НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Мобільний застосунок для самоорганізації та взаємодопомоги співвітчизників за кордоном»

Виконав:
студент IV курсу, групи КП-92
Проценко Андрій Павлович _____

Керівник:
Доцент кафедри ЕІ, к.т.н.,
Вунтесмері Юрій Володимирович _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Зав. кафедри СПіСКС, д-р. т. н.,
Романкевич Віталій Олексійович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Проценку Андрію Павловичу

1. Тема проєкту «Мобільний застосунок для самоорганізації та взаємодопомоги співвітчизників за кордоном», керівник проєкту Вунтесмері Юрій Володимирович, доцент, к.т.н., затверджені наказом по університету від «31» травня 2023 р. № 2107-с

2. Термін подання студентом проєкту «16» червня 2023 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - огляд існуючих програмних рішень;
   - обґрунтування обраних засобів розробки програмного забезпечення;
   - розбір розробленого мобільного застосунку;
   - аналіз розробленого програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
   - структура сутностей та баз даних (креслення);
   - функціональність застосунку (креслення);
   - алгоритм роботи застосунку (плакат);
   - дерево проблем проєктом (плакат).

6. Консультанти розділів проєкту.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	15.11.2022	
2.	Розроблення та узгодження технічного завдання	28.11.2022	
3.	Розроблення структури мобільного застосунку	17.12.2022	
4.	Підготовка матеріалів першого розділу дипломного проєкту	05.02.2022	
5.	Розроблення дизайну застосунку та графічних елементів	14.02.2023	
6.	Підготовка матеріалів другого розділу дипломного проєкту	20.02.2023	
7.	Програмна реалізація застосунку	10.03.2023	
8.	Тестування застосунку	06.04.2023	
9.	Підготовка матеріалів третього розділу дипломного проєкту	12.03.2023	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	17.05.2023	
11.	Підготовка графічної частини дипломного проєкту	26.05.2023	
12.	Оформлення документації дипломного проєкту	28.05.2023	

Студент Андрій ПРОЦЕНКО

Керівник проєкту Юрій ВУНТЕСМЕРІ

АНОТАЦІЯ

Даний дипломний проєкт присвячений розробленню мобільного застосонку для самоорганізації та взаємодопомоги співвітчизників за кордоном.

У роботі виконано порівняльний аналіз існуючих рішень можливої взаємодії українців за кордоном між собою, проаналізовано методи для найкращої побудови взаємопомочі у локальній спільноті, обґрунтовано вибір технологій та допоміжних бібліотек серверної та клієнтської частин для реалізації даного мобільного застосунку. Розроблений застосунок дає нашим співвітчизникам можливість дізнаватися про допомогу поруч, знаходити корисну інформацію через інших українців, які вже пробули якийсь час за кордоном та легко влитися у нове ком'юніті. Також українці з досвідом можуть самі допомогати іншим співвітчизникам не тільки словом, а і ділом.

В даному проєкті розроблено та досліджено: архітектуру серверної та клієнтської частини мобільного застосунку, створення нативного інтерфейсу, створення онлайн каналів комунікації між людьми, а також графічні елементи та дизайн мобільних застосунків.

ABSTRACT

This diploma project is dedicated to the development of a mobile application for self-organization and mutual assistance among fellow countrymen abroad.

The work includes a comparative analysis of existing solutions for potential interaction among Ukrainians abroad, an analysis of methods for establishing effective mutual support within a local community. The choice of technologies and auxiliary libraries for implementing this mobile application on both the server and client sides is substantiated. The developed application provides our fellow countrymen with the opportunity to find help nearby, access useful information through other Ukrainians who have already spent some time abroad, and easily integrate into a new community.

In this project, the architecture of the server and client parts of the mobile application, the creation of a native interface, the establishment of online communication channels between people, as well as the graphic elements and design of mobile applications have been developed and researched.

ДП.045440-01-90. Мобільний застосунок для самоорганізації та взаємодопомоги співвітчизників за кордоном. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045490-02-91	Мобільний застосунок	5	
	для самоорганізації та		
	взаємодопомоги		
	співвітчизників за		
	кордоном. Технічне		
	завдання		
ДП.045490-03-81	Мобільний застосунок	83	
	для самоорганізації та		
	взаємодопомоги		
	співвітчизників за		
	кордоном. Пояснювальна		
	записка		
ДП.045490-04-51	Мобільний застосунок	4	
	для самоорганізації та		
	взаємодопомоги		
	співвітчизників за		
	кордоном. Програма та		
	методика тестування		
ДП.045490-05-34	Мобільний застосунок	12	
	для самоорганізації та		
	взаємодопомоги		
	співвітчизників за		
	кордоном. Керівництво		
	користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045490-06-99	Мобільний застосунок	1	
	для самоорганізації та		
	взаємодопомоги		
	співвітчизників за		
	кордоном.		
	Функціональність		
	застосунку. UML-діаграма		
	прецедентів		
ДП.045490-07-99	Мобільний застосунок	1	
	для самоорганізації та		
	взаємодопомоги		
	співвітчизників за		
	кордоном. Структура		
	сутностей та баз даних.		
	ER-діаграма		
ДП.045490-08-98	Мобільний застосунок	1	
	для самоорганізації та		
	взаємодопомоги		
	співвітчизників за		
	кордоном. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ САМООРГАНІЗАЦІЇ ТА ВЗАЄМОДОПОМОГИ СПІВІТЧИЗНИКІВ ЗА КОРДОНОМ

Технічне завдання

ДП.045490-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Юрій ВУНТЕСМЕРІ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____Андрій ПРОЦЕНКО

2022

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Мобільний застосунок для самоорганізації та взаємодопомоги співвітчизників за кордоном.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості інструмента забезпечення взаємодії та допомоги між нашими співвітчизниками за кордоном.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Мобільний застосунок повинен забезпечувати такі основні функції:

1. можливість авторизації та реєстрації користувачів за одну з трьох ролей: Реципієнт, Помічник та Волонтер;
2. можливість переглядати та редагувати дані про свій обліковий запис для усіх користувачів;
3. доступ до перегляду мапи та міток поруч на ній для усіх користувачів;
4. можливість зберігати мітки у список «обраних» для Реципієнта та Помічника;

5. можливість створення, редагування та видалення міток на мапі для Помічника та Волонтера;
6. доступ до публічних та приватних чатів для усіх користувачів;
7. можливість створювати публічні та приватні чати для усіх користувачів.

Розробку виконати за допомогою мови програмування JavaScript у середовищі Node.js та фреймворку Express.js для серверної частини мобільного застосунку, мови програмування Dart та фреймворку Flutter для клієнтської частини та з використанням СУБД MongoDB та PostgreSQL.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Структура сутностей та баз даних. ER-діаграма»;
   - «Функціональність застосунку. UML-діаграма прецедентів».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою проєкту..... 15.11.2022
Розроблення та узгодження технічного завдання 28.11.2022
Розроблення структури мобільного застосунку 17.12.2022
Розроблення дизайну застосунку та графічних елементів..... 14.02.2023
Програмна реалізація застосунку..... 10.03.2023
Тестування застосунку..... 06.04.2023
Підготовка матеріалів текстової частини проєкту 17.05.2023
Підготовка матеріалів графічної частини проєкту..... 26.05.2023
Оформлення технічної документації проєкту 28.05.2023

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Євгенія СУЛЕМА
“___” _____ 2023 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ САМООРГАНІЗАЦІЇ ТА ВЗАЄМОДОПОМОГИ СПІВВІТЧИЗНИКІВ ЗА КОРДОНОМ

Пояснювальна записка

“ПОГОДЖЕНО”

Керівник проєкту:
_____Юрій ВУНТЕСМЕРІ

Нормоконтроль:
_____Микола ОНАЙ

Виконавець:
_____Андрій ПРОЦЕНКО

2023

ЗМІСТ

СПИСОК АБРЕВІАЦІЙ, ВИЗНАЧЕНЬ ТА ТЕРМІНІВ

ПЗ – програмне забезпечення.

СУБД – система управліннями базами даних.

Вебсервіси – рограмна система, що ідентифікується URI, і публічні інтерфейси та прив'язки якої визначені та описані мовою XML.

Meta Platforms – американська корпорація конгломерату соціальних мереж, раніше мала назву Facebook.

Facebook – соціальна мережа, що дозволяє користувачам спілкуватися з друзями, обмінюватися повідомленнями, ділитися фотографіями та відео, а також отримувати оновлення від компаній та медіа.

ОС – операційна система.

Windows – ОС, розроблена компанією Microsoft.

Linux – вільна та відкрита ОС, заснована на ядрі Linux.

Android – ОС для мобільних пристроїв розроблена компанією Google.

iOS – ОС розроблена компанією Apple, яка працює на мобільних пристроях.

macOS – це операційна система, розроблена компанією Apple, яка призначена для роботи на комп'ютерах Mac.

Вебверсія – версія програмного забезпечення або сервісу, яка доступна через веббраузер і не вимагає установки застосункового програмного забезпечення на пристрій.

Мобільна версія – адаптована версія ПЗ або вебсайту, яка оптимізована для використання на мобільних пристроях.

Інтерфейс – точка взаємодії між користувачем та системою.

Месенджер – це програмне забезпечення або сервіс, який дозволяє користувачам обмінюватися текстовими повідомленнями, медіа-файлами та відеозв'язком з іншими користувачами через мережу Інтернет.

Telegram – месенджер і соціальна платформа.

Viber – це месенджер та сервіс голосового та відеозв'язку.

Шифрування повідомлень – захищення вмісту повідомлень від несанкціонованого доступу шляхом перетворення їх у нерозбірливий текст.

Бекенд (backend) – складова ПЗ, яка виконує обробку та збереження даних, відповідає на запити від клієнтських пристроїв і надає функціональність через мережу для взаємодії з ними.

Фронтенд (frontend) – складова ПЗ, яку бачить та з якою взаємодіє користувач, що включає графічний інтерфейс, дизайн та функціональність.

Асинхронність – підхід у програмуванні, де операції розбиваються на незалежні завдання та виконуються паралельно, дозволяючи продовжувати роботу інших частин програми без очікування завершення певної операції.

ООП – об'єктно-орієнтоване програмування.

Кросплатформеність – властивість ПЗ, що дозволяє його використання та роботу на різних ОС або пристроях без необхідності значних змін коду.

JavaScript – високорівнева мова програмування, використовується для розробки різноманітних програм, включаючи веб, мобільні та серверні застосунки.

Python – високорівнева мова програмування, відзначається широким спектром вбудованих функцій та багатофункціональними бібліотеками, використовується для розробки вебзастосунків, наукових обчислень, штучного інтелекту, тощо.

Kotlin – мова програмування, працює на платформі Java та надає розширені можливості для розробки Android-застосунків, забезпечує безпеку типів, функціональність та високу сумісність з існуючим Java-кодом.

ECMAScript – стандарт мови скриптового програмування, використовується для розробки вебзастосунків та виконання скриптів у браузерах.

Java-аплети – програми написані на Java, можуть виконуватися в контексті вебсторінок і відтворюватися в браузері, надають розширені можливості, такі як графіка, анімація та взаємодія з користувачем.

LiveScript – мова програмування, яка є еволюційним нащадком JavaScript, пропонуючи більш компактний синтаксис, функціональні розширення та покращену читабельність коду.

API – набір правил та протоколів, які дозволяють різним програмам та сервісам взаємодіяти між собою.

React Native – це відкритий фреймворк розробки мобільних застосунків, що базується на JavaScript та React, дозволяє розробникам створювати нативні мобільні застосунків для платформ Android та iOS.

Строга динамічна типізація – властивість мови програмування, при якій типи змінних визначаються автоматично в процесі виконання програми і перевіряються на відповідність правилам типізації під час виконання.

Динамічна семантика – властивість мови програмування, яка визначає, як вирази та інструкції обробляються та оцінюються під час виконання програми.

PYPL PopularitY index – індекс популярності мов програмування, який визначається на основі обсягу пошукових запитів, пов'язаних з вивченням та навчанням певних мов програмування.

JVM (Java Virtual Machine) – віртуальна машина, яка виконує байт-код, створений під час компіляції програм на мові Java.

Нативна підтримка – можливість або функціональність, що вбудована в систему або платформу без потреби в застосункових розширеннях або залежностях.

Система нульової безпеки – підхід до розробки програмного забезпечення, який передбачає, що всі захисні заходи і перевірки повинні бути встановлені та виконуватися за замовчуванням, надійно захищаючи програму від можливих вразливостей.

Node.js – вільне та відкрите середовище виконання JavaScript, дозволяє виконувати JavaScript-код поза веббраузером, забезпечуючи можливість розробки серверних та мережевих застосунків на стороні сервера.

Express.js – легкий та гнучкий фреймворк для розробки серверних застосунків на базі Node.js, надає простий і зрозумілий спосіб створення вебсерверів та роутингу HTTP-запитів.

Socket.io – бібліотека JavaScript, дозволяє зручно реалізовувати двосторонню комунікацію в реальному часі між вебсервером і клієнтами за допомогою технології WebSockets.

WebSockets – протокол для забезпечення двостороннього, постійного з'єднання між клієнтом та сервером, що дозволяє реалізувати інтерактивну комунікацію в реальному часі.

Passport.js – модуль аутентифікації для Node.js, дозволяє зручно управляти процесом аутентифікації користувачів.

Flutter – відкритий фреймворк розробки користувацьких інтерфейсів, дозволяє створювати швидкі та кросплатформенні мобільні застосунки для iOS та Android з використанням єдиного коду.

Swift – мова програмування високого рівня, розроблена компанією Apple, поєднує в собі ефективність та безпеку мови C, експресивність та простоту використання мови Python, а також можливості розробки для платформ iOS, macOS, watchOS та tvOS.

Фреймворк – комплексний набір інструментів, бібліотек та правил, які надають загальну структуру та готові рішення для вирішення типових задач розробки.

Віджет – це невеликий графічний компонент інтерфейсу, що надає користувачам простий та інтуїтивний доступ до функціоналу або інформації в застосунку або на вебсайті.

Skia – високоефективна кросплатформена бібліотека рендерингу, розроблена Google, надає широкий спектр можливостей для візуалізації

графічних об'єктів, рендерингу векторної та растрової графіки, підтримку апаратного прискорення для швидкого відображення зображень на різних платформах.

ORM – об'єктно-реляційне відображення.

Prisma – сучасна ORM, яка працює з SQL та NoSQL базами даних, надаючи автоматичне генерування типів, що забезпечують зв'язок між схемою бази даних і кодом додатку.

MVC – шаблон проєктування програмного забезпечення, який розділяє застосунок на три основні логічні компоненти: модель даних, відображення та контролер для забезпечення чіткого розділу обов'язків.

MVVM – шаблон архітектури програмного забезпечення, який відокремлює логіку додатка від його інтерфейсу, розбиваючи застосунок на три частини: модель (дані), вид (інтерфейс) та ViewModel (посередник між першими двома).

GUI – графічний інтерфейс користувача.

ВСТУП

У світі, де сучасні конфлікти та війни стають частиною життя, проблема переселенців здобуває все більшого значення. З початком активної фази війни між Російською Федерацією та Україною 24 лютого 2022 року, мільйони українців були змушені покинути свої домівки і шукати притулок та безпеку за кордонами своєї країни.

Згідно даним Управління верховного комісара ООН, станом на 18 квітня 2023 року, загальна кількість виїхавших українців перевищує восьмимільйонний рубіж. При цьому, приблизно 45% з них прагнуть оселитися в нових країнах назавжди, в той час як 55% мають намір повернутися на Батьківщину, проте більшість – лише після перемоги.

Відповідаючи на цей виклик, ми прагнемо розробити мобільний застосунок, що стане надійним інструментом самоорганізації та взаємодопомоги українських переселенців у різних країнах світу. Наша мета – створити простір, що сприятиме ефективній комунікації, координації дій та обміну необхідною інформацією, проте з гарантованою безпекою та конфіденційністю переданих даних.

Основні функціональні складові застосунку будуть включати інтерактивну мапу для зручної орієнтації та пошуку потрібних об'єктів чи осіб, а також систему відкритих чатів для своєчасного та оперативного обміну інформацією між користувачами.

Сподіваємося, що наша праця сприятиме зміцненню спільнот українців, які знаходяться за кордоном, і допоможе їм впоратися з викликами, що постають перед ними на шляху адаптації в нових умовах життя.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Проблема яка вирішується ПЗ

З початком активної фази війни Російської Федерації з Україною 24 лютого 2022 року багато українців покинули свої домівки та поїхали шукати притулку та безпеки за кордон. Згідно порталу Управління верховного комісара ООН [1] на 18 квітня 2023 року записана загальна кількість виїхавших українців становить 8174189. Нижче на рис.1.1 наведена мапа європейських країн з загальною кількістю зареєстрованих на даний момент біженців. З них близько 45% хочуть залишитися там назавжди, а з 55% що планують повернутися на Батьківщину, 82% хочуть зробити це тільки після перемоги [2].

Рис. 1.1. Мапа розподілення біженців по країнах

Згідно обмеженому опитуванню Центра Розумкова з українців що тимчасово перебувають за кордоном постійну роботу там знайшли 12%, така ж кількість українців знайшла тимчасову роботу та підробітки, дистанційну роботу мають менше 10% опитуваних. Решта розраховує на соціальні виплати у країні де вони оформили статус біженця (30-50%) або/та на соціальні виплати з України (2%). Матеріальне становище 60% опитаних співвітчизників оцінюють як «мінімальне, вистачає для забезпечення себе

їжею та придбання інших необхідних дешевих речей», 12% рахують кожну копійку, іноді грошей не вистачає навіть на їжу, 21% оцінюють своє становище як оптимальне [3].

Зі статистики наведеної вище можемо зробити висновок, що далеко не усі з них мають можливість жити там на власні гроші, мають родичів або друзів за кордоном які живуть там довгий час та можуть допомогти з адаптацією або попередньо розпланували свій переїзд та не матимуть жодних проблем під час релокації.

Перетинаючи кордон наші співвітчизники часто можуть постати перед іншими проблемами: мовний бар'єр, незнайома країна, відсутність знайомих/родичів у країні до якої вони приїжджають.

Також варто зауважити що на даний момент багато країн скасовують виплати, скорочують кількість програм для переселенців, а безкоштовні сервіси та платформи з наданням різних видів допомог перестають бути активними, як наприклад оця німецька платформа для пошуку безкоштовного проживання у родині німців [4] або аналогічна платформа для Литви. Ці тренди ще раз підкреслюють необхідність існування зручного сервісу для допомоги нашим співвітчизникам за кордоном у такі важкі для нас часи.

Тому метою розроблюваного ПЗ є вирішення проблем українців, які шукають прихистку за кордоном.

1.2. Аналіз існуючих рішень

Під час даного аналізу знайшли вебзастосунки, які вирішують лише деяку кількість проблем та зазвичай працюють як чати між користувачами у особистих повідомленнях або у коментарях постів у групах. Далі ці рішення були проаналізовані.

1.2.1. *Facebook*

Facebook – це найбільша соціальна мережа, яка за останніми даними має 2.96 мільярди користувачів по всьому світу [5]. Вона доступна на ОС Windows, Linux, Android, iOS та має вебверсію.

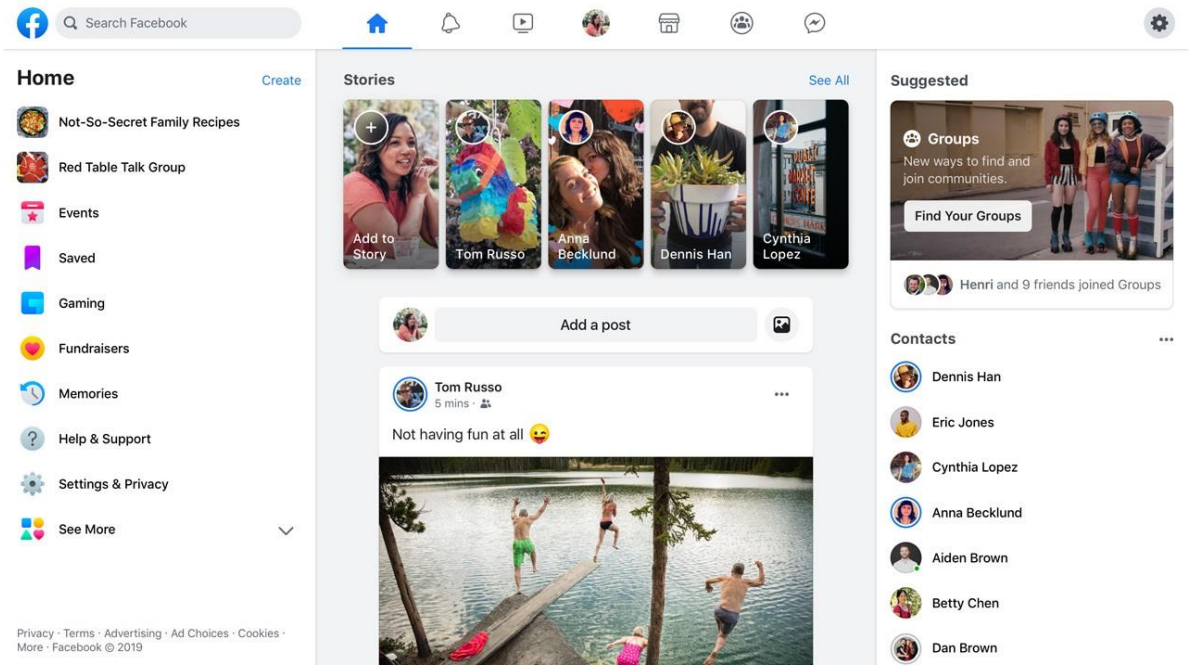


Рис. 1.2. Інтерфейс соціальної мережі Facebook

До основних функцій цього рішення відносяться:

- можливість обмінюватися повідомленнями;
- можливість знаходження груп по інтересах;
- інтегрованість соцмережі з іншими продуктами Meta;
- популярність застосунку серед українців;
- низький рівень загрози шахраїв через можливість перегляду активності профілів інших користувачів.

Недоліком Facebook є занадто велика кількість інформації яку користувачу потрібно перебрати щоб знайти необхідні йому дані або писати у публічних групах, чекаючи відповіді яку він може і не отримати через велику завантаженість повідомленнями інших користувачів і таким чином повідомлення користувача може просто загубитися.

1.2.2. *Telegram*

Telegram – месенджер запущений у 2013 році і наразі має 55.2 мільйони активних користувачів на день та 700 мільйонів щомісяця [6]. Він доступний на ОС Windows, Linux, macOS, Android, iOS та має вебверсію. З початком повномасштабного вторгнення став переважним ресурсом отримання новин як і інші месенджери, у зв'язку з чим 60% українців називають його найпопулярнішим новинним джерелом [7].

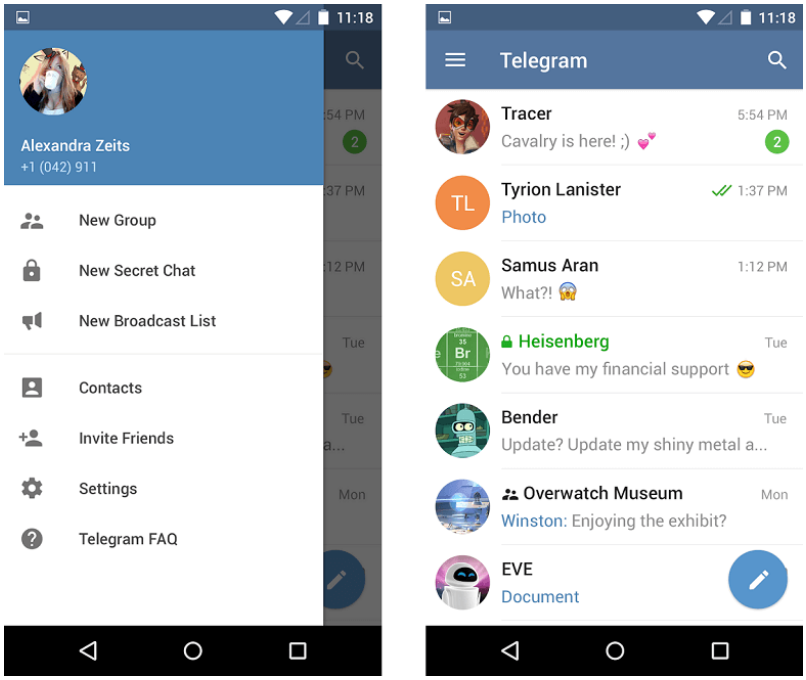


Рис. 1.3. Інтерфейс мобільної версії месенджеру Telegram

До основних функцій цього рішення відносяться:

- зручний та інтуїтивний інтерфейс;
- можливість обмінюватися повідомленнями;
- можливість знаходження груп по інтересах;
- швидкість надіслання та отримання повідомлень;
- можливість здійснювати аудіо- та відеодзвінки безпосередньо у застосунку;
- популярність застосунку в українців.

Недоліком Telegram як і у випадку з Facebook є велика кількість інформації, яку навіть за пошуковим запитом буває складно знайти у публічних чатах або групах, та велика ймовірність загублення повідомлення під купою інших. Також недоліком є поганий захист від шахрайства через анонімність яку гарантує шифрування месенджеру.

1.2.3. *Viber*

Viber – месенджер який був запущений у 2010 році та має 1.3 мільярди зареєстрованих користувачів на 2022 рік та є одним з найпопулярніших месенджерів яким користуються українці [8]. Він доступний на таких ОС, як Android, iOS, Linux, macOS та Windows.

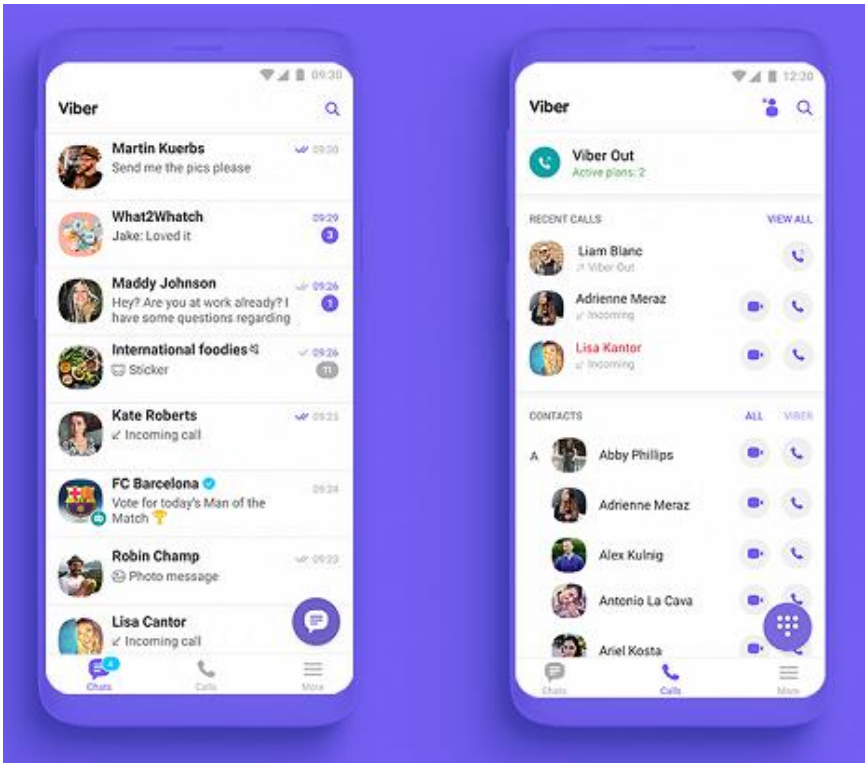


Рис. 1.4. Інтерфейс мобільної версії месенджеру Viber

Перевагами цього застосунку є:

- можливість обмінюватися повідомленнями;
- можливість знаходження груп по інтересах;

- швидкість надіслання та отримання повідомлень;
- можливість здійснювати аудіо- та відеодзвінки безпосередньо у застосунку;
- популярність застосунку в українців.

Недоліками є:

- велика кількість спаму та реклами;
- погана якість зв’язку при використанні відео- та аудіодзвінків за кордон;
- незрозумілий інтерфейс;
- низька захищеність файлів та повідомлень користувачів.

1.3. Опис вимог до розроблюваного ПЗ

Зробивши аналіз проблем з якими стикаються українські біженці приїжджаючі за кордон у пошуках безпеки та стабільності, були визначені головні цілі та бажані результати розроблюваного ПЗ. Також були сформульовані функціональні вимоги, які виглядають таким чином:

1. Українці можуть реєструватися у мобільному застосунку як Реципієнти або Помічники;
2. Благодійні організації та фонди можуть реєструватися як Волонтери;
3. Волонтери можуть створювати, редагувати та видаляти маркери своїх організацій з застосунковою інформацією про послуги які ця організація може надати;
4. Помічники можуть створювати, редагувати та видаляти свої маркери локальної допомоги з застосунковою інформацією про вид допомоги яку вони можуть надати;
5. Реципієнти можуть продивлятися доступні на мапі маркери Волонтерів та Помічників або продивлятися найближчі до них маркери при увімкненій геолокації;

6. Реципієнти можуть писати повідомлення та відповідати на вже створених обговореннях або створювати нові;
7. Помічники можуть писати повідомлення та відповідати на вже створених обговореннях або створювати нові, їх повідомлення позначаються маркером.

2. ОБГРУНТУВАННЯ ОБРАНИХ ЗАСОБІВ РОЗРОБКИ ПЗ

2.1. Вибір мови програмування для розроблення серверної частини

Після аналізу вимог до серверної частини, можемо описати бажані критерії до мови програмування:

- асинхронне виконання коду;
- підтримка ООП;
- швидкість виконання запитів;
- кросплатформеність;

Згідно цих вимог було обрано такі мови програмування: JavaScript, Python, Kotlin.

2.1.1. *JavaScript*

JavaScript (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Компанії мали на меті створити мову, що могла зв'язати різні частини вебсайтів: зображень, Java-аплетів, об'єктної моделі документа. Таким чином була створена LiveScript, яку сьогодні ми знаємо як JavaScript [9].

На травень 2023 року JavaScript є 3 з найбільш популярних мов програмування сьогодення, матеріали по ній шукають 9.52% людей від загальної кількості пошукових запитів по мовах програмування [10].

Перевагами цієї мови програмування є*:*

1. Кросплатформеність: JavaScript дозволяє писати застосунки які працюватимуть на будь-якій ОС при наявності необхідного інтерпретатора.
2. Асинхронність та швидкодія: ця мова заснована на асинхронному програмуванні, що дозволяє ефективно обробляти багато запитів одночасно без блокування потоку виконання. Це особливо корисно для серверної частини

мобільного застосунка, яка часто здійснює взаємодію з базою даних, зовнішніми API та іншими послугами.

3. Масштабованість: JavaScript побудована з урахуванням горизонтальної масштабованості. Можна легко масштабувати свої сервери, додаючи більше екземплярів та розподіляючи навантаження між ними. Це дозволяє обробляти більше запитів та забезпечувати високу продуктивність мобільного застосунка.

Недоліками є:

1. Продуктивність: Мобільні застосунки, написані на JavaScript, можуть бути повільнішими порівняно з нативними застосунками, особливо при використанні важких анімацій або великих даних.
2. Доступ до нативних API: Хоча рішення на базі JavaScript, такі як React Native, надають доступ до більшості нативних API, деякі специфічні для платформи API можуть бути недоступними.
3. Споживання батареї: Застосунки на цій мові можуть використовувати більше енергії порівняно з нативними застосунками.

2.1.2. *Python*

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів [11]. Є найпопулярнішою мовою згідно PYPL PopularitY index та має частку 27.27% на травень 2023 року [10].

Перевагами Python є:

1. Швидкодія та масштабованість: Python дозволяє швидко створювати прототипи та розробляти програми ефективно. Він також має можливості масштабування, що дозволяє розширювати існуючі сервери для обробки більшого обсягу запитів.
2. Багата екосистема та бібліотеки: Ця мова має велику кількість сторонніх бібліотек та модулів, які спрощують розробку серверної частини мобільних застосунків.
3. Переносимість коду: Python є інтерпретованою мовою, тому код може бути легко перенесений між різними платформами без необхідності в компіляції. Це полегшує розгортання та масштабування серверних застосунків.

Недоліками цієї мови є:

1. Високе споживання пам'яті: Python відомий своїм високим споживанням пам'яті порівняно з іншими мовами програмування. Це буде проблемою, якщо необхідно оптимізувати продуктивність або працювати з обсягом даних великого розміру.
2. Обмежена підтримка мобільних платформ: Ця мова не є основною мовою програмування для розробки мобільних застосунків, і має обмежену підтримку на деяких мобільних платформах, таких як iOS або Android.
3. Потреба в управлінні залежностями: Python має свою систему керування залежностями, але іноді можуть виникати проблеми з управлінням версіями бібліотек та їх сумісністю. Потрібно ретельно відслідковувати та оновлювати залежності, щоб уникнути конфліктів або проблем безпеки.

2.1.3. *Kotlin*

Kotlin – статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains. Мова розробляється з 2010 року, публічно представлена в липні 2011 [12]. Займає 13 місце у PYPL PopularitY index з часткою 1.77% [10].

Переваги мови програмування Kotlin:

1. Нативна підтримка Android: Kotlin є пріоритетною мовою для розробки Android-застосунків, вона офіційно підтримується Google.
2. Безпека та надійність: Kotlin покращує безпеку коду завдяки ряду інноваційних функцій, таких як система нульової безпеки, захищені типи, властивості та інші.
3. Функціональні можливості: Ця мова підтримує функціональне програмування, що дозволяє писати більш компактний та експресивний код. Це полегшує роботу з колекціями даних та обробку подій.
4. Підтримка корутинів: Kotlin має вбудовану підтримку корутинів, що дозволяє зручно працювати з асинхронним кодом та робити його більш конкурентоздатним.

Недоліки Kotlin:

1. Розрахована тільки на Android: Ця мова може використовуватися тільки для створення застосунків на Android, що сильно обмежує обсяг користувачів, які можуть бути залучені до користування застосунком.
2. Вимоги до ресурсів: Kotlin, порівняно з деякими іншими мовами програмування, може вимагати більше ресурсів для виконання. Його виконання може бути повільнішим та споживати більше пам'яті порівняно з іншими мовами згаданими раніше, особливо при обробці великого обсягу даних або при високих навантаженнях.

2.1.4. *Висновки*

Після детального дослідження різних мов програмування та їхньої сумісності з зазначеними потребами проєкту, було вирішено зупинити вибір на JavaScript для реалізації серверної частини цього мобільного застосунку. Ця мова програмування була обрана завдяки її широкому спектру можливостей, що найкраще відповідають потребам даного проєкту. Для застосунку було обрано платформу Node.js, яка дозволяє запускати JavaScript-код на сервері [13]. Фреймворк Express.js також було використано для конфігурації серверної середи [14]. Бібліотеку Socket.io було використано для реалізації функціоналу обміну повідомленнями між користувачами [15]. Passport.js було використано для автентифікації користувачів та їхньої реєстрації [16].

2.2. Вибір програмних засобів для розроблення клієнтської частини

В останні роки стали популярними мобільні застосунки, і для їх розробки на клієнтській частині використовуються різні мови програмування. Однак, основними мовами для створення сучасних мобільних застосунків є Flutter, Swift і Kotlin.

2.2.1. *Flutter*

Flutter – це молодий, але дуже швидко зростаючий фреймворк, який дозволяє розробляти нативні мобільні застосунки для платформ Android та iOS з одним кодом бази [17]. Він розроблений компанією Google що дає можливість інтеграції сервісів цієї компанії та використовує мову програмування Dart [18].

Перевагами цього фреймворку є:

1. Кросплатформність: Flutter дозволяє розробляти мобільні застосунки для платформ Android та iOS з одним кодом бази.
2. Гарний інтерфейс: Flutter надає багатий набір вбудованих віджетів та гнучкі можливості для створення красивого та

інтерактивного користувацького інтерфейсу. Ними можна легко втілити складні дизайни, анімації та переходи.

3. Швидкодія: Flutter використовує власний двигун рендерингу Skia, що дозволяє досягти високої продуктивності та швидкодії. Він дозволяє забезпечити плавність і швидкодію інтерфейсу.

Недоліки:

1. Розмір застосунку: Flutter включає в себе власну бібліотеку та двигун рендерингу, розмір кінцевого застосунку може бути більшим порівняно з застосунками, розробленими на нативних мовах.
2. Наявність нативних функцій: хоча Flutter надає багатий набір вбудованих віджетів, в ньому може бути обмежена підтримка для деяких специфічних функцій, які можуть бути доступні тільки на рівні нативної розробки.

2.2.2. *Swift*

Swift є мовою програмування, розробленою спеціально для платформи iOS [19]. Вона має чистий та експресивний синтаксис, що полегшує розробку мобільних застосунків для цієї платформи. Swift забезпечує високу продуктивність, масштабованість та безпеку. Ця мова займає 9 місце по популярності з часткою у 2.3% на травень 2023 року [10].

Переваги цієї мови:

1. Ефективність та швидкодія: Swift створена з орієнтацією на високу продуктивність. Вона використовує передові оптимізації та механізми компіляції, що дозволяють застосункам працювати швидко та ефективно.
2. Безпека та стабільність: Swift була розроблена з урахуванням безпеки, вона має систему типізації, що допомагає уникнути багатьох типових помилок. Вона також підтримує необов’язкові типи, які дозволяють обробляти значення, які можуть бути

нульовими. Це сприяє створенню більш стабільних та безпечних застосунків.

3. Читабельність та експресивність: Swift має сучасний та лаконічний синтаксис, що робить його дуже читабельним та зрозумілим. Вона підтримує високий рівень експресивності, що полегшує написання зрозумілого та зручного для розробника коду.

Недоліки:

1. Обмежена підтримка платформ: Swift підтримується тільки на платформах iOS, macOS, watchOS та tvOS.
2. Несталий стандарт: Swift ще в процесі розвитку, і стандарт мови може змінюватися з кожним оновленням, що може призвести до несумісності між різними версіями Swift та проблем зі сумісністю зі старішими версіями.

2.2.3. *Kotlin*

Переваги цієї мови для написання клієнтської частини:

1. Безпека та стабільність: Kotlin надає безпечний типізований підхід до програмування, що допомагає уникнути багатьох типових помилок. Вона має систему перевірки типів на етапі компіляції, що дозволяє виявляти помилки на ранніх етапах розробки.
2. Зручний синтаксис: Kotlin має чистий, зрозумілий та експресивний синтаксис, що полегшує розробку. Вона підтримує багато корисних функцій, таких як розширення функціональності класів, нульові оператори та інші, що полегшують писання чистогота зрозумілого коду.
3. Головним недоліком цієї мови для клієнтської частини є неможливість автономної розробки на iOS платформи, це буде

можливо тільки при використанні застосункових фреймворків та призведе до погіршення загальної читабельності коду.

2.2.4. *Висновки*

Після детального аналізу, єдиним підходящим з розгляданих варіантів є Flutter через свою кросплатформеність та можливості легкого використання сервісів Google через вбудовану інтеграцію. Також Flutter є найкращим фреймворком у плані інструментів для створення візуальної складової мобільного застосунку.

2.3. Вибір СУБД

Розподіл баз даних для мобільних застосунків також можна класифікувати за типом зберігання та представлення даних. Основні категорії включають реляційні бази даних (SQL) та нереляційні бази даних (NoSQL). Розуміння різниці між ними допомагає вибрати найкращу опцію для вашого мобільного застосунку.

Реляційні бази даних (SQL) використовуються для зберігання структурованих даних з визначеними залежностями між ними. Вони забезпечують гнучкість у взаємодії з різними типами даних та створення зв'язків між ними завдяки використанню статичної схеми.

Переваги реляційних баз даних включають:

1. Швидка робота з моделями даних різних типів та можливість створення відносин між ними.
2. Підтримка запитів за допомогою мови SQL, що спрощує вибірку та модифікацію даних.
3. Вища цілісність та захищеність даних, оскільки окремі операції з даними можуть бути інкапсульовані в рамках транзакцій.

Нереляційні бази даних (NoSQL), навпаки, призначені для зберігання неструктурованих даних, які можуть бути представлені у вигляді документів або пар ключ-значення.

Основні переваги нереляційних баз даних включають:

1. Висока швидкість обробки даних одного типу без необхідності установки складних залежностей.
2. Відсутність прив'язки до схеми даних, що дозволяє додавати, видаляти та змінювати поля без необхідності ускладнювати структуру даних.
3. Підтримка горизонтального масштабування, дозволяючи створювати систему баз даних з кількох екземплярів.

Враховуючи ці переваги, реляційні бази даних найкраще підходять для систем, які працюють зі структурованими даними або вимагають ізоляції та пріоритезації запитів до бази даних через використання транзакцій. З іншого боку, нереляційні бази даних є оптимальним вибором для обробки великих обсягів даних, які не обов'язково повністю структуровані.

Так як у розроблюваному мобільному застосунку будуть використані різні сутності, як структуровані так і неструктуровані, тому під час створення цього ПЗ буде застосовано обидва види БД.

2.3.1. *MySQL*

MySQL – вільна система керування реляційними базами даних, яка була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Має чудову підтримку з боку різноманітних мов програмування, завдяки чому є одною з найпопулярніших СКБД [20].

Її переваги:

1. Широкі можливості: MySQL підтримує багато функцій та операцій, включаючи складні запити, індексування, транзакції, відновлення даних та багато іншого.
2. Висока швидкодія: ця СКБД відома своєю високою продуктивністю та швидкодією обробки запитів, що робить її

привабливим вибором для мобільних застосунків з великим обсягом даних або високими вимогами до продуктивності.

3. Масштабованість: вона може бути легко масштабована вгору шляхом додавання нових серверів або розподілу навантаження на кілька реплік бази даних.

Недоліки:

1. Обмежена масштабованість горизонтального розширення: при масштабуванні MySQL може стикатися з обмеженнями та складнощами, пов'язаними з реплікацією та синхронізацією даних між серверами.
2. Вразливість до високого навантаження запису: якщо мобільний застосунок має високі вимоги до швидкості запису даних, ця СУБД може стати бутланеком через обмеження на швидкість запису на диск.
3. Обмежена географічна реплікація: якщо мобільний застосунок потребує реплікації даних на різних географічних місцях для поліпшення доступності та надійності, може потребувати застосункових рішень для реплікації даних на віддалених серверах.

2.3.2. *PostgreSQL*

PostgreSQL – об'єктно-реляційна система керування базами даних з відкритим кодом. Прототип був розроблений в Каліфорнійському університеті Берклі в 1987 році під назвою POSTGRES, після чого активно розвивався і доповнювався У 1996 році назву програмного продукту змінили на PostgreSQL, випустивши при цьому вихідну версію 6.0. Відтоді продовження підтримки та розробки PostgreSQL відбувається за допомогою групи експертів у сфері баз даних, які приєдналися до цього проєкту [21].

Її перевагами є:

1. Розширена функціональність: PostgreSQL надає багато розширених функцій, таких як підтримка геоданих, повний текстовий пошук, географічні запити та багато іншого. Це дозволяє створювати складні та потужні мобільні застосунки з багатограною функціональністю.
2. Висока надійність та цілісність даних: вона має вбудовану підтримку транзакцій, реплікації та механізмів відновлення даних. Це забезпечує надійність та захищеність даних в мобільному застосунку.
3. Розширені можливості розширень: ця СУБД має гнучку систему розширень, що дозволяє створювати та використовувати власні розширення для додавання нової функціональності до бази даних.

Недоліками:

1. Вимоги до ресурсів: вимагає більше ресурсів, таких як пам'ять та процесорний час, порівняно з деякими іншими СУБД. Це може вплинути на продуктивність мобільного застосунку, особливо якщо він працює на обмежених пристроях з обмеженими ресурсами.
2. Обмежена підтримка горизонтального масштабування: в порівнянні з деякими нереляційними базами даних, PostgreSQL має обмежену підтримку горизонтального масштабування.

2.3.3. *MongoDB*

MongoDB – документо-орієнтована система керування базами даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів [22].

Перевагами є:

1. Швидкість та продуктивність: MongoDB має високу швидкість доступу до даних, особливо при великих обсягах даних. Вона підтримує індексацію, реплікацію та географічне розташування даних, що сприяє покращенню продуктивності мобільного застосунку.
2. Гнучкість запитів: ця БД має потужну та гнучку мову запитів, яка дозволяє виконувати різноманітні операції з даними, включаючи складні запити, агрегацію та групування.

Недоліками:

1. Відсутність підтримки транзакцій: MongoDB не має повної підтримки транзакцій, що може бути проблемою для деяких мобільних застосунків, які вимагають гарантії цілісності даних та атомарних операцій.
2. Складність налаштування та адміністрування: MongoDB має ряд застосункових конфігураційних опцій та параметрів, які можуть змінювати його поведінку.

2.3.4. *Висновки*

Зважаючи на проаналізовані переваги та недоліки різних систем управління базами даних (СУБД), було вирішено вибрати дві основні СУБД для цього проєкту: реляційну PostgreSQL та нереляційну MongoDB. PostgreSQL була обрана через свою надійність, цілісність даних і розширений функціонал, тоді як MongoDB була вибрана через свою швидкість, високу продуктивність та гнучкість у формулюванні запитів. Для обробки взаємодії між цими базами даних та серверною частиною нашого застосунку, ми використали об'єктно-реляційний відображувач (ORM) Prisma. Prisma – це сучасний ORM, який значно полегшує роботу з базами даних у JavaScript. Prisma дозволяє легко виконувати CRUD-операції, здійснювати запити до бази даних, а також обробляти транзакції.

Prisma підтримує обидві вибрані нами СУБД – PostgreSQL і MongoDB, що робить її ідеальним інструментом для розроблюваного застосунку. Більше того, Prisma дозволяє ефективно працювати з різними базами даних одночасно в одному проєкті, що дозволяє нам максимально використовувати переваги обох баз даних [23].

3. ОГЛЯД РОЗРОБЛЕНОГО МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Загальний опис застосунку

Дана система була виконана у вигляді мобільного застосунку, згідно функціональним вимогам у першому розділі та розроблена за допомогою фреймворків та мов описаних у другому розділі. Вона ділиться на 2 частини, а саме клієнтську та серверну.

Клієнтська частина у цій системі забезпечує:

1. Дає можливість взаємодії між клієнтами та системою і її функціоналом.
2. Виводить необхідні користувачу дані у зрозумілій та необтяженій формі.
3. Надає можливість бачити мітки та взаємодіяти з мапою.

Серверна частина відповідає за:

1. Обробку запитів отриманих від клієнтів через клієнтську частину.
2. Роботу з базами даних.
3. Надсилання повідомлень для сповіщення користувачів про зміну у збережених мітках або нові повідомлення від збережених публічних чатах/користувачів.

Функціонал системи представимо як ключові сценарії користування мобільним застосунком Реципієнтом:

Сценарій 1: «Реєстрація або авторизація Реципієнта в системі»

1. Користувач заходить у застосунок.
2. Користувач пропонується форма авторизації та посилання на форму реєстрації.
3. Користувач вводить свої дані та обирає роль Реципієнта.
4. Користувач активує обліковий запис через лист-підтвердження з посиланням, який приходить йому на вказану пошту.

Сценарій 2: «Перегляд найближчих міток на мапі та додавання їх у “обрані”»

1. Реципієнт дозволяє застосунку використовувати поточну геолокацію.
2. Реципієнт може переглядати найближчі мітки з допомогою на мапі.
3. Реципієнт може додатии цікаві йому мітки до обраного.
4. Застосунок надішле push-сповіщення при початку роботи обраної мітки.

Сценарій 3: «Перегляд Реципієнтом міток на мапі за тегами»

1. Реципієнт відкриває строку пошуку.
2. Реципієнт вводить необхідні йому теги для пошуку міток у радіусі.
3. Реципієнт може переглянути мітки з шуканими тегами.
4. Реципієнт може додати знайдену мітку до обраного для отримання push-сповіщення про початок її роботи.

Сценарій 4: «Взаємодія з іншими користувачами застосунку через публічні чати»

1. Реципієнт переходить на сторінку публічних чатів.
2. Реципієнт може переглядати публічні чати.
3. Реципієнт може створювати свої чати, вони позначатимуться білою рамкою.
4. Реципієнт може писати повідомлення у будь-яких чатах.
5. Реципієнт може видаляти свої повідомлення.

Сценарій 5: «Взаємодія з іншими користувачами через приватні повідомлення»

1. Реципієнт може писати у приватні повідомлення іншим користувачам, за умови що у отримувача повідомлення відкритий профіль.

2. Реципієнт може отримувати повідомлення від інших користувачів у приватні повідомлення за тієї ж умови.

Функціонал системи представимо як ключові сценарії користування мобільним застосунком Помічником:

Сценарій 1: : «Реєстрація або авторизація Помічника в системі»

1. Користувач заходить у застосунок.
2. Користувач пропонується форма авторизації та посилання на форму реєстрації.
3. Користувач вводить свої дані та обирає роль Помічника.
4. Користувач активує обліковий запис через лист-підтвердження з посиланням, який приходить йому на вказану пошту.

Сценарій 2: «Перегляд найближчих міток на мапі та додавання їх у "обрані"»

1. Помічник дозволяє застосунку використовувати поточну геолокацію.
2. Помічник може переглядати найближчі мітки з допомогою на мапі.
3. Помічник може додати цікаві йому мітки до обраного.
4. Застосунок надішле push-сповіщення при початку роботи обраної мітки.

Сценарій 3: «Перегляд Помічником міток на мапі за тегами»

1. Помічник відкриває строку пошуку.
2. Помічник вводить необхідні йому теги для пошуку міток у радіусі.
3. Помічник може переглянути мітки з шуканими тегами.
4. Помічник може додати знайдену мітку до обраного для отримання push-сповіщення про початок її роботи.

Сценарій 4: «Створення Помічником мітки на мапі»

1. Помічник обирає доступну йому функцію "Створити мітку" на меню мапи.

2. Помічник заповнює форму потрібну для створення мітки, де вказує усю необхідну інформацію: адресу, вид допомоги, час її надання.
3. Якщо форма заповнена коректно, синя мітка з’явиться на мапі.
4. Якщо форма заповнена некоректно, застосунок запропонує Помічнику доповнити/перевірити на правильність надану інформацію.

Сценарій 5: «Взаємодія з іншими користувачами застосунку через публічні чати»

1. Помічник переходить на сторінку публічних чатів.
2. Помічник може переглядати публічні чати.
3. Помічник може створювати свої чати, вони позначатимуться синьою рамкою.
4. Помічник може писати повідомлення у будь-яких чатах.
5. Помічник може видаляти свої повідомлення.
6. При відсутності будь-якої активності у публічному чаті протягом місяця, він буде автоматично видалений.

Сценарій 6: «Взаємодія з іншими користувачами через приватні повідомлення»

1. Помічник може писати у приватні повідомлення іншим користувачам, за умови що у отримувача повідомлення відкритий профіль.
2. Помічник може отримувати повідомлення від інших користувачів у приватні повідомлення за тієї ж умови.

Функціонал системи представимо як ключові сценарії користування мобільним застосунком Волонтером:

Сценарій 1: «Реєстрація або авторизація Волонтера в системі»

1. Користувач заходить у застосунок.
2. Користувач пропонується форма авторизації та посилання на форму реєстрації.

3. Користувач вводить свої дані та обирає роль Волонтера.
4. Користувач активує обліковий запис через лист-підтвердження з посиланням, який приходить йому на вказану пошту.

Сценарій 2: «Перегляд Волонтером міток на мапі за тегами»

1. Волонтер відкриває строку пошуку.
2. Волонтер вводить необхідні йому теги для пошуку міток у радіусі.
3. Волонтер може переглянути мітки з шуканими тегами.

Сценарій 3: «Створення Волонтером мітки на мапі»

1. Волонтер обирає доступну йому функцію “Створити мітку” на меню мапи.
2. Волонтер заповнює форму потрібну для створення мітки, де вказує усю необхідну інформацію: адресу, вид допомоги, час її надання.
3. Якщо форма заповнена коректно, зелена мітка з’явиться на мапі.
4. Якщо форма заповнена некоректно, застосунок запропонує Волонтер доповнити/перевірити на правильність надану інформацію.

Сценарій 4: «Взаємодія з іншими користувачами застосунку через публічні чати»

1. Волонтер переходить на сторінку публічних чатів.
2. Волонтер може переглядати публічні чати.
3. Волонтер може створювати свої чати, вони позначатимуться синьою рамкою.
4. Волонтер може писати повідомлення у будь-яких чатах.
5. Волонтер може видаляти свої повідомлення.
6. При відсутності будь-якої активності у публічному чаті протягом місяця, він буде автоматично видалений.

Сценарій 5: «Взаємодія з іншими користувачами через приватні повідомлення»

1. Волонтер може писати у приватні повідомлення іншим користувачам, за умови що у отримувача повідомлення відкритий профіль.
2. Волонтер може отримувати повідомлення від інших користувачів у приватні повідомлення за тієї ж умови.

3.2 Структура застосунку

На сьогодні у світі існує безліч типів архітектури, у тому числі і для мобільних застосунків, але найвідоміші та розвинутішими є: Model-View-Controller (MVC) [24], Model-View-ViewModel (MVVM) [25] та мікросервісна архітектура [26].

3.2.1. *Архітектура MVC*

Модель-вид-контролер – це шаблон проєктування програмного забезпечення, який зазвичай використовується для розробки інтерфейсів користувача, він розділяє відповідну програмну логіку на три взаємопов’язані елементи. Це робиться для того, щоб відокремити внутрішнє представлення інформації від того, як інформація подається та приймається користувачем.

До переваг цієї архітектури належить:

1. Розділення відповідальності: MVC дозволяє розділити логіку бізнес-процесів, відображення даних та інтерфейс користувача та управління взаємодією між ними.
2. Перевикористання коду: компоненти MVC можуть бути перевикористані в інших ділянках застосунку або в інших проєктах. Модель може використовуватись безпосередньо, а вид і контролер можуть бути адаптовані до різних інтерфейсів користувача.

3. Легкість супроводу: зміни в одному компоненті не впливають на інші, що робить процес супроводу більш простим і менш помилковим.
4. Тестованість: Model-View-Controller дозволяє проводити одиничні та інтеграційні тести для кожного компонента окремо. Це сприяє виявленню та виправленню помилок, а також полегшує тестування окремих частин застосунку.

Недоліками цієї архітектури є:

1. Розмір коду: іноді використання шаблону MVC може призвести до збільшення кількості коду, оскільки кожен компонент має свої власні класи та методи.
2. Зайвий об'єм даних: комунікація між компонентами через інтерфейси може приводити до зайвого об'єму даних, особливо у великих проєктах.
3. Залежність від реалізації: шаблон MVC не визначає конкретну реалізацію компонентів. Залежно від платформи або фреймворку, реалізація може мати свої відмінності, що можуть вплинути на розробку та сумісність застосунку.

3.2.2. *Архітектура MVVM*

Модель-вигляд-модель-подання – це архітектурний шаблон у комп’ютерному програмному забезпеченні, який полегшує відокремлення розробки графічного інтерфейсу користувача – будь то за допомогою мови розмітки чи коду GUI – від розвитку бізнес логіки або серверної логіки (моделі), так що перегляд не залежить від конкретної платформи моделі.

Перевагами є:

1. Розділення відповідальності: MVVM розділяє застосунок на три основні компоненти: модель, вид та модель-подання. Це дозволяє відокремити бізнес-логіку від логіки взаємодії та представлення даних.

2. Байндинг даних: ця архітектура підтримує двостороннє зв'язування даних між моделлю та представленням. Зміни в моделі автоматично відображаються в представленні та навпаки, що зменшує необхідність вручну оновлювати дані.
3. Тестованість: вона сприяє легкості тестування, оскільки бізнес-логіка знаходиться в моделі, а логіка взаємодії – в ViewModel. Це дозволяє проводити одиничні та інтеграційні тести для кожного компонента окремо.
4. Легкість супроводу: розділення компонентів дозволяє полегшити супровід коду. Зміни в одному компоненті не впливають на інші, що спрощує внесення змін та розробку нових функцій.

Недоліками є:

1. Збільшена кількість коду: використання шаблону MVVM може призвести до збільшення кількості коду, оскільки кожен компонент має свої класи та методи. Це може збільшити складність розробки та підтримки застосунку.
2. Залежність від платформи: реалізація цієї архітектури може варіюватись в залежності від платформи, на якій розробляється застосунок. Це може призвести до залежності від конкретних інструментів та бібліотек.

3.2.3. *Мікросервісна архітектура*

Мікросервісна архітектура це архітектурний шаблон, що організовує застосунок як колекцію слабо зв'язаних, дрібнозернистих сервісів, які взаємодіють за допомогою легких протоколів. Одним із його цілей є можливість розробки та розгортання сервісів незалежно один від одного. Це досягається за рахунок зменшення залежностей у кодовій базі, що дозволяє розробникам розвивати свої сервіси з деякими обмеженнями від

користувачів, а також приховувати застосункову складність від користувачів.

Перевагами цієї архітектури є:

1. Розшарованість та складність: мікросервісна архітектура дозволяє розбити застосунок на невеликі сервіси, що спрощує управління та розгортання окремих компонентів.
2. Гнучкість та масштабованість: мікросервіси можуть бути масштабовані незалежно один від одного, що дозволяє гнучко адаптувати систему до змін навантаження.
3. Технологічна різноманітність: мікросервісна архітектура дозволяє використовувати різні технології та мови програмування для розробки окремих сервісів.

Недоліками є:

1. Складність управління: з кожним застосунком з'являється велика кількість сервісів, які потребують керування та координації.
2. Складність у тестуванні та відлагодженні: завдяки розподіленості та взаємодії між сервісами, тестування та відлагодження системи може стати складнішим.
3. Залежність від мережі: мікросервісна архітектура сильно залежить від мережевої доступності та надійності. Якщо один із сервісів стає недоступним або збоїть, це може вплинути на роботу системи в цілому.

Після детального аналізу даних архітектурних принципів, був здійснений висновок, що найбільш підходящою з них буде MVVM (Model-View-ViewModel), через свою гнучку структуру, можливість швидкого та окремого тестування компонентів ПЗ, модульність та розділення відповідальності на 3 частини.

3.3 Структура БД

У цьому мобільному застосунку використовуються 2 різні бази даних: реляційна PostgreSQL та нереляційна MongoDB.

3.3.1. *Структура реляційної PostgreSQL*

До цієї СУБД були створені такі сутності, які відповідають функціональним вимогам:

1. Користувач (User):
    - Ідентифікатор (Id) – унікальний ідентифікатор у форматі string.
    - Логін (LoginEmail) – електрона пошта користувача, використовується для реєстрації та авторизації, є унікальною для кожного користувача.
    - Пароль (Password) – захешований набір символів який користувач використовує для входу у систему.
    - Роль (Role) – обрана при реєстрації роль, яка визначає можливості взаємодії користувача. Вони поділяються на:
        - Реципієнт (Recipient) – користувач, який виключно шукає допомоги.
        - Помічник (Helper) – користувач, який може шукати допомоги та надавати локальну допомогу іншим.
        - Волонтер (Volunteer) – користувач, який представляє благодійну організацію, може надавати допомогу часто та у великих кількостях.
    - Ім’я (Nickname) – ім`я користувача, яке відображатиметься у профілі та при надсиланні повідомлень.
    - Посилання фото (ImageURL) – посилання на фото користувача, яке відображається у чатах та у профілі.
    - Про себе (Bio) – опис написаний користувачем.

- Статус (Status) – публічний булевий статус користувача, відповідає за доступність/недоступність отримання приватних повідомлень.

2. Чат (Chat):
   - Ідентифікатор (Id) – унікальний ідентифікатор у форматі string.
   - Назва (Title) – у випадку публічного чату тут буде тема чату, у разі приватного – ім’я користувача який надіслав повідомлення.
   - Дата створення (DateofCreation) – дата створення чату, для відслідковування активності.
   - Тип (Type) – тип чату у системі, поділяється на:
     - Публічний (Public) – чат для відповіді на поширенні запитання та взаємодією між користувачем та спільнотою.
     - Приватний (Private) – чат де тільки двоє користувачів можуть спілкуватися між собою.
3. Відношення чату до користувача (ChatUser):
   - Ідентифікатор (Id) – унікальний ідентифікатор у форматі string.
   - Ідентифікатор користувача (UserId) – зовнішній ключ, ідентифікатор користувача у відношенні.
   - Ідентифікатор чату (ChatId) – зовнішній ключ, ідентифікатор чату у відношенні.
   - Належність користувача до чату (InChat) – позначає що користувач є учасником чату.

3.3.2. *Структура нереляційної MongoDB*

До нереляційної БД MongoDB були створені такі сутності:

1. Мітка (Sign):
    - Ідентифікатор (Id) – унікальний ідентифікатор у форматі string.
    - Назва (Title) – назва мітки, яка відображатиметься на мапі.
    - Тег (Tag) – ключові слова які починаються з «#», використовуються для пошуку міток через пошукову строку.
    - Опис допомоги (DescriptionofHelp) – детальний опис допомоги, яку надають у цьому місці.
    - Адреса (Address) – адреса, за якою розташовується мітка.
    - Ідентифікатор власника (UserId) – унікальний ідентифікатор користувача, який створив цю мітку.
2. Список «обраних» міток (ListofFeaturedSigns):
    - Ідентифікатор (Id) – унікальний ідентифікатор у форматі string.
    - Ідентифікатор користувача (UserId) – унікальний ідентифікатор користувача у форматі string, якому належить цей список.
    - Список міток (Signs) – масив доданих користувачем міток.
3. Повідомлення (Message):
    - Ідентифікатор (Id) – унікальний ідентифікатор у форматі string.
    - Ідентифікатор чату (ChatId) – унікальний ідентифікатор чату у форматі string, у якому це повідомлення існує.
    - Час надіслання (Time) – час надіслання повідомлення у чат.
    - Вміст (Content) – вміст надісланого повідомлення.

4. АНАЛІЗ ЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості реалізації програмного забезпечення

Згідно функціональних вимог головні 3 функції, які повинні бути реалізовані перелічені нижче.

4.1.1. *Особливість реалізації системи додавання міток на мапу*

Після встановлення та налаштування Node.js та його фреймворку Express.js для швидкого розробки серверної логіки та обробки запитів, та підключення до них MongoDB для зберігання інформації міток, ми використали REST API для взаємодії з базою даних та обміну даними між сервером та клієнтом.

Після встановлення та налаштування Flutter для розробки мобільного застосунку та створення інтерфейсу користувача (UI) для відображення мапи та додавання міток знову використали REST API для відображення міток на мапі.

Використаємо MongoDB драйвера для Node.js для підключення та взаємодії з базою даних, далі були створені схеми даних для міток та їх опису та розроблені методи доступу до бази даних (створення, отримання, оновлення, видалення міток).

Організовані комунікації між клієнтом та сервером через використання HTTP протоколу для взаємодії через API. Реалізовані методи на сервері для обробки запитів на додавання, отримання, оновлення та видалення міток. Після інтерфейс користувача був оновлений для додавання міток на мапу та відображення описів міток.

4.1.2. *Особливість реалізації чату між користувачами*

Після встановлення та налаштування Node.js та його фреймворку Express.js для швидкого розробки серверної логіки та обробки запитів, та підключення до них MongoDB для зберігання змісту повідомлень, ми

використали REST API для взаємодії з базою даних та обміну даними між сервером та клієнтом.

Після встановлення та налаштування Flutter для розробки мобільного застосунку та створення інтерфейсу користувача (UI) для відправлення та отримання повідомлень знову використали REST API.

Використаємо MongoDB драйвера для Node.js для підключення та взаємодії з базою даних, далі були створені схеми даних для повідомлень та розроблені методи доступу до бази даних (створення, отримання, оновлення, видалення міток).

Після був використаний WebSocket для реалізації двосторонньої зв'язку між клієнтом та сервером. Реалізовано логіку обробки вхідних та вихідних повідомлень на сервері та клієнті та автоматичне оновлення інтерфейсу користувача для відображення отриманих повідомлень у режимі реального часу .

4.2. Демонстрація роботи застосунку

4.2.1. *Мапи з мітками*

На представлених скріншотах ми можемо бачити роботу мапи з мітками. Мапа відображає різні географічні точки, які були відзначені користувачами застосунку.

Кожна мітка на мапі може представляти різні типи місць з допомогою, що є важливими для українців за кордоном. Наприклад, мітка може позначати місце де видають гуманітарну допомогу, місце де можна отримати консульську допомогу та тощо.

Користувач може вибрати будь-яку мітку на мапі, щоб дізнатися більше інформації про позначене місце та вид допомоги яка надається. Ця функція також дозволяє користувачам застосунку легко знаходити потрібну інформацію про місцеву українську громаду та отримувати необхідну допомогу.

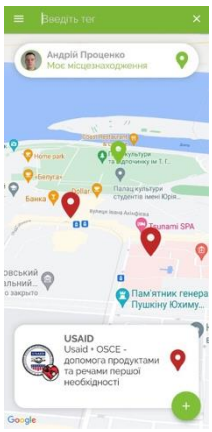


Рис. 4.1. Скріншоти роботи мапи з мітками

4.2.2. *Приватні та публічні чати*

Тут на ілюстраціях можна побачити розділи приватних та публічних чатів мобільного додатку для самоорганізації та взаємодопомоги співвітчизників за кордоном. Скріншоти ілюструють, як користувач може перемикатися між двома типами чатів, використовуючи дві кнопки на верхній частині екрана: "приватні" та "публічні".

Кнопка "приватні" веде до розділу, де користувач може бачити всі приватні розмови, в яких він бере участь. Приватні чати – це місце, де два користувачі можуть спілкуватися в чотири очі, обговорюючи, наприклад, деталі допомоги або обмінюючись інформацією.

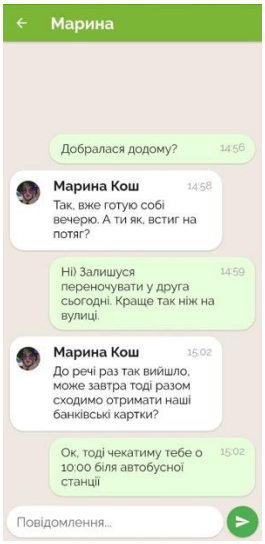


Рис. 4.2. Скріншот відображення та роботи приватних чатів

Натискаючи кнопку "публічні", користувач може переглянути всі доступні публічні чати. Публічні чати – це відкриті спільноти, в яких користувачі можуть обмінюватися думками, ідеями та надавати або отримувати допомогу від широкого кола осіб. Кожен чат має свою назву та короткий опис, що допомагає користувачам зрозуміти, яка тема обговорення в кожному з них. Такий інтерфейс дозволяє користувачам легко знайти потрібну інформацію або спільноту, що відповідає їх потребам. Всі ці особливості допомагають створити продуктивне середовище для взаємодопомоги та спілкування серед співвітчизників за кордоном.

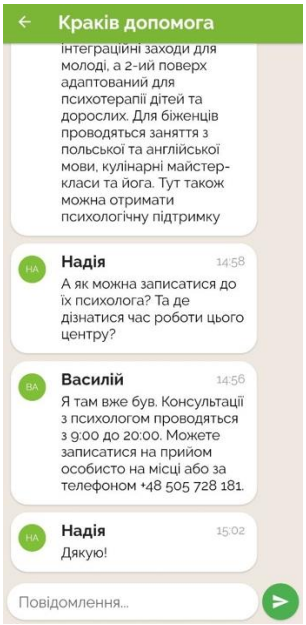


Рис. 4.3. Скріншот відображення та роботи публічних чатів

4.2.3. *Сторінка користувача*

На представленому скріні мобільного застосунку для самоорганізації та взаємодопомоги співвітчизників за кордоном видно розділ профілю користувача. У цьому розділі користувач може переглянути свої основні дані, включаючи ім'я, роль та інформацію, яку він ввів про себе. Поле для логіна, яке також видно, використовується для входу в систему. Всі ці дані

окрім ролі можна редагувати, що дозволяє користувачам оновлювати свою інформацію в будь-який момент.

Роль користувача (наприклад, реципієнт, помічник або волонтер) відображається в профілі, що допомагає іншим користувачам зрозуміти, яку допомогу може надати або потребує конкретний користувач. Ця зручна функція редактування допомагає користувачам підтримувати актуальність своїх даних у мобільному застосунку.

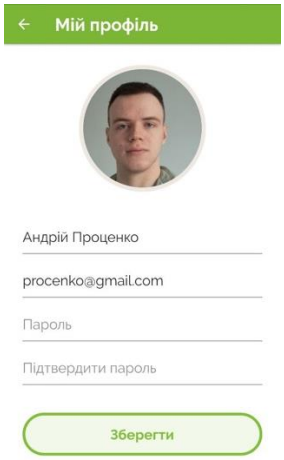


Рис. 4.4. Скріншот сторінки користувача

4.3. Рекомендації щодо подальшого вдосконалення

За час роботи над цим мобільним застосунком були знайдені ідеї для подальшого вдосконалення цього програмного продукту, а саме:

1. Вдосконалення інтерфейсу до інтуїтивного та зрозумілого будь-яким віковим групам.
2. Впровадження системи створення маршрутів різними видами транспорту до міток з поточної геолокації.
3. Додання рекомендованих міток на основі збережених користувачем раніше.

4. Додання можливості надсилати текстові та мультимедіа файли у чатах.
5. Додання дошки новин на головну сторінку.
6. Створення спеціальних можливостей для людей з вадами зору.

ВИСНОВКИ

У рамках даного проєкту було розроблено мобільний застосунок для самоорганізації та взаємодопомоги українців за кордоном, який включає мапу з мітками та можливість спілкування в публічних та приватних чатах. З метою адекватного задоволення потреб користувачів було вирішено використати Flutter для клієнтської частини та Node.js з Express.js для серверної частини.

В рамках аналізу виявилось, що існуючі рішення не надають достатнього функціоналу для спілкування та самоорганізації українців за кордоном. Тому було вирішено реалізувати цей функціонал у нашому мобільному застосунку.

Для забезпечення оптимальної роботи застосунку та зберігання даних було обрано поєднання MongoDB та PostgreSQL. MongoDB було використано для зберігання даних про мітки та чати, а PostgreSQL – для зберігання даних користувачів та управління правами доступу.

Застосунково було прийнято рішення використати MVC архітектуру для забезпечення гнучкості, масштабованості та легкості управління кодом проєкту.

Після розробки застосунку було проведено інтеграційне, функціональне та тестування сумісності, що підтвердило високу якість та надійність нашого рішення. За результатами тестування було визначено можливі напрямки для подальшого вдосконалення та розвитку проєкту.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

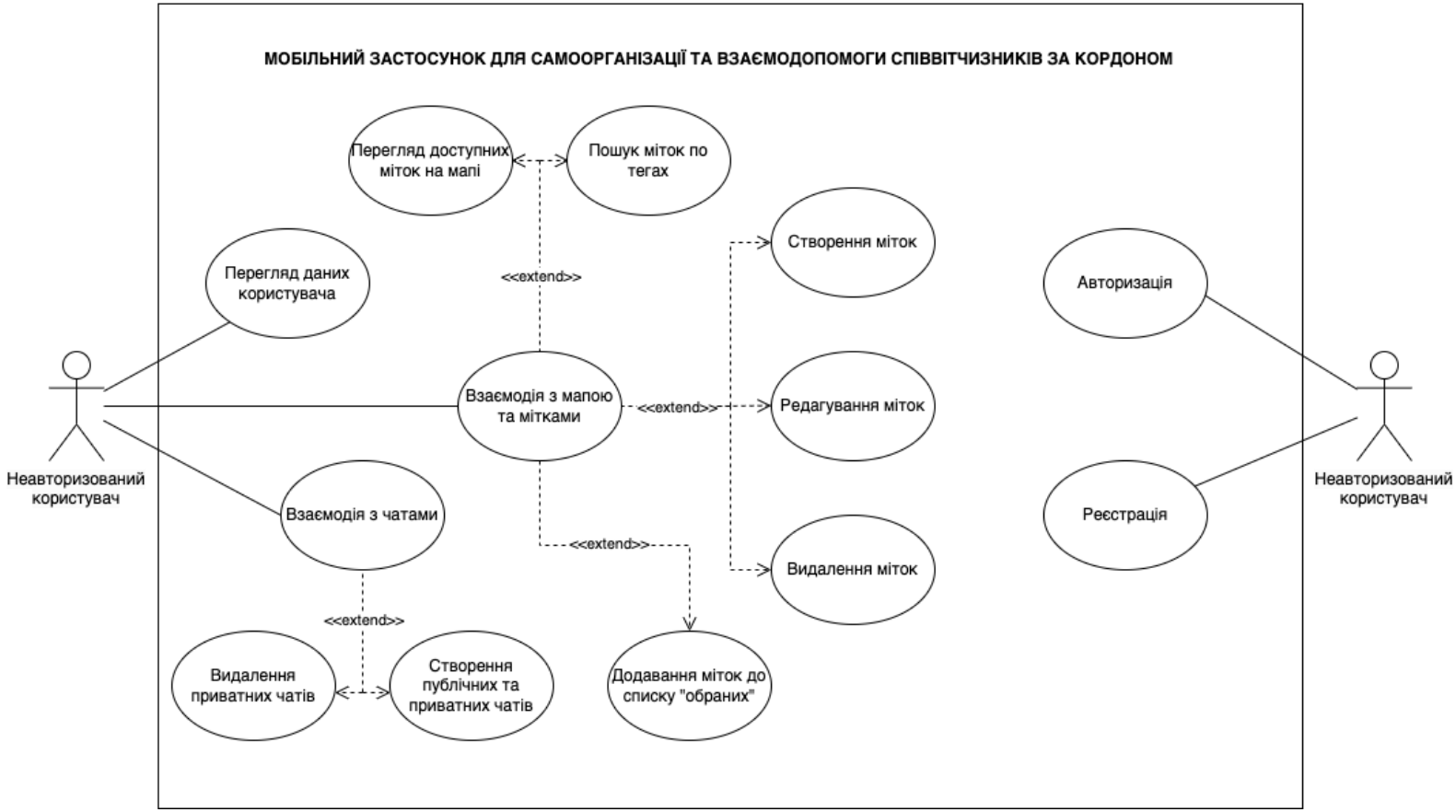
1. Ukraine Refugee Situation [Електронний ресурс]. — Режим доступу: https://data.unhcr.org/en/situations/ukraine (accessed on: 12.05.2023).
2. Міграція та соціально-політичні настрої під час повномасштабної війни Росії проти України — восьма хвиля дослідження: [Електронний ресурс]. — Режим доступу: https://gradus.app/uk/open-reports/ (доступ на: 05.09.2022).
3. Левін, Р.Я., Яценко, Г.Ю. Соціальні проблеми українських біженців за кордоном [Електронний ресурс] / Р. Я. Левін, Г.Ю. Яценко // Відділ моніторингових досліджень соціально-економічних трансформацій: 06.10.2022: тез. доп. / ДУ «Інститут економіки та прогнозування НАН України» — Режим доступу: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/6465cc19-5c46-4804-9722-27b3ec7e40d1/content. — (16.11.2022).
4. Seeker request [Електронний ресурс]. — Режим доступу: https://app.unterkunft-ukraine.de/seeker-request?lang=en (accessed on: 12.05.2023).
5. How Many Users Does Facebook Have? [Електронний ресурс]. — Режим доступу: https://www.oberlo.com/statistics/how-many-users-does-facebook-have (accessed on: 12.05.2023).
6. Telegram Users by Country 2023 [Електронний ресурс]. — Режим доступу: https://worldpopulationreview.com/country-rankings/telegram-users-by-country (accessed on: 12.05.2023).
7. Телеграм: чому пересічним українцям варто обмежити споживання, а владі — збільшити присутність [Електронний ресурс]. — Режим доступу: https://zmina.info/articles/telehram-chomu-peresichnym-ukrayintsyam-varto-obmezhyty-spozhyvannya-a-vladi-zbil%CA%B9shyty-prysutnist%CA%B9/ (доступ на: 12.05.2023).

8. Statista. Number of unique Viber user IDs from June 2011 to March 2020 [Електронний ресурс]. — Режим доступу: https://www.statista.com/statistics/316414/viber-messenger-registered-users/ (accessed on: 12.05.2023).
9. JavaScript [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/JavaScript (доступ на: 12.05.2023).
10. PYPL Index. PYPL PopularitY of Programming Language [Електронний ресурс]. — Режим доступу: https://pypl.github.io/PYPL.html (accessed on: 12.05.2023).
11. Python [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Python (доступ на: 12.05.2023).
12. Kotlin [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Kotlin (доступ на: 12.05.2023).
13. Node.js v18.16.0 documentation [Електронний ресурс]. — Режим доступу: https://nodejs.org/dist/latest-v18.x/docs/api/ (accessed on: 12.05.2023).
14. Express.js API reference [Електронний ресурс]. — Режим доступу: https://expressjs.com/en/4x/api.html (accessed on: 12.05.2023).
15. Passport.js [Електронний ресурс]. — Режим доступу: https://www.passportjs.org/ (accessed on: 12.05.2023).
16. What Socket.IO is [Електронний ресурс]. — Режим доступу: https://socket.io/docs/v4 (accessed on: 12.05.2023).
17. Flutter [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Flutter (доступ на: 12.05.2023)
18. Dart [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/Dart (доступ на: 12.05.2023).
19. Swift [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Swift_(programming_language) (accessed on: 12.05.2023)
20. MySQL [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/MySQL (доступ на: 12.05.2023)

21. PostgreSQL [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/PostgreSQL (доступ на: 12.05.2023).
22. MongoDB [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/MongoDB (доступ на: 12.05.2023).
23. What is Prisma? [Електронний ресурс]. — Режим доступу: https://www.prisma.io/docs/concepts/overview/what-is-prisma (accessed on: 12.05.2023).
24. MVC (Model-view-controller) [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (accessed on: 12.05.2023).
25. MVVM (Model-view-viewmodel) [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel (accessed on: 12.05.2023).
26. Microservices [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Microservices (accessed on: 12.05.2023).
27. WebSocket [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/WebSocket (accessed on: 12.05.2023).

ДОДАТКИ

Додаток 1
Копії графічних матеріалів

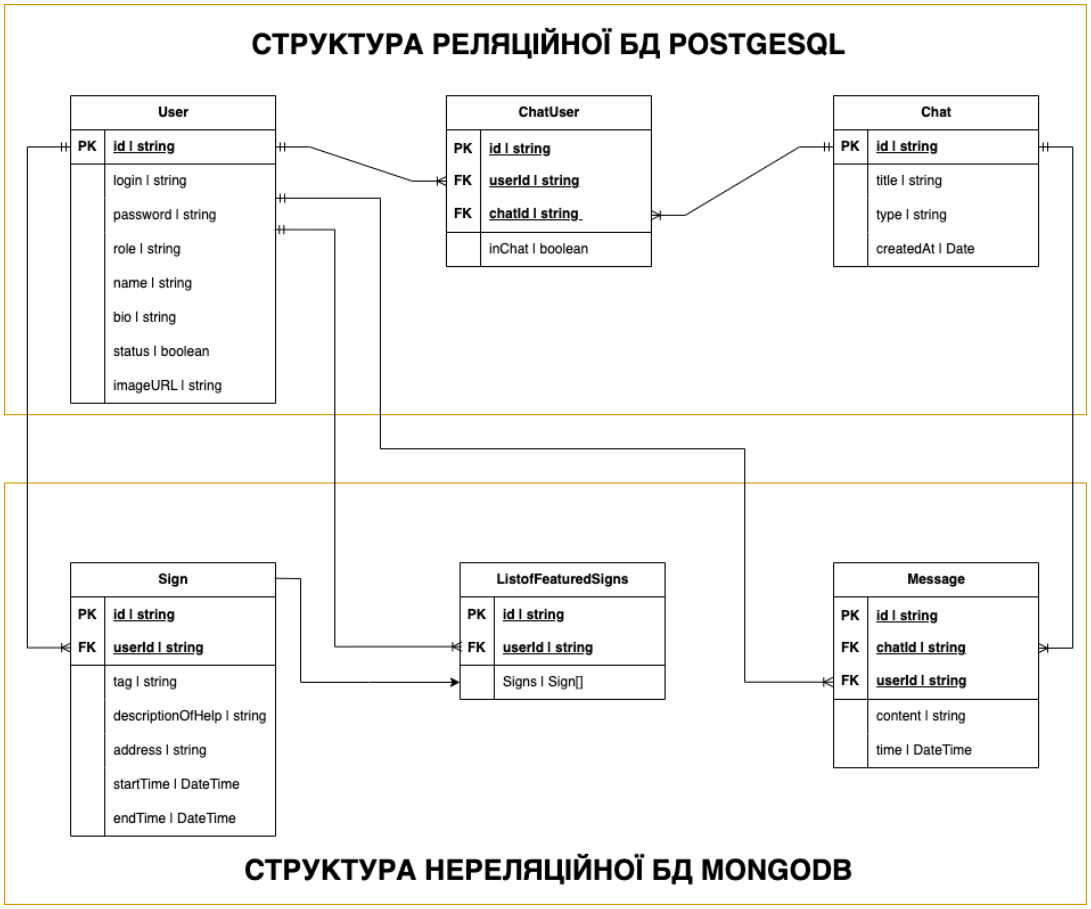


ДП.045490-06-99

Мобільний застосунок для самоорганізації взаємодопомоги співвітчизників за кордоном.

Функціональність застосунку.

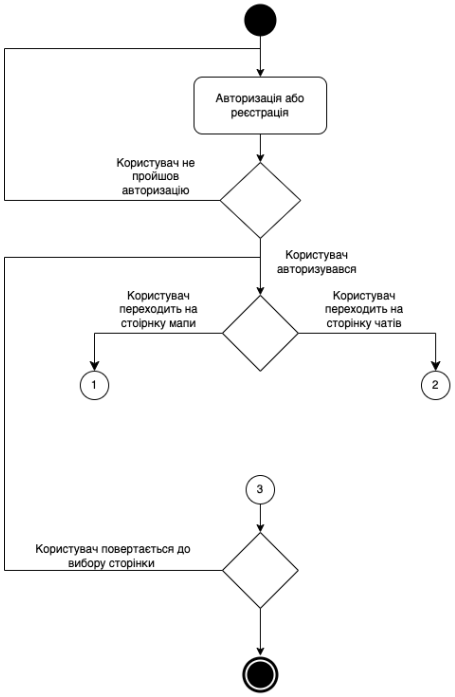
UML-діаграма сценаріїв

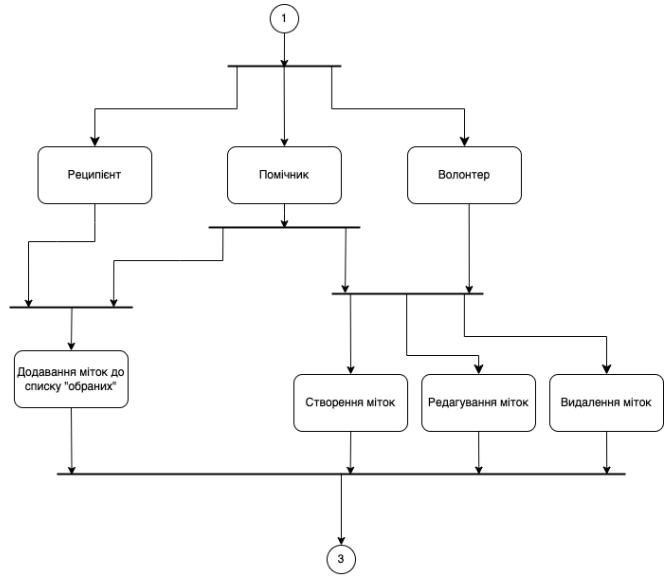


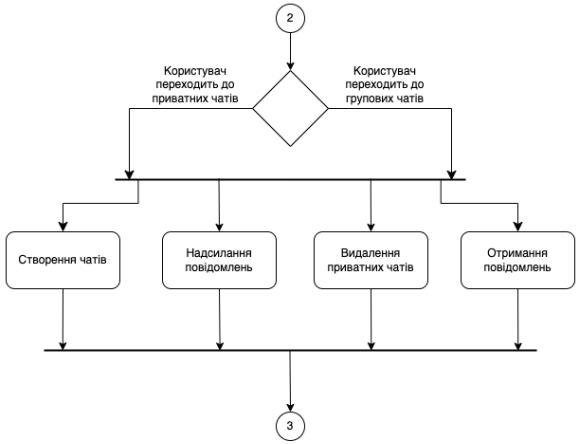
ДП.045490-07-99

Мобільний застосунок для самоорганізації взаємодопомоги співвітчизників за кордоном.

Структура сутностей та баз даних. ER-діаграма

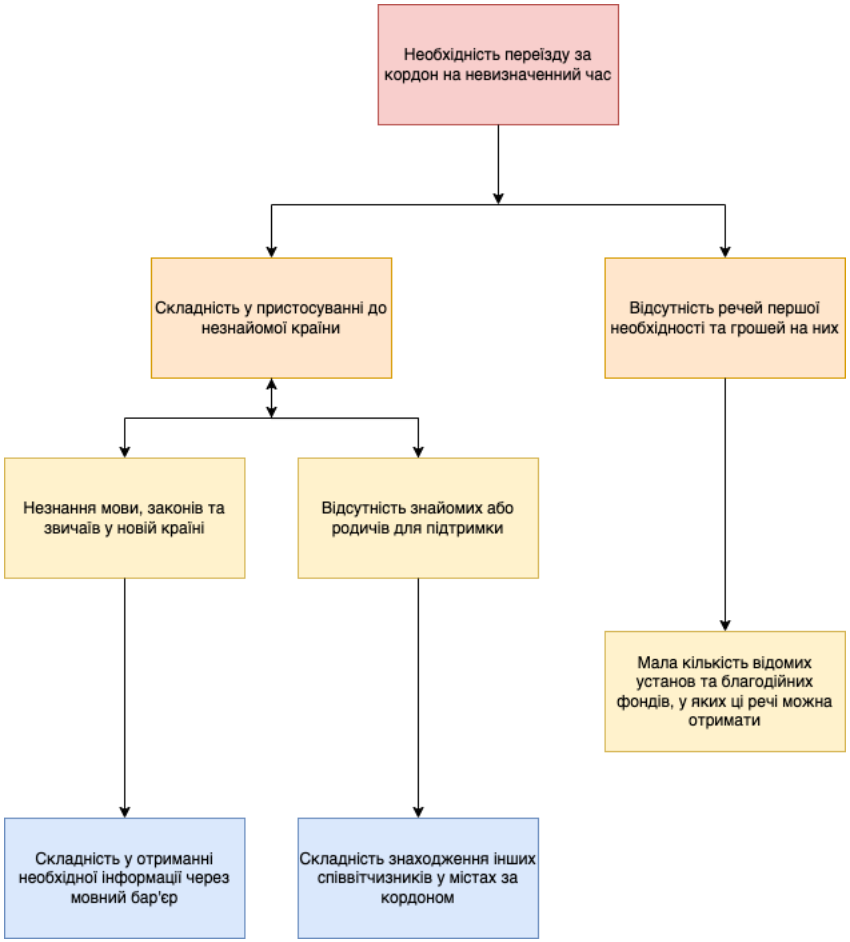






Алгоритм роботи застосунку

Проценко Андрій КП-92



Дерево проблем проєкту

Проценко Андрій КП-92

Додаток 2
Лістинг програми

Лістинг модуля мапи

```
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:goodeeds_ua/map/models/map_sign_model.dart';
import 'package:goodeeds_ua/map/providers/map_signs_provider.dart';
import 'package:goodeeds_ua/map/widgets/map_appbar.dart';
import 'package:goodeeds_ua/shared/helpers/app_colors.dart';
import 'package:goodeeds_ua/shared/widgets/app_drawer.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:goodeeds_ua/map/widgets/map_sign_info_item.dart';
import 'package:goodeeds_ua/map/widgets/map_user_badge.dart';

const double CAMERA_ZOOM = 16;
const double CAMERA_TILT = 80;
const double CAMERA_BEARING = 30;
const double PIN_VISIBLE_POSITION = 20;
const double PIN_INVISIBLE_POSITION = -220;

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

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

class _MapScreenState extends State<MapScreen> {
  List<MapSignModel> signs = [];

  bool userBadgeSelected = false;
  int selectedSignIndex = 0;
  PageController? _marker_page_controller;

  GoogleMapController? _map_controller;
  BitmapDescriptor? sourceIcon, markerIcon, newSignIcon;

  double signInfoPosition = PIN_VISIBLE_POSITION;
  LatLng userLocation = LatLng(48.46232417742038, 35.07236228860029);
  LatLng? pinnedLocation;


  @override
  void initState() {
    super.initState();

    _marker_page_controller = PageController();

    signs = getMapSigns();
  }

  @override
  void dispose() {
    super.dispose();
    _marker_page_controller!.dispose();
```

```
}

void setSourceAndDestinationMarkerIcons(BuildContext context) async {
  if (sourceIcon != null) return;

  sourceIcon = await BitmapDescriptor.fromAssetImage(
    ImageConfiguration(devicePixelRatio: 2.0),
    'assets/imgs/map/source_pin.png'
  );

  markerIcon = await BitmapDescriptor.fromAssetImage(
    ImageConfiguration(devicePixelRatio: 2.0),
    'assets/imgs/map/destination_pin.png'
  );

  newSignIcon = await BitmapDescriptor.fromAssetImage(
    ImageConfiguration(devicePixelRatio: 2.0),
    'assets/imgs/map/new_pin.png'
  );

  setState(() {});
}

@override
Widget build(BuildContext context) {
  setSourceAndDestinationMarkerIcons(context);

  CameraPosition initialCameraPosition = CameraPosition(
    zoom: CAMERA_ZOOM,
    tilt: CAMERA_TILT,
    bearing: CAMERA_BEARING,
    target: userLocation
  );

  return Scaffold(
    drawer: AppDrawer(),
    appBar: MapAppBar(),
    body: Stack(
      children: [
        Positioned.fill(
          child: GoogleMap(
            myLocationEnabled: true,
            compassEnabled: false,
            tiltGesturesEnabled: false,
            markers: getGMSMarkers(),
            mapType: MapType.normal,
            initialCameraPosition: initialCameraPosition,
            onTap: (LatLng loc) {
              onMapTapped(loc);
            },
            onMapCreated: (GoogleMapController controller) {
              _map_controller = controller;
              // showPinsOnMap();
```

```
          },
        ),
      ),
      Positioned(
        top: 10,
        left: 0,
        right: 0,
        child: MapUserBadge(
          isSelected: this.userBadgeSelected,
          onTap: () => onUserMarkerSelected(animateMap: true),
        ),
      ),
      AnimatedPositioned(
        duration: const Duration(milliseconds: 600),
        curve: Curves.easeInOut,
        left: 0,
        right: 0,
        bottom: this.signInfoPosition,
        child: Container(
          height: 220,
          child: PageView(
            controller: _marker_page_controller,
            onPageChanged: (int index) {
              onMarkerSelected(index, animateMap: true);
            },
            children: List.generate(
              signs.length,
              (index) => MapSignInfoItem(sign: signs[index])
            )
          )
        ),
      ],
    ),
    floatingActionButton: pinnedLocation != null ?
    FloatingActionButton(
      onPressed: () {
        GoRouter.of(context).push('/create_sign', extra: pinnedLocation);
      },
      backgroundColor: AppColors.GREEN1,
      child: const Icon(Icons.add_location_alt_outlined),
    ) : Container()
  );
}


Set<Marker> getGMSMarkers(){
  var markers = Set<Marker>();

  var srcIcon = sourceIcon;
  var mrkIcon = markerIcon;
  var newIcon = newSignIcon;

  if (srcIcon == null || mrkIcon == null || newIcon == null){
```

```
    return markers;
  }

  markers.add(Marker(
    markerId: MarkerId('my_location'),
    position: userLocation,
    icon: srcIcon,
    onTap: () {
      onUserMarkerSelected();
    }
  ));

  for (int i = 0; i < signs.length; ++i) {
    var m = signs[i];

    markers.add(Marker(
      markerId: MarkerId(m.location.toString()),
      position: m.location,
      icon: mrkIcon,
      onTap: () {
        onMarkerSelected(i, setPage: true);
      }
    ));
  }

  var pinLoc = pinnedLocation;
  if (pinLoc != null){
    markers.add(Marker(
      markerId: MarkerId('pinned_location'),
      position: pinLoc,
      icon: newIcon,
    ));
  }

  return markers;
}

void showPinsOnMap() {
  setState(() {

  });
}

void onMarkerSelected(int index, {animateMap = false, setPage = false}) {
  setState(() {
    selectedSignIndex = index;
    userBadgeSelected = false;
    signInfoPosition = PIN_VISIBLE_POSITION;
    pinnedLocation = null;
  });

  if(setPage) _marker_page_controller?.jumpToPage(index);
  if(animateMap) animateCameraToLocation(signs[index].location);
```

```
  }

  void onUserMarkerSelected({animateMap = false}) {
    setState(() {
      userBadgeSelected = true;
      signInfoPosition = PIN_INVISIBLE_POSITION;
      pinnedLocation = null;
    });

    if(animateMap) animateCameraToLocation(userLocation);
  }

  void onMapTapped(LatLng location) {
    setState(() {
      userBadgeSelected = false;
      signInfoPosition = PIN_INVISIBLE_POSITION;
      pinnedLocation = location;
    });
    animateCameraToLocation(location);
  }

  void animateCameraToLocation(LatLng location){
    _map_controller?.animateCamera(CameraUpdate.newLatLngZoom(location, CAMERA_ZOOM));
  }
}
```

Лістинг модуля міток

```
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:goodeeds_ua/map/models/map_sign_model.dart';
import 'package:goodeeds_ua/shared/helpers/app_colors.dart';
import 'package:goodeeds_ua/shared/helpers/utils.dart';
import 'package:goodeeds_ua/shared/widgets/big_button.dart';

class SignScreen extends StatelessWidget {
  MapSignModel sign;

  SignScreen({ required this.sign, Key? key }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: AppColors.WHITE,
      appBar: appBar(context),
      body: body(),
    );
  }

  PreferredSizeWidget appBar(context) {
    return AppBar(
      backgroundColor: AppColors.GREEN2,
      elevation: 2,
```

```
    title: Text(sign.name, style: TextStyle(fontWeight: FontWeight.w600)),
    actions: [
      IconButton(
        onPressed: () {
          GoRouter.of(context).push('/edit_sign', extra: sign);
        },
        icon: const Icon(Icons.edit_outlined),
      ),
      IconButton(
        onPressed: () {
        },
        icon: const Icon(Icons.delete_outline),
      ),
      IconButton(
        onPressed: () {
        },
        icon: Icon(
          sign.isBookmarked ? Icons.star : Icons.star_border,
          color: AppColors.WHITE,
          size: 28,
        )
      )
    ],
  );
}

Widget body() {
  return SingleChildScrollView(
    child: Container(
      padding: EdgeInsets.symmetric(vertical: 30, horizontal: 22),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Center(
            child: Container(
              width: 180,
              height: 180,
              decoration: BoxDecoration(
                borderRadius: BorderRadius.all(Radius.circular(90)),
                border: Border.all(
                  width: 3,
                  color: Colors.black
                )
              ),
              child: ClipOval(
                child: Image.asset(sign.imageUrl, fit: BoxFit.cover),
              )
            )
          ),
          const SizedBox(height: 40),
          Text('Опис:', style: TextStyle(color: Colors.black, fontSize: 18, fontWeight: FontWeight.w600)),
          const SizedBox(height: 10),
          Text(sign.description, style: TextStyle(fontSize: 18, fontWeight: FontWeight.w500)),
```

```
            const SizedBox(height: 20),
            Text('Теги:', style: TextStyle(color: Colors.black, fontSize: 18, fontWeight: FontWeight.w600)),
            const SizedBox(height: 10),
            Text(Utils.tagListToString(sign.tags), style: TextStyle(fontSize: 18, fontWeight:
FontWeight.w500)),

            const SizedBox(height: 40),
          ],
        ),
      )
    );
  }
}
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

МОБІЛЬНИЙ ДОДАТОК ДЛЯ САМООРГАНІЗАЦІЇ ВЗАЄМОДОПОМОГИ СПІВВІТЧИЗНИКІВ ЗА КОРДОНОМ

Виконав: студент групи КП-92 Проценко Андрій Павлович

Керівник: доцент кафедри ЕІ, к.т.н. Вунтесмері Юрій Володимирович

Київ — 2023

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: розробити мобільний додаток для самоорганізації взаємодопомоги співвітчизників за кордоном.

Аналіз:

- виявити аналоги розроблюваного ПЗ та виділити їх сильні та слабкі сторони у порівнянні з створюваним мобільним додатком
- окреслити функціональні вимоги до додатку для самоорганізації та взаємодопомоги співвітчизників за кордоном

Розроблення:

- проаналізувати та обрати засоби розроблення мобільного додатку
- реалізувати код додатку згідно описаних вимог

Результати:

- виялення плюсів і мінусів розроблюваного ПЗ у порівнянні з аналогами
- перевірка додатку шляхом тестування
- опис можливих покращень додатку у майбутньому

АКТУАЛЬНІСТЬ

1. Велика кількість українців за кордоном, які зацікавлене у отриманні різних видів безоплатної допомоги.
2. Скорочення програм підтримки біженців у країнах ЄС.
3. Недосконалість наявних рішень:
   - незручний доступ до необхідної допомоги через сторонні інструменти
   - відсутність централізованих інформаційних дошок та місць взаємодії між біженцями у конкретних країнах.

ІСНУЮЧІ АНАЛОГИ

Facebook

Telegram

Viber

ПОРІВНЯННЯ З АНАЛОГАМИ

Критерій	Аналоги			**Розроблене ПЗ**
	Facebook	Telegram	Viber	
Кросплатформеність на мобільних ОС	+	+	+	+
Можливість створювати приватних чатів	+	+	+	+
Можливість створення групових чатів	+	+	+	+
Наявність інтерактивної мапи з користувацькими мітками	–	–	–	+
Збереження міток у «обрані» в додатку	–	–	–	+
Можливість знаходити мітки поруч за допомогою геолокації	–	–	–	+
Спеціалізація на наданні зручних інструментів для отримання та відслідковування допомоги	–	–	–	+

ІДЕЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ САМООРГАНІЗАЦІЇ ТА ВЗАЄМОДОПОМОГИ СПІВВІТЧИЗНИКАМ ЗА КОРДОНОМ

Взаємодія з інтерактивною мапою:

1. Мітки з описом допомоги недалеко від користувача;
2. Створення користувацьких міток;
3. Теги для зручного пошуку по мітках;
4. Список «обраних» міток.

Взаємодія з приватними та груповими чатами:

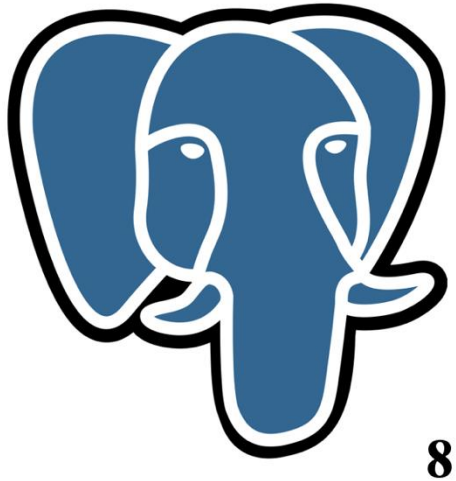
1. Перегляд усіх доступних групових чатів;
2. Можливість створення групових чатів;
3. Можливість писати повідомлення у групові та приватні чати.

ФУНКЦІОНАЛЬНІ ВИМОГИ ДО МОБІЛЬНОГО ДОДАТКУ

1. Користувачі можуть мати одну з 3 ролей: Реципієнт, Помічник або Волонтер;
2. Усі користувачі мають доступ до мап з мітками;
3. Реципієнт та Помічник можуть зберігати цікаві їм мітки у список «обраних»;
4. Помічник та Волонтер можуть створювати, редагувати та видаляти свої мітки або мітки своїх організацій;
5. Усі користувачі можуть спілкуватися через приватні та публічні чати;
6. Усі користувачі можуть створювати публічні чати;

ЗАСОБИ РЕАЛІЗАЦІЇ: ЗАГАЛЬНІ ВІДОМОСТІ

Тип ПЗ — **мобільний додаток**.
Мова програмування (сервер) — **JavaScript**, середовище **Node.js**.
Мова програмування (клієнт) — **Dart,** фреймворк **Flutter**.
База даних — **MongoDB, PostgreSQL**.
ORM для роботи з БД — **Prisma.**



ЗАСОБИ РЕАЛІЗАЦІЇ: ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ

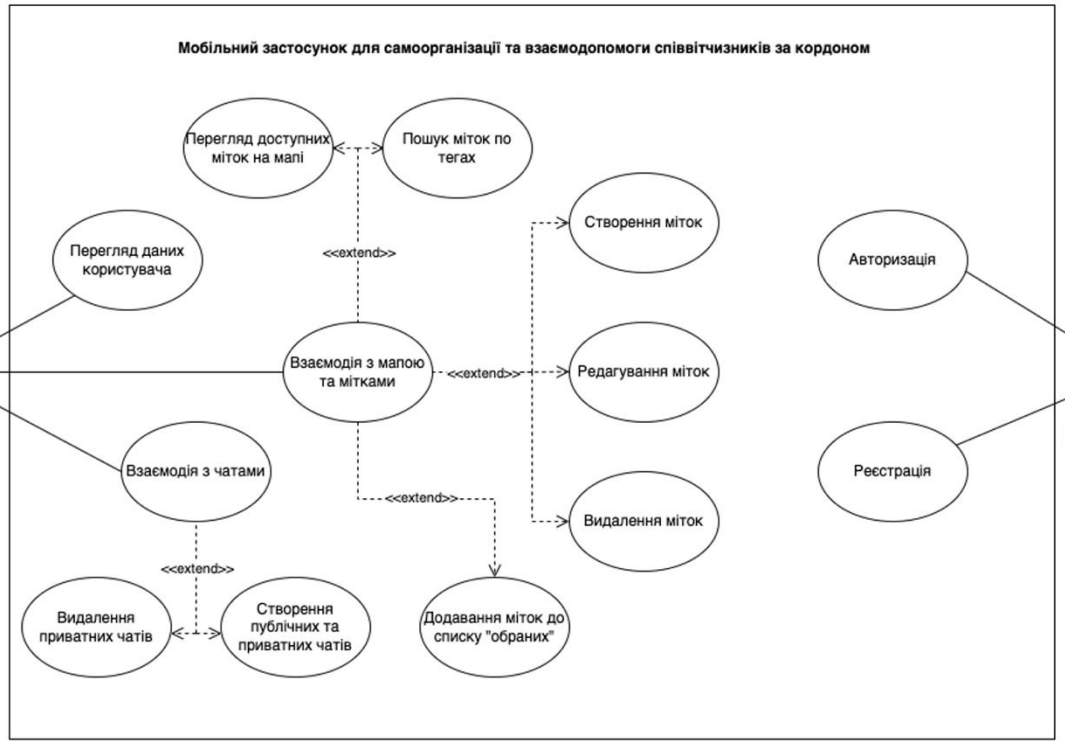
Клієнтська частина:

- Flutter
- Dart

Серверна частина:

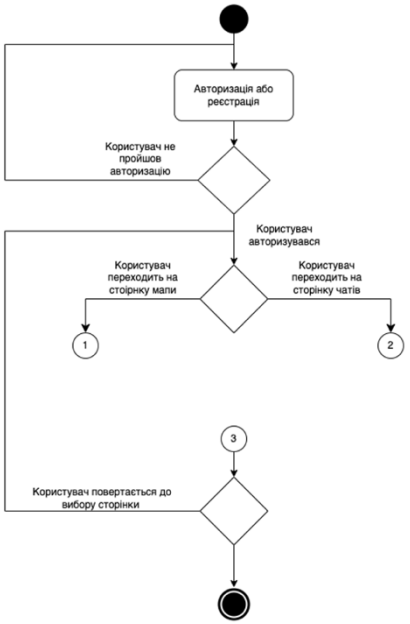
- Node.js
- Express.js

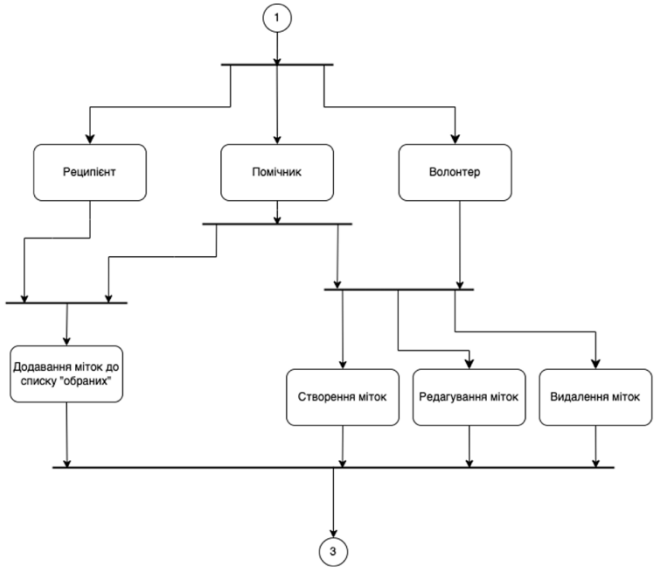
ФУНКЦІОНАЛЬНІСТЬ ЗАСТОСУНКУ

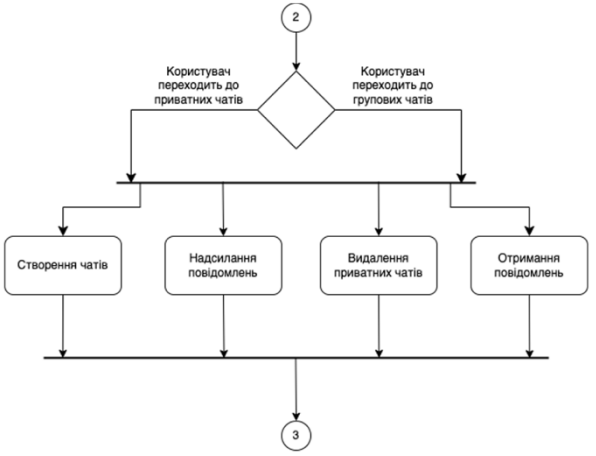


СТРУКТУРА СУТНОСТЕЙ ТА БАЗ ДАНИХ

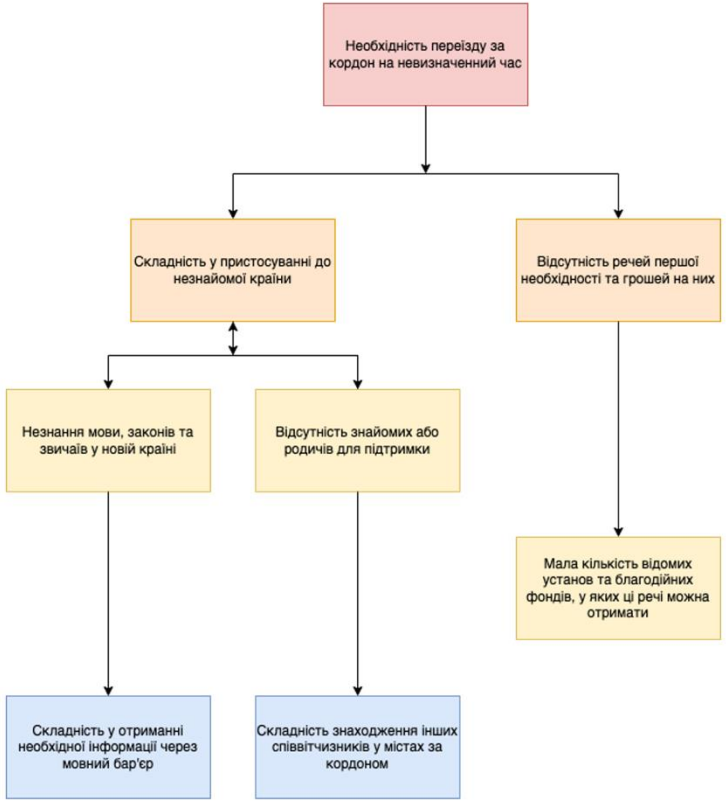
АЛГОРИТМ РОБОТИ ЗАСТОСУНКУ







ДЕРЕВО ПРОБЛЕМ ПРОЄКТУ



ТЕСТУВАННЯ ЗАСТОСУНКУ

Тестування буде проводитися на рівні системного тестування за методикою "Сірий ящик", де ми аналізуємо як внутрішні механізми коду, так і функціональність застосунку. Представлені такі методики:

1. Функціональне тестування;
2. Тестування продуктивності;
3. Тестування користувацького інтерфейсу.

НАПРЯМИ РОЗВИТКУ

Нові функції мапи:

1. Впровадження системи створення маршрутів до міток;
2. Автоматичне створення списку рекомендованих міток.

Нові функції чатів:

1. Додавання мультимедіа вкладень у повідомлення.

Нові функції чатів:

1. Додавання функцій для людей з вадами зору;
2. Покращення інтерфейсу.

ВИСНОВКИ

1. Були проаналізовані аналоги розроблюваного ПЗ: Facebook, Telegram та Viber.
2. Сформульовано ідею розроблюваного додатку.
3. Спроєктовано архітектуру шаблоном MVC та загальні структури баз даних.
4. Реалізовано реєстрацію та авторизацію, мапи з мітками, пошук міток за тегами, систему публічних та приватних чатів.
5. Проведено функціональне, тестування продуктивності та тестування інтерфейсу для перевірки роботоздатності критичних та некритичних функцій застосунку.
6. Представлені напрямки подальшого вдосконалення мобільного застосунку: створення системи маршрутів до міток, створення рекомендацій на основі вже обраних раніше міток, і т. д.

ПЕРЕВІРКА НА ПЛАГІАТ

UNICHECK
by Turnitin

Ім'я користувача:
Вунтесмері Юрій Володимирович

ID перевірки:
1015619207

Дата перевірки:
15.06.2023 22:22:50 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
16.06.2023 10:17:35 EEST

ID користувача:
100012490

Назва документа: **Пояснювальна_записка_КП_92_Проценко_v2**

Кількість сторінок: **36** **Кількість слів:** **6417** **Кількість символів:** **48433** **Розмір файлу:** **63.09 KB** **ID файлу:** **1015256292**

5.78%
Схожість

Найбільша схожість: **1.4%** з джерелом з Бібліотеки (**ID файлу:** **1004096809**)

3.97% Джерела з Інтернету	**307**	**Сторінка** 38
5.72% Джерела з Бібліотеки	**359**	**Сторінка** 39

1.75% Цитат

Цитати	**16**	**Сторінка** 40

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	**6**

ДЯКУЮ ЗА УВАГУ!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2022 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ САМООРГАНІЗАЦІЇ ТА ВЗАЄМОДОПОМОГИ СПІВВІТЧИЗНИКІВ ЗА КОРДОНОМ

Програма та методика тестування

ДП.045490-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Юрій ВУНТЕСМЕРІ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Андрій ПРОЦЕНКО

2022

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Мобільний застосунок для самоорганізації та взаємодопомоги співвітчизників за кордоном створений у оточенні Node.js з використанням мов програмування JavaScript і Dart, фреймворків Express.js, і Flutter з використанням протоколу WebSocket.

2. МЕТА ТЕСТУВАННЯ

У рамках тестування розробленого мобільного застосунку, потрібно переконатися в наступних аспектах:

1. Робота функціональних модулів застосунку відповідає специфікаціям.
2. Правильність обміну даними з сервером, реалізованою через Node.js та Express.js.
3. Сумісність з різними версіями мобільних операційних систем.
4. Дотримання основних принципів безпеки та конфіденційності даних користувача.
5. Чіткість та зручність користувацького інтерфейсу, розробленого на Flutter.
6. Відповідність стилю застосунка UI/UX вимогам технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування буде проводитися на рівні системного тестування за методикою "Сірий ящик", де ми аналізуємо як внутрішні механізми коду, так і функціональність застосунку. Представлені такі методики:

1. Функціональне тестування для перевірки коректності роботи основних функцій застосунку.

2. Тестування продуктивності, включаючи тестування навантаження (Load Testing) та тестування стабільності (Stability Testing) для аналізу роботи застосунку при високому навантаженні.
3. Тестування користувацького інтерфейсу для перевірки зручності та інтуїтивності роботи з застосунком.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування буде проводитися за допомогою інструментів, таких як Jest (для Node.js) та Flutter Test (для Flutter). Тестування застосунку буде включати:

1. Ручне тестування для валідації роботи користувацького інтерфейсу та функцій.
2. Автоматизоване тестування для перевірки відповідності коду функціональним вимогам.
3. Тестування API для перевірки взаємодії з сервером.
4. Тестування на різних версіях операційних систем.
5. Тестування при високому навантаженні для перевірки стабільності застосунку.
6. Тестування UI/UX для перевірки зручності користування застосунком.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ САМООРГАНІЗАЦІЇ ТА ВЗАЄМОДОПОМОГИ СПІВВІТЧИЗНИКІВ ЗА КОРДОНОМ

Керівництво користувача

ДП.045490-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Юрій ВУНТЕСМЕРІ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____Андрій ПРОЦЕНКО

2023

ЗМІСТ

1. ОПИС СТРУКТУРИ МОБІЛЬНОГО ЗАСТОСУНКУ

Мобільний застосунок для самоорганізації та взаємодопомоги співвітчизників за кордоном складається із динамічних сторінок. Інтерфейс застосунку має українську мову.

Мобільний застосунок має такі сторінки: сторінка авторизації/реєстрації користувача, сторінка з персональними даними користувача, сторінка редагування персональних даних користувача, сторінка з інтерактивною мапою та мітками, сторінка створення міток на мапі, сторінка редагування міток на мапі, сторінка приватних чатів, сторінка публічних чатів, сторінка створення публічних чатів.

Для усіх користувачів доступ до сторінок застосунку в першу чергу здійснюється за допомогою бокової панелі та подальших кнопок на сторінках.

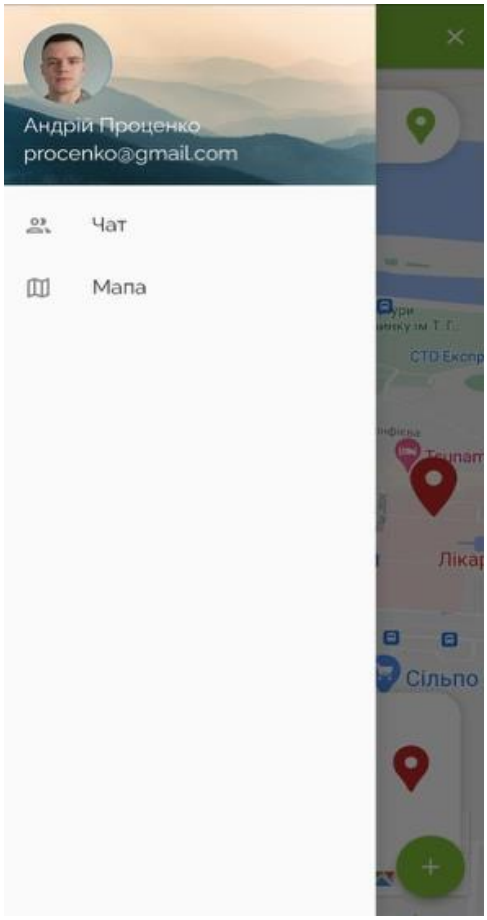


Рис. 1.1. Бокове меню мобільного застосунку

2. АВТОРИЗАЦІЯ ТА РЕЄСТРАЦІЯ У МОБІЛЬНОМУ ДОДАТКУ

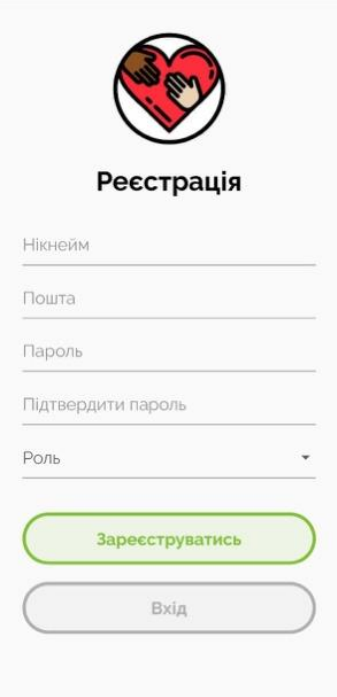


Рис. 2.1. Сторінка реєстрації користувача

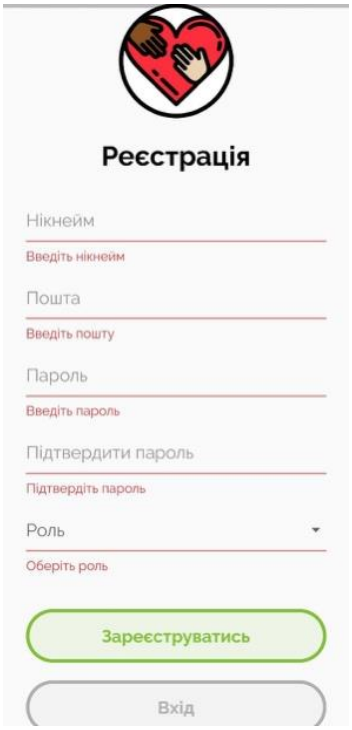


Рис. 2.2. Помилка при реєстрації

Рис. 2.3. Форма авторизації користувача

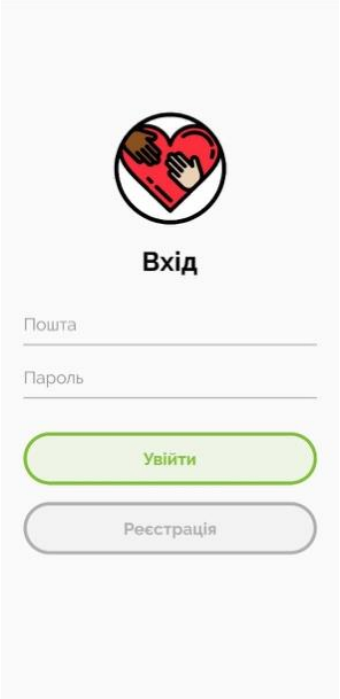


Рис. 2.4. Помилка при авторизації

Для доступу до сервісів мобільного застосунку кожен користувач повинен пройти процедуру реєстрації або авторизації.

Форма реєстрації має такі поля: поле імені/нікнейму, поле унікальної електронної пошти, поле паролю та маркер ролі (рис. 2.1). При введені некоректних даних або пустих полів, з’явиться помилка (рис. 2.2).

Після цього акаунт користувача буде успішно створено та його автоматично буде перенаправлено на сторінку мапи з мітками (рис. 2.3).

Форма авторизації має такі поля вводу як електрона пошта користувача та пароль (рис. 2.4). Якщо запис користувача з такими логіном або паролем не буде знайдено, форма оновиться та виведе повідомлення про помилку.

Після успішної авторизації користувача перенаправляє на сторінку мапи з мітками (рис. 2.5).

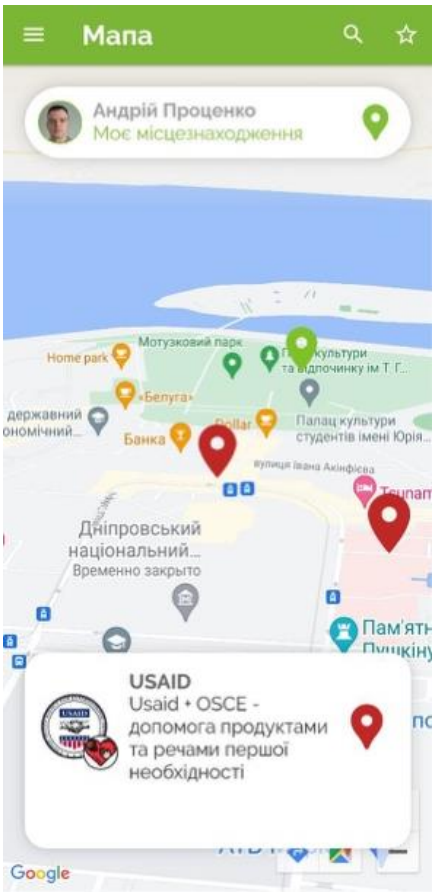


Рис. 2.5. Перенаправлення на сторінку мапи з мітками

3. РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОЇ МАПИ ТА МІТОК

Одною з основних складових цього застосунку є інтерактивна мапа з мітками допомоги. Згідно функціональних вимог будь-який користувач може переглядати сторінку мапи з мітками та пошуком (рис. 3.1).

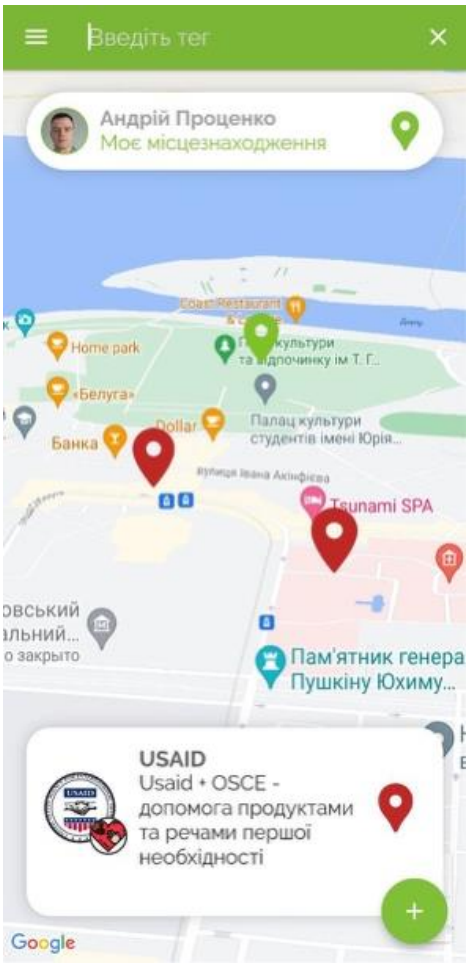


Рис. 3.1. Мапа з мітками та пошуком

Тепер розглянемо сценарій використання мапи Реципієнтом або Помічником. Окрім описаного вище функціоналу, вони мають кнопку справа зверху (рис. 3.2) для збереження цікавих їм міток у список «обраних» (рис. 3.3).

Якщо додаток використовує Помічник або Волонтер, вони мають можливість додавати нові мітки на мапу через взаємодію з кнопкою справа знизу екрану та натиск на мапу для обрання локації розташування (рис. 3.4)

через обрання локації шляхом натискання на бажаній позиції на мапі, і заповнення даних нової мітки через форму створення (рис. 3.5).

Після заповнення усієї необхідної інформації для створення мітки, вона буде створена на мапі (рис. 3.6).

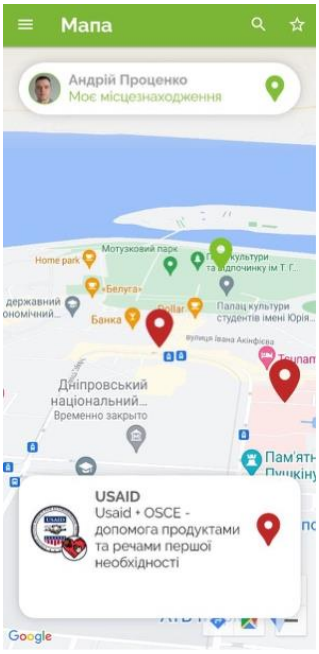


Рис. 3.2. Кнопка для збереження міток у список «обраних»

Рис. 3.3. Список збережених міток у «обраному»

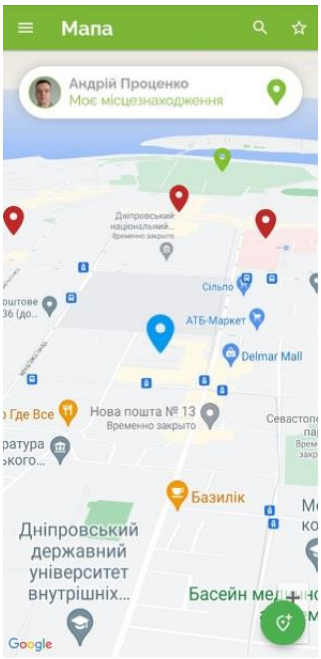


Рис. 3.4. Кнопка додавання міток

Рис. 3.5. Форма додавання мітки

Рис. 3.6. Створена мітка

Створені Помічником або Волонтером мітки можуть бути відредаговані (рис. 3.7) або видалені (рис. 3.8).

Рис. 3.7. Редагування мітки

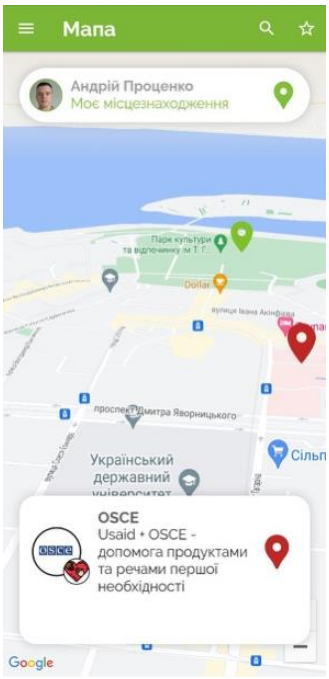


Рис. 3.8. Видалення мітки з мапи

4. РЕАЛІЗАЦІЯ ПРИВАТНИХ ТА ПУБЛІЧНИХ ЧАТІВ

Після обрання вкладки чатів на боковому меню користувачу відкриється сторінка усіх чатів, які поділяються на 2 групи: приватні та публічні (рис. 4.1, рис. 4.2).

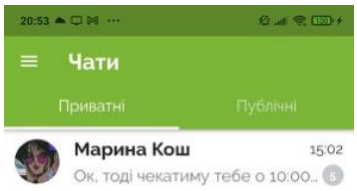


Рис. 4.1. Приватні чати

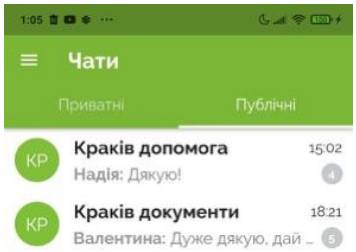


Рис. 4.2. Публічні чати

У приватних чатах відображуються діалоги користувача з іншими користувачами додатку з кожним по-окремості. Щоб створити приватний чат користувачу треба натиснути на кнопку справа знизу сторінки та заповнити інформацію про створюваний чат (рис. 4.3). Нижче приведена демонстрація роботи приватного чату (рис. 4.4).

У публічних чатах діалоги відбуваються між усіма користувачами що зайшли у цей діалог. Вони відкриті для усіх користувачів, тож кожен може приєднатися до них. Щоб створити публічний чат користувачу треба натиснути на кнопку справа знизу сторінки та написати назву створюваного чату (рис. 4.5). Після цього він буде доступний для написання повідомлень

(рис. 4.6) Нижче також приведена демонстрація роботи публічного чату (рис. 4.7, рис. 4.8).

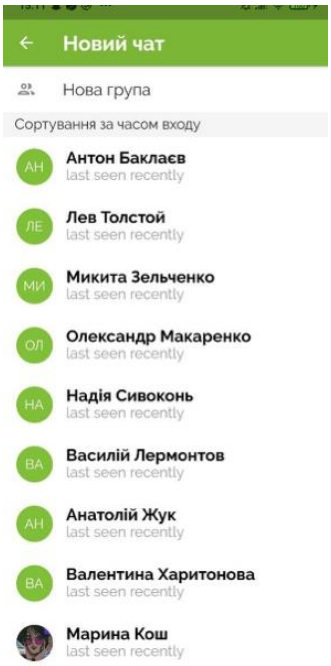


Рис. 4.3. Створення приватного чату

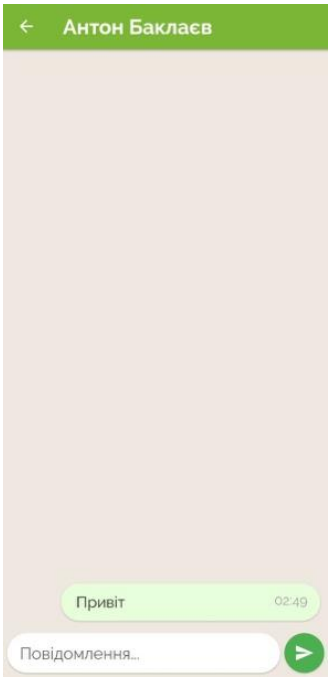


Рис. 4.4. Приватний чат між користувачами

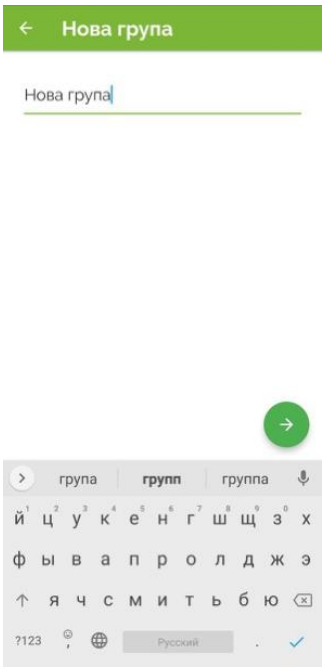


Рис. 4.5. Створення публічного чату

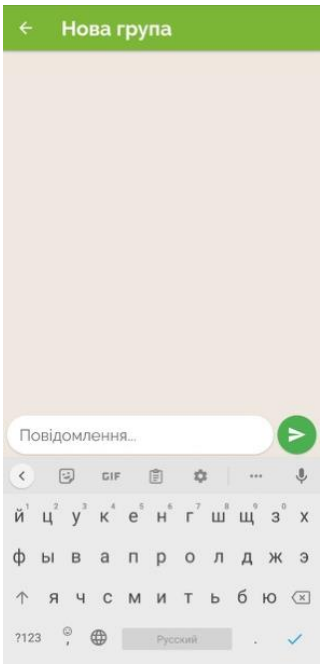


Рис. 4.6. Демонстрація створеного публічного чату

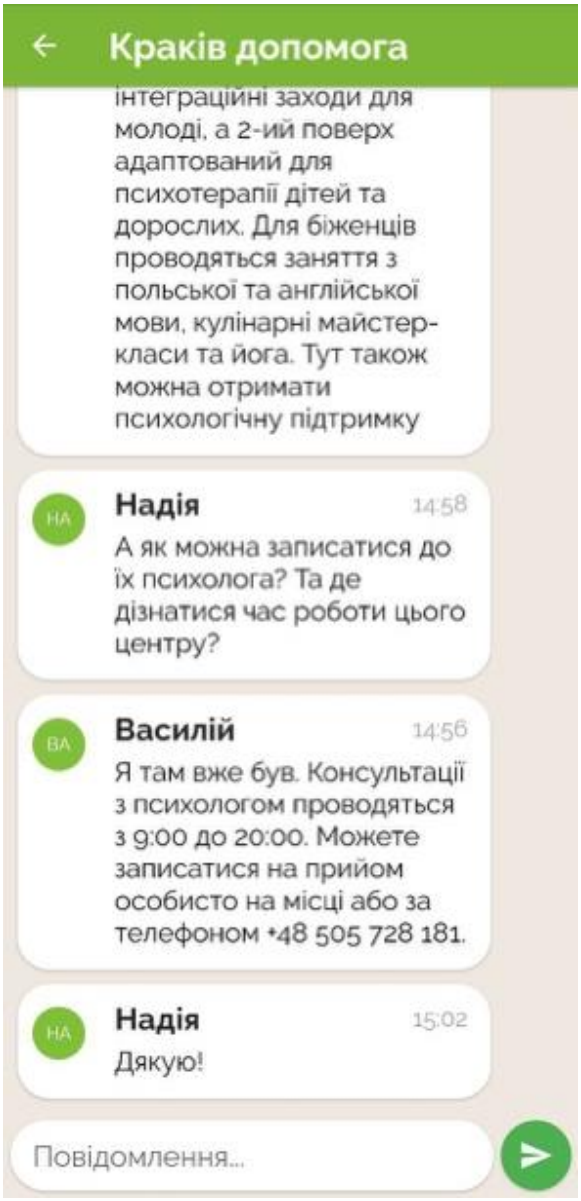


Рис. 4.7. Публічний чат допомоги

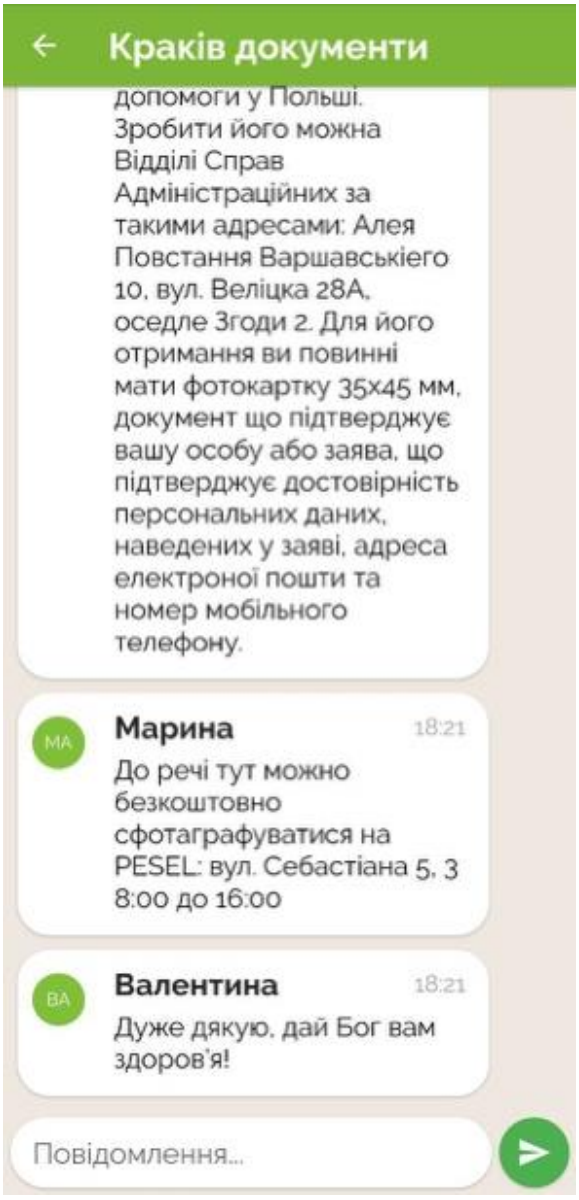


Рис. 4.8. Публічний чат документів

5. ПРОФІЛЬ КОРИСТУВАЧА

Також користувачі мають можливість редагувати свої дані у профілі через натискання на поле з їх інформацією у боковому меню (рис. 5.1).

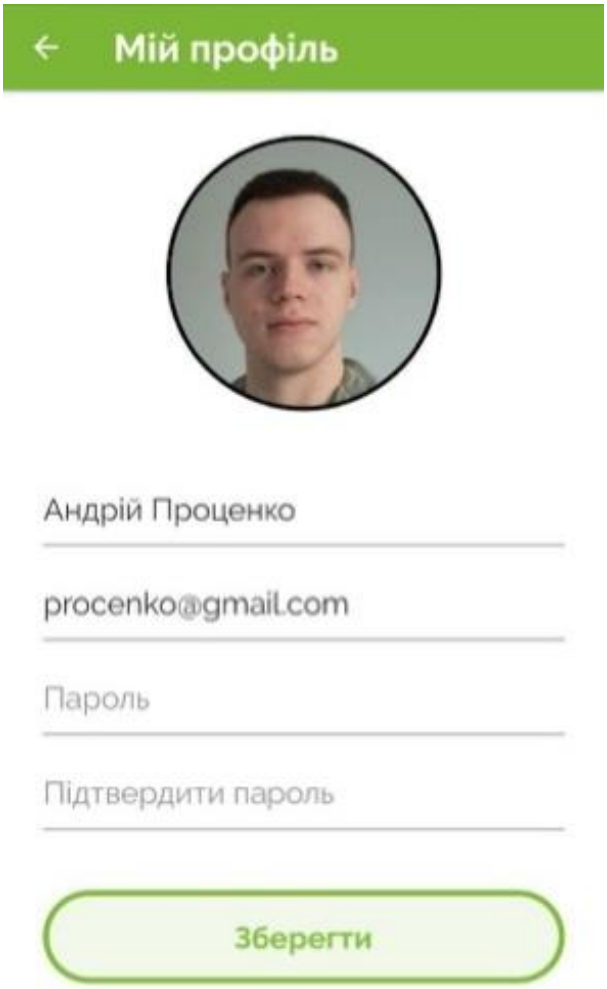


Рис. 5.1. Редагування інформації у профілі