НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ
ІНСТИТУТ імені ІГОРЯ
СІКОРСЬКОГО»
Приладобудівний факультет
Кафедра приладів і систем орієнтації і навігації

«На правах рукопису»
УДК

«До захисту допущено»
Завідувач кафедри
_____ Н.І., Бурау
«___»_____20__ р

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»(«Комп'ютерно-інтегровані технології оптичних та навігаційних систем»)

на тему: «Автоматизована Система інформування цивільного населення в умовах надзвичайної ситуації»

Виконав:
студент VI курсу, групи ПГ-
11мп Кравчук О.П. _____

Керівник:асистент,
Паздрій О.Я. _____

Консультант з Розроблення стартап-роекту:
завідувач кафедри економічної кібернетики,
д.е.н., проф. Бояринова К.О. _____

Рецензент:
Антонюк В.С. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студен _____

Київ – 2022 року

ЗАВДАННЯ

на магістерську дисертацію студента

Кравчука Олександра Петровича

1. Тема дисертації «Автоматизована система інформування цивільного населення в умовах надзвичайної ситуації», науковий керівник дисертації Паздрій Ольга Ярославівна, асистент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації « » грудня 2022 р.

3. Предмет дослідження: процес автоматизації інформування цивільного населення в умовах надзвичайних ситуацій

4. Об’єкт дослідження: автоматизована система інформування цивільного населення в умовах надзвичайних ситуацій.

5. Перелік завдань, які потрібно розробити

-зробити огляд основних засобів автоматичного інформування населення в умовах виникнення надзвичайних ситуацій;

- провести порівняльний аналіз існуючих систем для автоматизованого інформування населення в умовах виникнення надзвичайних ситуацій;

- розробка алгоритмічного забезпечення для реалізації автоматизованої інформаційно-пошукової ситеми;

- розробка програмного забезпечення для реалізації автоматизованої інформаційно-пошукової ситеми;

- створення прототипу (комп’ютерного/мобільного додатку) системи, перевірка працездатності;

6. Орієнтовний перелік ілюстративного матеріалу: структурні схеми алгоритмів, та діаграма мобільного додатку.

7. Орієнтовний перелік публікацій 1 матеріали науково-технічних конференцій у наукових збірниках

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Етапи розвитку засобів автоматичного інформування населення в умовах виникнення НС	04.10.2022	
2	Сучасний стан та методи автоматичного інформування населення в умовах виникнення НС	17.10.2022	
3	Класифікація систем автоматичного інформування населення в умовах виникнення НС	27.10.2022	
4	Розробка алгоритмічного забезпечення для системи автоматичного інформування населення в умовах виникнення НС	02.11.2022	
5	Написання програмного забезпечення	07.11.2022	
6	Створення прототипу системи. Перевірка працездатності	11.11.2022	
7	Дослідження характеристик системи для автоматичного інформування населення в умовах виникнення НС	17.11.2022	
8	Оформлення пояснювальної записки. Підготовка до захисту	28.11.2022	

Студент _____ Олександр КРАВЧУК

(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Ольга ПАЗДРІЙ

(підпис) (ініціали, прізвище)

РЕФЕРАТ

Кравчук О.П. Автоматизована система інформування цивільного населення в умовах надзвичайної ситуації.

Дипломна магістерська робота за спеціальністю 151 – «Автоматизація та комп'ютерно-інтегровані технології» – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2022 рік.

Розглянуто основні види автоматизованого сповіщення та принципи їх роботи, також проаналізовано існуючі програмні рішення, їх засоби реалізації, які використовуються для створення мобільних додатків, зокрема мову програмування dart, фреймворк flutter та підключення бази даних firebase. Розглянуто технології відкритого API.

В результаті даної роботи було розроблено автоматизовану інформаційно-пошукову системи у виді мобільного додатку для сповіщення та пошуку найближчого укриття за допомогою API та бази даних укриттів міста Київ. Також у розробленому додатку було реалізовано функціонал інформаційних сторінок, де можна прочитати інформацію про перелік дій під час повітряної тривоги та переглянути список небезпечних об'єктів.

Дисертаційна робота складається зі вступу, чотирьох розділів, висновків до розділів, загального висновку та переліку літеатурних посилань. Загальний обсяг дисертації становить 82 сторінок, 33 рисунків, 21 таблиць, 24 посилань.

Ключові слова: мобільний додаток, пошук, сповіщення, автоматизована система, dart, flutter, firebase, API, android studio.

ABSTRACT

O.P. Kravchuk Automated system of informing the civilian population in emergency situations.

Master's thesis in specialty 151 - "Automation and computer-integrated technologies" - National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute", Kyiv, 2022.

The main types of automated notifications and the principles of their operation are considered, as well as existing software solutions and their implementation tools used to create mobile applications, including the dart programming language, the flutter framework and firebase database connection, are analyzed. Open API technologies are considered.

As a result of this work, an automated information and search system was developed in the form of a mobile application for notification and search for the nearest shelter using the API and the Kyiv city shelter database. The developed application also implemented the functionality of information pages where you can read information about the list of actions during an air alert and view the list of dangerous objects.

The dissertation consists of an introduction, four chapters, conclusions to the chapters, a general conclusion and a list of literary references. The total volume of the dissertation is 82 pages, 33 figures, 21 tables, 24 references.

Keywords: mobile application, search, notification, automated system, dart, flutter, firebase, API, android studio.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

WEA – це сповіщення про надзвичайні ситуації, що передаються з мобільних телефонів через радіочастоту

SMS – служба коротких повідомлень (Short Message Service)

EAS – це система оповіщення населення, яка передає інформацію про надзвичайні ситуації через радіо та телевізійні передачі

НС – надзвичайні ситуації

АСРВО – автоматизована система раннього виявлення та оповіщення.

СРВНСО – автоматизована система раннього виявлення загрози виникнення надзвичайних ситуацій та оповіщення людей

ДСНС – Державна служба України з надзвичайних ситуацій

SDK – набір із засобів розробки, утиліт і документації, який дає програмістам змогу створювати прикладні програми

NDK – обхідний набір інструментарію для розробки компонентів програмного забезпечення для платформи Android

МВ – мегабайт

у.о. – умовні одиниці

СЦПТС – система централізованого пожежного та техногенного спостереження.

ЗАСЦО – загальнодержавна автоматизована система централізованого оповіщення.

A – ПЦС – пульт централізованого спостереження.

B – ТАСЦО – територіальна автоматизована система централізованого оповіщення.

C – ПК – пульт керування СРВНСО.

D – ПО – пристрій оповіщення .

E – КТЗІО – кінцеві технічні засоби інформування та оповіщення .

F – КП – комунікаційний пристрій.

H – джерела первинної інформації.

G – суміжні системи забезпечення безпеки.

Вступ

Кожного дня в світі відбувається багато надзвичайних ситуацій різного характеру, кожна держава світу має дбати про своє цивільне населення та сповіщати їх про можливу загрозу заздалегідь, це може врятувати безліч життів. Як ми знаємо, в 21 столітті важко уявити людей, які не користуються різними гаджетами, від мобільного телефону до комп'ютера. В більшості додатків, які інформують про повітряну тривогу, невирішена задача пошуку найближчого укриття. Це допоможе цивільному населенню не втрачати час на пошук інформації.

Саме тому розробка автоматизованої системи інформування цивільного населення під час надзвичайної ситуації є важливою і актуальною задачею під час військового стану та можливих військових дій на території України.

Метою дисертаційної роботи є створення автоматизованого мобільного додатка. основним функціоналом якого є отримання сповіщення про повітряну тривогу та пошук найближчого укриття у місті Київ, з додатковою функцією інформування користувачів про потенційно небезпечні об'єкти та поради щодо поведінки під час повітряної тривоги.

Для досягнення поставленої мети потрібно вирішити такі задачі, як:

- зробити огляд основних засобів автоматичного інформування населення в умовах виникнення надзвичайних ситуацій;
- провести порівняльний аналіз існуючих систем для автоматизованого інформування населення в умовах виникнення надзвичайних ситуацій;
- огляд методів та засобів реалізації поставленої мети;
- розробка алгоритмічного забезпечення для реалізації автоматизованої інформаційно-пошукової ситеми;
- створення прототипу (комп'ютерного/мобільного додатку) системи, перевірка працездатності;
- Тестування розробленого мобільного додатка.

Об'єкт дослідження: автоматизована інформаційно-пошукова система інформування цивільного населення в умовах надзвичайних ситуацій

Предметом дослідження є процес автоматизації інформування та пошуку даних для цивільного населення в умовах надзвичайних ситуацій

Методи дослідження – для розробки автоматизованої системи сповіщення та пошуку укриття використовувалось мова програмування dart, фреймворк flutter для обробки даних та відображення у вигляді графічного інтерфейсу.

Наукова новизна дисертації полягає в автоматизації процесу пошуку найближчого укриття від поточної геолокації після інформування користувача про повітряну тривогу та побудову маршруту до укриття, що, в свою чергу, зменшує час користувача на пошук потрібної інформації

Практична цінність отриманих результатів. В ході роботи було розроблено мобільний додаток для інформування цивільного населення про надзвичайні ситуації, а саме: про надзвичайну ситуацію військового характеру (повітряна тривога), було розроблено автоматизовану систему сповіщення цивільного населння та реалізовано автоматичний пошук найближчого укриття в залежності від поточного місцезнаходження.

Публікації. За матеріалами дисертації було опубліковано 1 тези доповідей на науково-практичній конференції.

РОЗДІЛ 1 АВТОМАТИЗОВАНІ СИСТЕМИ ІНФОРМУВАННЯ НАСЕЛЕННЯ

1.1. Огляд систем сповіщення

Системи масового сповіщення були невід'ємною частиною людського суспільства з тих пір, як була мова. Звичайно, мова необхідна для швидкого поширення інформації, але як люди традиційно спілкувалися один з одним.

Масова комунікація не завжди легка, але, тим не менш, вона була надзвичайно важливою протягом всієї історії людства. Щоб отримати глибше розуміння масової комунікації, яка просувається вперед, ми спочатку повинні зазирнути в минуле, щоб визначити основну мету вигукування повідомлення масам.

Як згадувалося, системи масового сповіщення існували з тих пір, як ми могли спілкуватися, і в історії існує велика кількість ключових моментів, які можна торкнутися щодо масового сповіщення. Давайте розглянемо кілька найвпливовіших і революційних на той час методів масового сповіщення.

Широко відомі як перша форма візуального спілкування, димові сигнали вперше були використані близько 200 року до нашої ери вздовж Великої китайської стіни. Димові сигнали мали здатність передавати дані на великі відстані за досить короткий проміжок часу Насправді в системі димової сигналізації був можливий досить високий ступінь налаштування, наскільки простим здається метод. Наприклад, колір диму може бути змінений і, отже, вказувати на інший тип повідомлення [1].

У середні віки для оповіщення почали використовувати дзвони – церковні або спеціальні пожежні дзвони. Багато вчених вважають, що головною функцією дзвону було не відтворення музики, а власне попередження. Дзвін протягом багатьох років служив засобом попередження населення про наближення лиха, громадських заворушень або наближення ворогів . З часом міста росли, розвивалися, і одного дзвона, щоб сповістити околиці, було вже недостатньо. Виникла потреба в нових технічних рішеннях у сфері раннього оповіщення та оповіщення про надзвичайні

ситуації. В кінці 18 століття були зроблені перші сміливі спроби створити механічний сигналізатор і автоматизувати його. Типовим прикладом того часу був пожежний дзвін – важкий вантаж, що висів на мотузці. Коли вогонь спалив мотузку, вантаж впав і вдарився об тривожний дзвін . Крім того, були інші спроби розробити обладнання, яке могло б реагувати на зміну температури, напругу або деформацію, кількість рідини тощо [2].

Коли Олександр Грем Белл здійснив перший телефонний дзвінок у 1876 році, навряд чи він міг уявити еволюцію, яка відбуватиметься протягом наступних 150 років і приведе нас у епоху 5G, у якій ми живемо сьогодні. Але давайте не будемо забігати вперед щодо телефонних технологій. Телефон, очевидно, змінив хід людського спілкування, і його здатність надсилати та отримувати аудіозв'язок у режимі реального часу стала справжньою зміною гри. Ніколи раніше двоє людей, які перебували за тисячі миль від нас, не могли розмовляти між собою так, ніби вони перебували в одній кімнаті.

До середини 1890-х років було створено перший пристрій радіозв'язку на великій відстані, що проклало шлях до дешевого масового зв'язку. Бездротову систему створив італійський винахідник Гульєльмо Марконі. До 1912 року всі американські кораблі, що плавали понад 200 миль від узбережжя, повинні були бути обладнані бездротовим радіо. А до 1920 року були створені перші громадські радіостанції. Золотий вік радіо розпочався приблизно в 1930-х роках, коли сталася Велика депресія. У цей важкий час люди зверталися до радіо, щоб отримати розвагу, новини та розраду. Цей момент в історії легко назвати ключовим для систем масового сповіщення. У певному сенсі це був «ідеальний шторм». Нова технологія розквітла якраз у той час в історії США, коли люди були досить довго вдома, їм потрібно було бути поінформованими та згаяти час. Радіо задовольнило велику потребу, і поки телебачення не стало мейнстрімом, воно мало кілька десятиліть шаленого успіху. З'явилися електромеханічні (моторні) сирени , пристрої стали потужнішими. Сирени цього типу використовували ротори та статори та активувалися електричним струмом для звукової сигналізації. Проте хід технічного прогресу не зупиняється. Швидкість і обсяг передачі даних зростає. Розробляється все більше і більше сучасних систем,

щоб надати своїм користувачам покращені функції безпеки. Швидкість каналів зв'язку між цими системами також зростає. Такі питання, як можливості інтеграції, автоматизація процесів, взаємодія систем, надійність даних і безпека передачі даних стали дуже важливими.

До того, як технології цифрової ери стали звичним явищем, спілкування можна було описати як «миттєве, але стаціонарне». У нас було телебачення та радіо, які могли передавати інформацію по всій планеті за лічені секунди, але якщо хтось відійде від цих пристроїв, вони навряд чи візьмуть із собою телевізор чи радіо, щоб бути в курсі на ходу. Популяризація Інтернету, а згодом і мобільних телефонів перетворила б масову комунікацію на ландшафт, з яким ми знайомі сьогодні. Незалежно від того, де ви перебуваєте в будь-який момент, ваш смартфон, швидше за все, буде під рукою. Незалежно від того, чи означає це фотографувати природу під час походів, шукати, що ви можете захотіти сьогодні на вечерю. Смартфони — це універсальний компаньйон, який допомагає навчати та розважати нас, одночасно забезпечуючи корисність в інших ключових сферах нашого життя. Ця залежність від наших смартфонів дає нам рівень захисту, про який ми часто не думаємо. Зручність смартфона також дозволяє нам отримувати електронні листи, текстові повідомлення, телефонні дзвінки та push-повідомлення в будь-який час. Хоча багато з цих повідомлень можна позначити як маркетинговий хід, ці методи комунікації також є основними методами, які використовуються критично важливими комунікаційними компаніями та федеральним урядом для надсилання відповідної інформації масам, якщо виникне надзвичайна ситуація, коли кожен у певній місцевості необхідно повідомити.

На даний момент я вважаю найактуальнішим методом інформування населення це інформування через смартфони, в 21 столітті майже у кожної людини є цей девайс, інформування може бути як текстовим так і звуковим, в наших реаліях це особливо важливо. Під час військового стану всі люди мають якнайскоріше дізнаватись про небезпеку, щоб мати змогу врятувати своє життя. Радіо та телевізор також є надійним методом інформування але уже втратив свою актуальність. Не зважаючи на різні

девайси які сповіщують про провітряну тривогу та інші надзвичайні ситуації, потрібні різні типи сирен для сповіщення населення.

1.1.2. Види систем сповіщення

Система оповіщення про надзвичайні ситуації – це загальнодоступна система, яка складається з телемовлення, супутникового та кабельного телебачення, радіо, соціальних мереж, мобільних пристроїв та будь-якого іншого пристрою попередження для надання інформації про поточну чи майбутню небезпеку. Бездротові екстренні сповіщення, або WEA – це сповіщення про надзвичайні ситуації, що передаються з мобільних телефонів через радіочастоту. Ці сповіщення використовуються органами управління надзвичайними ситуаціями, організаціями громадської безпеки та іншими державними органами для сповіщення громадськості про надзвичайні ситуації.

Системи екстренного оповіщення та системи попередження призначені для забезпечення безпеки людей. Однак у них є кілька незначних відмінностей, на які варто звернути увагу.

Бездротові екстрені сповіщення (WEA) - ця система, націлена на певну географічну область і надсилає SMS-повідомлення на всі мобільні пристрої в цільовій області. Бездротові оповіщення про надзвичайні ситуації настільки ж ефективні, як і SMS-повідомлення: вони можуть порушити беззвучний режим телефону та надіслати власнику чітке звукове повідомлення.

Зовнішні системи оповіщення - це системи, які розташовані в громадських місцях. Вони вмикають гучну сигналізацію та попередження щоразу, коли регіону загрожує небезпека. Такі сповіщення інформують людей, що їм не можна виходити з дому,а потрібно йти в бомбосховища тощо. Єдина проблема зовнішньої системи оповіщення полягає в тому, що не всі її почують. Ті, хто має проблеми зі слухом, не почують сигналу будильника і потрапляють у небезпеку.

EAS – це система оповіщення населення, яка передає інформацію про надзвичайні ситуації через радіо та телевізійні передачі. Федеральне агентство з

управління надзвичайними ситуаціями відповідає за надсилання екстрених повідомлень. Однак, президент також може вирішувати, яке послання і коли передати.

Рисунок 1.1 Класифікація надзвичайних ситуацій за походженням

Класифікація НС необхідна для оперативного вживання заходів для їх локалізації, навчання персоналу, фінансування, розподілу функцій та обов’язків, контролю і т.п. Відповідно до причин походження є такі види НС:

- природного характеру – це небезпечні геологічні, метеорологічні, гідрологічні явища, деградація ґрунтів або надр, природні пожежі, зміна стану повітряного басейну, інфекційні захворювання людей, сільськогосподарських тварин, масове ураження сільськогосподарських рослин хворобами або шкідниками, зміна стану водних ресурсів і біосфери;
- техногенного характеру – це транспортні аварії (катастрофи), пожежі, неспровоковані вибухи або їх погроза, аварії з викидом (погрозою викиду)

небезпечних хімічних, радіоактивних, біологічних речовин, раптове руйнування споруджень і будов, аварії на інженерних мережах і спорудженнях життєзабезпечення, гідродинамічні аварії на греблях і дамбах;

- соціальні – пов’язані з протиправними діями терористичної та антиконституційної направленості; здійснення або реальна погроза терористичного акту (збройний напад, захоплення й утримання важливих об’єктів, ядерних установок і матеріалів, систем зв’язку й телекомунікацій, напад або замах на екіпаж повітряного або морського судна), крадіжка (спроба крадіжки) або знищення суден, установлення вибухових пристроїв у громадських місцях, пропажа (крадіжка) зброї, виявлення застарілих боєприпасів;
- воєнні – пов’язані з наслідком застосування зброї масового ураження або звичайних засобів ураження, під час яких виникають вторинні фактори ураження населення внаслідок руйнування атомних і гідроелектричних станцій, складів і сховищ радіоактивних та токсичних речовин і відходів, нафтопродуктів, вибухівки, сильнодіючих отруйних речовин (СДОР), токсичних відходів, транспортних та інженерних комунікацій [3].

1.2. Автоматизовані системи сповіщення

Автоматизовані системи – це системи в яких частину керуючої функції виконує людина, а все інше це сукупність керованого об’єкту та автоматичних керуючих пристроїв. Автоматизована система забезпечую вироблення рішень на основі автоматизації інформаційних процесів у різних сферах діяльності.

Інформацією у сфері захисту населення і територій від НС становлять відомості про НС техногенного та природного характеру, що прогнозуються або виникли з визначенням їх класифікації, меж поширення і наслідків, а також способи та методи реагування на них [4].

Центральні та місцеві органи виконавчої влади зобов'язані надавати населенню через засоби масової інформації оперативну і достовірну інформацію про стан захисту населення і територій від НС, про їх виникнення, методи та способи їх захисту, вживання заходів що до забезпечення безпеки

Оповіщення про загрозу виникнення НС і постійне інформування населення про них забезпечується шляхом:

- завчасного створення і підтримки в постійній готовності автоматизованих систем централізованого оповіщення;
- організаційно-технічного об’єднання територіальних систем централізованого оповіщення із системами оповіщення на об’єктах господарювання;
- завчасного створення та організаційно-технічного з’єднання з системами спостереження і контролю постійно діючих об’єктових, локальних систем оповіщення та інформування населення в зонах можливого катастрофічного паводка, районах розміщення радіаційних і хімічних підприємств, інших об’єктів підвищеної небезпеки;
- централізованого використання загальнодержавних і галузевих систем зв’язку, радіо, телевізійного оповіщення, радіотрансляційних мереж та інших технічних засобів передачі інформації;

Сфера застосування автоматизованих систем раннього виявлення надзвичайних ситуацій та оповіщення:

- Цей стандарт установлює типи й загальні технічні вимоги до автоматизованих систем раннього виявлення техногенних і природних надзвичайних ситуацій та оповіщення працівників потенційно небезпечного об’єкту, посадових осіб, відповідальних за стан техногенної безпеки, органів виконавчої влади та місцевого самоврядування і населення.
- Вимоги цього стандарту поширюються на автоматизовані системи раннього виявлення техногенних і природних надзвичайних ситуацій та оповіщення, призначені для обладнання потенційно небезпечних об'єктів та будівель,

інженерних споруд та мереж, розташованих на територіях з ризиком прояву небезпечних природних процесів.

- Цей стандарт застосовують організації та підприємства, які проектують, виготовляють, монтують чи експлуатують такі системи.
- Цей стандарт не поширюється на спеціальні автоматизовані системи раннього виявлення надзвичайних ситуацій та оповіщення для атомних електростанцій

За призначенням автоматизовані системи раннього виявлення надзвичайних ситуацій та оповіщення поділяють на такі типи:

- АСРВО хімічно небезпечних об'єктів, на яких підлягають спостереженню та контролюванню;
- АСРВО вибухонебезпечних об'єктів, на яких підлягають спостереженню та контролюванню;
- АСРВО радіаційно небезпечних об'єктів (крім атомних електростанцій), на яких підлягають спостереженню та контролюванню;
- АСРВО біологічно небезпечних об'єктів, що пов'язані з біохімічним, біологічним і Фармацевтичним виробництвом, на яких підлягають спостереженню та контролюванню небезпечні біологічні чинники, що зазначають у технологічній документації на конкретне виробництва;
- АСРВО гідротехнічних споруд, на яких підлягають спостереженню та контролюванню;
- АСРВО будівель та споруд (у тому числі з покрівлею площею понад 1000 кв. м, виготовленою з використанням квантових і аркових конструкцій), на яких підлягають спостереженню та контролюванню;
- АСРВО контролю будівель, інженерних споруд та мереж, розташованих на територіях з ризиком прояву небезпечних природних явищ і процесів, на яких підлягають спостереженню та контролюванню [5];

Загальні вимоги:

У разі виявлення загрози або виникнення надзвичайної ситуації СРВНСО повинна:

- автоматично здійснювати інформування про виявлену загрозу відповідальних осіб, на яких покладено виконання певних дій щодо недопущення виникнення НС або мінімізації негативних наслідків у разі її виникнення;
- за командою оператора СРВНСО здійснювати оповіщення та передавання до СЦТПС відповідних тривожних сигналів разом із ідентифікатором формалізованого в електронних картках аварії прогнозованого сценарію розвитку НС, а за відсутності реагування оператора – автоматично відповідного найгіршого сценарію розвитку НС.

Для забезпечення оповіщення працівників об'єкта та населення у разі виникнення НС регіонального або державного рівня СРВНСО повинна забезпечити необхідне резервування і дублювання, бути технічно сполучена з територіальною автоматизованою системою централізованого оповіщення населення.

СРВНСО та суміжні системи повинні програмно і апаратно суміщатись із ієрархічними структурами вищого рівня та між собою.

СРВНСО повинна видавати відповідні сигнали до технічних засобів систем та устаткування, що не входять до складу СРВНСО, але які пов'язані із забезпеченням безпеки людей на об'єкті при загрозі або виникненні НС, а саме:

- ліфтів, ескалаторів, траволаторів, що повинні працювати в режимі НС;
- систем вентиляції та кондиціювання, що вимикаються (вмикаються) у разі виникнення НС;
- систем керування устаткуванням, яке має припиняти роботу або змінювати алгоритм роботи у разі виникнення НС;
- турнікетів, дверей, оснащених системою контролю доступу, які потребують необхідного розблокування у разі виникнення НС.

СРВНСО повинна автоматично здійснювати контроль:

- за діями оператора СРВНСО щодо обробляння отриманих з СРВНСО сигналів і повідомлень;
- працездатністю основних складових, каналів зв'язку та стану електроживлення.

Повідомлення, які використовуються для оповіщення населення, повинні передаватись державною мовою та мовою, якою користується більшість населення у регіоні. Якщо є загроза населенню, яке проживає (перебуває) у зоні ураження (можливого ураження) у разі виникнення надзвичайної ситуацій на потенційно небезпечному об'єкті (об'єкті підвищеної небезпеки), забезпечується локальне оповіщення. Приклад структури СРВНСО у виді схеми (рис. 1.2).

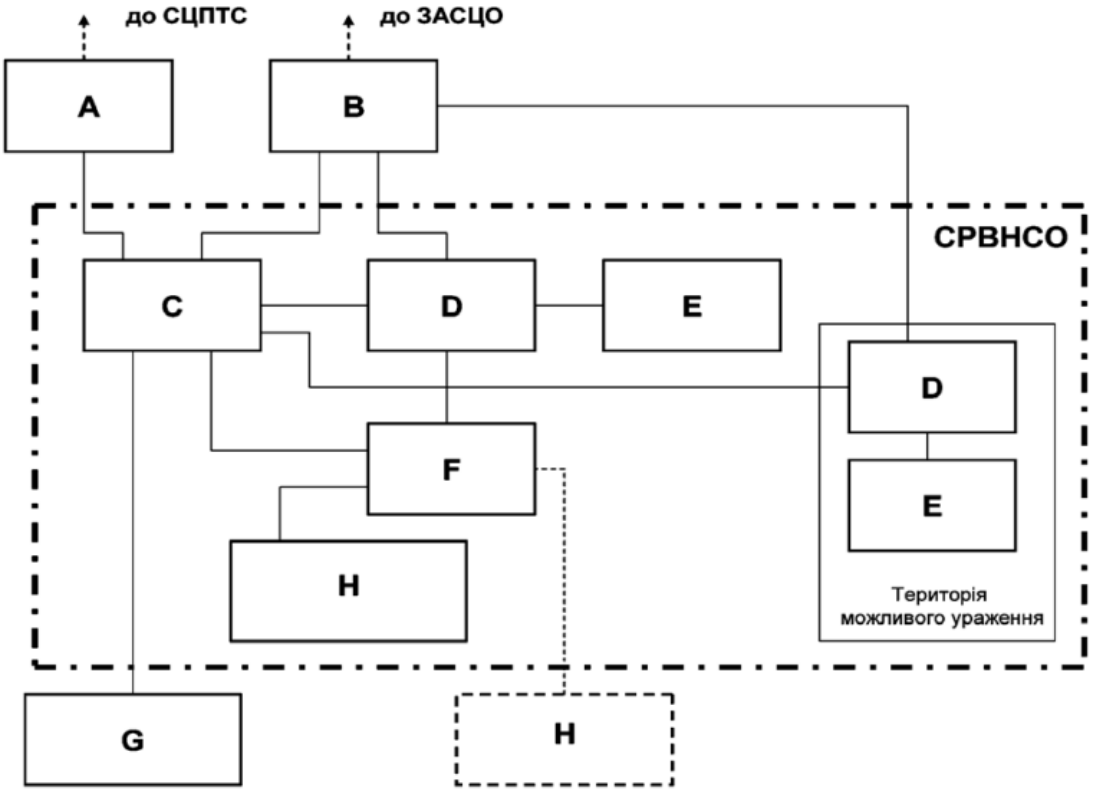

Рисунок 1.2 Структурна схема СРВНСО [6]

1.3. Аналіз існуючих програмних рішень

Після початку повномасштабної війни в України розпочався стрімкий розвиток систем інформування, з’явилася велика кількість різних додатків які в онлайн режимі інформують про повітряну тривогу. Все починалося з простих додатків в яких можна було вибрати свою область і отримувати звукові сигнали інформування, ці додатки постійно розвивали свій функціонал щоб бути більш зручними та корисними. На даний момент можна вибирати типи інформування такі як: вібросигнал, звукове сповіщення, текстове сповіщення. З’явився більш детальний функціонал місця де ви знаходитесь, було додано можливість регулювання рівню гучності звукового сповіщення, було додано мапу повітряної тривоги по областям України.

ДСНС України також почало тестування сповіщення різних типів, це сповіщення має текстовий та звуковий сигнал на телефон яке не має залежності від підключення до мережі інтернет. Розвиток цієї системи сповіщення буде корисним не тільки під час війни але й після неї, він буде інформувати населення про надзвичайні ситуації природнього, екологічно та техногенного характеру. Тестування системи буде завершено у грудні 2022 року та буде сповіщати у безперебійному режимі про загрозу життя та здоров’я людини.

На даний момент реалізовано велику кількість різних корисних додатків які допоможуть вам під час військового стану.

Перелік додатків:

- «Дія»
- «Київ цифровий»
- «TacticMedAid»
- «Перша мобільна допомога»
- «Helsi»
- «Доступні ліки»
- «Повітряна тривога»
- «MineFree»

- «Bachu»
- «Zrada»
- «Easy Way»
- «Travel post»
- «GeoZila»

1) Додаток «Київ цифровий»

Додаток був створений для покращення життя водіїв та жителів міста Київ. Насамперед там було реалізовано транспортну карту та погодинне паркування, уже після початку війни в Україні в додаток було добавлено функціонал маркерів укриття на відділеній вкладці, а також систему сповіщення про повітряну тривогу. Київ цифровий може дати наступну інформацію:

- Адресса укриття;
- Район;
- Тип укриття;
- Вид укриття;
- Тип будівлі;
- Власник;
- Форма власності;
- Номер телефону місця знаходження ключів;
- Наявність пристосування для осіб з інвалідністю та інших маломобільних груп;
- Сповіщення про повітряну тривогу ;
- Та інформаційне вікно з різною інформацією щодо надзвичайного стану військового характеру;

Одним із недоліків даного додатку є відображення даних тільки по місту Київ, також немає змоги редагування маркерів аунтифікованим користувачами, немає маркерів інших важливих галузь таких як аптеки, лікарні і тому подібне. Перевагой додатку є функціонал сповіщення про повітряну тривогу та функціонал

інформативної вкладки яка допоможе людям корисною інформацією у потрібний момент.

2) Додаток «Повітряна тривога»

Додаток був створений після початку війни, та мав ціль інформувати людей про повітряні тривоги, але його розвиток на цьому не закінчився і на даний момент додаток має наступний функціонал:

- Сповіщення про повітряну тривогу;
- Карту повітряних тривог по областям України;
- Функціонал налаштування гучності сповіщення;
- Функціонал вибору області або громади та його зміну при необхідності ;
- Має посилання на різні чат боти;
- Простий та зрозумілий інтерфейс та налаштування;

Додаток продуманий та має багато корисного функціоналу, є дуже комфортним в використанні та має потрібні налаштування

Нажаль в додатку немає реалізації пошуку найближчого укриття та посилань на такі додатки, одним з його мінусів є обмежене використання яке заточене тільки під сповіщення про повітряні тривоги.

3) Додаток «ДеУкриття»

Додаток створений для пошуку укриття в місті Тернопіль. Є змога фільтрування укриття в залежності від рівня небезпеки, а саме 3 види:

- Найпростіше укриття;
- Сховище;
- Протирадіаційне укриття;

Також до кожного укриття є інформаційне вікно яке надає змогу побачити таку інформацію:

- Тип укриття;
- Власник ;
- Місцезнаходження в виді назви вулиці та номеру будинку;
- Доступ до туалету;

- Доступ до WIFI;

Мінуси додатку:

- При фільтруванні можна відобразити тільки один тип сховищ;
- Додаток не сповіщає про повітряну тривогу ;
- Немає інформаційних вікон з порадами дій в відповідній ситуації;

Плюси додатку:

- Є можливість фільтрації укриття в залежності від небезпеки ;
- Для кожного виду укриття реалізовано свій дизайн маркеру;
- В програмі можна переглянути весь список укрить ;
- Реалізовано кнопку яка автоматично відкриває маршрут до укриття в Google maps ;

Проаналізовані мобільні додатки мають корисний функціонал, але швидка та доступна робота цих додатків дуже важлива, так як вони можуть врятувати життя людини за допомогою вчасного інформування, ці додатки повинні бути більш універсальними та мати більше функціонала для заощадження часу людини під час надзвичайної ситуації. Розроблений додаток поєднує в собі інформування щодо повітряної тривоги та пошук найближчого укриття, але крім цього в ньому можна буде знайти багато корисної інформації. Якщо порівнювати це з уже реалізованими проектами, додаток дасть змогу повідомити про повітряну тривогу а також знайти усю необхідно інформацію щодо дій під час надзвичайної ситуації, також не потрібно шукати спеціальний додаток для пошуку укриття та іншої корисної інфраструктури.

Висновок до розділу 1

Проаналізувавши аналогічні мобільні додатки, можна дійти висновку, що кожен з них має свої недоліки: це – відсутність мапи, з відображеними безпечними місцями, функціонал пошуку найближчого укриття від поточної геолокації користувача та інформаційні сторінки.

Один із найважливіших факторів додатків-аналогів є їх недостатній функціонал. Необхідно реалізувати мобільний додаток, який швидко та зручно інформує населення під час надзвичайної ситуації військового характеру та містить в собі тільки перевірену інформацію.

Отже, було з’ясовано, який саме функціонал повинен бути реалізованим в мобільному додатку:

- можливість перегляду детальної інформації про маркер;
- функціонал пошуку найближчого укриття;
- можливість користувачеві прокладати маршрут до укриття;
- підтримка актуальності бази даних;
- інформаційні сторінки з наявною інформацією про перелік дій під час повітряної тривоги та перелік потенційно небезпечних об’єктів;
- push сповіщення про повітряну тривогу;
- push сповіщення про розташування найближчого укриття.

Однією з переваг мобільного застосунку буде безкоштовне використання із повним доступом. Було розроблено оптимальний план реалізації додатку та його модулів для досягнення найкращого результату.

РОЗДІЛ 2 ВИБІР ТЕХНОЛОГІЇ ДЛЯ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ ІНФОРМУВАННЯ НАСЕЛЕННЯ В УМОВАХ НАДЗВИЧАЙНИХ СИТУАЦІЙ

2.1. Мова програмування Dart

Dart - однопоточна мова програмування, що накладає низку обмежень. Є можливість писати асинхронний код, але класу Thread тут немає. Натомість використовується поняття Isolate. На відміну від звичайного потоку «ізоляти» не поділяють загальну пам'ять, а взаємодіяти між собою можуть у вигляді повідомлень.

Dart має свій менеджер пакетів — pub, який дозволяє встановлювати пакети, що існують у сховищі. У більшості випадків немає потреби взаємодіяти з ним безпосередньо. Досить просто прописати як залежності проекту пакет, який необхідно встановити.

Середовище виконання та компілятори Dart підтримують комбінацію двох важливих функцій для Flutter: швидкий цикл розробки на основі JIT, який дозволяє змінювати форму та гаряче перезавантаження зі збереженням стану мовою з типами, а також AOT-компілятор, який генерує ефективний код ARM для швидкого запуску та передбачуваної продуктивності виробничих розгортань [7].

Складання сміття в програмуванні - одна з форм автоматичного керування пам'яттю. Спеціальний процес, званий збирачем сміття, періодично звільняє пам'ять, видаляючи об'єкти, які вже не будуть потрібні додатками.

У Dart використовується просунута схема збору сміття та виділення пам'яті з урахуванням поколінь об'єктів. Вона особливо швидко виділяє пам'ять для великої кількості об'єктів із коротким життєвим циклом. Це ідеально для реактивних інтерфейсів користувача як Flutter, де незмінне дерево віджетів перезбирається для кожного кадру.

Завдяки такому збирачеві Flutter приємний для розробки інтерфейсів користувача в декларативному. стиль. Ми використовуємо конструктори, створюючи

об'єкти, та описуємо за допомогою них верстку. Причому самі віджети досить легковажні і є лише інформацію для малювання. Самим малюванням займаються вже інші верстви.

Використовувати Dart можна різними способами:

1. Трансляція JavaScript, що підтримується деякими сучасними браузерами (Chrome,Safari 5+, Firefox 4+);

2. Виконання коду безпосередньо у віртуальній машині на серверній стороні;

3. Використовувати вбудований редактор Dartboard для написання, редагування та виконання простих скриптів у вікні браузера.

Переваги мови Dart:

1. Класи та інтерфейси, які забезпечують простий та зрозумілий механізм для чітко визначених API. Ці конструкції також забезпечують інкапсуляцію та повторне використання методів та даних;

2. Додаткові типи (optional types), завдяки яким можна переходити від найпростіших додатків до складних модульних систем, а також використовувати дебаггери для перевірки типів;

3. Інструментарій. Планується створити безліч додаткових програм на допомогу розробнику.

Проблеми web-розробки, які вирішені у Dart:

1. Невеликі скрипти, які часто використовуються у великих web-додатках у неструктурованому вигляді, важко підтримувати та налагоджувати. Монолітні програми проблематично розбити на частини та доручити розробку кожної з частин окремій команді програмістів. Чим більше стає web-додаток, тим важче його розвивати та підтримувати;

2. Скриптові мови популярні в основному через свою легкість у написанні та можливості дуже швидко створити працюючий код. Ціною такого підходу є проблема зі сприйняттям логіки роботи за структурою самого коду, що вимагає написання коментарів для деяких речей, очевидних у строго типизованих мов. У результаті стороннім розробникам важко розібратися з чужим кодом, забезпечити його підтримку та доопрацювання;

3. Для існуючих мов розробник змушений вибирати або мови зі статичною типізацією, або динамічною. Традиційні мови зі статичною типізацією вимагають використання важкого інструментарію та жорсткого дотримання стилю кодування, через що програміст може почуватися занадто обмеженим, а мова відчуватиметься як позбавлений гнучкості [8].

2.2. Фреймворк Flutter

Flutter - це крос-платформний фреймворк з єдиною кодовою базою, що працює мовою програмування Dart. Запущений тільки в 2018 році Google, Flutter зарекомендував себе як зручний набір інструментів, легкий для створення анімації та якісних компонентів інтерфейсу користувача. Незважаючи на те, що Google недавно почав використовувати крос-платформу, Flutter надає плавну анімацію та зручні елементи інтерфейсу [9-10].

Він поєднує в собі простоту розробки з високою продуктивністю. Це дає розробникам простий спосіб створити і розгорнути візуально привабливі, спочатку скомпіловані програми для мобільних (iOS, Android), Web і все з використанням єдиної кодової бази.

Flutter доступний програмістам, знайомим з об'єктно-орієнтованими (класами, методами, змінними тощо) та імперативними (циклами, умовами тощо) концепції програмування.

Кросплатформні фреймворки, схожі на React Native і Flutter, обговорювалися та впроваджувалися різними компаніями кілька разів раніше. Проте жодного з них достатньо для виконання вимог промислового розвитку. Незважаючи на всі ті невдалий попередник, React Native і Flutter, за підтримки Facebook і Google привертають величезну увагу, і люди з оптимізмом дивляться на їхні перспективи [11].

Екосистема пакетів Flutter підтримує широкий спектр апаратних засобів (камера, GPS, мережа, сховище) та послуг (платежі, хмарне зберігання, автентифікація, реклама).

Flutter відрізняється тим, що для створення мобільних додатків йому не потрібно покладатися ні на технологію веб-браузера, ні на набір віджетів, що поставляються з кожним пристроєм. Крім того, Flutter має тільки тонкий шар коду C/C++. Flutter реалізує велику частина своєї системи (композиція, жести, анімація, фреймворк, віджети і т. д.) в Dart (сучасний стислий об'єктно-орієнтована мова), до якої розробники можуть легко звернутися для внесення змін до код. Це дає розробникам величезний контроль за системою, а також значно знижує планку доступності для більшості систем. Приблизно раз на три місяці Flutter постачає оновлення, які покращують стабільність та продуктивність.

Flutter включає:

- оптимізований двигун 2D-рендерингу для мобільних пристроїв з відмінною підтримкою тексту;
- сучасний фреймворк у стилі React;
- великий набір віджетів, що реалізують матеріальний дизайн та стиль iOS;
- API для модульних та інтеграційних тестів;
- API взаємодії та підключаються модулі для підключення до системи та сторонніх SDK;
- Dart DevTools для тестування, налагодження та профілювання вашої програми;
- інструменти командного рядка для створення, складання, тестування та компіляції додатків.

Flutter framework підтримує безліч різних інструментів, включаючи Android Studio та Visual Studio Code. Він також забезпечує підтримку створення додатків із командного рядка. Dart DevTools, новий інструмент налагодження є більш гнучким. Віджетинспектор допомагає візуалізувати та досліджувати ієрархію дерева, що використовується для візуалізації інтерфейсу користувача.

Flutter побудований на таких мовах програмування, як C, C++, Dart та Skia (2D-движок рендерингу). Віджети є будівельними блоками будь-якої програми Flutter і можуть бути тематизовані, щоб виглядати як рідні компоненти UI Android (Material)

або iOS (Cupertino). Віджети відображаються на полотні Skia з підтримкою просунутої анімації та розпізнавання жестів (рис. 2.1).

Рисунок 2.1 Системна архітектура фреймфорку

Двигун Flutter містить ключові технології Skia. 2D-графічну бібліотеку рендерингу та мову Dart VM в конкретної платформи. Будь-яка оболонка реалізує відповідні API платформи та обробляє події життєвого циклу застосування системи.

Використання мови Dart дозволяє Flutter заздалегідь скомпілювати вихідний код до свого коду. Код двигуна C/C++ компілюється на Android NDK (Native Development Kit) або iOS LLVM (Low Level Virtual Machine). Обидві частини упаковані в проект Runner Android та iOS, в результаті чого з'являється apk- або ipa-файл відповідно.

При запуску програми будь-який рендеринг, введення або подія делегується компільованим двигуном Flutter і коду програми. Необхідність упаковки двигуна з файлом apk- або ipa-програми в даний час призводить до збільшення розміру програми до 4 MB [12].

2.3. Google Maps

Проект Google Maps – це картографічний сервіс, створений компанією Google спільно з NASA і що відрізняється високою точністю та глибокою деталізацією географічних об'єктів. Сервіс надає ефективний API (з кінця травня 2009 року по теперішній час чинна версія – v3), який на рівні JavaScript-класів, їх властивостей та методів дозволяє гнучко маніпулювати географічними об'єктами: промальовувати та центрувати фрагменти картки, розставляти маркери з позначками, виконувати різноманітні розрахунки .

Візуалізація географічних об'єктів за допомогою Google Maps API реалізується у вигляді земної поверхні з використанням різних шарів зображень, у тому числі – тривимірних графічних уявлень будівель та рельєфу.

Для використання JavaScript API Google Maps потрібен спеціальний ключ-ідентифікатор, доступний через обліковий запис користувача в Google. Застосування ключа API дозволяє відстежувати показники використання API

Google Maps і за необхідності дає можливість компанії Google збирати потрібну статистику. Якщо показники використання API Google Карт додатком перевищать передбачені обмеження на використання (на сьогоднішній день це 25 тисяч запитів на день), то для замовлення додаткових квот завантаження інтерфейсу повинно проводитись з використанням ключа API [13].

Для підключення Google Maps API використовується посилання на JavaScript-файл, який має такий вигляд:

<script type="text/javascript"src="http://maps.goo

gleapis.com/maps/api/js?key=Key&sensor=false"></script>

Параметр key приймає значення унікального ідентифікатора користувача. Параметр sensor вказує, використовується чи ні в програмі датчик позиціонування користувача (наприклад, локатор GPS) [14].

2.4. Android Studio

Android Studio – це продукт компанії Google. Розроблений програмним забезпеченням IntelliJ IDEA компанією JetBrains та є офіційним засобом розробки додатків Android. На сьогоднішній день останньою версією цього додатка є версія 2.2. Середовище розроблено для Windows, OS X та Linux. Для написання програмного коду використовується мова Java. Інтерфейс розробляється методом drag-n drop, а також із використанням XML. При розробці інтерфейсу, для зручності, використовуються шаблони для вирішення задачі при виконанні програми. Додаток має бібліотеку з інтерфейсом, що має вигляд випадаючого дерева, для якого необхідно багато місця в загальному інтерфейсі, інакше інформація не читається. З вікном налагодження аналогічна ситуація [15].

Функціонал Android Studio не має можливості підключати додаткові плагіни. У порівнянні з іншими засобами розробки Android Studio потребує хороших технічних даних ПК. Для цього продукту ОЗП має бути не менше 2 гігабайти, а для нормальної роботи з цією програмою ОЗП не повинно бути менше 8Гб і це не є проблемою для комп'ютерів з оптимальними системними параметрами. Для старих систем ПК це середовище працює дуже повільно. До сервісів контролю версій відсутнє пряме підключення, а це призводить до складнощів роботи з додатком. Для емуляції Android-пристрою є вбудований модуль. Вимога спеціальних ресурсів підвищує вимоги ПЗ [16].

За ці роки Android Studio зазнала великого числа оновлень, переробок, доповнень і зараз є найкращим середовищем для розробки мобільних додатків у порівнянні з іншими, як, наприклад, Eclipse.

Головним плюсом Android Studio можна назвати вбудований комплект засобів розробки, що дозволяє створювати програми для певних версій систем, використовуючи спеціальні елементи, недоступні в деяких версіях програм, програм – SDK. Android Studio вже містить всі версії API в базі і при перевірці коду автоматично визначає, яка версія API потрібна для роботи певних фрагментів, і при необхідності автоматично встановить усе необхідне для роботи.

Наступною перевагою є зручний конструктор інтерфейсів. В Android Studio міститься велика кількість емуляторів різних видів пристроїв, що дозволяє створювати інтерфейс програм, розміщувати елементи та миттєво переглядати результат на різних пристроях – від старих моделей смартфонів, до планшетів і телевізорів. Крім цього, всі елементи виглядатимуть аналогічно тому, як вони виглядають на конкретному пристрої. Крім цього, велику роботу було проведено в редакторі коду. При описі інтерфейсу, коду дій, подій, Android Studio автоматично відокремлює різні фрагменти один від одного, роблячи код більш читаним, виділяє рядки, які використовуються в застарілих версіях API, автоматично вказує зміст ресурсів і 107 даних, що зберігаються в файлах, що підключаються, нагадує про можливі помилки в тих або інших місцях і так далі. Завдяки цьому написання та редагування коду спрощується в рази - саме середовище розробки підказує і практично виправляє помилки, недоробки, і дає всю необхідну інформацію програмісту, захищаючи його від пошуку даних у проекті [17].

Також в Android Studio зручно реалізовано весь інтерфейс для роботи, меню структури проекту та пошук помилок. Весь інтерфейс можна налаштовувати, непотрібні вікна та панелі прибирати, згортати. Весь необхідний функціонал завжди буде під рукою. Вікно структури проекту завжди відображатиме список всіх файлів даного проекту, над яким ведеться розробка безпосередньо в даний момент, не відображаючи інші ранні роботи. Пошук помилок, їх відстеження ведеться через вікно «логів», що дозволяє стежити за роботою різних процесів, потоків і додатків і бачити, якому конкретно етапі трапилася помилка. Це дуже корисно під час створення високонавантажених та об'ємних проектів та додатків.

Всі вищезгадані переваги роблять Android Studio найпопулярнішим середовищем розробки мобільних додатків для операційної системи Android. Простий і зручний інтерфейс, зміст всього необхідного функціоналу, можливість вибору версій API та виду пристрою – все це допомагає початківцям і навіть досвідченим програмістам під час роботи, значно прискорюючи їхню продуктивність [18].

2.5. Firebase

Google Firebase – сервіс, що надає різні хмарні послуги та інструменти для розробки: база даних реального часу, хмарні функції, хостинг, сервіс для автентифікації користувачів та багато іншого. Для створення автоматизованої системи охорони праці використовується інструмент Cloud Firestore, це хмарна база даних NoSQL, що дозволяє зберігати та синхронізувати дані.

При проектуванні архітектури програми використовуються парадигма з 5 основних принципів (SOLID), методика розробки програмного коду, створена Kent Beck у 2003 році (TDD) та парадигма програмування, розділ дискретної математики (FP) [19].

Принципи SOLID із парадигми методології програмування, заснованої на представленні програми у вигляді сукупності об'єктів (ООР) дозволяють створювати структурований, адаптивний і код, що легко масштабується.

TDD підходить для розробки та проведення юніт-тестів – тести, які перевіряють свій ж функціонал, не перевіряючи при цьому роботу системи загалом. Такий підхід розробки дозволяє створити:

- Надійний та розширюваний код;
- Тести, які стануть документацією до програмного продукту, оскільки покривають велику функціональну частину програми;
- Контрольований код, захист від помилок.

Основне завдання TTD – розбити складну програму на частини. Тому виконуючи такий цикл для кожного маленького функціонального модуля програми, можна створювати великий керований програмний код [20].

Firebase чудово підходить для створення прототипів додатків, оскільки дозволяє швидко будувати бекенд. Даний сервіс надає велику кількість інструментів – від хмарної бази даних та хмарних функцій, аналітики. Хмарні функції дозволяють розробникам перенести API додатки в хмару і викликати в потрібних місцях.

Сервіс умовно безкоштовний, тобто до досягнення певної кількості читань та записів даних. Звідси можна дійти невтішного висновку, що потрібно правильно

організовувати читання даних, оскільки це сервісом буде стягуватися додаткова плата. Дані, що зберігаються є прив'язаними до Firebase. Це проблема практично всіх «Бекенд як сервіс» постачальників, що немає можливості та інструментів для перенесення даних на іншу платформу [21].

Для реалізації серверної частини програми використовувалася хмарна платформа Firebase. У розробленому додатку використовуються такі

компоненти Firebase:

– Firebase Authentication – для аутентифікації користувачів у додатку;

– Cloud Firestore – NoSQL база даних – необхідна для зберігання питань, відповідей, інформації користувачів (логіна, персональних даних);

– Firebase Realtime – це хмарна база даних (дані зберігаються у форматі JSON та синхронізуються в режимі реального часу з кожним підключеним клієнтом), використовується для зберігання діалогів користувачів

– Cloud Storage (хмарне сховище) – використовується для зберігання фотографій, документів та фотографій профілів користувачів.

База даних Cloud Firestore. У схемі бази даних програми представлені чотири колекції: category, posts, chats, users, які містять документи з інформацією про посади з питаннями, їх категорій, діалогів та користувачів відповідно.

База даних Realtime Database. У схемі бази даних програми представлена колекція chats, яка містить колекцію діалогів.

Колекція Chats містить документи діалогів, яких зберігаються документи повідомлень.

Сховище файлів Cloud Storage. У схемі сховища файлів програми представлені чотири папки для зберігання: avatars, comments, documents, photos, які містять документи та фотографії, що публікуються користувачами програми [22].

2.6. Business Logic Component

Найкращий спосіб підтримувати ваш код упорядкованим, чистим і придатним для обслуговування у Flutter — це мати компонент, який може посередничити те, що бачить користувач, і його логіку. Цей компонент, про який я маю на увазі, є Bloc (компонент бізнес-логіки).

Bloc – це шаблон дизайну, створений Google, щоб допомогти відокремити бізнес-логіку від рівня презентації та дозволити розробнику ефективніше повторно використовувати код.

Бібліотеку державного управління під назвою Bloc створив і підтримує Фелікс Анджело. Це допомагає розробникам реалізувати шаблон проектування Bloc у їхній програмі Flutter. Це означає, що розробник повинен знати стан програми в будь-який час. Щось має відображатися на екрані під час кожної взаємодії з програмою, щоб повідомляти користувачам, що відбувається.

Технічно для кожної взаємодії всередині програми має виникати певний стан. Наприклад, під час отримання даних програма має перебувати в стані завантаження з відображенням анімації завантаження на екрані. Коли Інтернет вимкнено, програма має відобразити спливаюче вікно, щоб повідомити користувача про відсутність підключення до Інтернету.

Хороший дизайн додатка включає в себе розгляд кожного можливого випадку та наявність стану, що випливає з цього. Отже, бібліотека Bloc відокремлює презентаційний рівень від бізнес-логіки та спрощує керування станом програми [23].

Bloc — хороший шаблон, який підійде майже для всіх типів програм. Це допомагає покращити якість коду та робить стани обробки в програмі набагато легшими.

Bloc має свої певні недоліки для тих, хто тільки починає використовувати Flutter, оскільки він використовує передові методи, такі як потокове та реактивне програмування, що може викликати певні труднощі на перших етапах.

Рисунок 2.2 Рівні в фреймворку Flutter

Патерн BLoC насправді є лише інтерфейсом навколо потоків Dart .Потоки, як і Futures, надаються dart:asyncпакетом. Потік схожий на Future, але замість того, щоб асинхронно повертати одне значення, потоки можуть давати кілька значень з часом. Якщо Future — це значення, яке буде надано врешті-решт, потік серії значень, які будуть надаватися спорадично з часом.

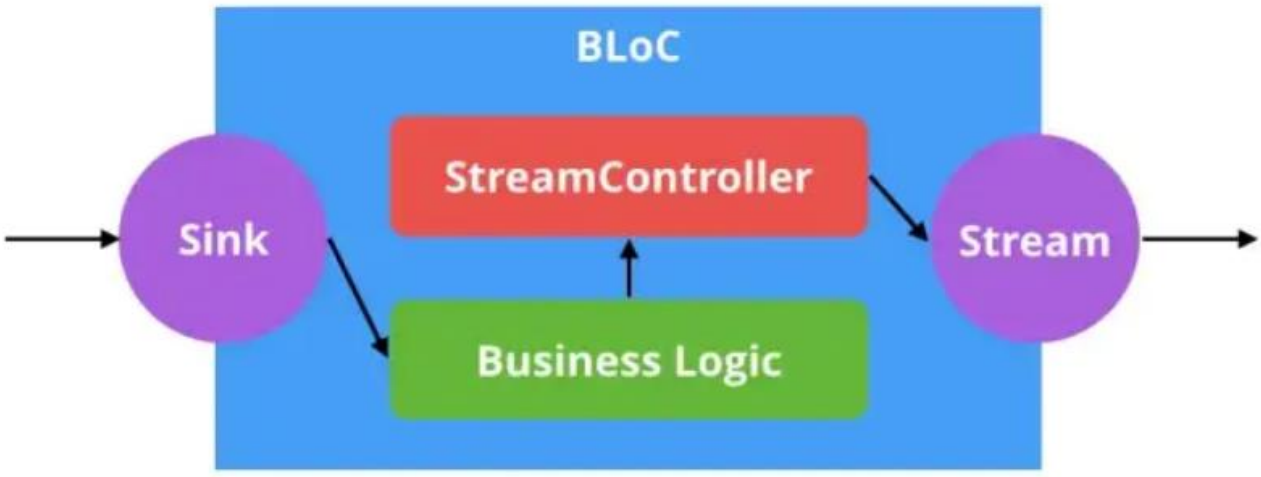

Рисунок 2.3 Патерн BLoC

Пакет dart:async містить об’єкт під назвою StreamController StreamControllers — це керуючі об’єкти, які створюють і потік, і приймач. Раковина протилежна струмку. Якщо потік дає вихідні значення з часом, приймач приймає вхідні значення з часом.

BLoC — це об’єкти, які обробляють і зберігають бізнес-логіку, використовують приймачі для прийому вхідних даних і забезпечують вихід через потоки [24].

2.7. Архітектура мобільного додатку

Перед тим, як перейти до реалізації проекту важливо продумати структуру мобільного додатку переходи між вкладками та функціонал програми (див. рисунок 2.4).

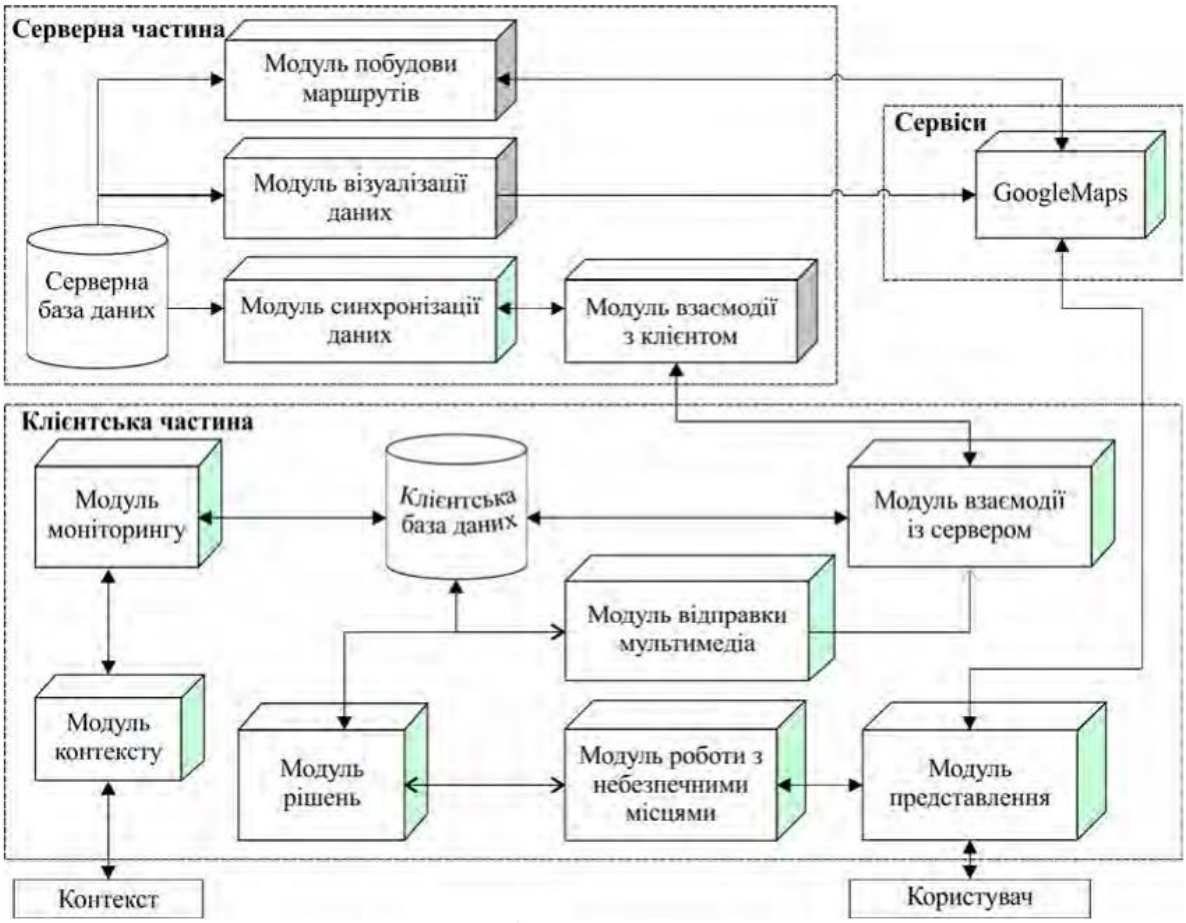

Рисунок 2.4 Структура мобільного додатку

Одним із найважливіших функціоналів даної програми є обчислення найблищого маршруту між розташуванням користувача до найблищого укриття, схематичний приклад блок-схеми можна побачити на рисунку 2.5.

Рисунок 2.5 Приклад блок-схеми для обчислення маршруту між двома точками

Не менш важливим є проектування бази в якій буде зберігатись вся необхідна для мобільного додатку інформація, в тому числі інформація про користувача, його пароль та електронний адрес (див. рисунок 2.6).

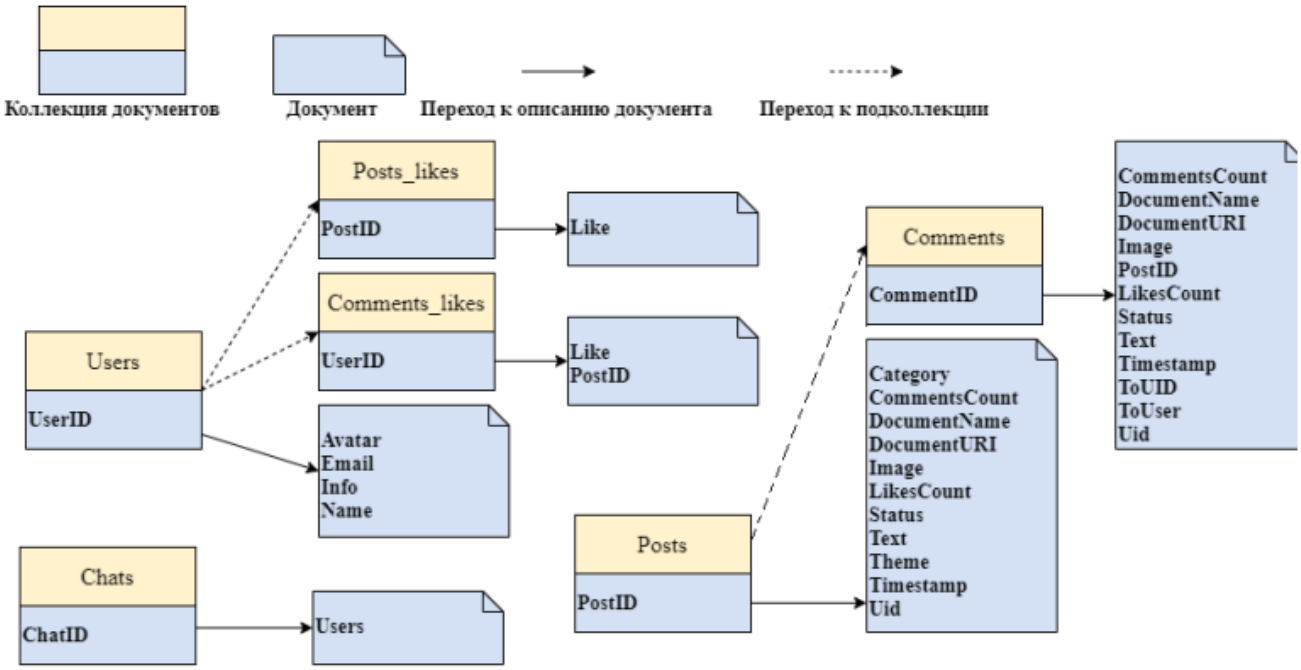

Рисунок 2.6 Схема бази даних в Firestore для мобільного додатку для інформування населення у випадку незвичайних ситуацій

Для реалізації серверної частини програми використовувалася хмарна платформа Firebase. У розробленому додатку використовуються такі компоненти Firebase:

– Firebase Authentication – для аутентифікації користувачів у додатку;

– Cloud Firestore – NoSQL база даних – необхідна для зберігання питань, відповідей, інформації користувачів (логіна, персональних даних);

– Firebase Realtime – це хмарна база даних (дані зберігаються у форматі JSON та синхронізуються в режимі реального часу з кожним підключеним клієнтом), використовується для зберігання діалогів користувачів;

– Cloud Storage – хмарне сховище – використовується для зберігання фотографій, документів та фотографій профілів користувачів

У додатку використовується автентифікація електронною поштою та паролем. Для включення функції автентифікації поштою необхідно активувати цю опцію у веб-сервісі Firebase. Коли користувач входить до програми або реєструється, він стає поточним користувачем екземпляра Auth. Екземпляр зберігає стан користувача, тому при перезапуску програми інформація користувача не втрачається. Коли користувач виходить із програми, екземпляр Auth перестає зберігати посилання на користувальницький об'єкт і не зберігає його стан.

Для авторизації у додатку використовується метод firebase.auth().signInWithEmail AndPassword (email, password). При натисканні на кнопку «Увійти» викликається метод, який надсилає введені дані: пошту та пароль. Firebase Authentication перевіряє правильність отриманих даних і повертає відповідь, токен доступу або помилку входу.

За відображення списків з потенційно небезпечними об’єктами та інфструкції відповідає компонент Flatlist. Цей компонент має безліч властивостей, такі як data, initialNumToRender, renderItem, onRefresh. Data – дані для малювання,

Список елементів. InitialNumToRender – список змальовується після отримання певної кількості елементів, використовується для оптимізації, щоб не чекати повного отримання списку елементів. RenderItem – задається розмітка для рендерингу списку. Для оновлення списку використовується жест "потягування списку вниз", який запускає функцію оновлення списку, зазначену у властивості OnRefresh.

Для підключення до сервера Firebase з автентифікацією, базами даних, хмарним сховищем та хмарними функціями необхідно встановити з'єднання з сервером Firebase. У розроблюваному додатку за допомогою пакетного менеджера Node.js потрібно встановити Firebase JavaScript SDK за допомогою наступної команди: npm і firebase. Після цього імпортувати його в проект через кодовий рядок: require (firebase).

У веб-інтерфейсі Firebase необхідно створити проект програми, що розробляється. Після чого в створеному проекті отримати об'єкт конфігурації для створення з'єднання з Firebase (див. рисунок 2.7).

Даний об'єкт конфігурації впроваджується програмним кодом мобільного додатка (див. рисунок 2.8).

Рисунок 2.7 Об’єкт конфігурації в Firebase для мобільного додатку для інформування населення у випадку незвичайних ситуацій

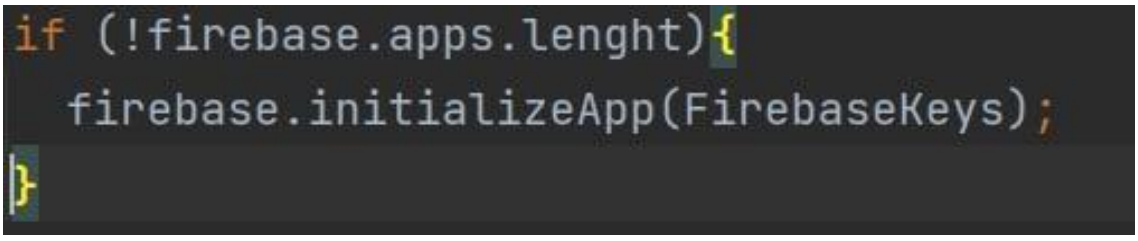

Рисунок 2.8 Підключення до сервера в Firebase для мобільного додатку для інформування населення у випадку незвичайних ситуацій

Для визначення і показу архітектури проекту потрібно реалізувати діаграму компонентів для мобільного додатку для інформування населення у випадку незвичайних ситуацій

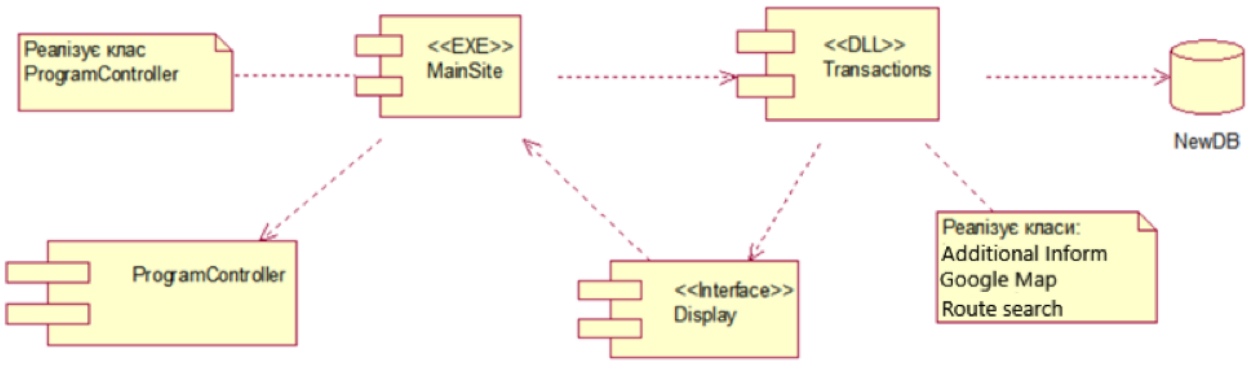

Рисунок 2.9 Діаграма компонентів для мобільного додатку для інформування населення у випадку незвичайних ситуацій

Також не менш важливим є розробка діаграми розгортання (див. 2.10).

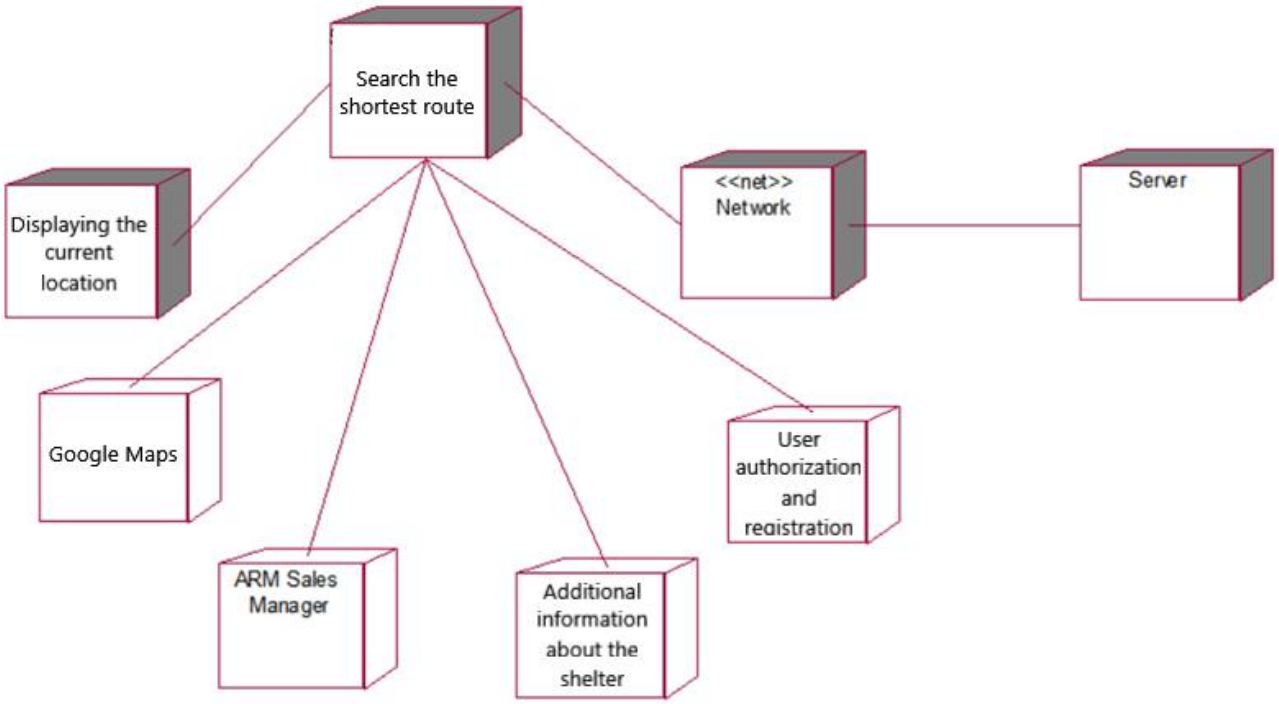

Рисунок 2.10 Діаграма розгортання для мобільного додатку для інформування населення у випадку незвичайних ситуацій

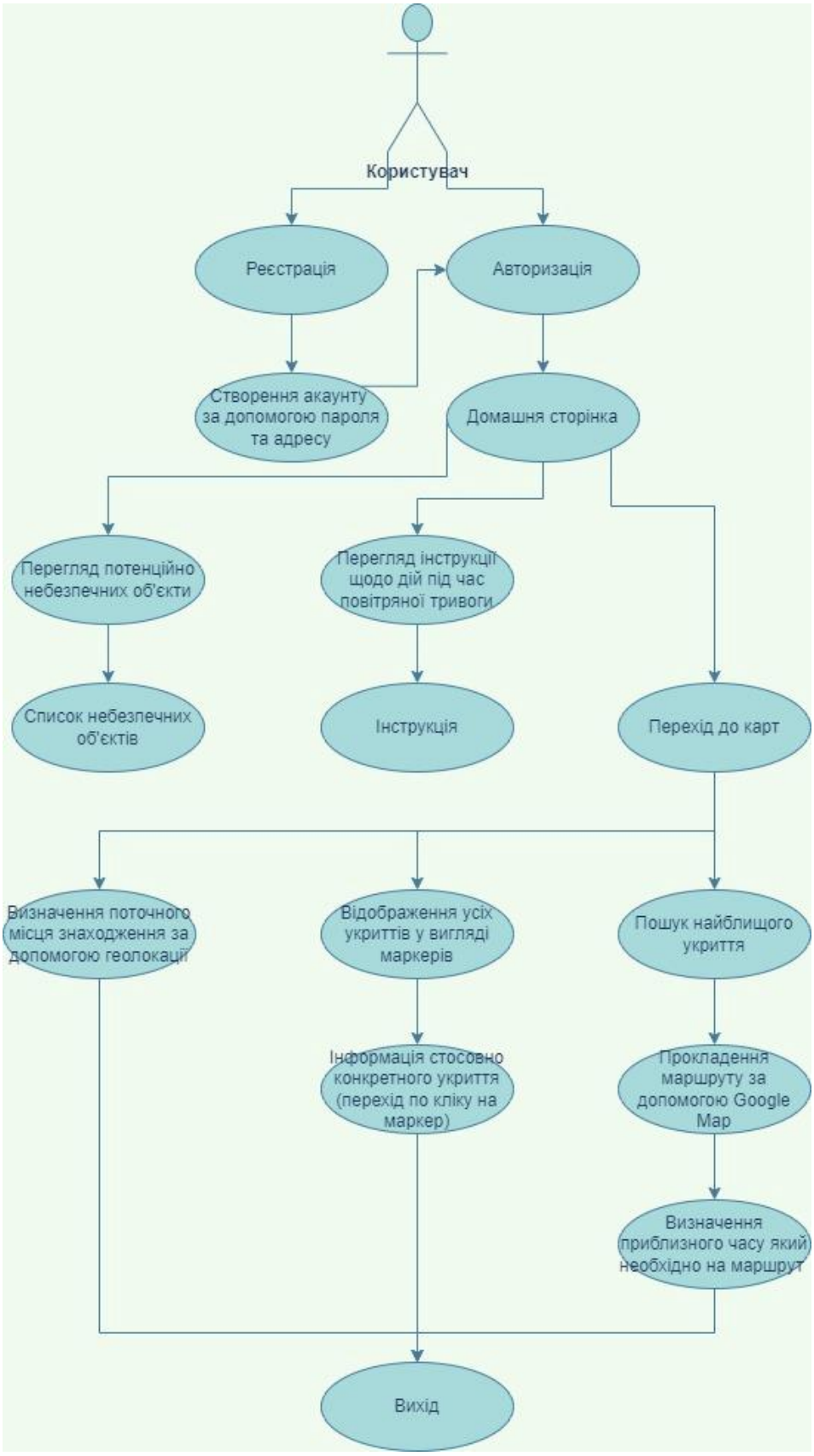

Рисунок 2.11 Приклад роботи мобільного додатку для інформування населення у випадку незвичайних ситуацій

Висновок до розділу 2

Було проаналізовано інструменти та середовище реалізації програмного продукту. Після чого було обрано найбільш відповідні та новітні засоби, які пришвидшать та збільшать продуктивність розробки проекту.

Було проаналізовано сучасні інструменти для розробки програмного додатку, описано їх плюси та мінуси, зазначені конкретні особливості різних інструментів.

Мова програмування dart та фреймворк flutter обрано через те, що Dart дозволяє Flutter уникнути необхідності в окремій мові декларативного компонування, такому як JSX або XML, або окремих конструкторів візуальних інтерфейсів.

Було наведено приклади архітектури мобільного додатку та роботи користувача з ним. Не менш важливим фактором є база даних та її вибір. Я обрав Firebase через її швидкість та простоту створення, а також легкого підключення бекенду для проекту.

РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1.Опис та основні функції середовища Android Studio

Android Studio – унікальне середовище розробки в якому можна розробляти додатки не тільки для Android та IOS а також для таких платформ як:

- Linux;
- MacOS;
- Web;
- Windows.

Це середовище доволі популярне серед розробників за можливість безкоштовного створення програмного продукту для великої кількості людей.

Розглянемо створення Flutter проекту та його налаштування.

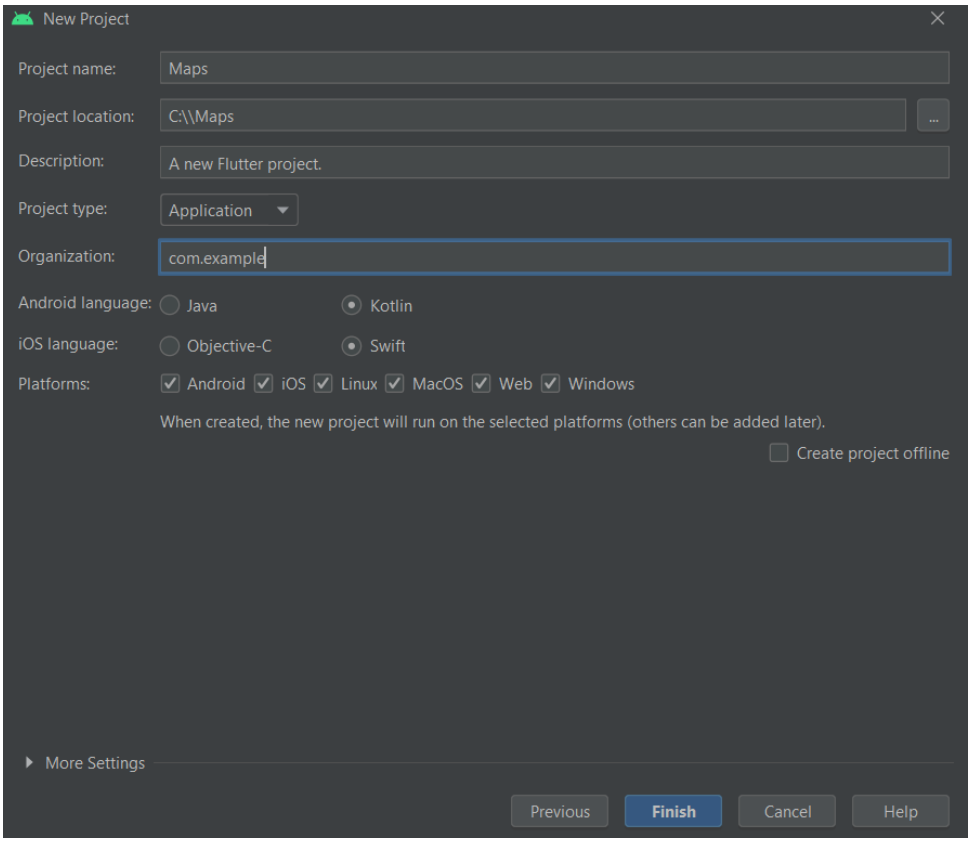

Рисунок 3.1 Налаштування проекту

Розглянемо налаштування проекту (рисунок 3.1):

- Project name – в цьому полі ми даємо назву нашому проекту;
- Project location – в цьому полі ми місце на диску де саме буде знаходитись проект;
- Description – в цьому полі ми можемо дати мінімальний опис проекту який буде відображатись на стартовому Widget;
- Project type – вибираємо тип проекту з наданого списку ;
- Organization – в цьому полі ми можемо вказати сайт організації розробника ;
- Android language – вибираємо мову програмування для Android версії додатку;
- IOS language – вибираємо мову програмування для IOS версії додатку;
- Platforms – Вибираємо платформи які будуть підтримувати наш додаток.

```
    <meta-data android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyAGezewoxTR4OjHXkUzDg-aFCHleAWWRt8"/>
```

Рисунок 3.2 Підключення API ключа

```
flutter:
  sdk: flutter

google_maps_flutter: ^2.2.1
geolocator: ^9.0.2
# The following adds the Cupertino Icons font to your application.
# Use with the CupertinoIcons class for iOS style icons.
cupertino_icons: ^1.0.2
clippy_flutter: ^1.1.1
custom_info_window: ^1.0.1
firebase_auth: ^4.1.5
cloud_firestore: ^4.1.0
provider: ^6.0.4
fluttertoast: ^8.1.1
firebase_core: ^2.3.0
google_map_polyline_new: ^0.3.0+2
flutter_google_places: ^0.3.0
google_maps_webservice: ^0.0.19
dev_dependencies:
flutter_test:
  sdk: flutter
```

Рисунок 3.3 Підключення бібліотек Google_maps

Для розробки додатку з картами необхідно підключити Google API Key (рисунок 3.2) та набір бібліотек які відповідають за створення карт та взаємодії з ними (рисунок 3.3)

3.2.Алгоритмічне відтворення Flutter проекту

Бібліотеки надають доступ до всіх функцій які вони містять в собі, це спрощує написання коду, Імпортування основних бібліотек

```
import 'dart:async';
import 'dart:convert';
import 'package:flutter/services.dart' show rootBundle;
import 'package:geolocator/geolocator.dart';
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
```

Рисунок 3.4

Наступним кроком є створення основної функції запуску проекту та виклик в ній всіх інших компонентів.

```
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Google Maps for Dartup',
      theme: ThemeData(
        primaryColor: Colors.grey
      ),  // ThemeData
      home: MapSample(),


    );  // MaterialApp
  }
```

Рисунок 3.5 Створення основного Widget.

Алгоритм сповіщення про тривогу та пошуку найближчого укриття (рис 3.6). На блок схемі зображено шлях роботи додатка від початку до кінця.

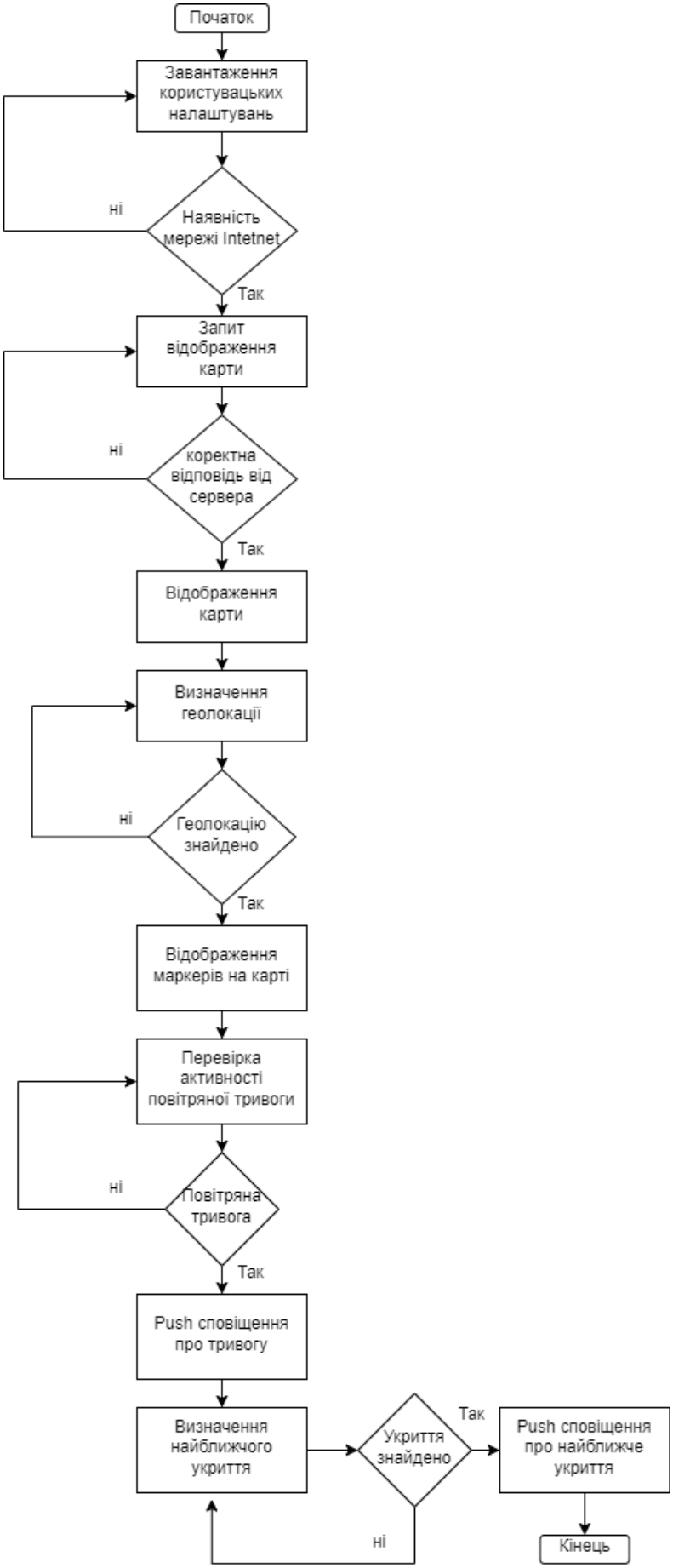

Рисунок 3.6 блок схема роботи додатку

3.3. Створення Widget мапи та маркерів

Організація методу побудови мапи та маркерів на ній. Спрочатку потрібно створити Widget який буде відповідати за створення мапи, за це відповідає спеціальний GoogleMap Widget

```
@override
Widget build(BuildContext context) {
  return Scaffold(
      body: GoogleMap(
        myLocationEnabled: true,


        markers: Set.from(_markers),

        initialCameraPosition: _Kyiv,
        onMapCreated: (GoogleMapController controller) {
          _controller.complete(controller);
        },

      ),  // GoogleMap
```

Рисунок 3.7 Створення мапи

Після цього потрібно створити функцію яка дістане всі наші маркери з json файлу та запише їх до масиву, також в цій функції ми ми можемо задати титульний опис маркеру та його вигляд «Icon».

Також в цьому випадку ми використовуємо бібліотеку для роботи з json файлами «dart.convert» , а саме нам потрібна функція «json.decode», яка розділить нам рядок з файлу json на об’єкти

```
Future loadMarkers() async {

  var jsonData = await rootBundle.loadString('assets/sss.json');

  var data = json.decode(jsonData);

  data["features"].forEach((item) {
    _markers.add(new Marker(
        markerId: MarkerId(item["properties"]["id"]),
        position: LatLng(
            double.parse(item["properties"]["latitude"]), double.parse(item["properties"]["longitude"])

        infoWindow: InfoWindow(
            title: item["properties"]["name"],
            snippet: item["properties"]["region"],

        ),  // InfoWindow
        //icon: BitmapDescriptor.defaultMarkerWithHue(
        //   BitmapDescriptor.hueGreen)));
        icon: BitmapDescriptor.defaultMarker));  // Marker
  });
}
```

3.8 Функція створення масиву маркерів


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "id": "1",
        "name": "м. Київ, вул. Антоновича, 10-БNBSP",
        "region": "Голосіївський",
        "latitude": "50.4384874",
        "longitude": "30.5120886",
        "type_shelter": "Подвійного призначення",
        "type_build": "Житловий будинок",
        "owner": "Власн. Разуменко Т.Є.",
        "ownership": "Приватна",
        "special_opportunities": "Відсутній"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          30.5120886,
          50.4384874,
          0
        ]
```


3.9 Вигляд маркера в json файлі

3.4. Взаємодія користувача з додатком

Базова інструкція користувача мобільного додатка:

- Призначення - мобільний додаток призначений для інформування цивільного населення та пошуку найближчого укриття, також є функціонал інформування про дії під час повітряної тривоги та потенційно небезпечних обєктів;
- Вимоги до мобільного телефону – мобільний додаток розрахований на роботу у операційній системі Android версією 5+, та не висуває надмірних вомог;
- Програма поставляєть у вигляді стандартного інсталяційного пакету для операційної системи Android.

Додаток створений для інформування населення під час надзвичайної ситуації. Розглянемо початкову сторінку додатку, на якій є можливість увійти або отримати доступ після реєстрації.

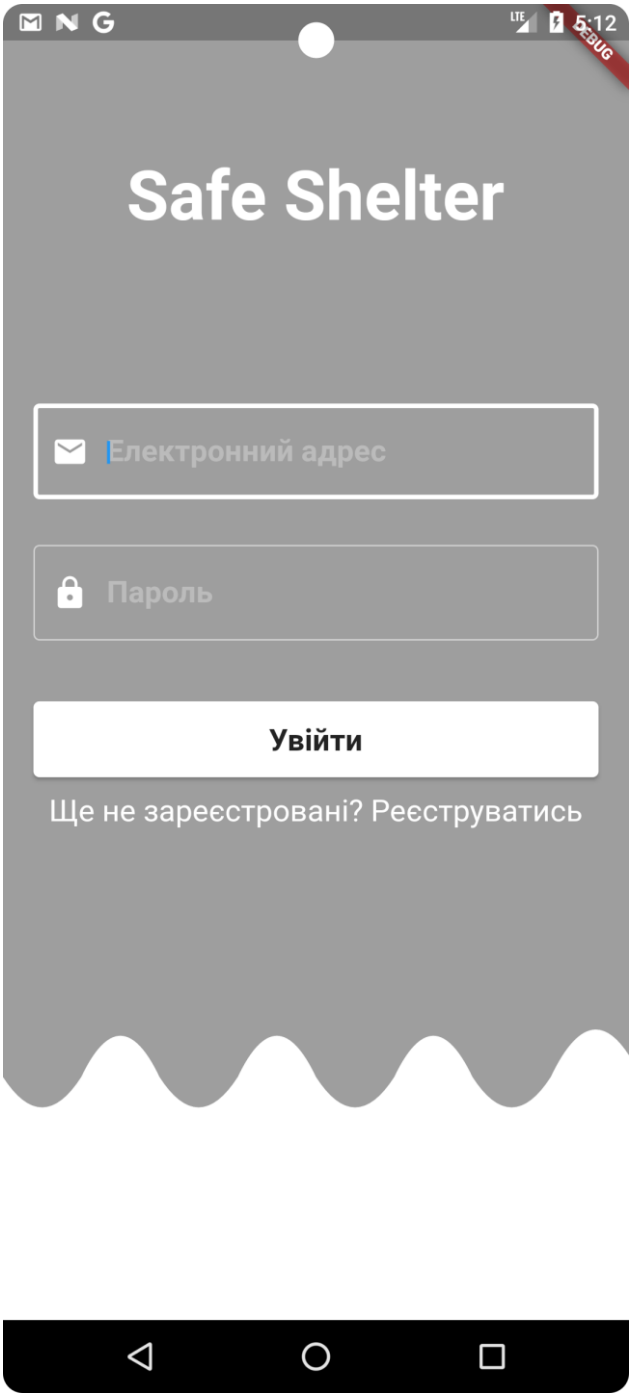

Рисунок 3.10 Сторінка авторизації та реєстрації

На рисунку зображенно дві форми для вводу електронного адресу та паралю щоб пройти аунтифікацію, форма вводу паролю має функціонал скривання тексту (рисунок 3.12). Також реалізована кнопка входу, яка порівнює введені дані з форм вводу з даними користувачів в базі даних «firebase». Якщо користувач не має аккаунта то він може змінити вікно авторизації на реєстрацію натиснувши на текст «Ще не зареєстровані? Реєструватись» (рисунок 3.11)

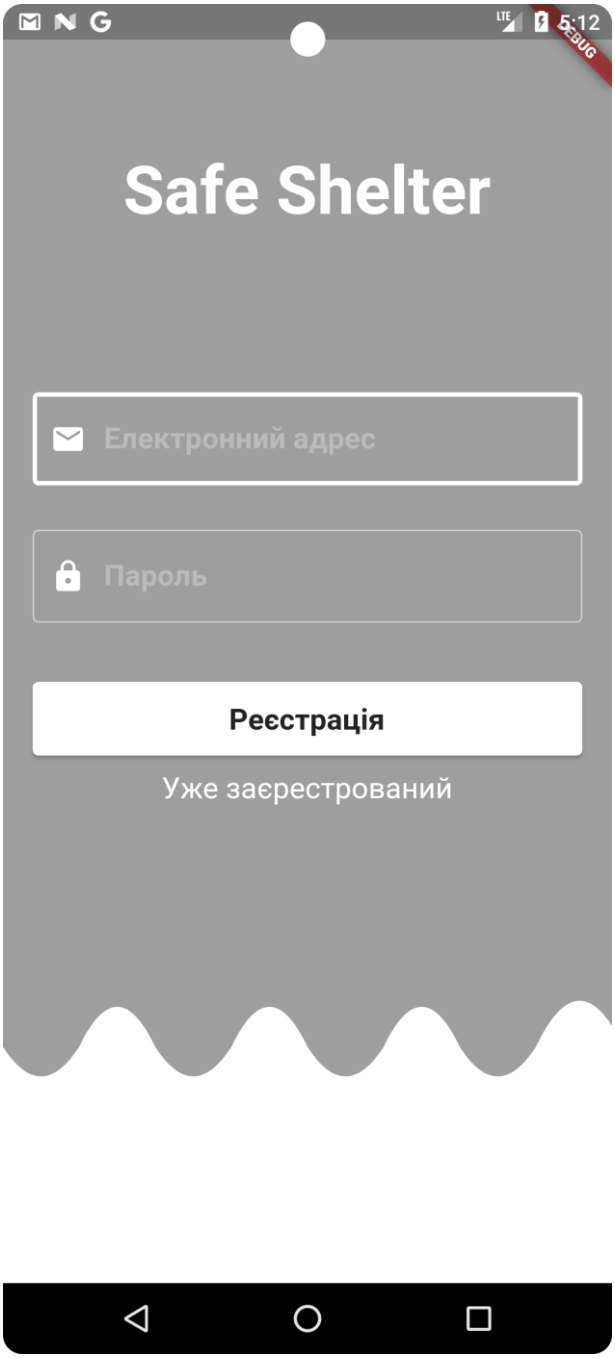

Рисунок 3.11 Сторінка реєстрації

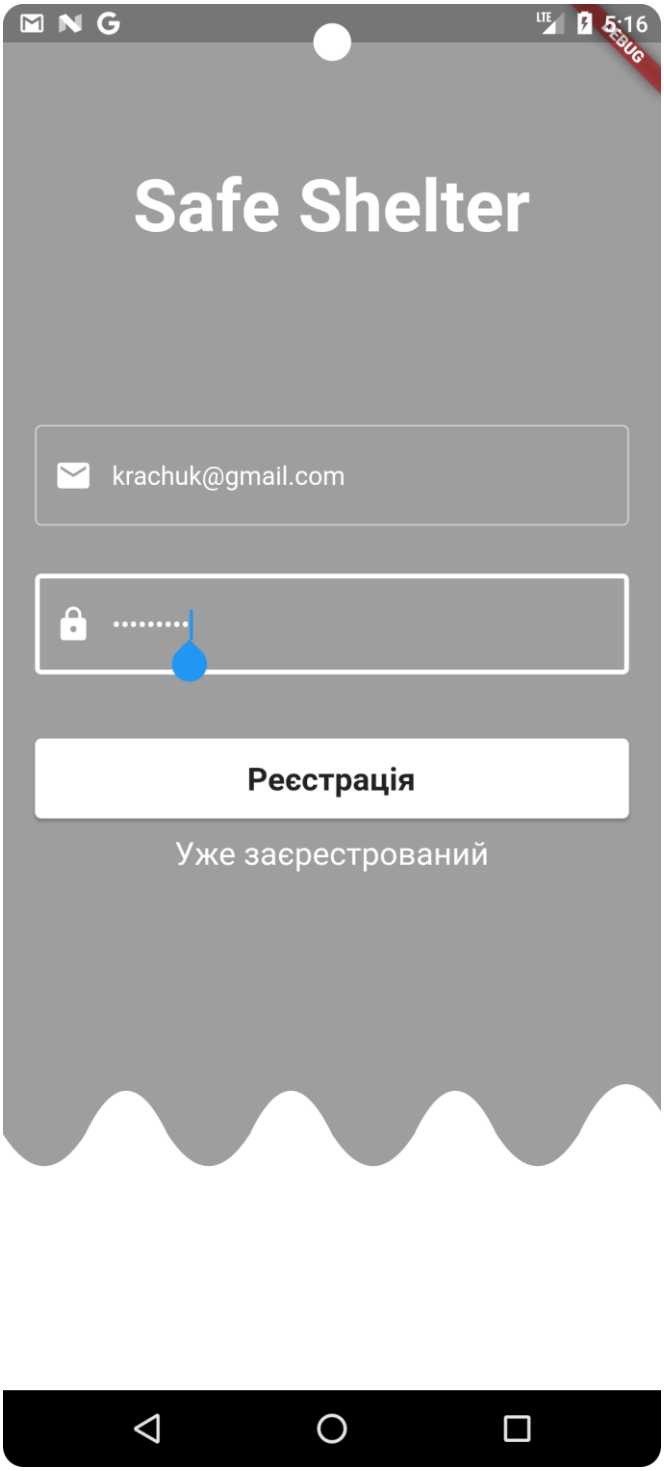

Рисунок 3.12 Демонстрація вводу даних

Після аунтифікації користувач потрапляє на домашню сторінку, на якій він може обрати потрібну йому інформацію (рисунок 3.13).

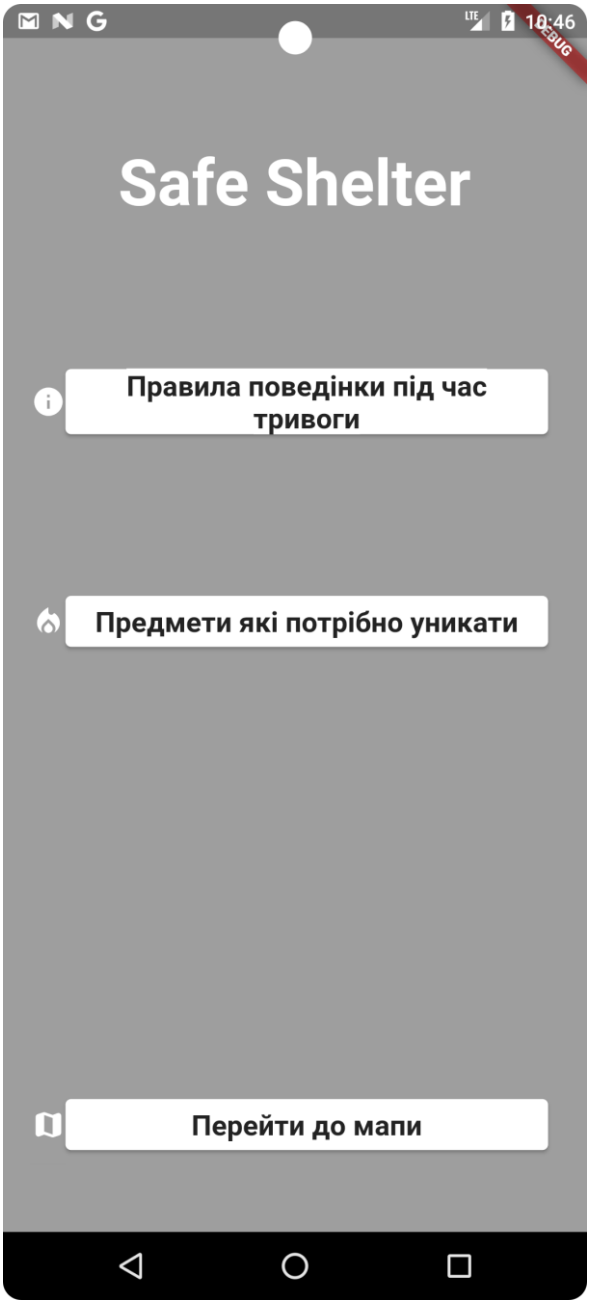

Рисунок 3.13 Домашня сторінка

Тут користувач має можливість обрати потрібну йому інформацію та перейти до ознайомлення з нею. Розглянемо кнопку «Предмети які потрібно уникати», тут ми може побачити список підозрілих та небезпечних предметів, з якими ніякому разі непотрібно взаємодіяти (рисунок 3.14). Також ми можемо переглянути правила та поради щодо дій під час повітряної тривоги натиснувши на кнопку «Правила поведінки під час тривоги».

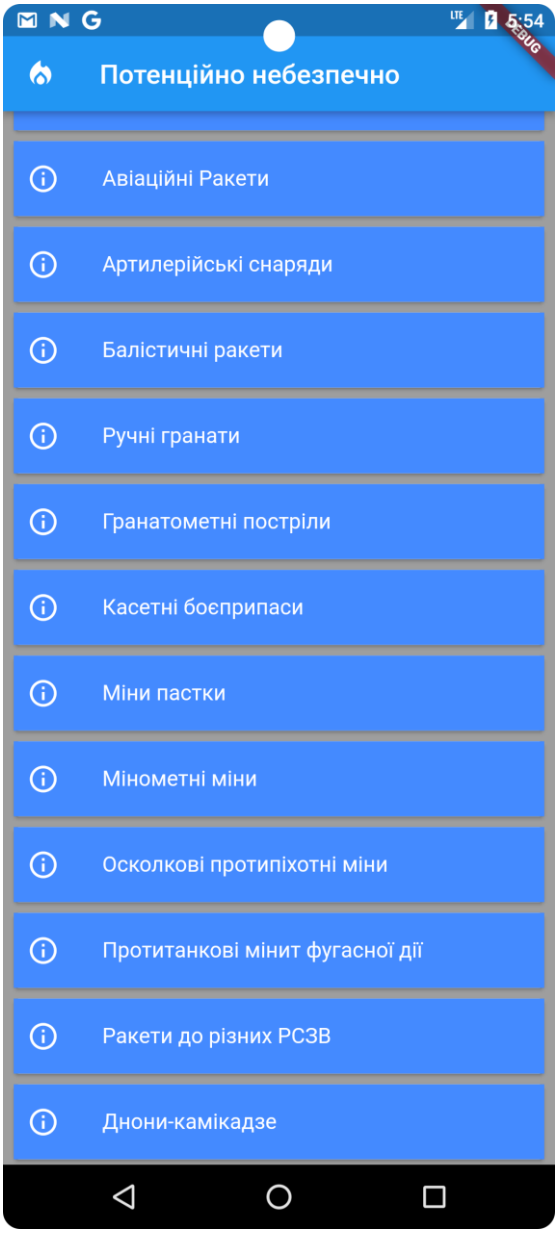

Рисунок 3.14 Потенційно небезпечні предмети

Натиснувши на кнопку «перейти до мапи», у користувача відкриється карта з укриттями міста Київ (рисунок 3.15), на якій він може по взаємодіяти з мараркерами переглядаючи інформацію до них та прокладаючи маршрут, також є функціонал визначення геолокації та найближчого укриття, натиснувши на будь який маркер, користувачу з’являється кнопка відкриття маркеру в додатку google maps та прокладання маршруту між наявною геолокацією користувача та маркером укриття (рисунок 3.18).

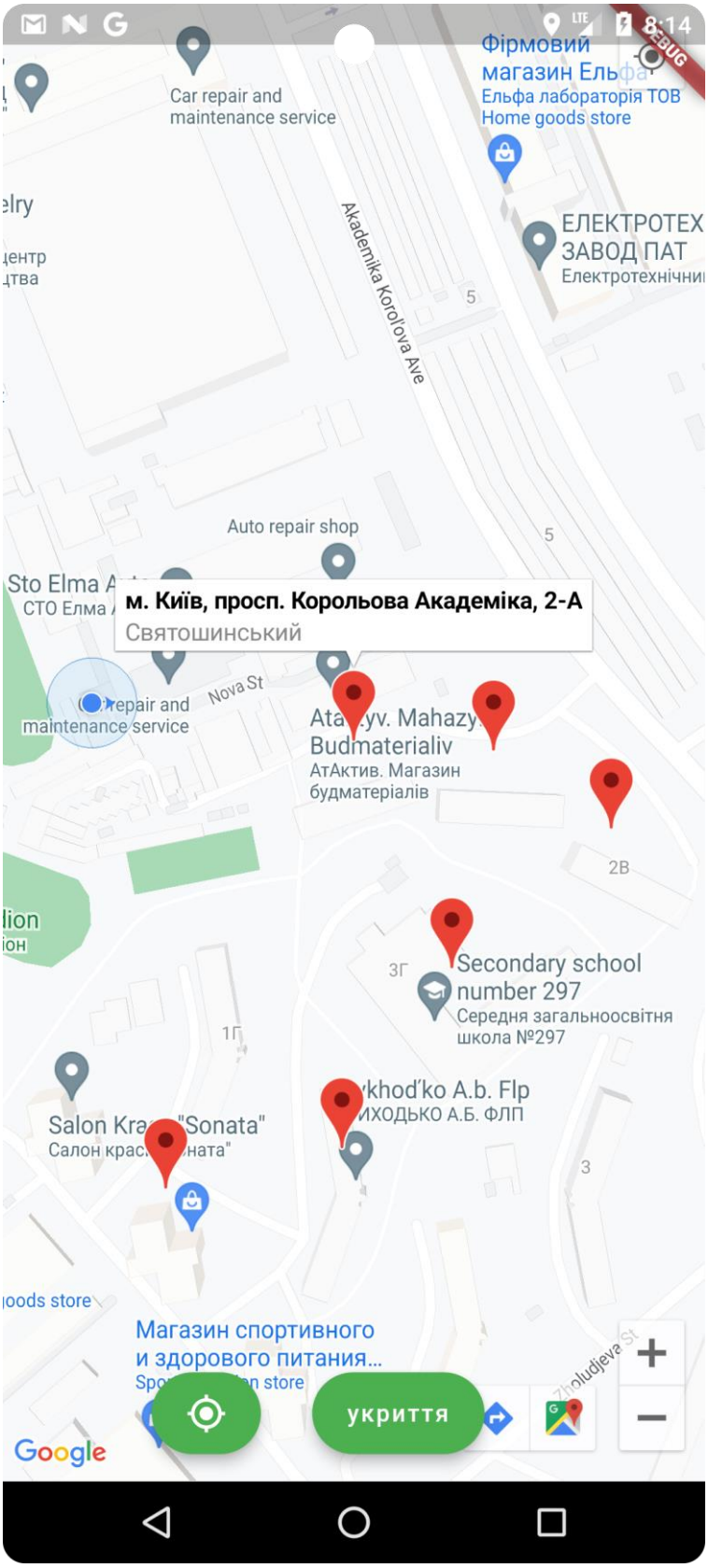

Рисунок 3.15 Карта укриттъ міста Київ

Користувач може знайти найближче укриття натиснувши на виділену червоним кольором кнопку «укриття» (рис. 3.16), після чого розташування камери на карті автоматично перейде до найближчого укриття від поточної геолокації.

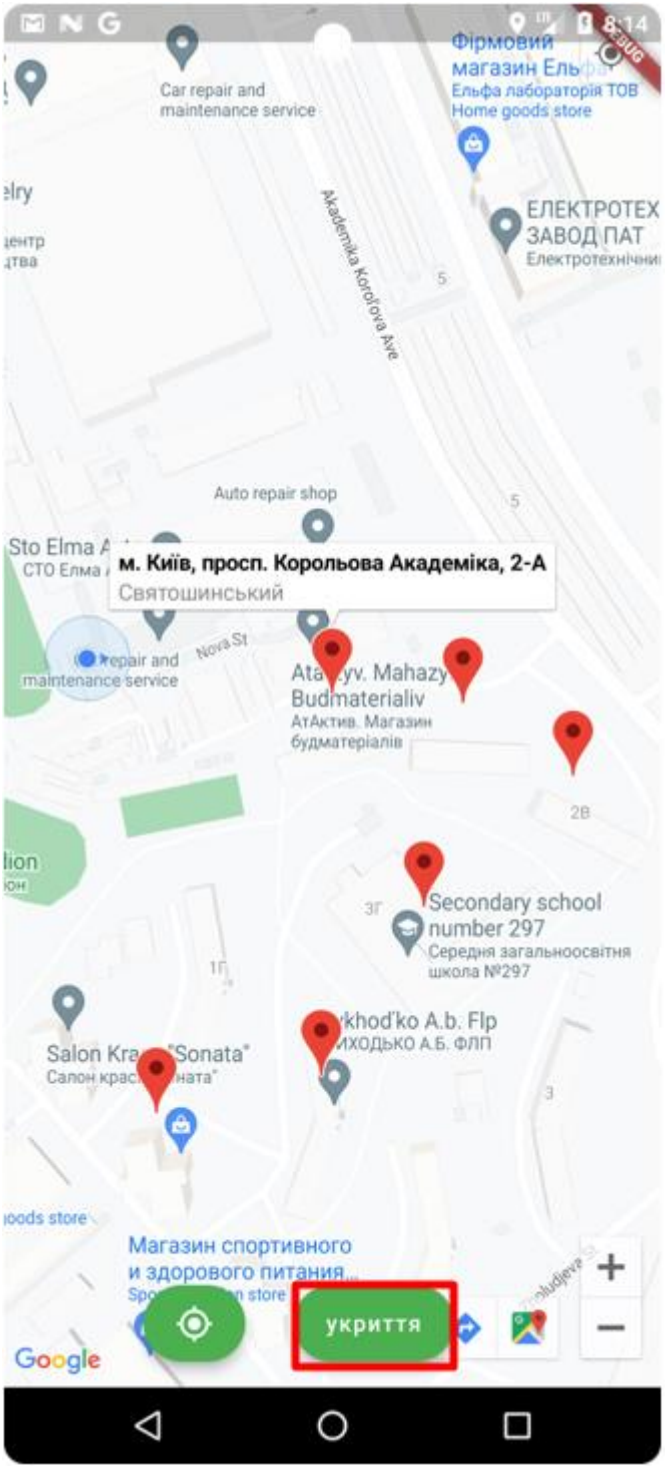

Рисунок 3.16 Кнопка пошуку найближчого укриття

Також користувач додатку може побудувати маршрут до найближчого укриття натиснувши на іконку маршруту (рис 3.17), йому автоматично відкриється додаток google maps і в ньому буде уже побудований піший маршрут.

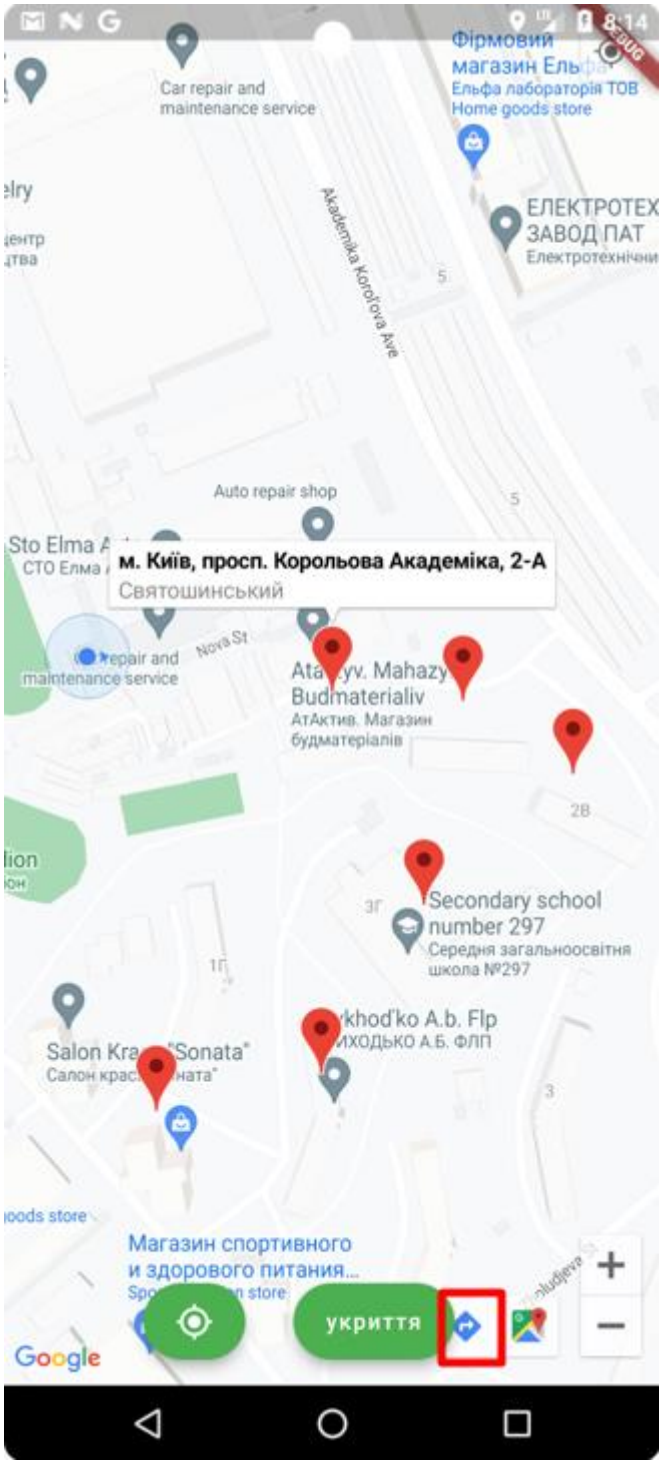

Рисунок 3.17 Кнопка побудови маршруту до укриття

Натиснувши на виділену кнопку (рис. 3.18) користувач визначає свою поточно геолокацію, після цього камера на карті автоматично переноситься до цієї точки.

Користувач обов’язково повинен мати включену функцію GPS та надати дозвіл додатку до визначення геолокації.

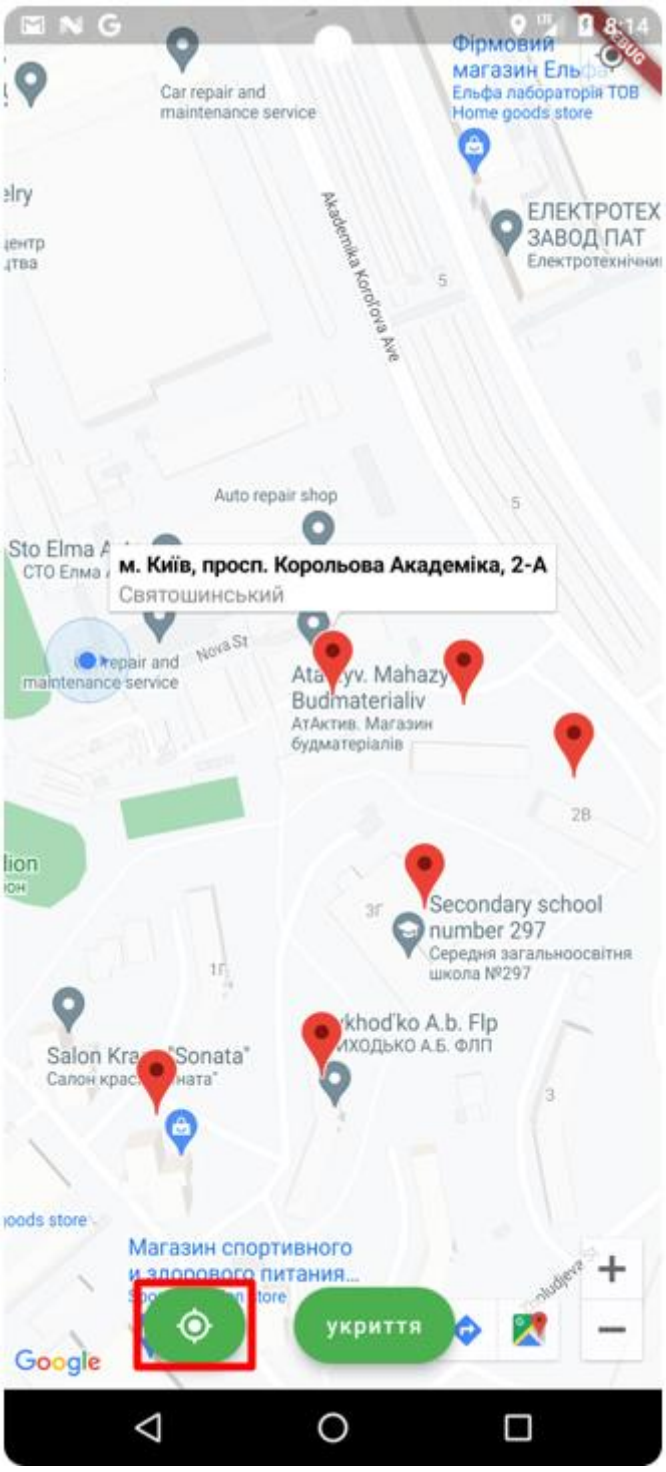

Рисунок 3.18 Кнопка визначення поточної геолокації

Якщо є сповіщення про повітряну тривогу, додаток надсилає користувачеві Push сповіщення про неї (рис. 3.19), після цього знаходиться найближче укриття та надсилається Push сповіщення про нього користувачеві, після натискання на яке у користувача відкривається маршрут до нього (рис. 3.20).

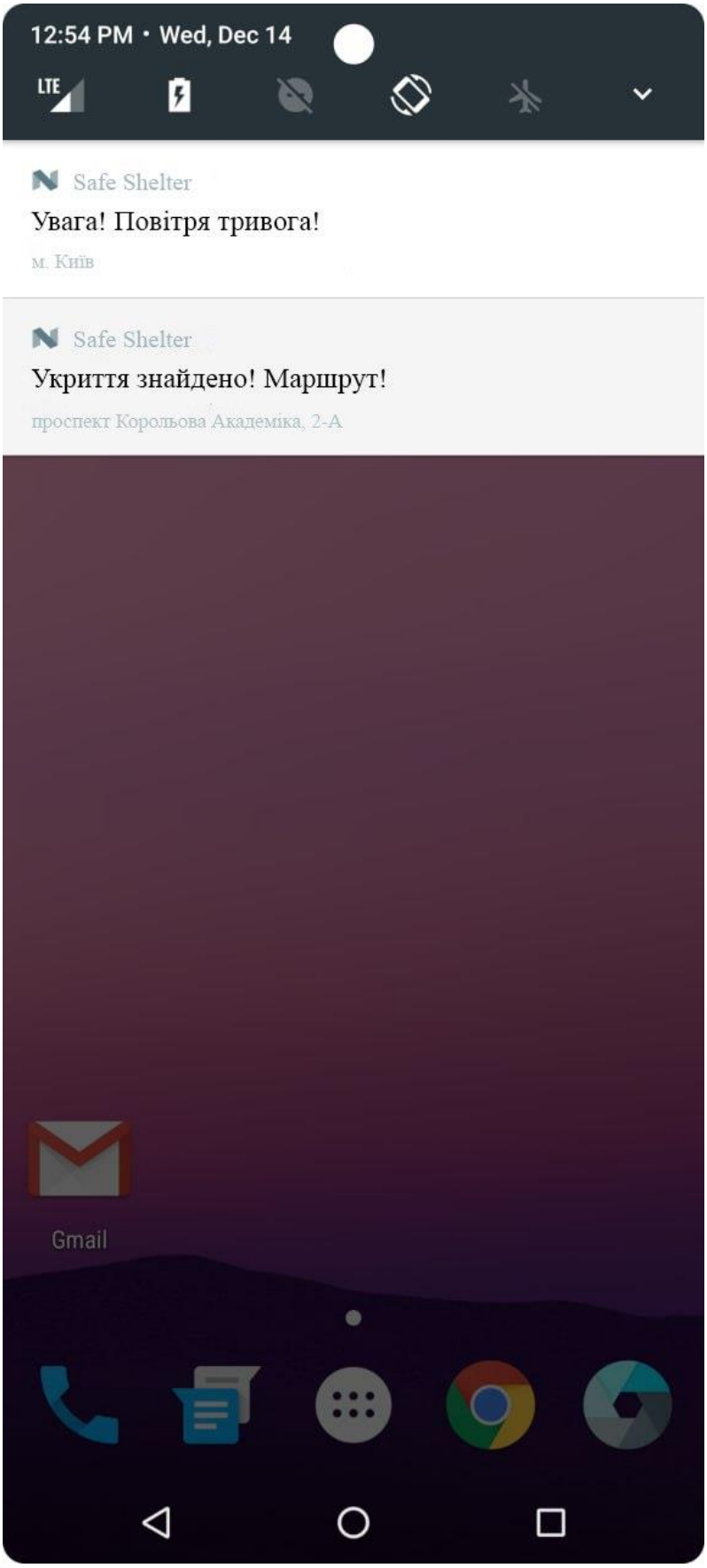

Рисунок 3.19 Приклад роботи push сповіщень

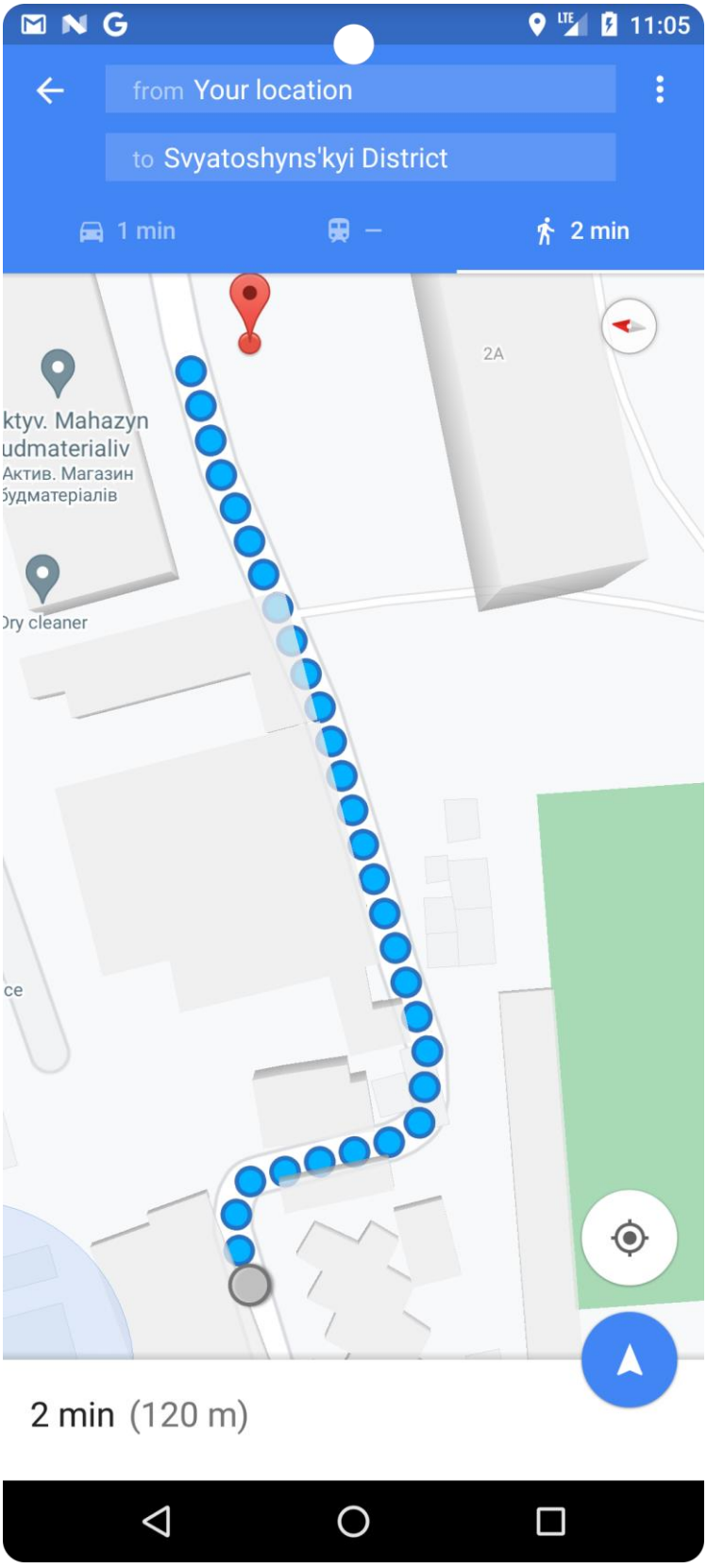

Рисунок 3.20 Прокладений маршрут до укриття

3.5. Результати тестування

Таблиця 3.1 тестування мобільного додатка

	Середовище тестування: Смартфон Xiaomi Mi 9T				
	Опис	Кроки тестування	Очікуваний результат	Фактичний результат	Результат тест-кейсу
1	Завантаження та встановлення	1.Завантаження додатку 2. Встановлення	Після завантаження додатку запускаємо APK файл та встановлюємо додаток	Відбувається встановлення мобільного додатка на смартфон	Працює
2	Авторизація або реєстрації	1. Ввод персональних данних від свого аккаунта 2. Реєстрація аккаунта	Користувач може увійти до додатка для отримання повного функціоналу або створити аккаунт	Вхід до системи або створення аккаунта	Працює
3	Маршрутизація в мобільному додатку за допомогою кнопоку	Перехід до нової сторінки за допомогою відповідної кнопки	Відкриття відповідної сторінки	Відкриття відповідної сторінки	Працює

	Відкриття мапи в мобільному додатку	Після натискання відповідної кнопки, мобільний додаток повинен відкривати сторінку з мапою	Відкриття сторінки з мапою	Відкриття сторінки з мапою	Працює
4	Відображення маркерів сховищ на мапі	Після відкриття сторінки з мапою на ній мають відображатися маркери з сховищами	Маркери відображено	Маркери відображено	Працює
5	Push сповіщення про тривогу	Після повітряної тривоги в Україні користувачеві приходить push сповіщення які сповіщають про тривогу	Push сповіщення надходить користувачеві	Push сповіщення надходить користувачеві	Працює
6	Після push сповіщення запускається алгоритм пошуку	Після сповіщення про повітряну тривогу відбувається	Найближче укриття знайдене	Найближче укриття знайдене	Працює

	найближчого укриття	початок пошуку найближчого укриття			
7	Після находження найближчого укриття користувач отримує push сповіщення	Користувач повинен отпримати push сповіщення з маршрутом до найближчого укриття	Користувач отримав push сповіщення	Push сповіщення було отримано	Працює
8	Відкриття маршруту до укриття	Після натискання на сповіщення у користувача має відкритись побудований маршрут до укриття	Користувач перейшов до google maps з побудованим маршрутом	Перехід до google maps з побудованим маршрутом	Працює

Висновок до розділу 3

В результаті розробки мобільного застосунка, було створено додаток для інформування цивільного населення та пошуку найближчого укриття. Було реалізовано інтуїтивно зрозумілий та простий в використанні інтерфейс, для відображення мапи використовувалось API google maps. Застосунок створений на базі мови програмування dart та бібліотеки flutter, розглянуто основний функціонал додатку та показано процес створення проекту.

Розглянуто середовище розробки, а саме Android studio та наведено приклади створення проекту та підключення API. Показано можливості сповіщення та інформування та автоматичної побудови маршруту.

РОЗДІЛ 4 РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

Автоматизований мобільний додаток для інформування цивільного населення про надзвичайну ситуацію

4.1.Опис ідеї проекту

Під час військового стану в Україні постає питання інформування населення про надзвичайні ситуації військового характеру, а саме «повітряна тривога». У теперішній час завдання розробки засобів інформування населення постає досить часто. Область застосування засобів інформування досить очевидна, це першочергова дія обласних адміністрацій для безпеки громадян. Різні засоби інформування можуть бути корисними не тільки під час війни, а й після її закінчення, тому що надзвичайних ситуацій різного характеру досить багато, і таким чином ми будемо готові до них заздалегідь. Мобільні додатки як спосіб інформування населення досить актуальні на даний момент, так як практично у кожного громадянина України є смартфон або планшет.

Цей додаток дозволить населенню не тільки отримувати сповіщення про повітряну тривогу, що продемонстровано уже існуючими аналогами, а й сповіщати про місцезнаходження найближчого укриття та будувати маршрут до нього в google maps. Крім цього в додатку реалізовано функціонал інформування населення щодо небезпечних об'єктів та запропоновані поради щодо дій під час повітряної тривоги.

В табл.4.1 описано інформаційну карту проекту.

Таблиця 4.1 – Інформаційна карта проекту

1. Назва проекту	Автоматизована система інформування цивільного населення під час надзвичайних ситуацій
2. Автор проекту	Кравчук О.П.

3. Коротка анотація	Кожного дня в світі відбувається багато надзвичайних ситуацій різного характеру, кожна держава світу має дбати про своє цивільне населення та сповіщати їх про можливу загрозу заздалегідь, це може врятувати безліч життів. Як ми знаємо в 21 столітті важко уявити людей які не користуються різними гаджетами, від мобільного телефону до комп'ютера. В більшості додатків які інформують про повітряну тривогу невирішена задача пошуку найближчого укриття, це допоможе цивільному населенні не втрачати час на пошук інформації.
4. Термін реалізації проекту	12 місяців
5. Необхідні ресурси	Матеріальне забезпечення: • Ноутбук Acer aspire A7 – 30000 грн Програмне забезпечення: • Середовище розробки Android Studio • Firebase Фінансові витрати: • Оренда офісу (6000 грн в місяць) • Оплата інтернет провайдера (180 грн в місяць) • Заробітна плата (36000 грн в місяць)
6. Опис проблеми яку вирішує проект	Даний проект вирішує проблему автоматичного пошуку укриття після сповіщення про повітряну тривогу.

7. Головні цілі та завдання проекту	Ціль: розробити мобільний додаток який буде конкурентоспроможним по відношенню до його аналогів, спростити користувачеві пошук інформації про найближче укриття та сповіщати його про повітряну тривогу
8. Очікувані результати	Очікуваний результат полягає в отриманні автоматизованої системи інформування населення під час надзвичайних ситуацій, Система повинна містити в собі рекомендації щодо базових дій під час повітряної тривоги, сповіщення про повітряну тривогу та автоматичний пошук найближчого укриття та будування маршруту до нього. Система повинна підтримуватися найпопулярнішими платформами, такими як Android, IOS та Web.

Аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів наведено в (табл. 4.2).

Таблиця 4.2 Визначення сильних, слабких та нейтральних характеристик

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтраль на сторона)	N (нейтраль на сторона)
		Мій проект	Київ цифровий	Повітряна тривога	Де Укриття			
1	Корисність функціоналу	91%	65%	84%	40%			+
2	Точність данних	висока	висока	висока	середня			+
3	Вартість додатку	безкоштовний	безкоштовний	безкоштовний	безкоштовний			+

4	Зручність	84%	80%	98%	42%		+	
5	Швидкість інформування	висока	середня	висока	низька			+

Висновки: як видно з таблиці головною перевагою проекту є корисний функціонал який дозволяє користувачеві виконувати пошук різної інформації, в свою чергу користувач витрачає менше часу на пошук інформації.

4.2.Технологічний аудит ідеї проекту

У табл. 4.3 проводить аудит технології, за допомогою якої можна реалізувати ідею проекту.

Таблиця 4.3 Технологічна здійсненність ідеї проекту

1	2	3	4	5
№ п/п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Сповіщення про тривогу	Використання API	Наявна	Доступна
2	Інформування користувачів	Використання мобільного додатку	Наявна	Доступна
3	Пошук найближчого укриття	Впровадження алгоритму пошуку	Наявна	Доступна
4	Створення маршруту до укриття	Використання google maps	Наявна	Доступна
Обрана технологія реалізації проекту: створення автоматизованої системи інформування цивільного населення під час надзвичайної ситуації				

Висновки: Проаналізувавши таблицю, можна дійти висновку, що даний проект може бути використано звичайними користувача для забезпечення власної безпеки та пошуку корисної інформаці.

4.3.Аналіз ринкових можливостей запуску стартап проекту

Для отримання чіткої ситуації на ринку, визначимо потенційні групи клієнтів (табл.4.4), їх характеристики, сформуємо орієнтовний перелік вимог до товару для кожної групи.

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Небезпека під час надзвичайної ситуації військового характеру	Населення віком від 18-25	Особливості, пов’язані з навичками користування смартфоном або планшетом, навички пошуку інформації	Швидкість роботи мобільного додатку, простота інтерфейсу, корисність функціоналу, інформування з перевірених джерел
2		Населення віком від 26-39		
3		Населення віком від 40+		

Висновки: проаналізувавши ринок виділимо три головні потреби для користувачів: Простота використання, швидкість роботи додатка, правдивість інформації.

Проведемо аналіз факторів ринкового середовища що сприяють ринковому впровадженню проекту (табл. 4.5) та факторів, що йому перешкоджають (табл. 4.6).

Таблиця 4.5 Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Збільшення попиту	Різке збільшення попиту на мобільний додаток	Збільшення функціоналу додатку
2	Необхідність до інтегрування	Відкриття свого API	Надати API в відкритий доступ
3	Державне замовлення	Клієнт потребує інтегрувати систему до їх додатку	Оцінка можливих рішень та їх реалцізація
4	Закінчення війни	Різкий спад попиду на мобільний додаток	Зміна основного напряму додатку на інший
5	Продаж додатку	Втрата всіх користувачів	Розробка нового проекту

Висновки: Такі системи зазвичай доступні на самих популярних платформах, таких як: IOS, Android та Web.

Таблиця 4.6 Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Роботоздатність	Збої у системі інформування	Відмова від додатка

2	Некомпетентність працівників	Баги та поломки в программному продукті	Звільнення працівника та пошук заміни
3	Конкуренція	На ринку зявився більший гравець	Бажання клієнтів використовувати кращий продукт
4	Втрата актуальності	Великий спад користувачів додатка	Відмова від додатка

Висновки: Головними факторами загрози є конкуренція та втрата актуальності. Користувачі зазвичай обирають додатки з більшим рейтенгом та перевірені часом.

Проведемо аналіз пропозиції (табл. 4.7) та визначимо загальні риси конкуренції на ринку.

Таблиця 4.7 Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції - чиста конкуренція	Присутні конкоренти які мають схожий додаток	Розширення функціоналу додатка
За рівнем конкуретної боротьби - міждународний	Відсутні конкуренти які мають аналогічний додаток	Поява нових гравців
За галузевою ознакою - міжгалузева	Використання у різних галузях	Робота менеджменту і реклами по залученню клієнтів

За інтенсивністю - марочна	Вибір серверів на яких буде знаходитися застосунок	Розкрутка бренду, його рекламування

Висновки: на ринку присутня чиста конкуренція через те, що окремі гравці мають свої аналоги, за рівнем конкуретної боротьби – міжнародна, так як можливі іноземні замовники.

Визначимо та обгрунтуємо перелік факторів конкурентоспроможності (Аналіз наведено в табл. 4.8).

Таблиця 4.8 Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	і Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Надійність	В додатку використовується інформація тільки з перевірених джерел
2	Простота	Додаток має простий та зрозумілий інтерфейс та легкий в використанні
3	Адаптивність	Додаток можна легко адаптувати під інші потреби якщо такі виникають
4	Готовий алгоритм	Автоматичний пошук найближчого укриття
5	Обслуговування	Команда проекту реагує на будь які повідомлення

Висновки: для підвищення надійності потрібно використовувати інформацію тільки з перевірених джерел. Простота у використанні додатка робить його більш привабливим для користувачів. Адаптивність дозволить проекту пережити спад актуальності додатку без втрати користувачів.

За визначеними факторами конкурентоспроможності (табл. 4.8) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 4.9).

Таблиця 4.9 Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг додатків-конкурентів						
			-3	-2	-1	0	+1	+2	+3
1	Надійність	18							+3
2	Простота	20						+2	
3	Адаптивність	17							+3
4	Готовий алгоритм	17				0			
5	Обслуговування	16							+3

Висновки: спираючись на фактори конкурентоспроможності (Табл. 4.9) та підсумовуючи рейтинг додатка відносно головного конкурента, запропонована система має більший рейтинг відносно прямих конкурентів. Дана таблиця показує якими саме особливостями розроблена система відрізняються від аналогів та в яку саме сторону. Детальний аналіз показує, що сильними сторонами є простота та надійність системи.

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities), що наведено в (табл. 4.10) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін в (табл. 4.9).

Таблиця 4.10 SWOT- аналіз стартап-проекту

Сильні сторони: 1. простота використання; 2. адаптивність; 3. стабільна робота; 4. універсальність; 5. перевіреність інформації;	Слабкі сторони: 1. низька репутація додатка після його впровадження на ринок; 2. додаток не пов'язаний з державними адміністраціями
Можливості:	Загрози:

1. вихід на міжнародний ринок; 2. збільшення попиту; 3. необхідність до інтеграції; 4. освоєння нових сфер; 5. індивідуальне замовлення; 6. співпраця з конкурентами	1. поява додатків аналогів; 2. втрата актуальності;

На основі SWOT-аналізу розробимо альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (табл. 4.11).

Таблиця 4.11 Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	індивідуалізм(максимізація власного виграшу)	середня	2 міс.
2	кооперація (максимізація спільного виграшу)	висока	3 міс.
3	суперництво	нижче середнього	5 міс.

Висновки: в результаті аналізу обрано кооперацію, як альтернативну ринкову поведінку через те, що за відносно короткий термін існує велика ймовірність отримання ресурсів.

4.4.Розроблення ринкової стратегії проекту

Розробка ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 4.12).

Таблиця 4.12 Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Цивільне населення	+	+	Середня	+
2	Державні установи	+	+	низька	-
Які цільові групи обрано: під час вибору цільової групи в першу чергу бралося до уваги готовність користувача сприйняти продукт. У випадку успішної реклами є шанс зайняти нішу.					

Висновки: За результатами оцінки були обрані групи, які потребують швидкого реагування та відповідно швидкого оцінювання надзвичайної ситуації для подальшого впровадження тих чи інших дії. З (табл. 4.12) можна побачити що обрані клієнти потребують дані рішення.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (табл. 4.13).

Таблиця 4.13 Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Індивідуалізм	Стратегія диференційованого маркетингу	Адаптація до вимог ринку Використання новацій Генерування ноу-хау	стратегія диференціації

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 4.14).

Таблиця 4.14 Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Не є першопрохідцем	Шукати нових споживачів із поступовим переманюванням від конкурентів	Компанія буде вдосконалювати наявний застосунок	Стратегія лідера

Висновок: оскільки проект не є першопрохідцем, але має суттєві переваги по відношенню до свого прямого конкурента, можливо обрати стратегію лідера. Це є можливим на фоні впровадження інновацій і технологій для вдосконалення продукту з метою ускладнення задачі конкурентів.

На основі вимог споживачів з обраних сегментів до постачальника та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробимо стратегію позиціонування (табл. 4.15).

Таблиця 4.15 Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Простота в користуванні	Стратегія спеціалізації	Розвиток та адаптація	Легкість
2	Можливість відкритого API	Стратегія спеціалізації	Обслуговування	Здатність мати відкритий API
3	Надійність інформації	Стратегія спеціалізації	Якість	Надійність

Висновки: Обрана стратегія є ключовою для розвитку цього проекту в майбутньому та отримання прибутку для реінвестування у проект та купівлю нових технологій з подальшою інтеграцією у проект.

4.5.Розроблення маркетингової програми стартап-проекту

Маркетингова концепція товару є першим кроком до формування ідеї яку отримує споживач, наведено у (табл. 4.16).

Таблиця 4.16 Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Необхідність швидкого пошуку укриття	Безкоштовна та надійна система	Надійність, простота, підтримка, достовірність
2	Необхідність інформування під час надзвичайної ситуації	Повний функціонал з використанням API	Ціна, достовірність, обслуговування, адаптивність

Висновки: в результаті оцінки маркетингової концепції є можливість створити дві цільової аудиторії для подальшого створення рекламних кампаній.

Розробимо трирівневу маркетингову модель товару (табл. 4.17).

Таблиця 4.17 Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Сповіщення про надзвичайну ситуацію та максимально швидкий пошук та відображення найближчого укриття та текстове інформування населення.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>Доступність
Швидкість роботи
Достовірність інформації
Вартість</td><td>Онлайн
1-3 с
99%
16</td><td>Тх
Тх
Тл
Е</td></tr>
<tr><td colspan="3">Якість: -</td></tr>
<tr><td colspan="3">Пакування: -</td></tr>
<tr><td colspan="3">Марка: -</td></tr>
<tr><td></td><td colspan="3">Додавання нового функціоналу для користувача</td></tr>
</table>

III. Товар із підкріпленням	Після оплати отримується доступ до візуальних та звукових налаштувань
Захист товару проводиться шляхом постійного додавання нового функціоналу та захоплення нових ніш ринку	

Висновки: систематичне оновлення та додавання нового функціоналу стане основним засобом захисту додатку. Також постійний аналіз та розробка новітнього функціоналу та подальша інтеграція його в систему.

Наступним кроком визначимо цінові межі (табл. 4.18).

Таблиця 4.18 Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	1100 у.о.	450 у.о.	30000 у.о.	50-450 у.о.

Висновки: Для даного товару планується зробити систему підписок для більш гнучкої системи для користувача, який готовий платити лише за окремий функіонал який не впливає на основну ціль додатку.

Визначимо оптимальну систему збуту, в межах якого будуть прийматися ті чи інші рішення(табл. 4.19).

Таблиця 4.19 Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Підписка	Оформлення підписки для доступу до додаткового функціоналу	Нульового рівня	Прямий

Висновки: Ефективною вважається система по оформлені підписки, яка допоже отримувати кошти регулярно та більш гнучкі для самого користувача. Через що виникає відносини він-він де кожен обирає той рівень підписки, якій йому потрібен.

Таблиця 4.20 Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Обставини змушують користувача шукати мобільний додаток для інформування	Реклама SMM	Висока швидкість роботи, іноваційний функціонал, достовірність інформації,	Показати іноваційний функціонал та швидкість роботи	Показ зручності та простоти використання

	про повітряну тривогу та пошук найближчого укриття		простота використання		

Висновки: Маркетинг проходить через різні канали розміщення рекламних об’яв, а саме Facebook Ads, Google Ads, TikTok, Instagramm, You Tube. Цільова аудиторія повинна побачити зручність та функціонал додатку менше ніж за 15 секунд.

Висновки до розділу 4

В цьому розділі було описано стратегію створення стартап-проекту. Для подальшого збуту розробленого додатку автоматичного інформування цивільного населення в умовах надзвичайної ситуації, який дозволяє користувачеві отримувати сповіщення про повітряну тривогу, та автоматично находити найближче укриття відносно поточної геолокації. Так як розроблений проект є досить актуальним, він може бути проданим зацікавленим особам.

Було створено інформаційну картку проекту (табл. 4.1) у якій коротко описано потреби та цілі проекту. Також було проведено аналіз слабких та сильних сторін додатка, його порівняння з прямими конкурентами.

Під час технічного аудиту проекту були отримане розуміння кращої технології виконання продукту. Виявлено, що аналогічні технології вже існують, але їхнього функціоналу недостатньо або вони не є інтуїтивно зрозумілими. У додатка є 2 шляхи розвитку, перший це постійне оновлення та розробка нового функціоналу або другий, це співпраця з обласними адміністраціями які будуть надавати актуальну інформацію та генерувати ідеї нового функціоналу.

Зважаючи на війну в Україні додаток інформування потребує розширення бази даних для території всієї країни та розробка нового функціоналу додавання сховищ через інтерактивну карту.

Висновки

1. В результаті роботи було розглянуто основні типи сучасних автоматизованих систем інформування цивільного населення під час надзвичайної ситуації військового характеру, а саме: повітряної тривоги та інформування населення про дії в умовах виникнення повітряної тривоги.
2. Проведено аналіз існуючих програмних рішень для реалізації автоматизованої інформаційно-пошукової системи та визначено переваги та недоліки додатків-аналогів.
3. Обрано сучасну мову програмування dart та фреймворк Флаттер за їх переваги та простоту в розробці мобільних додатків.
4. У ході роботи було визначено оптимальний план розробки мобільного додатка, структуру мобільного додатка, блок-схему обчислення маршруту між двома точками, схему бази даних в Firestore, діаграму компонентів для мобільного додатку, діаграму розгортання для мобільно додатку.
5. Було розроблено програмний продукт в вигляді мобільного застосунку, який має функціонал інформування про небезпечні об'єкти та поради щодо поведінки під час повітряної тривоги. Додаток інформує користувачів про повітряну тривогу та знаходить найближче укриття в виді push сповіщення, та має змогу автоматичної побудови маршруту за допомогою google maps. Продемонстровано приклади роботи додатка та показано його функціонал.
6. Було проведено тестування роботи мобільного додатка для автоматичного інформування та пошуку найближчого укриття.
7. Розроблено детальну інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Communications changing, are you behind the curve? [Електронний ресурс] – Режим доступу до ресурсу: https://blog.911cellular.com/mass-notification-systems.

2. History of Early Warning and Emergency Notification Systems [Електронний ресурс] – Режим доступу до ресурсу: http://www.electronic-sirens.com/history-early-warning-emergency-notification-systems/.

3. Охорона праці та цивільний захист / О. Г.Левченко, О. І. Полукаров, В. В. Зацарний, Ю. О. Полукаров. – Київ, 2019. – 408 с.

4. D. Mileti та J. Sorensen, "Communication of emergency public warnings: A social science perspective and state-of-the-art assessment", Oak Ridge National Lab, с. 11–12, січ. 1990.

5. Денисенко С. Автоматизовані системи раннього виявлення надзвичайних ситуацій та оповіщення / С. Денисенко, І. Гасек, О. Гуменюк. – Київ: МНС України, 2011.

6. О. Євгедін та К. Блажчук, "Автоматизовані системи раннього виявлення загрози виникнення надзвичайних ситуацій та оповіщення населення", Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, с. 19, 2014.

7. R. Ahmad, "Design and Development of Automotive Workshop Application Based on Android and IOS Using Dart Programming Language", 243rd ECS Meeting with the 18th International Symposium on Solid Oxide Fuel Cells (SOFC-XVIII), № 243, 2020.

8. A. Tashildar, N. Shah та R. Gala, "APPLICATION DEVELOPMENT USING FLUTTER", International Research Journal of Modernization in Engineering Technology and Science, с. 1262–1263, 2020.

9. Рябоконь О. С. Новый язык структурного веб-программирования dart / О. С. Рябоконь. // Информационно-экономические системы. – 2013. – №4. – С. 436–437.

10. H. Hussain та K. Khan, "Comparative Study of Android Native and Flutter App Development", KSII The 13 th International Conference on Internet (ICONI) 2021, № 13, с. 99–100, 2021.

11. W. Wu, "React Native vs Flutter, cross-platform mobile application frameworks", Metropolia University of Applied Sciences Bachelor of Engineering Information technology Thesis, c. 7–8, 2018.

12. C. Khawas, "Application of Firebase in Android App Development-A Study", International Journal of Computer Applications, June, c. 49–52, 2018.

13. S. Hu та T. Dai, "Online Map Application Development Using Google Maps API, SQL Database, and ASP.NET", International Journal of Information and Communication Technology Research, c. 102–103, 2013.

14. Воробьев А. В. Автоматизированный анализ невозмущенного геомагнитного поля на основе технологии картографических веб-сервисов / А. В. Воробьев, Г. Р. Шакирова. – 2017. – №5. – С. 177–187.

15. R. Thamizharasi, "Android Mobile Application Build on Android studio", International Journal of Modern Computer Science (IJMCS), c. 1–2, 2016.

16. Васильева К. Н. Обзор программных средств для разработки мобильных приложений / К. Н. Васильева, Г. Я. Хусаинова. // Technical science. – 2020. – №2. – С. 20–21.

17. N. Verma, S. Kansal та H. Malvi, "Development of Native Mobile Application Using Android Studio for Cabs and Some Glimpse of Cross Platform Apps", International Journal of Applied Engineering Research, c. 12527–12530, 2018.

18. Михеев Р. Э. Android Studio как платформа разработки электронных билетов / Р. Э. Михеев. – 2020. – №4. – С. 72–74.

19. N. Chatterjee та S. Chakraborty, "Real-time Communication Application Based on Android Using Google Firebase", International Journal of Advance Research in Computer Science and Management Studies, c. 74–76, 2018.

20. L. Dagne, "Flutter for cross-platform App and SDK development", Metropolia University of Applied Sciences, № 01 May, c. 2–7, 2019.

21. Родыгина И. В. Сравнительный анализ технологий для разработки серверной части системы управления продажами / И. В. Родыгина, А. В. Наливайко. // Технические науки. – 2018. – №2. – С. 256–266.

22. Данеев А. В. Об опыте разработки мобильного приложения вопросов и

ответов / А. В. Данеев, Р. А. Данеев, И. А. Рыжов. // Современные технологии. Системный анализ. Моделирование. – 2021. – №3. – С. 202–211.

23. Getting Started with Flutter Bloc Pattern [Електронний ресурс] – Режим доступу до ресурсу: https://www.mitrais.com/news-updates/getting-started-with-flutter-bloc-pattern/.

24. BLOC -Business Logic Components Pattern [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://medium.com/nerd-for-tech/bloc-business-logic-components-pattern-8b0b9a01d611.

Додаток А

Код мобільного додатка

Файл main.dart

```
import 'package:flutter/material.dart';
import 'map.dart';


void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Safe Shelter',
      theme: ThemeData(
        primaryColor: Colors.grey
      ),
      home: MapSample(),


    );
  }
}
```

Файл Start.dart:

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

class StartPage extends StatefulWidget {
  StartPage({Key? key}) : super(key: key);

@override
_StartPageState createState() => _StartPageState();
}

class _StartPageState extends State<StartPage> {

  @override
  Widget build(BuildContext context) {
    Widget _logo() {
      return Padding(
        padding: EdgeInsets.only(top: 100),
```

```
    child: Container(
      child: Align(
        child: Text(
          'Safe Shelter',
          style: TextStyle(
              fontSize: 45,
              fontWeight: FontWeight.bold,
              color: Colors.white),
        ),
      ),
    ),
  );

}

Widget _button(String text) {
  return ElevatedButton(
      style: ElevatedButton.styleFrom(

        alignment: Alignment.center,
          backgroundColor: Colors.white,

          textStyle: const TextStyle(
            fontSize: 20,

            fontWeight: FontWeight.bold,
            color: Colors.white,
            backgroundColor: Colors.white,
          )),
      onPressed: () {

      },
      child: Text(
        text,
        textAlign: TextAlign.center,

        style: TextStyle(

            fontWeight: FontWeight.bold,
            color: Colors.black87,
            fontSize: 20),
      ));
}
return Scaffold(
  backgroundColor: Theme.of(context).primaryColor,
```

```
      body: Column(

        children: <Widget>[
          _logo(),
          SizedBox(height: 100,),
          Container(child: Padding(
          padding: EdgeInsets.only(left: 20, right: 20),
          child:
          Row(
            children: <Widget>[
              Container(
                height: 55,
                color: Theme.of(context).primaryColor,
                child: Icon(
                  Icons.info_sharp,
                  color: Colors.white,
                ),
              ),
            Container(
              width: 340,
              child: Flexible(

                  child: _button('Правила поведінки під час
тривоги',
                  ),
              ),),
            ],
          ),),),
          SizedBox(height: 50,),

          SizedBox(height: 50,),
          Container(child: Padding(
            padding: EdgeInsets.only(left: 20, right: 20),

            child:
            Row(
              children: <Widget>[
                Container(
                  height: 55,

                  color: Theme.of(context).primaryColor,
                  child: Icon(
                    Icons.local_fire_department,
                    color: Colors.white,
                  ),
```

```
                ),
                Container(
                  width: 340,
                  child: Flexible(

                    child: _button('Предмети які потрібно
уникати',
                    ),
                  ),),
              ],
            ),),),),
          SizedBox(height: 300,),
          Container(child: Padding(
            padding: EdgeInsets.only(left: 20, right: 20),

            child:
            Row(
              children: <Widget>[
                Container(
                  height: 55,

                  color: Theme.of(context).primaryColor,
                  child: Icon(
                    Icons.map_rounded,
                    color: Colors.white,
                  ),
                ),
                Container(
                  width: 340,
                  child: Flexible(

                    child: _button('Перейти до мапи',
                    ),
                  ),),
              ],
            ),),),),
        ],
      ),
    );
}}
```

Файл register.dart:

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

class StartPage extends StatefulWidget {
  StartPage({Key? key}) : super(key: key);

@override
_StartPageState createState() => _StartPageState();
}

class _StartPageState extends State<StartPage> {

  @override
  Widget build(BuildContext context) {
    Widget _logo() {
      return Padding(
        padding: EdgeInsets.only(top: 100),
        child: Container(
          child: Align(
            child: Text(
              'Safe Shelter',
              style: TextStyle(
                  fontSize: 45,
                  fontWeight: FontWeight.bold,
                  color: Colors.white),
          ),
        ),
      ),
    );

    }

    Widget _button(String text) {
      return ElevatedButton(
          style: ElevatedButton.styleFrom(

            alignment: Alignment.center,
              backgroundColor: Colors.white,

              textStyle: const TextStyle(
                fontSize: 20,

                fontWeight: FontWeight.bold,
                color: Colors.white,
                backgroundColor: Colors.white,
```

```
            )),
        onPressed: () {

        },
        child: Text(
          text,
          textAlign: TextAlign.center,

          style: TextStyle(

              fontWeight: FontWeight.bold,
              color: Colors.black87,
              fontSize: 20),
        ));
    }
    return Scaffold(
      backgroundColor: Theme.of(context).primaryColor,
      body: Column(

        children: <Widget>[
          _logo(),
          SizedBox(height: 100,),
          Container(child: Padding(
          padding: EdgeInsets.only(left: 20, right: 20),
          child:
          Row(
            children: <Widget>[
              Container(
                height: 55,
                color: Theme.of(context).primaryColor,
                child: Icon(
                  Icons.info_sharp,
                  color: Colors.white,
                ),
              ),
            Container(
              width: 340,
              child: Flexible(

                  child: _button('Правила поведінки під час
тривоги',
                  ),
              ),),
            ],
        ),),),
```

```
            SizedBox(height: 50,),

            SizedBox(height: 50,),
            Container(child: Padding(
              padding: EdgeInsets.only(left: 20, right: 20),

              child:
              Row(
                children: <Widget>[
                  Container(
                    height: 55,

                    color: Theme.of(context).primaryColor,
                    child: Icon(
                      Icons.local_fire_department,
                      color: Colors.white,
                    ),
                  ),
                  Container(
                    width: 340,
                    child: Flexible(

                      child: _button('Предмети які потрібно
уникати',
                      ),
                    ),),
                ],
              ),),),
            SizedBox(height: 300,),
            Container(child: Padding(
              padding: EdgeInsets.only(left: 20, right: 20),

              child:
              Row(
                children: <Widget>[
                  Container(
                    height: 55,

                    color: Theme.of(context).primaryColor,
                    child: Icon(
                      Icons.map_rounded,
                      color: Colors.white,
                    ),
                  ),
                  Container(
```

```
                    width: 340,
                    child: Flexible(

                      child: _button('Перейти до мапи',
                      ),
                    ),),
                ],
              ),),),),
          ],
        ),
      );
}}
```

Файл map.dart:

```
import 'dart:async';
import 'dart:convert';
import 'package:flutter/services.dart' show rootBundle;
import 'package:geolocator/geolocator.dart';
import 'package:flutter/material.dart';
import
'package:google_maps_flutter/google_maps_flutter.dart';




import 'package:google_maps_webservice/places.dart';

class MapSample extends StatefulWidget {
  @override
  State<MapSample> createState() => MapSampleState();

}

class MapSampleState extends State<MapSample> {
  Set<Marker> _markers = {};


  Completer<GoogleMapController> _controller = Completer();



  @override
  void initState() {
```

```
    super.initState();
    setState(() {
      loadMarkers();

    });

  }

  static final CameraPosition _Kyiv = CameraPosition(
    target: LatLng(50.450001,30.523333),
    zoom: 14.4746,
  );




  Future loadMarkers() async {

    var jsonData = await
rootBundle.loadString('assets/sss.json');

    var data = json.decode(jsonData);

    data["features"].forEach((item) {
      _markers.add(new Marker(
          markerId: MarkerId(item["properties"]["id"]),
          position: LatLng(
              double.parse(item["properties"]["latitude"]),
double.parse(item["properties"]["longitude"])),

          infoWindow: InfoWindow(
              title: item["properties"]["name"],
              snippet: item["properties"]["region"],

          ),
          //icon: BitmapDescriptor.defaultMarkerWithHue(
          //   BitmapDescriptor.hueGreen)));
          icon: BitmapDescriptor.defaultMarker));
    });
  }

  var x;


  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
        body: GoogleMap(
          myLocationEnabled: true,


          markers: Set.from(_markers),

          initialCameraPosition: _Kyiv,
          onMapCreated: (GoogleMapController controller) {
            _controller.complete(controller);
          },

        ),
        floatingActionButton: Row(
          children: [
            Padding(
                padding: const EdgeInsets.only(left: 10,
right: 20),
                child: FloatingActionButton.extended(
                  onPressed: _goHome,
                  label: Icon(Icons.my_location, ),
                  backgroundColor: Colors.green,
                )),
            Padding(
                padding: const EdgeInsets.only(left: 10,
right: 20),
                child: FloatingActionButton.extended(

                  onPressed:  _goss,
                  label: Text("укриття"),

                  backgroundColor: Colors.green,
                )),
          ],
          mainAxisAlignment: MainAxisAlignment.center,
          mainAxisSize: MainAxisSize.max,
        ));
  }


  Future _goss() async {
```

```
     Set.from(_markers);



   final GoogleMapController controller = await
_controller.future;

controller.animateCamera(CameraUpdate.newCameraPosition(Came
raPosition(
       target: _markers.last.position,
       zoom: 17,
   )));
  }


  Future _goHome() async {
    Geolocator
        .getCurrentPosition(desiredAccuracy:
LocationAccuracy.best)
        .then((Position position) async {
    final GoogleMapController controller = await
_controller.future;

controller.animateCamera(CameraUpdate.newCameraPosition(Came
raPosition(
      target: LatLng(position.latitude, position.longitude),
      zoom: 17,
    )));
    return  LatLng(position.latitude, position.longitude);
  });
  }



 /* Future<void> _getCurrent() async {
    Geolocator
        .getCurrentPosition(desiredAccuracy:
LocationAccuracy.best)
        .then((Position position) async {
      final GoogleMapController controller = await
_controller.future;

controller.animateCamera(CameraUpdate.newCameraPosition(Came
```

```
raPosition(
        target: LatLng(position.latitude,
position.longitude),
        zoom: 14.4746,
      )));
    }).catchError((e) {
      print(e);
    });
  }
*/
}

class FirebaseKeys {
}
```

Файл Homepage.dart:

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'components/Homecom.dart';



class HomePage extends StatefulWidget{
  HomePage ({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int sectionsIndex = 0;

  @override
  Widget build(BuildContext context){
    return Container(
      child: Scaffold(
        backgroundColor: Theme.of(context).primaryColor,
        appBar: AppBar(
          title: Text('Алгоритм дій при сирені'),  leading:
Icon(Icons.local_fire_department),

        ),
```

```
        body:InfoList(),



      ),
    );
  }
}
```

Файл Homecon,dart:

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

import '../info.dart';

class InfoList extends StatelessWidget {


  final Infos = <Info>[
    Info(auth: 'my', description: 'Авіаційні бомби', title:
'test1'),
    Info(auth: 'my', description: 'Авіаційні Ракети', title:
'test1'),
    Info(auth: 'my', description: 'Артилерійські снаряди',
title: 'test1'),
    Info(auth: 'my', description: 'Балістичні ракети',
title: 'test1'),
    Info(auth: 'my', description: 'Ручні гранати', title:
'test1'),
    Info(auth: 'my', description: 'Гранатометні постріли',
title: 'test1'),
    Info(auth: 'my', description: 'Касетні боєприпаси',
title: 'test1'),
    Info(auth: 'my', description: 'Міни пастки', title:
'test1'),
    Info(auth: 'my', description: 'Мінометні міни', title:
'test1'),
    Info(auth: 'my', description: 'Осколкові протипіхотні
міни', title: 'test1'),
    Info(auth: 'my', description: 'Протитанкові мінит
фугасної дії', title: 'test1'),
    Info(auth: 'my', description: 'Ракети до різних РСЗВ',
title: 'test1'),
    Info(auth: 'my', description: 'Днони-камікадзе', title:
'test1'),
```

```
  ];

  final Infoss = <Info>[
    Info(auth: 'my', description: '  \n 1. Виконати заходи, передбачені на цей випадок Планом дій або Інструкцією, діяти за вказівками керівництва;  \n  2. Швидко, без паніки зайняти місце у захисній споруді (сховищі, підвальному приміщенні) та виконувати вимоги старшого (коменданта).', title: 'Якщо почули сирени, перебуваючи на роботі:'),

    Info(auth: 'my', description: ' \n1. Ввімкнути телевізор чи радіоприймач і уважно прослухати інформацію про характер тривоги;  \n 2. За можливості попередити сусідів і одиноких людей, що мешкають поруч;  \n3. Швидко одягнутися та одягнути дітей, перевірити наявність пришитих з внутрішньої сторони одягу у дітей дошкільного віку нашивок, на яких зазначено: прізвище, ім’я, по батькові, адреса, вік, номери телефонів батьків;\n 4. Закрити вікна, вимкнути усі електричні та нагрівальні прилади, перекрити газ, загасити печі, вимкнути світло (автоматичну коробку, рубильник тощо); \n 5. Взяти «тривожну валізу» (індивідуальні засоби захисту, запас продуктів і води, особисті документи, кишеньковий ліхтар) та найкоротшим шляхом прямувати до найближчої захисної споруди чи укриття.  \n У разі відсутності в радіусі 500 м від вашого будинку захисної споруди використовуйте для укриття підвальне приміщення під будинком.', title: 'Якщо почули сирени, перебуваючи вдома:'),

  ];

  static get auth => null;

  static get description => null;

  static get title => null;

  @override
  Widget build(BuildContext context) {
    return Container(

      child: Container(
          child: ListView.builder(
              itemCount: Infoss.length,
```

```
              itemBuilder: (context, i) {
                return Card(
                    elevation: 2.0,
                    margin: EdgeInsets.symmetric(horizontal:
8, vertical: 4),
                    child: Container(
                      decoration: BoxDecoration(color:
Colors.blueAccent),
                      child: ListTile(
                        contentPadding:
EdgeInsets.symmetric(horizontal:10),
                        leading: Container(
                          padding: EdgeInsets.only(right:
12),
                            child: Icon(Icons.info_outline,
color: Colors.white,),
                        ),
                        title: Text( Infoss[i].title, style:
TextStyle(color: Colors.white,fontSize: 22, fontWeight:
FontWeight.bold),),
                        subtitle: Text(
Infoss[i].description, style: TextStyle(color:
Colors.white,fontSize: 16,),)
                      ),
                    ));
              }
          )
      ),
    );
  }
}
```

ДОДАТОК В

Публцікація у вигляді тези: Вступ. Кожного дня в світі відбувається багато надзвичайних ситуацій різного характеру, кожна держава світу має дбати про своє цивільне населення та сповіщати їх про можливу загрозу заздалегідь, це може врятувати безліч життів. Як ми знаємо в 21 столітті важко уявити людей які не користуються різними гаджетами такими як: мобільні телефони, планшети, комп’ютери, тощо. Існує чимало відомих мобільних додатків сповіщення про повітряну тривогу, які досягли більшого чи меншого успіху у застосування, але в них є невирішена задача пошуку найближчого укриття.

Метою роботи є створення автоматизованого мобільного додатка основним функціоналом якого є отримання сповіщення про повітряну тривогу та пошук найближчого укриття у місті Київ, з додатковою функцією інформування користувачів про потенційно небезпечні об'єкти та поради щодо поведінки під час повітряної тривоги.

Матеріали та методи. Додаток був написаний за допомогою мови програмування dart та фреймворку flutter, в якось бази даних було використано firebase, базу даних укриттів було взято з офіційного сайту Київської міської ради [1]. Інформацію щодо дій під час повітряної тривоги було взято з офіційного сайту Рівненської обласної державної адміністрації. На (рис. 1.) зображено блок схему роботи додатку від початку до кінця.

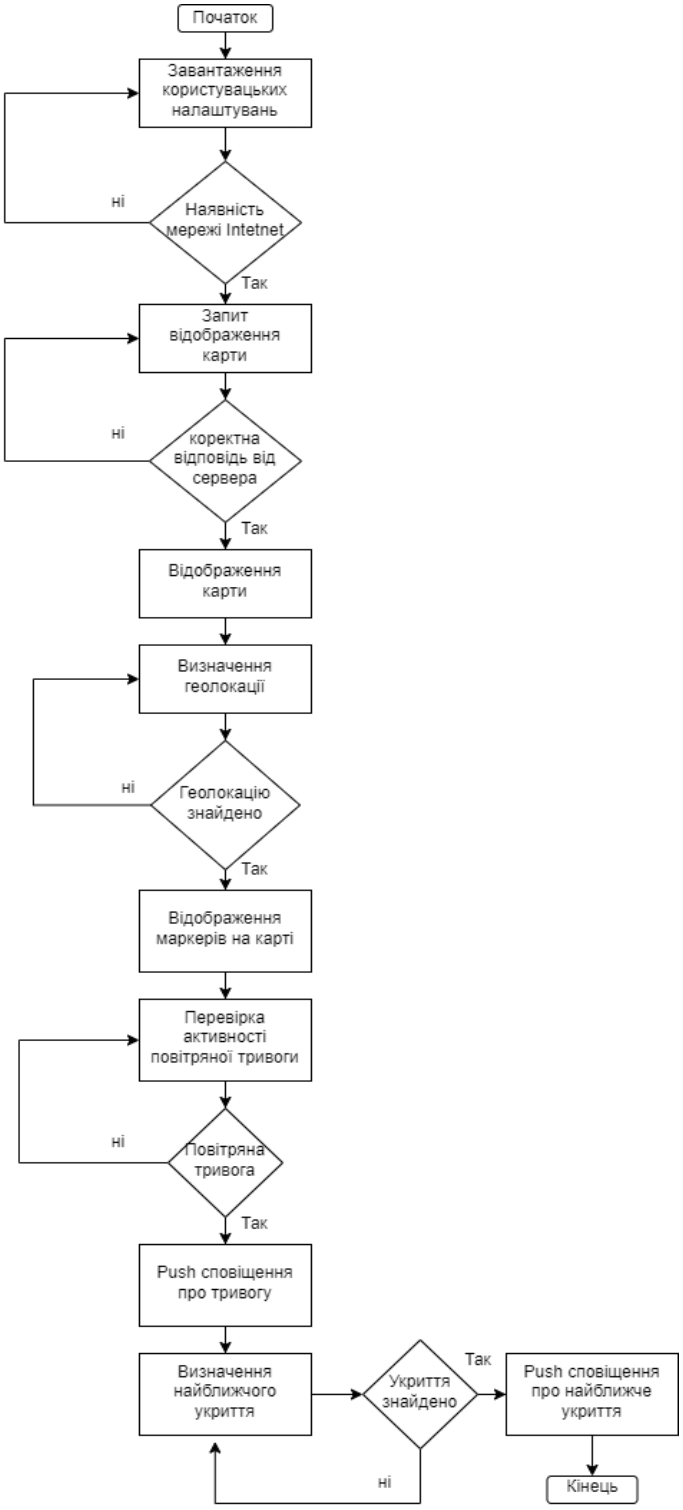

Рис. 1. Блок схема роботи додатка

Google Firebase – сервіс, що надає різні хмарні послуги та інструменти для розробки: база даних реального часу, хмарні функції, хостинг, сервіс для автентифікації користувачів та багато іншого. Для створення автоматизованої

системи охорони праці використовується інструмент Cloud Firestore, це хмарна база даних NoSQL, що дозволяє зберігати та синхронізувати дані [2].

Flutter - це крос-платформний фреймворк з єдиною кодовою базою, що працює мовою програмування Dart. Запущений тільки в 2018 році Google, Flutter зарекомендував себе як зручний набір інструментів, легкий для створення анімації та якісних компонентів інтерфейсу користувача. Незважаючи на те, що Google недавно почав використовувати крос-платформу, Flutter надає плавну анімацію та зручні елементи інтерфейсу [3].

Рузультати та обговорення. В результаті виконання магістерської дисертації було створено мобільний додаток який автоматично сповіщає користувача про повітряну тривогу, після цього відбувається автоматичний пошук найближчого укриття до поточної геолокації користувача. Користувач отримує Push-сповіщення на телефон в якому інформується про повітряну тривогу, а також користувач має можливість перейти до карти з маршрутом який веде до найближчого укриття.

Висновки. Додатки які сповіщують про повітряну тривогу можна назвити критичними, їх швидкість роботи та зручність використання мають бути на вищому рівні.

Завдяки реалізації мобільного додатку обраного в магістерській дисертації було значно зменшено час користувача для пошуку потрібної інформації, автоматизовано процес сповіщення та пошуку.

ЛІТЕРАТУРА

1. "Офіційний портал Києва". kyivcity. https://kyivcity.gov.ua/ (дата звернення 11 жовт. 2022).

2. Калиневич Н. Разработка кросс-платформенных приложений на языке Dart при помощи фреймворка Flutter / Н. Калиневич, Р. Гильванов. // Intellectual Technologies on Transport. – 2021. – №4. – С. 21–27.

3. Нургалиев Р. К. Внедрение цифровых технологий для обеспечения контроля передвижения работников в рамках организации производства / Р. К. Нургалиев, Е. Ю. Климанова,, О. В. Зеленко. // 6. – 2021. – С. 56–61.

Сертифікат публікації: