сторінки конфігурації ідентифікаційних карток

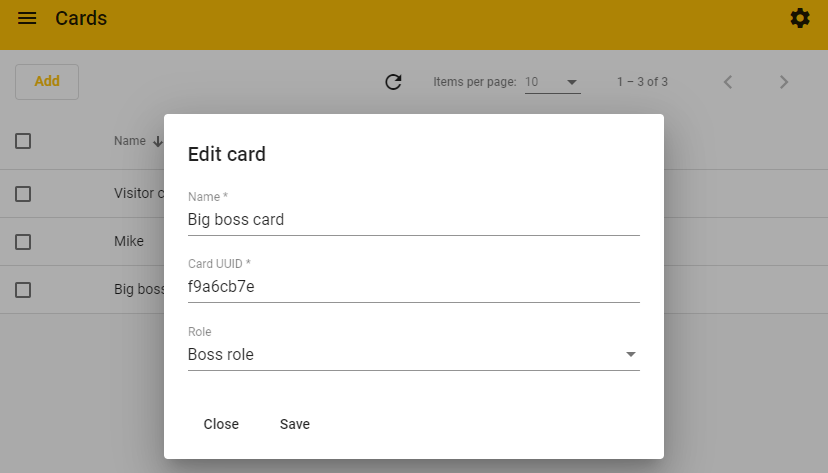


Рисунок 3.14 - Зображення діалогу редагування ідентифікаційної картки

* 1. **Взаємодія серверу та клієнта**

Взаємодія веб клієнта відбувається шляхом відправки запитів до серверу, приклад запитів до серверу можна побачити на рисунку 3.15, та приклад відповіді самого сервера на рис. 3.16.

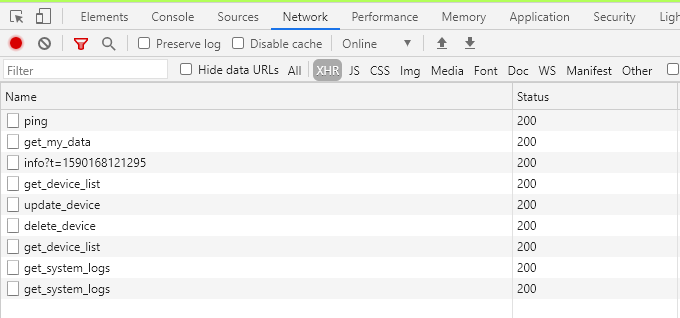


Рисунок 3.15 – Зображення відправки запитів від клієнта до сервера

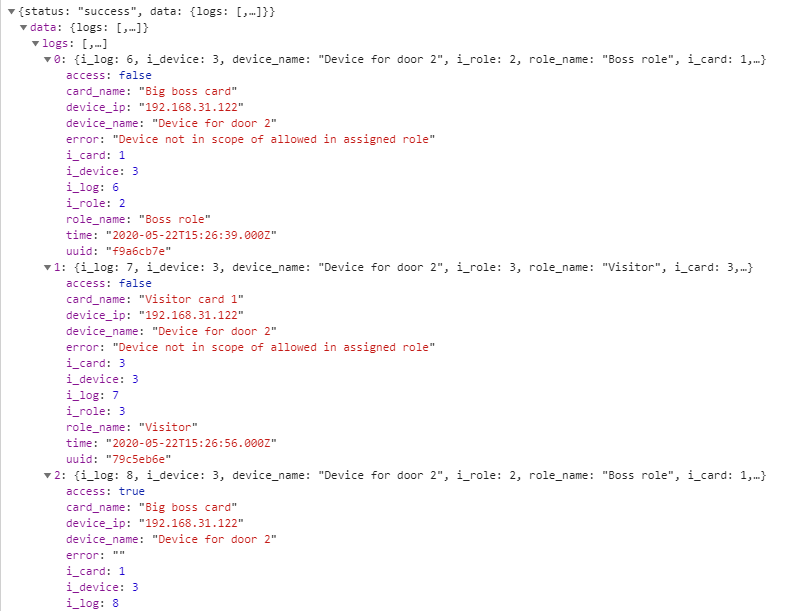


Рисунок 3.16 – Приклад відповіді від сервера, відповідь методу /logs/get\_system\_logs

* 1. **Реалізація автоматизованої збірки кожного модулю проекту та запуск головного серверу з графічним додатком**

Весь проект складається з 3х частин, кожній частині відповідає своя папка, для серверної частини це папка “server”, для графічної частини папка “ui” і для апаратної частини “nodemcu\_firmware”. Для спрощення збірки всіх частин та деплою серверу разом з графічною частиною було створено скрипт app-control.sh, який дозволяє встановити всі бібліотеки, зібрати серверну частини та графічну частини, створити всі таблиці в базі даних, скомпілювати прошивку для пристрою контролю доступу та завантажити її до підключеного пристрою. Для роботи скрипу необхідно щоб в системі був встановлений python3 та nodejs. Всі можливі операції можна дізнатися виконавши команду “./app-control.sh -h”, результат виконання команд зображений на рисунку 3.17

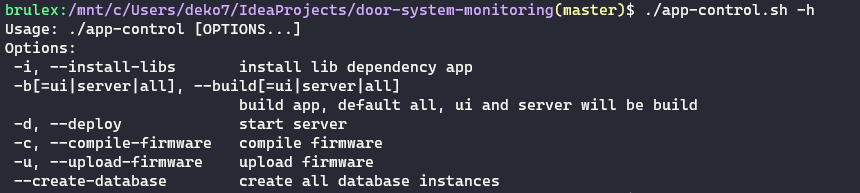


Рисунок 3.17 – Зображення виклику команди help в скрипті app-control

Перед початком роботи з збіркою будь якої частини треба спочатку запустити скачування та встановлення всіх необхідних бібліотек. Для цього треба виконати команду “app-control.sh --install-libs”, після чого почнеться процес закачування всіх необхідних бібліотека.

Для того щоб зібрати та запустити сервер з графічною частиною треба виконати після встановлення бібліотек команду “./app-control.sh --build --deploy”, перший ключ дає знати скрипту що необхідно спочатку скомпілювати серверну та графічні частини. Після збірки обох частин відбувається копіювання скомпільованого графічного інтерфейсу до скомпільованого серверу в папку “ui\_dist”. Другий ключ “--deploy" відповідає за запуск серверу, сервер в скомпільованому режиму буде також давати користувачам користуватися графічною частиною за шляхом “/ui”.

Для компіляції прошивки пристрою контролю доступу та її завантаження на нього необхідно виконати команду “./app-control.sh --compile-firmware --upload-firmware”, перший ключ “--compile-firmware” відповідає за компілювання коду в бінарний, другий ключ відповідає за завантаження прошивки на підключений пристрій. Компілювання та закачуванням відбувається за допомогою програми platformio, яка надає можливість компілювати та завантажувати прошивки до різних пристроїв, вона була згадана в розділі 2.3.1. Якщо ніякого пристрою не знайдено чи пристроїв декілька з’явиться повідомлення для вказування COM порту до якого підключений пристрій. Після завантаження нової прошивки пристрій необхідно перезавантажити, після цього він створить нову WIFI точку, до якої треба підключитися, після підключення треба перейти до браузеру, він повинен відкрити сторінку налаштування локальної мережі пристрою, якщо цього не сталося треба перейти по адресі 192.168.4.1, ця сторінка зображена на рис. 3.18 підключити до локальної мережі. Після відкриття сторінки треба натиснути на кнопку Configure WiFi де треба ввести необхідну локальну мережу та пароль до неї. Після цього пристрій повинен підключитися до цієї мережу, для того щоб підключити пристрій до системи необхідно додати IP адресу пристрою до системи на сторінці Devices.

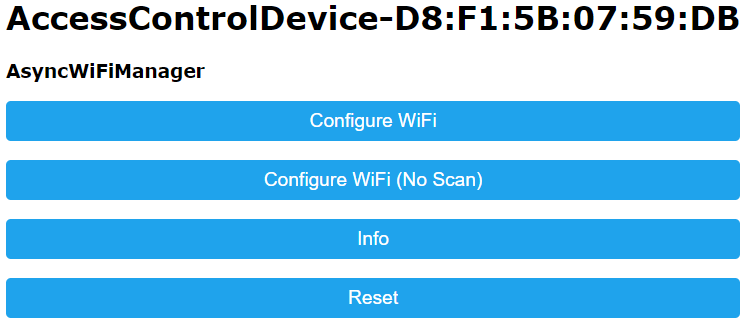


Рисунок 3.18 – Сторінка конфігурації локальної мережі для пристрою контролю доступу

**ВИСНОВКИ**

Під час виконання роботи були вивченні основні можливості існуючих систем контролю доступу, під час аналізу стало зрозуміло що існуючи системи не є досить зручними в користування, одні потребують покупки цілого нового комплексу, інші підтримують тільки ручне занесення карт, треті зовсім мають тільки обмежену кількість користувачів в системі.

Згідно висунутих вимог до системи контролю доступу з використання ідентифікаційних карт було створено архітектуру системи, були вибрані основні засоби для реалізації програмної частини всієї системи. Для апаратної частини було вибрано мову програмування C++, як середовища розробки було вибрано Visual studio code з плагіном platformio який дозволяє керувати різноманітними мікроконтролерами. Для програмної частини серверної та графічної частин було вибрано мову програмування typescript яка потом компілюється в мову програмування javascript, як основа для серверної частини було використано фреймворк fastify, для взаємодії з базою даних фреймворк sequalize. Основою для графічної частини став фреймворк Angular з базою графічних компонентів angular-material.

Було спроектований сервер який представляє API для пивного контролю системою, який можна в майбутньому інтегрувати з іншою системою. Розроблений сервер був спроектований так що масштабування буде безболісним для вже готової системи.

Був спроектований та реалізований графічний інтерфейс який використовує всі можливості API серверної частини і надає зручне налаштування системи через веб інтерфейс. Графічна система крім налаштування дає можливість моніторингу всіх пристроїв контролю доступу на головній сторінці інтерфейсу, а також на графічному інтерфейсі можна подивитися логи, події про всі події авторизації в системі.

Було спроектовано та створено пристрій контролю доступу який виконує роль контролера і зчитувача даних. Цей пристрій призначений для зчитування ідентифікаційних карток, після зчитування система відправляє данні на перевірку до головного серверу, який перевіряє і записую в базу даних подію про авторизації. Пристрій контролю доступу був спроектований таким чином що зчитувач можна замінити на будь який інший, наприклад на зчитувач відбитків пальців, треба лише додати функціонал для взаємодії з зчитувачем до модулю контролю доступу і налаштувати сервер для збереження такого роду ідентифікації.

**ПЕРЕЛІК ПОСИЛАНЬ**

1. Комплексный подход при идентификации личности / [Електронний ресурс]. – Режим доступу: http://lib.secuteck.ru/articles2/sys\_ogr\_dost/kompleksnyy-podhod-pri-identifikatsii-lichnosti
2. Сторінка з описанням пристрою Hikvision DS-K1T605E / [Електронний ресурс]. – Режим доступу: <https://hikvision.org.ua/uk/hikvision-ds-k1t605e>
3. Сторінка з описанням пристрою Hikvision DS-K1201EF / [Електронний ресурс]. – Режим доступу: <https://hikvision.org.ua/uk/hikvision-ds-k1201ef>
4. Toni Sellarès, "The Model View Controller: a Composed Pattern." / Toni Sellarès, Іспанія, Університет Жирони
5. Шарфельд Т. Системы RFID низкой стоимости с приложениями И. Девиля, Ж. Дамура, Н. Чаркани, С. Корнеева и А. Гуларии. Перевод с английского и научная редакция С. Корнеева. / Шарфельд Т. – Москва 2006 – с.6 – 18.
6. RFID / [Електронний ресурс]. – Режим доступу: <https://ru.wikipedia.org/wiki/RFID>
7. RFID тэги, метки, транспондеры / [Електронний ресурс]. – Режим доступу: <https://scancode.net.ua/p4317059-rfid-tegi-metki.html>
8. ISO/IEC 14443 / [Електронний ресурс]. – Режим доступу: <https://ru.wikipedia.org/wiki/ISO/IEC_14443>
9. Роль пластиковой карты («смарт-карты», проксимити-карты) в системах безопасности / [Електронний ресурс]. – Режим доступу: <http://gmpspb.ru/posts/post/plastic_card>
10. Near Field Communication / [Електронний ресурс]. – Режим доступу: <https://ru.wikipedia.org/wiki/Near_Field_Communication>
11. Офіційна документація Nodemcu / [Електронний ресурс]. – Режим доступу: <https://nodemcu.readthedocs.io>
12. Описання бібліотеки MFRC522 / [Електронний ресурс]. – Режим доступу: <https://github.com/miguelbalboa/rfid>
13. Посилання на документацію nodejs / [Електронний ресурс]. – Режим доступу:
14. Fastify official web site / [Електронний ресурс]. – Режим доступу: https://www.fastify.io
15. TypeScript Decorators overview / [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/docs/handbook/decorators.html
16. Fastify decorators readme / [Електронний ресурс]. – Режим доступу: https://github.com/L2jLiga/fastify-decorators/blob/next/lib/README.md
17. Sequelize ORM / [Електронний ресурс]. – Режим доступу: https://sequelize.org
18. Sequelize typescript readme / [Електронний ресурс]. – Режим доступу: https://github.com/RobinBuschmann/sequelize-typescript/blob/master/README.md
19. Introduction to the Angular Docs / [Електронний ресурс]. – Режим доступу: https://angular.io/docs
20. Angular material components / [Електронний ресурс]. – Режим доступу: <https://material.angular.io/components/categories>
21. Material design introduction / [Електронний ресурс]. – Режим доступу: <https://material.io/design/introduction>