ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка інтегрованої smart системи попередження про повітряні тривоги, пошуку бомбосховищ за рівнями захисту та комфорту з можливістю навігації»

(тема кваліфікаційної роботи українською мовою)

«Development of an integrated smart system for warning about air alarms, searching for bomb shelters by levels of protection and comfort with the possibility of navigation»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання

спеціальності 122 Комп'ютерні науки

(код, назва спеціальності)

Освітня програма Комп'ютерні науки

(назва)

Петров Ілля Володимирович

(прізвище, ім'я, по-батькові здобувача)

Керівник асистент Молчанова А.Ю. (підпис)

(науковий ступінь, вчене звання, прізвище, ініціали)

Рецензент к.т.н., Домаскін О.М.

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .

№ 1 від 09 червня 2024 р.

Завідувачка кафедри

(підпис) КАЗАКОВА Надія .
(прізвище,ім'я)

Захищено на засіданні ЕК № 13 .
протокол № 23 від 21 червня 2024 р.

Оцінка відмінно / A / 90 .
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

(підпис) КОПИЧЕНКО Іван .
(прізвище,ім'я)

Одеса 2024

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД — база даних

ОС — операційна система

ADT — Android Development Tools — Інструменти розробки в ОС Android

ADB — Android Debug Bridge — командний інтерфейс для взаємодії з пристроями Android

API — Application Programming Interface — програмний інтерфейс програми

API сервер — сервіс який надає мережеве API для роботи з серверною логікою

IDE — Integrated Development Environment — комплексне середовище розробки програмного забезпечення

SDK — Software Development Kit — набір інструментів, бібліотек, документації та інших компонентів

GPS — Global Positioning System — глобальна система навігації

GSM — Global System for Mobile Communications — глобальна мережа мобільного зв’язку

OSM — OpenStreetMap — відкрита (у сенсі вихідного коду) мапа вулиць

JSON — JavaScript Object Notation — опис об’єкту мови JavaScript який став стандартом

ВСТУП

У сучасних умовах зростаючих загроз безпеці цивільного населення виникає нагальна потреба в ефективних та надійних системах оповіщення про повітряні тривоги, а також у наданні інформації щодо доступних місць укриття. Особливо актуальним це питання стає для територій, що зазнають регулярних нападів, де своєчасне попередження та оперативний пошук безпечного укриття можуть врятувати життя.

Основною метою цієї дипломної роботи є розробка інтегрованої smart системи, яка дозволяє не лише оперативно повідомляти населення про повітряні тривоги, але й забезпечує функціональність пошуку найближчих бомбосховищ із вказівкою рівня їх захисту та комфорту. Додатково система має функцію навігації, що полегшує швидкий доступ до укриттів у випадку надзвичайної ситуації.

Розробка такої системи є важливою складовою стратегії забезпечення безпеки населення в умовах сучасних військових конфліктів. Інтеграція різноманітних технологій, зокрема, геоінформаційних систем (GIS), мобільних додатків та технологій інтернету речей (IoT), дозволяє створити комплексний підхід до вирішення завдання захисту цивільного населення.

На сьогоднішній день існує ряд рішень для оповіщення про повітряні тривоги, проте більшість з них не включають можливості пошуку та навігації до найближчих бомбосховищ, що суттєво знижує їх ефективність. Запропонована в даній роботі система поєднує в собі багатофункціональність та високу точність, що досягається завдяки інтеграції з актуальними базами даних про місцезнаходження та стан бомбосховищ, а також за рахунок використання сучасних навігаційних технологій.

Одним із ключових аспектів роботи є дослідження та врахування різних рівнів захисту та комфорту бомбосховищ, що дозволяє користувачам обирати оптимальні місця укриття залежно від поточної ситуації та індивідуальних

потреб. Це включає аналіз та класифікацію сховищ за такими параметрами, як місткість, доступність, наявність необхідних ресурсів (водопостачання, санітарні умови, зв'язок тощо).

Впровадження запропонованої системи матиме значний позитивний вплив на рівень безпеки населення, підвищуючи його готовність до надзвичайних ситуацій та зменшуючи ризик втрат серед цивільних. Зокрема, система допоможе забезпечити швидке та ефективне реагування у випадку повітряної тривоги, що є критично важливим у нинішніх реаліях.

Ця дипломна робота містить аналіз сучасних технологій, які можуть бути використані для розробки такої системи, детальний опис її функціоналу, а також результати тестування і рекомендації щодо подальшого вдосконалення. Таким чином, вона робить вагомий внесок у розвиток засобів захисту цивільного населення та підвищення рівня його безпеки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ СИСТЕМ

1.1 Аналіз предметної області

Мета даного застосунку полягає у забезпеченні користувачів інструментом для оперативного отримання сигналів про повітряні тривоги та знаходження найближчих бомбосховищ. У мінімальній реалізації цей застосунок дозволяє, по-перше, отримувати сигнали про повітряну тривогу, і, по-друге, знаходити найближчий притулок. Важливим аспектом є можливість використання списку притулків та їх розташування на карті без доступу до Інтернету.

Розглянемо декілька ключових сценаріїв використання цього застосунку:

- Перший сценарій: Людина, яка виїжджає зі свого рідного міста, потрапляє в дорозі в несподівану ситуацію з сигналом тривоги. Завдяки застосунку вона швидко знаходить притулок у незнайомому місці, що дозволяє уникнути небезпеки.
- Другий сценарій: Біженець у новому місті не орієнтується на місцевості. Застосунок у будь-який момент може підказати йому про найближчий притулок, забезпечуючи швидке реагування на тривогу.

Тема дипломної роботи - розробка інтегрованої smart системи попередження про повітряні тривоги, пошуку бомбосховищ за рівнями захисту та комфорту з можливістю навігації - є вкрай актуальною в контексті сучасних безпекових викликів. Основна мета полягає в створенні застосунку, який забезпечує базові функції оперативного оповіщення про тривогу та навігації до найближчого безпечного місця.

Війна та пошук сховищ

Сучасні конфлікти характеризуються високою інтенсивністю та використанням широкого спектру технологій і озброєнь. Повітряні атаки, ракетні удари та загроза з боку безпілотних літальних апаратів стають

звичайним явищем у зонах конфліктів. У таких умовах життєво важливо мати ефективні засоби попередження та захисту для цивільного населення.

Пошук сховищ під час війни стає однією з основних завдань для збереження життя і здоров'я людей. Бомбосховища повинні відповідати ряду вимог, включаючи:

– Рівень захисту: Сховища мають забезпечувати надійний захист від різних видів загроз, включаючи вибухові хвилі, уламки, хімічні та біологічні атаки.

– Комфорт: Наявність основних зручностей, таких як вода, санітарні умови, засоби зв'язку та інші ресурси, що забезпечують мінімально необхідний рівень комфорту під час перебування у сховищі.

– Доступність: Легкий доступ до сховищ, особливо для людей з обмеженими можливостями, дітей та літніх людей.

Застосунок, який ми розробляємо, має на меті полегшити процес пошуку таких сховищ, забезпечуючи користувачів актуальною інформацією про їх місцезнаходження, рівень захисту та комфорту. Окрім цього, можливість використання навігації без доступу до Інтернету є критично важливою у випадках, коли зв'язок може бути втрачений.

Розширення функціоналу

Планується розширення функціоналу застосунку для врахування різних типів повітряних тривог та загроз, таких як ракетна загроза, загроза безпілотників і т. д. Застосунок буде аналізувати рівень тривоги і підбирати для користувача притулок, який забезпечить максимальний захист та комфорт.

Основні компоненти застосунку:

– Оперативне оповіщення: Користувачі отримують інформацію про повітряну тривогу в реальному часі.

– Пошук бомбосховищ: Застосунок надає дані про найближчі притулки, враховуючи їх рівень захисту та комфорту.

– Навігація: Користувачі можуть прокладати маршрути до найближчого притулку навіть без доступу до Інтернету.

– Актуальність даних: Інтеграція з оновлюваними базами даних про розташування та стан бомбосховищ.

Переваги та впровадження

Розробка та впровадження інтегрованої smart системи попередження про повітряні тривоги, пошуку бомбосховищ за рівнями захисту та комфорту з можливістю навігації є важливим кроком у забезпеченні безпеки населення. Такий підхід підвищує готовність до надзвичайних ситуацій, знижуючи ризик втрат серед цивільних осіб, забезпечуючи швидке та ефективне реагування у випадку повітряної тривоги.

Враховуючи сучасні реалії, даний застосунок сприятиме захисту населення в умовах надзвичайних ситуацій, забезпечуючи користувачів своєчасною інформацією та ефективними засобами навігації до безпечних місць укриття. Це сприятиме зниженню паніки та хаосу під час повітряних тривог і забезпечить більш злагоджене та організоване реагування на загрози.

1.2 Аналіз існуючих програмних систем

Зараз створюються різноманітні додатки та сервіси, які надають можливість отримувати сповіщення про повітряну тривогу та знаходити найближче укриття. Розглянемо деякі з них.

Додаток «Повітряна тривога» (рис.1.1) розроблено компанією Ajax Systems за ініціативи IT-компанії stfalcon.com та за підтримки Міністерства цифрової трансформації України.

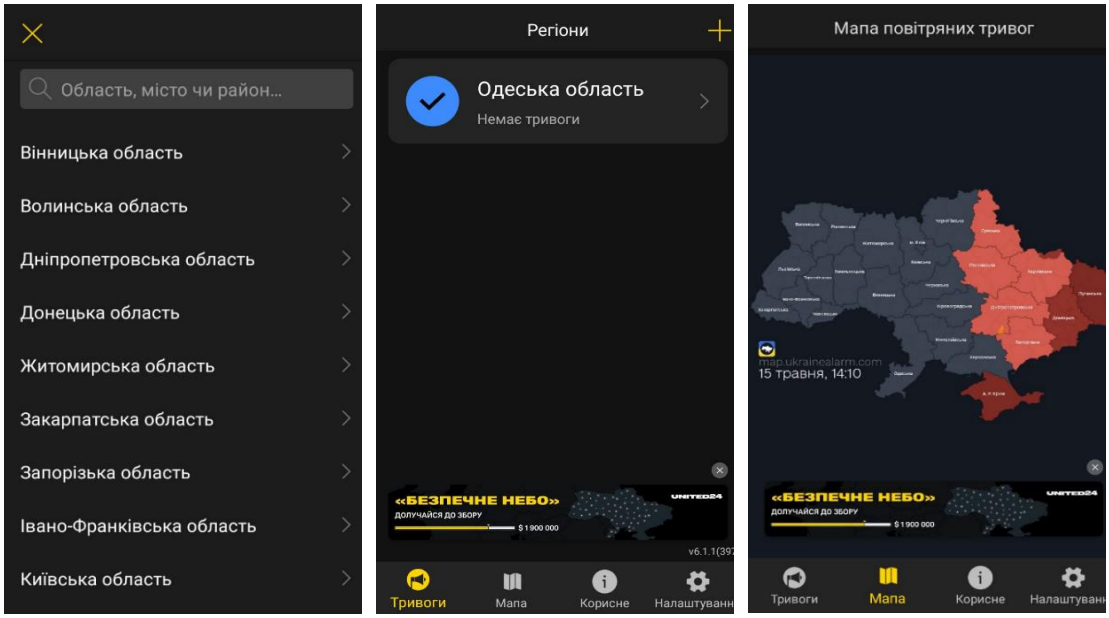


Рисунок 1.1 – Інтерфейс застосунку Повітряна тривога

Він надає вам миттєві сповіщення про повітря на ваш мобільний телефон. Додаток швидко повідомить про початок і закінчення будильника. Крім того, він максимально голосно передає критичну інформацію про повітря, хімічні речовини, техногенні та інші типи тривогів у системі цивільної оборони, навіть коли телефон перебуває в беззвучному або сплячому режимі. Програма призначена для iOS і Android.

Нещодавно додаток «Дія» (рис 1.2) оновлено новим картографічним сервісом, який пропонує укриття на випадок бомби. У цьому новому сервісі ви можете знайти найближчий притулок лише за кілька кліків.

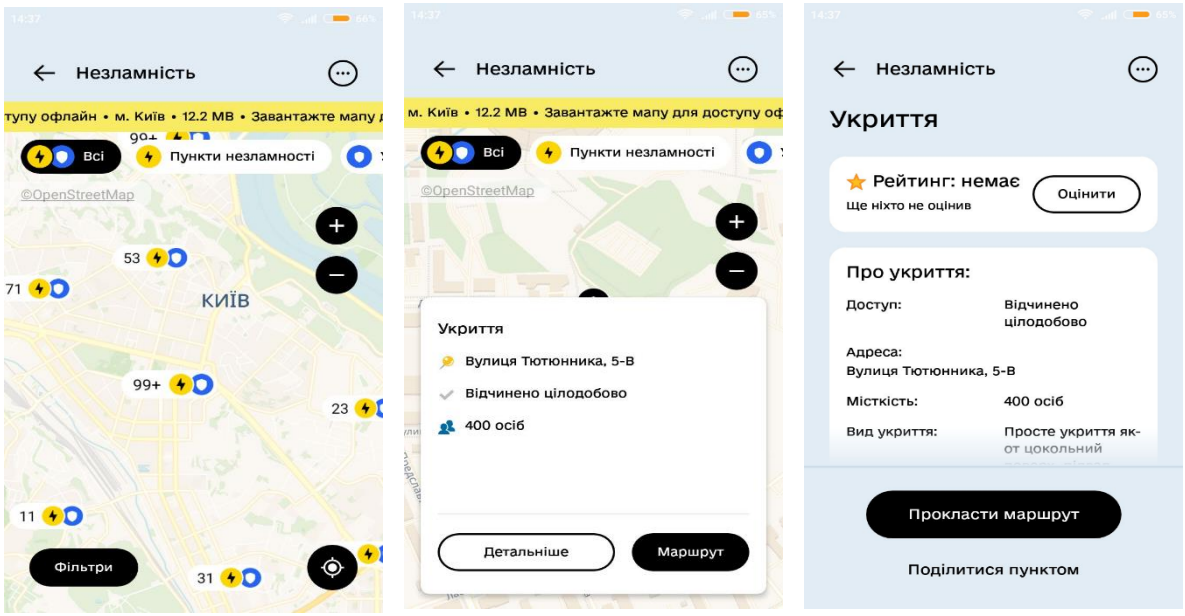


Рисунок 1.2 – Інтерфейс застосунку Дія

На інтерактивній карті також можна спостерігати адресу, вигляд, рейтинг, фото укриття та інформацію про доступ. Також дізнайтеся, скільки людей він може вмістити. Крім того, під час подання заявки ви можете оцінити стан притулку, залишити відгук і повідомити, чи він закритий. Зворотній зв’язок буде направлено до органів місцевого самоврядування, ці органи дадуть відповідь і вирішать питання на майбутнє. Ви також можете повідомити адресу членам родини або близьким друзям.

Щоб знайти найближче укриття в Дії:

– авторизуйтеся в застосунку;
– натисніть на розділ Незламність;
– виберіть категорію – Укриття;
– натисніть кнопку – фільтрувати, якщо хочете вказати необхідні параметри.
– Наприклад, ви шукаєте укриття для людей з інвалідністю;
– виберіть потрібне укриття на мапі, перегляньте інформацію про нього та прокладіть маршрут.

У Дії також можна завантажити мапу вашої області. Для цього треба перейти в розділ Незламність і натиснути – Завантажені мапи. Це дасть змогу шукати укриття, навіть якщо у вас не буде зв’язку чи інтернету.

2 ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ

2.1 Обґрунтування вибору операційної системи

Зважаючи на обмежений бюджет, я вирішив зосередитися на розробці для платформи Android. Ця стратегія обумовлена кількома вагомими технічними факторами. По-перше, розробка для Android дозволяє мені отримати ширший охоплення аудиторії, оскільки ця платформа має значно більшу кількість користувачів у порівнянні з iOS. По-друге, розвиток додатків для Android забезпечує мені можливість швидше та зручніше випробувати і вдосконалювати мої ідеї завдяки більш простому процесу розгортання та тестування. Крім того, ресурси для розробки для Android доступні без необхідності сплати за участь у програмі розробника, що робить цю платформу більш привабливою для стартапів та розробників з обмеженими фінансовими можливостями. Також за дослідженнями Statcounter (рис 2.1) частка мобільних пристроїв на ОС Android становить 71.5% проти 27.73% на iOS.

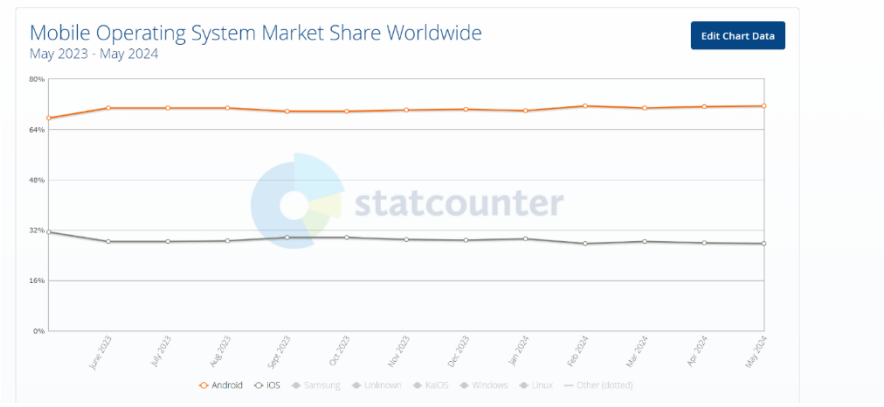


Рисунок 2.1– Дослідження Statcounter

2.2 Вибір мов програмування для розробки проекту

Проект має дві основні частини мобільний додаток та API сервер нижче я аргументую свій вибір яким є мобільний додаток під керуванням ОС Android

це є мова програмування Kotlin і API сервер під керуванням ОС Ubuntu – Golang.

Обравши Kotlin для розробки мобільного додатку під Android та Golang для API сервера на ОС Ubuntu, я врахував їхню актуальність та конкретні переваги, що вони надають.

Мова Kotlin стала де-факто стандартом для розробки мобільних додатків під управлінням ОС Android. Її активна підтримка від Google та швидке набуття популярності серед розробників свідчать про те, що вона є перспективною та ефективною мовою програмування для розвитку Android додатків. Kotlin пропонує зручний синтаксис, безпеку та високу продуктивність, що робить її ідеальним вибором для розробки мобільних застосунків [1].

Щодо Golang, ця мова програмування відома своєю швидкодією та ефективністю, що особливо важливо для серверних додатків, як API сервер. Обираючи Golang для розробки серверної частини проєкту на базі ОС Ubuntu, я врахував його здатність до обробки великого обсягу запитів та конкурентність, що є ключовими факторами для забезпечення високої продуктивності та масштабованості проєкту. Крім того, Golang має вбудовану підтримку для розробки веб-серверів та широкий вибір сторонніх бібліотек, що дозволяє швидко та ефективно створювати API сервери [2].

Таким чином, обираючи Kotlin для мобільного додатку та Golang для API сервера, я врахував їхню актуальність, потужність та специфіку для розробки конкретних частин проєкту, що дозволить забезпечити ефективність та успішний результат у реалізації цього проєкту.

2.3 Вибір середовища розробки

Популярними засобами розробки для операційної системи Android є Eclipse, Android Studio.

Eclipse – це інтегроване середовище розробки з відкритим вихідним кодом, яке полегшує розробку модульних програм для багатьох платформ. Розроблено theEclipse Foundation.

Щоб створювати програми Android для Eclipse, необхідно приєднати розширення Android Development Tools.

Однак додаток Eclipse ADT більше не підтримується, як було оголошено в червні 2015 року. Цей додаток має численні задокументовані та потенційні проблеми безпеки, які не будуть вирішені. Ось чому Google пропонує вам перейти на Android Studio.

Android Studio – офіційна платформа для розробки додатків для Android. Це середовище призначене для створення програм спеціально для операційної системи Android, ця система відрізняється від інших IDE.

Функції Android Studio включають:

– Середовище розробки полегшує використання кількох мов програмування, найпопулярніші з яких C/C++, Java.

– Програма, що полегшує кодування тексту;

– дозволяє створювати програми не лише для смартфонів/планшетів, але й для портативних комп'ютерів, приставок Android TV, Android Wear та мобільних пристроїв із незвичайною композицією

– як передня, так і задня частини екрана;

– оцінка легітимності нових ігор, інструментів та їх ефективності в роботі?

– та чи інша процедура відбувається безпосередньо в емуляторі;

– перероблення вже наявного коду;

– велика бібліотека, яка мала заздалегідь розроблені шаблони та компоненти для створення ПЗ;

– Розробка додатків для Android N, актуальна версія операційної системи;

– первинна перевірка вже реалізованої програми на наявність помилок.

– Велика різноманітність інструментів, які використовуються для тестування кожного компонента програми, ігор.

Враховуючи що фактично Eclipse фактично морально застарів і має дуже обмежену підтримку то безальтернативним вибором є Android Studio яка має офіційну підтримку компанією Google та часте оновлення.

Обравши Goland, я врахував не лише його потужність та функціональність, але й важливий фактор - сумісність з Android Studio.

Одним із ключових аспектів у моєму виборі стало те, що Goland та Android Studio мають багато спільних інструментів та схожий інтерфейс. Це дозволяє мені легко перемикатися між проектами, не витрачаючи час на адаптацію до різних середовищ розробки. Окрім того, спільні інструменти та інтерфейс роблять процес розробки більш послідовним та ефективним, адже я можу скористатися знайомими функціями та швидко вирішити завдання в обох середовищах.

Такий підхід дозволяє мені оптимізувати час розробки, забезпечуючи при цьому якість коду та продуктивність. Goland став для мене надійним партнером у розробці серверних додатків на базі Golang, забезпечуючи швидкість та зручність роботи.

3 ОПИС СТЕКА ТЕХНОЛОГІЙ

3.1 Опис технології OpenStreetMap

OpenStreetMap (OSM) - це вільна та відкрита геоданих карта світу, яка розробляється та підтримується спільнотою волонтерів. Ось деякі ключові особливості та переваги OpenStreetMap:

Преваги OpenStreetMap:

– Вільна та відкрита: OSM надає вільний доступ до своїх даних та API для використання, редагування та розповсюдження. Це робить його привабливим для широкого кола користувачів та розробників.

– Гнучкість: OSM дозволяє користувачам створювати та редагувати картографічні дані в режимі реального часу, що дозволяє швидко оновлювати та вдосконалювати карту.

– Широкий функціонал: OSM містить різноманітні типи географічних об'єктів, включаючи дороги, будівлі, річки, парки та інші, що робить його корисним для різних видів застосувань.

Недоліки OpenStreetMap:

– Менша покриття: OSM може мати меншу кількість даних порівняно з Mapbox та Google Maps, особливо в деяких регіонах світу, де спільнота користувачів менш активна.

– Обмежена підтримка служб: Оскільки OSM - це вільний проект, він може мати обмежену підтримку служб та інструментів порівняно з комерційними аналогами, такими як Mapbox та Google Maps.

Mapbox та Google Maps - це комерційні сервіси картографії, які надають широкий спектр функціоналу та підтримку.

Ось порівняння між ними та OpenStreetMap:

– Покриття та якість даних: Mapbox та Google Maps зазвичай мають більше інформації та кращу якість даних, особливо в розвинених країнах та мегаполісах.

– Продуктивність та надійність: Комерційні сервіси, такі як Mapbox та Google Maps, зазвичай мають кращу продуктивність та надійність, особливо при великих навантаженнях.

– Широкий функціонал: Mapbox та Google Maps можуть надавати більш широкий функціонал, такий як навігація, аналітика та інтеграція з іншими сервісами.

3.2 Опис технології GPS + резервна система геопозиціонування по GSM

Технологія GPS і додаткова технологія GSM для визначення місцезнаходження об'єктів є сучасними методами визначення положення. Ці технології використовуються в багатьох галузях, включаючи транспорт, логістику, безпеку та особисте використання. Ось вичерпний опис кожного з них і способу їх роботи.

GPS (система глобального позиціонування)

Основа операції

GPS - це глобальна супутникова навігаційна система, яка надає вам точні дані про широту, довготу та висоту об’єкта на Землі в будь-якій точці світу. Система складається з трьох основних компонентів:

– Супутники: у космосі є 24 активні супутники (плюс 10 резервних супутників), ці супутники знаходяться навколо Землі та завжди передають сигнали.

– Наземні станції: спостерігайте та коригуйте роботу супутників, переконавшись, що їхні дані точні.

– Приймачі GPS: Пристрої (наприклад, смартфони, автомобільні навігатори), які приймають сигнали від супутників і обчислюють своє місцезнаходження.

Як це працює

– Трілатація: Приймач GPS отримує сигнали щонайменше від чотирьох супутників.

– Часові відмітки: Кожен супутник передає інформацію про точний час відправлення сигналу та своє місцезнаходження.

– Обчислення координат: Приймач визначає час затримки сигналів від кожного супутника і обчислює відстані до них. З цих даних визначаються координати приймача.

Геопозиціонування по GSM

Принцип роботи

Геопозиціонування по GSM використовує мережу стільникового зв’язку для визначення місця розташування об'єкта. Це менш точна, але надійна альтернатива GPS, особливо в умовах обмеженого доступу до супутникових сигналів (наприклад, у приміщеннях, тунелях).

Методи геопозиціонування по GSM

– Cell ID (Ідентифікатор стільникової базової станції): Визначає місце розташування об'єкта на основі ідентифікатора базової станції, до якої він підключений. Точність залежить від радіусу покриття станції.

– Трилатерація по базових станціях: Використовує сигнали від кількох базових станцій для визначення місця розташування. Більш точний метод у порівнянні з Cell ID.

– OTDOA (Observed Time Difference of Arrival): Вимірює час приходу сигналів від різних базових станцій до мобільного пристрою для обчислення відстаней і визначення координат.

Комбіноване використання GPS та GSM

Для підвищення надійності і точності геопозиціонування часто використовують комбінований підхід:

– Переваги GPS: Висока точність, глобальне покриття, незалежність від наземних інфраструктур.

– Переваги GSM: Надійність у міських умовах, в приміщеннях, та при обмеженому доступі до супутників, швидке визначення приблизного місця розташування.

Приклади застосування

– Смартфони: Використовують GPS для точного позиціонування і GSM для швидкої оцінки місця розташування.

– Автомобільні навігатори: Відстежують транспортні засоби в режимі реального часу, використовуючи комбінацію GPS і GSM.

– Безпека і моніторинг: Використовують обидві технології для відстеження цінних вантажів або особистих місцезнаходжень у надзвичайних ситуаціях.

Комбіноване використання цих технологій забезпечує більш високу точність, надійність і стійкість до різних умов, що робить їх важливими інструментами у сучасному світі.

3.3 Опис технології VueJS

Vue.js - це прогресивний JavaScript фреймворк, призначений для розробки користувацьких інтерфейсів та веб-додатків. Заснований на компонентах, Vue.js дозволяє розробникам побудувати масштабовані та декларативні інтерфейси, просто поєднуючи HTML, CSS та JavaScript.

У третій версії Vue.js були внесені значні поліпшення та нововведення. Одним з ключових аспектів оновлення є впровадження Composition API, яка надає розробникам більшу гнучкість та ефективність при організації коду

компонентів. Крім того, у Vue.js 3 збільшено продуктивність та оптимізовано реактивність, що робить фреймворк ще більш привабливим для розробників.

Vue.js - це потужний фреймворк для розробки користувацьких інтерфейсів, який має свої переваги і недоліки порівняно з конкурентами, такими як React та Angular.

Головні переваги Vue.js:

– Легкість вивчення: Vue.js має простий та зрозумілий синтаксис, що робить його легким для вивчення новачками та швидким для впровадження в проект.

– Гнучкість: Фреймворк надає розробникам велику гнучкість в організації коду та компонентів, що дозволяє швидко реагувати на зміни вимог до проекту.

– Продуктивність: Vue.js має оптимізовану швидкодію та ефективну систему реактивності, що дозволяє створювати швидкі та реактивні веб-додатки.

– Екосистема: Велика та активна спільнота Vue розробників, багатий вибір сторонніх бібліотек та інструментів сприяють швидкому розвитку проектів на Vue.js.

Недоліки Vue.js в порівнянні з конкурентами:

– Менша популярність: У порівнянні з React та Angular, Vue.js має меншу кількість вакансій та меншу відомість на ринку праці.

– Менше офіційних інструментів: Порівняно з Angular, Vue.js може мати менше офіційних інструментів та бібліотек для додаткової функціональності.

– Недостатня підтримка: У порівнянні з React, деякі розробники вважають, що Vue.js має меншу підтримку від великих компаній та корпорацій.

– У підсумку, Vue.js - це потужний та гнучкий фреймворк з великим потенціалом для розробки сучасних веб-додатків. Хоча він може мати

свої недоліки порівняно з конкурентами, його простота та ефективність роблять його привабливим вибором для багатьох розробників.

3.4 Опис протоколу HTTP 3 + Websockets over HTTP 3

HTTP/3 – це нова версія протоколу HTTP, яка працює на основі транспортного протоколу QUIC (Quick UDP Internet Connections). QUIC – це транспортний протокол, що розроблявся Google для забезпечення швидших та безпечніших з'єднань в порівнянні з традиційним TCP [3].

Переваги HTTP/3:

– Швидший встановлення з'єднання: Завдяки використанню UDP і мінімізації кількості раундів для встановлення з'єднання.

– Менший час затримки: QUIC має покращене управління затримками, що знижує час реакції.

– Мультиплексування без головного блокування: У HTTP/3, на відміну від HTTP/2, помилки в одному потоці не блокують інші потоки.

– Підвищена безпека: QUIC включає шифрування з самого початку, забезпечуючи кращу захищеність даних.

– Мобільність: QUIC дозволяє змінювати IP-адресу без розриву з'єднання, що корисно для мобільних користувачів [4].

Недоліки HTTP/3:

– Непідтримка старими пристроями та браузерами: Новий протокол потребує оновлення інфраструктури та програмного забезпечення.

– Проблеми з мережевими пристроями: Деякі мережеві пристрої та брандмауери можуть блокувати або неправильно обробляти UDP трафік.

– Складність впровадження: Інтеграція нового протоколу може бути технічно складнішою для адміністраторів мереж та розробників [8].

WebSockets over HTTP/3

WebSockets – це протокол, який забезпечує дуплексне з'єднання між клієнтом і сервером поверх єдиного TCP-з'єднання. Перехід на HTTP/3 відкриває нові можливості для WebSockets [5].

Переваги WebSockets over HTTP/3:

– Покращена продуктивність: Завдяки меншій затримці та кращому мультиплексуванню QUIC.

– Більш стійкі з'єднання: QUIC дозволяє підтримувати з'єднання навіть при зміні мережі або IP-адреси.

– Швидке відновлення з'єднання: QUIC має механізми для швидшого відновлення з'єднання після втрати пакетів або розриву.

Недоліки WebSockets over HTTP/3:

– Обмежена підтримка: Як і HTTP/3, WebSockets over HTTP/3 ще не повністю підтримуються всіма браузерами та мережевими пристроями [11].

– Складність налаштування: Потребує оновлення серверного і клієнтського програмного забезпечення, а також налаштування мережевих пристроїв для підтримки QUIC.

Конкуренти

HTTP/2:

– Переваги: Широка підтримка, мультиплексування, стиснення заголовків.

– Недоліки: Проблеми з блокуванням потоків (головний потік може блокувати інші), залежність від TCP.

WebSockets over HTTP/2:

– Переваги: Більш зріла технологія, краще підтримується сучасними браузерами та серверами.

– Недоліки: Ті ж проблеми з блокуванням, що і в HTTP/2.

gRPC:

– Переваги: Ефективний для комунікацій між сервісами, підтримка HTTP/2, висока продуктивність.

– Недоліки: Може бути надмірно складним для простих випадків використання, залежність від HTTP/2.

SSE (Server-Sent Events):

– Переваги: Простий у використанні для однонаправлених повідомлень від сервера до клієнта.

– Недоліки: Підтримує тільки однонаправлені повідомлення, може бути менш ефективним для складних сценаріїв.

3.5 Опис протоколу Protocol Buffers

Protocol Buffers (Protobuf) - це механізм серіалізації даних, розроблений компанією Google для ефективного обміну структурованими даними між різними системами [6].

Ось головні відмінності і переваги Protobuf в порівнянні з XML/JSON та конкурентами:

Головні відмінності від XML/JSON:

– Ефективність: Protobuf використовує бінарний формат, що зменшує розмір переданих даних порівняно з текстовими форматами, такими як XML та JSON, що призводить до зменшення обсягу мережевого трафіку та швидшого оброблення даних [9].

– Швидкість: У зв'язку з використанням бінарного формату та компактного структурування даних, Protobuf зазвичай працює швидше за XML та JSON при серіалізації та десеріалізації.

– Версіонування: Protobuf має вбудовану підтримку версіонування даних, що дозволяє додавати та змінювати поля даних без руйнування сумісності з попередніми версіями.

Переваги та недоліки в порівнянні з конкурентами:

Переваги:

– Ефективність: Protobuf зазвичай працює швидше та має менший обсяг даних порівняно з іншими механізмами серіалізації.

– Менша зайнятість пам'яті: Бінарний формат Protobuf зазвичай потребує менше пам'яті для зберігання даних порівняно з текстовими форматами.

– Підтримка різних мов програмування: Protobuf має підтримку для різних мов програмування, що робить його універсальним інструментом для обміну даними між різними системами.

Недоліки:

– Складність налаштування: Використання Protobuf може вимагати додаткових зусиль для налаштування та інтеграції у додатки.

– Менша читабельність: Бінарний формат Protobuf не так простий для людського читання як XML або JSON [10], що може ускладнити відладку та розуміння даних.

– У підсумку, Protocol Buffers - це потужний та ефективний механізм серіалізації даних, який має свої переваги та недоліки порівняно з іншими форматами, такими як XML та JSON.

3.6 Опис системи Push-повідомлень (Firebase Cloud Messaging і Huawei Push Kit)

Push-повідомлення – це спливаюче повідомлення на екрані смартфона. Щоб надіслати його, потрібно скористатися сервісом розсилки. Можна надіслати повідомлення миттєво, можна налаштувати відправку на певний час, а можна зробити тригерну розсилку - вибудувати ланцюжок, який запуститься після певних дій користувача.

Firebase Cloud Messaging (FCM) - це крос-платформенна рішення від Google для надсилання повідомлень на мобільні пристрої та веб-браузери.

Ось деякі ключові особливості та можливості FCM:

– Миттєва доставка повідомлень: FCM забезпечує надсилання повідомлень на мобільні пристрої в режимі реального часу, що дозволяє користувачам отримувати сповіщення негайно.

– Підтримка крос-платформенності: FCM працює як з Android, так і з iOS, а також може надсилати повідомлення на веб-платформи, що робить його універсальним інструментом для розробки додатків на різних пристроях.

– Багатофункціональність: Окрім надсилання повідомлень, FCM також підтримує обмін даними в режимі реального часу, планування нагадувань та керування темпами надсилання повідомлень.

– Захист даних та безпека: Firebase Cloud Messaging забезпечує захист переданих даних за допомогою шифрування та ідентифікації, а також має механізми автентифікації та авторизації.

– Аналітика та моніторинг: Firebase надає інструменти для аналізу ефективності повідомлень, моніторингу їх доставки та взаємодії користувачів з ними.

FCM використовується для реалізації різноманітних сценаріїв, таких як надсилання сповіщень про нові повідомлення, оновлення додатків, сповіщення про акції та події, а також для створення реальні часу чатів та сповіщень. Його простота в інтеграції, широкий функціонал та надійність роблять його популярним вибором для розробників мобільних та веб-додатків.

Huawei Push Kit - це сервіс надсилання повідомлень (Push-сервіс) від компанії Huawei, який дозволяє розробникам надсилати повідомлення на мобільні пристрої, які працюють на платформі Huawei Mobile Services (HMS), такі як смартфони та планшети Huawei та Honor.

Ось деякі ключові особливості Huawei Push Kit :

– Крос-платформенна підтримка: Huawei Push Kit підтримує надсилання повідомлень на мобільні пристрої, що працюють на

платформі Android через Huawei Mobile Services (HMS). Він також може бути інтегрований у веб-додатки.

– Широкий функціонал: Huawei Push Kit дозволяє надсилати різноманітні типи повідомлень, включаючи текстові, зображення, звуки та сповіщення про події. Він також підтримує розсилання повідомлень по групах та географічним зонам.

– Безпека та конфіденційність: Huawei Push Kit забезпечує захист конфіденційності та безпеки переданих даних за допомогою шифрування та ідентифікації. Він також має механізми автентифікації та авторизації.

– Інструменти аналізу та відстеження: Huawei Push Kit надає інструменти для аналізу ефективності повідомлень, моніторингу їх доставки та взаємодії користувачів з ними.

За допомогою Huawei Push Kit розробники можуть надсилати повідомлення для сповіщення про нові повідомлення, оновлення додатків, сповіщення про акції та події, а також для створення реального часу чатів та сповіщень.

У підсумку, Huawei Push Kit є потужним інструментом для надсилання повідомлень на мобільні пристрої, які працюють на платформі Huawei Mobile Services, і відзначається своєю функціональністю та надійністю.

3.7 Механізм авторизації. JWT токени.

JWT (JSON Web Token) - це стандартний метод представлення токенів доступу у форматі JSON [7], що широко використовується для забезпечення аутентифікації та авторизації користувачів у розподілених системах. Порівняно з іншими методами аутентифікації, такими як сесійні куки або токени, що зберігаються на сервері, JWT має ряд переваг. Основна перевага полягає в тому, що він дозволяє зберігати інформацію про стан сеансу

безпосередньо на боці клієнта. Це зменшує навантаження на сервер та сприяє покращенню масштабованості архітектури.

Крім того, JWT дозволяє ефективно реалізувати механізм єдиного входу (Single Sign-On), де користувач може отримати доступ до різних систем, використовуючи лише один токен.

Ще однією важливою перевагою JWT є його компактність та зручність у обміні між клієнтом та сервером. Однак, варто мати на увазі, що при втраті контролю над JWT зловмисник може отримати доступ до захищених ресурсів. Для підвищення рівня безпеки рекомендується використовувати протокол HTTPS для передачі JWT між клієнтом та сервером, а також встановлювати обмеження доступу до JWT за допомогою відповідного заголовка HTTP.

Застосування цього механізму сприяє безпеці обміну даними між користувачем та сервером, роблячи процеси аутентифікації та авторизації більш ефективними та масштабованими.

3.8 OAuth 2.0

OAuth 2.0 - це протокол авторизації, розроблений для того, щоб надавати користувачам можливість контролювати обмежений доступ до своїх ресурсів (наприклад, інформації про профіль або інші ресурси) третім сторонам, таким як інші додатки або сервіси, без необхідності передачі своїх облікових даних. Цей протокол забезпечує безпечний обмін даними шляхом використання токенів доступу.

Основні концепції OAuth 2.0 включають наступне:

- Користувач: Власник ресурсу, який авторизує доступ до своїх даних. Користувач визначає, кому і які ресурси він надає доступ.
- Сервер авторизації: Це сервер, який контролює доступ до ресурсів користувача та видає токени доступу. Він використовується для автентифікації користувача та видачі токенів.

– Клієнтський додаток: Це додаток або сервіс, який отримує доступ до ресурсів користувача від його імені. Клієнтський додаток повинен отримати дозвіл від користувача перед тим, як отримати доступ до його ресурсів.

– Сервер ресурсів: Це сервер, який зберігає ресурси, до яких отримується доступ за допомогою OAuth 2.0.

Протокол OAuth 2.0 використовує різні типи токенів (наприклад, токени авторизації та токени оновлення) для забезпечення безпеки та автентифікації. Він включає в себе такі етапи, як отримання дозволу від користувача, обмін кодом авторизації на токен доступу та оновлення токенів при закінченні строку дії.

4 ПРОЕКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

4.1 Архітектура серверної частини додатку.

В процесі проектування було розроблено наступну спрощену UML діаграму серверної частини додатку. Ця діаграма представлена на рис 4.1.

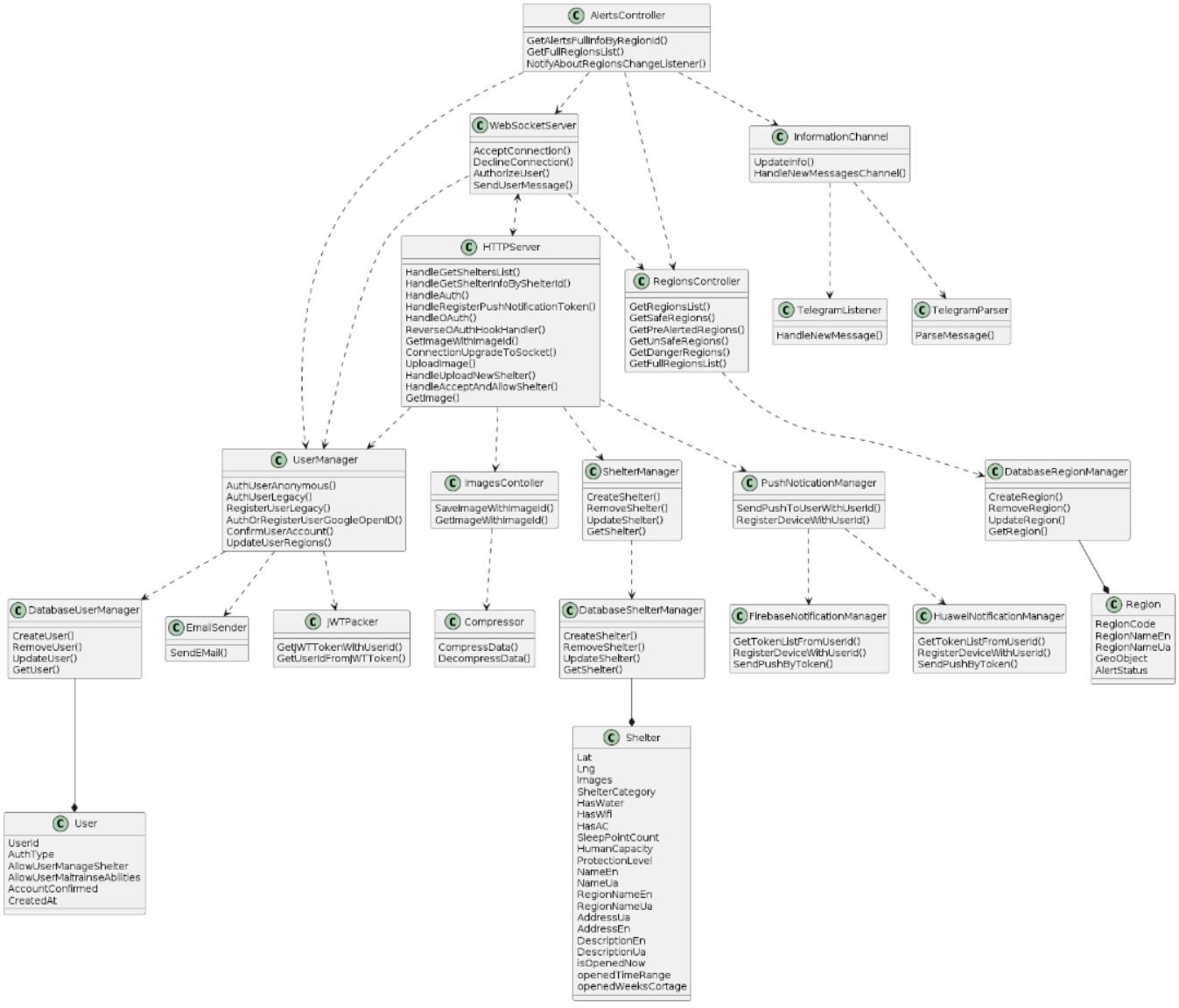


Рисунок 4.1 – Спрощена UML діаграма серверної частини

4.2 Архітектура мобільного додатку

В процесі проектування було розроблено наступну спрощену UML діаграму мобільного Android додатку. Ця діаграма представлена на рис 4.2.

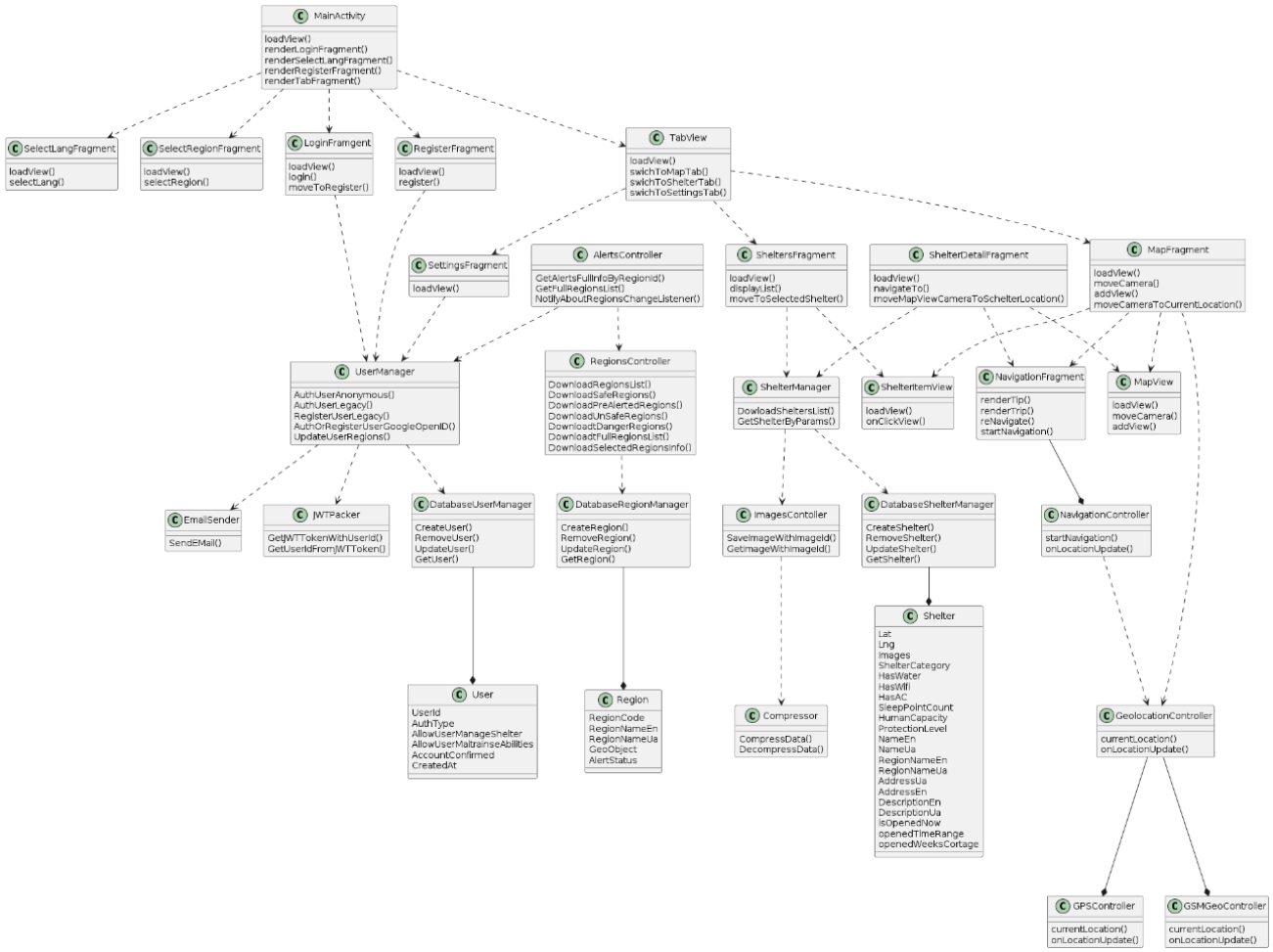


Рисунок 4.2 – Спрощена UML діаграма серверної частини

4.3 Архітектура системи

Оскільки в моєму додатку є 2 основні великі компоненти - це API-сервер і мобільний додаток, нижче я розділив цей пункт на 2 частини.

4.3.1 User-Case отримання сигналу тривоги та механізм раннього сповіщення

User-Case — це опис взаємодії користувача з системою для досягнення конкретної мети. Він є одним з основних інструментів аналізу вимог до системи в розробці програмного забезпечення. Юзер кейс описує, як

користувач взаємодіє з системою для виконання певного завдання або вирішення проблеми.

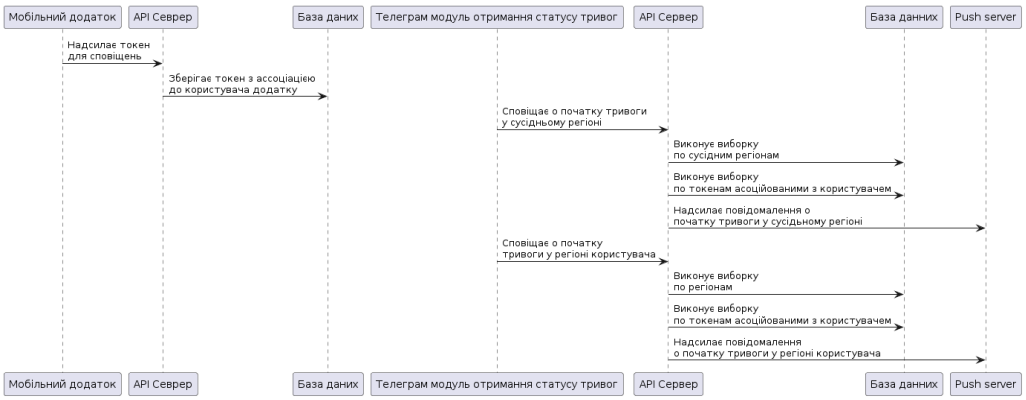


Рисунок 4.3 – Спрощена UML діаграма серверної частини

На рис 4.3 я відтворив один з цікавих механізмів моєї системи – це механізм сповіщаючий про оголошення сигналу повітряної тривоги у сусідніх регіонах включно з стандартним функціоналом сповіщення про небезпеку у регіоні. Також описані механізми безпосередньої взаємодії з механізмами доставки повідомлень.

4.3.2 User-Case система геопозиції користувача GPS + GSM Geo

При розробці системи навігації важливо враховувати що цей додаток буде використовуватися у важких умовах і з великою вірогідністю відказу роботи як системи GPS геолокації так і GSM геолокації і зв’язку з цим було передбачено насупні сценарії роботи.

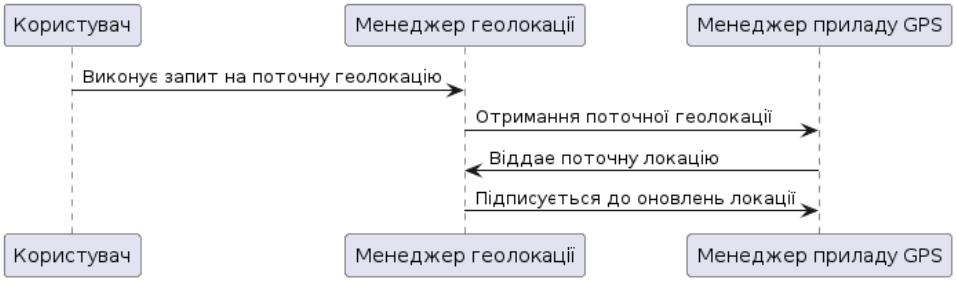


Рисунок 4.4 – Штатна робота GPS приладу

Перший сценарій відтворений на рис 4.4 це штатна робота GPS приладу - самого точного приладу який є на борту мобільного пристрою.

Наступним сценарієм є відказ GPS і зберігання роботи механізму знаходження поточної геопозиції через підключену соту GSM який я відобразив на рис 4.5

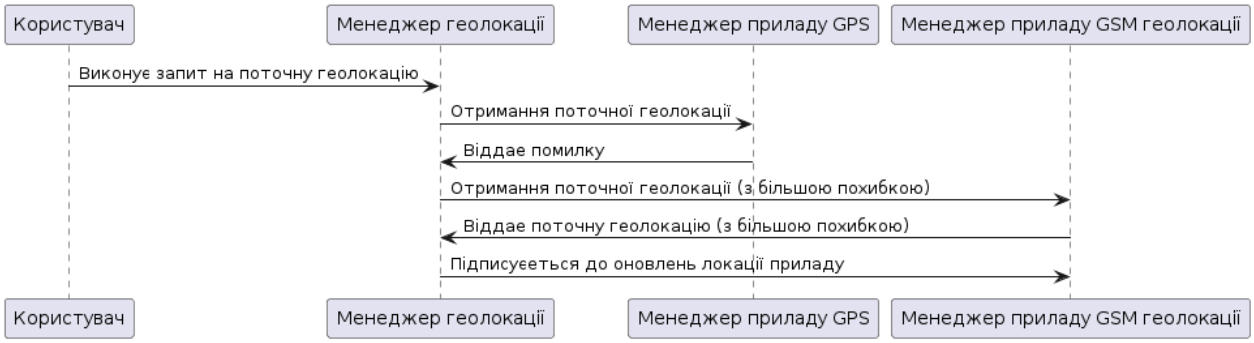


Рисунок 4.5 – Робота через резервний механізм приладу GSM геолокації

Нажаль робота з геолокацією через механізм приладу GSM геолокації має недолік у вигляді великої похибки але це резервний механізм бажано підтримувати свій функціонал і у важких умовах так як специфікація цього проекту вимагає. Але якщо робити якщо відкаже і цей прилад мною було передбачено і цей варіант з ручним вказанням геопозиції це не панацея але це є аварійний варіант роботи системи геолокації. І послідовність дій я відобразив на

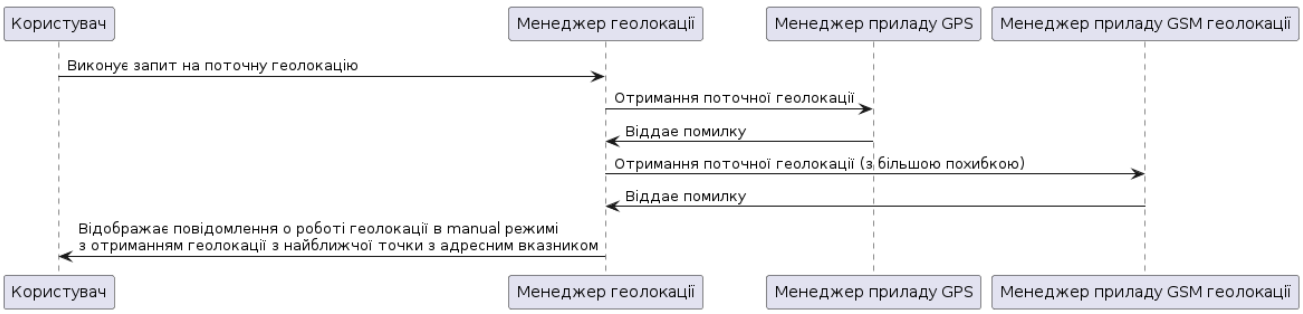
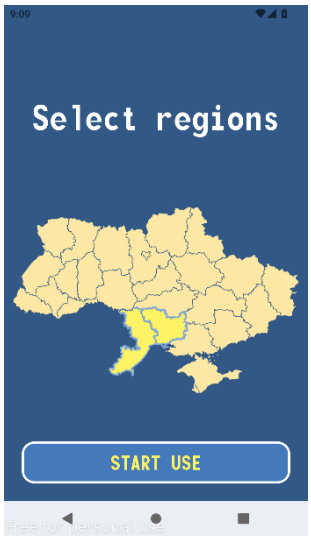


Рисунок 4.6 – Аварійний режим роботи системи геолокації

4.4 Розробка графічного прототипу

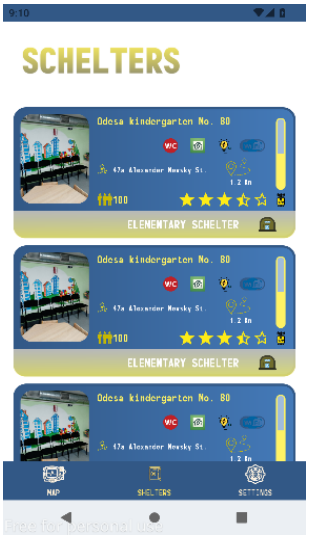
Нижче представлено галерею графічного прототипу мобільного додатка.Прототипи інтерфейсу наведені на рис. 4.7.

а)



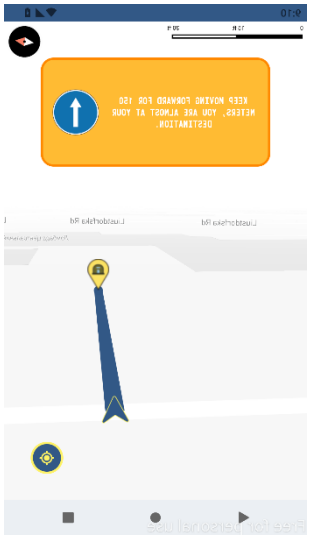
б)

в)



г)

ґ)



д)

Рисунок 4.7 – Макети інтерфейсу застосунку – а) сторінка вибору мови – б) сторінка вибору регіону – в) мапа з сховищами– г) лист сховищ – ґ) інформаційна сторінка сховища – д) екран навігації

На перших етапах проектування проекту було прийнято рішення про розробку графічного прототипу наживо за для економії часу та одночасного виконання 2 х задач - це проектування графічної частини додатку та відтворення кодової бази графічних елементів які будуть використовуватися на наступних етапах цього проекту.

5. ПРАКТИЧНА ЧАСТИНА РЕАЛІЗАЦІЇ ДОДАТКУ ДЛЯ GPS НАВІГАЦІЇ ТА ПОШУКУ СХОВИЩ

Метою проекту було створення інноваційного мобільного додатку для Android, який дозволяє користувачам знаходити найближчі сховища під час повітряної тривоги та отримувати сповіщення про стан тривоги в їхньому регіоні та сусідніх регіонах. В умовах війни в Україні цей додаток набуває особливої значущості, адже він може врятувати життя, оперативно інформуючи людей про небезпеку і вказуючи безпечні місця для укриття.

Для реалізації картографічної складової було використано OpenStreetMap, оскільки ця платформа надає детальні та актуальні карти, що є критично важливим для точного визначення розташування сховищ. Серверна частина була написана на Golang, який відомий своєю високою продуктивністю та ефективністю роботи з паралельними обчисленнями, що дозволяє обробляти великий обсяг запитів від користувачів в реальному часі. Для комунікації між сервером та клієнтом використовувалися сучасні протоколи HTTP/3, WebSockets та Protocol Buffers, що забезпечує швидку і надійну передачу даних.

Реалізація цього проекту була не просто технічним завданням, а справжнім викликом, що вимагав комплексного підходу та вирішення багатьох складних проблем.

1. Актуальність та Точність Даних

Основною складністю було забезпечити актуальність та точність даних про розташування сховищ. Для цього було необхідно інтегрувати кілька джерел даних та регулярно оновлювати інформацію. Важливим етапом стало впровадження механізмів перевірки та валідації даних, щоб уникнути потенційних помилок, які могли б коштувати життя користувачам.

2. Масштабованість та Продуктивність

В умовах кризових ситуацій, таких як повітряна тривога, додаток має працювати без збоїв навіть при значному навантаженні на сервер.

Використання Golang для серверної частини дозволило забезпечити високу продуктивність та здатність обробляти тисячі запитів одночасно. Проте, це потребувало глибокого оптимізації коду та налаштування серверів для досягнення стабільної роботи під високим навантаженням.

3. Реальний Час та Надійність Сповіщень

Важливим аспектом було забезпечити реальний час передачі сповіщень про повітряну тривогу. Використання WebSockets дозволило надсилати повідомлення миттєво, але виникали виклики з надійністю з'єднання, особливо в умовах поганої якості інтернету. Було впроваджено алгоритми повторного підключення та резервні канали для забезпечення безперервного потоку даних.

4. Безпека Даних

Безпека даних користувачів та захист від несанкціонованого доступу стали одними з пріоритетів проекту. Використання HTTPS для захищених з'єднань та протоколу HTTP/3, що забезпечує додатковий рівень безпеки та швидкості передачі даних, дозволило забезпечити високий рівень конфіденційності та захисту інформації.

5.1. Серверна частина

Серверна частина проекту була реалізована на Golang, що дозволило досягти високої продуктивності та ефективного управління ресурсами. Використання HTTP/3 забезпечило низькі затримки та високу швидкість передачі даних, а WebSockets дозволили реалізувати реальний час сповіщень.

Для обміну даними між сервером та клієнтом використовувалися Protocol Buffers (protobuf), що забезпечило компактність і швидкість серіалізації та десеріалізації даних. Це дозволило значно зменшити розмір переданих даних та підвищити швидкість їх обробки.

5.1.1. Налаштування середовища

Було налаштовано середовище для розробки на Golang. Для цього були встановлені необхідні пакети та залежності, включаючи gin-gonic, pgx для взаємодії з PostgreSQL, protobuf для роботи з Protocol Buffers та gorilla/websocket для реалізації WebSocket.

5.1.2. Розробка REST API

Створено REST API для отримання даних про сховища та статусу повітряних тривог.

1. Структура сервера налаштована у файлі main.go:

```
package main
import (
    "encoding/json"
    "fmt"
    "log"
    "net"
    "net/http"

    "github.com/gorilla/websocket"
    "github.com/lucas-clemente/quic-go/http3"
)
// Структура для представлення WebSocket сервера
type Server struct {
    clients   map[*websocket.Conn]bool
    broadcast chan []byte
    upgrader  websocket.Upgrader
}
// Функція для створення нового WebSocket сервера
func NewServer() *Server {
    return &Server{
        clients:   make(map[*websocket.Conn]bool),
        broadcast: make(chan []byte),
        upgrader:  websocket.Upgrader{},
    }
}
// Функція для запуску WebSocket сервера
func (s *Server) Run() {
    http.HandleFunc("/", s.handleConnections)
    go s.handleMessages()
    fmt.Println("WebSocket server started")
}
// Функція для обробки підключень до WebSocket сервера
func (s *Server) handleConnections(w http.ResponseWriter, r
*http.Request) {
```

```
    ws, err := s.upgrader.Upgrade(w, r, nil)
    if err != nil {
        fmt.Println("Upgrade:", err)
        return
    }
    defer ws.Close()
    s.clients[ws] = true
    for {
        var msg []byte
        err := ws.ReadJSON(&msg)
        if err != nil {
            fmt.Println("Read:", err)
            delete(s.clients, ws)
            break
        }
        s.broadcast <- msg
    }
}
// Функція для розсилки повідомлень всім підключеним клієнтам
func (s *Server) handleMessages() {
    for {
        msg := <-s.broadcast
        for client := range s.clients {
            err := client.WriteMessage(websocket.TextMessage,
msg)
            if err != nil {
                fmt.Println("Write:", err)
                delete(s.clients, client)
                client.Close()
            }
        }
    }
}
func main() {
    // WebSocket сервер
    server := NewServer()
    server.Run()
    // API сервер
    http.HandleFunc("/api/shelters", func(w http.ResponseWriter,
r *http.Request) {
        shelters := []string{"Shelter1", "Shelter2", "Shelter3"}
        json.NewEncoder(w).Encode(shelters)
    })
    // HTTP/3 сервер
    listener, err := net.Listen("udp", ":8080")
    if err != nil {
        log.Fatal(err)
    }
    handler := http3.Handler{}
    log.Fatal(handler.Serve(listener))
}
```

Налаштовано підключення до бази даних PostgreSQL:

```
db, err := sql.Open("postgres", "user=youruser dbname=yourdb
password=yourpassword host=yourhost sslmode=disable")
    if err != nil {
        panic(err)
```

```
    }
    defer db.Close()

    // Перевірка з'єднання з базою даних
    err = db.Ping()
    if err != nil {
        panic(err)
    }
    fmt.Println("Connected to PostgreSQL database!")
```

5.1.3. Розробка WebSocket сервісу

Реалізовано WebSocket сервіс для надсилання сповіщень про повітряні тривоги в реальному часі:

```
import (
     "log"
     "net/http"

     "github.com/gorilla/websocket"
)
var (
     upgrader = websocket.Upgrader{
          CheckOrigin: func(r *http.Request) bool {
               return true // Дозволяємо всім доменам
підключатися до WebSocket сервера
          },
     }
)
// Функція, яка буде обробляти підключення до WebSocket сервера
func wsHandler(w http.ResponseWriter, r *http.Request) {
     // Перевірка наявності токена у запиті
     token := r.URL.Query().Get("token")
     if token != "secret_token" {
          http.Error(w, "Unauthorized", http.StatusUnauthorized)
          return
```

```
        }
        // Підключення до WebSocket
        conn, err := upgrader.Upgrade(w, r, nil)
        if err != nil {
             log.Println("Upgrade:", err)
             return
        }
        defer conn.Close()

        // Безкінечний цикл для обробки повідомлень від клієнта
        for {
             // Отримання повідомлення від клієнта
             _, message, err := conn.ReadMessage()
             if err != nil {
                  log.Println("Read:", err)
                  break
             }
             resp, err := SocketManagerHandleClientGetResponse()
                  if err != nil {
                  log.Println("Write:", err)
                  break
             }
             // Відправка повідомлення клієнту
             err = conn.WriteMessage(resp, message)
             if err != nil {
                  log.Println("Write:", err)
                  break
             }
        }
}
```

5.2. Протоколи обміну даними (Protocol Buffers)

5.2.1. Визначення protobuf моделей

Для ефективного обміну даними між сервером та клієнтом, було створено protobuf моделі для опису структури даних.

1. Створено файл shelter.proto для опису даних про сховища:

```
syntax = "proto3";

package proto;

message Alert {
    int32 id = 1;
    string region = 2;
    string status = 3;
    int64 timestamp = 4;
}

message AlertList {
    repeated Alert alerts = 1;
}
```

2. Створено файл alert.proto для опису даних про тривоги

```
syntax = "proto3";

package proto;

message Shelter {
    int32 id = 1;
    string name = 2;
    double latitude = 3;
    double longitude = 4;
    string address = 5;
    int32 capacity = 6;
}

message ShelterList {
    repeated Shelter shelters = 1;
}
```

5.2.2. Генерація Go-коду з protobuf

Для генерації Go-коду з protobuf файлів використовується інструмент protoc:

```
protoc --go_out=. shelter.proto alert.proto
```

5.2.3. Генерація Kotlin-коду з protobuf

Для генерації Kotlin-коду з protobuf файлів використовується protoc з відповідними параметрами:

```
protoc --kotlin_out=. shelter.proto alert.proto
```

5.3. Клієнтська частина

Клієнтська частина була реалізована на Android з використанням Kotlin. Для відображення карт була інтегрована бібліотека OpenStreetMap, що дозволило використовувати деталізовані карти та забезпечити високу точність геолокації.

5.3.1. Налаштування середовища

Було налаштовано середовище розробки в Android Studio, створено новий проект з Empty Activity з використанням Kotlin.

5.3.2. Інтеграція OpenStreetMap

1. Додано залежності для роботи з OpenStreetMap:

```
dependencies { implementation 'org.osmdroid:osmdroid-
android:6.1.7' }
```

2. Відображено карту в activity_main.xml:

```
<org.osmdroid.views.MapView android:id="@+id/mapView"
android:layout_width="match_parent"
```

```
android:layout_height="match_parent" tilesource="Mapnik"
zoomControls="true" />
```

3. Налаштовано карту в MainActivity.kt:

```
import org.osmdroid.config.Configuration
import org.osmdroid.tileprovider.tilesource.TileSourceFactory
import org.osmdroid.views.MapView

class MainActivity : AppCompatActivity() {

    private lateinit var mapView: MapView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Ініціалізація конфігурації OSM
        Configuration.getInstance().load(applicationContext,
PreferenceManager.getDefaultSharedPreferences(applicationContext
))

        // Ініціалізація MapView
        mapView = findViewById(R.id.mapView)
        mapView.setTileSource(TileSourceFactory.MAPNIK)

mapView.zoomController.setVisibility(CustomZoomButtonsController
.Visibility.SHOW_AND_FADEOUT)
        mapView.setMultiTouchControls(true)
    }
}
```

5.3.3. Взаємодія з сервером

1. Реалізовано функцію для отримання сховищ:

```
   // Створення клієнта OkHttp
    val client = OkHttpClient()

    // Створення запиту GET
    val request = Request.Builder()
        .url("https://api.heimdallr.com.ua/api/shelters")
        .build()

    // Виконання запиту
    val response = client.newCall(request).execute()

    // Перевірка успішності виконання запиту
    if (!response.isSuccessful) {
        throw RuntimeException("Failed to execute request:
${response.code}")
    }

    // Отримання тіла відповіді
```

```
    val responseBody = response.body?.byteStream()

    // Десеріалізація Protocol Buffers даних
    val shelterList = ShelterList.parseFrom(responseBody)

    // Виведення результату
    for (shelter in shelterList.sheltersList) {
        println("Shelter ID: ${shelter.id}, Name:
${shelter.name}")
    }
```

2. Додано WebSocket для сповіщень:

```
import okhttp3.*
import okhttp3.ws.WebSocket
import okhttp3.ws.WebSocketListener
import okio.Buffer
import java.io.IOException

class WebSocketClient {

    private var webSocket: WebSocket? = null
    private val client = OkHttpClient()

    fun connect(token: String) {
        val request = Request.Builder()
            .url("wss://api.heimdallr.com.ua/ws")
            .addHeader("Authorization", "Bearer $token")
            .build()
        webSocket = client.newWebSocket(request, object :
WebSocketListener() {
            override fun onOpen(webSocket: WebSocket, response:
Response) {
                super.onOpen(webSocket, response)
                // Підключення успішне
            }

            override fun onFailure(webSocket: WebSocket?, t:
Throwable?, response: Response?) {
                super.onFailure(webSocket, t, response)
                // Помилка підключення
            }

            override fun onMessage(webSocket: WebSocket?, text:
String?) {
                super.onMessage(webSocket, text)
                // Обробка отриманих повідомлень
            }
        })
    }

    fun sendMessage(message: String) {
        webSocket?.send(message)
    }

    fun disconnect() {
        webSocket?.close(1000, "Goodbye!")
    }
```

5.4 Тестування

Так як сутностей в проекті багато то було вирішено створити 2 тести які будуть у змозі перевірити більшу частину проекту у полу автоматичному режимі як базова перевірка після оновлень, або покращень кодової бази.

5.4.1 Базовий Unit тест серверної частини

```
package main
import (
	"encoding/json"
	"net/http"
	"net/http/httptest"
	"testing"
)
// Тест для функції getShelters
func TestGetShelters(t *testing.T) {
	// Створюємо тестовий HTTP запит
	req, err := http.NewRequest("GET", "/shelters", nil)
	if err != nil {
		t.Fatal(err)
	}
	// Створюємо фейковий HTTP ResponseWriter для запису
відповіді
	rr := httptest.NewRecorder()
	// Викликаємо функцію, яку тестируємо
	handler := http.HandlerFunc(getShelters)
	handler.ServeHTTP(rr, req)
	// Перевіряємо код відповіді
	if status := rr.Code; status != http.StatusOK {
		t.Errorf("handler вернув неправильний статус код:
отримано %v, очікувано %v",
			status, http.StatusOK)
	}
	// Перевіряємо, чи повернулися правильні дані
	if rr.Body.String() != expected {
		t.Errorf("handler вернув неправильне тіло: отримано
%v, очікувано %v",
			rr.Body.String(), expected)
	}
}
```

У ситуації обмеженого часу для розробки для базової перевірки функціоналу було розроблено 1 unit тест функціонал якого покриває перевірку серіалізації та базового мережевого стеку і активує частину коду яка відповідає за роботу з базою даних.

5.4.2 Базовий запуск мобільного додатку

Appium використовується для автоматизації тестування мобільних додатків. У цьому проекті я використаю Appium для перевірки запуску додатку та ініціалізації мапи.

```
import os
import time
import pytest
from appium import webdriver
from appium.webdriver.common.touch_action import TouchAction
from appium.webdriver.common.mobileby import MobileBy
from appium.webdriver.common.multi_action import MultiAction
from appium.webdriver.common.touch_action import TouchAction
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import NoSuchElementException
# Шлях до додатку Heimdallr.apk
APP_PATH = os.path.join(os.getenv('HOME'), 'Downloads',
'Heimdallr.apk')

# Конфігурація Appium
desired_caps = {
    'platformName': 'Android',
    'deviceName': 'lolipop2353-generic',
    'app': APP_PATH
}
# Ініціалізація драйвера Appium
driver = webdriver.Remote('http://localhost:4723/wd/hub',
desired_caps)
# Завершення роботи драйвера після тесту
@pytest.fixture(scope="function")
def teardown():
    yield
    driver.quit()
    # Очікування завантаження карти
    map_loaded = WebDriverWait(driver,
30).until(EC.presence_of_element_located((MobileBy.ID,
‘map_v’)))

    # Перевірка, чи карта успішно завантажена
    assert map_loaded.is_displayed()
```

ВИСНОВКИ

Розробка мобільного додатку для GPS-навігації та пошуку сховищ під час повітряної тривоги в Україні є свідченням того, як технології можуть вирішувати соціально значущі завдання та врятувати життя. Цей проект став результатом складної, але надзвичайно важливої роботи, яку я провів, використовуючи передові технології, такі як Golang, OpenStreetMap, HTTP/3, WebSockets та Protocol Buffers.

Я впевнений, що мобільний додаток не лише надає користувачам засоби для швидкої реакції на небезпеку, але і створює можливість спільної дії та взаємодії в умовах кризи. Використання GPS-навігації та інші функції додатку дозволяють не лише уникнути небезпеки, але й координувати дії рятувальних служб та надавати підтримку один одному в критичних ситуаціях.

Цей проект відображає важливість технологій у покращенні безпеки та захисту громадян. Він служить прикладом того, як інновації можуть вирішувати актуальні проблеми суспільства та сприяти створенню безпечнішого середовища для всіх. Я вірю, що подібні ініціативи мають потенціал не лише змінити світ, але й рятувати життя в найважчі моменти.

Завдяки цьому проекту я роблю крок у майбутнє, де технології стають не лише інструментом для розваг, але і найважливішими засобами для захисту та підтримки людей у будь-яких умовах. Моя робота не завершена, і я з нетерпінням очікую подальших можливостей для впливу на суспільство та створення позитивних змін за допомогою технологій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Офіційний сайт Kotlin. URL: https://kotlinlang.org/
2. Офіційний сайт Go. URL: https://go.dev/
3. HTTP3 прототип HTTP over QUIC: огляд та специфікація на MDN. URL:https://developer.mozilla.org/enUS/search?q=HTTP%20over%20QUIC
4. QUIC і HTTP/3 у Вікіпедії. URL: https://uk.wikipedia.org/wiki/QUIC
5. WebSockets на MDN. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers
6. Огляд Protocol Buffers на Wikipedia. URL:https://uk.wikipedia.org/wiki/Protocol_Buffers
7. JSON на MDN. URL: https://developer.mozilla.org/en US/docs/Web/JavaScript/Reference/Global_Objects/JSON?retiredLocale=uk
8. RFC 9000: Hypertext Transfer Protocol Version 3 (HTTP/3). URL: https://datatracker.ietf.org/doc/html/rfc9000
9. RFC 8873: The protobuf Tag-Length-Value (TLV) Encoding. URL: https://datatracker.ietf.org/doc/html/rfc8873
10. RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format. URL: https://datatracker.ietf.org/doc/html/rfc8259
11. RFC 9220: Bootstrapping WebSockets with HTTP/3. URL: https://www.ietf.org/rfc/rfc9220.html