НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Комп'ютерних наук

_____ Голуб Б.Л.

"___" _____ 20 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

Інформаційна система обліку контроля доступу на приватну територію

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми

Д.е.н., професор _____ Руденський Р.А.

Керівник бакалаврської кваліфікаційної роботи

Кандидат педагогічних наук, доцент _____ Волошина Тетяна Володимирівна

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Асистент кафедри інформаційних систем та технологій _____ Понзель Ярослав Юрійович

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Пуха Максим Миколайович

(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

Комп'ютерних наук

_____ Голуб Б.Л.

"___" _____ 20 р.

.

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту

Пуха Максим Миколайович

(прізвище, ім'я, по батькові)

Спеціальність 122 – «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи Інформаційна система обліку контроля доступу на приватну територію

затверджена наказом ректора НУБіП України №2246с від 16.12.2024р.

№_

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи: опис програмного забезпечення

Перелік питань, які потрібно розробити:

1. Аналіз проблемної області.
2. Вибір та обґрунтування засобів.
3. Проектування системи.
4. Висновки

Перелік графічних документів (за потреби)

_____

Дата видачі завдання "_____"_____20___ р.

Керівник бакалаврської кваліфікаційної роботи

_____ _____ _Волошина Тетяна Володимирівна_

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ ____Пуха Максим Миколайович_____

(підпис) (ПІБ студента)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	23.03 - 01.04	Виконано
2	Проектування системи	01.04 - 25.04	Виконано
3	Реалізація системи	25.04 - 09.05	Виконано
4	Тестування системи	09.05 - 16.05	Виконано
5	Оформлення пояснювальної записки	16.05 - 18.05	Виконано
6	Оформлення графічного матеріалу	18.05 - 23.05	Виконано

Студент _____ _____
(підпис) (ініціали та прізвище)

Керівник роботи _____ _____
(підпис) (ініціали та прізвище)

РЕФЕРАТ

ІНФОРМАЦІЙНА СИСТЕМА, КОНТРОЛЬ ДОСТУПУ, ОБЛІК, БАЗА ДАНИХ, ПРОЕКТУВАННЯ, ESP32, PHP, MYSQL, WEB-ІНТЕРФЕЙС

Предметом дослідження є сучасні технології розробки автоматизованих інформаційних систем для контролю доступу на приватних об’єктах.

Об’єктом дослідження є процес організації автоматизованого обліку доступу осіб та транспортних засобів на приватну територію.

Мета дипломної роботи: проектування та створення інформаційної системи обліку контролю доступу на приватну територію, що включає в себе RFID-ідентифікацію та веб інтерфейса для адміністрування.

Завданням дипломної роботи є створення програмного та апаратного забезпечення для автоматизації процесу контролю, фіксації та зберігання інформації про події доступу, модернізації процесу адміністрування та підвищення рівня безпеки.

Предметом дослідження є сучасні технології розробки автоматизованих інформаційних систем для контролю доступу на приватних об’єктах.

Об’єктом дослідження є процес організації автоматизованого обліку доступу осіб та транспортних засобів на приватну територію.

Мета дипломної роботи: проектування та створення інформаційної системи обліку контролю доступу на приватну територію, що включає в себе RFID-ідентифікацію та web інтерфейса для адміністрування.

Завданням дипломної роботи є створення програмного та апаратного забезпечення для автоматизації процесу контролю, фіксації та зберігання інформації про події доступу, модернізації процесу адміністрування та підвищення рівня безпеки.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

xml	–	розширювана мова розмітки, Extensible Markup Language
PHP	–	скриптова мова програмування, Hypertext Preprocessor
СКБД	–	система керування базами даних
БД	–	база даних
URL	–	уніфікований локатор ресурсів, Uniform Resource Locator
IDE	–	інтегроване середовище розробки, Integrated Development Environment
RFID	–	Radio Frequency Identification
ANPR	–	Automatic Number Plate Recognition
HTTP	–	протокол передачі гіпер-текстових документів, Hyper Text Transfer Protocol
JSON	–	формат обміну даними JavaScriptObjectNotation
SQL	–	Structured Query Language — мова структурованих запитів

ВСТУП

Забезпечення безпеки на приватних територіях із обмеженим доступом – одна з актуальних задач, які стоять на шляху розвитку інформаційних технологій. Автоматизація контролю доступу на ці об’єкти дозволяє підвищити рівень захищеності, раціонально використовувати ресурси персоналу та усунути людський фактор в питаннях охорони.

Дана дипломна робота є розробкою інформаційної системи обліку доступу на приватну територію, що базується на програмних та апаратних компонентах для ефективного управління. Основною ціллю проекту є створення інтегрованої системи, що забезпечує автоматичний контроль доступу транспортних засобів та осіб за їх RFID-міток та ведення обліку всіх подій доступу у зручному електронному вигляді.

Основними завданнями, які вирішуються у процесі реалізації цього проєкту, є:

- проектування зручного веб-інтерфейсу для додавання, редагування, видалення та перегляду користувачів;
- реалізація автоматичного зчитування RFID-мітки контролером і передачі отриманих даних на сервер для перевірки;
- розробка програмного забезпечення для перевірки наявності мітки у базі даних та відправки відповіді на мікроконтролер;
- автоматичне відкриття шлагбаума у разі підтвердження доступу; реєстрація та зберігання інформації про всі факти доступу у базі даних з фіксацією дати та часу.

Впровадження подібної системи дозволяє організаціям автоматизувати контроль доступу на приватних територіях, підвищити рівень безпеки, забезпечити гнучке управління правами доступу та отримати повну інформацію про всі події, що відбуваються на об’єкті.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ НА ТЕРИТОРІЮ З ОБМЕЖЕНИМ ДОСТУПОМ

1.1. Система контролю і управління доступом

Система контролю та управління доступом (СКУД) — це комплекс технічних і програмних засобів, призначених для організації безпечного та контрольованого допуску осіб і транспортних засобів на територію з обмеженим доступом. Основна мета впровадження СКУД полягає в тому, щоб підвищити рівень безпеки приватних або промислових об’єктів, забезпечити автоматизацію обліку переміщень і скоротити ризик несанкціонованого проникнення.

Типова СКУД складається з таких основних елементів:

- ідентифікатори (наприклад, RFID-картки або брелоки) для персоналу та відвідувачів;
- зчитувачі, які розташовуються біля в’їзду або входу;
- контролери, що приймають інформацію від зчитувачів та приймають рішення про надання або відмову в доступі;
- пристрої блокування (електронні замки, шлагбауми, турнікети) для фізичного обмеження доступу;

центральний сервер або комп’ютер із базою даних, в якій зберігається інформація про користувачів та події доступу.

Принцип роботи сучасної СКУД базується на тому, що кожна особа, якій надано доступ, отримує унікальний електронний ідентифікатор. Під час наближення ідентифікатора до зчитувача відбувається передача даних на контролер системи, де ця інформація звіряється з базою даних. Якщо користувач має дозвіл, система

надсилає сигнал на виконавчий пристрій (наприклад, відкриває шлагбаум або двері), одночасно фіксуючи факт події у журналі подій.

Крім забезпечення фізичного контролю, СКУД дозволяє здійснювати облік часу перебування співробітників чи гостей на об’єкті, вести докладну статистику відвідувань та гнучко налаштовувати рівні доступу для різних категорій користувачів. У разі спроби несанкціонованого проходу система може активувати тривожну сигналізацію, повідомлення охороні або адміністратору.

СКУД знайшли широке застосування не лише на підприємствах і офісах, а й у житлових комплексах, паркінгах, гаражних кооперативах та приватних домоволодіннях. Завдяки впровадженню автоматизованих систем контролю доступу забезпечується не лише захист майна, а й створюються умови для більш ефективного управління ресурсами та персоналом [1].

1.2. Аналіз існуючих систем контролю доступу

Існує декілька основних типів систем, які забезпечують контроль доступу на приватних територіях та об’єктах з обмеженим доступом. Найбільш поширені серед них такі:

1) **Системи з використанням RFID-технологій**

RFID (радіочастотна ідентифікація) є однією з найпопулярніших технологій для організації автоматизованого контролю доступу. Користувачам чи транспортним засобам видаються RFID-мітки, які мають унікальний код. На контрольованих точках встановлюються RFID-зчитувачі, які ідентифікують мітку, після чого система приймає рішення про відкриття воріт або надання доступу. Основна перевага такого підходу — швидкість, безконтактність та можливість гнучко налаштовувати рівні доступу. []

Рисунок 1.1 – Приклад використання RFID-міток

2) Системи з автоматичним розпізнаванням номерних знаків (ANPR):

ANPR-системи (Automatic Number Plate Recognition) використовують камери відеоспостереження та спеціалізоване програмне забезпечення для зчитування та аналізу номерних знаків транспортних засобів при в’їзді та виїзді. Отримані дані порівнюються з базою дозволених номерів. Цей метод часто застосовується у великих парковках, бізнес-центрах та закритих житлових комплексах. [21]

Рисунок 1.2 – Приклад використання ANPR

3) Карткові системи доступу

Іншим поширеним методом є використання пластикових карт із магнітною смугою або чипом, які персонал чи мешканці підносять до зчитувача. Такий підхід

дозволяє не лише ідентифікувати користувача, а й вести облік часу перебування та відвідувань. [22]

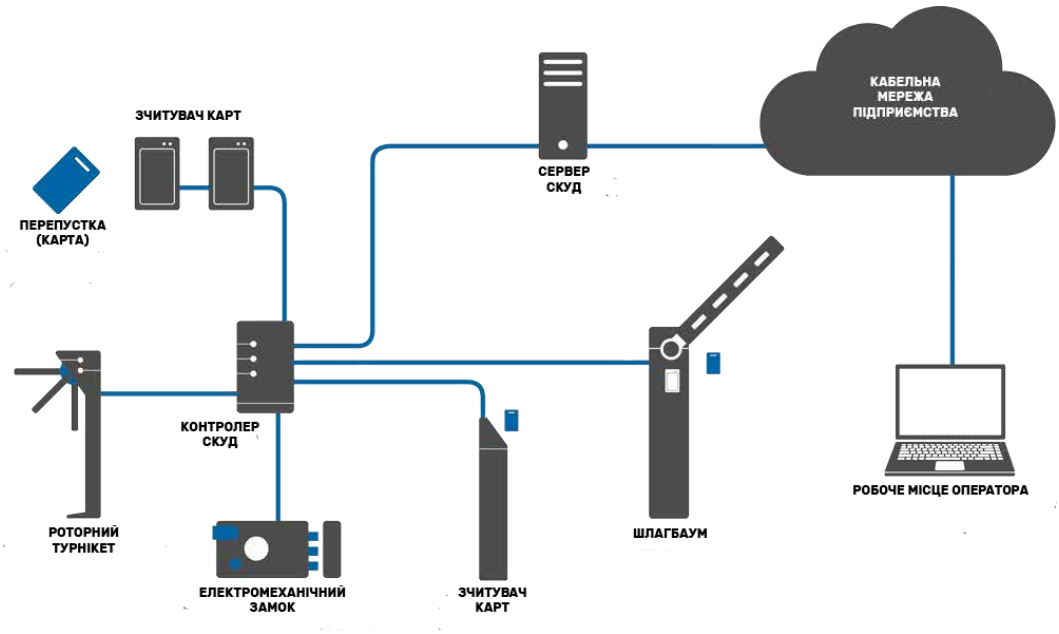


Рисунок 1.3 – Приклад використання карт доступу

4) Біометричні системи контролю:

Задля підвищення рівня безпеки дедалі частіше використовуються біометричні технології — сканування відбитків пальців, розпізнавання обличчя або навіть перевірка райдужки ока. Біометрія дозволяє ідентифікувати користувача з високою точністю, однак її застосування зазвичай потребує додаткових інвестицій та складнішого обслуговування.

Рисунок 1.4 – Приклад використання біометричного контролю

5) Інтегровані та комбіновані системи

Задля досягнення максимального рівня безпеки, на практиці часто впроваджують комплексні рішення, які поєднують декілька перелічених технологій. Наприклад, система може використовувати одночасно RFID-ідентифікацію та розпізнавання номерних знаків, або картки доступу та біометрію, що значно ускладнює несанкціонований доступ.

Такі сучасні системи дозволяють автоматизувати процес допуску, значно знизити ризик проникнення сторонніх осіб та забезпечити детальний облік усіх подій на контрольованій території [2].

1.3. Вибір компонентів для розробки системи доступу

Для розробки системи необхідний: модуль RFID, щоб при піднесені користувачем ключа-мітки, мікроконтролер міг її зчитати та передати дані на сервер, двигун який буде піднімати або опускати шлагбаум при піднесенні мітки, світловий індикатор доступу, який показуватиме червоне або зелене світло, сигналізуючи чи дозволено проїхати на територію чи ні.

В якості зчитувача RFID міток буде використовуватись RFID модуль PN532 NFC RFID. Він має сучасний приймач безконтактного зв'язку з частотою 13,56 МГц, дальність зв’язку до 6 см, живлення 3,3 В, низьке споживання струму – 26 мА [4].

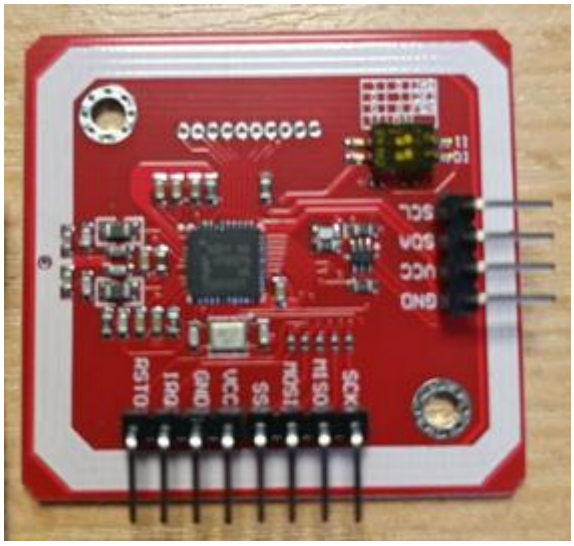


Рисунок 1.5 – зчитувач PN532 NFC RFID

Для приведення в рух шлагбауму буде використаний крокový двигун 28BYJ-48 з модулем драйвера Stepper Motor 28BYJ-48, оскільки ним можна відрегулювати кут та швидкість повороту, живлення 5 В [13].

Рисунок 1.6 – Крокový двигун 28BYJ-48 з модулем ULN2003

В якості світлового індикатора буде використано два світлодіоди, червоний та зелений. Їх живлення здійснюється від 5в через резистори[3].

Рисунок 1.7 – Світлодіод червоний/зелений

РОЗДІЛ 2 ПРАКТИЧНА ЧАСТИНА

2.1. Вибір мікроконтролера

Мікроконтролер — це компактний електронний пристрій, призначений для керування різними системами та пристроями на основі запрограмованих алгоритмів. Основна роль мікроконтролера полягає у виконанні завдань керування, обробки даних від сенсорів, взаємодії з зовнішніми пристроями, а також у забезпеченні обміну даними з іншими компонентами системи. На відміну від звичайних комп'ютерів, мікроконтролери створені для роботи з конкретними завданнями у реальному часі й часто використовуються у вбудованих системах, системах безпеки, автоматизації, а також в інтернеті речей (IoT).

Сьогодні на ринку існує багато різних платформ, серед яких кожна має свої особливості та переваги:

- **Raspberry Pi** — серія одноплатних комп'ютерів, що підтримують повноцінні операційні системи й широко застосовуються для прототипування, автоматизації та навчання. Raspberry Pi підходить для проектів, які потребують великої обчислювальної потужності або підтримки різних інтерфейсів підключення (див. Рисунок 2.1).

Рисунок 2.1 – Зовнішній вигляд плати RaspberryPi

- **Arduino** — платформа відкритого коду, яка завоювала популярність завдяки простоті використання та великій спільноті розробників. Плати Arduino ідеальні для початківців та прототипування автоматизованих пристроїв, підтримують багато різних сенсорів і виконавчих механізмів (див. Рисунок 2.2).

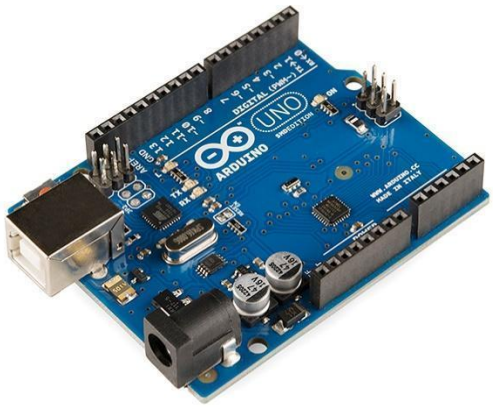


Рисунок 2.2 – Зовнішній вигляд плати Arduino

- **Orange Pi** — альтернатива Raspberry Pi, має схожий форм-фактор, набір периферії, можливість запуску різних ОС. Orange Pi підходить для проектів, де важлива бюджетність або потрібно більше портів вводу-виводу (див. Рисунок 2.3).

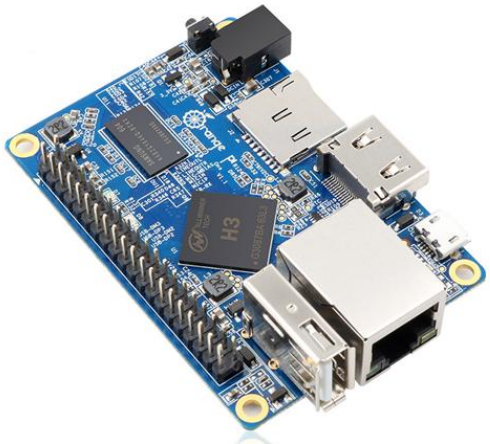


Рисунок 2.3 – Зовнішній вигляд плати Orange Pi

- **STM32** — це сімейство мікроконтролерів на базі ARM Cortex-M, які вирізняються високою продуктивністю, низьким енергоспоживанням та широкими можливостями для роботи з периферійними пристроями. STM32 часто використовують у промислових та критичних до надійності системах (див. Рисунок 2.4).

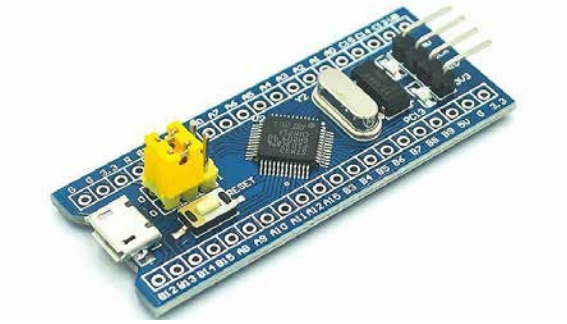


Рисунок 2.4 – Зовнішній вигляд плати STM32

- **ESP** (зокрема, ESP8266, ESP32) — серія сучасних мікроконтролерів із вбудованими модулями Wi-Fi і Bluetooth, що розроблені спеціально для застосувань в інтернеті речей (IoT). ESP-плати ідеальні для бездротового з'єднання з сервером чи віддаленим керуванням (див. Рисунок 2.5).

Рисунок 2.5 – Мікроконтролер ESP32

Після аналізу варіантів для розробки системи контролю доступу на приватну територію було обрано ESP32.
ESP32 — це мікроконтролер з двоядерним процесором, який підтримує одночасну роботу з декількома завданнями завдяки операційній системі реального часу. Він має широкий набір периферійних інтерфейсів (GPIO, UART, SPI, I2C, ADC, DAC, PWM), що дозволяє підключати різноманітні датчики та пристрої. Головною перевагою ESP32 є наявність вбудованих модулів Wi-Fi і Bluetooth, завдяки яким можна реалізувати віддалене підключення до серверу без необхідності прокладання кабелів.

Достатньо, щоб ESP32 знаходився в зоні покриття Wi-Fi та отримував живлення (наприклад, через стандартний блок живлення або портативну батарею). Це робить пристрій ідеальним для використання в автономних системах контролю доступу.

Серед недоліків можна відзначити дещо меншу кількість портів GPIO у порівнянні з класичними Arduino або STM32, однак для задачі контролю доступу (підключення RFID-зчитувача, крокового двигуна, світлодіодів тощо) функціоналу ESP32 цілком достатньо.

Таким чином, ESP32 був обраний як оптимальний варіант для побудови сучасної інформаційної системи контролю доступу завдяки своїй універсальності, підтримці бездротових технологій та простоті інтеграції з іншими пристроями системи.

2.2. Архітектура та комунікаційні протоколи

Для реалізації інформаційної системи обліку контролю доступу на приватну територію було проаналізовано різні архітектурні підходи та обрано оптимальний варіант з огляду на специфіку завдання, технічні можливості обладнання й потребу у простому адмініструванні.

Основні функціональні вимоги до системи:

- **Кожен користувач отримує RFID-мітку** з унікальним ідентифікатором.

- **Зчитування UID** відбувається за допомогою RFID-зчитувача, підключеного до мікроконтролера (ESP32).

- **Мікроконтролер** відправляє UID на сервер через Wi-Fi по HTTP-протоколу.

- **Центральний сервер** (на базі ОС Windows/Unix з OpenServer) приймає запит, перевіряє UID у базі даних MySQL.

- **У разі позитивної перевірки** сервер повертає відповідь "success", і контролер відкриває доступ (активує кроковий двигун на певний кут).

- **Вся інформація про події (ПІБ, UID, дата та час)** автоматично заноситься до журналу у базі даних.

- **Веб-інтерфейс** для адміністратора та директора забезпечує керування користувачами, перегляд журналу, а також експорт звітів.

Обрана архітектура — централізована (клієнт-серверна)

- **Центральний сервер:** виконує функції зберігання та обробки даних, адміністрування системи, журналювання доступу.
- **База даних MySQL**: містить таблиці користувачів, журналів доступу, налаштувань тощо.
- **Мікроконтролер ESP32**: виступає в ролі клієнта, здійснює зчитування UID та відправку запитів на сервер.
- **RFID-зчитувач PN532 (SPI)**: приймає картку користувача, передає UID на ESP32.
- **Кроковий двигун**: приводить у дію механізм відкриття/закриття (наприклад, шлагбаум, замок, турнікет).
- **Веб-інтерфейс (PHP/HTML/CSS/JS)**: адмініструє користувачів і дозволи, дає змогу переглядати журнал подій, формувати звіти.

Переваги централізованої архітектури:

- Простота впровадження та адміністрування.
- Одна точка входу для керування всією системою.
- Легке резервне копіювання даних та централізований моніторинг подій.
- Зручність розширення функціоналу через веб-інтерфейс.

Недоліки:

- Вся система залежить від надійності центрального сервера та стабільності Wi-Fi-мережі.

- У разі відсутності зв'язку з сервером — доступ не надається.

Опис комунікаційних протоколів

- **Зв’язок між ESP32 та сервером здійснюється через протокол HTTP (GET-запити).**

Наприклад:

http://access-system.local/verify_rfid.php?rfid=7CDD3C30

Сервер повертає JSON-відповідь зі статусом перевірки UID.

- **Обмін даними між клієнтом і сервером** є синхронним — після відправки UID ESP32 чекає на відповідь, і лише після її отримання приймає рішення щодо відкриття доступу.

- **Зберігання та обробка даних** реалізовані через СУБД MySQL, де кожен факт проходу, UID, час та ПІБ користувача фіксуються у таблиці журналу.

Чому обрано саме цю архітектуру?

Для задачі контролю доступу на приватній території централізована архітектура зручна тим, що вся логіка контролю і журналювання винесена на сервер. Це полегшує адміністрування, дозволяє централізовано додавати/видаляти користувачів та забезпечує повний контроль над всіма подіями на об'єкті. У разі необхідності систему легко масштабувати — додати нові зчитувачі, пристрої або функції, залишаючи всю логіку обробки на сервері.

Загальна схема комунікації:

RFID-картка (UID) → Зчитувач PN532 → ESP32 → Wi-Fi → HTTP-запит → Сервер (PHP/MySQL) → Відповідь → Крокový двигун

Хочу зауважити, що дана архітектура оптимальна для малих та середніх об'єктів, де важлива надійність і централізоване адміністрування, а також забезпечує легкість у розширенні функцій (наприклад, додавання інтеграції з мобільним застосунком або хмарними сервісами в майбутньому).

2.3. Вибір програмного забезпечення для написання коду

Для розробки програмного забезпечення системи контролю доступу на базі мікроконтролера ESP32 було обрано **Arduino IDE** (Integrated Development Environment). Це зручне та безкоштовне середовище розробки, яке підтримує не лише платформи Arduino, а й багато інших популярних мікроконтролерів, зокрема ESP8266 та ESP32. [6]

Arduino IDE має інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати, редагувати та завантажувати код на мікроконтролер. Середовище підтримує мову програмування C/C++ з додатковими бібліотеками для роботи з різними периферійними пристроями, включаючи RFID-зчитувачі (Adafruit_PN532), модулі Wi-Fi, крокові двигуни (AccelStepper) тощо.

Для налаштування взаємодії мікроконтролера з сервером також використовуються бібліотеки для роботи з HTTP-запитами та серіальним портом, що дозволяє організувати обмін даними з сервером через мережу Wi-Fi.

Для розробки веб-інтерфейсу (панель керування користувачами, журнал подій, налаштування тощо) застосовується стек технологій **PHP, HTML, CSS, JavaScript**.

Розгортання серверної частини виконується через OpenServer або XAMPP, які дозволяють швидко розгорнути локальний веб-сервер з підтримкою MySQL і PHP.

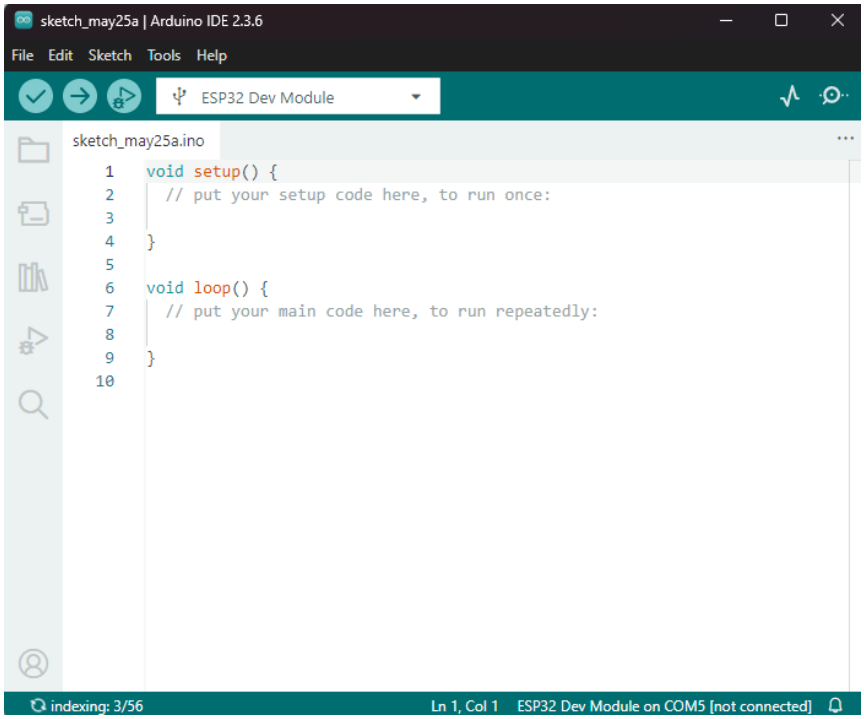


Рисунок 2.6 – Інтерфейс програми Arduino IDE

2.4. Програмування мікроконтролера системи доступу автотранспорта на територію з обмеженим доступом

Розглянемо структуру коду на ESP32, який безпосередньо контактує з датчиками та веб-інтерфейсом

```
#include <WiFi.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_PN532.h>
#include <AccelStepper.h>
```

Рисунок 2.7 – Підключення бібліотек

Програмування мікроконтролера ESP32 здійснюється в середовищі розробки Arduino IDE. У проєкті використовується низка спеціалізованих бібліотек для забезпечення коректної роботи обладнання:

- WiFi.h — для підключення мікроконтролера до мережі Wi-Fi;

- SPI.h — для організації обміну даними по SPI між ESP32 і RFID-зчитувачем PN532;

- Adafruit_PN532.h — для роботи з RFID-зчитувачем (зчитування UID мітки);

- AccelStepper.h — для керування кроковим двигуном через драйвер ULN2003;

- ArduinoJson.h — для роботи з відповідями сервера у форматі JSON.

Структура коду

На початку коду здійснюється підключення необхідних бібліотек, оголошення змінних для зберігання даних Wi-Fi, налаштування пінів для RFID-зчитувача, крокового двигуна та параметрів сервера (див. Рисунок 2.8).

```
// --- WiFi ---
const char* ssid = "TP-LINK_C948";
const char* password = "60945419";

// --- Сервер ---
const char* host = "192.168.0.103 // IP сервера
const uint16_t port = 80;           // HTTP

// --- PN532 SPI ---
#define PN532_SCK  (18)
#define PN532_MOSI (23)
#define PN532_SS   (5)
#define PN532_MISO (19)
Adafruit_PN532 nfc(PN532_SCK, PN532_MISO, PN532_MOSI, PN532_SS);

// --- Stepper (28BYJ-48 + ULN2003) ---
#define IN1 12
#define IN2 13
#define IN3 14
#define IN4 15
AccelStepper stepper(AccelStepper::HALF4WIRE, IN1, IN3, IN2, IN4);
```

Рисунок 2.8 – Структура

Підключення до Wi-Fi

Використовується стандартна ініціалізація підключення до Wi-Fi у функції setup(). Код забезпечує підключення до локальної мережі, яка надає доступ до серверу з базою даних.

Зчитування RFID-мітки

Після ініціалізації, у головному циклі loop(), відбувається опитування RFID-зчитувача. При піднесенні RFID-мітки UID зчитується і формується hex-рядок.

```
Serial.println("Очікування RFID мітки...");
uint8_t uid[7]; uint8_t uidLength;
if (nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength)) {
  // UID в hex-рядок
  String rfidKey = "";
  for (uint8_t i = 0; i < uidLength; i++) {
    if (uid[i] < 0x10) rfidKey += "0";
    rfidKey += String(uid[i], HEX);
  }
  rfidKey.toUpperCase();

  Serial.print("UID: "); Serial.println(rfidKey);

  // --- Перевіряємо на сервері ---
  if (checkUIDwithServer(rfidKey)) {
    Serial.println("Ключ валідний, обертаємо!");
    // 120 градусів вперед
    stepper.move(STEPS_120);
    while (stepper.distanceToGo() != 0) stepper.run();
    delay(3000);
    // Повертаємось назад
    stepper.move(-STEPS_120);
    while (stepper.distanceToGo() != 0) stepper.run();
  } else {
    Serial.println("Ключ не знайдено.");
  }

  // --- Очікуємо 5 сек до повторного зчитування ---
  delay(5000);
}
```

Рисунок 2.9 – Зчитування мітки

Перевірка ключа через сервер

Мікроконтролер надсилає GET-запит на сервер, передаючи UID мітки у параметрах запиту. Сервер перевіряє наявність такого UID у базі даних і повертає результат у форматі JSON. Відповідь аналізується в коді ESP32.

Реакція на валідний ключ

Якщо ключ знайдено в базі — система активує кроковий двигун (відкриває шлагбаум), чекає 3 секунди та повертає двигун у вихідне положення. Якщо ключ не знайдено — система просто очікує на наступну мітку, ніяких дій не виконується.

Алгоритм роботи програми:

1. Ініціалізація обладнання та Wi-Fi з’єднання.

2. Очікування піднесення RFID-мітки.

3. Зчитування UID мітки та формування hex-рядка.

4. Надсилання UID на сервер через HTTP-запит.

5. Аналіз відповіді сервера:

- якщо ключ знайдено, відкривається шлагбаум (кроковий двигун на 120 градусів вперед), фіксується у журналі подія доступу;

- через 3 секунди шлагбаум повертається у вихідне положення.

- якщо ключ не знайдено — ніяких дій не відбувається.

6. Після обробки одного зчитування — пауза 5 секунд для запобігання повторному зчитуванню тієї ж мітки.

Фрагмент коду зчитування мітки:

```
void setup() {
  Serial.begin(115200);

  // --- Stepper ---
  stepper.setMaxSpeed(500);
  stepper.setAcceleration(200);

  // --- PN532 ---
  nfc.begin();
  if (!nfc.getFirmwareVersion()) {
    Serial.println("❌ PN532 not found!");
    while (1);
  }
  Serial.println("✅ PN532 detected.");
  nfc.SAMConfig();

  // --- WiFi ---
  WiFi.disconnect(true);
  delay(1000);
  WiFi.begin(ssid, password);
  Serial.print("Connecting WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(300);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected.");
}

void loop() {
  stepper.run(); // обов'язково викликати завжди

  if (stepper.distanceToGo() != 0) {
    // Двигун ще крутиться, не зчитуємо мітку
    delay(20);
    return;
  }

  Serial.println("Очікування RFID мітки...");
```

Рисунок 2.10 - отримання інформації з мітки

Надсилання HTTP-запиту на сервер для перевірки RFID

Після зчитування UID з RFID-мітки ESP32 відправляє HTTP GET-запит на сервер для перевірки ключа.

```
bool checkUIDwithServer(String rfidKey) {
  WiFiClient client;
  String url = "/verify_rfid.php?rfid=" + rfidKey; // PHP-роутер
  Serial.print("Запит до: "); Serial.println(url);

  if (!client.connect(host, port)) {
    Serial.println("Помилка підключення до сервера");
    return false;
  }

  // --- HTTP GET ---
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");

  unsigned long timeout = millis();
  String response = "";
  while (client.connected() && millis() - timeout < 3000) {
    while (client.available()) {
      char c = client.read();
      response += c;
    }
  }
  client.stop();

  // --- Знаходимо JSON у відповіді ---
  int jsonStart = response.indexOf('{');
  int jsonEnd = response.lastIndexOf('}');
  if (jsonStart == -1 || jsonEnd == -1) {
    Serial.println("JSON not found in response.");
    return false;
  }
  String json = response.substring(jsonStart, jsonEnd + 1);
  Serial.print("Сервер відповів: "); Serial.println(json);

  // --- Простий аналіз без ArduinoJson ---
  if (json.indexOf("\"status\":\"success\"") != -1) {
    return true;
  }
  return false;
}
```

Рисунок 2.11 - Функція перевірки UID через HTTP-запит до сервера

Обробка відповіді та управління кроковим двигуном

Після отримання позитивної відповіді з сервера кроковий двигун відкриває та закриває шлагбаум:

```
if (checkUIDwithServer(rfidKey)) {
  Serial.println("Ключ валідний, обертаємо!");
  // 120 градусів вперед
  stepper.move(STEPS_120);
  while (stepper.distanceToGo() != 0) stepper.run();
  delay(3000);
  // Повертаємось назад
  stepper.move(-STEPS_120);
  while (stepper.distanceToGo() != 0) stepper.run();
} else {
  Serial.println("Ключ не знайдено.");
}
```

Рисунок 2.12 – Відкривання шлагбауму при успішній перевірці RFID-мітки

У процесі програмування системи контролю доступу на базі ESP32 для ефективного управління пристроями та обміну даними з сервером були оголошені ключові константи та змінні. Їх правильне налаштування забезпечує коректну роботу контролера з мережевим підключенням, обробкою RFID-міток і керуванням кроковим двигуном. Нижче наведено основні змінні, використані у програмному коді пристрою, а також їх короткий опис.

Змінна	**Опис**
ssid	Ім’я Wi-Fi мережі, до якої підключається ESP32
password	Пароль Wi-Fi мережі
PN532_SCK	Пін SPI-клоку для зчитувача PN532
stepper	Об’єкт керування кроковим двигуном
rfidKey	UID-мітки, переведений у рядок для передачі на сервер
FORWARD	Напрям руху крокового мотору вперед
REVERSE	Напрям руху крокового мотору назад

2.5. Розробка системи доступу на приватну територію

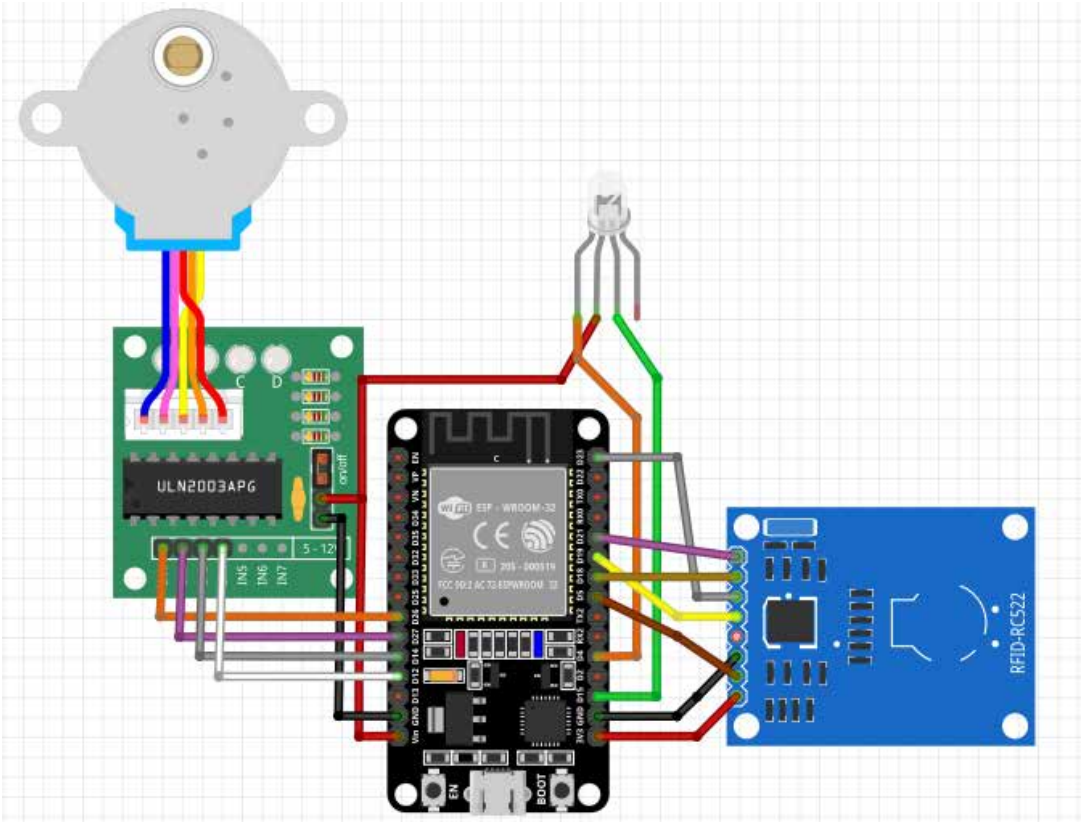


Рисунок 2.13 – Схема підключення датчиків з мікроконтролером

Рисунок 2.14 – Блок-схема роботи програми

2.6. Створення бази даних на сервері

Для реалізації інформаційної системи контролю доступу на приватну територію необхідно створити базу даних, яка забезпечить зберігання інформації про користувачів, їх RFID-мітки, а також журнал подій доступу. Для цього використовується веб-застосунок phpMyAdmin, що надає зручний інтерфейс для адміністрування бази даних MySQL.

Для початку роботи слід авторизуватися в phpMyAdmin під обліковим записом адміністратора (логін – root, пароль – порожній).

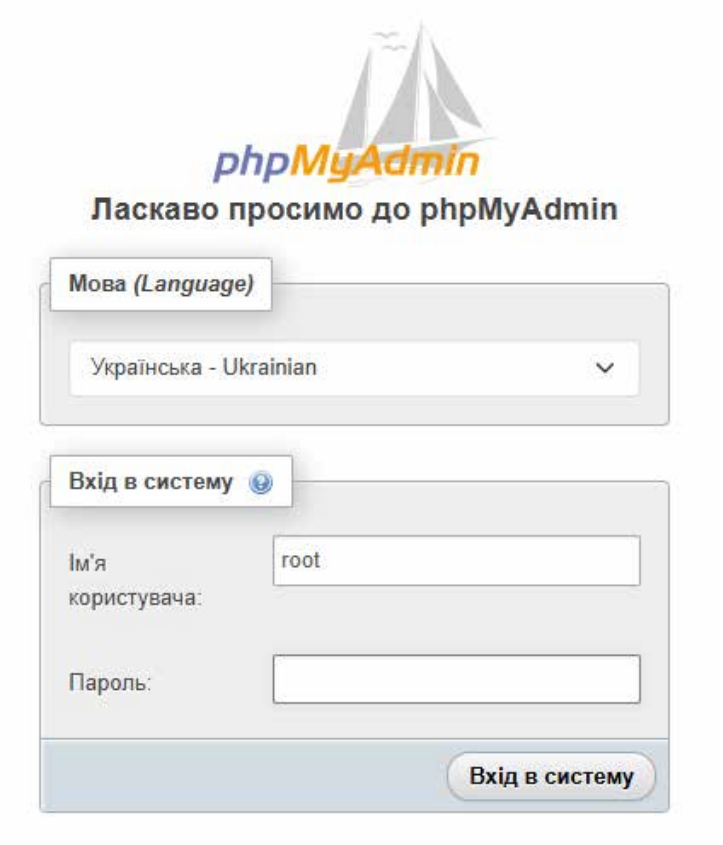


Рисунок 2.15 – Авторизація в phpMyAdmin

Далі, натиснувши кнопку Нова, створюється база даних з назвою, наприклад, access_system. У цій базі даних створюються такі основні таблиці: person_info, tags, access_log.

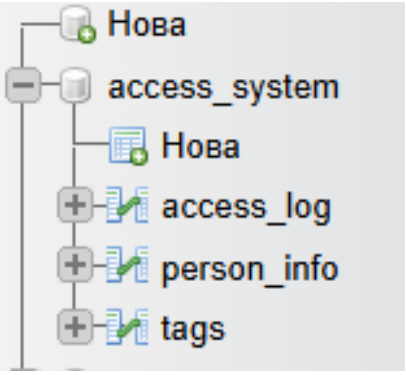


Рисунок 2.16 – Створена база даних esp32

Структура таблиць бази даних:

Таблиця person_info:

- id (INT, AUTO_INCREMENT, PRIMARY KEY) – унікальний ідентифікатор користувача;

- full_name (VARCHAR) – повне ім’я користувача;

- role (ENUM) – роль користувача (наприклад, admin, worker, director);

- tag_uid (VARCHAR) – RFID-ключ користувача;

- password (VARCHAR) – пароль користувача (для адміністрування).

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
1	id	int			Ні	*Немає*		AUTO_INCREMENT	Змінити Знищити Більше
2	full_name	varchar(100)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
3	role	enum('admin', 'director', 'guard', 'worker', 'user...	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
4	tag_uid	varchar(20)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше
5	password	varchar(255)	utf8mb4_unicode_ci		Так	*NULL*			Змінити Знищити Більше

Позначити все *Вибрані:* Переглянути Змінити Знищити Первинний Унікальне Індекс Просторовий ПовнТекст

Рисунок 2.17 – Структура таблиці person_info

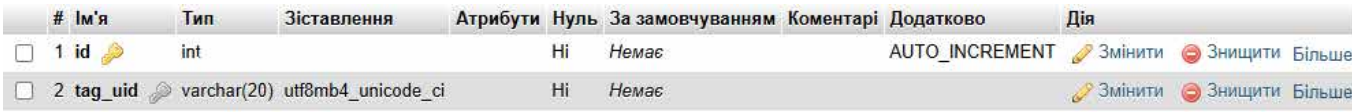
id	full_name	role	tag_uid	password
1	Пуха Максим Миколайович	admin	7CDD3C30	passwordmanager
2	Бровчук Дмитро Іванович	director	4370DF18	brovchukdir
3	Сухомлин Олексій Олександрович	worker	EE1E3F59	sosworkerpassword
6	Колеба Олег Павлович	worker	AA145FP7	*NULL*

Рисунок 2.18 - Заповнена таблиця person_info

Таблиця tags:

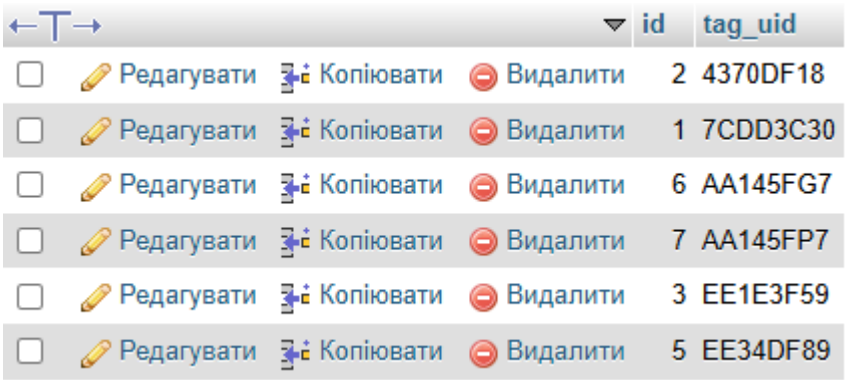
- id (INT, AUTO_INCREMENT, PRIMARY KEY) – унікальний ідентифікатор мітки;

- tag_uid (VARCHAR) – унікальний RFID-ключ.



#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
1	id	int			Ні	*Немає*		AUTO_INCREMENT	Змінити Знищити Більше
2	tag_uid	varchar(20)	utf8mb4_unicode_ci		Ні	*Немає*			Змінити Знищити Більше

Рисунок 2.19 – Структура таблиці tags



				id	tag_uid
☐	Редагувати	Копіювати	Видалити	2	4370DF18
☐	Редагувати	Копіювати	Видалити	1	7CDD3C30
☐	Редагувати	Копіювати	Видалити	6	AA145FG7
☐	Редагувати	Копіювати	Видалити	7	AA145FP7
☐	Редагувати	Копіювати	Видалити	3	EE1E3F59
☐	Редагувати	Копіювати	Видалити	5	EE34DF89

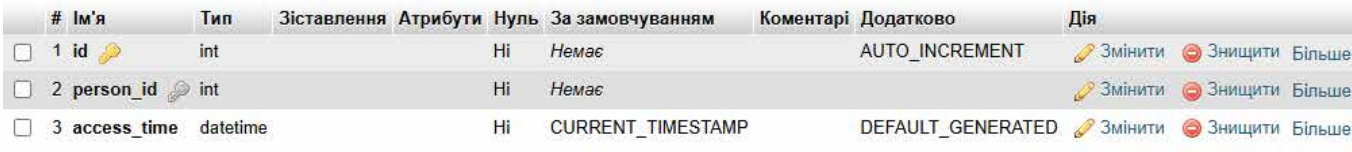
Рисунок 2.20 - Заповнена таблиця tags

Таблиця access_log:

- id (INT, AUTO_INCREMENT, PRIMARY KEY) – унікальний ідентифікатор запису;

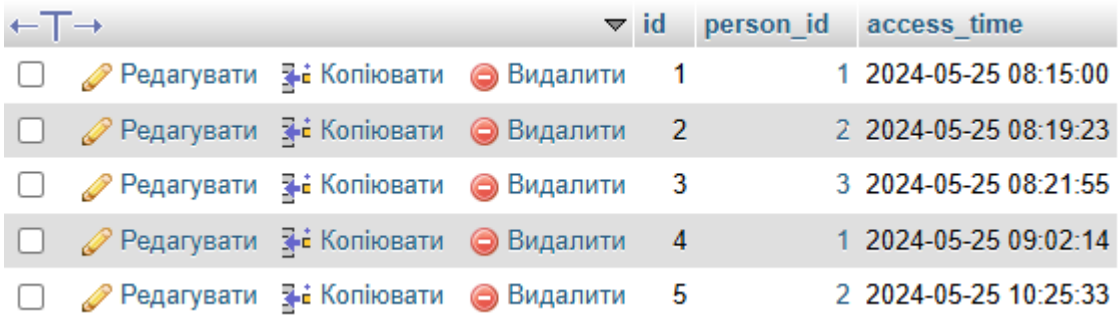
- person_id (INT) – зовнішній ключ, що пов’язує подію з користувачем;

- access_time (DATETIME) – дата і час події доступу.



#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
1	id	int			Ні	*Немає*		AUTO_INCREMENT	Змінити Знищити Більше
2	person_id	int			Ні	*Немає*			Змінити Знищити Більше
3	access_time	datetime			Ні	CURRENT_TIMESTAMP		DEFAULT_GENERATED	Змінити Знищити Більше

Рисунок 2.21 – Структура таблиці access_log



	id	person_id	access_time
Редагувати Копіювати Видалити	1	1	2024-05-25 08:15:00
Редагувати Копіювати Видалити	2	2	2024-05-25 08:19:23
Редагувати Копіювати Видалити	3	3	2024-05-25 08:21:55
Редагувати Копіювати Видалити	4	1	2024-05-25 09:02:14
Редагувати Копіювати Видалити	5	2	2024-05-25 10:25:33

Рисунок 2.22 - Заповнена таблиця access_log

Таблиця open_control містить три стовпці: count – кількість відкривань, rfid_key – ключ rfid мітки, користувача date – поточна дата відкриття шлагбаума.

Структура таблиці Вид відносин

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	**count**	int			Ні	*Немає*		AUTO_INCREMENT
2	**rfid_key**	varchar(50)	*utf8mb4_0900_ai_ci*		Ні	*Немає*		
3	**date**	datetime			Ні	CURRENT_TIMESTAMP		DEFAULT_GENERATED

Рисунок 2.22 – Структура таблиці open_control

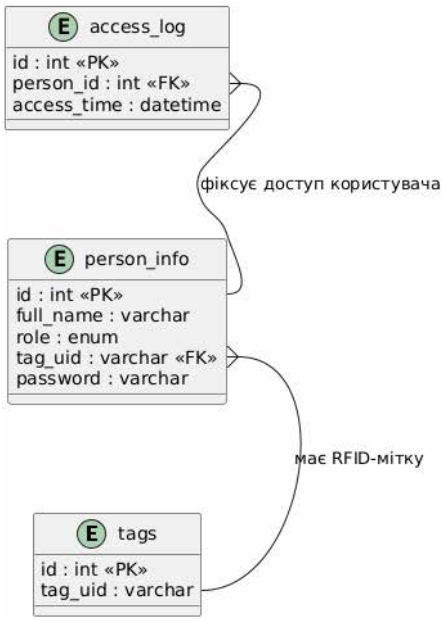


Рисунок 2.24 – Структура бази даних access_system

2.7. Обмін даних серверу та web-інтерфейсу

Для забезпечення роботи інформаційної системи контролю доступу на приватну територію необхідно організувати ефективний обмін даними між сервером і web-інтерфейсом. Для цього на сервері створюються PHP-скрипти, які працюють із базою даних та забезпечують отримання, додавання, редагування і видалення інформації.

Всі основні сторінки, що взаємодіють із базою даних, зберігаються у кореневому каталозі сайту (наприклад, у папці home/access-system.local/ на сервері OpenServer).

Підключення до бази даних

Підключення до бази даних здійснюється через окремий файл db.php. У цьому файлі задаються параметри підключення: адреса сервера MySQL, логін, пароль та назва бази даних. Якщо підключення невдале, скрипт видає повідомлення про помилку.

```php
<?php
$host = "MySQL-8.4";
$user = "root";
$password = "";
$dbname = "access_system";

$conn = new mysqli($host, $user, $password, $dbname);
if ($conn->connect_error) {
    die("Помилка підключення до БД: " . $conn->connect_error);
}
$conn->set_charset("utf8mb4"); // Додаємо кодування
?>
```

Рисунок 2.25 - файл bd.php

Отримання даних із бази для відображення у web-інтерфейсі

Основний web-інтерфейс (наприклад, journal.php або index.php) звертається до бази даних, виконує SQL-запит для вибірки потрібної інформації й формує таблицю для відображення користувачів, логів доступу тощо.

```php
<?php
session_start();
require_once 'db.php';
if (!isset($_SESSION['user_id'])) {
    header('Location: index.php');
    exit();
}
$query = "SELECT access_log.id, person_info.full_name, tags.tag_uid, access_log.access_time
          FROM access_log
          JOIN person_info ON access_log.person_id = person_info.id
          JOIN tags ON person_info.tag_uid = tags.tag_uid
          ORDER BY access_log.access_time DESC";
$result = $conn->query($query);
```

Рисунок 2.26 – Код отримання інформації про події доступу з бази даних

Виведення інформації у вигляді таблиці

Для зручного відображення даних у web-інтерфейсі формується HTML-таблиця із відповідними заголовками та рядками.

```php
echo "<table>";
echo "<tr><th>№</th><th>ПІБ</th><th>Ключ (UID)</th><th>Дата та час</th></tr>";
$n = 1;
while ($row = $result->fetch_assoc()) {
    echo "<tr>";
    echo "<td>" . $n++ . "</td>";
    echo "<td>" . htmlspecialchars($row['full_name']) . "</td>";
    echo "<td>" . htmlspecialchars($row['tag_uid']) . "</td>";
    echo "<td>" . htmlspecialchars($row['access_time']) . "</td>";
    echo "</tr>";
}
echo "</table>";
```

Рисунок 2.27 – Код відображення подій доступу у вигляді HTML-таблиці

Додавання, редагування та видалення користувачів

Для зміни інформації у базі даних використовуються окремі форми й скрипти (наприклад, add_user.php, edit_user.php, delete_user.php). Дані передаються методом POST або GET.

```php
// Додаємо користувача
$add = $conn->prepare("INSERT INTO person_info (full_name, tag_uid, role) VALUES (?, ?, ?)");
$add->bind_param("sss", $full_name, $tag_uid, $role);
$res = $add->execute();
$add->close();

if ($res) {
    echo json_encode(['status' => 'success', 'msg' => 'Користувача успішно додано']);
} else {
    echo json_encode(['status' => 'error', 'msg' => 'Помилка при додаванні користувача']);
}
exit();
```

Рисунок 2.28 - Додавання користувача

Цей код відповідає за додавання нового користувача до бази даних. За допомогою підготовленого SQL-запиту (INSERT INTO ... VALUES (?, ?, ?)), дані нового користувача (ПІБ, ключ RFID, роль) безпечно зберігаються у відповідній таблиці. Після успішного виконання запиту відправляється повідомлення про успішне додавання; у випадку помилки повертається відповідний статус.

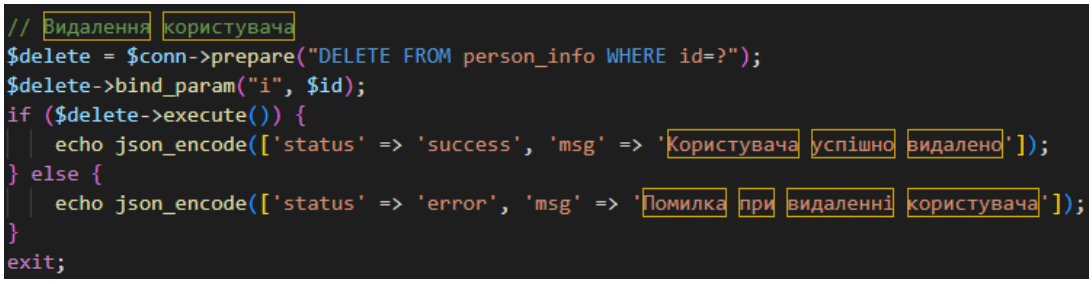


Рисунок 2.29 - Видалення користувача

Даний фрагмент забезпечує видалення користувача з бази даних за унікальним ідентифікатором (id). Код використовує підготовлений запит для видалення, що захищає від SQL-ін'єкцій. Успішне видалення підтверджується відповідним JSON-повідомленням. Якщо виникає помилка — система повідомляє про це.

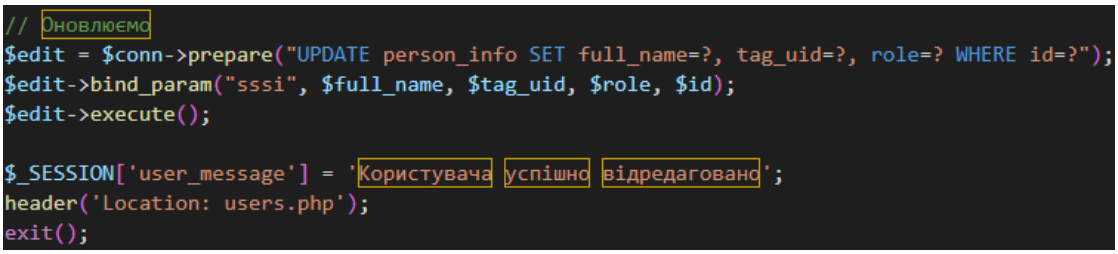


Рисунок 2.30 - Редагування користувача

У цьому прикладі здійснюється оновлення (редагування) даних користувача у базі. Код змінює ПІБ, ключ RFID та роль користувача за його унікальним ідентифікатором (id). Після оновлення даних користувача повертається повідомлення про успішне редагування, і користувач перенаправляється назад до списку користувачів.

Формування та завантаження звіту

Для формування звіту за журналом подій створюється окремий скрипт (наприклад, download_journal.php), який формує файл у форматі Excel або CSV для подальшого завантаження.

```php
// --- Витягуємо записи журналу ---
$sql = "SELECT access_log.id, person_info.full_name, tags.tag_uid, access_log.access_time
        FROM access_log
        JOIN person_info ON access_log.person_id = person_info.id
        JOIN tags ON person_info.tag_uid = tags.tag_uid
        ORDER BY access_log.access_time DESC";
$result = $conn->query($sql);

// --- Відправляємо CSV-заголовки ---
header('Content-Type: text/csv; charset=utf-8');
header('Content-Disposition: attachment; filename="journal.csv"');

$output = fopen('php://output', 'w');

// --- Пишемо перший рядок (заголовки) ---
fputcsv($output, ['№', 'ПІБ', 'Ключ (UID)', 'Дата та час']);

// --- Пишемо дані ---
$n = 1;
if ($result && $result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        fputcsv($output, [
            $n++,
            $row['full_name'],
            $row['tag_uid'],
            $row['access_time']
        ]);
    }
} else {
    // Якщо записів немає — пишемо порожній рядок
    fputcsv($output, ['', 'Немає даних', '', '']);
}
```

Рисунок 2.31 - Код для експорту журналу у CSV

Для забезпечення безпеки та захисту даних в інформаційній системі реалізовано механізм авторизації. Доступ до функціоналу керування (додавання, видалення, редагування користувачів, перегляд журналу подій) надається лише після введення коректного логіна та пароля адміністратора. Авторизація здійснюється через окрему форму входу (наприклад, файл login.php), де користувач вводить логін і пароль.

```php
// Запит до БД для перевірки користувача
$stmt = $conn->prepare("SELECT * FROM person_info WHERE full_name = ? AND password = ?");
$stmt->bind_param("ss", $full_name, $password);
$stmt->execute();
$result = $stmt->get_result();

if ($user = $result->fetch_assoc()) {
    // Авторизація успішна
    $_SESSION['user_id'] = $user['id'];
    $_SESSION['full_name'] = $user['full_name'];
    $_SESSION['role'] = $user['role'];
    header('Location: users.php');
    exit();
} else {
    // Помилка авторизації
    $_SESSION['login_error'] = "Невірний ПІБ або пароль!";
    header('Location: index.php');
    exit();
}
```

Рисунок 2.32 – Фрагмент коду авторизації login.php

Після успішної авторизації у сесії встановлюється ідентифікатор користувача, і він отримує доступ до всіх функцій системи. При спробі доступу без авторизації користувач автоматично перенаправляється на сторінку входу.

Отримання та перевірка RFID-мітки через сервер

У системі для ідентифікації користувачів використовується RFID-технологія. Кожна мітка має унікальний ідентифікатор (UID), який зчитується контролером (наприклад, ESP32 із підключеним модулем PN532).

Алгоритм взаємодії:

1. Користувач прикладає RFID-картку до зчитувача.

2. Контролер отримує UID мітки та формує HTTP-запит до серверного скрипта (наприклад, verify_rfid.php).

3. Серверний скрипт перевіряє наявність UID у базі даних, знаходить відповідного користувача, і повертає результат у форматі JSON.

4. У випадку позитивної відповіді керується виконавчий механізм (кроковий двигун відкриває прохід або шлагбаум), а інформація про доступ записується у журнал.

```php
<?php
header('Content-Type: application/json');
require_once 'db.php';

// Отримуємо tag_uid з GET або POST-запиту
$tag_uid = '';
if (isset($_GET['tag_uid'])) {
    $tag_uid = $_GET['tag_uid'];
} elseif (isset($_POST['tag_uid'])) {
    $tag_uid = $_POST['tag_uid'];
}

if ($tag_uid == '') {
    echo json_encode(['status' => 'error', 'message' => 'Немає tag_uid']);
    exit();
}
```

Рисунок 2.33 – Фрагмент коду отримання мітки

2.8. Налаштування серверу

Для роботи з сервером, було встановлено програму OpenServerPanel.

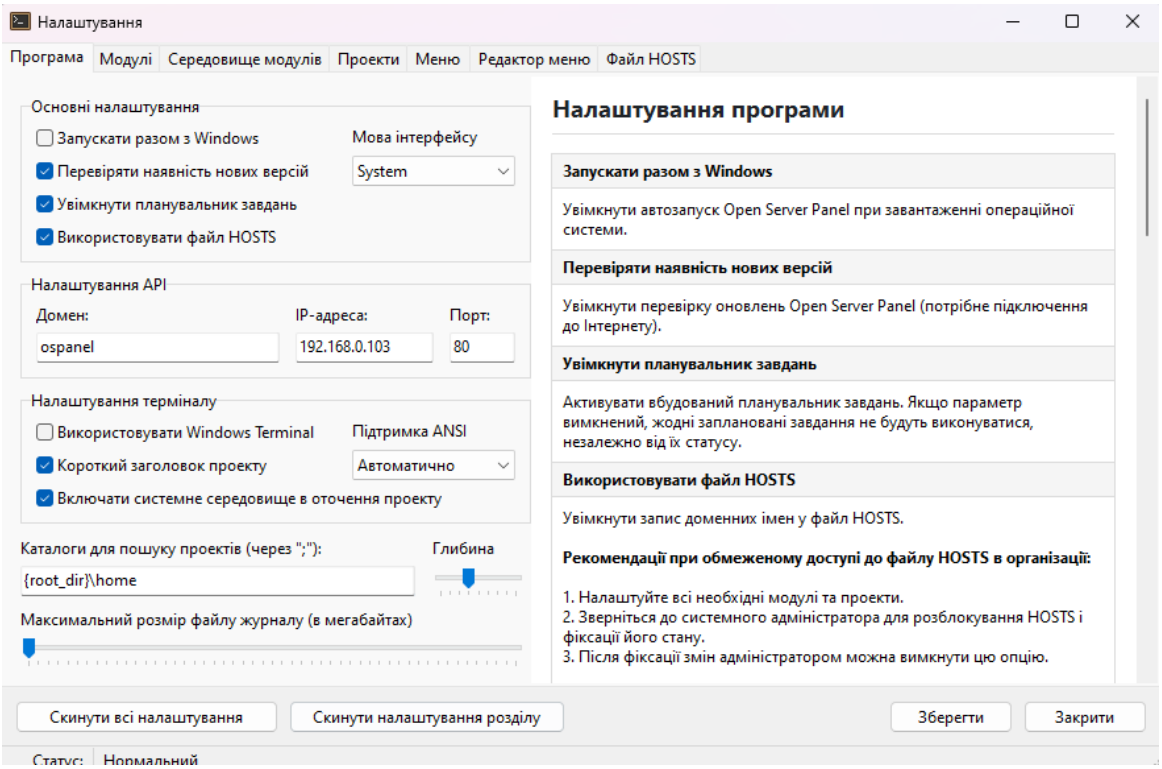


Рисунок 2.34 – Вікно налаштувань OpenServerPanel

Змінюємо IP-адресу сервера, на адресу IP-адресу мережевого адаптера, а саме 192.168.0.103 .

Розсташування файлів проекту:

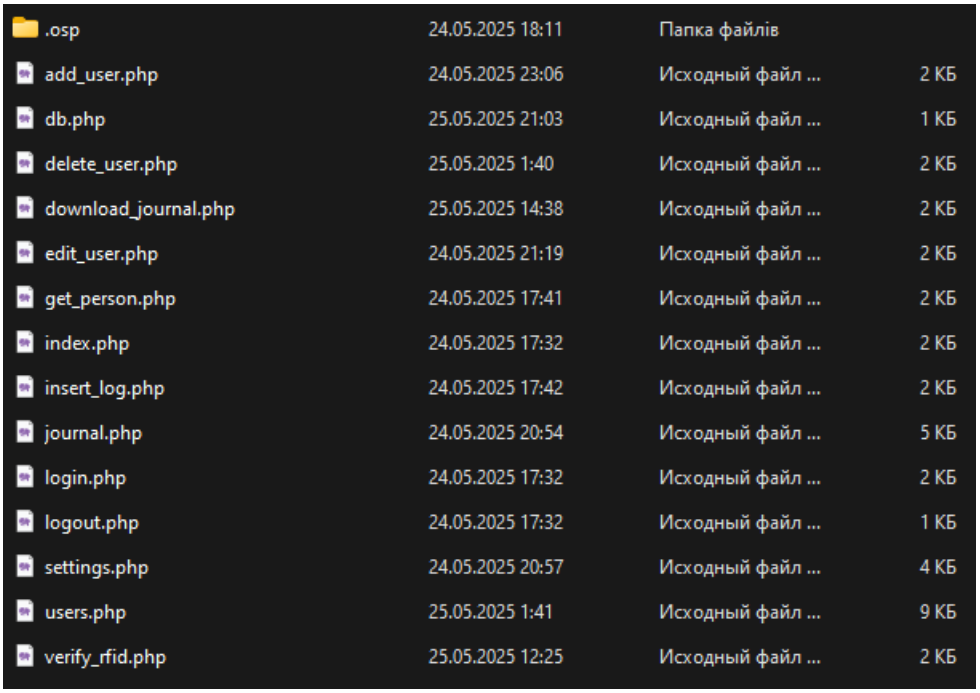


Рисунок 2.35 - файли для роботи проекту access_system.local

РОЗДІЛ 3 ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ

3.1. Використання web-інтерфейсу

У розробленій інформаційній системі контролю доступу на приватну територію для адміністрування, моніторингу та керування доступом використовується зручний web-інтерфейс. Для роботи з системою адміністратор відкриває веб-браузер та в адресному рядку вводить адресу сервера (наприклад, http://access-system.local/), після чого потрапляє на головну сторінку системи.

На веб-інтерфейсі відображаються основні вкладки:

- **Журнал** – сторінка для перегляду всіх подій доступу, які містять ПІБ користувача, ключ (UID), а також дату та час події.

- **Список користувачів** – вкладка для перегляду, додавання, редагування та видалення користувачів. Тут відображається перелік усіх зареєстрованих користувачів із зазначенням їхніх ролей і ключів доступу.

- **Налаштування** – сторінка для перегляду основної інформації про розробника та контактів для технічної підтримки.

Адміністратор має змогу швидко знаходити необхідного користувача, редагувати або видаляти записи. Додавання нового користувача здійснюється через відповідну форму, де вказуються ПІБ, ключ (UID) та роль користувача.

Крім цього, система дозволяє експортувати журнал подій у вигляді Excel-файлу для подальшого аналізу або архівування. Для цього достатньо натиснути кнопку "Завантажити звіт" на сторінці журналу, після чого сформується файл із усіма подіями за період.

3.2 Перегляд сторінок інформаційної системи

Сторінка авторизації

Опис:

Сторінка авторизації є початковою точкою входу до інформаційної системи контролю доступу. На ній користувач вводить логін та пароль адміністратора для отримання доступу до системи. Без успішної авторизації доступ до інших сторінок буде заборонено, що забезпечує базовий рівень безпеки.

Функції:

- Введення логіна та пароля.
- Перевірка даних на сервері.
- Перенаправлення до основного інтерфейсу у разі успіху.

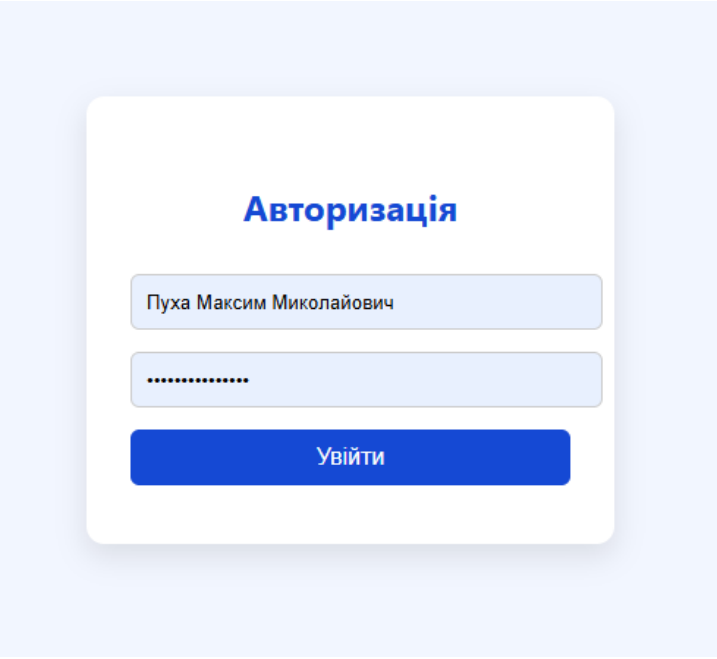


Рисунок 3.36 - Сторінка авторизації

Навігація та кнопка вийти

Після вдалого проходження авторизації користувач потрапляє на головну сторінку - Список користувачів, вгорі є навігація між вкладками - Журнал, Список користувачів, налаштування. Також для здійснення виходу з системи було створено кнопку - Вихід.

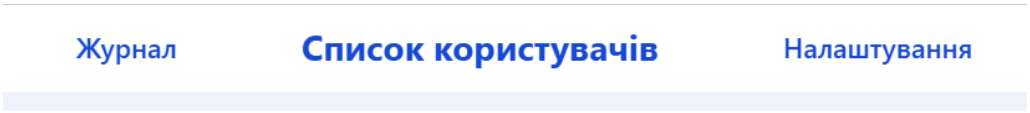


Рисунок 3.37 - Навігація

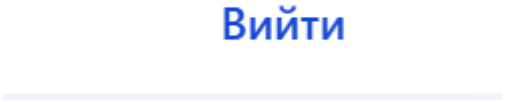


Рисунок 3.38 - Кнопка Вийти

Сторінка Список користувачів

Опис:

Ця сторінка призначена для перегляду, додавання, редагування та видалення користувачів, які мають право доступу на територію. Інтерфейс поділено на дві основні частини: зліва – форма для додавання нового користувача (вказуються ПІБ, ключ та роль), справа – форма для редагування обраного користувача з можливістю його видалення. Вгорі розташована таблиця зі списком усіх користувачів, їх ролями та UID-ключами.

Як працює:

- Оберіть користувача в таблиці — дані автоматично підтягуються у форму редагування.

- Для додавання нового користувача заповніть форму зліва і натисніть “Додати”.
- Для редагування змініть дані у формі справа і натисніть “Зберегти”.
- Для видалення натисніть “Видалити”.

Рисунок 3.39 - Список користувачів

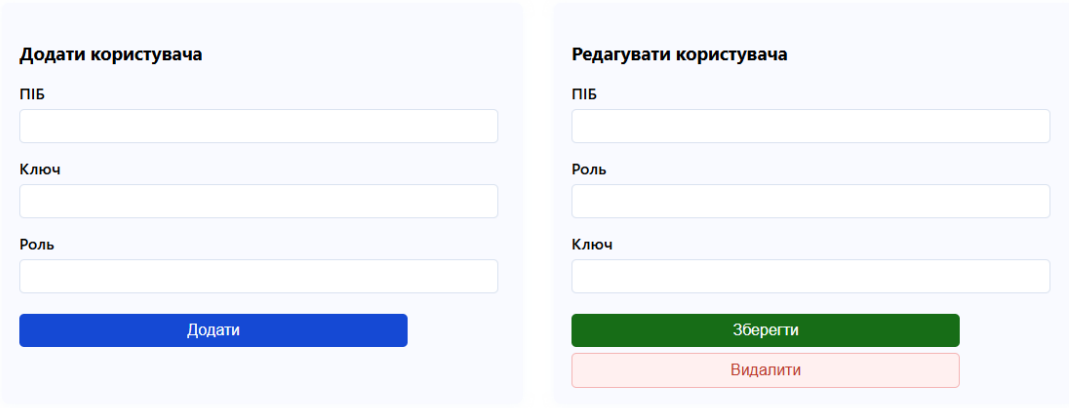


Рисунок 3.40 - Поля для редагування інформації

Сторінка “Журнал”

Опис:

Сторінка журналу відображає всі події доступу – хто, коли і з яким UID отримував доступ до території. Інформація представлена у вигляді таблиці, яка містить ПІБ, ключ користувача (UID), а також дату й час події. Є можливість експортувати журнал у форматі Excel для подальшого аналізу.

Як працює:

- У таблиці відображаються всі події доступу.
- Дані автоматично оновлюються при кожному відкритті сторінки.
- Для експорту даних потрібно натиснути кнопку “Завантажити звіт”.

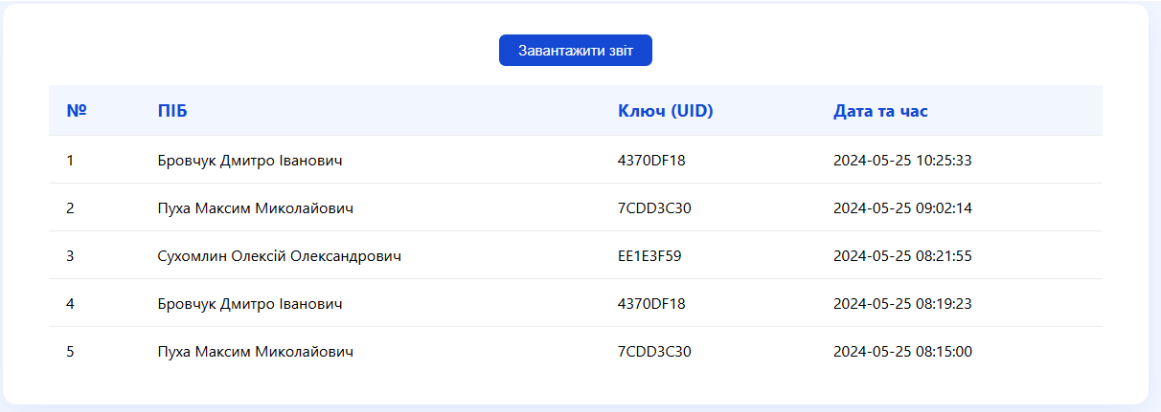


Рисунок 3.41 - Сторінка Журнал

	A	B	C	D
1	№	ПІБ	Ключ (UID)	Дата та час
2	1	Бровчук Дмитро Іванович	4370DF18	2024-05-25 10:25:33
3	2	Пуха Максим Миколайович	7CDD3C30	2024-05-25 9:02:14
4	3	Сухомлин Олексій Олександрович	EE1E3F59	2024-05-25 8:21:55
5	4	Бровчук Дмитро Іванович	4370DF18	2024-05-25 8:19:23
6	5	Пуха Максим Миколайович	7CDD3C30	2024-05-25 8:15:00

Рисунок 3.42 - Приклад сформованого звіту

Сторінка “Налаштування”

Опис:

На сторінці налаштувань міститься інформація про розробника системи, контакти для технічної підтримки, а також основні відомості про систему. Доступна лише для адміністратора, ця сторінка дозволяє швидко отримати інформацію для зв’язку у разі виникнення питань або проблем.

Рисунок 3.43 - Сторінка Налаштування

3.2. Перспективи розвитку та масштабування системи

Інформаційна система контролю доступу на приватну територію, побудована на базі ESP32, PHP та сучасної web-архітектури, є не лише гнучкою для поточних задач, а й має великий потенціал для подальшого розвитку і масштабування.

Можливості масштабування:

- **Додавання нових точок контролю:** Система дозволяє підключати додаткові контролери (наприклад, ще один в’їзд, виїзд або додаткові двері), які будуть працювати з тією ж базою даних.
- **Підтримка кількох об’єктів:** Серверна частина може бути доопрацьована для підтримки декількох територій або об’єктів (наприклад, кілька складів, офісних будівель тощо) з єдиною базою користувачів і журналом подій.
- **Інтеграція з іншими системами:** Система може бути інтегрована з відеоспостереженням, сигналізацією, системою пожежної безпеки або ERP/CRM-платформами підприємства.

Перспективи покращення функціоналу:

- Використання біометрії: Додавання біометричних модулів (відбиток пальця, розпізнавання обличчя) для підвищення рівня безпеки.
- Мобільний застосунок для адміністрування: Розробка мобільного додатку для віддаленого керування доступом, моніторингу подій та отримання сповіщень у реальному часі.
- Розширений журнал подій: Додавання можливості зберігати та візуалізувати статистику (кількість входів/виходів по користувачах, по часу, тощо).
- Автоматизовані сповіщення: Впровадження SMS/E-mail/Push-сповіщень у разі спроби несанкціонованого доступу або інших важливих подій.
- Гнучкі налаштування ролей: Впровадження ролей і прав доступу для різних категорій користувачів (адміністратор, охоронець, гість тощо).
- Хмарне зберігання даних: Перехід до хмарної бази даних для забезпечення надійності, доступу з будь-якої точки світу та резервного копіювання.

Можливості підвищення безпеки:

- Шифрування трафіку між контролером та сервером (наприклад, за допомогою HTTPS).
- Логування та сповіщення про підозрілі дії.
- Регулярне оновлення програмного забезпечення.

Технічне вдосконалення:

- Можливість переходу на потужніші мікроконтролери у разі збільшення навантаження.
- Застосування PoE (Power over Ethernet) для живлення та підключення контролерів.

ВИСНОВКИ

У ході виконання бакалаврської роботи було повністю реалізовано сучасну інформаційну систему для автоматизованого контролю доступу на приватну територію. В процесі роботи було проведено аналіз існуючих підходів до побудови систем доступу, здійснено вибір найоптимальніших апаратних та програмних компонентів, зокрема мікроконтролера ESP32, що забезпечує стабільний та безпечний зв’язок із сервером і базою даних.

На основі глибокого аналізу сучасних технологій було побудовано архітектуру системи, яка поєднує апаратну частину (RFID-зчитувач, кроковий двигун, ESP32) з інтуїтивно зрозумілим web-інтерфейсом, написаним із застосуванням PHP та MySQL. Особливу увагу приділено розробці зручної бази даних, що дозволяє зберігати та обробляти всю інформацію про користувачів, події доступу, журнал входів та виходів у режимі реального часу.

Програмне забезпечення було розроблено з акцентом на надійність, гнучкість та масштабованість, що дає змогу легко адаптувати систему під потреби конкретного об’єкту. В результаті розробки створено багатофункціональний web-інтерфейс, який дозволяє адміністратору здійснювати повний облік користувачів, контролювати історію подій, здійснювати додавання, редагування та видалення інформації безпосередньо з браузера, а також вивантажувати звіти для аналізу.

Тестування системи на практиці показало її стабільну роботу, зручність використання та високу швидкість обробки запитів. Розроблена система підвищує безпеку приватної території, дозволяє в автоматичному режимі вести детальний журнал усіх подій та мінімізує людський фактор у процесах контролю доступу.

Одержані результати мають універсальний характер і можуть бути впроваджені як на невеликих приватних об’єктах, так і на великих підприємствах, які потребують сучасного рівня захисту та обліку доступу. Впроваджена інформаційна система є

гнучкою до масштабування та модернізації, а її модульна архітектура дозволяє легко додавати нові функції у майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. [Електронний ресурс] Система контролю управління доступом: https://ssbb.ua/sistemy-kontrolya-dostupa/sistema-kontrolyu-dostupu/sistemy-kontrolya-dostupa-chto-eto-i-kak-eto-rabotaet/
2. [Електронний ресурс] Системи контролю доступу: https://ukrinfosystems.com.ua/uk/design-and-construction/access-control-systems
3. [Електронний ресурс] Світлодіод зелений / червоний: https://www.rcscomponents.kiev.ua/product/svitlodiod-5mm-chervonyi-zelenyi-bipoliarnyi-l-517lesgw_99051.html
4. [Електронний ресурс] RFID модуль PN532 NFC https://geekmatic.in.ua/pn532_nfc_rfid_module
5. [Електронний ресурс] Набір дротів: https://geekmatic.in.ua/wire_female_to_female_20cm
6. [Електронний ресурс] ArduinoIDE :https://www.arduino.cc/en/software
7. [Електронний ресурс] MySql :https://www.mysql.com/
8. [Електронний ресурс] Основи php : https://w3schoolsua.github.io/php/index.html#gsc.tab=0
9. [Електронний ресурс] ArduinoUno: https://geekmatic.in.ua/arduino_uno_r3
10. [Електронний ресурс] OrangePi: https://uamper.com/index.php?route=product/product&path=125&product_id=498&gad_source=1&gclid=Cj0KCQjw9vqyBhCKARIsAIIcLMF0n1vX-KDOGUMmGgaderoMSXQ-3IVhvoCs0uRF_k0sOXETwqMTDDEaAknZEALw_wcB
11. [Електронний ресурс] STM32: https://beegreen.com.ua/sistemna-plata-mikrokontroler-stm32f103c8t6-stm32-arm-17628
12. [Електронний ресурс] ESP32 : https://geekmatic.in.ua/esp32
13. [Електронний ресурс] кроковий двигун 28BYJ-48 з драйвером: https://geekmatic.in.ua/stepper_motor_28byj_48_with_drive

14. [Електронний ресурс] Автоматичне розпізнавання знаків : https://www.axis.com/solutions/vehicle-access-control
15. [Електронний ресурс] дослідження RFID: https://www.nedapidentification.com/insights/trends-and-developments-in-vehicle-access-control/
16. [Електронний ресурс] контроль доступу людей: https://alarm.lviv.ua/blog/shho-take-sistema-kontrolyu-dostupu-i-navishho-vona-potribna
17. [Електронний ресурс] Централізована архітектура :https://en.wikipedia.org/wiki/Access_control
18. [Електронний ресурс] Децентралізована архітектура : https://www.getkisi.com/guides/centralized-access-control
19. [Електронний ресурс] Хмарна архітектура:https://aws.amazon.com/blogs/big-data/build-a-centralized-granular-access-control-to-manage-assets-and-data-access-in-amazon-quicksight/
20. [Електронний ресурс] Гібридна архітектура:https://bezpeka.club/kontrol-dostupu-v-gibrydno-hmarnomu-seredovyshhi/
21. [Електронний ресурс] Система ANPR: https://bezpeka.club/anpr-klyuchovyj-element-rozumnogo-mista
22. [Електронний ресурс] Система електронних ключів: https://l-vorota.com.ua/ua/a422469-chto-takoe-sistemy.html

ДОДАТОК А.1 КОД МІКРОКОНТРОЛЕРА ESP32

Код програми мікроконтролера esp32:

```
#include <WiFi.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_PN532.h>
#include <AccelStepper.h>

// --- WiFi ---
const char* ssid = "TP-LINK_C948";
const char* password = "60945419";

// --- Сервер ---
const char* host = "192.168.0.103 // IP сервера
const uint16_t port = 80;           // HTTP

// --- PN532 SPI ---
#define PN532_SCK  (18)
#define PN532_MOSI (23)
#define PN532_SS   (5)
#define PN532_MISO (19)
Adafruit_PN532 nfc(PN532_SCK, PN532_MISO, PN532_MOSI, PN532_SS);

// --- Stepper (28BYJ-48 + ULN2003) ---
#define IN1 12
#define IN2 13
#define IN3 14
#define IN4 15
AccelStepper stepper(AccelStepper::HALF4WIRE, IN1, IN3, IN2, IN4);

// --- Кроки для 120 градусів --- (2048 кроків = 360°, тобто 120° = 682 кроки)
const int STEPS_120 = 2048 / 3;

void setup() {
  Serial.begin(115200);

  // --- Stepper ---
  stepper.setMaxSpeed(500);
  stepper.setAcceleration(200);
```

```
  // --- PN532 ---
  nfc.begin();
  if (!nfc.getFirmwareVersion()) {
    Serial.println("❌ PN532 not found!");
    while (1);
  }
  Serial.println("☑ PN532 detected.");
  nfc.SAMConfig();

  // --- WiFi ---
  WiFi.disconnect(true);
  delay(1000);
  WiFi.begin(ssid, password);
  Serial.print("Connecting WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(300);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected.");
}

void loop() {
  stepper.run(); // обов'язково викликати завжди

  if (stepper.distanceToGo() != 0) {
    // Двигун ще крутиться, не зчитуємо мітку
    delay(20);
    return;
  }

  Serial.println("Очікування RFID мітки...");
  uint8_t uid[7]; uint8_t uidLength;
  if (nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength)) {
    // UID в hex-рядок
    String rfidKey = "";
    for (uint8_t i = 0; i < uidLength; i++) {
      if (uid[i] < 0x10) rfidKey += "0";
      rfidKey += String(uid[i], HEX);
    }
    rfidKey.toUpperCase();

    Serial.print("UID: "); Serial.println(rfidKey);

    // --- Перевіряємо на сервері ---
```

```
    if (checkUIDwithServer(rfidKey)) {
      Serial.println("Ключ валідний, обертаємо!");
      // 120 градусів вперед
      stepper.move(STEPS_120);
      while (stepper.distanceToGo() != 0) stepper.run();
      delay(3000);
      // Повертаємось назад
      stepper.move(-STEPS_120);
      while (stepper.distanceToGo() != 0) stepper.run();
    } else {
      Serial.println("Ключ не знайдено.");
    }

    // --- Очікуємо 5 сек до повторного зчитування ---
    delay(5000);
  }
}

bool checkUIDwithServer(String rfidKey) {
  WiFiClient client;
  String url = "/verify_rfid.php?rfid=" + rfidKey; // PHP-роутер
  Serial.print("Запит до: "); Serial.println(url);

  if (!client.connect(host, port)) {
    Serial.println("Помилка підключення до сервера");
    return false;
  }

  // --- HTTP GET ---
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");

  unsigned long timeout = millis();
  String response = "";
  while (client.connected() && millis() - timeout < 3000) {
    while (client.available()) {
      char c = client.read();
      response += c;
    }
  }
  client.stop();

  // --- Знаходимо JSON у відповіді ---
```

```
  int jsonStart = response.indexOf('{');
  int jsonEnd = response.lastIndexOf('}');
  if (jsonStart == -1 || jsonEnd == -1) {
    Serial.println("JSON not found in response.");
    return false;
  }
  String json = response.substring(jsonStart, jsonEnd + 1);
  Serial.print("Сервер відповів: "); Serial.println(json);

  // --- Простий аналіз без ArduinoJson ---
  if (json.indexOf("\"status\":\"success\"") != -1) {
    return true;
  }
  return false;
}
```

ДОДАТОК Б.1

Код users.php:

```
<?php
session_start();
require_once 'db.php';

// Перевірка авторизації
if (!isset($_SESSION['user_id'])) {
    header('Location: index.php');
    exit();
}

// Отримуємо всіх користувачів
$query = "SELECT id, full_name, role, tag_uid FROM person_info";
$result = $conn->query($query);
$users = [];
while ($row = $result->fetch_assoc()) {
    $users[] = $row;
}
?>
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Список користувачів</title>
    <style>
        body { background: #f2f6ff; font-family: "Segoe UI", sans-
serif; margin: 0; padding: 0; }
        .navbar {
            width: 100%; background: #fff; box-shadow: 0 4px 16px
rgba(21, 73, 212, 0.04);
            display: flex; align-items: center; justify-content:
center; height: 70px;
            position: fixed; top: 0; left: 0; z-index: 999;
        }
        .navbar a, .navbar-title {
            color: #1549d4; text-decoration: none; font-size: 22px;
padding: 8px 30px;
            border-radius: 8px; font-weight: 500; margin: 0 10px;
            border-bottom: 3px solid transparent; transition: border-
bottom 0.2s, font-weight 0.2s;
        }
        .navbar-title { font-size: 28px; font-weight: bold; cursor:
default; margin: 0 30px; }
        .navbar a.active { font-weight: bold; border-bottom: 3px solid
#1549d4; text-decoration: underline; }
```

```
        .logout { position: absolute; right: 40px; color: #1549d4;
text-decoration: none; font-size: 18px; top: 50%; transform:
translateY(-50%); }
        .logout:hover { text-decoration: underline; }
        .main-content { padding-top: 100px; }
        .container {
            max-width: 1100px; margin: 0 auto; background: #fff;
border-radius: 14px;
            box-shadow: 0 8px 32px rgba(21, 73, 212, 0.07); padding:
32px 48px;
        }
        table { width: 100%; border-collapse: collapse; margin-top:
18px; }
        th, td { padding: 14px 18px; border-bottom: 1px solid #ececec;
text-align: left; }
        th { background: #f2f6ff; color: #1549d4; font-size: 18px; }
        tr:last-child td { border-bottom: none; }
        .forms-row { display: flex; justify-content: space-between;
gap: 32px; margin-top: 35px;}
        .user-form { flex: 1; background: #f9faff; border-radius: 8px;
box-shadow: 0 2px 10px #e5eefd40;
            padding: 20px 18px 16px 18px; min-width: 320px;
        }
        .user-form label { display: block; margin-bottom: 6px; font-
weight: 500; }
        .user-form input, .user-form select {
            width: 95%; padding: 8px; margin-bottom: 15px; border: 1px
solid #dbe3f1; border-radius: 5px; font-size: 15px;
        }
        .user-form button { width: 80%; padding: 8px 0; border: none;
border-radius: 5px; font-size: 16px; font-weight: 500; cursor:
pointer; margin-top: 6px; }
        .user-form .add-btn { background: #1549d4; color: #fff; }
        .user-form .edit-btn { background: #176d17; color: #fff; }
        .user-form .delete-btn { background: #fff0f0; color: #c0392b;
border: 1px solid #f5b5b5; }
        .user-msg { margin-bottom: 18px; padding: 8px 0; font-size:
16px; text-align: center; border-radius: 6px; }
        .user-msg.success { color: #0d7513; background: #ecffe6; }
        .user-msg.error { color: #b30000; background: #ffeaea; }
        .user-select-row { cursor: pointer; transition: background
0.12s; }
        .user-select-row:hover { background: #e6edff; }
    </style>
</head>
<body>
    <div class="navbar">
        <a href="journal.php" class="<?=
(basename($_SERVER['PHP_SELF'])=='journal.php')?'active':''
?>">Журнал</a>
```

```
        <span class="navbar-title <?=
(basename($_SERVER['PHP_SELF'])=='users.php')?'active':'' ?>">Список
користувачів</span>
        <a href="settings.php" class="<?=
(basename($_SERVER['PHP_SELF'])=='settings.php')?'active':''
?>">Налаштування</a>
        <a href="logout.php" class="logout">Вийти</a>
    </div>
    <div class="main-content">
        <div class="container">
            <div id="msgBox"></div>
            <table>
                <thead>
                    <tr>
                        <th>ПІБ</th>
                        <th>Роль</th>
                        <th>Ключ</th>
                    </tr>
                </thead>
                <tbody id="userTable">
                    <?php foreach ($users as $user): ?>
                    <tr class="user-select-row"
                        data-id="<?= $user['id'] ?>"
                        data-full_name="<?=
htmlspecialchars($user['full_name']) ?>"
                        data-role="<?= htmlspecialchars($user['role'])
?>"
                        data-tag_uid="<?=
htmlspecialchars($user['tag_uid']) ?>">
                        <td><?= htmlspecialchars($user['full_name'])
?></td>
                        <td><?= htmlspecialchars($user['role'])
?></td>
                        <td><?= htmlspecialchars($user['tag_uid'])
?></td>
                    </tr>
                    <?php endforeach; ?>
                </tbody>
            </table>
            <div class="forms-row">
                <!-- Додати користувача -->
                <form class="user-form" id="addForm" method="POST"
action="add_user.php">
                    <h3>Додати користувача</h3>
                    <label>ПІБ</label>
                    <input type="text" name="full_name" required>
                    <label>Ключ</label>
                    <input type="text" name="tag_uid" required>
                    <label>Роль</label>
                    <input type="text" name="role" required>
                    <button type="submit" class="add-
btn">Додати</button>
```

```
                </form>
                <!-- Редагувати користувача -->
                <form class="user-form" id="editForm" method="POST"
action="edit_user.php">
                    <h3>Редагувати користувача</h3>
                    <input type="hidden" name="id" id="edit_id">
                    <label>ПІБ</label>
                    <input type="text" name="full_name"
id="edit_full_name" required>
                    <label>Роль</label>
                    <input type="text" name="role" id="edit_role"
required>
                    <label>Ключ</label>
                    <input type="text" name="tag_uid"
id="edit_tag_uid" required>
                    <button type="submit" class="edit-
btn">Зберегти</button>
                    <button type="button" class="delete-btn"
id="deleteBtn">Видалити</button>
                </form>
            </div>
        </div>
    </div>
<script>
function showMsg(msg, status='success') {
    let box = document.getElementById('msgBox');
    box.innerHTML = `<div class="user-msg ${status}">${msg}</div>`;
    setTimeout(()=>{box.innerHTML=''}, 2200);
}
// Автозаповнення форми редагування по кліку на користувача
document.querySelectorAll('.user-select-row').forEach(row => {
    row.onclick = function() {
        document.getElementById('edit_id').value = this.dataset.id;
        document.getElementById('edit_full_name').value =
this.dataset.full_name;
        document.getElementById('edit_role').value =
this.dataset.role;
        document.getElementById('edit_tag_uid').value =
this.dataset.tag_uid;
    };
});
// AJAX-додавання користувача
document.getElementById('addForm').onsubmit = async function(e) {
    e.preventDefault();
    const fd = new FormData(this);
    let r = await fetch('add_user.php', { method:'POST', body:fd });
    let res = await r.json();
    showMsg(res.msg, res.status);
    if (res.status === 'success') setTimeout(()=>location.reload(),
1200);
};
// AJAX-редагування користувача
```

```
document.getElementById('editForm').onsubmit = async function(e) {
    e.preventDefault();
    const fd = new FormData(this);
    let r = await fetch('edit_user.php', { method:'POST', body:fd });
    let res = await r.json();
    showMsg(res.msg, res.status);
    if (res.status === 'success') setTimeout(()=>location.reload(),
1200);
};
// AJAX-видалення користувача
document.getElementById('deleteBtn').onclick = async function() {
    const id = document.getElementById('edit_id').value;
    if (!id) return showMsg('Оберіть користувача!', 'error');
    if (!confirm('Видалити користувача?')) return;
    const fd = new FormData();
    fd.append('id', id);
    let r = await fetch('delete_user.php', { method:'POST', body:fd
});
    let res = await r.json();
    showMsg(res.msg, res.status);
    if (res.status === 'success') setTimeout(()=>location.reload(),
1200);
};
</script>
</body>
</html>
```

ДОДАТОК Б.2

Код index.php:

```
<?php
session_start();
if (isset($_SESSION['user_id'])) {
    header('Location: users.php');
    exit();
}
?>

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Авторизація</title>
  <style>
    body {
      margin: 0;
      font-family: "Segoe UI", sans-serif;
      background-color: #f2f6ff;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }
    .login-container {
      background: white;
      padding: 40px 30px;
      border-radius: 12px;
      box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
      width: 300px;
      text-align: center;
    }
    .login-container h2 {
      margin-bottom: 30px;
      color: #1549d4;
      font-size: 24px;
    }
    .login-container input[type="text"],
    .login-container input[type="password"] {
      width: 100%;
      padding: 10px;
      margin-bottom: 15px;
      border: 1px solid #ccc;
```

```
      border-radius: 6px;
      font-size: 14px;
    }
    .login-container button {
      width: 100%;
      padding: 10px;
      background-color: #1549d4;
      color: white;
      border: none;
      border-radius: 6px;
      font-size: 16px;
      cursor: pointer;
    }
    .login-container button:hover {
      background-color: #123fc4;
    }
    .error-message {
      color: red;
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <div class="login-container">
    <h2>Авторизація</h2>
    <form method="POST" action="login.php">
      <input type="text" name="full_name" placeholder="ПІБ" required>
      <input type="password" name="password" placeholder="Пароль"
required>
      <button type="submit">Увійти</button>
    </form>
    <?php if (isset($_SESSION['login_error'])): ?>
      <div class="error-message"><?= $_SESSION['login_error'];
unset($_SESSION['login_error']); ?></div>
    <?php endif; ?>
  </div>
</body>
</html>
```

ДОДАТОК Б.3

Код login.php:

```
<?php

session_start();

require_once 'db.php'; // Файл підключення до БД

if (!isset($_POST['full_name'], $_POST['password'])) {

    $_SESSION['login_error'] = "Введіть ПІБ та пароль!";

    header('Location: index.php');

    exit();

}

$full_name = trim($_POST['full_name']);

$password = trim($_POST['password']);

$stmt = $conn->prepare("SELECT * FROM person_info WHERE full_name = ? AND
password = ?");

$stmt->bind_param("ss", $full_name, $password);

$stmt->execute();

$result = $stmt->get_result();

if ($user = $result->fetch_assoc()) {

    // Авторизація успішна

    $_SESSION['user_id'] = $user['id'];

    $_SESSION['full_name'] = $user['full_name'];

    $_SESSION['role'] = $user['role'];

    header('Location: users.php');

    exit();

} else {

    // Помилка авторизації

    $_SESSION['login_error'] = "Невірний ПІБ або пароль!";

    header('Location: index.php');
```

```
    exit();
}
```