

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА**

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту

" _____ " _____ 2023 р.

Зав. кафедри, к.т.н., доцент

_____ Ліщук Р.І.

**КВАЛІФІКАЦІЙНА РОБОТА
МАГІСТРА**

**Програмне забезпечення для віддаленого керування доступом до
об'єктів за допомогою веб-інтерфейсу**

**Виконав – студент 2 курсу, групи 21м-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки**

Гаркавий Владислав Олександрович

Науковий керівник – д.т.н., професор

Кулаков Павло Ігорович

Умань – 2023

Реферат
на кваліфікаційну роботу «Програмне забезпечення для віддаленого
керування доступом до об'єктів за допомогою веб-інтерфейсу»
студента факультету економіки та підприємництва
Уманського національного університету садівництва
Гаркавого Владислава Олександровича

Структура та обсяг роботи - пояснювальна записка до кваліфікаційної роботи магістра складається з 3 розділів, містить 74 сторінок, 22 рисунків, 6 додатків, використано 50 літературних джерел.

Кваліфікаційна робота магістра присвячена розробці програмного забезпечення для віддаленого керування доступом за допомогою веб-інтерфейсу. В процесі виконання роботи проведено аналіз способів побудови програмного забезпечення, в результаті якого було обрано архітектуру, сформульовано основні вимоги до нього, його складові та необхідний функціонал. Об'єктом дослідження є процеси передачі даних. Метою дослідження є розширення функціональних можливостей системи контролю доступу на об'єкти інфраструктури шляхом підтримки можливості дистанційного керування. Наукова новизна отриманих результатів полягає у тому, знайдено саме ефективне використання ресурсів. Практичне значення одержаних результатів полягає в тому, розроблено ефективну систему управління доступом.

У першому розділі кваліфікаційної роботи магістра проведено аналіз способів побудови програмного забезпечення, у другому розділі здійснено дослідження області, у третьому розділі розроблено програмне забезпечення управління доступом, та його тестування.

Анотація

Кваліфікаційна робота магістра присвячена розробці програмного забезпечення для віддаленого керування доступом за допомогою веб-інтерфейсу. Проведено аналіз предметної області, в результаті якого було обрано архітектуру, сформульовано вимоги, його складові та необхідний функціонал. Досліджено структурну та функціональну схеми системи керування доступом, розглянуті питання організації віддаленої взаємодії між пристроєм керування доступом та додатком адміністратора. Наукова новизна отриманих результатів кваліфікаційної роботи магістра полягає у тому, що досліджено весь можливий список приладів і визначено саме ефективне використання ресурсів.

Ключові слова: програмне забезпечення, віддалений доступ, дослідження предметної області, інтерфейс, веб-інтерфейс, база даних, генерація запитів бази даних, Java, організація структур автоматизації, інтегрування мікропроцесорів.

Annotation

The master's thesis is devoted to the development of software for remote access control using a web interface. An analysis of the subject area was conducted, as a result of which the architecture was chosen, requirements, its components and the necessary functionality were formulated. The structural and functional diagrams of the access control system were studied, and the issues of organizing remote interaction between the access control device and the administrator application were considered. The scientific novelty of the results of the master's qualification work is that the entire possible list of devices has been studied and the most efficient use of resources has been determined.

Keywords: software, remote access, domain research, interface, web interface, database, database query generation, Java, organization of automation structures, microprocessor integration.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
 Факультет економіки і підприємництва
 Кафедра _____ Інформаційних технологій
 Освітня програма _____ Комп'ютерні науки
 Спеціальність _____ 122 Комп'ютерні науки
 (код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Ліщук Р.І.

(підпис)

(ініціали, прізвище)

" ____ " _____ 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту _____ Гаркавому Владиславу Олександровичу _____

(прізвище, ім'я, по батькові)

1. Тема проекту Програмне забезпечення для віддаленого керування доступом до об'єктів за допомогою веб-інтерфейсу

керівник роботи Кулаков Павло Ігорович, д.т.н., професор кафедри інформаційних технологій

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від " ____ " _____ 2023 року № 0

2. Строк подання студентом роботи _____ 04.12.2023

3. Вихідні дані до роботи Розробити програмне забезпечення для віддаленого керування доступом до об'єктів за допомогою веб-інтерфейсу з використанням технологій проектування web-push-notification. Для обміну інформацією використати технологію push-notification.

4. Перелік питань, які потрібно розробити Аналіз способів побудови програмного забезпечення для віддаленого керування, вибір архітектури програмного забезпечення, розробка програмного забезпечення, тестування.

5. Перелік графічного матеріалу 22 рисунків, 6 додатків, 50 джерел

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми кваліфікаційної роботи		
2.	Одержання індивідуального завдання		
3.	Демонстрація календарного плану написання кваліфікаційної роботи		
4.	Підготовка до виконання роботи - підбір та вивчення літератури - участь у виконанні науково-дослідних робіт - виконання курсових робіт - інші заходи		
5.	Уточнення теми кваліфікаційної роботи та календарного плану-графіка, виходячи зі специфіки базового підприємства, установи		
6.	Підготовка: - першого розділу - другого розділу - третього розділу - висновків		
7.	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника		
8.	Написання та оформлення роботи в остаточному варіанті		
9.	Перевірка роботи на академічний плагіат		
10.	Одержання відгуку наукового керівника		
11.	Попередній захист роботи на кафедрі		
12.	Затвердження роботи завідувачем кафедри		
13.	Одержання рецензії		
14.	Захист кваліфікаційної роботи		

Студент _____
(підпис)

Гаркавий В.О. _____
(прізвище та ініціали)

Керівник роботи _____
(підпис)

Кулаков П.І. _____
(прізвище та ініціали)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	
ВСТУП.....	
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТУ.....	5
1.1. Огляд існуючих додатків.....	9
1.2. Огляд технологій програмування для створення веб додатків.....	10
1.3 Постановка задачі.....	16
РОЗДІЛ 2. ВИБІР АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	18
2.1 Розробка структурної схеми додатку	18
2.2. Структура програмного забезпечення.....	21
2.3. Напрямки та задачі оптимізації.....	30
РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ	34
3.1 Розробка-бази даних.....	34
3.2 Програмне забезпечення клієнтської системи	44
3.3 Тестування програмного забезпечення	53
ВИСНОВОК.....	58
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	60
ДОДАТКИ.....	Помилка! Закладку не визначено.
Додаток А	65
Додаток Б.....	66
Додаток В	67
Додаток Г.....	68
Додаток І	69
Додаток Д.....	77

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Andoid	-Операційна система для смартфонів
IT	-Інформаційно-комп'ютерні технології
JVM	-Java Virtual Machine
SDK	-Software Development Kit
IDE	-Integrated Development Environment
WIFI	-Wireless Fidelity
АЦП	-Аналого-цифровий перетворювач
IEEE	-Institute of Electrical and Electronics Engineers
OSI	-Open Systems Interconnection
БД	-База даних
СУБД	-Система управління бази даних
SQL	-Structured Query Language
MySQL	-Вільна реляційна система управління базами даних
РБД	-Реляційна база даних
TCP	-Transmission Control Protocol
UDP	-User Datagram Protocol
HTTP	-HyperText Transfer Protocol
FTP	-File Transfer Protocol
MAC	-Media Access Control
IP	-Internet Protocol
ОС	-Операційна система
JAR	-Java ARchive
USB	-Universal Serial Bus
API	-Application Programming Interface

ВСТУП

Система контролю та управління доступом (Access Control System, ACS) — комплекс програмно-апаратних засобів, що забезпечують можливість контролю та обмеження доступу до контрольованих об'єктів, реєстраційні форми осіб, які входять, виходять, переміщуються певними зонами [1].

Актуальність теми: Використання сучасних технологій дозволяє впроваджувати системи контролю та управління доступом, здатні ефективно вирішувати низку завдань, починаючи від боротьби з промисловим шпигунством і захисту конфіденційної інформації, закінчуючи моніторингом персоналу [2]. Сучасні системи контролю доступу можуть забезпечити рівень захисту, необхідний у великих установках з тисячами точок доступу та десятками тисяч користувачів.

Крім того, система також служить основою для побудови інтегрованої системи безпеки, яка поєднує охорону, пожежну сигналізацію та телевізійне спостереження [3].

Сьогодні системи контролю та управління доступом займають чималу частку ринку безпеки. Навіть під час кризи падіння в цьому сегменті ринку було незначним і відбулося лише за рахунок певного уповільнення реалізації масштабних проектів, тоді як динаміка компактних установок і локально практично не змінилася.

Це можна пояснити тим, що таких системи відносяться до категорії засобів безпеки, завдяки яким можна підвищити не тільки рівень стійкості до загроз, а й економічне зростання [3].

На відміну від інших технологій, робота системи контролю та управління доступом пов'язана з безпосередньою взаємодією з користувачем.

Інформація, яку ці системи отримують про дії співробітників і відвідувачів, може використовуватися іншими системами. Це не тільки спрощує роботу, але й економить ресурси. Наприклад, дані про час приходу/відходу співробітників дозволяють регулювати мікроклімат на робочому місці [4].

Одним із найперспективніших напрямків використання систем контролю та керування доступом є сфера автоматизації та забезпечення безпеки від проникнення сторонніх людей. Система контролю доступу – це сучасний інструмент, завдяки якому значно підвищується рівень безпеки, що забезпечує ефективне використання енергоресурсів [5].

На сьогоднішній день повсякчас використовується нові технології в повсякденному житті. Неможливо уявити сьогоднішнє життя без технологій “розумного” дому, використання автоматизації на різноманітних організаціях, використання програм для отримання необхідного функціоналу, що дуже спрощують і підвищують продуктивність праці [6]. Щоб отримати найкращий результат можливо використати прогресивні додатки які працюють значно швидше, та опрацьовувати поставлені дії, займаючи менше пам'яті їх зручно інтегрувати безліч всеможливих систем. За допомогою цього програмного рішення надається можливість неймовірно сильно оптимізувати поставлену ціль [7]. Можна також навести декілька прикладів, щодо актуальності даної теми:

1. Індустріальна автоматизація. Промисловість активно використовує дану технологію для оптимізації виробничих процесів, підвищення продуктивності та забезпечення якості продукції. Роботи, такі як автоматизовані виробничі лінії, роботи-маніпулятори та системи керування, стають необхідними для сучасних заводів.

2. IT та розробка програмного забезпечення. У сфері інформаційних технологій автоматизація використовується для автоматизації розробки, тестування та розгортання програмного забезпечення.

3. Обробка даних та аналітика. Автоматизація в обробці даних та аналітиці дозволяє автоматизовано аналізувати величезні обсяги даних для отримання цінної інформації. Такі як методи машинного навчання та штучного інтелекту.

Данні методи використовують нові технології, які було інтегровано для захисту і автоматизації процесів доступ система контролю та управління доступом дає можливість розділяти права доступу в різні приміщення для співробітників, попереджає можливість несанкціонованого проникнення до

підприємства. Як наслідок підвищується якість обслуговування та безпека працівників, а також рівень контролю за найманим персоналом, що сприяє підвищенню ефективності його роботи та запобігає можливим зловживанням, щоб зменшити і мінімізувати можливий допуск помилок персоналу.

Об'єкт дослідження: процеси передачі даних в інформаційних системах.

Предмет дослідження: Методи та засоби контролю та управління віддаленим доступом через Інтернет.

Мета дослідження: Дипломного проекту є розширення функціональних можливостей системи контролю доступу на об'єкти інфраструктури за рахунок підтримки можливості дистанційного керування доступом і використання параметрів мережевої ідентифікації.

Задачі дослідження, які необхідно виконати в даній дипломній роботі:

1. Аналіз сучасних тенденцій у сфері контролю та управління доступом .
2. Аналіз сучасних технології та бездротових засобів ідентифікації.
3. Визначити принципи побудови системи управління віддаленим доступом з розширеною функціональністю, структурно та функціональному рівні .
4. Створення та аналіз бази даних, використання доступних методів взаємодії з нею, інтегрування програми контролю доступу.

Методи дослідження: В дипломному проекті було використано декілька методів дослідження, щоб охарактеризувати і встановити більш детальний аналіз.

1. Метааналіз - даний метод був направлений на аналіз всіх можливих методів розробки програми в загальному доступі, аналіз інформації декількох досліджень дає уявлення про задачі поставлені перед доповідачем, проблемах з якими зіткнувся, та який результат отримав.

2. Експеримент - був проведений аналіз де буде визначено фактори, що впливають на досліджуване явище і їх взаємодія.

3. Контент-аналіз в якому було проведено аналіз текстового, візуального для визначення певних шаблону, який повинен мати вигляд програма.

Наукова новизна отриманих результатів полягає у тому, знайдено саме ефективне використання ресурсів з великого переліку можливих реалізацій схожих завдань, також аналіз всіх мікроконтролерів відкрив можливість для новітніх рішень ефективних при даному завданню.

Практичне значення одержаних результатів полягає в тому, розроблено ефективну систему управління доступом яка має чудовий захист всіх даних про співробітників, а також розділяє їх доступ по організації.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТУ

1.1. Огляд існуючих додатків

Delta Guard - Дана компанія представляє можливості інформаційної безпеки, так і фізичного захисту об'єктів можливої для атаки хакерів, має чудову обізнаність в сфері забезпечення захисту інформації використовуючи сучасні технології проводячи аналіз підприємства з виявленням можливих недоліків і вразливостей [8].

Переваги:

1. Додаток має комплексний захист інформації, від всеможливих атак “третьою” особою.
2. Своєчасне попередження загроз на основі аналізу, а також вироблення відповідні профілактичних заходів.
3. Системне підтримання програми.
4. Документовані регламенти про використання програми працівниками.
5. Гарантійне обслуговування обслуговування програми, а також пристроїв системи.

Недоліки:

1. Використання ресурсомістких даних і недостатня продуктивність системи.
2. Відсутність інтеграції в інші ефективні системи.
3. Не можливість отримання необхідної інформації для вирішення проблем системи.

Wise it - Компанія забезпечує та розширює можливості цифрового майбутнього у світі, орієнтованому на розробці, а також підтримання конфіденційної інформації будь-якої організації, отримати рекомендації щодо впорядкування ІТ-інфраструктури, придбати ІТ-обладнання та замовити його інтеграцію в наявну або ж побудувати нову інфраструктуру [9].

Переваги:

1. Своєчасне попередження загроз на основі аналізу, а також вироблення відповідні профілактичних заходів.
2. Системне підтримання програми.
3. Постійні заходи аудиту програми.
4. Заходи перевірки організації на проникнення.
5. Використання хмарного сховища для користувачів.
6. Системне підтримання програми.
7. Розробка та стратегії для покращення системи управління організації.

1.2. Огляд технологій програмування для створення веб додатків

Java - Офіційна мова програмування високого рівня на. Це універсальна, надійна, безпечна та об'єктно-орієнтована мова програмування першою і залишається найпоширенішою можна розробляти програми для смартфонів, планшетів, ноутбуків, електронних книг, розумних годинників, а ще безлічі інших пристроїв, що працюють під управлінням Android [10].

Переваги:

1. Кросплатформеність. Помагає скомпілювати його для Android і запускати на широкому спектрі пристроїв без необхідності великої модифікації коду яка полегшує розробку для різних пристроїв.
2. Велика спільнота і підтримка має широку та активну спільноту розробників, що забезпечує доступ до великої кількості ресурсів, документації, бібліотек та фреймворків для полегшення розробки.
3. Багатий екосистем має багато готових бібліотек, фреймворків та інструментів, що полегшують і прискорюють процес розробки. Використовування багато готових рішень для реалізації різних функціональностей додатків.
4. Широкі можливості оптимізації та розширення. Java пропонує багатий набір інструментів для оптимізації та профілювання коду проблем.

5. Безпека. Java володіє вбудованими механізмами безпеки, такими як механізми контролю доступу та обробки винятків, що дозволяє розробникам створювати більш безпечні додатки.

6. Можливості для розширення додатків підтримує різні сторонні бібліотеки та фреймворки, які полегшують використання розширених функцій у мобільних додатках, таких як робота з графікою, анімації, робота з базами даних.

Недоліки:

1. Недостатня оптимізація. Можуть бути менш ефективними за швидкістю порівняно з деякими іншими мовами.

2. Вимоги до пам'яті. Додатки на Java можуть вимагати більше пам'яті порівняно з іншими мовами.

3. Довгий час запуску. Може займати більше часу порівняно з додатками, написаними на інших мовах. Це може вплинути на користувацький досвід, особливо на пристроях із слабкими характеристиками.

4. Додатки Java використовують JVM, і це може впливати на величезний обсяг пам'яті та обсяги.

5. Більш великі артефакти. Можуть включати більше коду та більше бібліотек, що може зробити їх артефакти

6. Можливість несумісності між версіями. Виникає проблема несумісності між різними версіями.

Kotlin - Це сучасна, багатоцільова мова програмування використовуючи синтаксис платформи Java, для веб-розробки та багато іншого. Kotlin вирізняється високою виразністю, безпекою типів, і зручністю у використанні, що робить його привабливим вибором для широкого кола завдань і проєктів [10].

Переваги:

1. Сумісність з Java. Kotlin повністю сумісний з Java, тобто можете використовувати існуючий Java-код в проєкті Kotlin

2. Безпека та надійність. Kotlin надає безпечний і надійний код завдяки таким конструкціям, як нульовий тип (nullable types) та вбудована система перевірки на виключеннях.

3. Коротший та зрозуміліший код. Дозволяє розробникам писати менше коду для досягнення того самого результату, що сприяє збільшенню продуктивності і зменшенню кількості можливих помилок.

4. Функціональність. Підтримує функціональне програмування та має ряд функціональних конструкцій, інтерфейси функцій, що полегшують створення високопродуктивного та сучасного коду.

5. Розширення можливостей Android SDK. Полегшує роботу з деякими функціями, а також надає зручний синтаксис для деяких аспектів розробки.

6. Активна підтримка від Google, що гарантує стабільність та підтримку мови.

Недоліки:

1. Великий розмір файлів. Компільований код Kotlin може мати більший розмір порівняно з Java-кодом.

2. Проблеми з інтеграцією з деякими Java-бібліотеками в проект Kotlin і не всі бібліотеки можуть бути адаптовані для використання в ньому.

3. Швидкість компіляції. Залежно від розміру проекту, компіляція Kotlin може займати більше часу та ефективності роботи.

4. Можливість несумісності між версіями. Виникає проблема несумісності між різними версіями.

C++ - представляє високорівневу компілювану мову програмування загального призначення зі статичною типізацією, яка підходить для створення різних програм. Для розробки мобільних додатків, зокрема для платформи Android, може бути обґрунтованим у деяких сценаріях [11].

Переваги:

1. Висока продуктивність C++ є мовою низького рівня, що дозволяє оптимізувати виконання коду і досягати високої продуктивності. Це особливо важливо для додатків, які вимагають швидкодії.

2. Доступ до низькорівневих операцій. C++ надає розробникам можливість працювати з пам'яттю та проводити низькорівневі операції, що

робить його ідеальним для додатків, які вимагають точного управління ресурсами.

3. Кросплатформеність. Додатки, написані на C++, можуть бути легко переносимі між різними платформами, що дозволяє розробникам створювати кросплатформенні рішення для Android та інших систем, таких як iOS.

4. Багатий функціонал. C++ надає розробникам доступ до широкого спектру функцій і бібліотек, що може полегшити створення різноманітних додатків з різноманітними функціональностями.

5. Підтримка багатозадачності. C++ дозволяє розробникам створювати ефективні мобільні додатки, які можуть ефективно обробляти багатозадачні завдання та взаємодіяти з іншими компонентами пристрою.

6. Можливість використання існуючого коду. Розробники можуть використовувати існуючий C++-код, що може значно зменшити час розробки та спростити інтеграцію з іншими системами.

Недоліки:

1. Складність в порівнянні з іншими мовами. C++ є мовою програмування низького рівня, і вона може бути складною для вивчення та розуміння, особливо для початківців. Це може впливати на швидкість розробки та підтримки.

2. Можливість виникнення помилок пам'яті. Однією з характерних особливостей C++ є ручне управління пам'яттю, таких як витоки пам'яті або доступ до видаленої пам'яті.

3. Складніше виявлення помилок. Можуть бути важко виявити та виправити, що може призводити до виникнення проблем на етапі виконання.

4. Менше підтримки для низькорівневого доступу до апаратного забезпечення. У деяких випадках, для взаємодії з конкретними апаратними функціями мобільного пристрою.

В даній дипломній роботі буде використовуватись мова програмування Java, так як вона є найбільш адаптованою і призначеною для розробки веб додатків, використовуючи її бібліотеки, а також зручну можливість кросплатформенного адаптування можливо отримати найбільш зручний, а також

збалансований продукт для користувача з можливістю додаткових програмних покращень, які будуть відбуватись за допомогою написання і редагування бібліотек для оптимізації програмного продукту

Однак окрім мов програмування також необхідно вибрати і IDE за допомогою якого буде відбуватись написання коду

Android Studio - це інтегроване середовище розробки (IDE) для створення мобільних додатків для платформи Android. Воно базується на IntelliJ IDEA та надає спеціалізовані інструменти для розробки Android-застосунків. Шаблони основних макетів та компонентів Android. Підтримка розробки програм для Android Wear та Android TV. Вбудована підтримка Google Cloud Platform, яка включає інтеграцію з сервісами Google Cloud Messaging і App Engine [12].

Переваги:

1. Інтеграція з Android SDK. Android Studio поставляється з вбудованим Android SDK, який включає в себе необхідні інструменти та бібліотеки для розробки Android-додатків. Це спрощує налаштування середовища розробки та забезпечує легку інтеграцію з платформою Android.

2. Графічний редактор інтерфейсу користувача. Android Studio має вбудований графічний редактор інтерфейсу користувача, що дозволяє вам візуально створювати та налаштовувати інтерфейс додатка безпосередньо в IDE.

3. Емулятори та реальні пристрої. Засоби емуляції в Android Studio дозволяють вам тестувати додатки на різних віртуальних пристроях з різними версіями Android. Також легко підключити реальні пристрої для тестування.

4. Широкий вибір плагінів і розширень. Є багато розширень та плагінів для Android Studio, які дозволяють розширювати його функціональність та вдосконалювати робочий процес розробки.

Недоліки:

1. Обновлення IDE. Після виходу нового обновлення, можлива непрацездатність програми для розробки Android-застосунків, так як плагіни які знаходяться в ній будуть вважатися застарілими.

2. Повільне завантаження та складання проєктів. Деякі розробники відзначають, що процеси завантаження проєктів та складання можуть займати значний час, особливо для великих проєктів. Це може вплинути на загальну продуктивність і ефективність розробки.

3. Великий обсяг дистрибутива. Android Studio має досить великий розмір інсталяційного файлу, що може викликати складності при завантаженні та установці.

IntelliJ IDEA - це комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP) від компанії JetBrains. Програмний код Community-версії поширюється в рамках ліцензії Apache 2.0. Бінарні збірки підготовлені для Linux, Mac OS X і Windows. інтегроване середовище розробки (IDE), за допомогою плагінів, може використовуватися для розробки Android-додатків [13].

Переваги:

1. Ергономіка та Інтелектуальні функції. Дана IDE відомий своєю високою ергономікою та інтелектуальними функціями, такими як автозаповнення коду, аналіз коду, автоматичне виправлення помилок, рефакторинг. Це полегшує процес написання коду та підвищує продуктивність розробника.

2. Графічний редактор інтерфейсу користувача. IntelliJ IDEA має вбудований графічний редактор інтерфейсу користувача, що дозволяє вам легко створювати та редагувати UI-компоненти для Android-додатків.

3. Широкий вибір плагінів. IntelliJ IDEA підтримує широкий вибір плагінів, які розширюють його функціональність та можливості. Це дозволяє налаштовувати IDE під ваші потреби та використовувати додаткові інструменти.

4. Кросплатформенність. IntelliJ IDEA є кросплатформеним, тобто він підтримується на різних операційних системах, включаючи Windows, macOS та Linux.

Недоліки:

1. Вартість. IntelliJ IDEA має платну версію (Ultimate Edition) з додатковими функціями, і хоча існує безкоштовна Community Edition, вона не

містить всіх функцій. Для комерційного використання може знадобитися придбання ліцензії.

2. Високі системні вимоги. IntelliJ IDEA може вимагати значних ресурсів комп'ютера, що може призвести до сповільнення роботи на менш потужних пристроях або у віртуальних оточеннях з обмеженими ресурсами.

3. Оновлення IDE. Після виходу нового оновлення, можлива непрацездатність програми для розробки Android-застосунків, так як плагіни які знаходяться в ній будуть вважатися застарілими.

Рішення на прийняття якій IDE буду працювати залежало в легкій доступності і можливості отримати цілий продукт з всеможливими функціями, також вибір залежав в доступності додаткових бібліотек які постійно оновлюються і довга підтримка програми Android Studio

1.3 Постановка задачі

Проблема дослідницького завдання полягає у виборі найбільш прийнятної моделі авторизації за будь-якого структурованого набору вхідних параметрів, які визначатимуть ключові завдання та пріоритети бізнесу.

Іншими словами, це дослідження дасть відповідь на питання про те, як вибрати необхідний підхід, щоб він не був достатньо складним і дорогим для даної інформаційної системи, але в той же час, щоб форма авторизації, яка повністю включала все що потрібно і немає необхідності повністю міняти його через рік або два.

Для цього будуть використані найпопулярніші моделі доступу на сьогоднішній день, їх переваги та недоліки, а також сформовано набір критеріїв, за якими буде обрана та чи інша модель, що зробить оптимальний вибір для даного набору вимоги.

Критерії для вибору конкретної моделі доступу ґрунтуватимуться на створенні «паспорта системи». По суті, це дослідження конкретної інформаційної системи, у якій задокументовано всі параметри та властивості контролю доступу.

Очікуваним результатом після завершення роботи є створення алгоритму вибору і на його основі сформульованих даних, використовуючи певний набір вхідних даних шляхом обчислення кількості необхідних ролей, привілеїв і атрибутів. Ця інформація покращить процес створення інформаційної системи, допоможе зробити правильний вибір моделі авторизації, що в подальшому безпосередньо вплине на безпеку даних та часові витрати операцій, підтримки та розробки нових політик безпеки.

РОЗДІЛ 2. ВИБІР АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Розробка структурної схеми додатку

Розвиток Інтернету, а також бездротових технологій дозволив інтегрувати дистанційне керування електронними пристроями і враховуючи те, що для поставленої задачі дипломної роботи необхідно створити систему в якій буде реалізовано передачу інформації (сигналів) за допомогою WIFI або інших бездротових технологій для доступу до глобальної мережі та виконати обробку даних шляхом передачі їх на мікропроцесорний пристрій [14].

Також необхідно забезпечити стабільне підключення та продуктивність пристрою для використання доступу до глобальної мережі.

Однак, крім можливості підключення до електричної мережі, слід зазначити, що робота приладу здійснюється від генератора, для автономної роботи в разі аварійної ситуації.

Використовуючи дані, необхідні для вирішення цієї проблеми, було створено схему взаємодії з користувачем між нашим мікропроцесорним пристроєм та користувачем, який отримує до неї доступ через мережу. (Додаток А).

Існує три способи введення аналогових сигналів у мікроконтролер. Кожне з цих джерел має певні переваги для різних застосувань.

Перший спосіб полягає у використанні датчика, за допомогою якого мікроконтролер визначає фізичне положення движка потенціометра.

Другий спосіб полягає у включенні аналогового компаратора, який визначає, чи є значення вхідної напруги вищим або нижчим за заданий рівень (опорної напруги).

Третій тип полягає у використанні мікроконтролера з аналого-цифровим перетворювачем (АЦП), вбудованим у кристал, що дозволяє вимірювати значення напруги на вході.

Даний транзисторний ключ, який представлений вище відноситься до першого типу входу аналогового сигналу. При цьому аналого-цифрове перетворення фактично не виконується, а поточне значення резистора потенціометра визначається за допомогою введення-виведення цифрових даних.

Даний транзистор використовується в тому випадку, коли достатньо контролювати момент досягнення сигналом заданого рівня напруги приймає керує мікропроцесором взаємодіючи з користувачем, тобто після запиту доступу до даного замка, був отриманий позитивний результат, мікропроцесорний ключ буде відчиняти замок.

Електромагнітним елементом комутації є електромагнітні реле, в основі роботи якого покладене явище електромагнітної індукції. Замикання механічних електричних контактів реле відбувається під дією магнітного поля, що виникає у котушці при протіканні крізь неї струму керування.

Електромагнітні реле поділяються на поляризовані, що реагують на полярність сигналу керування та нейтральні, які реагують лише на значення струму, що протікає крізь обмотку. Існують також і імпульсні реле, в яких для керування станом вихідних контактів використовується імпульсний сигнал.

Після зняття імпульсу керування реле залишається в одну з двох стійких станів – увімкнено або вимкнено. Такі реле споживають енергію лише в момент перемикавання, на відміну від звичайних, які для збереження увімкненого стану потребують постійного прикладання напруги керування.

Електромеханічними твердотільні реле характеризуються меншим часом спарювання та більшим ресурсом роботи, який у електромеханічних реле обмежений і визначається максимально можливою кількістю перемикань. При дотриманні допустимих умов і умов експлуатації кількість перемикань напівпровідникових реле практично не обмежена.

Перевагами електромагнітні реле за порівнянням з твердотільними є забезпечення можливості комутації навантажень загальною потужністю до 4 кВт при малих габаритних розмірах, стійкість до імпульсів перенапруги, високий

ступінь ізоляції між вихідною контактною групою та вхідним колом керування, мала розсіювана теплова потужність замкнених контактів, низька вартість.

Враховуючи ці переваги та те, що імпульсні електромагнітні реле на відміну від інших мають найменшу споживану потужність керування, у блоці комутації пристрою ідентифікації та надання доступу будемо використовувати реле саме цього типу.

Останнім елементом пристрою є блок індикації, що забезпечує надання інформації про стан замка та результат підтвердження запиту. Введення цього структурного блоку до складу пристрою дозволяє реалізувати інтерфейс взаємодії з користувачем для відображення реакцію пристрою на його дії.

Для відображення стану замка (відкритий чи закритий) застосуємо світлову індикацію з використанням світлодіодів. Для того, щоб підвищити інформативність, будемо здійснювати світлову індикацію як відкритого, так і закритого стану замка з використанням двокольорового світлодіода з червоним та зеленим кольорами світіння. Випромінювання зеленого кольору буде відображати відкритий доступ, червоного – закритий.

Для індикації процесу та результату запиту застосуємо як світлову, так і звукову індикацію. Світлова індикація буде відображати активність зчитувача. Її активація буде відбуватися при отриманні запиту на отримання доступу. Звукова індикація буде використовуватися для інформування користувача про результат зчитування та розпізнавання коду. Для цього застосуємо звукові сигнали різної тривалості.

При отриманні позитивного результату зчитування, що дозволяє доступ, буде отримуватися короткочасний звуковий сигнал. Невдалий результат сканування, що може мати місце при відхиленні доступу, буде супроводжуватися звуковим сигналом збільшеної тривалості. Для отримання звукового сигналу доцільно використати п'єзоелектричний акустичний випромінювач. Відмінною ознакою випромінювачів такого типу є малі габаритні розміри та сумісність сигналів керування із сигналами цифрових пристроїв.

2.2. Структура програмного забезпечення

Відповідно до проведеного аналізу необхідно виконати надання прав доступу до мікропроцесора в розробленій системі.

Мікропроцесор оснащений транзисторним блокуючим елементом, який дозволяє доступ, ідентифікація якого включатиме дозвіл на доступ, отриманий працівником, що супроводжується підтвердженням його запиту, яке буде надіслано йому.

Щоб уникнути протиправних дій, необхідно забезпечити порівняння цього працівника зі списком усіх співробітників, присутніх у базі даних. Тому в розробленій системі управління віддаленим доступом необхідно не тільки підтримувати можливість надання доступу працівникам, але й оновлювати його в базі даних. Щоб він міг отримати доступ і взаємодіяти з мікропроцесором [15].

Оновлення бази даних для надання доступу до ідентифікатора пристрою найефективніше за допомогою Інтернет-інструментів . Для оновлення бази даних необхідно бути підключеним до Інтернету, можна використовувати як дротовий і бездротовий канали передачі даних. Дротовий зв'язок зазвичай здійснюється за допомогою витії пари.

Перевагами дротового зв'язку є висока швидкість передачі та надійний зв'язок. Однак цей вимагає додаткових витрат на прокладку кабелю.

Для передачі інформації в бездротовій технології використовуються радіохвилі або інфрачервоне випромінювання. У порівнянні з дротовими технологіями, бездротова мережа пропонує більш гнучке розміщення обладнання, що робить простішим у використанні, характеризується простотою та низькими витратами на встановлення після встановлення. Однак бездротовий спосіб передачі менш надійний і менш захищений, характеризується меншою швидкістю обміну даними [16 - 17].

Враховуючи, що розроблена система має бути простою в розгортанні, не залежно від внутрішнього простору організації, тобто всередині приміщення, ми

будемо використовувати технологію бездротової передачі даних для забезпечення зв'язку з Інтернетом.

Технологія Wi-Fi є потужним і надійним бездротовим рішенням, яке успішно використовується для побудови локальних мереж протягом багатьох років, і містить сімейство протоколів бездротової передачі даних IEEE 802.11x. Стандарт бездротової мережі 802.11 став де-факто глобальним стандартом зв'язку, оскільки він пропонує широкий спектр гнучких функцій і постійно вдосконалюється [18].

Розроблений для швидкого обміну великими обсягами даних на невеликих відстанях, Wi-Fi чудово виконує свою роботу. Основні параметри, такі як діапазон покриття або швидкість передачі даних, відрізняються між різними версіями стандарту, але в більшості випадків типового домашнього бездротового маршрутизатора достатньо, щоб забезпечити мережеве покриття невеликої квартири. У великих будівлях ви можете збільшити кількість точок доступу або ретрансляторів сигналу, щоб збільшити покриття [19 - 21].

Крім того, однією з головних переваг Wi-Fi є універсальна доступність інфраструктури 802.11. Той факт, що цей стандарт інтегрований у нові ноутбуки, смартфони та планшети, має велике значення з точки зору впровадження програм керування для організацій та «Інтернету речей» [20].

Завдяки різкому здешевленню мікропроцесорної техніки, одночасному збільшенню надійності і технічних характеристик, зменшенню розмірів і збільшенню функціональних можливостей з'явилася велика кількість малогабаритних контролерів і мікрокомп'ютерів. Так, мікропроцесори і мікроконтролери широко використовуються у побутовій техніці, автомобільній електроніці, аерокосмічній та військовій галузях і, звичайно ж, у промисловому виробництві [22].

Наявність розвинених засобів передачі інформації дозволяє об'єднати ці контролери в єдину мережу, причому різні вузли цієї мережі можуть розташовуватися на досить великій відстані один від одного [23].

Така архітектура системи управління має такі переваги:

1. Висока надійність роботи системи. Чіткий розподіл обов'язків між кожним окремим вузлом системи дозволяє системі працювати навіть у разі збою або зависання пристрою.

2. Менше дротових підключень. Контролер здатний працювати в важких промислових умовах, тому його часто встановлюють поблизу об'єкта управління.

3. Можливість легкого розширення системи. Коли в системі з'являються додаткові контрольні точки, ви просто додаєте в систему новий вузол.

4. Період модернізації був відносно коротким.

5. Вміння використовувати комп'ютери та контролери меншої потужності. Оскільки кожен вузол виконує лише окреме завдання, немає потреби в дорогих комп'ютерах.

6. Легко тестувати та налагоджувати. Оскільки всі компоненти системи активні, легко запровадити самодіагностику та пошук несправностей [24 - 25].

Необхідність модернізації інтерфейсів управління в основному пов'язана з необхідністю підвищення функціональності пристроїв, що входять до системи, а також зниження витрат на установку та експлуатацію. Традиційний спосіб зв'язку в розподілених системах управління за допомогою уніфікованих аналогових електричних сигналів все ще активно використовується, але його можливостей для деяких суб'єктів вже недостатньо [26].

У зв'язку з цим нещодавно стало популярним підключення до польової шини, яке дозволяє передавати команди керування, запити статусу, сигнали відповідей та іншу інформацію в цифровій формі, включаючи використання існуючих ліній зв'язку. При цьому можуть використовуватися різні протоколи обміну даними. Однак проблема при використанні цього методу полягає в тому, що зі збільшенням швидкості передачі інформації максимальна відстань між вузлами мережі значно зменшується (рис. 2.1) [27].

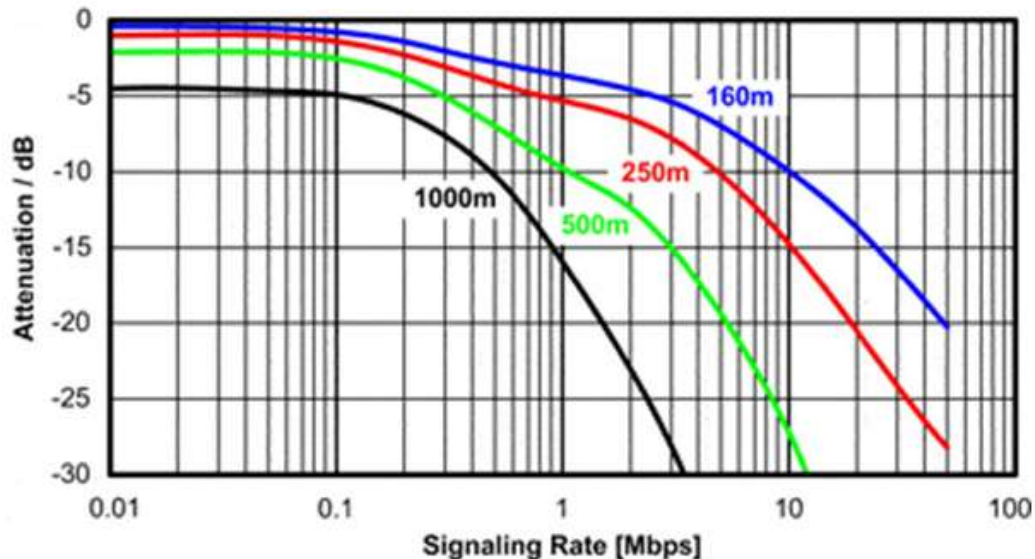


Рисунок 2.1 - Залежність затухання корисного сигналу від швидкості передачі даних на різних відстанях(RS-485)

Найпоширенішими протоколами передачі даних у промисловій автоматичі є:

1. PROFIBUS (PROcess Field BUS) - це відкрита промислова мережа польового рівня, яка відповідає вимогам міжнародного стандарту IEC 61 158 / EN 50170, призначена для побудови розподілених систем введення-виведення, а також організації обміну даними між системами автоматизації [28].

2. Modbus - Комунікаційний протокол зв'язку на основі технології «ведучий-підлеглий». Він широко використовується в промисловості для організації зв'язку між електронними пристроями. В основному використовується для передачі даних по послідовних лініях RS-485, RS-422, RS-232, також модифікація для мережі TCP/IP (Modbus TCP) [29].

3. Технологія Foundation Fieldbus це двостороння, послідовна, цифрова система зв'язку, яка служить базовим мережевим рівнем у заводських або фабричних системах автоматизації. Це відкрита архітектура, розроблена та впроваджена компанією Foundation Fieldbus [30].

4. HART це протокол передачі цифрових даних у промислових мережах був розроблений компанією Rosemount Fischer (США) у середині 1980-х рр. З 1990-х рр. він був удосконалений і став загальновизнаним стандартом комунікації, що дозволяє обмінюватися інформацією між інтелектуальними передавачами та приладами [31].

5. Controller Area Network (CAN) стандарт для організації високонадійних і недорогих каналів зв'язку в розподілених системах управління. CAN використовується в обладнанні промислової автоматизації, обладнанні для інтеграції технологій розумного будинку, автомобільній промисловості та інших галузях. Як правило, інтерфейс CAN використовується як сполучна ланка між основним контролером і різними допоміжними датчиками, механізмами і виконавчими механізмами, підключення яких до центральної лінії не завжди підходить [32].

Для того, щоб обрати найкращий протокол передачі даних, складемо порівняльну таблицю їхніх характеристик(табл. 2.1).

Таблиця 2.1. Порівняння різних протоколів

Протоколи	Максимальна швидкість	Максимальна кількість пристроїв на одній лінії	Максимальна довжина кабелю
Profibus	12 Мбіт/с	126	100 м
Modbus	115,2 кбіт/с	247	1,2 км
Foundation Fieldbus	31,25 кбіт/с	32	1,9 км
HART	1,2 кбіт/с	15	3 км
CAN	1 Мбіт/с	30	25 м

Як показано в таблиці, жоден протокол не є найкращим за всіма параметрами одночасно. Якщо протокол підтримує дуже високу швидкість передачі даних (пропускну здатність), він часто не може передати цей сигнал

на достатньо велику відстань і навпаки, якщо протокол здатний передавати інформаційні пакети на дуже великі відстані, це відбувається за рахунок втрати кількості підтримуваних пристроїв на шині та значно нижчої максимальної швидкості.

Це означає, що вибір найкращого протоколу для передачі даних між пристроями в промисловій автоматизації залежить від конкретної ситуації.

Якщо максимальна відстань від головного до керованих пристроїв знаходиться у межах до 300 м, тоді одним з найкращих варіантів стане Profibus, однак якщо відстань перевищує 300 м, тоді Modbus стане у нагоді. Не слід забувати про максимальну кількість пристроїв на одній шині.

Наприклад, якщо керованих пристроїв більше 50, тоді буде доволі складно об'єднати їх у єдину систему за допомогою Foundation Fieldbus, HART та CAN.

Стандарт протоколу описує рівні 1, 2 і 7 моделі OSI. PROFIBUS використовує гібридну процедуру доступу головний/підлеглий і децентралізовану процедуру передачі маркерів [33]. В якості середовища передачі використовується екранована вита пара, скляні або пластикові оптоволоконна. Довжина кабелів сегмента шини залежить від швидкості передачі даних. 100 м з найвищою швидкістю передачі (12 Мбод) і 1000 м з найнижчою швидкістю передачі (9,6 Кбод). Повторювачі та модулі оптичного зв'язку можна використовувати для розширення діапазону вашої мережі.

Мережа PROFIBUS може складатися зі 122 вузлів, з яких 32 можуть бути Master-вузлами. У середовищі Master-вузлів за схемою зростаючих номерів передається маркер, який надає право вузлам здійснювати на шині цикли читання або запису. Активний вузол одержує права доступу до шини на певний інтервал часу, після закінчення якого права доступу передаються наступному вузлу. Якщо активному вузлу (майстру) призначені пасивні вузли (підпорядковані), майстер обмінюється з ними даними поки володіє правами доступу [34 - 35].

При побудові багаторівневих систем автоматизації часто постає завдання організації обміну інформацією між рівнями.

PROFIBUS-FMS - вимагає обміну складними повідомленнями на середніх швидкостях, призначений для роботи на так званому цеховому рівні [36].

PROFIBUS-DP вимагає швидкого обміну короткими повідомленнями за допомогою спрощеного протоколу обміну (рівень датчика). Для його комутації пристроїв потрібна всього одна кручена пара, яка може одночасно використовуватися і для інформаційного обміну, та для підведення живлення до пристроїв польового рівня [38].

PROFIBUS-PA. вимагає роботи на небезпечному виробництві (нафтопродукти). Газової техніки, виробництва хімічне виробництво [37].

У завданнях управління, що вимагають реального часу, на перше місце висувається такий параметр, як тривалість циклу шини. Реалізація протоколу PROFIBUS-DP дає збільшення продуктивності шини (наприклад, для передачі 512 біт даних, розподілених по 32 станціям, потрібний всього 6 мс). Протокол PROFIBUS-DP підтримується пристроями різних виробників (рис. 2.2) [38].

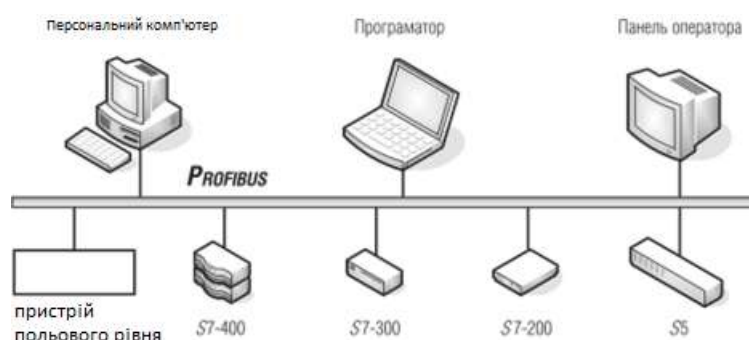


Рисунок 2.2 - Контролери в мережі Profibus-DP

Одним з основних критеріїв при виборі контролера для автоматизації є наявність програмного забезпечення, здатного ефективно реалізовувати поставлені користувачем завдання. Для ефективної реалізації поставленої задачі передусім необхідна швидка розробка програмних додатків для ПЛК та наявність готових рішень стандартних функцій контролю та управління.

Успіх програмованих логічних контролерів швидко призвів до розколу на одноканальні програмовані цифрові контролери та багатоканальні програмовані цифрові контролери. У програмуванні мова функціональних блоків повторює інженерні прийоми створення систем керування, які раніше використовувалися під час з'єднання окремих електронних або пневматичних функціональних блоків для отримання більш складної функціональності мають спрощену систему команд, випускаються серійно і досить дешеві порівняно із вартістю розроблення та виготовлення пристрою керування з жорсткою логікою. Але як пристрої керування технологічного устаткування програмовані логічні контролери значного поширення не знайшли, оскільки були спочатку громіздкими, все ж таки досить дорогими і для багатьох можливих застосувань [39].

Принцип адаптації універсальної мови програмування був простим - в мову вводилися нові типи змінних - «входи» і «виходи» контролера, а також створювалася бібліотека специфічних підпрограм, що реалізують найпоширеніші алгоритми.

Незважаючи на те, що програмувати стало легше, звичайним інженерам і «автоматчикам» все ще було важко.

Це створення процедурної, спеціалізованої технологічної мови, заснованої на загальній мові програмування, спеціально спрямованої на кінцевих користувачів об'єкта.

Основна відмінність текстових технологічних мов від мов загального призначення полягає в тому, що, з одного боку, значно спрощені синтаксис і семантика (зменшена різноманітність операторів, складність виразів), а з іншого боку, це введення в мову спеціальних технічних концепцій, які реалізують типові функції контролю та управління [40].

У зв'язку з цим відкрився величезний простір для творчості, і кількість таких мов на першому етапі майже зрівнялася з кількістю компаній, які бажають їх використовувати.

З врахуванням викладеного вище та тих задач, які повинні вирішуватися у системі віддаленого керування доступом, приходимо до її загальної структурної схеми. Система складається з: додатку адміністратора, бази даних, пристрою ідентифікації та надання доступу, смартфона як носія ідентифікатора. Додаток адміністратора являє собою одну частину програмного забезпечення. Він може працювати як на локальному сервері (відразу на комп'ютері власника чи адміністратора), так і на хмарному WEB-сервері. Додаток забезпечує контроль над пристроєм, взаємодію з базою даних клієнтів та запитом доступу в системи.

Смартфон є носієм ідентифікатора, через мережу інтернет згенерований запит на відправляється доступу клієнту. Так при потребі потрапити до приміщення клієнту необхідно лише

Так для отримання доступу необхідно відправити запит на мобільному пристрої. Основними структурними блоками пристрою є: мікроконтролер, база даних, Wi-Fi модуль, блоки комутації та індикації.

Основним структурним блоком пристрою, що забезпечує його функціонування, є мікроконтролер. Основними завданнями мікроконтролера є ідентифікація клієнта та надання доступу. Обмеження доступу забезпечується електромеханічним замком, керування яким здійснюється мікроконтролером через блок комутації. Ключ доступу у вигляді запиту, який використовується для ідентифікації запиту співробітника, записується в пам'ять мікроконтролера дистанційно через Інтернет. Обмін даними між мікроконтролером та додатком адміністратора, що формує запит, здійснюється за моделлю клієнт-сервер через точку доступу бездротової мережі Wi-Fi відповідно до протоколу TCP/IP [41].

Організації обміну даними між мікроконтролером та точкою доступу через бездротове Wi-Fi з'єднання забезпечується за допомогою Wi-Fi модуля, що реалізує обмін повідомленнями по радіоканалу відповідно до стандарту 802.11x. Wi-Fi модуль виконує функції прямого та зворотного перетворення цифрових даних у радіохвилю, управління доступом до радіоканалу, передачу та прийом пакетів даних. Використання спеціалізованого Wi-Fi модуля, що, підтримує стек протоколу TCP/IP, дозволяє суттєво спростити реалізацію взаємодії з пристроєм

через Інтернет, зводячи обмін інформації між віддаленим додатком та мікроконтролером до простого обміну даними між мікроконтролером та Wi-Fi модулем.

Надання прав доступу здійснюється за опрацювання запиту користувачем, який використовується як можливість відчинити замок. Рішення про дозвіл на доступ приймається мікроконтролером за інформацією, що представлена підтвердженням запиту до доступу. Отримання даних, що зашифровані, здійснюється мікроконтролером за допомогою прирівнення даних. Апаратно-програмне забезпечення зчитувача перетворює електричний імпульс у послідовність двійкових слів, які зчитуються мікроконтролером.

Доступ надається за допомогою електромеханічного замка, керування яким здійснюється мікроконтролером через блок комутації. Основним завданням блока комутації є перетворення сигналів, що формуються мікроконтролером, у сигнали, що забезпечують спрацювання замикаючого елемента замка.

Основним елементом комутаційного блоку є напівпровідниковий або електромагнітний комутаційний елемент. Твердотільні або напівпровідникові пристрої в основному включають резистори та сімістори. Робоча напруга забезпечують від 24 В до 480 В постійного або змінного струму дозволяє безконтактно перемикає навантаження.

2.3. Напрямки та задачі оптимізації

Процес виконання програми з конкретними наборами даних може бути представлений динамічною структурою програми, тобто у тому порядку виконання команд, як його оптимізує процесор разом з транслятором. Обмеженнями при оптимізації є залежність і по командах, і по даних. При виконанні програми процесор просувається по ній за допомогою вікна виконання. Якщо команди, які попали у вікно, є незалежні, то вони можуть виконуватись паралельно. Для усунення залежностей, викликаних командами переходів, використовується метод передбачення, який може викликати у конвеєр команд

та умовно виконувати команди передбачених переходів. Якщо передбачення зроблено вірно, то результати виконуваних команд враховуються, якщо ж ні, то стан процесора відновлюється на момент прийняття рішення про виконання команд.

Команди у вікні виконання можуть також бути залежними по даних, що обумовлено використанням тих самих регістрів, комірок пам'яті або результатів попередніх команд. Деякі з цих залежностей можна усунути, використовуючи додаткові ресурси процесора, після чого команди можна виконувати паралельно. Основні блоки суперскалярного процесора – це блок вибірки команд і передбачення переходів, блок декодування команд, аналізу залежності між командами, перейменування регістрів, диспетчеризація, блоки регістрів та оброблювальних пристроїв з плаваючою та фіксованою точками, блок керування пам'яттю, а також блок упорядкування виконаних команд.

Роздільні багаторівневі кеш-пам'яті даних та команд забезпечують одночасне введення у паралельно працюючі функціональні пристрої процесора кількох команд за один такт; у кеш-пам'яті введені засоби передбачення переходів. Одні засоби передбачення застосовують інформацію з двійкового коду команд або вироблену компілятором. Так, певні коди операцій частіше викликають розгалуження ніж інші, або розгалуження більш вірогідне, наприклад, у циклах, або компілятор у процесі перетворення програми у машинні коди виставляє прапорець, який установлює напрямок переходу. Може використовуватись також статистична інформація, отримана під час трасування програми. Інші засоби передбачення використовують інформацію щодо історії розгалуження, яка запам'ятовується у таблицях розгалужень та передбачень. Якщо передбачення виявилось невірним, результати команд, які були умовно (спекулятивно) виконані, анулюються.

Після декодування команд створюються групи даних щодо виконуваної операції, адрес операндів та адреси розміщення результату. На наступному кроці відбувається відображення логічних ресурсів, яку потребує програма, на фізичні ресурси мікропроцесора. Якщо один логічний ресурс відображається на кілька

фізичних ресурсів, кожний з яких відповідає значенню логічної величини в один з моментів послідовного виконання програми, і команда створює нове значення для логічного ресурсу; фізичний ресурс, в якому розміщується це значення, отримує ім'я. Наступні команди, які потім це значення використовують, сповіщаються відносно імені фізичного ресурсу, частіше за все це стосується перейменування регістрів. Цей прийом усуває залежність різних команд від даних.

У суперскалярних процесорах послідовні програми розбиваються на фрагменти, в яких команди виконуються паралельно. Після заповнення конвеєра командами між ними встановлюються тільки необхідні залежності по даних. Ефективне використання суперскалярних процесорів обмежується двома обставинами.

По-перше, через наявність умовних переходів паралелізм обчислювань на рівні команд є обмежений. Розмір вікна виконання також обмежує можливий притаманний програмі паралелізм, тому що зовсім не розглядається паралельне виконання команд, які знаходяться за межами вікна.

По-друге, складність суперскалярного процесора зростає швидше, ніж кількість паралельно виконуваних команд. Процесори VLIW використовують задання у командному слові сукупності паралельно виконуваних команд. Підготовка таких програм виконується компілятором. Перевагою VLIW є наявність програмних засобів, які більш ефективно, ніж апаратні, обмежені розміром вікна виконання, можуть аналізувати залежності між командами і вибирати паралельно виконувані команди. Крім того, що є важливим, VLIW-процесор має більш простий пристрій керування і потенціально може мати більш високу тактову частоту.

Недоліком VLIW є також обмеження його продуктивності за рахунок обмеженості вікна виконання, вже реалізованого програмно, та проблеми з умовними переходами. Представниками процесорів типу VLIW є процесори фірми Transmeta (МП типу Crusoe моделей TM3120, TM5400, TM5600 з

тактовою частотою 100 МГц), Intel (МП Itanium, 800 МГц) і Hewlett-Packard – модель McKinley [42].

Перспективними є також мультискалярні процесори, які розбивають послідовний потік команд на задачі, і забезпечують більшу глибину передбачення та високу вірогідність вибору правильного напрямку обчислень, а також більш широке вікно виконання. Мультискалярний процесор схожий на багатопроцесорну систему з розподілюваною пам'яттю і не вимагає ніяких апріорних знань щодо зв'язків команд з керування та даним. Розпаралелювання задач потребує використання компіляторів з мов високого рівня, які їх розпаралелюють. На ринку мікропроцесорів суперскалярні мікропроцесори є лідерами. При виборі цих процесорів для розв'язання задач у проблемній області, для якої створюється обчислювальна система, слід перевіряти адекватність прийомів підвищення продуктивності розв'язуванням задачам.

Архітектура 64-розрядних процесорів не є ані 64-розрядним розширенням архітектури CISC, ані переробленням архітектури RISC. Це нова архітектура, яка для забезпечення більшого паралелізму виконання команд використовує довгі слова команд (LIW), предикати команд, попереднє завантаження даних, сумісність на етапі декодування команд VLIW та CISC. Для цього операційні системи повинні мати підтримку як 64-розрядних додатків, так і 32-розрядних.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Розробка-бази даних

База даних (БД) — це організована за певними правилами та збережена в пам'яті комп'ютера сукупність інформації про об'єкти, процеси, події чи явища, пов'язані з певною галуззю, певною темою чи завданням.

Основна причина складності проектування бази даних полягає в тому, що об'єкти реального світу та зв'язки між ними не повинні мати і, як правило, не мають структури, узгодженої з реляційною моделлю даних. У процесі проектування розробник повинен забезпечити представлення реальних об'єктів та їхніх зв'язків у формі таблиць, полів, властивостей, записів тощо, тобто як абстракцію зв'язку моделі даних. Тому в цьому контексті термін «дизайн» можна розуміти як процес, результатом якого є проект, так і процес, результатом якого є проект.

Ефективна розробка бази даних складається з кількох етапів. Першим етапом розробки є аналіз вимог. Проектувальник повинен знайти відповіді на наступні питання: які елементи даних слід зберігати, хто повинен мати до них доступ і як [43]. Проаналізувавши всі данні і зрозумівши, які данні необхідні для поставленої задачі для персоналізації співробітників на підприємстві. Для цього необхідно створити таблиці які будуть мати їх персональні данні, а також інформації. про отримання, або ж заборони для доступу для користувачів. Також зважаючи на це, можливо зробити і таблицю запитів співробітників, дана таблиця залишає данні про запити, час цього запиту, та надання/заборони їм доступу.

На другому кроці створюється логічна структура бази даних. Для цього визначте, як дані будуть логічно згруповані. На цьому етапі структура бази даних представлена в термінах прикладних об'єктів і зв'язків між ними.

На останньому (третьому) кроці логічна структура бази даних перетворюється на фізичну структуру з урахуванням аспектів продуктивності.

На цьому етапі елементи даних містять атрибути та ідентифікуються як стовпці в таблицях СУБД, вибраних для реалізації.

Реляційна база даних — це набір взаємопов'язаних таблиць, по одній містить інформацію про об'єкти певного типу. Кожен рядок таблиці містить дані про об'єкт (наприклад, автомобіль, комп'ютер, клієнт), а стовпці таблиці містять різні характеристики цих об'єктів - атрибути (наприклад, номер двигуна, марка процесора, номер телефону, номер компанії або клієнта)

В БД повинні зберігатися дані, логічно пов'язані між собою. Для того щоб дані можна було пов'язати між собою, і зв'язати так, щоб ці зв'язки відповідали реально існуючим в даній предметній області, останню піддають детальному аналізу, виділяючи сутності чи об'єкти.

Сутність або об'єкт - це те, в чому повинна зберігатися інформація. Сутності з певними характеристиками, які називаються атрибутами, також повинні зберігатися в базі даних. Коли ми визначили сутності та їхні атрибути, ми повинні продовжити виявлення зв'язків, які можуть існувати між певними сутностями.

Зв'язок - це те, що об'єднує дві або більше сутностей. Відносини між сутностями також є частиною даних і також повинні зберігатися в базі даних.

Для роботи з даними використовується система управління базами даних (СУБД). Основні функції СУБД:

1. Визначення даних (описує структуру бази даних);
2. Обробка даних;

3. Управління даними. Розробка структури бази даних є найважливішим завданням, яке вирішується в процесі проектування бази даних. Структура бази даних (сукупність, форма зв'язку її таблиць) є одним з основних проектних рішень при створенні додатків, що використовують базу даних. Створена розробником структура бази даних описана за допомогою мови визначення даних СУБД.

Будь-яка СУБД дозволяє виконувати такі операції з даними:

1. Додавати записи до таблиці;

2. Видалити записи з таблиці;

3. Оновлювати значення певних полів в одному або кількох записах таблиці бази даних;

4. Знайди всі записи, які задовольняють певну умову.

Для виконання цих операцій використовується механізм запитів. Результатом виконання запиту є набір записів, відібраних за певними критеріями або змінами в таблиці. Запити до бази даних виконуються за допомогою мови, створеної спеціально для цієї мети, яка називається Structured Query Language (SQL) [44].

Під керуванням даними звичайно розуміють захист даних від несанкціонованого доступу, підтримку режиму роботи багатьох користувачів з даними і забезпечення цілісності і узгодженості даних. Забезпечення захисту і неможливості отримання несанкціонованого доступу незважаючи на те, що в будь-якій мові програмування є тип даних, масив, що виконує збереження великих блоків даних і можуть інтерпретуватись, як найпростіша база даних. Однак це не зручно, вносити зміни в код для додавання в масив даних, код цієї програми буде зовсім небезпечний, так як користувачі отримавши дані до контролю кодом можуть слідувати і задати свої властивості, використавши вразливості вводу кейсів отримати незаконні дані, що не розуміється системою.

Проектована база даних повинна володіти певними властивостями:

1. Цілісність бази даних досягається внаслідок введення обмежень, пов'язаних з нормалізацією БД. У кожен момент часу існування БД відомості, що містяться в ній, повинні бути несуперечливі.

2. Відновлювання - можливість відновлення БД після збою системи або окремих видів поломок системи.

3. Безпека - припускає захист даних від навмисного і ненавмисного доступу, модифікації або руйнування.

4. Ефективність. Властивість ефективності зазвичай оцінюється двома параметрами: мінімальний час реакції на запит користувача і мінімальні потреби в пам'яті, а так же поєднанням цих параметрів.

Сам процес розробки бази даних є складним процесом, який передбачає розробку відображення опису домену на внутрішню схему моделі даних.

Потік цього процесу — це серія простіших процедур для розробки менш складних відображень. Ця послідовність у процесі проектування постійно вдосконалюється та вдосконалюється, щоб ідентифікувати об'єкти, їхні властивості та зв'язки були необхідні майбутнім користувачам системи.

Етапи розробки бази даних у середовищі MySQL і Microsoft які ми будемо реалізовувати:

- ☐ 1. Визначення таблиць, які повинна містити база даних;
- ☐ 2. Визначення структури таблиць (полів та їх типів);
- ☐ 3. Призначення ключів таблиць та створення потрібних індексів;
- 4. Визначення зв'язків між таблицями;
- 5. Завантаження даних;
- 6. Створення додаткових об'єктів бази даних: запитів, форм и звітів;

Таблиці складають основу бази даних, в якій зберігаються всі дані. Спочатку нам потрібно побудувати схему структури кожної таблиці. Структура таблиці визначається вмістом вихідних форм, запитів, звітів, які має отримати при використанні бази даних.

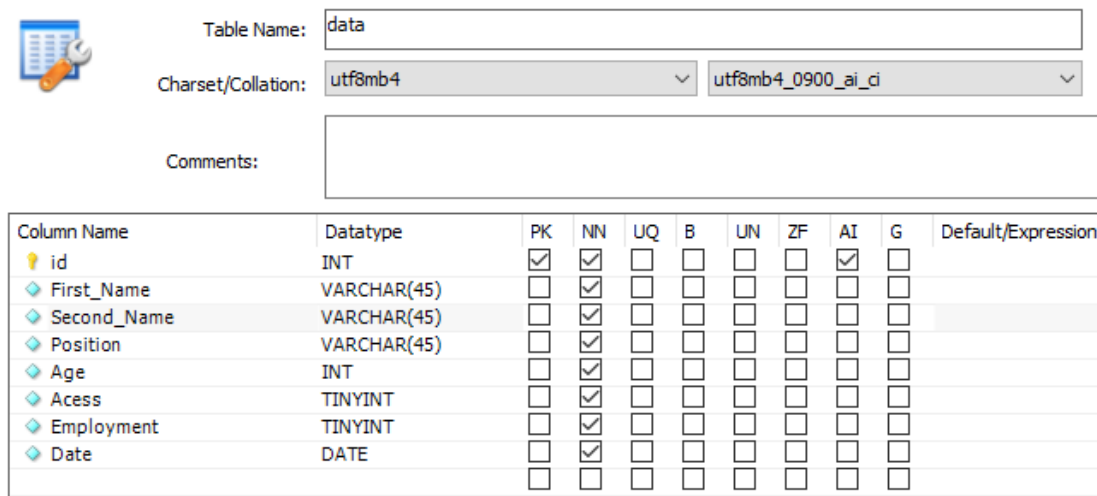
Під час створення таблиць необхідно уникати повторення стовпців у різних таблицях, лише якщо вони не використовуються для визначення зв'язків між ними. Таблиця складається із записів (рядків), кожен з яких описує сутність. Кожен стовпець у таблиці є полем.

Це поле містить однотипну інформацію, яка ідентифікує тип даних. Тип даних визначає тип і обмеження дійсних значень, які можна ввести в поле, а також обсяг пам'яті, виділеного цьому полю, що важливо при розробці великих баз даних.

Давайте створимо таблицю data з такими властивостями за допомогою команди SQL Create(рис. 3.1):

- Ідентифікаційний номер (id) - внутрішній ключ
- Ім'я співробітника(First Name)
- Прізвище співробітника(Second Name)
- Посада(Position)
- Права доступу(Acess)
- Працевлаштований (Employment)
- Дата працевлаштування(Date)

```
CREATE TABLE `basedata`.`data` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `First_Name` VARCHAR(45) NOT NULL,
  `Second_Name` VARCHAR(45) NOT NULL,
  `Position` VARCHAR(45) NOT NULL,
  `Age` INT NOT NULL,
  `Acess` TINYINT NOT NULL,
  `Employment` TINYINT NOT NULL,
  `Date` DATE NOT NULL,
  PRIMARY KEY (`id`));
```



The screenshot shows a database management tool interface. At the top, there is a 'Table Name' field containing 'data'. Below it, 'Charset/Collation' is set to 'utf8mb4' and 'utf8mb4_0900_ai_ci'. A 'Comments' field is empty. Below these fields is a table listing the columns of the 'data' table.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id	INT	☒	☒	☐	☐	☐	☐	☒	☐	
 First_Name	VARCHAR(45)	☐	☒	☐	☐	☐	☐	☐	☐	
 Second_Name	VARCHAR(45)	☐	☒	☐	☐	☐	☐	☐	☐	
 Position	VARCHAR(45)	☐	☒	☐	☐	☐	☐	☐	☐	
 Age	INT	☐	☒	☐	☐	☐	☐	☐	☐	
 Acess	TINYINT	☐	☒	☐	☐	☐	☐	☐	☐	
 Employment	TINYINT	☐	☒	☐	☐	☐	☐	☐	☐	
 Date	DATE	☐	☒	☐	☐	☐	☐	☐	☐	

Рисунок 3.1 - Дані таблиці data

Дана таблиця містить необхідні дані співробітників, такі як Ім'я (First Name) та Прізвище(Second Name), ці характеристики в подальшому будуть використовуватись для співставлення та персоналізації співробітників на підприємстві. Далі вносяться дані, посаду(Position) на яку спеціальність працевлаштувався співробітник, необхідні дані використовуються для розмежування доступу персоналу, тобто використовуються деякі умови характеризуючи працівників, кому можливо надати доступ. Вік(Age) є біометричними даними користувача, призначенням якого можуть використовуватись для розмежування співробітників по їхньому віку. Наступні дані вказують, що наданий доступ(Access) даному працівнику, дана властивість приймає, значення опера булевого типу, приймаючи два можливих значення, "1" та "0", що характеризує доступ до даного замка працівником. Також в цій таблиці є значення працює цей працівник(Employment) зараз. Також можливо реалізувати функцію, якщо працівник уже не працює, то воно буде видаляти всі дані пов'язані з ним. Останнім даними таблиці буде дата коли був добавлений працівник(Date) в базу даних

Наступної створимо таблицю запис про отримані запити(request_log) з наступними атрибутами, так само за допомогою SQL-команди Create(рис. 3.2):

- Ідентифікаційний номер (id) - внутрішній ключ
- Ім'я співробітника(First Name)
- Прізвище співробітника(Second Name)
- Права доступу(Acess)
- Час отримання запиту (Date)

```
CREATE TABLE `basedata`.`request_log` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `First Name` VARCHAR(45) NOT NULL,
  `Second Name` VARCHAR(45) NOT NULL,
  `Acess` TINYINT NOT NULL,,
  `Time` TIME NOT NULL,
  PRIMARY KEY (`id`));
```

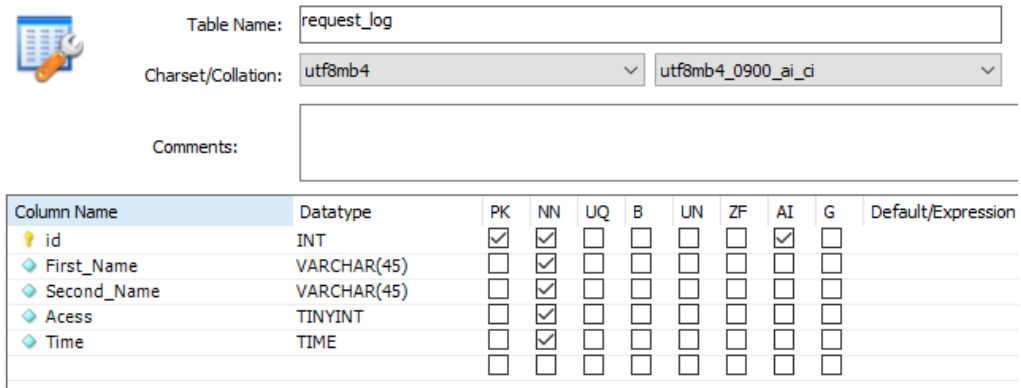


Table Name: request_log

Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☒	☒	☐	☐	☐	☐	☒	☐	
First_Name	VARCHAR(45)	☐	☒	☐	☐	☐	☐	☐	☐	
Second_Name	VARCHAR(45)	☐	☒	☐	☐	☐	☐	☐	☐	
Acess	TINYINT	☐	☒	☐	☐	☐	☐	☐	☐	
Time	TIME	☐	☒	☐	☐	☐	☐	☐	☐	

Рисунок 3.2 - Таблиця request_log

Дана таблиця містить необхідні дані співробітників, такі як Ім'я(First Name) та Прізвище(Second Name), ці характеристики використовуються для демонстрації який співробітників має доступ з замком на підприємстві. Наступні дані вказують, що наданий доступ(Access), що характеризує отримання доступу працівником. А також аргумент Часу(Time) характеризує коли був відправлений запит.

Також існують дві додаткові таблиці, які містять властивості для працівника і запис про отримані запити, як наприклад список назв на яку спеціальність(position) працевлаштування і створимо таблицю за допомогою SQL-команди Create(рис. 3.3).

```
CREATE TABLE `basedata`.`position` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Position` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`));
```

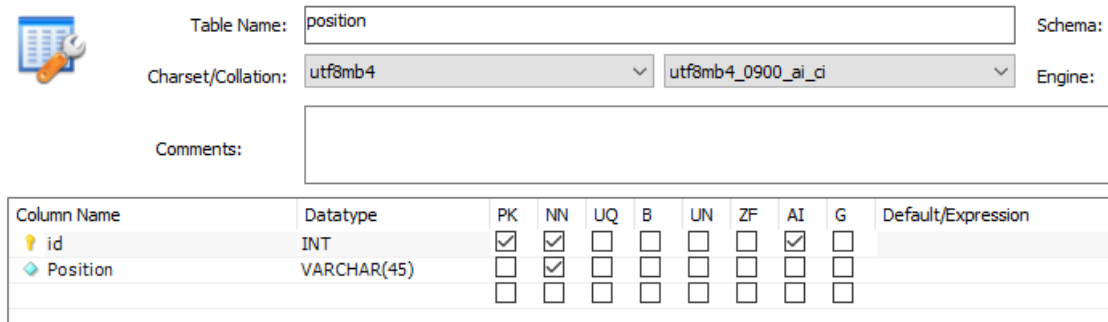


Table Name: position Schema:

Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☒	☒	☐	☐	☐	☐	☒	☐	
Position	VARCHAR(45)	☐	☒	☐	☐	☐	☐	☐	☐	

Рисунок 3.3 - Властивості спеціальностей

Остання таблиця яка буде містити властивості о статусі запиту(meaning) створимо таблицю(рис. 3.4).

```
CREATE TABLE `basedata`.`meaning` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `Acess` TINYINT NOT NULL,
  `Comment` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`));
```

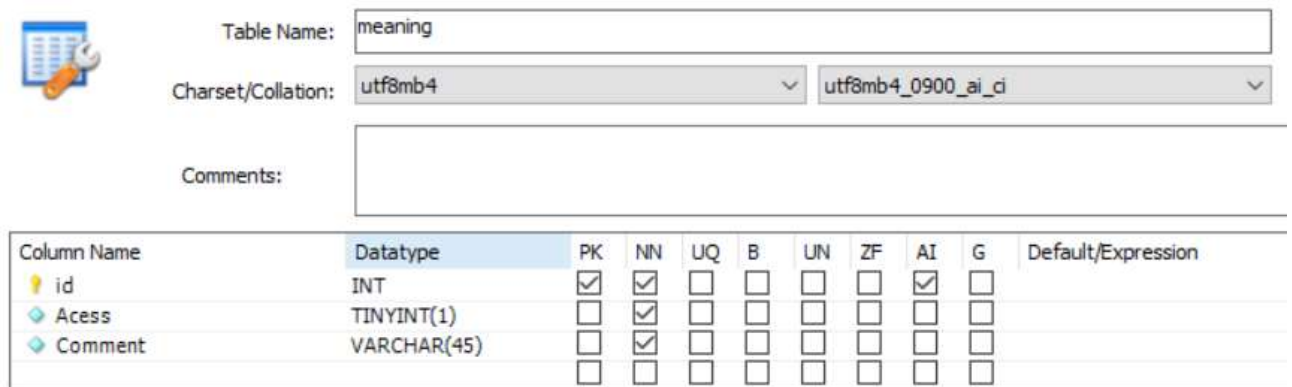


Table Name: meaning

Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☒	☒	☐	☐	☐	☐	☒	☐	
Acess	TINYINT(1)	☐	☒	☐	☐	☐	☐	☐	☐	
Comment	VARCHAR(45)	☐	☒	☐	☐	☐	☐	☐	☐	

Рисунок 3.4 - Властивості таблиці meaning

Результатом створення таблиць є наступна діаграма(рис. 3.5)

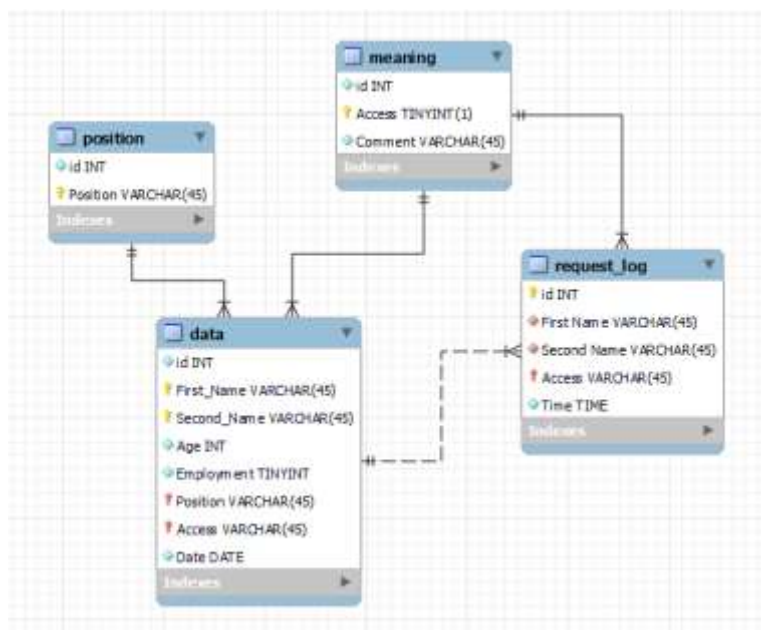


Рисунок 3.5 - Діаграма таблиць даних

У процесі проектування бази даних був розроблений ряд запитів, що забезпечує пошук і сортування інформації, що зберігається в базі даних. Ці запити надають можливість в зручній формі переглянути інформацію про клієнтів, співробітників, постачальників, товарах.

Було продемонстровано запит в бази даних, які спеціальності знаходяться в даній організації, що дасть повний аналіз всіх можливих співробітників(рис. 3.6)

```
SELECT * FROM basedata.position;
```

	id	Position
▶	1	Головний
	2	Адміністратор
	3	Бухгалтер
	4	Вантажник
	5	Різнорабочий
	6	Охоронець
*	NULL	NULL

Рисунок 3.6 - Аналіз всіх спеціальностей в організації

Також необхідно продивитись хто може отримати доступ(рис. 3.7) після відправки запиту, дана команда буде працювати з пошуком атрибута Access, який відповідає за цю властивість.

```
SELECT id, First_Name, Second_Name, Age, Access, Employment, Date
FROM basedata.data
Where Access = 1;
```

	id	First_Name	Second_Name	Age	Position	Access	Employment	Date
▶	1	Владислав	Гаркавий	22	Адмістратор	1	1	2023-11-30
	3	Есфір	Мацько	23	Головний	1	1	2023-03-04
	4	Живорід	Долженко	43	Вантажник	1	1	2023-01-02
	6	Шерлок	Караванський	19	Охоронець	1	1	2023-04-12
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 3.7 - Співробітники що мають доступ

Як можна бачити, дані співробітники часто взаємодіють з цим пристроєм і знаходяться в цьому приміщенні, однак щоб підтвердити дану думку можемо перевірити, хто не має необхідного доступу(рис. 3.8).

```
SELECT id, First_Name, Second_Name, Age, Position, Access, Employment,
Date FROM basedata.data
Where Access = 0;
```

	id	First_Name	Second_Name	Age	Position	Access	Employment	Date
▶	2	Нігослав	Щербацький	25	Бухгалтер	0	1	2023-07-03
	5	Мстибог	Циганик	32	Різноробочий	0	1	2023-07-23
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 3.8 - Співробітники що не мають доступу

Так думка має пряме підтвердження, даний персонал знаходиться в другому корпусі і ніяк не взаємодіє.

Також можна створити тригер для таблиці request_log, що буде автоматично при отриманні запиту базою даних буде виставлений системний час.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `request_log_AFTER_INSERT`  
AFTER INSERT ON `request_log` FOR EACH ROW BEGIN  
Insert into request_log(Time)  
values (CURRENT_TIME());  
END
```

3.2 Програмне забезпечення клієнтської системи

Клієнт-сервер і дозволить вам краще зрозуміти, як організовано взаємодію яка використовує сучасні технології, присутні елементи клієнт-серверного взаємодії. З однорівневої архітектури, вона стає дворівневою, через те що між клієнтом та сервером необхідний бути розподіл трьох основних компонентів, взаємодію з користувачем, бізнес логіку та управління ресурсами(рис. 12). Ця модель завжди була предметів суперечок серед розробників ПЗ, що стосуються в, першу чергу, розділення функцій компонентів системи між клієнтом і сервером. Зазвичай чіткої відмінності немає. Наприклад, сервер РБД може виступати клієнтом, який передає запити на файлові сервери, що відповідають за реалізацію таблиць цієї БД. Проте, розглядаючи різні застосування типу клієнт-сервер, для роботи з БД, рекомендують розділяти їх на три логічні рівні:

1. рівень інтерфейсу користувача реалізується на клієнті;
2. рівень обробки реалізується основна логіка компонента (функціональність);
3. рівень даних. Рівень інтерфейсу. Зазвичай особливою властивістю цього рівня є вимога збереження (persistence). Це означає, що коли застосування не працює, дані повинні зберігатися у визначеному місці (файлі або БД) для

подальшого використання. Цей рівень зазвичай реалізується на сервері(рис. 3.9) [45].

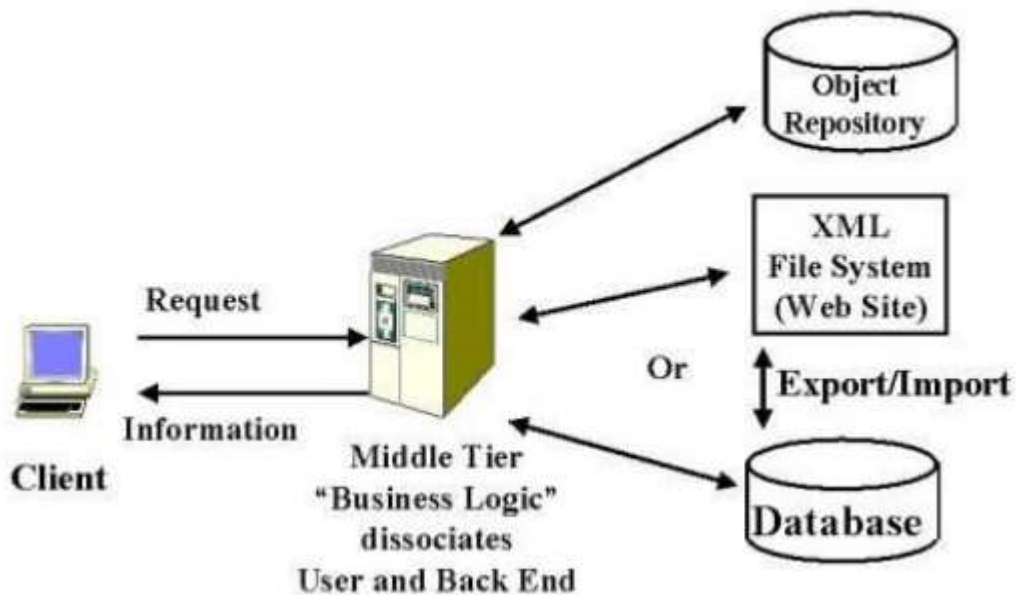


Рисунок 3.9 - Дворівнева архітектура клієнт-сервер”

Із назви зрозуміло, що в цій концепції беруть участь дві сторони: клієнт і сервер. Тут все як у житті: клієнт – це замовник тієї чи іншої послуги, а сервер – постачальник послуг. Клієнт і сервер фізично являють собою програми, наприклад, типовим клієнтом є браузер.

Серед серверів можна навести такі приклади:

- Веб-сервери, наприклад, Tomcat.
- Сервери баз даних, наприклад, MySQL.
- Платіжні шлюзи, наприклад, Stripe.

Клієнт із сервером зазвичай спілкуються через інтернет, хоча можуть працювати і в одній локальній мережі і в будь-яких інших типах мереж. Спілкування відбувається за стандартними протоколами, такими як HTTP, FTP або більш низькорівневими, такими як TCP або UDP [46].

Протокол зазвичай обирається під тип послуги, яку надають сервери. Наприклад, якщо це відеозв'язок, зазвичай використовується UDP. Існує 7 видів протоколів таких як:

1. Фізичний — середовище, де здійснюється обмін даними, на цьому рівні трансформуються в бінарний код електричні імпульси, які далі передаються на більш високі рівні (на даному рівні функціонують медіаконвертери, сигнальні ретранслятори, хаби);
2. Канальний — рівень, на якому дані передаються на хост з метою обробки, а щоб ідентифікувати інформацію, застосовується MAC-адреси;
3. Мережевий — актуалізуються IP-адреси, завдяки яким в інтернеті ідентифікуються користувачі, дані надходять пакетами;
4. Транспортний — в обов'язковому порядку здійснюється доставка пакетів до адресатів, протокол відстежує цілісність і коректність донесення інформації, для цих цілей використовуються алгоритми фрагментування або об'єднання;
5. Сесійний — протоколи забезпечують підтримку мережевого сеансу, синхронність початку і кінця з'єднання, а також перевіряють дозволи на доступ;
6. Репрезентативний (рівень представлення) — отримана інформація декодується або кодується, файли розпаковуються або стискаються, тобто здійснюється переклад даних на рівень, який підійде конкретному браузеру (з додатком);
7. Прикладний — відбувається регулювання зв'язку користувачів і мережі, даються дозволи на доступ, реалізовується робота протоколів вищого рівня.

Правила взаємодії між клієнтом і сервером називаються протоколом обміну (протоколом взаємодії). Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

1. Рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

2. Прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;

3. Рівень управління даними, який забезпечує зберігання даних та доступ до них.

Основна ідея архітектури «клієнт-сервер» полягає в поділі мережевого додатку на кілька компонентів, кожен з яких реалізує специфічний набір сервісів. Компоненти такого додатку можуть виконуватися на різних комп'ютерах, виконуючи серверні і/або клієнтські функції. Це дозволяє підвищити надійність, безпеку і продуктивність мережевих додатків і мережі в цілому [47].

Серед сервери можна охарактеризувати наприклад баз даних, сервер зберігає в собі всі данні користувачів і їх характеристики, для подальшого аналізу і їх використання. Ці всі данні є можливість інтегрувати в програму, для організації бази даних MySQL необхідно в першу чергу підключитись до неї, а потім виконати ряд запитів відповідно з поставленим завданням [48].

Для ОС Linux Mint 19 після встановлення файл JAR можна виявити за адресою /usr/share/java/mysql-connector-java-8.0.28.jar (тут замість номера версії 8.0.28 може бути інший номер). Файл JAR можна залишити на місці встановлення або перенести у довільне інше місце. Наприклад, у теку проекту.

У будь-якому випадку встановлений файл JAR необхідно приєднати до проекту, що використовує SQL

```
SET collation_connection = 'utf8_unicode_ci';
CREATE DATABASE basedata;
ALTER DATABASE basedata CHARACTER SET utf8 COLLATE
utf8_unicode_ci;
USE basedata;
SOURCE request_log.sql;
SOURCE data.sql;
```

У цьому випадку програма мовою Java може мати такий вигляд:

```
package work;
import java.sql.*; // під'єднання бібліотеки підпрограм роботи з SQL
```

```

import java.util.*;
class Work
{ public static void main(String args[])
    { try { Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection
        ("jdbc:mysql://localhost:3306/basedata",      //
        "root",
        "root" );
        Statement stmt=con.createStatement();
        ResultSet rs=stmt.executeQuery("id, First_Name, Second_Name,
Age, Access, Employment, Date FROM basedata.data");
        while (rs.next()) System.out.println
        (rs.getString(1)+" "+rs.getString(2)+" "
        +rs.getString(3)+" "+rs.getString(4));
        con.close();
    }
    catch(Exception e) {System.out.println("Отримано виняток "+e);}
    }
}

```

Результат виведенням у консоль.

1,Владислав,Гаркавий,Адміністратор,22,1,1,2023-11-30

3,Есфір,Мацько,Головний,23,1,1,2023-03-04

4,Живорід,Долженко,Вантажник,43,1,1,2023-01-02

6,Шерлок,Караванський,Охоронець,19,1,1,2023-04-12

Даний список з бази даних прирівнюється в програмі с ведений значенням в строку.

Спочатку на екрані необхідно зробити інтерфейс для реагування користувачем і інтеграцію даних.

Програма доступу

Введіть назви Отримання доступу

Введіть прізвище

Час відправки запиту Отправка запросу

Рисунок 3.10 - Програма запиту доступу

Даний інтерфейс (рис.3.10) має в функціонал запис про отримані запити(request_log) з наступними атрибутами:

- Ім'я співробітника(First Name)
- Прізвище співробітника(Second Name)
- Права доступу(Acess)
- Час отримання запиту (Date)

Вона містить необхідні дані співробітників, такі як Ім'я(First Name) та Прізвище(Second Name), ці характеристики використовуються для демонстрації який співробітників взаємодіяв з замком на підприємстві, однак в кодї програми він використовується як прирівнення input строки з базою даних. Наступні дані вказують, що наданий доступ(Access), що характеризує сповіщення про отриманий доступ. А також аргумент Часу(Time) характеризує коли був відправлений запит.

Спочатку в кодї запиту було проведено ідентифікацію всіх елементів графічної структури(рис. 3.11).

```

5 usages
private TextView acessProtect;
2 usages
private EditText editTextText;
2 usages
private EditText editTextText2;
2 usages
private Button buttonRequest;
2 usages
private EditText timeRequest;

```

Рисунок 3.11 - Ініціалізація всіх елементів програмами

Дана ініціалізація відбувається за допомогою пошуку цих самих блоків в активній частині програми (active_main.xml), після пошуку воно присвоює йому змінну, за якою буде знаходитися ця ініціалізація.

Данні блоки мають різні властивості тобто:

TextView - основою функцію якого, є демонстрація тексту, який був заданий в текст, або ж програмно добавлений.

EditText - строка має в собі властивості вводу даних, використовується для отримання програмою необхідних змінних для співставлення.

Button - використовується для виконання алгоритмів, активується вони під час натискання кнопки, встановить прослуховування натискань для об'єкта кнопки.

Окрім знаходження блоків, важливою характеристикою об'єкта також є його id в системі, звертаючись на нього ми можемо отримати необхідні дані(рис. 3.12).

```

<TextView
    android:id="@+id/textView4"
    android:layout_width="233dp"
    android:layout_height="39dp"
    android:layout_marginStart="100dp"
    android:layout_marginTop="156dp"
    android:text="@string/programName"
    android:textSize="24sp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

Рисунок 3.12 - Об'єкт назви програми

Після цього користувачу необхідно знайти об'єкти інтерфейсу в програмі, даний пошук відбувається по id(рис. 3.13).

```

accessProtect = findViewById(R.id.accessProtect);
editTextText = findViewById(R.id.editTextText);
editTextText2 = findViewById(R.id.editTextText2);
buttonRequest = findViewById(R.id.buttonRequest);
timeRequest = findViewById(R.id.timeRequest);

```

Рисунок 3.13 - Пошук об'єктів

Робота програми відбувається за допомогою функції onClick, об'єкта button, після введення даних в строку для імені та прізвища, необхідно натиснути на кнопку “Отправка запису”, що далі буде відбуватись порівняння елементів бази даних, а також введені користувачем біометричні дані, в випадку коли данні співпадають, співробітнику надається доступ, щоб зрозуміти це на екрані замінюється елемент “отримання доступу”, на слово “Success” (Успіх), коли ж не невдача, результатом буде “Fail”. А також дата і час створення запиту (рис. 3.14).

```

buttonRequest.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String FirstName = editTextText.getText().toString();
        String SecondName = editTextText2.getText().toString();
        for (int i = 0; i < firstName.length; ) {
            if (FirstName.equals(firstName[i])) {
                acessProtect.setText("Success");
                break;
            } else if (!FirstName.equals(firstName[i])) {
                acessProtect.setText("Fail");
            }
            i++;
        }
        for (int c = 0; c < secondName.length; ) {
            if (SecondName.equals(secondName[c])) {
                acessProtect.setText("Success");
                break;
            } else if (!SecondName.equals(secondName[c])) {
                acessProtect.setText("Fail");
            }
            c++;
        }

        Date currentDate = new Date();

        timeRequest.setText(String.valueOf(currentDate));
    }
}

```

Рисунок 3.14 - Використання функції onClick

3.3 Тестування програмного забезпечення

Тестування програмного забезпечення відбувається для виявлення інформації про якість продукту відносно контексту, в якому його мають використовувати. В ході тестування працівник також буде виконувати процес пошуку помилок або інших дефектів, так і випробування програмних складових із метою оцінки [49].

В ході перевірки даного програми буде перевірено декілька видів тестів, а саме:

1. Функціональне тестування : чи робить мою програму те, що має?
2. Тестування продуктивності: чи робиться воно швидко та ефективно?

Перший тест відповідає на головне питання, чи виповнює він отримання запиту базою даних, опрацьовує запит і забирає данні з неї, де після прирівнюється з строчкою введення даних програми. Для того щоб протестувати програму її необхідно запустити, це відбувається за допомогою IDE Android Studio, в якому вибирається налаштування пристрою смартфону. Окрім можливості вибрати емульований телефон в програму, є також і фізичний, що прискорить активацію програми в декілька раз Також необхідно вибрати файл для перетворення коду, на зрозумілий для пристроїв(рис. 3.15).



Рисунок 3.15 - Інструменти для запуску програми

Після активації програми буде видано пусту форму в яку необхідно заповнити самотужки(рис. 3.16). За допомогою різних апаратних прискорень і правильного розподілення пам'яті, та використання виконування тільки необхідних команд із завантаженої в пам'ять програми. Майже всі ініціалізування процесора — це щось на кшталт узяти з деяких комірок

дані → щось із ними зробити → результат помістити в інші комірки, данні перезаписи в програмі трудомісткі і необхідно також і звільнювати зарезервовані блоки [50]. Дані ресурси допомогли в двічі швидше ініціалізувати скомпільований код в зрозумілий для комп'ютера набір команд, а після використання програми, дані зарезервовані блоки, будуть стиратись отримуючи більше ресурсів для подальшої роботи.

Програма доступу

Введіть назви

Отримання доступу

Введіть прізвище

Час відправки запиту

Отправка запросу

Рисунок 3.16 - Пуста форма отримання доступу

Після заповнення форми і активації кнопки опправки, запит успішно був виконаний(рис. 3.17)

Програма доступу

Владислав

Гаркавий

Sat Dec 02 11:21:35 GMT

Success

Отправка запросу

Рисунок 3.17 - Успішний запит

Також для перевірки подивимося в базу даних і вона демонструє що було успішно передано данні(рис. 3.18).

	id	First_Name	Second_Name	Acess	Time
▶	1	Владислав	Гаркавий	1	16:32:12
	4	Владислав	Гаркавий	1	11:21:35
✱	NULL	NULL	NULL	NULL	NULL

Рисунок 3.18 - База даних

Так даний тест є успішним, відбувається порівняння і отримання результатів, в ході виконання була зроблена успішна з базою даних організації, ця таблиця складалась з строкового значення імені та прізвище працівника, а також булеве значення його доступу до необхідного об'єкта, основою методою було фільтрація працівників і можливих третіх осіб, які незаконно хотіли отримати комерційні дані.

Також можливо перевірити інструментальні та локальні тести. Данні тести закладаються на тому, що як і де виконуються.

Програмна інтегрована середовище розробки для роботи Android програм можливе виконання як на пристрої фізичному так і емульованому. Фізичний пристрій виконується за допомогою USB-кабеля(рис. 3.19), та має свої переваги в швидкодії, даний метод відкривається в декілька раз швидше ніж протилежний, це відбувається через те, що на цей пристрій немає необхідності виділяти оперативну пам'ять комп'ютера, так як всі ресурси пересилаються на телефон для подальшої ініціалізації.

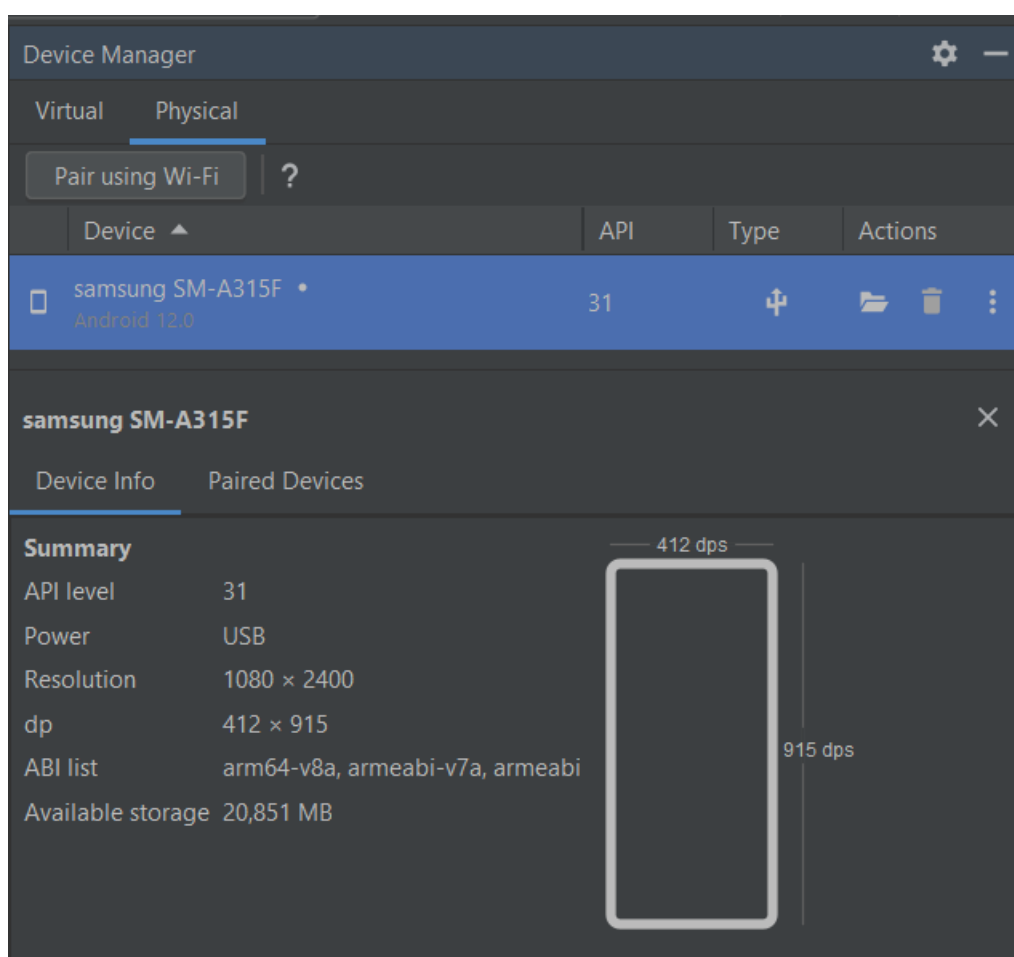


Рисунок 3.19 - Фізичний пристрій

Також більшої актуальності і розробки сучасних Android програм, так як важливою складовою є версія API. Однак незважаючи на це емульовані також

мають свої переваги, так як в них не має сталої версії Android(рис. 3.20) і можлива інсталяція любої для перевірки працездатності всіх можливих систем

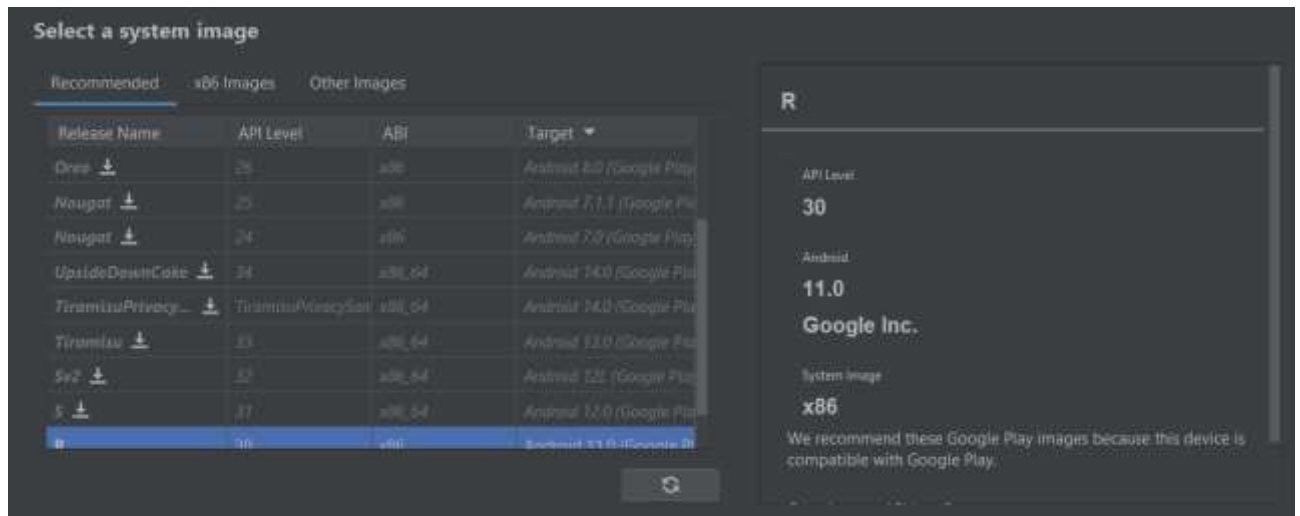


Рисунок 3.20 - Варіативність систем Android

ВИСНОВОК

В ході дипломної роботи було проаналізовано методи для розробки Android програм, їх ключові особливості, а також взаємодія з різними структурами і компонентами.

Також важливою складовою для виконання дипломної роботи, було аналіз особливостей структури їх програм, висвітлення головних задач, а також визначення їх структури. Що особливості важливим результатом, що було отримано в ході виконання дипломної роботи являється дослідження мікропроцесорів і їх особливості. Так як їх є велике різноманіття і можливе нецільове використання їх в системі, а також необхідний був якомога скоріший відгук. Однак крім його переваг, також було зроблено адаптований і оптимізований відгук системи.

Це було досягнуто, за допомогою розподілення пам'яті в програмі, перед виконанням поставленого завдання перед програмою їй виділялось біти зарезервованих даних для опрацювання. Після же виконання програми дані зарезервовані біти звільняються і надлишкова оперативна пам'ять переходить до загальної.

Для того щоб автоматизувати створення та аналіз бази даних, використовується за допомогою доступних методів створення таблиць їх взаємодії з нею, а також інтегрування в мову програмування.

З широким використанням застосування Інтернет технологій було збільшення актуальності і необхідності в даній технології, за допомогою її можна зменшити що створює передумови необхідності керування та управління віддалено доступом. Це особливо актуально для організацій, які використовуються його обмеження. Дає можливість розділяти права доступу в різні приміщення для співробітників, попереджає можливість несанкційованого проникнення до підприємства. Як наслідок підвищується якість обслуговування та безпека працівників, а також рівень контролю за найманим персоналом, що сприяє підвищенню ефективності його роботи та запобігає можливим

зловживанням, щоб зменшити і мінімізувати можливий допуск помилок персоналу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Система контролю і управління доступом [Електронний ресурс]. Режим доступу:
https://uk.wikipedia.org/wiki/Система_контролю_і_управління_доступом
2. Система контролю і управління доступом [Електронний ресурс]. Режим доступу: <https://vistplus.com/it-poslugi/skud/>
3. Види систем контролю та керування доступом. [Електронний ресурс]. Режим доступу: <https://karabiner.ua/stati/vidy-sistem-kontrolya-i-upravleniyadostupom-skud/>
4. Контроль роботи співробітників: види , методи , помилки [Електронний ресурс] - Режим доступу до ресурсу: <https://aboutmarketing.info/internet-marketynh/kontrol-roboty-spivrobitnykiv-vydy-metody-pomylyky/>
5. Перспективи автоматизації внутрішнього контролю [Електронний ресурс]. Режим доступу:
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKewjxn6SCz_OCAxWbYPEDHc6kB00QFnoECCYQAQ&url=https%3A%2F%2Fbiem.sumdu.edu.ua%2Fimages%2Fstories%2Fdocs%2FK_EK%2FYarovenko_2.pdf&usg=AOvVaw2ANTAHpnOhi0LeEQ7xDbIB&opi=89978449
6. Використання сучасних інформаційних технологій і програмних продуктів [Електронний ресурс]. Режим доступу:
https://osvita.ua/vnz/reports/econom_pidpr/21931/
7. Що таке прогресивні веб-додатки [Електронний ресурс] - Режим доступу до ресурсу: <https://smile-ukraine.com/ua/pwa/introduction>
8. Организация комплексной системы обеспечения безопасности предприятия [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.deltaguard.com.ua/uk/база-знань/організація-комплексної-системи-заб/>
9. Wise IT [Електронний ресурс] - Режим доступу до ресурсу:
https://wiseit.com.ua/about_us/

10. Переваги та недоліки Java [Електронний ресурс] - Режим доступу до ресурсу: <https://from-ua.info/perevahy-ta-nedoliky-java/>
11. Kotlin docs [Електронний ресурс] - Режим доступу до ресурсу: <https://kotlinlang.org/docs/home.html>
12. Переваги та недоліки C++ [Електронний ресурс] - Режим доступу до ресурсу: <http://informatikpm11.blogspot.com/p/c.html>
13. Why IntelliJ IDEA [Електронний ресурс] - Режим доступу до ресурсу: <https://www.jetbrains.com/idea/>
14. Д. Д. Татарчук, Ю. В. Діденко. – Мікропроцесори та мікроконтролери курс лекцій, 2020. – 238 с
15. Класифікація мікропроцесорів [Електронний ресурс]. Режим доступу: http://ni.biz.ua/9/9_7/9_72217_klassifikatsiya-mikroprotessorov.html
16. Дротові та бездротові локальні мережі - переваги та недоліки [Електронний ресурс]. Режим доступу: <http://alternativa.dp.ua/provodnye-i-besprovodnye-lokalnye-seti-preimushhestva-inedostatki/>
17. Wi-Fi або Ethernet: плюси і мінуси підключень [Електронний ресурс]. Режим доступу: <https://www.agsat.com.ua/ua/info/wi-fi-abo-ethernet-plyusy-minusy-pidklyuchen/>
18. Стандарт 802.1X IEEE [Електронний ресурс]. Режим доступу: <https://nettech.ua/news/802-1x-ieee>
19. Z-Wave vs ZigBee, WiFi, Thread, Bluetooth BLE: обираємо протокол управління розумним будинком. [Електронний ресурс]. Режим доступу: <https://xterm.com.ua/novosti/porivniannia-zigbee-ta-z-wave-shcho-obrati>
20. Бездротовий доступ WI-FI як складова телекомунікаційної мережі університету. [Електронний ресурс]. Режим доступу: <https://kpi.ua/809-7>
21. Огляд технології Wi-Fi - Network Tools. [Електронний ресурс]. Режим доступу: <https://ntools.com.ua/uk/information/faq/tehnologija-wi-fi>
22. Мережі Wi-Fi. Робота та стандарти. Застосування та особливості. [Електронний ресурс]. Режим доступу: <https://apps4business.com.ua/merezhi-wi-fi-robota-ta-standarti-zastosuvannya-ta-osoblivosti/>

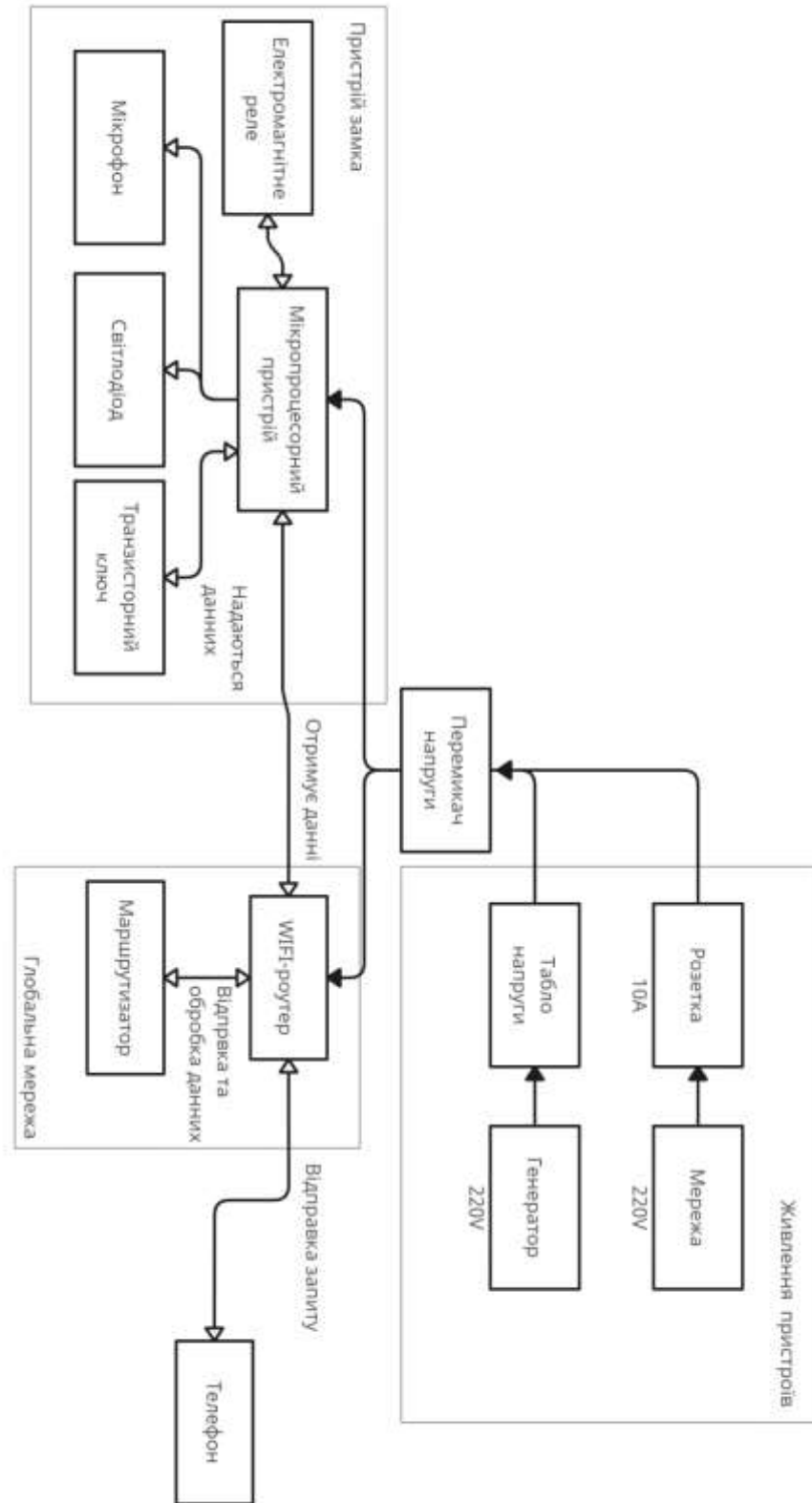
23. Контролери віддаленого управління – майбутнє вже сьогодні [Електронний ресурс]. Режим доступу: <https://novatek-electro.com/blog/kontroleri-viddalenogo-upravlinnya.html>
24. Що таке мікропроцесор, мікроконтролер та програмований логічний контролер. [Електронний ресурс]. Режим доступу: <https://naukam.triada.in.ua/index.php/konferentsiji/70-tridtsyat-dev-yata-vseukrajinska-praktichno-piznavalna-internet-konferentsiya/931-vibir-protokolu-obminu-danimi-mizh-pristroyami-promislovoji-avtomatiki>
25. Масштабована архітектура системи. [Електронний ресурс]. Режим доступу: <https://www.it.ua/knowledge-base/architecture-security/masshtabiruemaja-arhitektura-sistemy-it-enterprise>
26. Основні проблеми розумних будинків і як їх можна вирішити? [Електронний ресурс]. Режим доступу: <https://klaster.ua/ua/stati-i-obzory/osnovnye-problemy-umnyh-domov-i-kak-ih-mozhno-reshit/>
27. Мікросервісна архітектура: плюси та мінуси [Електронний ресурс]. Режим доступу: <https://blog.iteducenter.ua/articles/microservices-architecture-advantages-and-disadvantages/>
28. Profibus [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/Profibus>
29. Modbus [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/Modbus>
30. Fieldbus [Електронний ресурс]. Режим доступу: <https://www.auma.com/en/products/communication-systems/fieldbus/>
31. HART-протокол [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/HART-%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB>
32. А.М.Ніколаєнко Н.О.Міняйло - Мікропроцесорні та програмні засоби автоматизації, 2011, – 439с
33. Profibus [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/Profibus>

34. Архітектура та проектування компонентних систем [Електронний ресурс].
Режим доступу: [https://victana.lviv.ua/knyhy/konspekty-lektsii/133-kros-platformenne-prohramuvannia-ta-khmarni-servisy/568-lektsiia-20-arkhitektura-ta-proektuvannia-komponentnykh-system-2016-r#:~:text=20.2.1.-,Дворівнева%20архітектура%20клієнт-сервер,\(two-tiered%20archite](https://victana.lviv.ua/knyhy/konspekty-lektsii/133-kros-platformenne-prohramuvannia-ta-khmarni-servisy/568-lektsiia-20-arkhitektura-ta-proektuvannia-komponentnykh-system-2016-r#:~:text=20.2.1.-,Дворівнева%20архітектура%20клієнт-сервер,(two-tiered%20archite)
35. Міжмережева взаємодія у промислових мережах [Електронний ресурс].
Режим доступу: <http://www.picad.com.ua/0304/pdf/6-11.pdf>
36. Системи автоматизації і управління технологічними процесами [Електронний ресурс]. Режим доступу: http://4ua.co.ua/manufacture/qa2ac68b4c43a89521316c36_1.html
37. Мережа Profibus [Електронний ресурс]. Режим доступу: <http://um.co.ua/12/12-3/12-36519.html>
38. Комунікаційні можливості контролерів [Електронний ресурс]. Режим доступу: https://stud.com.ua/160486/tehnika/komunikatsiyni_mozhливosti_kontroleriv
39. Програмуємі логічні контролери. Монтаж, налагодження та експлуатація ПЛК в умовах підприємства [Електронний ресурс]. Режим доступу: https://studwood.net/1151964/tovarovedenie/programuyemi_logichni_kontroleri_montazh_nalagodzhennya_ekspluatatsiya_umovah_pidpriyemstva
40. Мова моделювання [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Мова_моделювання
41. TCP/IP [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/TCP/IP>
42. VLIW [Електронний ресурс]. Режим доступу: <https://uk.wikipedia.org/wiki/VLIW>
43. Основні етапи проектування баз даних [Електронний ресурс]. Режим доступу: <https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture01>

44. Основні відомості про запити [Електронний ресурс]. Режим доступу: <https://support.microsoft.com/uk-ua/office/основні-відомості-про-запити-a9739a09-d3ff-4f36-8ac3-5760249fb65c>
45. Як створюється інтерфейс мобільних додатків[Електронний ресурс]. Режим доступу: <https://wezom.com.ua/ua/blog/dizajn-interfejsov-mobilnyh-prilozhenij>
46. Типи мережових протоколів і їх призначення (HTTP, IP, SSH, FTP, POP3, MAC) [Електронний ресурс]. Режим доступу: <https://deltahost.ua/ua/tipi-merezhevix-protokoliv-i-ih-priznachennya-http-ip-ssh-ftp-pop3-mac.html>
47. Клієнт-серверна архітектура [Електронний ресурс]. Режим доступу: <https://javarush.com/ua/quests/lectures/ua.questservlets.level14.lecture00>
48. Клієнт-серверна архітектура та ролі серверів [Електронний ресурс]. Режим доступу: <https://medium.com/@IvanZmerzlyi/клієнт-серверна-архітектура-та-ролі-серверів-9893d8048229>
49. Тестування програмного забезпечення [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення
50. Адресація пам'яті та змінні [Електронний ресурс]. Режим доступу: <https://javarush.com/ua/quests/lectures/ua.questsyntaxpro.level02.lecture04>

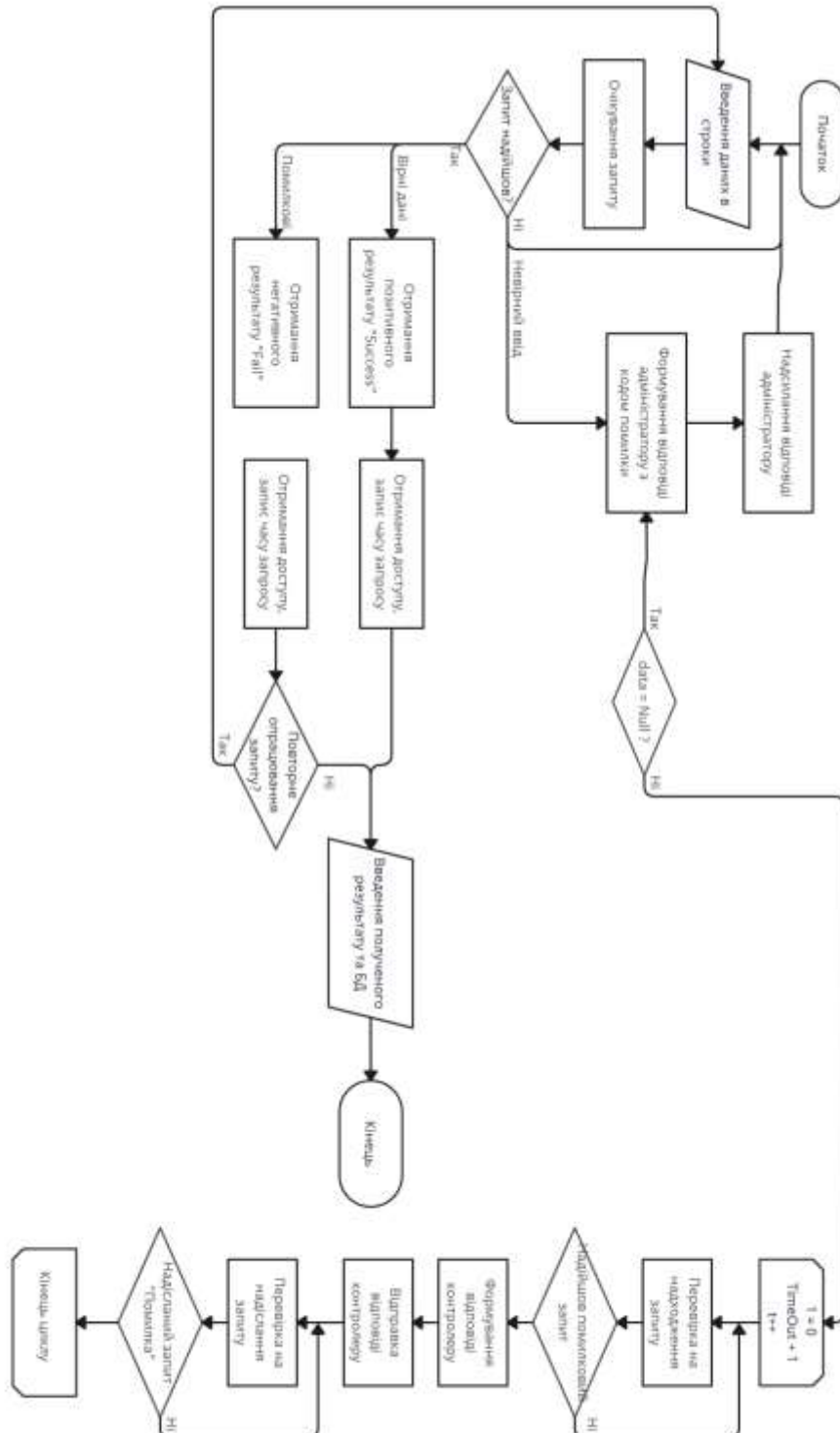
Додаток А

Схема взаємодії користувача між мікропроцесорним пристроєм



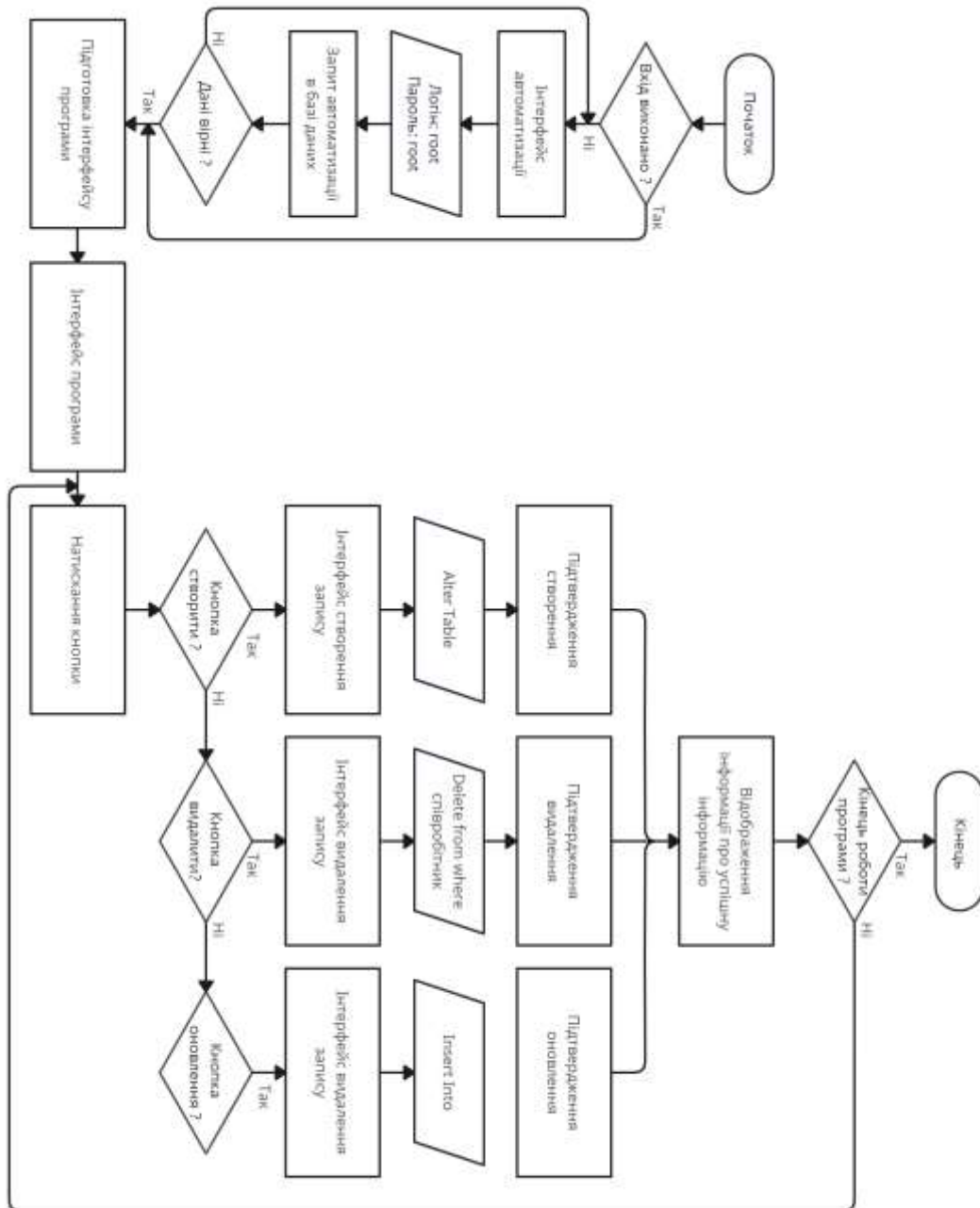
Додаток В

Схема роботи алгоритму бази даних



Додаток Г

Схема роботи алгоритму додатку адміністратора



Додаток Д

Код візуальної демонстрації програми

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity"
tools:layout_editor_absoluteX="0dp"
tools:layout_editor_absoluteY="23dp">

<Button
android:id="@+id/buttonRequest"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="64dp"
android:text="@string/buttonName"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.304"
app:layout_constraintStart_toEndOf="@+id/timeRequest"
app:layout_constraintTop_toBottomOf="@+id/acesProtect" />

<TextView
android:id="@+id/acesProtect"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="32dp"

```

```

android:layout_marginTop="48dp"
android:text="@string/takeAwayLock"
android:textSize="14sp"
app:layout_constraintStart_toEndOf="@+id/editTextText2"
app:layout_constraintTop_toBottomOf="@+id/textView4"
tools:ignore="MissingConstraints" />

```

```

<TextView
    android:id="@+id/textView4"
    android:layout_width="233dp"
    android:layout_height="39dp"
    android:layout_marginStart="100dp"
    android:layout_marginTop="156dp"
    android:text="@string/programName"
    android:textSize="24sp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

```

<EditText
    android:id="@+id/timeRequest"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="20dp"
    android:layout_marginTop="12dp"
    android:ems="10"
    android:hint="@string/timeRequest"
    android:inputType="time"
    android:minHeight="48dp"
    app:layout_constraintStart_toStartOf="parent"

```

```
app:layout_constraintTop_toBottomOf="@+id/editTextText2"
tools:ignore="MissingConstraints" />
```

```
<EditText
    android:id="@+id/editTextText"
    android:layout_width="143dp"
    android:layout_height="48dp"
    android:layout_marginStart="28dp"
    android:layout_marginTop="20dp"
    android:ems="10"
    android:hint="@string/firstNameWorker"
    android:inputType=""
    android:text=""
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView4" />
```

```
<EditText
    android:id="@+id/editTextText2"
    android:layout_width="175dp"
    android:layout_height="48dp"
    android:layout_marginStart="28dp"
    android:layout_marginTop="16dp"
    android:ems="10"
    android:hint="@string/secondNameWorker"
    android:inputType=""
    android:text=""
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextText" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

Додаток Е

Активна частина взаємодії з користувачем

```
package com.example.programmlock;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import org.w3c.dom.Text;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Date;

public class MainActivity extends AppCompatActivity {

    private TextView accessProtect;
    private EditText editTextText;
    private EditText editTextText2;
    private Button buttonRequest;
    private EditText timeRequest;
```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    acessProtect = findViewById(R.id.acessProtect);
    editTextText = findViewById(R.id.editTextText);
    editTextText2 = findViewById(R.id.editTextText2);
    buttonRequest = findViewById(R.id.buttonRequest);
    timeRequest = findViewById(R.id.timeRequest);

    buttonRequest.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String FirstName = editTextText.getText().toString();
            String SecondName = editTextText2.getText().toString();
            for (int i = 0; i < firstName.length;) {
                if (FirstName.equals(firstName[i])) {
                    acessProtect.setText("Success");
                    break;
                } else if (!FirstName.equals(firstName[i])) {
                    acessProtect.setText("Fail");
                }
                i++;
            }
            for (int c = 0; c < secondName.length;) {
                if (SecondName.equals(secondName[c])) {
                    acessProtect.setText("Success");
                    break;
                } else if (!SecondName.equals(secondName[c])) {

```

```
        acessProtect.setText("Fail");
    }
    c++;
}

Date currentData = new Date();

timeRequest.setText(String.valueOf(currentData));
}
});
}
```