МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ
д-р техн. наук, проф.
_____ І. М. Журавська
« __ » _____ 2023 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

МОНІТОРИНГОВА IOT-МЕРЕЖА НА БАЗІ LORAWAN ТА MQTT

Спеціальність 123 Комп'ютерна інженерія
123 – КМР.1 – 605м.21710533

Студент _____А. Г. Шевченко
підпис

«__»_____2023 р.

Керівник канд. фіз.–мат. наук, доцент _____С. В. Пузирьов
підпис
«__»_____2023 р.

Миколаїв – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

BIM – Building Information Modeling

BSN – Body Sensor Network

COVID-19 – Coronavirus Disease 2019

CoAP – Constrained Application Protocol

GPS – Global Positioning System

HTTPS – Hypertext Transfer Protocol Secure

IoT – Internet of Things

LoRaWAN – Long Range Wide Area Network

MQTT – Message Queuing Telemetry Transport

NCISP – National Criminal Intelligence Sharing Plan

RFID – Radio-Frequency Identification

ДТП – Дорожньо-транспортна пригода

ЦАД – Центр аналітичних досліджень

ШІ – Штучний інтелект

ВСТУП

Актуальність кваліфікаційної роботи: Останнім часом людство зіткнулося з великою кількістю викликів, які руйнують буденний темп життя та наражають людей на небезпеку. Одним із таких викликів сьогодення є епідемія COVID-19, яка спричинила локдауни, що викликало проблеми економічного та психологічного характеру в суспільстві. За сучасними прогнозами це не єдиний та не останній випадок в історії людства. Тому на сьогодні досі залишається актуальним питання запобігання розповсюдження хвороб, не змінюючи суспільний ритм життя та не вдаючись до карантинних обмежень.

Мета кваліфікаційної роботи: Розробити концептуальну модель системи моніторингу географічних даних скупчень людей та загальних параметрів їх здоров'я (з дотриманням приватності), а також реалізувати її на базі модулів LoRaWAN та протоколу MQTT задля запобігання утворень осередків розповсюдження епідемії шляхом оповіщень та надання інформації людям, щодо небезпечних зон.

Об'єкт кваліфікаційної роботи: Процес збору, обробки, зберігання та надання в узагальненому та анонімізованому вигляді даних суспільства про пересування та стан здоров'я скупчень людей задля виявлення потенційних зон осередків розповсюдження хвороб.

Предмет кваліфікаційної роботи: Методи побудови моніторингових мереж на основі модулів LoRaWAN, збір даних на базі протоколу MQTT; обробка, фільтрація, анонімізація даних за допомогою мови програмування Rust.

Завдання кваліфікаційної роботи:

– Проаналізувати сучасні моніторингові системи та аналогічні рішення, визначити недоліки та переваги.

– Розробити концептуальну модель моніторингової мережі, виходячи з ідеї оптимальності рішення, та покращень в порівняні з аналогами.

– Побудувати математичну модель обробки даних. Обрати алгоритми кластеризації та фільтрації даних.

– Проаналізувати та обрати оптимальні за ціною та якістю компоненти.

– Спроєктувати апаратну частину системи.

– Спроєктувати програмну частину системи.

– Визначити шляхи оптимізації та покращення отриманого рішення, подальший напрямок розвитку.

Методи дослідження кваліфікаційної роботи: історичний метод (розвиток систем збору даних), функціональний метод (системи моніторингу), системний метод (IoT мережі збору даних LoRaWAN), метод формалізації (аналіз даних), метод математичного моделювання (кластеризація та фільтрація даних), прототипування (створення прототипу системи), метод порівняльного аналізу (порівняння наведеного рішення з аналогами).

Гіпотеза кваліфікаційної роботи: дослідження полягає у виявленні переваг моніторингової системи з використанням технологій LoRaWAN та MQTT в порівнянні з існуючими технологіями за параметрами швидкодії, безпеки, приватності та ціни.

Наукова новизна кваліфікаційної роботи полягає у створені сучасного рішення виявлення та попередження утворень осередків розповсюдження хвороб у громадських місцях на основі моніторингу даних та оповіщень за допомогою мережі LoRaWAN та протоколу MQTT.

Практичне значення кваліфікаційної роботи: можливість впровадження в міську систему або в систему розумного міста технології виявлення та попередження осередків виникнення хвороб, наведеної у даній роботі, з метою зменшення випадків захворюваності та виникнення епідемій, покращення якості охорони здоров'я та життя суспільства.

Робота пройшла апробацію під час XXV Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 07–11 листопада 2022 р.).

Публікації. Основні положення та результати магістерської роботи опубліковані у збірнику матеріалів XXV Всеукраїнської науково-практичної конференції «Могилянські читання–2022» [42].

1 АНАЛІТИЧНА ЧАСТИНА. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ СИСТЕМИ. ФОРМУВАННЯ ВИМОГ ДО АПАРАТНО-ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальний огляд технічних рішень

В даному розділі проводиться загальний огляд сучасних технологічних рішень, які подібні або мають спільні риси з темою дипломної роботи, тобто моніторингової мережі для збору даних контактів людей на відкритих ділянках місцевості або у приміщеннях для подальшого виявлення небезпеки розповсюдження хвороб та попереднього оповіщення.

1.1.1 Contact tracing apps на основі IoT

Цифрове відстеження контактів - це система, яка дозволяє смартфонам, на яких запущено один і той самий додаток для відстеження контактів, реєструвати близькі контакти з іншими пристроями. У цьому розділі розглядається архітектура платформи для цифрового відстеження контактів. Крім того, в цьому розділі також розглядаються рішення для цифрового відстеження контактів на основі Інтернету речей.

Система обчислювальних пристроїв, відома як Інтернет речей (IoT), здатна передавати дані через мережу без необхідності взаємодії людини з людиною або людини з комп'ютером. Стає все важче покладатися виключно на відстеження людських контактів, оскільки кількість інфікованих людей в умовах нинішньої епідемії зростає щодня [1].

Масштабована, автоматизована система відстеження контактів, яка може впоратися з постійно зростаючим навантаженням відстеження контактів, може стати можливою завдяки цифровому відстеженню контактів на основі IoT. Для допомоги в процесі відстеження контактів використовується метод під назвою "цифрове відстеження контактів", який знаходить контакти, коли людина знаходиться поруч з інфікованою особою. Для цього може використовуватися або GPS-локація, або сигнал Bluetooth. Ймовірність зараження вірусом залежить від

того, на якій відстані (скажімо, менше 1 м) і як довго людина перебуває в контакті з хворим. Однак, через бар'єри між двома людьми, такі як стіна або використання засобів індивідуального захисту, безконтактне відстеження не може запропонувати всебічну оцінку при відстеженні контактів з інфікованою особою.

Численні системи відстеження контактів були розроблені відповідно до національних та міжнародних законів про безпеку, конфіденційність та сумісність. Більшість цих рішень покладаються на датчики для визначення місце‑ знаходження користувачів або їхніх близьких контактів з метою оцінки того, як поширюється хвороба (рішення для відстеження). Ці програми попереджають або попереджають користувачів про небезпеку передачі вірусу, запитуючи певні дозволи або сенсорні дані.

Завдяки співпраці з користувачами такі платформи відстеження, як Google COVID-19 Community Mobility Reports та Apple COVID-19 Mobility Trends Reports, збирають дані від мільйонів користувачів та анонімізують їх. Це робиться для моніторингу пересування натовпів та окремих осіб, виявлення гарячих точок, про які ходять чутки, і, що найважливіше, для моніторингу успішності локдаунів у різних місцевостях та країнах. Ці додатки для відстеження контактів призначені для здорових людей, яким цікаво дізнатися про свій ризик заразитися хворобою, якщо вони контактують з кимось, у кого згодом буде виявлено COVID-19. За допомогою цих цифрових додатків для відстеження контактів та прогнозування транспортних даних можна управляти потоком пасажирів у залах очікування [2].

Крім того, були розроблені більш жорсткі підходи до геозонування людей на карантині з використанням спеціалізованих додатків для смартфонів або більш прямої взаємодії з користувачем за допомогою телефонного дзвінка або текстового повідомлення, яке розкриває місцезнаходження користувача. Переміщення та контакти предметів реєструються за допомогою RFID-міток в апаратному рішенні IoT [3]. Таке впровадження є скоріше обов'язковим, ніж добровільним, і відмова від нього є неправильною. Платформа під назвою BubbleBox нещодавно була представлена, де набір інтегрованих IoT-пристроїв та

програмної платформи використовується для контролю та виявлення додаткових спалахів інфекції COVID-19. Браслет від BubbleBox відстежує контакти, зберігаючи безпечну соціальну дистанцію. Користувачі веб-додатку можуть реєструвати свої симптоми та співставляти їх ідентифікацію зі своїм гаджетом. Таким чином, вони надають уповноваженим медичним працівникам швидкий спосіб зрозуміти поширення вірусу, відстежити, хто має пройти тестування, та оперативно зв'язатися з хворими. Нарешті, зібрані анонімні дані можуть допомогти дослідникам виявити закономірності передачі інфекції. Для того, щоб максимізувати охоплення відстеженням контактів та збільшити кількість користувачів системи, важливе значення також має зручність використання. Нарешті, оскільки численні програми та системи відстеження контактів набувають популярності, такі системи будуть успішними лише тоді, коли вони будуть функціонально сумісними. Як наслідок, необхідно встановити стандарт для зібраних даних. IoT - це інноваційний пристрій, який може допомогти у відстеженні всіх пацієнтів з групи ризику під час карантину.

Переваги викроситання Digital contact tracing apps на основі IoT:

- зменшення кількість помилок;
- покращені методики лікування;
- ефективна діагностика;
- розширена діагностика;
- менші витрати.

Після спалаху COVID-19 цифрові програми відстеження контактів набули широкого міжнародного використання. Цифрове відстеження контактів є більш масштабованим, ніж ручне відстеження контактів, і має можливість виявляти контакти, які неможливо знайти за допомогою ручних методів, наприклад, випадкові зустрічі з незнайомцями в кав'ярні або в системі громадського транспорту. У цьому документі ми представляємо ретельний аналіз програм і методів відстеження контактів у світлі початкового досвіду розгортання цих цифрових технологій відстеження контактів та висвітлюємо їхні успіхи, невдачі

та підводні камені, незважаючи на те, що кілька дослідницьких робіт розглядали програми та методи відстеження контрактів. Додатки для цифрового відстеження контактів зіткнулися з проблемами, включаючи, але не обмежуючись ними, низький рівень проникнення мобільних телефонів, низький рівень використання користувачами та проблеми з конфіденційністю. У зв'язку з цим ми розглянемо, як різні країни і додатки по-різному реагували на COVID-19 і, як наслідок, мали різний ступінь успіху. Багатостороннє реагування на COVID-19 вимагає цифрового відстеження контактів разом з такими доповненнями, як ручне відстеження контактів, ефективна координація та використання превентивних заходів, таких як карантинна ізоляція, соціальне дистанціювання, гігієнічний контроль та використання масок. Слід зазначити, що хоча цифрові додатки для відстеження контактів мають свої переваги, вони не є панацеєю.

1.1.2 Social Credit Systems

Соціальна кредитна система Китаю - це комплексна правова база, призначена для оцінки "благонадійності" людей, підприємств та державних організацій по всьому Китаю. Термін "соціальний кредит" є навмисно широким та неоднозначним для того, щоб забезпечити найбільшу гнучкість у регулюванні.

"Китайська система соціального кредиту" (також відома як "Китайська рейтингова система") стосується широкої мережі заходів, спрямованих на підвищення рівня "довіри" в китайському суспільстві, і вписана в правову базу.

Китайська система соціального кредитування оцінює фізичних осіб на основі накопичення та аналізу даних. Деякі спроби використовували або єдину літерну оцінку, або єдину числову оцінку, часто від 1 до 1000, подібно до оцінки FICO (зазвичай від A-D) [4].

На рис. 1.1 зображена загальноприйнята інфографіка, створена німецькою компанією Bertelsmann Stiftung, яка демонструє принцип роботи Китайської рейтингової системи.

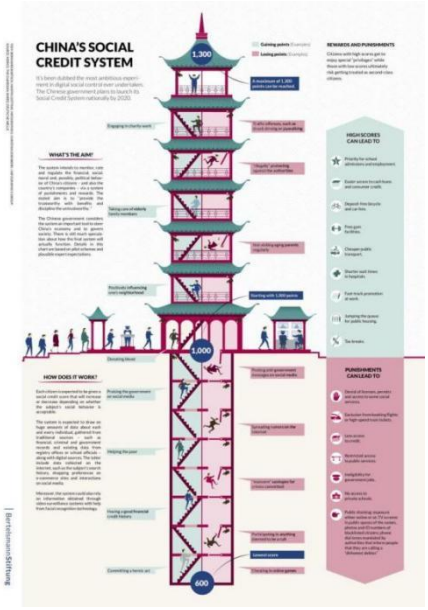


Рисунок 1.1 – Інфографіка китайської системи соціального рейтингу від компанії Bertelsmann Stiftung

Для збору інформації використовуються різні джерела, включаючи приватні компанії (в тому числі "великі технологічні" компанії) та урядові організації. Деякі дані є "закритими", тобто доступ до них має лише певний регіональний або національний орган влади. Але часто дані поширюються серед інших регуляторів через централізовану базу даних, наприклад, NCISP [4].

Очікується, що нещодавні технологічні досягнення суттєво вплинуть на систему соціального кредитування в країні. Згідно з повідомленнями, Китай наразі використовує програмне забезпечення для розпізнавання облич на основі штучного інтелекту (ШІ) разом з більш ніж 200 мільйонами камер спостереження [5].

Метою широкомасштабних методів спостереження є надання китайським чиновникам можливості відстежувати своє населення в кожній сфері повсякденного життя: це, в свою чергу, надає масиви даних для оцінки того, чи відбулося діяння, гідне занесення до чорного списку.

На додаток до цієї зовнішньої тактики моніторингу, китайський уряд також стежить за діяльністю своїх громадян в Інтернеті. Китайська влада може шукати

різноманітні порушення, в тому числі як доказ авторства та поширення антиурядової ідеології.

Більшу частину цієї роботи від імені уряду може виконати програмне забезпечення зі штучним інтелектом, яке також може повідомляти владу про порушення. Сучасний рівень розвитку технологій дозволяє ШІ розпізнавати кадри антиурядових демонстрацій і не допускати їх перегляду користувачами.

Соціальна кредитна система Китаю має на меті стати всеосяжною системою оцінки надійності людей, бізнесу та державних суб'єктів у Китаї.

На сьогодні не існує єдиного універсального "соціального кредитного балу", який ведеться на кожну особу для оцінки її надійності. Замість цього існує цілий ряд окремих рейтингів, які мають різні муніципальні та провінційні уряди, а також урядові міністерства.

Інформації щодо технічних особливостей реалізації системи соціального кредиту в Китаї майже не має, так як система є закритою та не має уніфікованого вигляду, як вже було наведено - це певний комплекс різних технічних та політичних рішень, що застосовуються на різних рівнях суспільного життя.

1.1.3 Digital twin та Social Digital Twin

Цифровий двійник - це цифрова версія реального об'єкта, такого як система, людина, місце або предмет. Цифрові двійники спочатку призначалися для використання в симуляціях з дуже точними моделями окремих компонентів для вдосконалення виробничих процесів. Однак тепер стало можливим розробляти розумні міста-двійники завдяки все більш точним інформаційним моделям будівель (BIM) і величезним обсягам даних, що надходять від датчиків Інтернету речей в розумному місті [6]. Громадськість може вивчити детальну 3D-модель міста, яка була опублікована в Інтернеті, щоб побачити запропоновані зміни в міському плануванні та політиці. Перед тим, як втілити ці рішення в життя, це полегшує публічне розкриття інформації та прозорість.

Соціальний цифровий двійник забезпечує середовище для моделювання, прогнозування та прийняття рішень для вирішення різноманітних та складних соціальних питань шляхом цифрового відтворення зв'язків та відносин між людьми, речами, економікою та суспільством. Першою спробою описати та реалізувати концепцію цифрових двійників було дослідження компанії Fujitsu при університеті Карнегі-Меллона.

Центр аналізу даних про мобільність (ЦАД) при університеті Карнегі‑Меллона працює над проектом, який використовуватиме реальні дані, такі як вхідні дані з правил дорожнього руху та руху транспортних засобів, для оцінки ефективності стратегій контролю та динамічної оцінки транспортних потоків. Ще одна спільна робота з Лабораторією обчислювальної поведінки університету Карнегі-Меллона в Інституті робототехніки Школи комп'ютерних наук покращить існуючі навички 3D-моделювання пішоходів та прогнозування їхньої поведінки в міських умовах. Ця технологія може бути використана для відстеження руху на вулицях та визначення потенційних проблемних зон або місць концентрації ДТП [7].

Також у проєкті Fujitsu та університету Карнегі-Меллона використали так звані "конвергентні технології" - передові інновації, які об'єднують комп'ютерні науки зі знаннями з гуманітарних та соціальних наук, з метою вирішення різноманітних та складних проблем, з якими стикаються міста, працюючи над реалізацією сталого суспільства [8]. Дослідники хочуть створити нову платформу, яка пропонує широкий спектр рішень для різних соціальних проблем, заснованих на надзвичайно точному моделюванні руху людей і транспортних засобів і здатності візуалізувати і передбачати майбутні дії, а також потенційні ризики, засновані на поведінкових особливостях людини. Для покращення результатів міського планування та політики можна заздалегідь відобразити наслідки та потенційні небезпеки дій, використовуючи нещодавно розроблену платформу Social Digital Twin для вивчення та прогнозування поведінки людей та руху транспортних засобів. Інфографіку принципів роботи соціального цифрового

двійника, створену університетом Карнегі-Меллона за підтримки компанією Fujitsu, можна спостерігати на рис. 1.2.

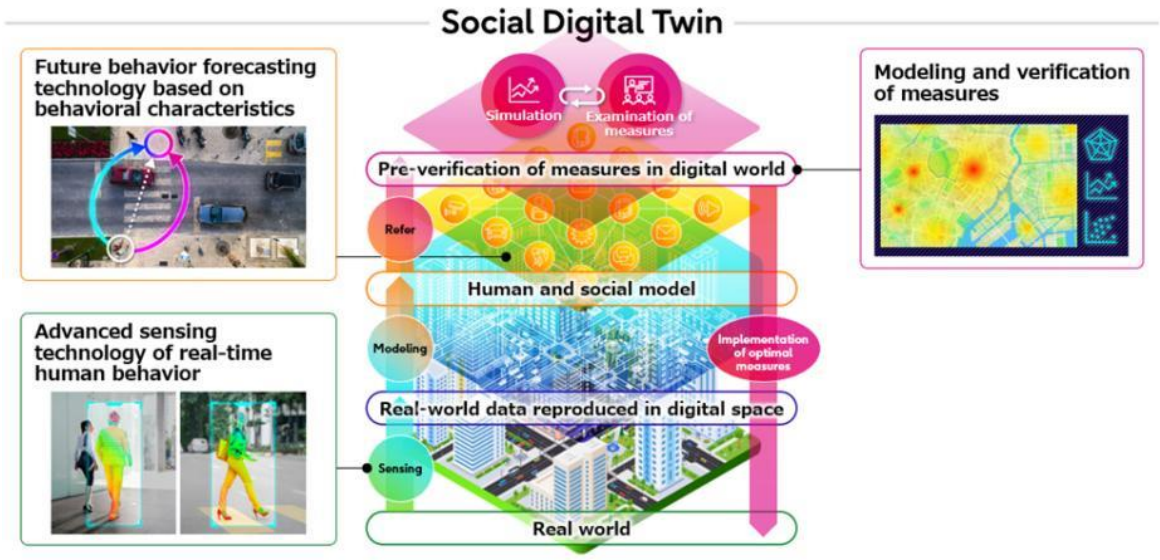


Рисунок 1.2 – Інфографіка соціального цифрового двійника міста від університету Карнегі-Меллона та компанії Fujitsu

Fujitsu та університет Карнегі-Меллона використовуватимуть платформу Social Digital Twin для сприяння перевірці конкретних заходів, спрямованих на вирішення екологічних проблем, включаючи скорочення викидів CO2 та вдосконалення міських транспортних мереж, на додаток до аналізу заторів на дорогах та шляхів забезпечення економічної ефективності. Дослідники університету Карнегі-Меллона також продовжать працювати над підтримкою політики, яка дозволить гнучко і ефективно розподіляти медичні ресурси, одночасно сприяючи економічному зростанню, допомагаючи реалізувати безпечні і стійкі "розумні" міста наступного покоління [8].

1.1.4 Health tracking та BSN

Сучасна система охорони здоров'я стає зручнішою для пацієнтів та лікарів завдяки впровадженню технологій Інтернету речей. Платформа Body Sensor Network (BSN), яка використовує мережу бездротових сенсорних вузлів, що є легкими або малопотужними, дозволяє здійснювати моніторинг пацієнтів як інновацію IoT в медицині. Безпека була найважливішим компонентом технології

health tracking. Оскільки різні люди мають різні думки з цього приводу, захист визначався по-різному. Безпека, як правило, розглядається як щось подібне до захисту системи в цілому [9]. Більшість додатків сенсорних мереж, що використовуються в медицині, в даний час взаємодіють бездротовим способом [10].

Застосування BSN дозволило використовувати комп'ютерні технології в охороні здоров'я. Останнім часом було зроблено ряд пропозицій та ініціатив в галузі клітинної медицини, які мають потенціал для відкритого, амбулаторного, клінічного та безперервного моніторингу пацієнтів. У цій главі обговорюється ряд відомих досліджень, які використовують мережі детекторів тіла в галузі медицини. Гарвардська лабораторія сенсорних мереж заснувала широко використовуваний медичний дослідницький проект BSN CodeBlue. В рамках цього проекту на тілі пацієнта розміщуються численні біосенсори. Ці програми стежать за фізичним станом пацієнта і дистанційно передають результати аналізів користувачеві [11]. Основна ідея CodeBlue була простою: лікар або інший медичний працівник запитує інформацію про стан здоров'я людини, використовуючи інтелектуальний пристрій у публічному, зареєстрованому місці. Розробники CodeBlue також визнають необхідність забезпечення безпеки в медичному застосуванні, хоча до цього часу або питання безпеки залишалося невирішеним, або компоненти безпеки були навмисно залишені для більш ранніх розробок.

BSN полегшує інтеграцію мережі крихітних, інтелектуальних мікросенсорів в тіло людини, щоб вони могли контролювати внутрішні органи і навколишнє середовище. В останні роки він привернув увагу численних експертів в академічному та бізнес-середовищі [12]. Він має потенціал кардинально змінити медичні технології в майбутньому. BSN часто мають детектуюче обладнання на тілі або в тілі. Вузли датчиків всередині тіла були підключені до інвазивних процедур і точки доступу. В якості альтернативи, сенсорні вузли на тілі можуть бути використані для підключення контролера або неінвазивних пристроїв.

1.1.5 Corporate surveillance

Корпоративне спостереження - це слово, яке позначає будь-який прихований моніторинг діяльності всередині компанії або в галузі, яка пов'язана з цією компанією. Внутрішній моніторинг такого роду часто проводиться з метою захисту інтересів бізнесу. Це досягається шляхом використання різноманітних методів моніторингу співробітників для забезпечення дотримання всіх політик і процедур, а також етичних і законних дій співробітників в особистому житті та при використанні корпоративної власності. Корпоративне спостереження часто включає в себе спостереження за засобами масової інформації, а також використання технологій для відстеження будь-яких матеріалів у відкритому доступі, що стосуються ключових конкурентів, коли мова йде про моніторинг їх дій [13].

Програмне забезпечення, яке відстежує комп'ютерну активність співробітників, часто використовується в рамках внутрішньокорпоративних процесів спостереження. Це програмне забезпечення, яке також відоме як комп'ютерне шпигунське програмне забезпечення, перевіряє, чи використовуються інструменти та з'єднання з Інтернетом лише в робочих цілях, а не для особистого використання. Корпоративне шпигунське програмне забезпечення часто дозволяє отримати доступ до надісланих та отриманих електронних листів, відстежувати пошукові запити, зроблені за допомогою різних пошукових систем, та ідентифікувати відвідані веб-сайти. Такий вид діяльності часто вважається необхідним для підтримки продуктивності праці та забезпечення того, щоб працівники були зосереджені на успішному виконанні завдань, пов'язаних з їхніми посадовими обов'язками.

Корпоративне спостереження може також включати використання інструментів, які відстежують переміщення через об'єкти, що належать компанії, на підприємствах, де безпека має першорядне значення. Знаючи, коли працівник входить до місця з обмеженим доступом, можна запобігти потенційному порушенню безпеки та вжити відповідних заходів. Компанії, які працюють з

конфіденційною інформацією, доступ до якої має лише персонал, що має службовий доступ, часто застосовують цей вид корпоративного захисту.

1.2 Порівняльний аналіз запатентованих рішень

В даному розділі розглядається патентна база. Усього буде наведено три патенти, які вирішують спільну проблему, мають подібну архітектуру або суміжні технологічні рішення відносно теми дипломної роботи. Усі три патенти, використовують IoT технології при вирішенні спільної проблеми моніторингу виявлення, ідентифікації та оповіщення груп осіб про наявну небезпеку розповсюдження хвороб на певній місцевості, шляхом збору та аналізу отриманих даних с сенсорів закріплених на особах.

1.2.1 Патент US20210296008A1. Система моніторингу здоров'я для обмеження поширення інфекції в організації

Основний принцип роботи системи заснований на програмних інструкціях. Програмні інструкції, що зберігаються на запам'ятовуючому пристрої, до якого має доступ обчислювальний пристрій, є частиною системи дистанційного керування та моніторингу стану здоров'я декількох користувачів. Програмні інструкції призводять до того, що апаратний процесор обчислювального пристрою отримує декілька наборів даних, пов'язаних зі станом здоров'я, з декількох мобільних обчислювальних пристроїв, що використовуються декількома користувачами. Інформація, пов'язана зі здоров'ям, включає інформацію про відстеження контактів, діагностичну інформацію та фізіологічну інформацію, зібрану з натільних пристроїв користувачів, що вказує на появу симптомів, пов'язаних з інфекцією. Апаратний процесор оцінює рівні впливу на основі фізіологічних даних, діагностичних даних та рівнів впливу, а також визначає небезпечні стани для конкретного користувача на основі принаймні даних про відстеження контактів [14].

1.2.2 Патент US10902955B1. Виявлення COVID-19 за допомогою сурогатів

Система сортування, яка використовує дані датчиків з пристрою користувача для визначення ймовірності захворювання (наприклад, смартфона або трекера активності). Шляхом порівняння даних датчиків з базовою лінією та зміною до визначеного порогу симптомів виявляється кожен симптом. Система сортування додатково використовує сурогати для ідентифікації деяких симптомів, оскільки пряме вимірювання симптомів за допомогою датчиків, доступних користувачеві, може бути неможливим або недостатньо точним. Наприклад, лихоманка може бути виявлена за допомогою серцевих даних, кашель або задишка можуть бути виявлені шляхом прослуховування записаного звуку, рух пристрою користувача може бути використаний для виявлення втоми, а втрата смаку або запаху може бути виявлена шляхом запису звуку і використання алгоритмів розпізнавання мови для пошуку фраз, які вказують на стан [15].

1.2.3 Патент KR102399702B1. Система знаходження інфекційних хворих та метод розшуку з її використанням

Він відноситься до способу і системи відстеження пацієнтів з інфекційними захворюваннями, більш конкретно, коли користувачі з терміналами знаходяться в безпосередній близькості один від одного і можуть здійснювати бездротовий зв'язок один з одним для обміну інформацією, такою як положення, час і відстань між користувачами. На основі інформації про користувача терміналу, через який було встановлено бездротовий зв'язок з терміналом інфікованої особи, при виникненні інфекційного захворювання точно здійснюється відстеження та управління контактами, які вступили в контакт з інфікованою особою. Він стосується системи відстеження пацієнтів з інфекційними захворюваннями, яка може ефективно відстежувати, спостерігати та контролювати не лише інфікованих людей у місті, але й їхніх контактів, щоб зупинити поширення

інфекційного захворювання, а також способу відстеження пацієнтів з інфекційними захворюваннями з використанням цієї системи [16].

1.3 Теоретичні відомості моніторингових мереж та формування вимог

Збір даних IoT використовує датчики для моніторингу функціональності гаджетів, підключених до Інтернету речей. Датчики збирають і передають дані в режимі реального часу, які зберігаються і витягуються в будь-який час з метою моніторингу стану мережі IoT.

Моніторингові IoT мережі базуються на таких рівнях:

Рівень пристроїв. Фундаментальний рівень архітектури IoT складають підключені пристрої. Ці пристрої включають в себе виконавчі механізми, Bluetooth-пристрої, малопотужні радіопристрої, датчики відстеження даних про навколишнє середовище та інше.

Рівень зв'язку. Цей рівень дозволяє пристроям обмінюватися даними за допомогою таких протоколів, як:

– **CoAP** - цей протокол має більший розмір і заснований на семантиці HTTP. CoAP має гіршу підтримку бібліотек і більш складний для підключення до брандмауерів, ніж MQTT.

– **MQTT** - це протокол, який був створений для вбудованих систем і вдосконалений для Інтернету речей. Він відомий тим, що має значну кількість прихильників і значну колекцію активів.

– Навіть 8-розрядні пристрої низького класу, які підтримують текстові протоколи, можуть використовувати **HTTP/HTTPS.**

Граничний рівень. Операційна система, вбудоване програмне забезпечення та апаратне забезпечення пристроїв Інтернету речей складають цей рівень.

Рівень обробки подій. Дані, зібрані з пристроїв Інтернету речей, обробляються та зберігаються на цьому рівні. На цьому рівні виконуються подальші процедури: організація даних, додавання метаданих до даних IoT,

очищення даних. Для обробки та зберігання даних IoT також можна створити цей рівень, використовуючи хмарні сервіси IoT.

Клієнтський комунікаційний рівень. Через цей рівень передаються дані від пристрою до користувача. Він служить сполучною ланкою між базами даних на внутрішній стороні і користувацькими інтерфейсами на зовнішній стороні.

Щоб вирішувати сучасні виклики моніторингова мережа повинна відповідати таким вимогам:

Узгодженість. Всі пристрої повинні бути узгодженні у мережі. Це також включаючи обробку, зберігання інформації та процедури проактивного зворотного зв'язку, потрібно виконувати безперервно протягом дня.

Обробка великих масивів даних. Під час збору даних IoT часто збираються величезні обсяги потоків даних. Крім того, ці дані повинні бути підготовлені для того, щоб програмне забезпечення могло отримати з них цінну інформацію.

Сумісність. Архітектури та протоколи систем Інтернету речей неймовірно різноманітні. Дані з датчиків та інформація зі спеціалізованих пристроїв Інтернету речей (IoT) повинні підтримуватися багатьма інструментами.

Безпека. Безпека є головним пріоритетом у будь-який час, у тому числі на етапах збору, обробки та зберігання даних. Оскільки значна частина матеріалів, що розглядаються, може бути чутливою, необхідно приділяти особливу увагу захисту їх від вторгнення.

Висновки до розділу 1

В даному розділі були розглянуті сучасні моніторингові системи для виявлення та попередження розповсюдження хвороб серед населення. Були виявлені їх головні переваги, недоліки, та особливості на основі яких розробляється система приведена в даній роботі. Також були проаналізовані запатентовані рішення подібні до розроблювальної системи. В кінці на основі зібраної інформації та інформації з відкритих джерел були сформовані вимоги, щодо побудови моніторингової мережі.

2 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ МОНІТОРИНГОВОЇ МЕРЕЖІ. КОНЦЕПТУАЛЬНІ СХЕМИ. ФУНКЦІОНАЛЬНІ ТА МАТЕМАТИЧНІ МОДЕЛІ

В даному розділі розглядається проєктування системи моніторингової мережі в кілка етапів. Розглянута концептуальна модель, алгоритми роботи системи, схеми взаємодії компонентів, математичні моделі, апаратна та програмна комплекси, а також архітектура хмарної компоненти.

2.1 Проєктування концептуальної схеми роботи моніторингової мережі

Першим кроком було спроєктовано концептуальну модель системи, яка описує основні компоненти та принципи роботи, функціювання моніторингової мережі. Концептуальна схему проєкту наведена на рис. 2.1.

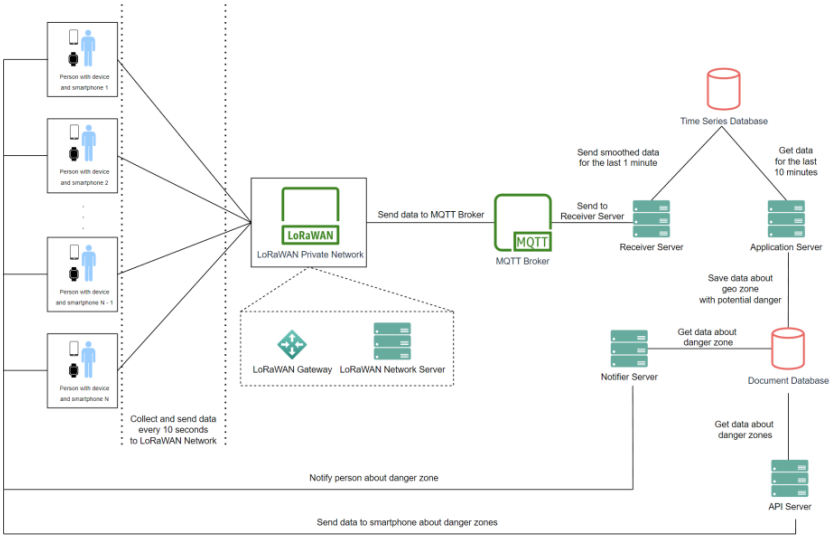


Рисунок 2.1 – Концептуальна схема моніторингової мережі

Систему можна поділити на основні три компоненти: кінцеві користувачі, мережа LoRaWAN та серверна архітектура. Як позначено на рис. 2.1 система починається з кінцевого користувача, на якому розміщені браслет та мобільний телефон з ввімкненим Bluetooth модулем. Браслет збирає основну інформацію: геолокація користувача, температура тіла, пульс, фотоплетизмографію та електрокардіограму. Також браслет отримає дані щодо розташування небезпечних

зон та індексів загроз небезпечних зон через додаток на мобільному телефоні шляхом передачі даних через Bluetooth, які свою чергу отримуються від серверу. На основі зібраних даних формується статус положення користувача у небезпечній зоні та його індекс потенційного зараження. Попередньо дані показників здоров'я та параметрів геолокації згладжуються. Після цього дані через мережу LoRaWAN та MQTT брокер відправляються на сервер у хмарному середовищі для обробки. На сервер передаються лише дані геолокації та індекс потенційного зараження користувача, дані щодо показників здоров'я та попередні індекси зберігаються лише на кінцевому пристрої і нікуди не передаються.

На проміжному етапі між хмарним середовищем та кінцевими користувачами розташована приватна мережа LoRaWAN. Мережа складається з певної кількості шлюзів (кількість залежить від фізичних параметрів шлюзу, площі та параметрів забудови міста, де буде встановлюватись), підключених до них кінцевих пристроїв, у даному випадку браслетів та мережевих серверів, які керують та управляють мережею та пристроями, а також спрямовують трафік з пристроїв в брокер MQTT. Мережевий сервер LoRaWAN розташований у хмарному середовищі разом з брокером MQTT.

Дані користувачів збираються та спрямовуються на брокер MQTT, на який підписаний сервіс Receiver. Receiver відповідає за отримання даних за певний проміжок часу, їх згладжування та розміщення у базі даних часових рядів. Після чого сервіс Application отримує дані користувачів з бази даних часових рядів за певний проміжок часу після чого кластеризує їх за географічним параметром, таким чином утворюючи зони скупчення людей. Також для кожної зони на основі параметру потенційного зараження користувачів формуються індекси потенційної небезпеки зон. Після чого дані відносно небезпечних зон зберігаються у документній базі даних. Останнім кроком сервіс Notify під'єднується до смартфону користувачів через WebSocket Secure протокол та застосунок на смартфоні, і у разі заходження користувача у небезпечну зону надсилає оповіщення та дані про зону небезпеки для подальшого формування індексу

потенційного зараження користувача. Сервіс API в свою чергу отримує дані зон з документної бази даних. Дані щодо розташування користувача отримуються застосунком через Bluetooth інтерфейс з браслету, це означає, що користувачу не потрібно вмикати GPS на смартфоні. Також сервіс API має REST API інтерфейс взаємодії, для отримання даних на смартфоні через мобільний застосунок, щодо всіх найближчих зон зараження, їх географічного розташування та рівня небезпеки.

2.2 Проєктування алгоритму роботи системи

Наступним етапом було формування детального алгоритму роботи опираючись на поставлені на початку роботи вимоги до системи та теоретичних відомостях про системи моніторингу зібраних та проаналізованих в першому розділі дослідження. Повну схему алгоритму можна знайти у додатку А.

Робота алгоритму починається на рівні пристрою, що зображено на рис. 2.2.

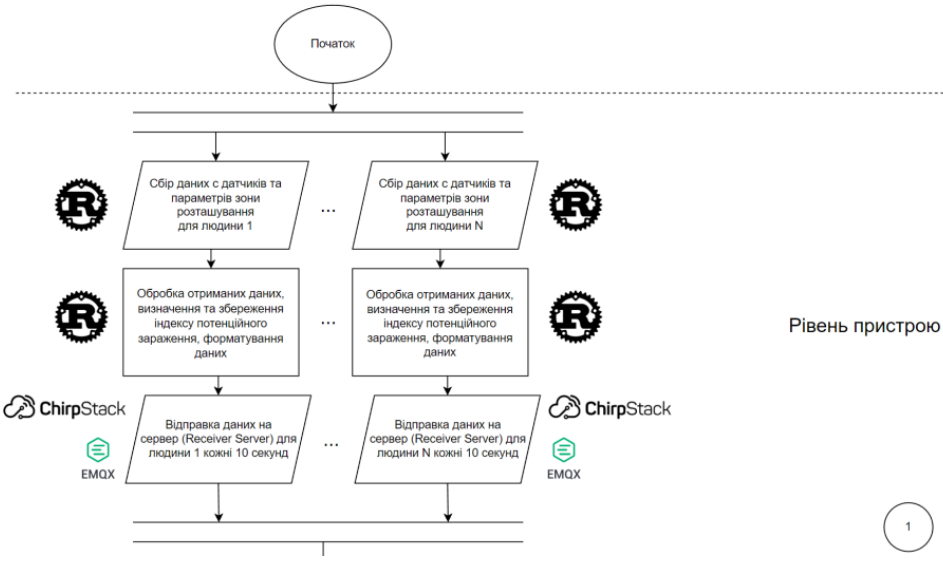


Рисунок 2.2 – Початок алгоритму роботи системи моніторингової мережі

Першим кроком відбувається збір даних (геолокація, параметри здоров'я користувача) з датчиків, кожну секунду. Також з смартфону зчитуються дані щодо небезпечних зон. Наступним кроком дані (геолокація, параметри здоров'я користувача) за останні 10 секунд згладжуються за допомогою фільтру Калмана, а також формується індекс потенційного зараження користувача на основі

параметрів здоров'я та інформації про зону небезпеки. Після цього дані (індекс потенційного зараження, геолокація) відправляються через мережу LoRaWAN та MQTT брокер на сервер Receiver. Ці процеси відбуваються паралельно та окремо для кожного користувача в системі.

Далі відбувається процес збору інформації користувачів на сервісі Receiver. Сегмент алгоритму позначено на рис. 2.3.

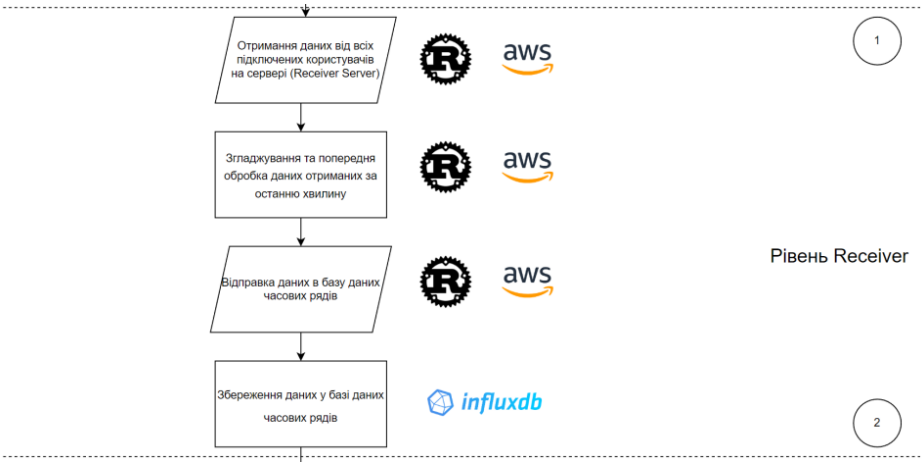


Рисунок 2.3 – Алгоритм роботи сервісу Receiver

Спочатку сервіс Receiver збирає дані індексів потенційного зараження та геолокації всіх користувачів протягом 1 хвилини. Після чого за допомогою експоненційного згладжування згладжує отримані дані та зберігає їх в базу даних часових рядів.

Передостаннім етапом сервіс Application опрацьовує дані користувачів за останні 10 хвилин та формує зони зараження, що зображено на рис. 2.4.

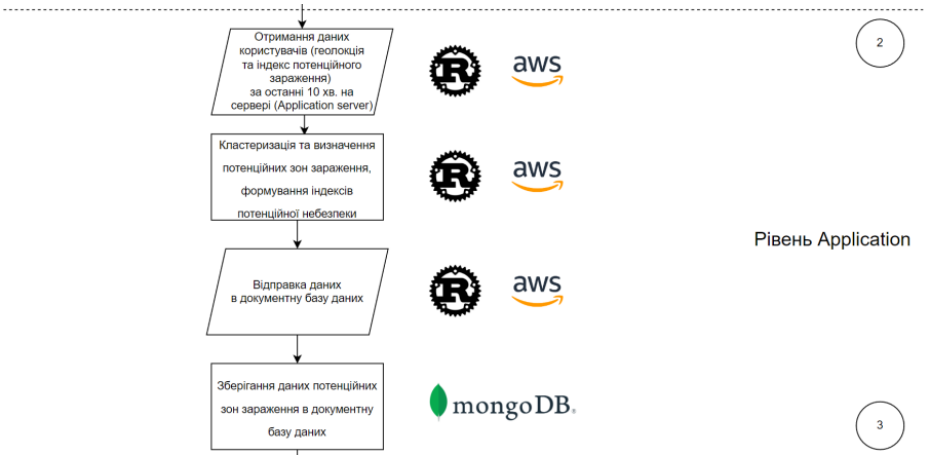


Рисунок 2.4 – Алгоритм формування потенційних зон зараження

Сервіс Application отримує дані з бази даних числових рядів, зібрані за останні 10 хвилин. Після чого за географічним параметром дані користувачів кластеризуються алгоритмом DBSCAN (Density-based spatial clustering with the application of noise). Потім для отриманих зон формуються індекси потенційної небезпеки на основі параметрів щільності та суми індексів потенційного зараження користувачів. Отримані дані про зони небезпеки форматуються в документи (кожен окремий документ – окрема зона скупчення користувачів) та зберігаються в документній базі даних.

Останнім кроком дані з документної бази даних отримує API сервіс який передає ці дані на смартфон до застосунку. Застосунок отримує дані лише, щодо найближчих небезпечних зон, або зон на певній географічній місцевості. Дані передаються або за запитом користувача, або через сервіс Notify при оповіщенні в ситуації, коли користувач опинився в зоні потенційної небезпеки. Робота алгоритму передачі даних зон небезпеки користувачу зображена на рис. 2.5.

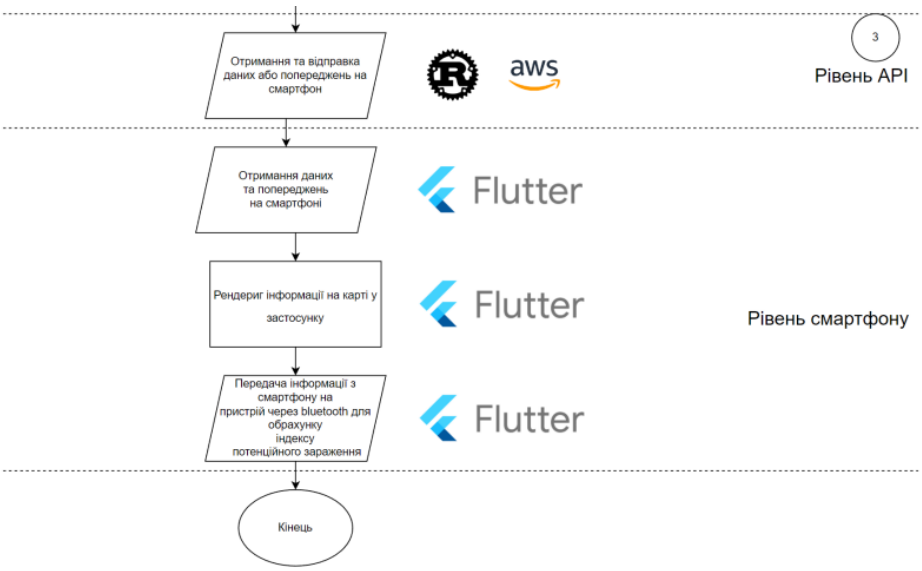


Рисунок 2.5 – Алгоритм передачі інформації про зони небезпеки користувачу

Застосунок відповідає за рендерінг отриманої інформації на карті, а також за вивід корисної інформації в зручному для користувача вигляді. Додатково за запитом застосунок може відправляти дані через Bluetooth до браслету для

формування індексу потенційного зараження користувача, а також отримувати дані, щодо місце-знаходження користувача.

Після побудови концептуальної моделі та створення алгоритму роботи системи було спроєктовано модель взаємодії компонентів під час роботи за допомогою діаграми послідовності. Діаграма послідовності, що зображена на рис. 2.6, описує взаємодію компонентів, область обов'язків компонентів та виконання операцій розподілених у часі.

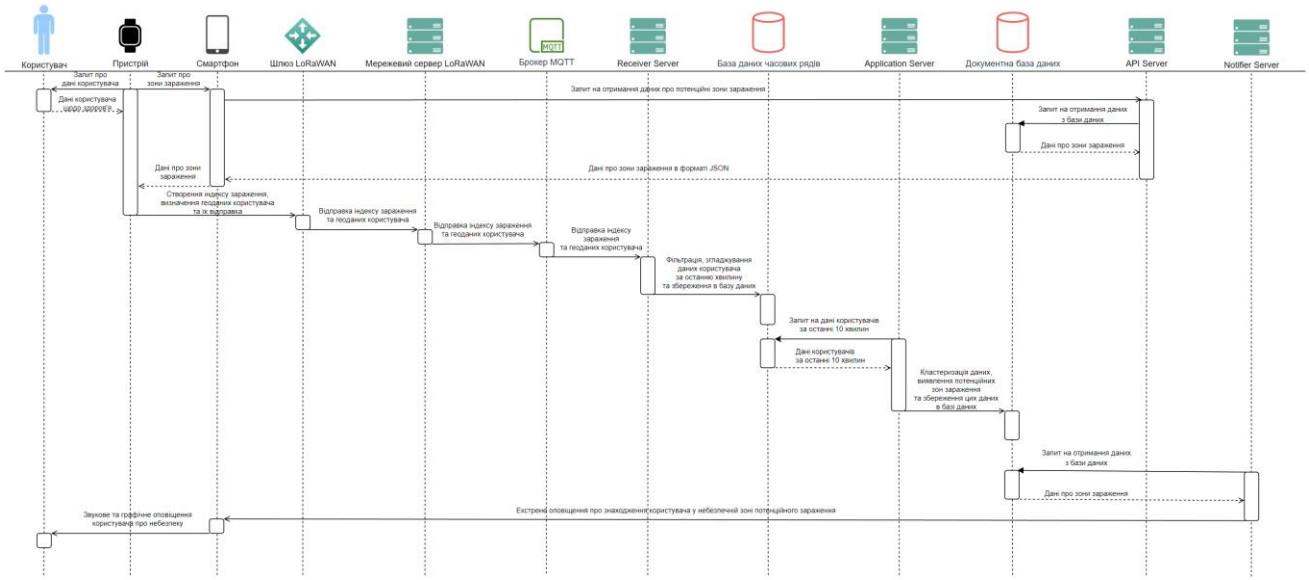


Рисунок 2.6 – Діаграма послідовності моніторингової мережі

Як можна бачити на рис. 2.6 пристрій у вигляді браслету зчитує паралельно дані з смартфону, щодо потенційних зон зараження та дані здоров'я користувача. Під час отримання даних зон потенційної небезпеки, смартфон у синхронному режимі робить запит до API сервісу, який в свою чергу в синхронному режимі отримує дані з документної бази даних. Після чого пристрій оброблює та асинхронно надсилає дані на Receiver сервіс через мережу LoRaWAN та MQTT. Далі дані за одну хвилину згладжуються та зберігаються в базу даних часових рядів. Після чого Application сервіс в синхронному режимі отримує дані користувачів за останні 10 хвилин, формує зони потенційної небезпеки та в асинхронному режимі зберігає їх в документну базу даних. Після чого застосунок кожного разу перевіряє місцезнаходження користувача, звіряє їх з отриманими

даними, щодо потенційно небезпечних зон з API сервісу, і у разі перетину небезпечної зони, надсилає оповіщення за допомогою Notify сервісу в асинхронному режимі. З діаграми можна зробити висновок, що майже всі операції відбуваються послідовно і лише декілька паралельно.

2.3 Математичні моделі роботи с даними. Алгоритми згладжування та кластеризації даних

Під час проєктування системи була виявлена необхідність створення математичної моделі формування індексів загроз. Першим кроком було сформульовано індекс PPI (Personal Potential Infection), тобто індекс потенційного зараження користувача (2.1):

$$PPI = \frac{tmp}{tmp_{max}} * a_{tmp} + \frac{ZPD}{ZPD_{max}} * (1 + \frac{t}{t_{max}}) * a_{zpd} + \frac{PPI_{prev}}{PPI_{max}} * a_{PPI} + \frac{pl}{pl_{max}} * a_{pl} + \frac{e}{e_{max}} * a_{\text{e}} \quad (2.1)$$

де $a_{\text{tmp}} = 3;\ a_{\text{zpd}} = 1.5; a_{\text{PPI}} = 2; a_{\text{pl}} = 1; a_{\text{e}} = 1$

$a\ -$ вагові коофіцієнти розподілу;

t_{max} має функціональну залежність від ZPD: чим більший ZPD тим менший t_{max}.

Даний індекс описує потенційний рівень можливого зараження користувача на сонові параметрів здоров'я та географічного розташування. Формула складається з суми компонентів, тобто з показників здоров'я, значення попереднього індексу потенційного зараження (PPI_{prev}), та рівня небезпечності зони (ZPD) помножений на час проведений там (t). До складу показників здоров'я входять такі параметри як температура тіла (tmp), значення пульсу (pl) та електрокардіограма (e). Пульс та електрокардіограма в даній формулі представлені у вигляді десяткових дробів, які в свою чергу є числовими показниками відхилення поточних значень від норми. Значення норм пульсу та електрокардіограми визначаються програмно на основі зібраних даних користувача на кінцевому пристрої. Кожний параметр зводиться до значення десяткового дробу в межах від 0 до 1 шляхом ділення на максимально допустиме

значення параметру. Максимально допустиме значення встановлюється для кожного параметру окремо відповідно до природних особливостей параметру та особистих характеристик користувача. Також слід зазначити, що параметр максимально допустимого часу t_{max} функціонально залежний від поточного значення рівня небезпеки зони ZPD, чим небезпечніша зона (більший рівень загрози), тим менше значення допустимого часу для зараження t_{max}. Залежність значення максимально допустимого часу зараження t_{max} до індексу потенційної небезпеки зони ZPD, встановлюються експериментальним шляхом і залежать від багатьох параметрів, в першу чергу від особливостей хвороби та епідемії. Також кожний компонент у формулі помножений на ваговий коефіцієнт розподілу, так що кінцеве значення індексу потенційного зараження (PPI) має дорівнювати значенню від 0 до 10 у вигляді десяткового дробу. Вагові коефіцієнти задають пріоритетності параметрам в формулі та встановлюються відповідно до особливостей хвороби. Чим більший коефіцієнт тим більше значення компонент має у формулі. Наприклад, для хвороби COVID-19, температура користувача має більше значення ніж пульс та електрокардіограма, тому серед них має більший коефіцієнт, у даному випадку $a_{\text{tmp}} = 3$. Коефіцієнти для формули індексу потенційного зараження (PPI) були встановлені відповідно до особливостей COVID-19, для інших хвороб їх слід перераховувати. Також вагові коефіцієнти розподілу будуть коригуватися у подальшому після отримання більшої кількості експериментальних даних.

Як зазначалось раніше для формування індексу потенційного зараження користувача (PPI), в якості компоненти використовуватися індекс потенційної загрози зони (ZPD). Формула індексу потенційної загрози зони (Zone Potential Danger) зображена нижче (2.2):

$$\text{ZPD} = \frac{\left(\frac{\text{c}_\text{p}}{\pi r^2}\right)}{\rho_{\text{max}}} * a_{\rho} + \frac{\sum_{\text{i}=1}^{\text{c}_\text{p}} \text{PPI}_\text{i}}{\text{c}_\text{p} * 10} * a_{\text{ppi}} \quad (2.2)$$

де $a_{\rho} = 5;\ a_{\text{ppi}} = 5;$

a – вагові коофіцієнти розподілу.

Формула використовується для формування індексу, який визначає рівень небезпеки зони та впливає на рівень потенційного зараження користувачів, які знаходяться в даній зоні протягом певного часу. Формула складається з двох параметрів: щільності зони ρ (кількість людей c_p поділена на площу зони πr^2, так як зона - це круг з певним радіусом) та суми індексів потенційного зараження всіх користувачів в зоні. Також кожний параметр приведений до значень десяткового дробу від 0 до 1 та помножений на вагові коефіцієнти розподілу a, так що кінцеве значення індексу потенційної небезпеки зони (ZPD) знаходиться в межах від 0 до 10 у вигляді десяткового дробу. В даному випадку вагові коефіцієнти розподілені порівну між параметрами, так як в даному випадку пріоритетність параметрів однакова, але при подальших дослідженнях ці значення будуть коригуватися в залежності від отриманих експериментальних даних.

Слід зазначити, що значення параметрів здоров'я та географічні дані положення користувача збираються у певній кількості на кінцевому пристрої та перед відправкою на сервер згладжуються алгоритмом фільтру Калмана

Фільтр Калмана - це алгоритм, який на основі вимірів, зібраних протягом певного часу, дає оцінки невідомих змінних. Щоб отримати ідеальну оцінку стану, він інтегрує інформацію із зашумлених спостережень та прогнозованих станів. Існує кілька технологічних застосувань фільтрації Калмана. Наведення, навігація та керування рухомими об'єктами, особливо кораблями, літаками та космічними апаратами, є популярними сферами застосування. Оскільки фільтри Калмана вимагають мало пам'яті, вони ідеально підходять для систем, які постійно змінюються [17]. Алгоритм фільтру Калмана реалізований на мові програмування Rust можна знайти у додатку В.

Для фільтрації індексів потенційного зараження на стороні серверу сервіс Receiver використовує алгоритм експоненційного згладжування. Експоненціальне згладжування - це емпіричний метод, який використовує експоненціальну віконну функцію (2.3) для згладжування даних часових рядів. Він генерує прогнози, які є

середньозваженими попередніми даними, зі зменшуваними вагами, які експоненціально масштабуються з кількістю спостережень.

$$s_t = \begin{cases} c_1 & : \ t = 1 \\ s_{t-1} + \alpha * (c_t - s_{t-1}) & : \ t > 1 \end{cases} \quad (2.3)$$

де s_t – згладжений ряд;

c_t – первинний ряд;

α – коефіцієнт згладжування, який обирається апріорі з діапазону $(0 < \alpha < 1)$.

Алгоритм експоненціального згладжування написаний на мові програмування Rust наведений у додатку Г.

Також перед визначенням індексів зон потенційного зараження, зони зараження формуються шляхом кластеризації географічних даних місцезнаходження людей. Кластеризація відбувається за допомогою сервісу Application на стороні серверу алгоритмом Density-based spatial clustering of applications with noise (DBSCAN). Реалізація алгоритму DBSCAN на мові програмування Rust наведена у додатку Ґ.

Density-based spatial clustering of applications with noise (DBSCAN) - це алгоритм кластеризації даних, створений Гансом-Петером Крігелем, Мартіном Естером та іншими. Він працює шляхом знаходження основних зразків високої щільності та розширення кластерів з них, і має хорошу продуктивність для наборів даних із зашумленими характеристиками. Він виявляє відхилення в областях з низькою щільністю і об'єднує точки, які знаходяться близько одна до одної, на основі вимірювання відстані (часто евклідової відстані) і мінімальної кількості точок [18].

2.4 Проєктування апаратної частини проєкту

Одним із головних етапів є побудова апаратної частини моніторингової мережі, так як даний компонент суттєво впливає на ціну кінцевого виробу та особливості розгортання системи у місті. Тому система проєктувалась з

урахуванням оптимізації споживання ресурсів та географії, місцевості та особливостей міст в Європі. Також під час проєктування враховувались потреби користувачів, такі як приватність та безпека для здоров'я, та ціна браслету для користувача.

Повну діаграму апаратної частини моніторингової мережі можна спостерігати на рис. 2.7.

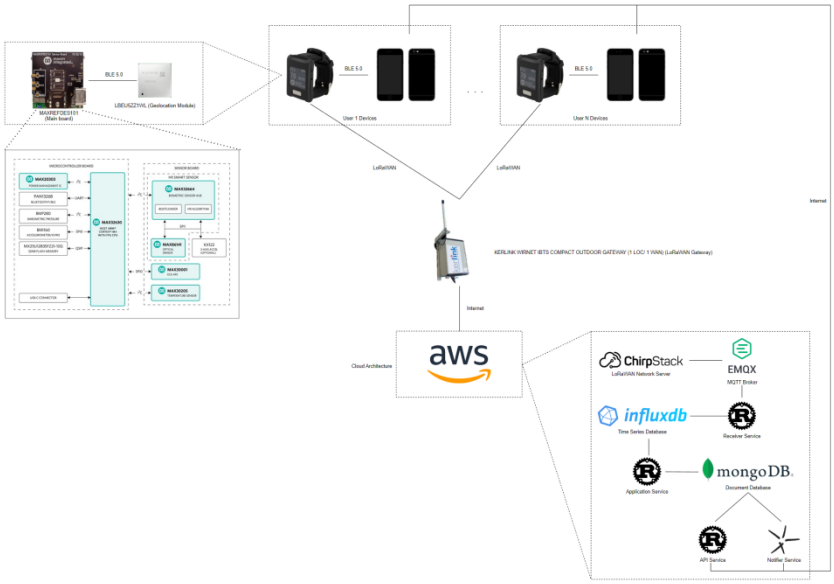


Рисунок 2.7 – Діаграма апаратної частини моніторингової мережі

Пристрої користувача можна розділити на два компоненти: смартфон користувача та браслет, які з'єднані між собою за допомогою інтерфейсу Bluetooth. Смартфон отримує дані з хмарної частини про зони потенційної небезпеки розповсюдження хвороби та передає, ці дані на браслет. Смартфон може бути будь-який, модель залежить від користувача, головна вимога - наявність Bluetooth модулю. В якості браслету використовується виріб із двох компонентів: плати сенсорів здоров'я MAXREFDES101 та модулю геолокації LBEU5ZZ1WL, які також з'єднані за допомогою протоколу Bluetooth. Плата MAXREFDES101 відповідає за збір даних показників здоров'я та складається з таких модулів: MAX20303, MAX30001, MAX30205, MAX32630, MAX32664, MAX86141. Детальний опис плати та її модулів можна знайти у 3 розділі даної роботи. Зібрані дані плата MAXREFDES101 передає за допомогою Bluetooth на

модуль LBEU5ZZ1WL, який в свою чергу збирає дані про географічне положення користувача та надсилає всі зібрані дані на LoRaWAN шлюз KERLINK WIRNET IBTS COMPACT OUTDOOR GATEWAY через мережу LoRaWAN. LoRaWAN шлюз в свою чергу надсилає отримані дані до хмарної частини для подальшої обробки. Користувацьких пристроїв в системі може бути велика кількість, також з урахуванням особливостей урбаністичного середовища, мережа LoRaWAN у більшості міст Європи буде сегментована, тобто мати декілька шлюзів KERLINK WIRNET IBTS COMPACT OUTDOOR GATEWAY. Відповідно до сегментації у хмарному середовищі, кожному сегменту відповідає мережевий сервер LoRaWAN ChirpStack з власним MQTT брокером на Mosquitto, який керує своїм сегментом мережі, а також відповідає за початковий збір даних користувачів у певному сегменті. Один сегмент може включати в себе декілька шлюзів та лише один мережевий сервер. На даному етапі можна збору та обробки даних можна виділити окремий компонент системи - хмарну архітектуру. Хмарна архітектура містить мережеві сервери LoRaWAN ChirpStack, дві бази даних: часову (InfluxDB) та документну (MongoDB), які горизонтально масштабуються, а також сервіси обробки даних, програмних інтерфейсів та сервісів оповіщення написаних на мові Rust та Gleam, які також в свою чергу підтримують горизонтальне та вертикальне масштабування. Вся програмна архітектура керуванням мережею LoRaWAN, збором, обробкою та зберіганням даних розгорнута у хмарному середовищі AWS. В хмарному середовищі після збору даних по сегментам дані відправляються на загальний MQTT брокер EMQX. Після чого дані згладжуються та зберігаються у базу даних часових рядів InfluxDB сервісом Receiver. Далі за допомогою сервісу Application кластеризуються та формуються зони небезпеки та їх індекси, після чого вся інформація зберігається в базі даних MongoDB. Останнім кроком сервіси API та Notify використовують інформацію про зони потенційної небезпеки, отримавши їх з бази даних MongoDB, та відправляють її на смартфон за допомогою інтернет підключення.

2.5 Проєктування програмного компоненту системи

Наступним важливим проєктування було створення програмної частини. Програмна частина була розроблена з урахуванням потреб системи та з розрахунком на великі навантаження. Програмна частина в проєкті представлена у вигляді горизонтально масштабованих мікросервісів. Для кожного мікросервісу використовувався свій балансувальник на HAProxy. Програмна компонента також містить мережевий сервер LoRaWAN на ChripStack з локальним брокером повідомлень Mosquitto, центральний брокер MQTT EMQX, та дві бази даних InfluxDB та MongoDB. Повну діаграму програмного комплексу можна знайти у додатку Б.

Система починає працювати з кінцевих пристроїв тобто з браслетів для яких була написана прошивка на мові програмування Rust, що можна спостерігати на рис. 2.8.

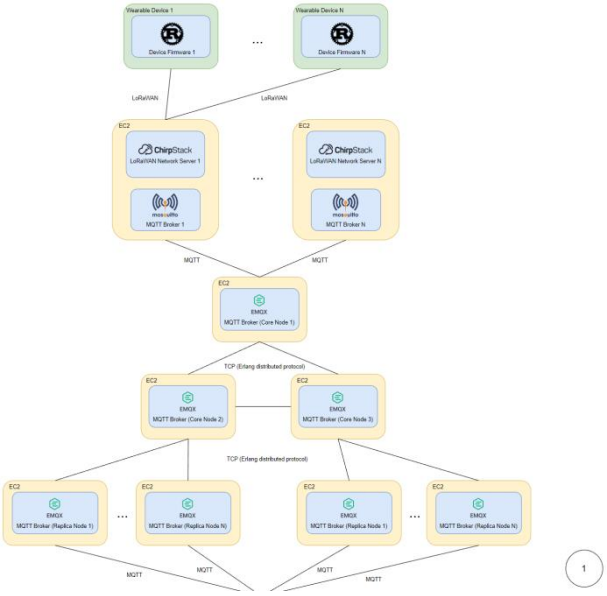


Рисунок 2.8 – Рівень пристроїв програмної діаграми моніторингової мережі

Прошивка запускається на bare-metal без проміжних операційних систем, і відповідає за збір, обробку (згладжування за допомогою фільтру Калмана) та передачу даних через протокол LoRaWAN. Далі як видно на рис. 2.8 на екземплярах EC2, тобто віртуальних машинах хмарного середовища AWS, розміщений сервіс мережевого серверу LoRaWAN ChirpStack та локальний

брокер MQTT Mosquitto. Мережевий сервер LoRaWAN ChirpStack масштабується в системі за допомогою шардування та сегментації, тобто кожний екземпляр відповідає за певну кількість шлюзів LoRaWAN до яких під'єднані кінцеві пристрої за протоколом LoRaWAN. Кожний екземпляр Mosquitto збирає дані лише певного сегменту та передає їх за протоколом MQTT до центрального брокеру EMQX. Центральний брокер повідомлень EMQX, також горизонтально масштабується: має три екземпляри типу Core та 4 екземпляри типу Replica. Екземпляри типу Core в EMQX - це повноцінні сервіси, які можуть як читати так і писати дані, в свою чергу сервіси EMQX типу Replica можуть лише читати дані. Core екземпляри підключені між собою один до одного, обмінюються даними між собою за спеціальним протоколом написаним на мові Erlang поверх протоколу TCP та досягають консенсусу за RAFT алгоритмом. Тому в системі екземплярів Core має бути саме непарна кількість. До кожного екземпляру Core підключені декілька екземплярів Replica, які реплікують дані з Core для читання сторонніми сервісами. Replica сервіси підключені до Core через той же протокол, що й Core до Core.

Наступним кроком на дані, що знаходяться в центральному брокері EMQX підписується сервіс Receiver, що можна спостерігати на рис. 2.9.

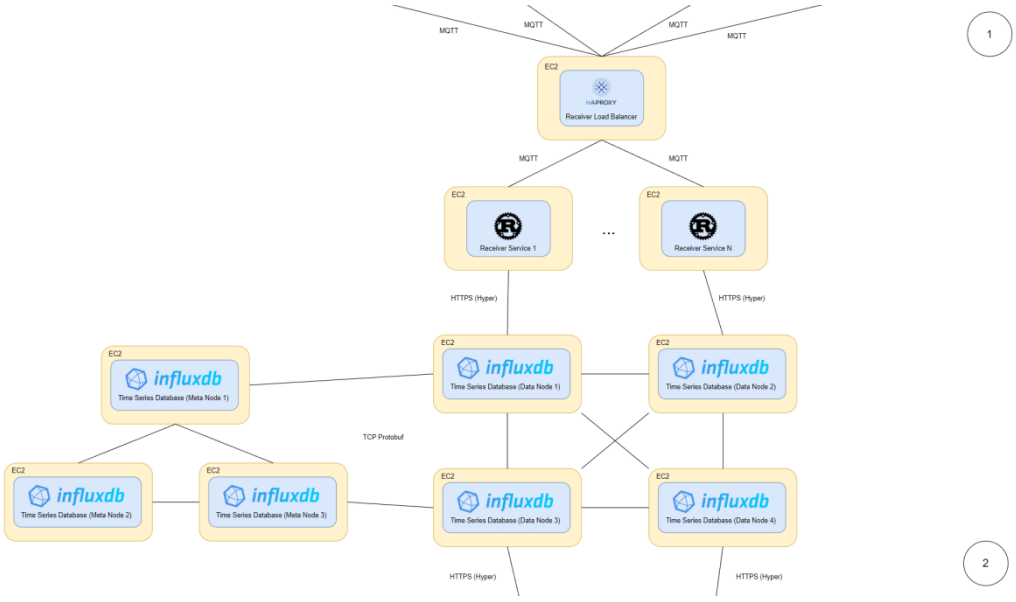


Рисунок 2.9 – Рівень сервісу Receiver програмної діаграми моніторингової мережі

Сервіс Receiver отримує дані через протокол MQTT з екземплярів Replica EMQX брокеру через балансувальник HAProxy. Сервіс Receiver написаний на мові програмування Rust, горизонтально масштабується. Сервіс Receiver відповідає за отримання індексів потенційного зараження та географічного положення, згладжування отриманих даних експоненціальним згладжуванням, та зберігання даних у базі даних часових рядів InfluxDB. Формат даних для збереження у базу даних часових рядів можна спостерігати на рис. 2.10.

user_id	time	ppi_index	latitude	longitude
String	Time	Float	Float	Float

Рисунок 2.10 – Схема даних бази даних InfluxDB

В базі даних часових рядів зберігаються такі дані як індекс користувача (user_id) для його ідентифікації в системі; часова мітка (time), яка визначає точний час отримання даних; індекс потенційного зараження користувача (ppi_index) та координати місцезнаходження користувача (latitude, longitude) на момент отримання даних.

Також слід зазначити, що сервіс Receiver “спілкується” з базою даних за допомогою протоколу HTTPS, та в кодовій частині використовує бібліотеку Hyper для цих потреб. База даних InfluxDB горизонтально масштабується за допомогою двох видів вузлів: Meta та Data. Meta вузли відповідають за адміністрування та управління системою, а також розкривають HTTP API, який використовує команда influxd-ctl. Системні адміністратори використовують цю команду для виконання дій на кластері, таких як додавання і видалення серверів, переміщення шардів (величезних блоків даних) і виконання інших адміністративних обов'язків. Для їх спілкування використовується протокол TCP Protobuf і група консенсусу RAFT. Тому Meta вузлів в системі має буди непарна кількість. Data вузли в свою чергу відповідають за маніпуляції з даними, тобто читання та писання в базу даних. Саме до них підключені всі інші сервіси в системі, такі як Receiver та Application, через протокол HTTPS, коли в свою чергу Meta вузли керують Data

вузлами та їх кластером. TCP і протокол Protobuf використовуються Data вузлами для зв'язку один з одним. Кожен Meta вузол у кластері повинен взаємодіяти з кожним іншим Meta вузлом. Кожен Data вузол повинен взаємодіяти з кожним іншим Data вузлом і кожним Meta вузлом. Використовуючи протокол Protobuf через TCP, Data вузли взаємодіють один з одним і реплікують дані. Всі записи і запити повинні оброблятися Data вузлами.

Далі сервіси Application отримують дані користувачів з бази даних часових рядів InfluxDB через протокол HTTPS, що зображено на рис. 2.11.

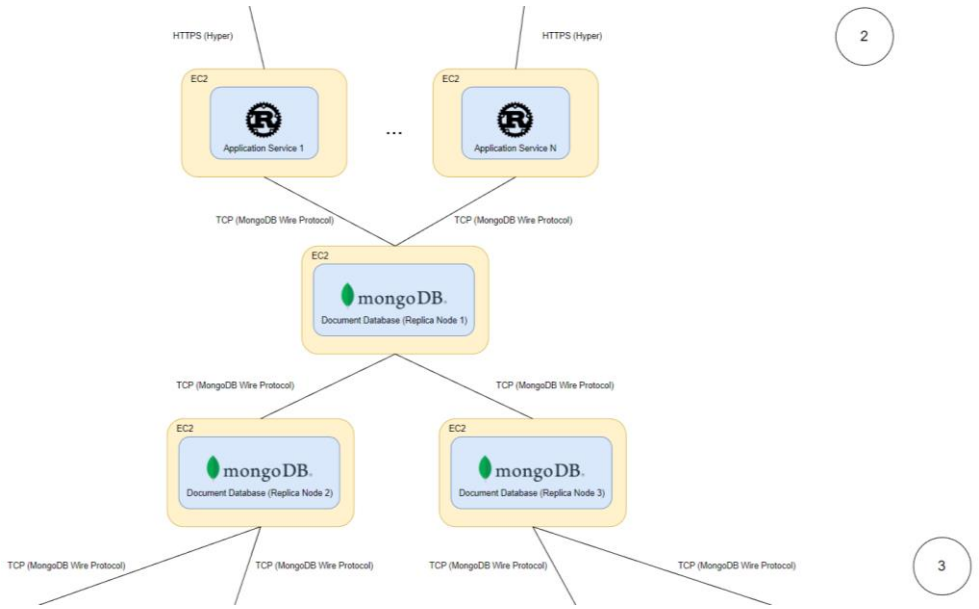


Рисунок 2.11 – Рівень сервісу Application програмної діаграми моніторингової мережі

Сервіс Application також написаний на мові програмування Rust та у якості бібліотеки підключення до бази даних часових рядів використовує Hyper. Сервіс Application відповідає за отримання даних геолокації та параметрів здоров'я користувачів та подальше згладжування та кластеризацію цих даних. Для згладжування використовується алгоритм експоненційного згладжування, а для кластеризації - Density-based spatial clustering of applications with noise (DBSCAN). Сервіс Application також горизонтально масштабується за рахунок розпаралелювання виконання кластеризації шляхом географічного сегментування та модифікованого алгоритму DBSCAN HY-DBSCAN [19]. Після кластеризації та

визначення індексу потенційної небезпеки зони сервіс Application зберігає дані в базу даних MongoDB у вигляді документів, де один документ - це одна небезпечна географічна зона скупчень людей. Схему даних MongoDB можна спостерігати на рис. 12.

Рисунок 2.12 – Схема даних бази даних MongoDB

Схема даних MongoDB вміщає в себе такі компоненти: час створення зони небезпеки (date), що відповідає за значення актуальності небезпеки; індекс потенційної небезпеки зони (zpd_index), радіус зони та центральна географічна точка зони (location). Зони потенційної небезпеки перебудовюються кожні 10 хвилин у системі, з подальшим видаленням попередніх даних. Але, у випадку проводження детальної аналітики розповсюдження епідемії, застарілі дані можуть зберігатися.

Сервіс Application взаємодіє з базою даних на програмному рівні через спеціальний протокол написаний поверх TCP MongoDB Wire Protocol, та для цих потреб використовує стандартні бібліотеки. База даних MongoDB в даному випадку масштабується за допомогою звичайної реплікації. Тобто система обирає один головний вузол, з яким взаємодіють інші сервіси, в той час інші вузли реплікують зміни зроблені в головному вузлі. У випадку коли головний вузол відмовляє, система обирає головним одного з реплікантів. Консенсус в кластері MongoDB досягаються за допомогою MongoDB Primary Election алгоритму. Головний вузол підключений до реплікантів та надсилає їм зміни через протокол

MongoDB Wire Protocol. В подальшому зі збільшенням потреб та навантаження для масштабування разом із реплікацією буде задіяне шардування.

Далі користувач через мобільний застосунок написаний за допомогою фреймворку Flatter та мови програмування Dart, отримує дані щодо зон небезпеки, а також поточне положення спираючись на дані отримані з браслету через Bluetooth. Дані на смартфоні отримуються двома способами в яких задіяні два сервіси API та Notify. Смартфон також відповідає за оповіщення користувача у випадку потенційної небезпеки чи зараження, а також за інформування користувача про можливі наявні небезпечні зони поблизу, з метою попередження. Оповіщення відбувається звуковим чи вібро сигналом, а також на екрані смартфону у застосунку. Вся інформація по небезпечним зонам також відображається на карті у застосунку. Слід зазначити, що у майбутньому планується додавання алгоритму побудови оптимального маршруту до кінцевої точки на карті з урахуванням небезпечних зон та урбаністичного ландшафту, а також відображеня маршруту в користувацькому застосунку. Кінцеву схему роботи програмної частини можна спостерігати на рис. 2.13.

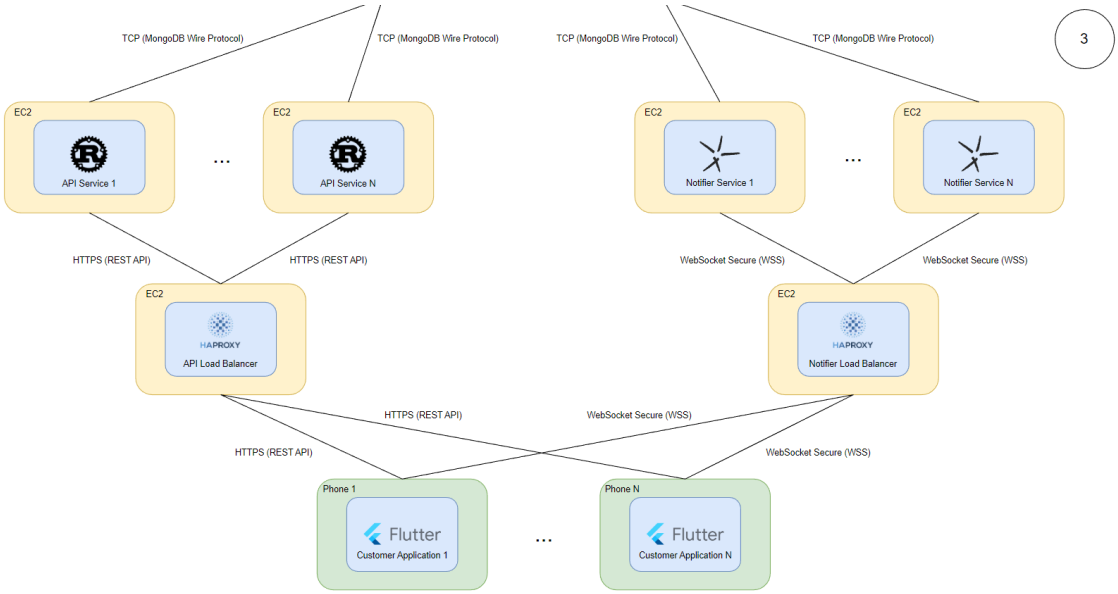


Рисунок 2.13 – Кінцевий рівень API програмної діаграми моніторингової мережі

Дані щодо зон потенційної небезпеки сервіси API та Notify беруть з бази даних MongoDB через протокол MongoDB Wire Protocol. Доступ до сервісів

застосунок отримує через балансувальники. Балансувальники балансують навантаження в горизонтально масштабованих сервісах API та Notify на протоколах HTTPS та WebSocket Secure. Яка зазначалось сервіси API та Notify горизонтально масштабовані задля витримування навантаження великої кількості користувацьких запитів через смартфони. Сервіс API представляє собою web‑застосунок написаний на мові програмування Rust та надає REST API інтерфейс для отримання даних небезпечних зон скупчень людей. API сервіс в системі передає дані тільки за запитому застосунку у фоновому режимі, або за запитом користувача. Сервіс Notify написаний на мові програмування Gleam, який свою чергу інтерпретується на віртуальній машині BEAM, що є дуже релевантним варіантом для телекомунікаційних та розподілених системи [20]. Сервіс Notify відповідає лише за оповіщення користувача, у випадку коли останній опинився у зоні потенційної небезпеки зараження. Також сервіс Notify з певною невеликою періодичністю відправляє запит на смартфон для отримання смартфоном даних, щодо небезпечних зон через REST HTTPS API запит до сервісу API. Ніякі персональні дані застосунок не передає до хмарного середовища, тільки запити на отримання даних. Також застосунок створює та залишає відкритим WebSocket на смартфоні для “спілкуванням” з сервісом Notify.

2.6 Проєктування хмарної архітектури

Кінцевим кроком було створення хмарної архітектури системи моніторингової мережі. Хмарна архітектура була створена за допомогою сервісу CloudCraft та використовувала компоненти хмарного провайдера AWS, на якому в подальшому буде розгорнута система. Схема хмарної архітектури можна знайти на рис. 2.14.

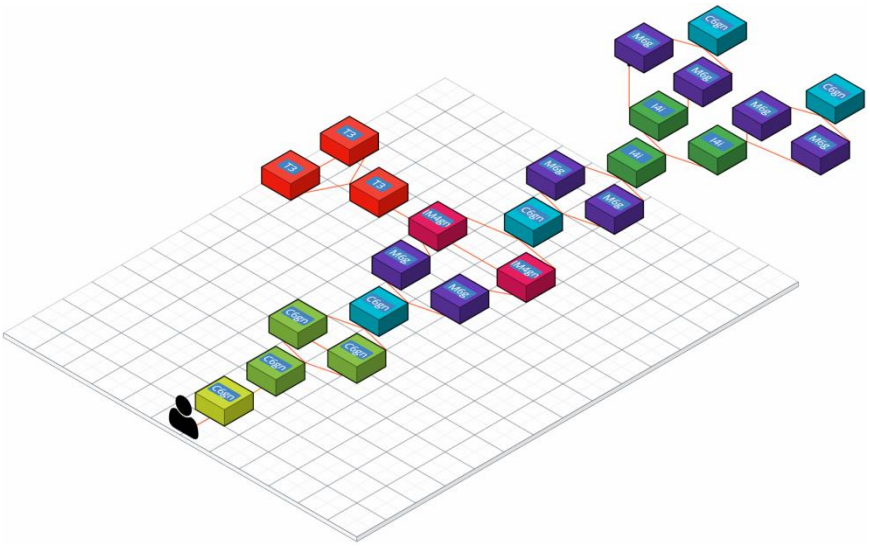


Рисунок 2.14 – Схема хмарної архітектури моніторингової мережі

Створена архітектура є реалізацією спрощеної схеми програмної частини системи. Слід також зазначити, що архітектура повністю побудована на EC2 віртуальних машинах, де кожний тип віртуальної машини підбирався відповідно до задачі. Тобто для балансувальників були обрані C6gn екземпляри з розрахунком на великі мережеві навантаження; для сервісів - M6g екземпляри (тип загального призначення з балансом ресурсів); для розміщення бази даних MongoDB - I4i екземпляри (оптимізовані для швидкої роботи з даними); для Meta вузлів бази даних InfluxDB використовувались екземпляри типу T3 (дешеві та економні по ресурсам, так як Meta вузли не потребують великих навантажень і відповідають лише за невеликий менеджмент кластеру), в свою чергу для Data вузлів кластеру бази даних InfluxDB використовувалися екземпляри Im4gn оптимізовані для роботи з даними (на момент написання цієї роботи є передовими в серед підтипів Storage Optimized [21]). Тип віртуальної машини EC2 для брокеру MQTT було підібрано C6gn, також з розрахунком на великі мережеві та обчислювальні навантаження. Віртуальні машини були підібрані відповідно до горизонтального масштабування сервісів, де кожному сервісу відповідає свій екземпляр віртуальної машини. Розмір та кількість ресурсів кожного екземпляру EC2 підбиралися відповідно до теоретично можливих навантажень з розрахунком на зменшення ціни та у подальшому будуть коригуватися відповідно до

отриманих даних роботи системи у реальному світі. Більш детально про обрані типи віртуальних машин описано в 3 розділі даної роботи.

Наступним кроком хмарну архітектуру було оптимізовано з розрахунком на потенційне місце розгортання першого прототипу (Тулон, Франція). Головним критерієм оптимізації було зменшення ціни з розрахунком на потенційне навантаження системи (тобто обчислювальні можливості мали бути меншими не більш ніж 20% від ресурсів початкової архітектури), при залишку можливості горизонтального масштабування у випадку зростання навантаження. Результат оптимізації зображено на рис. 2.15.

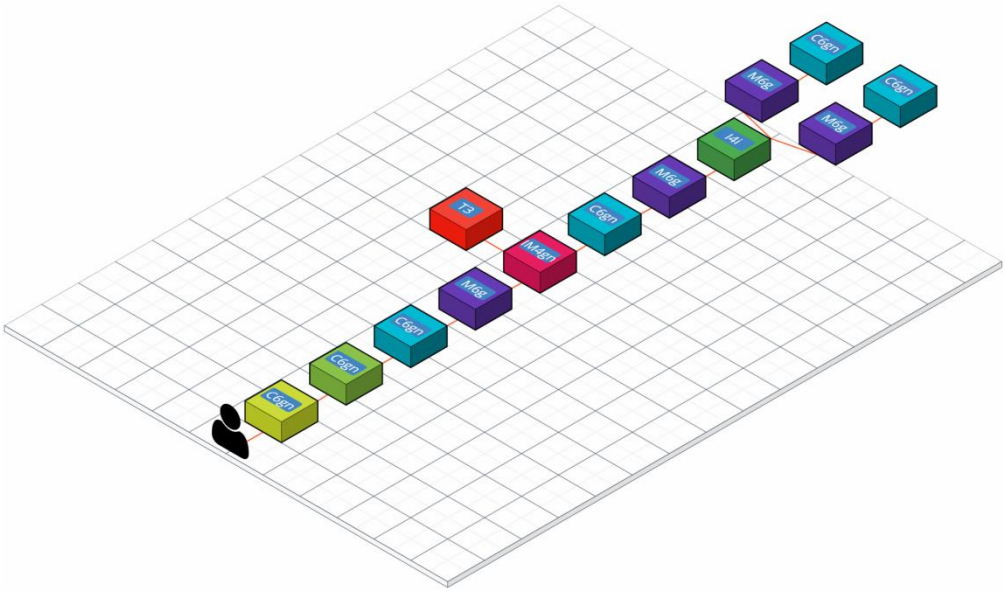


Рисунок 2.15 – Схема хмарної архітектури моніторингової мережі після оптимізації

Основна оптимізація полягала в заміні фізичного горизонтального масштабування на програмне за рахунок скорочення кількості екземплярів віртуальних машин EC2 та вертикального масштабування. В кінцевому результаті кожному сервісу так само як і в початкові архітектурі відповідає свій екземпляр, але всі кластери баз даних та горизонтально масштабовані сервіси запущені на одній віртуальній машині. Результати оптимізації за ціною зображено на рис. 2.16.

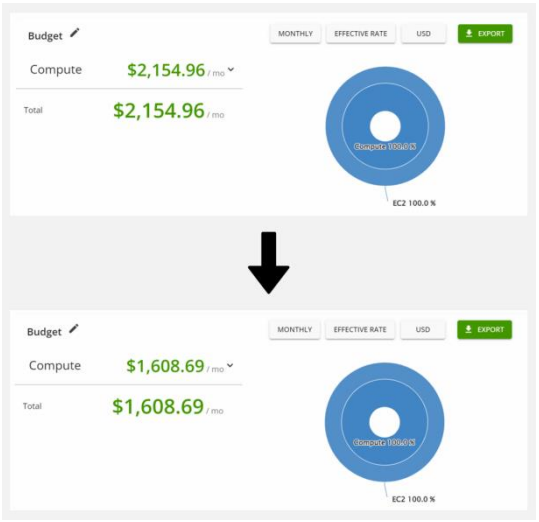


Рисунок 2.16 – Вплив результатів оптимізації на кінцеву вартість серверних ресурсів

Як видно з рисунку кінцеві витрати на хмарні ресурси вдалося оптимізувати з 2154 доларів до 1608 доларів на місяць, тобто на приблизно на 25% від початкової ціни. При цьому загальна потужність по обчислювальним ресурсам віртуальних машин AWS EC2 знизилась лише на 12%. Можливість горизонтального масштабування при такій оптимізації залишається.

Висновки до розділу 2

Розглянуто проєктування системи моніторингової мережі в кілька етапів, включаючи концептуальну модель та алгоритмічну модель роботи системи. Створено апаратні та програмні схеми системи, включаючи опис технічних компонентів та протоколів взаємодії. Створено хмарну архітектуру за допомогою сервісів AWS, з оптимізацією для першого прототипу та вирішенням питань масштабування системи.

3 АПАРАТНО-ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ. ПЕРШИЙ ПРОТОТИП ТА РОЗГОРТАННЯ СИСТЕМИ

Даний розділ присвячений опису підібраних апаратних компонентів та програмних рішень. В даному розділі детально наведені технічні характеристики апаратних компонентів та принципи роботи програмних рішень. Також були відділені особливості обраного апаратно-програмного забезпечення та аргументовано доцільність використання їх в системі.

3.1 Огляд апаратних компонентів

В цьому підрозділі велика увага приділяється апаратним компонентам, які є складовою апаратної частини моніторингової мережі. Тут наведений детальний опис обраних апаратних рішень та особливості, які були ключовими при підборі цих компонентів для системи.

3.1.1 MAXREFDES101 (Health sensor development platforms)

Плата розробника MAXREFDES101, яка являє собою комплекс сенсорів показників здоров'я використовувалась у системі як основа для побудови браслету-пристрою для зчитування важливих параметрів здоров'я користувача.

MAXREFDES101 - це платформа для бездротової платформи датчиків здоров'я, розроблена Maxim Integrated, напівпровідниковою компанією, яка виробляє аналогові та змішані інтегральні мікросхеми. Платформа призначена для моніторингу життєво важливих показників людського організму, включаючи частоту серцевих скорочень, варіабельність серцевого ритму, частоту дихання і температуру тіла, що робить її придатною для використання в різних медичних і фітнес-програмах. Зовнішній вигляд плати можна спостерігати на рис. 3.1.

Рисунок 3.1 – Зовнішній вигляд плати MAXREFDES101

Еталонна плата MAXREFDES101 включає декілька компонентів Maxim Integrated, в тому числі мікроконтролер MAX32664 з наднизьким енергоспоживанням, оптичний пульсоксиметр і датчик частоти серцевих скорочень MAX86141, датчик температури людського тіла MAX30205 і аналоговий модуль біопотенціалу та біоімпедансу MAX30001. Ці компоненти працюють разом, щоб забезпечити точне і надійне вимірювання, передачу і обробку даних [22]. Детальну інформацію по датчикам та модулям можна знайти у табл. 3.1.

Таблиця 3.1 – Перелік модулів плати MAXREFDES101

Назва	Опис
MAX20303	PMIC з стабілізаторами напруги з наднизьким IQ, зарядним пристроєм і паливним датчиком для малих літій-іонних систем
MAX30001	Наднизькопотужний одноканальний інтегрований біопотенціал (ЕКГ, R-to-R та визначення темпу) та біоімпеданс (BioZ) АФЕ
MAX30205	Датчик температури тіла людини
MAX32630	Наднизькопотужна рука Cortex-M4 з мікроконтролером (MCU) на базі FPU з 2 МБ флеш-пам'яті та 512 КБ оперативної пам'яті
MAX32664	Концентратор біометричних датчиків з наднизьким енергоспоживанням
MAX86141	Найкращий у своєму класі оптичний пульсоксиметр і датчик частоти серцевих скорочень для носимого здоров'я

Плата MAXREFDES101 також має бездротовий зв'язок за технологією Bluetooth Low Energy (BLE), що дозволяє передавати дані на смартфон або планшет для моніторингу та аналізу. Платформа призначена для живлення від невеликої літій-іонної батареї, що робить її дуже портативною і зручною для щоденного використання. Повна схема компонентів на платі зображена на рис. 3.2.

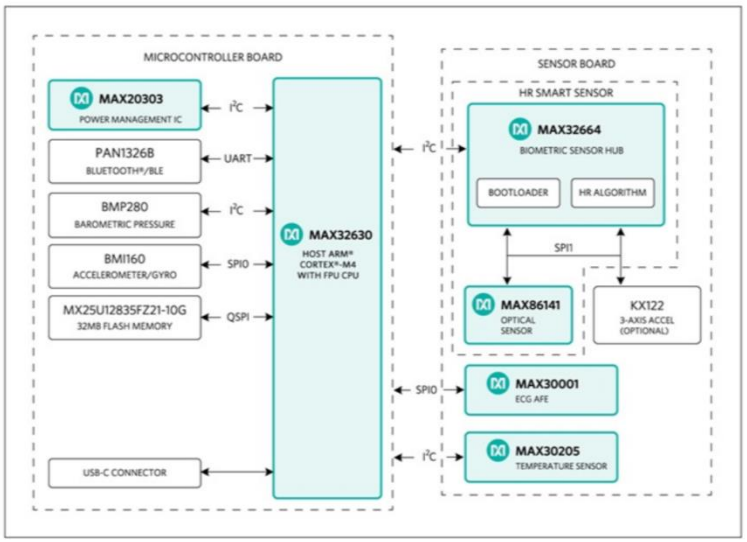


Рисунок 3.2 – Схема розміщення компонентів на платі MAXREFDES101

Еталонна плата MAXREFDES101 доступна у вигляді оціночного набору, що надає розробникам готову до використання платформу для прототипування власних додатків для датчиків здоров'я, що носяться. Дизайн повністю налаштовується і може бути адаптований до широкого спектру випадків використання, що робить його цінним інструментом для виробників медичного обладнання, компаній, що займаються виробництвом фітнес-обладнання, та науково-дослідних установ [23].

3.1.2 LBEU5ZZ1WL (Geolocation Module)

Модуль визначення геопозиції користувача LBEU5ZZ1WL використовується в системі як копмонент для збору геоданих з GPS модулю та даних показників здоров'я через BLE модуль з плати MAXREFDES101, і

подальшої передачі отриманих даних до мережі LoRaWAN через відповідний модуль на чіпі.

Abeeway LBEU5ZZ1WL - це модуль геолокації, призначений для відстеження та моніторингу активів. Це невеликий, малопотужний і економічно ефективний пристрій, який поєднує в собі кілька технологій позиціонування для отримання точних і надійних даних про місцезнаходження. Зовнішній вигляд модулю LBEU5ZZ1WL можна спостерігати на рис. 3.3.

Рисунок 3.3 – Зовнішній вигляд модулю LBEU5ZZ1WL

Модуль оснащений низкою датчиків, включаючи GPS, малопотужну широкосмугову мережу (LPWAN) і позиціювання з підтримкою GPS (A-GPS), що дозволяє йому точно відстежувати і контролювати об'єкти навіть у складних умовах, в тому числі глибоко в приміщенні, на відкритому повітрі і під землею. Модуль також використовує технологію LoRaWAN для зв'язку, яка забезпечує бездротове з'єднання з низьким енергоспоживанням на великі відстані [24].

Основні технічні характеристики модуля Abeeway LBEU5ZZ1WL:

– Мережеві технології: підтримка мереж LoRaWAN Class A / C та BLE.

– Підтримка GNSS: модуль оснащений вбудованим GNSS-приймачем (GPS, GLONASS, Galileo, BeiDou) для визначення географічного положення.

– Інтерфейси: модуль має інтерфейси для підключення до датчиків температури, вологості, освітлення тощо. Крім того, модуль має інтерфейси для підключення до антени, мікро-USB та SPI.

– Дальність передачі даних: в залежності від умов середовища та конфігурації мережі LoRaWAN, модуль може передавати дані на відстань до 15 км.

– Живлення: модуль можна живити від батареї або від зовнішнього джерела живлення. Він має вбудовану літій-іонну батарею ємністю 700 мАг, яка забезпечує роботу в режимі очікування до 5 років.

– Розміри: модуль має компактні розміри 45 x 38 x 15 мм.

– Відповідність стандартам: модуль відповідає стандартам CE та RoHS.

Abeeway LBEU5ZZ1WL компактний і може бути легко встановлений на об'єктах, таких як контейнери, транспортні засоби та обладнання. Він живиться від невеликої акумуляторної батареї, яка забезпечує тривалий час автономної роботи і дозволяє безперервно працювати до декількох років.

Модуль легко інтегрується з існуючими системами управління активами, дозволяючи менеджерам активів відстежувати та контролювати свої активи в режимі реального часу. Крім того, він надає корисну інформацію, таку як місцезнаходження, температура, рух і виявлення ударів, яка може бути використана для оптимізації операцій, скорочення витрат і підвищення безпеки.

В цілому, модуль геолокації Abeeway LBEU5ZZ1WL є надійним і економічно ефективним рішенням для відстеження і моніторингу активів, що підходить для широкого спектру галузей, включаючи логістику, транспорт і управління ланцюгами поставок [25].

3.1.3 Kerlink wirnet ibts compact outdoor gateway (LoRaWAN Gateway)

У якості LoRaWAN шлюзу для підключення браслетів в системі використовується Kerlink wirnet ibts compact outdoor gateway. Компактний вуличний шлюз Kerlink Wirnet iBTS - це невеликий, надійний і високопродуктивний шлюз, призначений для зовнішніх додатків Інтернету речей (IoT). Це шлюз LoRaWAN, який забезпечує бездротове з'єднання з низьким

енергоспоживанням для пристроїв IoT на великі відстані, дозволяючи передавати дані і віддалено керувати пристроями.

Шлюз призначений для розгортання в суворих зовнішніх умовах і може витримувати екстремальні температури, вологість та інші погодні умови. Він має компактну і міцну конструкцію і може бути легко встановлений на стовпах, стінах або інших конструкціях [26].

Зовнішній вигляд шлюзу Kerlink wirnet ibts compact outdoor gateway зображено на рис. 3.4.

Рисунок 3.4 – Зовнішній вигляд шлюзу Kerlink wirnet ibts compact outdoor gateway

Компактний зовнішній шлюз Kerlink Wirnet iBTS має одне місце розташування і один інтерфейс глобальної мережі (WAN), що робить його придатним для невеликих додатків IoT. Він підтримує до 30 000 повідомлень в день і до 1000 пристроїв, що робить його придатним для широкого спектру додатків IoT, включаючи розумні міста, розумне сільське господарство і промислову автоматизацію [26].

Шлюз працює на високопродуктивному процесорі з низьким енергоспоживанням і може дистанційно керуватися через веб-інтерфейс, що дозволяє легко налаштовувати, моніторити та оновлювати прошивку пристроїв. Він також підтримує кілька мережевих протоколів, включаючи MQTT, HTTP і

UDP, що забезпечує безперешкодну інтеграцію з існуючими платформами і додатками IoT.

Основні технічні характеристики Wirnet iBTS Compact Outdoor Gateway:

– Продуктивність: шлюз підтримує до 1000 одночасних з'єднань з дальністю покриття від 5 до 15 км, залежно від умов середовища.

– Пропускна здатність: шлюз може передавати дані зі швидкістю до 300 Кбіт / с.

– Підтримка протоколів: шлюз підтримує різні протоколи передачі даних, включаючи LoRaWAN та MQTT.

– Захист від випадкового доступу: вбудовані заходи захисту дозволяють запобігати випадковому доступу до мережі.

– Відповідність стандартам: Wirnet iBTS Compact Outdoor Gateway відповідає стандартам CE, FCC, IC, RCM та TELEC.

– Інтерфейси: шлюз має різні інтерфейси, включаючи Ethernet, RS-232, RS-485, USB та вхід / вихід відключення живлення.

– Конфігурування та керування: шлюз можна налаштовувати та керувати за допомогою веб-інтерфейсу або за допомогою програмного забезпечення Kerlink Wanesy Management Center.

– Стійкість до атмосферних впливів: Wirnet iBTS Compact Outdoor Gateway має високий рівень стійкості до атмосферних впливів і може використовуватися в різних кліматичних умовах.

– Живлення: шлюз можна живити від мережі живлення або від акумуляторів з можливістю заряд

В цілому, компактний зовнішній шлюз Kerlink Wirnet iBTS є надійним і економічно ефективним рішенням для невеликих додатків IoT. Його міцна і компактна конструкція в поєднанні з великою дальністю дії і низьким енергоспоживанням робить його правильним вибором для різних випадків використання IoT на відкритому повітрі [27].

3.2 Огляд програмних компонентів

В даному розділі розглянуті всі програмні компоненти задіяні при програмному та хмарному проєктуваннях. Були наведені детальні описи використаного ПЗ, мов програмування, їх головні особливості та переваги при виборі для побудови системи моніторингової мережі.

3.2.1 ChirpStack (LoRaWAN Network Server)

У якості мережевого серверу був обраний продукт з відкритим кодом ChirpStack. ChirpStack - це відкрита платформа для розробки та управління мережами Інтернету речей (IoT) на базі протоколу LoRaWAN. Протокол LoRaWAN - це бездротовий протокол мережі дальнього зв'язку, спеціально розроблений для забезпечення низького споживання енергії та великого діапазону покриття для IoT-пристроїв, що працюють на довгих відстанях.

ChirpStack - це мережевий сервер LoRaWAN корпоративного рівня з відкритим вихідним кодом, який дозволяє розгортати і керувати великомасштабними мережами Інтернету речей. Він надає набір мережевих сервісів і API, які дозволяють розробникам і операторам мереж керувати і контролювати свої LoRaWAN-пристрої, шлюзи і додатки.

ChirpStack забезпечує підтримку специфікацій LoRaWAN 1.0, 1.0.2 і 1.1 і сумісний з широким спектром шлюзів і пристроїв LoRaWAN. Він розроблений як масштабований, гнучкий і модульний, що дозволяє мережевим операторам налаштовувати і оптимізувати розгортання IoT відповідно до їхніх конкретних вимог.

ChirpStack надає ряд послуг, включаючи управління пристроями, управління мережею та управління додатками. Служба управління пристроями дозволяє реєструвати, активувати та де-активувати пристрої в мережі, а також керувати профілями пристроїв, прошивкою та ключами безпеки. Служба керування мережею надає такі функції, як мережеве планування, керування чергою низхідної лінії зв'язку та керування шлюзами. Сервіс управління

додатками дозволяє розробникам створювати і керувати своїми IoT-додатками, включаючи обробку даних, тригери подій і інтеграцію з зовнішніми системами.

ChirpStack також надає набір API, які дозволяють розробникам інтегрувати свої додатки з мережевим сервером. Ці API включають RESTful HTTP API, gRPC API та WebSocket API, що дозволяють розробникам взаємодіяти з мережевим сервером різними способами. Платформа складається з декількох складових частин, включаючи ChirpStack Network Server, ChirpStack Application Server та ChirpStack Gateway Bridge.

ChirpStack Network Server - це сервер, який управляє мережею LoRaWAN, приймає та обробляє дані від IoT-пристроїв та передає їх до ChirpStack Application Server.

ChirpStack Application Server - це сервер, який дозволяє розробникам обробляти та аналізувати дані, що надходять від IoT-пристроїв, виконувати операції на даних та керувати IoT-пристроями.

ChirpStack Gateway Bridge - це компонент, який дозволяє ChirpStack Network Server спілкуватися з мережами воріт (gateway), що використовуються для забезпечення зв'язку між IoT-пристроями та сервером.

За допомогою ChirpStack можна легко створювати та керувати мережами LoRaWAN, збирати та аналізувати дані від IoT-пристроїв та виконувати інші операції, пов'язані з IoT [28].

3.2.2 EMQX (MQTT Broker)

У якості центрального брокеру повідомлень в системі моніторингової мережі було обрано EMQX брокер. EMQ X - це відкрите програмне забезпечення для повідомлень, яке використовується для розробки та розгортання масштабних систем Інтернету речей (IoT) та миттєвого обміну повідомленнями. EMQ X підтримує більше 10 протоколів підключення, включаючи MQTT, MQTT-SN, CoAP та LwM2M, що робить його одним з найбільш універсальних рішень для розробки IoT-систем [29].

EMQ X забезпечує масштабованість, високу доступність та стійкість до відмов. Він може обробляти мільйони одночасних з'єднань та мільярди повідомлень на день. Більше того, він забезпечує підтримку кластеризації, що дозволяє розгортати системи з великою кількістю вузлів та розділяти навантаження між ними [30].

EMQ X також має велику кількість функцій, включаючи можливість налаштування даних користувачів, підтримку SSL / TLS, можливість налаштування прав доступу до каналів, збереження повідомлень у разі відключення та багато іншого. Крім того, EMQ X надає можливість розробникам розширювати та налаштовувати систему з використанням плагінів та API.

Ось декілька переваг використання EMQX в моніторингових LoRaWAN мережах [31]:

– Масштабованість: EMQ X може масштабуватися для підтримки тисяч пристроїв IoT, що дозволяє створювати великі мережі LoRaWAN та моніторити їх з однієї центральної точки.

– Висока продуктивність: EMQ X підтримує швидку обробку даних та високу продуктивність, що дозволяє моніторити великі мережі з високим трафіком.

– Підтримка протоколу LoRaWAN: EMQ X підтримує протокол LoRaWAN, що дозволяє збирати дані від пристроїв, підключених до мережі LoRaWAN.

– Гнучкість: EMQ X дозволяє налаштовувати мережі LoRaWAN залежно від вимог конкретної задачі та рівня безпеки, що необхідний для конкретного проекту.

– Підтримка різних пристроїв: EMQ X підтримує більше 10 протоколів підключення, що дозволяє збирати дані з різних типів пристроїв, підключених до мережі.

– Безпека: EMQ X забезпечує високий рівень безпеки для моніторингових мереж LoRaWAN, включаючи шифрування трафіку та контроль доступу до мережі.

– Легка інтеграція: EMQ X легко інтегрується з іншими системами та платформами для збору та аналізу даних, що дозволяє легко і швидко створювати повноцінні IoT-рішення.

EMQ X широко використовується в багатьох великих проектах Інтернету речей, таких як віддалений моніторинг та управління університетським кампусом, системи смарт-електромережі, системи контролю якості повітря та багато інших. Він також має досить добру документацію та спільноту розробників, які готові надавати допомогу в розв'язанні проблем та налаштуванні системи.

3.2.3 InfluxDB (Time series database)

У якості бази даних часових рядів для збереження сирих користуватскьих даних (індексу потенційного зараження та геолокації) використовувалась база даних InfluxDB. InfluxDB - це база даних часових рядів з відкритим вихідним кодом, яка була розроблена спеціально для зберігання, пошуку та аналізу даних, що змінюються з часом. Її основною метою є зберігання інформації про велику кількість часових рядів даних, які збираються в реальному часі від різних джерел.

InfluxDB використовує власну мову запитів InfluxQL для взаємодії з даними. Ця мова спрощує пошук даних за часом та іншими параметрами, що змінюються з часом. База даних також може зберігати дані в форматі JSON і пропонує HTTP API для звернення до даних [32].

Одна з головних переваг InfluxDB полягає в її високій продуктивності при роботі з часовими рядами. База даних розроблена таким чином, щоб забезпечити швидкий доступ до даних, що змінюються з часом. Крім того, InfluxDB пропонує можливості реплікації даних для забезпечення надійності та зберігання даних у випадку відмови системи [33].

InfluxDB може бути використаний для зберігання та обробки даних з підвищеними потребами щодо приватності користувачів. Деякі з переваг використання InfluxDB у таких застосунках включають [34]:

– Відкритий вихідний код: InfluxDB має відкритий вихідний код, що означає, що ви можете перевірити, як саме він збирає та обробляє дані. Це забезпечує більшу прозорість в роботі з даними та сприяє відповідності з нормативними вимогами щодо захисту персональних даних.

– Захист даних: InfluxDB забезпечує шифрування даних у стані спокою та під час передачі між вузлами кластера. Крім того, вона має вбудовану систему автентифікації та авторизації користувачів, що забезпечує контроль доступу до даних та забезпечує захист від несанкціонованого доступу.

– Масштабованість: InfluxDB може бути масштабована для забезпечення роботи з великою кількістю даних, що змінюються з часом. Вона може бути використана як в окремій машині, так і в розподіленому кластері, що забезпечує підтримку великих обсягів даних та підвищену доступність.

– Можливості аналітики: InfluxDB має вбудовані можливості аналізу даних, такі як агрегація даних за різними періодами часу та статистичні обчислення. Це дозволяє здійснювати аналіз даних в реальному часі та швидко реагувати на події.

InfluxDB використовується в багатьох великих проектах з IoT, моніторингу та аналізу даних, що змінюються з часом. Також вона є однією з найпопулярніших баз даних для моніторингу систем, які використовуються в хмарних сервісах.

3.2.4 MongoDB (Document database)

Для збереження інформації, щодо зон потенційної небезпеки у вигляді документів у системі використовується база даних MongoDB. MongoDB - це нереляційна база даних з відкритим вихідним кодом, яка зберігає дані у форматі JSON-подібних документів. MongoDB створена для зберігання та роботи з

документами у великому масштабі, і має розподілену архітектуру, що дозволяє масштабувати роботу з базою даних горизонтально.

Основні характеристики MongoDB [35]:

– Схема-менше: MongoDB не має фіксованої схеми, що дозволяє зберігати документи різної структури в одній колекції. Це дає можливість додавати поля та оновлювати документи без необхідності модифікувати структуру бази даних.

– Гнучкі запити: MongoDB дозволяє створювати складні запити з використанням декількох параметрів, наприклад, діапазонів, регулярних виразів та агрегації.

– Масштабованість: MongoDB підтримує розподілені бази даних та може бути масштабована горизонтально за допомогою класування та реплікації.

– Висока продуктивність: MongoDB дозволяє виконувати запити до десятків мільйонів документів за декілька мілісекунд.

MongoDB є гнучкою базою даних з JSON-подібним форматом даних, що дозволяє зберігати та обробляти дані більш ефективно. Вона має вбудований механізм реплікації та шардування даних, який забезпечує високу доступність та масштабованість системи.

Окрім цього, MongoDB має вбудовану підтримку геопросторових запитів, що дозволяє ефективно зберігати та запитувати геодані. Вона також має велику та активну спільноту розробників, яка створює різноманітні додатки та інструменти для підтримки та розширення бази даних. MongoDB використовується у багатьох великих інтернет-проектах, таких як Google, Adobe, eBay, Cisco, та багатьох інших. Вона також є однією з найпопулярніших баз даних для розробки сучасних веб-додатків та мобільних додатків [36].

MongoDB є популярним вибором для зберігання даних IoT через його ряд переваг [37]:

– Гнучкість: MongoDB має гнучкий JSON-подібний формат даних, що дозволяє легко зберігати дані в різних форматах та розмірностях.

– Масштабованість: MongoDB має вбудовані механізми шардування та реплікації, що дозволяє масштабувати базу даних при збільшенні обсягів даних.

– Швидкість: MongoDB працює дуже швидко та ефективно завдяки використанню індексів, які дозволяють ефективно шукати та фільтрувати дані.

– Геопросторові запити: MongoDB має вбудовану підтримку геопросторових запитів, що дозволяє ефективно зберігати та запитувати геодані.

– Гнучкий API: MongoDB має гнучкий API, який дозволяє легко взаємодіяти з базою даних з будь-якої мови програмування.

– Підтримка ACID-транзакцій: MongoDB підтримує ACID-транзакції, що забезпечує цілісність та стійкість даних в системі.

– Легкість використання: MongoDB має простий та зрозумілий інтерфейс користувача, що дозволяє швидко та ефективно взаємодіяти з базою даних.

– Велика спільнота розробників: MongoDB має велику та активну спільноту розробників, що створює різноманітні додатки та інструменти для підтримки та розширення бази даних.

MongoDB підтримує різноманітні операції з даними, такі як вставка, оновлення, видалення та запити до документів. Крім того, MongoDB має широкий спектр інструментів для розробників, таких як драйвери для різних мов програмування, засоби адміністрування та моніторингу, а також інструменти для розгортання та автоматизації.

3.2.5 Мови програмування для написання сервісів, прошивок та застосунків

Для написання прошивок та сервісів обробки даних була використана мова програмування Rust. Для браслету була написана прошивка для bare-metal. Rust - це системна мова програмування, що була розроблена Mozilla Research. Вона комбінує в собі властивості системних мов, таких як C і C++, із сучасними конструкціями мов програмування вищого рівня. Rust надає програмістам

можливість писати ефективний код, який має низьку вартість абстракції, що забезпечує високу продуктивність і мінімізує можливість помилок.

Rust є мовою програмування, яка надає кілька переваг для розробки IoT wearable пристроїв:

– Безпека: Rust дозволяє уникнути багатьох типів програмних помилок, таких як витіки пам'яті, дефектні вказівники, збої програми та багато інших. Це особливо важливо для IoT пристроїв, оскільки вони мають обмежені ресурси та можуть бути піддані атакам, які можуть використовувати програмні помилки.

– Швидкість та ефективність: Rust є дуже швидкою та ефективною мовою програмування, що дозволяє запускати програми на низьких рівнях системи. Це забезпечує зменшення споживання енергії та збільшення продуктивності, що особливо важливо для пристроїв з обмеженими ресурсами.

– Простота у використанні: Rust має досить простий та зрозумілий синтаксис, що дозволяє швидко створювати функціональні програми. Крім того, Rust надає кілька інструментів, які спрощують розробку програм, зокрема систему пакетного менеджера Cargo.

– Підтримка платформи: Rust підтримує багато платформ, такі як ARM, AVR, x86, x86_64 та багато інших. Це дозволяє розробникам легко переносити свої програми на різні платформи та апаратні засоби.

Узагальнюючи, Rust є гарним вибором для розробки IoT wearable пристроїв, які потребують високої швидкості, ефективності та безпеки.

Крім того, Rust має вбудовану підтримку багатьох парадигм програмування, включаючи функційний та об'єктно-орієнтований підходи. Це дозволяє розробникам створювати гнучкі та легко збережені програми, що можуть виконуватися на різних платформах.

Також варто зазначити, що Rust має вбудовану підтримку багатьох інструментів, що полегшують процес розробки. Наприклад, у Rust є менеджер пакетів Cargo, що дозволяє легко встановлювати та управляти залежностями, а також забезпечує автоматичну генерацію документації [38].

Однією з ключових переваг Rust є його швидкодія. Rust компілюється у нативний машинний код, що дозволяє йому працювати на високій швидкості. Це робить Rust ідеальним вибором для розробки високопродуктивних додатків, включаючи IoT рішення.

Для сервісу оповіщень було обрано мову програмування Gleam, яка інтерпретується на віртуальній машині Beam. Gleam - це нова функціональна мова програмування, яка побудована на основі Erlang VM (BEAM), та має на меті забезпечити більш високий рівень безпеки та довіру до програм, що пишуться для платформи Erlang/OTP. Головною метою Gleam є спрощення розробки надійних, масштабованих систем та ділитися можливостями програмних продуктів.

Однією з особливостей Gleam є сильна статична типізація, що дозволяє забезпечити безпеку під час компіляції. Це означає, що ви можете бути впевнені, що ваш код не буде містити помилок типізації. Крім того, це забезпечує більшу надійність та швидкодію у порівнянні з динамічними мовами програмування, такими як Python та Ruby.

Gleam підтримує багато функцій, які роблять його привабливим для програмування IoT-пристроїв та інших розподілених систем. Наприклад, мова має багато вбудованих функцій для маніпулювання та опрацювання списків, кортежів та інших типів даних. Можливість збільшити швидкодію та ефективність за допомогою спеціальних функцій, які дозволяють виконувати операції безпосередньо на рівні байткоду.

Gleam також має дуже чистий синтаксис, який дозволяє легко читати та зрозуміти код. Багато конструкцій мови було запозичено з інших функціональних мов програмування, таких як Haskell та Elm, що забезпечує зручність та зрозумілість коду. Крім того, Gleam може легко інтегруватися з Erlang та іншими мовами, що дає змогу використовувати вже наявний код та додаткові бібліотеки.

Також Gleam має вбудовану підтримку віддаленого виконання коду (RPC), що дозволяє взаємодіяти з іншими процесами та вузлами з максимальною ефективністю.

Завдяки зручному синтаксису, статичній типізації та підтримці функцій вищого порядку, Gleam дозволяє швидко розробляти безпечний та стабільний код з мінімальною кількістю помилок. Це робить мову привабливою для розробників, які працюють з великими системами та додатками з високими вимогами до надійності та ефективності.

Крім того, Gleam надає можливість використовувати багатопотоковість та розподілений підхід для роботи з багатоядерними та розподіленими системами. Це дозволяє створювати розподілені системи з високою масштабованістю та продуктивністю.

Загалом, Gleam є потужним та простим інструментом для розробки різноманітних додатків та систем з високими вимогами до надійності та ефективності, а також з високими потребами до захисту даних та приватності.

Мова програмування Gleam має декілька переваг для розподілених систем оповіщення. Ось декілька з них [39]:

– Безпека: Gleam є мовою з сильною типізацією та безпечним виконанням, що дозволяє зменшити кількість помилок під час виконання програми. Це особливо важливо для розподілених систем, де помилки можуть призвести до серйозних проблем.

– Розширюваність: Gleam базується на Erlang VM, що дозволяє легко розширювати систему. Erlang VM забезпечує високу доступність та розподілену обробку даних, що є ключовими вимогами для систем оповіщення.

– Швидкість: Gleam має компілятор, що дозволяє генерувати швидкий та ефективний код, що дозволяє оптимально використовувати ресурси машини та швидко оброблювати великі обсяги даних.

– Масштабованість: Gleam дозволяє легко створювати розподілені системи, що можуть легко масштабуватися, використовуючи високоякісні механізми кластеризації, що забезпечують високу доступність та стійкість до збоїв.

– Легка інтеграція з іншими мовами: Gleam може легко інтегруватися з Erlang та іншими мовами, що дає змогу використовувати вже існуючі рішення та бібліотеки для створення систем оповіщення.

Застосунок написаний на мові програмування Dart за допомогою фреймворку Flutter. Flutter - це відкрите програмне забезпечення з відкритим вихідним кодом, створене компанією Google для розробки мобільних та веб-додатків з використанням одного кодової бази. Він включає в себе віджети, бібліотеки та інструменти для розробки графічного інтерфейсу користувача, а також платформу розробки на мові Dart.

Flutter використовує швидкий двигун Skia для візуалізації графічних елементів та віджетів і виконується на віртуальній машині Dart. Використання одного коду для декількох платформ дозволяє розробникам ефективно створювати додатки для Android, iOS, web та desktop.

Однією з головних переваг Flutter є гнучкість в розробці. Він має багато різноманітних віджетів та компонентів, які можна легко комбінувати, що дозволяє розробникам створювати користувацькі інтерфейси з високим рівнем специфікації. Крім того, Flutter надає багато вбудованих інструментів для тестування, дебагу та оптимізації додатків, що допомагає розробникам забезпечувати високу якість своїх додатків.

Flutter також дозволяє розробникам швидко та легко реалізовувати віджети анімації та взаємодії з користувачем, що робить його дуже привабливим для розробки додатків з багатим інтерфейсом користувача. Він також підтримує гарний рівень переносимості між платформами та широку підтримку спільноти, що дозволяє розробникам отримувати підтримку та допомогу від інших учасників [40].

Flutter - це відкрита платформа розробки мобільних додатків, що базується на мові програмування Dart. Flutter має декілька переваг для розробки Contact Tracing App з використанням IoT мережі на LoRaWAN:

– Швидкість розробки: Flutter надає готовий набір віджетів і бібліотек, які дозволяють швидко розробляти інтерактивні додатки з високоякісним інтерфейсом користувача.

– Кросплатформеність: з допомогою Flutter можна створювати додатки для Android та iOS. Це дозволяє розробникам використовувати один і той же код для створення додатків для різних платформ.

– Легкість розгортання: Flutter дозволяє легко розгортати додатки на різних платформах. Крім того, додатки, розроблені на Flutter, мають невеликий розмір, що дозволяє їх швидко завантажувати та виконувати.

– Висока продуктивність: Flutter має вбудовану підтримку гарячої перезавантаження, що дозволяє розробникам швидко вносити зміни та перевіряти їх в реальному часі. Крім того, Flutter має високу продуктивність завдяки використанню власного движка рендерингу.

– Підтримка IoT мережі LoRaWAN: Flutter дозволяє легко інтегруватися з IoT мережею LoRaWAN за допомогою бібліотек, таких як flutter_blue і flutter_ble_libs. Це дозволяє розробникам підключати пристрої IoT до своїх додатків та обмінюватися даними між ними.

Отже, Flutter має декілька переваг для розробки Contact Tracing App з використанням IoT мережі на LoRaWAN. Він забезпечує швидкість розробки, кросплатформеність та легкість розгортання.

3.3 Огляд серверних компонентів хмарної архітектури

Для створення серверної частини хмарної архітектури на AWS були обрані такі типи віртуальних машин EC2: C6gn, I4i, T3, Im4gn та M6g різної величини. Всього за ресурсами в залежності від навантаження екземпляри обиралися large або xlarge, тільки у випадку екземпляру T3 був обраний розмір small так як даний вузол має містити систему не вибагливу до навантажень. Повний перелік екземплярів віртуальних машин з невеликим описом, особливостями та технічними характеристиками зображено на рис. 3.5.

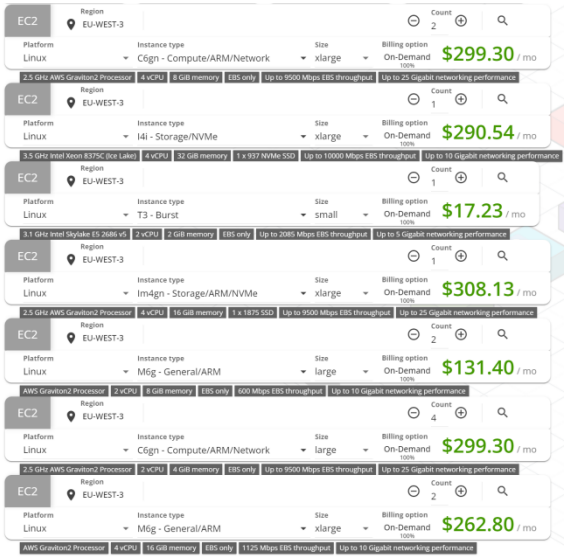


Рисунок 3.5 – Перелік використаних екземплярів EC2 в хмарній архітектурі

Amazon Elastic Compute Cloud (EC2) - це хмарна послуга, що надає можливість орендувати віртуальні сервери з можливістю масштабування під різні потреби користувачів. В рамках даної послуги доступні різні типи інстансів, які відрізняються обсягом ресурсів, швидкістю та продуктивністю.

В системі використовувались саме найчастіше два типи інстансів за об'ємом, а саме:

– Large: Інстанси з цим префіксом мають менше ресурсів, ніж інстанси з префіксом xlarge. Наприклад, c4.large має 2 ядра процесора та 3,75 Гб оперативної пам'яті.

– Xlarge: Інстанси з цим префіксом мають більше ресурсів, ніж інстанси з префіксом large. Наприклад, c4.xlarge має 4 ядра процесора та 7,5 Гб оперативної пам'яті.

Різниця між large та xlarge EC2 інстансами полягає у кількості ресурсів, таких як кількість ядер процесора, обсяг оперативної та сховищевої пам'яті, максимальний обсяг сховища та інші параметри, що визначають їх продуктивність. Зазвичай, xlarge інстанси є потужнішими та забезпечують більшу продуктивність, але і коштують дорожче за рахунок більшого обсягу ресурсів. Слід також зазначити, що класифікація екземплярів EC2 за ресурсами значно ширша і не обмежується типами large та xlarge.

C6gn (Compute-optimized instances with 6th generation Nitro system) є одним із класів EC2 інстансів у серії Amazon Elastic Compute Cloud (EC2). Ці інстанси були випущені Amazon Web Services для покращення можливостей обробки, зберігання та передачі даних в хмарному середовищі AWS.

C6gn інстанси є оптимізованими для обчислень і використовують процесори Intel Cascade Lake. Вони також підтримують мережевий інтерфейс Elastic Fabric Adapter (EFA), що забезпечує високопродуктивну мережеву взаємодію між екземплярами C6gn в межах однієї або більше віртуальних мереж AWS. Інші технічні характеристики C6gn інстансів включають високопродуктивні NVMe-накопичувачі та можливість забезпечення гіпервізорної ізоляції за допомогою технології AWS Nitro.

Основними перевагами використання C6gn інстансів є висока продуктивність обчислень, ефективність передачі даних та здатність до інтеграції з Elastic Fabric Adapter (EFA), що дозволяє підвищити продуктивність мережі між екземплярами. Крім того, C6gn інстанси можуть бути використані для великої кількості завдань, включаючи інтенсивні обчислення, машинне навчання, аналіз даних та інші завдання, які вимагають багато ресурсів обчислень та мережі.

I4i - це один з типів інстансів Amazon Elastic Compute Cloud (EC2), що пропонується Amazon Web Services (AWS). Цей інстанс призначений для роботи з високопродуктивними обчислювальними завданнями та іншими завданнями, які потребують великої кількості обчислювальних ресурсів.

I4i інстанси використовують процесори Intel Xeon Platinum або Gold, а також мають високопродуктивні сховища SSD. Вони мають від 16 до 96 віртуальних процесорів, від 122 до 768 гігабайт оперативної пам'яті та можуть пропонувати до 9,6 мільйонів випадків вводу-виводу за секунду (IOPS) на один екземпляр.

Основні переваги I4i інстансів EC2 включають високу продуктивність та масштабованість, можливість вибору операційної системи та налаштування різних типів сховищ даних, таких як SSD. Ці інстанси також можуть бути

використані для широкого спектру завдань, від високопродуктивних обчислювань до машинного навчання та аналізу даних.

T3-інстанс (T3 Instance) є одним з типів віртуальних серверів (EC2 Instances), які надаються Amazon Web Services (AWS). Ці інстанси пропонуються високопродуктивні та економічно ефективні рішення для обчислювальних завдань з невеликими і середніми вимогами до процесорних ресурсів.

T3-інстанси працюють на базі процесорів Intel Xeon Platinum 8000 серії з підтримкою Hyper-Threading технології. Вони мають динамічну многопотокову обробку, що дає можливість ефективно використовувати процесорні ресурси при зміні навантаження. Крім того, T3-інстанси підтримують технологію Intel Turbo Boost, яка автоматично підвищує частоту процесора в залежності від навантаження, що забезпечує підвищену продуктивність.

T3-інстанси мають можливість налаштування мережевих параметрів, в тому числі регулювання пропускної здатності мережі та використання Elastic Network Adapter для оптимізації мережевої продуктивності.

Однією з головних переваг T3-інстансів є їх економічність. Вони підтримують безкоштовну можливість використання T3.micro-інстансів протягом 12 місяців. Крім того, вартість використання T3-інстансів менша порівняно з іншими типами EC2-інстансів.

У загальному, T3-інстанси забезпечують високу продуктивність та економічність, що робить їх хорошим варіантом для різноманітних робіт з обробки даних, використання веб-серверів та іншого ПЗ. Вони також можуть бути використані в багатьох інших застосунках, де потрібна достатня продуктивність, але не настільки висока, щоб виправдати використання більш потужних інстансів. Завдяки своїй економічності, T3-інстанси дозволяють значно знизити витрати на хостинг, що робить їх особливо привабливими для малих та середніх підприємств, які мають обмежений бюджет.

Однак, варто мати на увазі, що T3-інстанси мають деякі обмеження, такі як обмежений об'єм оперативної пам'яті, обмеження на час виконання, а також

обмеження на введення/виведення даних. Ці обмеження можуть стати перешкодою для деяких застосувань з великим обсягом даних або потребують високої продуктивності. У цьому випадку, може знадобитися використання більш потужних інстансів EC2, таких як C6gn або I4i, залежно від потреб застосунку.

Amazon Elastic Compute Cloud (EC2) Im4gn інстанси є одними з найпотужніших інстансів серії EC2. Вони призначені для виконання високопродуктивних завдань, які вимагають великої кількості процесорних ресурсів та оперативної пам'яті.

Im4gn інстанси мають до 48 процесорів Intel Xeon Platinum 8000 серії і 384 ГБ оперативної пам'яті. Крім того, вони оснащені технологією NVMe-дискового простору, що забезпечує високу продуктивність вводу-виводу для роботи з великими обсягами даних.

Im4gn інстанси можуть бути використані для високопродуктивних завдань, таких як аналіз даних, машинне навчання, високопродуктивні бази даних та інші завдання, які вимагають багато ресурсів обчислень. Оскільки вони мають високу продуктивність та багато ресурсів, ці інстанси можуть бути вартісним вибором для організацій, які працюють з великими обсягами даних та вимогливими до продуктивності застосунками.

M6g є екземпляром Amazon Elastic Compute Cloud (EC2), який базується на процесорах Graviton2 від Amazon Web Services (AWS) і призначений для роботи з високопродуктивними та вимогливими до ресурсів додатками, такими як великі бази даних, веб-сервери та інші застосунки. Ці інстанси пропонують до 40% більше продуктивності в порівнянні з попередньою поколіннями інстансів M5, а також до 50% економії витрат на збереження та обробку даних. M6g є хорошим варіантом для додатків з високими вимогами до продуктивності та масштабованості.

3.4 Перший прототип та кошторис.

Для розгортання першого робочого прототипу системи моніторингової мережі на основі протоколів LoRaWAN та MQTT було обрано місто Тулон, Франція. Основна орієнтація та направлення розробки з самого початку була спрямована на європейській ринок та при подальшій розробці та розгортанні повинна буде відповідати усім європейським стандартам та повістю відповідати вимогам GDPR. Проєктувалась система спираючись на вимоги GDPR та на даний момент повністю їм відповідає. Місто Тулон в Франції було обране саме через ідеальне співвідношення кількості людей до їх щільності. При виборі початкового міста головними критеріями були невелика кількість населення, для економії витрат для побудови першого робочого прототипу, а також велика щільність населення, для зручного тестування працездатності системи в урбаністичних умовах. Населення Тулону складає приблизно 176200, а щільність 4200\км2 [41], що є досить оптимальним в даному випадку.

Отримавши відповідні дані, щодо початкових умов розгортання, приблизної складності створення системи та інших зовнішніх параметрів було сформовано кошторис для створення першого прототипу моніторингової мережі:

Таблиця 3.2 – Кошторис реалізації та розгортання моніторингової мережі у місті Тулон, Франція

Назва	Час (Місяць)	Вартість ($)
Додатковий аналіз ринку	1-3	~15,000
Апаратна частина	6-12	~57,073,271
UX/UI Дизайн	1-2	~3,000
Програмна розробка системи	8-15	~1,500,000
Розробка мобільного застосунку	4-8	~150,000
Хмарні обчислення	8 (в експлуатації)	~12,856
Отримання ліцензій та юридичне оформлення	4-10	~50,000
Розгортання системи	2-6	~400,000

Маркетинг	3-5	~100,000
Всього	8-24	~59,304,127

Кошторис складається з багатьох компонентів і є лише попередньо оцінним і відображає терміни та витрати лише приблизно. Кошторис можна розділити на декілька важливих компонентів: розробка мережі, розробка застосунку, експлантація, маркетинг та юридичні витрати. Більше всього ресурсів витрачається на створення апаратної частини, так як туди входить закупівля необхідних компонентів, замовлення друкованих плат та збір усіх необхідних компонентів в кінцеві пристрої. Кінцевих пристроїв має вистачити на 60% населення міста Тулон, що є оптимальним порогом злагодженої роботи системи. Також до апаратної частини відноситься закупівля та розгортання шлюзів LoRaWAN, усього для міста Тулон з розрахунком на особливості шлюзу (радіусу роботи в урбаністичних умовах, для обраного шлюзу Kerlink Wirnet iBTS даний параметр становить 2 км) та ландшафтних, географічних особливостей міста Тулон необхідно 21 одиниця екземплярів. На другому місці по витратам становить розробка програмного забезпечення для системи моніторингу, так як для розробки буде залучено декілька команд спеціалістів для розробки окремих компонентів системи. Останнім кроком було створено концептуальну модель робочого застосунку, яку можна спостерігати на рис. 3.6.

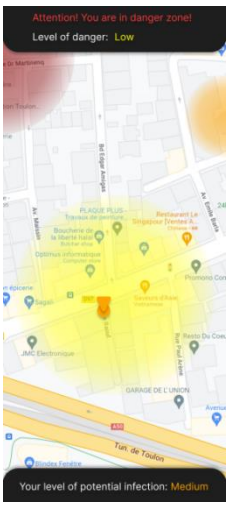


Рисунок 3.6 – Концептуальний макет мобільного застосунку системи

Макет застосунку розроблявся з урахуванням зручності користування та легкого надання інформації. Головною вимогою при створенні макету було не перевантаження інтерфейсу системи та лаконічне відображення лише головної інформації на екрані, важлива інформація повинна мати найвищій пріоритет в системі. Зображений інтерфейс застосунку не є остаточним та у майбутньому буде перероблений відділом UX\UI дизайну.

Висновки до розділу 3

У даному розділі було детально проаналізовано апаратні та програмні компоненти системи моніторингу, зокрема були розглянуті переваги використання певних технологій та пристроїв з урахуванням особливостей та вимог системи. Були проведені дослідження та порівняння різних протоколів та пристроїв, що можуть бути використані для забезпечення ефективної роботи системи.

Наступним етапом було проаналізовано компоненти хмарної архітектури, описано їхню функціональність та взаємодію з іншими компонентами системи. Було обране місто для розгортання першого прототипу, з урахуванням його географічного положення та наявності необхідної інфраструктури. Також були сформовані кошториси для реалізації системи, в яких були враховані всі складові витрат на придбання технічних засобів, програмного забезпечення, встановлення та обслуговування системи.

Окремо було розроблено макет користувацького інтерфейсу системи, що дозволило ознайомитись з функціональністю та зручністю використання системи для кінцевих користувачів. Були враховані рекомендації та вимоги до дизайну інтерфейсу, зокрема його зручність, зрозумілість та ергономічність.

ВИСНОВКИ

Першим кроком даної роботи було виявлення сучасних проблем контролю розповсюдження епідемій, визначення актуальності проблеми. Наступним етапом було запропоновано рішення, визначено наукову новизну та шляхи вирішення проблеми. Після цього були сформовані завдання задля досягнення поставленої мети. За мету було взято вирішення проблеми контролю скупчень людей для запобігання розповсюдження епідемії з урахуванням потреб приватності та свободи переміщення осіб шляхом проєктування сучасної GDPR відповідної системи моніторингової мережі на основі модулів LoRaWAN та протоколу MQTT.

Відповідно до завдання спочатку було проаналізовано сучасні рішення моніторингових мереж та виявлені їх основні особливості, недоліки та переваги. Далі був проведений аналіз аналогічних та запатентованих рішень, були виділені основні проблеми та недоліки відповідно до сучасних вимог та визначені шляхи вдосконалення та покращення систем. На основі зібраних даних були закладені основи для для розроблювальної системи, а також визначені вимоги.

Основним етапом було проектування системи моніторингової мережі. Під час проектування було створена та проаналізована концептуальна схема роботи мережі, створені моделі та схеми зберігання даних відповідно до вимог GDPR, визначені аспекти взаємодії компонентів через діаграму послідовності, сформульовані математичні моделі знаходження індексів потенційного зараження користувача та потенційної небезпеки зони, розроблені апаратні та програмні частини системи, а також спроєктована та оптимізована хмарна архітектура на базі сервісів AWS.

Кінцевим кроком були детально розглянуті апаратні та програмні компоненти, приведені переваги використання відносно проєктованої системи, детально наведені характеристики систем. Також в третьому розділі було підібрано місто для розгортання першого прототипу системи, а також

розроблений план розгортання в місті. На основі плану був сформований кошторис з визначенням часових та грошових витрат на реалізацію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Shahroz, M., Ahmad, F., Younis, M. S., та ін. COVID-19 digital contact tracing applications and techniques: A review post initial deployments. Transportation Engineering. 2021. Vol. 5. С. 100072.

2. Shen, J., Duan, H., Zhang, B., та ін. Prevention and control of COVID-19 in public transportation: Experience from China. Environmental Pollution. 2020. Vol. 266. С. 115291.

3. Garg, L., Chukwu, E., Nasser, N., та ін. Anonymity Preserving IoT-Based COVID-19 and Other Infectious Disease Contact Tracing Model. IEEE Access. 2020. Vol. 8. С. 159402–159414.

4. Drew Donnelly, P. China Social Credit System Explained – What is it & How Does it Work? URL: https://nhglobalpartners.com/china-social-credit-system-explained (дата звернення: 20.12.22).

5. Vincent Ni and Yitsing Wang. How to "disappear" on Happiness Avenue in Beijing: URL: https://www.bbc.com/news/technology-55053978 (дата звернення: 20.12.22).

6. White, G., Zink, A., Codecá, L., та ін. A digital twin smart city for citizen feedback. Cities. 2021. Vol. 110. С. 103064.

7. Aaron Aupperlee. CMU, Fujitsu Collaborate To Develop Social Digital Twin Technology for Smart Cities: URL: https://www.cmu.edu/news/stories/archives/2022/february/digital-twin-technology (дата звернення: 20.12.22).

8. Aaron Aupperlee. Fujitsu, Carnegie Mellon University Collaborate to Develop 'Social Digital Twin' Technology for Smart Cities. URL: https://www.fujitsu.com/global/about/resources/news/press-releases/2022/0208-01.html (дата звернення: 20.12.22).

9. G.Naga Rama Devi, Sivakumar Ponnusamy, Jyotsna Pandit, та ін. Development Of Medicinal Industries In Building A Replica To The Damaged Human Tissue For Artificial Organs With The Application Of Micro- And Nano Technology (Mnt). Journal of Optoelectronics Laser. 2022. Vol. 41, No. 3. С. 79–83.

10. Satyanarayana, T. V. V., Mohana Roopa, Y., Maheswari, M., та ін. A secured IoT-based model for human health through sensor data. Measurement: Sensors. 2022. Vol. 24. С. 100516.

11. Ullah, I., Amin, N. U., Khan, M. A., та ін. An Efficient and Provable Secure Certificate-Based Combined Signature, Encryption and Signcryption Scheme for Internet of Things (IoT) in Mobile Health (M-Health) System. Journal of Medical Systems. 2021. Vol. 45, No. 1. С. 4.

12. Rehman, A., Haseeb, K., Saba, T., та ін. Secured Big Data Analytics for Decision-Oriented Medical System Using Internet of Things. Electronics. 2021. Vol. 10, No. 11. С. 1273.

13. Malcolm Tatum. What is Corporate Surveillance? URL: https://www.smartcapitalmind.com/what-is-corporate-surveillance.html (дата звернення: 20.12.22).

14. Health monitoring system for limiting the spread of an infection in an organization: пат. US20210296008A1 United States. Jerome J. Novak, Jr., Omar Ahmed, та ін.; опубл 2021.

15. Detecting COVID-19 using surrogates: пат. US10902955B1 United States. Howard Federoff, Ophir Frieder; опубл 2020.

16. Trace system for Infectious people and trace method using it: пат. KR102399702B1 South Korea. Hyunwook Lee; опубл 2020.

17. Kalman Filtering: A Simple Introduction. URL: https://towardsdatascience.com/kalman-filtering-a-simple-introduction-df9a84307add (дата звернення: 10.01.23)

18. DBSCAN Clustering in ML | Density based clustering. URL: https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering (дата звернення: 10.01.23)

19. Wu, G., Cao, L., Tian, H., та ін. HY-DBSCAN: A hybrid parallel DBSCAN clustering algorithm scalable on distributed-memory computers. Journal of Parallel and Distributed Computing. 2022. Vol. 168. С. 57–69.

20. Optimising for Concurrency: Comparing and contrasting the BEAM and JVM virtual machines. URL: https://www.erlang-solutions.com/blog/optimising-for-concurrency-comparing-and-contrasting-the-beam-and-jvm-virtual-machines (дата звернення: 12.01.23)

21. Amazon EC2 Instance Types - Amazon Web Services. URL: https://aws.amazon.com/ec2/instance-types (дата звернення: 14.01.23)

22. MAXREFDES101 Health Sensor Platform 2.0 User Guide. URL: https://pdfserv.maximintegrated.com/en/an/AN6780.pdf (дата звернення: 15.01.23)

23. MAXREFDES101 | reference design | Analog Devices. URL: https://www.analog.com/en/design-center/reference-designs/maxrefdes101.html#rd-overview (дата звернення: 16.01.23)

24. Abeeway Geolocation module data-sheet. URL: https://www.abeeway.com/wp-content/uploads/2022/04/Abeeway_Geolocation-module-data-sheet_2022-v05.pdf (дата звернення: 17.01.23)

25. Abeeway Geolocation Module - Abeeway. URL: https://www.abeeway.com/abeeway-geolocation-module (дата звернення: 18.01.23)

26. WirnetTM iBTS LoRaWAN® Outdoor Gateway for the Internet of Things. URL: https://lora-alliance.org/wp-content/uploads/2019/09/Commercial_Leaflet_Wirnet_iBTS_2019-1_1.pdf (дата звернення: 18.01.23)

27. Kerlink Wirnet iBTS Compact Outdoor Gateway (1 LOC/ 1 WAN) - ThingPark Market. URL: https://market.thingpark.com/kerlink-compact-ibts.html (дата звернення: 19.01.23)

28. ChirpStack, open-source LoRaWAN® Network Server. URL: https://www.chirpstack.io (дата звернення: 19.01.23)

29. Protocol | EMQX 3.0 Documentation. URL: https://www.emqx.io/docs/en/v3.0/protocol.html (дата звернення: 20.01.23)

30. EMQX: The World's #1 Open Source Distributed MQTT Broker. URL: https://www.emqx.io (дата звернення: 20.01.23)

31. Project Roadmap: MQTT 5.0, CoAP/LwM2M, LoraWAN supported; emqtt->emqx and more. URL: https://github.com/emqx/emqx/issues/1659 (дата звернення: 20.01.23)

32. Get started with InfluxDB | InfluxDB OSS 2.6 Documentation. URL: https://docs.influxdata.com/influxdb/v2.6/get-started (дата звернення: 21.01.23)

33. Clustering in InfluxDB Enterprise | InfluxDB Enterprise 1.10 Documentation. URL: https://docs.influxdata.com/enterprise_influxdb/v1.10/concepts/clustering (дата звернення: 21.01.23)

34. Manage InfluxDB security | InfluxDB OSS 1.8 Documentation. URL: https://docs.influxdata.com/influxdb/v1.8/administration/security (дата звернення: 21.01.23)

35. MongoDB Features & Key Characteristics | MongoDB | MongoDB. URL: https://www.mongodb.com/features (дата звернення: 22.01.23)

36. Use Cases | MongoDB. URL: https://www.mongodb.com/use-cases (дата звернення: 22.01.23)

37. IoT And MongoDB's Developer Data Platform | MongoDB. URL: https://www.mongodb.com/use-cases/internet-of-things (дата звернення: 22.01.23)

38. Tools - Rust Programming Language. URL: https://www.rust-lang.org/tools (дата звернення: 22.01.23)

39. Gleam. URL: https://gleam.run (дата звернення: 23.01.23)

40. Flutter - Build apps for any screen. URL: https://flutter.dev (дата звернення: 24.01.23)

41. Comparateur de territoires | Insee. URL: https://www.insee.fr/fr/statistiques/1405599?geo=UU2020-00757+COM-83137 (дата звернення: 24.01.23)

42. Шевченко А. Г., Пузирьов С. В. Проєктуваннямоніторингової мережі IoT на основі модулів LORAWAN та протоколу MQTT. Могилянські читання – 2022 : тези доп. XXV Всеукр. наук.-метод. конф. Миколаїв, 7–11 листоп. 2022 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2022. С. 117–121.

ДОДАТОК А
БЛОК-СХЕМА РОБОТИ МОНІТОРИНГОВОЇ МЕРЕЖІ

Рисунок А.1 – Блок-схема роботи моніторингової мережі

ДОДАТОК Б
ДІАГРАМА ПРОГРАМНОГО КОМПЛЕКСУ У ВИГЛЯДІ СХЕМИ

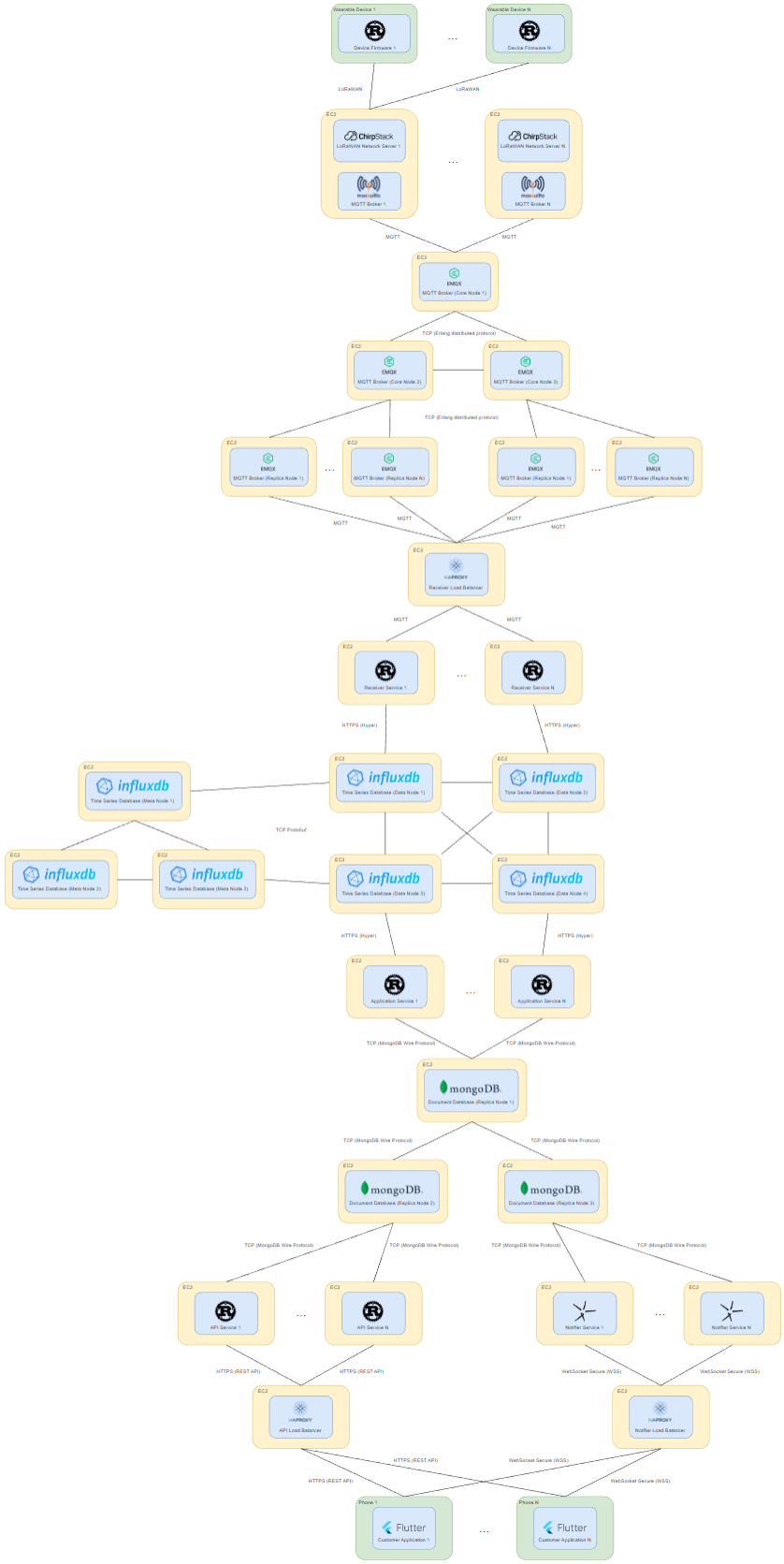


Рисунок Б.1 – Діаграма програмного комплексу у вигляді схеми

ДОДАТОК В
ЛІСТИНГ АЛГОРИТМУ ФІЛЬТРУ КАЛМАНА

```
use nalgebra::{DMatrix, DVector};

struct KalmanFilter {
    x: DVector<f64>,  // state vector
    p: DMatrix<f64>,  // state covariance matrix
    f: DMatrix<f64>,  // state transition matrix
    h: DMatrix<f64>,  // measurement matrix
    r: DMatrix<f64>,  // measurement noise covariance matrix
    q: DMatrix<f64>,  // process noise covariance matrix
}

impl KalmanFilter {
    pub fn new(x0: DVector<f64>, p0: DMatrix<f64>, f: DMatrix<f64>, h: DMatrix<f64>, r:
DMatrix<f64>, q: DMatrix<f64>) -> KalmanFilter {
        KalmanFilter {
            x: x0,
            p: p0,
            f,
            h,
            r,
            q,
        }
    }

    pub fn predict(&mut self) {
        // predict next state
        self.x = &self.f * &self.x;
        self.p = &self.f * &self.p * &self.f.transpose() + &self.q;
    }

    pub fn update(&mut self, z: &DVector<f64>) {
        // update state estimate based on measurement
        let y = z - &self.h * &self.x;
        let s = &self.h * &self.p * &self.h.transpose() + &self.r;
        let k = &self.p * &self.h.transpose() * s.try_inverse().unwrap();
        self.x = &self.x + &k * y;
        self.p = (&DMatrix::identity(self.p.nrows(), self.p.ncols()) - &k * &self.h) *
&self.p;
    }
}
```

ДОДАТОК Г
ЛІСТИНГ АЛГОРИТМУ ЕКСПОНЕНЦІЙНОГО ЗГЛАДЖУВАННЯ

```
use ndarray::{Array1, Array2};

fn exponential_smoothing(series: &Array1<f64>, alpha: f64) -> Array1<f64> {
    let n = series.len();

    // initialize smoothed series with first observation
    let mut smoothed = Array1::zeros(n);
    smoothed[0] = series[0];

    for i in 1..n {
        smoothed[i] = alpha * series[i] + (1.0 - alpha) * smoothed[i - 1];
    }

    smoothed
}
```

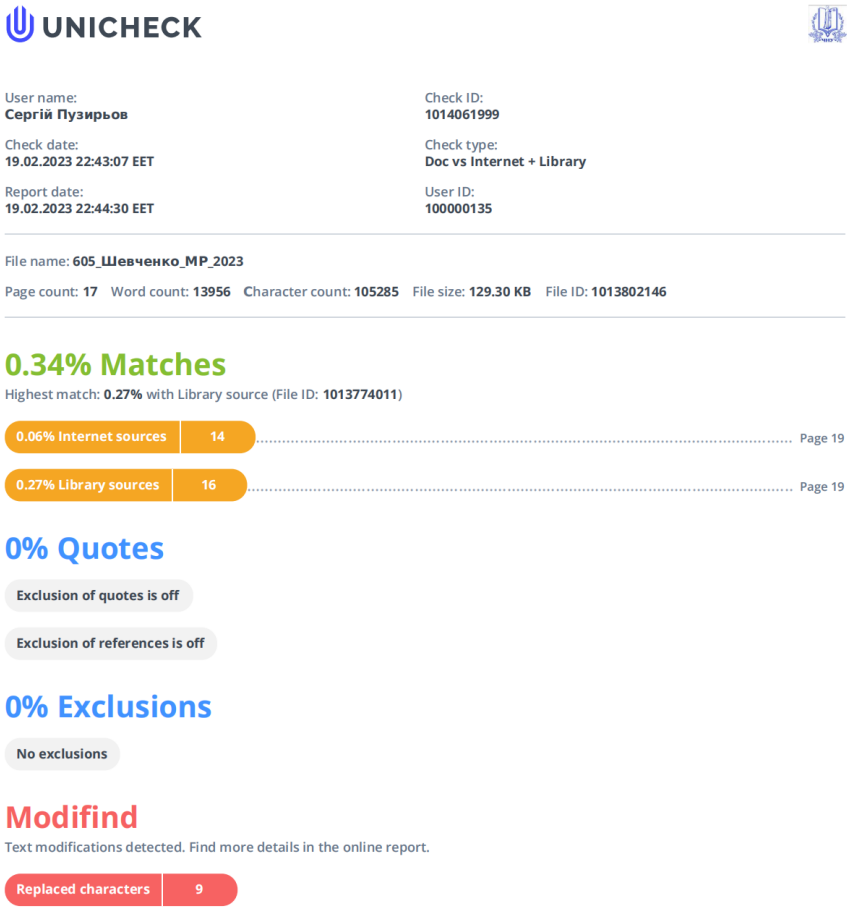
ДОДАТОК Ґ
ЛІСТИНГ АЛГОРИТМУ DBSCAN

```
use ndarray::{Array2, Axis};
use ndarray_stats::Distance;

#[derive(Debug, PartialEq)]
enum ClusterType {
    Core,
    Border,
    Noise,
}

fn dbscan(data: &Array2<f64>, eps: f64, min_pts: usize) -> Vec<usize> {
    let mut labels = vec![None; data.nrows()];

    let mut cluster_id = 0;
    for i in 0..data.nrows() {
        if labels[i].is_some() {
            continue;
        }

        let neighbors = find_neighbors(data, i, eps);
        if neighbors.len() < min_pts {
            labels[i] = Some(ClusterType::Noise);
        } else {
            cluster_id += 1;
            expand_cluster(data, i, &neighbors, eps, min_pts, cluster_id, &mut labels);
        }
    }

    labels.into_iter().map(|label| label.unwrap_or(ClusterType::Noise) as usize).collect()
}

fn find_neighbors(data: &Array2<f64>, point_idx: usize, eps: f64) -> Vec<usize> {
    let mut neighbors = vec![];
    for (i, row) in data.axis_iter(Axis(0)).enumerate() {
        if Distance::euclidean(row, &data.row(point_idx)).unwrap() <= eps {
            neighbors.push(i);
        }
    }
    neighbors
}

fn expand_cluster(
    data: &Array2<f64>,
    point_idx: usize,
    neighbors: &[usize],
    eps: f64,
    min_pts: usize,
    cluster_id: usize,
    labels: &mut Vec<Option<ClusterType>>,
) {
    labels[point_idx] = Some(ClusterType::Core);
    labels[neighbors.iter().cloned().filter(|&idx|
labels[idx].is_none()).collect::<Vec<_>>().as_slice()].iter_mut().for_each(|label| {
        let neighbors = find_neighbors(data, *label.unwrap(), eps);
        if neighbors.len() >= min_pts {
            labels[*label.unwrap()] = Some(ClusterType::Core);
            expand_cluster(data, *label.unwrap(), &neighbors, eps, min_pts, cluster_id, labels);
        } else {
            labels[*label.unwrap()] = Some(ClusterType::Border);
        }
    });
    labels[neighbors].iter_mut().for_each(|label| {
        if label.is_none() {
            labels[*label.unwrap()] = Some(ClusterType::Border);
        }
    });
}
```

ДОДАТОК Д
ДОВІДКА

про перевірку на унікальність пояснювальної записки
кваліфікаційної магістерської роботи
на тему: «Моніторингова IoT-мережа на базі LORAWAN та MQTT»
студента спеціальності 123 «Комп'ютерна інженерія», групи
605м
Шевченко Артем Геннадійович
прізвище, ім'я, по-батькові

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck.Результат перевірки тексту роботи: схожість складає 0,34%.


UNICHECK

User name: Сергій Пузирьов
Check ID: 1014061999
Check date: 19.02.2023 22:43:07 EET
Check type: Doc vs Internet + Library
Report date: 19.02.2023 22:44:30 EET
User ID: 100000135

File name: 605_Шевченко_МР_2023

Page count: 17 Word count: 13956 Character count: 105285 File size: 129.30 KB File ID: 1013802146

0.34% Matches
Highest match: 0.27% with Library source (File ID: 1013774011)

0.06% Internet sources 14 Page 19
0.27% Library sources 16 Page 19

0% Quotes
Exclusion of quotes is off
Exclusion of references is off

0% Exclusions
No exclusions

Modifind
Text modifications detected. Find more details in the online report.
Replaced characters 9

Студент
_____А. Г. Шевченко
підпис ініціали, прізвище

Керівник
канд. фіз.-мат. наук, доцент
_____С. В. Пузирьов
підпис ініціали, прізвище

Дата: «__»_____2023 р.