Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

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

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Другий (магістерський)

(рівень вищої освіти)

Розроблення мобільного застосунку для моніторингу автоматизованих систем
(тема)

Виконав:

студент 2 курсу, групи КІТПВм 21-1

Руденко І. В.
(прізвище, ініціали)

Спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)

Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Ком'ютерно-інтегровані технологічні процеси та виробництва
(повна назва освітньої програми)

Керівник доц. Бронніков А. І.
(посада, прізвище, ініціали)

Допускається до захисту

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

2022 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАМ
Рівень вищої освіти другий (магістерський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми Освітньо-професійна
Освітня програма

(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАМ
_____
(підпис)
«__»_____20_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Руденку Ігорю Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення мобільного застосунку для моніторингу автоматизованих систем
Затверджена наказом по університету від 07.11.2022 № 1464 Ст
2. Термін подання студентом роботи до екзаменаційної комісії

3. Вихідні дані до роботи
3.1 Scada-системи;
3.2 Моніторинг автоматизованих систем;
3.3 Технології віддленого доступу;
3.4 Децентралізовані та централізовані системи;
3.5 Архітектура клієнт-серверного застосунку.

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Аналіз сучасного моніторингу автоматизованих систем;
4.2 Методи реалізації програмної системи;
4.3 Реалізація алгоритму та розробка архітектурної системи;
4.4 Процес роботи програми мобільного застосунку.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Демонстраційні матеріали, представлені у форматі презентації PowerPoint (*.ppt), 12 стор.формату А4.

6. Консультанти розділів роботи

Найменування Розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		Підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області та технічного завдання	7.11 – 8.11.22	Виконано
2	Вибір архітектури та модулів моніторингу	9.11 – 15.11.22	Виконано
3	Аналіз Scada-систем	15.11 – 9.11.22	Виконано
4	Створення застосунку	19.11 – 14.12.22	Виконано
5	Подання роботи на перевірку Інтернет сервісом Unicheck	15.12 – 17.12.22	Виконано
6	Оформлення пояснювальної записки	17.12 – 18.12.22	Виконано
7	Подання роботи на рецензію	18.12 – 19.12.22	Виконано
8	Подання роботи на підпис зав.кафедри	19.12 – 20.12.22	Виконано
9	Подання атестаційної роботи в ЕК	21.12 – 22.12.22	Виконано

Дата видачі завдання 01.09.2022

Студент _____ Руденко І.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Бронніков А.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 119 с., 49 рис., 2 дод., 31 джерело.

АВТОМАТИЗОВАНІ СИСТЕМИ , МОНІТОРИНГ, ІНТЕРНЕТ, СУЧАСНІ ЦЕНТРИ ОБРОБКИ ДАНИХ, МОБІЛЬНИЙ ЗАСТОСУНОК, SCADA-СИСТЕМИ.

Об’єкт дослідження – моніторинг автоматизованих систем.

Предмет дослідження – моніторинг автоматизованих систем через мобільний застосунок, який написаний на мові програмування Kotlin в інтегрованому середовищі розробки Android Studio.

Мета кваліфікаційної роботи – підвищення ефективності ведення моніторингу автоматизованих систем за рахунок створення сучасних інструментів обліку та аналізу виробничих процесів та їх реалізації у мобільному застосунку.

Методи дослідження – методи розробки мобільних застосунків під операційну систему Android, методи роботи з базою даних SQLite, методи передачі даних по HTTP протоколах при синхронізації локальної і серверної баз даних.

У кваліфікаційній роботі досліджено найпопулярніші методи створення системи моніторингу автоматизованих систем через мобільний застосунок. Також було проведено аналіз економічної доречності установки сучасної системи.

Система моніторингу дозволяє підприємству заздалегідь покращити процеси розробки продукції, вносити зміни до її виготовлення з метою покращення якості продукції та зниження витрат, що потребує ефективного аналізу даних або DM-алгоритмів. Ця система допомагає людям економити час та кошти.

Основною вимогою, що ставиться до засобів моніторингу, є мобільність, яка повинна забезпечувати можливість доступу до необхідної інформації у будь-який час з будь-якого місця за допомогою різних засобів.

Результати кваліфікаційної роботи апробовані в 2 фахових статтях.

ABSTRACT

Explanatory note: 119 p., 49 figures, 2 appendix, 31 sources.

AUTOMATED SYSTEMS, MONITORING, INTERNET, MODERN DATA PROCESSING CENTERS, MOBILE APPLICATION, SCADA SYSTEMS.

The object of research is the monitoring of automated systems.

The subject of the research is the monitoring of automated systems through a mobile application written in the Kotlin programming language in the integrated Android Studio development environment.

The purpose of the qualification work is to increase the efficiency of monitoring of automated systems by creating modern accounting and analysis tools for production processes and their implementation in a mobile application.

The research methods are methods of development of mobile applications for the Android operating system, methods of workwith the SQLite database, methods of data transfer using HTTP protocols at synchronization of local and server databases.

The most popular methods of creating a system for monitoring automated systems through a mobile application were investigated in the qualification work. An analysis of the economic appropriateness of installing a modern system was also conducted.

The monitoring system allows the enterprise to improve product development processes in advance, to make changes to its production in order to improve product quality and reduce costs, which requires effective data analysis or DM-algorithms. This system helps people save time and money.

The main requirement for monitoring tools is mobility, which should ensure access to the necessary information at any time from any place using various means.

The results of the qualification work are tested in 2 professional articles.

ЗМІСТ

ВСТУП

Останнє десятиріччя відзначається прискореними темпами проникнення інформаційних технологій в усі сфери життя людини. Зокрема, комп'ютерні системи повністю змінили характер взаємодій між людьми, а також між людиною і машиною.

Одним із напрямів розвитку сучасного підприємства є моніторинг автоматизованих систем в реальному часі. Основною вимогою, що ставиться до засобів моніторингу, є мобільність, яка повинна забезпечувати можливість доступу до необхідної інформації у будь-який час з будь-якого місця за допомогою різних засобів. Для такого моніторингу використовують мініатюрні мобільні пристрої, мережі давачів, безпровідні мережі, супутникову навігацію, хмарні технології та сховища даних. Засоби моніторингу повинні забезпечувати спостереження і прогнозування всіх стадій виробничого процесу з метою підвищення ефективності внесенням необхідних поправок у реальному часі.

Основні переваги при впровадженні моніторингу автоматизованих систем:

− збільшується продуктивність виробничого обладнання;

− підвищується надійність роботи обладнання, знижується ризик важких аварій;

− забезпечується автоматизоване управління технологічними процесами;

− персонал інформується про хід технологічного процесу та стає захищеним при загрозі аварії.

Таким чином, метою кваліфікаційної роботи є створення системи моніторингу автоматизованих систем через мобільний застосунок.

Об'єкт дослідження – моніторинг автоматизованих систем.

Предмет дослідження – моніторинг автоматизованих систем через мобільний застосунок, який написаний на мові програмування Kotlin в інтегрованому середовищі розробки Android Studio.

Методи дослідження – методи розробки мобільних додатків під операційну систему Android, методи роботи з базою даних SQLite, методи передачі даних по HTTP протоколах при синхронізації локальної і серверної баз даних.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих систем моніторингу автоматизованих систем;

– проаналізувати специфіку розробки мобільного застосунку під різні мобільні платформи;

– визначити перелік завдань для системи моніторингу автоматизованих систем на підприємстві;

– сформулювати вимоги та вибрати принципи розроблення системи моніторингу автоматизованих систем на підприємстві;

– розробити блок-схему алгоритму роботи застосунку;

– проаналізувати сучасну розробку мобільних застосунків, вибрати актуальні технології для розробки;

– розробити програмне забезпечення мобільного застосунку;

– розробити базу даних та алгоритм функціонування мобільного додатку;

– оформити пояснювальну записку згідно [1] та [2].

У дослідженні було проаналізовано уже створені мобільні застосунки. Усі розглянуті програми мають широкий функціонал і досить вагомі доповнення, але не всі продукти мають безкоштовний контент. У деяких немає жодних функцій або навантажені безліччю зображень та тексту. Програма Мобільний застосунок на базі ОС Android для моніторингу автоматизованих систем вирішить проблему з набором необхідних функцій та інформативністю. Головний плюс цієї розробки – абсолютно безкоштовне придбання та користування.

Практичне значення одержаних результатів полягає в тому, що пропозиції, висновки та результати наукового дослідження в ході апробації підтвердили свою економічну ефективність та є перспективними для застосування на підприємствах України.

Результати дослідження опубліковані в збірнику наукових статей «Automation of Electronic Devices» ADED-2022 [4].

1 АНАЛІЗ СУЧАСНОГО СТАНУ МОНІТОРИНГУ АВТОМАТИЗОВАНИХ СИСТЕМ

1.1 Використання автоматизованих систем у різних сферах діяльності

1.1.1 Поняття автоматизованої системи

Автоматизована система (АС) – сукупність керованого об’єкта й автоматичних керуючих пристроїв, у якій частину функцій керування виконує людина. АС являє собою організаційно-технічну систему, що забезпечує вироблення рішень на основі автоматизації інформаційних процесів у різних сферах діяльності (управління, проектування, виробництво тощо) або їх поєднаннях [6].

Створенню АС у нашій країні приділяється багато уваги. За масштабами, темпами зростання, витратами матеріальних, фінансових і трудових ресурсів, а також за ступенем впливу на процеси управління проблема створення АС перетворилася на велике народногосподарське завдання.

Інформаційні системи можуть значно різнитися за типами об’єктів управління, характером та обсягом розв’язуваних завдань і рядом інших ознак. Загальноприйнятої класифікації АС на сьогодні не існує, тому їх можна класифікувати за різними ознаками:

– за рівнем або сферою діяльності – державні, територіальні (регіональні), галузеві;

– за рівнем автоматизації процесів управління – інформаційно-пошукові, інформаційно-довідкові, інформаційно-керівні, системи підтримки прийняття рішень, інтелектуальні АС;

– за ступенем централізації обробки інформації – централізовані АС, децентралізовані АС, інформаційні системи колективного використання;

– за ступенем інтеграції функцій – багаторівневі АС з інтеграцією за рівнями управління (підприємство – об’єднання, об’єднання – галузь тощо), багаторівневі АС з інтеграцією за рівнями планування тощо [7].

Державні АС призначені для вирішення найважливіших народногосподарських проблем країни. На базі використання обчислювальних комплексів та економіко-математичних методів складають перспективні та поточні плани розвитку країни, ведуть облік результатів та регулюють діяльність окремих ланцюгів народного господарства, розробляють державний бюджет та контролюють його виконання й т. ін.

Територіальні (регіональні) АС призначені для управління адміністративно-територіальним регіоном. Сюди належать АС області, міста, району. Ці системи виконують роботи з обробки інформації, яка необхідна для реалізації функцій управління регіоном, формування звітності й видачі оперативних даних місцевим і керівним державним та господарським органам.

Галузеві інформаційні системи управління призначені для управління підвідомчими підприємствами та організаціями. Галузеві АС діють у промисловості та в сільському господарстві, будівництві, на транспорті тощо. У них розв'язуються задачі інформаційного обслуговування апарату управління галузевих міністерств і їх підрозділів. Галузеві АС відрізняються сферами застосування – промислова, наукова. Популярними АС на підприємствах є система контролю та обліку енергоресурсів, які забезпечують оперативний контроль поточного навантаження, комерційний облік і оперативний контроль споживання або надання енергоносіїв, підтримку прийняття рішень при плануванні енергоспоживання та вироблення енергозберігаючої політики, приклад зовнішнього вигляду такої системи представлено на рис. 1.1.

Загалом АС – це система, яка складається з персоналу й комплексу засобів автоматизації його діяльності та реалізує інформаційну технологію виконання встановлених функцій.

Залежно від виду діяльності розрізняють такі різновиди АС: автоматизовані системи управління (АСУ), системи автоматизованого проектування (САПР), автоматизовані системи наукових досліджень (АСНД) [7].

Рисунок 1.1 – Автоматизована система контролю та обліку енергоресурсів

Людино-машинна АСУ вже на стадії проектування потребує як удосконалення організації основної діяльності економічного об’єкта (виробничого, господарського), так і поліпшення організації управлінських процедур.

Масове проектування АСУ, яке почалося тридцять років тому, вимагало розробки єдиних теоретичних положень, методичних підходів до їх створення і функціонування, без чого неможлива взаємодія різноманітних економічних об’єктів, їх нормальне функціонування в складеному багаторівневому народногосподарському комплексі.

1.1.2 Принципи побудови АСУ

Початково сформульовані академіком В. М. Глушковим науково-методичні положення та рекомендації з проектування автоматизованих систем управління тепер склались як принципи побудови АСУ, закріплені державним стандартом. До них належать принципи системності, розвитку, сумісності, стандартизації та уніфікації, ефективності.

Принцип системності є основоположним при створенні, функціонуванні і розвитку АСУ. Він дає змогу розглядати досліджуваний об'єкт як одне ціле; виявляти на цій підставі різноманітні типи зв'язків між структурними елементами, які забезпечують цілісність системи; установлювати напрямок виробничо-господарської діяльності системи і реалізовані нею конкретні функції. Системний підхід передбачає проведення двохаспектного аналізу, відомого під назвою макро- і мікро-підходів [8].

При мікроаналізі система або її елемент розглядається як частина системи вищого порядку. Особлива увага приділяється інформаційним зв'язкам: встановлюється їх кількість; виокремлюються та аналізуються ті зв'язки, які зумовлені метою вивчення системи, а далі відбираються найперспективніші, які реалізують задану цільову функцію. При мікроаналізі вивчається структура об'єкта, аналізуються її складові елементи з погляду їх функціональних характеристик, які виявляються через зв'язки з іншими елементами та зовнішнім середовищем. У процесі проектування АСУ системний підхід дає змогу використовувати математичний опис функціонування, дослідження різноманітних властивостей окремих елементів і системи в цілому, моделювати процеси, що вивчаються, для аналізу роботи створюваних систем.

У системах управління особливо великими, протяжними об'єктами замість децентралізованих передуючих сигналів застосовують централізовані, здійснювані за допомогою відповідних телемеханічних пристроїв (рис. 1.2).

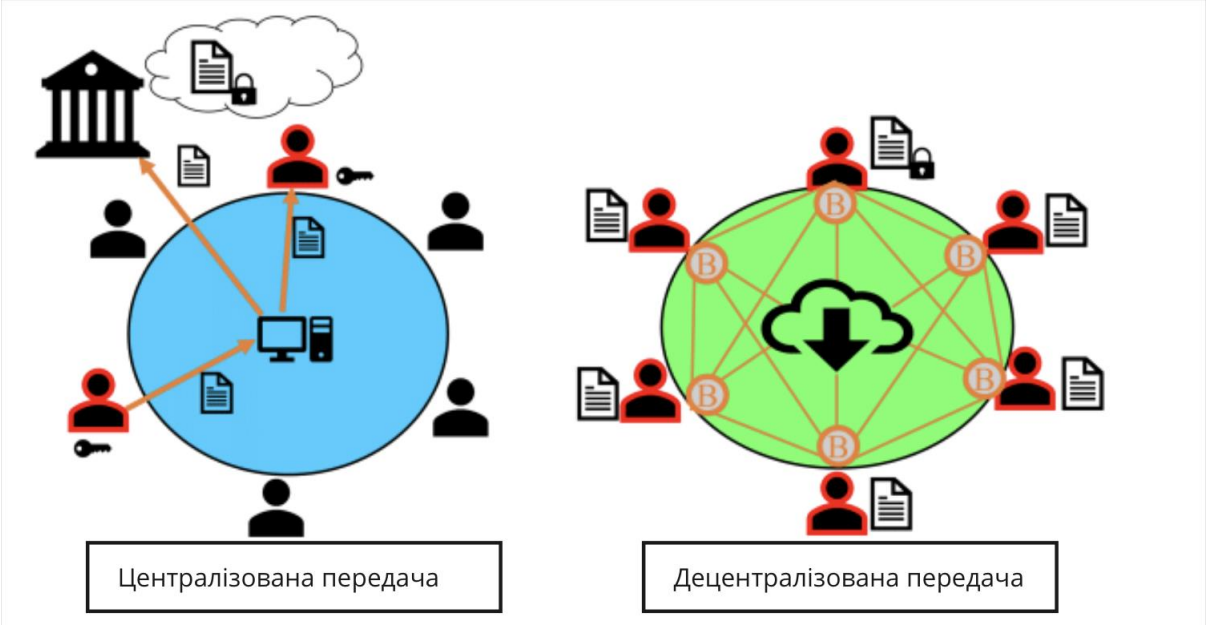

Рисунок 1.2 – Децентралізована та централізована передача сигналів

Для автоматизованих систем управління характерна багаторівнева ієрархія з вертикально субпідрядними елементами (підсистемами). Ієрархічні структури в системах управління набули значного поширення завдяки своїм перевагам. Так, ієрархічна структура створює відносну волю дій над окремими елементами для кожного рівня системи і можливість різних поєднань (комбінацій) локальних критеріїв оптимальності функціонування системи в цілому; забезпечує відносну гнучкість системи управління і можливість пристосування до умов, які постійно змінюються; підвищує надійність за рахунок можливості введення елементної надмірності, реалізації напрямків потоків інформації.

При розробці, впровадженні та експлуатації на одному підприємстві розглядаються як взаємозалежні, але окремі системи, між якими існують відносини ієрархічної співпідпорядкованості як молодшого до старшого, а не як частини до цілого. Аналогічно трактуються співвідношення між автоматизованою системою управління підприємства (АСУП) і галузевими АСУ (ОАСУ), АСУ агрегатами і АСУ виробництвами: всі вони не вкладені одна в іншу, а утворюють багаторівневу ієрархію автоматизованих систем управління промисловими об’єктами, яка схематично показана на рис. 1.3.

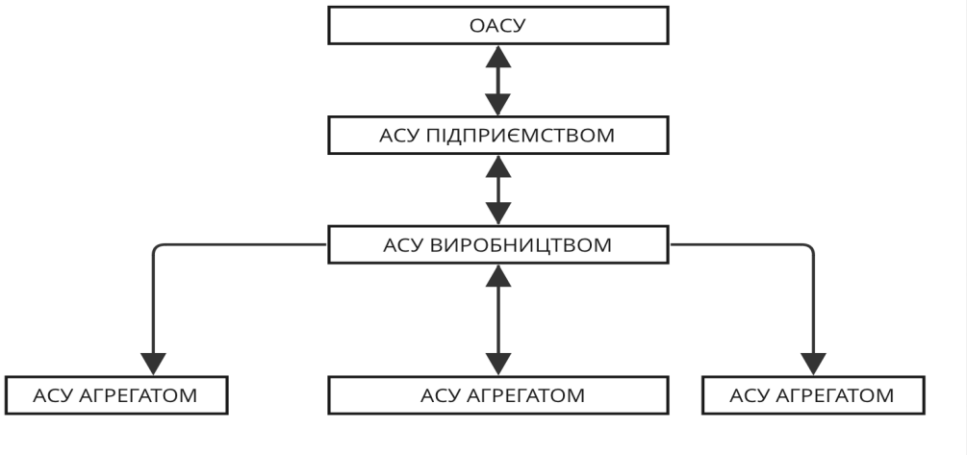

Рисунок 1.3 – Ієрархія автоматизованих систем управління

Принцип розвитку полягає в тому, що АСУ створюється з урахуванням можливості постійного поповнення та оновлення функцій системи й типів її забезпечення. Передбачається, що автоматизована система має нарощувати свої обчислювальні можливості, оснащуватись новими технічними і програмними засобами, бути здатною постійно розширювати й поновлювати склад задач та інформаційний фонд, який створюється у вигляді баз даних.

Принцип сумісності полягає в забезпеченні здатності взаємодії АСУ різних видів, рівнів у процесі їх спільного функціонування. Реалізація цього принципу дає змогу забезпечити якісне функціонування економічних об'єктів, підвищити ефективність управління народним господарством та його окремими ланками.

Принцип стандартизації та уніфікації полягає в необхідності застосування типових уніфікованих і стандартизованих елементів функціонування АСУ. Упровадження в практику створення і розвитку АСУ цього принципу дає змогу скоротити часові, трудові і вартісні витрати на створення АСУ за максимально можливого використання нагромадженого досвіду у формуванні проектних рішень і впровадженні автоматизації проектних робіт.

Принцип ефективності полягає в досягненні раціонального співвідношення між витратами на створення АСУ й цільовим ефектом, одержаним при її функціонуванні [8].

Як правило, крім основних принципів для ефективного здійснення управління вирізняють також низку часткових принципів, які деталізують загальні. Додержання кожного з часткових принципів дає змогу отримати певний економічний ефект:

– принцип декомпозиції – використовується при вивченні особливостей, властивостей елементів і системи в цілому. Він ґрунтується на розбитті системи на частини, виокремленні деяких комплексів робіт, створенні умов для ефективного аналізу системи та її проектування;

– принцип першого керівника передбачає закріплення відповідальності під час створення системи за замовником – керівником підприємства, установи, галузі, тобто майбутнім користувачем, який відповідає за введення у дію та функціонування АСУ;

– принцип нових задач – пошук постійного розширення можливостей системи, удосконалення процесів управління, одержання додаткових результатних показників з метою оптимізації управлінських рішень. Це може супроводжуватись постановкою й реалізацією на ЕОМ нових задач управління;

– принцип автоматизації інформаційних потоків і документообігу передбачає комплексне використання технічних засобів на всіх стадіях проходження інформації від моменту її реєстрації до одержання результативних показників і формування управлінських рішень;

– принцип автоматизації проектування має на меті підвищити ефективність самого процесу проектування і створення АСУ на всіх рівнях народного господарства, при цьому забезпечується скорочення часових, трудових і вартісних витрат за рахунок введення індустріальних методів [8].

АСУ ТП – це комплекс технічних та програмних засобів, призначений для управління технологічними процесами на виробництвах.

АСУ ТП в загальному випадку має трирівневу архітектуру:

– нижній (польовий) рівень;

– контролери (абревіатури: ПЛК, PLC);

– комп’ютери, сервери для задач диспетчеризації, характерне використання SCADA.

На нижньому рівні працюємо з датчиками (перетворювачами) та виконавчими механізмами будь-яких виробників, що підключаються по будь-яким аналоговим, дискретним, імпульсним, цифровим інтерфейсам зв’язку.

На середньому рівні ієрархії АСУ ТП функціонують контролери, що виконують функції вимірювання, керування, регулювання, сигналізації, діагностування, захисту, блокування. Технічні засоби середнього рівня, як правило, компонуються у вигляді електротехнічних шаф, щитів, пультів, панелей.

На верхньому рівні ієрархії АСУ ТП функціонують сервери та комп’ютери промислового або не промислового (широкого вжитку) виконання.

Зв’язок між технічними засобами верхнього та середнього рівня забезпечується за рахунок використання мережевого інтерфейсу Ethernet.

Сервери та комп’ютери є платформою для функціонування Систем управління базами даних (СУБД) та програми диспетчеризації. Програми диспетчеризації реалізуємо на базі SCADA.

Організаційна схема автоматизації процесів управління виробництвом представлена на рис. 1.4.

Рисунок 1.4 – Організаційна схема автоматизації процесів управління виробництвом

Сучасний рівень розробки й впровадження систем дає змогу широко використовувати типізацію проектних рішень, уніфікацію методів і засобів при підготовці проектних матеріалів, стандартизації підходів під час проектування окремих елементів систем і підсистем, методи автоматизації ведення проектних робіт з використанням персональних ЕОМ і організованих на їх базі автоматизованих робочих місць проектувальника АСУ.

1.1.3 Класифікація АСУ

Складниками автоматизованої системи є групи персоналу, необхідні комплекси автоматизації та правила (алгоритми) дій.

Залежно від визначеної мети автоматизовані системи поділяють на кілька видів:

– системи автоматизованого проектування (САПР);

– автоматизовані системи наукових досліджень (АСНД);

– автоматизовані системи керування (АСК).

У свою чергу АСК поділяють на два підкласи:

– системи керування технологічними процесами;

– системи керування підприємством (галуззю) [5].

АСУ адаптовані під усі сфери діяльності, незважаючи на форму власності, обсяг виробництва, кількість працівників. Узагальнюючу класифікацію систем АСУ наведено на рис. 1.5.

У процесі функціонування АС є поєднанням:

а) комплексу технічних засобів автоматизації (ТЗА) – сукупності взаємоузгоджених компонентів і комплексів програмного, технічного та інформаційного забезпечення, що розробляються, виготовляються й постачаються як продукція виробничо-технічного призначення:

1) програмне забезпечення автоматизованої системи – сукупність програм на носіях інформації з програмною документацією;

2) технічне забезпечення автоматизованої системи – сукупність засобів реалізації керуючих впливів, засобів отримання, введення, підготовки,

перетворення, обробки, зберігання, реєстрації, виведення, відображення, використання та передачі даних з конструкторською та експлуатаційною документацією;

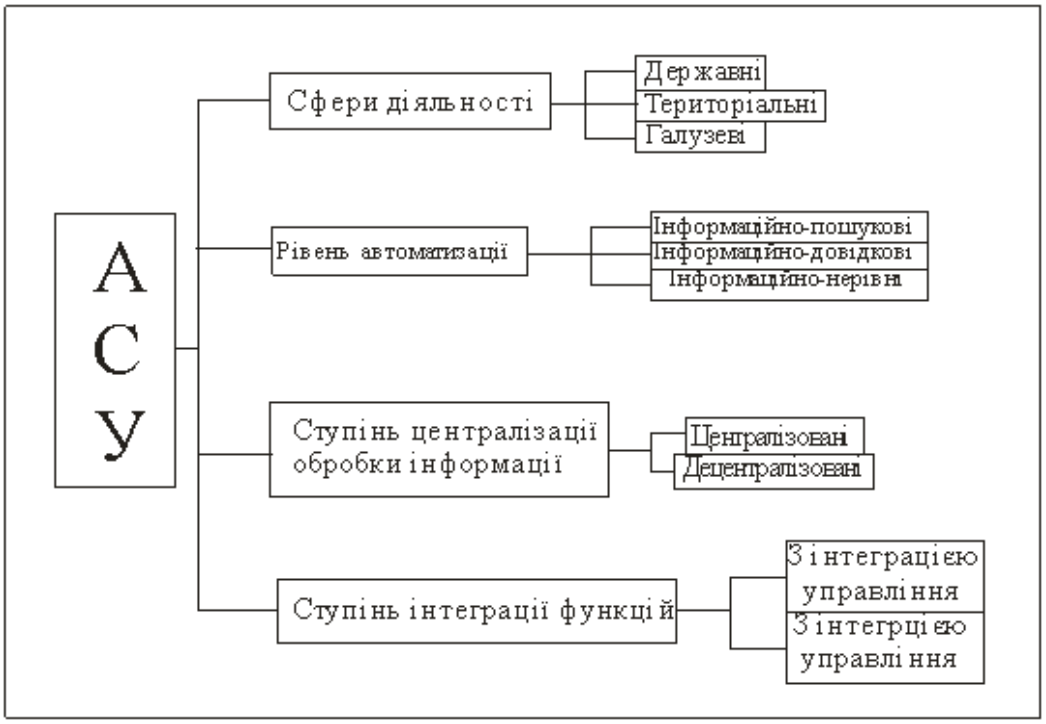

Рисунок 1.5 – Класифікація АСУ

3) інформаційне забезпечення автоматизованої системи – сукупність системно-орієнтованих даних, що описують прийнятий у системі словник базових описів (класифікатори, типові моделі, елементи автоматизації, формати документів), та актуалізованих даних про стан інформаційної моделі об'єкта автоматизації (об'єкта управління, об'єкта проектування) на всіх етапах його життєвого циклу.

б) організаційно-методичного забезпечення автоматизованої системи – сукупності документів, що визначають: організаційну структуру об'єкта та системи автоматизації, необхідних для виконання конкретних функцій, що автоматизуються; діяльність в умовах функціонування системи, а також форми представлення результатів діяльності;

в) фахівців, які використовують вище перераховане в процесі своєї професійної діяльності [9].

Внутрішню побудову систем характеризують за допомогою структур, що описують стійкі зв'язки між їх елементами. При описі АС використовують такі види структур, що відрізняються типами елементів і зв'язків між ними:

- функціональні (елементи – функції, завдання, процедури; зв'язки – інформаційні);
- технічні (елементи – пристрої, компоненти та комплекси; зв'язки – лінії і канали зв'язку);
- організаційні (елементи – колективи людей і окремі виконавці; зв'язки – інформаційні, підпорядкування та взаємодії);
- документальні (елементи – неподільні складові частини та документи АС; зв'язки – взаємодії, вхідність та підпорядкування);
- алгоритмічні (елементи – алгоритми; зв'язки – інформаційні);
- програмні (елементи – програмні модулі та вироби; зв'язки – керуючі);
- інформаційні (елементи – форми існування і подання інформації в системі; зв'язки – операції перетворення інформації в системі) [9].

1.1.4 Напрямки автоматизації

Автоматизація системи відбувається за трьома напрямами:

– розроблення методів вивчення закономірностей об’єктів керування й розроблення ефективних алгоритмів керування;

– визначення економічної доцільності автоматизації системи в цілому або її частин;

– створення необхідних технічних засобів [10].

Спільною властивістю автоматизованих систем будь-якого призначення є необхідність обробки великих масивів даних: введення, виведення, запам’ятовування, аналізування тощо. Ефективність дії автоматизованої системи в цілому часто залежить від ефективності виконання цих функцій. Це призвело до створення спеціалізованих автоматизованих систем – автоматизованих інформаційних систем (АІС), мета яких полягає в оптимізації оброблення цифрових даних незалежно від джерела їх походження. Визначальною рисою АІС

є їхня гнучкість, тобто можливість обробляти різні за походженням дані в стандартизованому технологічному засобі, змінюючи тільки алгоритм оброблення даних. Автоматизація системи будь-якого призначення є способом підвищення її продуктивності та якості її продукту.

1.2 Аналіз проблеми моніторингу автоматизованих систем

1.2.1 Об’єкт і суб’єкт моніторингу

Моніторинг – спеціально організоване, систематичне спостереження за станом об'єктів, явищ, процесів з метою їх оцінки, моніторингу, прогнозу й управління [11]. Моніторинг є не тільки інформаційною базою для виконання функцій управління, а й сам реалізується за допомогою загальних функцій управління. З точки зору технічних наук під системою моніторингу розуміється взаємопов'язана сукупність елементів, що забезпечує здійснення моніторингу. Можна виділити такі складові структури системи моніторингу:

– суб'єкти моніторингу;
– об'єкт моніторингу;
– комплекс показників та індикаторів моніторингу;
– інструментарій моніторингу;
– процедура моніторингу.

Суб'єктами моніторингу є носії функцій систематичного спостереження за станом об'єкта моніторингу, тобто безпосередньо функцій моніторингу. Умовно їх можна розділити на дві групи:

– ті, що виробляють збір інформації;
– ті, що виконують обробку інформації [11].

Об'єктом моніторингу в загальному випадку є деякий об'єкт або процес реального світу. Види об'єктів моніторингу різноманітні, їх масштабність, динамічність, складність істотно відрізняється, що частково й обумовлює різноманітність видів і систем моніторингу. Слід розмежувати об'єкт адміністративного моніторингу та об'єкт управління АСУ. Об'єкт моніторингу

аналізує менше інформації, ніж об'єкт управління інтегрованої АСУ, функціональні підсистеми якої можуть охоплювати всі сфери основної та допоміжної діяльності ОТС (у тому числі і організаційне управління) [12]. Об'єктами моніторингу в загальному випадку можуть бути як складні системні об'єкти (наприклад, галузь промисловості, виробниче об'єднання, корпорація, холдинг і їх діяльність, здоров'я персоналу, екологічний стан території підприємства та ін.), так і досить локальними (наприклад, процес впровадження у виробництво нового виробу, впровадження енергозберігаючої технології на підприємстві та ін.) [12]. З точки зору адміністративного моніторингу на підприємстві об'єктом моніторингу можуть бути процеси (основні, допоміжні, що забезпечують) або структурні елементи підприємства (засоби виробництва, об'єкти інфраструктури, елементи організаційної структури та ін.). Однак, незважаючи на різноманітність об'єктів моніторингу, можна виділити загальні властивості, які об'єднують всі ці різнорідні об'єкти, що належать навіть до різних сфер діяльності:

– об'єкти моніторингу динамічні, знаходяться в постійному розвитку, вони схильні до впливу зовнішніх впливів;

– реалізація моніторингу передбачає організацію стеження за об'єктом;

– організація стеження передбачає відбір обґрунтованих показників та індикаторів, що характеризують об'єкт моніторингу;

– стеження здійснюється шляхом безпосереднього або опосередкованого вимірювання параметрів об'єкта [14].

1.2.2 Комплекс показників та індикаторів моніторингу

Здійснення моніторингу неможливо без чіткого визначення показників та індикаторів досліджуваного об'єкта моніторингу. Поняття показника використовується досить широко в різних областях. Зокрема, показник є методологічним інструментом, що забезпечує можливість перевірки теоретичних положень за допомогою емпіричних даних [15]. У рамках цієї тематики під показником розуміється узагальнена характеристика властивості об'єкта моніторингу.

Зазвичай виділяють наступні види показників:

– якісні показники, що фіксують наявність або відсутність певної властивості;

– кількісні показники, що фіксують міру виразності, розвитку певного властивості.

Індикатор – доступна спостереженню й виміру характеристика досліджуваного об'єкта. Індикаторами заміщують, за допомогою індикаторів виявляють і представляють інші характеристики (показники) досліджуваного об'єкта, зазвичай недоступні спостереженню або незручні (непридатні) для інтерпретації. Таким чином, значення індикаторів є результатом вивчення, узагальнення та порівняння значень деяких показників [16]. Комплекс показників та індикаторів забезпечують уявлення про стан системи, про якісні та кількісні зміни в ній, необхідна й достатня для реалізації функцій управління.

1.2.3 Процедура моніторингу

Процедура моніторингу визначається як сукупність операцій, що проводяться в ході моніторингу. Наприклад, підготовка моніторингу, збір, обробка, аналіз і представлення інформації, забезпечення моніторингових процедур. Основна сфера практичного застосування моніторингу – це управління, а, точніше, інформаційне обслуговування управління в різних сферах діяльності [16]. Залежно від сфери застосування результатів моніторингу можна виділити радіолокаційний, авіаційний, космічний, супутниковий, інструментальний, педагогічний, психологічний, соціологічний, медичний, статистичний, адміністративний та інші види моніторингу.

1.2.4 Класифікація на підставі способу збору інформації

На підставі способу збору інформації можна виділити три групи видів моніторингу. До першої групи входять ті види моніторингу, у процесі здійснення яких можливе безпосереднє опис об'єкта моніторингу без будь-яких вимірювань, з використанням технології структуризації результатів, побудови схеми і технології

збору інформації (наприклад, моніторинг засобів масової інформації, поточного законодавства, виборів). Другу групу складають види моніторингу, в процесі яких здійснюється безпосереднє фізичне вимірювання параметрів об'єкта (наприклад, моніторинг шуму, рівня моря, корозії металів, комп'ютерних мереж). Третя група включає види моніторингу, в ході яких вимірювання параметрів об'єкта проводиться опосередковано з використанням системи добре розроблених і загальноприйнятих критеріїв або індикаторів (наприклад, моніторинг якості повітря, води, серцевої діяльності, грунтово-хімічний моніторинг). Четверту групу складають ті види моніторингу, при проведенні яких немає можливості використовувати існуючі системи критеріїв та індикаторів. У цьому випадку необхідна опосередкована зміна і залучення технологій наукового дослідження, що забезпечують перетворення результатів безпосередніх вимірювань у відповідний вид (наприклад, моніторинг санітарно-гігієнічний, соціально-політичний, соціально-економічний) [17].

Відповідно до характеру обробки інформації виділяють наступні типи інформаційних систем:

– інформаційні системи управління;

– системи аналітичної обробки даних;

– системи інтелектуального аналізу даних (data mining);

– інструменти кінцевого користувача для виконання запитів і побудови звітів (query and reporting tools) [18].

Широке поширення комп'ютерних та мережевих технологій, впровадження нових систем збору даних дозволяє застосовувати ці системи для моніторингу процесів та операцій, формувати електронні бази даних. Такі відомості доступні для аналізу з метою отримання знань: тенденції, моделі, причинно-наслідкові зв'язки між якістю кінцевої продукції та організацією виробництва, сформулювати план дій щодо вирішення проблеми та покращення якості.

Моніторинг автоматизованих систем повинен розв’язувати такі завдання:

– збір технологічних даних у реальному часі;

– збереження та попереднє опрацювання даних з використанням хмарних технологій;

– аналітичне та інтелектуальне опрацювання даних;

– візуалізація накопичених даних про технологічні процеси і подання результатів опрацювання даних у вигляді графіків і діаграм;

– формування сигналів управління для виконавчих механізмів;

– прийняття управлінських рішень для управління автоматизованими системами;

– формування звітів про стан технологічного процесу.

1.2.5 Сучасний стан систем керування інженерними мережами

Сучасні центри обробки даних (ЦОД) – це економічно виправдані рішення, що консолідують IT-ресурси організації та здатні значно скоротити загальні витрати на IT за рахунок впровадження централізованої моделі обчислень. Однак постійне ускладнення IT-інфраструктури, збільшення енергоспоживання та тепловиділення в ЦОД накладають на роботу обслуговуючих інженерних підсистем низку додаткових вимог: дуже висока надійність, керованість, безпека, адаптивність до змін довкілля. Приклад сучасного центру обробки даних представлено на рис. 1.6.

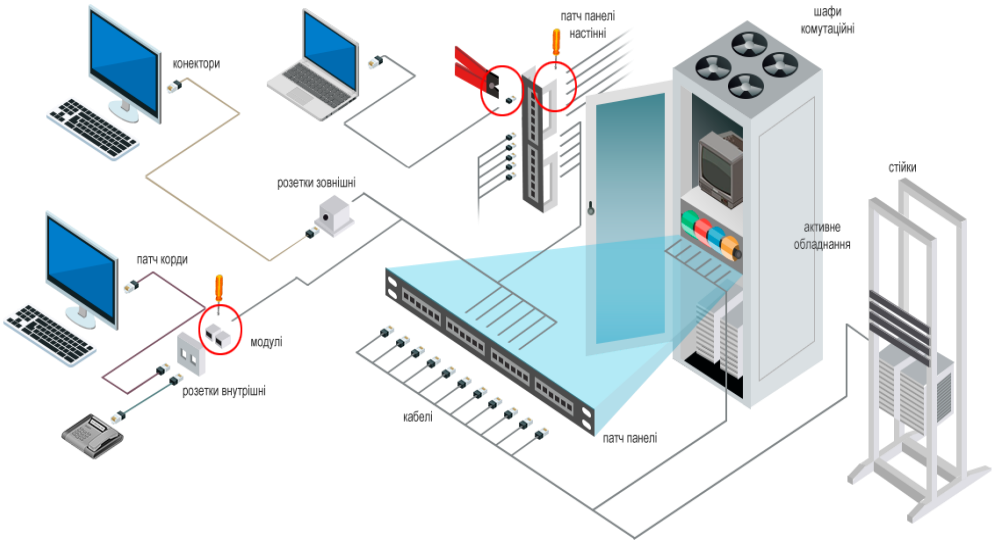

Рисунок 1.6 – Зразок ЦОД

Системи моніторингу та управління забезпечують автоматичне керування та диспетчеризацію інженерного обладнання будівлі, роботу агрегатів, підтримання заданих параметрів роботи обладнання та їх оперативної зміни.

Автоматизації та диспетчеризації підлягає наступне інженерне обладнання:

– система вентиляції та кондиціювання повітря;

– система теплопостачання;

– система холодопостачання;

– системи електропостачання;

– протипожежне обладнання та пристрої пожежогасіння;

– системи безпеки [19].

Цілодобовий моніторинг, комплексний аналіз параметрів обладнання, попередження відмов та мінімальний час реакції – це найважливіші вимоги до диспетчерських служб, що контролюють інженерні підсистеми ЦОД, а робота персоналу в подібних службах стає все більш відповідальною.

Автоматизована система диспетчеризації та управління (АСДУ) є цілісною платформою для управління всіма інженерними підсистемами й створюється як багаторівнева автоматична система, що забезпечує контроль стану та управління технологічним обладнанням ЦОД з виведенням даних на екрани автоматизованих робочих місць операторів. АСДУ веде безперервний моніторинг інженерних систем із реєстрацією основних параметрів та забезпечує контроль та управління інженерним комплексом з єдиного диспетчерського центру.

Організація диспетчерського центру на основі рішення АСДУ дозволяє запровадити нові стандарти якості в управлінні експлуатаційно-забезпечуючим обладнанням, підвищити експлуатаційну готовність ЦОД, знизити поточні витрати на керування інженерними системами, забезпечити документування та протоколювання збоїв, створити базу для оперативного усунення аварійних ситуацій.

Сучасна АСДУ має трирівневу архітектуру:

– нижній рівень утворюють периферійні пристрої та інженерне обладнання, що формують первинні дані;

– другий рівень – контролери, які приймають і обробляють інформацію, та мережу передачі даних;

– верхній рівень – це ПЗ, що надає засоби візуалізації, архівації, публікації даних, що надходять [20].

На робочі місця диспетчерів (АРМ) надходить структурована консолідована інформація у потрібному форматі. Аналітичний модуль постійно відстежує робочі параметри систем на предмет відхилення від норми й здатний автоматично запускати процедури згідно з закладеними інструкціями, наприклад, подати тривожний сигнал або запустити аварійний дизель-генератор. Важливе завдання аналітичного модуля – завчасні попередження про майбутні відмови.

Зібрані дані можна:

– передати операторам та подати їх у легко читаному вигляді;

– зберегти в базі даних;

– проаналізувати та подати у вигляді статистичних звітів;

– використовувати як сигнал, що управляє, при реакції на певні події для запуску систем в автоматичному режимі.

До складу рішення може входити система відеоспостереження, що одночасно з сигналом тривоги виводить картинку з аварійною підсистемою на монітор оператора. Як правило, у системі передбачено Web-інтерфейс, крім того, її можна інтегрувати з системами моніторингу IT-інфраструктури ЦОД.

Інженерні системи ЦОД складаються з великої кількості взаємопов'язаного обладнання, тому при настанні будь-якої тривожної події важко визначити, де конкретно виникла проблема. Для прикладу візьмемо проблему в контурі живлення між розподільчим щитом і активним мережевим обладнанням. Система локалізує проблему, визначає рівень можливих наслідків та відображає інформацію про конкретну систему у вікні тривог. Екранна форма зі схемою системи показує відносини між взаємозалежним обладнанням та можливими наслідками неполадок в окремих компонентах.

АСДУ централізовано фіксує подію у базі даних та сповіщає диспетчера про виникнення проблеми та необхідність її вирішення. Далі система визначає рівень

серйозності події та надає події певний пріоритет. Пріоритет необхідний, щоб підвищити ефективність реакції персоналу на подію. Наприклад, якщо сигналізація, що спрацювала, говорить про необхідність заміни фільтра системи кондиціювання повітря, оператор повинен розуміти, у які терміни й з яким пріоритетом вирішити ситуацію, що склалася.

Одна з найважливіших функцій АСДУ – своєчасне сповіщення про ситуації, що виникли, всіх відповідальних осіб, які обслуговують підсистеми ЦОД. Система має функції оперативного оповіщення диспетчерів, адміністраторів та керівних осіб об'єкта електронною поштою або за допомогою SMS-повідомлень, а також інтегрується з іншими доступними способами сигналізації відповідно до встановленого регламенту [19].

Система виводить повідомлення про вихід параметрів, що відстежуються за встановлені раніше межі, а також повідомлення про критичний час напрацювання експлуатованого інженерного обладнання. Наприклад, це можуть бути дані про стан акумуляторних батарей, температуру та вологість у стійках. Інформація надається в доступному вигляді для адміністраторів і диспетчерів, а також легко читається.

Відмова обладнання може бути наслідком не лише надто високої температури, а й швидкої зміни. Система відстежує температуру та вологість на рівні стійок з обладнанням та сповіщає диспетчера про те, що зафіксовано потенційно небезпечні значення температури та вологості. Хронологічні дані та параметри навколишнього середовища можуть виводитися у вигляді графіків, що легко читаються.

З появою в ЦОД нового обладнання, потреби в електроживленні та охолодженні можуть перевершити наявні ресурси, результатом чого стануть перебої у роботі. Зокрема, інженерні системи ЦОД вимагають додаткової уваги в міру старіння батарей ДБЖ. Рівень старіння батарей залежить від інтенсивності їх використання та температури. АСДУ відстежує споживання струму для кожної гілки ланцюга або стійки і повідомляє відповідальних осіб про ситуації, що загрожують виникненням навантаження. Вона також інформує їх про всіх ДБЖ, у

яких час автономної роботи виявляється меншим за мінімум або у яких перевищується порогове значення навантаження.

Несправність обладнання або ліній подачі електроживлення, а також некоректні дії обслуговуючого персоналу можуть призвести до знеструмлення обладнання. АСДУ оперативно повідомляє диспетчера про наявність або відсутність напруги живлення на споживачах.

Неякісне електроживлення призводить до виходу з ладу або передчасного зношення обладнання. Зміна навантаження на систему електроживлення (включення/вимкнення кліматичного обладнання, додавання обладнання ЦОД тощо) може спричинити ситуацію, коли система безперебійного електроживлення не в змозі забезпечити резервування. АСДУ надає обслуговуючому персоналу централізовану інформацію про якість електроживлення та розподіл навантаження по ЦОД у режимі реального часу, а також зберігає цю інформацію в базі даних для подальшого з'ясування причин відмови обладнання.

Оперативне відстеження стану обладнання, що забезпечує гарантоване та безперебійне електроживлення (ДБЖ, ДДУ), неможливе без централізованого збору та відображення інформації з цих пристроїв. АСДУ надає диспетчеру централізовану інформацію про стан обладнання, що забезпечує.

Кліматичний режим ЦОД може порушуватися через неправильні режими роботи кліматичного обладнання. Через нерівномірний розподіл обладнання в ЦОД іноді виникають зони локального перегріву, що може вимагати змін у режимах роботи кліматичного обладнання. Обслуговуючий персонал не завжди помічає тимчасовий вихід температури і вологості за межі норми, що призведе до проблем щодо причин збоїв у роботі активного обладнання. Крім того, кліматичний режим ЦОД може порушуватися через неправильні режими роботи або аварії на кліматичному обладнанні. АСДУ відстежує температуру і вологість у телекомунікаційних стійках і повідомляє диспетчера про те, що вони досягли потенційно небезпечних значень, а також зберігає цю інформацію в БД і видає її у зручному для подальшого аналізу вигляді. Система надає диспетчеру інтерфейс для

зміни режимів роботи кліматичного обладнання та оперативно повідомляє відповідальних про збої в його роботі.

На АСДУ також покладено функції мінімізації наслідків пожежі у ЦОД. У разі виникнення пожежі несвоєчасне оповіщення персоналу, а також робота кондиціонерів та неузгодженість роботи інших підсистем у ЦОД може ускладнити роботу системи пожежогасіння та знизити її ефективність. АСДУ повідомляє диспетчера про спрацювання пожежної сигналізації та станції пожежогасіння, а також має можливість автоматично відключити кондиціонери та вентиляцію. Після спрацьовування системи пожежогасіння необхідно визначати якість повітря у приміщеннях та виводити цю інформацію на АРМ диспетчера.

1.3 Особливості автоматизованої системи SCADA

1.3.1 Основні завдання SCADA-систем

Для моніторингу й управління параметрами технологічних процесів в автоматизованих системах промислових підприємств, міської інфраструктури та будівель використовуються системи SCADA.

SCADA представляє з себе програмний пакет, призначений щоб виконувати функції обробки зібраних даних, архівування збереженої інформації про певний об'єкт моніторингу та допомагає забезпечити управління ним в реальному часі. ПЗ даного класу як правило є частиною АСК ТП, АСКОЕ, або системи екологічного моніторингу, також широко використовується в наукових експериментах, автоматизації будівлі і т. д. Ці системи використовують в усіх галузях діяльності, де необхідно забезпечувати операторний контроль технологічних процесів. В основному, SCADA є частиною АСК, диспетчерської системи, що відповідає за моніторинг технологічних параметрів і дистанційне керування обладнанням [20].

Сучасний ринок SCADA-систем пропонує дуже великий вибір ПЗ для різних задач, що будуть працювати безпосередньо на комп’ютері, такі як OpenSCADA, Rapid SCADA, FreeSCADA, SCADA -ГИНЭС, Inductive Automation Ignition – для функціонування на комп’ютері, та WebSCADA – для функціонування на Веб-

сервері, тобто вона включає сервер WebSCADA і клієнтські термінали, такі як КПК, ПК або мобільні телефони з веб-браузером. Таким чином, маючи пристрій з виходом до мережі Інтернет можна керувати системою, що підключена до мережі WebSCADA віддалено з будь-якої точки планети.

Сучасні системи SCADA повинні мати значний набір функцій, щоб бути конкурентноспроможними на ринку таких послуг. Серед усіх можна виділити такі основні завдання:

– збір даних від апаратури процесу й дистанційне керування обладнанням. Ведення баз даних реального часу;

– створення графічного інтерфейсу для моніторингу та управління процесом оператором. Витяг інформації з баз даних і представлення оператору в зручному вигляді для аналізу;

– автоматизація виконання робочих процесів, прийняття рішень оператором;

– розрахунок вторинних показників ефективності виробництва, статистики ходу процесу, роботи устаткування тощо;

– виконання деяких функцій управління (блокування, некритичне регулювання тощо);

– автоматична генерація тривог і повідомлень;

– підготовка рапортів, зведень, звітів та іншої експлуатаційної документації;

– архівування історії, тривог і дій оператора;

– розмежування прав доступу по категоріям користувачів. Забезпечення безпеки і контролю над діями оператора;

– резервування найбільш важливих складових системи (серверів, мереж, клієнтів);

– горизонтальний обмін даними з суміжними системами АСК ТП і передача даних на верхні рівні управління.

Пристрої SCADA є дуже важливими в вирішенні різних проблем, що можуть виникати й вони, як ніщо інше, найкраще підходять для реалізації моніторингу, контролю і управління підприємством.

SCADA працює з сигналами, які обмінюються даними по каналах, щоб забезпечити користувача дистанційним управлінням будь-яким обладнанням у цій системі. Також SCADA реалізує розподілену базу даних, або базу даних тегів, яка містить в собі теги або точки на всій установці. Ці точки представляють єдине вхідне або вихідне значення, яке контролюється системою SCADA в централізованій диспетчерській кімнаті. Точки зберігаються у розподіленій базі даних мітки значення-час. Дуже поширеним є налаштування систем SCADA для отримання метаданих, таких як шляхи реєстрації програмованого логічного контролера (ПЛК) та статистика тривоги [21]. Хоча ці системи спрощують задану інфраструктуру, їх компоненти досить складні. Існує п’ять важливих складових частин системи SCADA:

– людино-машинний інтерфейс (HMI);

– диспетчерська система;

– віддалений термінал (RTU);

– програмовані логічні контролери (ПЛК);

– інфраструктури зв’язку.

1.3.2 Основні компоненти SCADA-систем

SCADA-системи містять такі компоненти [20]:

– драйвери або сервери введення-виведення – програми, що забезпечують зв'язок SCADA з промисловими контролерами, лічильниками, АЦП й іншими пристроями введення-виведення інформації;

– диспетчерська система (головний термінал) (MTU англ. Master Terminal Unit) – збирає дані про процес і відправляє команди процесору (керування);

– програмований логічний контролер (PLC англ. Programmable Logic Controller) використовується як польовий пристрій у зв'язку з вищою, ніж у RTU спеціального призначення економічністю, універсальністю і гнучкістю;

– комунікаційна інфраструктура (CS англ. Communication System) для реалізації промислової мережі;

– система реального часу – програма, яка забезпечує обробку даних у межах заданого тимчасового циклу з урахуванням пріоритетів;

– людино-машинний інтерфейс (англ. Human Machine Interface, HMI) – пристрій, який подає дані про хід процесу людині і це дозволяє операторові контролювати процес і управляти ним;

– абонентський кінцевий блок (віддалений термінал) (RTU англ. Remote Terminal Unit), що під'єднується до датчиків процесу, перетворює сигнал з датчика в цифровий код і відправляє дані в диспетчерську систему;

– програма-редактор для розробки людино-машинного інтерфейсу;

– система логічного управління – програма, яка відповідає за виконання призначених для користувача скриптів логічного управління в SCADA-системі. Набір редакторів для їх розробки;

– база даних реального часу – програма, що забезпечує збереження історії процесу в режимі реального часу;

– система управління тривогами – програма, що забезпечує автоматичний контроль технологічних подій, віднесення їх до категорії нормальних, що попереджають або аварійних, а також обробку подій оператором або комп'ютером;

– генератор звітів – програма, що забезпечує створення призначених для користувача звітів про технологічні події. Набір редакторів для їх розробки;

– зовнішні інтерфейси – стандартні інтерфейси обміну даними між SCADA та іншими додатками. Зазвичай OPC, DDE, ODBC, DLL тощо.

Системи SCADA мають доступну потужність хмарних обчислень; ці системи можуть повідомляти про точність в режимі реального часу та використовувати хмарні середовища для реалізації більш складних алгоритмів. В іншому випадку ці алгоритми не можна було б реалізувати на традиційних PLC або RTU. Навіть не будучи на підприємстві, працівники можуть отримати доступ до обчислювальних ресурсів, таких як мережі, сховища, сервери та засоби управління [21].

Загальна схема АС SCADA представлена на рис. 1.7.

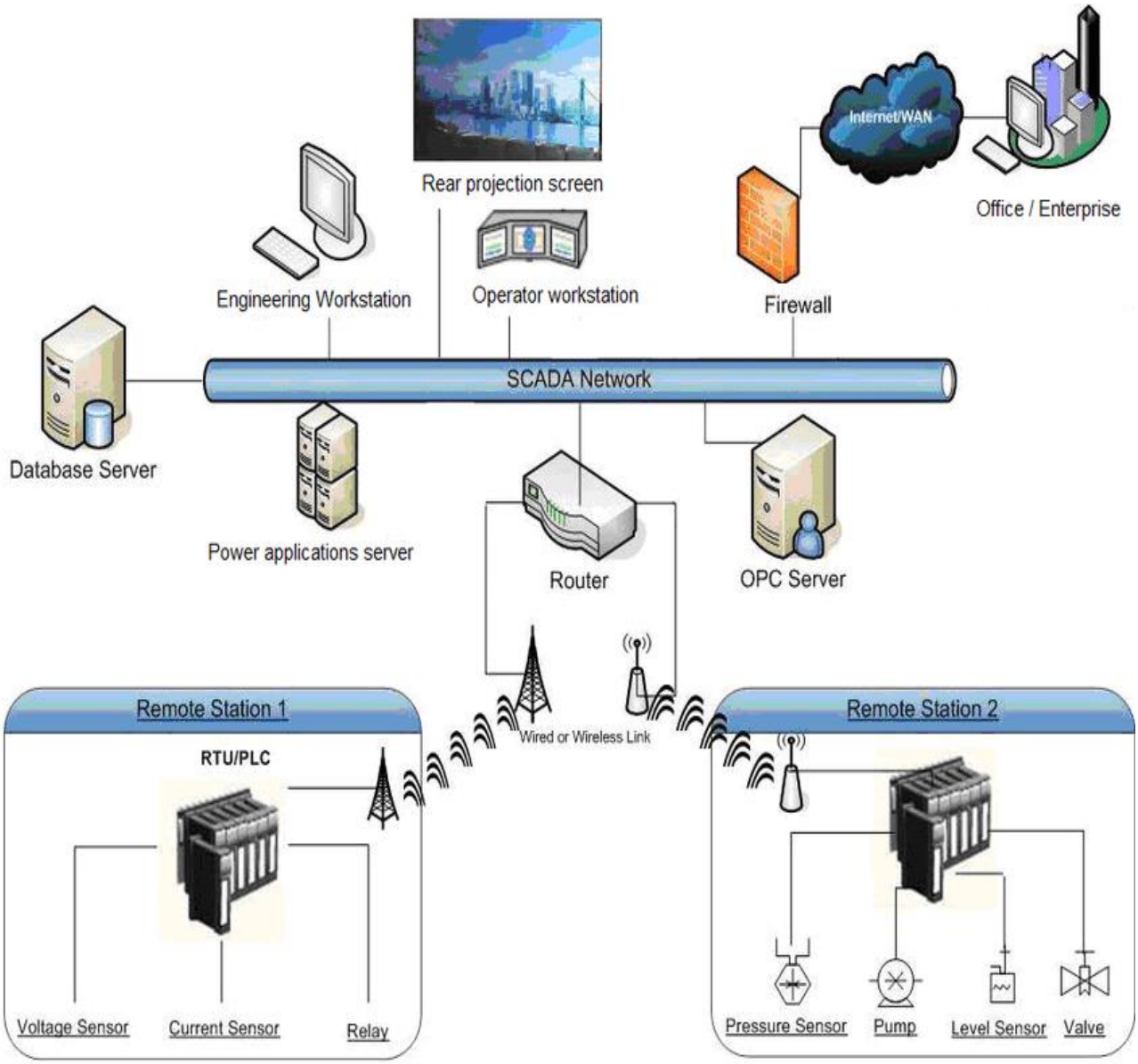

Рисунок 1.7 – Загальна схема АС SCADA

Хмарні обчислення можуть підтримуватися двома способами: система SCADA працює на місці, підключена безпосередньо до комунікаційної інфраструктури, і доставляє інформацію до хмари, або SCADA-система повністю працює в хмарній мережі та віддалено підключена до комунікаційної інфраструктури. Якби практичним не був доступ до елементів керування на відстані, хмарні обчислення за допомогою додатків SCADA все ще дуже вразливі до кібератак. Якби система була атакована, хакери могли отримати доступ до організаційних даних та ресурсів [21].

Базова система SCADA представлена на рис. 1.8.

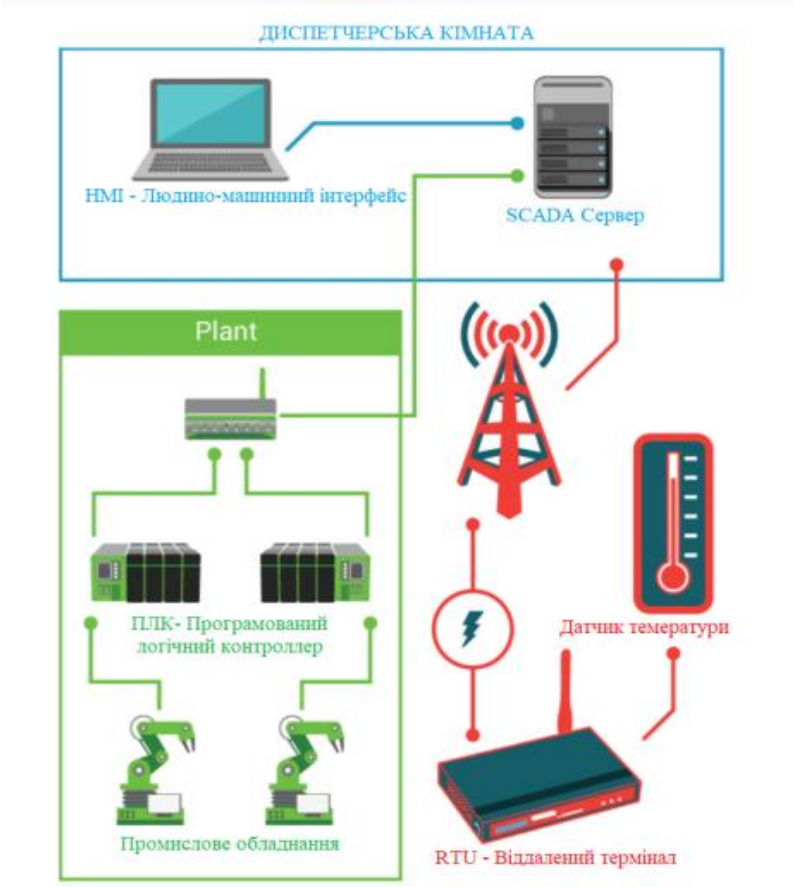

Рисунок 1.8 – Базова система SCADA

Перевагами програмного забезпечення SCADA є:

– зменшення собівартість виробництва – застарілим виробничим підприємствам потрібні були сотні операторів, щоб точно синхронізувати та керувати заводським обладнанням під час виробництва. Це значно збільшує собівартість продукції. Використовуючи інструменти SCADA для управління виробничим підприємством можна керувати цілим виробничим підрозділом лише з 2 або 3 операторами;

– глобальний виробничий центр – підприємства, які працюють у кількох місцях для виробництва або операцій, використовують програмне забезпечення SCADA для контролю за всім процесом;

– контроль стихійних лих – керівники виробництва високого рівня в глобальних центрах управління можуть дистанційно зупинити виробниче підприємство, коли в локальній системі SCADA виникає помилка або промислова аварія;

– точність – виробничі процеси, які повинні забезпечити максимальну точність, отримують вигоду від SCADA. Ця програма без особливих зусиль керує сотнями роботизованих рукояток, механізмів подачі та виробників, щоб виробляти послідовну партію продукції за партією;

– технічне обслуговування та безпека заводського обладнання – програма SCADA постійно перевіряє систему, щоб виявити, чи є у будь-якої машини механічні проблеми. Якщо він виявляє такий інструмент, негайно зупиняє процес збереження кінцевих продуктів. Додаток також нагадує групі технічного обслуговування про майбутні профілактичні заходи та продовжує термін служби виробничих інструментів;

– покращення процесу – аналізуючи дані, зібрані програмою HMI, можна без особливих зусиль покращити продукт, процес і продуктивність кожної партії. База даних також дає чітке уявлення про кількість споживчих товарів, що виробляються. Збільшити або зменшити виробництво можна одним натисканням кнопки.

1.3.3 Вибір SCADA-системи

Сьогодні існує досить велика кількість SCADA-систем, що різняться за своїми можливостями, вартістю, зручністю розробки і т. д. Проте кожна із існуючих систем має свої труднощі у використанні. Спробуємо сформулювати вимоги до ідеальної SCADA:

– необхідна висока швидкість роботи. Це означає, що не повинно бути ніяких інтерпретаторів, на виході треба отримати машинний код, що виконується;

– можливість легко та без істотних ризиків змінювати поведінку існуючих компонентів або додавати свої;

– прозорість форматів зберігання налаштувань та історичних даних. Наприклад, необхідність зробити специфічну вибірку з архівів для побудови звітів повинна виливатися в тривалий реверс-інжиніринг інструментів, які входять до складу SCADA;

– простота та швидкість розробки. Необхідно звести до мінімуму написання коду і максимально використовувати візуальне програмування;

– зручне та сучасне середовище розробки (IDE). Необхідні звичні інструменти будь-якого програміста: автодоповнення коду, контроль версій тощо;

– низька вартість стороннього ПЗ, а в ідеалі безкоштовність та відкритість вихідного коду.

Ураховуючи наведені вище вимоги, можна запропонувати рішення – треба взяти існуюче якісне середовище для візуального програмування і створити бібліотеку компонентів, заточених під специфічні завдання SCADA-систем. Розмірковуючи таким чином, зупинимо свій вибір на Qt. Тут і безліч готових компонентів, і відмінна IDE, і величезна спільнота розробників. Qt вражає внутрішньою логічністю та багатством бібліотеки.

Створений набір можна умовно поділити на кілька груп:

а) компоненти, призначені для забезпечення обміну даними з ПЛК. Система тегів. Фактично деякий буфер між драйверами та іншими частинами бібліотеки, що забезпечує доступ до даних із різних компонентів програми. Драйвер-клієнт для OPC DA2. На даний момент це найпопулярніший спосіб обміну даними з ПЛК і досить складно знайти хоч скільки поширений пристрій без OPC-сервера;

б) забезпечення запису та доступу до архівної інформації:

1) система аварійних повідомлень;

2) журнали технологічних властивостей;

3) набір графічних компонентів (widgets).

Побудова графіків та трендів із журналів технологічних параметрів. Тут все класично – вибір і налаштування відображення накопичених даних.

Робота з аварійними повідомленнями – виведення активних повідомлень, підтвердження оператором (квітування), доступ до архівної інформації.

Відображення різних елементів мнемосхем. Як показали опитування, більшість компаній використовують власні іконки для показу станів технологічного устаткування. З цієї причини було створено компонент, що дозволяє виводити графічні зображення (в тому числі й з ефектом миготіння) залежно від значень тегів.

Набір компонентів для полегшення створення елементів користувача.

1.3.4 Принцип роботи програмної системи SCADA

Розглянемо докладніше завдання доступу на сервер SCADA із мобільних пристроїв. Порівняємо особливості такого доступу. Проаналізуємо 5 поширених методів, як експортувати дані із системи SCADA та запустити її в роботу мобільного застосунку. Експорт даних із системи SCADA представлений на рис. 1.9.

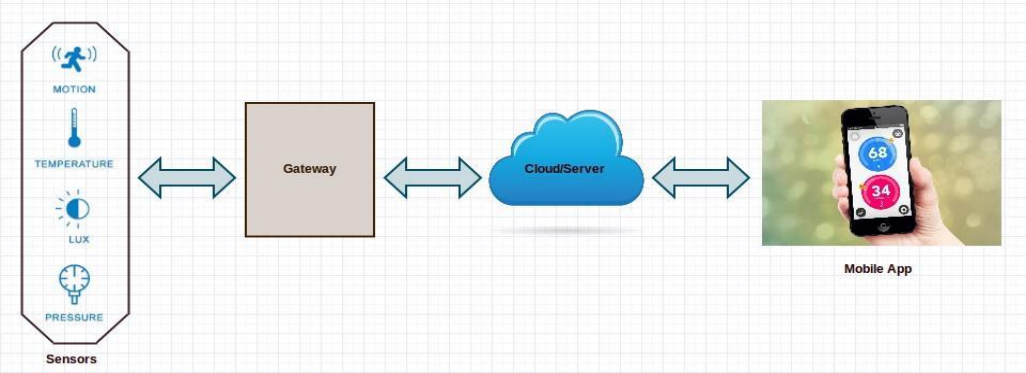

Рисунок 1.9 – Експорт даних із системи SCADA

Конектори бази даних. Одним із поширених способів з'єднання двох різних баз даних є використання конекторів баз даних. Приклади конекторів бази даних включають ODBC, JDBC або CLI. Ці з'єднувачі баз даних, по суті, є API, розробленими спеціально для того, щоб дозволити виробничому програмному забезпеченню отримувати дані з архіву даних SCADA.

Спосіб налаштування конекторів бази даних залежить від типів серверів даних, до яких слід підключитися, і драйверів, які підтримуються виробничим програмним забезпеченням. Загалом процес стає складнішим, якщо архіватор даних використовує інший формат або систему керування базою даних, ніж виробниче програмне забезпечення, або якщо потрібно отримати дані з кількох систем SCADA чи джерел збору польових даних.

У багатьох випадках потрібно буде виконати процес вилучення, перетворення та завантаження (ETL), щоб переконатися, що дані успішно отримані з першої бази даних, правильно відформатовані для другої, а потім ефективно завантажені.

Проста схема, що демонструє процес ETL, зображена на рис. 1.10.

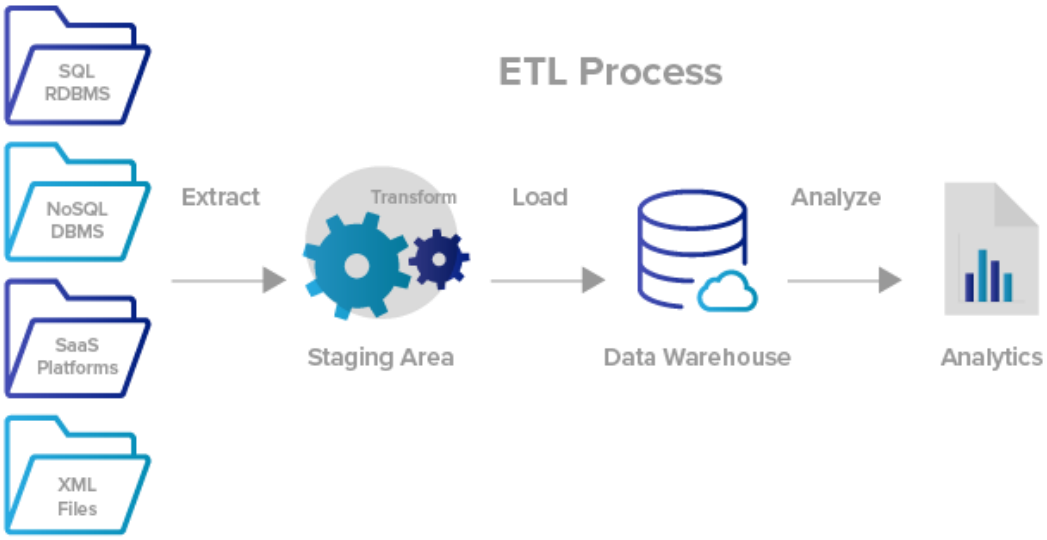

Рисунок 1.10 – Проста схема, що демонструє процес ETL

З’єднувачі баз даних, наприклад ODBC, як правило, є компетенцією IT-команд, оскільки вони вимагають глибокого розуміння мереж баз даних і мов запитів. Компанії з великими серверами даних і хостами IT-ресурсів, як правило, простіше з конекторами баз даних, оскільки вони частіше використовуються великими організаціями. З іншого боку, невеликі компанії з обмеженими IT-ресурсами можуть виявити, що їм потрібна стороння допомога для ефективного впровадження такого роду підключень до баз даних.

Протоколи передачі файлів. Другим підходом до інтеграції даних SCADA у виробниче програмне забезпечення є використання протоколу передачі файлів (FTP). Як випливає з назви, FTP призначений для переміщення файлів, які містять робочі дані, між двома серверами.

FTP, по суті, є формою ETL, яка використовує файли як носій. Витягуються дані з одного сервера, перетворюються (поміщаючи у файл із рядками та стовпцями, які може зрозуміти другий сервер), а потім налаштовується система, за допомогою якої сервер-одержувач може інтерпретувати та завантажувати їх у

якусь базу даних. FTP працює майже з будь-якими типами файлів. Інтегруючи дані часових рядів SCADA з виробничим програмним забезпеченням, інженери часто використовують такі формати, як CSV, XLS, MDB або ODS. Загалом FTP забезпечує відносно простий та інтуїтивно зрозумілий спосіб передачі даних від SCADA до виробничого програмного забезпечення, за умови, що кожна сторона підтримує процес створення/споживання файлів.

Проте FTP зазвичай потребує певної IT-підтримки, оскільки компаніям доводиться налаштовувати FTP-сервери для надсилання та отримання файлів і гарантувати, що все безпечно. Існує два типи протоколів безпеки FTP (SFTP і FTPS).

Проста діаграма FTP, що демонструє додаткову безпеку, яку пропонує SFTP, представлена на рис. 1.11.

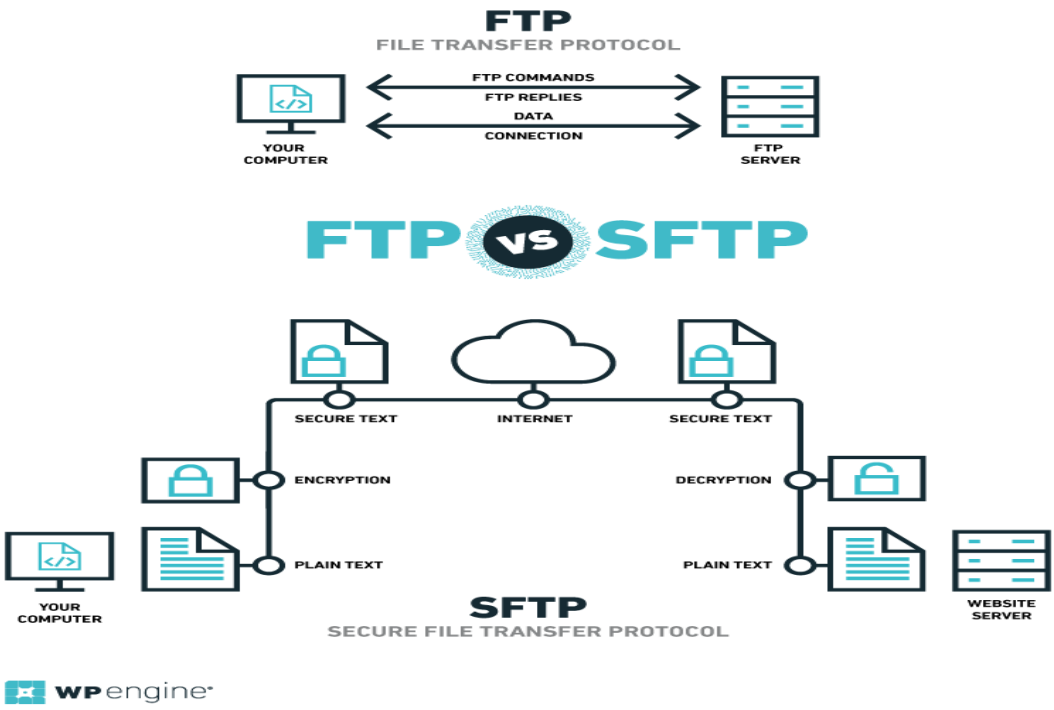

Рисунок 1.11 – Проста діаграма FTP, що демонструє додаткову безпеку, яку пропонує SFTP

Це випробуваний і надійний метод. Однак якщо підприємство не працює із зовнішнім постачальником, це може зайняти багато часу або бути надто складним

для зайнятої внутрішньої команди. FTP все ще потребує певних IT-ноу-хау, оскільки передбачає налаштування FTP-сервера, а потім направлення клієнта Production Accounting на цей сервер для отримання файлів за розкладом.

Інтегровані інструменти Cloud. Якщо дані SCADA та/або виробниче програмне забезпечення використовує бази даних, розміщені на хмарних платформах сторонніх розробників, як-от AWS або Azure, можна використовувати інструменти, створені для цих хмар, щоб спростити передачу даних.

Наприклад, Azure пропонує AZ Copy, утиліту командного рядка для перенесення даних до баз даних Azure або з іншої бази даних Azure, або з зовсім іншого джерела. AWS пропонує такі інструменти, як DataSync, для передачі даних між серверами або обліковими записами.

Більшість інженерів-виробничників також потребуватимуть IT-підтримки в цій сфері. Але оскільки ці фірмові інструменти розроблені для підтримки певних хмарних інфраструктур, вони досить добре працюють для цієї інфраструктури та пропонують певну гнучкість щодо джерел даних, з яких ви можете отримувати дані. Цей підхід може бути хорошим варіантом, якщо підприємство вже використовує хмарне сховище для SCADA та/або виробниче програмне забезпечення.

API. Деякі сучасні хмарні рішення SCADA зазвичай постачаються з інтерфейсом прикладного програмування (API). На високому рівні API функціонує як перекладач між програмами. Це спосіб взаємодії з багатьма функціями та послугами системи SCADA, включаючи експорт даних часових рядів у програмне забезпечення для виробництва або інструменти аналізу.

Щоб отримати дані з SCADA API, робиться «запит» до API. API взаємодіє з програмою SCADA, щоб отримати дані, які запитували, а потім повертає ці дані в певному форматі. Наприклад, REST API часто передають дані як структуровані словники ata у формі файлів JSON.

Інженери-технологи задоволені роботою API, оскільки більшість із цих мов пропонують готові бібліотеки, які дозволяють інженерам легко надсилати запити

API, щоб отримати саме ті дані, які їм потрібні, не потребуючи значної чи будь-якої підтримки IT.

Хоча вони не часто використовуються для створення постійних зв'язків між SCADA та виробничим програмним забезпеченням, хмарні API SCADA дають інженерам можливість створювати сценарії на мові, яку вони вибирають, для доступу до певних типів даних.

Хороша документація є важливою особливістю будь-якого хмарного SCADA API.

Крім того, API призначені не лише для запиту даних, вони також корисні для повернення інформації до SCADA, тобто інженери можуть створювати сценарії для автоматизації критичних функцій.

MQTT. Останній метод (див. рис. 1.12) – це Message Queuing Telemetry Transport (MQTT). Сьогодні MQTT набирає обертів, оскільки команди IoT і SCADA все частіше використовують цю технологію для передачі даних між програмами. MQTT використовує методологію «публікувати / підписуватися» (скорочено pub-sub), за якою клієнт MQTT може як публікувати, так і підписуватися на дані з різних тем. Брокери MQTT також можуть надсилати дані між клієнтами.

Однією з переваг MQTT є те, що цей метод надзвичайно легкий і ідеально підходить для потоків даних часових рядів, які використовуються в SCADA та виробничому програмному забезпеченні. MQTT також неймовірно масштабований завдяки своїй структурі pub-sub. Користувачам не потрібно керувати з'єднаннями між кожним клієнтом у мережі – для цього створені брокери.

Брокер «обслуговує» потрібний кредит на підставі заданих параметрів. Брокери MQTT схожі тим, що вони автоматично підключають клієнтів. Користувачі отримують саме ті дані, які їм потрібні, залежно від тем, на які вони підписані.

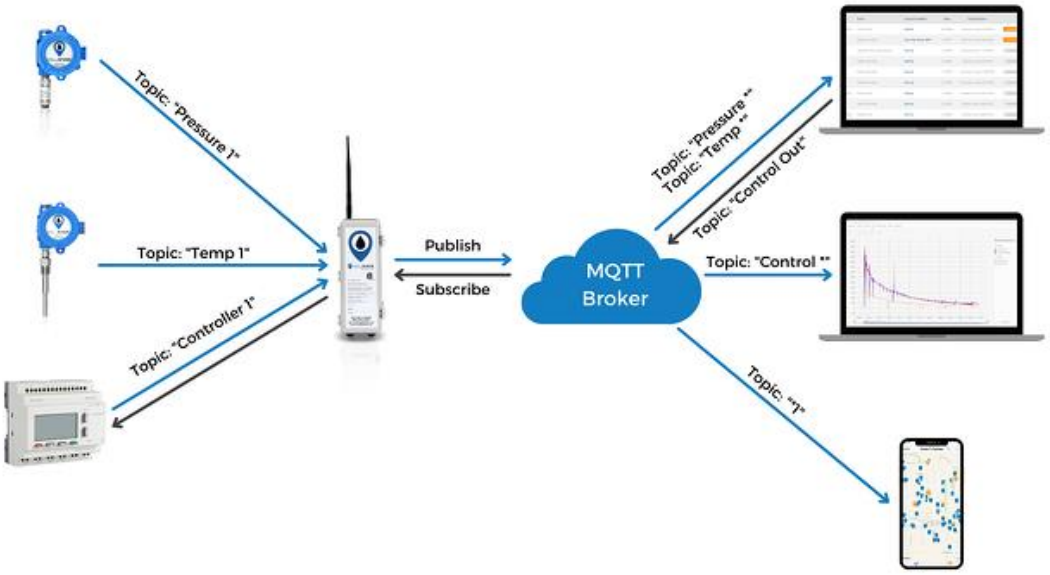

Рисунок 1.12 – MQTT

MQTT дозволяє пристроям і клієнтам підписуватися лише на актуальну інформацію. Як і API, MQTT починає набувати популярності в промислових умовах. Сучасні команди SCADA ефективно використовують MQTT, але застарілі системи SCADA та виробниче програмне забезпечення не завжди його підтримують. Оскільки API надає розробнику засоби для швидкої розробки програмного забезпечення, для нашого дослідження оберемо API.

Інтерфейс користувача (UI) мобільного застосунку або веб-панелі, представленої вище, є HMI системи SCADA. Це разом із серверною бізнес-логікою, базою даних (хмарним сервером) і шлюзом становить рішення SCADA для контролю та моніторингу пристроїв у мережі IoT.

Шлюз IoT забезпечує сумісність між сенсорною мережею IoT і хмарним сервером. Через шлюз дані датчиків зберігаються на хмарному сервері. Хмарний сервер є місцем розміщення алгоритмів, які реалізують бізнес-логіку.

Датчик виявляє зміни в навколишньому середовищі, такі як зміна температури, вологості, а також діє як блок накопичення даних.

У рамках IoT-проекту на основі рішення SCADA може бути підключено кілька автоматизованих систем. Розглянемо детально особливості програмної системи SCADA.

Функції, які підтримуються SCADA:

а) Функції керування системою:

1) управління пристроями: кількість вузлів або пристроїв, які контролюються або обслуговуються, видно з інтерфейсу користувача та може бути розподілено за різними параметрами.

2) керування користувачами: дозволи та ролі можна визначити на панелі керування користувачами відповідно до рівня користувачів. Можна призначити певні ролі адміністратора та обмежити доступ користувачам іншого рівня до різних даних, а також до інтерфейсу.

б) Функції керування пристроєм і системою. У системі SCADA переважає як віддалений, так і локальний доступ, який широко варіюється залежно від користувачів і галузей, для яких створено програму. Цікаво взяти до уваги, що програмне забезпечення SCADA має інтелект, щоб розпізнавати різні комунікаційні сигнали встановлених різних пристроїв.

Система SCADA, як і будь-яка інша, повинна мати надійне шифрування/дешифрування даних, що є процесом захисту даних під час обміну даними. Під час кодування також слід дотримуватися криптографічних протоколів, таких як TLS/SSL. Усе це гарантує створення повністю функціональної, захищеної та потужної системи програмного забезпечення SCADA.

1.4 Висновки до першого розділу

У першому розділі кваліфікаційної роботи розкрито поняття автоматизованих систем, принципи побудови АСУ. Визначено спільні властивості автоматизованих систем будь-якого призначення. Було доведено, що додержання кожного з принципів дає змогу отримати певний економічний ефект.

Оглянуто види моніторингу, основну сферу практичного застосування моніторингу, завдання, які повинен розв’язувати моніторинг. Доведено, що використання моніторингу дозволяє підприємствам покращити процеси розробки та якість продукції, сприяє зниженню витрат підприємства.

Перший розділ містить також технічне обґрунтування доцільності використання для моніторингу і управління параметрами технологічних процесів в автоматизованих системах промислових підприємств системи SCADA. У розділі проаналізовано обрану предметну область, виявлено переваги програмного забезпечення SCADA-систем,сформульовано вимоги до ідеальної SCADA, обґрунтовано вибір SCADA для дослідження. Докладно розглянуто завдання доступу на сервер SCADA із мобільних пристроїв.

2 МЕТОДИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

2.1 Аналіз програмних продуктів-аналогів

Підприємцю потрібна мобільність: можливість підключення до робочого місця з будь-якої точки світу, швидке розгортання та переміщення робочих місць, доступ до інформації та робочих інструментів для управління. Для бізнесмена мобільність означає завжди залишатися на зв'язку та бути в курсі справ, контролювати свій бізнес, а для співробітників – можливість працювати де завгодно з мінімальною кількістю обладнання.

Для того щоб сформулювати вимоги до нашого мобільного застосунку, потрібно проаналізувати існуючі аналоги. Ми розглянемо чотири застосунки: Мобільний застосунок IGSS, INDAS IT Solutions, CloudView NMS, Універсальний мобільний застосунок до платформи SIMATIC.

Мобільний застосунок IGSS надає можливість візуалізації та управління автоматизованими системами з IGSS зі смартфону. Програма надає можливість операторам, які мають відповідні права, підключитися до одного або декількох підприємств з системою IGSS і відстежувати та контролювати певні речі за допомогою смартфонів:

– підключитися до вибраного підприємства;
– моніторити активні тривоги;
– змінити налаштування тривог та підтвердити їх;
– оновити значення об'єктів IGSS;
– надіслати команди вибраним об’єктам;
– переглянути тренди за тиждень.

INDAS IT Solutions – програмне забезпечення SCADA на замовлення для будь-якого бізнесу. Застосунок використовується для збору даних, керування виробничими інструментами та віддаленого моніторингу виробничих підприємств, які працюють з нафтою та природним газом.

Для розробки інструментів SCADA він використовує такі основні компетенції:

– мобільний інтернет-зв’язок із наскрізним шифруванням даних для безпеки виробництва;

– телеметричні технології, дистанційне управління обладнанням і моніторинг процесів;

– мультимодальне підключення даних через GPRS, WLAN, 3G тощо забезпечує максимальний час безвідмовної роботи системи.

Будучи системою керування та моніторингу мережі, мобільний застосунок CloudView NMS здатний здійснювати моніторинг SCADA для підприємства чи галузі. Це програмне забезпечення показує узгоджену географічну картину повної мережі підприємства і, таким чином, дозволяє також налаштовувати та усувати несправності.

Його можна запускати як спрощену програму з графічним інтерфейсом користувача або як розподілену систему для кількох віддалених користувачів, які працюють одночасно.

Мобільний застосунок пропонує цілодобовий моніторинг у режимі реального часу, SMS-повідомлення та сповіщення електронною поштою. За допомогою нього компанії можуть відстежувати доступність і продуктивність мереж, серверів, пристроїв, інтерфейсів, карт, програм, VoIP та IT-автоматизації.

Компанія SIMATIC створила інноваційну, але масштабовану систему SCADA для візуалізації процесів, які дозволяють контролювати автоматизовані процеси. Ця відкрита система містить функціональні можливості, необхідні в усіх галузях для візуалізації дуже складних завдань і додатків SCADA. Також є мобільне рішення SCADA для планшетів і смартфонів, яке можна використовувати для надання інформації.

SIMATIC пропонує поєднання ефективного проектування, захисту даних найвищого рівня та потужного архівування. З його допомогою компанії можуть контролювати операції та ефективно аналізувати виробництво.

Усі розглянуті мобільні застосунки та аналогічні розробки, написані під конкретні апаратні можливості виконавчих механізмів та завдань підприємств, фізичних осіб, мають широкий функціонал і досить цікаві доповнення.

Проаналізувавши представлені застосунки, ми можемо сформулювати функціональні вимоги до власного програмного продукту, а саме:

– вхід до облікового запису шляхом авторизації на початку роботи застосунку;

– обрання потрібної філії, якщо їх декілька;

– перегляд списку характеристик та значень, параметрів, характеристик системи підтримки діяльності підприємства;

– перегляд архівних значень параметрів характеристик системи підтримки діяльності підприємства у вигляді графіків;

– редагування налаштувань керуючих параметрів системи підтримки діяльності підприємства шляхом вибору дискретних значень з фіксованого діапазону.

2.2 Особливості класифікації мобільних застосунків

Перед тим, як починати розробку мобільних застосунків, слід чітко визначитися з головними функціями майбутньої розробки. Від того, які функції має виконувати мобільний застосунок, залежить тривалість його розробки, рівень складності і, звичайно, ціна. Загалом визначають чотири основні типи мобільних застосунків [16]:

– корпоративні – їх метою є спрощення роботи компанії, швидка передача даних між працівниками або отримання корпоративної інформації. Цільовою аудиторією таких мобільних застосунків є, насамперед, працівники компанії, а також реальні та потенційні клієнти та партнери. Розробка мобільних застосунків такого типу простіша, для їх дизайну використовуються кольори та елементи корпоративного стилю компанії;

– контентні – це мобільні застосунки, основна мета яких надавати різного роду інформацію (текстову або у відео чи аудіоформаті). Головним чином, така розробка мобільних застосунків здійснюється для засобів масової інформації, радіо, телеканалів чи порталів;

– сервісні – мета яких випливає з назви, а саме вони надають певні сервісні послуги, тобто виконувати завдання, які ставить користувач, у режимі реального часу. Розробка мобільних застосунків такого типу є складною, адже вони повинні працювати, як годинник: починаючи від калькулятора або будильника і закінчуючи програмами для роботи з великими обсягами тексту або графіки;

– ігрові – основне завдання таких мобільних застосунків – розважати, але це не означає, що воно єдине.

Отже, розробка мобільних застосунків такого типу ускладнюється необхідністю вдало розмістити контекстну або пряму рекламу. Для розробників мобільних застосунків більш важливою є класифікація з точки зору їх структури:

– веб-застосунки можна назвати мобільною версією сайту з розширеним інтерактивом. Веб-застосунки не розміщуються в спеціалізованих магазинах, а використовують для роботи браузер. Швидкість роботи таких застосунків залежить від якості Інтернет-з'єднання, крім того вони характеризуються низькою вартістю і швидкими термінами реалізації, є кроссплатформенними;

– гібридні застосунки або генератори мобільних застосунків – щось середнє між нативними і веб-застосунками, встановлюються через офіційні магазини, мають обмежений доступ до апаратної частини пристрою, за функціоналом і якістю наближені більше до нативних застосунків, проте дешевше їх, залежать від фреймворку, який використовувався розробником цього застосунку;

– нативні застосунки пишуться на мовах програмування під конкретну платформу і вбудовуються в операційну систему, працюють швидко і коректно, володіють перевагою як по функціоналу, так і за швидкістю роботи інших мобільних застосунків. Дані програми також мають доступ до апаратної частини пристроїв: камери, мікрофону, акселерометру, телефонною книги і т. п., економно витрачають ресурси, працюють повністю або частково при відключеному Інтернет-

з'єднанні. Звичайно, мають високу вартість і великі витрати за часом написання програми, як наслідок того, що розробник повинен володіти спеціальними знаннями в середовищі розробки, а також з тієї причини, що кожній платформі відповідає своя мова програмування.

Розроблений мобільний застосунок є сервісним нативним мобільним застосунком, оскільки він написаний під конкретну платформу (Android) та надає користувачу сервісні послуги, а саме – можливість моніторити автоматизовані системи.

2.3 Функціонал мобільного застосунку

Мобільний застосунок надає можливість операторам, які мають відповідні права, підключитися до одного або декількох підприємств і відстежувати та контролювати певні речі за допомогою смартфонів:

– підключитися до вибраного підприємства;
– моніторити активні тривоги;
– змінити налаштування тривог та підтвердити їх;
– оновити значення об'єктів;
– надіслати команди вибраним об’єктам;
– переглянути звіти за тиждень.

Мобільний застосунок дозволяє автоматично виконувати такі функції:

– контроль і регулювання температури всередині підприємства;
– моніторинг вологості повітря;
– моніторинг споживання струму, додаткове освітлення при нестачі природного світла;
– моніторинг водопостачання;
– стан обладнання;
– пожежна безпека.

Розроблений застосунок – це новий спосіб моніторингу, який пропонує кінцевим користувачам промислових систем швидко створювати власні

персональні огляди процесів. Крім того, за допомогою застосунку можна отримати кілька нових способів відображення різних даних процесу, наприклад, графіки, дані LOG та BCL можуть відображатися в одному поданні та оновлюватись автоматично.

2.4 Вибір засобів реалізації

2.4.1 Огляд мобільних операційних систем

Оскільки програма, що розробляється, є мобільною, необхідно визначитися з вибором операційної системи, на якій виконуватиметься розробка клієнтської частини цієї програми.

Мобільний застосунок, який створюється має бути уніфікований для усіх гаджетів з операційною системою Android, яка є на цей час найпопулярнішою у світі, як і IOS.

Згідно з даним World Mobi за 2020 рік, 70% ринку мобільних операційних систем займає саме Android (рис. 2.1).

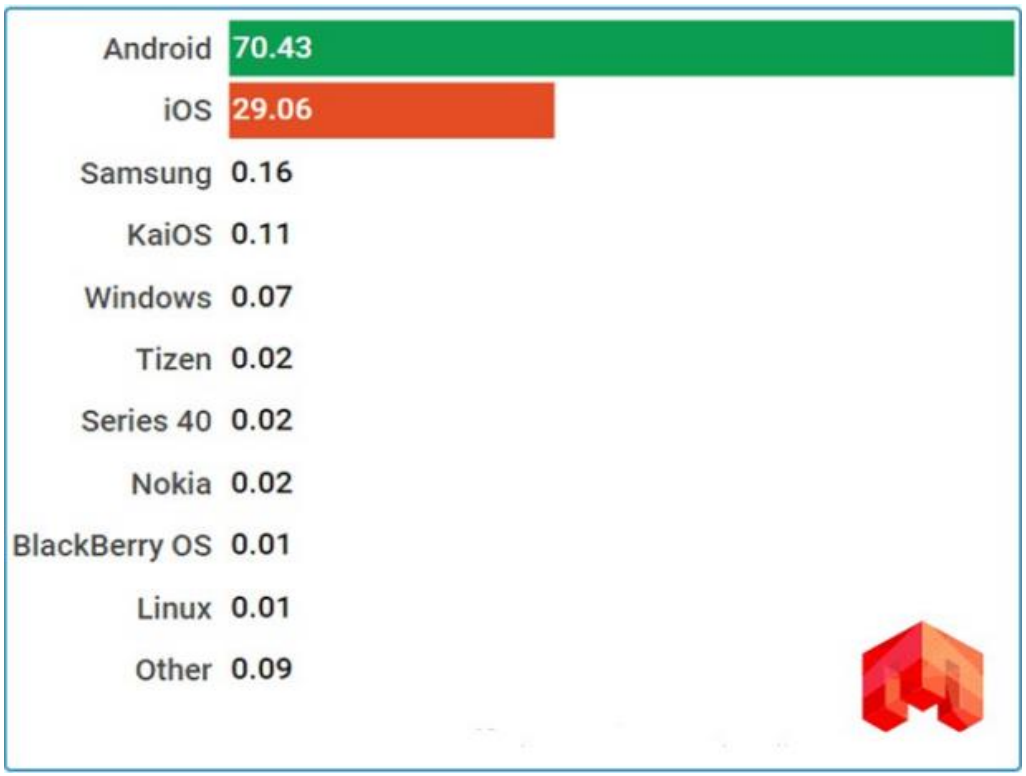

Рисунок 2.1 – Статистика встановлених мобільних операційних систем

Android – операційна система, розроблена на базі Linux компанією Open Handset Alliance за допомогою Google. У числі переваг Android можна назвати гнучкість, відкриті вихідні коди, висока швидкодія, відсутність утруднень при взаємодії з сервісами Google та багатозадачність.

Одним з недоліків Android є наявність великої кількості актуальних версій, оскільки версії для багатьох пристроїв виходять із значним запізненням, через що розробникам доводиться створювати застосунки, орієнтовані на старі версії. Відкритість коду має і свій зворотний бік – зниження рівня безпеки.

У процесі розробки застосунків для Android є ряд особливостей:

– на відміну від iOS, мобільні застосунки для Android являють собою взаємозв'язок окремих, логічно відокремлених елементів. Тобто не можна просто взяти і перенести застосунок на іншу мобільну операційну систему, переписавши код з однієї мови програмування на іншу. Потрібно закладати зовсім іншу архітектуру. Інший підхід спостерігається також і в інших аспектах. Наприклад, сучасна іконка Android застосунків може мати різну форму в залежності від налаштувань операційної системи. Дизайнер повинен це враховувати і переконатися, що логотип виглядає прекрасно і гармонійно у всіх варіантах;

– Material Design – ціла філософія побудови призначеного для користувача інтерфейсу. Офіційна документація по цьому підходу містить сотні документів, детально описують як його принципи, так і конкретні приклади правильного і неправильного використання правил для кожного елемента інтерфейсу. Іконка програми може мати різну форму на різних екранах. Але вона завжди повинна гарно виглядати. Кнопка, яка виступає над активною областю, повинна бути тільки одна (одного кольору). Не можна використовувати кілька виступаючих кнопок різних кольорів. Кнопка, панель навігації, іконка і всі інші елементи повинні слідувати цим правилам, якщо потрібно сконструювати гарний Material інтерфейс і отримати пропозицію від Google на просування нашого мобільного застосунку в Google Play;

– рекомендованою Google мовою програмування для Android на сьогодні є Kotlin, а не Java. Різниця між ними суттєво менша, ніж між Objective-C і Swift для iOS, але все ж таки це різні підходи до розробки;

– тестування на великій різноманітності фізичних пристроїв (не емуляторів) має надзвичайно важливе значення. Навіть воно в силу величезної кількості різних телефонів на ринку не забезпечує безпроблемне функціонування на всіх доступних моделях, але принаймні знижує ймовірність проблем на найбільш популярних пристроях.

Створення подібного продукту тягне за собою дослідження і пізнання в таких предметних областях, як мережеві протоколи передачі даних, програмування для мобільних пристроїв на мові програмування, розробка дизайну для мобільних застосунків, проектування інтерфейсів мобільних застосунків. Створення мобільних застосунків для Android на об’єктно-орієнтованій мові, найчастіше виконуються на Java, на якій розроблено 90% та на інших. Серед інших набуває популярності нова мова Kotlin, яка була офіційно представлена на Google I/O у травні 2017 року. Через 2 роки Google визнали Kotlin кращою мовою програмування для розробки Android-застосунків. Після чого мова програмування Java відійшла на другий план. Насамперед це означало, що розробка нових інструментів, таких як бібліотеки та функції Android Studio, буде націлена саме на мову Kotlin. Серед його основних переваг можна виділити автоматичне виділення типів даних, функції-розширення та підтримка, окрім об’єктно-орієнтованої, функціональної парадигми. Проте найголовнішою перевагою Kotlin над мовою Java, є захист від виключення NullPointerException. Саме через це його популярність різко виросла згідно з даними опитування на StackOverflow. У 2019 році Kotlin увійшов до п’ятірки найбільш улюблених мов програмування серед спільноти.

2.4.2 Процес розробки дизайну

У створенні мобільного застосунку найважливішу роль виконує розробка дизайну. Користувачів, у першу чергу, утримує саме дизайн, його зовнішній

вигляд, функціональність і зручність. Дизайн програми – це візуальне оформлення програми, а також створення структури, заснованої на логіці користувача поведінки. Іншими словами, це не тільки зовнішній вигляд, але і зручність використання. Дизайн застосунку можна поділити на UX і UI частини. Дизайн мобільного застосунку повинен бути не тільки привабливим зовні і функціональним. Користування програмою має бути інтуїтивно зрозумілим для користувачів. Вчинення будь-яких дій не повинно займати багато зусиль або часу, адже користувачі від такого застосунку найімовірніше відмовляться, не бажаючи розібратися в тому, як це працює. Виявити, проаналізувати, протестувати і впровадити інтуїтивно зрозумілий дизайн допоможе UX (User eXperience). Ґрунтуючись на досвіді попередніх програм, застосунків конкурентів, UX дизайн застосунків дозволить виробити найбільш ефективний і інтуїтивно зрозумілий інтерфейс. Корисний мобільний застосунок має бути орієнтованим в першу чергу на працівників.

Не менш важливий елемент – це UI дизайн програми. UI визначає кольори і загальне візуальне оформлення застосунку, то наскільки зручно користувачеві буде натискати на кнопки і наскільки читабельним буде текст.

Увесь процес розробки дизайну можна поділити на декілька етапів:

– спочатку необхідно визначити цілі, що і навіщо потрібно користувачам застосунку. Треба обдумати реалізацію з боку дизайну;

– створити начерки ескізів дизайну застосунка, для подальшого втілення в життя цих ескізів. Заздалегідь опрацювати на папері приблизний зовнішній вигляд програми і функціонал який в ньому буде. Це дозволить уникнути зайвої роботи при повноцінному відображенні екранів;

– після того, як функціонал визначений, необхідно створити схему того як цей функціонал буде пов’язаний. Зручніше за все, буде реалізувати цю схему у вигляді пов’язаних між собою блоків різних екранів з описом функцій в кожному з них. Ця схема має назву user flow. Основне призначення створення user flow – це опрацювання логіки функціоналу застосунка, того як, що і навіщо має працювати.

Зазвичай user flow складається з трьох типів фігур:

– прямокутники – використовуються для представлення екранів;

– ромби – використовуються для умов (наприклад, натискання кнопки входу в систему, свайп вліво, збільшення);

– стрілки – з’єднують екрани і умови разом. User flow дуже корисний, тому що він дає логічне уявлення про те, як програма має працювати і вирішувати завдання;

– wireframes – створення ескізів для всіх екранів. Необхідний для створення каркасної структури пов’язаних між собою сторінок застосунка, визначити розташування кнопок, іконок, картинок і ярликів. Бажано його створити на етапі створення скетчів. Найчастіше, дизайнери використовують готові шаблони з сервісів на зразок UI Stencils. Це дозволяє їм пристойно заощадити час на малюванні ескізів wireframes;

– розробка дизайну застосунка і прототипів. Для повноцінної розробки дизайну, раніше використовувався Adobe Photoshop. Однак з розвитком технологій, більшість дизайнерів мобільних застосунків від нього відмовилися через непотрібність. Зараз найчастіше використовують Figma, Sketch, Adobe XD.

2.4.3 Створення дизайну застосунку

Насамперед, було розроблено user flow, у якому описано основну логіку застосунку. На другому етапі роздруковано макет із сервісу UI Stencils, на якому було намальовано ескіз для головного екрану, а також визначено загальний стиль користувацького інтерфейсу для всього застосунка. Після визначення основної логіки та створення ескізу, зобразимо наявну інформацію на макеті за допомогою Adobe XD. Користуючись цією програмою, розробимо детальний макет, який містить дизайн всіх екранів застосунку, а також задамо логіку переходу між ними. А ще є змога протестувати створений інтерфейс за допомогою вбудованого функціонала Adobe XD.

На рис 2.2 міститься поведінка застосунку для головного екрану, з якого можна продовжити навігацію по цьому застосунку та переходити на екрани моніторингу автоматизованих систем.

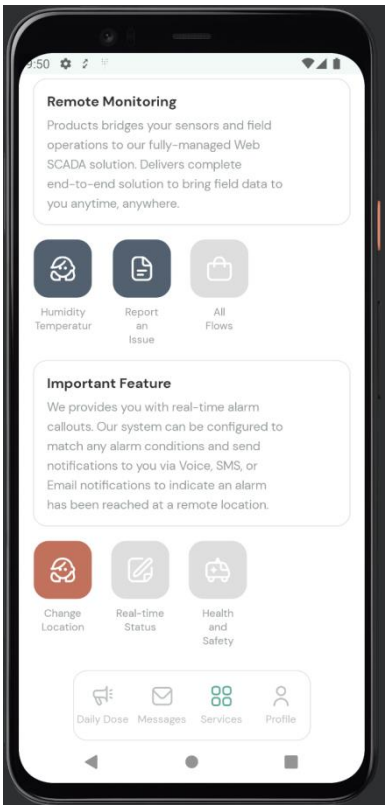

Рисунок 2.2 – Поведінка застосунку для головного екрану

2.5 Висновки до другого розділу

У другому розділі проаналізовано чотири програмні продукти – аналоги мобільного застосунку, сформульовано функціональні вимоги до власного програмного продукту. Визначено головні функції мобільного застосунку, вибрано засоби реалізації розробки мобільного застосунку, описано процес розробки дизайну.

3 РЕАЛІЗАЦІЯ АЛГОРИТМУ ТА РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ

3.1 Концепція будівлі клієнт-серверної системи

Особливості клієнт-серверної системи полягають у тому, що користувач надсилає певний запит на сервер, де той системно обробляється та кінцевий результат надсилається клієнту. У можливості сервера входить одночасне обслуговування кількох клієнтів.

Клієнт – локальний комп'ютер на стороні віртуального користувача, який виконує надсилання запиту до сервера для надання даних або виконання певної групи системних дій.

Сервер – дуже потужний комп'ютер або спеціальне системне обладнання, яке призначається для вирішення певного кола завдань процесу виконання програмних кодів. Він виконує роботи сервісного обслуговування за запитами клієнтів, надає користувачам доступ до певних системних ресурсів, зберігає дані або БД (див. рис. 3.1).

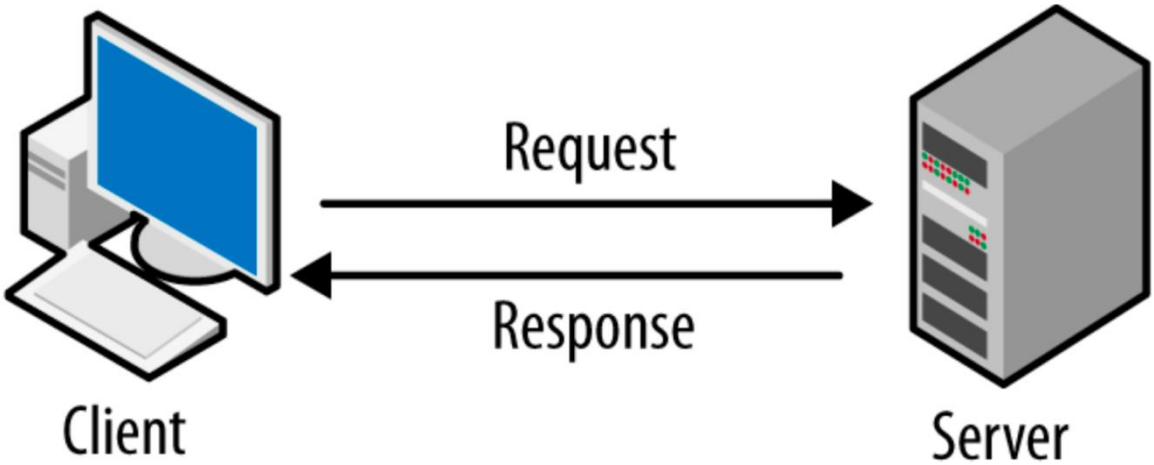

Рисунок 3.1 – Взаємодія клієнт-серверної системи

Якщо одночасно надходить більше одного запиту, такі запити встановлюються у певну чергу й відповідно сервером виконуються по черзі. Іноді

запити можуть мати свої пріоритети. Частина запитів із вищими пріоритетами постійно виконуватимуться в першочерговому порядку.

Параметри, які можуть бути реалізовані на стороні сервера:

– зберігання, захист та доступ до даних;

– робота з клієнтськими запитами;

– процес надсилання відповіді клієнту.

Параметри, які можуть бути реалізовані з боку клієнта:

– майданчик з надання користувальницького графічного інтерфейсу;

– формулювання запиту до сервера та його подальше відправлення;

– отримання результатів запиту та надсилання додаткової групи команд (запити на додавання, оновлення інформації, видалення групи даних).

Архітектура системи клієнт-сервер формулює принципи віртуального спілкування між локальними комп'ютерами, проте правила й принципи взаємодії перебувають усередині протоколу.

Мережевий протокол – це особливий набір правил, виходячи з якого виконується точне взаємодія між комп'ютерами всередині віртуальної мережі. TCP/IP – сукупність протоколів передачі. TCP/IP – це особливе позначення всієї мережі, яка функціонує з урахуванням протоколів TCP, і навіть IP.

TCP – вид протоколу, який є сполучною ланкою для встановлення якісного з'єднання між 2 пристроями, передачі даних та верифікації їх отримання.

IP – протокол, до якого входить коректність доставки повідомлень за обраною адресою. У цьому інформація ділиться на пакети, які можуть постачатися по-різному.

MAC – вид протоколу, виходячи з якого відбувається процес верифікації мережевих пристроїв. Усі пристрої, які підключені до мережі Інтернет, містять свою оригінальну MAC-адресу.

ICMP – протокол, відповідальний за обмін даними, але з використання процесу передачі інформації.

UDP – протокол, керуючий передачею даних, але дані проходять верифікацію при отриманні. Цей протокол працює швидше, ніж протокол TCP.

HTTP – протокол передачі інформації (гіпертексту), з урахуванням якого функціонують все сьогоднішні сайти. У його можливості входить процес запитування необхідних даних у віртуально віддаленої системи (файли, веб-сторінки та інше).

FTP – протокол передачі з особливого файлового сервера на ПК кінцевого пользователя.

POP3 – це класичний протокол простого поштового з'єднання, який відповідає за передачу пошти.

SMTP – вид протоколу, який може встановлювати правила передачі віртуальної пошти. Він відповідальний за передачу та верифікацію доставки, а також оповіщення про можливі помилки.

Є два види клієнт-серверних архітектур:

а) двохрівнева, що складається відразу з 2 вузлів:

1) сервер, який відповідає за отримання вхідних запитів та надсилання відповіді користувачу, застосовуючи при цьому власні ресурси системи;

2) клієнт, який може надавати користувальницький графічний інтерфейс.

Особливості роботи у тому, що у сервер надходить певний запит, потім його обробляють і дають безпосередньо, без додаткового застосування групи зовнішніх ресурсів.

б) трирівнева система складається з використання таких компонентів:

1) надання інформації – графічний користувальницький, прикладний об'єкт у вигляді сервера програми;

2) менеджмент ресурсів – сервер БД, який може надавати дані.

Особливість роботи у тому, що кілька серверів можуть обробляти клієнтські запити. Процес розподілу операцій може істотно знизити навантаження на сервер, що використовується.

Трирівнева архітектура може трансформуватися до багаторівневої можливістю встановлення групи додаткових серверів. Подібна віртуальна архітектура дозволяє значно підвищити ефективність функціонування

інформаційних систем, а також виконати оптимізований розподіл частини її програмно-апаратних ресурсів.

Клієнт-серверна архітектура представлена на рис. 3.2.

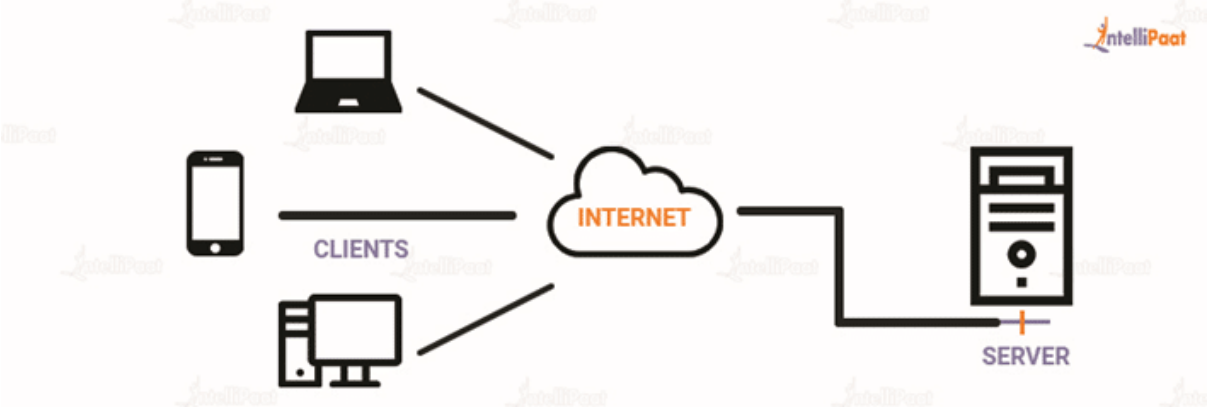

Рисунок 3.2 – Клієнт-серверна архітектура

Поки програмний застосунок відповідає архітектурі клієнт-сервер (тобто клієнт може надсилати та отримувати дані до API на сервері), можна створювати будь-який інтерфейс користувача на будь-якій платформі. Це вигідно, оскільки очікується, що сучасні програмні застосунки будуть доступні на кількох платформах і забезпечуватимуть узгоджену роботу на всіх пристроях.

Розроблений мобільний застосунок для моніторингу автоматизованих систем буде працювати в режимі онлайн. Клієнт-серверна версія роботи реалізована на основі класичної архітектури «клієнт-сервер», де взаємодіють такі частинки, як:

– клієнтський (мобільний) застосунок;
– сервер баз даних.

3.2 Алгоритм роботи мобільного застосунку

Алгоритм – набір інструкцій, що описують порядок дій виконавця для досягнення результату рішення задачі за визначене число дій. Для створення алгоритму необхідно знати:

– повний набір вихідних даних завдання (початковий стан об'єкта);
– мета створення алгоритму (кінцевий стан об'єкта);

– систему команд виконавця (тобто набір команд, які виконавець розуміє і може виконати).

Існує два методи розробки складних алгоритмів. Метод послідовної деталізації завдання (зверху-вниз) полягає в тому, що вихідна складна задача розбивається на підзадачі. Кожна з підзадач розглядається і вирішується окремо. Якщо які-небудь з підзадач складні, вони також розбиваються на підзадачі. Процес продовжується до тих пір, поки підзадачі не зведуться до елементарних. Вирішення окремих підзадач потім збираються в єдиний алгоритм розв'язання вихідної задачі. Метод широко використовується, тому що дозволяє вести розробку загального алгоритму одночасно кільком програмістам, які вирішують локальні підзадачі. Це необхідна умова швидкої розробки програмних продуктів. Складальний метод (знизу-вверх) полягає у створенні безлічі програмних модулів, що реалізують рішення типових задач. При вирішенні складного завдання програміст може використовувати розроблені модулі в якості допоміжних алгоритмів (процедур). Кожний алгоритм повинен відповідати ряду загальних вимог [24], а саме:

– дискретність – алгоритм повинен представляти процес вирішення завдання як послідовне виконання деяких простих кроків. При цьому для виконання кожного кроку алгоритму потрібно кінцевий відрізок часу, тобто перетворення вихідних даних у результат здійснюється в часі дискретно;

– детермінованість (визначеність) – у кожен момент часу наступний крок роботи однозначно визначається станом системи. Таким чином, алгоритм видає один і той же результат (відповідь) для одних і тих же початкових даних. У сучасному трактуванні у різних реалізацій одного і того ж алгоритму має бути ізоморфний граф. З іншого боку, існують імовірнісні алгоритми, в яких наступний крок роботи залежить від поточного стану системи і випадкового числа, що генерується. Однак при включенні методу генерації випадкових чисел в список «початкових даних», імовірнісний алгоритм стає підвидом звичайного;

– зрозумілість – алгоритм повинен включати тільки ті команди, які доступні виконавцю і входять в його систему команд;

– завершеність – при коректно заданих початкових даних алгоритм повинен завершувати роботу і видавати результат за кінцеве число кроків;

– масовість (універсальність) – алгоритм повинен бути застосовний до різних наборі початкових даних;

– результативність – завершення алгоритму певними результатами;

– алгоритм містить помилки, якщо призводить до отримання неправильних результатів або не дає результатів зовсім;

– алгоритм не містить помилок, якщо він дає правильні результати для будь-яких допустимих початкових даних.

Алгоритми в залежності від мети, початкових умов завдання, шляхів її вирішення, визначення дій виконавця поділяються таким чином:

– механічні алгоритми – задають певні дії, позначаючи їх в єдиній і достовірної послідовності, забезпечуючи тим самим однозначний необхідний або шуканий результат, якщо виконуються ті умови процесу, завдання, для яких розроблений алгоритм;

– гнучкі алгоритми, наприклад стохастичні, тобто імовірнісні та евристичні;

– імовірнісний алгоритм дає програму рішення задачі кількома шляхами або способами, що призводять до ймовірного досягненню результату;

– евристичний алгоритм (від грецького слова «еврика») – алгоритм, що використовує різні розумні міркування без строгих обґрунтувань;

– лінійний алгоритм – набір команд (вказівок), виконуваних послідовно в часі один за одним;

– розгалужений алгоритм – алгоритм, який містить хоча б одну умову, в результаті перевірки якої може здійснюватися поділ на кілька паралельних гілок алгоритму;

– циклічний алгоритм – алгоритм, що передбачає багаторазове повторення однієї і тієї ж дії (одних і тих же операцій) над новими вихідними даними. Цикл програми – послідовність команд, яка може виконуватися багато разів (для нових початкових даних) до задоволення деякої умови;

– структурна блок-схема, граф-схема алгоритму – графічне зображення алгоритму у вигляді схеми пов'язаних між собою за допомогою стрілок (ліній переходу) блоків – графічних символів, кожен з яких відповідає одному кроку алгоритму. Всередині блоку дається опис відповідної дії. Графічне зображення алгоритму широко використовується перед програмуванням завдання внаслідок його наочності.

Алгоритм розробленої автоматизованої системи є розгалуженим, так як містить декілька умов, у результаті перевірки яких здійснюється поділ на кілька паралельних гілок алгоритму.

3.3 Розробка ієрархічної моделі мобільного застосунку

У плані обробки взаємодії між інтерфейсом користувача і його логікою Android слідує архітектурному шаблону Model-View-ViewModel (MVVM) [23]. Model-View-ViewModel – це шаблон проектування застосунків для розділення коду інтерфейсу користувача і іншого коду. За допомогою MVVM декларативно визначається інтерфейс користувача (у нашому випадку, використовуючи XML) і використовується розмітка прив'язки даних, щоб пов'язати його з іншими рівнями, що містять дані і команди користувача. Шаблон MVVM організовує код так, що можна змінювати окремі його частини, не впливаючи на інші. Це дає багато переваг, серед яких:

– можливість використання ітеративного, довільного стилю написання коду;

– спрощене тестування модулів;

– більш ефективне використання інструментів проектування;

– підтримка взаємодії в команді.

При використанні шаблону MVVM Android-застосунок ділиться на такі частини:

– інтерфейс, який розробляється за допомогою технології XML;

– логіка користувацького інтерфейсу реалізується розробником як компонент ViewModel;

– функціональні зв'язки між інтерфейсом користувача і ViewModel реалізуються через Біндинг (bindings). Вони можуть бути написані в коді або визначені декларативним шляхом (Android використовує обидва типи). View – базовий клас для всіх віджетів користувацького інтерфейсу. Інтерфейс Android-застосунку є деревом екземплярів нащадків цього класу. Клас Activity і його підкласи містять логіку, що реалізує інтерфейс користувача. Цей клас відповідає ViewModel в архітектурному шаблоні ModelView–ViewModel (MVVM). Відношення між підкласом Activity і інтерфейсом користувача – це відношення один до одного; зазвичай кожен підклас Activity має тільки один пов'язаний з ним користувацький інтерфейс, і навпаки.

У кваліфікаційній роботі мобільний застосунок містить чотири рівня моделей представлення підкласів класу Activity (рис. 3.3).

Перший рівень – це вікно програми, де користувач повинен ввести свій логін і пароль. Якщо ж він не зареєстрований, то спочатку необхідно перейти на другий рівень – рівень реєстрації користувача, а потім повернутися на перший. Третій рівень – це рівень нижньої панелі навігації, де можна перейти в свій профіль, чати, щоденні звіти та побачити головний екран навігації. Четвертий рівень – це головні функції застосунка. Тут є можливість моніторити різні прилади, дивитися звіти з автоматизованих систем, а також надсилати власні помилки до обробки на сервер.

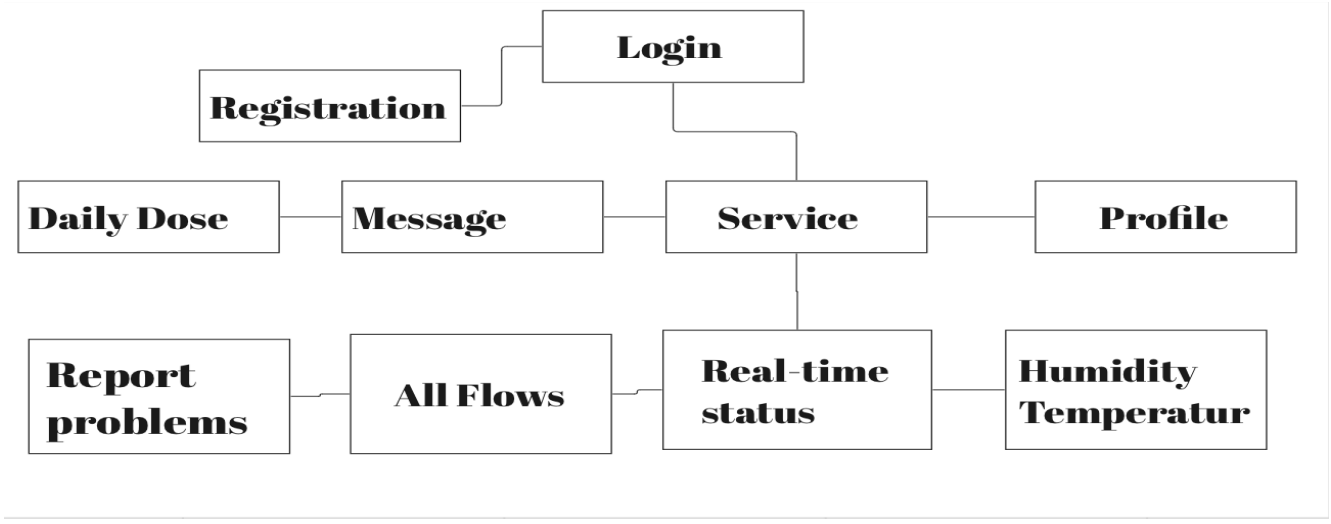

Рисунок 3.3 – Ієрархічна модель мобільного застосунку

3.4 Висновки до третього розділу

У третьому розділі розкрито поняття алгоритму роботи мобільного застосунку, описано процес розробки архітектури системи, ієрархічну модель. Описано чотири рівня моделей представлення підкласів класу Activity.

У розділі доведено переваги використання у розробленому мобільному застосунку розгалуженого алгоритму та класичної архітектури клієнт-сервер.

4 ПРОЦЕС РОБОТИ ПРОГРАМИ МОБІЛЬНОГО ЗАСТОСУНКУ

4.1 Завантаження застосунку

Наразі застосунок знаходиться у бета-тестуванні. Інсталяційний APK-файл був залитий в телеграм группу Android Developer, де програмісти можуть скачати застосунок та виявити наявні баги. Помилки будуть виявлені після бета-фази тестування, будуть враховуватися побажання користувачів. У результаті вийде перша стабільна версія застосунку, яка буде завантажена в Play Market, з якого кожна людина зможе встановити програму.

4.2 Вхід у застосунок

При першому запуску системи на екран виводиться вікно авторизації користувача, за допомогою якого необхідно задати логін та пароль для входу в аккаунт системи. На рис. 4.1 представлений скріншот авторизації.

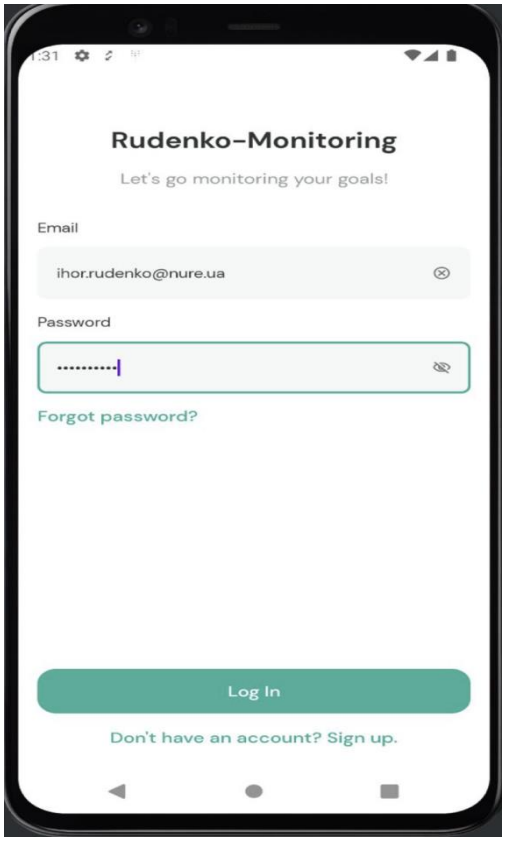

Рисунок 4.1 – Скріншот авторизації

4.3 Реєстрація

Реєстрація давно вже стала звичним процесом. Проте реєстрація повинна бути швидкою і не втомлювати користувачів великою кількістю полів для заповнення. Тому користувача просимо ввести тільки основну інформацію – ім’я, прізвище, пошту й пароль (див рис. 4.2). Поле електронна пошта є унікальним, тому не можна зареєструвати пошту, яка вже є в системі.

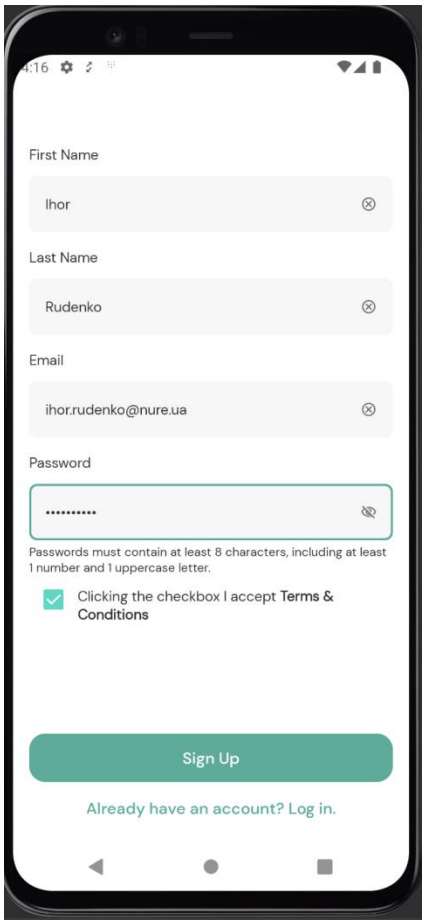

Рисунок 4.2 – Скріншот реєстрації

4.4 Головний екран

Після успішної авторизації буде здійснено вхід у систему і перед користувачем відкриється головна сторінка мобільного застосунку, на якій відображається нижня панель навігації та головні функції застосунку.

Головний екран застосунку розділений на 4 частини нижньої навігації: Profile, Message, Daily Dose, Service. Між ними можна перемикатися клацанням на вкладках внизу екрана (рис. 4.3). На екрані також зображені основні вкладки, куди можна перейти. Це перегляд звітів, моніторинг температури та вологості в приміщеннях, відкриття екрану з переліком пристроїв та їх показниками в реальному часі та багато інших корисних функцій.

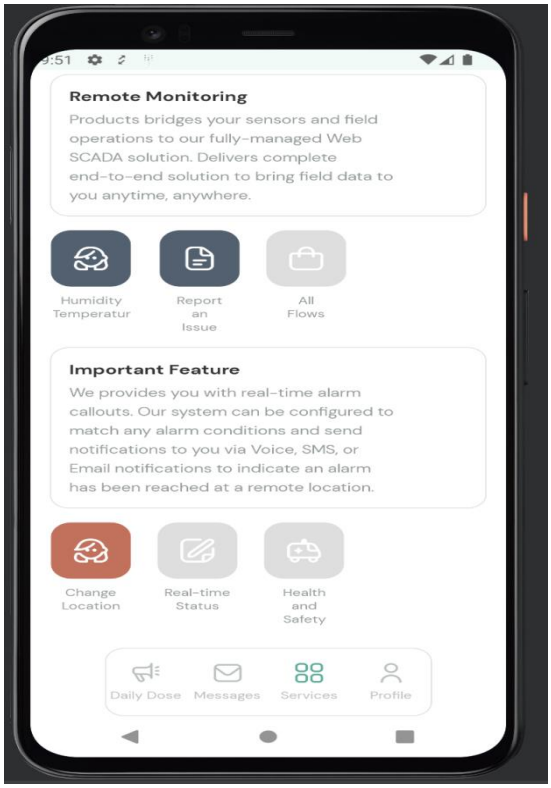

Рисунок 4.3 – Головний екран

4.5 Екран Daily Dose та процес отримання даних із серверу

Цей екран знаходиться у нижній панелі навігації, зображеній на рис. 4.4.

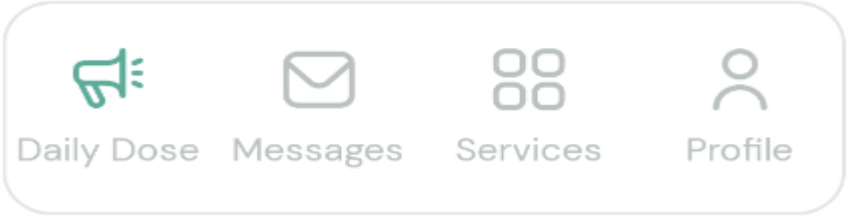

Рисунок 4.4 – Екран Daily Dose

На цій вкладці можна отримати звіти з серверу, оформлені вже в готові графіки та діаграми. Отримувати дані з серверу будемо по технології API, тому створимо ключ API для нашого застосунку за допомогою відкритого API сайту: https://www.scadacore.com/live/features/api.

Перш ніж створити API-ключ для інтерфейсу, спочатку потрібно зареєструватися на сайті. Після реєстрації в розділі API веб-сайту побачимо автоматично створений ключ API Scada-системи. Збережемо цей ключ API, оскільки нам знадобиться цей параметр для виклику з програми.

Створений Api key:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHBpcmVzQXQiOjE2NzA3NjkwMzcsImNyZWF0ZWRBdCI6MTY3MDY4MjYzNywiZW1haWwiOiJpaWlAaWlpLmlpaSIsImlhdCI6MTY3MDY4MjYzN30.UB1vsph2MXW4rIYhBJME0aU3SnbFizI92TLxQlBKEVQ.

Цей ключ будемо підставляти в наш базовий Url та підключатися до серверу. Повний базовий Url виглядає так:

https://www.scadacore.com/live/features/api/js?key=YOUR_API_KEY&callback,

де в YOUR_API_KEY будемо підставляти нещодавно згенерований api key для нашого аккаунту.

Наступним кроком буде виклик API з даними існуючої Scada-системи.

У дослідженні використаємо Retrofit для типу безпечного виклику клієнта HTTP.

Бібліотека retrofit спрощує взаємодію з REST API сайту, взявши на себе частину рутинної роботи. Авторами бібліотеки retrofit є розробники з компанії Square, які написали багато корисних бібліотек, наприклад, picasso, ohttp, otto.

Бібліотека корисна для запитів до різноманітних веб-служб з командами GET, POST, PUT, DELETE. Також вона може працювати в асинхронному режимі, що виключає зайвий код.

У цілому нам потрібно буде працювати лише з методами GET та POST для отримання та передачі інформації.

Для виклику API необхідна базова адреса URL-адреси та унікальний ключ користувача. Цей Url потрібно підставити при створенні retrofit клієнту в поле baseUrl на рисунку 4.5.

```
val key = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHBpcmVzQXQiOjE2NzA3NjkwMz
val url = "https://www.scadacore.com/live/features/api/js?key=${key}&callback"
    var retrofit = Retrofit.Builder()
        .baseUrl(url)
        .addConverterFactory(GsonConverterFactory.create()).build()
```

Рисунок 4.5 – Створення Retrofit клієнту для роботи із сервером

SCADACore Live надсилає різноманітні дані, щоб дозволити користувачам стежити за їх виробництвом та експлуатаційними даними в будь-який час, будь-де. Крім того, можна отримати багато різної інформації. Зокрема, перегляд даних вибраного пристрою, інформацію про включення сигналів тривог, нагадування про події, історичні дані та характеристики систем моніторингу, температуру, вологість у приміщенні та багато іншого.

Дані, які отримуємо назад з API, можуть бути використані для відображення в UI і зберігатимуться в локальній базі даних (SQLite), якщо користувач знаходиться в автономному режимі. Отримуючи дані назад у форматі JSON, потрібно перетворити їх на локальні прості Java-об'єкти (POJO) для доступу до даних.

У створеному застосунку необхідно вивести дані про приміщення, в якому розташовуються датчики Scada-системи. Увагу привертають такі параметри:

– температура;

– вологість;

– швидкість зміни температури в приміщенні;

– середньодобове сумарне енергоспоживання, розраховане на кожен місяць.

Для цього потрібно викликати API по Get-запиту – getInformationSpace (рис. 4.6).

```
@GET("information/space")
@RequiresAuth
suspend fun getInformationSpace(): ApiResponse<BaseResponse<SpaceInformation>>
```

Рисунок 4.6 – Get-запит на сервер

Нижче наведено дані JSON, які повертаємо:

"system": [

{

"id": 803,

"main": "Viasat-100",

"description": "Monitoring for Flow, Tank Level, Water Level, Pressures, and More",

"icon_current_temp": "https://www.scadacore.com/.s3.zonaws.com//uploads/user/profile_picture/343349/images/1668687890170_Screenshot_from_2022-04-13_17-20-08.png"

}

],

"base_average_daily_energy":
"https://www.scadacore.com/.s3.amazonaws.com//uploads/user/profile_picture/343434699/images/1668687890170_Screenshot_from_2022-04-13_17-20-08.png",

"main": {

"temp": 15,

"pressure": 1022,

"humidity": 72,

"temp_min": 15,

"temp_max": 15

},

"rate_temperature_change": {

"icon":
"https://www.scadacore.com/.s3.amazonaws.com//uploads/user/profile_picture/6943349/images/1668687890170_Screenshot_from_2022-04-13_17-20-08.png",

"name": "General",

"cod": 200

}

SCADACore Api пропонує різні варіанти отримання інформації. У нашому випадку отримуємо вже готові картинки з графіками швидкості зміни температури в приміщенні та середньодобове сумарне енергоспоживання.

Ці дані використовуємо для відображення в створеному застосунку (рис. 4.7).

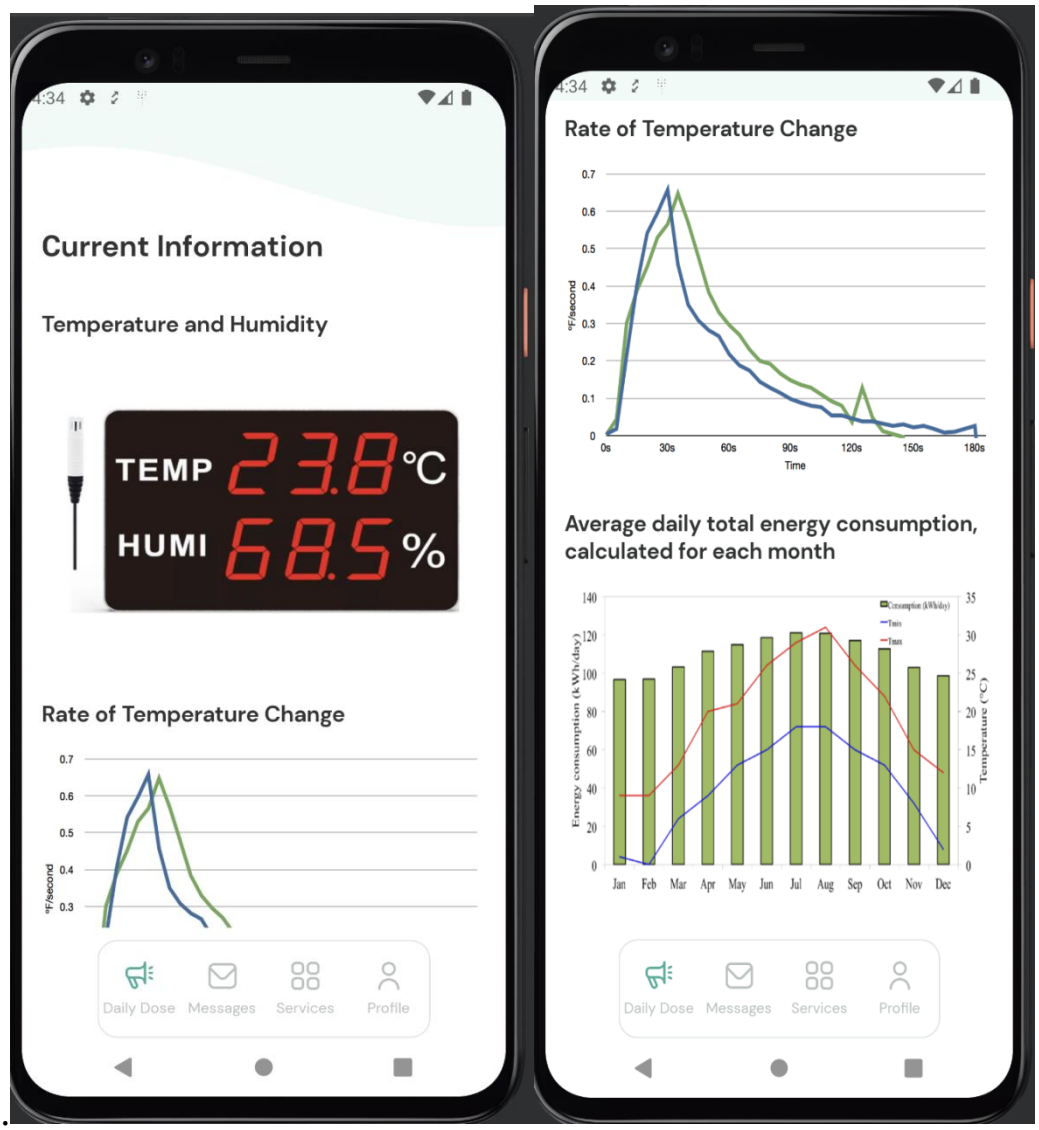

Рисунок 4.7 – Графіки швидкості зміни температури в приміщенні та середньодобове сумарне енергоспоживання

4.6 Report a Problem

Ця функція буде корисною більше для середніх та великих підприємств, де працівник за допомогою цього застосунку зможе доповісти про якусь проблему з пристроями, яку він помітив. Щоб створити звіт про власну проблему, потрібно з головного екрану перейти на кнопку Report an Issue. З’явиться нове вікно, де буде історія всіх звітів користувача та коментарі від сервісного центру (рис. 4.8).

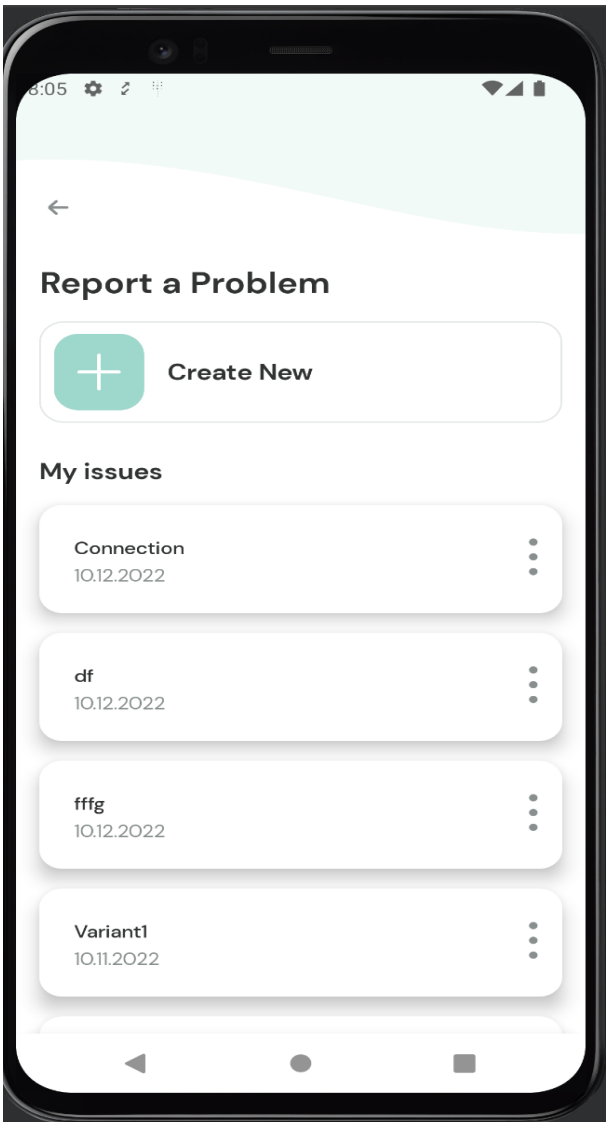

Рисунок 4.8 – Функція Report a Problem

Для створення нового звіту потрібно натиснути на Create New. З’явиться нове вікно, куди буде запропоновано написати інформацію про помилку (рис. 4.9).

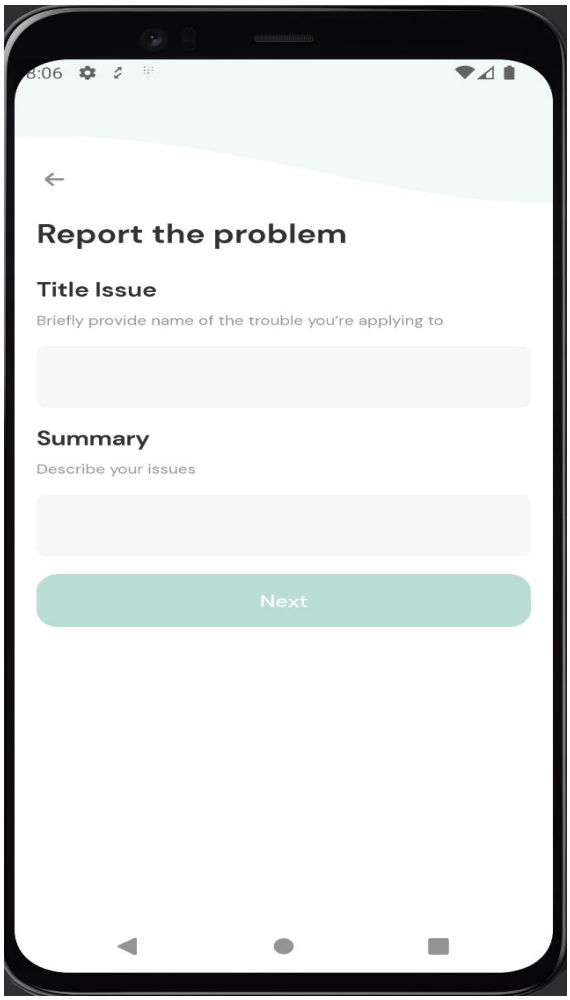

Рисунок 4.9 – Вікно Create New

Заголовок має бути інформативним і містити не більше 60 символів. Якщо два поля заповнено, то після натиску на Next подія буде успішною та буде відправлена на сервер через POST запит, як в прикладі описаному в пункті 4.5, а повний код міститься в тексті (Додаток А).

4.7 Real-time Status

Щоб відкрити екран з моніторингом усіх доступних приладів, які надсилає нам сайт www.scadacore.com для безкоштовного ознайомлення з їх сервісами та можливостями, потрібно з головного екрану натиснути на кнопку Real-time Status. З’явиться нове вікно, де буде перелік пристроїв та значень, які отримуємо кожну хвилину з серверу і представлено на рис. 4.10.

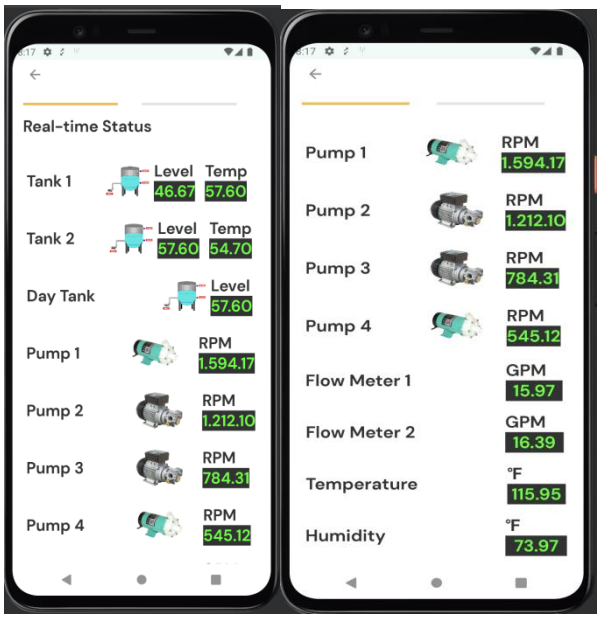

Рисунок 4.10 – Вікно Real-time Status

4.8 Екран All Flows

Щоб відкрити екран зі звітами по існуючим автоматизованим системам, потрібно на головному екрані натиснути на кнопку All Flows. З’явиться нове вікно, де буде перелік систем (flows) та дати формування звіту (рис. 4.11).

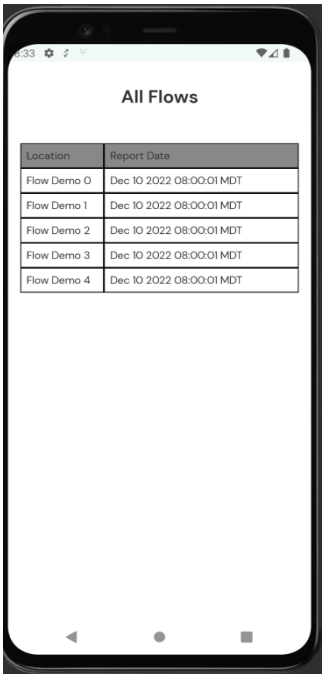

Рисунок 4.11 – Вікно All Flows

Якщо натиснути на один із звітів, то отримаємо інформацію про нього. На рис. 4.12 можна побачити перелік та значення, досліджувані в цій системі.

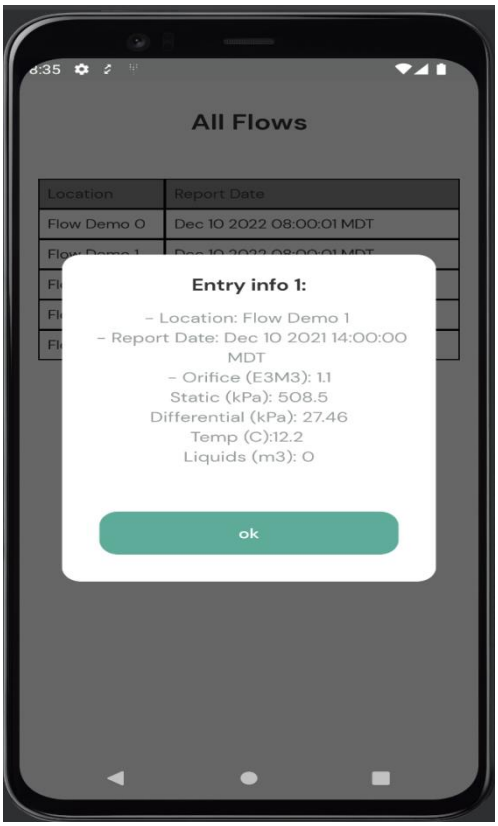

Рисунок 4.12 – Інформація про звіт

4.9 Створення та редагування профілю користувача

Для середніх та великих підприємств буде корисною функція створення та редагування профілю (рис. 4.13). Вона допоможе розрізняти працівників одного підприємства, які хочуть надсилати та отримувати звіти про проблеми, які виникли. Користувач може редагувати свій профіль у будь-який час. Для цього потрібно перейти до профілю і натиснути кнопку edit profile. Після цього буде відкрито вікно з редагуванням. До редагування доступні поля імені, дати народження, статі, аватару, місця роботи та посади.

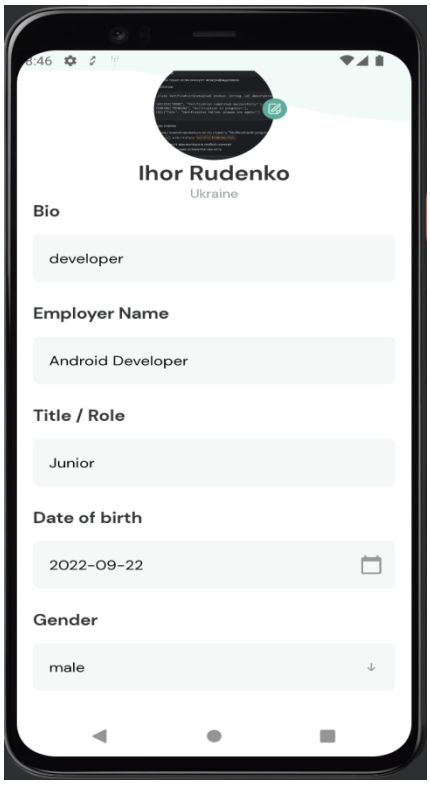

Рисунок 4.13 – Редагування профілю користувача

Щоб уникнути зловживань користувача, здатність редагувати ім’я, стать і вік тощо. доступна лише три рази. Для підтвердження зміни профілю необхідно ввести поточний пароль.

4.10 Розробка локальної бази даних

Після отримання даних із серверу необхідно зберегти їх локально в базі даних, щоб у разі відсутності інтернет-зв’язку застосунок продовжував відображати останні дані, які до нього надійшли.

4.11 Аналіз існуючих інструментів для збереження інформації

4.11.1 SQLite

В Android існує вбудована підтримка бази даних SQLite. Підтримуються всі функції SQLite, надається API оболонки з сумісним інтерфейсом. API AndroidSQLite є типовим; розробнику слід реалізувати всю обробку бази даних, зокрема створення, управління версіями, оновлення бази даних та інші

налаштування. У застосунку при підключенні до бази даних необхідно вказати ім'я бази даних та її версію. При цьому можуть виникнути такі ситуації:

– бази даних не існує – це може бути, наприклад, у випадку первинної установки програми. У цьому випадку застосунок повинен сам створити базу даних і всі таблиці в ній. І далі він вже працює з щойно створеною базою;

–база даних існує, але її версія застаріла. Це може бути при оновленні програми. Наприклад, новій версії програми потрібні додаткові поля в старих таблицях або нові таблиці. У цьому випадку застосунок повинен оновити існуючі таблиці та створити нові, якщо це необхідно;

– база даних існує і її версія актуальна. У цьому випадку застосунок успішно підключається до бази даних і працює.

SQLite передує Android і використовується в інших основних програмах, крім Android. Незважаючи на те, що SQLite сьогодні широко використовується в розробці Android, однак велика кількість розробників не задоволені її складністю. Тому було зроблено багато спроб відмовитися від SQLite.

Акронім SQL у SQLite означає Structured Query Language. Це означає, що SQLite є SQL-сумісним механізмом баз даних, тобто використовується для зберігання структурованих даних, на відміну від SharedPreference, який використовується для зберігання пар даних ключ-значення. Робота з SQLite в Android означає, що потрібно буде використовувати дві мови програмування. Дані, які необхідно зберегти, і логіка, яка генерує дані, написані на об’єктно-орієнтованій мові, такій як Java. Тоді як SQLite розуміє мову SQL.

Нижче наведено три компоненти SQLite, які є центральними для її роботи в Android.

SQLiteOpenHelper – це найважливіший клас, який використовується в Android SQLite. Він використовується для створення та оновлення бази даних SQLite. Іншими словами, SQLiteOpenHelper усуває зусилля, необхідні для встановлення та налаштування бази даних в інших системах.

SQLiteDatabase – це фактична база даних, де зберігаються дані користувачів. Коли користувач створює базу даних за допомогою класу SQLiteOpenHelper, вона

запускає все для створення цієї бази даних, але затримується, доки користувач буде готовий використовувати цю базу даних. SQLiteOpenHelper дізнається, що ви готові використовувати свою базу даних, коли ми отримуємо доступ до цієї бази даних за допомогою getReadableDatabase() або getWritableDatabase() для операцій читання та запису відповідно.

Cursor – це компонент, який зберігає ваші дані в базі даних, щоб мати до них доступ пізніше. Цей доступ називається запитом і успішний запит поверне список елементів, які запитували. Якщо цей список такий довгий, пристрій Android може видавати помилки, якщо необхідно отримати доступ до всіх елементів у результаті, який повертається. Ось тут і з’являється Cursor, список елементів, які були запитані, розміщується в Cursor, а він передає їх нам партіями будь-якої кількості.

4.11.2 Room

У пакеті розробки Jetpack Compose було створено зовнішню обгортку над SQLite, яка має назву Room.

Room надає шар абстракції над SQLite, щоб дозволити вільний доступ до бази даних, використовуючи повну потужність SQLite.

Зараз Room розглядається як кращий підхід для перманентності даних, ніж SQLiteDatabase. Це полегшує роботу з об'єктами SQLiteDatabase у застосуванні, зменшуючи кількість коду boilerplate і верифікації запитів SQL при компіляції.

Плюси користуванням Room. По-перше – це час компіляції верифікації SQL-запитів. Кожен @Query і @Entity перевіряється в момент компіляції, що захищає застосунок від проблем під час виконання. Крім того, він перевіряє не тільки єдиний синтаксис, але і відсутні таблиці. Легко інтегрується з іншими компонентами архітектури (наприклад, LiveData).

Компоненти Room DB. Room DB має три основні компоненти:

– Entity;

– Dao;

– Database.

Взаємодію компонентів Room зображено на рис. 4.14.

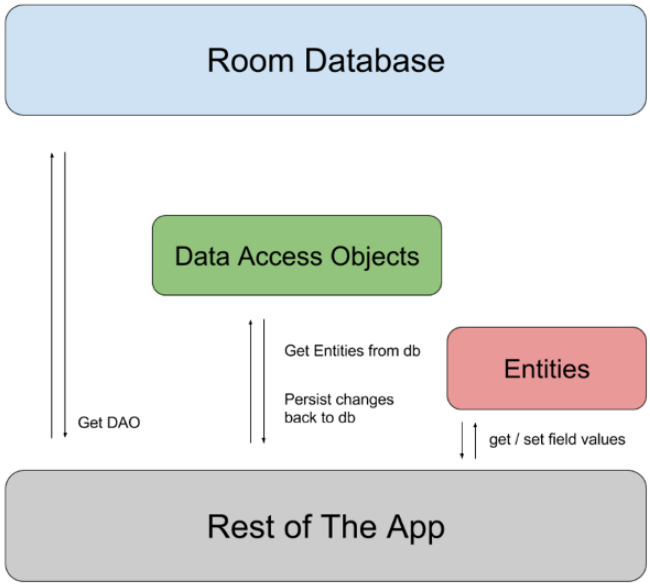

Рисунок 4.14 – Компоненти Room

– Entity є таблицею в базі даних. У Room DB створюється таблиця для кожного класу, що має @Entity анотацію, поля в класі відповідають стовпчикам в таблиці. Тому класи сутності, як правило, є невеликими класами моделей, які не містять жодної логіки;

– Dao відповідають за визначення методів доступу до бази даних. У початковій SQLite використовуються об'єкти Cursor. З Room нам не потрібен весь код, пов'язаний з Cursor, і ми можемо просто визначити наші запити, використовуючи анотації в класі Dao;

– Database містить утримувач бази даних і служить основною точкою доступу для базового з'єднання з реляційними даними вашого застосунку.

4.11.3 Різниця між способами збереження через SQLite та Room

У випадку SQLite відсутня компіляція – час верифікації запитів RAW SQLite. Але в Room під час компіляції відбувається перевірка SQL.

Для перетворення між SQL-запитами і об'єктами даних Java потрібно використовувати багато коду boilerplate. Але Room відображає наші об'єкти бази даних на Java Object без коду boilerplate. У міру зміни схеми потрібно вручну оновлювати постраждалі запити SQL. Room вирішує цю проблему.

Room побудований для роботи з LiveData і RxJava для спостереження за даними, в той час як SQLite – ні.

Для створення бази даних потрібно визначити абстрактний клас, який розширює RoomDatabase. Цей клас анотований за допомогою @Database, перераховує елементи, що містяться в базі даних, і DAOS, які отримують доступ до них. Клас, який анотований за допомогою @Database, повинен відповідати умовам:

– включити список сутностей, пов'язаних з базою даних в анотацію;

– містить абстрактний метод, який має 0 аргументів і повертає клас, що анотований за допомогою @Dao;

– при запуску можна отримати екземпляр бази даних, викликом Room.databaseBuilder() або Room.inMemoryDatabaseBuilder().

4.11.3 Створення основних компонентів Room для класу Profile

Для збереження даних, отриманих із серверу, будемо використовувати локальну базу даних, користуючись Room. Тому нам потрібно реалізувати три основних компонената Room: Entity, Dao, Database.

Під час реєстрації потрібно ввести повне ім’я та email, тому Room компонент Entity повинен містити поля email, fullName, firstName, lastName.

Також потрібно дати назву таблиці в базі даних. Так як за назвою таблиці зможемо отримувати дані про користувача, назвемо її «profile».

Щоб повідомити Room про клас Profile, потрібно додати анотацію Entity до класу і анотацію PrimaryKey до ключа (рис. 4.15).

```
@Entity(tableName = ProfileEntity.TABLE_NAME)
data class ProfileEntity(
    @PrimaryKey(autoGenerate = false) val id: String,
    val email: String,
    val fullName: String,
    val firstName: String,
    val lastName: String,
) {
    companion object {
        const val TABLE_NAME = "profile"
    }
}
```

Рисунок 4.15 – Компонент Room – Entity

Наступним кроком буде написання основних функцій в класі з анотацією Dao. На рис. 4.16 написано класичні методи з анотаціями @Query, @Insert та @Update для отримання та редагування інформації нашого профілю. При кожному вході в застосунок буде викликатися метод getProfile, а при редагуванні профілю ми будемо користуватися методом insertProfile.

Рисунок 4.16 – Компонент Room – Dao з класичними для нього методами

Третім компонентом Room є створення класу DataBase, який зв'язує попередні два компоненти та слідкує за версіями нашої бази даних. У поле entities потрібно написати наш компонент Entity (рис. 4.17). Також в компоненті DataBase прописується головний метод getProfileDao, який зробить вибірку з бази та поверне нам актуальні дані. На рис. 4.17 написаний class AppDataBase.

Рисунок 4.17 – Компонент Room – DataBase

Отже, після реєстрації в застосунку введені дані будуть збережені локально та будуть використовуватися для відображення нашого профілю (див. рис.4.18).

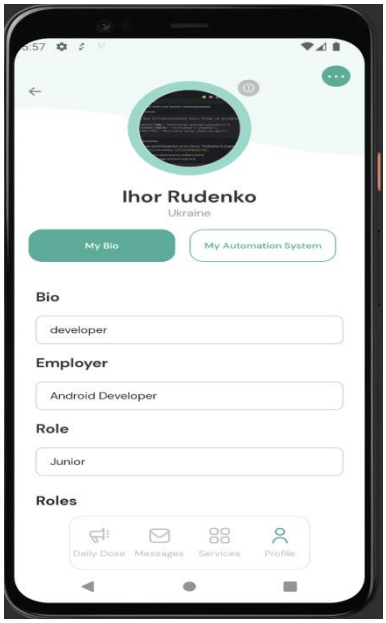

Рисунок 4.18 – Відображення профілю

4.12 Експеримент з відображення отриманих даних у вигляді графіку

Для створення фінальних висновків треба отримати інформацію про те, як система виконує свою основну ціль. У цьому розділі опишемо експеримент.

Основною метою експерименту є виявлення та фіксація властивостей, зв’язків та помилок при виведенні отриманої інформації із серверу на екран користувача у вигляді графіків. Експеримент дає можливість дослідити, по-перше, об’єкти в так званому чистому вигляді; по-друге, в нестандартних умовах, що сприяє більш глибокому проникненню в їхню сутність; по-третє, важливою перевагою експерименту є його повторюваність.

У ході цього експерименту побудуємо графік зміни тиску та температури упродовж 6.11.2022–20.11.2022. Для проведення експерименту нам потрібні вихідні дані із серверу або статично прописані з метою їхнього змінення, це допоможе краще протестувати програмний код. Також потрібно написати програмний код, який може на основі цих даних побудувати графік.

Потрібно зрозуміти, як можна накреслити графік та розробити універсальну функцію. Для цього візьмемо вісь X, яка представляє кількість днів та вісь Y, яка представляє кількість кроків, які відбулися у цей конкретний день. Для прикладу візьмемо 10 днів по осі X і 350 кроків по осі Y з інтервалом кроку 50 (рис. 4.19).

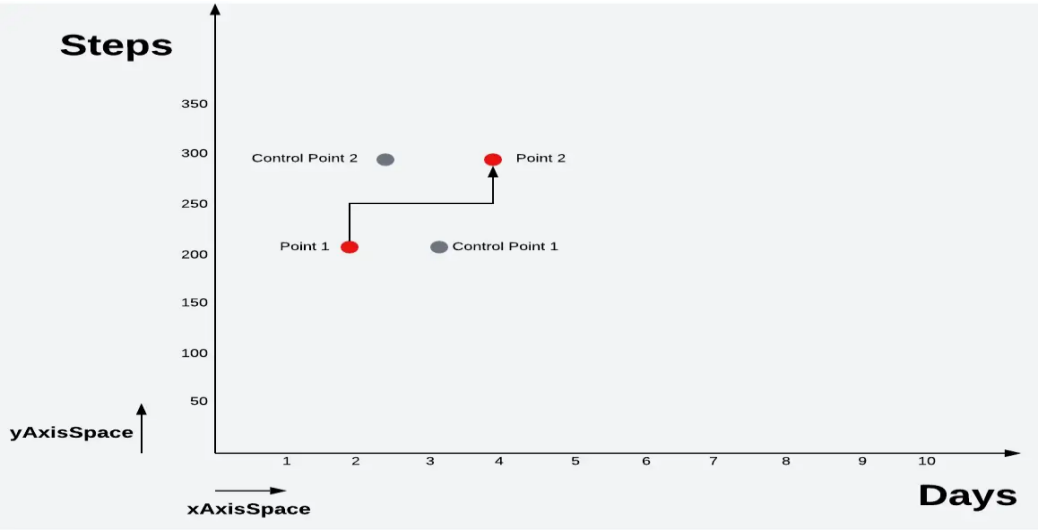

Рисунок 4.19 – Графік залежності days від steps

На рисунку зображено деякі ключові терміни, які допомагають будувати графік:

– xAxisSpace – це мінімальний або рівний інтервал, передбачений для значення на осі X;

– yAxisSpace – це мінімальний або рівний інтервал, передбачений для значення по осі Y;

– Point 1 – це перша початкова точка на графіку;

– Point 2 – це друга точка на графіку.

Control point 1 and 2 – це точки, за допомогою яких відображається крива з точок 1 і 2.

Згадаємо, як працює крива Безьє (рис. 4.20).

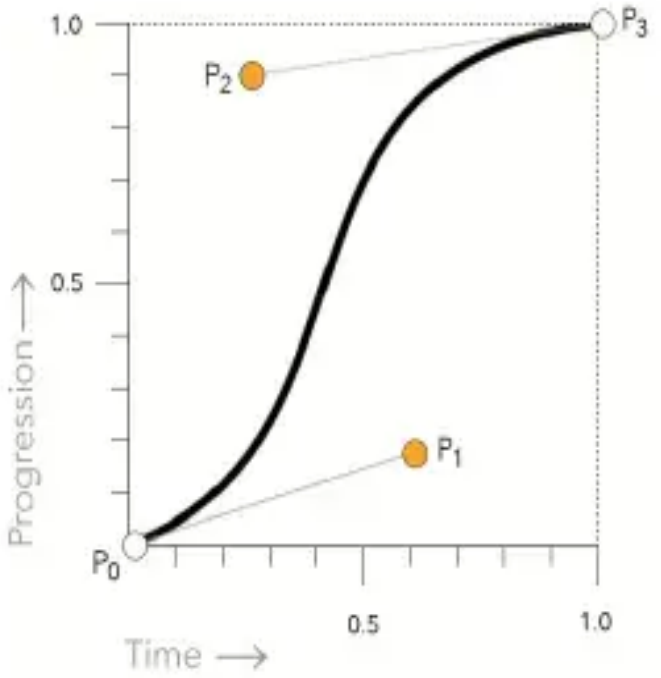

Рисунок 4.20 – Крива Безьє

Крива Безьє має точку початку (P0) і кінця (P3), між якими розташовується крива. Також має дві контрольні точки P1 та P2, які допомагають контролювати форму кривої. Наприклад, якщо ми змінимо обидві контрольні точки для різних значень, то деякі форми можуть мати різний вигляд (рис. 4.21).

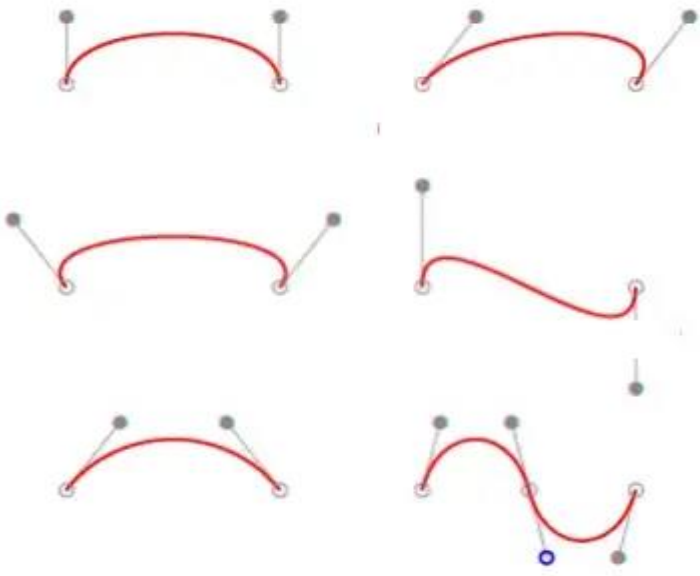

Рисунок 4.21 – Приклад форм Кривої Безьє

Наступним етапом є створення коду для функції графіку з такими параметрами (рис. 4.22).

```
@Composable
fun Graph(
    modifier : Modifier,
    xValues: List<Int>,
    yValues: List<Int>,
    points: List<Float>,
    paddingSpace: Dp,
    verticalStep: Int
) {}
```

Рисунок 4.22 – Код функції графіку

Важливими параметрами функції будуть xValues – це інтервали по осі X, yValues – інтервали по осі Y, points – це точки на графіку. Для побудови графіку потрібно розрахувати відстань між точками на осі X та Y, відняти розрахунок розміру, який ми надаємо нашій функції та поділити на загальні точки на осі. Зобразимо також маленькі кола на точках (координатах), щоб можна було побачити, де знаходяться точки. Повний код зображено на рис. 4.23.

```
val xAxisSpace = (size.width - paddingSpace.toPx()) / xValues.size
val yAxisSpace = size.height / yValues.size
/** placing x axis points */
for (i in xValues.indices) {
    drawContext.canvas.nativeCanvas.drawText(
        "${xValues[i]}",
        xAxisSpace * (i + 1),
        size.height - 30,
        textPaint
    )
}
 /** placing y axis points */
for (i in yValues.indices) {
    drawContext.canvas.nativeCanvas.drawText(
        "${yValues[i]}",
        paddingSpace.toPx() / 2f,
        size.height - yAxisSpace * (i + 1),
        textPaint
    )
}
/** placing points */
for (i in points.indices) {
    val x1 = xAxisSpace * xValues[i]
    val y1 = size.height - (yAxisSpace * (points[i]/verticalStep.toFloat()))
    coordinates.add(PointF(x1,y1))
    /** drawing circles to indicate all the points */
    drawCircle(
        color = Color.Red,
        radius = 10f,
        center = Offset(x1,y1)
    )
}
```

Рисунок 4.23 – Код роботи функції, яка будує графік

Після запуску в Android Studio цього коду отримуємо результат, зображений на рис. 4.24.

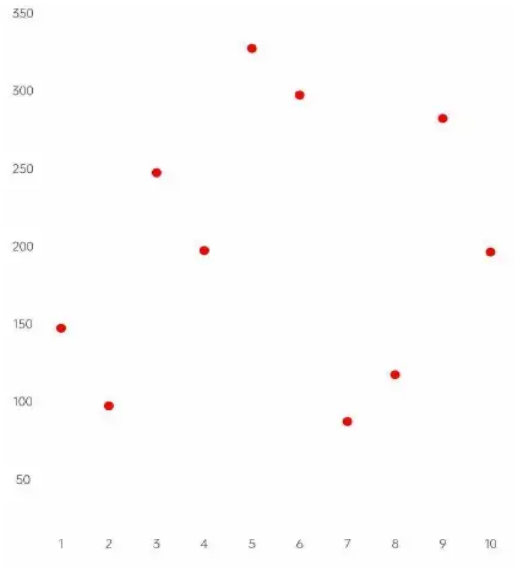

Рисунок 4.24 – Точки на графіку

Щоб приєднатися до цих точок, використовуючи криву Безьє, потрібно визначити контрольні точки для кожної пари точок, починаючи з нуля (рис. 4.25). Коли є координати і контрольні точки для всіх пар, можемо об'єднати координати. Спочатку переходимо до першої координати і починаємо приєднуватися до інших.

```
/** drawing the path */
val stroke = Path().apply {
  reset()
  moveTo(coordinates.first().x, coordinates.first().y)
  for (i in 0 until coordinates.size - 1) {
      cubicTo(
        controlPoints1[i].x,controlPoints1[i].y,
        controlPoints2[i].x,controlPoints2[i].y,
        coordinates[i + 1].x,coordinates[i + 1].y
      )
  }
}

drawPath(
  stroke,
  color = Color.Black,
  style = Stroke(
      width = 5f,
      cap = StrokeCap.Round
  )
)
```

Рисунок 4.25 – Код з’єднання всіх точок кривою Безьє

Запустивши цю функцію, отримаємо результат, що зображений на рис. 4.26.

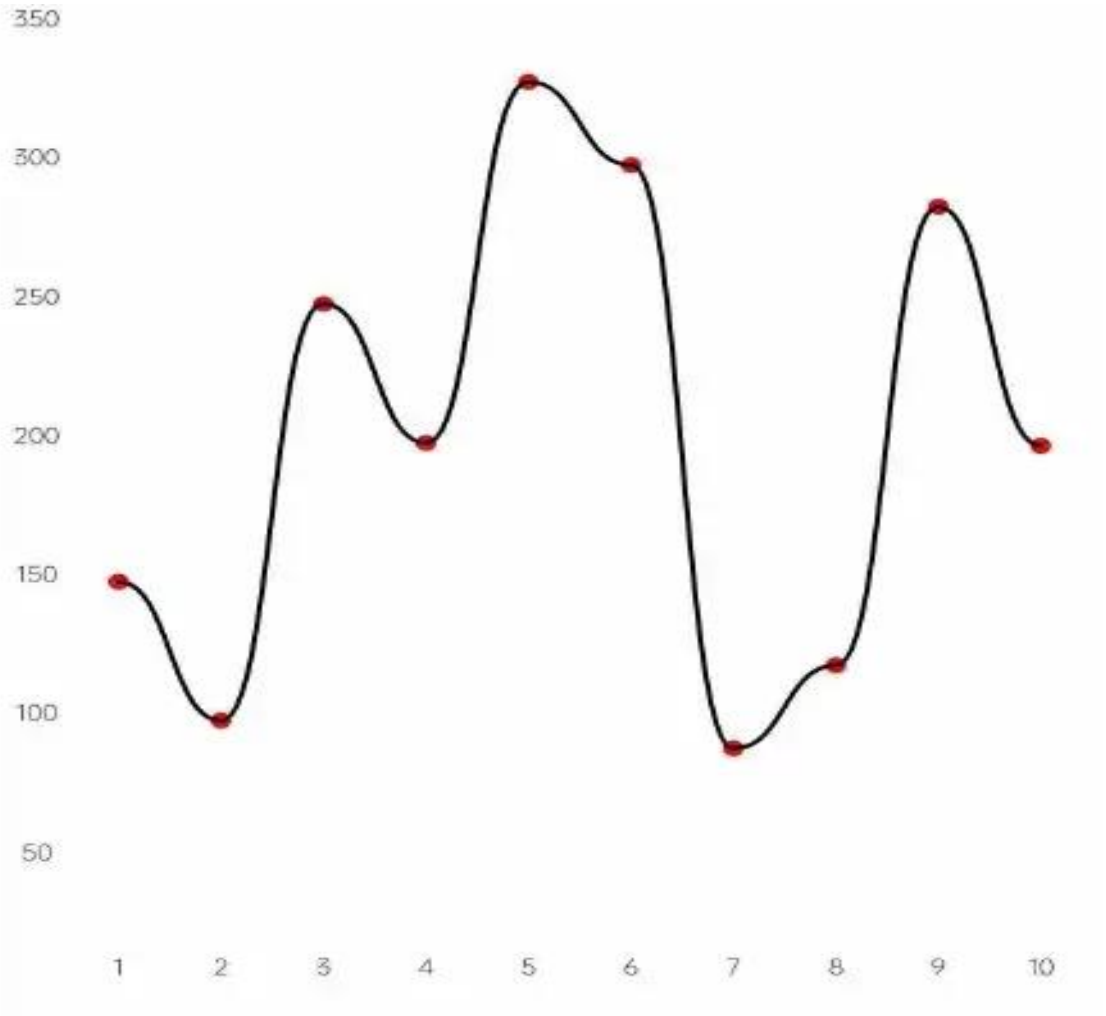

Рисунок 4.26 – Результат роботи функції створення графіку

Залишилось завершити дизайн відображення функції створення графіків та програмну логіку його візуальної зміни (Додаток А).

Проведемо експеримент, змінюючи вхідні дані функції. В Android Studio створимо 2 класси ChartTemp() та ChartPres. У першому зберігатиметься значення дати та температури, а в другому – дати та тиску. Вхідні дані для графіку зміни температури зображені на рис. 4.27.

```
val dataTemp = listOf(
    Pair("6.11", 30.45),
    Pair("7.11", 31.0),
    Pair("8.11", 31.45),
    Pair("9.11", 35.25),
    Pair("10.11", 36.45),
    Pair("11.11", 34.35),
    Pair("12.11", 33.65),
    Pair("13.11", 38.15),
    Pair("14.11", 35.05),
    Pair("15.11", 35.25),
    Pair("16.11", 37.35),
    Pair("17.11", 34.45),
    Pair("18.11", 34.65),
    Pair("19.11", 33.45),
    Pair("20.11", 31.85)
)
```

Рисунок 4.27 – Вхідні дані з показниками температури по днях

Вхідні дані для графіку зміни тиску зображені на рис. 4.28.

```
val dataPres = listOf(
    Pair("6.11", 111.45),
    Pair("7.11", 111.0),
    Pair("8.11", 113.45),
    Pair("9.11", 112.25),
    Pair("10.11", 116.45),
    Pair("11.11", 113.35),
    Pair("12.11", 118.65),
    Pair("13.11", 110.15),
    Pair("14.11", 113.05),
    Pair("15.11", 114.25),
    Pair("16.11", 116.35),
    Pair("17.11", 117.45),
    Pair("18.11", 112.65),
    Pair("19.11", 115.45),
    Pair("20.11", 111.85)
)
```

Рисунок 4.28 – Вхідні дані з показниками тиску по днях

Перша колонка буде використовуватися для побудови осі X, а друга колонка-для осі Y. Маючи вхідні дані,зможемо запустити функцію в мобільному застосунку. Результат зображений на рис. 4.29.

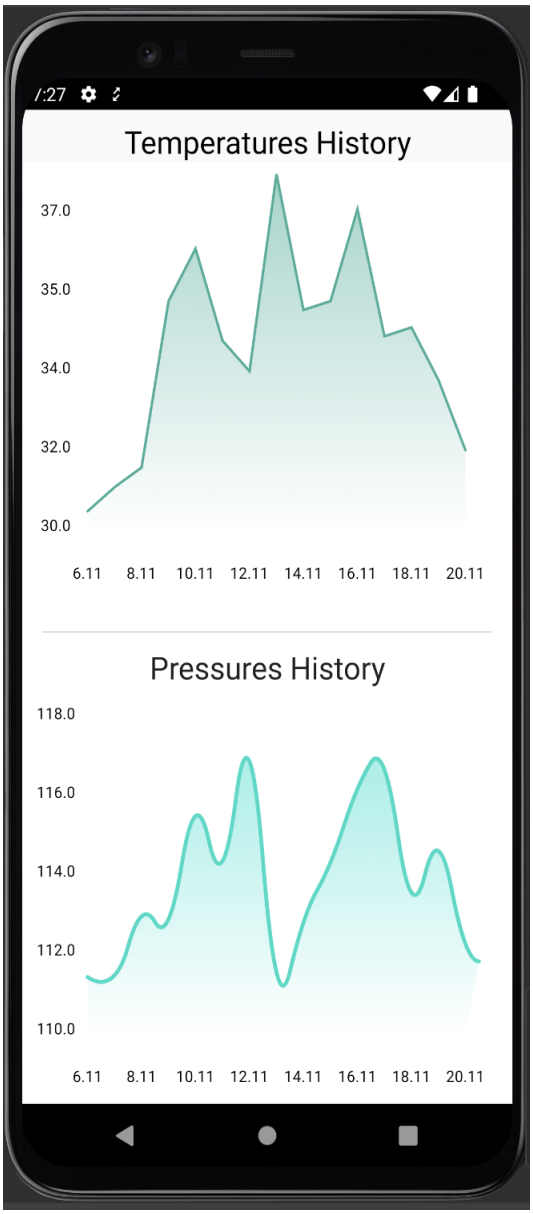

Рисунок 4.29 – Моніторинг температури та тиску на екрані

Лінії та точки по осі X та Y точно відповідають вхідним даним. Для завершального етапу експерименту потрібно змінити вхідні дані. Підставимо новий клас з різким зменшенням або збільшенням значень температури та тиску.

Змінені вхідні для графіку зміни тиску на рис. 4.30.

```
val data = listOf(
    Pair("6.11", 118.0),
    Pair("7.11", 111.0),
    Pair("8.11", 110.0),
    Pair("9.11", 112.25),
    Pair("10.11", 116.45),
    Pair("11.11", 113.35),
    Pair("12.11", 118.65),
    Pair("13.11", 110.15),
    Pair("14.11", 113.05),
    Pair("15.11", 114.25),
    Pair("16.11", 116.35),
    Pair("17.11", 117.45),
    Pair("18.11", 112.65),
    Pair("19.11", 110.0),
    Pair("20.11", 110.0)
)
```

Рисунок 4.30 – Вхідні дані з показниками тиску по днях

Змінені вхідні для графіку зміни температури зображені на рис. 4.31.

```
val data2 = listOf(
    Pair("6.11", 30.0),
    Pair("7.11", 30.0),
    Pair("8.11", 30.0),
    Pair("9.11", 30.0),
    Pair("10.11", 33.0),
    Pair("11.11", 34.5),
    Pair("12.11", 35.0),
    Pair("13.11", 36.15),
    Pair("14.11", 37.0),
    Pair("15.11", 35.25),
    Pair("16.11", 37.35),
    Pair("17.11", 34.45),
    Pair("18.11", 34.65),
    Pair("19.11", 30.0),
    Pair("20.11", 30.0)
```

Рисунок 4.31 – Вхідні дані з показниками температури по днях

Після запуску створеного застосунку з оновленими вхідними даними отримаємо результат, що зображено на рис. 4.32.

Рисунок 4.32 – Графіки зміни температури та тиску

Аналізуючи вхідні дані за графіком температури, можна побачити, що 6.11, 8.11 та 20.11 температура була на рівні 30 градусів. Це відповідає результату на рис. 4.32.

Отже, експеримент пройшов успішно. Розроблена функція побудови графіків на основі вхідних даних працює задовільно, тому будемо її використовувати в нашому застосунку.

4.13 Забезпечення безпеки умов праці при функціонуванні моніторингу автоматизованих систем

При проведенні моніторингу автоматизованих систем, які в свою чергу використовують комп’ютери та мережеве обладнання, зокрема: маршрутизатори, комутатори, повторювачі, сервери, необхідно дотримуватися техніки безпеки, правил охорони праці та протипожежної безпеки. Відповідно до закону України, нормативно-правові акти з охорони праці є обов'язковими для виконання у виробничих майстернях, лабораторіях, цехах, на дільницях та в інших місцях трудового і професійного навчання, облаштованих у будь-яких навчальних закладах [29].

Усі пристрої мережевої взаємодії потребують живлення від електромереж, при цьому електромагнітні поля, які створюються цими пристроями, особливо негативно впливають на організм людини, яка безпосередньо працює з джерелом випромінювання. При дії випромінювання на людину можливі гострі та хронічні форми порушення фізіологічних функцій організму. Такі порушення виникають в результаті дії електричної складової випромінювання на нервову систему, а також на структуру кори головного та спинного мозку, серцево-судинної системи. Приміщення, в яких планується установка та подальша робота з комп’ютеризованими приладами, за допомогою яких буде будуватися віртуальний макет комп’ютерної мережі, повинні відповідати проектній документації будинку, погоджений з уповноваженими державними органами.

Крім того, повинні бути враховані санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення. Конкретні показники зазначених санітарних норм знаходяться в Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. Відповідне приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря. При установці зазначених систем,

необхідно переконатись, що батареї опалення, водопровідні труби, вентиляційні кабелі тощо, надійно сховані під захисними щитками, які перешкоджатимуть можливому потраплянню робітника під напругу [31].

У процесі роботи з модельованою комп'ютерною мережею, необхідно дотримувати правильний режим праці та відпочинку. В іншому випадку у робочого персоналу значно збільшується напруга зорового апарату, з'являються скарги на незадоволеність роботою, головні болі, дратівливість, порушення сну, втома і хворобливі відчуття в очах, попереку, в області шиї і руках. Робочі місця слід розташовувати так, щоб уникнути попадання в очі прямого світла. Джерела освітлення рекомендується розташовувати з обох боків екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану, клавіатури в напрямку очей користувача, від світильників загального освітлення або сонячних променів, необхідно використовувати антиполискові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах – жалюзі. Екран дисплея повинен бути розташованим перпендикулярно до напрямку погляду. Якщо він розташований під кутом, то стає причиною сутулості. Відстань від дисплея до очей повинна трохи перевищувати звичну відстань між книгою та очима. Монітор повинен бути розташований на робочому місці так, щоб поверхня екрана знаходилась в центрі поля зору на відстані 400-700 мм від очей користувача.

Залежно від джерела світла виробниче освітлення може бути: природним, що створюється прямими сонячними променями та розсіяним світлом небосхилу; штучним, що створюється електричними джерелами світла, та суміщеним, при якому недостатнє за нормами природне освітлення доповнюється штучним. Природне освітлення поділяється на: бокове (одно- або двостороннє), що здійснюється через світлові отвори (вікна) в зовнішніх стінах; верхнє – через ліхтарі та отвори в дахах і перекриттях; комбіноване – поєднання верхнього та бокового освітлення. Штучне освітлення може бути загальним та комбінованим [30].

Крім вищезазначеного, важливою складовою охорони праці є пожежна безпека приміщень. Основними напрямками забезпечення пожежної безпеки є

усунення умов виникнення пожежі та мінімізація її наслідків. Об'єкти повинні мати системи пожежної безпеки, спрямовані на запобігання пожежі, дії на людей та матеріальні цінності небезпечних факторів пожежі, в тому числі їх вторинних проявів. До таких факторів, згідно з ГОСТ 12.1.004-91, належать:

– підвищена температура навколишнього середовища;
– полум'я та іскри;
– знижена концентрація кисню;
– токсичні продукти горіння й термічного розкладу матеріалів і речовин дим.

Відповідно до ГОСТ 12.1.004-91, пожежна безпека будь-якого об'єкта повинна забезпечуватися системою протипожежного захисту, системою запобігання пожежі і системою організаційно-технічних заходів [32]. Основними вихідними даними при розробці комплексу організаційних і технічних рішень щодо забезпечення потрібного рівня пожежної безпеки в кожному конкретному випадку є чинна законодавча і нормативно-технічна база з питань пожежної безпеки, властивості матеріалів і речовин, що застосовуються у виробничому циклі, матеріалів, речовин і особливості виробництва. Під час виконання роботи, був проведений аналіз діючих законодавчих актів та норм щодо облаштування робочих приміщень.

4.14 Висновки до четвертого розділу

У четвертому розділі детально описано процес роботи програми мобільного застосунку, його функціонал. Проаналізовано існуючі інструменти для збереження інформації. Описано експеримент , який довів задовільну роботу застосунку.

У цьому розділі було проведено аналіз діючих законодавчих актів та норм щодо облаштування робочих приміщень, було проаналізовано забезпечення нівелювання негативного впливу електромагнітного впливу на людину та апаратуру.

ВИСНОВКИ

Ефективність функціонування промислового підприємства безпосередньо залежить від технічного оснащення виробництва та від ступеня виробничої автоматизації. Необхідно регулярно здійснювати моніторинг автоматизованих систем. Впровадження електронних систем моніторингу та управління значно спрощує роботу персоналу, дозволяє ефективно використовувати ресурси, швидко виявляти та вирішувати проблеми.

Підприємство є складною системою, і тому його можна представити як єдину систему сфер діяльності. Отже, моніторинг та управління має відбуватися цілком у всіх напрямках (виробництво, кадри, реалізація продукції, фінанси). На рівні окремого підприємства ефективний розвиток передбачає впровадження прибуткових нововведень, які допомагають підприємству залишатися конкурентоспроможним на ринку.

Одним із факторів, що сприяють конкурентоспроможності великих підприємств, є використання автоматизованих систем моніторингу та управління. Такі системи передбачають глибоке опрацювання даних, спеціально перетворених для зручного використання у процесі прийняття рішень. Моніторинг процесів промислових підприємств передбачає покращення технологій накопичення та зберігання даних через інтеграцію сховищ даних та інтелектуального аналізу даних. Завдяки цьому усуваються проблеми недостовірності даних, низької продуктивності при аналітичних запитах, неможливість перетворення різнорідних даних на єдину інформацію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з підготовки й оформлення кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / / упоряд. : І. Ш. Невлюдов, Р. В. Артюх, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків, ХНУРЕ, 2021. 50 с.

2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, 2016. 31 с.

3. Бронніков А. І., Руденко І. В. Система моніторингу автоматизованих систем на підприємстві. Виробництво & Мехатронні Системи 2022: матеріали Міжнародної VI конференції : Міжнар. конф., м. Харків, 21 жовт. 2022 р. 2022. С. 71.

4. Руденко І. В. Основні напрямки застосування автоматизованих систем управління технологічними процесами. «Автоматизація та приладобудування» ADED-2022 : Зб. студент. наук. ст. С. 117–121.

5. Основні напрямки застосування автоматизованих систем управління технологічними процесами. «Автоматизація та приладобудування» ADED-2022 : Зб. студент. наук. ст. С. 117–121.

6. Автоматизована система управління. Лексика – українські енциклопедії та словники. URL: https://leksika.com.ua/16010716/ure/avtomatizovana_sistema_upravlinnya (дата звернення: 02.10.2022).

7. Автоматизована система. URL: http://www.wiki.uk-ua.nina.az/Автоматизована_система.html (дата звернення: 02.10.2022).

8. Поняття та призначення АРМ. *StudFiles*. URL: https://studfile.net/preview/9998000/page:2/ (дата звернення: 16.12.2022).

9. Барало О. В., Гранат С. Є., Ковальов В. О., Самойленко П. Г. Автоматизація технологічних процесів і систем автоматичного керування. К. : Аграрна освіта, 2018. 557 с.

10. Ельперін І. В. Автоматизація виробничих процесів. Ліра-К, 2017. 378 с.

11. Ata A. A. Autonomous mobile robot for mine detection. Alexandria University, 2017. P. 607–608.

12. How to Create – 7 Steps to Make an app (2022). GoodBarber. URL: https://www.goodbarber.com/blog/how-to-make-an-app/ (дата звернення: 15.12.2022).

13. Liliane Lona. A Step by Step Approach to the Modelling of Chemical Engineering Processes. School of Chemical Engineering University of Campinas, 2019. 182 p.

14. Сенченко П. В. Моніторинг соціально-економічного розвитку територій в контексті інформаційного забезпечення системи управління за результатами. 2018. С. 293–299.

15. Фролов А. І. Адміністративний моніторинг як елемент процесу організації управління в організаційно-технічних системах. Управління розвитком великомасштабних систем (MLSD'2011) : матеріали п’ятої міжнародної конференції, 2017. Т. 2. С. 383–386.

16. Іванишин Т. В, Мазепа С. С. Основи автоматики та автоматизація виробничих процесів лісових і деревообробних підприємств. Магнолія, 2019. 354 с.

17. Васильківський І. С, Федінець В. О, Юсик Я. П. Виконавчі пристрої систем автоматизації. Львів. політехніка, 2020. 220 с.

18. Automation System: What Is It? How Is It Used? Advantages. *OEM Manufacturers | OEM Manufacturing Companies | IQS Directory*. URL: https://www.iqsdirectory.com/articles/automation-equipment/automation-system.html (дата звернення: 21.11.2022).

19. What is SCADA? Supervisory Control and Data Acquisition | COPA-DATA. Automatisierungs- & Industriesoftware | zenon COPA-DATA. URL: https://www.copadata.com/en/product/zenon-software-platform-for-industrial-

automation-energy-automation/visualization-control/what-is-scada/ (дата звернення: 21.11.2022).

20. Learn all about SCADA systems: What is SCADA? | SCADApedia. SCADA International. URL: https://scada-international.com/what-is-scada/ (дата звернення: 21.11.2022).

21. Parves Kamal. Identifying and Scoring Vulnerability in SCADA Environments [Текст] / Parves Kamal, Abdullah Abuhussein – Ванкувер, 2017. 17 с.

22. SIMATIC SCADA Systems. siemens.com Global Website. URL: https://new.siemens.com/global/en/products/automation/industry-software/automation-software/scada.html (дата звернення: 21.11.2022).

23. Top 10 Features of the Best SCADA Systems | ICONICS Software Solutions. ICONICS Automation Software Solutions | ICONICS Software Solutions. URL: https://iconics.com/Resources/ICONICS-Blog/Top-10-Features-of-the-Best-SCADA-Systems (дата звернення: 21.11.2022).

24. Global Commerce Review, Interactive World Map | Criteo. Criteo. URL: https://bit.ly/2AkZ5Re (дата звернення: 14.12.2022).

25. Annual number of mobile app downloads worldwide 2021 | Statista. Statista. URL: https://bit.ly/2zTtudv(дата звернення: 14.12.2022).

26. Average Time Spent per Day with Mobile Internet Among US Adults, In– App vs. Mobile Web, 2015–2019 URL: bit.ly/(дата звернення: 14.12.2022).

27. Mobile App Cost Calculator. URL: http://bit.ly/2AgRNlB(дата звернення: 14.12.2022)

28. Mobile Operating System Market Share Ukraine [Електронний ресурс] – Режим доступу: bit.ly/2HyPdKr – 29.08.2019р. –Загл. з екрана.

29. Наказ Міністерства соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» № 207, 2018. URL: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 07.11.2022).

30. Закон України «Про охорону праці» № 49. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 07.11.2022).

31. Наказ Міністерства внутрішніх справ України «Про затвердження Правил пожежної безпеки в Україні» № 1417. URL: https://zakon.rada.gov.ua/laws/show/z0252-15 (дата звернення: 09.11.2022).