Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка системи автоматизації для контролю доступу та обліку робочого часу співробітників на основі QR-кодів
(тема)

Виконав:
студент 4 курсу, групи АКТАКІТ-20-1
Стасенко Я. О.
(прізвище, ініціали)

Спеціальність 151 – автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва освітньої програми)

Керівник проф. Сезонова І. К.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024р.

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
Кафедра КІТАР
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва)
Тип програми освітньо-професійна
Освітня програма Автоматизація та комп'ютерно-інтегровані технології
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри_____
(підпис)
«____»_____20___р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Стасенку Ярославу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи автоматизації для контролю доступу та обліку робочого часу співробітників на основі QR-кодів

затверджена наказом університету від 03.06 2024 р. № 544 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 20 06 2024 р.
3. Вихідні дані до роботи Середовище програмування Rider, .NET
мова програмування: C#, база даних MSSQL, Blazor, Web API, WPF

4. Перелік питань, що потрібно опрацювати в роботі
Вступ;
Аналіз предметної області;
Розробка архітектури та вибір технологій для розроблюваної системи;
Опис практичної реалізації розроблюваної системи;
Технічно-економічні розрахунки;
Охорона праці;
Висновки;
Додатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускної кафедри) Демонстрацію та матеріал представлений у форматі презентації PowerPoint (*.ppt) = 16 с. формату А4

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Видача теми проекту, узгодження та затвердження	01.05.2024	виконано
2	Аналіз вимог технічного завдання	06.05.2024	виконано
3	Аналіз і опис систем контролю доступу, обліку робочого часу та опис QR-кодів	08.05.2024	виконано
4	Вибір компонентів для програмної частини проекту	10.05.2024	виконано
5	Вибір компонентів для апаратної частини проекту	11.05.2024	виконано
6	Розробка серверної частини системи	15.05.2024	виконано
7	Розробка структури бази даних	20.05.2024	виконано
8	Розробка веб-інтерфейсу	22.05.2024	виконано
9	Розробка алгоритму роботи для симуляції роботи апаратної частини системи	25.05.2024	виконано
10	Розробка десктоп-застосунку для симуляції роботи апаратних компонентів системи	1.06.2024	виконано
11	Розрахунок техніко-економічних показників системи	5.06.2024	виконано
12	Оформлення пояснювальної записки	10.06.2024	виконано
13	Перевірка виконаного проекту керівником	14.06.2024	виконано
14	Захист проекту	25.06.2024	

Дата видачі завдання 01 квітня 2024 р.

Студент_____
(підпис)

Керівник роботи_____ проф. Сезонова І. К.
(підпис) (посада, прізвище, ініціали)

Я, Стасенко Ярослав Олександрович, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на віповідне джерело.

14.06.2024 р.  Стасенко Я. О.

(підпис) (ПІБ)

РЕФЕРАТ

Пояснювальна записка містить 84 с., 2 табл., 40 рис., 5 дод., 20 джерел.

СИСТЕМА КОНТРОЛЮ ДОСТУПУ, КОНТРОЛЬ ДОСТУПУ, ОБЛІК РОБОЧОГО ЧАСУ, QR-КОД, ЗАПИТИ, ВЕБ-ІНТЕРФЕЙС, СИМУЛЯЦІЯ, C#, MSSQL.

Метою кваліфікаційної роботи є підвищення безпеки контролю доступу співробітників, а також точного обліку їх робочого часу шляхом впровадження автоматизованої системи на основі QR-кодів. Система повинна забезпечувати зручний інтерфейс для взаємодії користувачів та адміністраторів, спрощувати процес контролю доступу та обліку робочого часу, а також зменшувати ймовірність помилок та зловживань.

Об'єктом розробки є процес контролю доступу та обліку робочого часу співробітників на основі QR-кодів системи автоматизації.

Предметом розробки є програмні та апаратні засоби, алгоритми, що забезпечують функціональність системи контролю доступу та обліку робочого часу на основі QR-кодів.

Проведено аналіз предметної області, де було визначено існуючі системи контролю доступу та обліку робочого часу, основні засоби реалізації типових систем та описано QR-коди та їх використання. Було обрано програмні та апаратні засоби для розробки системи автоматизації.

Розроблено систему, яка включає сервер, що складається з API, який, у свою чергу, виконує основну логіку системи, взаємодіючі з базою даних, веб-інтерфейс для зручної та ефективної взаємодії користувачів із системою, та створено десктоп-додаток для симуляції роботи апаратних компонентів системи.

ABSTRACT

The explanatory note contains 84 p., 2 tabl., 40 fig., 5 adj, 20 sources.

ACCESS CONTROL SYSTEM, ACCESS CONTROL, WORK TIME ACCOUNTING, QR CODE, REQUESTS, WEB INTERFACE, SIMULATION, C#, MSSQL.

The aim of the qualification work is to improve the security of employee access control, as well as the accurate tracking of their working hours by implementing an automated system based on QR codes. The system should provide a convenient interface for user and administrator interaction, simplify the process of access control and working time tracking, and reduce the likelihood of errors and misuse.

The object of development is the process of access control and working time tracking of employees based on the QR code automation system.

The subject of development is the software and hardware tools, algorithms that ensure the functionality of the QR code-based access control and working time tracking system.

An analysis of the subject area was conducted, where existing access control and working time tracking systems, the main tools for implementing typical systems, and the use of QR codes were described. The software and hardware tools for developing the automation system were selected.

A system was developed that includes a server consisting of an API, which performs the main logic of the system by interacting with the database, a web interface for convenient and efficient user interaction with the system, and a desktop application to simulate the operation of the system's hardware components.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних;

ООП – об'єктно-орієнтоване програмування;

РСУБД – реляційна система управління базами даних;

СКД – система контролю доступу;

СКУД – система контролю та управління доступом;

API – application programming interface (інтерфейс програмування додатків);

CORS – cross-origin resource sharing (крос-доменний обмін ресурсами);

CRM – customer relationship management (управління взаємовідносинами з клієнтами);

DI – dependency injection (впровадження залежностей);

DTO – data transfer object (об'єкт передачі даних);

ERP – enterprise resource planning (планування ресурсів підприємства);

HRM – human resource management (управління людськими ресурсами);

HTTPS – hypertext transfer protocol secure (захищений протокол передачі гіпертексту);

IP – internet protocol (інтернет-протокол);

JSON – javascript object notation (нотація об'єктів javascript);

JWT – JSON web token (веб-токен JSON);

NFC – near field communication (близьке безконтактне спілкування);

PIN – personal identification number (персональний ідентифікаційний номер);

QR – quick response (швидка відповідь);

RFID – radio frequency identification (радіочастотна ідентифікація);

SQL – structured query language (мова структурованих запитів);

SSL – secure sockets layer (рівень захищених сокетів);

TCP – transmission control protocol (протокол керування передачею);

TSL – transport layer security (захист транспортного рівня);

URL – uniform resource locator (уніфікований покажчик ресурсу);

USB – universal serial bus (універсальна послідовна шина);

WASM – webassembly (веб-збірка);

WPF – windows presentation foundation (фундамент презентації Windows);

XAML – extensible application markup language (розширювана мова розмітки додатків).

ВСТУП

У сучасному світі інформаційні технології швидко розвиваються, що створює нові можливості для автоматизації бізнес-процесів і підвищення ефективності підприємств. Особливо актуальним стає питання автоматизації контролю доступу та обліку робочого часу співробітників, оскільки це дозволяє оптимізувати управлінські процеси, підвищити безпеку та продуктивність праці. Серед технологій, що можуть бути використані для цих цілей, виділяються QR-коди, які завдяки своїй простоті, зручності та надійності знаходять широке застосування в різних сферах.

Актуальність дослідження обумовлена зростаючими вимогами до безпеки та ефективності управління персоналом, особливо в умовах нинішньої військової ситуації в Україні. На сьогодні підприємства стикаються з додатковими викликами щодо забезпечення безпеки своїх співробітників і збереження контролю над внутрішніми процесами. Традиційні методи контролю доступу, такі як магнітні картки або паролі, мають свої обмеження, пов’язані з можливістю їх втрати, крадіжки або забування. QR-коди, навпаки, забезпечують високий рівень захисту інформації, оскільки їх можна швидко зчитати за допомогою спеціальних пристроїв або звичайних смартфонів, що робить їх зручними для використання в будь-яких умовах.

Тема роботи сприяє досягненню цілі сталого розвитку 9.4, яка передбачає прискорене зростання високотехнологічних секторів переробної промисловості, зокрема через застосування інформаційно-телекомунікаційних технологій у цій галузі [1].

Метою кваліфікаційної роботи є підвищення безпеки контролю доступу співробітників, а також точного обліку їх робочого часу шляхом впровадження автоматизованої системи на основі QR-кодів. Система повинна забезпечувати зручний інтерфейс для взаємодії користувачів та адміністраторів, спрощувати

процес контролю доступу та обліку робочого часу, а також зменшувати ймовірність помилок та зловживань.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– провести аналіз існуючих систем контролю доступу та обліку робочого часу, визначити їх переваги та недоліки;

– розробити архітектуру системи та обґрунтувати вибір технологій для її реалізації;

– створити API для управління працівниками, запрошеннями та QR-кодами (генерація, перевірка, створення зображення);

– розробити веб-інтерфейс для забезпечення взаємодії користувачів із системою;

– спроектувати та реалізувати апаратну частину системи, яка включатиме турнікет, QR-сканер та інфрачервоні датчики відстані;

– провести тестування розробленої системи та оцінити її ефективність;

– оформити кваліфікаційну роботу згідно ДСТУ 3008:2015 [2], а також за методичними вказівками з підготовки кваліфікаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» освітньої програми «Автоматизація та комп’ютерно-інтегровані технології» [3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз систем контролю доступу

Сучасні системи контролю та управління доступом (СКУД) являють собою комплекс електронних, механічних, електротехнічних, апаратно-програмних та інших засобів, які дозволяють визначеному персоналу отримати доступ до певних зон або обладнання, обмежуючи доступ тим, хто не має відповідних прав. Це критично важливо для забезпечення інформаційної безпеки об'єктів. СКУД також можуть відстежувати переміщення працівників і транспорту на охоронюваній території, гарантувати безпеку персоналу та відвідувачів, а також зберігання матеріальних і інформаційних ресурсів підприємства. Крім того, такі системи автоматизують моніторинг виконання завдань, пов'язаних із безпекою та контролем доступу [4].

Система контролю доступу вирішує такі задачі:

– захист від несанкціонованого доступу;

– контроль доступу до серверів і баз даних;

– ведення журналу подій для моніторингу та реагування на загрози;

– фіксація робочого часу співробітників;

– інтеграція з іншими системами безпеки;

– автоматичне керування зонами в надзвичайних ситуаціях;

– відповідність законодавчим вимогам щодо безпеки [5].

За контролем доступу розрізняють автономні, мережеві та універсальні системи контролю доступу.

Автономні системи контролю доступу є самодостатніми та незалежними від централізованого керування або підключення до мережі. Вони призначені для забезпечення безпеки на невеликих об'єктах, таких як окремі приміщення, магазини чи офіси, де немає потреби у складних та дорогих комплексних системах безпеки.

Автономні системи контролю доступу вбудовані в двері чи прохідні та містять засоби ідентифікації, такі як зчитувачі карток, клавіатури для PIN-кодів або біометричні датчики. Вони не потребують підключення до мережі чи серверів, оскільки дані зберігаються в пристрої. Це спрощує встановлення та експлуатацію, оскільки немає потреби в мережевих кабелях чи складних серверах.

Хоча автономні системи не забезпечують високого рівня безпеки та функціональності, як централізовані, вони є економічно ефективним рішенням для малих об'єктів з обмеженими потребами в контролі доступу. Вони забезпечують базовий захист, дозволяючи керувати списками користувачів та контролювати доступ через певні точки.

Прикладом частини автономної системи контролю доступу може бути біометрична панель контролю доступу ZKTeco X8s (рис. 1.1). Для роботи біометричного терміналу необхідно додатково обладнати двері: електрозамком, кнопкою виходу та системою електроживлення DC12B. Цей контролер зберігає в пам'яті не персоналізовану базу шаблонів відбитків користувачів без можливості вилучення статистики проходів та інших даних. Панель має сенсорну панель-клавіатуру для взаємодії з системою [6].

Рисунок 1.1 – Автономна біометрична панель контролю доступу ZKTeco X8s [6]

Мережева система контролю доступу є комплексним рішенням, де всі контролери доступу на різних точках об'єкта об'єднані в єдину комп'ютерну мережу і керуються централізовано. На відміну від автономних систем, у мережевій СКУД контролери постійно обмінюються даними з центральним сервером або робочими станціями.

Управління всією системою здійснюється через спеціалізоване програмне забезпечення на одному або декількох персональних комп'ютерах. Адміністратори мають повний контроль, можливість налаштовувати права доступу для користувачів, переглядати журнали подій у реальному часі тощо. У системі зберігається централізована база даних усіх зареєстрованих користувачів, їхніх облікових записів та правил доступу, що спрощує керування великою кількістю співробітників.

Мережеві СКУД є масштабованими і можуть охоплювати сотні окремих контролерів на дверях, турнікетах, шлагбаумах і забезпечувати контроль доступу для десятків тисяч користувачів. Така архітектура ідеально підходить для великих об'єктів, таких як заводи, бізнес-центри, аеропорти тощо. Систему легко розширювати за потреби, додаючи нові контролери.

Мережеві системи контролю доступу забезпечують високий рівень безпеки, гнучкість і зручність централізованого управління на великих об'єктах з багатьма точками проходу. Єдина база даних, оперативний моніторинг подій та інтеграція з іншими системами дозволяють комплексно вирішувати питання безпеки, своєчасно реагувати на інциденти та оптимізувати процеси за допомогою аналітики.

Прикладом мережевої системи може послугувати комплект (рис 1.2), який складається з мережевого контролеру KTeco C3-100, який дозволяє організувати облік робочого часу співробітників, а також два універсальних зчитувача ZKTeco KR310 Дані зчитувачі працюють з безконтактними картами форматів Em-marine і Mifare. Це дозволить безболісно модернізувати існуючий об'єкт, на якому використовуються різні типи ідентифікаторів. Зчитувачі підключаються до контролера по протоколу Wiegand [7].

Рисунок 1.2 – Мережевий комплект СКД [7]

Системи контролю та управління доступом складаються з декількох ключових компонентів, кожен із яких відіграє важливу роль у забезпеченні ефективності та безпеки.

Одною з головних частин системи контролю доступу є контролери. Контролери – це інтелектуальні пристрої, що керують доступом на основі отриманих даних від різних датчиків, зчитувачів і системи загалом. Вони слугують для прийняття рішення щодо надання чи обмеження доступу, можуть зберігати журнали подій, ідентифікаторів користувачів та налаштування безпеки, а також взаємодіють з іншими системами через мережеві протоколи або централізований сервер. В залежності від потреби контролер може бути мережевий або автономний.

Для того, щоб контролери мали підставу для виконання певних рішень, до них приєднують певні зчитувачі, що сканують дані ідентифікаторів.

До ідентифікаторів належать:

– картки доступу. До карток доступу можна віднести магнітні картки, смарт картки, безконтактні картки (RFID (Radio Frequency Identification, радіочастотна ідентифікація)), та інші види різноманітних карток-ідентифікаторів;

– біометричні ідентифікатори. Сюди відносяться відбитки пальців, розпізнавання обличчя, розпізнавання оболонки ока, розпізнавання голосу, аналіз геометрії долоні або рук тощо;

– коди або паролі;

– мобільні ідентифікатори. Зазвичай використовують різні мобільні додатки для ідентифікації у відповідній системи завдяки різним технологіям таким, як Bluetooth або NFC. А також на сьогоднішній день є дуже популярними додатки, що генерують короткочасні коди.

Відповідно до кожного типу ідентифікатора є свій зчитувач, який може зчитувати інформацію та передавати її до контролера.

Для безпеки та правильної роботи СКД в системи додають замки та інші виконавчі прилади. Сюди можна віднести різного роду замки, які встановлюються відповідно до рівня безпеки, електроприводи, що слугують для автоматичного відкривання та закривання дверей, турнікетів, шлагбаумів, відповідно самі турнікети, різні ворота та шлагбауми. Сюди можна також віднести різні типи датчиків, наприклад, датчики відчинення дверей, датчики руху або присутності [8].

Відповідно до системи встановлюються центральний сервер та програмне забезпечення для зручного централізованого управління і контроля усіма компонентами.

Прикладом простої мережевої системи контролю доступу може стати схема, що наведена на рисунку 1.3. До цієї схеми входять:

– пристрої блокування (замки, шлагбауми, турнікети);

– контролер – мозок системи;

– зчитувач;

– ідентифікатори (картки, біометрика);

– робоче місце адміністратора СКД.

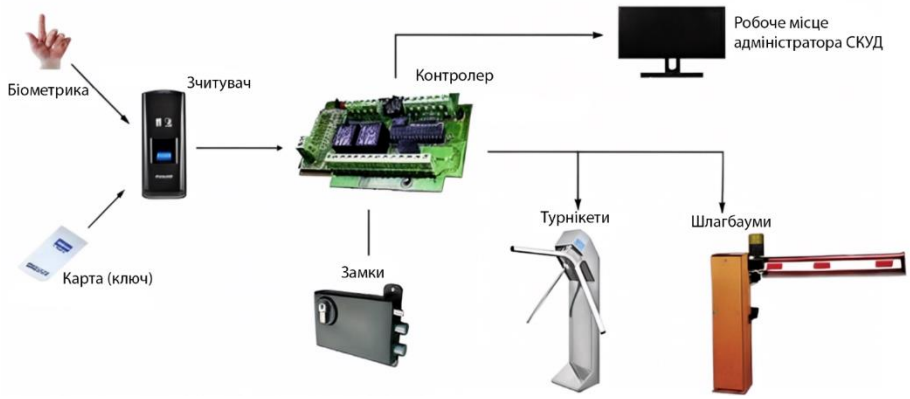

Рисунок 1.3 – Приклад простої мережевої СКД [9]

Робота цієї система полягає в наступному: зчитувач отримує інформацію через карту, брелок або зчитує біометрику, передає дані на контролер, а той, у свою чергу, ідентифікує користувача, передає інформацію до серверу і «віддає команду» виконавчому пристрою пропустити людину або транспортний засіб на об'єкт [9].

1.2 Аналіз систем обліку робочого часу

Система обліку робочого часу – це автоматизована система, яка використовується для контролю та реєстрації часу, проведеного працівниками на роботі. Вона допомагає організаціям відстежувати присутність співробітників, оптимізувати розклад, а також покращити дисципліну та продуктивність.

На сьогоднішній день в Україні існує три види обліку робочого часу:

– щоденний облік. Він застосовується в тому разі, коли працівник має однакову тривалість добової роботи;

– щотижневий облік використовується, коли тривалість щоденної роботи може бути неоднаковою, але за тиждень працівник реалізує встановлену норму робочого часу – 24, 36 год. тощо, але не більше 40 год;

– підсумований облік робочого часу. Цей облік допускається на безперервно діючих підприємствах, а також при виконанні окремих видів робіт, де за умовами виробництва не може додержуватися встановлена для певної

категорії працівників стабільна щоденна або щотижнева тривалість робочого часу.

Підсумований облік робочого часу застосовується також при використанні режиму гнучкого робочого часу, обов’язковою умовою впровадження якого є забезпечення точного обліку відпрацьованого часу. Практична реалізація цієї вимоги в Україні в цілому та в окремих видах економічної діяльності зокрема спричиняє певні труднощі [10].

Системи обліку робочого часу існують давно. Першою такою системою є паперовий журнал обліку та контролю. До більш сучасних належать:

– електронний табель обліку робочого часу;

– відеоспостереження та відеофіксація;

– автоматичні контролери, які відстежують час прибуття на роботу співробітників та відходу з роботи за допомогою їх особистих карт або брелоків;

– біометричні термінали – пристрої, які використовують для ідентифікації співробітників унікальні фізіологічні особливості, такі як райдужка очей, відбиток пальця, будова обличчя, геометрія долоні та малюнок вен. Також для ідентифікації можна використовувати почерк і голос. Крім оптичного зчитувача, термінал має дисплей, клавіші та вбудовану пам'ять для ведення журналу подій;

– тайм-трекери — спеціальні програми для відстеження часу, який витрачають співробітники на виконання кожного з поставлених перед ними завдань.

CRM-системи (Customer Relationship Management) – це операційні або аналітичні програми, що автоматизують взаємодію з клієнтами. CRM-системи зберігають дані про клієнтів, історію відносин, відстежують та аналізують результати, дозволяючи більш поінформовано, а отже, і якісно керувати обслуговуванням та маркетингом. CRM-системи оптимізують бізнес-процеси та використовуються для підвищення лояльності клієнтів до компанії [11].

Так як існує багато сучасних систем для обліку робочого часу, що відрізняються своїми функціями, застосування та впливом на робоче

середовище, необхідно визначити всі плюси та недоліки найпопулярніших систем.

Біометричні системи використовують унікальні фізіологічні характеристики для ідентифікації особи. Ці системи включають сканери відбитків пальців, обличчя, сітківки ока, що дозволяє з високою точністю ідентифікувати співробітника при вході та виході з роботи. До плюсів такої системи можна віднести точність ідентифікації, що унеможливлює підроблення або передачу даних для ідентифікації іншому співробітнику, зменшення шахрайства, а також ефективне управління даними робітників.

До мінусів такої системи можна віднести високу вартість, що включає в себе встановлення та обслуговування біометричних даних, приватність біометричних даних співробітників, а також чутливість до умов.

Прикладом біометричної системи для обліку робочого часу може стати біометрична система ZKTeco S922 (рис. 1.4) [12].

Рисунок 1.4 – Біометрична система обліку робочого часу ZKTeco S922

Дані з віддаленого об'єкта (будмайданчик, поле, ферма, захід, тимчасова локація. кар'єру) передаватимуться в режимі реального часу безпосередньо на сервер (центральний офіс) вашої компанії через вбудований модуль зв'язку GPRS/3G. Процес фіксації відвідувань відбувається після того, як працівник

підприємства сканує на пристрої свій відбиток пальця або за допомогою використання безконтактних карток. Далі термінал фіксує інформацію у внутрішній пам'яті та передає інформацію на сервер ПЗ BioTime 8 або WDMS-сервер ПЗ ZKTimeNET3.0 [12].

Дуже популярною технологією у системах обліку робочого часу є RFID-технологія. RFID-технології використовують радіочастотні ідентифікаційні мітки для автоматичного відстеження присутності співробітників. Співробітники носять спеціальні картки або брелоки, які взаємодіють з читачами, розміщеними на входах і виходах.

До плюсів такої системи можна віднести:

– швидкість і зручність використання;

– гнучкість розгортання;

– низька вартість обладнання.

До недоліків такої системи відносяться:

– ризик безпеки – картки можуть бути втрачені або викрадені;

– обмежена інформація про робітника;

– вразливість до порушень.

Мобільні додатки для обліку робочого часу забезпечують доступність і гнучкість, дозволяючи співробітникам відмічати час роботи через смартфони. Це знижує потребу в спеціальному обладнанні та ідеально підходить для віддаленої роботи. Однак, вони потребують смартфону у кожного співробітника, викликають занепокоєння щодо приватності та залежать від інтернету і електроживлення.

Веб-базовані системи дозволяють доступ з будь-якого пристрою з інтернетом і легко інтегруються з іншими HR-системами. Вони добре масштабуються, але залежать від мережі, вимагають захисту даних і комплексного управління.

Традиційні системи обліку використовують фізичні картки для відмітки часу. Вони прості у використанні, але мають обмежену функціональність, можуть бути об'єктом шахрайства і потребують фізичного обслуговування.

Сучасні системи для обліку робочого часу об'єднують в собі як апаратне, так і програмне забезпечення для ефективного управління та моніторингу робочих годин співробітників. Ці системи використовують різні технології ідентифікації, включаючи біометричні сканери та RFID-читачі, які забезпечують точну реєстрацію приходу і відходу персоналу. Всі дані, зібрані на термінальних точках, передаються на центральний сервер, де вони обробляються і зберігаються.

Центральний сервер використовує спеціалізоване програмне забезпечення для управління цими даними, що дозволяє керівництву компанії вести звітність, налаштовувати графіки роботи та аналізувати ефективність співробітників. Це програмне забезпечення також надає інструменти для розрахунку заробітної плати та управління відпустками та овертаймами, забезпечуючи цілісність даних та їхню актуальність.

Однією з ключових особливостей сучасних систем обліку робочого часу є їхня здатність інтегруватися з іншими корпоративними системами, такими як HRM (Human Resource Management) – управління людськими ресурсами та ERP (Enterprise Resource Planning) – планування ресурсів підприємства, а також з системами контролю доступу. Ця інтеграція підвищує рівень безпеки, дозволяючи не тільки відстежувати час, але й контролювати доступ до чутливих зон в приміщенні підприємства. Інтеграція з системами контролю доступу допомагає запобігати несанкціонованому входу, а також дозволяє вести повну історію переміщень співробітників всередині організації.

Крім того, сучасні системи забезпечують високий рівень захисту даних завдяки використанню сучасних технологій шифрування та автентифікації. Вони також дозволяють налаштувати різні рівні доступу до інформації, забезпечуючи, що конфіденційна інформація залишається захищеною від несанкціонованого доступу.

1.3 Застосування QR-кодів у сучасних системах контролю доступу та обліку робочого часу працівників

QR-код (Quick Response code, що в перекладі, – код швидкої відповіді) – це вид двовимірного штрих-коду, здатного зберігати значну кількість інформації у компактній формі. Ця технологія була розроблена в Японії в 1994 році компанією Denso Wave, спочатку з метою оптимізації процесу відстеження компонентів у виробництві автомобілів. Однак з часом її застосування значно розширилося.

QR-коди ефективно використовуються в багатьох сферах завдяки своїй здатності швидко бути розпізнаними сканувальним обладнанням та камерами мобільних телефонів. Вони можуть містити різні формати даних, включаючи URL-адреси вебсторінок, зашифрований текст, адреси електронної пошти, SMS-повідомлення, географічні координати, а також файли зображень у форматах gif, jpg, png. Ця багатофункціональність робить QR-коди незамінними в сучасних маркетингових та інформаційних стратегіях [13].

QR-коди широко використовуються в рекламі, де вони допомагають залучити потенційних клієнтів до візиту веб-сайтів, акцій або спеціальних пропозицій, просто скануючи код мобільним пристроєм. У соціальних мережах і на візитках QR-коди дозволяють миттєво обмінюватися контактною інформацією, спрощуючи процес додавання нових контактів.

Крім того, в умовах глобальної пандемії COVID-19, QR-коди набули особливої актуальності, дозволяючи безконтактну взаємодію. Вони використовуються для перегляду меню в ресторанах, реєстрації відвідувань у медичних та інших установах, замінюючи традиційні методи, що вимагають фізичного контакту.

Структура QR-коду досить проста – у формі квадрата. Це тому, що сканери QR-кодів краще розпізнають цю форму. Крім того, квадрат може максимізувати ефективність зберігання та передачі інформації. Всередині нього піксельне зображення дуже схоже на пазл і містить сім ключових елементів (рис. 1.5) [14].

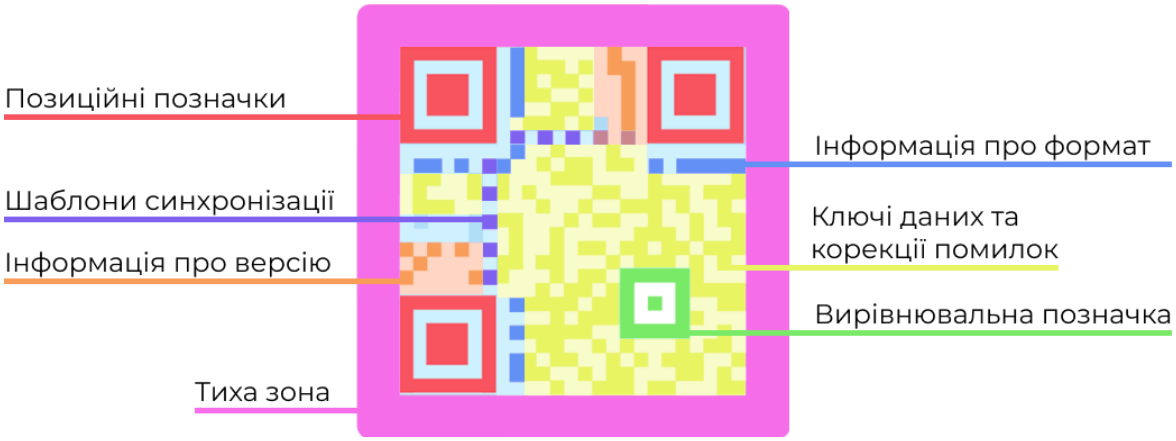

Рисунок 1.5 – Структура QR-коду

Складові QR-коду:

– позиційні позначки вказують, у який спосіб надруковано код;

– вирівнювальні позначки можуть допомогти з орієнтацією при використанні великих кодів;

– шаблони синхронізації – лінії, які повідомляють сканеру розмір матриці даних;

– інформація про версію використовується для позначення версій QR кодів, що використовуються. Їх існує понад 40 версій, але зазвичай часто використовуються версії від 1 до 7;

– у інформації про формат міститься інформація про допуск до помилок і шаблон маски даних для спрощення сканування коду;

– у ключах даних та корекції помилок відображаються самі дані;

– тиха зона – порожній простір, який допомагає сканувати програми, щоб відрізнити код від оточення [14].

Розрізняють два основних види таких кодів: статичні та динамічні. Кожен з цих типів має свої унікальні особливості та застосування, що дозволяє вибрати найбільш ефективний спосіб використання залежно від потреб користувачів та бізнес-сценаріїв.

Статичні коди – коди, які безпосередньо кодують інформацію. Після створення дані в ньому не можуть бути змінені без генерації нового коду.

На відміну від статичних, динамічні QR-коди дозволяють змінювати інформацію, на яку вони ведуть, не змінюючи сам QR-код. Це досягається завдяки тому, що QR-код вказує на URL, що веде до веб-сервера, де можна оновлювати цільовий контент.

У контексті обліку робочого часу та контролю доступу QR-коди можуть використовуватися для ідентифікації працівників при вході та виході з робочого місця. Працівник просто сканує QR-код за допомогою мобільного пристрою або спеціалізованого сканера, і система реєструє час приходу або відходу. Також є і інший варіант, коли у кожного працівника є особистий QR-код, який він сканує за допомогою спеціального сканера на підприємстві. Ці підходи спрощують процес обліку та зменшують можливість помилок.

Використання QR-кодів для обліку робочого часу має численні переваги. Вони включають в себе:

– підвищення точності та надійності обліку;

– зменшення адміністративних витрат та людських помилок;

– можливість отримувати миттєвий доступ до інформації про робочий час;

– вплив на продуктивність та ефективність працівників.

Незважаючи на численні переваги, існують певні недоліки використання QR-кодів для обліку робочого часу. Ось деякі з них:

– залежність від технічних засобів. Вимагає спеціалізовані сканери для підприємства, а також певних навичок у працівників;

– можливість підробки. QR-коди можуть бути підроблені, що створює ризик недостовірної інформації;

– безпека та конфіденційність. Збереження даних у цифровій формі може створювати проблеми з безпекою та конфіденційністю;

– необхідність оновлення та підтримки. Вимагає постійного оновлення та технічної підтримки;

– можливі проблеми зі зчитуванням. QR-коди можуть бути прочитані не завжди з першої спроби, особливо при наявності бруду або пошкоджень [15].

QR-коди є технологічно доцільними для інтеграції з сучасними системами контролю доступу, адже їх легко адаптувати для роботи з різними типами вхідних систем, такими як турнікети, двері та ворота. Їх компактна форма та можливість зберігати значну кількість інформації роблять їх ідеальними для швидкого зчитування цифровими камерами та сканерами. Така технологія може бути налаштована для інтеграції з RFID системами та біометричними технологіями, забезпечуючи додатковий рівень безпеки та ідентифікації.

Легкість впровадження таких систем визначається мінімальними технологічними вимогами. Установка та налаштування системи, що використовує QR-коди, не потребує спеціалізованого обладнання або великих технологічних вкладень. Більшість мобільних пристроїв та простих веб-камер можуть сканувати QR-коди, що знижує бар'єри впровадження та експлуатаційні витрати.

QR-коди вартісно ефективні, оскільки їх генерація майже безкоштовна і вони не потребують великих витрат на обслуговування. Це забезпечує зниження загальних витрат на систему контролю доступу, що особливо важливо для організацій з обмеженим бюджетом. Застосування QR-кодів сприяє підвищенню точності обліку робочого часу та зменшує можливість людських помилок, що є важливим фактором у підвищенні продуктивності та ефективності.

Використання QR-кодів забезпечує високий рівень прийняття з боку користувачів завдяки їхній простоті та зручності. Водночас, існують певні виклики, пов'язані з безпекою, оскільки QR-коди можуть бути схильні до копіювання та маніпуляцій. Однак, впровадження динамічних або шифрованих QR-кодів може значно знизити ці ризики, забезпечуючи додатковий рівень захисту інформації. Також, QR-коди легко інтегруються з іншими системами управління персоналом та безпеки, що дозволяє створювати комплексні рішення.

З огляду на вищезазначене, QR-коди вирізняються високою надійністю та легкістю масштабування, що робить їх ідеальними для використання в компаніях будь-якого розміру. При впровадженні такої системи важливо також врахувати вимоги до дотримання нормативних актів, зокрема у сфері захисту даних.

2 РОЗРОБКА АРХІТЕКТУРИ ТА ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Загальна архітектура проектованої системи

Система автоматизації контролю доступу та обліку робочого часу співробітників на основі QR-кодів призначена для точного обліку часу робітників, контролю доступу до підприємства, об’єктів, забезпечення безпеки та зручності використання. Було обрано мережеву систему, яка складається з кількох основних компонентів, які взаємодіють між собою для забезпечення точності та безпеки даних. Користувачами такої системи є адміністратори та звичайні користувачі. Основними функціями системи є аутентифікація та авторизація користувачів в системі, управління користувачами та їхніми даними в системі, генерація та зчитування QR-кодів для доступу, облік та аналіз відвідуваності. Для відображення основних компонентів та їх взаємозв’язків було розроблено структурну схему (рис. 2.1).

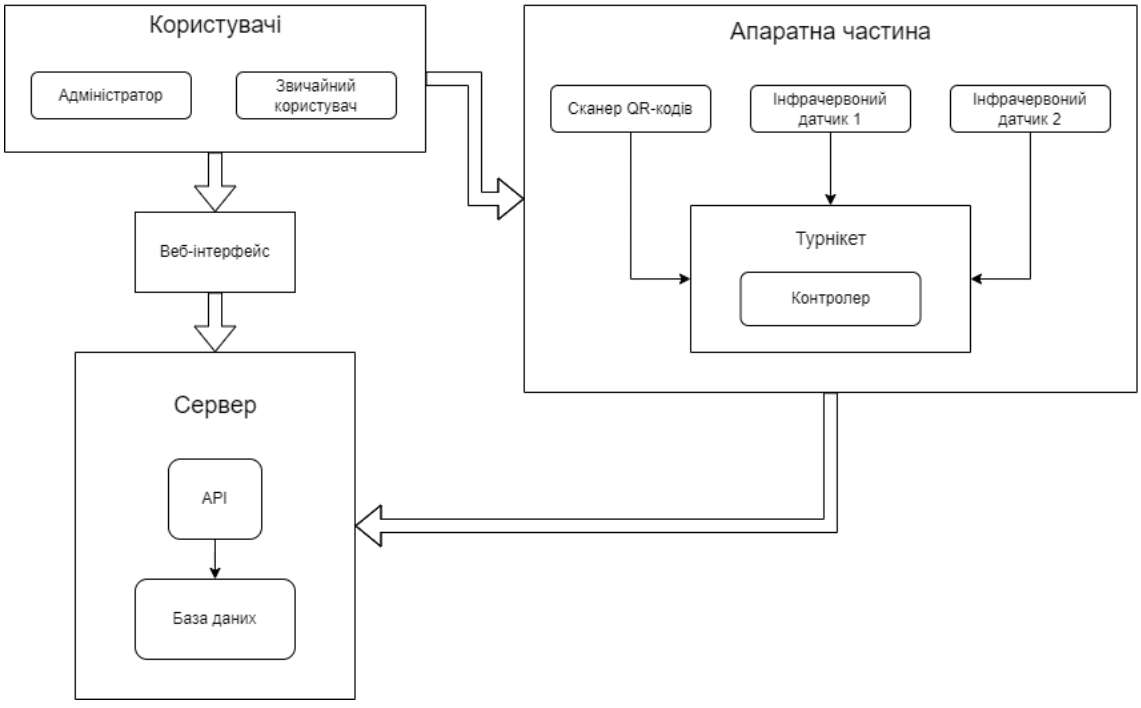

Рисунок 2.1 – Високорівнева структурна схема розроблюваної системи

У вищенаведеній схемі (рис. 2.1) видно, що система складається з декількох частин. Користувачі, які поділяються на адміністраторів та звичайних користувачів, взаємодіють з системою через веб-інтерфейс, а також є учасниками в процесах, які оброблює апаратна частина.

Веб-інтерфейс дозволяє користувачам, в залежності від ролі, взаємодіяти з системою, наприклад, переглядаючи власну статистику, щодо обліку робочого часу, а адміністраторам – управляти користувачами та переглядати загальну статистику. Веб-інтерфейс надсилає запити до серверної частини через API (Application Programming Interface) для отримання та відправки даних.

Серверна частина, яка являє собою API, оброблює запити від веб-інтерфейсу, взаємодіє з базою даних та апаратною частиною, що в свою чергу дозволяє всій системі працювати ефективно.

Апаратна ж частина включає в себе турнікет, до якого звертаються сканер QR-кодів та два інфрачервоні датчики. В свою чергу турнікет має контролер, який оброблює всю інформацію з відповідних пристроїв та взаємодіє з сервером для правильної обробки даних користувача та безпеки системи.

Зображення роботи системи в цілому наведено на рисунку 2.2.

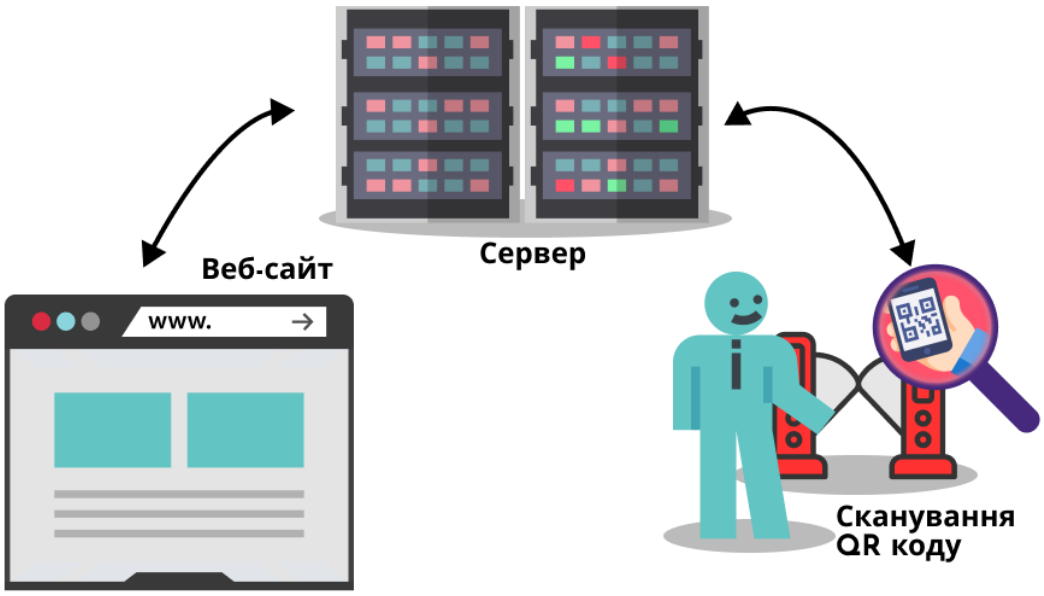

Рисунок 2.2 – Схема роботи розроблюваної системи

2.2 Опис технологій та компонентів

2.2.1 Опис технологій для програмної частини

2.2.1.1 Фреймворк ASP.NET Core

Сервер відіграє важливу роль в системі, так як він є центром логіки та управління. Він відповідає за обробку всіх запитів, що надходять від клієнтської частини (веб-інтерфейсу) та апаратних компонентів (турнікету). Сервер виконує функції авторизації та аутентифікації користувачів, управління даними, отриманими від апаратних компонентів, та їх збереження в базі даних. Він також забезпечує реалізацію бізнес-логіки, такої як облік робочого часу, обробка даних про відвідуваність та іншої логіки системи.

Для написання серверу було обрано технологію ASP.NET Core, що має шаблон Web API. Для використання цієї технології було обрано мову C#.

C# – це сучасна, об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона є основною мовою для розробки на платформі .NET, включаючи ASP.NET Core Web API. C# відзначається простотою синтаксису, високою продуктивністю та широкими можливостями для розробки різноманітних застосунків, від веб і десктопних до мобільних та хмарних. Завдяки потужним засобам для роботи з асинхронними операціями, обробкою винятків, та інтеграцією з іншими технологіями, C# є ідеальним вибором для створення надійних та масштабованих рішень.

ASP.NET Core було обрано відповідно до наступних плюсів:

– кросплатформеність, що дозволяє працювати на різних операційних системах включаючи Windows, macOS та Linux. Це забезпечує гнучкість у виборі середовища для розгортання застосунків та знижує залежність від конкретної платформи;

– продуктивність є одним із найбільших переваг цієї технології. Так як ASP.NET Core використовують на C#, що має вбудований збірник сміття, дуже рідко виникають проблеми з пам’яттю;

– фреймворк має вбудовані засоби для аутентифікації та авторизації, захисту від підробки міжсайтових запитів та шифрування даних, що в свою чергу забезпечує високий рівень безпеки веб-застосунків та API;

– ASP.NET Core побудований на основі модульної архітектури, що дозволяє легко додавати або вилучати необхідні компоненти, спрощуючи процес розробки та підтримки системи, а також масштабуючи її функціональність;

– технологія дуже легко інтегрується з іншими технологіями, а також має дуже багато бібліотек, які можуть спростити написання додатків та різного роду застосунків;

– фреймворк має активну спільноту шанувальників та розробників, що допомагає у вирішенні різного роду питань та проблем.

Досить вагомим мінусом цієї технології є поріг входу. Тобто для новачків або розробників, як не мають досвіду роботи з платформами Microsoft, необхідно задіяти досить багато вільного часу та зусиль для освоєння. Також ASP.NET Core є сам по собі великим і складним фреймворком, що збільшує час розробки та потребу в налаштуванні.

2.2.1.2 Фреймворк Blazor WASM

Для написання та використання Web-інтерфейсу було обрано Blazor WebAssembly. Цей фреймворк використовується для створення інтерактивних веб-застосунків за допомогою C# та .NET. Ця технологія дозволяє запустити код C# безпосередньо у браузері завдяки технології WebAssembly, що усуває потребу в JavaScript для клієнтської логіки. Blazor WASM (WebAssembly) забезпечує високу продуктивність, дозволяє повторно використовувати код між серверною та клієнтською частинами, та спрощує розробку завдяки використанню єдиної мови програмування. Інтеграція з існуючою екосистемою .NET та підтримка компонентно-орієнтованої архітектури роблять Blazor WASM відмінним вибором для створення сучасних, масштабованих та багатофункціональних веб-застосунків.

Поряд з багатьма плюсами фреймворк має кілька недоліків:

– великий обсяг початкового завантаження може уповільнити старт застосунку;

– високе навантаження на ресурси клієнтського пристрою може бути проблемою для старих або малопотужних пристроїв;

– обмежена підтримка багатопоточності та доступу до деяких браузерних API;

– можливі проблеми зі сумісністю на старих версіях браузерів або менш популярних платформах.

2.2.1.3 РСУБД Microsoft SQL Server

При виборі баз даних було зосереджено увагу на реляційних базах даних, так як вони використовують структуру таблицю для організації та зберігання даних, що є неймовірно зручно відповідно до розроблюваної системи контролю доступу та обліку робочого часу.

Одним із найпопулярніших реляційних систем управління базами даних (РСУБД) є Microsoft SQL Server. Вона легко інтегрується з іншими продуктами Microsoft, підтримує різні типи даних, індексацію, збережені процедури, транзакції та реплікацію. SQL Server також пропонує розширені можливості для резервного копіювання та відновлення даних, що робить його надійним вибором для створення сучасних інформаційних систем.

Крім того, SQL Server має високу продуктивність та безпеку, що дозволяє забезпечити швидкий доступ до даних та захистити їх від несанкціонованого доступу. Завдяки масштабованості та підтримці великих обсягів даних, SQL Server є ідеальним рішенням для підприємств будь-якого розміру.

2.2.1.4 Фреймворк WPF

У зв’язку з відсутністю фізичної можливості розробити фізичний макет турнікету, було прийнято рішення провести симуляцію його роботи за допомогою програмного забезпечення.

При виборі програмного забезпечення було зосереджено увагу на стек технологій Microsoft та відповідно мови програмування C#. Тому було обрано один із найпопулярніших платформ для побудови інтерактивних настільних застосунків – WPF (Windows Presentation Foundation).

Цей фреймворк використовує XAML (Extensible Application Markup Language) для опису інтерфейсі користувача за допомогою мови розмітки. Основні переваги WPF включають багаті можливості для створення користувацьких інтерфейсів, підтримку прив'язки даних (data binding) і шаблонів (templates), а також потужний механізм подій та команд. Завдяки цим характеристикам, WPF дозволяє створювати високоякісні, інтерактивні та масштабовані настільні застосунки.

Мінуси WPF включають складність вивчення та використання, особливо для новачків, високу вимогливість до апаратних ресурсів, що може вплинути на продуктивність на старих або малопотужних пристроях, та обмежену підтримку інших платформ, оскільки WPF працює виключно на Windows.

2.2.2 Опис апаратних компонентів

Так як апаратна частина системи займає ключову позицію в системі, то вибір складових апаратної частини є важливою і дуже відповідальною задачею. Однак, через відсутність можливості створити фізичний макет апаратної частини, у цій роботі буде проводитись симуляція її функціонування. Це дозволить детально розглянути всі аспекти роботи системи та оцінити її ефективність.

Симуляція буде виконана з використанням програмних засобів, які дозволяють моделювати роботу апаратних компонентів Такий підхід забезпечить можливість проаналізувати взаємодію та загальну працездатність системи без необхідності створення реального фізичного макету.

Це рішення дозволяє зосередитись на програмному забезпеченні та логіці роботи системи, що є важливим етапом розробки.

2.2.2.1 Турнікет TS1000 Pro

TS1000 Pro – це триподовий турнікет зі шліфованої нержавіючої сталі (рис. 2.3), який оснащений автоматичною функцією антипаніка.

В порівнянні з іншими схожими турнікетами, до TS1000Pro можна додавати інші модулі, які можуть покращувати систему, а також замінювати існуючі, це робить TS1000Pro гнучким рішенням для будь-якого підприємства.

Турнікет забезпечує надійний контроль доступу з одиночною смугою для швидкого проходу. У режимі очікування він залишається заблокованим, і розблоковується натисканням кнопки на пульті керування. Під час екстрених ситуацій або відключення електроенергії штанги автоматично опускаються, дозволяючи безпечний вихід.

Рисунок 2.3 – Турнікет TS1000 Pro [16]

TS1000 Pro вирізняється високою надійністю та безпекою. Маточина штанг виготовлена з легованої сталі, а корпус і штанги – з нержавіючої сталі SUS304, що забезпечує довговічність і стійкість до зовнішніх впливів. Турнікет працює в енергозберігаючому режимі, де електромагнітні замки знаходяться в стані очікування до активації. Запатентована система змащення маслом знижує потребу в обслуговуванні, а високоякісні електричні компоненти гарантують стабільну роботу.

Турнікет оснащений функціями безпеки, які включають автоматичне розблокування та опускання штанг під час надзвичайних ситуацій або відключення живлення, що дозволяє людям швидко і безпечно покинути територію. Гладка поверхня без відкритих гвинтів забезпечує додаткову безпеку, а візуальні індикатори надають інформацію про надання доступу або відмову. Ергономічний дизайн дозволяє швидко і зручно виконувати аутентифікацію по карті або відбитку пальця, а функція «Anti-tailgating» запобігає несанкціонованому проходу [16].

Основні технічні характеристики турнікету наведено в таблиці 2.1 [16].

Таблиця 2.1 – Технічні характеристики TS1000Pro

Назва характеристики	Значення
Вимоги до живлення	AC 100 В / 220 В, 50/60 Гц Старт – 68 В Очікування – 15 В Відкриття – 60 В
Робоча температура	від -28 °C до 60 °C
Робоча вологість	від 5% до 80%
Ширина смуги руху	500 мм
Розміри (мм × мм)	520 мм × 810 мм
Розміри (мм)	Довжина – 520 мм Ширина – 310 мм Висота – 1010 мм
Вага	34 кг.
Світлодіодний індикатор	Так
Матеріал	SUS304 Нержавіюча сталь
Бар’єрний рух	Обертання
Аварійний режим	Так
Рівень безпеки	Середній

Для взаємодії з турнікетом сторонніх компонентів, використовується сама плата управління та панель контролю доступу, до якої приєднуються зовнішні компоненти. Загальний вигляд схеми з’єднання плати управління та панелі контролю доступу наведено на рисунку 2.4.

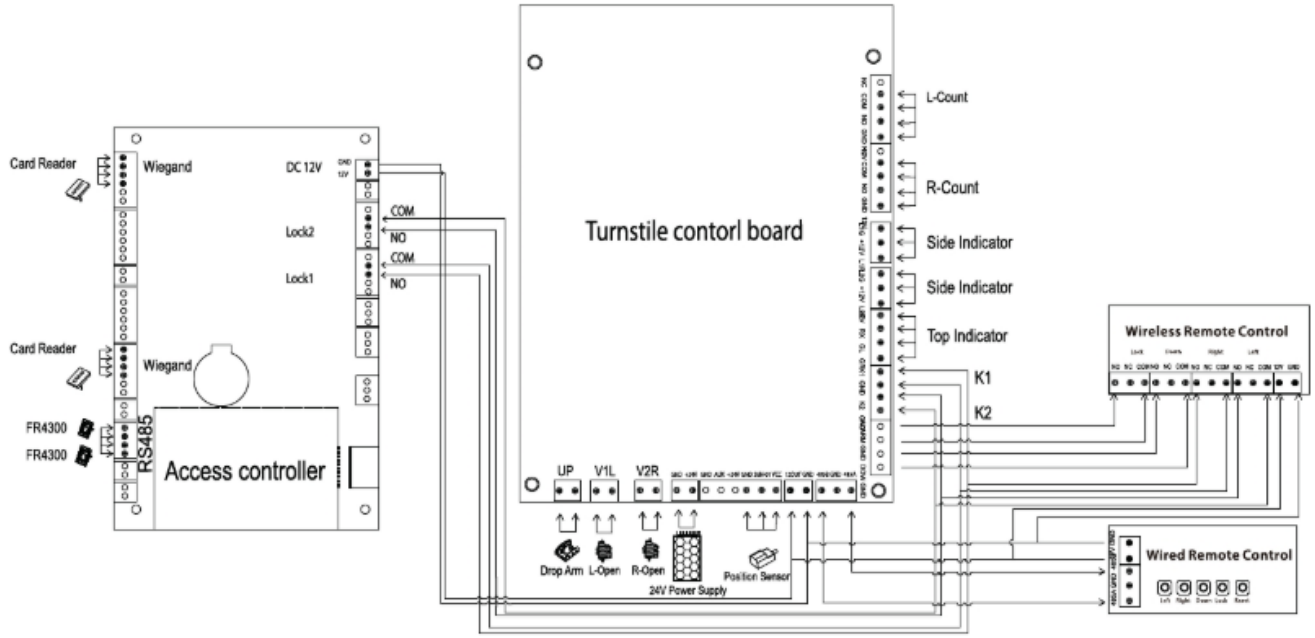

Рисунок 2.4 – Схема з’єднання плати управління та панелі контролю доступу [16]

2.2.2.2 Сканер QR-коду QR500

QR500 є інноваційним зчитувачем QR-кодів нового покоління, призначеним для контролю доступу. Він характеризується швидким скануванням та високою точністю розпізнавання, а також відмінною сумісністю з будь-якими контролерами доступу, які підтримують інтерфейс Wiegand.

До основних характеристик зчитувача входять:

– нові технології контролю доступу за допомогою QR-кодів;

– вбудована антена зчитувача карт з робочою частотою 13,56 МГц;

– підтримка MF, CPU, NFC (аналогової карти), DESFire EV1, китайських ID-карт, QR-кодів;

– підтримка Wiegand, RS485, USB, TCP/IP мережевих режимів виводу [17].

Загальний вигляд зчитувача QR-коду наведено на рисунку 2.5.

Для підключення цього зчитувача QR-коду QR500, спочатку необхідно підключити його до контролера через Wiegand або RS485, а потім підключити до джерела живлення +12 В.

Рисунок 2.5 – Зчитувач QR-коду QR500 [17]

Загальне підключення через Wiegand або RS485, наведено на рисунку 2.6.

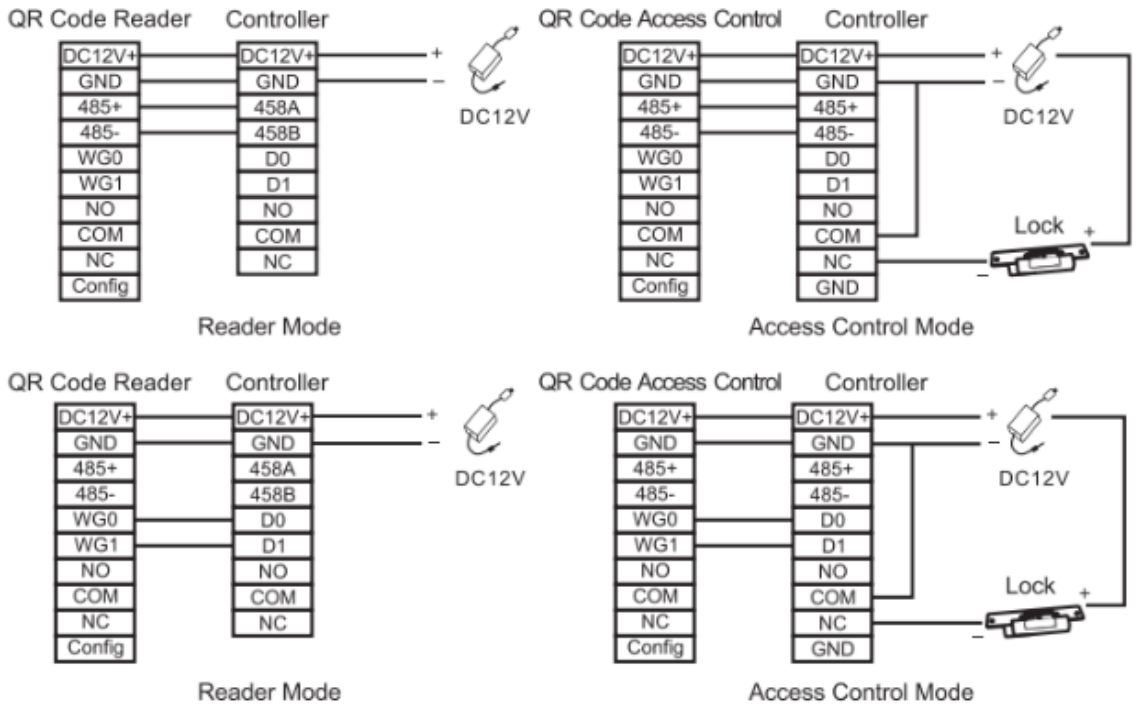

Рисунок 2.6 – Загальне підключення QR500 через Wiegand або RS485 [17]

Технічні характеристики зчитувача QR-коду наведено в таблиці 2.2 [17].

Таблиця 2.2 – Технічні характеристики QR500

Назва характеристики	Значення
Стандарт	Протокол ISO / IEC 14443A RS485
Код	QR, Data Matrix, PDF417, GS1 Databar, Code 128/EAN12, UPC/EAN, Codabar, Code 39/Code 93
Частота	13,56 МГц
Живлення	12 В, USB 5 В.
Відстань зчитування	QR до 5 см
Зв’язок	Wiegand, RS485, USB
Робоча температура	від -10 С° до 50 С°
Вологість	від 20% до 80%
Габарити	86 мм × 86 мм × 50 мм

Цей зчитувач було обрано, так як він підтримує декілька інтерфейсів, таких як Wiegang, RS485, USB та є можливість зв’язку за допомогою RS232, що є відмінним варіантом для турнікетів, так як контролери доступу у турнікетах можуть бути різними, з різними підключеннями. Це полегшує його інтеграцію забезпечуючи гнучкість та універсальність в підключенні. Якщо порівнювати з іншим сканером, наприклад Zebra DS2208, там також декілька інтерфейсів підключення, але значно менше і, все ж таки, більш орієнтований на підключення по USB. Також порівнюючи з Zebra DS2208, QR500 має більшу робочу температуру та захист від пилі та води, що також є великим плюсом на підприємстві. Zebra DS2208 більш розрахований для внутрішнього використання, наприклад, в офісах.

2.2.2.3 Інфрачервоний датчик відстані E18-D80NK

Інфрачервоний датчик відстані E18-D80NK (рис. 2.7). Датчик може визначати наявність об'єктів на відстані від 10 см до 80 см, яку можна налаштувати за допомогою підлаштовного резистора, розташованого на зворотному боці пристрою. Вихідний сигнал – цифровий бінарний, який інформує про наявність перешкоди в зоні видимості. Працює при живленні 5 В.

Датчик складається з інфрачервоного передавача та приймача. На відміну від ультразвукового датчика відстані, цей пристрій не вимірює точну відстань до перешкоди, а лише визначає, чи є перешкода в його зоні видимості. Коли датчик активується, вихідна напруга стає рівною "1" [18].

Для проекту було обрано інфрачервоний датчик E18-D80NK завдяки його перевагам над аналогами HC-SR501 та E3F-DS30P2. E18-D80NK має налаштовуваний діапазон від 10 см до 80 см, що забезпечує точне виявлення об'єктів на коротких відстанях. Працює при живленні 5 В, тоді як E3F-DS30P2 потребує від 10 В до 30 В DC. Вихідний сигнал E18-D80NK є цифровим бінарним, що спрощує інтеграцію з системами контролю доступу. Компактний розмір 50 мм х 18 мм і простота налаштування роблять E18-D80NK зручним для використання у вузьких просторах. У порівнянні, HC-SR501 більше підходить для виявлення руху на великих відстанях, а E3F-DS30P2 має більш складні налаштування і габарити.

Таким чином, E18-D80NK був обраний за його точність, універсальність у живленні, простоту інтеграції та компактний дизайн.

Рисунок 2.7 – Інфрачервоний датчик відстані E18-D80NK [18]

2.3 Функціональні можливості системи

2.3.1 Функціональні можливості Backend частини

Серверна частина системи контролю доступу та обліку робочого часу включає в себе API та БД.

База даних буде відповідати за зберігання наступної інформації:

– дані про користувачів;

– дані про запрошених користувачів та самі запрошення;

– дані про ролі;

– дані про посади підприємства;

– дані про QR-коди;

– дані про відвідування.

API включає в себе реалізацію бізнес-логіки системи, тобто виконує обробку як запитів від апаратної частини, так і від Frontend частини. До функцій API відносяться:

– реалізація аутентифікації та авторизації користувачів, за допомогою яких забезпечується реєстрація нових користувачів, автентифікація існуючих користувачів, а також керування дозволами та ролями, для забезпечення надійності системи;

– можливість керування користувачами, що включає в себе перегляд наявної інформації про користувача, його статистики, можливість редагування та видалення користувача з системи;

– створення функціоналу пов’язаного із запрошеннями нових робітників через пошту, так як необхідно, щоб в системі могли реєструватися тільки запрошені користувачі. До цього функціоналу відносяться: заповнення даних про робітника, відправлення на пошту запрошення, перегляд запрошених користувачів та статус запрошення, видалення запрошених користувачів;

– створення функціоналу із генерацією QR-кодів: статичних та динамічних. Тобто створення при кожній генерації нового коду з датою створення, самим кодом шифрування та з датою закінчення дії коду. Також створення сервісу для

автоматичного видалення QR-кодів, в яких закінчився термін дії, а також перевірка валідності QR-коду системою;

– реалізація функцій щодо обліку робочого часу користувачів, тобто створення логіки «входу» та «виходу» користувачів на основі QR-коду, можливість перегляду інформації, щодо обліку робочого часу;

– забезпечення валідацією всіх вхідних даних;

– реалізація взаємодії з БД, створення об’єктів бази даних, а також налаштування підключення до БД;

– налаштування конфігураційного файлу, до якого входять базові налаштування системи, наприклад, час дії QR-кодів, рядок підключення до БД, дані для забезпечення безпеки системи і інші дані для коректної роботи системи;

– створення кінцевих точок для комунікації інших компонентів із серверною частиною.

2.3.2 Функціональні можливості Frontend частини

Frontend частина системи включає в себе веб-інтерфейс для користувачів та адміністраторів. Основні сторінки та функціонал:

– привітальна сторінка, яка забезпечує початок роботи з веб-інтерфейсом;

– сторінка логіну та реєстрації для аутентифікації та авторизації користувачів;

– сторінка для звичайного користувача, де відображається основна інформація про облік робочого часу цього користувача;

– сторінка для генерації QR-кодів та можливості їх збереження;

– сторінка для адміністратора, яка надає доступ до можливості управління працівниками, перегляду інформації про користувача, створення запрошення користувача, управління запрошеннями, перегляду відвідуваності всіх користувачів та окремо користувача, перегляд статистики робітників;

– сторінки з помилками різного типу, наприклад, «сторінку не знайдено», «не надано доступ», «невірне запрошення-посилання» та «користувач не авторизований».

2.3.3 Функціональні можливості апаратної частини

Апаратна частина системи повинна включати заходи щодо безпеки підприємства та передавання інформації про відвідування працівників на сервер.

Так як було вирішено зробити симуляцію роботи апаратної частини, отже їй необхідно мати наступні функції та можливості:

– перевірка QR-коду, шляхом сканування та відправлення запиту до серверу, отримання відповіді;

– відмова у доступі у разі невдачі при сканування QR-коду;

– забезпечення доступу до входу чи виходу при вдалому скануванні;

– зчитування інформації з інфрачервоних датчиків;

– відправлення інформації на сервер, якщо людина повністю пройшла через турнікет.

2.4 Комунікація компонентів системи

Серверна частина є центральним вузлом в системі контролю доступу та обліку робочого часу, що об’єднує всі компоненти системи. Вона складається з API, реалізованого за допомогою ASP.NET Core, та бази даних на основі SQL Server. API забезпечує зв'язок між frontend, апаратними компонентами та базою даних.

Процес передачі даних від апаратної частини до серверу та від Frontend до серверу відбувається за допомогою HTTPS протоколу. Використовуючи адрес API, а також завчасно налаштовані кінцеві точки, можна звертатися до API.

Отже, для того, щоб звернутися до API і отримати якісь дані, наприклад, список співробітників, необхідно написати адресу API, тобто адресу на якій знаходиться наша серверна частина. У випадку з локальними застосунками це може бути наступна адреса: «https://localhost:7271». До цієї адреси додається шлях, який містить контрольна точка, наприклад, «api/employee/list» і, таким чином, формується повний запит «https://localhost:7271/api/employee/list», який працює по протоколу HTTPS.

Для наглядної демонстрації контрольних точок, було використано інструмент Swagger, який використовується для автоматичної генерації документації по API. Він дозволяє розробникам легко переглядати та тестувати всі доступні кінцеві точки API. Як приклад, вигляд зображення кінцевих точок API за допомогою Swagger наведено на рисунку 2.8.

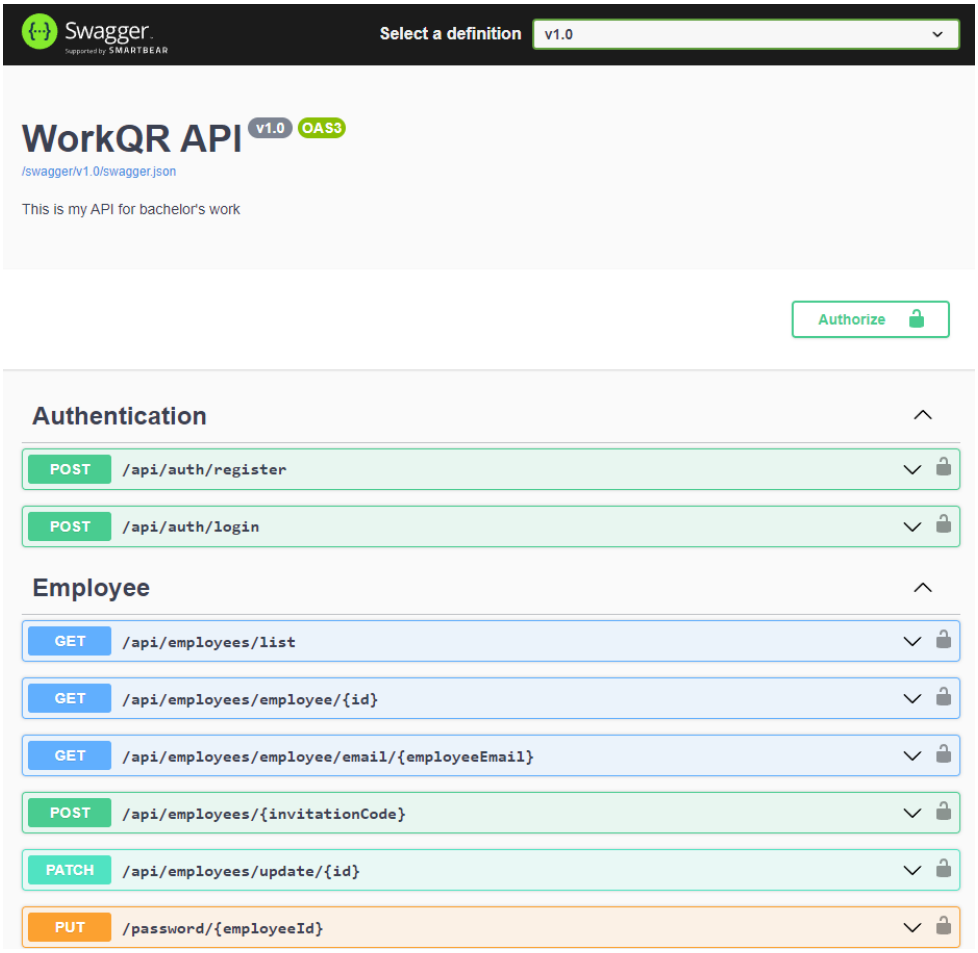

Рисунок 2.8 – Вигляд кінцевих точок API, використовуючи Swagger

В результаті звернення до запиту, буде отримано відповідь, в яку входить код та тіло відповіді. Код може бути декількох варіацій:

– «1xx» – це коди статусу, які інформують клієнта про те, що запит отриманий та обробляється;

– «2xx» – це коди статусу, що інформують про успішний запит;

– коди зі статусом «3xx», інформують клієнта про те, що для завершення дії необхідні додаткові дії;

– «4xx» – це коди помилки клієнта, що означають помилку запиту або запит не може бути виконаний, наприклад, «Not Found (не знайдено)»;

– «5xx» – коди статусу, що вказують на помилку зі сторони сервера.

Отже, сформувавши запит і відправивши його на API, наприклад, по адресу «https://localhost:7271/api/employee/list», у разі успіху ми отримаємо, код «2xx» та тіло відповіді, що наведено на рисунку 2.9.

Рисунок 2.9 – Результат успішного виконання запиту до API

У разі неуспішного виконання запиту, буде отримано код «4xx» або «5xx» та тіло відповіді з помилкою (рис. 2.10).

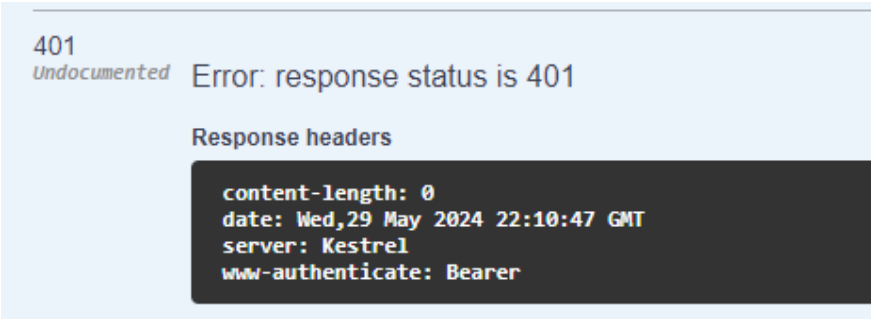

Рисунок 2.10 – Результат невдалого виконання запиту

Як видно з рисунка 2.9, тіло відповіді має відповідає формату JSON (JavaScript Object Notation). JSON є одним із найпопулярніших форматів для обміну даних між клієнтом і сервером завдяки своїй простоті та читабельності.

JSON складається з пар «ключ-значення» і має підтримку багатьох типів, наприклад, «об’єкти», «масиви», «рядки», «числа» і так далі. Тобто саме JSON дозволяє зручно обмінюватись даними з клієнтом та сервером.

2.5 Забезпечення безпеки системи

В системі при роботі API та веб-інтерфейсу використовується протокол HTTPS (HyperText Transfer Protocol Secure). HTTPS – це протокол, що забезпечує безпечний обмін даними між веб-браузером користувача та веб-сервером. Завдяки сертифікатам SSL(Secure Sockets Layer)/TLS(Transport Layer Security) забезпечується автентичність сайту, тобто користувач може бути впевнений, що взаємодіє саме з тим сайтом, який заявляє про себе, а не з фальшивим. Також цей протокол гарантує, що дані в процесі передачі не будуть підроблені або змінені.

Для захисту самої API та веб-інтерфейсу застосовується аутентифікація та авторизація за допомогою JWT токенів. JWT (JSON Web Token) – це сам по собі компактний, самодостатній спосіб передачі інформація між сторонами у вигляді JSON-об’єкта. Інформація в JWT може бути перевірена і довірена, оскільки вона підписана цифровим підписом. Підпис у JWT токені шифрується за різними стандартами, наприклад SHA256. JWT токен складається з трьох частин:

– заголовок (header), що містить тип токена і алгоритм підпису;

– головна частина (payload), яка містить дані про користувача;

– підпис (signature), що забезпечує автентичність токену.

JWT виглядає як «xxxxx.yyyyy.zzzzz», де кожна частина закодована в Base64Url.

У аутентифікації JWT токен використовується для передачі тіла запиту. API вимагає активний JWT-токен в заголовках запиту, бо без нього повертається помилка «Unauthorized».

Важливу роль у безпеці відіграє розподіл користувачів за ролями, що обмежує доступ до функцій системи залежно від ролі користувача.

3 ОПИС ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ РОЗРОБЛЮВАНОЇ СИСТЕМИ

3.1 Опис програмної частини розроблюваної системи

3.1.1 Розробка та структура API

Серверна частина в системі контролю доступу та обліку робочого часу займає головну роль, бо вона оброблює всі запити від інших компонентів системи. Саме завдяки API, що працює разом з БД, система має змогу працювати злагоджено та повноцінно.

API було розроблено з використанням RESTful принципів, що забезпечує гнучкість взаємодії з веб-інтерфейсом та з програмним застосунком, що симулює апаратну частину.

Перед початком створення головної логіки для API, спочатку було проведено загальне налаштування головних компонентів для загальної роботи API. В основному файлі, який ініціює запуск роботи API, «Program.cs», лістинг коду якого наведено в лістингу А.1 додатка А, було налаштовано наступне:

– додано та налаштовано файли конфігурації для API: «appDescription.json» (лістинг А.2) та «appsetting.json» (лістинг А.3), в яких зберігається базова інформація щодо головної логіки, наприклад, час роботи довготривалого QR-коду, час закінчення валідності JWT-токену та інші налаштування;

– впроваджено шаблон проєктування програмного забезпечення DI;

– було додано опцію роботи з контролерами, через які будуть встановлюватись кінцеві точки для реалізації шаблону RESTful API, а також налаштовано роботу з файлами типу «JSON» в цих контролерах;

– налаштовано документацію для API – «Swagger»;

– налаштовано підключення до БД;

– додано необхідні кастомні сервіси в DI, для роботи з сервісним принципом програмування;

– впроваджено та налаштовано валідацію та «мапування»(перетворення) даних;

– створено та додано в конвеєр проміжних програмних засобів DI «middleware», який оброблює всі помилки API, для того, щоб при виникненні помилок, API продовжував працювати;

– налаштовано CORS (Cross-origin resource sharing), для надання доступу до API зовнішнім компонентам системи, наприклад, веб-інтерфейсу;

– додано та налаштовано аутентифікацію та авторизацію.

Для подальшої взаємодії з БД та використання в повній мірі ООП, було створено сутності, які є абстрактною моделлю реального об’єкта у рамках інформаційної системи. Для бізнес-логіки було спроектовано та реалізовано наступні сутності, лістинг змісту яких наведено в додатку А у лістингах від А.2 до А.8 відповідно:

– «EmployeeEntity», який відображає основні дані робітника, а також його зв’язки та залежності з іншими сутностями;

– сутність «InvitationInfoEntity», що представляє собою модель запрошення, яке містить основні дані запрошення, тобто дата створення запрошення, термін закінчення запрошення, код запрошення, активність запрошення, а також інші зв’язки з сутностями;

– «InvitedEmployeesEntity» – це модель, яка відображає інформацію про запрошеного робітника, а також містить зв’язки з іншими моделями;

– сутність «JobPositionEntity» являє собою відображення моделі позиції робітника на підприємстві та має назву позиції та зв’язок з деякими іншими сутностями системи;

– «QrCodeEntity» відображає основні можливі дані, щодо QR-коду, такі як унікальний зашифрований код, який вкладається в QR-код, термін придатності коду, тип коду та зв’язки з відповідними сутностями, які використовують QR-код;

– «RoleEntity» як і «JobPositionEntity» є відображенням моделі, але саме ролі робітника у системі, що містить назву ролі та залежності з іншими сутностями;

– «VisitRecordEntity», що являє собою сутність, яка відображає модель даних про відвідування, тобто дата та час запису відвідування, дата та час входу та виходу, а також залежності з іншими сутностями.

Відповідно до логіки роботи API, було додано два переліки з позиціями та ролями користувачів. В системі міститься 3 позиції: «Employee», «SecurityGuard» та «Cleaner», які при необхідності можна розширити чи, навпаки, скоротити. Також є дві ролі: «User» та «Admin».

Для можливості повідомляти користувачам про помилки було розроблено власну обробку помилок та створено власні класи, які відображають тип помилок. Лістинг коду обробника помилок «ExceptionHandlerMiddleware» наведено в додатку А, в лістингу А.9.

До власних класів, що відображають тип та суть помилок належать:

– «CustomException», що є базовим класом, від якого наслідуються всі інші власні класи для обробки помилок. Тіло класу наведено в лістингу А.10 додатка А. Він містить HttpStatusCode, який буде вказувати на код помилки, а також рядок с повідомленням про помилку. Ці поля повинні визначати всі інші класи, що наслідуються від головного;

– «BadRequestException» являє собою помилку про неправильний запит. Він оброблює помилки, які відносяться до запитів, що не відповідають вимогам системи та головної логіки, наприклад, невірні або неповні дані для входу в систему, та має код помилки «400»;

– «NotFoundException» – це клас, який оброблює помилки, що стосуються не знайдених даних у системі, наприклад, не знайдено відповідного користувача за його ідентифікатором. Ця помилка має код «404»;

– «ForbiddenException» відповідає за обробку помилок, які виникають при спробі запиту отримати доступ до частини застосунку, до якого прав цей запит не має. Цей тип помилок має код «403».

Для реалізації основної логіки роботи API, було створено сервіси, що включають основні методи для роботи з даними, які необхідні системі.

Так як API складається із сервісів, які відповідають конкретно за якусь частину логіки нашої системи, було вирішено розділити сервіси на області, які відповідають певній логікі.

Майже у кожній області є моделі DTO, тобто моделі, які використовуються для того, щоб частково або в певній формі передати дані сутностей зовнішнім компонентам. Тобто якщо сутність «EmployeeEntity» містить ім’я, прізвище, пошту, захешований пароль та різні залежності та зв’язки, то DTO відображає лише частину даних головної сутності, що дозволяє в певній мірі безпечно та без зайвої інформації передавати дані іншим компонентам, які звертаються до API.

Для того, щоб перетворювати властивості головної сутності до моделі DTO, було використано «мапування» (перетворення) даних за допомогою бібліотеки «AutoMapper», який при налаштуванні дозволяє, автоматично перетворювати дані головної сутності у вказану модель. Приклад налаштування та використання «мапування» наведено в лістингу А.11 та А.12 додатка А.

Також при отриманні інформації від інших компонентів необхідно перевіряти правильність цих даних, так як використання неправильно надісланих даних може підірвати безпеку системи, а також її логіку. Для валідації даних було використано бібліотеку «FluentValidation», яка так само, як і «AutoMapper», при правильному налаштуванні, може автоматично валідувати вхідні дані. Приклад налаштування валідації наведено в лістингу А.13 додатка А.

Для використання в повній мірі DI, в кожній області створюється інтерфейс, який представляє загальні методи та поля, які повинні бути у сервісі відповідної області, а також сам клас-сервіс, який реалізує логіку цих методів та визначає поля.

Одна з найголовніших областей – область «Employee», яка містить сервіс для роботи з працівниками підприємства. Інтерфейс та клас реалізації логіки з працівниками наведено в додатку А лістингах А.14 та А.15 відповідно.

Як видно з лістингу А.14, сервіс містить декілька наступних методів:

– «GetListAsync» – дозволяє отримати список всіх користувачів системи;

– «GetByIdAsync» – дозволяє отримати дані про користувача за його ідентифікатором;

– «GetByEmailAsync» – метод для отримання даних про користувача за його поштою;

– «CreateAsync» – метод, який дозволяє створити користувача за кодом запрошення та введеним паролем;

– «UpdateByIdAsync» – метод, для оновлення даних про користувача за його ідентифікатором та самими даними для оновлення;

– «DeleteByIdAsync» дозволяє видалити користувача за його ідентифікатором;

– «UpdatePasswordByIdAsync» – метод для оновлення пароля для користувача за його ідентифікатором та відповідними даними для оновлення пароля.

Детальна реалізація цих методів наведена в лістингу А.15 додатка А.

Також було створено такі області як «Authentication», «InvitedEmployee», «JobPosition», «QrCode», «Role», «VisitRecords». Згідно до назв цих областей, було створено сервіси, які реалізують певну логіку. Методи цих сервісів майже у всіх схожі на ті, що має область «Employee», тобто, отримати всі дані, отримати дані по ідентифікатору, отримати дані по іншій властивості, створити, оновити, видалити, але в залежності від області реалізація цих методів відрізняється.

Область «InvitedEmployee» містить реалізацію запрошень нового користувача, використовуючи засоби пошти, а також логіку зі створенням нового запрошеного користувача, яким можна керувати.

Сервіс пов’язаний з «VisitRecords» має методи щодо реєстрації відвідувань користувача, тобто реєстрація входу та виходу користувача із системи.

До області «QrCode» відноситься сервіс, який дозволяє генерувати різні типи коду, зображення коду, а також перевіряти QR-код на валідність.

Після того, як було створено сервіси, які відповідають за головну логіку нашої системи, необхідно зареєструвати їх в DI. Для цих цілей було створено окремий клас «ApiExtensions», який дозволяє розширити функціонал головного

виконуючого файлу «Program.cs». В цьому файлі було створено метод «ServicesExtensions», який є розширенням до «IServiceCollection», з якого складається наш DI, тому в цьому методі було зареєстровано всі сервіси, що були створені раніше. Загальний вигляд реєстрації всіх сервісів, що було створено для реалізації головної логіки системи, наведено в лістингу А.16 додатка А.

Для того, щоб перейти до реалізації контролерів, які відповідають за обробку вхідних запитів від інших компонентів системи, необхідно налаштувати аутентифікацію, авторизацію, а також CORS. Для цього у файлі «ApiExtensions» було створено метод «AddApiAuthentication», який відповідає за налаштування JWT-токену в системі, встановлює авторизацію, а також налаштовує CORS API для доступу до API зовнішніх компонентів. Лістинг коду метода «AddApiAuthentication» наведено у лістингу А.17 додатка А. В цьому методі було обрано метод аутентифікації «JwtBearer», налаштовано валідацію цього метода аутентифікації за часом, а також за ключем.

Після налаштування аутентифікації та авторизації, для того, щоб обмежити доступ до відповідних контролерів, необхідно використати атрибут «Authorize».

Наступним кроком було створення контролерів, які відповідають описаним раніше областям. Ці контролери використовують функціонал відповідного сервісу. Для того, щоб створити контролер, необхідно створити клас, який буде закінчуватися на «Controller» та пронаслідуватися від «ControllerBase», щоб використати весь функціонал контролерів. Зверху контролерів додаються основні атрибути такі як:

– «ApiController» – вказує, що цей клас є контролером;

– «Route» – задає в параметрах атрибута шлях до цього контролера, наприклад, «api/auth»;

– «Produces(MediaTypeNames.Application.Json)» – встановлює тип запитів та відповідей від контролера у вигляді «JSON»;

– «Authorize» – є опціональним атрибутом, який може встановлюватись як на весь контролер, так і на його методи. Якщо встановити «Authorize» на весь контролер, тоді цей атрибут буде застосовуватись до всіх методів цього класу.

В кожному контролері є методи, які являють собою кінцеві точки, до яких звертаються зовнішні компоненти системи. Для встановлення шляху до кінцевої точки, необхідно використати атрибут з відповідним типом запиту, тобто Get, Post, Put, Delete, Patch та інші RESTful запити, а також сам шлях. Відповідно, щоб звернутися до цього контролеру спочатку, необхідно звернутися до адреси контролера, тобто, наприклад, «api/auth», а потім поставити «/» та написати шлях до методу контролера, що в результаті буде мати наступний вигляд: «api/auth/register».

До створених контролерів належать:

– «AuthenticationController» – містить методи реєстрації та авторизації, генерує JWT токен для аутентифікації;

– «EmployeeController» – використовує всі методи «IEmployeeService», налаштовує типи запитів і шляхи;

– «InvitationController» – реалізує методи «IInvitationService», вказуючи типи запитів і шляхи;

– «InvitedEmployeeController» – використовує «IInvitedEmployeeService» і налаштовує кінцеві точки;

– «JobPositionsController» та «RolesController» – мають схожий функціонал, використовують відповідні сервіси;

– «QrCodeController» – обробляє та створює QR-коди, використовує «IQrCodeService»;

– «VisitRecordsController» – встановлює кінцеві точки для роботи з даними про відвідування, використовує «IVisitRecordService».

Після всіх цих налаштувань та реалізацій відповідних компонентів нашої API, запустивши виконуючий файл «Program.cs», завдяки документації «Swagger», можна побачити всі налаштовані кінцеві точки з їх шляхами, завдяки яким можна звертатися до методів контролера та отримувати або оброблювати певні дані. Зображення загального вигляду всіх кінцевих точок наведено на рисунках 3.2 та 3.3.

Рисунок 3.2 – Загальний вигляд першої частини всіх кінцевих точок

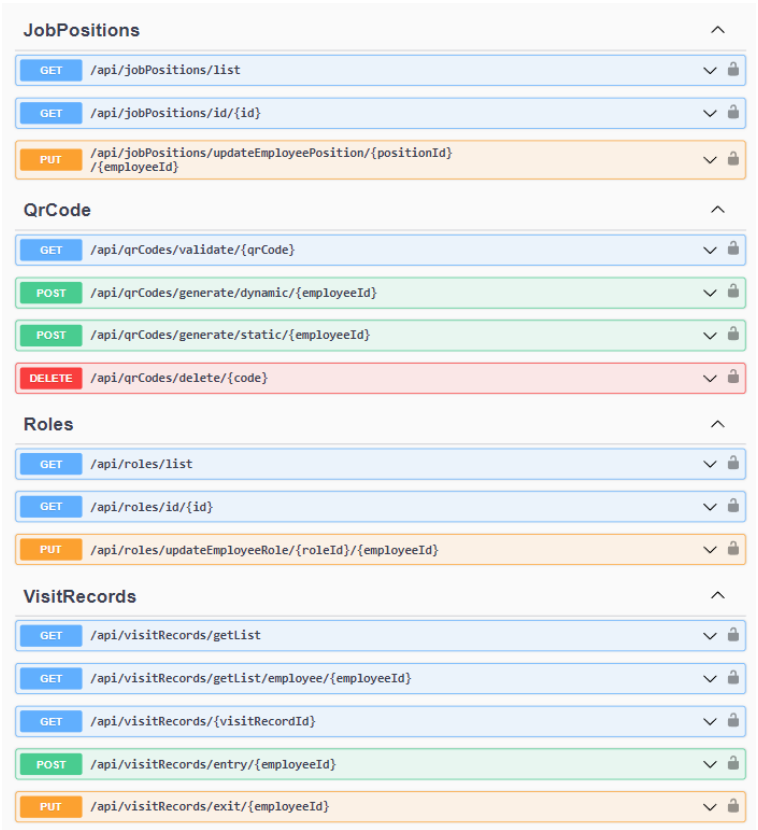

Рисунок 3.3 – Загальний вигляд другої частини всіх кінцевих точок

3.1.2 База даних

Для правильної роботи серверної частини, API повинне мати сховище, де буде зберігатися вся головна інформація щодо робітників та їх відвідувань. Для того, щоб пов’язати API та базу даних, було вирішено використати ORM «Entity Framework Core», який дозволяє, використовуючи методи, звертатися до БД та керувати даними.

Для того, щоб звертатися до БД, необхідно було створити сутності, які відповідають моделі таблиць бази даних. Так як раніше ці сутності було реалізовано (див. лістинги від А.2 до А.8 додатка А), їх необхідно налаштувати, створивши спеціальний клас, який буде наслідуватися від класу «DbContext». Реалізація цього класу наведено в додатку А лістингу А.18.

В класі «WorkQrDbContext» реалізовано властивості, які мають тип «DbSet<T>», що створюють таблиці, які будуть називатися так само як і назва властивості, та ці таблиці матимуть назви стовпців, що відповідають властивостям типу «T». Тобто, якщо у нас встановлено властивість «DbSet<EmployeeEntity> Employees», тоді назва таблиці в базі даних буде називатися Employees, та матиме стовпці, що відповідають властивостям класу «EmployeeEntity».

Використовуючи метод «OnModelCreating», можна, при створенні бази даних вперше, реалізувати якусь логіку, а саме, в даному випадку генеруються стандартні ролі та позиції для системи.

Для створення БД було використано міграції, які дозволяють створити базу даних та зберегти в певній таблиці міграцій версію БД, таким чином можна переключатися між міграціями, змінюючи структуру БД та дані, які відносяться до певних таблиць, якщо в цьому є необхідність.

Після використання міграцій, було створено БД з таблицями, структуру та зв’язки яких наведено на рисунку 3.4.

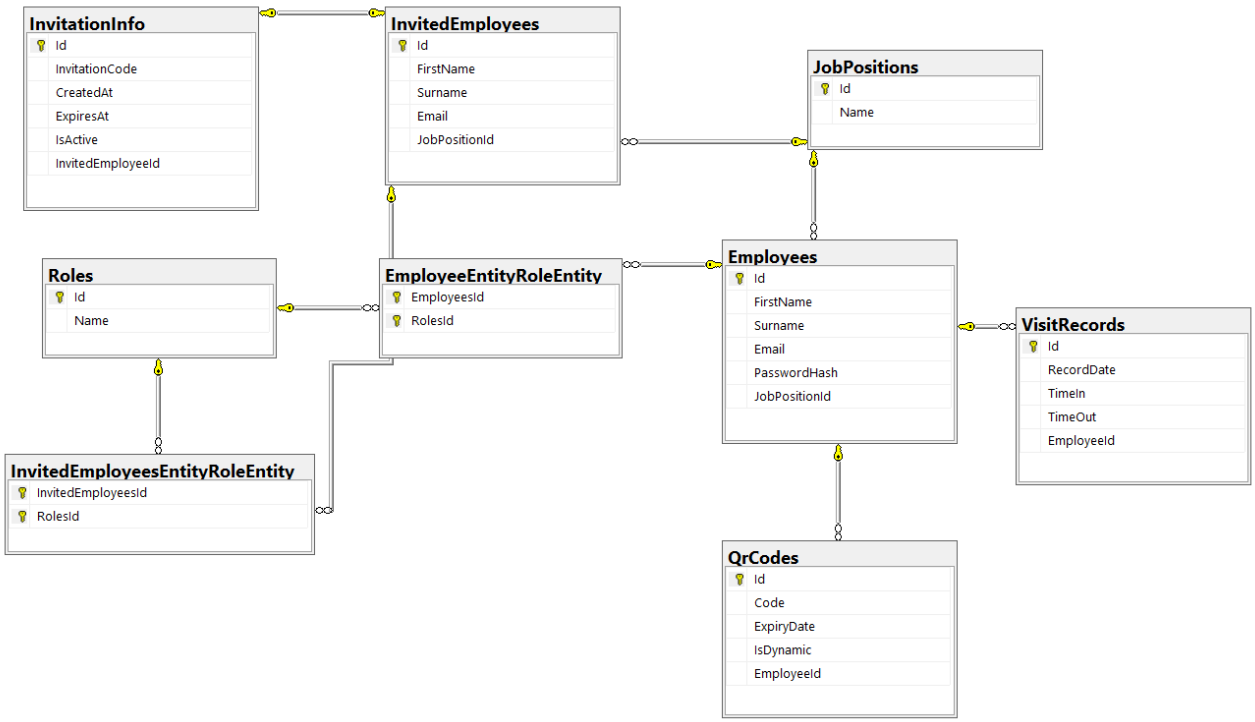

Рисунок 3.4 – Зображення структури БД у вигляді діаграми

Відповідно до рисунка 3.4, можна побачити дані, які зберігаються в таблицях та зв’язки між таблицями. Згідно діаграми можна побачити наступні зв’язки:

– відповідно до таблиці «Employees» один співробітник може мати багато ролей, багато записів про відвідування, багато QR-кодів та кожен співробітник має одну посаду;

– з таблиці «InvitationInfo» видно, що одне запрошення пов’язане з одним запрошеним робітником;

– згідно таблиці «InvitedEmployees» маємо, що один запрошений робітник може мати одне запрошення, може мати багато ролей, має одну посаду;

– відповідно до таблиць «JobPositions» та «Roles» бачимо, що одна посада та роль може мати багато співробітників та запрошених користувачів;

– таблиця «QrCodes» має одну залежність, відповідно один QR-код може належати тільки одному співробітнику;

– відповідно до «VisitRecords», один запис відвідування належить одному співробітнику.

Також, згідно рисунка 3.4, видно, що є додаткові таблиці, які створюються коли є зв’язок «багато до багатьох».

3.1.3 Розробка веб-інтерфейсу

Головною метою створення веб-інтерфейсу є забезпечення інтуїтивно зрозумілого, зручного та безпечного способу взаємодії користувачів з системою. В системі реалізовано веб-інтерфейс для двох типів користувачів – для робітників та адміністраторів. Сторінка адміністратора повинна мати можливості для управління користувачами та мати інформацію щодо їх обліку робочого часу. Сторінка для звичайного робітника представляє собою сторінку, в якій розміщуються базові дані про робітника та його відвідування, а також, як і на сторінці адміністратора, повинна бути можливість згенерувати QR-код.

Для того, щоб мати можливість створювати сторінки, спочатку необхідно провести базові налаштування для технології Blazor. По-перше необхідно налаштувати аутентифікацію системи, для цього необхідно у головному виконуючому файлі додати сервіс «AddAuthorizationCore», який дозволяє використовувати атрибут «Authorize» на сторінках, да під час «роутингу» перевіряти стан аутентифікації та авторизації.

Далі, у головному файлі розмітки «App.razor», необхідно додати код, лістинг якого наведено в лістингу Б.1 додатка Б. В цій розмітці є важливий тег «CascadingAuthenticationState», який впроваджує авторизацію у свої дочірні теги та контент, таким чином, звичайний «RouteView» замінюється на «AuthorizeRouteView», що дозволяє використовувати навігацію разом з перевіркою аутентифікації.

Наступний важливим кроком для роботи веб-інтерфейсу нашої системи є створення сервісів та використання сервісних підходів для зв’язку з раніше створеним API. Ці сервіси відповідають контролерам нашого API, та мають ті ж самі методи, що й контролери. В цих методах йде звертання до кінцевих точок, що налаштовані в контролерах, для того, щоб отримувати, редагувати, видаляти оброблювати інформацію, що надходить з API. Прикладом одного із сервісів є

«VisitRecords», в якому є інтерфейс та клас, що реалізує цей інтерфейс, так само, як і в API. Лістинг коду сервісу наведено в лістингу Б.2 додатка Б.

Відповідно до лістингу Б.2 додатка Б, видно, що методи звертаються до API, використовуючи «HttpClient», який містить в собі базову адресу API, а також в конкретних методах, в залежності від типу звертання, використовується адреса до кінцевих точок API, наприклад, «GetAsync("api/visitRecords/getList")». Також виходячи з потреби, було створено два «HttpClient», перший використовується для звичайних запитів, а інший – «HttpAuthorizedClient», використовується для запитів, в яких необхідно передавати статус авторизації для кінцевих точок API, які потребують авторизації.

Після створення всіх інших сервісів, вони так само як і в API були зареєстровані в DI, для того, щоб можна було їх використовувати.

В результаті було налаштовано середовище для створення сторінок. Перед початком створення сторінок було прийнято рішення назвати веб-інтерфейс «WorkQR».

Для початку було створено сторінку привітання «WelcomePage», яку зображено на рисунку 3.5.

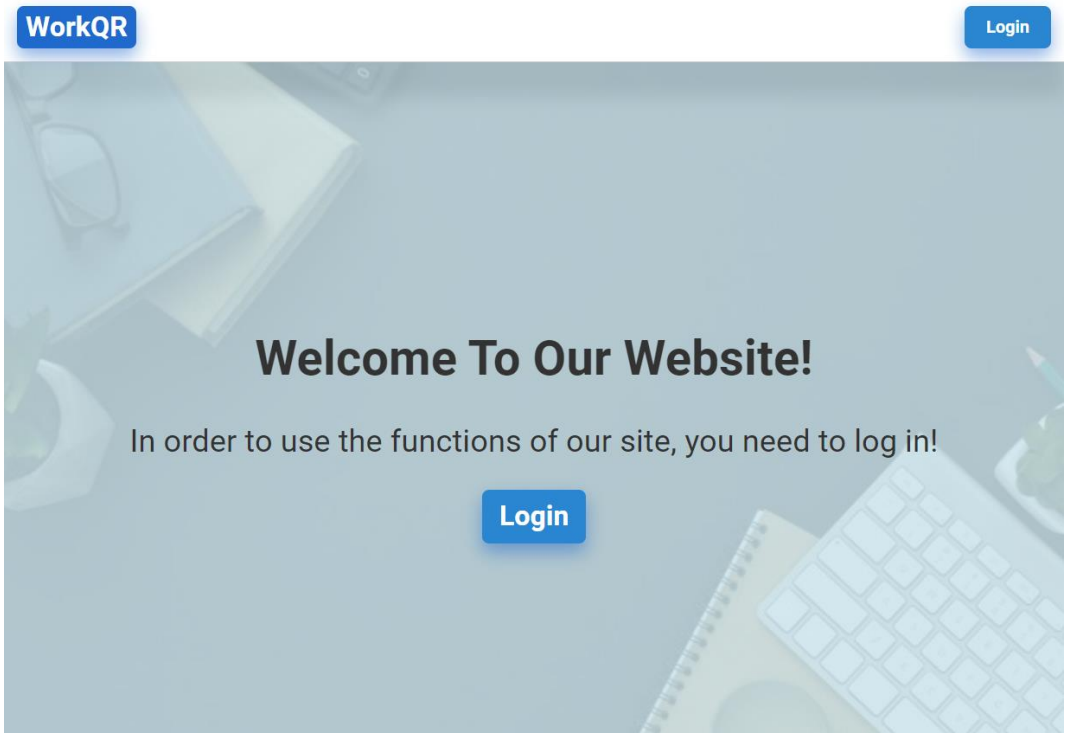

Рисунок 3.5 – Вигляд привітальної сторінки «WelcomePage»

Ця сторінка була створена, як початкове вікно при першому відвідуванні веб-інтерфейсу.

Наступною сторінкою було реалізовано сторінку «LoginPage», що містить поля для вводу логіну у вигляді пошти та паролю. Ці поля валідуються відповідно до правил встановлених системою, а також, при натисненні на кнопку «Login», надсилається запит до API, де перевіряються дані для входу в систему. Якщо дані валідні, то на веб-інтерфейс повертається рядок із «JWT-токеном», який встановлюється в локальному сховищі та, в подальших запитах до серверу, використовується для авторизації. Якщо дані не валідні, то веб-інтерфейс видає повідомлення про невірно введені дані. Після все ж таки вдалої авторизації, користувача переадресовує на сторінку, яка перевіряє роль користувача та перенаправляє на відповідно сторінку. Отже, якщо користувач має роль адміністратора, його переадресовує на сторінку адміністрування, якщо роль працівника – на сторінку працівника. Загальний вигляд сторінки «LoginPage» наведено на рисунку 3.6.

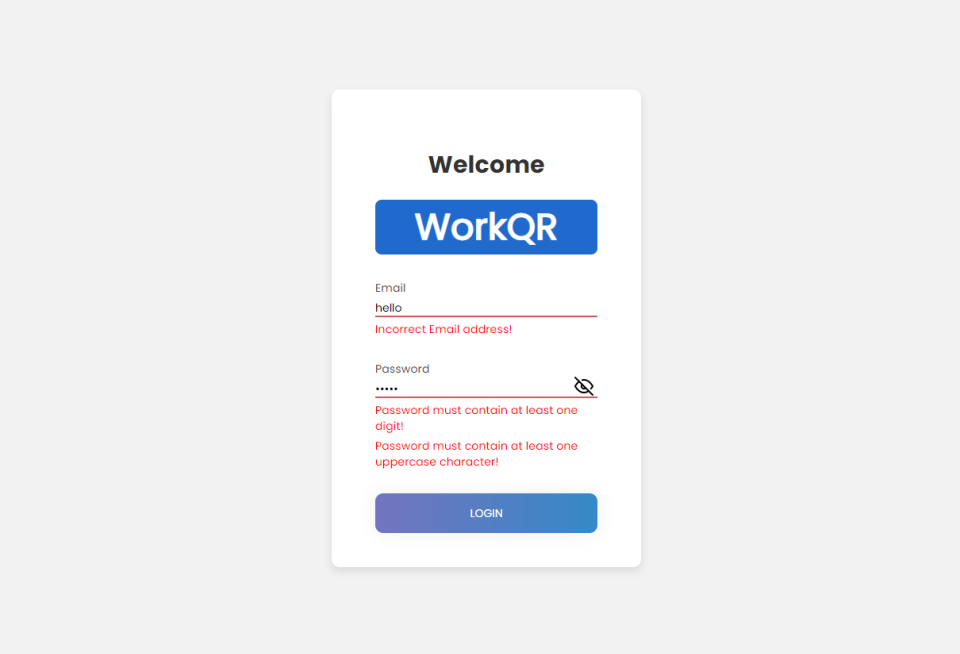

Рисунок 3.6 – Вигляд сторінки «LoginPage»

Першу розглянемо сторінку адміністрування, яка дозволяє керувати системою, використовуючи функції адміністратора, що наведена на рисунку 3.7.

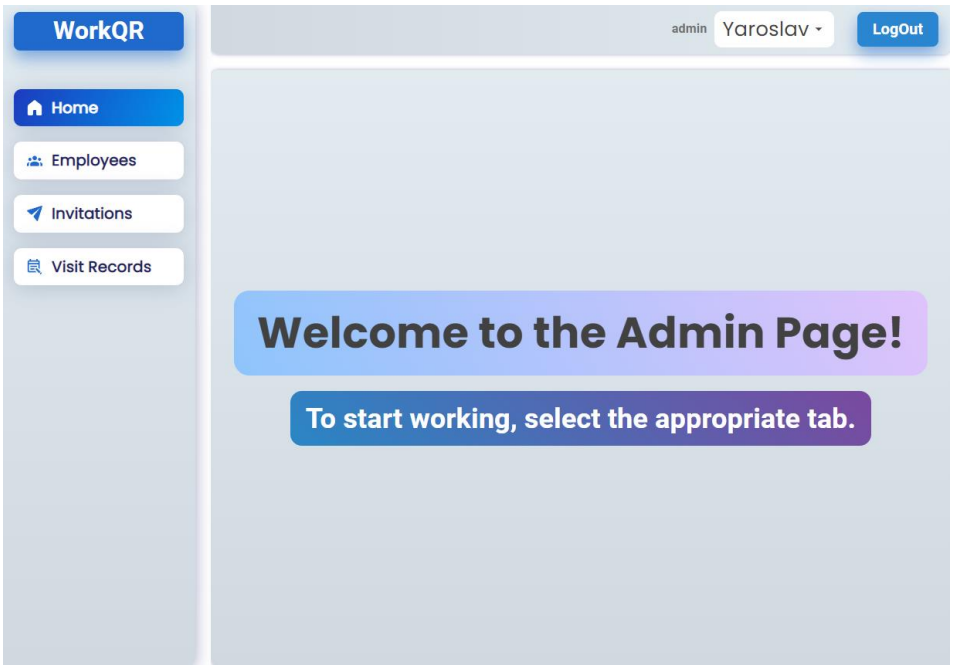

Рисунок 3.7 – Початкова сторінка адміністратора

При переході на сторінку адміністрування, нас зустрічає інтерфейс який візуально розділений на три частини:

– ліва частина або «sidebar» представляє собою список можливих сторінок, які мають певний функціонал;

– верхня частина або «header» відображає роль користувача, його ім’я, яке є кнопкою, а також кнопка для виходу із системи;

– головна частина або «main» відображає головний контент вибраної вкладки «sidebar».

При першому потраплянні на сторінку адміністратора, відразу відображається сторінка «HomePage», яка містить текст з привітанням та невеличкою вказівкою, щодо роботи.

При переході на сторінку «EmployeePage» (рис. 3.8), можна побачити список користувачів у вигляді таблиці, яка має заголовок «Employees», кнопку для оновлення таблиці, рядок з пошуком любих даних з таблиці, а також наступні колонки:

– «First Name» відображає ім’я робітника;

– «Surname» відображає прізвище користувача;

– «Email» відображає пошту користувача;

– «Role» та «Position» відповідно відображають роль та позицію робітника на підприємстві;

– «Status» дозволяє побачити активність користувача на даних момент, тобто присутній робітник чи ні. Відповідно загорається зелений чи червоний круг;

– «Actions» представляє собою стовбець, який містить дії для адміністратора, а саме можливість відредагувати дані користувача, видалити користувача та переглянути його дані та інформацію щодо обліку робочого часу. При натисканні на редагування чи видалення з’являються відповідні діалогові вікна, що наведені на рисунку 3.9.

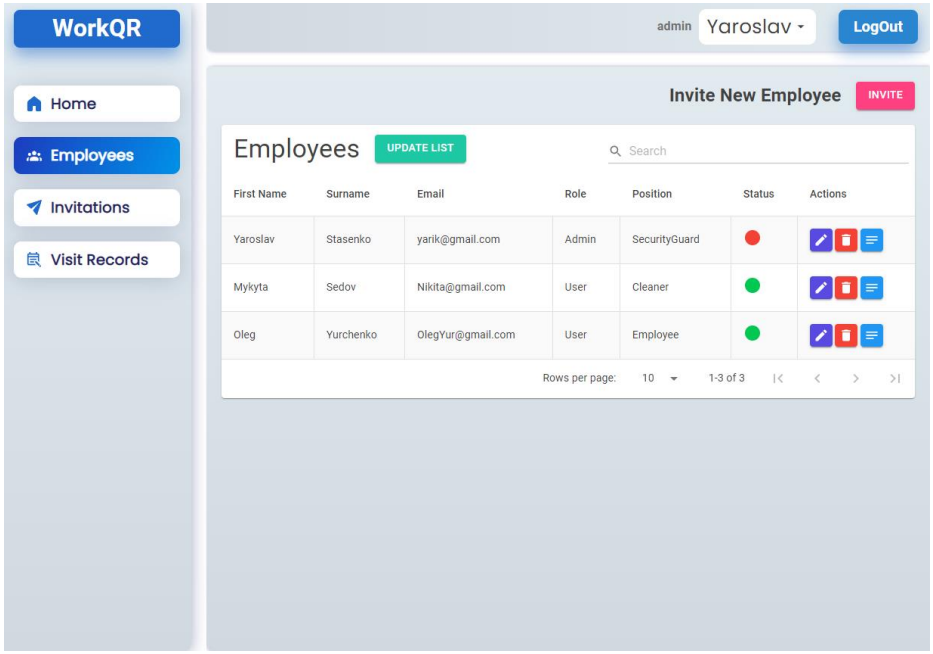

Рисунок 3.8 – Загальний вигляд сторінки при виборі вкладки «Employee»

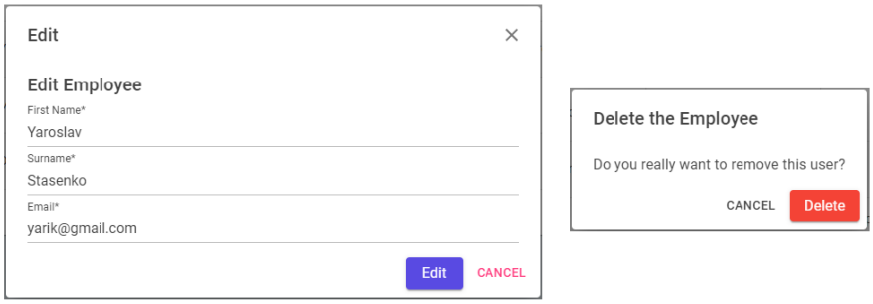

Рисунок 3.9 – Зображення діалогових вікон функцій редагування та видалення

Перейшовши на сторінку «EmployeeStatisticsPage», можна побачити всю головну інформацію про користувача, а саме скільки годин та хвилин відпрацював цей користувач в цьому місяці, в цьому році та в обраному місяці, а також можна побачити його імя та прізвище, пошту та позицію користувача. Нижче відображається таблиця, яка демонструє всі відвідування користувача. Таблицю можна оновлювати завдяки кнопці «Update List», фільтрувати за датою та виконати пошук. Також в таблиці відображаються дані, щодо дати та часу запису, дати та час входу, виходу, а також скільки працівник відпрацював за певний проміжок між входом та виходом. Вигляд сторінки, яка містить головну інформацію про працівника наведено на рисунку 3.10.

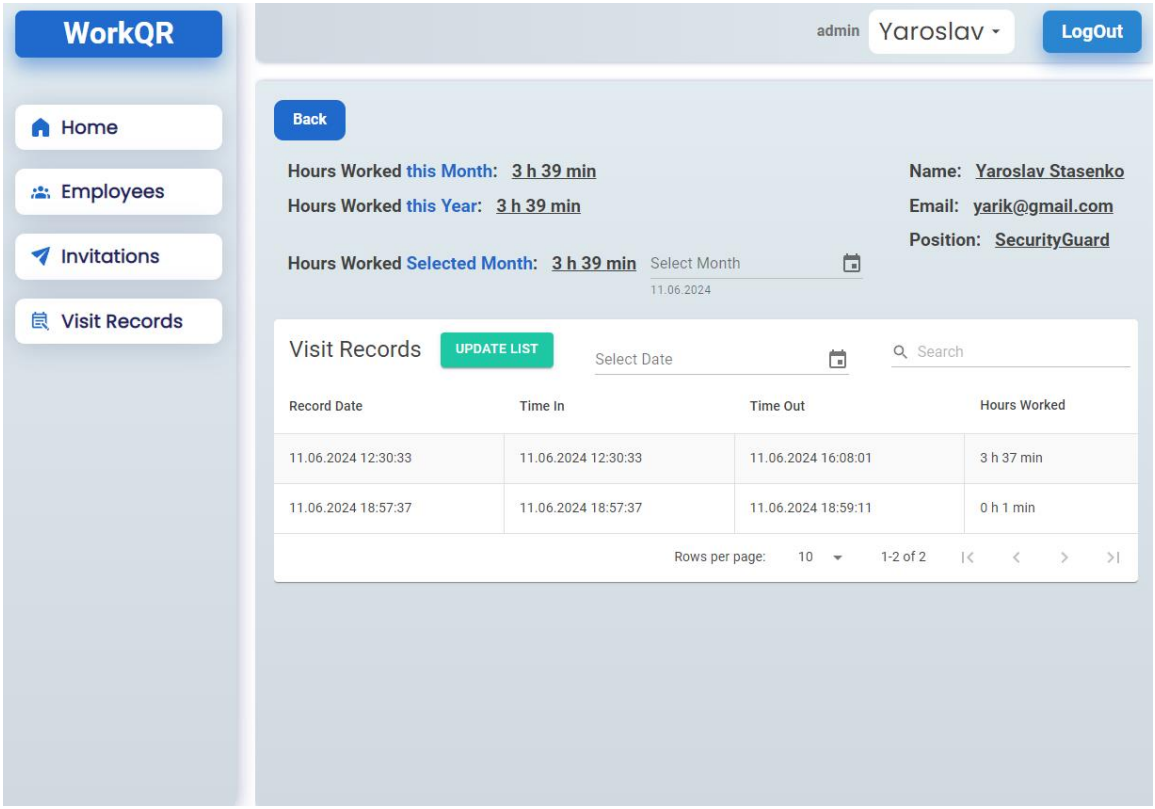

Рисунок 3.10 – Загальний вигляд сторінки з інформаціє про працівника та його даними щодо обліку робочого часу

Сторінка «Employees» має кнопку «Invite», яка переадресовує на сторінку із запрошенням нового користувача. Сторінка із запрошенням нового користувача містить форму, в якій необхідно вести ім’я, прізвище, пошту та обрати позицію нового робітника та його роль в системі. Всі перераховані поля

валідуються відповідно до політик системи. Після натиснення на кнопку «Invite», йде звертання до API і, якщо дані зі сторони API також валідні, тоді за вказаною електронною поштою відправляється повідомлення про запрошення, яке містить посилання для завершення реєстрації. Вигляд сторінки з формою для запрошення, а також повідомлення, яке отримую запрошений користувач наведено на рисунках 3.11 та 3.12 відповідно.

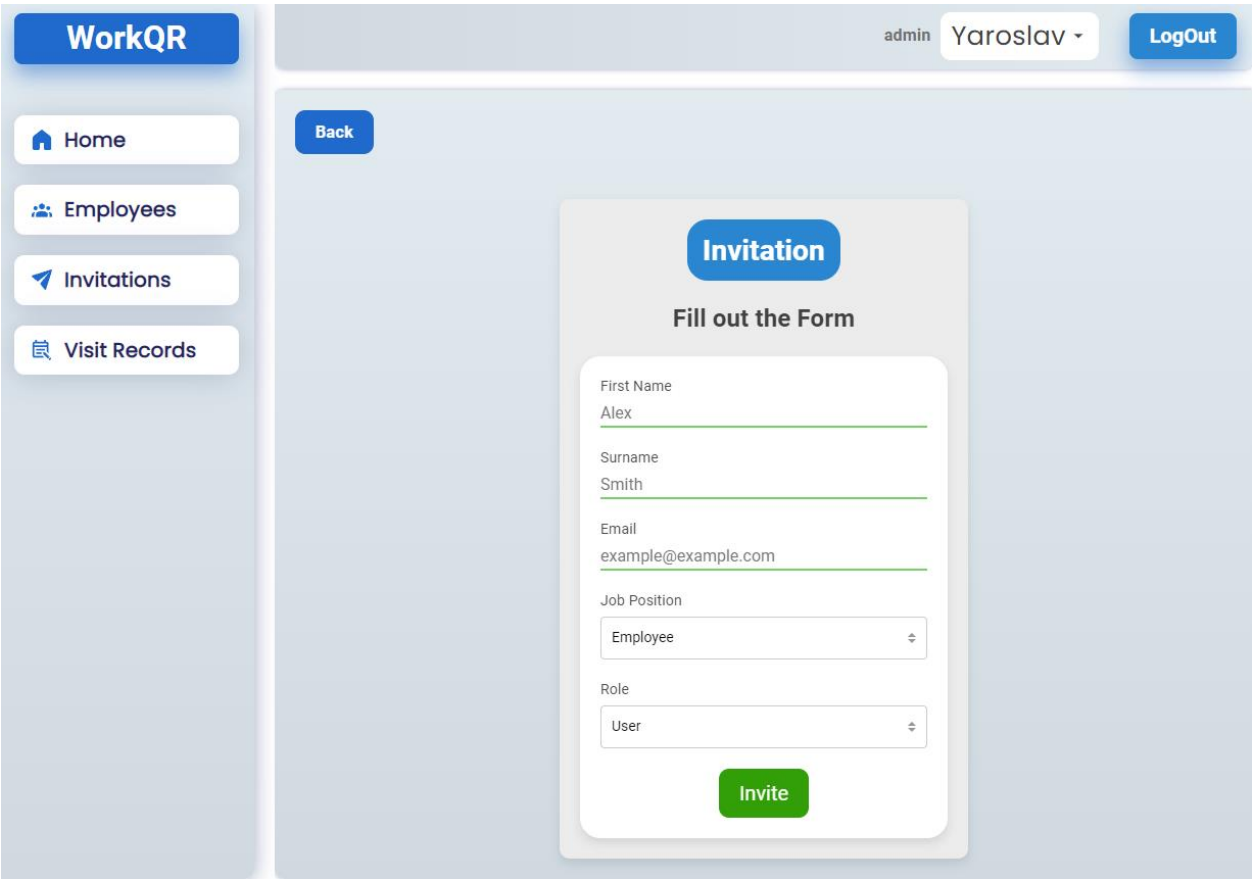

Рисунок 3.11 – Загальний вигляд сторінки для запрошення нового працівника

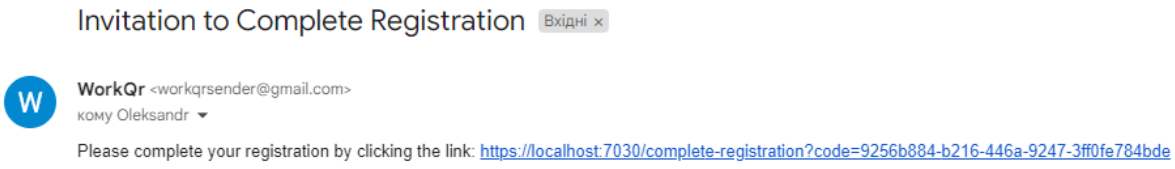

Рисунок 3.12 – Приклад повідомлення-запрошення на пошті для завершення реєстрації в системі

Після того, як запрошений користувач перейде по посиланню, відбудеться перевірка посилання на валідність та дійсність. Якщо посилання не валідне, користувачеві відобразиться повідомлення про невірне посилання або недійсність запрошення, якщо ж посилання є валідним, користувача переадресує на сторінку із завершенням реєстрації в системі, що зображено на рисунку 3.13.

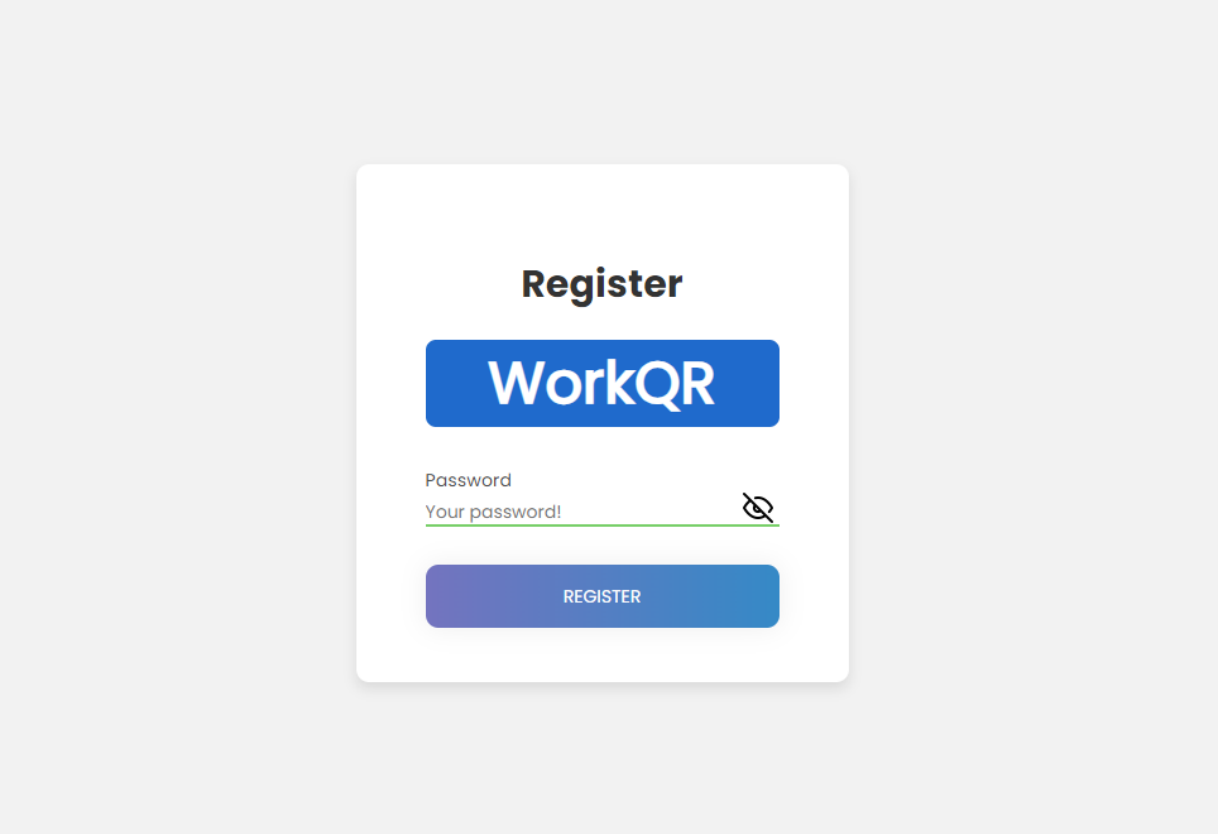

Рисунок 3.13 – Сторінка із завершенням реєстрації

На наступній сторінці «Invitations» можна переглянути список всіх запрошених користувачів, що наведені в таблиці. Таблицю можна оновити, та відсортувати за колонками. В таблицю входять ім’я запрошеного робітника, пошта, роль, позиція, дата створення запрошення, дата до якої дійсне запрошення, стан запрошення (активне чи ні) та можливість видалити запрошення. Вигляд загальної сторінки із запрошенням наведено на рисунку 3.14.

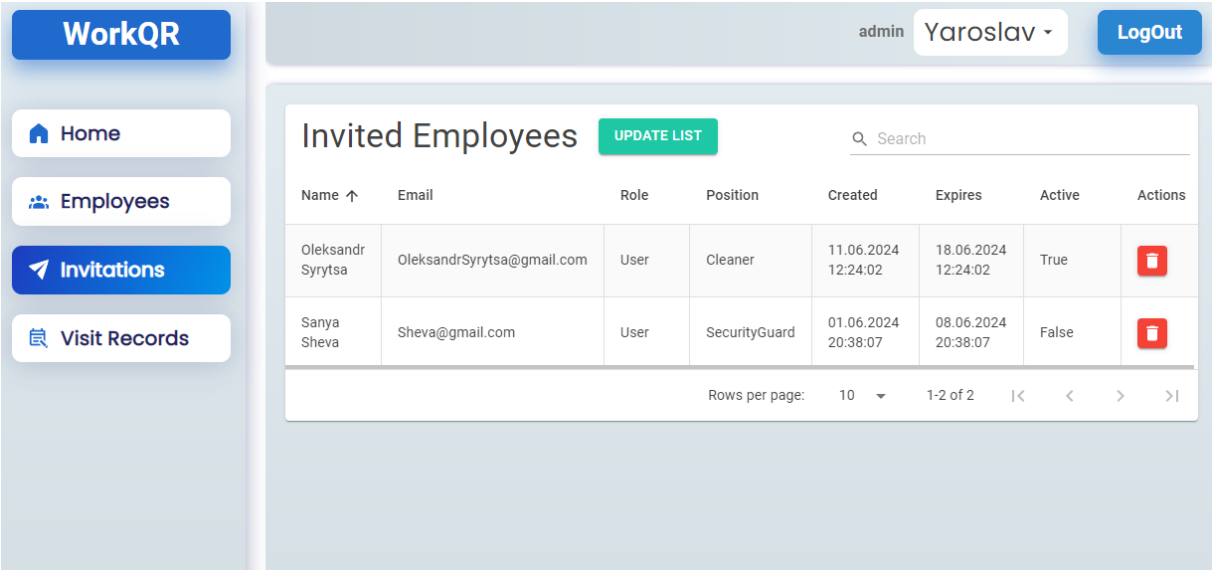

Рисунок 3.14 – Вигляд сторінки із списком запрошених користувачів

Остання сторінка «Visit Records» (див. рис. 3.15) містить інформацію у вигляді таблиці щодо відвідувань всіх користувачів системи. Таблиця має інформацію, щодо імені працівника, який скористався системою, позиції, дати та часу відвідування, дати та часу входу та виходу.

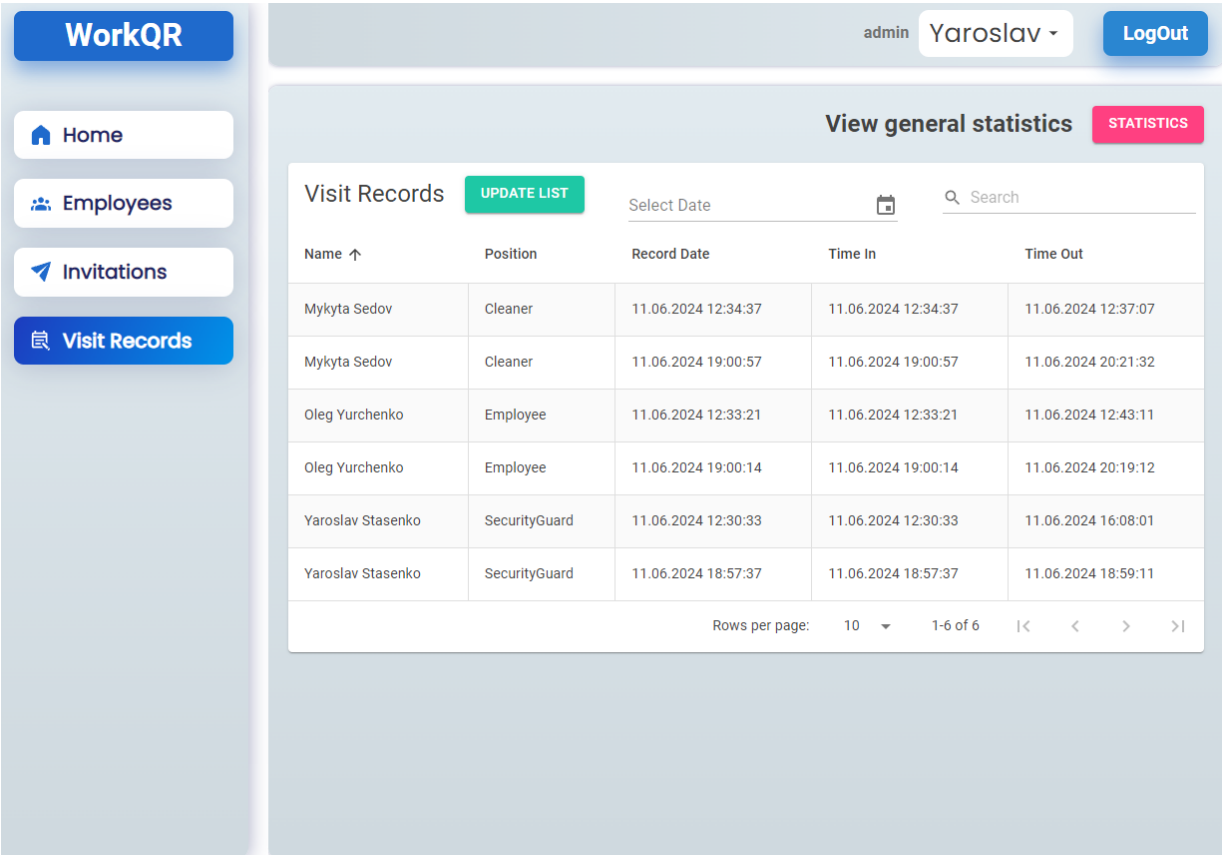

Рисунок 3.15 – Сторінка відвідувань користувачів

Також сторінка «Visit Records» має кнопку «Statistics», при натисненні на яку відбувається перенаправлення на сторінку, на якій наведена загальна статистика щодо відвідувань у вигляді графіків та діаграм. Ця сторінка була розроблена як приклад на використання даних про відвідування у якості обліку робочого часу. В даному випадку було відтворено лише три графіки, але цей функціонал можна розширювати та додавати нові графіки, діаграми або інші дані, що стосуються обліку робочого часу. Перший графік демонструє загальне значення відпрацьованих годин за конкретний місяць. Кругова діаграма відображає по кольорам найвідвідуваніші дні тижня. Остання діаграма показує, яка професія відпрацювала найбільшу кількість годин за весь час. Також сторінка має кнопку «Refresh Charts» для оновлення графіків. Сторінка з прикладом графіків та діаграм наведена на рисунку 3.16.

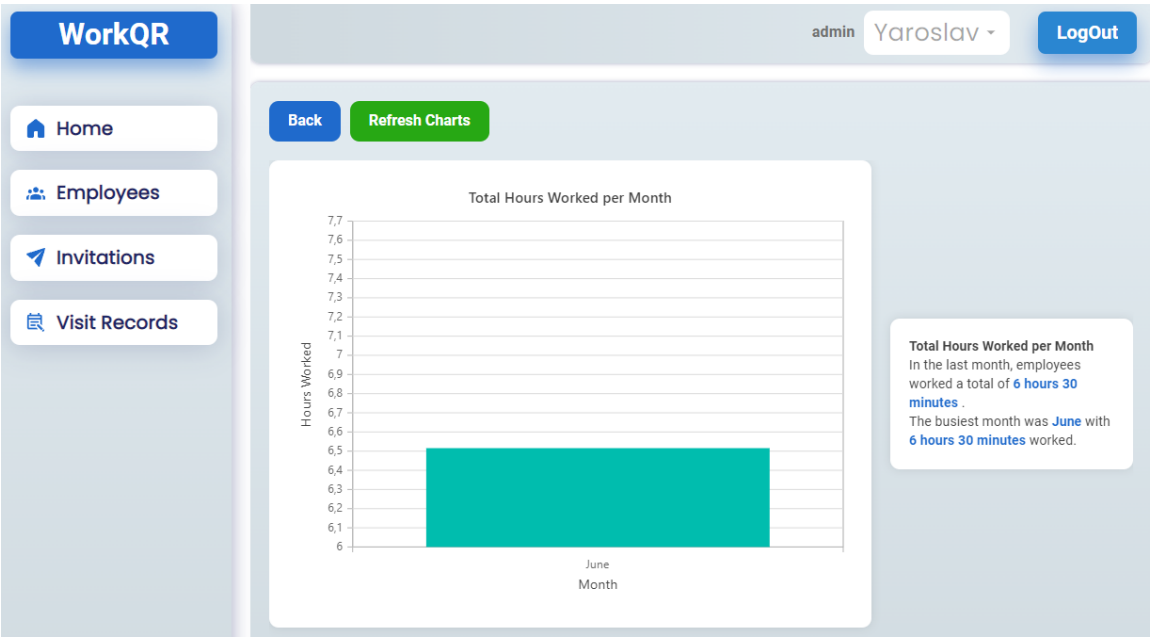

Рисунок 3.16 – Сторінка із загальною статистикою щодо обліка робочого часу у вигляді графіків

При натисненні на ім’я користувача, що знаходиться зверху, відкривається спливаюче вікно, яке містить кнопку «Change Password», що дозволяє користувачеві змінити пароль, перейшовши на відповідну вкладку веб-інтерфейсу. Також вікно містить іншу кнопку – «Generate QR Code», натиснувши

на яку, можна перейти на вкладку, де можливо буде згенерувати QR-код. Зображення спливаючого вікна наведено на рисунку 3.17.

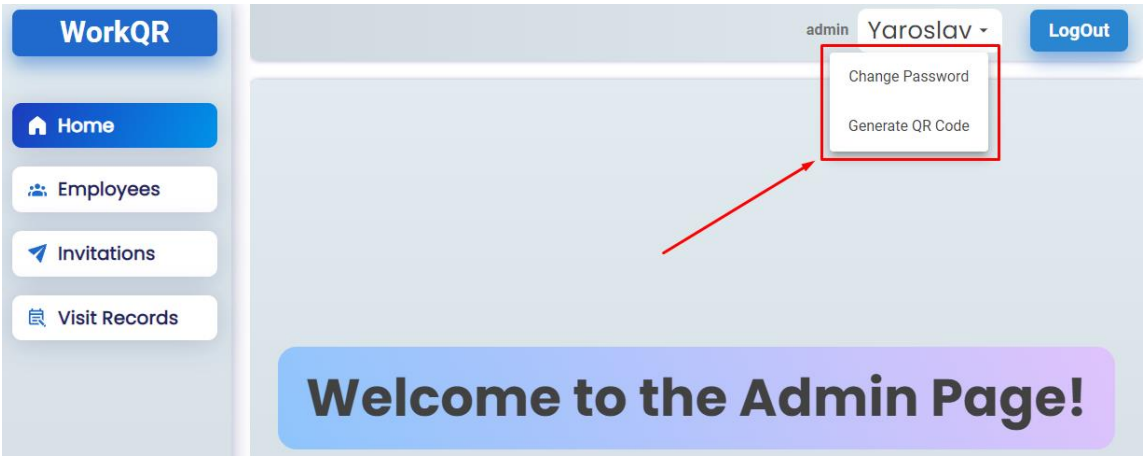

Рисунок 3.17 – Зображення спливаючого вікна з кнопками

Після переходу на сторінку «GenerateQrCodePage», можна побачити вікно, в якому є кнопка «Long-term QRCode» та «Short-term QrCode». При натисненні на першу кнопку виникає діалогове вікно з попередженням про небезпеку використання довготривалих кодів та кнопкою генерації кода та кнопкою відміни дії. При натисненні на іншу кнопку, також виникає діалогове вікно, але з інформацією про те, що цей тип коду працює не довго.

Після підтвердження генерації будь-якого типу QR-коду, веб-інтерфейс відправляє запит до API, який в свою чергу генерує у відповідь зображення QR-коду у вигляді байтів та надсилає це до веб-інтерфейсу. Після отримання зображення від API, веб-інтерфейс виводить на екран зображення та відображає кнопку, яка дозволяє завантажити QR-код на пристрій.

Також на сторінці міститься кнопка «Back», яка повертає на сторінку, яка ініціювала перехід на сторінку із генерацією QR-кода.

Загальний вигляд сторінки із згенерованим QR-кодом, а також із діалоговими вікнами при натисненні на відповідну кнопку, наведено на рисунках 3.18 та 3.19 відповідно.

Рисунок 3.18 – Зображення вкладки з можливістю згенерувати QR-код

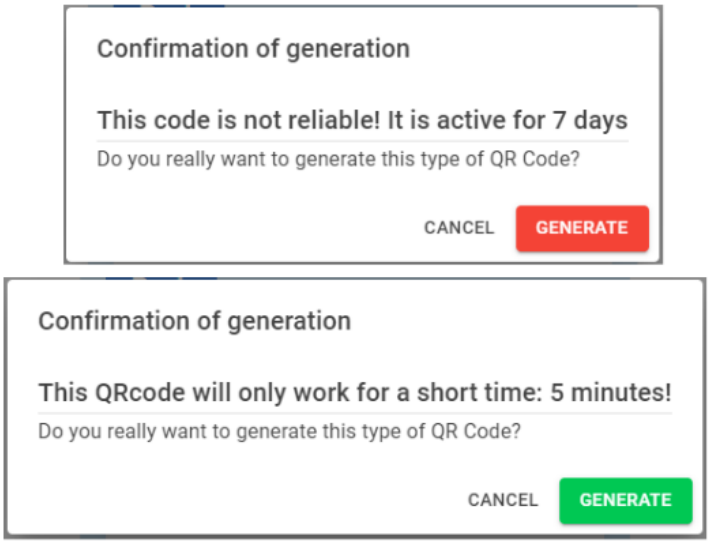

Рисунок 3.19 – Зображення діалогових вікон відповідно до кнопок генерації QR-кодів вибраного типу

3.2 Симуляція апаратної частини

У рамках розробки комплексної системи контролю доступу та обліку робочого часу, значну роль відіграє точне тестування та валідація апаратних компонентів. Оскільки пряме впровадження та тестування реальних апаратних

рішень може бути часомістким та вимагати значних ресурсів, симуляція апаратної частини стає важливою складовою процесу розробки.

Як було описано раніше, для симуляції було використано технологію WPF, для створення десктоп-застосунку на Windows. При написанні застосунку використовувався шаблон проєктування «MVVM», який дозволяє чітко розділили частини розробки на «Model», «View» та «ViewModel», що в свою чергу дозволяє писати чистий код то робити якісну розмітку XAML.

Для того, щоб реалізувати у застосунку симуляцію апаратної частини, спочатку необхідно створити алгоритм за допомогою якого буде працювати апаратна частина. Реалізований алгоритм роботи апаратної частини у вигляді блок-схем наведено на рисунку 3.20.

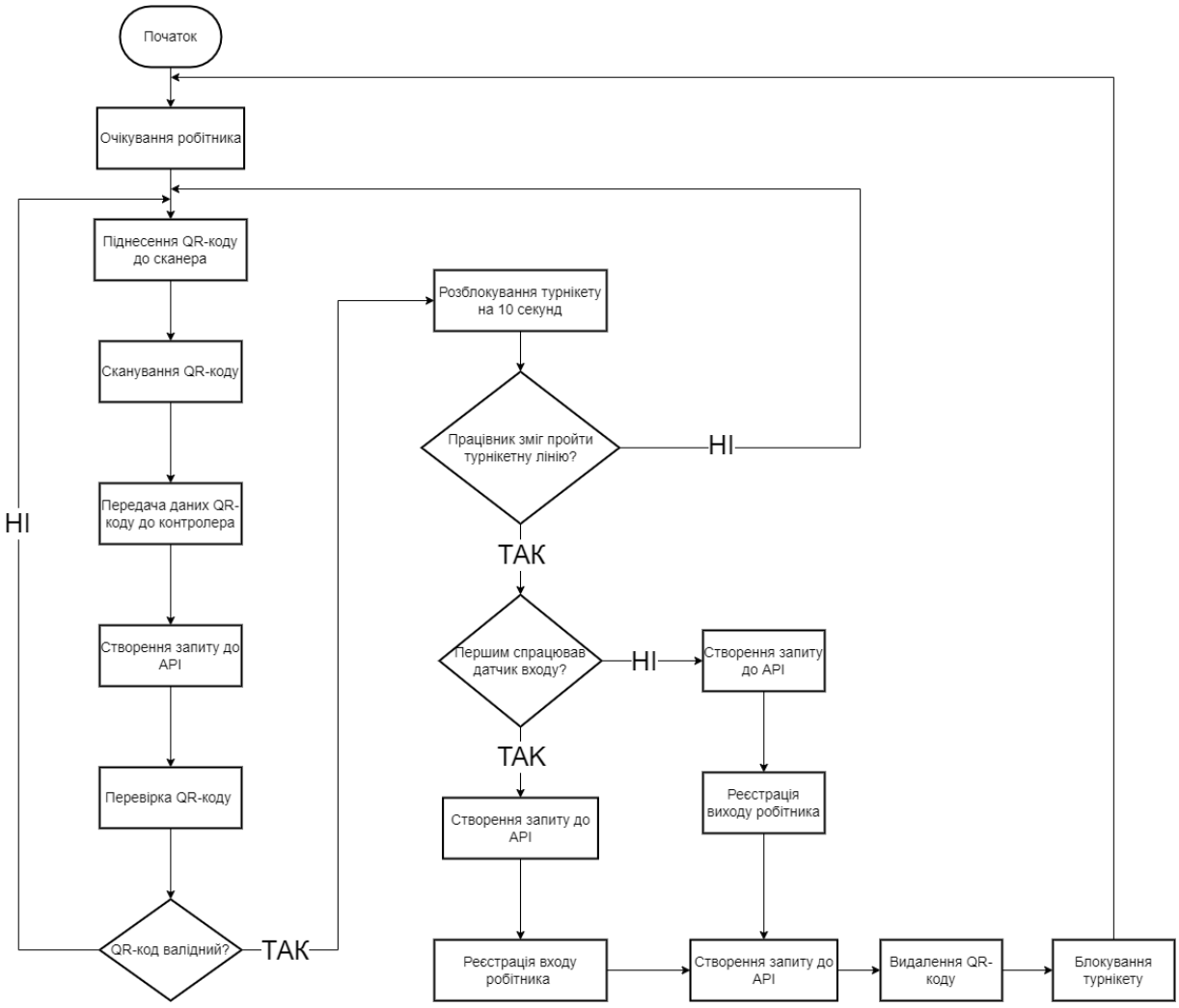

Рисунок 3.20 – Алгоритм роботи апаратної системи

Відповідно до розробленого алгоритму, апаратна система очікує на робітника. Працівник повинен піднести свій QR-код та відсканувати його, використовуючи QR-сканер. Після цього інформація щодо коду передається до контролера, де відправляється запит до API, завдяки якому відбувається перевірка валідності QR-коду. Пройшовши перевірку валідності, якщо код був неправильним, потрібно використати інший QR-код, якщо правильний, тоді турнікет розблоковується на 10 секунд, що дає змогу робітнику пройти через турнікетну лінію. Якщо працівник не зміг пройти, за будь-яких причин, йому необхідно знову піднести код до сканеру, якщо йому вдалося повністю пройти, вирішується який датчик спрацював першим. Якщо було задіяно першим датчик входу, тоді після повного проходження турнікетної лінії, створиться запит до API, який зареєструє вхід робітника, після чого знову створиться запит, який видалить QR-код та турнікет буде заблоковано. Якщо було задіяно першим другий датчик, тобто датчик виходу, тоді відбудуться ті ж самі дії, що і з датчиком входу, тільки створиться запит до API щодо реєстрації виходу робітника з підприємства. Після цих всіх дій, апаратна система буде очікувати наступного користувача.

Відповідно до цієї логіки було реалізовано функціональність, використовуючи WPF. Вигляд створеного десктоп-застосунку наведено на рисунку 3.21.

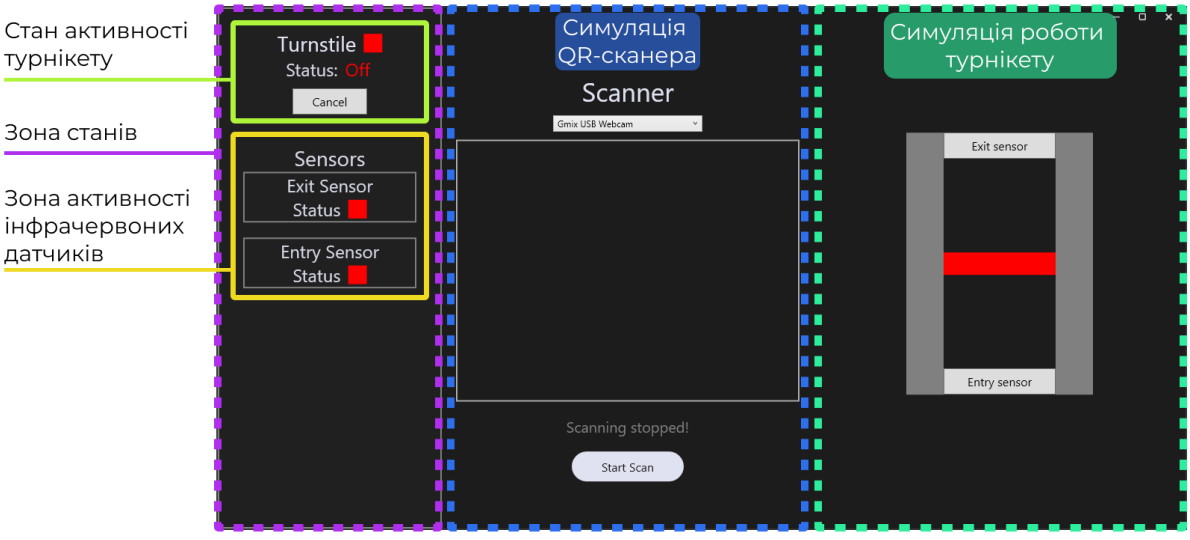

Рисунок 3.21 – Зображення створеного застосунку разом із описом його частин

Виходячи з рисунку 3.21, можна відмітити, що візуально застосунок поділяється на три частини. Перша, сама ліва частина, яка обведена фіолетовим кольором, відповідає за відображення інформації щодо стану частин апаратної частини. До синьої зони відноситься реалізація QR-код сканера, який працює через камеру. Зелена зона симулює роботу турнікета та інфрачервоних датчиків відстані. Остання зона має дві кнопки, які при натисканні будуть симулювати проходження людини через відповідні датчики входу та виходу, а також сам турнікет, який на зображенні наведено червоним кольором.

При натисненні на кнопку «Start Scan» вмикається камера, яка дозволяє симулювати роботу сканера QR-кодів, після цього необхідно піднести QR-код. Якщо код є валідним, у зоні стану активності турнікету квадрат загориться зеленим кольором, що буде свідчити про те, що турнікет розблоковано. Також турнікет загориться зеленим і у зоні з симуляцією роботи турнікету та інфрачервоних датчиків. Після цього необхідно провести симуляцію роботи інфрачервоних датчиків, тобто створити видимість проходу людини через турнікет. Вигляд програми, під час симуляції роботи апаратних компонентів, наведено на рисунку 3.22.

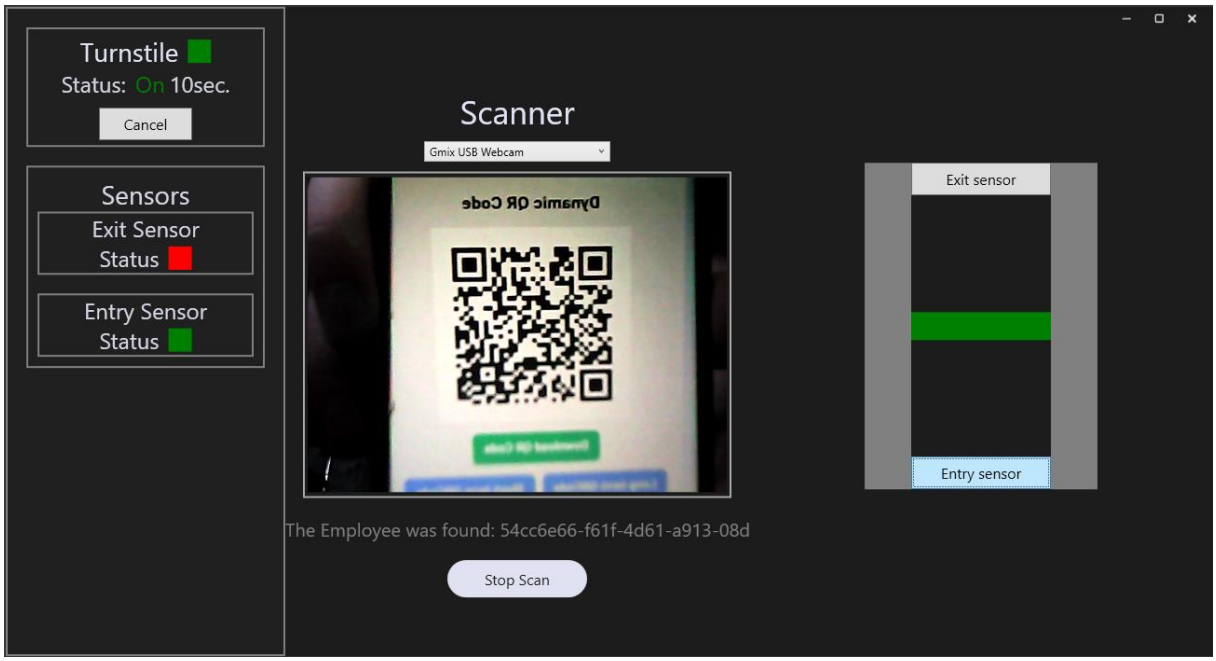

Рисунок 2.22 – Вигляд програми під час симуляції роботи апаратної частини

Така реалізація дозволяє цілком провести симуляцію роботи реальних апаратних компонентів, перевірити логіку роботи системи та перевірити систему на можливі помилки в алгоритмах до їх реалізації в апаратному забезпеченні.

3.3 Порівняльний аналіз із аналогами

3.3.1 Порівняння із системою «Genea»

Розроблена система контролю доступу та обліку робочого часу використовує QR-коди, API для обробки запитів та верифікації, веб-інтерфейс для управління і моніторингу, а також апаратну частину, що складається з турнікетів, сканера QR-кодів та інфрачервоних датчиків. Це дозволяє зручно управляти доступом і обліком часу в невеликих і середніх підприємствах, забезпечуючи економічне і ефективне рішення.

Система Genea також використовує QR-коди, але має ширші можливості і складніші технології, орієнтовані на великі корпорації. Genea інтегрує хмарні технології, що дозволяє централізовано управляти доступом та безпекою через глобальну мережу. Це забезпечує високу масштабованість і можливість керування доступом з будь-якої точки світу. Genea також використовує передові технології шифрування для захисту даних, забезпечуючи високий рівень безпеки, що є критично важливим для великих підприємств. Додатково, система Genea може інтегруватися з мобільними додатками, що дозволяє користувачам управляти доступом через свої смартфони.

Таким чином, розроблена система є більш простою і доступною для малих та середніх підприємств, тоді як Genea пропонує більш високий рівень безпеки і інтеграції, що підходить для великих організацій. Вибір між цими системами залежить від розміру компанії та специфічних потреб у контролі доступу та обліку робочого часу.

3.3.2 Порівняння з системою «Doordeck»

Doordeck використовує NFC для забезпечення безконтактного доступу. NFC-технологія дозволяє користувачам використовувати свої смартфони або інші NFC-пристрої для входу. Doordeck інтегрується з існуючими системами контролю доступу, не потребуючи додаткового апаратного забезпечення. Система забезпечує високий рівень безпеки завдяки військовому рівню шифрування і багатофакторній автентифікації, а також дозволяє керувати доступом і правами через хмарну інфраструктуру.

Розроблена система, яка використовує QR-коди для контролю доступу та обліку робочого часу, є економічною, легкою у впровадженні та інтеграції, що робить її ідеальною для малих та середніх підприємств. Вона забезпечує необхідний рівень безпеки та гнучкості за менших витрат на обладнання.

Система Doordeck, що базується на NFC, надає вищий рівень безпеки та зручності завдяки "tap-and-go" технології. Користувачі можуть використовувати свої смартфони для входу, що зменшує потребу у фізичних ключах або картках. Проте, вона вимагає більше ресурсів для інтеграції та є дорожчою у впровадженні.

Якщо порівнювати QR-коди з NFC, то QR-коди є економічними і простими у впровадженні, що робить їх доступними для малих і середніх підприємств. Вони не потребують дорогого обладнання і легко інтегруються з існуючими системами, забезпечуючи гнучкість і можливість швидкої налаштування під різні умови. Користувачі можуть сканувати QR-коди за допомогою камер смартфонів або спеціальних зчитувачів, що також додає зручності. Однак, QR-коди можуть бути менш безпечними, оскільки їх можна скопіювати або підробити, особливо якщо вони не змінюються динамічно.

Отже, розроблена система краща для економічного та простого впровадження, тоді як Doordeck забезпечує вищий рівень безпеки та зручності, але є більш дорогою.

4 РОЗРАХУНКОВИЙ АНАЛІЗ ПРОДУКТИВНОСТІ ТА СТІЙКОСТІ СИСТЕМИ

4.1 Оцінка продуктивності системи

Так як серверна частина є основною роботи мережевої системи контролю доступу та обліку робочого часу, то час відгуку цієї системи є важливим показником її продуктивності. Цей параметр визначає швидкість обробки запитів користувачів.

Для того, щоб оцінити продуктивність системи, було вирішено визначити середній час відгуку API, яке є основою розробленої системи. Для оцінки такої системи було написано невеличкий програмний код, який генерував запити до API, і відповідно враховував час відгуку API на запит. Процес відправлення запитів та отримання результату, у вигляді номеру запиту разом з часом відгуку, наведено на рисунку 4.1.

```
1 - Response time: 141 ms
2 - Response time: 2 ms
3 - Response time: 1 ms
4 - Response time: 1 ms
5 - Response time: 1 ms
6 - Response time: 0 ms
7 - Response time: 1 ms
8 - Response time: 1 ms
9 - Response time: 1 ms
10 - Response time: 1 ms
```

Рисунок 4.1 – Приклад тестування часу відгуку API на запити

Для розрахунку середнього часу відгуку системи було використано формулу (4.1):

$$T_{avg} = \frac{\sum_{i=1}^{N} T_i}{N}, \tag{4.1}$$

де T_{avg}– середній час відгуку API на запит;

N – кількість запитів;

T_i – час відгуку для і-го запиту.

Відповідно до формули (4.1) було розраховано час середнього відгуку системи:

$$T_{avg} = \frac{141 + 2 + 1 + 1 + 1 \ldots + 1 + 1}{100} = 2{,}21 \text{ мс.}$$

Відповідно до результатів середній час відгуку API на запити становить 2,21 мс, а також видно, що під час першого запиту триває найбільша затримка в 141 мс.

Виходячи з попередніх розрахунків, з’являється необхідні розрахувати кількість запитів, які система може обробити за одиницю часу. Для розрахунку пропускної здатності було використано формулу (4.2):

$$\text{Пропускна здатність} = \frac{N}{T}, \tag{4.2}$$

де N – кількість запитів;

T – час, протягом якого оброблялися ці запити.

Провівши тести системи було визначено що пропускна здібність API включає 493,31 запит на секунду, що являється дуже високим показником для роботи API.

Також для оцінки загальної продуктивності мережевих взаємодій та ефективності серверної частини можна розрахувати час до першого байту. Це є метрикою, яка вимірює проміжок часу між відправленням запиту клієнтом і отримання першого байту відповіді від сервера. Для того, щоб розрахувати час до першого байту необхідно скористатися формулою (4.3):

$$T_{TTFB} = T_{start} - T_{send}, \quad (4.3)$$

де T_{TTFB} – час до першого байту;

T_{start} – час отримання першого байту відповіді;

T_{send} – час відправлення запиту.

Згідно формули було отримано, що час до першого байту першого запиту становить 255,9386 мс. Для об'єктивності вимірювань необхідно розрахувати середнє значення часу до першого байту для декількох запитів. Для цього необхідно скористатися формулою (4.4).

$$T_{avgTTFB} = \frac{\sum_{i=1}^{N} T_{TTFB_i}}{N}, \quad (4.4)$$

де $T_{avgTTFB}$ – середній час відгуку API на запит;

N – кількість запитів;

T_{TTFB_i} – час відгуку для і-го запиту.

Відповідно було проведено 100 запитів та визначено середній час до першого байту, що складає 3,25 мс. Приклад результатів вимірювань наведено на рисунку 4.2.

```
Request 93: TTFB = 1,0428 ms
Request 94: TTFB = 0,9789 ms
Request 95: TTFB = 0,9041 ms
Request 96: TTFB = 1,2995 ms
Request 97: TTFB = 0,9422 ms
Request 98: TTFB = 0,838 ms
Request 99: TTFB = 0,8781 ms
Request 100: TTFB = 1,0257 ms
Average TTFB: 3,25 ms
```

Рисунок 4.2 – Приклад результатів розрахунку часу до першого байту відповідно до запиту

4.2 Аналіз стійкості апаратної частини системи

Для того, щоб провести аналіз стійкості апаратної частини системи, що складається з QR-сканера QR500, турнікета TS1000 Pro з його контролером, та двох інфрачервоних датчиків E18-D80NK, необхідно використати декілька критеріїв.

Спочатку необхідно визначити передаточні функції компонентів.

Передаточна функція QR-сканера QR500, за припущенням, що він має низьку інерційність, буде мати наступний вигляд:

$$G_q(s) = K_q = 1.$$

Відповідно, якщо коефіцієнт підсилення $K_q = 1$, що означає, що сканер має одиничне підсилення.

Контролер турнікета TS1000 Pro має PI-регулятор, який описується формулою (4.5):

$$G_c(s) = K_c\left(1 + \frac{1}{T_c s}\right), \tag{4.5}$$

де K_c – коефіцієнт підсилення пропорційної частини регулятора;

T_c – постійна часу інтегральної частини регулятора. Ці параметри визначають, як контролер реагує на зміни вхідного сигналу.

У випадку з цим контролером K_c = 2 та $T_c s$ = 1,5. Тоді передавальна функція контролера буде мати наступний вигляд:

$$G_c(s) = 2\left(1 + \frac{1}{1{,}5s}\right) = \frac{2(1{,}5s + 1)}{1{,}5s} = \frac{6s + 4}{3s}.$$

Передаточна функція турнікета TS1000 Pro з постійною часу $T_t = 2{,}5$ с має наступний вигляд:

$$G_t(s) = \frac{1}{2{,}5s + 1}.$$

Відповідно до характеристик інфрачервоних датчиків, їх передаточна функція наступна:

$$G_d(s) = 1.$$

Беручи до уваги всі попередні передаточні функції, можна визначити загальну передаточну функцію апаратної частини системи за формулою (4.6):

$$G(s) = G_q(s) \cdot G_c(s) \cdot G_t(s) \cdot G_d(s), \quad (4.6)$$

де $G_q(s)$ – передаточна функція QR-сканера;

$G_c(s)$ – передаточна функція контролеру турнікета;

$G_t(s)$ – передаточна функція турнікету;

$G_t(s)$ – передаточна функція інфрачервоних датчиків.

Виходячи з формули (4.6), отримаємо наступне рівняння:

$$G(s) = 1 \cdot \frac{6s + 4}{3s} \cdot \frac{1}{2{,}5s + 1} \cdot 1 = \frac{6s + 4}{7{,}5s^2 + 3s}.$$

Першим критерієм для перевірки стійкості було обрано критерій Найквіста. За цим критерієм: для того, щоб замкнута система була стійка, необхідно і достатньо, щоб різниця між числами позитивних і негативних переходів амплітудно-фазової частотної характеристики розімкненої системи через відрізок (-∞, -1) була рівна 1/2 (1 – число правих коренів характеристичного рівняння розімкненої системи) [16].

Відповідно до цього критерія було написано MATLAB код, який генерує діаграму, за якою можна визначити стійкість апаратної частини системи. Лістинг коду наведено в лістингу В.1 в додатку В. В результаті було отримано діаграму, що наведена на рисунку 4.3.

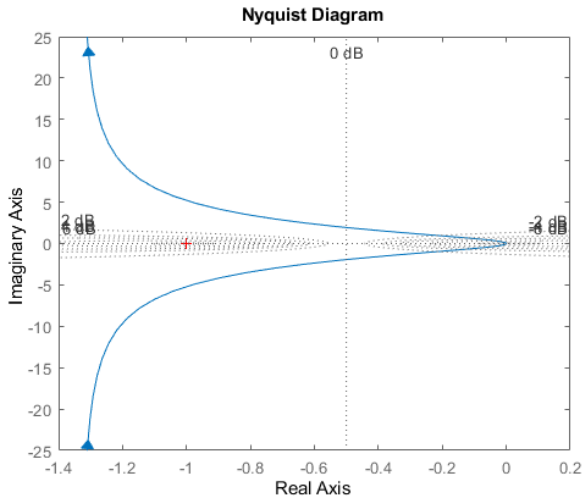

Рисунок 4.3 – Діаграма Найквіста

Відповідно до рисунка 4.3, діаграма не охоплює точку (-1,0), а отже є стійкою.

Наступним критерієм для перевірки стійкості є критерій Гурвіца. Цей критерій дозволяє визначити стійкість системи, аналізуючи коефіцієнти характеристичного рівняння:

$$P(s) = 7{,}5s^2 + 3s + 4.$$

Для цього рівняння необхідно порахувати визначники, використовуючи матрицю другого порядку:

$$H = \begin{vmatrix} a_1 & a_3 \\ a_0 & a_2 \end{vmatrix},$$

де $a_0 = 4$, $a_1 = 3$, $a_2 = 7{,}5$, $a_3 = 0$.

Відповідно перший визначник буде дорівнювати: $\Delta_1 = a_1 = 3$, а другий визначник буде дорівнювати:

$$H = \begin{vmatrix} 3 & 0 \\ 4 & 7{,}5 \end{vmatrix} = 3 \cdot 7{,}5 - 0 = 22{,}5.$$

Тому, можна сказати, що система є стійкою, бо всі визначники Гурвіца є додатними, а $a_0 > 0$.

Останнім критерієм, за яким буде перевірятися стійкість апаратної частини системи, є критерій Михайлова. Критерій Михайлова використовує характеристичне рівняння системи для побудови графіка Михайлова. Відповідно до характеристичного рівняння, де s необхідно замінити на jw і побудувати графік уявної та дійсної частини. Для цього було написано MATLAB код, що наведено в лістингу В.2 додатка В.

Відповідно до скрипту було побудовано графік Михайлова, що наведено на рисунку 4.4.

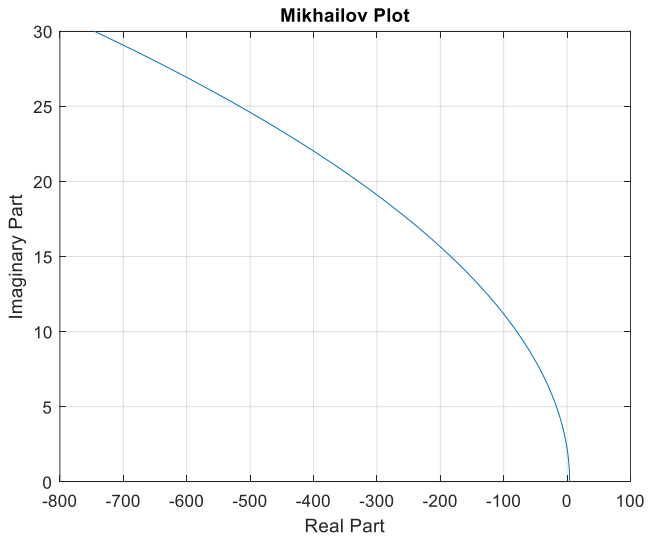

Рисунок 4.4 – Графік Михайлова

Графік починається на правій площині і рухається проти годинникової стрілки, тому можна вважати, що система є стійкою.

5 ОХОРОНА ПРАЦІ

Важливість охорони праці на підприємствах важко переоцінити, особливо в контексті впровадження автоматизованих систем контролю доступу та обліку робочого часу. Система, що базується на використанні QR-кодів, включає в себе апаратні та програмні компоненти, кожен з яких має потенційні ризики для безпеки працівників.

Основними фізичними ризиками є можливість механічних травм при використанні турнікетів, сканерів та інфрачервоних датчиків. Неправильна експлуатація або несправність цих пристроїв можуть спричинити небезпечні ситуації. Електричні ризики також становлять значну загрозу, оскільки система включає в себе електронні компоненти, які потребують безпечного підключення та обслуговування. Крім того, не можна ігнорувати психологічні ризики, такі як стрес, пов'язаний з нововведеннями, які можуть вплинути на робочий процес.

Для мінімізації цих ризиків необхідно впровадити ряд технічних засобів захисту, включаючи, наприклад, різні сигнальні пристрої, які попереджають про небезпеку. Організаційні заходи, такі як регулярні інструктажі з техніки безпеки та перевірки обладнання, допомагають забезпечити безпеку під час експлуатації системи. Особисті засоби захисту, такі як спеціальний одяг та взуття, є необхідними для тих, хто безпосередньо працює з обладнанням.

Обслуговування та ремонт системи потребують особливої уваги до правил безпеки. Персонал, який виконує ці роботи, повинен бути належним чином підготовлений і проінструктований. Дотримання регламентованих процедур з обслуговування обладнання є ключовим для запобігання нещасним випадкам.

Впровадження системи автоматизації контролю доступу та обліку робочого часу співробітників позитивно впливає на умови праці, забезпечуючи більш точний та надійний облік робочого часу. Однак, необхідно уважно стежити за можливими ускладненнями, які можуть виникнути в процесі експлуатації системи [20].

Охорона праці регламентується низкою законів та нормативних актів, які визначають основні вимоги до безпеки на робочих місцях. Важливо розробити внутрішні інструкції та положення, що забезпечать дотримання всіх вимог охорони праці. Це включає як загальні правила, так і специфічні рекомендації для користувачів системи та технічного персоналу.

Отже, охорона праці в контексті розробленої системи автоматизації контролю доступу та обліку робочого часу співробітників є комплексним процесом, що вимагає ретельного планування та впровадження заходів безпеки на всіх етапах експлуатації системи.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено систему автоматизації для контролю доступу та обліку робочого часу співробітників на основі QR-кодів, що є актуальним у сучасних умовах розвитку інформаційних технологій та нестабільної військової ситуації в Україні.

В процесі аналізу існуючих систем контролю доступу та обліку робочого часу, було виявлено основні їх переваги та недоліки, що дозволило визначити ключові вимоги до розроблювано системи.

Для розробки системи було обрано програмні засоби, такі як мова програмування C#, ASP.NET Core Web API, Blazor WASM, база даних MSSQL та WPF, який дозволив провести симуляцію апаратної частини. Для апаратної частини, яка повинна існувати в системі, було обрано турнікет TS1000 Pro, який поставляється з контролером, сканер QR500 для сканування QR-кодів та два інфрачервоні датчики E18-D80NK для реагування на прохід турнікетної лінії.

В ході розробки було створено API для реалізації основної логіки роботи нашої системи, яка працює разом з БД. API містить сервіси для аутентифікації та авторизації, управління робітниками, управління запрошеннями та запрошеними робітниками, управління QR-кодами, керуванням над позиціями та ролями робітників, а також для управління відвідуваннями працівників на підприємстві.

Для зручної взаємодії з системою було створено веб-інтерфейс, який в залежності від ролі працівника представляє різний інтерфейс. Адміністратор має можливість використовувати весь функціонал API, а звичайний робітник може переглядати свою статистику щодо обліку робочого часу та генерувати різні види QR-коду за потреби.

Для цілісності роботи було проведено симуляцію роботи апаратних компонентів, за допомогою якої можна сканувати QR-код, симулювати роботу турнікету та інфрачервоних датчиків та взаємодіяти з API, для реєстрації відвідувань робітників.

Розроблена система значно підвищує ефективність контролю доступу та обліку робочого часу, забезпечуючи при цьому високий рівень безпеки та зручності для користувачів. У контексті нинішньої військової ситуації в Україні, така система також сприяє підвищенню безпеки підприємств та їхніх співробітників, що є особливо важливим для стабільної роботи організацій у кризових умовах.

Для покращення зручності для робітників доцільно впровадити мобільний додаток або застосунок, який забезпечить зручну генерацію QR-кодів та отримання повідомлень про успішний прихід чи вихід з роботи. Це сприятиме підвищенню ефективності та задоволеності користувачів системи.

Результати даної роботи можуть бути використані для впровадження на підприємствах різних галузей, де важливо забезпечити контроль доступу та точний облік робочого часу співробітників. Це сприятиме підвищенню ефективності управління персоналом, оптимізації робочих процесів та покращенню загальної продуктивності підприємств.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ціль 9. Промисловість, інновації та інфраструктура [Електронний ресурс] / – Режим доступу: https://business.diia.gov.ua/handbook/sustainable-development-goals/cil-9-promislovist-innovacii-ta-infrastruktura (дата звернення: 10.05.2024).

2. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України, 2017. – 29 с.

3. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд.: І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2022. – 55 с.

4. Писаренко Д. Г. Сучасна система контролю та управління доступом [Електронний ресурс] / Д. Г. Писаренко, Ю. Ю. Нестюк, А. С. Васюра // Матеріали XLIX науково-технічної конференції підрозділів ВНТУ, Вінниця, 27-28 квітня 2020 р. – Електрон. текст. дані. – 2020. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2020/paper/view/9077 (дата звернення: 01.05.2024).

5. Системи контролю і управління доступом від А до Я [Електронний ресурс] / – Режим доступу: https://deps.ua/ua/knowegable-base/reference-information/7824.html (дата звернення: 01.05.2024).

6. Автономна біометрична панель контролю доступу ZKTeco X8s [Електронний ресурс] / – Режим доступу: https://zkstore.com.ua/ua/p566001643-avtonomnaya-biometricheskaya-panel.html (дата звернення: 01.05.2024).

7. Мережевий комплект СКД на одні двері з урахуванням робочого часу. [Електронний ресурс] / – Режим доступу: https://vario.com.ua/setevoy-komplekt-skd-na-odnu-dver-s-uchetom-rabochego-vremeni (дата звернення: 01.05.2024).

8. Системи контролю доступу: різновиди та поради щодо вибору / Insider-Media [Електронний ресурс] / – Режим доступу: https://insider-media.net/life/systemy-kontroliu-dostupu-riznovydy-ta-porady-shchodo-vyboru (дата звернення: 03.05.2024).

9. Що таке СКУД / Ohrana.ua [Електронний доступ] / – Режим доступу: https://ohrana.ua/uk/stati-i-obzory/chto-takoe-skud.html (дата звернення: 03.05.2024).

10. Про організацію обліку робочого часу на підприємстві / Головний кадровий журнал України / КАДРОВИК.UA [Електронний ресурс] / – Режим доступу: https://www.kadrovik.ua/content/pro-organizaciyu-obliku-robochogo-chasu-na-pidpriiemstvi (дата звернення: 03.05.2024).

11. Система обліку робочого часу [Електронний ресурс] / – Режим доступу: https://www.bezpeka-shop.com/ua/catalog/uchet-rabochego-vremeni/ (дата звернення: 07.05.2024).

12. Біометрична система обліку робочого часу ZKTeco S922 [Електронний ресурс] / – Режим доступу: https://zkstore.com.ua/ua/p888367239-biometricheskaya-sistema-ucheta.html (дата звернення: 07.05.2024).

13. Історія QR-кодів та їх використання в комунікаціях / Bazilik Media [Електронний ресурс] / – Режим доступу: https://bazilik.media/istoriia-qr-kodiv-ta-ikh-vykorystannia-v-komunikatsiiakh (дата звернення: 08.05.2024).

14. Structure of The QR Code: How Is The Data Coded / My QR BC – Digital Business Card [Електронний ресурс] / – Режим доступу: https://myqrbc.com/structure-of-the-qr-code-how-is-the-data-coded/ (дата звернення: 08.05.2024).

15. Стасенко Я., Сезонова І. АВТОМАТИЗАЦІЯ ОБЛІКУ РОБОЧОГО ЧАСУ З ВИКОРИСТАННЯМ QR-КОДІВ // Collection of scientific papers «ΛΌГОΣ». – 2023. – С. 120-122 (дата звернення: 08.05.2024).

16. Турнікет-трипод з антипанікою TS1000Pro [Електронний ресурс] / – Режим доступу: https://zktecoua.com/ua/products/turniket-seriya-ts1000-pro. (дата звернення: 25.05.2024).

17. Зчитувач QR-коду QR500 [Електронний ресурс] / – Режим доступу: https://zktecoua.com/ua/products/schityvatel-qr-koda-qr500/ (дата звернення: 25.05.2024).

18. Інфрачервоний датчик відстані 10-80 см E18-D80NK [Електронний ресурс] / – Режим доступу: https://arduino.ua/prod224-infrakrasnii-datchik-rasstoyaniya-3-80-sm (дата звернення: 25.05.2024).

19. Теорія автоматичного управління (збірник задач) [Текст]: навч. посіб. для студентів спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології / І.Ш. Невлюдов, О.В. Токарєва; Харків. нац. ун-т радіоелектроніки. - Харків: Панов А.М., 2020. – 240 с (дата звернення: 10.06.2024).

20. Методичні вказівки до лабораторних робіт з дисципліни «Основи охорони праці» для студентів усіх напрямів та форм навчання / Упоряд.: Т.Є. Стиценко, В.А. Айвазов, О.В. Мамонтов. – Харків: ХНУРЕ, 2018. – 120 с (дата звернення: 10.06.2024).