МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Катерина Нестеренко

“____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ
“ МАГІСТРА”

Тема: “Методика та застосунок для оповіщення сигналу тревоги та формування маршруту до укриття”

Виконавець: Абібуллаєв Мустафа Едемович

Керівник: к.т.н. доцент Олександр Андрійович Ткаченко

Нормоконтролер: Гололобов Дмитро Олександрович ст.в

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Катерина Нестеренко
"___" _____ 2023 р

ЗАВДАННЯ
на виконання дипломного проекту студента
Абібуллаєва Мустафи Едемовича

1. Тема проекту: «Методика та застосунок для оповіщення сигналу тревоги та формування маршруту до укриття»
затверджена наказом ректора від 29.09.2023 р. № 1994/ст

2. Термін виконання проекту: з 02.10.2023 р. до 31.12.2023 р.

3. Вихідні данні до проекту: розробка програмного продукту на основі .NET 6.0 та Telegram Bot API.

4. Зміст пояснювальної записки:
   1. Аналіз предметної області та дослідження ідеї.
   2. Опис вимог та методика оповіщення сигналу тревоги та формування маршруту до укриття.
   3. Розробка та опис структури інформаційної системи.
   4. Оцінка та тестування програмного застосунку.

5. Перелік обов'язкових слайдів презентації:
   1. Мета роботи.
   2. Вимоги до системи.
   3. Архітектура системи.
   4. Робота системи.
   5. Майбутній розвиток.

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Оформлення перших 2-х сторінок пояснювальної записки (ПЗ) – титулка, завдання.	02.10.23 – 08.10.23	
2.	Ознайомлення з постановкою задачі та вивчення літератури. Написання 1 розділу, представлення керівнику. 1-ий нормо-контроль.	09.10.23– 15.10.23	
3.	Підготовка остаточного плану та узгодження його з керівником	16.10.23– 22.10.23	
4.	Обговорення та редагування першого розділу з керівником, підготовка до написання другого розділу	23.10.23– 29.10.23	
5.	Написання другого розділу, представлення керівнику, підготовка до написання третього розділу	30.10.23 – 05.11.23	
6.	Написання 3 розділу, представлення керівнику, форматування та виправлення існуючого тексту.	06.11.23 – 19.11.23	
7.	Загальне редагування та друк пояснювальної записки, графічного матеріалу. Отримання відгуку керівника, рецензії та заповнення листа про доброчесність. Розробка доповіді	20.11.23 – 03.12.23	
8.	Завершення написання ПЗ. Проходження нормо-контролю. Друк ПЗ. Отримання відгуку керівника. Підготовка презентації та доповіді на перед захист.	04.12.23 - 10.12.23	
9.	Передзахист каліф. Роботи. Отримання рецензії	11.12.23 - 17.12.23	
10.	Підготовка документів до захисту та здача їх секретарю ДЕК	18.12.23 - 24.12.23	
11.	Захист дипломної роботи перед ЕК	25.12.23 - 31.12.23	

Дата видачі завдання 02.10.2023

Керівник: к.т.н. доцент Олександр ТКАЧЕНКО

Завдання прийняв до виконання: Мустафа АБІБУЛЛАЄВ

Дата 02.10.2023

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Методика та застосунок для оповіщення сигналу тревоги та формування маршруту до укриття»: 98 с., 42 рис., 0 табл., 34 інформаційних джерел.

ТЕЛЕГРАМ БОТ, OPEN STREET MAP, ФОРМУВАННЯ МАРШРУТІВ, TELEGRAM BOT API, .NET, C#

Об'єкт дослідження – можливості для оповіщення сигналу тревоги та формування маршруту до укриття.

Мета дипломної роботи – підвищення безпеки населення шляхом оповіщення про повітряні тривоги та полегшення процесу пошуку найближчих безпечних укриттів.

Метод дослідження – методи управління процесами життєвого циклу розробки програмного забезпечення у гнучких методологіях, аналіз предметної області.

В процесі роботи було досліджено: 1) методи формування маршруту; 2) зміни у діях населення під час повітряної тривоги, що виникли в результаті змін у психіці людей; 3) засоби інформації, що могли попередити населення про повітряну тривогу та сформувати маршрут до найближчого укриття.

Результат роботи може бути використаним для проектування та імплементації застосунків, пов'язаних з роботою з мапами або з оповіщенням про повітряні тривоги.

Розробка та дослідження проводилися під управлінням ОС Windows XP. Розробка програми проводилася у середовищі .NET Core, а саме 6-ої версії, на мові програмування C#.

ABSTRACT

Explanatory note to the diploma thesis "Methodology and application for alerting an alarm signal and forming a route to shelter": 98 pages, 42 figures, 0 table, 34 information sources.

TELEGRAM BOT, OPEN STREET MAP, ROUTE FORMATION, TELEGRAM BOT API, .NET, C#

Object of research are possibilities for alerting an alarm signal and forming a route to shelter.

Purpose of research is to increase the safety of the population by notifying about air alarms and facilitating the process of finding the nearest safe shelters.

Research method – methods of managing software development life cycle processes in flexible methodologies, subject area analysis.

During the work, the following were investigated: 1) methods of route formation; 2) changes in the actions of the population during an air raid, which arose as a result of changes in the psyche of people; 3) means of information that could warn the population about an air alert and form a route to the nearest shelter.

The result of the work can be used for the design and implementation of applications related to working with maps or with the notification of air alarms.

Development and research were carried out under Windows XP operating system. The program was developed in the .NET Core environment, as well as the 6th version, in the C# programming language.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

API - Appliccation Programming Interface

IDE - Integrated Development Environment

ОДА - Обласна державна адміністрація

DFS - Depth-First Search

BFS - Breadth-First Search

OSPF - Open Shortest Path First

HTTPS - HyperText Transfer Protocol Secure

ВСТУП

Головна мета даної дипломної роботи полягає в розробці методики та застосунку для ефективного оповіщення сигналу тривоги та формування маршруту до укриття в умовах реальної загрози, що виникає внаслідок війни на території України. Сучасна обстановка визначає необхідність створення надійних та ефективних систем, призначених для захисту населення у випадку надзвичайних ситуацій.

Наразі існує багато систем, які працюють на оповіщення населення про небезпеку, проте з різних причин вони можуть не працювати з деякими категоріями населення, що і вплинуло на створення та розвиток даної ідеї.

Дослідження предметної області в даній темі передбачає кінцевий продукт, проект, що використовує сучасні технології мапування та геолокації для надання точної інформації щодо потенційних загроз для користувачів та способів убезпечити себе. Інтерфейс та сповіщення бота будуть спрощеними і інтуїтивно зрозумілими, дозволяючи швидко реагувати на небезпеку та вживати необхідні заходи для захисту.

У роботі використані методи системного аналізу, такі як метод моніторингу, індукції та дедукції.

Окрім того, у роботі проводиться аналіз існуючих систем та додатків, що використовуються для оповіщення та навігації в екстрених ситуаціях. Цей огляд дозволить ідентифікувати сильні та слабкі сторони існуючих рішень, а також визначити напрямки подальших вдосконалень для нашого додатку з метою забезпечення максимальної ефективності та безпеки для користувачів.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ ОПОВІЩЕННЯ ТА УБЕЗПЕЧЕННЯ РІЗНИХ КАТЕГОРІЙ НАСЕЛЕННЯ

У сучасних умовах в Україні зросла актуальність простих та зрозумілих засобів для передачі важливої інформації. Це і відповідне апаратне забезпечення, і нові програмні застосунки. Багато людських життів залежать від подібних систем. Через велику ціну помилок ефективність цих розробок з часом лише збільшується, все більш зрозумілими та дієвими вони стають. Одним з типів таких застосунків є додатки для оповіщення людей про повітряну тривогу. Звісно, для цієї мети існують фізичні сирени, які характерним звуком попереджують про небезпеку (рис 1.1). Ми чуємо сигнал повітряної тривоги, коли електродвигун приводить у рух великі маховики, що, завдяки повітряному тиску, виробляють характерний протяжний звук.

Існують кілька видів сигналів.

- Довгий і рівномірний гудок тривалістю до 7 секунд - це сигнал попередження. По його виявленні слід ввімкнути офіційні радіо- чи телеканали та слухати подальші інструкції.
- Звук, що наростає і спадає, триває одну хвилину і повторюється принаймні тричі, свідчить про необхідність перейти в укриття. Це сигнал загальної тривоги.
- Короткий сигнал тривалістю у п'ять секунд, який лунає після загальної тривоги, позначає завершення повітряної тривоги.

В Україні досі працює радянська система оповіщення цивільного населення, вона не оновлювалася понад 50 років. Якщо через, наприклад, відсутність електроенергії, сирена не працює, населення повинно бути оповіщене за допомогою гучномовців, що встановлені на автомобілях патрульної поліції, пожежників, рятувальників або аварійної служби.

Проте не всюди подібні системи є в наявності або нормально працюють.

Рис. 1.1. Сигнальна сирена повітряної тривоги

Наприклад, не всюди є подібні сирени, малі містечка та села час від часу не мають такої можливості. Також є побутові причини, не завжди люди спроможні почути таку сирену, наприклад, особи з вадами слуху, люди, що працюють у шумному середовищі або молодь, що вже звикла до технологій та не обійдеться без навушників з музикою. Ці пункти призвели до необхідності в більш доступних та поширених засобах оповіщення.

Одним зі способів стали офіційні оповіщення на телефони від Державної служби України з надзвичайних ситуацій (рис. 1.2). Ця система була протестована у вересні 2022 року. На мобільні телефони приходили текстові повідомлення та звукові сигнали, що попереджували населення про небезпеку. Тестування пройшло успішно, проте систему досі не використовують. Причиною цьому є те, що рішення про використання повинно бути прийняте органами місцевої влади кожного району згідно з Кодексом цивільного захисту України.[1]

Рис. 1.2. Оповіщення про повітряну тривогу від Державної служби України з надзвичайних ситуацій

Але все ж найочевиднішим варіантом інформування, звісно, став Інтернет. Зважаючи на те, що технології врешті-решт проникають усюди, в усіх типах населених пунктів зараз є достатня кількість людей зі смартфонам або комп'ютерами, що мають доступ до мережі Інтернет, тому почали з'являтися мобільні додатки та програми для комп'ютерів, що надсилають повідомлення у разі загрози.

Тим не менш, з часом люди також просто звикають до того, що час від часу виникає подібна загроза, в тому числі через неефективність засобів оповіщення. Тому саме час звернутися до психології.

1.1. Аналіз психологічної сторони проблеми

Війна – тяжке випробування для психіки кожної людини, а особливо тих, хто не стикався з подібним раніше. Військова служба є фактором ризику розвитку посттравматичного стресового розладу (ПТСР). Близько 78% людей, які зазнали бойових дій, не розвивають ПТСР; приблизно у 25% військовослужбовців, у яких розвивається ПТСР, його поява відстрочена.[2] Біженці також мають підвищений ризик посттравматичних стресових розладів через те, що вони пережили війну, труднощі та травматичні події. Рівень ПТСР серед біженців коливається від 4% до 86%.[3] Незважаючи на

те, що стреси війни впливають на всіх, хто бере участь, переміщені особи, як було показано, страждають більше, ніж інші.[4]

Проблеми, пов’язані із загальним психосоціальним благополуччям біженців, є складними та мають індивідуальні нюанси. Біженці мають знижений рівень добробуту та високий рівень психічних розладів через минулі та поточні травми. Особливо постраждалі групи, чиї потреби часто залишаються незадоволеними, це жінки, люди похилого віку та неповнолітні без супроводу. Посттравматичний стрес і депресія серед біженців також, як правило, впливають на їхні успіхи в навчанні.[5]

Подібні ситуації викликають сильний стрес. Але врешті-решт людина починає підлаштовуватися під нову реальність і в середньому за місяць звикає. Адаптація – нормальна реакція на швидкі зміни в житті. Проте, до звикання стан стресу викликає різні емоції. Час від часу можуть відбуватися сплески емоцій. Наприклад, стан постійного страху. Є люди, які, усвідомлюючи небезпеку, не бажають їхати зі звичного дому у безпечне місце і кажуть, що дуже налякані. Вони не можуть замінити зручність на відчуття захищеності. У звичному місці їм комфортно, але небезпечно. Але дуже важливо, щоб цивільні розуміли ризики та проводили заходи для убезпечення себе і близьких. Страх збуджує роботу лімбічної системи, що відповідає за емоції, замість ділянки мозку, що працює з критичним мисленням. Змусивши себе подолати ці емоції, людина зможе увімкнути раціональне мислення, підвищуючи свою ефективність.

Також, такі умови викликають коливання емоцій, що є звичайним явищем (рис 1.3). Емоції повинні змінюватися, і це абсолютно нормально. Людина не може постійно перебувати в стані радості або суму. Під час війни коливання настрою стають більш вираженими: радість може перерости в ейфорію, а сум може перейти у жах.[6]

Рис. 1.3. Цикл змін настрою людини під час війни

Одним з факторів психологічних змін є сигнал повітряної тривоги, що став неодмінною частиною повсякденного життя усіх українців. Але навіть у цього фактора є власні деталі, пов'язані з психологією людей, що теж змінюються з часом.

За словами психологів, є два крайні випадки реакції на тривогу, що є найгіршими. Перший – людина ігнорує тривоги. Другий – людина настільки звикла реагувати на тривогу та спускатися в укриття, що їй вже здається, що це буденна справа. В обох випадках це означає вимкнення інстинкту самозахисту, що може спричинити ситуації, в яких місце цього інстинкту посяде дезорієнтованість і невпевненість. Вони можуть спричинити затримки в усвідомленні ситуації та у діях людини, що, в свою чергу, може стати причиною жахливих наслідків.

Якщо ці тривоги стають регулярними, люди можуть відзначити, що починають реагувати на них швидше, стають надто чутливими до кожного звуку. Це може бути розглянуто як нормальна реакція, оскільки всі реагують на буденні події. Люди відповідають на звуки навколишнього середовища, що є інстинктом самозахисту, виниклим від тваринного інстинкту. Якою повинна бути реакція? Якщо паніка стає постійним станом, і людина реагує тільки тривогою без можливості спокійно вирішити ситуацію, то вона не зможе належно орієнтуватися і раціонально вирішувати проблеми. Це може

лише створити додаткові труднощі. Коли люди дивляться на це як на «ситуацію», яку можна вирішити, вони можуть заспокоїтися. Якщо людина вже має чіткий план дій під час тривоги, вона стає організованою і діє відповідно до розробленого плану.[7]

Другою стороною медалі є те, що за останній час все більший відсоток населення ігнорує звук сирени, що може покласти під загрозу їх власне життя. Чому це відбувається?

Відсутність реакції на повітряну тривогу може бути розглянута як підлаштування до теперішньої реальності. Це своєрідний сигнал того, що населення вже пристосувалося до нових умов. Адаптація — це невід'ємна частина виживання. Проте сам процес адаптації зазнав певних змін. У зв'язку з тим, що люди вже дуже втомилися, як морально, так і фізично, інколи вони просто нехтують правилами через втомленість, проте свідомо розуміють ризики. Внаслідок цього перехід в укриття вже не є пріоритетом, і кожна людина несе свою відповідальність за своє життя та життя близьких.[8][9]

Окрім втоми можна виділити ще кілька причин для такої поведінки. Наприклад, апатія та фаталізм. Постійне перебування у стресовому середовищі, що іноді розбавляється історіями з хорошим закінченням про неочікуваний порятунок людей, створює у деяких осіб відчуття того, що від них нічого не залежить, і якщо щось станеться, то вони ніяк не могли на це вплинути.

Також іноді можна почути аргумент про «статистику»: коли людина часто чує тривоги, але нічого не відбувається, створюється хибне відчуття «безпеки», що зменшує пильність та раціональність суджень про тривогу, що, звісно, не є правдою.

Схожим чином інші можуть покластися на свій власний досвід, бути занадто самовпевненими, щиро думати, що саме з ними в жодному випадку нічого не станеться. Такі люди зазвичай працюють на якихось небезпечних роботах у мирний час, проте постійна небезпека, яка нависла над країною зараз має зовсім інший масштаб.

Нарешті, є ще один простий мотив для людей, що ігнорують сигнали тривоги – стадний інстинкт. Коли у когось в оточенні з'являється перший приклад нехтування безпекою, це може вплинути на погляд цієї людини, і, в свою чергу, викликати ланцюгову реакцію на інших. Бажання не виділятися серед інших може створювати подібні ірраціональні звички, що прямо наражають усіх на небезпеку.[10]

Ще одним фактором є неефективність систем оповіщення населення про повітряні тривоги. Вона може впливати на реакцію людей у випадку небезпеки різними способами.

Наприклад:

- Затримки - якщо механізми оповіщення працюють неналежним чином, це може викликати затримку у передачі важливої інформації громаді. Люди можуть отримати повідомлення про тривогу пізніше необхідного або не отримати його взагалі, що ускладнить їх здатність прийняти ефективні заходи безпеки.
- Недостатня інформація – у випадку частих помилкових тривог у громадськості може зменшитися довіра до системи. Люди можуть стати байдужими або менш обережними, оскільки не можуть відрізнити серйозну тривогу від невірного сигналу. Особливо це стосується ситуацій, коли потенційні носії ракет «Кинджал» викликають тривогу на всій території України, що зупиняє життя багатьох людей. Але з часом люди звикли до таких ситуацій, і реакція на подібні сигнали стає все слабкішою.
- Недоступність для всіх шарів населення - якщо система оповіщення не адаптована до потреб всіх груп населення, це може створити ризик для тих, хто перебуває в уразливих ситуаціях, таких як люди з обмеженими можливостями. Інклюзивність системи оповіщення є важливим аспектом, оскільки всі повинні мати можливість отримати і розуміти тривоги.

- Соціокультурні аспекти - урахування соціокультурних реалій важливо при розробці систем оповіщення, оскільки різні групи можуть реагувати по-різному. Наприклад, ефективне оповіщення в міських та сільських районах може вимагати різних підходів.

Врахування цих аспектів та систематична перевірка й удосконалення систем оповіщення є ключовими для забезпечення найвищого рівня безпеки населення під час екстрених ситуацій. Також важливою має бути уніфікація усіх методів, бажано під егідою уряду, з метою збільшення достовірності інформації.

Байдужість до проблеми становить велику загрозу для життів людей, і через це потрібно розробляти нові варіанти вдосконалення подібних механізмів.

1.2. Актуальність та обґрунтування варіантів вдосконалення систем оповіщення

Враховуючи вищесказане, потрібно виділити дієві шляхи покращення існуючих методів або створення нових для підвищення безпеки людей.

Вдосконалення неефективних систем оповіщення населення про повітряну тривогу може включати різні заходи з метою покращення ефективності та реакції людей. Ось деякі можливі варіанти поліпшення ситуації в цілому:

- Технічні зміни:

  Модернізація технічних засобів: застосування сучасних технологій для покращення систем оповіщення, таких як розумні мобільні додатки, системи автоматизованого надання інформації тощо.

  Резервні канали зв'язку: використання кількох різних способів оповіщення, таких як SMS, електронна пошта, соціальні мережі, щоб забезпечити найширшу можливу охопленість.
- Покращення інформаційного повідомлення:

Чіткі інструкції та рекомендації: забезпечення зрозумілих та конкретних вказівок для населення щодо дій під час повітряної тривоги.

Мовлення та мова: використання доступної та лаконічної мови, уникання термінів та технічних термінів, щоб забезпечити розуміння інформації найширшим колом аудиторії.

- Тестування та навчання:

  Регулярні тести систем: проведення регулярних тестів систем оповіщення для перевірки їх ефективності та виправлення можливих недоліків.

  Навчання населення: проведення навчань та навчальних кампаній, щоб підвищити обізнаність населення щодо дій під час повітряної тривоги.

- Аналіз та звітність:

  Аналіз відгуків та виправлення недоліків: систематичний аналіз відгуків та дослідження випадків, коли система не була ефективною, з метою вдосконалення.

  Швидка реакція на зміни: розробка процедур для швидкого внесення змін у систему на основі виявлених проблем.

- Залучення спільноти:

  Активна взаємодія з громадськістю: забезпечення двостороннього зв'язку між владою та населенням, приймаючи до уваги думки та запитання громадян.

  Спільнотні програми підготовки: організація тренінгів та вправ для громадян щодо ефективної реакції на повітряну тривогу.

Ці заходи спрямовані на створення більш надійних та ефективних систем оповіщення, які максимально забезпечують безпеку та реакцію населення в умовах повітряної небезпеки.

В першу чергу потрібно зосередити увагу на середовищі, що є найефективнішим та стане основою для попередження найбільшого обсягу людей.

Інтернет – одна з найпоширеніших на даний мереж в Україні. Розвиток зв’язку та мобільних пристроїв робить саме цей напрям подальшого покращення системи оповіщення найбільш привабливим.

За статистикою, у 2018-ому році 85% населення України віком від 18-30 років користувалися смартфонами (рис. 1.4). Найактивніше смартфони використовуються мешканцями великих і середніх міст - 54-55%. В маленьких містах під цю категорію підпадають близько 43% населення, а в селах - 32%.[11]

Порівняно з попередніми роками, частка користування смартфонами в Україні за останні два роки виросла на 26%. При цьому в середньому українці мали по 3-4 гаджета на особу. Якщо звернути увагу на способи купівлі пристроїв, то в 50% випадків люди передивлювалися товари через смартфон, а 43% здійснювали покупку також через мобільні гаджети. Окрім того, 39% жителів України використовують мобільні пристрої для пошуку інформації про різноманітні товари, 36% переглядають відео за їх допомогою, а 25% здійснюють покупки онлайн.[12]

Рис. 1.4. Статистика користування смартфонами молоддю в 2018-ому році

При цьому 91% українських власників сенсорних смартфонів користується мобільними додатками. Такий же відсоток українців користується месенджерами. Найпопулярнішим додатком для обміну повідомленнями виявився Viber. На другому місці - Facebook та Skype на третьому (рис. 1.5).[11]

Рис. 1.5. Користування месенджерами в 2018-ому році

За статистикою, отриманою після другого кварталу 2023 року, Україна займає високу позицію серед країн світу за обсягом безконтактних платежів, входячи до ТОП-5 у цьому рейтингу. Протягом останніх п'яти років наша країна стабільно входить до першої десятки лідерів за кількістю NFC-оплат.

Згідно інформації від компанії Mastercard, в другому кварталі поточного року частка безконтактних розрахунків через смартфони та гаджети з технологією NFC у фізичному ритейлі в Україні зросла до 60%. Таким чином, Україна опинилася в першій п'ятірці разом із Австралією, Великою Британією, Нідерландами та США.

Цей стрімкий розвиток NFC-оплат пов'язаний із збільшенням популярності цифрових гаманців, які перетворюють гаджети з NFC (смартфони, смартгодинники, фітнес-браслети) на зручний засіб оплати.

Також цей ріст пов'язаний з поширенням технології безконтактних розрахунків. Серед найпопулярніших категорій оплат вирізняються продуктовий ритейл, заклади харчування, одяг і взуття, а також автозаправні станції.

Mastercard також зауважила, що розвиток мережі платіжних терміналів, які підтримують безконтактні розрахунки, відіграв важливу роль у формуванні культури безготівкових оплат в Україні. За даними Національного банку України, майже 97% POS-терміналів в країні приймають безконтактні платежі.

Загальний підхід до розвитку безконтактних платежів та відкритість українців до інновацій стали ключовими чинниками, які забезпечили Україні визнане місце серед лідерів світового ринку NFC-оплат.[13]

Усі вищезазначені факти підтверджують ідею того, що варіанти покращення ситуації повинні рухатися в бік мобільних пристроїв, що все більше проникають у життя населення України. Але тут потрібна конкретика.

Одним з найзначущіших інтернет-трендів у минулі роки стало використання соціальних мереж не тільки для взаємодії та розваг, але й як джерело новин та інформації. Ця тенденція є помітною як у країнах Заходу (наприклад, близько 53% громадян у США), так і в Україні, де відсоток користувачів, які отримують новини через соціальні мережі, зросла з 45% до 63%. Під час повномасштабного вторгнення цей відсоток виріс до 76.7% (рис. 1.6).

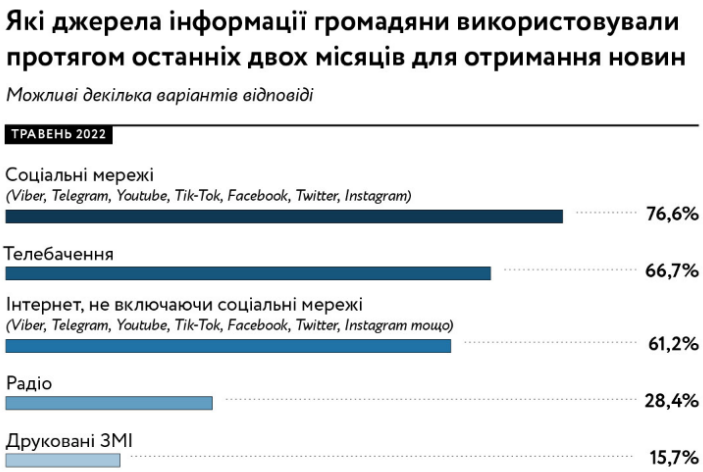


Рис. 1.6. Джерела інформації для отримання новин громадянами України влітку 2022 року

Найпопулярнішими соцмережами для отримання новин серед українців влітку 2022 року були Telegram (65,7%), YouTube (61,2%) і Facebook (57,8%). 48% відповідачів використовували Viber, 29,1% обирали Instagram, 19,5% віддали перевагу TikTok, а 8,9% звертались до Twitter. Близько 2% опитаних використовували інші соціальні мережі для отримання новин, такі як WhatsApp, Signal і так далі (рис. 1.7).[14]

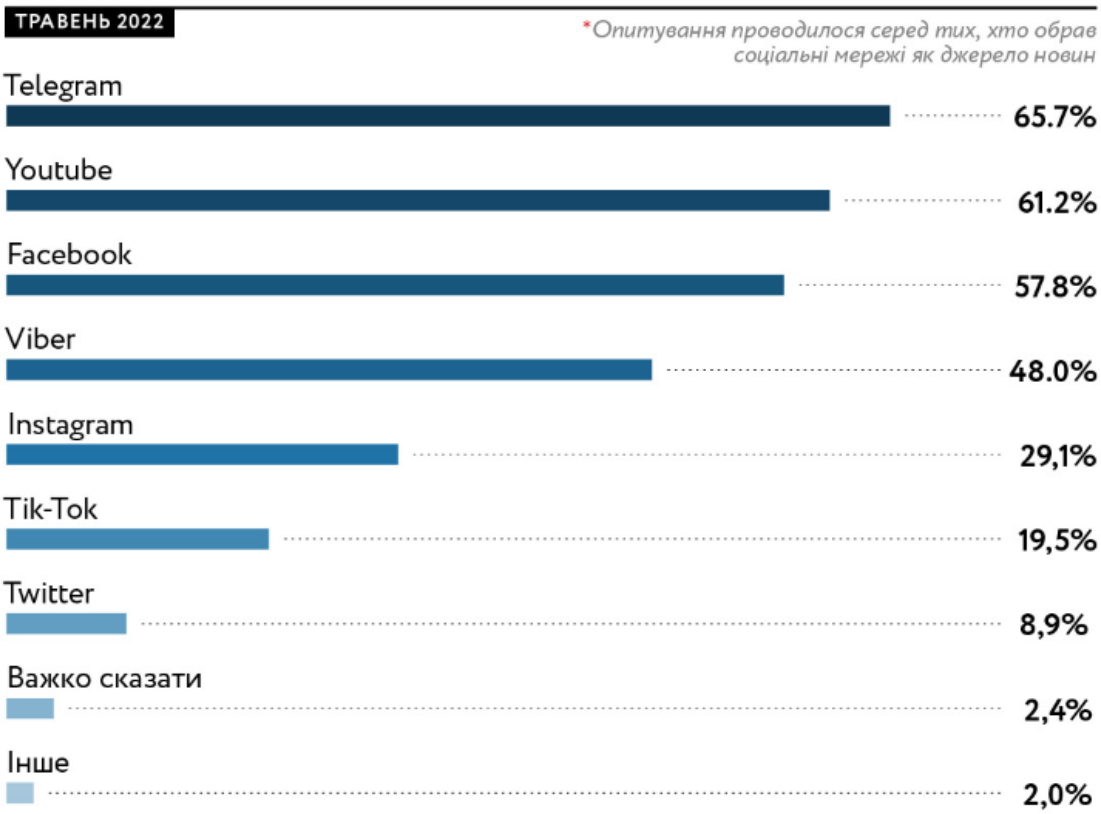


Рис. 1.7. Найбільш використовувані в якості джерела інформації соцмережі влітку 2022 року.

Розбіжності в споживанні новин між чоловіками та жінками через різні соціальні мережі також варто відзначити. Так, чоловіки переважно використовують YouTube (67,2%), Telegram (63%) та Facebook (54,8%) для отримання новин, а найменше зацікавлені в TikTok (19,5%) і Twitter (11,8%).

Серед жінок основними джерелами новин також виявились Telegram (68%), Facebook (62,1%) та YouTube (56%), хоча з дещо іншими відсотковими співвідношеннями. Менше уваги жінки приділяють TikTok (19,5%) та Twitter (6,5%) як джерелам новин. Новини через Viber та Instagram отримують відповідно 44,5% та 24,6% чоловіків і 51,1% та 32,9% жінок (рис. 1.8).[14]

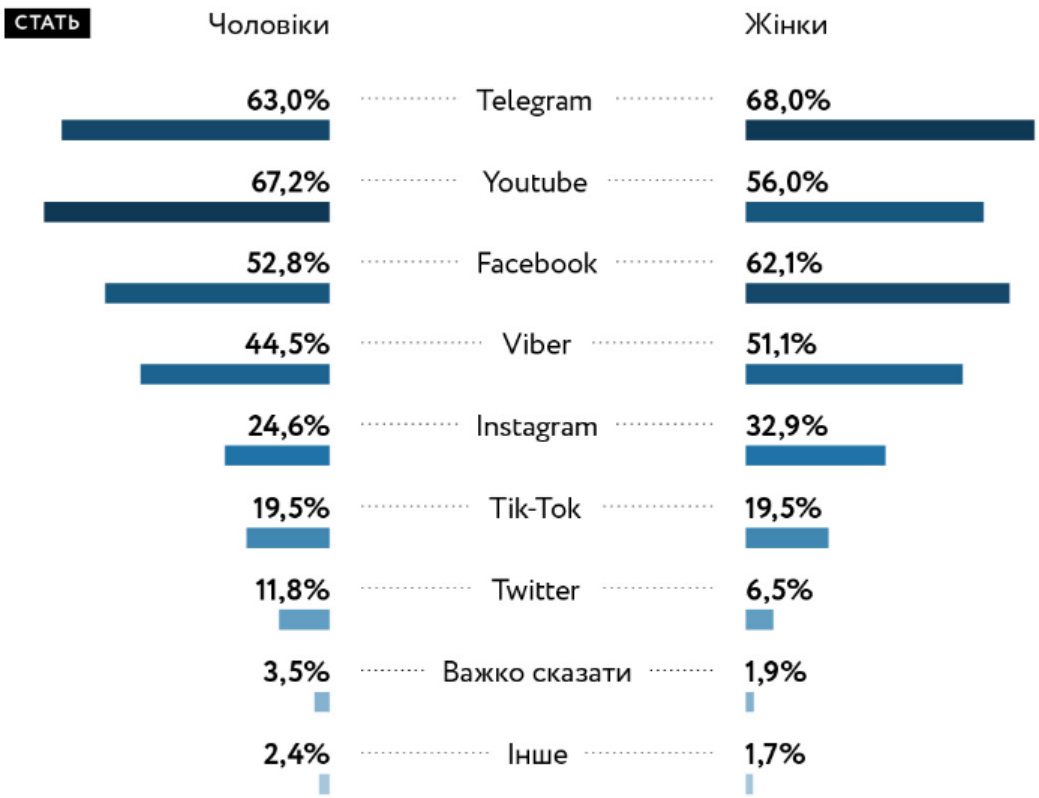


Рис. 1.8. Розподілення використання соціальних мереж за статтю

Також різні соціальні мережі мають різний віковий розподіл аудиторії. Молодь віком від 18 до 39 років в основному отримує новини через Telegram (від 72,4% серед людей віком 30-39 років до 86,7% серед людей віком 18-29 років) та YouTube (від 57,6% серед людей віком 30-39 років до 58,5% серед людей віком 18-29 років). З іншого боку, люди похилого віку (від 60 років) віддають перевагу YouTube (понад 61,4% опитаних) та Facebook (від 43,3% серед людей віком понад 70 років до 59,3% серед людей віком 60-69 років). Viber найчастіше використовують користувачі середнього віку: від 52,1% серед категорії 40-49 років до 59,6% серед людей віком 60-69 років. Серед

цієї вікової категорії також популярний Facebook (від 62,2% серед людей віком 40-49 років до 59,3% серед людей віком 60-69 років). Twitter є більш популярним серед молоді - максимальний відсоток від 10,7% до 10,9% спостерігається серед людей віком 18-39 років. З іншого боку, TikTok знайшов своє застосування не тільки серед 18-29-річних, ним користуються 24,2% людей віком 18-29 років та 23,4% людей віком 60-69 років (рис. 1.9).[14]

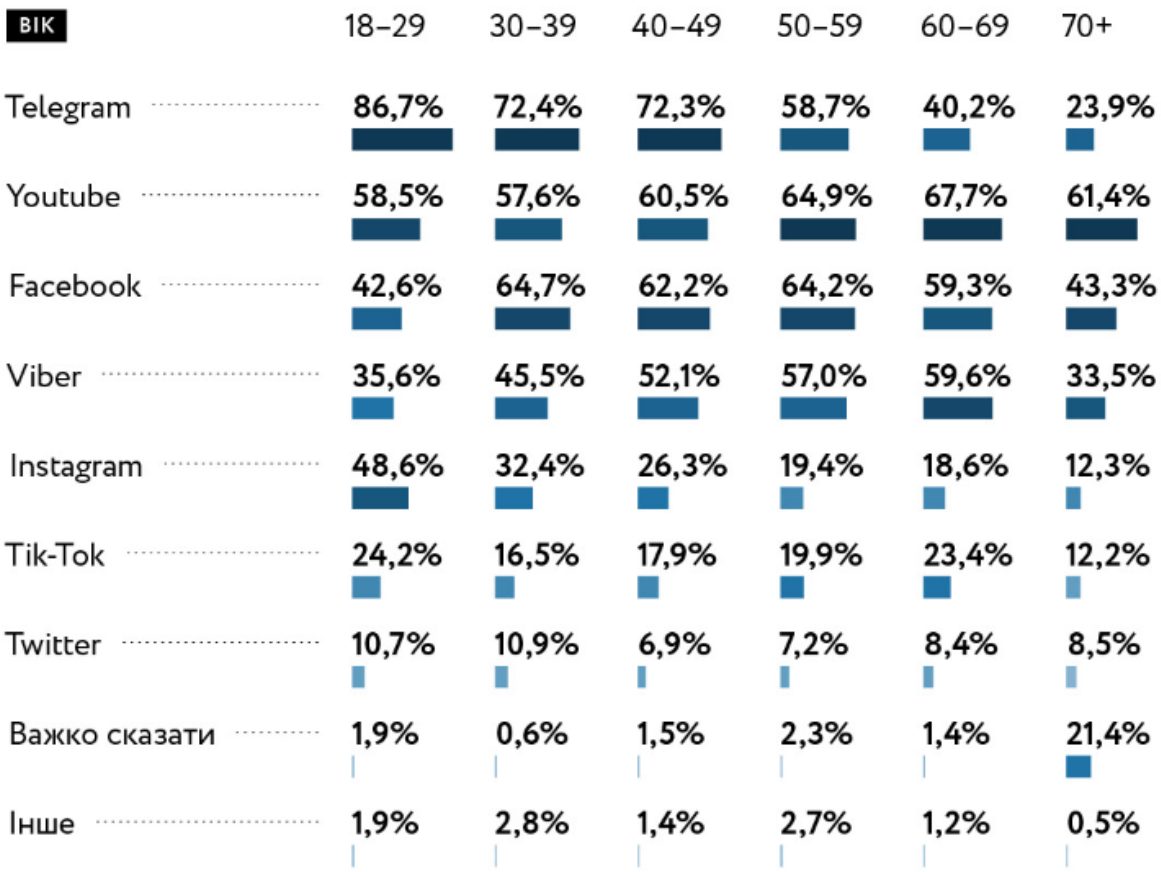


Рис. 1.9. Розподілення використання соціальних мереж за віковою категорією населення

Відмінності в користуванні різними соціальними мережами між мешканцями сільської місцевості та містянами, хоч і невеликі, але відзначаються. У сільських районах більша частина користувачів отримує новини через Facebook (62%, порівняно з 55,9% у місті) та TikTok (24,3%, у

порівнянні з 17,3%). З іншого боку, у місті Telegram (69,6%, у порівнянні з 57,1% в сільській місцевості), YouTube (66%, у порівнянні з 50,7%) та Viber (50,9%, порівняно з 41,8%) є більш популярними. Щодо Instagram та Twitter, через ці соціальні мережі отримують новини приблизно однакова кількість опитаних (приблизно 30% та 9% відповідно) (рис. 1.10).[14]

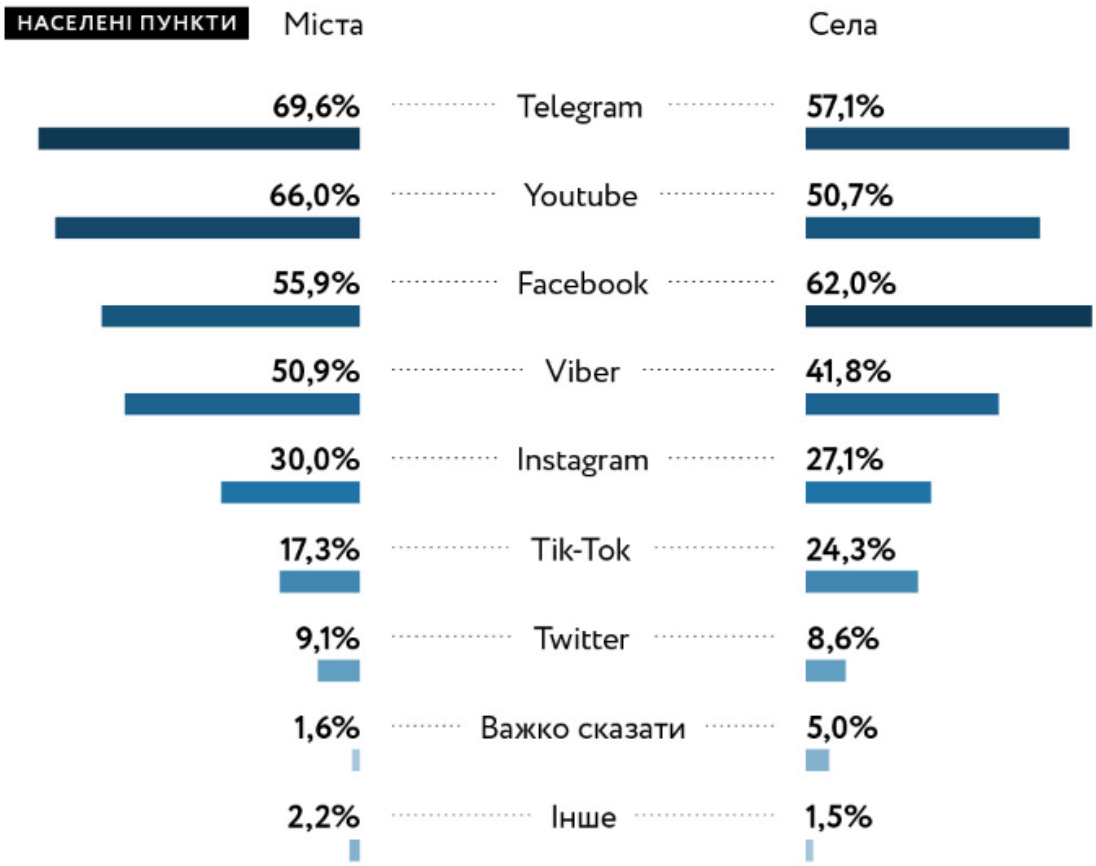


Рис. 1.10. Розподілення використання соціальних мереж за типом населеного пункту

Цікавим аспектом є регіональні відмінності. Зокрема, на Сході України люди значно частіше використовують Telegram як джерело новин (71% людей порівняно з 61,1% у західних регіонах), Viber (54,3% проти 43% у західній Україні), YouTube (62,9% проти 59,3% на Заході) та TikTok (23,4% проти 18,4% у центральній Україні). Facebook найбільш популярний для отримання новин саме у західних регіонах (63% проти 50,6% на півдні України). Серед користувачів Instagram, які використовують його як джерело

новин, найбільше знаходиться у південних регіонах України (31,5%), а Twitter найпоширеніший у центральних регіонах (10,6%) (рис 1.11).[14]

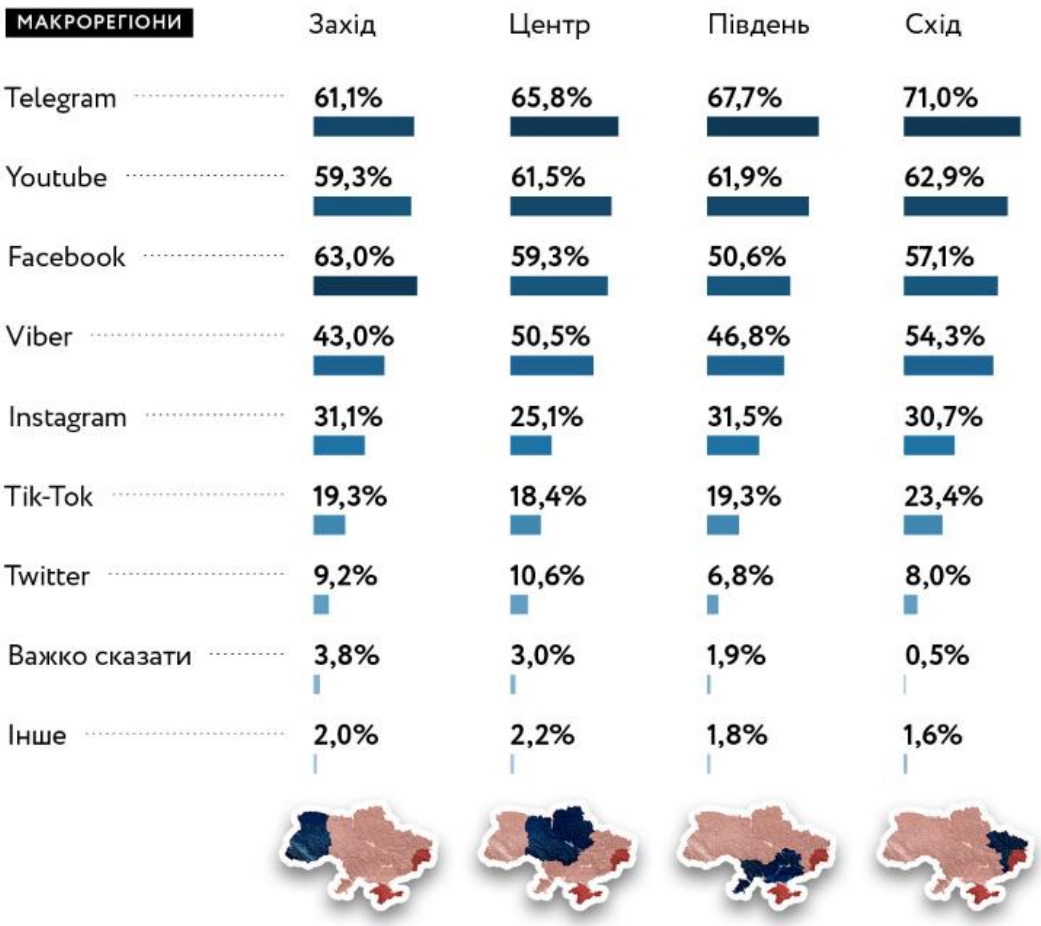


Рис. 1.11. Розподілення використання соціальних мереж за макрорегіонами

Якщо порівнювати за категоріями обсягу користувачів та легкості розробки, то найкращим варіантом є мессенджер Telegram.

Також варто зазначити, що у період з 2021 по 2022 рік аудиторія Telegram в Україні зросла з 20% до 65%. Такий приріст свідчить про те, що саме цей мессенджер є найбільш комфортним для отримання інформації населенням.

Одним з рішень проблеми оповіщення людей про повітряну небезпеку є Telegram канали. Вони охоплюють велику аудиторію, і потребують лише невеликий обсяг автоматизації або пильного адміністратора каналу. Проте, з

цим пов’язана одна проблема, яку мають і інші засоби масової інформації, а саме – відсутність персоналізації даних. Такі засоби можуть повідомити населення про тривогу, що вирішує проблему інформування населення, але це не допоможе з іншим нюансом. Цей нюанс – інформація про найближчі укриття.

1.3. Аналіз проблеми інформованості населення про укриття

По-перше, потрібно чітко зрозуміти, що таке укриття. За офіційною інформацією держустанов, укриття - це ізольована структура для довготривалого перебування людей у випадку надзвичайної ситуації.

Найпростішими укриттями є підвальні та цокольні приміщення будівель, підземні переходи та паркінги. Вони є відносно безпечними схованками у разі нетривалої загрози. Найкраще для таких цілей підходять ті, що мають декілька виходів (а в ідеалі – хоч один з них має вести за межі будівлі).

У період воєнного часу в якості укриття використовуються станції метро у Києві, Харкові та Дніпрі, а також підземні станції швидкісного трамваю у Кривому Розі. Заходити в підземку для укриття можна цілодобово, включаючи період комендантської години.

У Києві вхід безкоштовний під час повітряної тривоги, але лише через турнікети ручного контролю. Щоб потрапити у вестибюль під час комендантської години, вам слід викликати постового поліцейського, звернувшись за номером телефону, що розміщений біля входу в підземку, або натиснути на дзвінок у вестибюлі біля входу в метро. Правоохоронці організовують прохід в укриття.

У Харкові, Дніпрі та Кривому Розі станції доступні для входу цілодобово.[15]

Станції метро – найочевидніший варіант, куди слід прямувати під час повітряної тривоги. Проте вони мають обмежений об’єм, тому слід розглядати й інші варіанти. Але як можна знайти найближчі укриття?

Щоб знайти найближчі укриття під час повітряної тривоги, слід дотримуватися наступних кроків:

- Долучіться до системи оповіщення: Багато міст та регіонів мають системи електронного оповіщення або мобільних додатків, які надсилають сповіщення про надзвичайні ситуації, включаючи повітряні тривоги.
- Слідкуйте за офіційними джерелами: Використовуйте офіційні веб-сайти чи додатки місцевих влад для отримання інформації про місця укриття. Часто така інформація доступна на сайтах місцевої мерії, цивільного захисту чи інших компетентних організацій.
- Дізнайтеся про місця укриття вперед. Багато міст мають карту укриттів, яка може бути доступна онлайн або у місцевих офісах цивільного захисту. Зазначте місця укриття близько до вашого місця проживання чи роботи.
- Спостерігайте за сигналами: Міста можуть розміщувати сигнали або знаки, що вказують на місця укриття. Ця інформація може бути розташована на вулицях чи в об'єктах громадського призначення.
- Звертайте увагу на місцеве населення: Якщо ви перебуваєте в новому місці, запитайте місцевих мешканців або працівників, де знаходяться найближчі укриття під час повітряної тривоги.

Серед усіх варіантів пошуку, що заздалегідь, що під час тривоги, найбільш комфортним варіантом є Інтернет. Є багато онлайн мап різного ступеню точності. Найбільш достовірними є ресурси, які отримують інформацію безпосередньо від органів місцевої влади, але і вони не точні на 100%, бо проблема полягає ще й у відсутності належної інформаційної стратегії. Багато людей не знають, куди вирушити під час тривоги. Існує невизначеність стосовно шляхів евакуації, наприклад, деякі намагаються пройти через підземний торговий центр у центрі міста, але стикаються із закритими дверима. Інші можуть спробувати проникнути в підвальний магазин, але і ця спроба залишається невдачею. Дуже важливо перевіряти

отримані дані, бо, наприклад, в Києві в середньому після початку тривоги ракети можуть долетіти до міста за 5-6 хвилин, що робіть швидкість реакції та дій дуже важливими. Абсолютно необхідно встигнути дістатися укриття до моменту підльоту ракет, щоб убезпечити себе.[16]

А тепер розглянемо існуючі варіанти для оповіщення про тривоги та пошуку укриттів.

1.4. Аналіз існуючих рішенн

В першу чергу слід зазначити найпоширеніші Інтернет-ресурси, що інформують населення про небезпеку.

1.4.1. Telegram канал «Повітряна Тривога»

Найпершим джерелом інформації про загрозу став Telegram канал «Повітряна Тривога», розроблений українською компанією Ajax Systems, та сповіщує усіх підписників постами на каналі.[17]

Плюси цього рішення:

- Достовірність інформації
- Швидкість реагування
- Співпраця з офіційними джерелами

З мінусів для пересічного користувача варто відокремити є глобальність цього ресурсу. Він попереджує про абсолютно усі тривоги на території України, що не є дуже корисним якщо людина налаштує собі гучні повідомлення від цього каналу, або навпаки, вимкне повідомлення повністю і не одразу дізнається про небезпеку.

1.4.2. MapUkraineAlarm.com

Офіційна карта повітряних тривог України. Розроблена компанією Stfalcon.

Ця мапа є основним джерелом інформації для людей на даний момент. Розробники співпрацюють з головами усіх облдержадміністрацій для найбільш точного та оперативного сповіщення людей про небезпеку.[18]

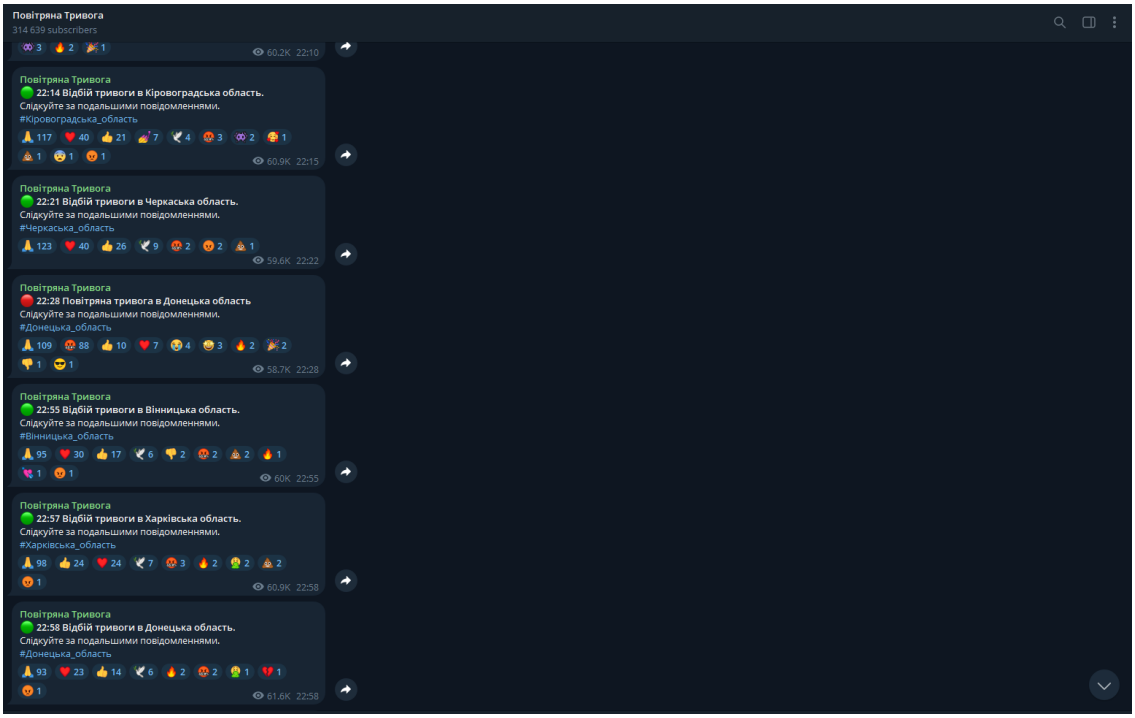


Рис. 1.12. Telegram канал «Повітряна Тривога»

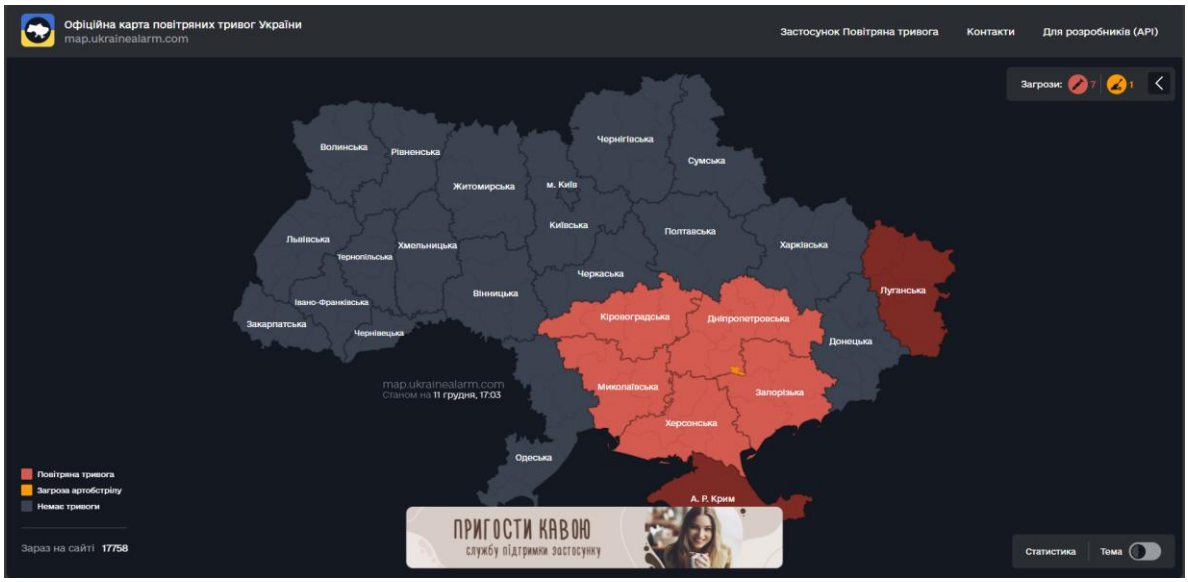


Рис. 1.13. MapUkraineAlarm.com

Переваги цього ресурсу, в цілому, співпадають з попереднім джерелом, проте слід зазначити допомогу розробникам інших ресурсів за допомогою внутрішнього API, для доступу до якого потрібно заповнити заявку на офіційному сайти додатку. Також на сайті є статистика повітряних тривог, але вона не дуже детальна і потрібна лише для інших ресурсів, наприклад, новин, або допитливих громадян.

Також варто зазначити, що розробники цього сайту також розробили окремий застосунок для iOS, Android та Windows під назвою «Повітряна тривога».[19]

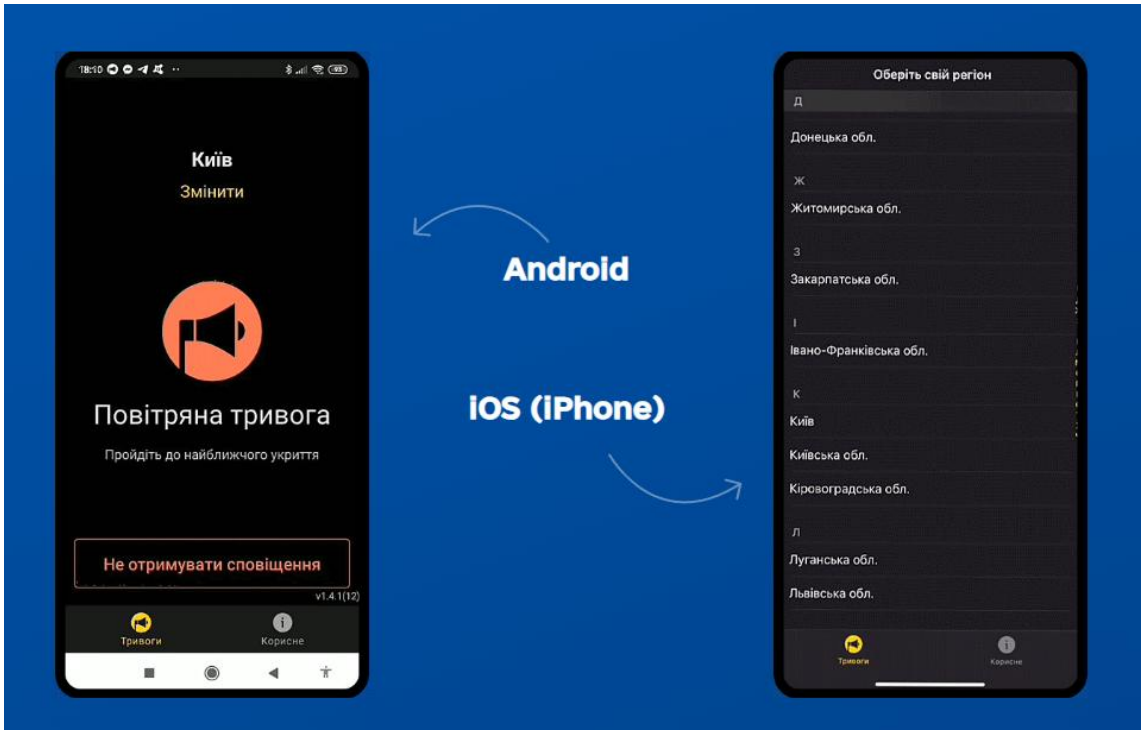


Рис. 1.14. Інтерфейс застосунку «Повітряна тривога» для різних мобільних операційних систем

Цей додаток користується тим же API, що і основна мапа, тому різниці в наданій інформації в цих ресурсів немає.

Недоліком усієї даної системи, як і інших ресурсів Інтернет-ресурсів для оповіщення, є її вузька спеціалізація. Також, сам сайт з мапою не може надсилати повідомлення користувачам, тому потрібно або встановлювати

додаток, або підписуватися на канал в Telegram, або просто постійно слідкувати за мапою.

1.4.3. Alerts.in.ua

Неофіційна мапа тривог, дані отримує не напряму від кожної ОДА, але з їх офіційних Telegram каналів та каналу «Повітряна Тривога», що був згаданий раніше. Сервіс було розроблено волонтерською командою УкрДзен, яка його і підтримує.[20]

Ця система багато в чому нагадує MapUkraineAlarm.com, проте все ж є більш цікавим прикладом. Після ввімкнення додаткових налаштувань на цій мапі відображаються не лише області або райони з тривогою, але й причина тривоги, повідомлення про вибухи, і потенційні загрози, які визначаються штучним інтелектом після перегляду точної інформації про тривоги і напрямок польоту ракет. Ця особливість сайту може допомогти заздалегідь підготуватися до можливої тривоги, проте варто все ж діяти лише за фактом офіційного початку тривоги.

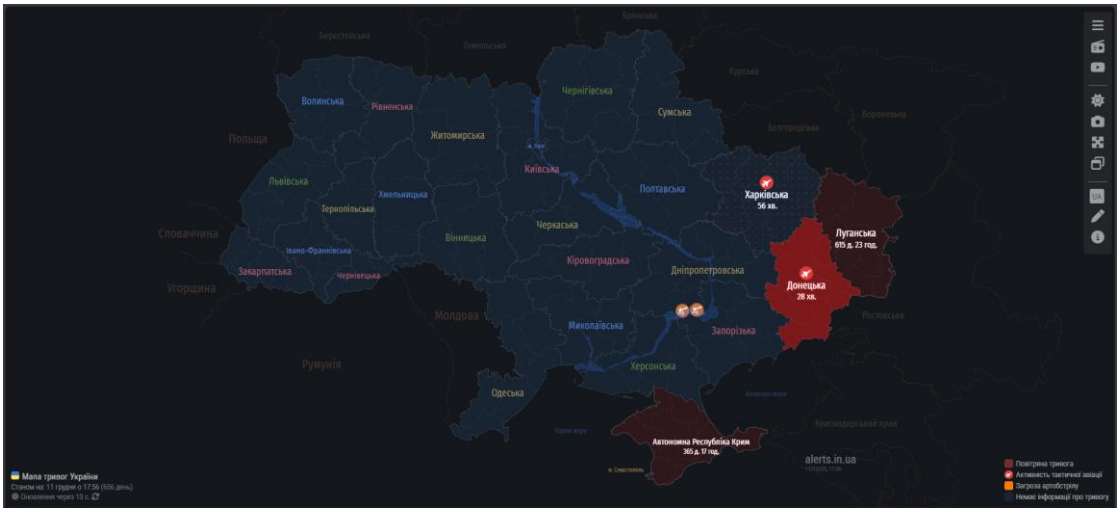


Рис. 1.15. Alerts.in.ua

Мінуси частково повторюють MapUkraineAlarm.com – неможливість надсилати повідомлення користувачам, спрямованість тільки на

інформування про тривоги. Але додатково варто зазначити, що у цього ресурсу немає альтернатив для того, щоб якимось чином отримувати повідомлення, таких як канали у соціальних мережах або офіційні додатки на мобільні платформи.

Тепер варто звернути увагу на основні джерела інформації про укриття. Вони відрізняються від систем оповіщення тим, що створені щоб спрацьовувати лише за запитом користувача.

1.4.4. Telegram бот «Укриття!»

Бот, що розробляється компанією Stfalcon. Він має базу укриттів по всій Україні, і ці дані надаються місцевими органами влади.[21]

Саме цей бот наразі є основним джерелом для простих громадян, і він працює досить ефективно. Якщо людині потрібно знайти укриття, вона може просто відправити відповідну команду боту, після чого користувач отримає запит на його поточну геолокацію. Після цього потрібно відправити місцезнаходження, і у відповідь отримати найближчі укриття в радіусі 5 км з координатами та короткими даними.

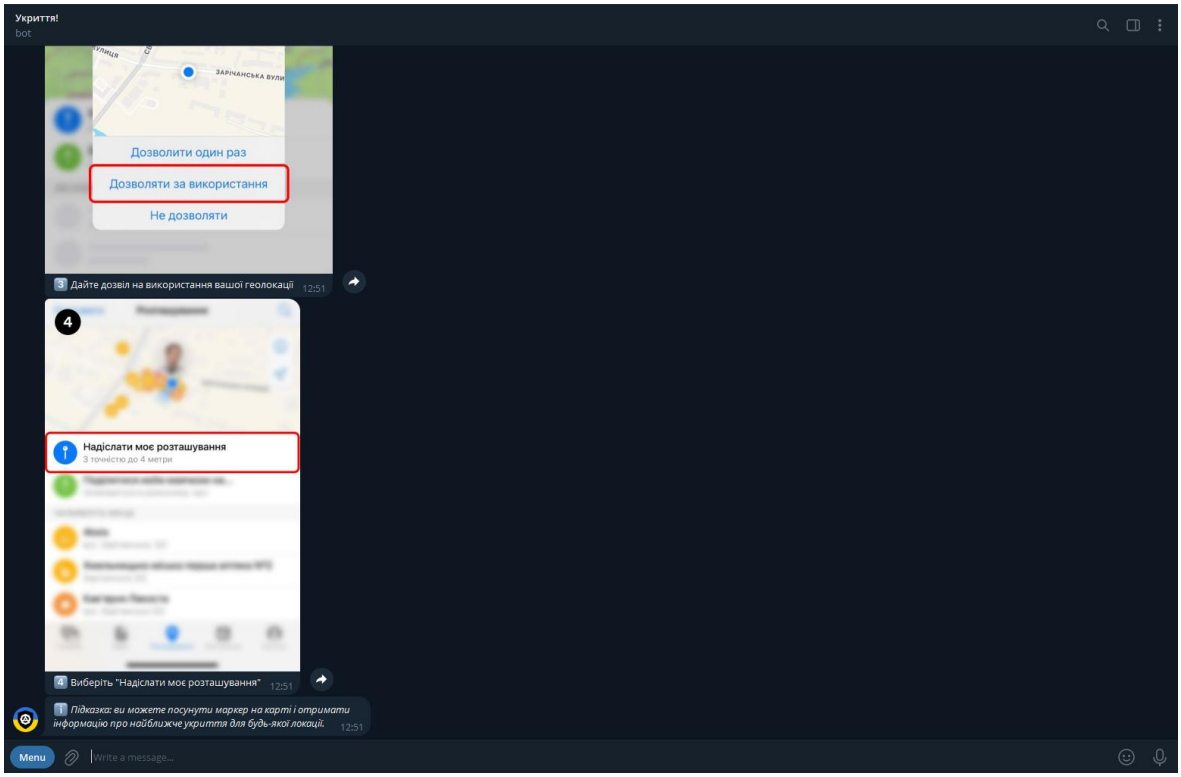


Рис. 1.16. Інструкція з використання від Telegram боту «Укриття!»

Переваги:

- Дані перевірені, ці укриття дійсно існують і вважаються прийнятними для перебування під час тривоги
- Використання максимально примітивне, простий алгоритм дій для отримання бажаної інформації

Недоліки:

- Хоча дані і передані офіційними джерелами, на жаль, 100% вірогідності того, що укриття відкрите чи дійсно підходить для виконання своєї задачі нема. Але це залежить не від розробників, тому недолік швидше відноситься до системи в цілому, а не саме до даного бота
- Бот лише відправляє локацію на мапі, але не формує маршрут до неї, що потребує додаткові дії для отримання саме маршруту

1.4.5. Лун Місто

Карта, розроблена організацією Лун Місто, що опирається на дані, які розробники отримають від користувачів.

Це один з непоганих варіантів для отримання інформації у великих містах України, в якому вказані заклади, що були додані користувачами та перевірені організацією.

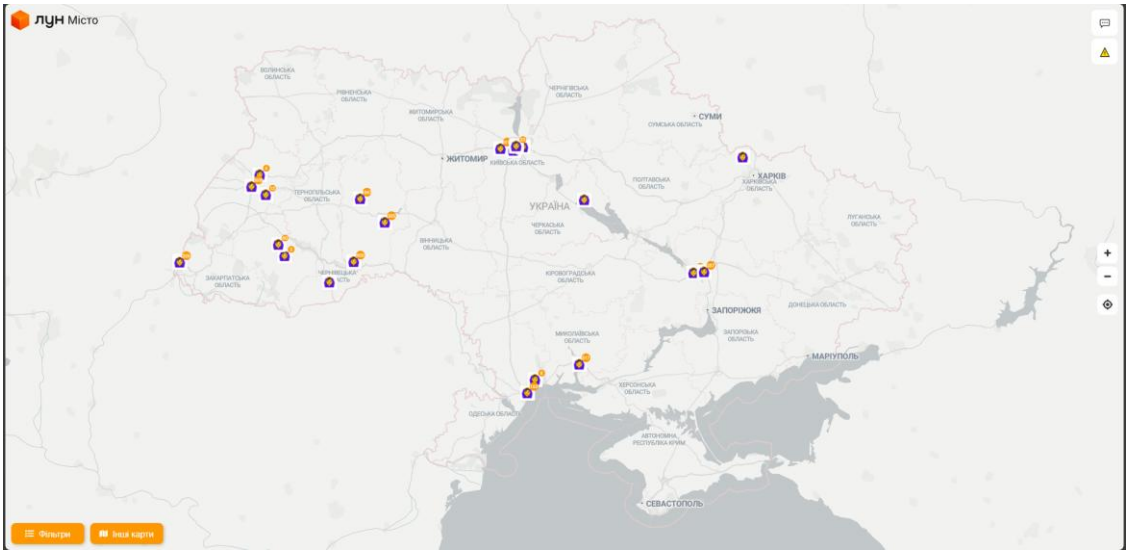


Рис. 1.17. Мапа «Лун Місто»

Переваги над іншими сервісами відокремити важко. Ця система працює лише там, де користувачі додають знайдені ними укриття. Через це мапа не є точною, а також покриває лише малу територію. Ну і вже традиційним в цій роботі недоліком для інтернет ресурсів є неможливість персоналізації повідомлень або пошуків.

Висновки

В цьому розділі було досліджено особливості механізму оповіщення населення, проведено порівняльний аналіз найпоширеніших технологій, що використовуються з цією метою, обрано найбільш ефективну середу для реалізації вирішення цієї проблеми та проаналізовано існуючі системи, що реалізують цей механізм, визначено їх переваги та недоліки.

На основі проведеного в першому розділі дослідження будуть сформовані вимоги для майбутнього застосунку.

РОЗДІЛ 2
МЕТОДИКА РОБОТИ З МАПАМИ ТА ОПИС ВИМОГ ДО ПРОГРАМНОГО ЗАСТОСУНКУ

2.1. Історія цифрових мап та аналіз методів роботи з ними

Історію використання географічної інформації для кращого розуміння та вирішення складних проблем можна простежити аж до 19 століття. У 1832 році французький географ Шарль Піке використовував різні градієнти кольорів на карті Парижа, щоб відобразити кількість смертей від холери, зробивши ранній внесок у розвиток епідеміології. Двадцять два роки потому англійський лікар Джон Сноу пішов далі в цю концепцію та продемонстрував потенціал карт для вирішення проблем, визначивши зв’язок між спалахом холери в Лондоні та забрудненою водою.

Сьогодні географічна інформаційна наука (ГІС) має широке застосування від розробки стратегії ініціатив у сфері охорони здоров’я до вибору місць для археологічних розкопок. Протягом історії ГІС дослідники, програмісти та аналітики продовжували впроваджувати інновації, відкриваючи нові перспективи та технологічний прорив. Результатом їхніх зусиль стали універсальні інструменти та методи, які розширюють можливості проектів для різноманітних організацій.

ГІС виникла як окрема дисципліна в 1960-х роках завдяки піонерським зусиллям Роджера Томлінсона для Департаменту лісового господарства та розвитку сільських районів Канади. Томлінсону було доручено створити перелік природних ресурсів країни на основі карти. У співпраці з програмістами IBM він розробив автоматизовані процеси для зіставлення даних з усіх провінцій Канади.

Перша в світі діюча ГІС, Канадська геоінформаційна система, також дала назву новому напряму. Ця нова технологія стала серйозним кроком

вперед у порівнянні з попередніми підходами до комп’ютеризованого картографування, наприклад:

- Можливість конвертувати дані цифрових сканів у топологічно закодовані формати карт
- Національна система координат
- Окреме зберігання файлів для інформації про атрибути та розташування
- Накладення, операція, яка накладає кілька наборів даних на карту, щоб ідентифікувати їхні зв’язки

Коли команда Томлінсона розробляла CGIS, працювали й інші новатори, наприклад Говард Т. Фішер. У 1964 році Фішер розробив прототип однієї з перших картографічних програм Symap у Північно-Західному університеті. У 1965 році він заснував лабораторію комп’ютерної графіки та просторового аналізу Гарвардської вищої школи дизайну. LCGSA лідирував у розробці методів. для управління, синтезу, аналізу та візуалізації просторової інформації, яка уможливила все більш надійні програми для ГІС.

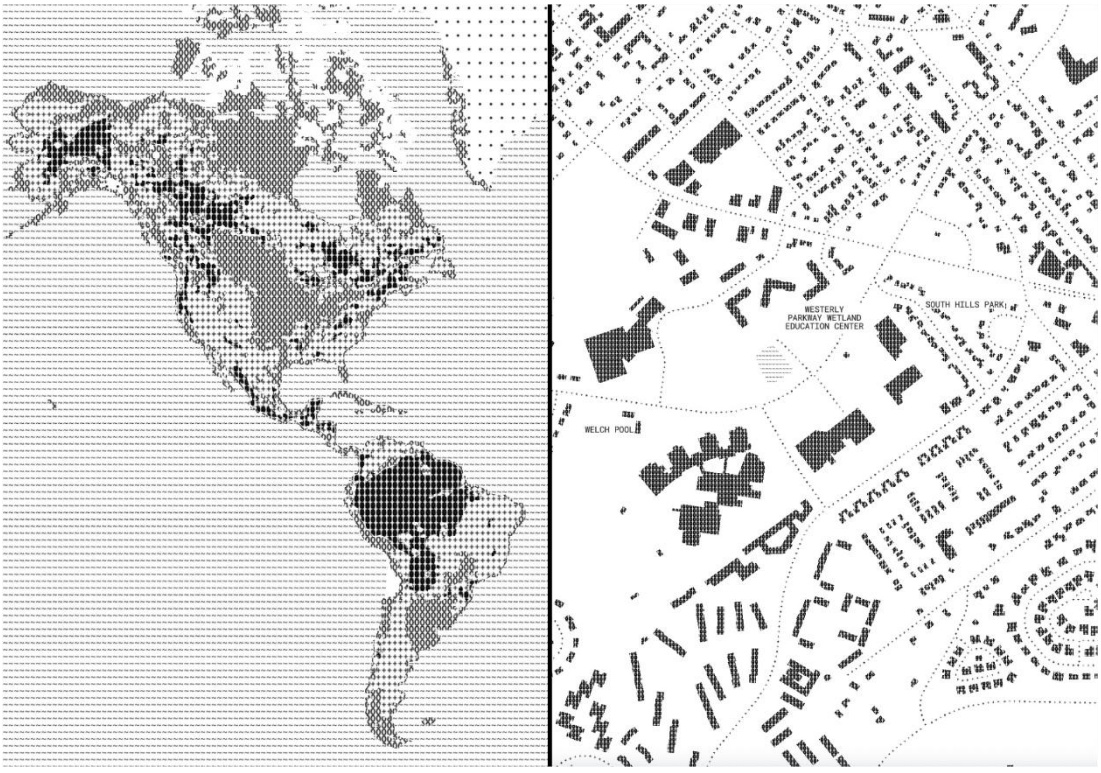


Рис. 2.1. Приклад роботи системи SYMAP

Протягом десятиліть після винаходу ГІС програмісти та дослідники розширили межі збору геопросторових даних і вирішення проблем. Удосконалення цифрових інструментів для збору, візуалізації та аналізу просторової інформації відкрило нові можливості для організацій як у державному, так і в приватному секторах.

Esri – це одна з компаній, яка відіграла ключову роль у цих змінах з моменту свого заснування в 1969 році як Інституту досліджень екологічних систем. Фірма Редлендс, штат Каліфорнія, спочатку спеціалізувалася на використанні ГІС для надання консультаційних послуг щодо планування землекористування та управління ресурсами.

Після багатьох років удосконалення цих методів компанія Esri представила першу комерційно доступну геоінформаційну систему ARC/INFO у 1981 році. Цей програмний пакет надав користувачам базу даних і набір інструментів для введення, обробки та виведення просторової інформації. Створення на основі цієї початкової платформи закріпило місце Esri як найбільшої у світі фірми ГІС, навіть коли на ринку з'явилася низка альтернатив.[22]

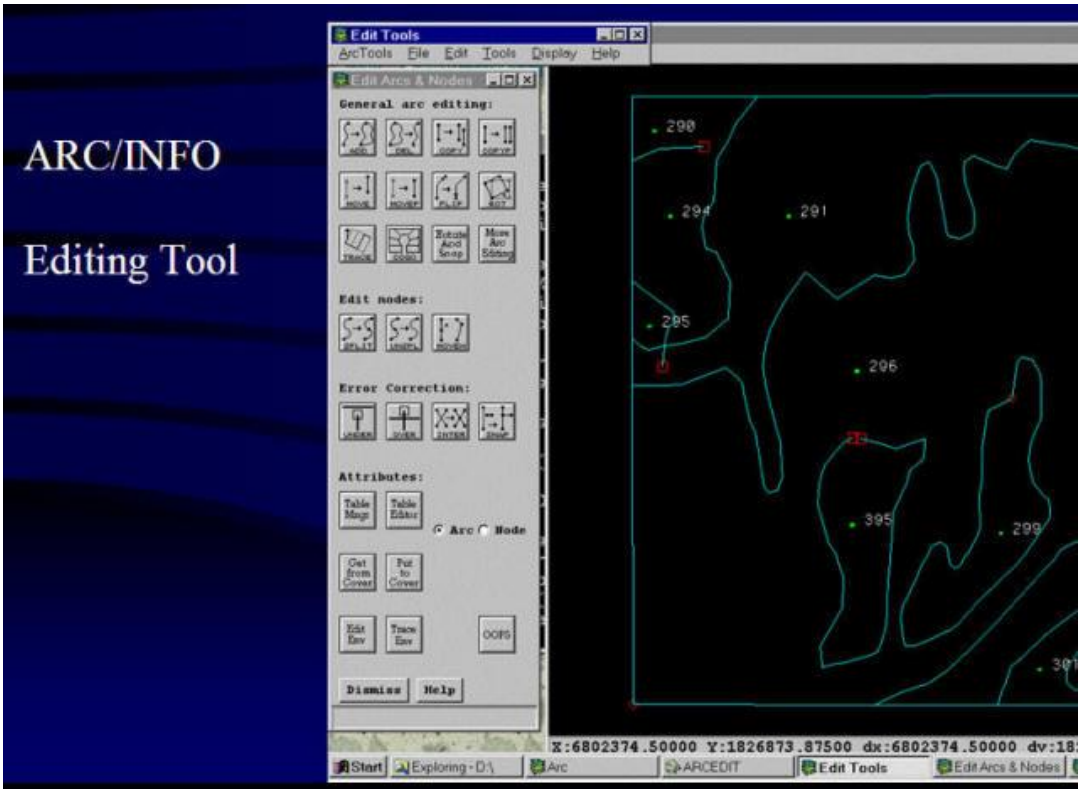


Рис. 2.2. Одна з перших версій ARC/INFO

З появою Інтернету в 1990-х роках ГІС-технології стали доступнішими. Дебют онлайн-картографічних платформ, таких як Google Maps, привернув увагу до ролі легкодоступних карт у повсякденному житті. У 1993 році компанія Xerox PARC представила перший веб-сервіс картографування Xerox PARC Map Viewer, що стало важливою віхою в історії ГІС.

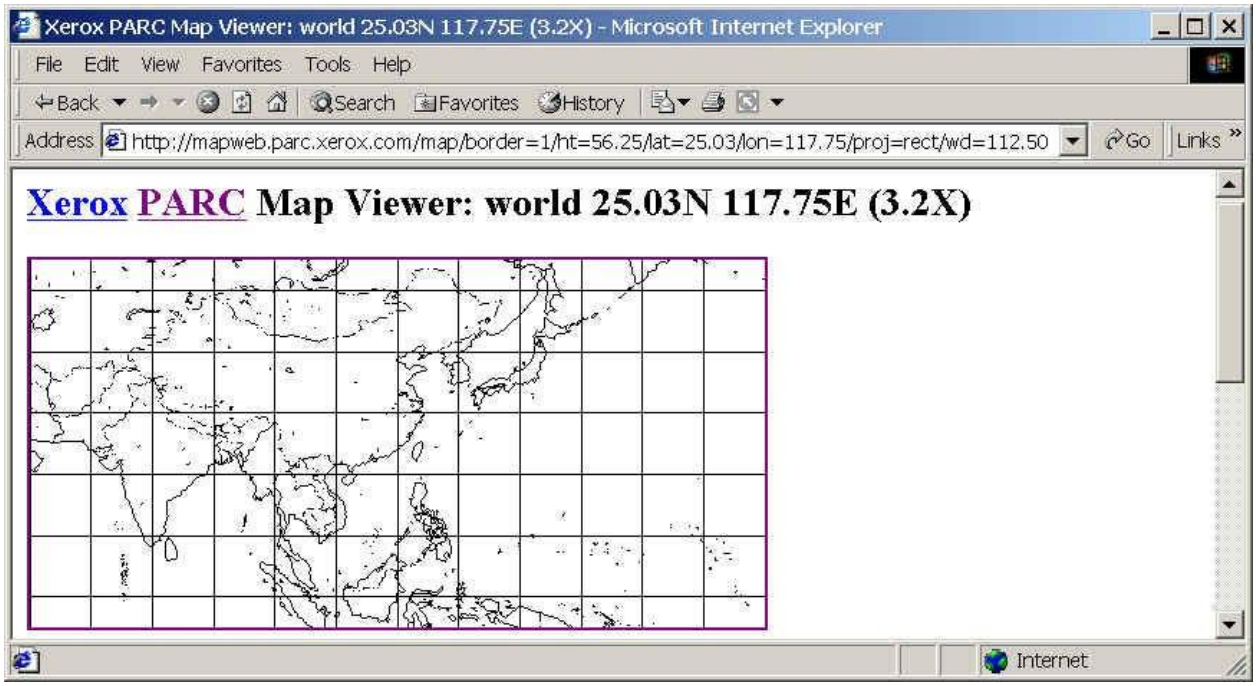


Рис. 2.3. Приклад роботи Xerox PARC Map Viewer

Еволюція ГІС триває, оскільки геопросторові системи трансформувалися разом зі збільшенням потужності комп'ютерів і зростанням ємності зберігання даних. Зараз інструменти та методи постійно вдосконалюються, щоб служити таким цілям, як міське планування, реагування на стихійні лиха, дослідження ринку, управління ресурсами та військові операції. Дистанційне зондування дозволяє збирати географічні дані з віддалених районів світу і навіть з космосу.

Надійні ГІС-інструменти дозволяють організаціям упорядковувати та обмінюватися складною інформацією так, щоб користувачі її легко зрозуміли. Виробники та роздрібні торговці можуть використовувати свої висновки для створення більш ефективних ланцюжків поставок, тоді як

департаменти охорони здоров’я можуть приймати більш обґрунтовані рішення щодо розподілу обмежених ресурсів. За допомогою картографування з відкритим вихідним кодом і веб-інструментів будь-хто може взяти участь у створенні даних про моделі дорожнього руху або шляхи небезпечного шторму.

Коли ми вступили в 21-е століття, ГІС перетворилася на далекосяжну технологію, яка перетинає межі академічних і наукових досліджень. Удосконалені інструменти, такі як Google Earth і Google Maps, змінили нашу взаємодію з геопросторовими даними, зробивши просторове розуміння частиною нашого повсякденного життя. Інтеграція ГІС із мобільними технологіями стала ще одним проривом, який уможливив відстеження та навігацію в реальному часі.[23]

Креативні люди, які вирішують проблеми, продовжуватимуть розширювати межі геопросторового мислення в наступні роки. Деякі з найбільш перспективних напрямків для майбутнього розвитку ГІС включають:

- Використання 3D ГІС для створення детальніших візуалізацій
- Програми доповненої реальності на основі визначення місця розташування
- Створення навігаційних систем для безпілотних автомобілів
- Картографування внутрішнього середовища

Тепер варто проаналізувати основні цифрові картографічні системи, що використовуються на даний момент та визначити, які з них будуть більш ефективними та зручними у використанні в процесі реалізації програмного застосунку.

2.1.1. Google Maps

Google Maps, широко використовувана веб-картографічна служба, створена компанією Google, пропонує користувачам безліч функцій, пов’язаних із картами, навігацією та службами визначення

місцезнаходження. Можна досліджувати інтерактивні карти, і користувачі можуть шукати адреси, підприємства та визначні місця, а також отримувати інформацію про дорожній рух у режимі реального часу для напрямків проїзду. Перегляд вулиць, включена функція, дозволяє користувачам віртуально досліджувати вулиці на рівні землі з оглядом на 360 градусів. Додаткові функції включають інформацію про громадський транспорт, пішохідні та велосипедні маршрути та офлайн-доступ до карти. Спрощено обмін інформацією про місцезнаходження в реальному часі, і користувачі можуть активно брати участь у програмі «Місцеві експерти», надаючи відгуки, фотографії та інформацію про місця. Розробники можуть легко інтегрувати картографічні функції у свої програми за допомогою платформи Google Maps Platform, яка містить API для карт, маршрутів і місць. Постійно оновлюючись новими функціями, Карти Google залишаються універсальною та потужною картографічною платформою, яка широко використовується для навігації, дослідження та отримання інформації на основі місцезнаходження окремими особами та компаніями.

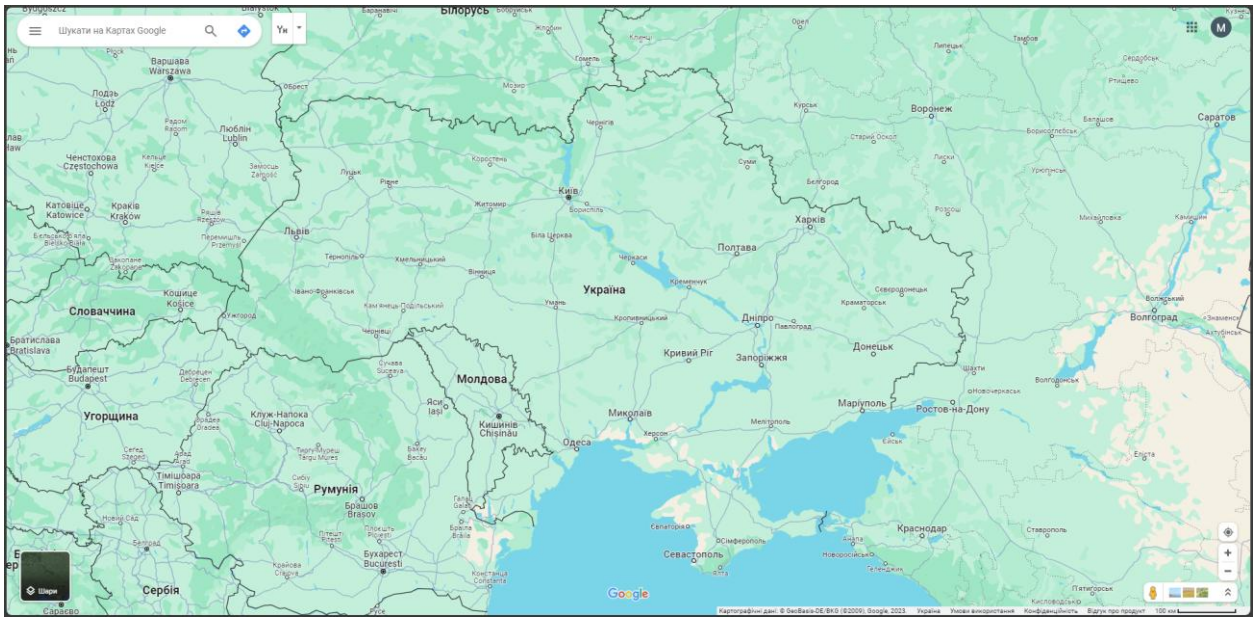


Рис. 2.4. Україна на мапі Google Maps

Проблемами Google Maps є необхідність у реєстрації в сервісі, отриманні особистого API ключа та підключенні особистих платіжних

засобів до сервісу, об’єми даних, що можна отримувати безкоштовно досить обмежені.

Основною перевагою є власне компанія, яка стоїть за цією системою. Google має дуже великі ресурси, що дозволяють перевіряти усю інформацію на картах та вносити нові дані.

2.1.2. OpenStreetMap (OSM)

OpenStreetMap (OSM), заснована в 2004 році, є спільною ініціативою, яка надає користувачам у всьому світі географічні дані та карти, які можна редагувати, у вільному доступі. OSM дозволяє учасникам постійно вдосконалювати карти, додаючи такі деталі, як дороги, стежки, кафе та залізничні станції. Різноманітне співтовариство картографів підтримує проект, надаючи дані, виправляючи помилки та вдосконалюючи тонкощі карти.

Основні особливості та характеристики:

- Модель відкритих даних: OSM використовує модель відкритих даних, що дозволяє необмежений доступ, перегляд і завантаження картографічних даних як для особистого, так і для комерційного використання. Ця характеристика відрізняє OSM.
- Спільне картографування: OSM спирається на модель участі спільноти, де окремі особи вносять дані через опитування, відстеження супутникових зображень і редагування. Ці колективні зусилля забезпечують постійну та детальну карту.
- Розширюваність: можливість адаптації моделі даних OSM враховує широкий спектр географічних об’єктів, дозволяючи учасникам відображати різноманітні елементи, такі як інфраструктура, природні об’єкти та зручності.
- Система тегів: OSM використовує систему тегів для характеристики об’єктів карти. Теги, що вказують такі атрибути, як ім’я, тип і адреса, надають контекст картографованим об’єктам.

- Глобальне покриття: OSM забезпечує глобальне покриття, виходячи за рамки окремих регіонів або країн. Цей глобальний охоплення робить OSM цінним ресурсом для картографування різноманітних міських і сільських територій по всьому світу.
- Настроювані карти: розробники можуть створювати персоналізовані карти, використовуючи дані OSM, налаштовуючи візуальне представлення та стиль відповідно до своїх конкретних вимог.
- Спільнота та події: спільнота OSM організовує такі заходи, як мапатони та конференції, сприяючи співпраці та обміну знаннями між учасниками. Природа OSM, керована спільнотою, заохочує дискусії та обмін найкращими практиками.
- Різні випадки використання: OSM знаходить застосування в різних контекстах, включаючи навігацію, географічний аналіз, реагування на стихійні лиха, міське планування та активний відпочинок. Це універсальна та адаптована картографічна платформа.[24]

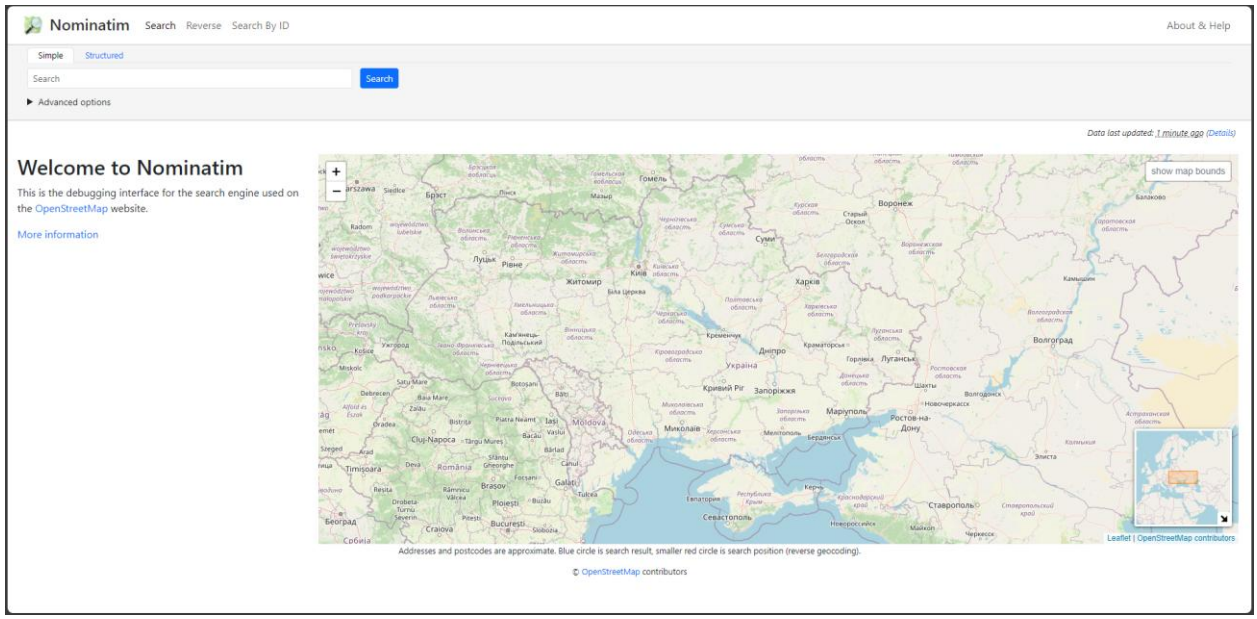


Рис. 2.5. Nominatim – пошукова система, розроблена для сервісу OpenStreetMap

Для отримання даних з бази даних OpenStreetMap розробниками було створено Overpass API. Цей сервіс працює з власною мовою запитів, та

повертає у відповіді на запит дані в обраному форматі. Система підтримує багато загальновживаних форматів, такі як JSON та XML.

Overpass API підтримує отримання трьох основних типів елементів із бази даних OpenStreetMap: вузли (окремі точки з географічними координатами), шляхи (упорядковані списки вузлів, що представляють лінійні об’єкти, як-от дороги), і зв’язки (упорядковані чи невпорядковані групи вузлів, шляхів, або відносини).

Окрім того, в інфраструктуру OpenStreetMap входить Overpass Turbo — веб-інтерфейс, який спрощує процес створення та тестування запитів Overpass API. Це дозволяє користувачам інтерактивно створювати та запускати запити, візуалізувати результати на карті та уточнювати запити в режимі реального часу.[25]

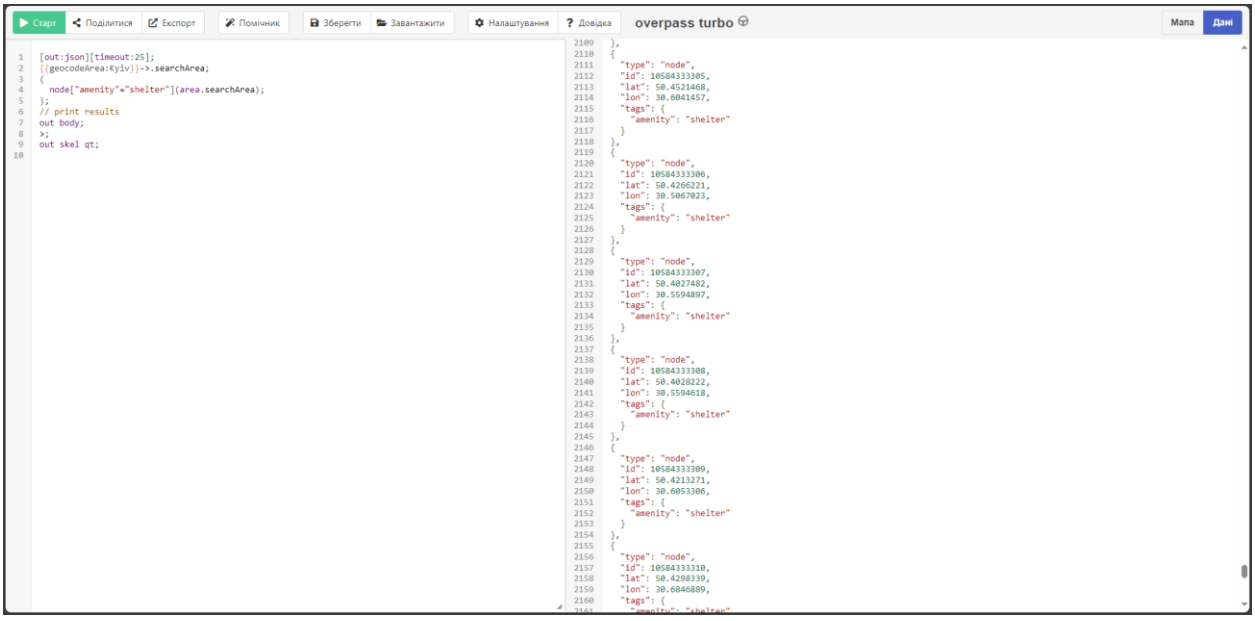


Рис. 2.6. Приклад пошуку укриття в Києві за допомогою Overpass Turbo з використанням мови запитів

Окрім Overpass API OpenStreetMap використовують ще два сервіси – Mapbox та Leaflet.

Mapbox пропонує картографічні та локаційні послуги, включаючи настроювані карти та навігацію. Розробники можуть використовувати

Mapbox API для інтеграції карт у свої додатки з параметрами налаштування та стилю.

Leaflet — це бібліотека JavaScript з відкритим кодом для інтерактивних карт. Він легкий і простий у використанні, що робить його популярним вибором для розробників, які хочуть просте, але потужне рішення для картографування.

2.1.3. Bing Maps API

Bing Maps API пропонує набір геопросторових служб і інструментів із платформи Bing Maps від Microsoft. Розробники можуть легко інтегрувати ці служби у свої додатки для картографування, геокодування, маршрутизації та інших функцій на основі місцезнаходження.

API надає елемент керування картою, який полегшує вбудовування динамічних та інтерактивних карт у веб-програми та програми для мобільних пристроїв, підтримуючи такі функції, як налаштування стилів карти, масштабування та панорамування.

Послуги геокодування дозволяють перетворювати назви місць або адреси в географічні координати та навпаки, допомагаючи в картографуванні та додатках на основі розташування.

Служби маршрутизації допомагають розраховувати маршрути між місцями, враховуючи такі фактори, як стан дорожнього руху в реальному часі. Розробники можуть реалізувати покрокові маршрути та оцінити час у дорозі.

Служби просторових даних пропонують такі функції, як пошук найближчих визначних місць (POI), пошук підприємств і отримання детальної інформації про конкретні місця.

Служби Imagery Services надають доступ до різних стилів і шарів карт, включаючи супутникові зображення, покращуючи візуальне представлення карт у програмах.

API карт Bing включає API часового поясу, що дозволяє розробникам отримувати інформацію про часові пояси для певних місць, що є цінним для планування та завдань, чутливих до часу.

Autosuggest API покращує взаємодію з користувачем, надаючи пропозиції в реальному часі щодо місць і адрес під час операцій пошуку та геокодування.

Можливості Spatial Analytics дають змогу розробникам виконувати геопросторові обчислення, аналізувати шаблони та проводити аналіз наближення, додаючи рівень просторового інтелекту до програм.

API карт Bing легко інтегрується зі службами Microsoft Azure, надаючи додаткові хмарні функції для масштабованості, безпеки та зберігання.

Для джерел даних API карт Bing переважно покладається на картографічні дані Microsoft, які включають внески від різних партнерів і джерел. Ці дані регулярно оновлюються та покращуються для забезпечення точності та актуальності.

З точки зору ліцензування та ціноутворення, Bing Maps API працює на основі моделі передплати, і розробники повинні отримати ключі API. Витрати можуть бути пов’язані з обсягом використання, а докладну інформацію про ціни можна отримати в Microsoft.

Корпорація Microsoft надає вичерпну документацію, включаючи посібники, навчальні посібники та довідкові матеріали, щоб допомогти розробникам ефективно інтегрувати та використовувати API карт Bing для різноманітних геопросторових програм.

Недоліки цього API в більшості співпадають з недоліками Google Maps, але варто додати меншу точність інформації, бо дані оновлюються не так якісно.

Зважаючи на вільний доступ до даних, простоту інтегрування системи до застосунку та сумісність з обраними технологіями та мовою програмування, Overpass API є найбільш комфортним сервісом для імплементації рішення проблеми, що досліджується в цій роботі.

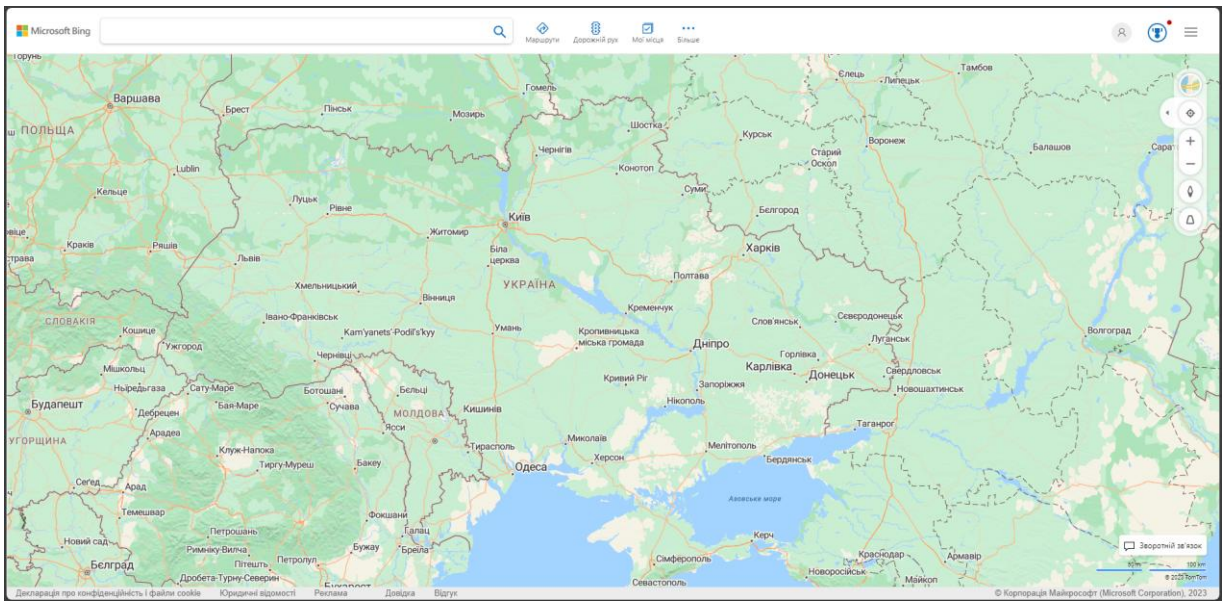


Рис. 2.7. Microsoft Bing Map

2.2. Аналіз алгоритмів формування маршруту між точками на мапі

Алгоритми пошуку шляху вирішують проблему пошуку шляху від джерела до пункту призначення, уникаючи перешкод і мінімізуючи витрати (час, відстань, ризики, паливо, ціну тощо).

2.2.1. Алгоритм Дейкстри

Алгоритм Дейкстри, розроблений голландським комп’ютерним науковцем Едсгером Дейкстрою в 1959 році [26], є алгоритмом пошуку графа, який розв’язує проблему найкоротшого шляху з одним джерелом для графа з невід’ємною вартістю шляху ребра, створюючи дерево найкоротших шляхів. Цей алгоритм часто використовується при маршрутизації. Еквівалентний алгоритм був розроблений Едвардом Ф. Муром у 1957 році.[27]

Для даної вихідної вершини (вузла) на графі алгоритм знаходить шлях із найменшою вартістю (тобто найкоротший шлях) між цією вершиною та будь-якою іншою вершиною. Він також може бути використаний для знаходження вартості найкоротших шляхів від однієї вершини до однієї вершини призначення шляхом зупинки алгоритму після визначення

найкоротшого шляху до вершини призначення. Наприклад, якщо вершини графіка представляють міста, а витрати на краї шляху представляють відстані між парами міст, з'єднаних прямою дорогою, алгоритм Дейкстри можна використовувати для пошуку найкоротшого маршруту між одним містом та всіма іншими містами. Як наслідок, найкоротший шлях спочатку широко використовується в мережевих протоколах маршрутизації, особливо IS-IS і OSPF (Open Shortest Path First).[28]

Назвемо вузол, з якого ми починаємо, початковим вузлом. Нехай відстань вузла Y буде відстанню від початкового вузла до нього. Алгоритм Дейкстри призначить деякі початкові значення відстані та намагатиметься покращити їх крок за кроком.

- Призначте кожному вузлу значення відстані. Встановіть його на нуль для нашого початкового вузла та на нескінченність для всіх інших вузлів.
- Позначте всі вузли як невідвідані. Встановіть початковий вузол як поточний.
- Для поточного вузла розгляньте всіх його невідвіданих сусідів і обчисліть їх відстань (від початкового вузла). Наприклад, якщо поточний вузол (A) має відстань 6, а ребро, що з'єднує його з іншим вузлом (B), дорівнює 2, відстань до B через A буде 6+2=8. Якщо ця відстань менша за попередньо записану відстань (нескінченність на початку, нуль для початкового вузла), перезапишіть відстань.
- Коли ми закінчимо розглядати всіх сусідів поточного вузла, позначте його як відвіданий. Відвіданий вузол більше ніколи не перевірятиметься; його відстань, зареєстрована зараз, є остаточною та мінімальною.
- Встановіть невідвіданий вузол із найменшою відстанню (від початкового вузла) як наступний «поточний вузол» і продовжуйте з кроку 3.[28]

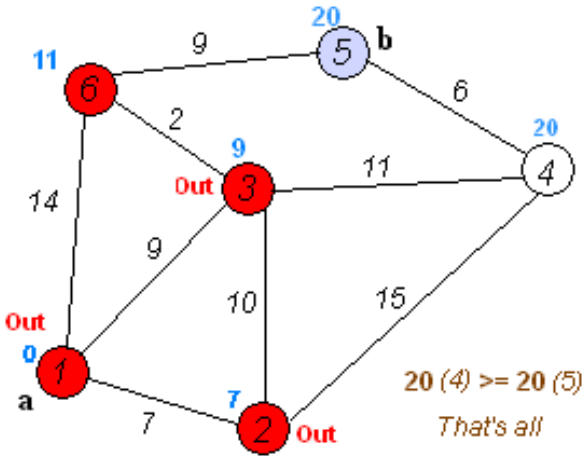


Рис. 2.8. Приклад роботи алгоритма Дейкстри на графі

Лістинг 2.1

Псевдокод алгоритму Дейкстри

```
Dijkstra(Graph, source, destination):
  initialize distances
  initialize previous
  initialize priority_queue
  initialize visited
  while the top of priority_queue is not destination:
      get the current_node from priority_queue
      for neighbor in current_node's neighbors and not in visited:
          if update its neighbor's distances:
              previous[neighbor] = current_node
      save current_node into visited
  extract path from previous
  return path and distance
```

Алгоритм Дейкстри має різні застосування в реальному часі в різних областях:

- Системи GPS-навігації. Алгоритм Дейкстри зазвичай використовується в системах GPS-навігації для пошуку найкоротшого шляху між джерелом і пунктом призначення, що дозволяє користувачам знаходити оптимальні маршрути для автомобіля, пішки або громадського транспорту.
- Планування маршруту авіакомпанії або транспорту: алгоритм Дейкстри можна застосувати для визначення найкоротших шляхів і відстаней між аеропортами або містами, допомагаючи

авіакомпаніям або транспортним компаніям оптимізувати свої маршрути та розклади.

- Управління ланцюгом поставок: алгоритм Дейкстри може допомогти оптимізувати маршрути доставки в управлінні ланцюгом поставок, забезпечуючи ефективне транспортування товарів від постачальників до клієнтів.[29]

Цей алгоритм – один з основних, що використовується у Google Maps.[30]

2.2.2. A* (A star) алгоритм

A* (A-star) - це алгоритм пошуку шляху в графах, який комбінує в собі властивості алгоритму Дейкстри та евристичний підхід для оптимізації пошуку. Його основна мета - знайти найоптимальніший шлях від початкового вузла до кінцевого.

На відміну від алгоритму Дейкстри, де вага ребра визначається тільки реальною відстанню між вузлами, A* використовує додаткову евристичну функцію, яка надає приблизне значення відстані від поточного вузла до кінцевого вузла. Ця евристична оцінка дозволяє алгоритму враховувати інформацію про віддаленість між поточним і кінцевим вузлами, що допомагає ефективно вибирати напрямок пошуку.[28]

Алгоритм працює наступним чином:

- Починаємо з початкового вузла і встановлюємо його відстань від початку рівною 0, а також розраховуємо евристичну оцінку від поточного вузла до кінцевого.
- Розглядаємо всі можливі наступні вузли, обчислюємо вагу шляху від початку до кожного з них, враховуючи реальну відстань і евристичну оцінку.
- Обираємо вузол з мінімальною загальною оцінкою і розглядаємо його сусідів.

- Повторюємо кроки 2-3 до тих пір, поки не досягнемо кінцевого вузла або не вичерпаємо всі можливі варіанти.
- Завершуємо алгоритм, коли досягаємо кінцевого вузла і маємо знайдений найоптимальніший шлях.

Таким чином, A* використовує евристику для вибору наступного вузла для розгляду, оптимізуючи процес пошуку найкоротшого шляху в графі.

A* (A-star) є одним із найефективніших алгоритмів пошуку шляху, і його ефективність базується на експлуатації якісної інформації про шлях. Він знаходить оптимальний шлях, враховуючи як реальні відстані між вузлами, так і евристичні оцінки відстаней до кінцевого пункту.

Одна з ключових особливостей A* - це використання функції оцінки f(n) для кожного вузла n, яка враховує і реальну відстань від початкового вузла до n (g(n)), і евристичну оцінку відстані від n до кінцевого вузла (h(n)). Таким чином, f(n) = g(n) + h(n). Вибираючи вузли для розгляду за значенням f(n), алгоритм уникає безплідних гілок у пошуку та прискорює збіжність до оптимального шляху.

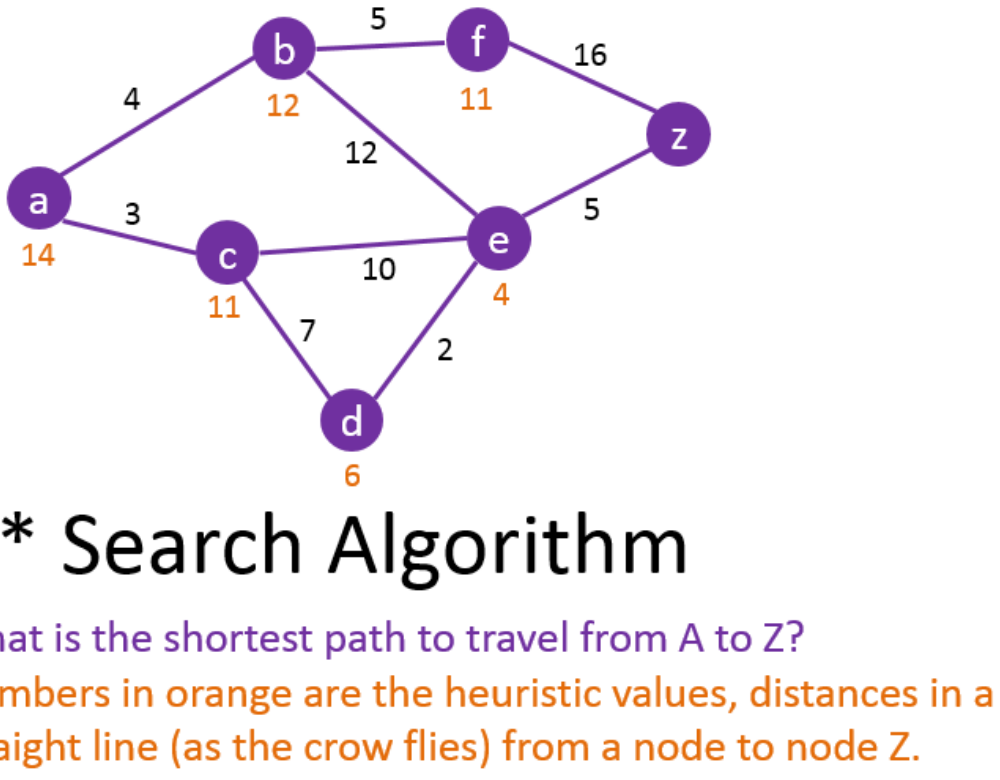


Рис. 2.9. Приклад роботи A* алгоритму

Евристична функція h(n) повинна бути допустимою, тобто вона не може переоцінювати відстань до кінцевого вузла. Це забезпечує те, що A* завжди знаходить оптимальний шлях.

Евристична функція, в контексті A*-алгоритму, грає ключову роль у визначенні того, які вузли слід розглядати в першу чергу під час пошуку найкоротшого шляху в графі. Ця функція оцінює апріорну інформацію про відстань від поточного вузла до кінцевого вузла і використовується для вибору оптимального напрямку пошуку.

Основні вимоги до евристичної функції (часто позначається як "h(n)"):

- допустимість (admissibility): Функція не повинна переоцінювати відстань між двома вузлами. Тобто, h(n) повинна бути меншою або рівною реальній відстані від вузла n до кінцевого вузла. Це забезпечує оптимальність алгоритму.
- монотонність (consistency або triangle inequality): Для будь-яких трьох вузлів A, B і C, відстань від A до B, плюс відстань від B до C, повинна бути завжди більшою або рівною відстані від A до C. Формально виражається як $h(A) \leq c(A, B) + h(B)$, де c(A, B) - вага ребра між вузлами A і B. Це гарантує, що евристична функція не буде вводити "фальшиві" покращення шляху.

Приклади евристичних функцій можуть включати евклідову відстань у двовимірному просторі, мангаттенську (місто) відстань або інші, залежно від природи проблеми.

Важливо вибирати евристичну функцію, яка відображає особливості конкретного завдання, щоб забезпечити ефективність A*-алгоритму та отримання точних та швидких результатів.

Існує декілька найбільш використовуваних у A* алгоритмі евристичних функцій.

Мангаттенська відстань (також відома як метрика міста, відстань місто-квартал або L1-норма) - це метрика в просторі, яка вимірює відстань між двома точками, обчислюючи суму абсолютних різниць їхніх координат.

Формула для обчислення мангаттенської відстані між двома точками A(x_1 , y_1) і B(x_2 , y_2) у двовимірному просторі виглядає наступним чином:

$$D(A,B) = |x_2 - x_1| + |y_2 - y_1| \quad (2.1)$$

Ця формула відображає суму горизонтальної і вертикальної відстаней між двома точками. Вона визначає "мангаттенський шлях", який можна пройти по решітці міста, де вам дозволено рухатися тільки вгору, вниз, вліво та вправо, але не по діагоналі.

Мангаттенська відстань часто використовується в алгоритмах пошуку шляху, таких як A*-алгоритм, коли обчислення відстані між точками повинно враховувати тільки вертикальні та горизонтальні рухи, а не діагональні. Ця метрика є ефективною в обчисленнях та дозволяє просто моделювати рух у міському середовищі, де можна пересуватися лише вздовж вулиць.

Евклідова відстань (Euclidean Heuristic): Вимірює пряму відстань між поточним і кінцевим вузлами у просторі. Формула:

$$h(n) = \sqrt{(x_{goal} - x_n)^2 + (y_{goal} - y_n)^2} \quad (2.2)$$

Чебишева відстань (Chebyshev Heuristic): Максимальна з абсолютних різниць по координатах. Це відстань "через клітинку".

$$h(n) = \max\left(|x_{goal} - x_n|, |y_{goal} - y_n|\right) \quad (2.3)$$

У деяких випадках може бути доцільно використовувати специфічні евристичні функції, залежно від особливостей конкретного випадку. Наприклад, в графічних додатках може використовуватися евристична функція, яка базується на складності пересування в конкретних областях зображення.

Деякі задачі можуть вимагати врахування додаткових факторів або вагованих коефіцієнтів в евристичній функції, щоб підкреслити або зменшити важливість певних аспектів відстані.

Обираючи евристичну функцію, важливо враховувати природу задачі та особливості середовища. Оптимальний вибір евристичної функції може значно покращити ефективність A* алгоритму.

Один із важливих випадків використання A* - це в сфері штучного інтелекту для пошуку шляхів в графах, що моделюють великі системи, такі як мережі доріг, графічні об'єкти або рухомі об'єкти в середовищі. Завдяки своїй ефективності та можливості знаходження оптимальних шляхів, A* став популярним в ігровій індустрії, робототехніці та інших областях, де важливо знаходження шляхів в графах.

Лістинг 2.2

Псевдокод A* алгоритму

```
A* Algorithm(Graph, start, goal):
    Create an open list and a closed list
    Add the start node to the open list
    while the open list is not empty:
        CurrentNode = node in the open list with the lowest cost (f = g + h)
        Move CurrentNode from the open list to the closed list
        if CurrentNode is the goal:
            Construct the path from start to goal
            return the path
        for each neighbor of CurrentNode:
            if neighbor is in the closed list:
                continue
            calculate the tentative g score:
            tentative_g = CurrentNode.g + distance(CurrentNode, neighbor)
            if neighbor is not in the open list or tentative_g < neighbor.g:
                set neighbor.g = tentative_g
                set neighbor.h = heuristic distance from neighbor to goal
                set neighbor.f = neighbor.g + neighbor.h
                set neighbor.parent = CurrentNode
                if neighbor is not in the open list:
                    add neighbor to the open list
    return failure (no path found)
```

У цьому псевдокоді g являє собою акумульовану вартість шляху від початкової вершини до поточної вершини, h - евристичну оцінку вартості шляху від поточної вершини до кінцевої вершини, а f - сумарну оцінку (f = g

+ h). Алгоритм продовжує виконання, доки відкритий список не спорожніє або не буде знайдено шлях до цільової вершини.

2.2.3. Алгоритм Беллмана-Форда

Алгоритм Беллмана-Форда - це алгоритм для знаходження найкоротших шляхів в графі, який може працювати з вагованими графами, де ребра можуть мати від'ємні ваги. Його основна ідея полягає в тому, щоб поступово оцінювати відстані від початкового вузла до всіх інших вузлів, оновлюючи ці оцінки в кожному кроці, поки можливо поліпшення.

Процес алгоритму можна уявити так: починаючи з початкового вузла, алгоритм спробує покращити оцінки відстаней до всіх сусідніх вузлів. Це повторюється для кожного вузла в графі. Алгоритм продовжує виконуватися, доки відбуваються оновлення відстаней, інакше він завершується.[31]

Сам процес можна поділити на наступні етапи:

- Ініціалізація: всі відстані від початкового вузла до інших вузлів встановлюються на нескінченність, крім відстані до самого себе, яка встановлюється на 0. Також створюється масив для зберігання попередніх вузлів на шляху до кожного вузла.
- Ітерації: виконується $V - 1$ ітерацій, V – кількість вузлів в графі. Кожна ітерація спробує покращити оцінки відстаней.
- Оновлення відстаней: для кожного ребра (u, v) у графі перевіряється, чи можливо скоротити відстань до вузла v через вузол u. Умова для оновлення відстані:

$$d[v] > d[u] + w(u, v) \tag{2.4}$$

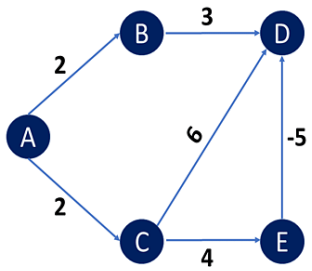
де d – масив відстаней, $w(u, v)$ – вага ребра (u, v). Якщо така умова виконується, відстань до v оновлюється:

$$(2.5)$$

$$d[v] = d[u] + w(u, v)$$

і зберігається попередній вузол $p[v]$, який лежить на найкоротшому шляху до v.

- Перевірка від'ємних циклів: Після виконання всіх ітерацій алгоритм може виявити наявність від'ємних циклів. Для цього виконується ще одна ітерація. Якщо під час цієї ітерації відбувається оновлення відстаней, то у графі є від'ємний цикл.
- Вивід результатів: Якщо граф не має від'ємних циклів, алгоритм повертає знайдені найкоротші шляхи від початкового вузла до всіх інших. Якщо граф має від'ємний цикл, алгоритм може вивести повідомлення про це або іншим способом інформувати про наявність проблем.



	B	C	D	E
0	∞	∞	∞	∞
0	2	2	∞	∞
0	2	2	2	6
0	2	2	3	6
0	2	2	3	6

Рис. 2.10. Приклад роботи алгоритму Беллмана-Форда

Алгоритм Беллмана-Форда може застосовуватися в графах з ваговими ребрами, що відкриває можливість роботи з різноманітними задачами, такими як пошук найкоротших шляхів у мережах та транспортних системах.

Хоча алгоритм Беллмана-Форда не завжди є найефективнішим для всіх випадків, його важливість полягає в тому, що він працює з графами з

від'ємними вагами і може виявляти від'ємні цикли. Це може бути важливим, оскільки від'ємні цикли можуть призводити до нескінченного зменшення відстаней, і алгоритм виявить цю ситуацію.

Лістинг 2.3

Псевдокод алгоритму Беллмана-Форда

```
function bellmanFordAlgorithm(G, s)
  for each vertex V in G
    dist[V] <- infinite // dist is distance
      prev[V] <- NULL // prev is previous
  dist[s] <- 0
  for each vertex V in G
    for each edge (u,v) in G
      temporaryDist <- dist[u] + edgeweight(u, v)
      if temporaryDist < dist[v]
        dist[v] <- temporaryDist
        prev[v] <- u
  for each edge (U,V) in G
    If dist[U] + edgeweight(U, V) < dist[V}
      Error: Negative Cycle Exists
  return dist[], previ[]
```

2.2.4. Алгоритм пошуку в глибину

Алгоритм пошуку в глибину (DFS), це метод обходу графа або дерева, при якому всі глибини графа досліджуються перед тим, як переходити до наступного рівня. Зазвичай алгоритм "поглиблюється" якнайдалі в глибину графа перед тим, як вибрати інший шлях.

Основна ідея полягає в тому, що алгоритм рекурсивно або за допомогою стеку просувається вглиб графа. Коли вузол відвідується, його помічають як відвіданий, і рекурсивно або через стек переходять до всіх його сусідів. Якщо сусід ще не відвіданий, алгоритм продовжує глибше; якщо всі сусіди вже відвідані, алгоритм повертається в попередній вузол і продовжує пошук іншим шляхом.

Алгоритм пошуку в глибину застосовується для знаходження компонент зв'язності в графі, виявлення циклів, топологічного сортування та інших задач, пов'язаних з обходом графів чи дерев. Щоб почати, вибирається початковий вузол, його помічають як відвіданий, і обхід розпочинається з

цього вузла. Після відвідання вузла, рекурсивно (через функцію або стек) відбувається перехід до всіх його ще не відвіданих сусідів, кожен вузол позначається як відвіданий. Це важливо, щоб уникнути зациклювання у графі. Обхід сусідів виконується рекурсивно або за допомогою стеку. Рекурсивний метод використовує властивість стека, де останній відвіданий вузол стає першим, який повертається. Це призводить до глибокого спуску вниз графа перед тим, як рухатися вгору. Коли всі сусіди вузла вже відвідані, алгоритм повертається вверх по стеку або викликає стек рекурсивних викликів, знаходячись на попередньому рівні. Цей процес повторюється для інших не відвіданих вузлів, доки не буде відвідано всі вузли у графі.[32]

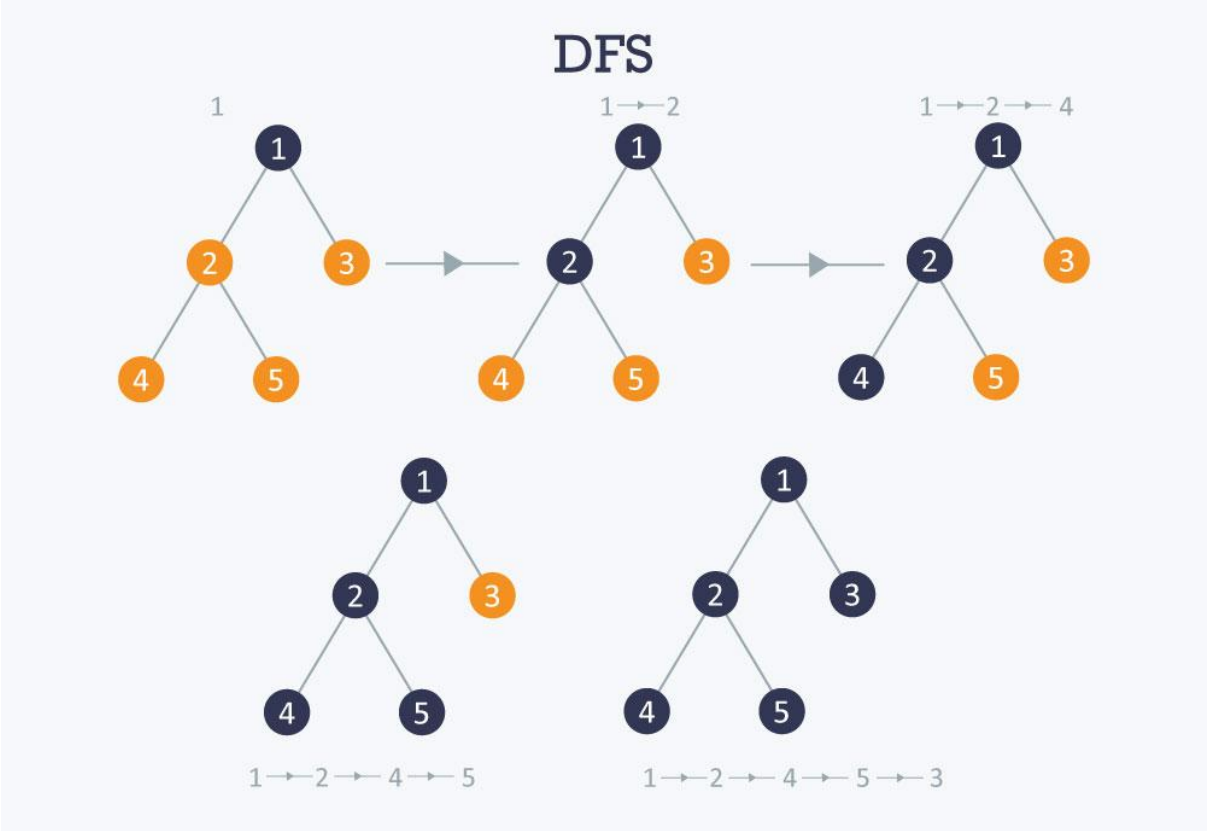


Рис. 2.11. Приклад роботи алгоритму пошуку в глибину

Ітеративний варіант алгоритму пошуку в глибину використовує цикл та стек для відстеження вузлів, що дозволяє обійти граф без використання

рекурсії. Основна ідея полягає в тому, щоб вручну керувати стеком вузлів для обходу графа у глиб.

Нижче наведений загальний алгоритм ітеративного DFS:

- Створити порожній стек для відстеження вузлів.
- Помістити початковий вузол до стеку.
- Поки стек не порожній:
- Взяти вузол з верху стеку.
- Якщо він не відвіданий:
- Відмітити його як відвіданий.
- Додати всі його невідвідані сусіди до стеку.
- Коли стек порожній, алгоритм завершено.

Ітеративний підхід дає можливість більш гнучкого керування процесом обходу графа, особливо коли вам потрібно докладніше визначити порядок обходу сусідніх вузлів. Він також може бути корисний у випадках, коли рекурсія може викликати переповнення стеку через велику глибину графа.

Лістинг 2.4

Псевдокод алгоритму пошуку в глибину (ітеративний варіант)

```
procedure DFS_iterative(graph, start_node):
    stack = empty stack
    mark start_node as visited
    push start_node onto stack
    while stack is not empty:
        current_node = pop from stack
        for each neighbor in neighbors of current_node:
            if neighbor is not visited:
                mark neighbor as visited
                push neighbor onto stack
```

У рекурсивному варіанті алгоритму використовується рекурсивна функція, яка викликає себе для кожного невідвіданого сусіда поточного вузла. Перед тим, як викликати себе, поточний вузол помічається як відвіданий і, за необхідністю, над цим вузлом виконуються потрібні дії.

Рекурсивний варіант використовує стек викликів для відстеження вузлів, зручний у реалізації, зокрема для простих графів чи дерев, але може

призводити до переповнення стеку при роботі з великими графами або глибокими рекурсивними викликами, тому важливо враховувати обмеження на глибину рекурсії.

Лістинг 2.5

Псевдокод алгоритму пошуку в глибину (рекурсивний варіант)

```
procedure DFS_recursive(graph, node):
    if node is not visited:
        mark node as visited
        for each neighbor in neighbors of node:
            DFS_recursive(graph, neighbor)
```

Обидва варіанти еквівалентні в своїй логіці, і обираються в залежності від конкретної задачі та особливостей графа.

Алгоритм пошуку в глибину (DFS) знаходить широке застосування в області комп'ютерних наук і інших дисциплінах.

При обході графів і дерев він потрібен для таких операцій:

- Визначення компонент зв'язності графа.
- Знаходження та аналіз циклів у графах.
- Топологічне сортування графа.

Для задачі знаходження шляхів цей алгоритм застосовується в пошуку шляхів у графі між двома вузлами та знаходженні всіх можливих шляхів у графі.

Окрім того, в нього є і графічні застосування: в області картографії і маршрутизації алгоритм DFS може використовуватися для пошуку оптимального маршруту на мапі. Також додатки, що стосуються шляхопроводження, можуть використовувати DFS для визначення оптимального шляху між двома точками. Проте слід зазначити, що цей алгоритм може бути застосований в основному простих випадках, DFS не завжди гарантує знаходження найкоротшого шляху.

Штучний інтелект використовує даний алгоритм в рішенні головоломок або задач на логічне мислення.

2.2.5. Алгоритм пошуку в ширину

Алгоритм пошуку в ширину (BFS) представляє собою спосіб обходу графа або дерева, при якому спочатку розглядаються всі вузли на одному рівні перед тим, як переходити до наступного рівня. Початковий вузол служить початком процесу, який розгалужується широко, включаючи всіх його найближчих сусідів, а потім переходить до наступного рівня вузлів.

Для відстеження порядку обходу вузлів алгоритм використовує чергу. Вузли додаються в чергу для подальшого відвідування, і потім виймаються з черги в тому порядку, в якому були додані. Цей процес триває доти, поки всі вузли не будуть відвідані або досягнута мета.

Основна концепція алгоритму полягає в тому, щоб спершу розглядати вузли на тому самому рівні перед переходом до наступного, що дозволяє ефективно знаходити найкоротші шляхи в незважених графах та деревах.

Процес BFS можна уявити як розповсюдження хвилі, що розповсюджується від початкового вузла і охоплює всі його найближчі вузли, а потім переходить на наступний рівень.

BFS часто використовується для пошуку найкоротшого шляху між двома вузлами у незваженому графі. Оскільки алгоритм розглядає вузли на одному рівні перед переходом на наступний рівень, перший знайдений шлях буде найкоротшим. BFS може служити для визначення компонент зв'язності у графі, оброблюючи кожну компоненту окремо.

У випадку графів з орієнтованими ребрами BFS може бути використаний для пошуку вузлів, до яких можна дістатися з даного вузла, або вузлів, які можуть дістатися до даного вузла. Завдяки BFS можна знаходити відстані від початкового вузла до всіх інших вузлів у графі.

Алгоритм також корисний для виявлення циклів у графах. Якщо вже відвіданий вузол потрапляє знову в чергу, то у графі існує цикл. Крім того, BFS може використовуватися для моделювання розповсюдження інформації чи пошуку оптимальних рішень.

Застосування BFS може бути різноманітним залежно від конкретних потреб задачі. Цей алгоритм ефективно використовує структуру даних черги для реалізації порядку обходу вузлів.

У галузі мереж та комунікації цей алгоритм застосовується для знаходження найкоротших шляхів у мережах зв'язку, виявлення груп та підграфів у складних мережах даних та виявлення найменшої кількості переадресацій для досягнення вузла в комп'ютерних мережах.

Для GPS та мап він виконує пошук найкоротших маршрутів між точками на карті та визначення оптимального шляху для автомобілів, пішоходів або інших транспортних засобів.

У сфері комп’ютерних ігор це реалізація логіки пошуку шляхів і моделювання поведінки об'єктів в графічних сценах.

Цей алгоритм також має практичну цінність для соціальних мереж: виявлення ступенів віддаленості між користувачами в соціальних мережах та аналіз графа взаємозв'язків для рекомендацій – результат роботи саме BFS.

Окрім того, він потрібний у біології та медицині. Ось деякі з його застосувань:

- Моделювання розповсюдження інфекцій у популяції.
- Вивчення генетичних взаємозв'язків та еволюції.
- Моделювання поширення захворювань та вакцинаційних кампаній.

У логістиці BFS потрібен для пошуку оптимальних маршрутів для транспорту та маршрутизації вантажів та вантажівок.

Також однією зі сфер використання є виробництво та інженерія, де алгоритм необхідний для планування ліній виробництва та потоку матеріалів і оптимізації виробничих процесів.

Алгоритм виглядає так:

- Вибір початкового вузла: Робота BFS починається з вибору початкового вузла в графі. Це стає точкою виходу для алгоритму.

- Позначення вузлів: Початковий вузол позначається як відвіданий. Це важливо для того, щоб уникнути зациклювання у випадку циклів у графі.
- Використання черги: BFS використовує структуру даних "черга" для відстеження порядку обходу вузлів. Початковий вузол додається до черги.

- Цикл обходу: Алгоритм починає цикл, в якому вузли видаляються з черги один за одним, і їх сусіди, які ще не були відвідані, додаються до черги. Цей процес відбувається для кожного вузла на поточному рівні перед переходом на наступний рівень.
- Помічання відвіданих вузлів: Кожен вузол, який додається до черги, помічається як відвіданий. Це запобігає повторному відвідуванню вузлів та забезпечує ефективність алгоритму.
- Продовження циклу: Цей процес повторюється, доки черга не стане порожньою. Після завершення алгоритм відвідав усі вузли графа в порядку їх віддаленості від початкового вузла.
- Визначення шляхів і відстаней: Під час обходу можна визначити найкоротші шляхи від початкового вузла до всіх інших вузлів. Також можна визначити відстані від початкового вузла до кожного іншого вузла у графі.

Принциповою особливістю BFS є те, що він розглядає всі вузли на одному рівні перед переходом до наступного рівня, що дозволяє ефективно визначати найкоротші шляхи та використовується для різноманітних застосувань у різних областях.[33]

Лістинг 2.6

Псевдокод алгоритму пошуку в ширину

```
BFS(Graph, start):
    Create a queue Q and an empty set visited
    Enqueue start into Q
```

```
    Add start to visited set
    while Q is not empty:
        current = dequeue from Q
        for each neighbor of current:
            if neighbor is not in visited:
                enqueue neighbor into Q
                add neighbor to visited set
```

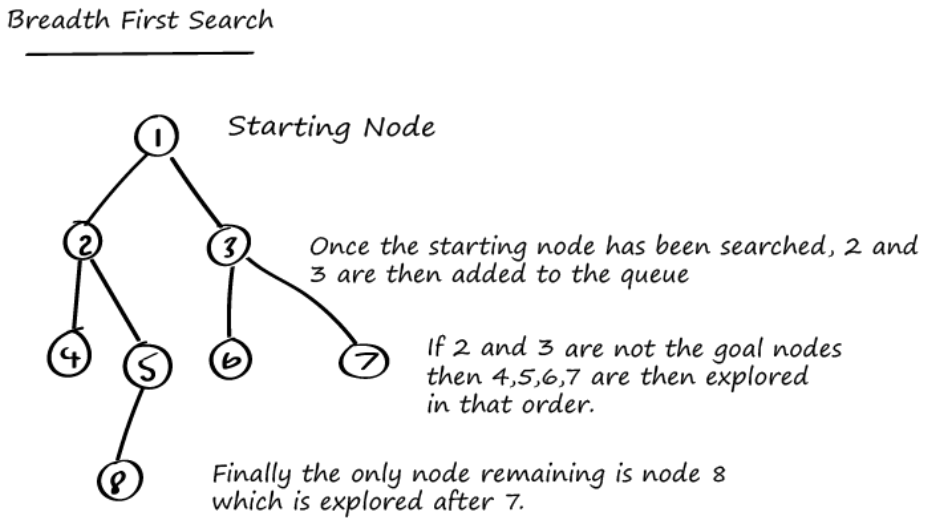


Рис. 2.12. Порядок дій алгоритму пошуку в ширину

Після аналізу ефективності різних алгоритмів було вирішено використовувати алгоритм A*, який є найбільш використовуваним для таких задач через оптимальні потреби у часі та пам’яті, відносно невелику складність та універсальність, яка дозволяє йому працювати на різних типах графів.

2.3. Вимоги до програмного застосунку

Аналіз вимог до програмного забезпечення є ключовим етапом у процесі розробки програмного продукту і визначає його успішність та відповідність потребам користувачів. Цей процес включає в себе ретельне вивчення, збір і формалізацію вимог, які визначають функціональність, характеристики та обмеження програми. Важливою метою аналізу вимог є

створення бази для подальшого проектування, розробки та тестування програмного продукту.

Етап аналізу вимог у розробці програмного забезпечення поділяється на кілька ключових етапів. На початку стоїть збір вимог, де команда розробників розмовляє з клієнтом і іншими зацікавленими особами, щоб зрозуміти, яким повинен бути продукт та які завдання він повинен вирішувати. Збір вимог - це, насамперед, розмови зі замовником та іншими людьми, які мають відношення до програмного продукту. На цьому етапі важливо з'ясувати, яким чином вони бачать роботу продукту, які проблеми вони хочуть вирішити, і які конкретні функції є для них важливими.

У загальному вигляді збір вимог має такий вигляд:

- Визначення цілей та завдань: Визначення, що саме має вирішувати програма, та які задачі повинна виконувати.
- Ідентифікація зацікавлених сторін: Визначення основних зацікавлених сторін, таких як клієнти, користувачі, адміністратори і т.д.
- Проведення спільних зустрічей: Взаємодія із зацікавленими сторонами для збору деталей і уточнення вимог.

Наступним важливим етапом є аналіз цих вимог, де вже розглядається, як ці вимоги можуть бути впроваджені в програмному продукті. Це означає розуміння того, як вони взаємодіють між собою, визначення пріоритетів і можливість розбиття їх на більш дрібні частини. Він складається з наступних етапів:

- Розбір і декомпозіція: Розбиття загальних вимог на менші, конкретні компоненти для кращого розуміння та управління.
- Прийняття рішень про пріоритети: Визначення важливості та пріоритетів різних вимог для коректного розподілу ресурсів.

Після аналізу вимоги формалізуються. Це може включати в себе створення документації, яка визначає кожну функцію, характеристику і

обмеження продукту. Також можуть бути визначені критерії прийняття, тобто умови, за якими можна визначити, що вимоги виконані.

Важливим етапом є також валідація та верифікація вимог. Під час валідації перевіряється, чи вимоги відповідають потребам та очікуванням клієнта. Верифікація стосується перевірки, чи вони консистентні, чи вони відповідають стандартам та чи враховують всі нюанси.

Останнім пунктом є управління вимогами. Оскільки розробка може змінюватися, вимоги також можуть змінюватися, і важливо слідкувати за цими змінами, забезпечуючи, щоб вони враховувались та впроваджувались систематично і без проблем. Цей етап розділяється на:

- Систематичний підхід до змін: Визначення процесу управління змінами та контролю за змінами у вимогах.
- Впровадження ретельного документування: Збереження записів про всі зміни та їх вплив на різні аспекти програмного продукту.

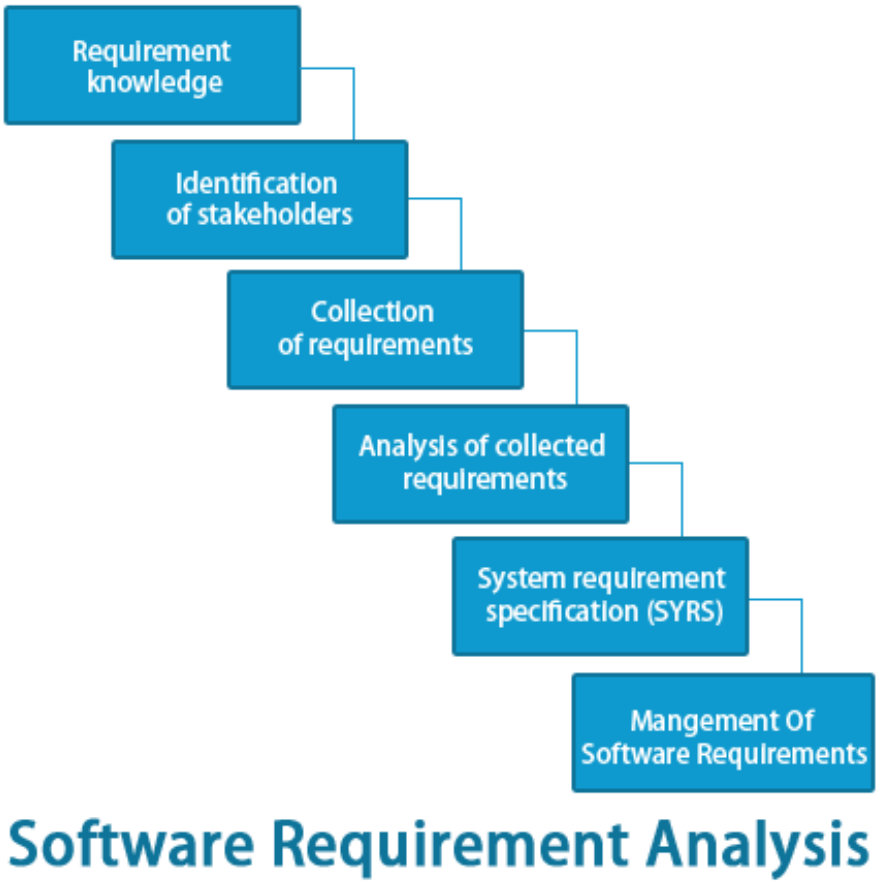


Рис. 2.13. Структура і порядок етапів аналізу вимог до програмного забезпечення

Вимоги до програмного забезпечення повинні бути докладно визначеними, зрозумілими, мінімізованими та відповідати потребам та очікуванням користувачів і замовників. Ось деякі ключові характеристики, які повинні бути властиві добре сформульованим вимогам:

- Вимоги повинні бути виражені чітко та зрозумілою мовою, щоб уникнути можливих непорозумінь.
- Важливо уникати великої кількості термінів та термінології, невідомої користувачеві.
- Всі важливі аспекти функціональності та характеристик повинні бути включені в вимоги.
- Вимоги повинні бути докладними та не залишати місця для тлумачень чи розуміння.
- Уникати двозначних формулювань, які можуть викликати різне розуміння.
- Вимоги повинні бути конкретними та вимірюваними, щоб їх можна було точно перевірити.
- Використання кількісних показників, які можна виміряти, робить вимоги більш об'єктивними.
- Вимоги повинні бути гнучкими та відкритими до змін, оскільки вони можуть змінюватися протягом розробки.
- Забезпечення системи управління змінами може полегшити цей процес.
- Вимоги повинні враховувати контекст використання програмного продукту та потреби користувачів у цьому контексті.

- Вимоги повинні визначати необхідність забезпечення ефективності та продуктивності системи, щоб вона відповідала очікуванням користувачів.
- Якщо програмний продукт має враховувати аспекти безпеки та конфіденційності, вимоги повинні чітко визначати вимоги до захисту інформації.
- Якщо система повинна бути сумісною з іншими продуктами або масштабованою, вимоги повинні враховувати ці аспекти.

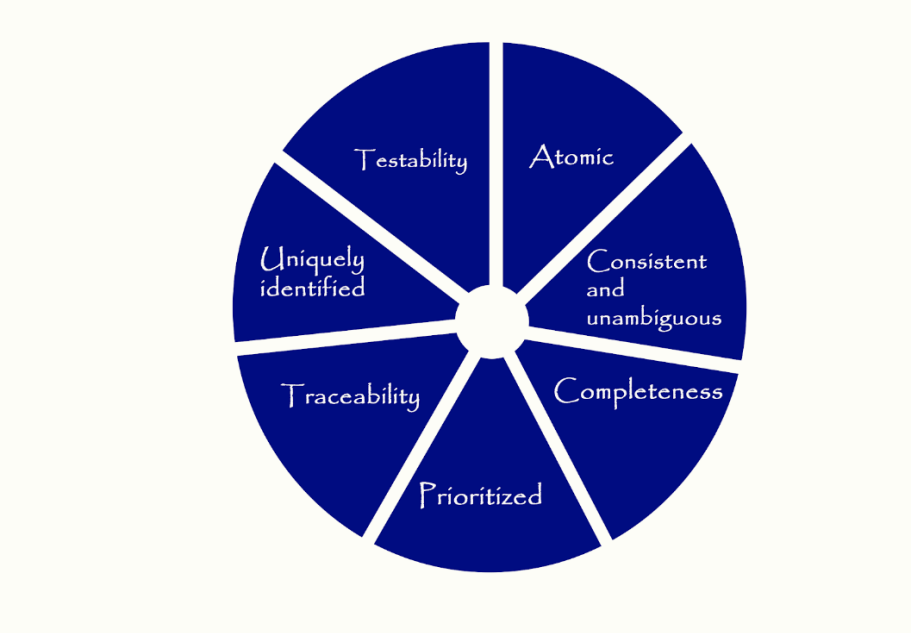


Рис. 2.14. Якості коректних вимог до програмного забезпечення

Вимоги до ПЗ поділяються на такі види:

- Бізнес-вимоги (Business Requirements):

  Ці вимоги визначають бізнес-цілі та завдання, які програмне забезпечення повинно вирішити для підтримки бізнес-процесів.

  Приклад: "Розробити систему електронного магазину, яка забезпечить зручне онлайн-замовлення та обробку платежів."
- Користувацькі вимоги (User Requirements):

  Ці вимоги визначають очікування та потреби кінцевих користувачів, які використовуватимуть програмне забезпечення.

Приклад: "Система повинна мати інтуїтивний інтерфейс для зручного використання покупцями без попереднього навчання."

- Функціональні вимоги (Functional Requirements):

  Ці вимоги визначають конкретні функції та операції, які повинно виконувати програмне забезпечення.

  Приклад: "Система повинна забезпечувати можливість додавання товарів до кошика, оформлення замовлення та перегляд історії покупок."

- Нефункціональні вимоги (Non-functional Requirements):

  Ці вимоги визначають якості, властивості та обмеження програмного забезпечення, які не стосуються конкретних функцій.

  Приклад: "Система повинна мати час відгуку менше 2 секунд для покращення користувацького досвіду."

- Доменні вимоги (Domain Requirements):

  Ці вимоги визначають контекст або домен, в якому використовуватиметься програмне забезпечення, та враховують специфічні вимоги цього домену.

  Приклад: "Система повинна відповідати законодавчим вимогам щодо обробки особистих даних."

Кожен вид вимог виконує свою роль у процесі розробки програмного забезпечення. Бізнес-вимоги визначають контекст та цілі проекту, користувацькі вимоги виражають потреби кінцевих користувачів, функціональні вимоги описують конкретні функції, нефункціональні вимоги визначають якості системи, а доменні вимоги враховують специфічні аспекти діяльності визначеного домену.[34]

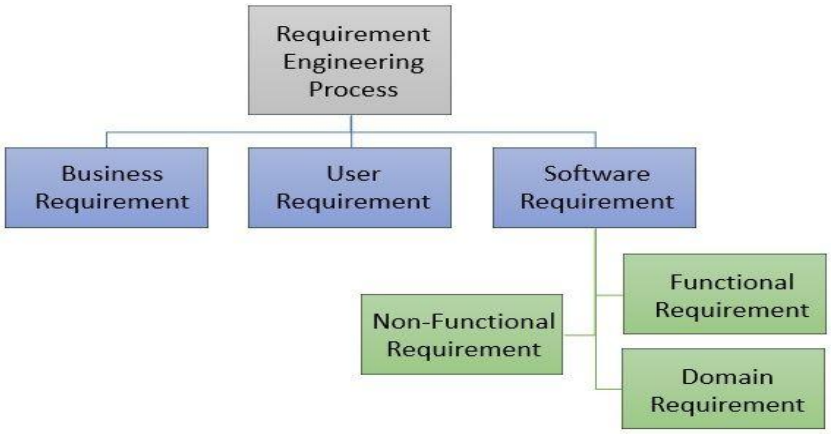


Рис. 2.15. Класифікація вимог до програмного забезпечення

2.3.1. Бізнес вимоги

Програмний продукт розрахований на людей усіх категорій, тому повинен бути доступним і зрозумілим незалежно від користувача. Після дослідження було виведено такі бізнес вимоги до застосунку для оповіщення сигналу тревоги та формування маршруту до укриття:

- Попередження про повітряну тривогу: бот повинен отримувати інформацію в режимі реального часу від системи попередження про повітряні тривоги. Це включає офіційні повідомлення від органів влади.
- Геолокаційні послуги: бізнес-вимога полягає в тому, щоб за умови надання точного місцезнаходження користувачами надавати персоналізовану інформацію та маршрутизацію.
- Інтуїтивний інтерфейс: розробка інтуїтивного інтерфейсу, який дозволяє користувачам швидко та ефективно отримувати необхідну інформацію та взаємодіяти з ботом без зайвих ускладнень.
- Маршрутизація: бот повинен надавати користувачам оптимальні маршрути від їхнього поточного місцезнаходження до найближчого укриття.
- Високий рівень сервісу: бот повинен надавати високий рівень сервісу, який включає у себе надійність, оперативність та точність інформації. Безпека та ефективність повинні бути в основі дизайну бота.

- Інформування про укриття: забезпечення користувачів інформацією про доступні укриття. Інформування про стан укриттів не входить в вимоги, бо ця інформація може бути отримана лише від офіційних джерел у реальному часі, проте вони не надаються.

Ці бізнес-вимоги було узгоджено з потребами потенційних користувачів та можливостями технічної реалізації, щоб забезпечити успішну реалізацію та використання бота в умовах повітряних тривог.

2.3.2. Функціональні вимоги

Функціональні вимоги повинні бути детально визначені та враховувати потреби користувачів та бізнес-цілей. Їхній розвиток і впровадження визначать успішність та ефективність роботи бота у виконанні своєї основної функції - забезпеченні безпеки під час повітряних тривог.

Було сформовано такі функціональні вимоги:

- Отримання інформації про повітряні тривоги: бот повинен мати можливість отримувати повідомлення про повітряні тривоги в режимі реального часу від відповідної системи або джерела інформації.
- Запит місцезнаходження: функціональність для ініціювання запиту від бота до користувача щодо надання геолокаційної інформації.
- Пошук найближчого укриття: розробка алгоритму для пошуку найближчого укриття на основі геолокаційних даних користувача та інформації про доступні укриття.
- Маршрутизація: функціональність для побудови оптимального маршруту від поточного місцезнаходження користувача до обраного укриття.
- Інформування про укриття: надання користувачам інформації про найближчі укриття.

- Система повідомлень: забезпечення системи повідомлень для сповіщення користувачів про тривоги, оновлення маршрутів, стан укриттів та іншої важливої інформації.
- Інтерфейс з картами та навігацією: використання відповідних API для карт та навігації для забезпечення візуалізації маршрутів та укриттів.

2.3.3. Нефункціональні вимоги

Специфіка проекту потребує особливих нефункціональних вимог, що забезпечать правильну роботу системи. Кінцевий список вимог виглядає так:

- Очікувана швидкість відгуку: бот повинен забезпечити швидкий відгук на запитання та повідомлення користувачів, особливо в ситуаціях повітряної тривоги, де час критично важливий.
- Продуктивність системи: забезпечення ефективної роботи системи навіть при великому потоці запитань та обробці даних.
- Захист особистих даних: забезпечення конфіденційності та захисту особистих даних користувачів, зокрема, геолокаційної інформації.
- Стабільність системи: забезпечення стабільної роботи бота, навіть при великому обсязі запитань та непередбачених обставинах.
- Зручність використання: інтерфейс має бути зручним та інтуїтивно зрозумілим для різних категорій користувачів, включаючи людей з різним рівнем технічної підготовки.
- Здатність до розширення: бот повинен мати здатність до розширення для врахування зростання числа користувачів та обсягу запитань у випадку великомасштабної екстреної ситуації.
- Доступність в будь-який час: забезпечення доступності бота в будь-який час доби, особливо в ситуаціях екстреної необхідності.

- Сумісність з різними пристроями: забезпечення роботи бота на різних платформах та пристроях, включаючи смартфони, планшети та комп'ютери.

2.3.4. Системні вимоги

Оскільки застосунок створюється на базі Telegram Bot API, якихось особливих системних вимог він не має. Єдині вимоги, яким повинний відповідати пристрій, з якого людина буде користуватися ботом – це вимоги самого додатку Telegram. Цей додаток підтримується усіма найбільшими операційними системами, як для ПК, так і для мобільних пристроїв.

Висновки

В цьому розділі було досліджено цифрові мапи та методи формування маршрутів, що використовуються цими сервісами. В результаті дослідження серед перелічених інструментів було обрано ті, що найбільше підходять для поставленої задачі.

На базі усіх попередніх аналізів було сформовано вимоги до розроблюваного бота. Бот повинен забезпечувати користувача усією функціональністю, що була описана у вимогах.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ

3.1. Архітектура

Архітектура програмного забезпечення (АПЗ) є ключовим елементом у розробці програмних продуктів, визначаючи їхню структуру, організацію та взаємодію компонентів. Існують різноманітні типи архітектур програмного забезпечення, кожен з яких має свої переваги та відповідає різним вимогам проектів.

До найтиповіших архітектур ПЗ відносяться монолітна, клієнт-серверна, мікросервісна, та багатошарова архітектури.

3.1.1. Монолітна архітектура

Монолітна архітектура визначається як метод розробки програмного забезпечення, в якому весь функціонал програми об'єднується в єдиний блок, що взаємодіє з базою даних та користувацьким інтерфейсом. На відміну від інших архітектурних підходів, які передбачають розподілені компоненти програми, моноліт представляє собою єдиний компактний блок.

У програмах з монолітною архітектурою весь код і логіка, зазвичай, розміщені в одній основній інсталяції або додатку. Це означає, що всі компоненти, такі як база даних, серверна логіка та користувацький інтерфейс, об'єднані в одному місці. Такий підхід полегшує розробку та тестування, оскільки всі частини програми доступні для локальних змін.

Однак монолітна архітектура може викликати певні обмеження, наприклад, у складності масштабування та розширення, оскільки всі компоненти взаємозалежні. Також зміна одного компонента може вимагати перекомпіляції та випуску всього моноліту. Такі системи часто менш гнучкі порівняно з розподіленими архітектурами, такими як мікросервіси чи клієнт-серверні системи.

Монолітна архітектура є традиційним та простим способом розробки програмного забезпечення, де всі компоненти додатку взаємодіють безпосередньо, у єдиному процесі. Її переваги включають простоту розробки, тестування та розгортання, адже всі частини додатку розташовані в єдиному середовищі.

Незважаючи на це, монолітні системи можуть стати проблематичними при розширенні масштабів проектів або зміні вимог. Інтеграція нового функціоналу чи змін в одній частині може вплинути на весь код, а розширення та масштабування можуть стати складнішими завданнями. Монолітна архітектура часто застосовується в простих чи невеликих проектах, де простота має важливе значення, але великі або складні системи можуть вигідно використовувати розподілені архітектурні підходи, такі як мікросервіси чи клієнт-серверні рішення.

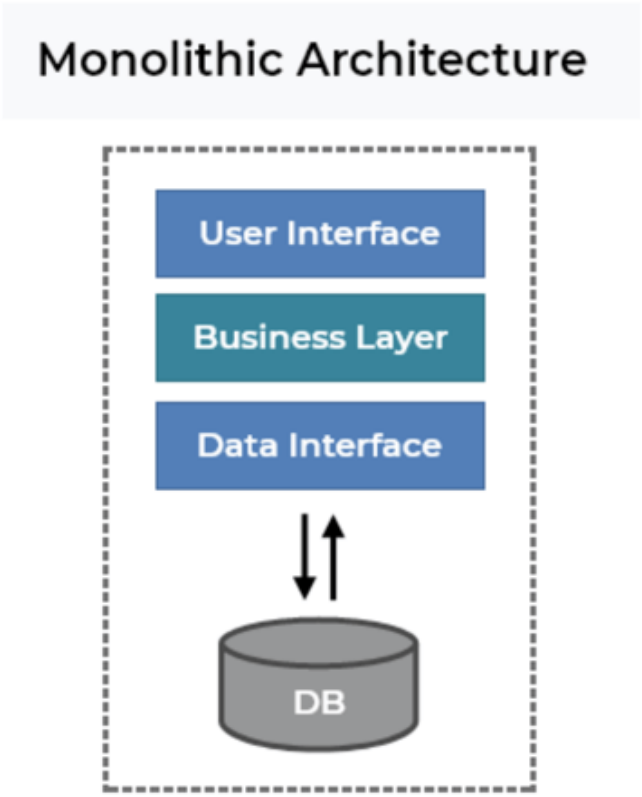


Рис. 3.1. Монолітна архітектура

Багато відомих програм та сервісів використовують монолітну архітектуру. Наприклад, WordPress – це система управління контентом для веб-сайтів, де всі функції, включаючи роботу з базою даних та інтерфейс адміністратора, об'єднані в одному додатку. Також, електронна комерційна платформа Magento та веб-фреймворк Django для Python мають схожий підхід, об'єднуючи всі компоненти в одному проекті. Ще одним прикладом є Joomla – система управління контентом, подібна до WordPress, яка також використовує монолітну архітектуру. Ці приклади свідчать про популярність монолітної архітектури в розробці різноманітних систем і додатків, особливо тих, які не потребують значної розподіленої гнучкості чи масштабованості.

3.1.2. Мікросервісна архітектура

Мікросервісна архітектура – це сучасний підхід до розробки програмного забезпечення, спрямований на декомпозицію великих та складних систем на менші, автономні компоненти, що отримали назву мікросервіси. Кожен мікросервіс виконує конкретну функцію та може функціонувати незалежно. Взаємодія між мікросервісами відбувається за допомогою API, що дозволяє їм обмінюватися даними та функціоналом.

Ключова концепція мікросервісної архітектури полягає в поділі програмного застосунку на компактні, легко управляючі компоненти, спрощуючи розробку, розгортання та масштабування системи. Кожен мікросервіс може бути розроблений автономно, використовуючи технології та мови програмування, які найбільше підходять для конкретної задачі.

Одним з ключових аспектів мікросервісної архітектури є її гнучкість. Кожен мікросервіс може масштабуватися незалежно, що сприяє ефективному використанню ресурсів та адаптації до змін у навантаженні. Крім того, цей підхід дозволяє проводити оновлення окремих сервісів без впливу на інші частини системи, сприяючи полегшенню підтримки та розвитку.

Проте, зі збільшенням кількості мікросервісів стає важливим ефективне управління конфігурацією, моніторинг, тестуванням та безпекою системи в

цілому. Успішна реалізація мікросервісної архітектури вимагає детального планування та ефективного управління. Комунікація між сервісами повинна бути оптимізованою, а також слід вирішувати завдання управління конфігурацією та моніторингу. Крім того, обов'язково враховувати аспекти безпеки та надійності в розподіленому середовищі.

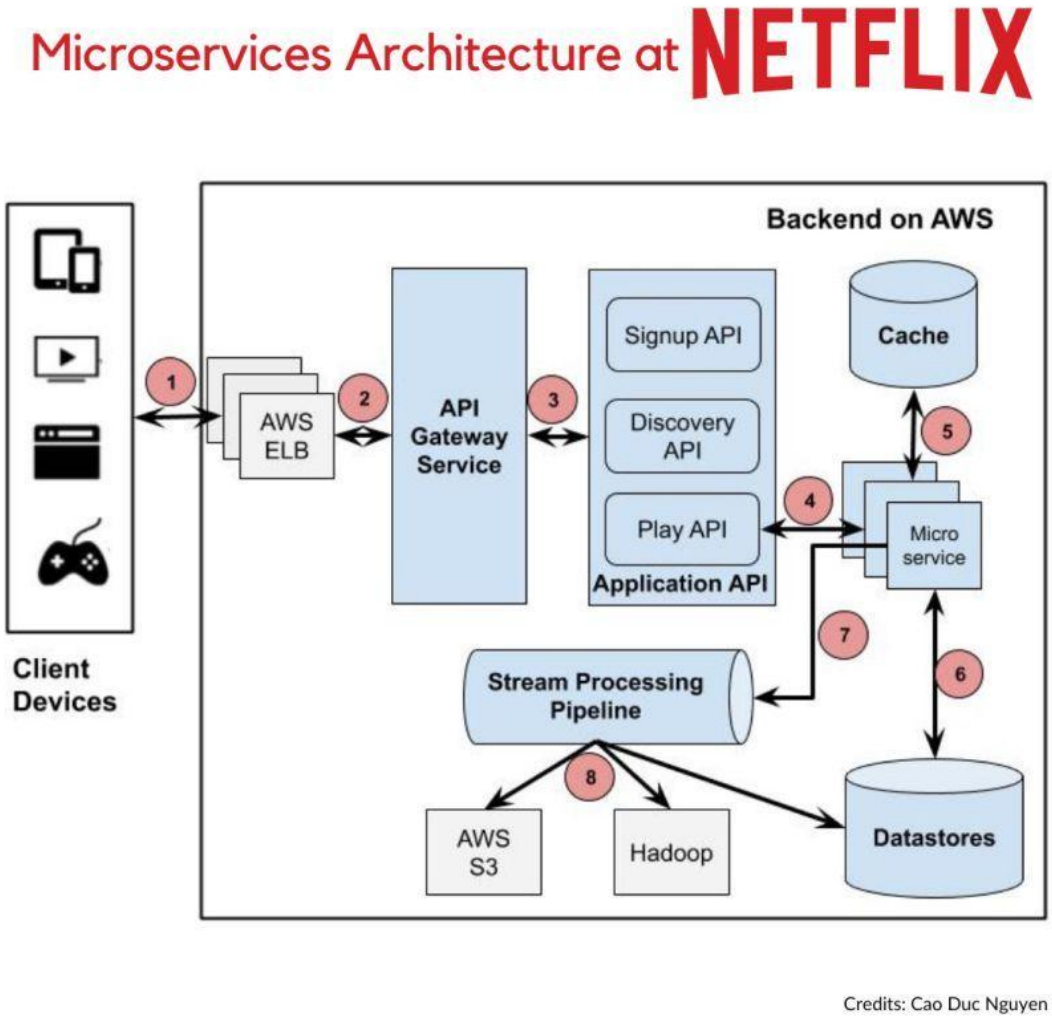


Рис. 3.2. Мікросервісна архітектура компанії Netflix

Багато сучасних систем та сервісів вибирають мікросервісну архітектуру для досягнення високої гнучкості, масштабованості та швидкості реакції на зміни вимог. Декілька прикладів систем, що використовують мікросервісну архітектуру:

- Netflix використовує цей підхід для свого стрімінгового сервісу, де різні мікросервіси відповідають за рекомендації, оплату, управління відеоконтентом тощо.
- Uber використовує мікросервіси для розбиття своєї платформи на невеликі, незалежні частини, які можуть функціонувати та розвиватися незалежно одна від одної.
- Amazon Web Services (AWS) засновані на мікросервісній архітектурі, де різні сервіси, такі як Amazon S3 та Amazon EC2, представлені як окремі мікросервіси.
- Spotify використовує мікросервіси для обробки рекомендацій, управління користувачами та потокового відтворення музики.
- Airbnb застосовує мікросервіси для реалізації різних аспектів свого функціоналу, таких як пошук житла, обробка платежів та взаємодія з користувачами.

Ці приклади ілюструють успішне використання мікросервісної архітектури в різних галузях, таких як стрімінгові сервіси, транспортні платформи, хмарні послуги тощо.

3.1.3. Клієнт-серверна архітектура

Клієнт-серверна архітектура є однією з традиційних та широко використовуваних архітектур у розробці програмного забезпечення. У цьому підході програма або система розділена на дві основні компоненти: клієнт і сервер.

Клієнт - це складова системи, яка відповідає за виведення інформації для користувача та обробку його введеного. Клієнтська частина може бути реалізована на різних платформах, включаючи десктопні комп'ютери, мобільні пристрої та веб-браузери. Існують два види клієнтів:

- Тонкий клієнт (Thin Client): Тут більша частина логіки та обчислень знаходиться на сервері. Клієнт відповідає за виведення інформації та

взаємодію з користувачем, але для обчислень зазвичай використовується потужний сервер.

- Товстий клієнт (Thick Client): У цьому випадку значна частина логіки розташована на самому клієнтському пристрої. Клієнт може обробляти дані та виконувати обчислення, взаємодіяти з локальною базою даних тощо.

Сервер - це компонент системи, який забезпечує ресурси та послуги, необхідні для виконання завдань клієнта. Сервер може бути фізичним пристроєм або програмним модулем, який виконується на віддаленому сервері. Він обробляє запити від клієнтів та повертає їм відповіді або результати обчислень. Також існують кілька типів серверів:

- Веб-сервер: Використовується для обслуговування HTTP-запитів від клієнтів. Надає веб-ресурси, такі як HTML-сторінки, статичні файли та інші.
- Додатковий сервер (Application Server): Може виконувати бізнес-логіку, обробку запитів, взаємодію з базою даних та інші завдання, які потребують обчислень.
- База даних: Це сервер, який забезпечує зберігання та обробку даних. Клієнти взаємодіють з базою даних через запити, які оброблюються на сервері.

Клієнт та сервер можуть бути розташовані на різних машинах або в різних мережевих середовищах. Це дозволяє створювати розподілені системи, де різні компоненти можуть виконуватися на віддалених серверах.

Взаємодія між клієнтом та сервером здійснюється через мережеві з'єднання. Клієнт відправляє запити на сервер, і сервер відповідає їм, забезпечуючи необхідну інформацію чи виконуючи конкретні дії. Взаємодія забезпечується такими протоколами:

- HTTP (Hypertext Transfer Protocol): Використовується для взаємодії між веб-клієнтами та серверами.

- TCP/IP (Transmission Control Protocol/Internet Protocol): Забезпечує надійний обмін даними між клієнтом та сервером.

Клієнт-серверна архітектура дає можливість масштабування системи, оскільки можна додавати нові клієнти чи сервери для розширення функціоналу та навантаження.

Цей підхід часто використовується в інтерактивних додатках, де клієнт відповідає за реакцію на введення користувача, а сервер - за обробку та забезпечення необхідних даних.

Клієнт-серверна архітектура забезпечує стандартизований та ефективний спосіб розподіленої роботи в системах різного роду, від веб-додатків до корпоративних програм. Вона надає гнучкість та можливість масштабування, а також дозволяє використовувати різні технології для клієнтської та серверної частин.

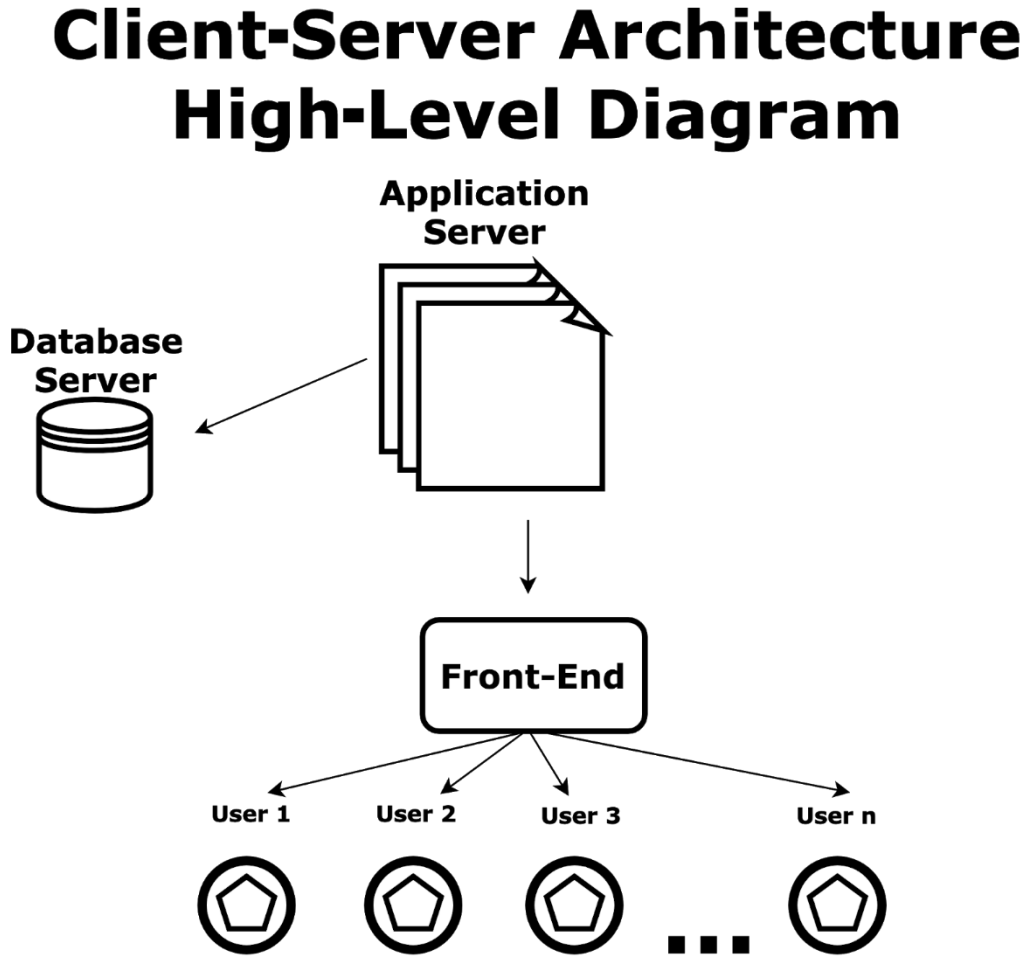


Рис. 3.3. Клієнт-серверна архітектура

Багато сучасних систем та додатків використовують клієнт-серверну архітектуру для свого функціонування. Наприклад, Microsoft Exchange є системою електронної пошти та групового календаря, де клієнти, такі як Microsoft Outlook, взаємодіють з сервером Exchange для роботи з електронною поштою та календарем. Також, системи управління базами даних, наприклад, Microsoft SQL Server чи Oracle Database, використовують клієнт-серверну архітектуру для взаємодії з базою даних через різні клієнтські інтерфейси. File Servers, Print Servers, Game Servers, VoIP Servers та Proxy Servers також використовують цей підхід для обробки відповідних завдань у мережевих середовищах. Ці приклади розкривають варіативність використання клієнт-серверної архітектури в різних областях, забезпечуючи ефективну взаємодію між клієнтами та серверами.

3.1.4. Багатошарова архітектура

Багатошарова архітектура — це методологія проектування програмного забезпечення, де функціональність розділена між різними "шарами" або рівнями. Кожен з цих рівнів виконує конкретні завдання та несе відповідальність за свою область. Такий підхід широко використовується для полегшення розподіленості, підтримки модульності, поліпшення масштабованості та сприяння ефективному розвитку та обслуговуванню програм.

Багатошарова архітектура включає в себе кілька ключових компонентів. Перший з них – шар представлення, або інтерфейс. Він відповідає за взаємодію з користувачем. Включає всі компоненти, які відображають інформацію та обробляють введення користувача.

Наступний шар - бізнес-логіка, або доменний шар. Він включає в себе логіку, пов'язану з бізнес-процесами та правилами додатка, а також забезпечує обробку даних, валідацію та інші ключові операції.

Шар доступ до даних взаємодіє з базою даних або іншими джерелами даних. Включає операції читання, запису та валідацію даних.

Комунікація між шарами забезпечується за допомогою визначених методів комунікації, таких як API.

Безпека та автентифікація відповідає за забезпечення безпеки системи, перевірку доступу та автентифікацію користувачів.

Багатошарова архітектура допомагає розділити відповідальності, полегшити підтримку, розвиток та тестування програм. Вона сприяє модульності та взаємозамінності, робить систему більш гнучкою та легко масштабується.

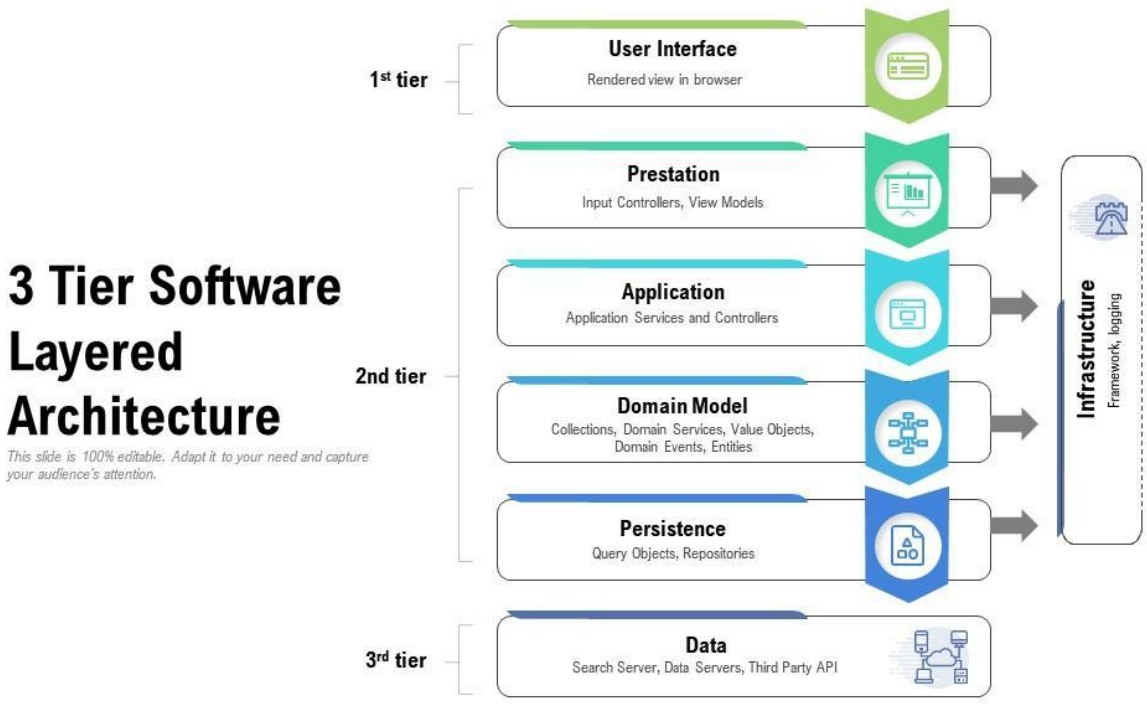


Рис. 3.4. Багатошарова архітектура (в цьому випадку – тришарова)

В результаті дослідження для розроблюваної системи було обрано мікросервісну архітектуру, що найбільше підходить для подібних додатків.

Мікросервісна архітектура для Telegram бота передбачає розгортання окремих компонентів (мікросервісів), кожен з яких відповідає за конкретну

функціональність. Кожен мікросервіс може розвиватися та масштабуватися незалежно, що надає гнучкість та можливість швидко реагувати на зміни та розвивати окремі компоненти системи незалежно один від одного.

Основні мікросервіси:

- Сервіс обробки повітряних тривог: отримання та аналіз інформації щодо повітряних тривог, фільтрація та прийняття рішень щодо сповіщення користувачів.
- Сервіс геолокації та навігації: збір геолокаційних даних користувачів, пошук укриття та побудова маршрутів евакуації.
- Сервіс інформації про укриття: забезпечення доступу до бази даних укриттів, надання інформації користувачам щодо укриття.
- Сервіс нотифікацій: відправлення повідомлень користувачам про повітряні тривоги та оновлення.
- Сервіс інтеграції з Telegram API: взаємодія з Telegram для отримання повідомлень та відправки інформації.

Перевагами такого підходу є:

- Гнучкість та розширюваність: Кожен сервіс може розвиватися та масштабуватися незалежно, що дозволяє швидко впроваджувати нові функції та реагувати на зміни вимог.
- Незалежність: Кожен мікросервіс може бути розгорнутий, оновлений та масштабований незалежно від інших, що зменшує ризик виникнення проблем при впровадженні змін.
- Оптимальне використання ресурсів: Дозволяє використовувати ресурси та технології, які найкраще підходять для кожного окремого сервісу.
- Покращена масштабованість: Легко масштабувати лише ті частини системи, які потребують додаткових ресурсів, замість масштабування всього моноліту.

- Легша відладка та тестування: Відокремлені мікросервіси спрощують відладку та тестування, оскільки проблеми можна шукати в межах конкретного сервісу.

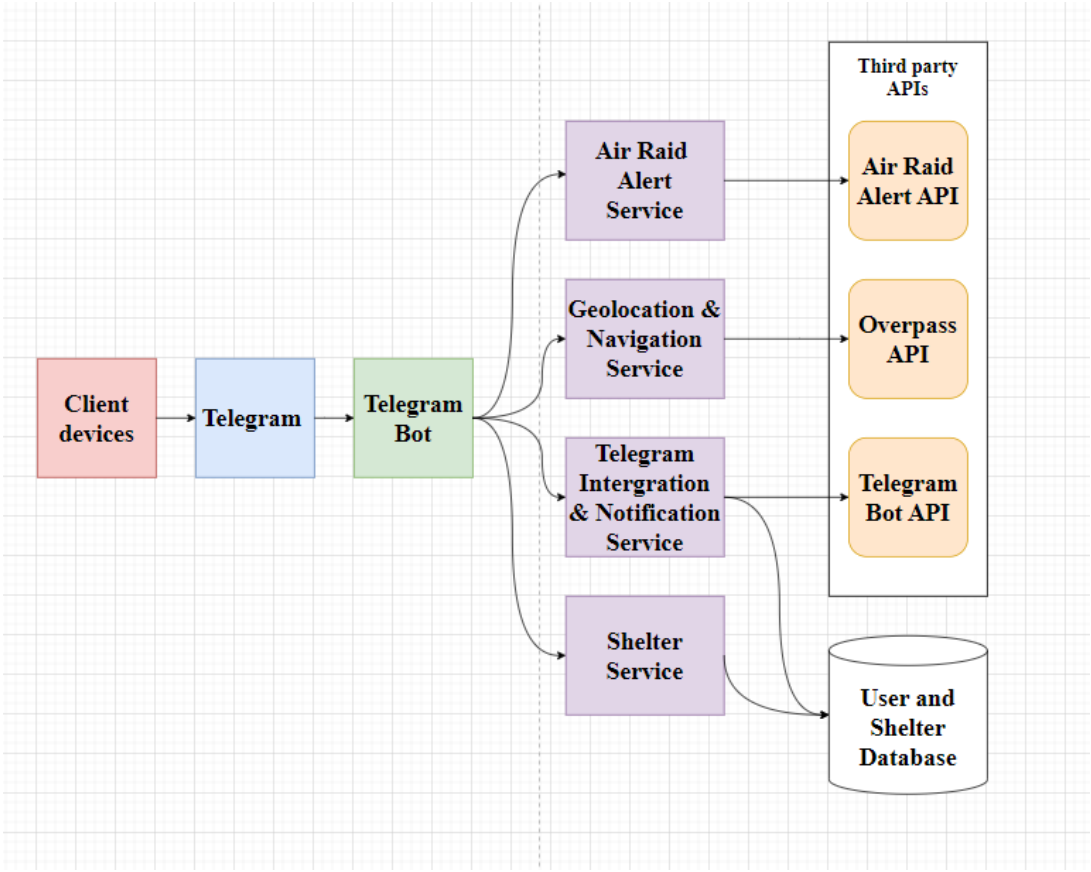


Рис. 3.5. Структура розроблюваного застосунку

3.2. Застосовані технології

3.2.1. Мова програмування і платформа для розробки

Для розробки даного бота було обрано мову C# та платформу .NET.

.NET представляє собою платформу розробки програм, яку надає Microsoft. Вона включає різноманітні інструменти, бібліотеки та ресурси, призначені для створення різних типів програм, таких як веб-додатки, настільні додатки, мобільні додатки, служби та інші. Ключовим компонентом платформи .NET є середовище виконання коду Common Language Runtime

(CLR), яке забезпечує виконання коду на різних мовах програмування, спрямованих на платформу .NET.

C# (вимовляється як "С-шарп") - це одна з основних мов програмування, яку підтримує платформа .NET. Розроблена Microsoft, C# спроектована для створення програм, які працюють на платформі .NET. Вона вирізняється такими ключовими характеристиками:

- Синтаксис: має синтаксис, схожий на мови програмування C і C++, що полегшує вивчення та використання для розробників, знайомих із цими мовами.
- Об'єктно-орієнтованість: повністю об'єктно-орієнтована мова, підтримує концепції об'єктно-орієнтованого програмування, такі як класи, успадкування, поліморфізм та інкапсуляція.
- Багатозадачність та асинхронне програмування: підтримує асинхронне програмування за допомогою ключових слів async та await, спрощуючи розробку ефективних та чуйних додатків.
- Керований код: код на C# компілюється в проміжний код (Intermediate Language - IL), який виконується CLR, забезпечуючи керування пам'яттю та безпеку типів.
- Велике різноманіття бібліотек: використовує бібліотеки класів .NET для роботи з файлами, мережами, базами даних та іншими аспектами розробки.
- Сучасні можливості мови: постійно оновлюється, впроваджуючи сучасні можливості та поліпшення, такі як паттерни та записи.

C# та платформа .NET надають розробникам потужний інструментарій для створення різноманітних та високопродуктивних додатків.

3.2.2. Telegram Bot API

Боти — це зовнішні програми, розроблені для роботи на платформі Telegram. Користувачі можуть взаємодіяти з ботами за допомогою

повідомлень, команд і вбудованих запитів. Відмінності між ботом і звичайним обліковим записом включають відсутність статусів «онлайн» або «був онлайн» для ботів, які ідентифікуються міткою «бот». Крім того, боти працюють в обмеженому серверному просторі, що призводить до автоматичного видалення повідомлень через певний проміжок часу.

На відміну від звичайних акаунтів, боти не можуть ініціювати спілкування з користувачами; натомість користувачі повинні або додати бота до групи, або розпочати з ним діалог. Ботів можна ідентифікувати за іменами користувачів, які закінчуються на "bot" (наприклад, @controllerbot). При додаванні в групу бот за замовчуванням не отримує всі повідомлення.

Боти служать різним цілям, включаючи інтеграцію із зовнішніми службами (наприклад, GitHub Bot, Image Bot), службовими функціями (наприклад, відображення погоди, переклад тексту), іграми (наприклад, шашки, шахи, вікторини) і соціальними службами (наприклад, компаньйон). -пошук ботів, таких як HotOrBot).

Користувачі можуть взаємодіяти з ботами, надсилаючи повідомлення та команди в чаті або додаючи їх до груп. Крім того, користувачі можуть робити запити безпосередньо з поля введення, ввівши @username бота та запит, що дозволяє надсилати вміст із вбудованих ботів у будь-який чат, групу чи канал.

Створені користувачем повідомлення, команди та запити передаються на сервер, на якому запущено програмне забезпечення. Проміжний сервер керує шифруванням і зв’язком із Telegram API від імені користувачів. Цей зв’язок відбувається через спрощений інтерфейс HTTPS, який пропонує спрощену версію Telegram API.

Створення та активація бота Telegram передбачає використання BotFather, офіційного інструменту Telegram. BotFather полегшує початкове налаштування нового бота, включаючи встановлення імені бота, імені користувача, розділу біографії та зображення профілю. Згодом BotFather генерує унікальний токен для доступу до бота через Telegram Bot API.

Другий крок включає фактичну розробку ботів, де ви маєте можливість вибирати з різних мов програмування та ініціювати процес розробки, використовуючи офіційні або спеціальні API для ботів. Третій крок — розгорнути бота на віртуальній машині, щоб забезпечити його безперервну роботу, навіть коли комп'ютер вимкнено. Після розгортання ваш бот готовий до використання.

Основною функцією бота є надсилання та отримання повідомлень, підтримка широкого діапазону типів повідомлень, таких як фотографії, відео, файли, опитування та голосові повідомлення. У той час як Telegram дозволяє обмінюватися файлами розміром до 2 ГБ, API Bot накладає суворіші обмеження, дозволяючи роботам завантажувати файли розміром до 20 МБ і надсилати файли розміром до 50 МБ. Боти можуть спілкуватися через особисті повідомлення лише з користувачами, які ініціювали контакт, і користувачі мають можливість заблокувати бота, запобігаючи подальшому спілкуванню. Боти не можуть надсилати повідомлення іншим роботам.

Ботів можна додавати до груп із налаштуваннями за замовчуванням, які обмежують їх видимість для певних повідомлень (розглянемо далі в розділі «Видимість повідомлень у групах»). У групі боту можуть бути надані права адміністратора для виконання дій, аналогічних адмінам, і в одній групі може бути до 20 ботів. Лише адміністратори мають право додавати ботів у публічні групи з іменами користувачів.

Боти зазвичай відповідають на команди, які починаються з "/" і складаються з латинських літер, допускаючи цифри та підкреслення. Команди розглядаються як посилання, і в налаштуваннях групи рекомендується включати ім'я користувача бота в кінці команди (наприклад, /start@examplebot), щоб відрізняти команди від різних ботів.

Режим конфіденційності, увімкнений за замовчуванням і налаштований у BotFather, гарантує, що бот у групах бачить певні повідомлення, включаючи згадки ботів, відповіді на повідомлення ботів, системні повідомлення та команди. Якщо режим конфіденційності вимкнено, бот

може бачити всі повідомлення в групі. Боти-адміністратори, незалежно від режиму приватності, мають доступ до всіх повідомлень у групі.

Спілкування з ботами зазвичай передбачає використання команд, які починаються з "/" і можуть містити латинські літери, цифри та підкреслення. Telegram пропонує додавати ім’я користувача бота до команд у групах, щоб розрізняти команди від різних ботів. Крім того, BotFather дозволяє вказувати командні підказки для бота, відображаючи їх під час введення «/» та команд із кнопкою відкриття меню поруч із кнопкою «Надіслати», якщо підказки доступні.

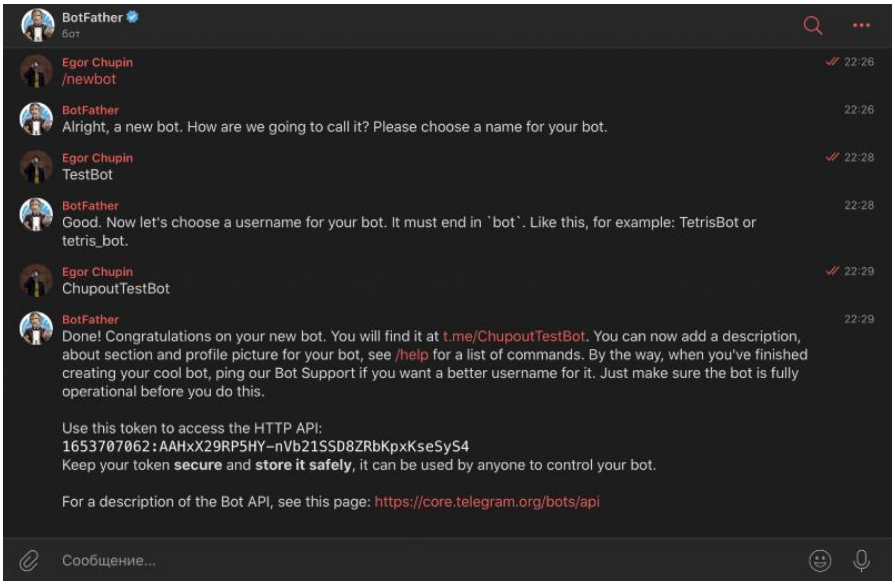


Рис. 3.6. Початкове налаштування бота у BotFather

Висновки

Після розгляду різних варіантів, було затверджено наступні вимоги до розробки застосунку – мікросервісна архітектура, платформа .NET та мова C#, яка має якісну бібліотеку для роботи з Telegram Bot API.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ДЕМОНСТРАЦІЯ РОБОТИ ПЗ

4.1. Огляд системи

Застосунок розроблено у вигляді консольного додатку на базі платформи .NET 6.0.

Перш за все, у боті є 2 основні обробники: обробник оновлень і обробник помилок. Обробник оновлень використовується для реагування на отримані повідомлення та подальших операцій з їх вмістом. Обробник помилок використовується для належної обробки будь-яких винятків.

Залежно від типу повідомлення обробник оновлень викличе спеціальні методи. Наприклад, якщо ми отримуємо текстове повідомлення, бот викличе метод HandleTextMessage, який аналізує повідомлення користувача та визначає, що робити.

Текстовий обробник розпізнає команди та передає їх відповідним сервісам, які виконають ці команди.

4.2. Початок роботи з ботом

Першою дією після запуску боту потрібно обрати область, в якій знаходиться користувач, за допомогою команди /setregion. Це зроблено для надання більш точної інформації, бо не завжди безпечно тримати геолокаційні сервіси пристрою ввімкненими. Окрім того, вони збільшують витрати заряду аккумуляторів.

Вибір робиться на спеціальній клавіатурі (рис. 4.1), що містить в собі майже усі області України (окрім АР Крим, актуальність інформації в якій не може бути підтверджена навіть офіційними джерелами).

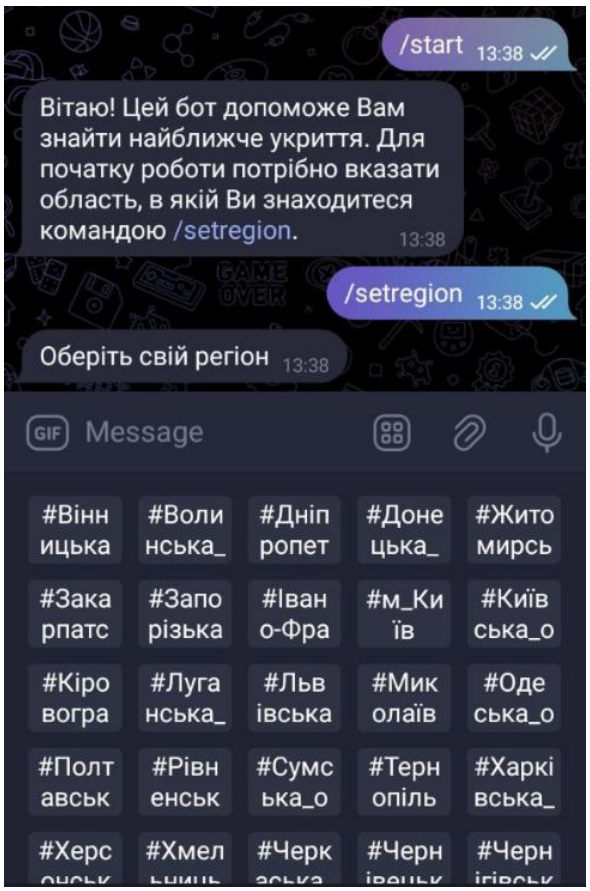


Рис. 4.1. Вибір регіону

Після вибору області, користувач отримає повідомлення (рис. 4.2), що підтвердить отримання ботом інформації про місцезнаходження користувача.

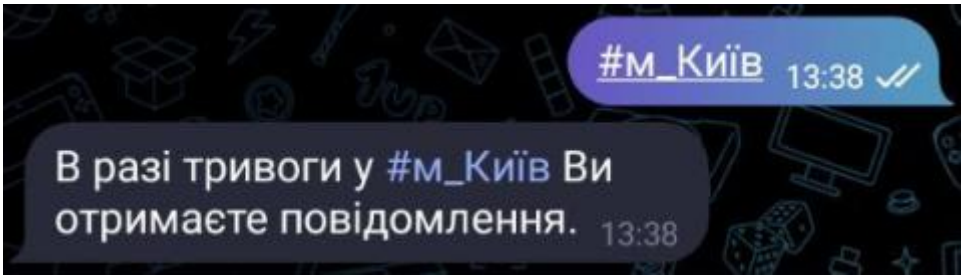


Рис. 4.2. Відповідь бота на вибір регіону

4.3. Функція оповіщення населення та пошуку укриттів

В момент, коли оголошується повітряна тривога, усі користувачі в областях, яких ця тривога торкнулася, отримують повідомлення про початок тривоги та запит на отримання точного місцезнаходження (рис. 4.3).

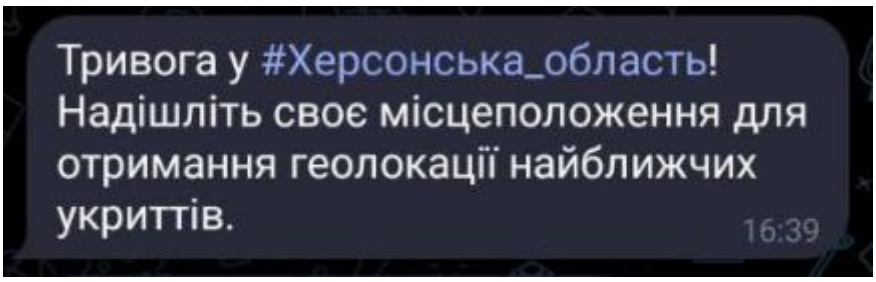


Рис. 4.3. Повідомлення бота про тривогу

При отриманні геолокації бот почне пошук найближчих укриттів і надішле у відповідь їх список (рис. 4.4).

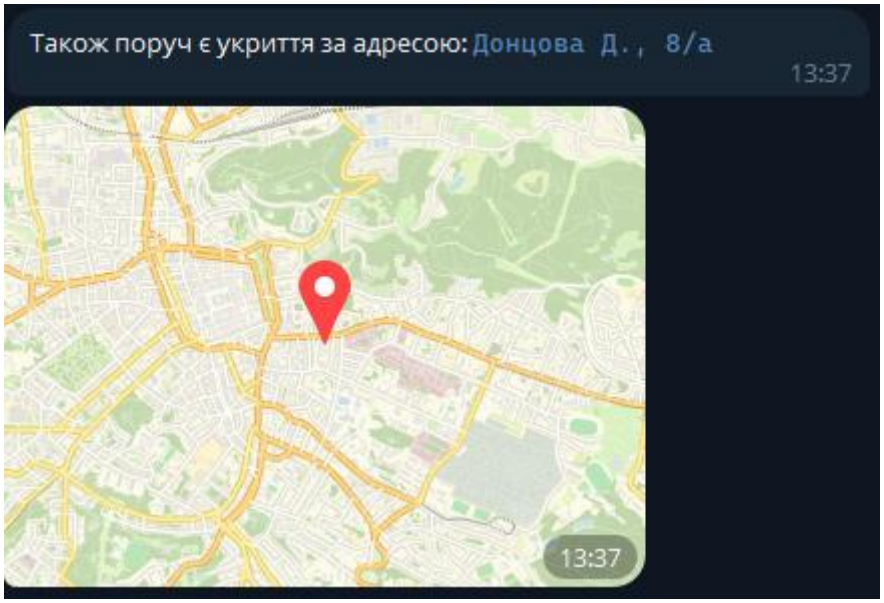


Рис. 4.4. Локація найближчого укриття

Для вибору потрібно обрати повідомлення з обраним укриттям на написати боту команду /choose (рис. 4.5).

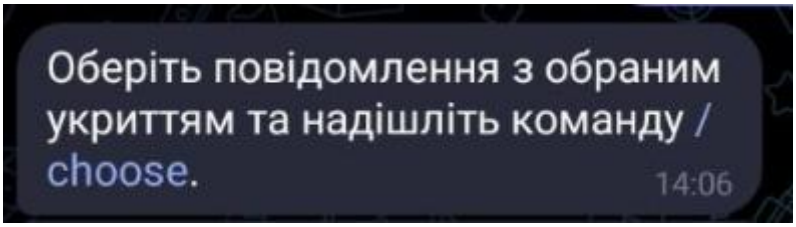


Рис. 4.5. Повідомлення про вибір укриття

У відповідь бот надішле користувачу маршрут до обраного укриття (рис. 4.6).

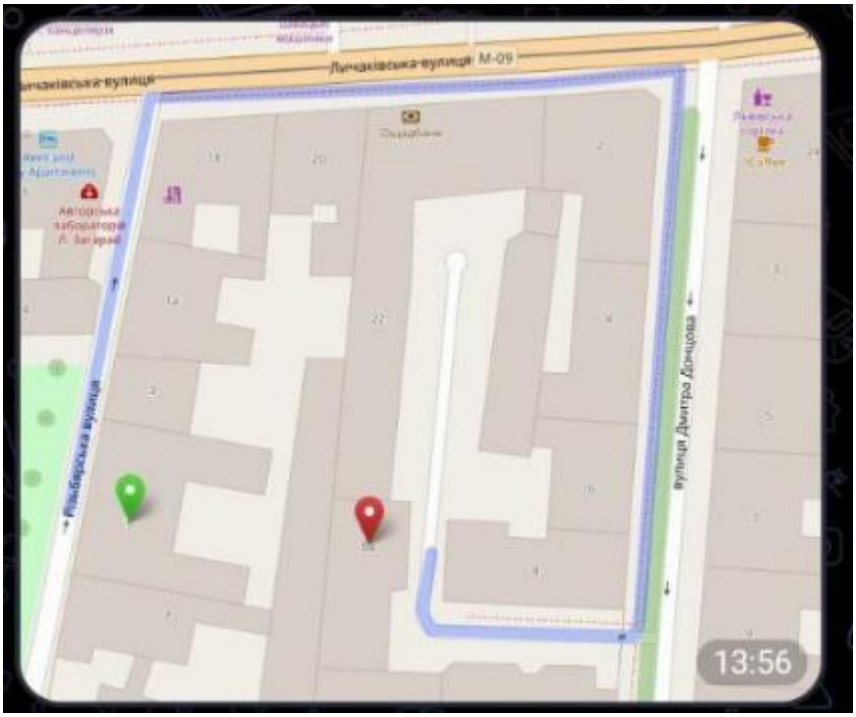


Рис. 4.6. Маршрут до обраного укриття

Висновки

В цьому розділі було розглянуто готовий програмний застосунок та продемонстровано його функціонал.

Основною перевагою над іншими подібними сервісами є об’єднання усіх основних потреб населення в разі оголошення повітряної тривоги в один застосунок, що зменшує час, необхідний на швидкість рішень та дій.

Якщо ж розглядати можливі модифікації, то в цілому вони співпадають з іншими додатками такого характеру. Більш детальне оповіщення про повітряні тривоги, що прив’язане не до області, а до району або територіальної громади, більш актуальна інформація про укриття. Проте, ці поліпшення неможливі без втручання місцевих органів влади. Тим не менш, є варіант покращення ситуації з даними про укриття. Додання системи оцінювання укриттів зможе допомогти людям у виборі, до якого саме укриття прямувати.

ВИСНОВКИ

В ході виконання дипломної роботи було досліджено проблеми оповіщення населення про повітряні тривоги та надання інформації про найближчі укриття. Було проаналізовано психологічну та технічну сторони питання, проведено порівнювальну характеристику аналогів у цій сфері, а також методів пошуку шляху і технологій роботи з цифровими мапами.

Результати дослідження було систематизовано та на цій основі було сформовано вимоги до майбутнього програмного застосунку та визначено технології, що будуть використовуватися у розробці для максимальної ефективності.

У подальшому аналізі було обрано архітектуру програмного застосунку і описано деталі реалізації обраних технологій розробки.

У четвертому розділі роботи було проведено тестування і демонстрацію готового програмного засобу, а також аналіз його можливих модифікацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Оповіщення про повітряну тривогу від ДСНС: чи приходитимуть ці сигнали на телефон найближчим часом [Електронний ресурс] // Вільне радіо. Режим доступу до ресурсу: https://freeradio.com.ua/opovishchennia-pro-povitrianu-tryvohu-vid-dsns-chy-prykhodytymut-tsi-syhnaly-na-telefon-naiblyzhchym-chasom/
2. Shalev A, Liberzon I, Marmar C (June 2017). "Post-Traumatic Stress Disorder". The New England Journal of Medicine. 376 (25): 2459–2469.
3. Hollifield M, Warner TD, Lian N, Krakow B, Jenkins JH, Kesler J, Stevenson J, Westermeyer J (August 2002). "Measuring trauma and health status in refugees: a critical review". JAMA. 288 (5): 611–21.
4. Porter M, Haslam N (October 2001). "Forced displacement in Yugoslavia: a meta-analysis of psychological consequences and their moderators". Journal of Traumatic Stress. 14 (4): 817–34.
5. UNESCO (2018). A Lifeline to learning: leveraging mobile technology to support education for refugees. UNESCO.
6. "Далі буде краще," — військовий психолог про підтримку психологічного стану під час війни [Електронний ресурс] // Суспільне. Режим доступу до ресурсу: https://suspilne.media/221165-dali-bude-krase-vijskovij-psiholog-pro-pidtrimku-psihologicnogo-stanu-pid-cas-vijni/
7. Реальні та «фантомні» сигнали повітряної тривоги: психолог дала поради, як себе заспокоїти [Електронний ресурс] // Українські національні новини. Режим доступу до ресурсу: https://unn.ua/news/realni-ta-fantomni-signali-povitryanoyi-trivogi-psikholog-dala-poradi-yak-sebe-zaspokoyiti
8. Куди поділася реакція на повітряні тривоги? [Електронний ресурс] // Світловодськ. Режим доступу до ресурсу: https://svetlovodsk.com.ua/14184-siren.html

9. Понад два десятки повітряних тривог за тиждень: чи змінилася реакція сумчан на загрози [Електронний ресурс] // Суспільне. Режим доступу до ресурсу: https://suspilne.media/255414-ponad-dva-desatki-povitranih-trivog-za-tizden-ci-zminilasa-reakcia-sumcan-na-zagrozi/
10. Уже ігнорую повітряну тривогу, або чому нам стало байдуже на небезпеку [Електронний ресурс] // Вікна. Режим доступу до ресурсу: https://vikna.tv/dlia-tebe/psykholohiia/uzhe-ignoruyu-povitryanu-tryvogu-abo-chomu-nam-stalo-bajduzhe-na-nebezpeku/
11. Viber виявився найпопулярнішим месенджером серед українців - дослідження [Електронний ресурс] // Media sapiens. Режим доступу до ресурсу: https://ms.detector.media/mediadoslidzhennya/post/20836/2018-03-22-viber-vyyavyvsya-naypopulyarnishym-mesendzherom-sered-ukraintsiv-doslidzhennya/
12. Кількість користувачів смартфонів в Україні збільшилася до 85% — дослідження [Електронний ресурс] // Media sapiens. Режим доступу до ресурсу: https://ms.detector.media/mediadoslidzhennya/post/21573/2018-08-03-kilkist-korystuvachiv-smartfoniv-v-ukraini-zbilshylasya-do-85-doslidzhennya/#:~:text=%D0%97%D0%B0%20%D1%97%D1%85%20%D0%B4%D0%B0%D0%BD%D0%B8%D0%BC%D0%B8%2C%2085%25%20%D1%83%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D1%86%D1%96%D0%B2,%D0%BC%D1%96%D1%81%D1%82%20%2D%2054%2D55%25.
13. Україна - у лідерах діджиталізації: ми найбільше розраховуємося смартфоном [Електронний ресурс] // Уніан. Режим доступу до ресурсу: https://www.unian.ua/economics/finance/ukrajina-uviyshla-u-svitovi-lideri-za-kilkistyu-bezkontaktnih-oplat-gadzhetami-12362256.html
14. Медіаспоживання українців в умовах повномасштабної війни. [Електронний ресурс] // Опора. Режим доступу до ресурсу: https://oporaua.org/polit_ad/mediaspozhivannia-ukrayintsiv-v-umovakh-povnomasshtabnoyi-viini-opituvannia-opori-24068

15. Укриття [Електронний ресурс] // dovidka.info. Режим доступу до ресурсу: https://dovidka.info/ukryttya/
16. Проблема бомбосховищ та укриттів в Києві: все дуже серйозно [Електронний ресурс] // Україна кримінальна. Режим доступу до ресурсу: https://cripo.com.ua/vojna-s-rf/problema-bomboshovyshh-ta-ukryttiv-v-kyyevi-vse-duzhe-serjozno/
17. Нове у застосунку «Повітряна тривога»: налаштування гучності, нульовий трафік та окремий Telegram-канал [Електронний ресурс] // Ajax. Режим доступу до ресурсу: https://ajax.systems/ua/blog/air-alert-telegram-channel/
18. Найчастіші запитання [Електронний ресурс] // Повітряна тривога. Режим доступу до ресурсу: https://www.ukrainealarm.com/faq
19. Головна | Повітряна тривога [Електронний ресурс] // Повітряна тривога. Режим доступу до ресурсу: https://www.ukrainealarm.com/
20. Українські фахівці створили онлайн-карти для відстеження повітряних тривог одночасно в усіх областях [Електронний ресурс] // Суспільне. Режим доступу до ресурсу: https://suspilne.media/222147-ukrainski-fahivci-stvorili-onlajn-karti-dla-vidstezenna-povitranih-trivog-odnocasno-v-usih-oblastah/
21. Телеграм-бот "Укриття!" з повною базою укриттів по Україні [Електронний ресурс] // stfalcon. Режим доступу до ресурсу: https://stfalcon.com/uk/blog/post/telegram-bot-baza-ukryttiv
22. The Evolution of GIS [Електронний ресурс] // University of Southern California. Режим доступу до ресурсу: https://gis.usc.edu/blog/the-evolution-of-gis/
23. The Remarkable History of GIS: The Evolution [Електронний ресурс] // SpatialPost. Режим доступу до ресурсу: https://www.spatialpost.com/history-of-gis/
24. About & Help [Електронний ресурс] // Nominatim. Режим доступу до ресурсу: https://nominatim.openstreetmap.org/ui/about.html

25. Overpass API [Електронний ресурс] // OpenStreetMap Wiki. Режим доступу до ресурсу: https://wiki.openstreetmap.org/wiki/Overpass_API
26. E. W. Dijkstra: A note on two problems in connexion with graphs. In Numerische Mathematik, 1 (1959), S. 269–271.
27. Moore, E.F., (1959), The shortest path through a maze, pp. 285-292 in Proceedings of an International Symposium on the Theory of Switching (Cambridge, Massachusetts, 2-5 April, 1957), Harvard University Press, Cambridge.
28. Dijkstra's algorithm [Електронний ресурс] // GIS Wiki. Режим доступу до ресурсу: https://wiki.gis.com/wiki/index.php/Dijkstra%27s_algorithm
29. Dijkstra Algorithm [Електронний ресурс] // RPubs. Режим доступу до ресурсу: https://rpubs.com/yogesh665/DJA#:~:text=GPS%20navigation%20systems%3A%20Dijkstra%27s%20algorithm,%2C%20walking%2C%20or%20public%20transportation.
30. An overview of Google Maps Algorithms [Електронний ресурс] // LinkedIn. Режим доступу до ресурсу: https://www.linkedin.com/pulse/overview-google-maps-algorithms-shanmugapriya-thangavel/
31. Bellman-Ford Algorithm [Електронний ресурс] // Stanford University. Режим доступу до ресурсу: https://web.stanford.edu/class/archive/cs/cs161/cs161.1168/lecture14.pdf
32. Running Time Analysis of DFS [Електронний ресурс] // Hong Kong University of Science and Technology. Режим доступу до ресурсу: https://home.cse.ust.hk/~dekai/271/notes/L06/L06.pdf
33. Applications, Advantages and Disadvantages of Breadth First Search (BFS) [Електронний ресурс] // GeeksForGeeks. Режим доступу до ресурсу: https://www.geeksforgeeks.org/applications-of-breadth-first-traversal/
34. Software Requirements [Електронний ресурс] // University of Edinburgh. Режим доступу до ресурсу:

https://www.inf.ed.ac.uk/teaching/courses/ip/CS2Ah0405-SoftwareRequirements.pdf