Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Навчально-науковий інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня ___магістра___
(бакалавра, спеціаліста, магістра)

студента _____Шарипова Дмитра Олександровича_____
(ПІБ)

академічної групи _____123м-21-1_____
(шифр)

спеціальності _____123 «Комп'ютерна інженерія»_____
(код і назва спеціальності)

за освітньо-професійною програмою ___«Комп'ютерна інженерія»___
(офіційна назва)

на тему «Програмно-технічна реалізація комп'ютерної системи контролю і управління доступом котеджного містечка Sun Coast Dnipro»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Шедловська Я.І.			
розділів:				
теоретичний розділ	доц. Шедловська Я.І.			
синтез системи	доц. Бешта Д.О.			
розроблення програмного забезпечення	ас. Панферова Я.В.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ __Гнатушенко В.В._
(підпис) (прізвище, ініціали)
«___»_____ 2022 року

ЗАВДАННЯ на кваліфікаційну роботу

ступеня _____магістра _____
(бакалавра, спеціаліста, магістра)

студенту _____Шарипов Д.О. ___ **академічної групи** ____123м-21-1____
(прізвище та ініціали) (шифр)
спеціальності _____123 «Комп'ютерна інженерія»_____

за освітньою-професійною програмою _123 «Комп'ютерна інженерія» __
(офіційна назва)
на тему Програмно-технічна реалізація комп'ютерної системи контролю і управління доступом котеджного містечка Sun Coast Dnipro, затверджену наказом ректора НТУ «Дніпровська політехніка» від _31 жовтня 2022 р. №1200

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	10.10.2022
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	24.10.2022
Синтез системи	Розробка комп'ютерної системи	14.11.2022
Розроблення програмного забезпечення	Розробка програмного забезпечення	28.11.2022
Експериментальний розділ	Проведення і обробка результатів експериментів	05.12.2022
Графічна частина	Графічні результати роботи подати у вигляді рисунків схем таблиць на 10 арк. формату А4.	10.12.2022

Завдання видано _____ доц. Шедловська Я.І.
(підпис керівника) (прізвище, ініціали)

Дата видачі 10 жовтня_2022 р.

Дата подання до екзаменаційної комісії 15.12.2022 р.

Прийнято до виконання _____ Шарипов Д.О.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 77 с., 32 рис., 11 табл., 11 джерел, 1 додаток.

Об’єкт розробки: система контролю доступу котеджного містечка Sun Coast Dnipro.

Мета роботи: створити програмно-технічну реалізацію системи контролю і управління доступом на об’єкті впровадження. Надати теоретичне обгрунтування розробленої системи.

Пояснювальна записка включає аналіз типових способів створення систем контролю і управління доступом .На основі цих даних було створено завдання дослідження. Після цього були наведені вимоги до розроблюваної системи і побудована структурна і функціональна модель. На основі цього було провдене встановлення системи на об’єкті та розробка додаткового програмного забезпечення. Останнім кроком було проведення випробування створеної системи для виявлення проблем у її роботі.

СИСТЕМА УПРАВЛІННЯ І КОНТРОЛЮ ДОСТУПУ, СКУД, КОНТРОЛЬ ДОСТУПУ, TELEGRAM, UPROX

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

СКУД – система керування і управління доступом;

Аутентифікація – ідентифікація користувача у системі за наданими ним даними;

Авторизація – перевірка прав користувача при виконанні деякої дії;

API – опис засобів взаємодії однієї програми іншими;

REST – набір правил для взаємодії застосунків що використовують протокол HTTP;

HTTP – протокол прикладного рівня сетевої моделі OSI;

Месенджер – програма, створена для обміну повідомленнями між користувачами;

Бот – у мессенджерах, віртуальний користувач, який може взаємодіяти із реальними користувачами за допомогою програмних інтерфейсів;

Зчитувач – пристрій для зчитування цифрових ключів;

Контрольно-пропускний пункт - у системах СКУД, місце де відбувається контроль допуску користувачів до об’єкту;

КТП - скорочення контрольно-пропускного пункту;

ВСТУП

Для будь-якого об’єкту господарювання надзвичайно важливим є питання безпеки і охорони. Для об’єктів із великою кількістю користувачів важливою ланкою загальної системи безпеки є контроль і управління доступом.

Системи контролю і управління доступом мають на меті протидію незаконному проникненню на об’єкт впровадження, протидію розкрадання власності і забезпечення конфіденційності. Ці цілі досягаються за рахунок використання набору пристроїв і апаратних засобів, які об’єднуються в єдину систему програмними методами.

Системи такого роду є одним із найбільш розвинених сегментів ринку безпеки України, адже забезпечують надзвичайно високу ефективність роботи та заощаджують кошти при правильному впровадженні.

Мета і завдання дослідження. Метою даної роботи є програмно-технічна реалізація комп'ютерної системи контролю і управління доступу комплексу «Sun Coast Dnipro».

Для досягнення поставленої мети необхідно виконати такі завдання:

- дослідити існуючі рішення в області створення систем контролю і управління доступом такого типу
- на основі дослідження змоделювати створювану систему і сформулювати вимоги до неї
- синтезувати створювану систему, виділити її функціональні компоненти і здійснити підбір обладання
- встановити систему на об’єкті впровадження і за необхідності розробити програмне забезпечення
- провести тестування системи для визначення доцільності її використання

Об'єкт дослідження – процес створення системи контролю і управління доступом на території комплексу “Sun Coast Dnipro”.

Предмет дослідження – методи створення відмовостійких систем контролю доступу на території об’єктів значної площі.

Методи дослідження - для досягнення отриманої цілі були застосовані методи аналізу та синтезу, порівняння і методу моделювання.

СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Особливості об’єкта дослідження

Котеджне містечко – це, земельна ділянка за містом, викуплена компанією-забудовником для подальшого розбиття на менші за розміром ділянки та подальшого продажу фізичним або юридичним особам у якості житлового фонду. Зазвичай, на території таких ділянок, забудовник зводить типові будинки. Після продажу усіх котеджів, компанія-забудовник за певну платню бере на себе догляд за інфраструктурою прилеглих територій, наприклад: підведення і обслуговування міських комунікацій (вода, газ, електричний струм), вивезення сміття, благоустрій території, охорона території, тощо.

Таким чином, особа, яка викупає ділянку у котеджному містечку отримує право власності на заміське житло у охоронюваному та добре облаштованому районі та звільняється від типових проблем нерозвиненої інфраструктури за містом. В Україні практика котеджних містечок ще недостатньо популярна, проте поступово розвивається через великий попит на заміське житло.

Для зручності і безпеки мешканців, доступ до містечка має бути обмежений і контрольований.

Система контролю і управління доступом повинна виконувати такі функції:

- Фізичне обмеження доступу до об’єкту небажаних людей/транспорту.
- Ведення журналу доступу до об'єкту.
- Надання доступу до об’єкту після ідентифікації людей;

Засобами ідентифікації можуть виступати:

- Фізичні ключі (nfc мітки, брелки тощо).
- Цифрові ключі (аутентифікація через довірені пристрої за допомогою глобальної мережі)

1.2 Аналіз існуючих систем СКУД

Сьогодні популярними є системи СКУД, види яких умовно можна розподілити на дві категорії

- автономна система контролю доступу
- мережева система контролю доступу.

Кожна з них має свої особливості і відмінності.

1.2.1 Автономна система

Ця система контролю доступу є децентралізованою, без єдиного центру прийняття рішень.

Основними складовими системи є:

- Зчитувач, який передає код ідентифікаційний контролеру, який приймає рішення про обмеження чи надання доступу. Цей пристрій може бути як контактним, так і безконтактним.
- Ідентифікатори - пристрої, що зберігають ідентифікаційний ключ. Вони працюють у парі з зчитувачем, для того щоб визначити рівень допуску користувача. Прикладом безконтактного ідентифікатора для зчитувача є радіо-брелок, аналогічний до брелків які використовуються у сучасних автомобілях. Прикладом контактного ідентифікатора є магнітна картка, яку необхідну підносити до зчитувача.

- Обмежувальні заходи - ворота, турнікети, електричні замки. Фізично обмежують доступ до об’єкту допоки не отримають сигнал від контроллеру.
- Контролери - пристрій, який приймає рішення для допуску чи недопуску людей на територію об’єкта. У пам’яті контроллера міститься список ключів для користувачів, які надають право доступу. Вони працюють у парі з зчитувачами і обмежувальними заходами. Після отримання ключа від зчитувача, або передають сигнал до обмежувальних заходів, або ігнорують користувача з недостатніми правами.

У такому типі СКУД кожен пункт пропуску являє собою закриту систему - контролер, зчитувач, ідентифікатор, обмежувальний захід. Кожна така система створюються навколо одного пункту пропуску (калитка, ворота, шлагбаум).

Дана система гарно підходить для невеликих підприємств з невеликою кількістю користувачів. Оскільки кожен контролер має список користувачів, немає необхідності у існуванні постійного з’єднання між ними для обміну інформацією. Таким чином, значно зменшуються кошти на прокладання мережі на об’єкті. У той же час, у цієї системи є великі проблеми з масштабованістю, адже зі збільшенням кількості контролерів забезпечувати синхронізацію користувачів між ними стає надалі важче.

1.2.2 Мережева система контролю доступу

На відміну від попередньої системи, системи такого типу мають централізований сервер, який є єдиним джерелом прийняття рішень. Зазвичай він є віддаленим, тому такі системи для роботи потребують інтернет.

Основними складовими системи є:

- Зчитувач - аналогічно до автономних СКУД

- Ідентифікатори – аналогічно до автономних СКУД
- Обмежувальні заходи – аналогічно до автономних СКУД
- Контролери – на відміну від попередньої системи, є необов’язковою частиною. В залежності від типів використовуваних приладів, якщо зчитувачі і обмежувальні заходи можуть напряму до мережі, можуть бути непотрібні. Якщо ж такої можливості немає, виступають проміжною ланкою між центральним сервером і зчитувачами/обмежувальними заходами.
- Центральний сервер – центр прийняття рішень системи, який веде облік користувачів, журнал відвідувань тощо. З’єднується у єдину мережу з складовими системи.

В свою чергу центральні сервери можуть бути:

- Віддаленим – з’єднується з локальними пристроями через мережу інтернет. Таким чином, немає необхідності локального розміщення потужного обладнання. Такий варіант серверу є доволі популярним, адже це зазвичай спрощує монтаж системи і полегшує адміністрування. Недоліком є необхідність постійного підключення усієї системи до глобальної мережі.
- Локальним – сервер встановлено на об’єкті.

Системи такого типу зазвичай є більш дорогими і складними ніж автономні СКУД, проте для великих об'єктів з великою кількістю точок доступу вони є більш раціональним вибором.

Одним із основних недоліків таких систем є те, що вони потребують постійного підключення до мережі інтернет. У випадку відключення глобальної мережі, вони або перестають функціонувати загалом (у випадку систем з віддаленим сервером), або просто втрачають свої переваги (можливість віддаленого надання допуску).

Більшість постачальників обладнання для сучасних систем такого типу орієнтуються на використання власних хмарних віддалених серверів. Таке рішення не завжди є найоптимальнішим, адже дані користувачів знаходяться у руках третіх осіб, а системи потребують інтернет для роботи.

1.3 Проблеми сучасних систем

Обидва типи існуючих систем мають свої переваги і недоліки.

Системи можна аналізувати за декількома чинниками:

- Ціна розгортання – для обох типів систем необхідна закупівля та використування великої кількості зазвичай недешевого обладнання. Для мережевих систем контролю доступу також необхідне налаштування центрального серверу, що робить процес дорожче і довше.
- Ціна підтримки/розширення системи – централізовані мережеві системи значно простіше адмініструвати і моніторити. Додавання нових користувачів/видалення старих на автономних системах значно ускладнене і може призводити до людських помилок і в результаті, до неадекватної роботи окремих частин системи.
- Масштабованість – централізовані мережеві системи значно легше розширяти, тоді як у випадку автономних систем із збільшенням кількості користувачів/пунктів допуску підвищується складність підтримки системи.
- Відмовостійкість – оскільки мережеві системи в більшості випадків потребують постійного з’єднання з мережею інтернет, вони менш стійкі і більш залежать від зовнішніх умов.
- Стандарти безпеки – у автономних систем немає централізованого журналу доступу, тому за рівнем безпеки вони поступаються мережевим.

- Інтеграція з іншими охоронними системами – мережеві системи значно простіше інтегрувати з іншими системами, такими як відеоспостереження.

Основною проблемою сучаних систем СКУД є забезпечення належного рівня безпеки, включаючи ведення моніторингу доступу і журналу користувачів, зі збереженням відмовостійкості та максимальної адекватності системи не зважаючи на зовнішні умови. Багато сучасних рішень орієнтовані на інтеграцію з хмарними сервісами і застосунками виробників обладнання. Це створює великі проблеми для користувачів, які хочуть забезпечити максимальну безпеку даних власних клієнтів.

Сучасна система СКУД має покривати усі потреби фізичної безпеки, надавати засоби моніторингу та інформацію для подальшого аналізу доступу до об'єкта, і в той же час бути максимально стійкою до зовнішніх факторів. Підбір правильного обладнання для таких систем стає критичною задачею, проте після розробки вона стане надзвичайно важливою і надійною складовою ланкою забезпечення безпеки об'єкта.

1.4 Постановка завдання дослідження

Метою цієї роботи є створення системи контролю і управління доступом, яка надаватиме мешканцям котеджного комплексу необхідний рівень безпеки. Вона має включати позитивні риси як автономних так і мережевих систем – це означає забезпечувати централізоване керування системою, ведення журналу обліку, можливості інтеграції з іншими сервісами, а також надання максимальної відмовостійкості та безпеки зберігання інформації.

Для досягнення поставленої мети к роботі необхідно виконати низку завдань:

- описати стан питання і здійснити постановку завдання;

- описати переваги і недоліки існуючих систем
- навести теоретичне обгрунтування створюваної системи
- навести технічні вимоги до комп’ютерної системи контролю та керування доступом;
- підібрати необхідні апаратні засоби для використання у системі;
- виконати проектування програмного забезпечення системи та перевірку роботи комп’ютерної системи об’єкта;
- провести тестування створеної системи

ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Загальна характеристика об`єкту впровадження

Об’єктом впровадження системи є котеджне містечко Sun Coast Dnipro. Об’єкт знаходиться на березі ріки Мокра Сура у с. Новоалександрівка, що знаходиться за 8 кілометрів від міста Дніпро.

Керуючою компанією об’єкта є забудовник DDC. Компанія знаходиться на ринку 9 років і спеціалізується виключно на створенні котеджних містечок. Дане котеджне містечко було першим об'єктом такого роду, створеним компанією.

Територію містечка складає 7.5 га, які розподілені на 63 окремі ділянки. Приблизну схему ділянок відображено на рис 2.1.

Рисунок 2.1 – Схема котеджного містечка

На цих ділянках збудовано будинки декількох типів. Після завершення будівництва, будинки виставляються на продаж як приватизовані об’єкти з кадастровим номером. До кожної ділянки підведено комунікації, постачальником яких є компанія забудовник:

- Водопостачання
- Електроживлення
- Каналізація
- Природний газ

Крім комунікацій, компанія-забудовник надає послуги централізованого вивозу сміття і охорони об’єкту.

Засобами забезпечення охорони об’єкту є:

- Огорожа по всьому периметру містечка
- Підтримання безпеки охоронцями, які здійснюють нагляд за територією і можуть бути викликані мешканцями при надзвичайнх ситуаціях.

На виїздах з території об’єкту розташовані пропускні пункти з воротами і хвіртками, які вручну контролюються охоронцями. Один із пропускних пунктів зображено на рис 2.2.

Рисунок 2.2 – Пропускний пункт котеджного містечка

Інтернет у домівки і адмінбудівлі комплексу постачається не компанією-забудовником, а ПрАТ Київстар – телекомунікаційної компанії, у якої укладено контракт із забудовником. До кожної ділянки глобальна мережа підводиться за допомогою оптичного волокна із конвертером, а локальної мережі на об’єкті не існує.

Аналізуючи загальні відомості про об’єкт можна виділити декілька чинників, що створюють проблеми для забезпечення безпеки:

- Відсутність автоматизації контролю доступу, оскільки доступ на об’єкті наразі контролюється персоналом охорони, що робить систему вразливою для людського фактору.
- Несистематизованість ведення журналу доступу, який доводиться заповнювати вручну
- Відсутність мережі, які б об’єднувала пристрої об’єкту, що створює проблеми для інтеграції автоматизованої охоронної системи

2.1.2 Структура адміністративної системи об’єкту впровадження

У існуючій системі адмінстрації об’єкту можна виділити такі складові:

- Охоронний периметр, який являє собою цегляний паркан висотою 2 метри. Слугує для фізичного обмеження доступу на об’єкт за межею пропускних пунктів. Зображено на схемі за допомогою чорної лінії.
- Контрольно-пропускні пункти, у вигляді воріт та хвірток через які мешканці отримують доступ до містечка. Позначено на схемі як «КПП».
- Адміністративна будівля, де знаходяться резервні охоронці на випадок надзвичайних ситуацій і зберігається журнал доступу. Позначено на схемі як «АДМ».

Розташування зазначених складові зазначено на рис 2.3:

Рисунок 2.3 – Розташування складових адміністрації

На котрольно-пропускних пунктах знаходиться 1-2 охоронці які контролюють допуск людей до території містечка.

У адміністративній будівлі знаходиться декілька відділів, які створені для підтримання ефективної роботи служб містечка:

- Відділ продажів
- Бухгалтерія
- Серверна із робочим місцем системного адміністратора
- Кімната охоронців
- Кабінет директора

До адміністративної будівлі підключено глобальну мережу провайдером ПРаТ Київстар. Зв’язок забезпечується за допомогою одного конвертора (оптоволоконне підключення), одного маршрутизатора і одного коммутатора.

Схему адміністративної будівлі можна зобразити так:

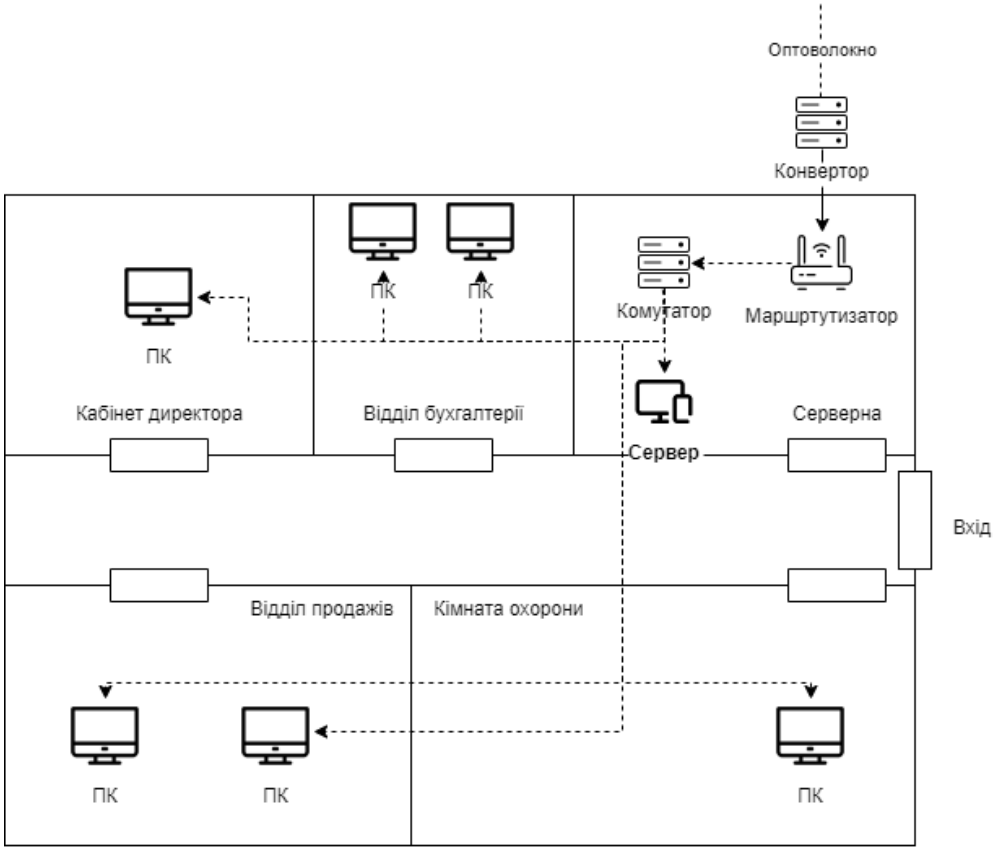


Рисунок 2.4 – Схема адміністративної будівлі

Ієрархічна структура підприємсства зображена на рис 2.5:

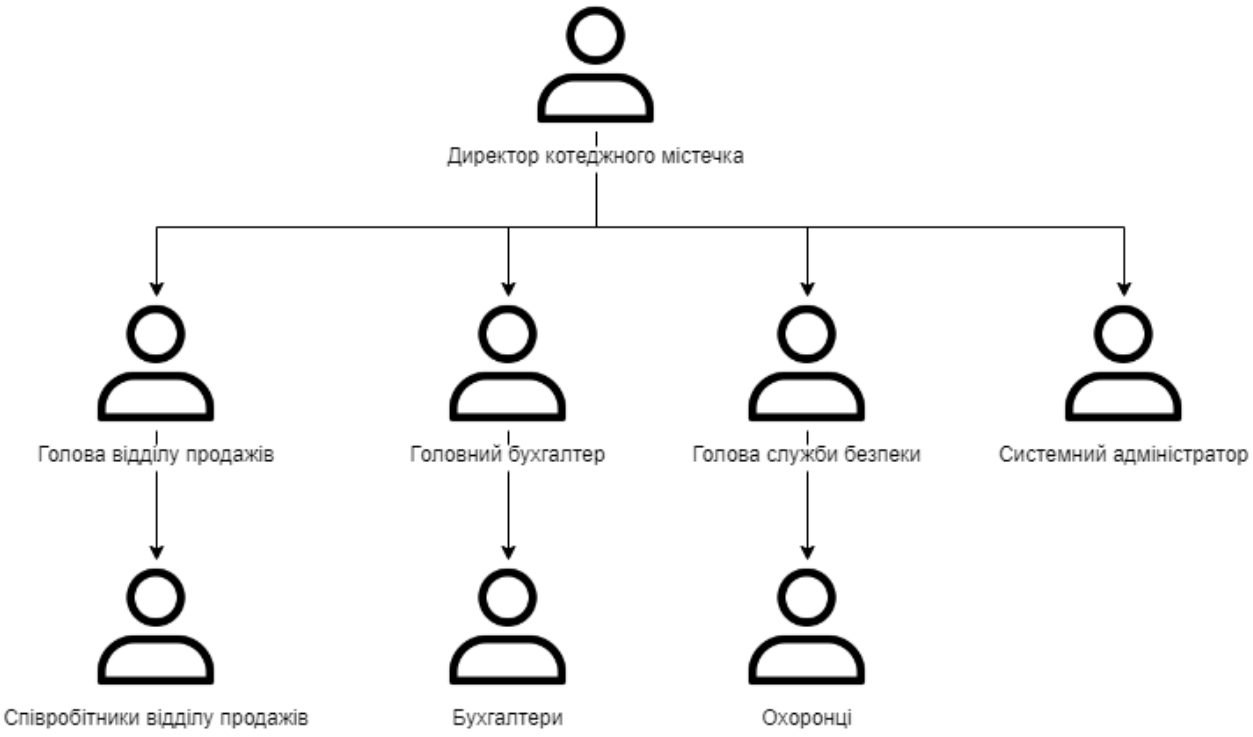


Рисунок 2.5 – Ієрархіна структура

2.2 Обґрунтування і вибір методів дослідження

Для виконання поставлення завдання необхідно розробити автоматизовану систему керування доступом для об’єкту впровадження.

Для розроблення системи, необхідно обрати методи і критерії, що допоможуть синтезувати розроблювану систему. У результаті дослідження, для досягнення цієї мети було вирішено обрати такі методи:

— метод порівняння;

— метод аналізу та синтезу;

— метод моделювання

2.2.1 Метод порівняння

Для того, щоби визначитися із типом створюваної системи, необхідно порівняти існуючі типові рішення для систем СКУД.

На даний момент на об’єкті немає автоматизованої системи контролю доступу. Процес доступу до об’єкта контролюється охоронцями вручну.

Як вже описано у попередньому розділі, систему керування доступом поділяються на два типи: автономні і мережеві. Необхідно дослідити переваги і недоліки кожної із них.

Для автономної системи їх можна сформувати так:

Переваги:

- Проста у виконанні
- Дешева
- Не потребує мережі для роботи

Недоліки:

- Не підходить для об'єктів з великою кількістю пунктів пропуску
- Важко підтримувати
- Не має централізованого журналу доступу

Для мережевої системи їх можна сформулювати так:

Переваги:

- Гарна масштабованість
- Має журнал централізованого доступу
- Підвищена безпека зберігання даних

Недоліки:

- Зазвичай дорожча ніж автономні системи
- Зберігання даних на серверах третіх осіб
- Необхідність постійного з’єднання з глобальною мережею

Таким чином можна скласти таблицю порівніння типів систем:

Таблиця 2.1 Порівняння мережевої і автономної систем СКУД

Показник	Автономні СКУД	Мережеві СКУД
Ціна розгортання	+	-
Ціна підтримки	-	+
Масштабованість	-	+
Відмовостійкість	+	-
Стандарти безпеки	-	+
Інтеграція з іншими системами	-	+

2.2.2 Методи аналізу і синтезу

Виходячи з порівняння автономної і мережевої систем і загальної характеристики системи, необхідно проаналізувати основні критерії для синтезу розроблюваної системи.

Для цього необхідно проаналізувати дані подані у таблиці 2.1 у площині об’єкта впровадження:

- Ціна розгортання – для компанії-забудовника початкова ціна впровадження системи не є критичною, адже вона має значні економічні ресурси, а впровадження системи дозволить зекономити кошти у майбутньому.
- Ціна підтримки – важливий критерій для об’єкту, що планується підтримувати довгостроково. Для даного конкретного об’єкту це питання не є критичним, адже у штаті вже маються системні адміністратори, проте будь-які кошти зекономлені у довгостроковій перспективі мають велике значення для успішного ведення бізнесу.
- Масштабованість – об`’єкт розробки планує розширення у майбутньому із доданням 20-30 будинків і 1 контрольно-пропускного пункту. Також можливе створення інтегрованої системи контролю доступа усередині містечка. Тому цей пункт є важливим для впровадження системи.
- Відмовостійкість – цей компонент э важливим для будь-якої системи контролю доступу. Мешканці мають отримувати доступ до своєї оселі цілодобово.
- Стандарти безпеки – незважаючи на відсутність державного регулювання стандартів безпеки котеджних містечок, компанія-забудовник зацікавлена в наданні найвищого можливого рівня безпеки мешканцям.
- Інтеграція з іншими системами – комплекс заходів щодо забезпечення безпеки планується розширювати у майбутньому тому для об’єкту впровадження цей критерій є важливий.

Таким чином, можна зробити висновок, що для об’єкту впровадження доцільно обрати мережеву модель створення системи контролю і управління доступом.

Проаналізуємо також основні недоліки системи мережевого типу систем СКУД для об’єкта впровадження:

- Необхідність функціонування локальної мережі на об’єкті впровадження
- Забезпечення відмовостійкості системи в умовах відключення електропостачання/зв’язку
- Забезпечення безпеки зберігання даних

2.2.3 Модель комбінованої системи контролю і управління доступом

У будь-якої системи контролю і управління доступом можливо виділити дві логічні складові:

- Пам'ять, де зберігаються дані про користувачів і їх ідентифікатори доступу
- Контролер, який приймає рішення про надання доступу, використовуючи інформацію, що знаходиться у пам’яті.

У пам’яті зберігаються структуровані дані про складові моделі за допомогою яких отримується доступ. У базовому варіанті системи це:

- Користувачі - компонована модель користувача, що містить про нього дані
- Двері/замки та інші обмежуючи пристрої - точки доступу, через які можна його отримати
- Ідентифікатори, що прив’язані до дверей і користувачів - ключі ідентифікування користувачів у системі

Структуру пам’яті можна зобразити так (рис. 2.6):

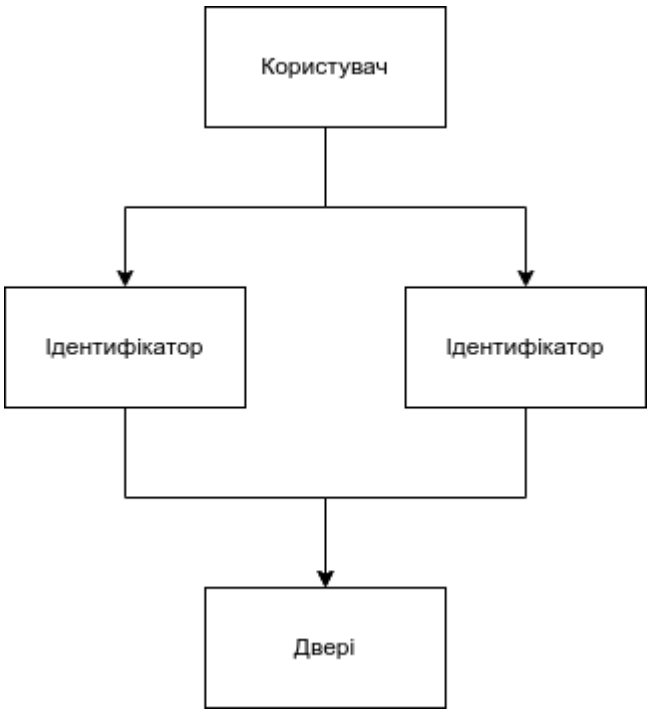


Рисунок 2.6 – Структура пам’яті

Реляційні зв’язки між складовими пам’яті мають певні обмеження:

– Система може мати нескінченну кількість користувачів і дверей
– У кожного ідентифікатора має бути один і лише один власник, тому їх кількість має відповідати або перевищувати кількість користувачів
– Користувач може мати нескінченну кількість ідентифікаторів
– Кожен ідентифікатор може бути прив’язаний до будь-якої кількості дверей

Контролер порівнює ідентифікатори, що містяться в пам’яті з ідентифікаторами, наданими користувачем. Він має скінченну кількість станів і переходів із них, тому його логічну схему можна зобразити за допомогою графу скінченного автомату [1]. Його стани можна описати таким чином:

– S0 - очікування запиту на доступ від користувача, доступ закритий

– S1 - ідентифікація користувача, порівняння наданого ідентифікатора із тими, що зберігаються в пам’яті
– S2 - надання доступу

Переходи між станами можна описати таким чином:

– Λ1 - отримання ідентифікатора від користуваача
– Λ2 - успішне співставлення наданого ідентифікатора із наявним у пам’яті
– Λ3 - наданий ідентифікатор у пам’яті не знайдено
– Λ4 - повернення до початкового замкненого стану

Результуючий граф зображено на рис 2.7:

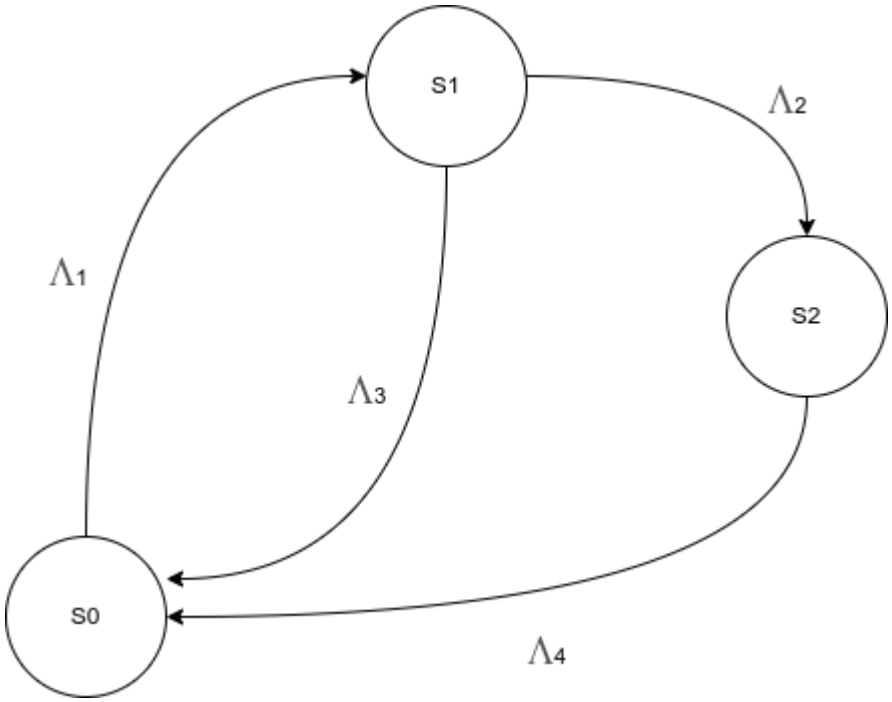


Рисунок 2.7 – Граф станів контролеру

Для мережевих систем контролю доступу характерне використання одного центрального контролеру, який керує усіма пристроями, приєднаними за допомогою мережі. Це створює проблеми з точки зору відмовостійкості системи, адже будь-яка несправність у мережі може спровокувати вихід з ладу одного або декількох пропускних пунктів.

Для того, щоб цьому заподіяти можливе використання контролерів на пунктах пропуску. В такому випадку, система складається із двох типів контролерів - одного центрального і довільної кількості напівавтономних із різним набором фукціонального призначення.

Характеристика центрального контролеру:

– Являє собою єдине джерело інформації системи
– Синхронізує зміст пам’яті між напівавтономними контролерами
– Веде централізований журнал доступу

Характеристика напівавтономних контролерів:

– Встановлюються на кожному окремому пункті пропуску
– Ідентифікують користувачів незалежно від центрального контролера
– Повідомляють центральний контролер про події надання доступу

Схематично таку систему можна зобразити так: (рис 2.8)

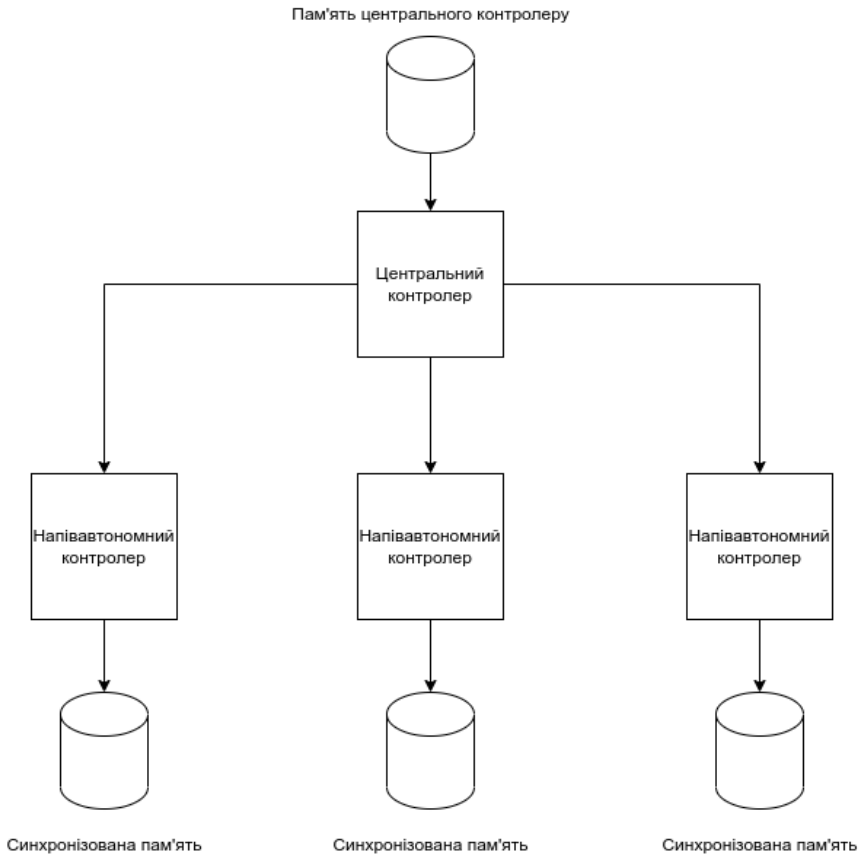


Рисунок 2.8 – Схема системи із напівавтономними контролерами

2.3 Формулювання факторів, що впливають на синтез системи

Використовуючи дані результатів аналізу та моделювання, можна виділити такі критерії для синтезу і розробки створюваної системи:

- Система повинна завжди поводитись однаково і передбечувано за умови незмінності пам’яті
- Система повинна відповідати мережевому типу систем СКУД
- Для створення системи необхідно створити локальну мережу на об’єкті впровадження
- Система має реалізовувати відмовостійкість за допомогою використання центрального і напівавтономних контролерів
- Безпека зберігання даних повинна реалізуватися за допомогою централізованого зберігання даних пам’яті системи на локальному сервері

2.4 Висновки до розділу

У даному розділі проведено аналіз об’єкта дослідження для досягнення поставленої мети. Проведено дослідження метою, якого було опис вимог до розроблюваної системи і синтез абстракції розроблюваної системи. За допомогою методу порівняння було досліджено переваги і недоліки існуючих типів систем, за допомогою методу аналізу було визначено базовий тип розроблювальної, а за допомогою методу моделювання створено базову структуру розроблювальної системи.

СИНТЕЗ СИСТЕМИ

3.1 Вибір і обгрунтування принципів побудови системи

Перед впровадженням системи контролю і управління доступом задано наступні цілі:

- фізичне обмеження доступу неавторизваних осіб до об’єкта
- можливість отримання доступу до об’єкту за допомогою цифрових ключів доступу
- можливість отримання доступу до об’єкту віддалено
- ведення журналу відвідувань користувачів і можливість його перегляду

Для побудови системи необхідно визначитися із складовими системи, створити структурну схему на базі моделі із теоретичного розділу, описати вимоги до системи, синтезувати функціональні складові та створити функціональну схему системи.

3.2 Обгрунтування прийнятих засобів проектування і характеристик системи

Для проектування системи необхідно виділити основні її частини.

Система має складатись з таких частин:

— сервіс управління пристроями з адміністративною панеллю і відкритим інтерфейсом;

— інтерфейс користувача для віддаленого доступу до системи;

— локальна база даних для зберігання даних про користувачів системи

Система буде розміщуватись на центральному сервері. Всі інші пристрої будуть під’єднані до однієї підмережі і зв’язуватись із системою за допомогою IP технологій (Ethernet).

Реалізація сервісу управління пристроями буде залежати від виробника обладнання. Він має надавати широки можливості адміністрування і розширення системи за допомогою відкритого інтерфейсу.

Для віддаленого доступу користувачів необхідно надати можливість керування пристроями, до яких користувач має доступ за допомогою мобільного телефону. Для цього доцільно використати популярні месенджери, зокрема Telegram [2]. Створивши бота, можна надати користувачам можливість зручно взаємодіяти з системою.

Для реалізації бота було обрано мову програмування JavaScript [3] як основну, оскільки за допомогою неї можна написати усі складові частини системи, включаючи інтерфейс адміністративної панелі. У якості допоміжного інструменту буде використана система типізації TypeScript [4].

Доцільно використати Telegraf [5] - бібліотеку JavaScript, що полегшує розробку ботів для Telegram.

Система контролю і управління доступом має забезечувати безпечний і контрольований доступ до об’єкту. Система повинна виконувати такі функції.

- дозволяти авторизованим мешканцям отримувати доступ до об’єкту за допомогою фізичних і цифрових ключів
- обмежувати доступ до об’єкту небажаним особам
- вести журнал доступу користувачів до об’єкту
- надавати можливість адміністратору віддаленого керування пристроями і системою в цілому через адміністративну панель

Для роботи системи необхідне розміщення центрального сервера на територіїї об’єкту. Центральний сервер має бути з'єднаним з усіма приладами у одну мережу.

Центральний сервер системи мусить відповідати таким вимогам:

– мінімальний об’єм диска - 250 ГБ з потенційною можливістю розширення;
– процесор з 4 фізичними ядрами тактовою частотою не менш 1.8 ГГц;
– 16 ГБ оперативної пам’яті.

Сервер повинен мати встановлену одну з таких операційних систем:

- Windows 10;
- Windows Server 2016;
- Windows Server 2019;
- Windows Server 2022;

Захист інформації в цій комп’ютерній системи мусить складатися з низки заходів, направлених на захист конфіденційності інформації і захисту від зовнішнього втручання у систему.

Доступ до адміністративної панелі буде обмежений пристроями у локальній мережі і закритий до доступу зовні. У разі виникнення необхідності отримання віддаленого доступу, адміністратор повинен буде налаштувати VPN сервіс на маршрутизаторі.

Система матиме одного адміністратора, і лише він одночасно матиме доступ до неї. Пароль адміністратора системи повинен бути завдовжки 15 символів, та повинен бути згенерований випадково.

Всі зовнішні порти доступу до центрального серверу задля безпеки мають бути закриті.

Журнал користувачів має мати можливість бекапу на віддалені сервери, для того щоб запобігти втраті важливих даних у випадку дій зловмисників.

Сервер має знаходитися у будівлі, де знаходиться охорона, у спеціально відведеній для цього кімнаті. Фізичний доступ до сервера має бути лише у системних адміністраторів. У адміністраторів має бути можливість швидкого доступу до системи для зручного обслуговування.

Персонал системи має складатися з двох адміністраторів, які мають академічну степінь бакалавра або вище у галузі комп’ютерної інженерії. Періодичність інструктаж з техніки безпеки – 2 рази на рік. Режим роботи − 1 зміна по 12 годин, 2 дні через 2 дні. Тривалість робочого місяця – від 28 до 31 календарних діб, в залежності від місяця.

3.3 Синтез структурної схеми

На основі даних теоретичного розділу було створено структурну схему для системи контролю доступу, що зображено на рисунку 3.1.

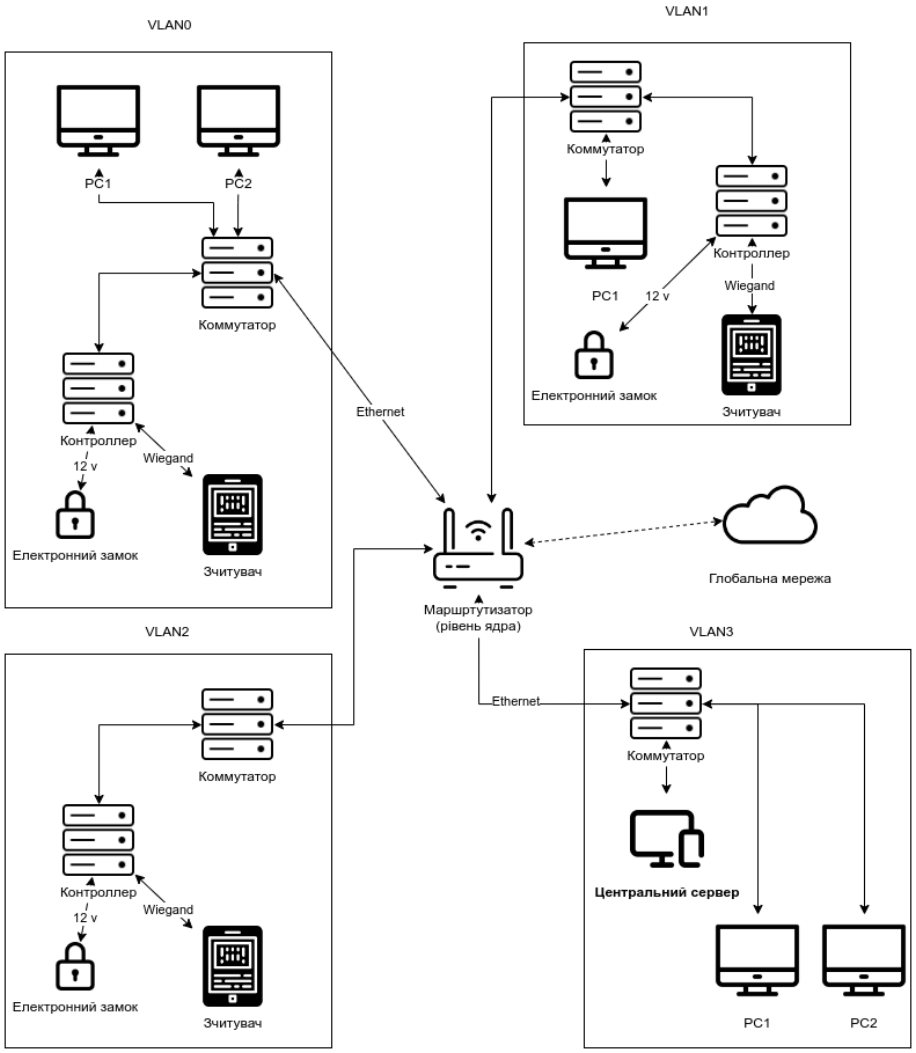


Рисунок 3.1 – Структурна схема системи контролю доступу

3.4 Проектування системи

3.4.1 Опис складових функціональної схеми

У попередніх розділах були виділені складові системи. Необхідно характеризувати кожен із них.

Сервіс управління пристроями:

- Надає можливості віддаленого управління пристроями для адміністратора
- Веде журнал доступу користувачів до системи
- Має доступ до списку ідентифікаторів доступу користувачів для синхронізації між пристроями
- Має відкритий інтерфейс для розширення і автоматизації системи

Telegram бот для користувачів:

- Надає можливість користувачам взаємодіяти з системою віддалено за допомогою глобальної мережі
- Надає можливість аутентифікації за допомогою номеру телефону

Локальна база даних:

- Забезпечує зручне і надійне зберігання даних системи
- Має можливість створення резервних копій

СКУД пристрої:

- Напівавтономні контролери, що обробляють запити від зчитувачів
- Зчитувачі, що отримують запити на доступ від контактних засобів ідентифікації користувачів
- Електронні замки, що фізично обмежують доступ людей до об’єкту

3.4.2 Побудова функціональної схеми

Враховуючи ці дані можна створити схему функціональної структури, зображену на рис 3.2.

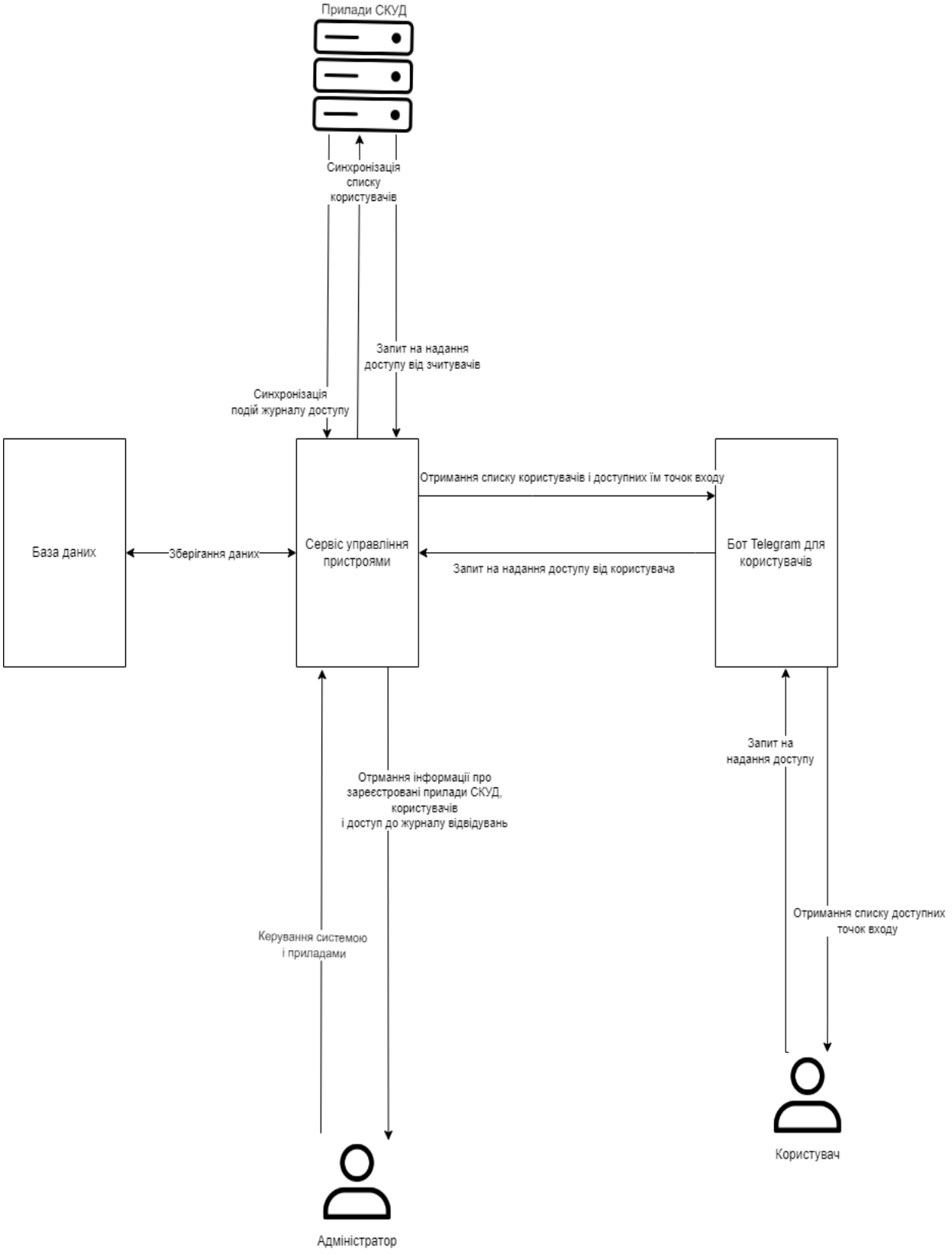


Рисунок 3.2 - Схема функціональної структури

Центральним елементом створеної системи буде сервіс управління пристроями. Він буде обробляти логіку надання користувачам доступу, базуючись на даних що зберігаються у базі даних.

У системі можна виділити два типи клієнтів:

- Адміністратор, який керує системою через адміністративну панель сервісу управління пристроями.
- Користувач, який використовує бот Telegram для отримання доступу

Система налаштовується адміністратором, який додає у систему прилади і користувачів і надає їм права доступу. Після цього сервіс синхронізує дані про користувачів із компонентами системи. Після налаштування системи, у користувачів є два способи отримання доступу до об'єкту:

- Приклавши цифровий ідентифікатор до зчитувача на пропускному пункті. У цьому випадку контролер СКУД, встановлений на згаданому пропускному пункті, має співставити наданий ключ ідентифікації з одним із користувачем системи. У випадку успішного надання доступу, контролер повідомляє центральний сервер для створення запису в журналі доступу.
- Віддалено, використовуючи бот Telegram. Після аутентифікації у додатку, обліковий запис Telegram користувача прив'язується до користувача системи. Надалі використовуючи спеціальні команди, користувач може відправлять запити на надання доступу на той чи інший пропускний пункт. При відправленні подібного запиту, центральний сервер повідомляє відповідний контролер та створює запис у журналі.

Усі дані системи мають зберігатися у базі даних.

3.4.3 Вибір та обгрунтування застосування апаратних засобів

Як вказано у вимогах до системи, необхідно підібрати центральний сервер для системи з підтримкою операційної системи Windows.

Оскільки на об’єкті вже існує стійка для серверів, у цілях економії місця та організації простору, доцільно використовувати сервер сумісний із кріпленням на стойку.

У якості виробника сервера було обрано компанію Dell.

Dell — американська компанія з головним офісом в Остіні, штат Техас, вона є одним із провідних лідерів в галузі розробки і виробництва комп'ютерних систем. «Dell» виробляє сервери, системи зберігання даних, робочі станції, мережне устаткування, персональні комп'ютери тощо.

Оскільки вимоги до потужності системи невеликі за сучасними показниками, було обрано недеорогий сервер початкового рівня Dell R420 (рис 3.3).

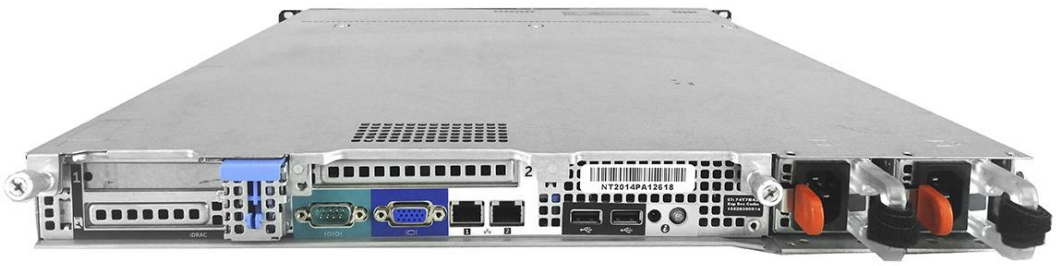


Рисунок 3.3 – Сервер Dell R420

Цей сервер може бути конфігурований запчастинами в залежності від вимог замовника. У випадку нашої системи його було конфігуровано таким чином:

- Процесор - Intel XEON E5-2420 - за вимогами до видів забезпечення

Таблиця 3.1 – Технічні характеристики процесора

Показник	Значення
Кількість фізичних ядер	6
Кількість віртуальних ядер	12
Частота	1,9 – 2,4 ГГц
Потужність	95 ватт
Підтримувані типи пам’яті	DDR3 800/1066/1333

- Оперативна пам’ять – Samsung 4 GB DDR 1333 (2 шт.) – за вимогами до видів забезпечення
- Диски – Seagate 300 GB HDD 3.5” (2 шт.) – оскільки прогнозована кількість операцій запису/читання з диску не є високою, доцільно використати дешеві HDD диски у масиві RAID1 для підвищення надійності. Таким чином якщо 1 диск вийде з ладу, його можна буде замінити не втрачаючи дані.

У випадку нашої системи, важливим фактором для серверу є можлвість під’єднання периферійних пристроїв, адже доступ до сервера планується отримувати тільки з серверної кімнати. Як можна бачити на рис 3.2, для цього у серверу Dell R420 є набір портів – USB 2.0 (2 шт.) для підключення периферії, VGA для підключення монітору, Ethernet (2 шт.) для під’єднання до мережі. Також на сервері є інтегрована у материнську плату відеокарта, якої вистачить для потреб системи.

Таким чином можна скласти загальні характеристики серверу:

Таблиця 3.2 - Загальні характеристики серверу

Показник	Значення
Процесор	Intel XEON E5-2420
Оперативна пам’ять	Samsung 4GB DDR 1333 (2шт.)
Жорсткі диски	Seagate 300 GB HDD 3.5” (2 шт.)
RAID контроллер	PERC H130
Живлення	Блок живлення DELL від мережі 220V
Відеоадаптер	Вбудований
Зовнішні порти	USB 2.0 (2 шт.), VGA (2шт.), Ethernet (2 шт.)
Кріплення стійки	Рейки універсальні нерухомі 19”
Фізичні розміри	434x42.8x607 мм

Для того, щоб забезпечити комфортність роботи адміністраторів необхідно організувати підключення до сервера як до локальної робочої станції. Таким чином, необхідно забезпечити сервер монітором і периферійний обладнанням (а саме клавіатура та комп’ютерна миша).

Основними критеріями для монітору є роздільна здатність і наявність VGA входу відповідно для під’єднання до сервера.

У якості монітору було підібрано Монітор DELL E2720H (зображений на рис. 3.4) від вже описаного виробника DELL.

Він відповідає всім вимогам, описаним вище а саме:

- Роздільна здатність – 1920x1080 пікселів
- Частота оновлення – 60 Гц
- Доступні типи портів – VGA, DisplayPort, HDMI

Рисунок 3.4 - DELL E2720H

Периферійні пристрої мусять використовувати USB порт для під’єднання до серверу. Оскільки жорстких критеріїв для підбору немає, необхідно використати найдешевші компоненти, які при цьому забезпечують належний рівень якості.

У якості виробника було обрано компанію Logitech – швейцарську компанію, яка відома на ринку відносно недорогими, проте довговічними пристроями. Було обрано набір клавіатури та миші комп’ютерної MK120.

Рисунок 3.5 - Logitech MK12

Запроваджувана мережа комп’ютерної системи матиме два рівні: рівень ядра та рівень доступу.

Для використання на рівня ядра мережі необхідно обрати маршрутизатори, відповідаючі таким мінімальним вимогам:

Висока пропускна здатність - до об’єкту підведено з’єднання глобальної мережі 1000 мбіт/с, тому для повного використання можливостей, наданих провайдером необхідна підтримка такої швидкості на рівні маршрутизатора.

Наявність великої кількості портів - об’єкт, на якому реалізується система постійно розширюється і добудовується, тому не можна виключати майбутнє розширення системи.

Високі стандарти безпеки і захищенності інформації - оскільки система містить велику кількість конфіденційної інформації, необхідно забезпечити відповідний рівень безпеки.

Аналізуючи ринок і виходячи з даних вимог, було обрано обладнання компанії Cisco - одного зі світових лідерів мережевого обладнання. У даному конкретному випадку, можна обрати маршрутизатор Cisco 2911 (Рис 3.6).

Рис 3.6 – Маршрутизатор Cisco 2911

Таблиця 3.3 - Характеристики Cisco 2911

Показники	Значення
Інтерфейси	4 x RJ 45 з пропускною здастністю до 1000 мбіт/с з можливістю додання до 4 карт розширення EWSIC 2 x USB
Форм фактор	Стійка 1U
Підтримувані протоколи передачі даних	Ethernet
Підтримувані протоколи маршрутизації	OSPF, IS-IS, BGP, EIGRP, DVMRP, PIM-SM, IGMPv3, GRE, PIM-SSM, IPV4, IPV6
Підтримувані міжнародні стандарти	IEEE 802.1Q, IEEE 802.3af, IEEE 802.3ah, IEEE 802.1ah, IEEE 802.1ag
Живлення	220-240V 50/60 гц
Особливості	Підтримка Firewall, підтримка створення VPN тунелів, підтримка апаратного шифрування, підтримка IPV6, наявність системного журналу і фільтрації вмісту пакетів

Для рівня доступу необхідно підібрати комутатори для об’єднання кінцевих пристроїв у підмережу. Для цієї задачі доцільно використовувати також коммутатори компанії Cisco.

Критеріями вибору комутатора є підтримка з’єднання 1000 мбіт/с і наявність достатньої кількості портів. На базі цих критеріїв було обрано коммутатор Cisco CBS350-16P-2G-EU (рис 3.7).

Рисунок 3.7 – Коммутатор Cisco CBS350-16P-2G-EU

Таблиця 3.3 - Характеристики CBS350-16P-2G-EU

Показники	Значення
Тип	Керований
Інтерфейси	16 x RJ 45 з пропускною здастністю до 100 мбіт/с 16 x RJ 45 з пропускною здастністю до 1000 мбіт/с
Форм фактор	Стійка 1U
Підтримувані протоколи передачі даних	Ethernet
Підтримувані протоколи маршрутизації	OSPF, IS-IS, BGP, EIGRP, DVMRP, PIM-SM, IGMPv3, GRE, PIM-SSM, IPV4, IPV6
Підтримувані міжнародні стандарти	IEEE 802.1Q, IEEE 802.3af, IEEE 802.3ah, IEEE 802.1ah, IEEE 802.1ag
Живлення	220-240V 50/60 гц

Обладнання системи керування доступом є найважливішою складовою системи. Зазвичай, для створення пункту пропуску мінімально необхідні три типа приладів:

- Контролер
- Зчитувач
- Електронний замок

Виробники систем СКУД створюють комплексні рішення, які включають усі типи вищезгаданих пристроїв, тому надзвичайно важливою ціллю є зважений підбір виробника пристроїв систем СКУД. Пристрої мають відповідати таким критеріям:

- Наявність відкритого інтерфейсу для розширення системи; пристрої мусять мати документацію і не бути повністю прив’язані до ПО виробника або надавати можливості програмної взаємодії
- Використання загальноприйнятих стандартів обміну інформації
- Забезпечення захисту інформації у мережі

Виходячи з наданих критерій оцінки обладнання, після аналізу ринку для використання у розроблювальній системи було обрано українську команію UProx. Існуючи на ринку з 1992 року, вона займає лідерські позиції у галузі виробництва систем СКУД. Великою перевагою використання продукції UProx є UProx IP - програмно апаратний комплекс для створення мережевої СКУД із подальшою можливість розширення через REST інтерфейс прикладного рівня.

У якості контроллера для системи було обрано контроллер UProx IP400.(рис 3.8)

Таблиця 3.4 - Характеристики UProx IP400

Показники	Значення
Мережевий інтерфейс	Ethernet

Кількість ідентифікаторів користувачів	До 31768
Додаткові інтерфейси	2 реле (NO, NC, COM), 2 реле (NO, COM) 8 програмованих шлейфів із контролем по струму: кнопка “вихід”, дверний контакт, пожежа, напад та інші охоронні шлейфи; тампер, контроль заряду акумулятора, контроль мережі живлення. Порт USB для мережевих налаштувань.
Живлення	220-240V 50/60 гц із можливістю встановлення резервного акумулятора
Особливості	Можливість синхронізації із системою UProx IP

Для того, щоб користувачі могли авторизуватися у системі, необхідне встановлення зчитувача на пропускних пунктах. Для цієї задачі було обрано зчитувач U-Prox SL mini (рис 3.9).

.

Рисунок 3.8 – Контроллер UProx IP400

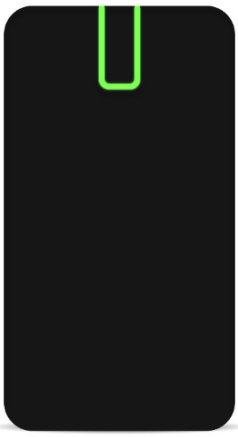


Рисунок 3.9 – Зчитувач U-Prox SL mini

Таблиця 3.5 - Характеристики U-Prox SL mini

Показники	Значення
Інтерфейси	Wiegand Auto, Wiegand 26-80, RS, TouchMemory
Підтримувані типи цифрових ключів	ASK/FSK/Mifare/Mifare Plus/радіоінтерфейс 2,4 ГГц/NFC.
Живлення	12V

Особливості	Режим синхронізації для близько розташованих зчитувачів.

За фізичне обмеження доступку у системі відповідають електронні замки, які зазвичай контролюються живленням 12 або 24 вольт.

Для встановлення на воротах і хвіртках пропускних пунктів було обрано замок SEVEN EL-7729 (рис 3.10).

Рисунок 3.10 – Електронний замок SEVEN EL-7729

Таблиця 3.6 - Характеристики SEVEN EL-7729

Показники	Значення
Матеріал корпусу	Хромована сталь
Тип встановлення	Накладний
Живлення	12V
Особливості	Наявність кнопки для виходу у випадку аварійних ситуацій

Таким чином, можна скласти загальну специфікацію обладнання (таблиця 3.7):

Таблиця 3.7 - Загальна специфікація обладнання

Позиція	Найменування і технічна характеристика	Тип, марка, позначення документа, опитувального листа	Одиниці виміру	Кількість
1	Dell R420	Сервер, Dell	шт	1
2	Dell E2720H	Монітор, Dell	шт	1
3	Logitech MK120	Периферійне обладнання, Logitech	шт	1
4	Cisco 2911	маршрутизатор, cisco	шт	1
5	Cisco CBS350-16P-2G-EU	коммутатор, cisco	шт	4
6	UProx IP400	Контроллер СКУД, UProx	шт	4
7	UProx SL mini	Зчитувач, UProx	шт	8
8	SEVEN EL-7729	Електронний замок, SEVEN	шт	4

3.5 Висновки до розділу

В даному розділі були розроблені вимоги до створюваної системи контролю і управління, які мусять гарантувати надійність і стабільне функціонування системи. Була розроблені схема функціональної структури, яка описує логічні зв’язки між складовими системи. На основі заданих критеріїв до видів забезпечення було підібране обладнання, необхідне для створення системи СКУД на об’єкті.

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Призначення й область застосування програмного забезпечення

Призначенням програмного-технічного комплексу є забезпечення централізованого управління пристроями і моніторингу статусу журналу доступу для адміністратора і можливості віддаленого відкриття замків на пропускних пунктах для користувача (мешканця котеджного містечка).

Виробником обладнання СКУД UProx, продукція якого була обрана для створення системи у попередньому розділі, пропонується програмно-технічний комплекс UProx IP [6] для створення такої централізованої системи. Ця система надає такі можливості:

- Під'єднання та управління контроллерами UProx
- Налаштування контроллерів і ассоціація їх з електронними замками на пунктах пропуску
- Додавання користувачів і адміністраторів у систему
- Створення груп користувачів з правами доступу до одного або декількох пунктів пропуску
- Автоматичне ведення і можливість перегляду журналу доступу користувачів до об’єкту
- Наявність інтерфейсу для програмної взаємодії інших застосунків з компонентами системи

Таким чином, запропонований виробником обладнання технічний комплекс покриває потреби системи із адміністрування приладів і користувачів і його можна використовувати як основну адміністративну панель.

Недоліком запропонованої системи є відсутність інтерфейсу користувача для зручного віддаленого користування дверима на пропускних пунктах.

Оскільки у системи виробника, що лежить у основі програмно-технічного комплексу, існує відкритий інтерфейс для взаємодії з іншими застосунками, призначенням розроблюваного програмного забезпечення буде розширення системи виробника і створення інтерфейсу користувача.

Програмне забезпечення СКУД повинно бути встановлено на центральний сервер, який знаходиться у одній мережі з усіма іншими пристроями СКУД.

Область застосування – котеджне містечко “Sun Coast Dnipro”.

4.2 Обґрунтування технічних характеристик системи

Система складатиметься із трьох частин:

- Застосунку UProx IP, який виконує функцію сервісу керування пристроями і включає браузерний клієнт для роботи адміністратора
- Інтерфейсу користувача, що дозволить віддалено взаємодіяти з системою мешканцям містечка
- Бази даних для зберігання інформації із можливістю резервних копій

Застосунок UProx IP постачається компанією UProx у вигляді інсталлятора для операційної системи Windows. Для початку роботи адміністратору необхідно встановити застосунок як системний сервіс і провести початкове налаштування користувачів і приладів. Перевагами використання готового рішення є те, що його взаємодія з приладами є уже протестованою виробником.

Для работи UProxIP необхідне використання бази даних MSSQL версій 2014, 2016 або 2019. Рекомендованою для стабільності роботи є версія MSSQL 2014.

У якості інтерфейсу користувача було обрано створення бота для мессенджеру Telegram. Перевагами такого підходу є:

- Спрощення аутентифікації - у якості ідентифікатора для прив’язки користувача Telegram до користувача системи може бути використаний номер мобільного телефону, який можливо отримати на запит від API Telegram. Сеcсія зберігається на сервері Telegram.
- Підтримка пристроїв - у Telegram є кліенти як для стаціонарних комп’ютерів і ноутбуків, так і мобільних телефонів користувачів. Ще однією перевагою для мобільних телефонів є відсутність необхідності встановлювати додатковий застосунок, адже цей мессенджер є одним із найбільш популярних в Україні.
- Можливість створення простого адаптивного інтерфейсу - за допомогою API можливе створеннння простого кнопкового інтерфейсу, який буде зручним на усіх платформах і типах пристроїв.

Для реалізації вихідного коду бота було обрано мову програмування Javascript з інструментарієм TypeScript [4] для типізації коду. Мова програмування JavaScript забезпечує високу швидкість розробки, достатню продуктивність і забезпечує можливості кросплатформеності для застосунку.

Для спрощення взаемодії бота з системою буде використан Telegraf [5] – бібліотека для мови JavaScript, що спрощує написання ботів для Telegram.

Проміжні дані для боту Telegram будуть зберігатися окремо від основної бази даних, бо застосунок виробника UProx IP при кожному перезавантаженні системи перевіряє структуру бази даних і видаляє невідомі таблиці, що може призвести до втрати даних. Оскільки кількість даних невелика, для цієї задачі доцільно використати SQL базу даних SqLite [7]. Її особливість полягає у тому, що базу даних можна зберігати у якості одного файлу.

4.3 Опис розробленої програми

4.3.1 Загальні відомості

Розроблюваним програмним забезпеченням є Telegram бот для системи СКУД у якості інтерфейсу користувача. Воно виступає у якості інтерфейсу користувача для системи UProx. Програму написано мовою JavaScript.

4.3.2 Функціональне призначення

Функції, які буде виконувати застосунок:

– аутентифікація користувачів;

– перегляд доступних дверей для користувача;

– відкриття дверей за допомогою інтерфейсу бота;

4.3.3 Установка і налаштування системи

Після встановлення операційної системи, необхідно буде зробити початкове налаштування системи. Для цього необхідно покроково виконати наступні дії:

1) Встановлення комплексу UProx (рис 4.1)

Користуючись наданим виробником обладнання інсталлятором, необхідно встановити компоненти системи у такому порядку: Microsoft SQL Server, UProx IP Server, UProx Web Server.

Рисунок 4.1 – Встановлення UProx

2) Вхід до адміністративної панелі

Після встановлення системи Web сервіс буде знаходитися на порту 40001 сервера. Для подальшого налаштування системи необхідно виконати вхід у систему як адміністратор. Для цього необхідно використати логін/пароль за замовчуванням (admin).

Рисунок 4.2 – Вхід до адміністративної панелі

Після входу можна буде побачити головну сторінку системи, через яку можна проводити подальші налаштування.

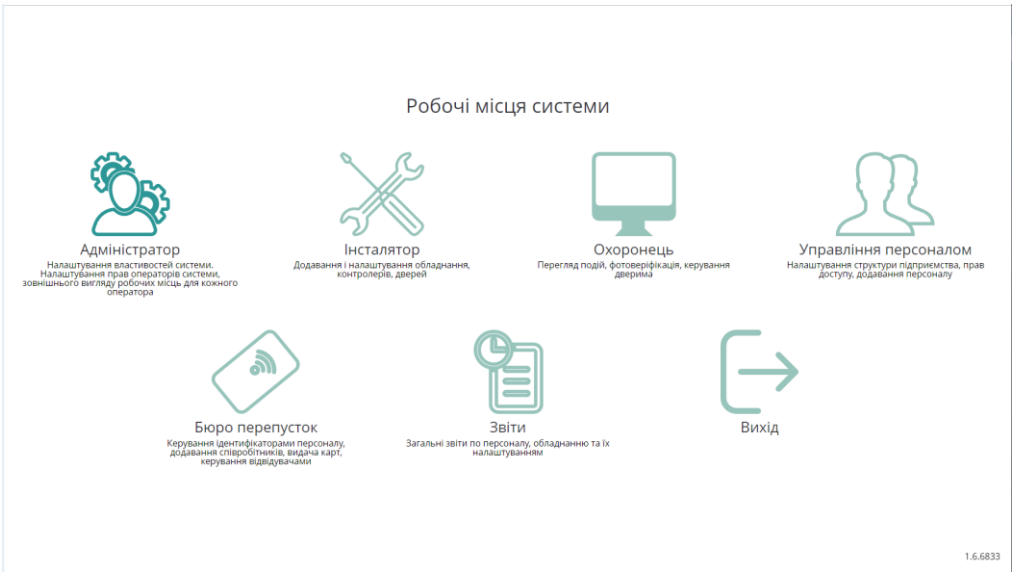


Рисунок 4.3 – Головна сторінка UProx

3) Додавання мешканців у систему

Мешканці і групи мешканців додаються до системи за допомогою сервісу «Управління персоналом» з головної сторінки. Спершу необхідні створити группу для мешканців, яка буде визначати їх права доступу (рис 4.4)

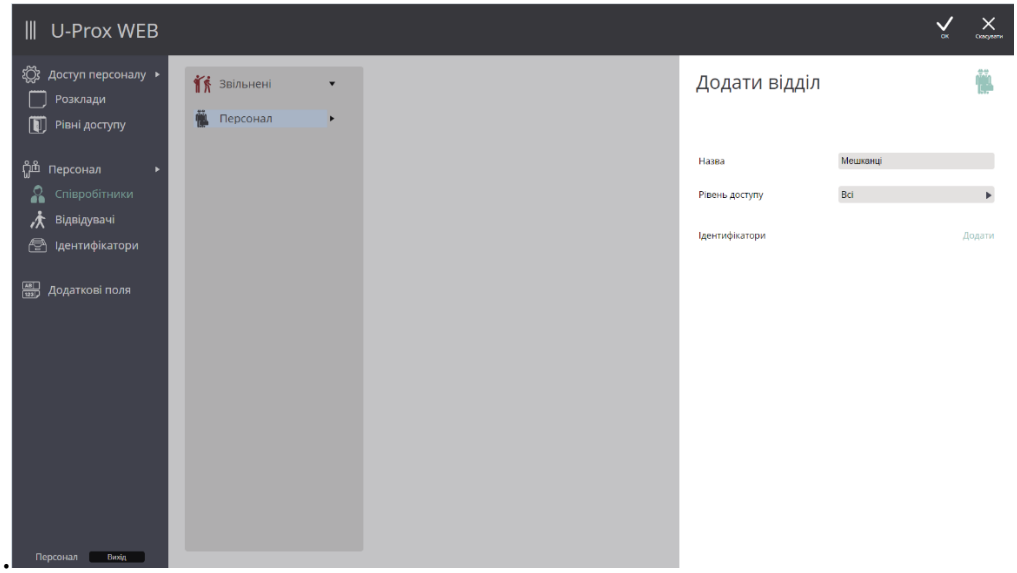


Рисунок 4.4 – Створення групи мешканців

Для правильної роботи бота Telegram необхідно щоби користувачі ідентифікувалися за допомогою телефону. Для цього необхідно додати додаткові поля до моделі користувача у системі (рис 4.5).

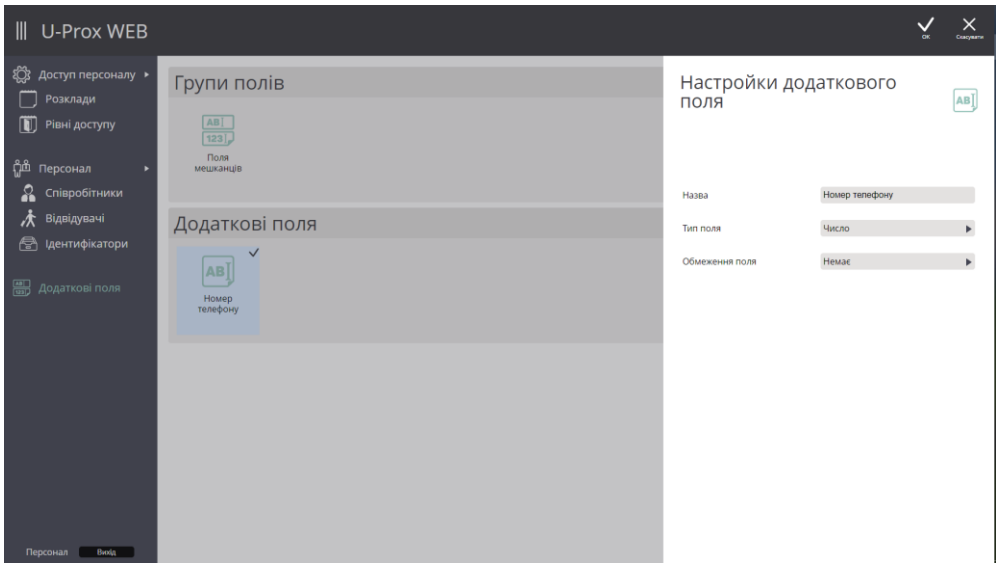


Рисунок 4.5 – Додавання номеру телефону у якості додаткового поля

Після цього можна перейти до безпосереднього додавання мешканців котеджного містечка (рис 4.6).

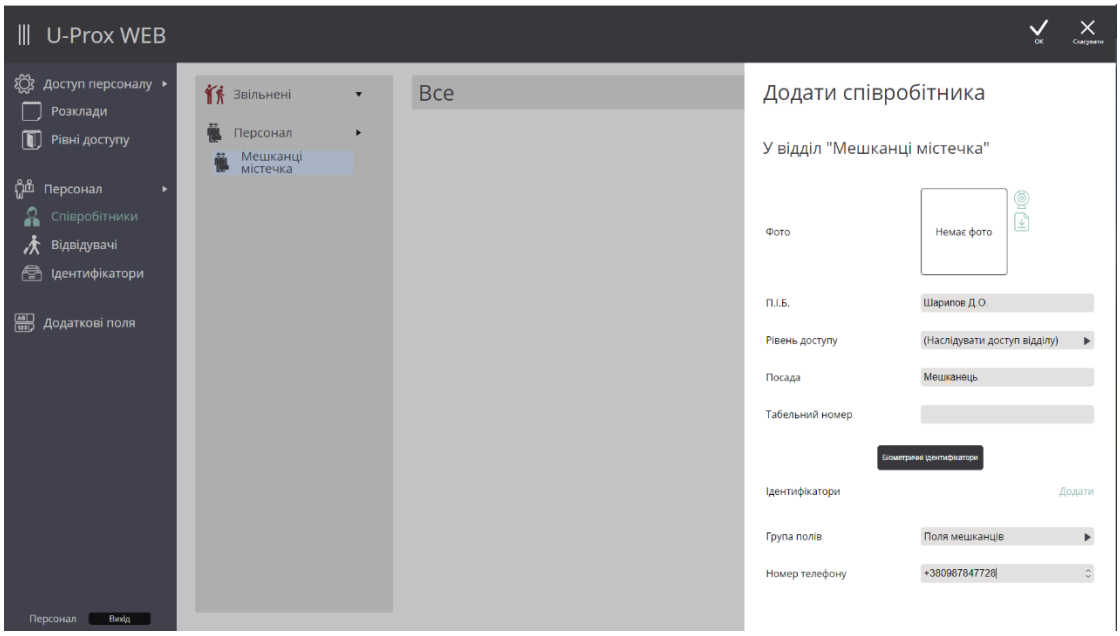


Рисунок 4.6 – Додавання мешканців

4) Реєстрації пристроїв СКУД у системі

Після створення системи треба під’єднати до неї встановлені на об’єкті пристрої. Вони додаються до системи за допомогою сервісу «Інсталятор» з головної сторінки. Спочатку необхідно під’єднати контролери (рис 4.7)

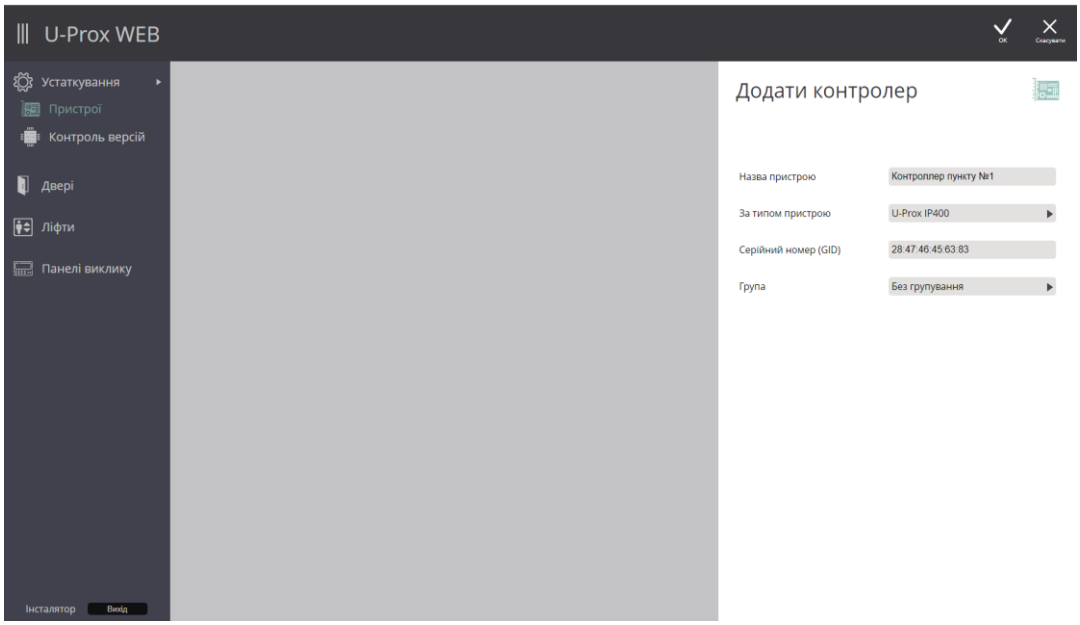


Рисунок 4.7 - Реєстрація контролерів

Після реєстрації контролерів, необхідно описати під’єднані до них двері, оскільки один контролер може контролювати одну або декілька дверей чи воріт. Цей процес зображено на рис 4.8.

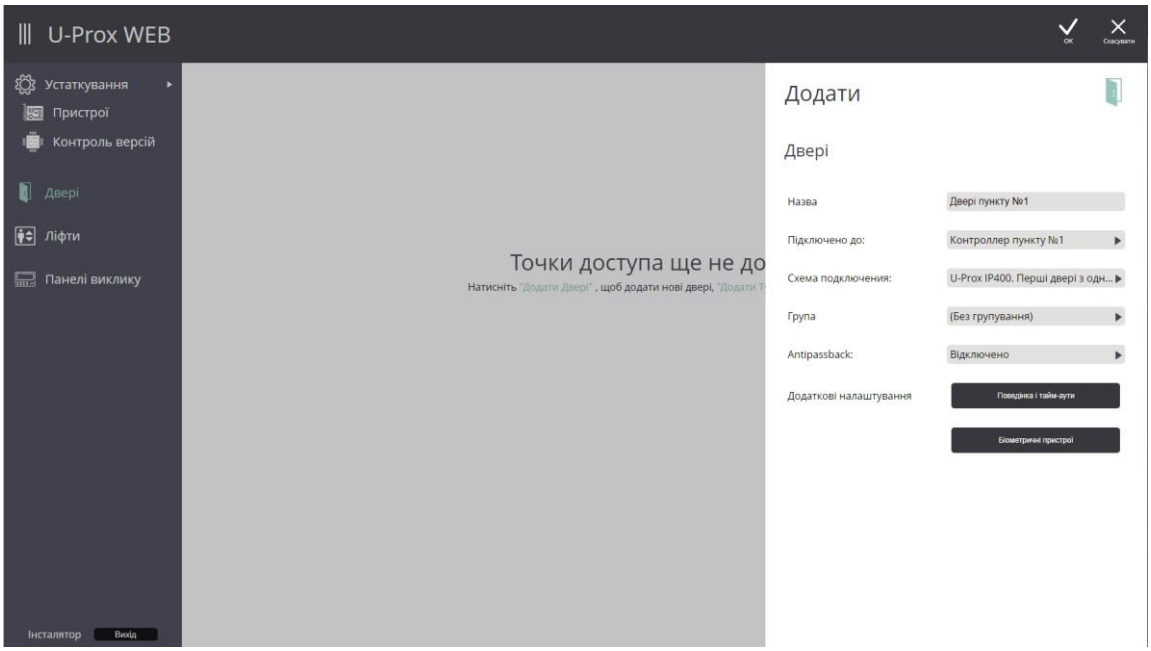


Рисунок 4.8 - Реєстрація дверей

Для роботи бота Telegram також необхідно встановити NodeJS 16 версії – програму для виконання скриптів, написаною мовою JavaScript.

Після цього процес початкового налаштування системи можна вважати закінченим.

4.3.4 Опис логічної структури програми

При ініціалізації програми створюється процес, що очікує повідомлення від API Telegram. Отримувати повідомлення від Telegram можна двома спообами:

- сповістити Telegram про адрес і порт, на який необхідно висилати оновлення
- постійно відсилати запрос до серверів Telegram для отримання нових повідомлень (так званий полінг)

Оскільки всі порти сервера вважаються закритими для глобальної мережі, необхідно використовувати полінг.

Для зв'язку із системою UProx бот використовує REST API UProx Web – інтерфейс управління системою через HTTP запити.

При першому використанні бота користувача буде запропоновано відіслати власний контакт із номером телефону для аутентифікації у системі. Всі повідомлення користувача, які не містять контактної інформації будуть проігноровані. Після отримання контактної інформації від користувача бот повинен використовуючи метод API UProx отримати список всіх користувачів і серед них знайти відповідного користувача бота. Якщо такого користувача знайдено, то інформація про користувача записується у базу SqLite, а у інтерфейсі він отримує повідомлення про аутентифікацію. Після аутентифікації користувач ідентифікується у системі за номером чату.

Якщо користувач вже аутентифікований, йому необхідно надіслати список доступних йому пунктів пропуску. Для цього робиться запит до API UProx з метою отримання доступних дверей. Після цього користувачу надсилається список цих дверей у вигляді кнопок. При натискані користувачем однієї з цих кнопок боту надсилається команда. Отримавши таку команду, бот повинен відправити запит на відкриття дверей на API UProx з номером обраної двері. Якщо такі двері будуть знайдені, вони відкриються, а користувачу прийде про це повідомлення. Після цього він знову отримає список доступних дверей для подальшої взаємодії.

Таким чином, можна сформувати два основних алгоритми дії бота:

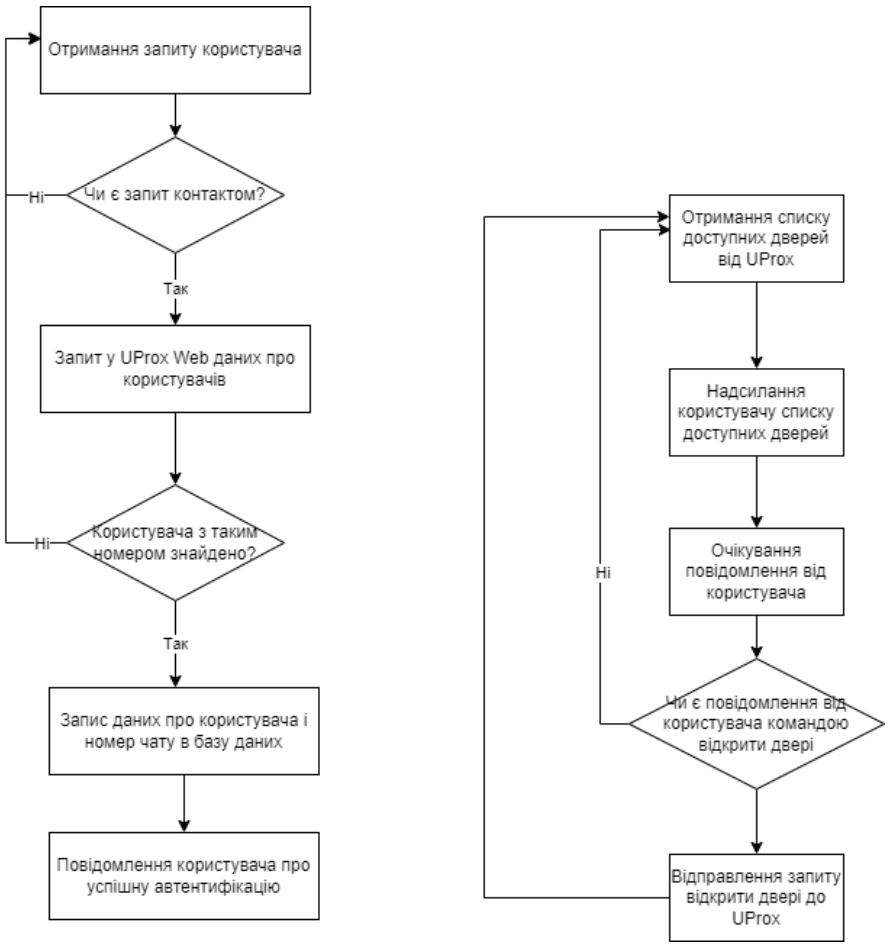


Рисунок 4.9 - Алгоритм «Аутентифікація» (зліва) і алгоритм «Взаємодія з дверима» (справа).

Їх можна об’єднати таким чином:

Рисунок 4.10 - Загальний алгоритм роботи бота

За сформованими алгоритмами була розроблена програма, текст якої наведено у Додатку А (77 сторінка пояснювальної записки).

Текст програми розбитий на декілька файлів за логічними складовими:

- datasource.ts – ініціалізація підключення до бази даних
- entity/Log.ts – модель таблиці логів застосунку у базі даних
- entity/UserChat.ts – модель таблиці ідентифікованих користувачів у базі даних
- uproxService.ts – програмний клас для взаємодії з сервісом UProx
- bot.ts – ініціалізація бота Telegram та алгоритму його поведінки
- index.ts – точка входу
- package.json – опис залежностей для NPM

4.3.5 Опис інтерфейсу користувача

При першому відвідуванні сторінки бота користувачу пропонується розпочати з ним роботу (рис 4.11):

Рисунок 4.11 - Початок роботи

Після надсилання першого повідомлення, користувачу надається запит відправити контактні дані і кнопка для цього:

Рисунок 4.12 - Перше повідомлення

Якщо користувач відправить не власні контакти, його запит буде відхилено:

Рисунок 4.13 - Відправлення чужого контакту

Якщо користувача системи з таким номером телефона як у користувача буде знайдено, він отримає список доступних йому дверей:

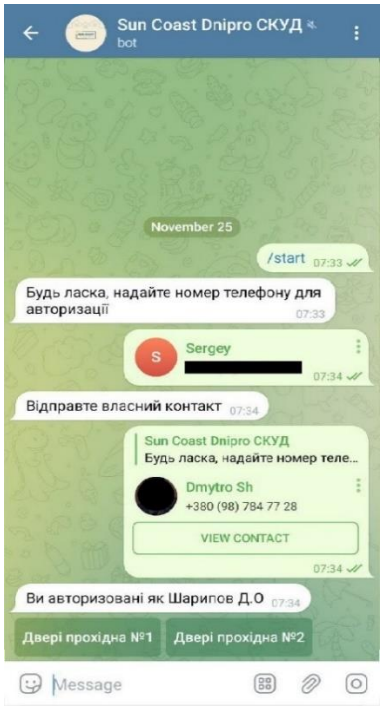


Рисунок 4.14 - Авторизація у системі

При натисканні на одну з наданих кнопок дверей відбудеться їх відкриття на об’єкті, а список доступних дверей оновиться.

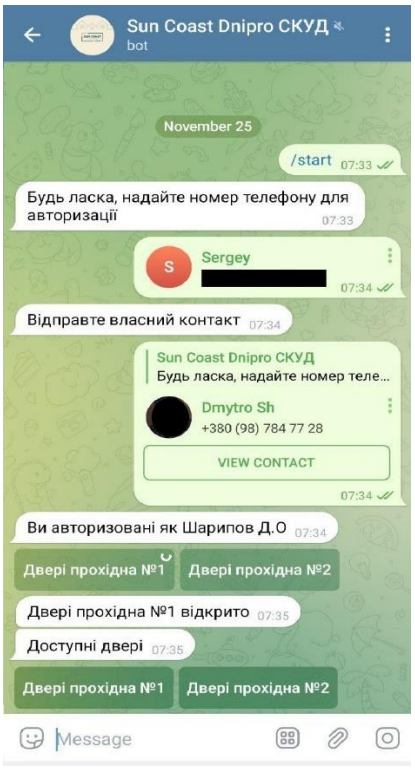


Рисунок 4.15 - Відкриття дверів

4.3.6 Виклик і завантаження

Для встановлення залежностей у Telegram боті було використано менеджер залежностей NPM [8]. Для встановлення залежностей потрібно використати команду “npm install”.

Для запуску застосунку необхідно використати команду “npm run start”.

4.3.7 Вхідні і вихідні дані

Для взаємодії з користувачем використовується API Telegram, для взаємодії з сервісом UProx – API UProx Web. Для передачі даних використовується протокол прикладного рівня HTTP, формат передачі даних – JSON.

Змінні програми знаходяться у файлі .env у кореневій папці бота.

Існують такі змінні:

- TG_TOKEN - токен API Telegram для підписування запитів від бота
- UPROX_USERNAME – логін користувача, через якого відбувається взаємодія з UProx Web API
- UPROX_PASSWORD – пароль користувача, через якого відбувається взаємодія з UProx Web API

4.4 Очікувані техніко-економічні показники

Використання розробленого програмного забезпечення і системи UProx IP у котеджному містечку дозволить:

- збільшити ефективність безпеки об’єкту через автоматизацію процесу контролю доступу і таким чином зменшити кількість необхідних охоронців
- покращити якість моніторингу доступу на об’єкт через автоматичне ведення журналу доступу
- збільшити комфорт мешканців містечка, надавши їм можливість більш безпечного проживання на території містечка, при цьому не

викликаючи незручностей щодо переміщення, що підвищить конкурентоспроможність містечка

4.5 Висновки до розділу

Для вирішення поставленої задачі, були задані функціональні вимоги та обґрунтовано технічні характеристики. У якості технічного завдання програмно-технічного комплексу, було визначено та виконано такі цілі:

— створення і налаштування централізованого сервісу управління системою контролю і управління доступу за допомогою застосунку UProx IP.

— розроблення користувацького інтерфейсу системи у якості боту Telegram для віддаленого керування дверима для мешканців містечка.

ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Формулювання завдання та мети експерименту

Мета експерименту: підтвердження відповідності роботи системи контролю і управління доступом очікуваним показникам. Перевірка правильного функціонування системи. Аналіз відповідності реальних алгоритмів роботи системи очікуваним показникам.

Задача експерименту: використовуючи метод тестування перевірити поведінку системи в умовах реального використання. Порівняти з очікуваним результатом. Виявити недоліки у алгоритмах роботи системи.

5.2 Методика проведення експерименту

Для перевірки отриманої системи було обрано метод тестування. Суть методу тестування полягає у тому, що експериментатор або група експериментаторів отримає доступ до готової системи у режимі, наближеному до реального використання. Після цього дослідники мають перевірити заздалегідь підготовлені сценарії, що мають на меті підтвердити заявлені характеристики системи. Висновок про стан системи робиться на базі кількісної відповідності вдалих експериментів відносно до кількості заявлених. Есперимент вважається успішним якщо переважаюча більшість (95-99%) заявлених сценаріїв може бути успішно відтворена у системі. Сценарії, які не можуть бути виконані є підставою для доопрацювання системи.

Виділяють такі підтипи тестування:

- функціональне тестування – перевірка відповідності системи запитам замовника
- тестування стабільності – перевірка стабільності системи при нештатних умовах використання

– тестування інтерфейсу – перевірка інтерфейса на зручність
– тестування безпеки – перевірка захисту даних
– тестування навантаження – перевірка стабільності системи при великому обсягу запитів

Для перевірки вимог до системи, описаних у попередніх розділах, необхідно провести 2 експерименти:

– групове тестування вибірковою групою
– функціональне тестування

5.2.1 Методика групового тестування вибірковою групою

Цей експеримент має на меті провести тестування стабільності, тестування безпеки, тестування навантаження і тестування інтерфейсу.

Для проведення експерименту було відібрано 20 мешканців котеджного містечка, які згодились брати у ньому участь. Вони будуть виступати дослідниками.

Перед дослідниками було поставлено задачі взаємодії з системою:

– отримання доступу до системи за допомогою nfc мітки на пунктах пропуску
– отримання доступу до системи за допомогою розробленого Telegram боту

Для цього для дослідників було створено облікові записи у системі та надано зареєстровані ідентифікатори. Чверть дослідників отримають незареєстровані ідентифікатори. Їхною ціллю буде отримання несанкціонованого доступу до системи. Експеримент має проводитися 1 годину, після цього можна перейти до аналізу результатів.

Для аналізу результатів експерименту доцільно використовувати такі засоби:

– логи системи UProx Web

- журнал доступу системи UProx Web
- логи розробленого бота Telegram

Після виконання експерименту дослідникам необхідно пройти опитування для суб’єктивної оцінки зручності та ефективності функціонування системи. Для виконання цього завдання користувачам необхідно зазначити власну оцінку системи за шкалою від 1 до 5 за такими чинниками:

- загальна зручність користування
- швидкість роботи системи
- вплив розробленої системи на безпекову систуацію на об’єкті впровадження

Висновок про успішне тестування стабільності можна скласти, якщо не відбудеться програмної помилки у системі під час тестування, що призведе до відмови оброблення запиту. Висновок про успішне тестування безпеки можна скласти, якщо ніхто з користувачів з недійсними ідентифікатора не зможе отримати доступ .Висновок про успішне тестування навантаження можна скласти, якщо під час одночасного тестування системи 20 користувачам не відбудеться затримки більше ніж 5 секунд при обробленні запитів. Висновок про успішне тестування зручності інтерфейсу формується на базі опитування дослідників як вибіркової групи подальших користувачів.

5.2.2 Методика функціонального тестування

Метою проведення функціонального тестування є перевірка алгоритмів розробленої програми. Сценарії для тестування формуються на базі вимог синтезованої системи. Дослідник має перевірити кожен сценарій окремо і порівняти поведінку системи із очікуваними. Після цього дані заносяться в таблицю для подальшого аналізу.

Функціонально тестувати систему UProx немає необхідності, адже вона вже протестована виробником обладнання. Для тестування розробленої програми користувацького інтерфейсу було обрано такі сценарії:

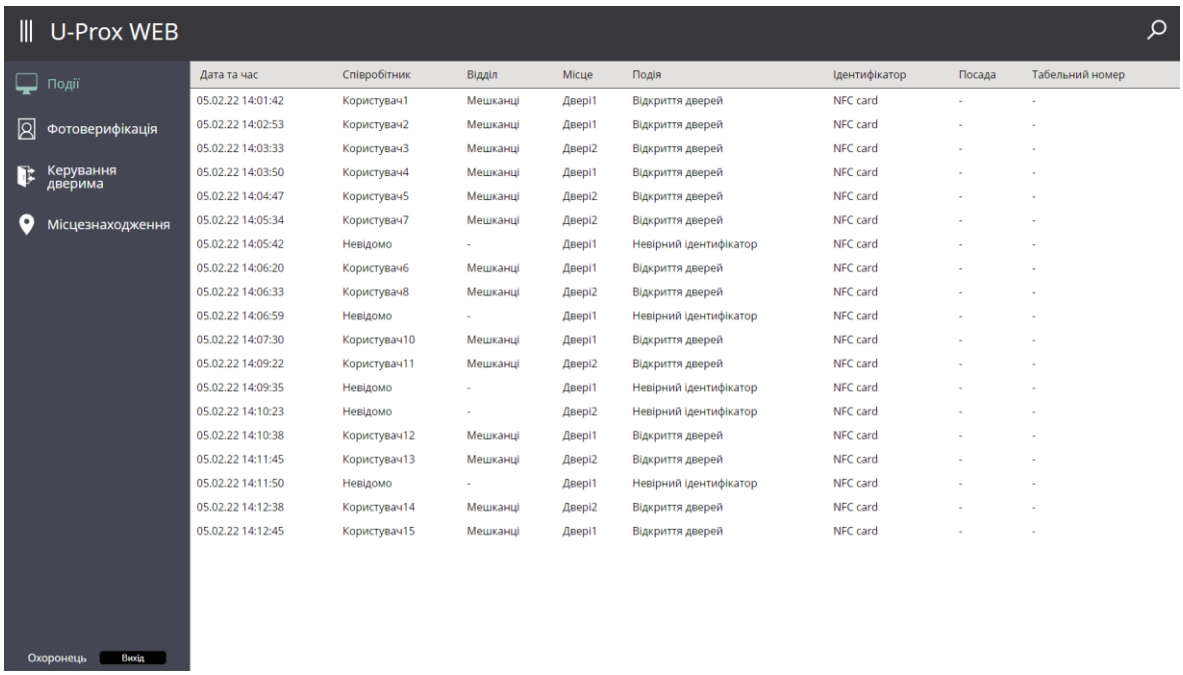
- Користувач може успішно авторизуватись за умови наявності про нього інформації в системі UProx
- Користувачу не надається доступ якщо його не знайдено у системі
- Користувач не може авторизуватись повторно
- Користувач не може відправити дані іншого користувача
- Користувач може відкривати двері
- Список дверей доступних користувачу може бути оновлений
- Користувач не може відкривати двері якщо його обліковий запис було заблоковано адміністратором

Висновок про успішне функціональне тестування системи можна зробити, якщо поведінка системи відповідатиме всім заявленим сценаріям.

5.3 Аналіз результатів

5.3.1 Результати групового тестування вибірковою групою

Спочатку розглянемо дані журналу доступу системи системи UProx:



Дата та час	Співробітник	Відділ	Місце	Подія	Ідентифікатор	Посада	Табельний номер
05.02.22 14:01:42	Користувач1	Мешканці	Двері1	Відкриття дверей	NFC card	-	-
05.02.22 14:02:53	Користувач2	Мешканці	Двері1	Відкриття дверей	NFC card	-	-
05.02.22 14:03:33	Користувач3	Мешканці	Двері2	Відкриття дверей	NFC card	-	-
05.02.22 14:03:50	Користувач4	Мешканці	Двері1	Відкриття дверей	NFC card	-	-
05.02.22 14:04:47	Користувач5	Мешканці	Двері2	Відкриття дверей	NFC card	-	-
05.02.22 14:05:34	Користувач7	Мешканці	Двері2	Відкриття дверей	NFC card	-	-
05.02.22 14:05:42	Невідомо	-	Двері1	Невірний ідентифікатор	NFC card	-	-
05.02.22 14:06:20	Користувач6	Мешканці	Двері1	Відкриття дверей	NFC card	-	-
05.02.22 14:06:33	Користувач8	Мешканці	Двері2	Відкриття дверей	NFC card	-	-
05.02.22 14:06:59	Невідомо	-	Двері1	Невірний ідентифікатор	NFC card	-	-
05.02.22 14:07:30	Користувач10	Мешканці	Двері1	Відкриття дверей	NFC card	-	-
05.02.22 14:09:22	Користувач11	Мешканці	Двері2	Відкриття дверей	NFC card	-	-
05.02.22 14:09:35	Невідомо	-	Двері1	Невірний ідентифікатор	NFC card	-	-
05.02.22 14:10:23	Невідомо	-	Двері2	Невірний ідентифікатор	NFC card	-	-
05.02.22 14:10:38	Користувач12	Мешканці	Двері1	Відкриття дверей	NFC card	-	-
05.02.22 14:11:45	Користувач13	Мешканці	Двері2	Відкриття дверей	NFC card	-	-
05.02.22 14:11:50	Невідомо	-	Двері1	Невірний ідентифікатор	NFC card	-	-
05.02.22 14:12:38	Користувач14	Мешканці	Двері2	Відкриття дверей	NFC card	-	-
05.02.22 14:12:45	Користувач15	Мешканці	Двері1	Відкриття дверей	NFC card	-	-

Рисунок 5.1 - Журнал доступу після експерименту

Аналізуючи зміст журналу після проведення експерименту, можна прийти до висновків:

- Кількість успішних подій входу (не включаючи доступу за допомогою боту Telegram) співпадає з заявленою кількістю користувачів з реальними ідентифікаторами
- Кількість відхилених запитів входу відповідає кількості користувачів, які отримали незареєстровані ідентифікатори

Виходячи з цього, можна зробити висновок, що система виконує відповідні функції обмеження доступу за допомогою ідентифікаторів.

Далі необхідно проаналізувати логи UProx. Логи знаходяться на сервері у папці C:\ProgramData\A2SoftIn\Ess\Logs\. Після проведення тестування нові записи про помилки у логах не створені. Під час роботи системи нештатних ситуацій не відбувалось. На основі цього можна зробити висновок що система стабільна.

Для оцінки роботи розроблюваного застосунку користувацького інтерфейсу необхідно також проаналізувати і логи бота Telegram. Логи знаходяться у базі даних SQLite у таблиці Logs.

Для аналізу даних необхідно агрегувати дані за типом події. Результуюча таблиця має такий вигляд:

Таблиця 5.1 - Агрегація логів розробленої програми

Тип події	Кількість
Успішна авторизація	15
Користувача не знайдено	5
Надано чужі контакти	5
Спроба повторної авторизації	1
Відкриття дверей	18

Аналізуючі надані результати, можна прийти до висновку:

- Кількість подій успішної авторизації співпадає із кількістю створених користувачів у системі
- Кількість подій відсутності користувача відповідає кількості дослідників, що не мали облікових записів
- Кількість подій надання чужих контактів відповідає очікуваним, адже дослідники без облікових записів надавали чужі контакти для авторизації
- Одним із дослідників була випадкова проведена спроба повторної авторизації
- Усі дослідники змоги відкрити двері за допомогою бота Telegram; деякі з них здійснили це декілька разів.

Загалом, показники відповідають очікуваним.

Після проведення експерименту було проведене опитування щодо суб’єктивної оцінки роботи системи контрольною групою мешканців містечка. Із результатів опитування сформовано таблицю:

Таблиця 5.2 - Результати опитування

Номер користувача	Оцінка загальної зручності системи	Оцінка швидкості системи	Оцінка впливу системи на безпеку об'єкта
1	5	5	5
2	4	3	5
3	5	3	3
4	3	5	3
5	5	5	5
6	3	4	3
7	5	4	5
8	4	4	4
9	4	2	3
10	3	3	5
11	3	3	5

12	5	4	4
13	5	5	5
14	3	5	5
15	4	3	3
16	5	4	3
17	3	3	4
18	5	5	4
19	5	5	3
20	4	5	5
Середня величина	**4,15**	**4**	**4,1**

Якщо враховувати середню величину оцінок мешканців містечка, система забезпечує задовільний рівень комфорту споживачів.

5.3.2 Результати тестування очікуваних сценаріїв роботи системи

Для більш детальної перевірки правильного відпрацюванню алгоритмів розробленої програми необхідно також вручну перевірити сценарії її роботи, що були синтезовані у попередній частині розділу. Після проведення експериментів було створено таблицю результатів:

Таблиця 5.3 - Результати функціонального тестування

Номер сценарію	Очікуваний результат	Реальний результат	Відповідність реального результату очікованому
1	Користувач може успішно авторизуватись за умови наявності про нього інформації в системі UProx	Після надання контактних даних, користувач отримує ім’я авторизованого облікового запису і список доступних дверей	Так

2	Користувачу не надається доступ якщо його не знайдено у системі	Після відправлення контактних даних, користувачу приходить повідмолення про відсутність користувача у системі	Так
3	Користувач не може авторизуватись повторно	Після відправлення контактних даних, користувачу приходить повідомлення про те що він вже авторизований і список доступних дверей	Так
4	Користувач не може відправити дані іншого користувача	Після відправлення контактних даних, користувачу приходить повідомлення що він мусить відправити власні контактні дані	Так
5	Користувач може відкривати двері	Після натискання на кнопку із іменем двері, двері будуть відкриті а користувач отримає про це повідомлення	Так
6	Список дверей доступних користувачу може бути оновлений	Після відкриття будь-якої двері за допомогою інтерфейсу бота, користувачу	Так

		відправляється новий список доступних дверей	
7	Користувач не може відкривати двері якщо його обліковий запис було заблоковано адміністратором	Після натискання на кнопку із іменем двері, користувач отримає повідомлення що відкрити двері не вдалося і пропозицію звернутися до адміністратора	Так

Оскільки всі реальні результати сценаріїв збігаються з очікуваними, можна прийти до висновку що функціональне тестування успішне.

5.3 Характеристика новизни результатів

Наукове значення експерименту, який проводився з системою СКУД полягало у перевірці правильної роботи системи, обґрунтуванні відповідності системи поставленим вимогам і обгрунтуванню доцільності використання системи на об’єкті впровадження. Практичне значення результатів експерименту полягає в тому, що система задовільняє висунуті до неї вимоги як засіб забезпечення охорони об’єкту.

5.4 Висновок до розділу

У даному розділі було проведено випробування системи для виявлення недоліків у її роботи. Для перевірки системи було використано метод тестування, а саме:

- Групове тестування вибірковою групою
- Функціональне тестування

У результаті тестування системи проблем, що впливали би на стабільність і адекватність роботи системи не виявлено. Опитування контрольної групи виявило задовільний рівень сприйняття системи майбутніми користувачами.

ВИСНОВКИ

Кваліфікаційна робота є завершеною науковою роботою, в якій вирішена задача розробки комплексу програмно-технічних засобів системи контролю і управління доступом котеджного містечка Sun Coast Dnipro. Дана система була розроблена за рахунок аналізу типових засобів побудови подібних систем, розробки моделей, використання методів аналізу та синтезу а також розробки програмного забезпечення.

Основні результати роботи полягають у наступному:

1. Використовуючи методи аналізу та синтезу на основі відомих рішень систем СКУД були створені вимоги до розроблювальної системи і створено комбіновану модель системи.

2. Під час синтезу системи, були запропоновані і обгрунтовані принципи побудови і проектування, на основі них було сформовано структурну і функціональну схеми.

3. На основі сформульованих вимог було здійснено підбір мережевого і серверного обладнання, підібрано виробника пристроїв СКУД для подальшої реалізації системи.

4. На основі підібраного обладнання і програмного забезпечення від виробника на системі створено СКУД мережевого типу, що відповідає синтезованим вимогам.

5. Для допрацювання отриманої системи було створено інтерфейс користувача за допомогою мови програмування JavaScript, що дозволить користувачам взаємодіяти з системою віддалено.

6. Виконано експеримент, метою якого є перевірка стабільності, надійності, ефективності та зручності системи за допомогою функціонального та групового тестування. Експеримент пройшов вдало.

7. Результатом експерименту є висновок про доцільність використання створеної системи на об'єкті впровадження для покращення рівня безпеки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Скінченний автомат.
   URL: https://uk.wikipedia.org/wiki/Скінченний_автомат
2. Telegram
   URL: https://telegram.org
3. Javascript
   URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript
4. Typescript
   URL: https://www.typescriptlang.org
5. Telegraf
   URL: https://telegraf.js.org
6. UProx IP
   URL: https://access.u-prox.systems/uk/
7. База даних SqLite
   URL: https://www.sqlite.org/index.html
8. NPM
   URL: https://www.npmjs.com
9. Журавська І. М. Проектування та монтаж локальних комп'ютерних мереж : навчальний посібник / І. М. Журавська. – Миколаїв : Видавництво ЧДУ ім. Петра Могили, 2016. – 396 с.
10. Жуков, І. А. Комп'ютерні мережі та технології : навч. посіб./І. А. Жуков, В. О. Гуменюк, І. Є. Альтман. – К. : НАУ, 2004. – 276 с.
11. Розробка програмного забезпечення комп'ютерних систем. Програмування: навч. посібник / Л.І. Цвіркун, А.А. Євстігнєєва, Я.В. Панферова. – 2-ге вид., випр. – Д.: Національний гірничий університет, 2011. – 222 с.

ДОДАТОК А

Текст розробленої програми

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ДЛЯ СИСТЕМИ КОНТРОЛЮ І УПРАВЛІННЯ ДОСТУПОМ

ТЕКСТ ПРОГРАМИ

804.02070743.22018-01 12 01

Листів 17

2022

АНОТАЦІЯ

У цьому документі представлено код програмного бота Telegram для системи СКУД котеджного містечка Sun Coast Dnipro.

Програми реалізована використовуючи програмну мову JavaScript із використанням компілятора TypeScript.

ЗМІСТ

package.json – опис команд запуску та використаних бібілотек

```
{
  "name": "skud-suncoast-bot",
  "version": "0.0.1",
  "type": "commonjs",
  "devDependencies": {
    "@types/node": "^16.11.10",
    "ts-node": "10.7.0",
    "typescript": "4.5.2"
  },
  "dependencies": {
    "dotenv": "^16.0.3",
    "md5": "^2.3.0",
    "node-fetch": "^2.6.6",
    "reflect-metadata": "^0.1.13",
    "rimraf": "^3.0.2",
    "sqlite3": "^5.1.2",
    "telegraf": "^4.11.2",
    "telegram-keyboard": "^2.3.2",
    "typeorm": "0.3.10"
  },
  "scripts": {
    "clean-start": "npm run reset && npm run start",
    "start": "ts-node src/index.ts",
    "reset": "rimraf local.db",
    "test": "ts-node src/testScenario.ts"
  }
```

```
}
```

src/index.ts – точка входу застосунку

```
import { AppDataSource } from "./datasource"
import * as dotenv from 'dotenv'
import { UserChat } from "./entity/UserChat"
import { createSkudBot } from "./bot";
import { UProxService } from "./uproxService";

dotenv.config()

async function main() {
    await AppDataSource.initialize();

    const service = new UProxService(process.env.UPROX_USERNAME,
process.env.UPROX_PASSWORD);
    const bot = createSkudBot(process.env.TG_TOKEN, service);

    bot.launch();

    console.log('Bot started');
}

main();
```

src/datasource.ts – ініціалізація підключення до бази даних

```
import "reflect-metadata"
import { DataSource } from "typeorm"

export const AppDataSource = new DataSource({
```

```
    type: 'sqlite',
    database: 'local.db',
    synchronize: true,
    entities: [__dirname + '/entity/*.ts']
})
```

src/entity/UserChat.ts - модель таблиці даних про авторизованих користувачів

```
import { Entity, PrimaryGeneratedColumn, Column, BaseEntity } from "typeorm"


@Entity('userchat')
export class UserChat extends BaseEntity {

    @PrimaryGeneratedColumn()
    id: number;

    @Column()
    chatId: number;

    @Column()
    remoteName: string

    @Column()
    remoteId: string;

    static async isUserAuthenticated(chatId: number) {
        return Boolean(await this.findByChatId(chatId));
    }
```

```
    static findByChatId(chatId: number): Promise<UserChat | null> {
        return this.createQueryBuilder('userchat').where("userchat_chatId=:chatId", {
chatId }).getOne();
    }

}
```

src/entity/Log.ts – модель логів подій бота

```
import { Entity, PrimaryGeneratedColumn, Column, BaseEntity } from "typeorm"

export enum LogMessages {
    USER_ALREADY_EXISTS = 'Спроба повторної авторизації',
    USER_NOT_FOUND = 'Невідомий користувач',
    USER_WRONG_CONTACT = 'Спроба надати чужий контакт',
    USER_AUTHENTICATED = 'Користувач увійшов в систему',
    DOOR_OPENED = 'Двері відкрито'
}

@Entity('log')
export class Log extends BaseEntity {

    @PrimaryGeneratedColumn()
    id: number;

    @Column()
    message: string;

    @Column()
    description: string
```

```
    @Column()
    chat_id: number;

    @Column()
    created_at: Date;


    static async createLog(message: LogMessages, chatId: number, description?:
Object) {
        const instance = new this();

        instance.created_at = new Date();
        instance.message = message;
        instance.chat_id = chatId;
        instance.description = JSON.stringify(description) ?? '';

        return instance.save();
    }
}
```

src/bot.ts – логіка обробки повідомлень користувача

```
import { Telegraf } from 'telegraf'
import { ForceReply, InlineKeyboardMarkup, Message, ReplyKeyboardMarkup }
from 'telegraf/typings/core/types/typegram';
import { Log, LogMessages } from './entity/Log';
import { UserChat } from './entity/UserChat';
import { ButtonDescription, UProxService } from './uproxService';
```

```
function buildDoorButtonsReplyMarkup(desc: ButtonDescription[]):
InlineKeyboardMarkup {
    return {
        inline_keyboard: [desc]
    }
}

export function createSkudBot(token: string, service: UProxService) {
    const bot = new Telegraf(token);

    bot.on(['message', 'contact', 'callback_query'], async (ctx) => {

        const contact = (ctx.message as Message.ContactMessage)?.contact;
        const chatId = ctx?.message?.chat?.id ?? ctx.callbackQuery.message.chat.id;
        const isAuthenticated = await UserChat.isUserAuthenticated(chatId);

        if (contact && isAuthenticated) {
            ctx.reply('Ви вже авторизовані');
            Log.createLog(LogMessages.USER_ALREADY_EXISTS, chatId);
            return;
        }

        if (contact && !isAuthenticated) {

            if (!contact.user_id) {
                ctx.reply('Відправте власний контакт');
                Log.createLog(LogMessages.USER_WRONG_CONTACT, chatId, {
contact });
```

```
            return;
        }

        const user = await service.findUserByPhoneNumber(contact.phone_number);

        if (!user) {
            ctx.reply('Такий користувач не знайдений');
            Log.createLog(LogMessages.USER_NOT_FOUND, chatId, {
phone_number: contact.phone_number });
            return;
        }

        const userChat = new UserChat();
        userChat.chatId = ctx.message.chat.id;
        userChat.remoteName = user.remoteName;
        userChat.remoteId = user.remoteId;
        await userChat.save();

        const buttonDesc = await service.getAvailableButtons(userChat.remoteId);
        ctx.reply(`Ви авторизовані як ${userChat.remoteName}`, { reply_markup:
buildDoorButtonsReplyMarkup(buttonDesc) });
        Log.createLog(LogMessages.USER_AUTHENTICATED, chatId, { userChat
});
        return;
    }

    if (!isAuthenticated) {
        ctx.reply('Будь ласка, надайте номер телефону для авторизації', {
```

```
            reply_markup: {
                keyboard: [[{ text: 'Відправити контакти', request_contact: true }]],
                one_time_keyboard: true,
                resize_keyboard: true
            }
        });

        return;
    }

    const user = await UserChat.findByChatId(chatId);

    if (ctx?.callbackQuery?.data) {
        const [name, id] = (ctx.callbackQuery.data as string).split(';');

        const doorOpenResult = await service.openDoor(user.remoteId, id);

        if (!doorOpenResult) {
            ctx.reply(`Помилка відкриття дверей. Зверніться до адміністратора.`);
        }


        ctx.reply(`${name} відкрито`);
        Log.createLog(LogMessages.DOOR_OPENED, chatId, { doorName: name,
doorId: id, user });
    }

    const buttonDesc = await service.getAvailableButtons(user.remoteId);
```

```
        ctx.reply('Доступні двері', { reply_markup:
buildDoorButtonsReplyMarkup(buttonDesc) });
    });

    return bot;
}
```

src/uproxService.ts – логіка взаємодії з системою UProx

```
import { UserChat } from "./entity/UserChat";
import fetch from 'node-fetch';
import * as md5 from 'md5';

export type ButtonDescription = {
    text: string;
    callback_data: string;
}

export class UProxService {

    constructor(adminLogin: string, adminPass: string) { this.adminLogin =
adminLogin; this.adminPass = adminPass }

    adminLogin: string;
    adminPass: string;
    usersid: string = '0';
    usertoken: number = 0;

    async authenticate() {
        const hash = md5(md5(md5(this.adminPass).toUpperCase() +
```

```
"F593B01C562548C6B7A31B30884BDE53").toUpperCase()).toUpperCase();


        const body = {
            UserName: this.adminLogin,
            PasswordHash: hash
        }

        const response = await fetch("http://localhost:40001/json/Authenticate", {
method: "POST", body: JSON.stringify(body), headers: { 'Content-Type':
'application/json' } });

        const json = await response.json();

        this.usersid = json.UserSID;
        this.usertoken = json.UserToken;

        return;
    }

    async getUsers() {
        if (this.usertoken == 0) {
            await this.authenticate();
        }


        const requestBody = {
```

```
            "Language": '',
            "UserSID": this.usersid,
            "SubscriptionEnabled": true,
            "Limit": 2147483647,
            "StartToken": 0,
            "AdditionalFieldsRequired": true,
            "DepartmentUsed": false,
            "HideDismissed": true,
        }

        const response = await fetch("http://localhost:40001/json/EmployeeGetList",
            { method: "POST", body: JSON.stringify(requestBody), headers: { 'Content-
Type': 'application/json', 'Cookie': `UserSID=${this.usersid}` } });

        const json = await response.json();

        return json.Employee;
    }

    async getCards() {
        if (this.usertoken == 0) {
            await this.authenticate();
        }


        const requestBody =
        {
            "Language": "",
```

```
            "UserSID": this.usersid,
            "SubscriptionEnabled": false,
            "Limit": 2147483647,
            "StartToken": 0,
            "UserTokenUsed": false,
        }

        const response = await fetch("http://localhost:40001/json/CardGetList",
            { method: "POST", body: JSON.stringify(requestBody), headers: { 'Content-
Type': 'application/json', 'Cookie': `UserSID=${this.usersid}` } });

        const json = await response.json();

        return json.Card;
    }

    async getDoors() {
        if (this.usertoken == 0) {
            await this.authenticate();
        }


        const requestBody =
        {
            "Language": "",
            "UserSID": this.usersid,
            "SubscriptionEnabled": false,
            "Limit": 2147483647,
```

```
            "StartToken": 0,
        }

        const response = await fetch("http://localhost:40001/json/DoorGetList",
            { method: "POST", body: JSON.stringify(requestBody), headers: { 'Content-
Type': 'application/json', 'Cookie': `UserSID=${this.usersid}` } });

        const json = await response.json();

        return json.Door;
    }

    async getUserCardByUserToken(userId: string) {
        const cards = await this.getCards();

        return cards.find(card => card.UserToken = userId);
    }

    async doorAccessOut(userId: string, doorId: string) {
        if (this.usertoken == 0) {
            await this.authenticate();
        }

        const card = await this.getUserCardByUserToken(userId);

        if (!card) return false;

        const requestBody =
```

```
        {
            "Language": "",
            "UserSID": this.usersid,
            "CardCode": card.Code,
            "Token": doorId,
        }

        const response = await fetch("http://localhost:40001/json/DoorAccessOut",
            { method: "POST", body: JSON.stringify(requestBody), headers: { 'Content-
Type': 'application/json', 'Cookie': `UserSID=${this.usersid}` } });

        return response.status === 200;
    }


    async findUserByPhoneNumber(phone: string): Promise<Pick<UserChat,
"remoteName" | "remoteId"> | null> {
        const users = await this.getUsers();

        if (!users) return null;

        const selectedUser = users.map(user => ({
            token: user.Token,
            name: user.Name,
            phone: user.AdditionalFields.find(field => field.Name = "Номер
телефону")?.Value
        })).find(user => user.phone == phone);
```

```
        if (selectedUser) {
            return {
                remoteId: selectedUser.token,
                remoteName: selectedUser.name
            }
        }

        return null;
    }

    async getAvailableButtons(userId: string): Promise<ButtonDescription[]> {

        const doors = await this.getDoors();

        const buttons: ButtonDescription[] = doors.map(door => ({
            text: door.Name,
            callback_data: `${door.name};${door.Token}`
        }))

        return buttons;
    }

    async openDoor(userId: string, doorId: string): Promise<boolean> {
        return this.doorAccessOut(userId, doorId);
    }
}
```