МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерна інженерія»

Група: 2БКС-28

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

здобувача освіти денної форми навчання

БКС.28.02.000.КРБ

ВЄЛКОВА
ІВАНА ВАСИЛЬОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**
Освітньо-професійна програма: **«Комп'ютерна інженерія»**
Група: **2БКС-28**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему: **«Розробка геоінформаційного мобільного додатку для ідентифікації пунктів незламності»**

Проектний матеріал складається з пояснювальної записки на 93 сторінках та графічного (презентаційного) матеріалу на 20 аркушах (слайдах)

Виконавець _____ (Вєлков І.В.)
Керівник проекту _____ (Іванова Л.В.)

Консультанти:
з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)
з нормоконтролю _____ (Петрашова В.І.)
старший консультант _____ (Кривченко Ю.В.)

До захисту допущений
Завідувач кафедри _____ (Іванова Л.В.)
Завідувач відділення _____ (Скорнякова О.В.)

Захист « 24 » 06 2024 р. Протокол ЕК № 1
Оцінка ЕК 5 (відмінно) 95.

Секретар ЕК _____

АНОТАЦІЯ

У випускній кваліфікаційній роботі розглянуто методи та засоби створення мобільних додатків геоінформаційного типу під ОС Android. При цьому створено реально діючий додаток для ідентифікації пунктів незламності.

У випускній кваліфікаційній роботі виконано аналітичний огляд існуючих рішень, визначена актуальність створення подібного додатку. Обґрунтовано вибір технологій для створення геоінформаційних систем та мобільних додатків. Визначено переваги недоліки кожного з них, обрано найбільш зручний і доречний для виконання технічного завдання фреймворк, мову програмування та сервіси. Виконано аналіз архітектури та компонентів фреймворку Flutter, можливості мови програмування Dart. Визначені сервіси для інтеграції графічних карт у мобільні додатки.

Обрано архітектуру та компоненти створюваного мобільного додатку, розроблені варіанти його використання, розроблено загальну структуру геоінформаційного додатку. Реалізовано архітектуру мобільного додатку на базі API, створено схему клієнт-серверної взаємодії.

Реалізовано Frontend- та Backend-частину мобільного додатку мовою програмування Dart, його елементи та візуальний інтерфейс користувача. Розроблено та реалізовано блок-схему алгоритму обробки дій користувача у додатку, виконано його тестування та розміщення у магазині додатків Google Play для вільного скачування. Розроблений мобільний додаток призначений для надання користувачам інформації про розташування найближчих пунктів незламності, де забезпечено доступ до мережі електроживлення.

Описано заходи з охорони праці та техніки безпеки.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *Комп'ютерних систем* Кафедра *Комп'ютерної інженерії*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"11" 07 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

здобувачеві освіти *Вєлкова Івана Васильовича*

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Розробка геоінформаційного мобільного додатку для ідентифікації пунктів незламності*

затверджена наказом по коледжу від "02" 11 2023 р. № 244-А2-ОД

2. Термін здачі студентом кваліфікаційної роботи *15.06.2024*

3. Вихідні дані до роботи *Вимоги до геоінформаційних інформаційних систем*
Функціональні вимоги до програмного забезпечення
Нефункціональні вимоги до програмного забезпечення
Вимоги до дизайну та інтерфейсу користувача

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1. Аналітичний огляд існуючих рішень. 2. Обгрунтування вибору технологій для створення геоінформаційних систем та мобільних додатків. 3. Архітектура мобільного додатку. 4. Розробка елементів дизайну та інтерфейсу користувача. 5. Створення макету та прототипу мобільного додатку. 6. Тестування мобільного додатку. 7.Охорона праці. 8. Висновки. 9. Список використаних джерел інформації

5. Перелік графічного матеріалу (слайдів мультимедійної презентації)

1. *Порівняльний аналіз існуючих рішень.*
2. *Технології геоінформаційних систем та мобільних додатків.*
3. *Архітектура мобільного додатку.*
4. *Макет мобільного додатку*
5. *Скріншоти /відео роботи мобільного додатку*

6. Консультанти по кваліфікаційній роботі, із зазначенням розділів, що їх стосуються

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Основний розділ	*Іванова Л.В.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 15.01.2024

Керівник роботи *Іванова Л.В.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Аналіз технічного завдання та пошук джерел інформації*	*4.05.2024*	Виконано
2.	*Формування функціональних вимог до ПЗ*	*8.05.2024*	Виконано
3.	*Визначення технічного завдання на розробку*	*10.05.2024*	Виконано
4.	*Вибір технологій для реалізації додатку*	*12.05.2024*	Виконано
5.	*Розробка арфітектури мобільного додатку*	*15.05.2024*	Виконано
6.	*Розробка алгоритмів роботи додатку*	*18.05.204*	Виконано
7.	*Розробка елементів дизайну*	*22.05.2024*	Виконано
8.	*Розробка елементів інтерфейсу*	*24.05.2024*	Виконано
9.	*Створення макету додатку*	*26.05.2024*	Виконано
10.	*Створення прототипу*	*28.05.2024*	Виконано
11.	*Тестування ПЗ*	*31.05.2024*	Виконано
12.	*Тестування виконання вимог користувача*	*2.06.2024*	Виконано
13.	*Перевірка роботи на плагіат*	*6.06.2024*	Виконано
14.	*Підготовка до малого захисту*	*10.06.2024*	Виконано
15.	*Розробка питань з охорони праці*	*12.062024*	Виконано
16.	*Підготовка роботи до захисту та тестування ПЗ*	*15.06.2024*	Виконано
17.	*Оформлення слайдів мультимедійної презентації*	*18.06.2024*	Виконано

Здобувач освіти _____
(підпис)

Керівник роботи _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 28. 02 000. 00 КРБ	Випускна робота		
А4			БКС 28. 02 000. 00 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 28. 02 000. 00 КРБ		
Розробив		Вєлков І.В.		10.06	Розробка геоінформаційного мобільного додатку для ідентифікації пунктів незламності	Літ.	Аркуш 4 / Аркушів 93
Перевірив		Іванова Л.В.		10.06		Н Д П	
Н. Контр.		Петрашова В.І.		10.06		ВСП "ОТФК ОНТУ" гр.2БКС-28	
Затверд.		Іванова Л.В.		10.06			

ЗМІСТ

ВСТУП

Відомо, що завдяки сучасним мобільним додаткам можливо одержати швидкий доступ до різноманітної інформації і виконувати різні завдання і спілкуватися із користувачами у будь-котрий час й із будь-якого місця. Мобільні додатки дозволяють застосовувати смартфони і планшети для розваг, освіти, роботи, здоров'я, фітнесу, подорожей і багатьох інших сфер життя. Вони пропонують широкий вибір функціональності, включаючи сповіщення, геолокацію, камеру, датчики, інтеграцію із соціальними мережами і немало іншого.

ІЗ ростом популярності мобільних додатків зростає й їхня роль в різноманітних галузях. Розроблення мобільних додатків вимагає не тільки технічної експертизи, але й уважного аналізу потреб користувачів і створення зручного і привабливого інтерфейса. Постійне зростання ринку мобільних додатків викликає конкуренцію серед розробників і підтримку нових інновацій й технологій. Мобільні додатки надають більш персоналізований і зручний користувацький досвід порівняно із веб-сайтами. Вони спроможні застосовувати функції пристрою, такі як камера, GPS і сповіщення, для створення більш інтерактивного і привабливого досвіду. Застосування мобільних функцій дозволить розробникам утворювати унікальні і інноваційні функції, котрі використовують потужні можливості мобільних пристроїв. Зокрема, додаток може застосовувати геолокацію для визначення місцеположення користувача і надання актуальної інформації відносно найближчих об’єктів, визначати стан окремих об’єктів і керувати ними. Мобільні додатки дозволяють утворювати більш глибоку і безпосередню взаємодію із користувачами крізь застосування різноманітних функцій, таких як оповіщення, сповіщення, інтерактивність жестами і немало іншого. Це дозволить створити більш імерсивний і захоплюючий досвід для користувачів, що сприяє залученню і задоволенню клієнтів. Однією із переваг мобільних додатків є їх здатність працювати у offline-режимі. Це означає, що користувачі спроможні застосовувати додаток без доступу до Інтернету. Зокрема, в разі повітряної тривоги, коли з'єднання може бути обмеженим, мобільний додаток

може надавати користувачам доступну інформацію про найближчі укриття на основі технології GPS-позиціювання, навіть в відсутності мережі GSM.

Розроблення мобільного додатку на базі фрейм-ворка Flutter дозволить швидко і ефективно застосовувати ці переваги, надаючи клієнтам зручну, інноваційну і персоналізовану взаємодію. Важливість реалізації гео-інформаційного мобільного додатку, що вміщує інформацію про пункти незламності, стає особливо актуальною і необхідною. Забезпечення безпеки даних від кібератак і несанкціонованого доступу відіграє вирішальну роль в підтримці довіри користувачів до мобільного додатку. Це дозволить здійснити конфіденційність, цілісність і доступність інформації, що є важливим для функціонування будь-якого проекту. Так само крім безпеки, реалізація гео-інформаційного мобільного додатку для ідентифікації пунктів незламності має ще один важливий аспект – забезпечення надійності і стабільності функціонування застосунку. Враховуючи потужність інфраструктури і надійність системи, можливо запобігти можливим перебоям у роботі, зниженню продуктивності і негативним наслідкам для користувачів. Реалізація такого мобільного додатку із інформацією про пункти незламності так само сприяє забезпеченню відновлюваності і резервуванню даних. Це важливо для збереження інформації у разі випадкового видалення, системних збоїв чи інших непередбачуваних ситуацій.

Метою роботи є розроблення гео-інформаційного мобільного додатку ESvitlo!, котрий має надати користувачам інформацію про розміщення найближчих пунктів незламності і допомогти їм віднайти місця, де забезпечено доступ до мережі електроживлення, але в режимі адміністратора – додавати на карту додатку окремі заклади (кафе, ресторани, бізнес-центри й т.д.). Таким чином, розроблюваний додаток спрямований на зберігання і відображення інформації про пункти незламності й має стати ефективним і зручним інструментом, що дозволить користувачам легко віднайти і одержати доступ до необхідних даних про місця, де забезпечується безперебійний доступ до важливих ресурсів. Так само протягом виконання роботи необхідно проаналізувати сучасні технології розроблення мобільних додатків, що використовують клієнт-серверні моделі.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

1.1.1 Визначення актуальності створення додатку

На сьогоднішні день розгортання пунктів незламності у Україні має соціально-економічне значення. Країна стикається із економічними викликами, агресією ворога і соціальною нерівністю, що призводить до того, що деякі групи населення мають обмежений доступ до резервних джерел енергії і можливості здійснити собі безпечне місце перебування під час відключення електроенергії крізь атаки ворога на об’єкти критичної інфраструктури.

«Пункт незламності» – проект української влади, котрий ініціювали в листопаді 2022 року на фоні ушкодження критичної інфраструктури під час російсько-української війни. Пункти незламності є місцями, де передбачені тепло, вода, електрика, мобільний зв'язок тощо. Зважаючи на це, пункти незламності спроможні допомогти здійснити рівні можливості для всіх громадян, забезпечуючи їм необхідні умови перебування у умовах відключення електропостачання.

Важливо зазначити, що проблема пунктів незламності у Україні має не тільки матеріальний аспект, але й соціальний і психологічний. Відключення електроенергії і тепла може викликати почуття безпорадності, страху і вразливості в населення. Пункти незламності спроможні здійснити не тільки фізичну безпеку, але й підтримку, сприяючи збереженню психологічного благополуччя і єдності громади. Відповідно до вищезазначеного, актуальність проблеми пунктів незламності у Україні на сьогоднішній день надзвичайно висока.

Забезпечення безпеки, тепла і підтримки населення під час тривалого відключення електроенергії є невідкладним завданням. Розвиток і впровадження пунктів незламності, розширення їх мережі і покращення їх функціональності є важливими кроками для забезпечення безпеки і добробуту українського населення.

Додатки для пошуку інформації про пункти незламності допомагають зменшити бар'єри й покращити доступність для тих, хто бажає віднайти місце із постійною електроенергією і опаленням. За поміччю мобільного додатку

потенційні користувачі зможуть швидко віднайти пункт незламності і ідентифікувати їх. Отже, додаток із пошуком пунктів незламності може стати цінним інструментом для поміччі людям в знаходженні безпечних місць перебування під час тривалого відключення електроенергії. Він надає спроможність швидко і зручно віднайти доступні пункти незламності в конкретній локації, що може бути особливо корисним під час екстрених ситуацій, котрі нажаль цілком можливі в 2024 році.

1.1.2 Огляд існуючих рішень

Для успішної розроблення і впровадження гео-інформаційного мобільного додатку для ідентифікації пунктів незламності необхідно провести аналіз ринку відповідного програмного забезпечення.

При виборі теми випускної роботи ставилася мета задовольнити ринкову потребу в зручному і доступному способі отримання інформації про пункти незламності. Зростаюча популярність мобільних додатків, спрямованих на безпеку і надзвичайні ситуації, свідчить про високу зацікавленість користувачів в таких рішеннях. Попередній аналіз конкурентного середовища вказує на наявність схожих додатків на ринку, проте в розроблюваного додатку передбачається так само потенційне керування вмиканням/вимиканням електроживлення в окремому закладі, простота застосування.

Оцінка потенційних конкурентів допоможе удосконалити розроблюваний програмний продукт і розробити ефективну маркетингову стратегію для залучення користувачів. Так само важливим елементом аналізу ринку є вивчення цільової аудиторії і її потреб. Зусилля будуть направлені на привернення уваги і задоволення потреб потенційних користувачів, надаючи їм цінну інформацію і забезпечуючи доступ під час виняткових ситуацій. Загалом, аналіз ринку підтверджує актуальність і потенціал успіху гео-інформаційного мобільного додатку для ідентифікації пунктів незламності.

У Україні і світі на сьогоднішній день існує декілька застосунків і платформ, корисних для пошуку пунктів незламності. Деякі із них розглянуто далі.

Веб-додаток Nezlamnist спирається на офіційну платформу Пункт

незламності (nezlamnist.gov.ua) у Україні й є надає інформацію про пункти незламності і безпечні місця у разі виняткових ситуацій. Ця платформа створена із метою забезпечення безпеки і підтримки населення під час виняткових ситуацій, зокрема відключення електроенергії.

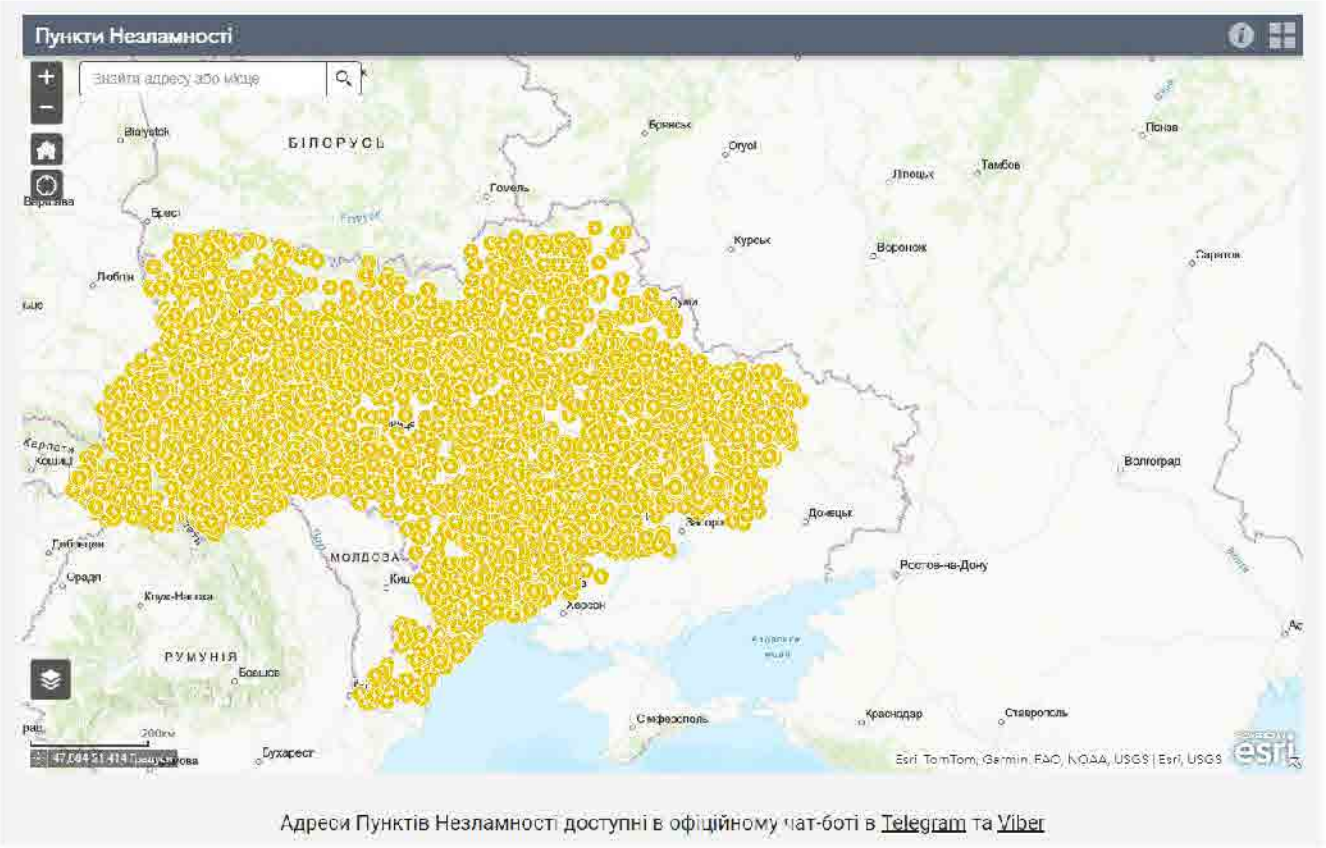


Рисунок 1.1. Інтерфейс веб-додатку Nezlamnist із розташуванням пунктів незламності на карті України

Веб-додаток Nezlamnist передбачає сервіс, що надає користувачам ряд можливостей для пошуку і надання інформації відносно пунктів незламності, але саме:

- сервіс Nezlamnist дозволить користувачам швидко знаходити розміщення пунктів незламності на території. В користувача є спроможність шукати притулки, лікарні, пункти збору і інші об'єкти, котрі спроможні бути важливими у виняткових ситуаціях;
- користувачі мають спроможність одержати детальну інформацію про кожний пункт незламності, включаючи адресу, контактні дані і послуги, котрі надаються. Це допомагає здійснити належну підготовку

і знати, куди звернутися в разі потреби.

Веб-додаток Nezlamnist має наступні переваги:

- сервіс надає інформацію про пункти незламності по всій території України;
- платформа регулярно оновлюється, аби здійснити користувачам актуальну й достовірну інформацію про пункти незламності;
- сервіс надає інформацію про різні категорії пунктів незламності.

Веб-додаток Nezlamnist має наступні недоліки:

- дизайн вебсайту не є досить зручними і інтуїтивно зрозумілими для користувачів;
- спроможність реєстрування пункту незламності крізь гугл форму;
- відсутність вбудованого чату на сайті. Для зв'язку між користувачами
- сервісу необхідно користуватись сторонніми додатками, такими як Viber чи Telegram, котрі спроможні бути не завжди доступними в надзвичайній ситуації.

Таким чином веб-додаток Nezlamnist має свої переваги й недоліки, котрі варто враховувати при оцінці його ефективності. ІЗ одного боку, сервіс надає користувачам широкий спектр можливостей, включаючи швидкий пошук пунктів незламності, доступ до детальної інформації про них. Це дозволить людям здійснити належну підготовку і знати куди звернутися в виняткових ситуаціях. Крім того, незалежно від переваг, варто пам'ятати, що сервіс базується на наявності актуальних даних й оновленнях, що спроможні вплинути на точність й доступність інформації. Загалом, Сервіс Nezlamnist може бути корисним інструментом для користувачів в пошуку пунктів незламності, але слід зважати на необхідність додаткової перевірки актуальності інформації і його обмеження.

Веб-додаток Shelter safe (sheltersafe.ca) є вебплатформою, розробленою для надання інформації про пункти поміччі і укриття у Канаді. Цей сервіс створений із метою допомогти особам, котрі перебувають в вразливому стані, віднайти безпечний притулок в різноманітних регіонах.

Веб-додаток Shelter safe передбачає сервіс, що надає користувачам ряд

можливостей для пошуку і надання інформації відносно пунктів поміччі, але саме:

- дозволить користувачам швидко віднайти найближчі до їх поточного місцезнаходження пункти поміччі, такі як притулки, лікарні, поліцейські дільниці і інші важливі об'єкти;
- сервіс надає інтерактивну карту із позначками розміщення пунктів незламності по всій території;
- користувачі спроможні зручно переглядати цю карту й знаходити потрібні об'єкти у конкретному регіоні чи місті. Користувачі спроможні одержати докладну інформацію про кожний пункт поміччі, таку як його адреса, контактні дані, час роботи, послуги, котрі надаються і інші важливі відомості. Це дозволить людям знати, куди звернутися в разі потреби.

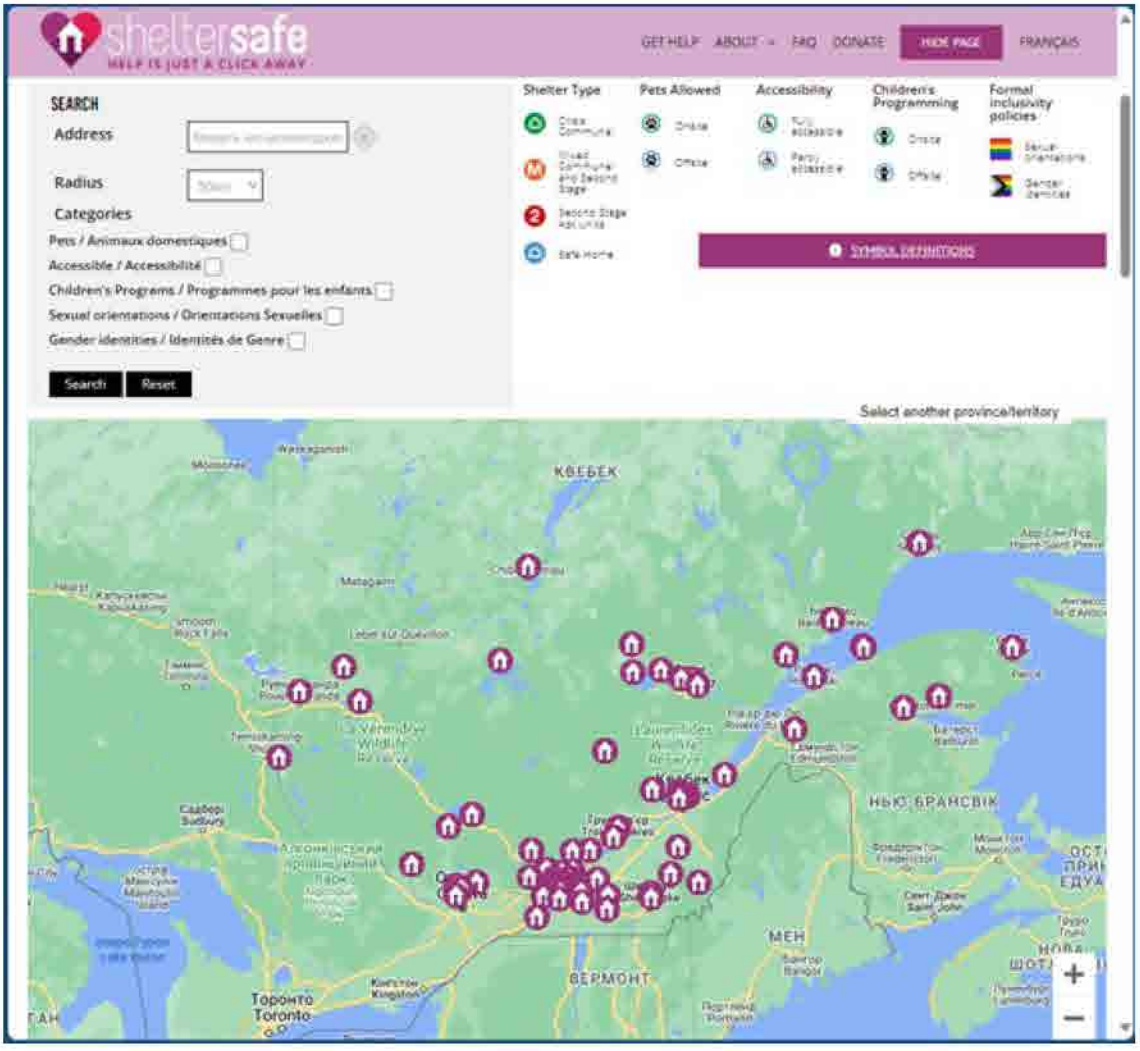


Рисунок 1.2. Інтерфейс веб-додатку Shelter safe із розташуванням пунктів допомоги на карті Канади

Переваги веб-додатку Shelter safe:

- доступ до інформації;
- різноманітність послуг;
- інтерактивна карта;
- детальна інформація про пункти допомоги;
- наявність мобільного додатку під різні платформи.

Недоліки веб-додатку Shelter safe:

- обмежене покриття географії;
- відсутність детальної інформації;
- відсутність безпосередньої підтримки.

Сервіс Shelter safe має кілька переваг, зокрема широкий вибір категорій і послуг для пошуку укриття, детальна інформація про кожний пункт поміччі, наявність мобільних додатків для зручного доступу і спроможність користувачів залишати відгуки і оцінки. Однак, сервіс так само має свої недоліки, зокрема обмежену базу даних пунктів поміччі, іноді неінтуїтивний інтерфейс і відсутність чат-бота безпосередньо на сайті. Незважаючи на це, Shelter Safe є корисним інструментом для пошуку безпечного укриття.

Веб-додаток American Red Cross (www.redcross.org) є офіційним вебсайтом Американського Червоного Хреста. Червоний Хрест є глобальною гуманітарною організацією, котра надає допомогу у разі кризових ситуацій. Сайт вміщує широкий спектр інформації, ресурсів і послуг, котрі спроможні бути корисними у виняткових ситуаціях. На сайті можливо віднайти інформацію про пункти поміччі, розміщення притулків, допомогу постраждалим від стихійних лих і немало іншого. Вебсайт так само надає спроможність зробити внесок, стати волонтером чи зареєструватися для отримання навчання із першої поміччі. Сервіс American Red Cross надає ряд можливостей для пошуку пунктів поміччі і отримання додаткової інформації про них, спроможність швидкого й зручного пошуку пунктів поміччі за різними характеристиками, такими як місцезнаходження, тип пункту і доступні послуги. Передбачено спроможність отримання докладної інформації про кожний пункт поміччі, включаючи адресу, контактні дані, графік роботи і опис наданих

послуг; спроможність перегляду відгуків й оцінок інших користувачів, що дозволить одержати об'єктивну думку про пункт поміччі від людей, котрі вже скористалися ним; наявність додаткової інформації про безпеку і поради відносно поведінки у виняткових ситуаціях.

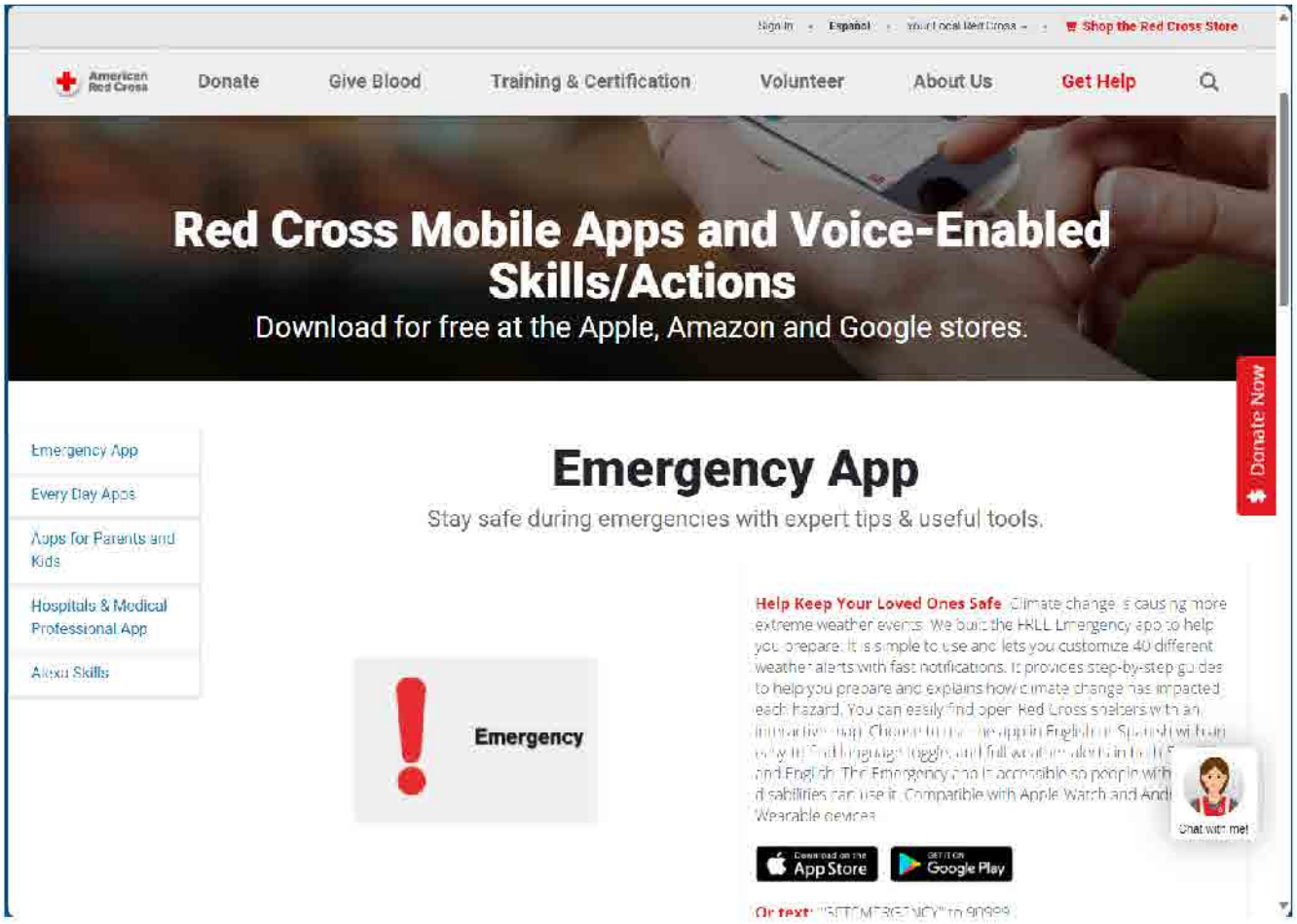


Рисунок 1.3. Інтерфейс веб-додатку American Red Cross для скачування мобільного додатку

Переваги веб-додатку American Red Cross:

- розширена база даних про пункти поміччі;
- зручний і легкий пошук пунктів поміччі;
- наявність системи відгуків і оцінок;
- додаткова інформація і поради відносно безпеки і дій в виняткових ситуаціях.

Недоліки веб-додатку American Red Cross:

- обмежене географічне покриття сервісу;
- відсутність безпосередньої підтримки чи можливості одержати

консультацію у режимі реального часу;

– залежність від актуалізації і оновлення бази даних.

Враховуючи переваги і недоліки, сервіс American Red Cross може бути цінним ресурсом для отримання інформації про пункти поміччі, але користувачам слід бути свідомими відносно його обмежень.

1.1.3 Визначення вимог до розроблюваного гео-інформаційного мобільного додатку

Відповідно до виконаного вище аналізу можливо відзначити, що реалізація й подальше супроводження мобільного додатку для надання інформації відносно пунктів незламності є вкрай важливим для вирішення проблем, як у Україні, так й за кордоном. Розроблення такого гео-інформаційного мобільного додатку є важливим аспектом із огляду на покращення доступності і якості послуг для користувачів. Розглянуті в п.1.1.2 додатки, такі як Nezlamnist, Shelter safe і American Red Cross, хоч й надають певний рівень інформації, проте не відповідають в повній мірі потребам користувачів. Тому розроблюваний геоінформаційний мобільний додаток ESvitlo! має надавати користувачеві карту розміщення пунктів незламності, в якості котрих спроможні бути громадські заклади (кафе, ресторани, бізнес-центри й т.д.) із більш змістовною інформацією, зокрема їх статусу, додаткових послуг, меню й т.д., що дозволить користувачам мати повне уявлення про них. Все це може мати значний вплив на покращення доступності і організації інформації про пункти незламності для користувачів.

Шляхом поєднання функціональності пошуку, статусу наявності електроживлення в закладі у даний час, детальних описів і, зокрема, наявного меню, такий додаток створить спроможність віднайти надійні і безпечні місця в разі виняткових ситуацій, де так само можливо провести час “за чашкою кави”. За поміччю цього додатку користувачі зможуть легко зорієнтуватись, в котрих громадських закладах м.Одеса наразі є електроживлення, і одержати докладну інформацію про їх розміщення на карті, оснащення, додаткові послуги. Це значно полегшить пошук безпечних місць для людей, котрі шукають притулок, медичну допомогу чи інші необхідні послуги під час виняткових ситуацій.

1.2 Обґрунтування вибору технологій для створення геоінформаційних систем і мобільних додатків

Для розроблення додатку буде використано компонентний підхід створення інтерфейса користувача і принципи об'єктно-орієнтованого програмування.

Вибір фреймворка для розроблення мобільного додатку залежить від потреб розробника і вимог проекту. В світі мобільних додатків є немало інструментів для розроблення мобільних додатків, тому розробники повинні знати про найбільш популярні фреймворки і їх переваги і недоліки. React Native, Flutter, Ionic, Xamarin і Phone-gap є найпопулярнішими фреймворками для розроблення мобільних додатків в 2023-2024 році. Кожний із них має свої особливості і переваги, котрі дозволяють розробникам утворювати високоякісні мобільні додатки для різноманітних платформ.

1.2.1 Аналіз фреймворків для розроблення мобільних додатків

В світі мобільних додатків на ринку з'являється все більше технологій, що дають змогу розробляти високоякісні і продуктивні мобільні додатки. Далі описано п'ять найпопулярніших фреймворків, котрі зараз на піку популярності.

Фрейм-ворк React Native (рис.1.4) є одним із найпопулярніших фреймворків для розроблення мобільних додатків. Він є фреймворком із відкритим кодом, розробленим Facebook. React Native дозволить розробникам утворювати мобільні додатки для платформ Android і iOS із використанням JavaScript. Поява фрейм-ворка React Native в 2015 році викликала хвилю на ринку гібридних фреймворків. Протягом кількох років після його виходу на ринок він став найбільш популярним серед 5 платформ, що обговорюються у спільноті розробників. Цей фрейм-ворк дозволить розробникам ефективно застосовувати переваги реактивного програмування і декларативного стилю коду. React Native так само дає змогу розробникам застосовувати немало звичайних елементів, котрі доступні у бібліотеці React, тому розроблення мобільних додатків із React Native є дуже швидкою і ефективною.

Фрейм-ворк React Native має наступні переваги:

– до 80% кодової бази можливо розподіляти між платформами залежно від складності програми. Окрім багаторазового застосування коду це дозволить відразу переглядати результати, крім того, є готові до застосування елементи, тим самим значно скорочуючи час розроблення;
– функція «гарячого перезавантаження» дозволить розробникам бачити зміни, внесені у код, протягом декількох секунд;
– фрейм-ворк React Native фокусується значною мірою на UI, надаючи високочутливий інтерфейс;
– фрейм-ворк React Native так само надає доступ до таких функціональних можливостей пристрою, як акселерометр і камера.

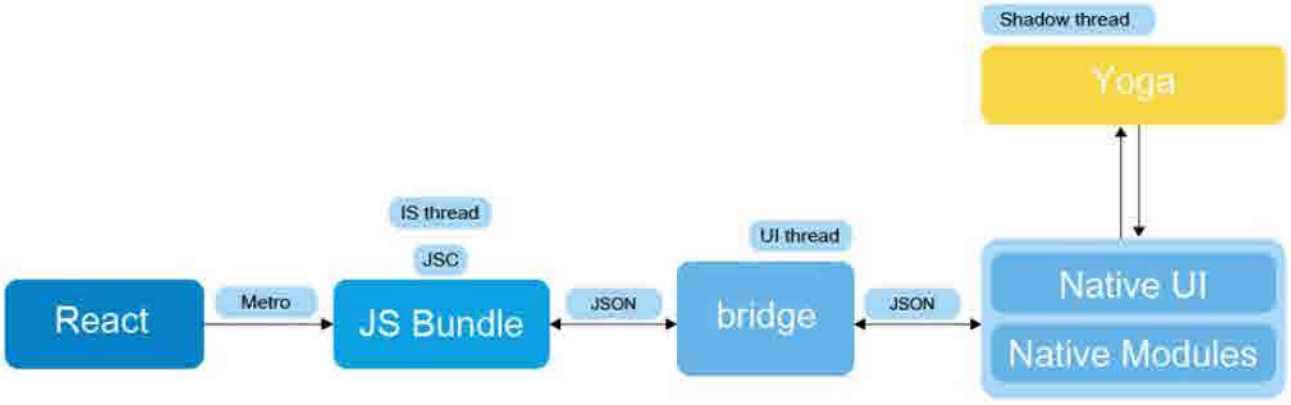


Рисунок 1.4. Спрощена архітектура фрейм-ворка React Native

Фрейм-ворк React Native має наступні недоліки:

– фрейм-ворк React Native не є повністю крос-платформним фреймворком додатків. Для застосування деяких функцій, таких як камера чи акселерометр, треба застосовувати власні компоненти, тому буде окремий код для Android і iOS;
– адже фрейм-ворк не будується спільно із iOS чи Android, він часом відстає від спеціалізованих платформ;
– фрейм-ворк React Native не має послідовності відносно випуску оновлень. React Native покращує швидкість розроблення, але так само збільшує тривалість процесу налагодження, особливо на Android.

Фрейм-ворк Ionic (рис.1.5) створений за популярно технологією для

розроблення мобільних додатків із відкритим кодом. Цей фрейм-ворк базується на HTML, CSS і JavaScript технологіях і дозволить розробникам утворювати мобільні додатки, котрі спроможні працювати на будь-якій платформі. Ionic має вбудовану бібліотеку, що вміщує компоненти, котрі допомагають здійснити єдність дизайну додатку і полегшують розробку. Ionic так само має можливості для розроблення гібридних мобільних додатків, що дозволить застосовувати веб-технології для розроблення мобільних додатків.

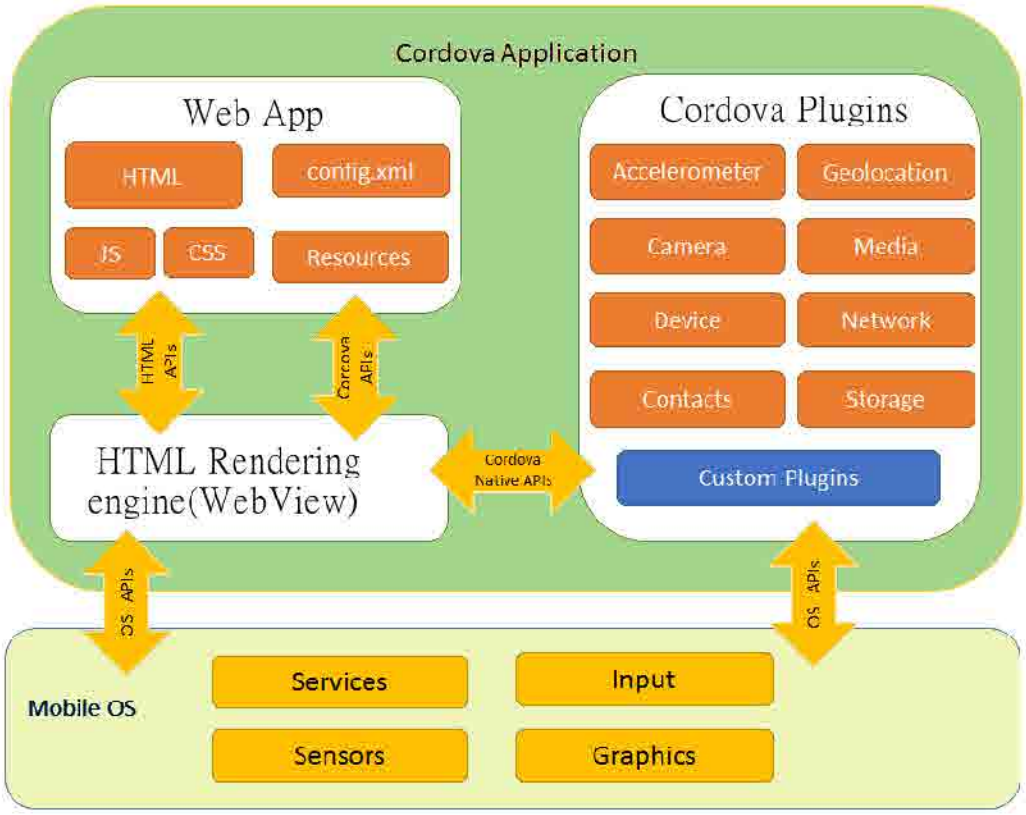


Рисунок 1.5. Спрощена архітектура фрейм-ворка Ionic

Фрейм-ворк Ionic має наступні переваги:

Швидка розроблення і мінімальний час виходу на ринок.

- можливо вести основну частину розроблення в браузері (крім нативної функціональності смартфона – тут знадобиться застосовувати смартфон для дебага);
- можливо розробляти програму для iOS і Android одночасно (із деякими обмеженнями – такими як особливості платформ, що стосуються стилів і плагінів);
- нема необхідності знати Java і Swift чи Objective-C;

– немало UI-компонентів доступні й прості в використанні – картки, кнопки, перемикачі, сегменти, Pop-Up, поля введення, списки, сітка із рядків й колонок й т.д.
– немало плагінів, що дозволяють застосовувати функції смартфонів, такі як: камера, сканер відбитків пальців, NFC, геолокація, відправка аналітики на Firebase, оповіщення і глибинні посилання.

Фрейм-ворк Ionic має наступні недоліки:

– нативні плагіни спроможні стати проблемою, коли якісь із плагінів, що використовуються, конфліктують чи коли у одному із них баг. Плагін FCM (Firebase Cloud Messaging) не працює із firebase-analytics, але є плагін під назвою Firebase, яким можливо його замінити. Відлагодження може бути досить складним: іноді важко зрозуміти, звідки приходить помилка, адже повідомлення про помилки спроможні бути неінформативними;
– збірка може порушуватися без причин, коли щось виявляється пошкодженим у оригінальній папці. Отже треба застосовувати гілки для кожної нової функції чи сторінки.

Фрейм-ворк Xamarin (рис.1.6) – це інша популярна технологія для розроблення мобільних додатків, розроблена компанією Microsoft. Цей фрейм-ворк дозволить розробникам утворювати мобільні додатки для платформ Android, iOS і Windows, використовуючи мову програмування C#. Фрейм-ворк Xamarin був випущений у 2011 році як незалежний фрейм-ворк для мобільних додатків, але у майбутньому був придбаний корпорацією Microsoft. Він має відкритий код й був запущений для вирішення проблем роз'єднаних стеків нативних технологій. Xamarin має вбудовані бібліотеки, що допомагають здійснити єдність дизайну додатку і полегшують розробку. Цей фрейм-ворк дозволить розробникам утворювати додатки, котрі мають високу продуктивність і швидкість.

Фрейм-ворк Xamarin має наступні переваги:

– при розробці додатків фрейм-ворк Xamarin застосовується мова програмування C#, відповідно він працює на багатьох платформ (включаючи

Android і iOS);

– фрейм-ворк Xamarin має добру підтримку із боку спільноти (понад 60000 учасників із понад 3700 компаній.

Фрейм-ворк Xamarin має наступні недоліки:

– фрейм-ворк Xamarin, хоч й є безкоштовно поширюваною платформою для приватних осіб і стартапів, але підприємства мають придбати ліцензію від Microsoft, вартість якої дуже висока;

– фрейм-ворк Xamarin не рекомендується для застосунків, котрі вимагають важкої графіки, адже ця платформа має інший метод візуального розміщення екранів. Застосунок, що активно використовує UX/UI, рекомендується реалізовувати власноруч;

– фрейм-ворк Xamarin має обмежений доступ до певних важливих бібліотек, необхідних розробникам програм для розроблення мобільних додатків. Крім того, адже ядро для створення інтерфейса не є мобільним, створення інтерфейса займає немало часу.

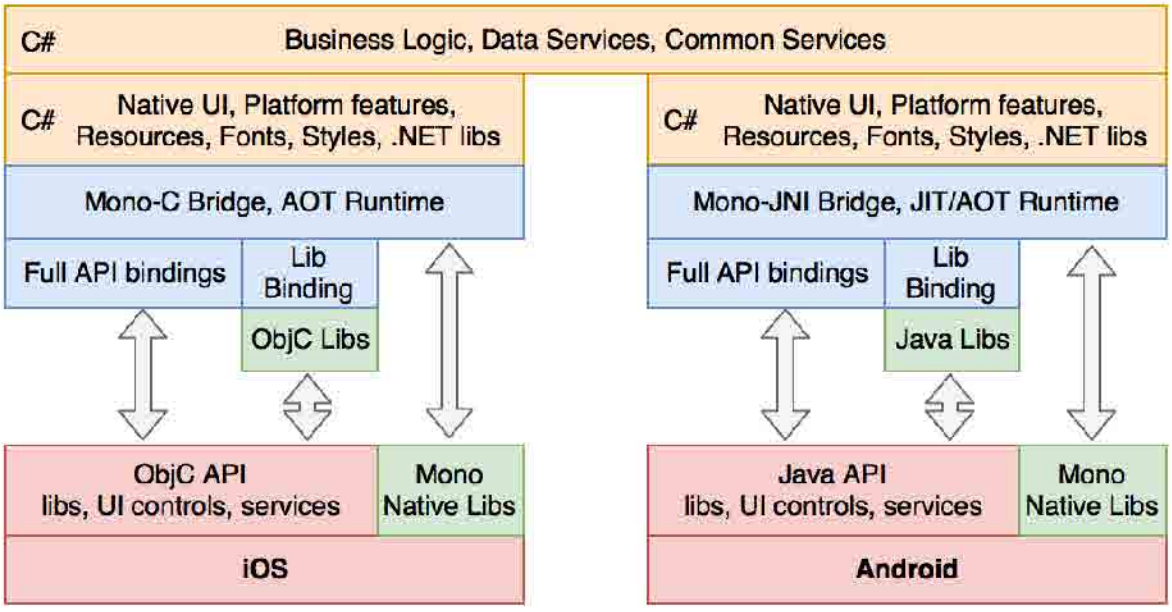


Рисунок 1.6. Спрощена архітектура фрейм-ворка Xamarin

Фрейм-ворк Phone-gap (рис.1.7) – це фрейм-ворк для розроблення мобільних додатків із відкритим кодом, розроблений компанією Adobe. Цей фрейм-ворк дозволить розробникам утворювати мобільні додатки для платформ Android, iOS і Windows із використанням веб-технологій, таких як HTML, CSS і JavaScript. Phone-

gap дає змогу розробникам утворювати гібридні мобільні додатки, що дозволить економити час і зусилля при розробці додатків для різноманітних платформ. Phone-gap так само має вбудовану бібліотеку, котра вміщує готові компоненти і стилі, що допомагає здійснити єдність дизайну додатку і полегшує розробку. Phone-gap є платформою із відкритим вихідним кодом від компанії Nitobi (нині в складі Adobe), котра дозволить розробляти програми для декількох мобільних платформ, використовуючи стандартні веб-технології. У даний час підтримуються такі операційні системи, як: Android, iOS, Symbian, Windows Phone, Bada, WebOS. Кожна із перерахованих ОС має повну чи обмежену функціональність. Написання програми ведеться на JavaScript із використанням HTML і CSS для розмітки. Мобільний додаток розробляється як звичайний сайт чи веб-сервіс. Платформа Phone-gap розширює API браузера й додає наступні можливості: доступ до акселерометра, доступ до камери, доступ до компаса, доступ до списку контактів, запис і прослуховування аудіо-файлів, надає доступ до файлової системи, дозволить працювати із різними HTML5 сховищами localStorage, Web SQL і але так само дозволить безболісно звертатися до будь-якої крос-доменної адреси.

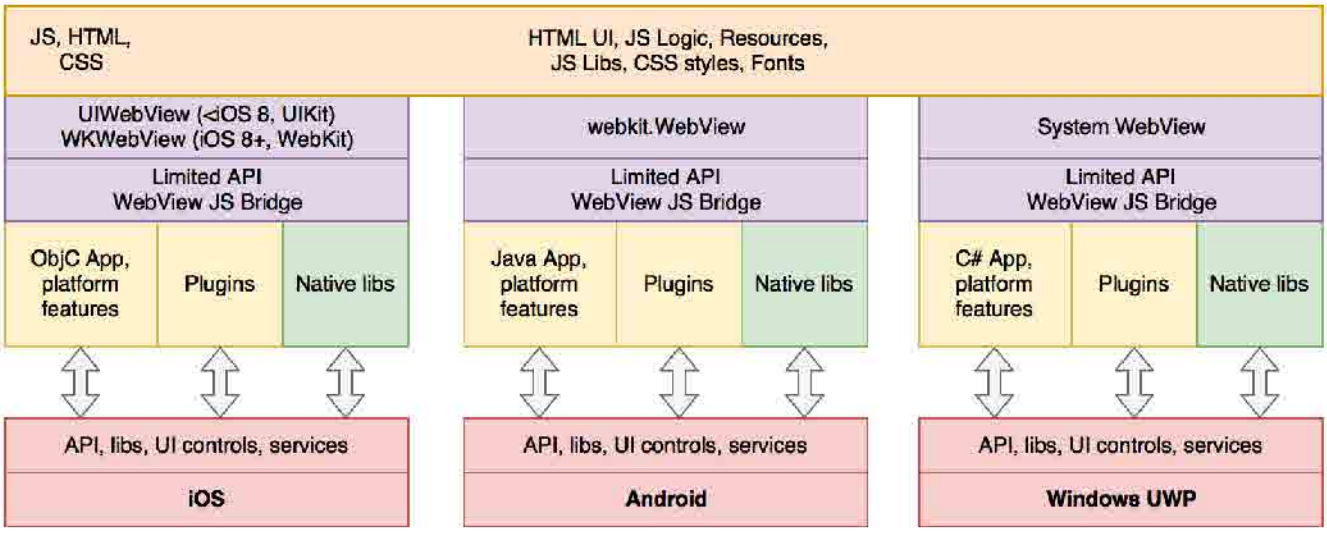


Рисунок 1.7. Спрощена архітектура фрейм-ворка Phone-gap

Фрейм-ворк Phone-gap має наступні переваги:

- дуже просто реалізувати спільну розробку програми;
- спроможність застосування сторонніх бібліотек;
- налагодження програм за поміччю браузера;

– спроможність конвертації програми під усі необхідні платформи за поміччю Phone-gap.

Фрейм-ворк Phone-gap має наступні недоліки:

– програми не підтримують багатопоточність;
– проблемна реалізація довгих списків (понад 1000);
– не усі програми спроможні бути оформлені як Web-програми;
– звернення до апаратних частин мобільного пристрою ведеться по-різному, що може спричинити несподіваний ефект;
– складність налаштування програмного середовища для написання програм під Phone-gap.

В результаті, коли додаток не є обчислювальним і ресурсномістким, то розроблення за поміччю платформи Phone-gap у сумі займе менше часу, порівняно із розробкою кросплатформного додатка під різні ОС. Коли ж програма складна, ресурсномістка, й її не можливо реалізувати як web-додаток, то краще розробити нативний додаток під кожну платформу.

Фрейм-ворк Kotlin Multi-platform Mobile (рис.1.8) – прагматичний, сучасний й має статичний тип. Kotlin є не тільки Java-подібною мовою JVM, адже JVM – тільки одна із доступних цілей. Kotlin може бути скомпільований / переведений у JavaScript (й працювати із React) і у код рідної платформи за поміччю Kotlin / Native. Kotlin / Native дозволить створити виконуваний файл, статичну чи динамічну бібліотеку із заголовками C чи фрейм-ворк Apple. Він так само може застосовувати існуючі бібліотеки C, Swift чи Objective-C. Типовий проект Kotlin Multi-platform Mobile складається із 3 модулів:

– Android: власна програма JVM для Android із власним інтерфейсом;
– iOS: власна програма, котра базується на LLVM для iOS із власним інтерфейсом;
– Common: тільки бізнес логіка Kotlin, що застосовується як програмами Android, так й iOS.

Kotlin Multi-platform Mobile є набором інструментів, що спрощують створення мультиплатформних програм для Android і iOS. Архітектура додатку

Kotlin Multi-platform Mobile може застосовувати патерн MVP (Model-View-Presenter). Тоді додаток буде поділятися на три головних шари коду: Model, відповідальний за сховище даних, такі як API чи бази даних; View, відповідальний за відображення даних від Presenter і передачу вводу до нього; Presenter, відповідальний за обробку даних для View.

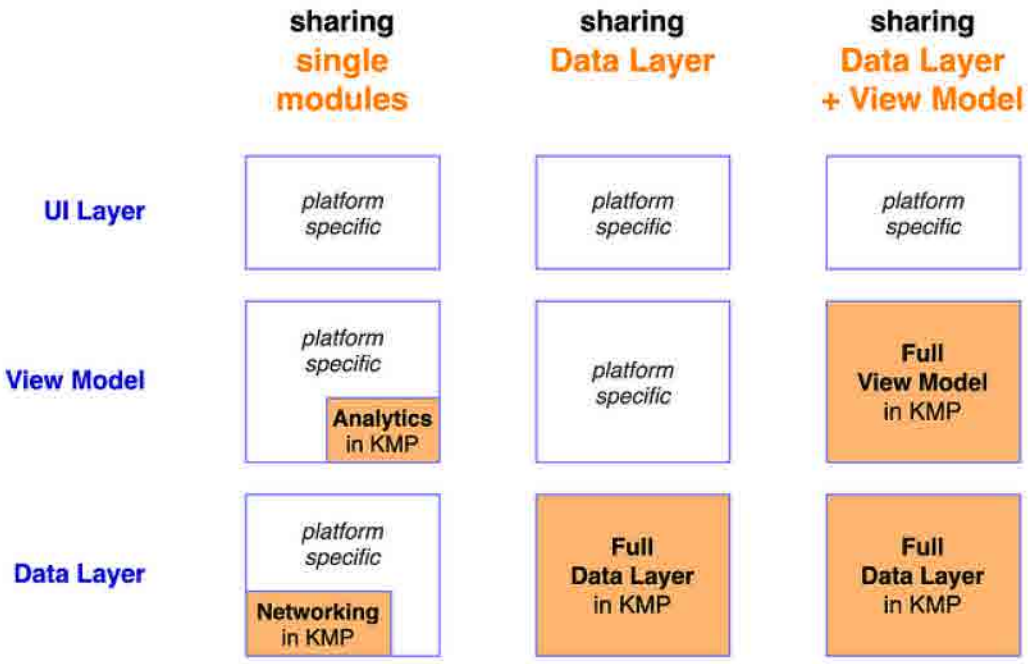


Рисунок 1.8. Спрощена архітектура фрейм-ворка Kotlin Multi-platform Mobile

Фрейм-ворк Kotlin Multi-platform Mobile має наступні переваги:

- власна продуктивність: без додаткового часу роботи у наборі додатків;
- нема застарілого коду: можливо почати інтегрувати Kotlin Multi-platform Mobile в будь-котрий час;
- нема спільного інтерфейса: ідеальні для пікселів власні види;
- просто додати iOS до існуючої програми для Android.

Фрейм-ворк Kotlin Multi-platform Mobile має наступні недоліки:

- в загальному спільному модулі не дуже немало доступних чистих бібліотек Kotlin;
- неможливо застосовувати ViewModel API у Android чи LiveData у загальному спільному модулі, адже він написаний на Java;
- неможливо застосовувати RxJava і модернізацію. Цей модуль повинен

містити чистий код Kotlin, що трохи ускладнює ситуацію. Але Kotlin Multi-platform має рішення для цього в вигляді двох нових ключових слів: очікуваного і фактичного, що легко можливо зрозуміти як декларацію і реалізацію.

Фрейм-ворк Flutter (рис.1.9) – це фрейм-ворк для розроблення мобільних додатків із відкритим кодом, розроблений компанією Google. Він дає змогу розробникам утворювати високоякісні мобільні додатки для Android і iOS за поміччю одного коду. Flutter має вбудовану бібліотеку, котра дозволить розробникам утворювати привабливі і ефективні інтерфейси для користувачів. Цей фрейм-ворк дозволить розробникам утворювати мобільні додатки, котрі працюють дуже швидко і плавно. Flutter так само дозволить розробникам застосовувати вбудовані готові компоненти і стилі, що дозволить економити час на розробці і забезпечує єдність стилю додатку. Flutter має відкритий код і безкоштовну платформу для створення власних інтерфейсів для Android, але так само iOS. В результаті опитування розробників Flutter входить до трійки найбільш улюблених фреймворків, й це додало ще однієї складності поточній популярності Reactive Native framework.

Фрейм-ворк Flutter має наступні переваги:

- функція "гарячого перезавантаження" дозволить розробникам бачити зміни, внесені у код, протягом декількох секунд. Це ідеальна основа для розвитку MVP. Замість того, аби витрачати зайві гроші і час на дві окремі програми, можливо швидко створити мобільний додаток Flutter, котрий буде нативним як на Android, так й на iOS;
- фрейм-ворк Flutter базується на Dart, об'єктно-орієнтованій мові програмування, для якої розробникам досить легко набути навичок. Фрейм-ворк Flutter має повний набір віджетів в матеріальному дизайні Google і у стилі Apple із пакетом Cupertino;
- немало готових рішень для власних додатків Android і iOS дозволяють працювати із платформами безперервної інтеграції, такими як Travis і Jenkins.

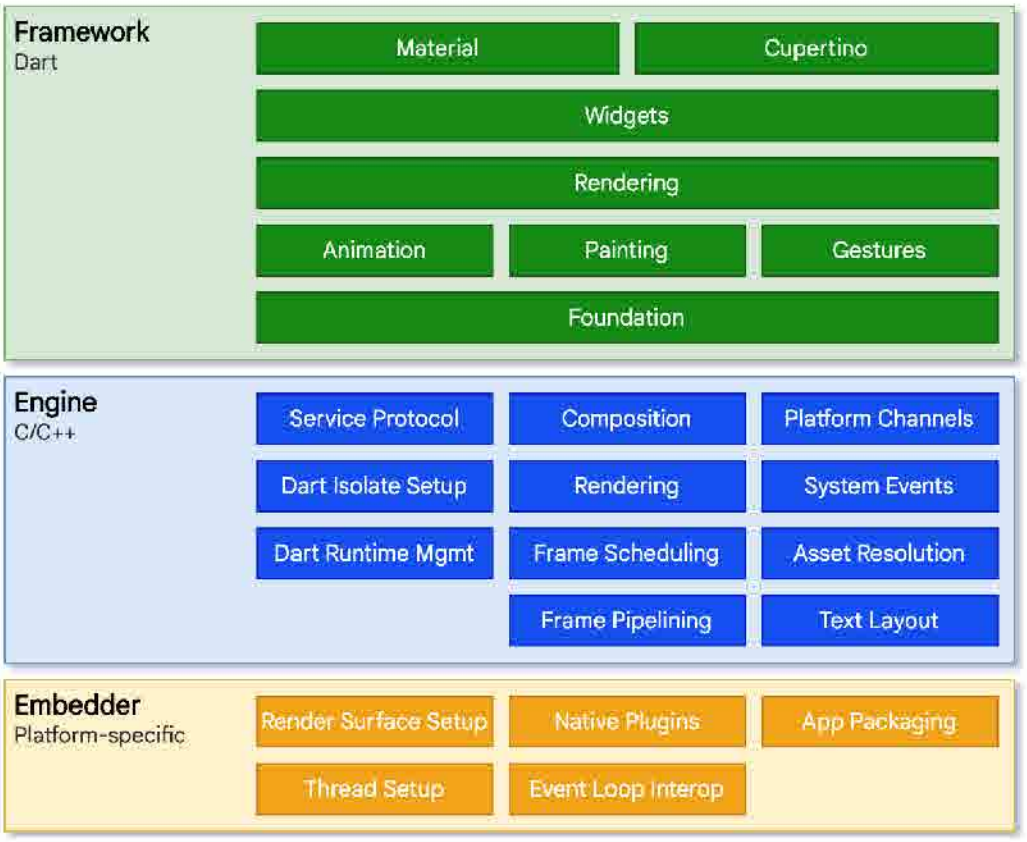


Рисунок 1.9. Компоненти фрейм-ворка Flutter

Фрейм-ворк Flutter має наступні недоліки:

- Flutter не пропонує ніякої підтримки для Android TV і Apple TV;
- адже додатки із підтримкою Flutter використовують вбудовані віджети, але не віджети платформи, розмір програми, як правило, більший.

1.2.2 Визначення фрейм-ворка для розроблення додатку

Підсумовуючи результати проведеного в п.1.2.1 огляду і із врахуванням особливостей кожного із наведених вище крос-платформних фреймворків виконано порівняння (табл.1.1).

ІЗ проведеного огляду й аналізу стає зрозумілим, що деякі із крос-платформних фреймворків, котрі надають змогу побудови інтерфейса користувача, мають недоліки із продуктивністю виконання додатка, але так само незручним використанням API платформи, коли треба працювати, зокрема, із геолокацією чи камерою на девайсі. Тому було обрано застосовувати фрейм-ворк Flutter, котрий підтримується корпорацією Google і має інтеграцію із технологією Google Firebase.

Таблиця 1.1. Порівняння крос-платформних фреймворків

	React Native	**Flutter**	**Xamarin**	**Kotlin Multi-platform Mobile**
Мова програмування	JavaScript і React	Dart	C# і .Net	Kotlin
Перший реліз	2015	2017	2011	2017
Розробник платформи	Facebook і спільнота	Google і спільнота	Microsoft	JetBrains
Спільнота	Невелика	Достатня спільнота, активний розвиток	Мала спільнота	Велика
Відкритий вихідний код	Так	Так	Так	Ні
Інструменти побудови UI	Компоненти Native і декларативний UI	Віджети і декларативний UI	Xamarin.Forms	Платформний UI
Продуктивність роботи	Висока	Висока	Середня крізь Xamarin.Forms	Висока

1.2.3 Аналіз архітектури і компонентів фрейм-ворка Flutter

В якості мови програмування фрейм-ворк Flutter використовує Dart. Основними перевагами Dart є:

- AOT-компіляція. У результаті утворюється ARM чи x86-код із високою швидкодією на пристрої користувача;
- JIT-компіляція. Дає спроможність запускати проект в віртуальній машині на емуляторі і реалізувати hot reload тривалістю менше однієї секунди із збереженням стану програми;
- Сувора система типів. Дозволить забути про вибір між TS і Flow;
- Tree shaking compiler. При складанні програми видаляє весь код, котрий не застосовується;
- Generational garbage collection. Швидке виділення і очищення пам'яті.

Фрейм-ворк Flutter складається із трьох шарів: Embeder, Engine і Framework (рис.1.9). Embeder передбачає платформозалежний код, що відповідає за взаємодію

і запуск на кінцевій платформі. Engine написаний на C/C++ й включає Dart як рівень абстракції для роботи із CPU, Skia – 2D-бібліотеку для малювання й підсистему для рендерингу тексту, позичену в Android. Найбільша частина Framework повністю написана на Dart. Поділ на шари дозволить легко побудувати програми будь-якої складності. Можливо застосовувати готові віджети Material-UI чи реалізувати специфічні компоненти на основі базових віджетів. Є спроможність гнучко працювати із анімацією і обробкою жестів. Таким чином, можливо комбінувати швидку роботу із високорівневими віджетами із можливістю внесення/перевизначення глибших шарів.

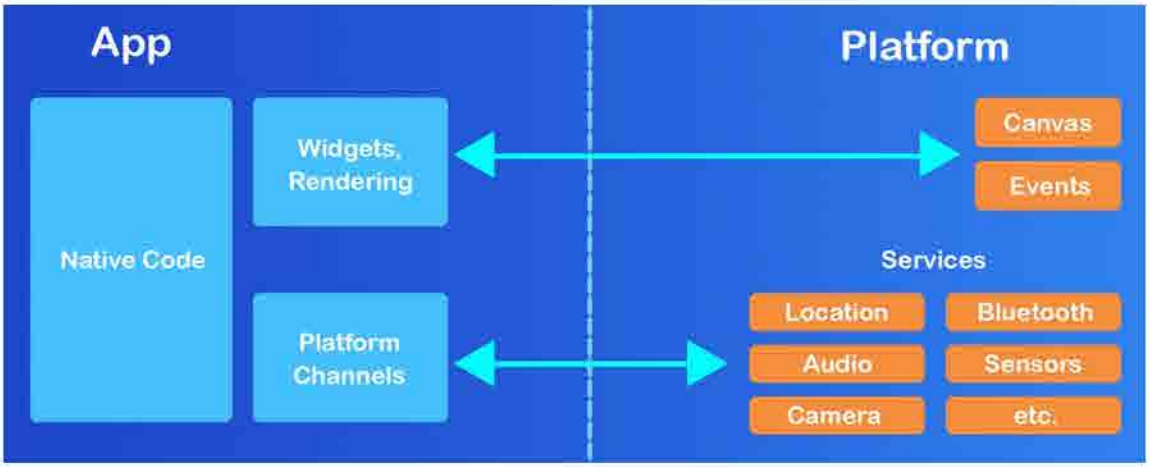


Рисунок 1.10. Взаємодія додатку користувача із платформою Flutter

Особливістю архітектури Flutter є те, що він сам малює кожний піксель, контролює жести і анімацію. Він не використовує OEM-віджети, як це робить React Native. Натомість створені два набори віджетів для основних мобільних платформ: Material для Android і Cupertino для iOS. Таким чином відмальовані усі UI-компоненти із обох мобільних платформ. Безпосередньо із мобільною платформою (геолокація, звук, bluetooth) взаємодія відбувається крізь Platform Channels.

Крос-платформність в Flutter реалізована за поміччю емуляції нативних елементів на канвасі – подібний підхід застосовують ігрові двигуни. Вся UI-малювання виконується бібліотекою Skia.

Dart є строго типізованою об'єктно-орієнтованою мовою програмування. Й вона має особливості дострокової компіляції. Dart ближче до Java із підходом OOP, ніж до JS із функціональним програмуванням. Основними перевагами мови Dart є:

1. AOT-компіляція. У результаті утворюється ARM- чи x86-код із високою швидкодією на пристрої користувача;

2. JIT-компіляція. Дає спроможність запускати проект в віртуальній машині на емуляторі і реалізувати hot reload тривалістю менше однієї секунди із збереженням стану програми;

3. Сувора система типів. Дозволить забути про вибір між TS і Flow;

4. Tree shaking compiler. При складанні програми видаляє весь код, котрий не застосовується;

5. Generational garbage collection. Швидке виділення і очищення пам'яті.

Flutter підтримує JIT і AOT компіляцію. JIT – тільки для функціонала Hot Reload. Мова Dart є однопотоковою, як й node.js. Для реалізації асинхронності використовують механізм Event Loop. Дострокова компіляція робить Flutter SDK й Dart придатними для створення власного коду ARM, котрий може бути скомпільований на Android й iOS. Для встановлення бібліотек застосовується Pub менеджер пакетів. В Flutter компоненти називають віджетами (рис.1.11). Так само Flutter дає дві візуальні бібліотеки: Material і Cupertino. Flutter постачає бібліотеки для роботи із графікою, більшість із котрих це й є віджети: Animation, Cupertino, Foundation, Gestures, Material, Painting, Physics, Rendering, Scheduler, Semantics, Services, Widgets, Dart:io.

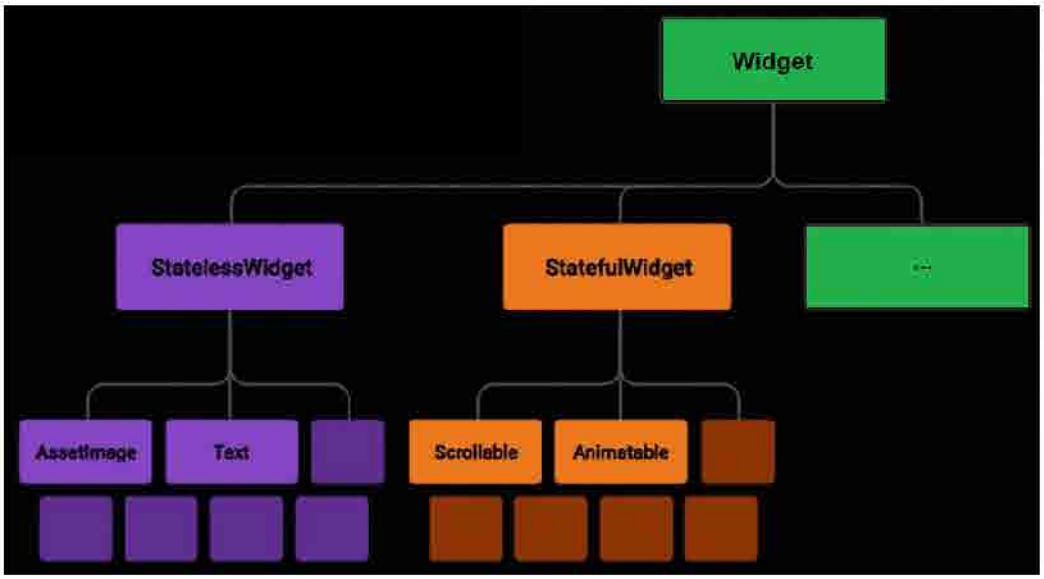


Рисунок 1.11. Ієрархія класів Flutter

Віджети в Flutter спроможні бути видимими, невидимими, містити дочірні віджети і взаємодіяти між собою. Кожний із них може бути як віджетом без стану (Stateless Widget), так й віджетом, котрий має стан (Stateful Widget). Основна відмінність – спроможність повторно малювати віджети під час виконання програми. Stateless Widget (рис.1.12) буде малюватись тільки один раз й є незмінним. Stateful Widget може малюватись немало разів залежно від зміни внутрішнього стану віджету. Для створення Stateful Widget треба створити 2 класи. Перший клас має успадковуватися від Stateful Widget, котрий в свою чергу успадковується від Widget й є незмінним. Примірник цього класу не перетворюється при кожному малюванні і застосовується для зберігання переданих параметрів і ініціалізації стану. Другий – клас стану, котрий має доступ до Stateful Widget крізь внутрішню властивість й займається безпосередньо відмальовуванням стану, реагуючи на його зміну.

Рисунок 1.12. Stateful Widget в Flutter

Глобальний стан програми, так само як й локальний, може бути реалізований за поміччю Stateful Widget. Цей віджет створюється в верхньому вузлі програми й передається вниз по віджетам за поміччю InheritedWidget. Кожний дочірній віджет програми може одержати доступ до віджету глобального стану для зміни і застосування його полів. Для управління станами із сайд-ефектами треба керуватися тими самими принципами. Треба викликати асинхронну функцію, котра послідовно встановлює прапор для індикації про завантаження із сервера, виконати потрібний нам запит на сервер й змінити стан залежно від даних. Перевагами такого підходу є простота і швидкість впровадження в додаток із

невеликою кількістю екранів, що виражається в відсутності будь-котрих додаткових бібліотек. ІЗ недоліків можливо виділити такі:

1. Подання і бізнес-логіка не розділені;
2. Складність в модульному тестуванні;
3. Складність в масштабуванні і підтримці;
4. Велика кількість коду, котрий не може бути перевикористаний.

Цей підхід бажано застосовувати тільки до дуже маленьких програм із фіксованою кількістю екранів.

Тип архітектури Provider (Scoped Model) (рис.1.13) дозволить винести бізнес-логіку із вистави і дає спроможність перевикористовувати цю логіку у різноманітних модулях системи. Результат досягається за поміччю створення моделі і реагування підписаних віджетів на її зміну. Для локального стану віджету необхідно створити модель із усіма полями, котрі будуть використовуватися в віджеті. Після зміни кожного поля (чи полів) треба повідомити що модель змінилася й виконати малювання підписаних віджетів. Аби підписати віджет на модель, застосовується клас ChangeNotifierProvider, котрий є частиною бібліотеки provider. Підписка відбувається безпосередньо до того віджету, котрий залежатиме від даних із створеної моделі.

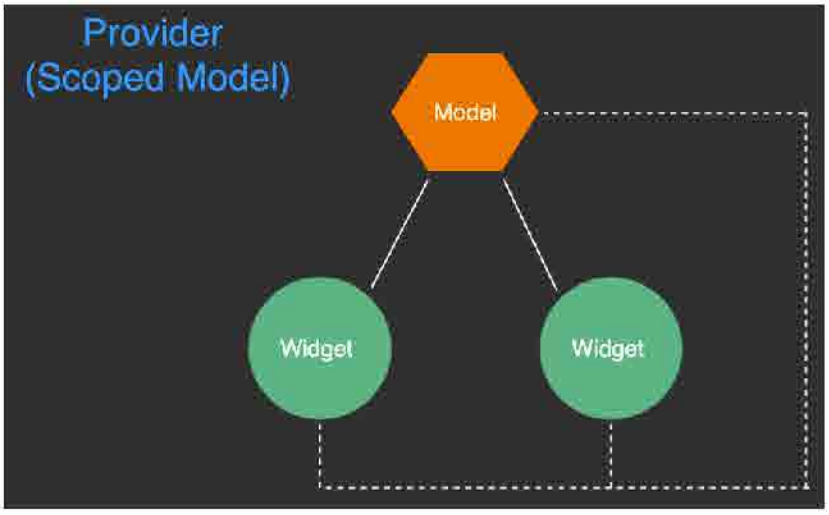


Рисунок 1.13. Тип архітектури Provider (Scoped Model)

Глобальний стан нічим не відрізняється від локального, крім того, що модель підписується до верхнього віджету програми. Сама модель повинна мати унікальний клас, аби не було конфліктів під час її пошуку в дочірніх віджетах.

В моделі для зберігання стану із сайд-ефектами необхідно передбачити спеціальний метод, котрий встановить індикатор початку асинхронного завантаження даних із сервера і встановить дані, що прийшли із сервера, скинувши при цьому індикатор завантаження. Однак тут криється проблема подвійного повідомлення про зміну моделі. Перше повідомлення відбувається під час зміни установки даних із сервера, друге після зміни індикатора завантаження. ІЗ переваг можливо виділити поділ бізнес-логіки і уявлення за поміччю створення моделей і event-based-архітектури. Тестувати такі моделі легше, ніж в Native state за рахунок відсутності додаткових зусиль створення віджетів, в котрих цей стан застосовується. Важливо, що цей тип архітектури підтримується Google, тому можливо не турбуватися про популярність й підтримку цього рішення. Однією із основних проблем такої архітектури є складність в розумінні того, котра властивість була змінена й із якої моделі відбулося повідомлення віджетів про зміну. Варіантом вирішення цієї проблеми є угода на рівні команди відносно запровадження так званих екшенів. Це єдине місце, де модель може викликати спосіб сповіщення чи інші екшени. Так само вам треба бути готовим, що немало речей на кшталт persist-сховища моделей недоступні із коробки й доведеться написати велику кількість супроводжуючого коду для їх впровадження.

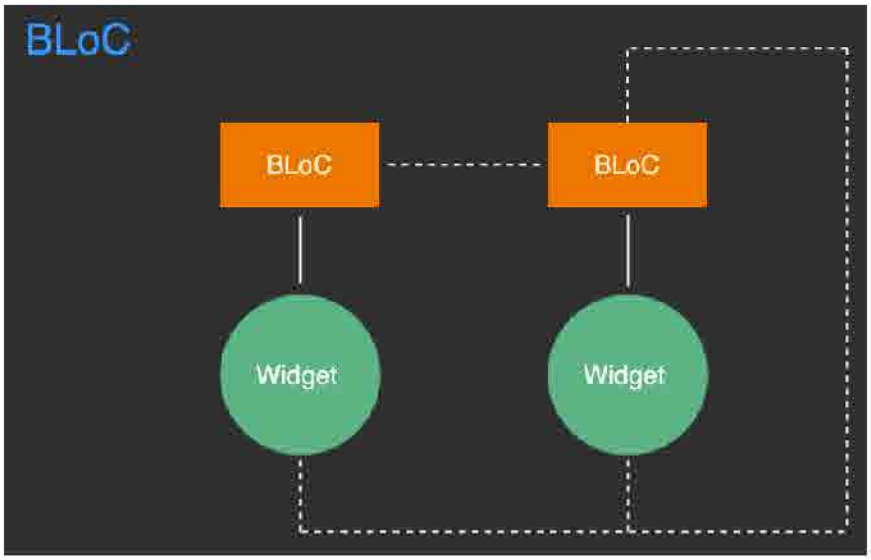


Рисунок 1.14. Тип архітектури BLoC

Provider чудово підходить для середніх проектів, в котрих нема великого зачеплення між модулями. BLoC (Business Logic Component) (рис.1.14) – шаблон,

створений Google для керування складним станом програми, ґрунтуючись на реактивній парадигмі. Основна ідея полягає у тому, що наша програма розбита на модулі, що реалізують бізнес-логіку. Кожний модуль має одну чи кілька Sink (труб), котрі є деякими вхідними потоками для агрегування подій ззовні. В якості вихідних даних виступає Stream (потік), котрий визначає асинхронний формат даних для наших віджетів. Аби скористатися модулем на рівні віджету, застосовують StreamBuilder, котрий керує потоком даних і автоматично вирішує проблеми підписки і перемальовування дочірнього дерева віджетів. Незважаючи на це, застосовувати BLoC в чистому вигляді – досить складна робота, адже треба застосовувати бібліотеку RxDart для маніпуляції із потоками, вручну відписуватись від потоків, інакше можливо одержати серйозний витік пам'яті на великих додатках. Важливою перевагою цього пакету є спроможність авто-генерування коду за поміччю плагінів для найпопулярніших IDE (IntelliJ, VS Code). Сам підхід цікавий й має велику кількість переваг:

– багате API під час роботи із потоками, що дозволить їх легко групувати, поєднувати і трансформувати;

– угруповання логіки у одному місці;

– легкість в тестуванні стану із сайд-ефектами за рахунок вбудованого у Dart API тестування потоків;

– мінімальна кількість малювань завдяки використанню StreamBuilder.

ІЗ недоліків треба виділити тільки завищену складність для початківців, адже не усі розробники спроможні швидко вникнути у суть роботи потоків, й відсутність інструменту для налагодження кожного із модулів.

Найпопулярніша реалізація Flux-архітектури – Redux. Причини прості:

– централізованість – стан усієї програми знаходиться у одному місці, що дозволить зберігати його у будь-якому зручному для вас сховищі;

– передбачуваність – не треба імперативно змінювати моделі, що залежать одна від одної, треба просто реагувати на дії, котрі надсилає система;

– простота налагодження – завжди можливо подивитися повне дерево стану, але так само спроможність time-travel-налагодження, коли можна послідовно

пройтися по всіх змінах в стейті й своєчасно віднайти та виправити помилки;

– гнучкість – існує велика кількість middleware-розширень на усі потреби програміста у управлінні станом.

В dartpub є пакет redux (рис.1.15), котрий може бути використаний як в Інтернеті, так й на мобільних платформах із впровадженням додаткового пакету flutter_redux. В Redux нема поділу на локальне і глобальне. Стан завжди глобальний – доступний будь-якому віджету й доступний до зміни крізь екшени у будь-якому місці системи. Формально при цьому існують такі поняття:

– State – модель стану, котра може бути як скалярним, так й будь-яким іншим складовим типом;
– Action – клас-ідентифікатор події, котрий зберігає у собі payload для передачі потрібних параметрів;
– Reducer – обробник екшенів, має доступ до поточного стану і екшену, котрий чекає на обробку;
– Dispatch – метод виклику екшену, котрий обробляється одним із редьюсерів;
– Store – дерево стану програми, що комбінує в собі усі редьюсери, котрі визначаються в додатку;
– StoreConnector – дозволить дочірньому віджету одержати доступ до store.

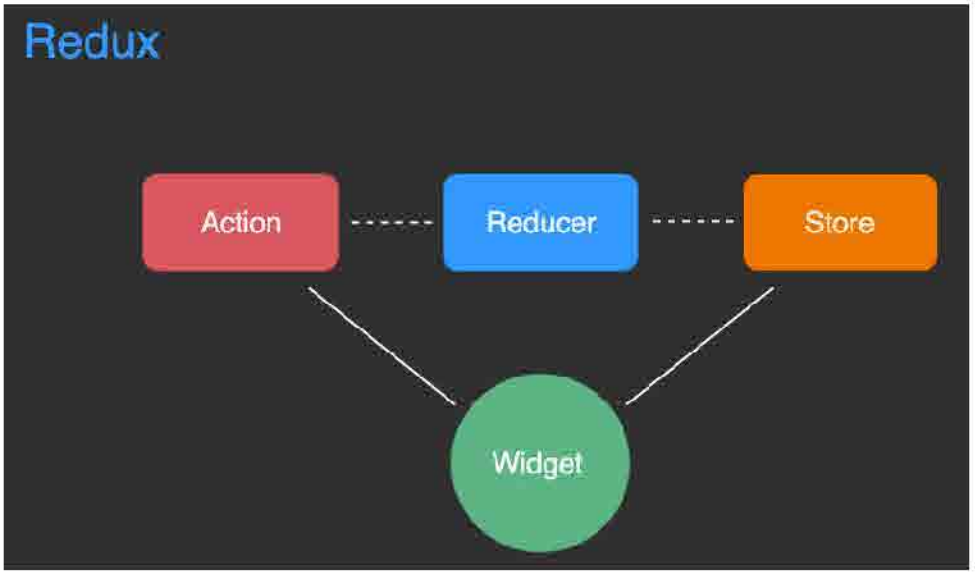


Рисунок 1.15. Тип архітектури Redux

В чистій реалізації Redux нема можливості працювати із побічними ефектами. Зазвичай при цьому використовуються додаткові бібліотеки redux-thunk, redux-saga тощо. В dartpub є такі пакети із цією метою: redux_thunk й redux_epics.

Працювати із Redux досить зручно завдяки бібліотекам, що розвиваються:

- flutter_redux_dev_tools – налагодження і time-travel debug;
- redux_thunk – робота із сайд-ефектами за поміччю thunk;
- redux_epics – робота із сайд-ефектами за поміччю епіків, що базуються на потоках;
- redux_logging – логування стейту чи екшенів;
- redux_persist_flutter – збереження стану в постійному сховищі.

Однак недоліки успадковуються від старшого батька:

- кожний віджет може одержати доступ до всього стану програми, що може легко порушити принцип єдиної відповідальності;
- локальний стан віджетів зберігається в глобальному дереві стану, що суттєво збільшує його розміри;
- проблема сайд-ефектів вирішується тільки крізь додаткові middleware й може змінюватися від проекту до проекту.

Усі розглянуті підходи спроможні використовуватися у фазі продакшен й спроможні надовго бути у екосистемі Flutter. Вибір архітектури для проекту залежатиме від багатьох факторів: розміру і типу програми, рівня володіння технологією в команди, минулого досвіду роботи зі схожими технологіями і бібліотеками. Для невеликих проектів чи MVP найкращим рішенням є архітектура Provider, яку й обрано для реалізації даного проекту гео-інформаційного мобільного додатку для ідентифікації пунктів незламності.

1.2.4 Аналіз технологій для створення геоінформаційних систем

При створенні геоінформаційних систем функціональність картографування є невід'ємною частиною майже кожного типу логістичного програмного забезпечення. Зокрема, планування маршруту і відстеження у реальному часі на порталі клієнтів значною мірою покладаються на карти. Правильний вибір засобу

картографування може допомогти побудувати прозорі відносини зі своїми клієнтами і зменшити експлуатаційні витрати.

Фрейм-ворк MapKit (рис.1.16) допомагає відображати карти чи супутникові зображення в додатку, так само є спроможність визначати інформацію про позначки місця для координат на карті. MapKit може знадобитися, аби дати додатку відчуття місця із картами і інформацією про місцезнаходження. Фрейм-ворк MapKit можливо застосовувати для того, аби:

– вставляти карти безпосередньо відображення програми;
– додавати анотації і накладання на карту для виклику визначних місць;
– забезпечте доповнення тексту, аби полегшити користувачам пошук місця призначення чи визначної точки.

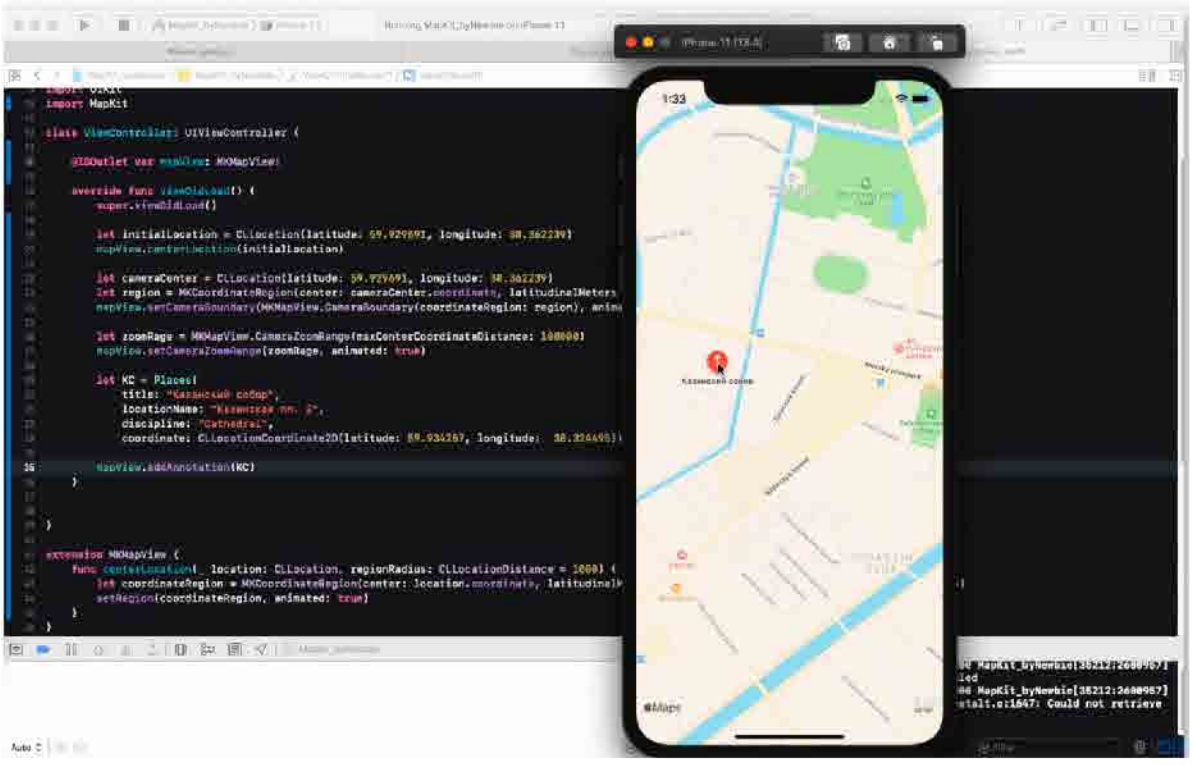


Рисунок 1.16. Застосування фрейм-ворка MapKit

Коли додаток пропонує маршрути, можливо зробити їх доступними за поміччю MapKit. Так само можливо застосовувати цей фрейм-ворк, аби доповнити вказівки, котрі спроможні надатися додатку.

Фрейм-ворк Mapbox (рис.1.17) є одним із найбільших постачальників спеціально розроблених карт для веб-сайтів і мобільних додатків. Цією послугою користуються відомі компанії із доставки і транспортування, такі як DHL,

DPDgroup, Grubhub, Instacart. Більшість даних, котрі використовує Mapbox, є відкритими, й Mapbox підтримує спільноту добровольчих картографів. Вони часто надають найсвіжіші оновлення, включаючи дані про місцезнаходження, що швидко змінюються. Джерела даних Mapbox включають OpenStreetMap (OSM), USGS, Landsat, Natural Earth і OpenAddresses. Платформа використовує OpenStreetMap як базову карту й дозволить розробникам додавати різні маркери, лінії, полілінії і багатокутники, але так само шари із зовнішніх джерел (в GeoJSON, GPX і інших форматах). Mapbox пропонує безліч інструментів, котрі допоможуть інтегрувати карти і інші веб-служби веб-картографування Mapbox такі як напрямки, геокодування і статичні зображення в мобільний додаток. Для того, аби SDK Maps для iOS і Android був невеликим, Mapbox пропонує різні API відображення для взаємодії із веб-службами Mapbox: API Mapbox Directions API, API статичних зображень і API геокодування.

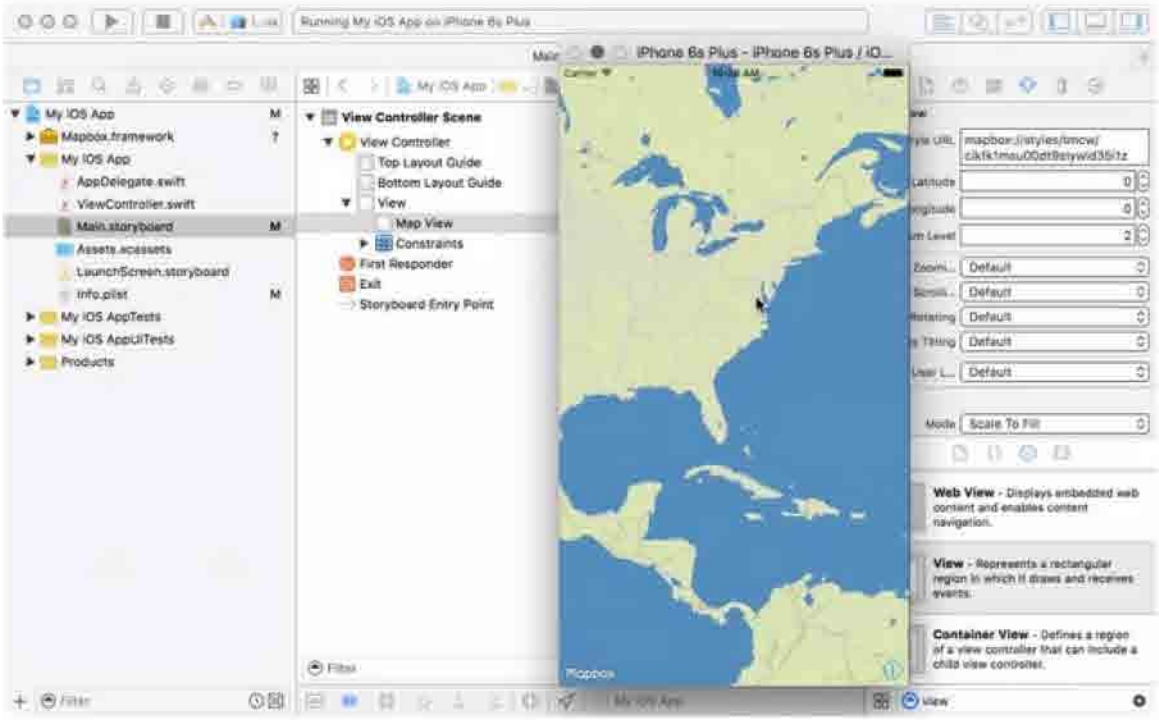


Рисунок 1.17. Застосування фрейм-ворка Mapbox

Сервіс MapBox має наступні переваги:

– Унікальні варіанти налаштування. Огляди Mapbox показують, що Mapbox є більш настроюваним, ніж Google Maps. Розробники спроможні встановити шрифти і колірну схему і додати такі функції,

як покрокові вказівки і інформація про місцевість;

- SDK із відкритим кодом. SDK для карт Mapbox є відкритим кодом. Mapbox ділиться своїм кодом на GitHub, аби його завжди можливо було побачити, проаналізувати і вдосконалити. Пакети SDK Maps засновані на Mapbox GL Native. Ця бібліотека дозволить вбудовувати інтерактивні, настроювані векторні карти в власні програми на декількох платформах;
- Інтеграція із PubNub. Mapbox співпрацює із PubNub, котрий пропонує інфраструктуру як послугу для потокового передавання даних у реальному часі, створює динамічні візуалізації карт із даних в реальному часі і включає такі функції, як відстеження об'єктів, геокодування і теплові карти;
- Mapbox AR. SDK Mapbox Maps для Unity дозволить утворювати послуги на основі розміщення із використанням визначних місць (POI) в всьому світі. Можливо додавати місцеположення за поміччю перетягування карт й POI, 3D будівель і місцевості, AR на основі місць;
- Офлайн-карти. Mapbox забезпечує більшу гнучкість відносно режиму роботи у автономному режимі. Завдяки використанню векторних карт, Mapbox підтримує функціонування у автономному режимі. Додатки, створені за поміччю мобільних SDK Mapbox, спроможні завантажувати карти для вибраних географічних районів для застосування, коли пристрій не має мережевих зв'язків. Крім того, пакети SDK для мобільних пристроїв Mapbox автоматично кешують плитки і інші ресурси, що вимагаються під час звичайного застосування;

В сервісі MapBox є так само недолік, пов'язаний із порівняно слабким охопленням. Є немало місць, де Google Maps чи MapKit мають вищий рівень покриття, ніж послуги на основі OSM. Причиною цього є головним чином той факт, що карти Google отримують постійні оновлення.

Фрейм-ворк Google Maps є продуктом Google, пов'язаним із картою, котрий

базується на надійних навігаційних даних і має понад мільярд активних користувачів щомісяця, що робить цю послугу найбільш популярною.

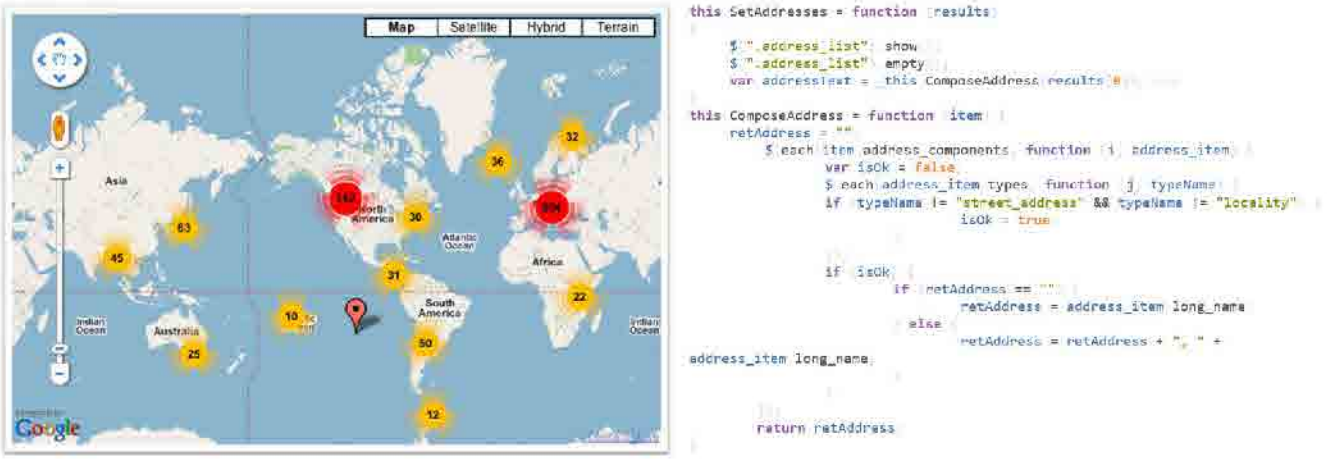


Рисунок 1.18. Застосування фрейм-ворка Google Maps

Сервіс Google Maps має наступні переваги переваги:

- Завдяки супутникам Google, транспортним засобам перегляду вулиць і створеним користувачами виправленням, географічне покриття Google вважається найкращим;
- Синтаксис, подібний до JSON, що застосовується Google Maps, негайно завантажується разом із картою. Є спроможність керувати видимістю, кольором і непрозорістю всіх елементів карти;
- Перегляд вулиць є функцією, котра забезпечує інтерактивні панорами із різноманітних позицій уздовж багатьох вулиць по всьому світу. Ця функція може візуалізувати на карті мову розмітки замкової щілини (KML) і дані GeoRSS. Зокрема, синтаксис KML повідомляє Google Maps про те, як відображати географічні об'єкти, такі як визначні місця, зображення, багатокутники тощо, й може надати все: від статистики злочинів до інформації про недавні землетруси;

Сервіс Google Maps має наступні переваги недоліки:

- Карти Google обмежують кількість запитів, котрі можливо надсилати до продуктів Google Maps Platform за секунду. Зокрема, можливо надіслати 50 запитів до API напрямків. Треба постійно контролювати застосування API, аби перевіряти витрати, й коли перевищити ліміти,

рахунок буде заблокований;

– Складна й не дуже прийнятна модель ціноутворення.

Проект OpenStreetMap (OSM) (рис.1.19) відкритих геоінформаційних даних під вільною ліцензією Open Database Licence відомий досить широко. Головною особливістю проекту і його основною перевагою порівняно із будь-якими іншими аналогами був принцип повністю відкритих географічних даних, котрі спроможні вільно доповнюватись і уточнюватись будь-яким учасником проекту. OpenStreetMap використовує відкриті джерела даних, такі як супутникові знімки, надані постачальниками для цілей проекту, відкриті геопросторові дані, що поширюються урядами країн на вільних засадах, дані із інших вільних відкритих джерел, але так само власні внески учасників проекту.

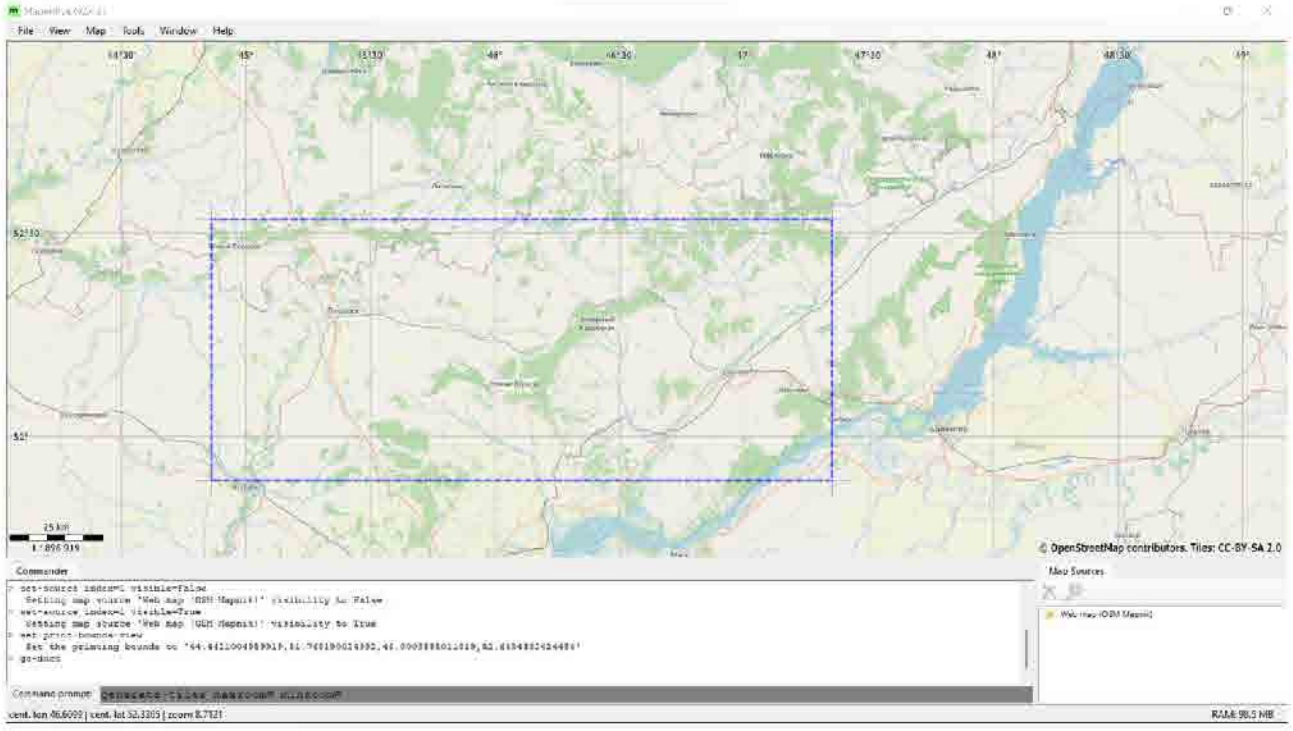


Рисунок 1.18. Застосування сервісу OpenStreetMap

Це дозволить утворювати більш прозорі і перевірені дані, котрі спроможні бути корисними для громадськості, дослідників і навіть гуманітарних організацій. OpenStreetMap, залежно від регіону, може мати різний рівень актуальності даних. Тут все залежить від кількості і активності волонтерів, котрі вносять зміни до бази даних. Однак там, де є активна спільнота учасників проекту, дані спроможні бути досить актуальними і детальними. На головній сторінці OpenStreetMap знаходяться

елементи: мапа; поле для пошуку; спроможність побудувати маршрут; перегляд відомостей про об'єкт на мапі. При цьому використовуються різні API, зокрема API для показу мап; API для пошуку (геокодування); API для прокладання маршрутів; API для ведення по маршруту і надавання покрокових інструкцій і ін. API OpenStreetMap призначений для отримання, редагування і збереження даних у спільній базі геопросторових даних учасниками проекту. Тобто єдине, що гарантує OpenStreetMap, це те, що можливо буде скористатись даними, й вже використовуючи ці дані можливо створити власну мапу, геокодер, пропонувати своїм клієнтам послуги навігації чи ще щось. Мапа, пошук, прокладання маршрутів на головній OpenStreetMap – це демонстрація того, що можливо зробити із даними проекту, усі ці елементи самі собою є окремими незалежними проектами й єдине, що їх пов'язує із OpenStreetMap – дані, котрі OpenStreetMap надає. Мапа – тайловий сервер (Mapnik) і стиль подання даних (OSM-Carto) – 2 окремих проктит, котрі жодним чином не залежать від OSMF (OpenStreetMap Foundation); бібліотека показу тайлів – leaflet.js, бібліотека JavaScript із відкритими сирцями для показу інтерактивних мап; пошук – геокодер Nominatim, теж самостійний проект; прокладання маршрутів – OSRM, Valhalla, GraphHopper, теж сторонні проекти; експорт/видобування даних – Overpass API (Overpass-Turbo); iD – редактор даних OSM, зараз під управлінням OSMF. Усі ці проекти є вільними проектами із відкритими вихідними кодами й, за потреби і наявності відповідної кваліфікації можливо зібрати із цього конструктора потрібне рішення. Тим паче, що є вибір серед різноманітних проектів візуалізації даних, створення мапи; різноманітних рушіїв геокодування, пошуку; рушіїв прокладання маршрутів і навігації. Крім того, можливо застосовувати дані OpenStreetMap для проведення аналізу в власних проектах, поєднуючи їх із іншими даними, – це те, що неможна робити із Google Maps. Як й будь-котрі інші дані, географічні дані так само підлягають структуруванню при зберіганні і обробці. У цілому всю структуру даних можливо представити схематично (рис.1.20).

Усі дані можливо умовно розбити на три основні групи: типи даних, що описують в вигляді ієрархічного зв'язку сам об'єкт, як таку собі просторову

сутність, що має свій кінцевий результат – відомі координати всіх частин об'єкта; інформаційна частина – це описова характеристика об'єкта, що не має до просторової географічної структури об'єкта прямого відношення (його назва, фізичні, логічні і інші властивості); службові атрибути об'єкта, необхідні для організації процесу зберігання і обробки інформації в вигляді набору даних, такі як унікальний ідентифікатор, стан об'єкта у базі, час останньої правки об'єкта у базі й т.д.

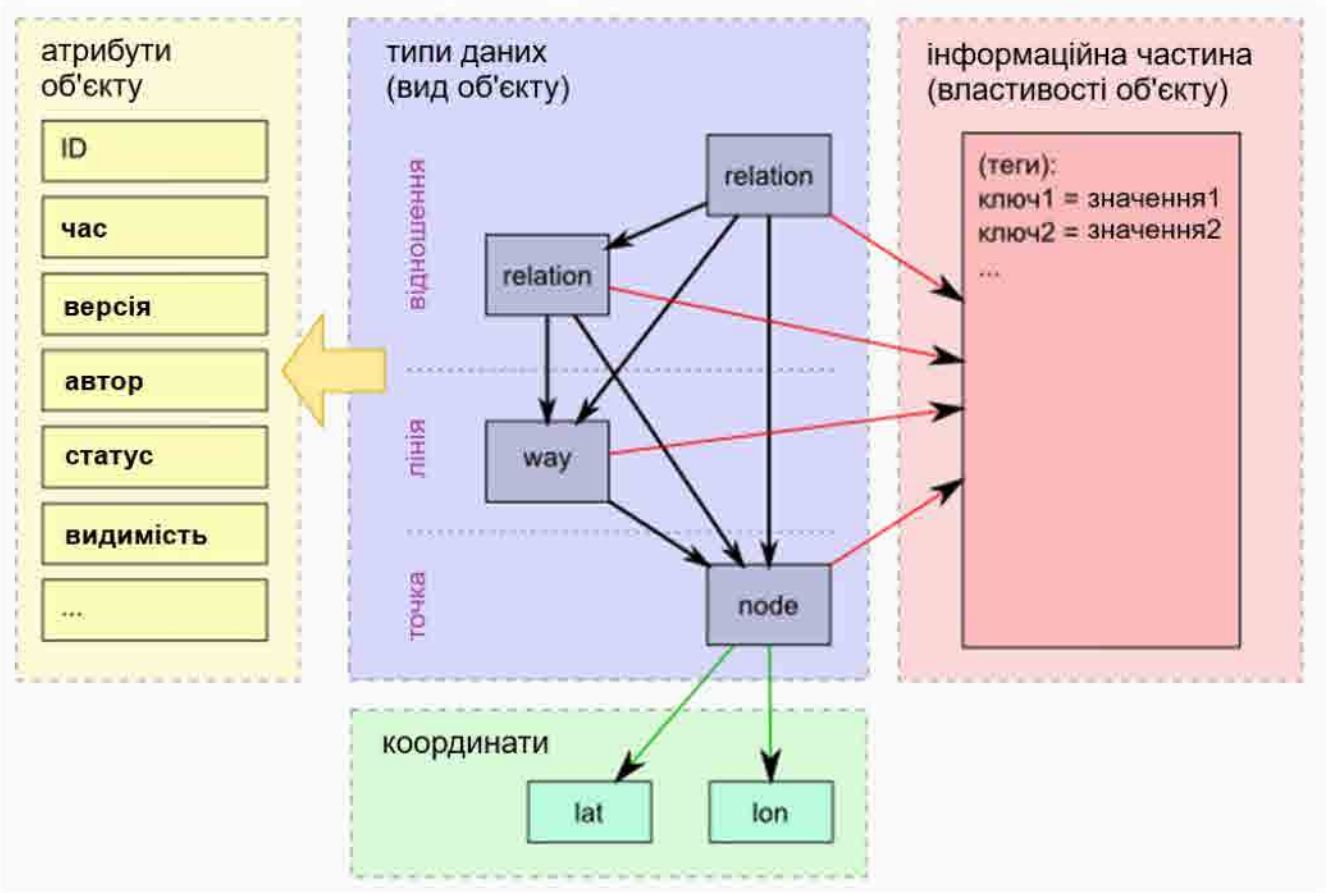


Рисунок 1.20. Структура даних OpenStreetMap

1.2.5 Визначення сервісу для інтеграції графічної карти

Після виконаного в п.1.2.4 аналізу кожного із актуальних геоінформаційних сервісів для інтеграції карт й визначення переваг й недоліків кожного із них обрано сервіс OpenStreetMap (табл.1.2).

При аналізі зазначених геоінформаційних сервісів було зроблено висновок, що саме OpenStreetMap є оптимальним вибором для вирішуваної задачі. Сервіс MapKit, хоч й є нативним рішенням від компанії Apple, але не має немало можливостей для вирішуваної задачі: нема кешування карти, що треба

реалізовувати самостійно, нема API для навігації по побудованому маршруту користувача. Коли аналізувати GoogleMaps, то таке рішення так само не має кешування карт, але так само Google забороняє утворювати кешування карти вже самим розробникам згідно с «Умовами застосування» їх SDK. Це зрозуміло, адже офлайн-режим є перевагою їх власного додатка Google Maps, але іншою проблемою є вартість застосування їх карт, що дуже важливо при виборі вендору карт у умові стартапу.

Таблиця 1.2. Порівняння геоінформаційних сервісів

	MapKit	GoogleMaps	MapBox	OpenStreetMap
Нативний фреймворк	Так	Ні	Ні	Ні
Відкритий вихідний код	Ні	Ні	Так, частинами	Так
Розробник платформи	Apple	Google	MapBox	Стив Кост і вільні розробники
Стилізація карти	Ні	Так	Так	Так
Кешування карти	Нема, треба розробляти самому	Нема, не можливо розробити крізь умови застосування	Існує	Існує
Побудова маршрутів офлайн	Ні	Ні	Так	Так

З іншої сторони, OpenStreetMap надає офлайн-режим для карт, котрий дуже просто реалізувати у мобільному додатку, так само наявний API для побудови маршрутів на картах, безкоштовне застосування.

1.3 Архітектура і компоненти мобільного додатку

Перед виконанням розроблення гео-інформаційного мобільного додатку для ідентифікації пунктів незламності треба поточнити функціональні і нефункціональні вимоги, вимоги до інтерфейса користувача мобільного додатку. На основі зібраних вимог треба створити діаграму варіантів застосування системи,

структуру мобільного додатку, описати архітектуру створюваного додатку і базу даних.

1.3.1 Поточнення функціональних і нефункціональних вимог, визначення вимог до інтерфейса

Геоінформаційний мобільний додаток для ідентифікації пунктів незламності має задовольняти наступним функціональним вимогам:

- мобільний додаток має надавати клієнтам спроможність в будь-котрий час переглянути карту розміщення пунктів незламності на базі різноманітних громадських закладів і визначити їх статус (наявність електроживлення, додаткові послуги на кшталт меню в кафе);
- мобільний додаток має надавати власникам громадських закладів, котрі виявили бажання під'єднатись до системи, спроможність авторизуватися, додавати новий пункт незламності на карту і змінювати статус пункту незламності (наявність електроживлення);
- мобільний додаток має надавати спроможність клієнту переглянути адреси найближчих пунктів незламності на карті, де є електроенергія;
- мобільний додаток має надавати клієнту інформацію про даний громадський заклад, графік його роботи й т.д.

У ході аналізу функціональних вимог і виконаного вище огляду сформовані наступні нефункціональні вимоги:

- мобільний додаток має бути написаним мовою програмування Dart за поміччю фрейм-ворка Flutter;
- мобільний додаток має бути написаний під операційну систему Android версії 6.0 і вище;
- сервер і база даних мобільного додатку повинні бути розміщені на платформі Firebase;
- інтеграція географічної карти м.Одеса має бути виконана за поміччю сервісу OpenStreetMap;
- для збереження даних треба застосовувати Firebase Realtime Database;
- для забезпечення аутентифікації користувачів додатку треба застосовувати

Firebase Auth;

- для збирання метаданих треба застосовувати Firebase analytics;
- для збирання помилок треба застосовувати Firebase crashlytics.

Інтерфейс користувача мобільного додатку повинен відповідати таким вимогам:

- зручне представлення даних;
- застосування звичних елементів інтерфейса користувача: прямокутні кнопки, звичне розміщення меню;
- важливі елементи повинні знаходитися на першому плані;
- повинен бути заклик до дії, де вона необхідна.

Для підвищення якості і ефективності інтерфейса користувача мобільний додаток повинен відповідати наступним критеріям:

- інтуїтивно-зрозумілий інтерфейс. При першому вході у додаток користувач повинен легко й самостійно розібратися із функціоналом додатку;
- швидкий доступ. Важливим є те, скільки часу витрачає користувач на пошук необхідної інформації чи функції. Тому необхідно провести оптимізацію інтерфейса й зробити його більш простим і зрозумілим.
- емоційний вплив. Інтерфейс додатку не має бути перевантаженим і відволікати від основної мети – віднайти найближчий пункт незламності на карті м.Одеса. Аби зробити інтерфейс більш дружнім, треба гармонічно застосовувати кольори і форми об'єктів.

1.3.2 Розроблення варіантів застосування мобільного додатку

На рис. 1.21 представлено розроблену діаграму варіантів застосування мобільного додатку. Система для ідентифікації пунктів незламності вміщує два актора, котрі взаємодіють із системою. Перший актор – це клієнт, але другий – власник громадського закладу (пункту незламності).

Власник громадського закладу може зареєструватися і авторизуватися у додатку, розмістити на карті свій заклад в якості пункту незламності. Власник може змінювати статус свого закладу відносно забезпечення функцій виконання ролі

пункту незламності у залежності від наявності чи відсутності електроенергії в даний період після авторизації у додатку.

Рисунок 1.21. Узагальнена схема варіантів застосування мобільного додатку

Клієнт може зареєструватися й увійти до мобільного додатку із свого смартфону, авторизуватися в додатку, шукати найближчі пункти незламності на карті міста і переглядати послуги, котрі вони надають, зокрема меню для кафе.

На рис. 1.22 зображено діаграму компонентів мобільного додатку для ідентифікації пунктів незламності. Компонент інтерфейса користувача призначений для взаємодії із системою. Компонент авторизації визначає необхідність авторизації й надає спроможність реєстрування чи авторизації в систему. Компонент карти надає спроможність користувачу перейти до однієї із можливостей додатку, аби користуватися іншими можливостями програмного забезпечення. Компонент надає інформацію про розміщення пунктів незламності в місті Одеса і відповідає за пошук. Компонент інформації користувача дозволить переглянути інформацію при авторизації чи редагувати її. Компонент пам'яток надає спроможність переглянути додаткову інформацію про громадські заклади і відповідає за пошук. Компонент оновлення даних оновлює дані у базі даних програмного забезпечення, коли сервер системи доступний і пройшов певний час після попереднього поновлення даних.

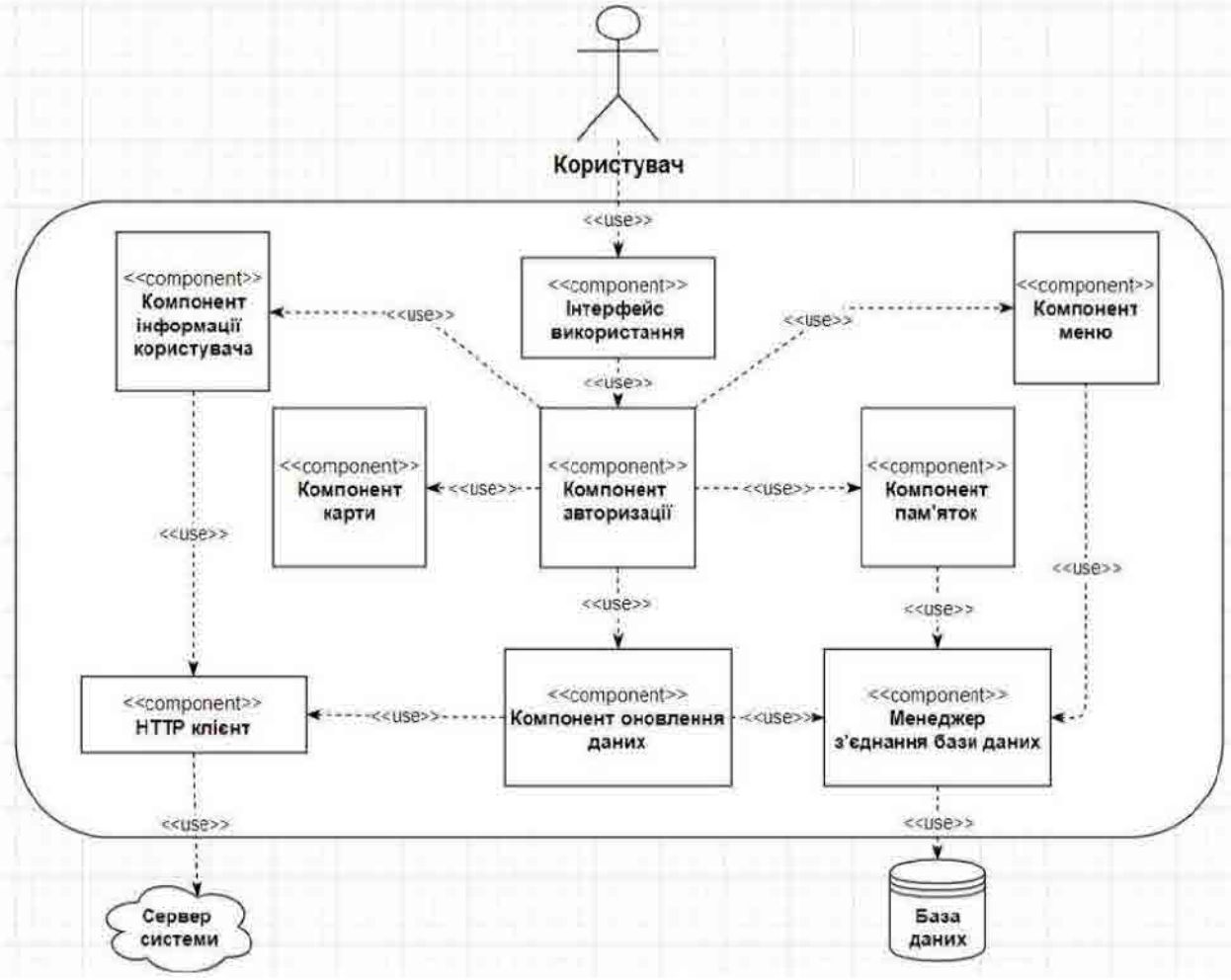


Рисунок 1.22. Діаграма компонентів мобільного додатку для ідентифікації пунктів незламності

Менеджер з'єднання бази даних вміщує конфігураційні дані для з'єднання із базою даних, скрипт ініціалізації бази даних, але так само скрипт створення схеми бази даних. HTTP-клієнт реалізує взаємодію із сервером системи за поміччю звернення до REST-інтерфейса сервера.

На рис. 1.23 показана вся взаємодія користувача із мобільним додатком. За діаграмою послідовності взаємодії користувача із системою видно, що додаток не виконує ніяких дій, доки користувач не взаємодіє із програмою, коли користувач виконує дію (зокрема реєстрацію чи авторизацію, перехід до розділів, що відображають інформацію, до карти розміщення пунктів незламності). Коли користувач виконав дію, котра потребує підтягування інформації зі сторони бази даних, то система відправляє запит до Firebase в форматі JSON, тоді база даних оброблює запит й вертає системі результат, але користувач отримує результат відповідно заданої ним дії.

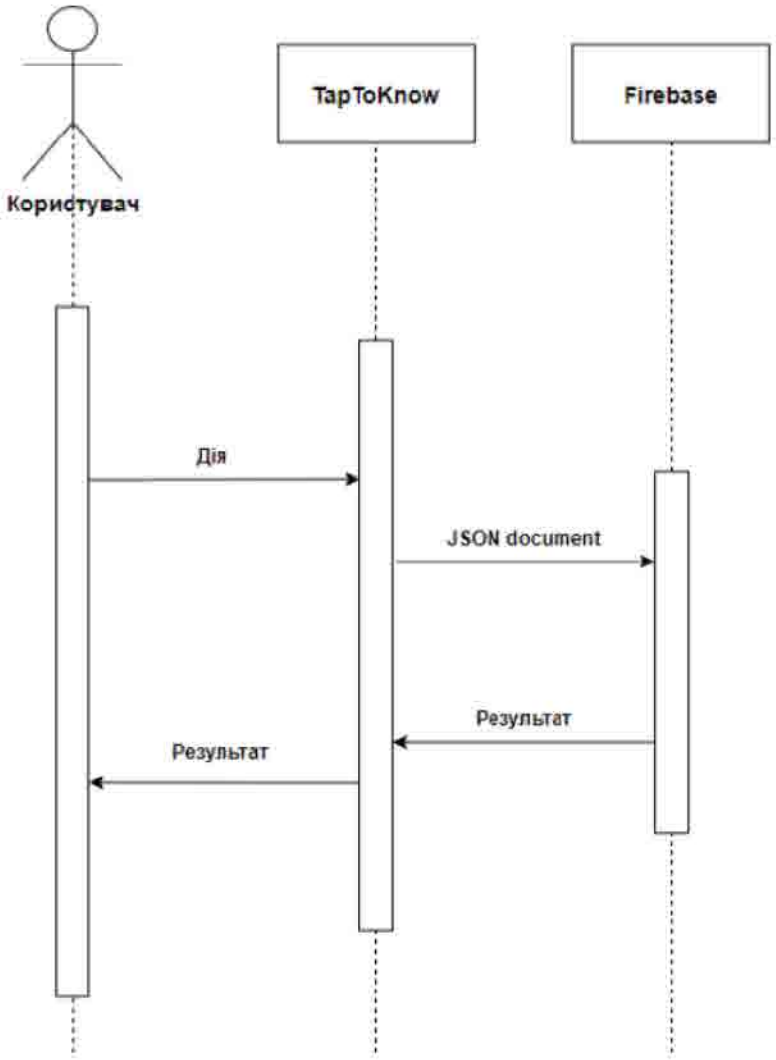


Рисунок 1.23. Діаграма послідовності взаємодії користувача із додатком

1.3.3 Розроблення загальної структури мобільного додатку

Створюваний мобільний додаток для ідентифікації пунктів незламності вміщує два актора, котрі взаємодіють із системою, й складається з екранів:

- стартовий екран;
- екран авторизації;
- екран авторизації для власника громадського закладу (потенційного пункту незламності);
- екран авторизації для клієнта ;
- екран власника громадського закладу;
- екран клієнта із відображенням карти розміщення пунктів незламності і інформації про пункти незламності;
- екран налаштувань.

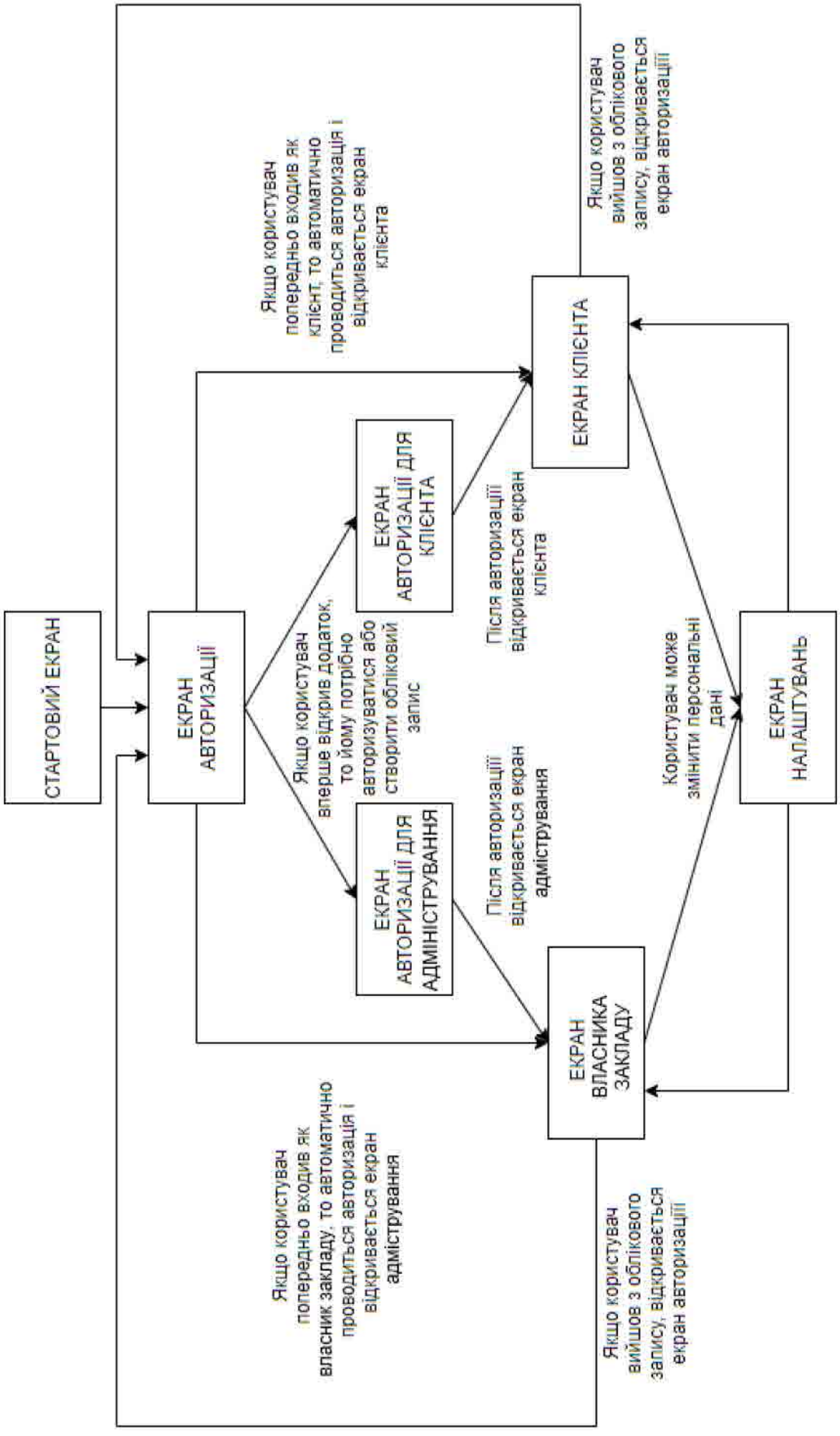


Рисунок 1.24. Узагальнена структурна схема мобільного додатку

Узагальнену структурну схему гео-інформаційного мобільного додатку для ідентифікації пунктів незламності наведено на рис. 1.24. в вигляді деревовидної діаграми, котра показує зв'язки між структурними елементами автоматизованої системи. Після запуску додатку відкривається стартовий екран із зображенням логотипу додатку eSvitlo!. Потім здійснюється перехід у екран авторизації. Коли користувач попередньо входив в додаток, то автоматично проводиться авторизація й відкривається його екран, зокрема карта розміщення пункті незламності для клієнта. Коли користувач вперше відкрив додаток, то на екрані авторизації треба обрати бажану кнопку, котра буде вести на екран авторизації для власника закладу, де від може задати нову точку пункту незламності і ввести інформацію про заклад, чи екран авторизації для клієнта. На екрані користувача можливо вийти із обліково запису й працювати із додатком без реєстрування.

1.3.4 Реалізація архітектури мобільного додатку

Розроблювана програмна система побудована на базі клієнт-серверної архітектури й складається із Android-додатка, сервера, бази даних Firebase. FrontEnd-частина додатку буде реалізована із використанням мови програмування Dart і фрейм-ворка Flutter. Так само буде використано OpenStreetMap для інтеграції географічної карти м.Одеса (рис.1.25).

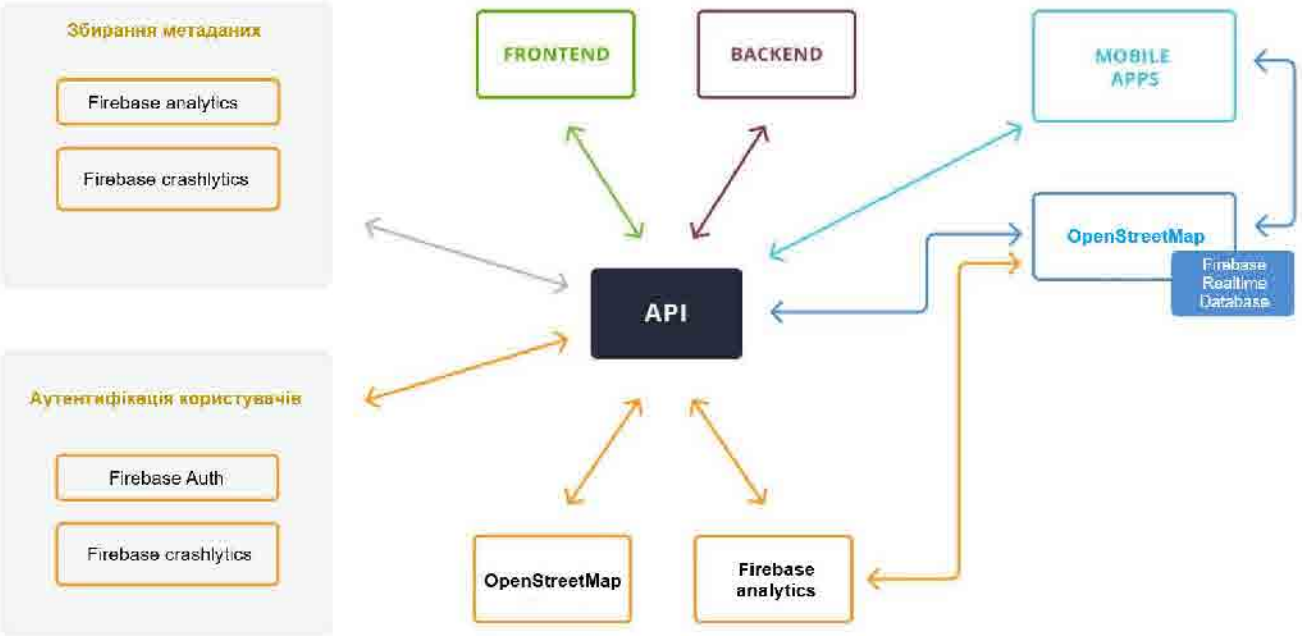


Рисунок 1.25. Архітектурна модель API додатку

BackEnd-частина додатку буде побудована на базі технологій Google Firebase. Для збереження даних буде використовуватися Firebase Realtime Database, що забезпечує ефективне і масштабоване збереження інформації. Для забезпечення аутентифікації користувачів додатку буде застосовано Firebase Auth. Для збирання метаданих буде застосовано Firebase Analytics, але для збирання помилок – Firebase Crashlytics.

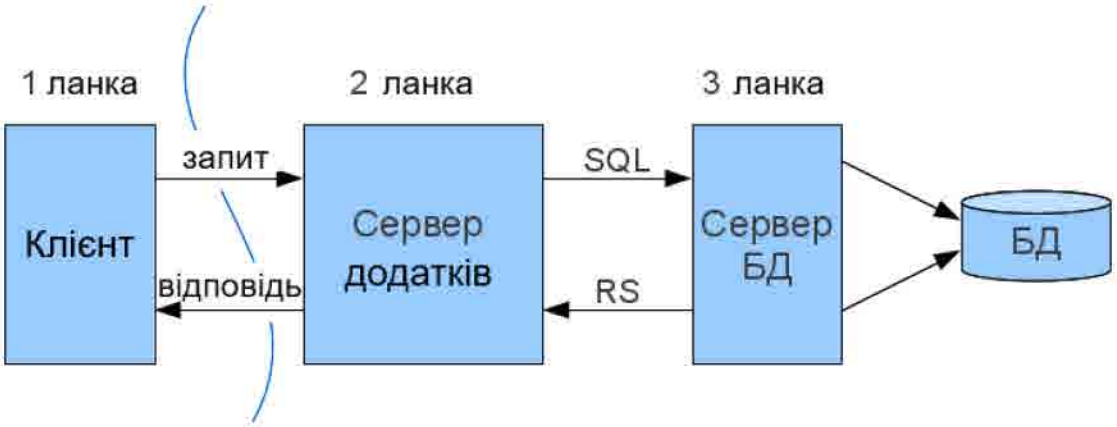


Рисунок 1.26. Схема клієнт-серверної взаємодії мобільного додатку

Архітектура клієнт-сервер (рис.1.26) більше всього підходить для даного типу програмного забезпечення, тому що усі дані зберігаються на сервері й швидко синхронізуються із різними пристроями. Клієнт передає на сервер дані, котрі використовуються службою автентифікації і зберігаються у базі даних. Для зберігання і синхронізації даних між клієнтом і сервером була обрана хмарна нереляційна база даних Firebase Realtime Database від компанії Firebase. Вона дозволить працювати із даними у текстовому форматі JSON і синхронізувати їх у реальному часі із кожним підключеним клієнтом. Дані зберігаються у будь-якому вигляді, не маючи визначеного типу даних. На відміну від реляційних баз даних, тут не має таблиць і записів. Коли дані додаються у дерево JSON, вони стають вузлом із прив'язаним ключем у існуючій структурі JSON. Для того, аби дані могли зчитуватись і записуватись, треба встановити наступні правила безпеки:

– «read» – визначає, чи спроможні зчитуватись дані із бази даних;
– «write» – визначає, чи спроможні дані записуватись у базу даних.

Спроектована база даних має чотири основних вузли:

– вузол запитів клієнтів «Customers Requests»;

– вузол власників громадських закладів «Owner Available»;
– вузол активних пунктів незламності на карті «Owner Working»;
– вузол користувачів «Users».

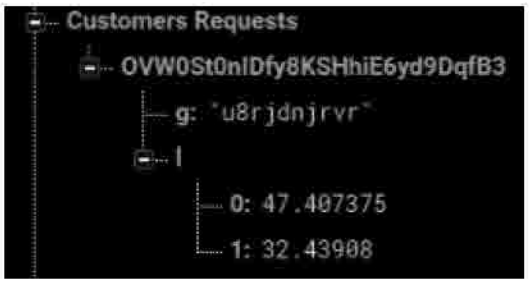


Рисунок 1.27. Структура вузла Customers Requests в структурі бази даних

Вузол «Customers Requests» вміщує дочірні вузли, ключ котрих визначається ключем зареєстрованого користувача, із точним місцезнаходженням пунктів незламності. Структура вузла «Customers Requests» показана на рис. 1.27.

Рисунок 1.28. Структура вузла Owner Available в структурі бази даних

В базі даних вузол «Owner Available» вміщує дочірні вузли, ключ котрих визначається ключем зареєстрованого користувача, із точним місцезнаходженням пунктів незламності на карті, котрі є активними у даний момент часу. Структура вузла «Owner Available» показана на рис. 1.28.

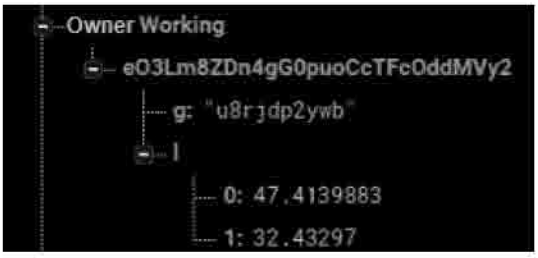


Рисунок 1.29. Структура вузла Owner Working в структурі бази даних

В базі даних вузол «Owner Working» вміщує дочірні вузли, ключ котрих визначається ключем зареєстрованого користувача, із точним місцезнаходженням

активних пунктів незламності на карті, в котрих є електроживлення. Структура вузла «Owner Working» показана на рис. 1.29.

Рисунок 1.30. Структура вузла Users в структурі бази даних

В базі даних вузол «Users» вміщує дочірні вузли «Customers» і «Owners». Вузол «Customers» вміщує дочірні вузли із даними клієнтів, ключ котрих визначається ключем зареєстрованого користувача.

Рисунок 1.31. Загальна структура спроектованої бази даних

Вузол «Owners» вміщує дочірні вузли із персональними даними власників громадських закладів, ключ котрих визначається ключем зареєстрованого користувача. Структура вузла «Users» показана на рис. 1.30.

На рис. 1.31 наведено загальний вигляд спроектованої бази даних для геоінформаційного мобільного додатку для ідентифікації пунктів незламності.

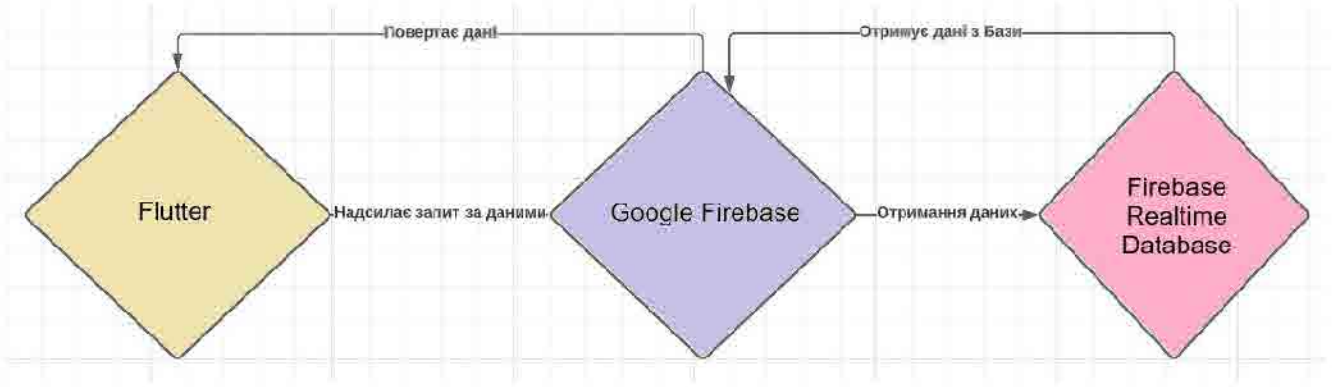


Рисунок 1.32. Схема запитів до спроектованої бази даних

У результаті при використанні механізмів фрейм-ворка Flutter і сервісів Firebase схема буде мати вигляд, наведений на рис.1.32.

Під час розроблення мобільного додатку eSvitlo використано компонентний підхід для створення інтерфейса користувача. Цей підхід дозволить розбити додаток на невеликі й самодостатні компоненти, котрі можливо повторно застосовувати і легко управляти. Кожний компонент відповідає за відображення певної частини інтерфейса чи виконання певної функціональності. Зокрема, компонент для відображення заголовка сторінки, компонент для відображення списку елементів чи компонент для взаємодії із сервером. Компоненти спроможні бути вкладені один у одного, що дозволить утворювати складні ієрархії інтерфейса. Зокрема, компонент сторінки, котрий вміщує компоненти заголовка, списку елементів й форми вводу.

В даному додатку використано модульний підхід для розроблення Backend (рис.1.33). Це означає, що код розбито на невеликі, самодостатні модулі, кожний із котрих відповідає за свою конкретну функціональність. Кожний модуль виконує певну роботу й має чітко визначені обов'язки, що спрощує розробку і підтримку коду. Можливо розробляти і тестувати кожний модуль незалежно, що дозволить прискорити розвиток проекту. Крім того, модульна структура дозволить

використовувати код, зокрема, можливо застосовувати модулі у інших проектах чи у різноманітних частинах додатку. Завдяки модульному підходу отримано більш організований і масштабований код, що полегшує його розуміння і роботу із ним.

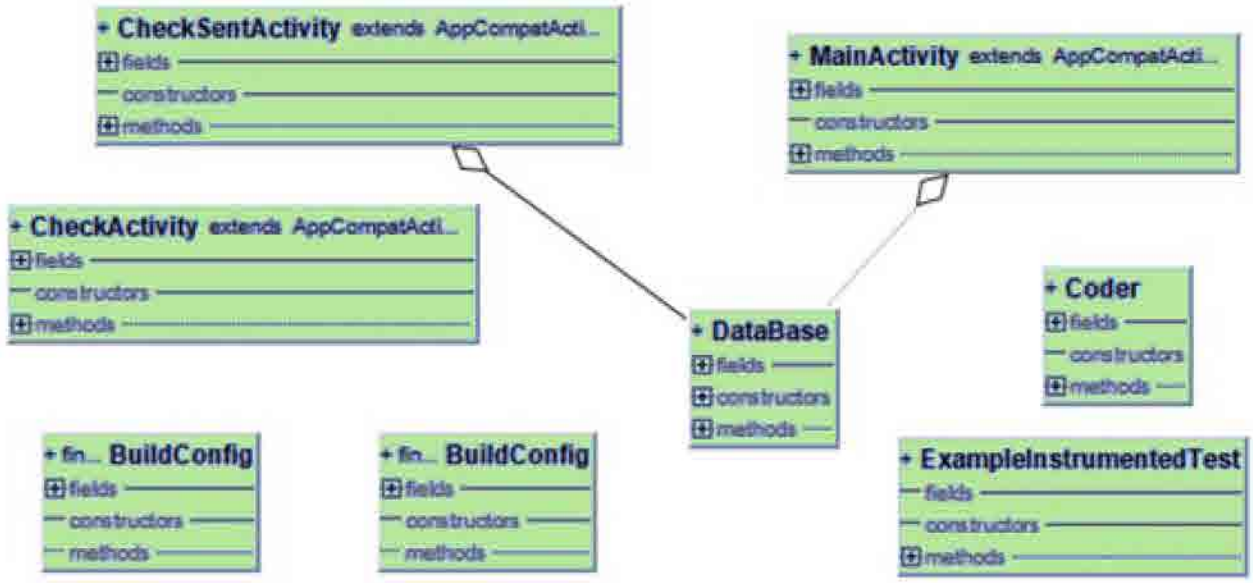


Рисунок 1.33. Схема класів в мобільному додатку

1.4 Розроблення елементів додатку і інтерфейса користувача

Розроблення додатка для ОС Android виконувалась в середовищі Flutter (рис.1.34).

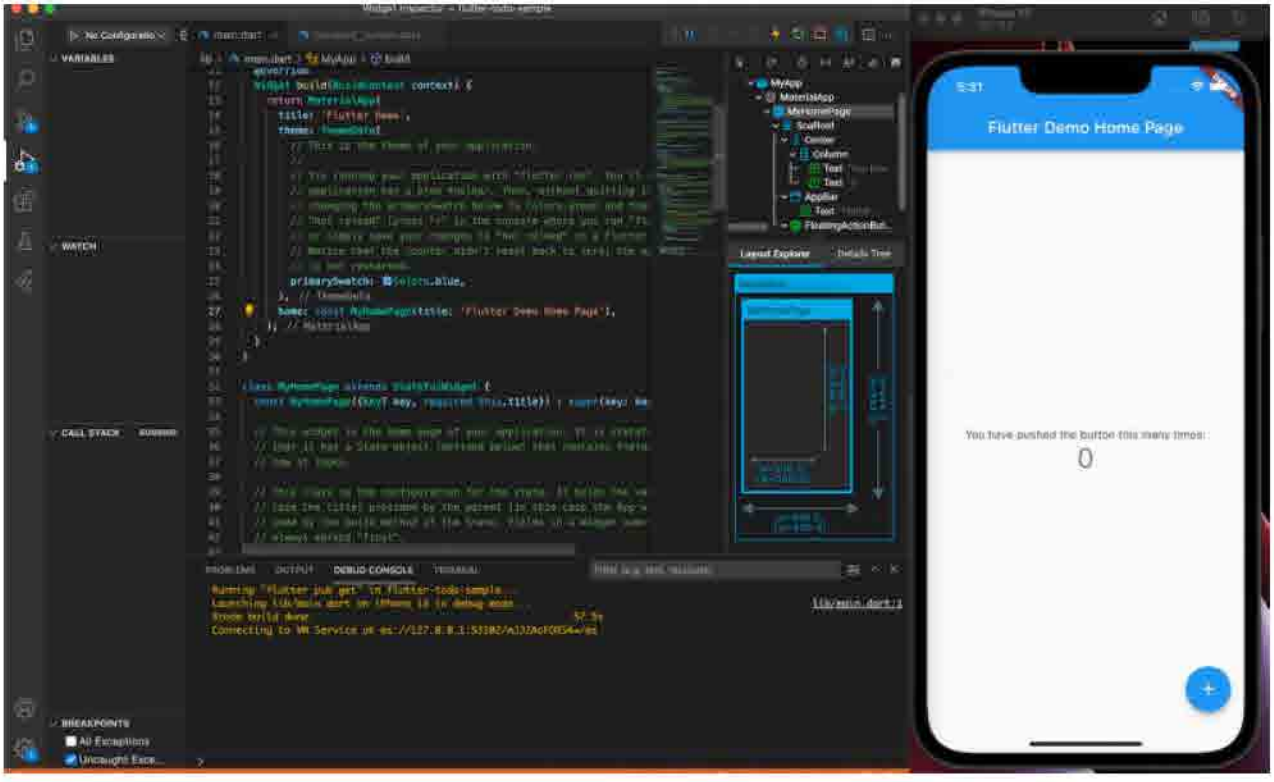


Рисунок 1.34. Етап розроблення інтерфейса користувача в фрейм-ворка Flutter

Сервіс Firebase Authentication було використано для реалізації функцій автентифікації. Після реєстрування користувачу присвоюється унікальний ідентифікатор, але його профіль зберігається в хмарі Firebase. В додатку був використаний єдиний контролер для реєстрування і входу, що потребувало розробити алгоритм для реалізації цього підходу (рис. 1.35). Коли користувач входить до системи, відбувається запит до серверу автентифікації із даними, що впровадив користувач, після цього відбувається перевірка чи була автентифікація вдалою, й коли невдала – відбувається відображення помилки.

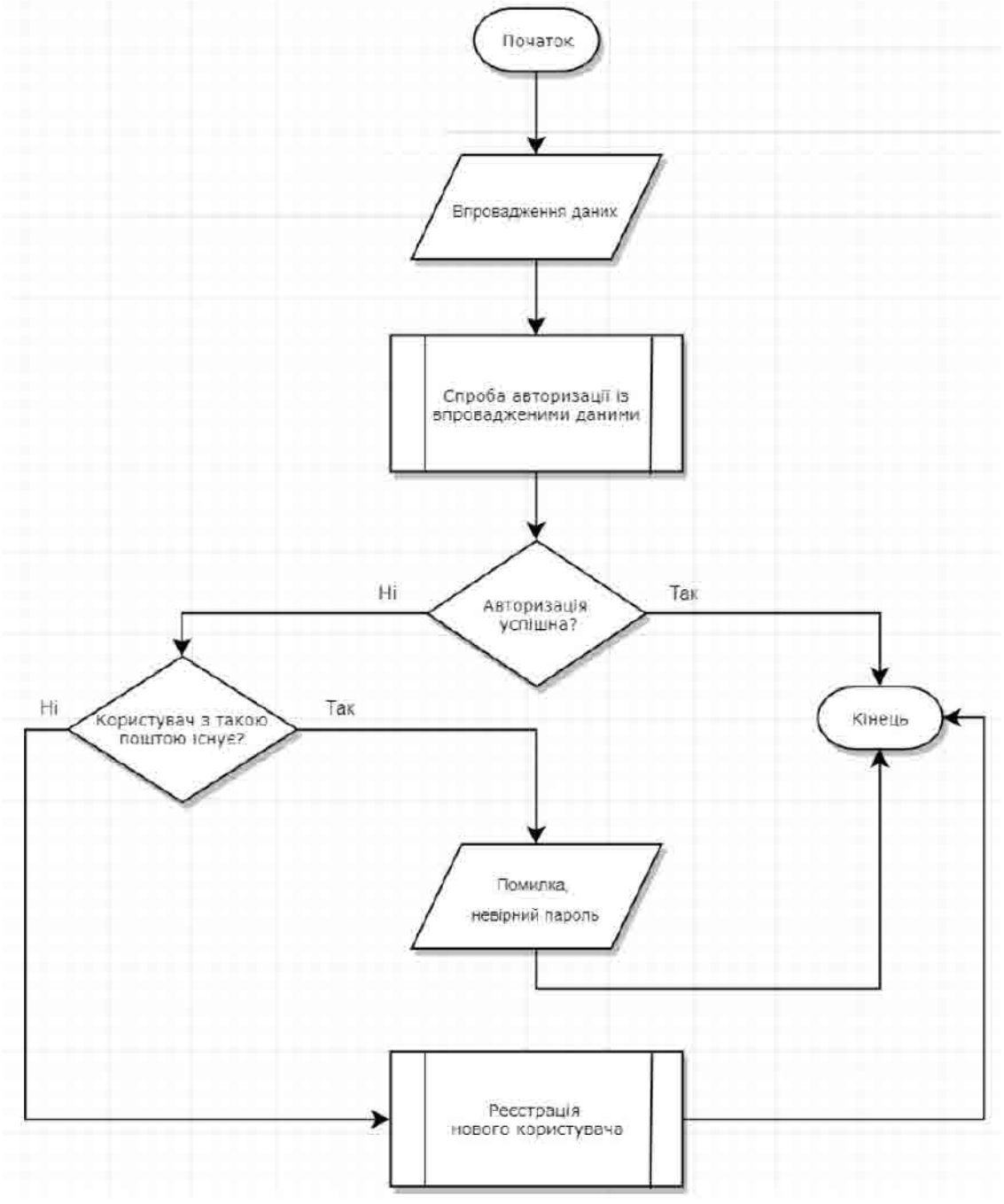


Рисунок 1.35. БСА реєстрування і входу до додатку

Таблиця 1.3. Клас фрейм-ворка Firebase Auth

Клас Firebase Auth		
Властивості		
app	currentUser	languageCode
settings		
Функції		
signInWithCustomToken	signInWithCredential	signInWithEmailLink
signInWithPopup	signOut	createUser

Клас Auth в фрейм-ворка Firebase є головним класом для авторизації, автентифікації і реєстрування. В табл. 1.3 наведено таблицю основних функцій класу Auth. В додатку застосовується функція “createUser”, реалізація якої інкапсульована в фрейм-ворка Firebase в класі Auth. В цю функцію передаються дані користувача. В випадку, коли після запиту до серверу Firebase не відбулося помилок, викликана функція має повернути об’єкт класу User, що вміщує в собі функції і властивості (табл. 1.4).

Таблиця 1.4. Клас фрейм-ворка Firebase User

Клас Firebase User		
Властивості		
displayName	email	emailVerified
isAnonymous	metadata	phoneNumber
photoURL	providerData	providerId
refreshToken	uid	
Функції		
delete	getIdToken	getIdTokenResult
updateEmail	updatePassword	updatePhoneNumber

Об’єкт класу User має властивість “uid”, що є унікальним ідентифікатором, котрий асоціюється із об’єктом користувача. Унікальний ідентифікатор (UID) є числовим чи буквено-цифровим значенням, що являє собою унікальне ім’я об’єкту в межах певної системи. Основна задача UID – бути унікальним показником певного об’єкту, зокрема в базі даних, для того аби одержати доступ до певної

сутності і взаємодіяти із нею.

UID користувача об'єкту класу Points використано в базі даних реального часу Realtime Database в якості ідентифікатору, за яким можливо одержати інформацію про розміщення пункту незламності, але саме про його координати, координати, статус, але так само інші логічні значення (табл.1.5).

Users		
Унікальний ідентифікатор користувача (UID)		
Ключ	provider	Строкове значення що визначає походження реєстрації
Coordinate		
Ключ	0	Широта
Ключ	1	Довгота

На базі впроваджених адміністратором додатку даних про розміщення на карті пунктів незламності, створюються об'єкти в базі даних в середині об'єкту "points" чи "users". Адже база даних є нереляційною, кожний запис є об'єктом із набором властивостей і інших об'єктів (рис. 1.36).

Рисунок 1.36. Організація нереляційної бази геоданих в додатку

Кожне місцезнаходження на землі має глобальну адресу – широту й довготу. Адже ця "адреса" відображається в цифрах, можливо повідомляти про місцезнаходження незалежно від обраної в додатку мови, але так само застосовувати ці значення для геолокації в форматі чисел. Глобальна адреса

надається як два числа, котрі є координатами пунктів незламності. При цьому реалізується фільтрація геоданих в додатку на базі сервісу OpenStreetMap відповідно до рис. 1.37.

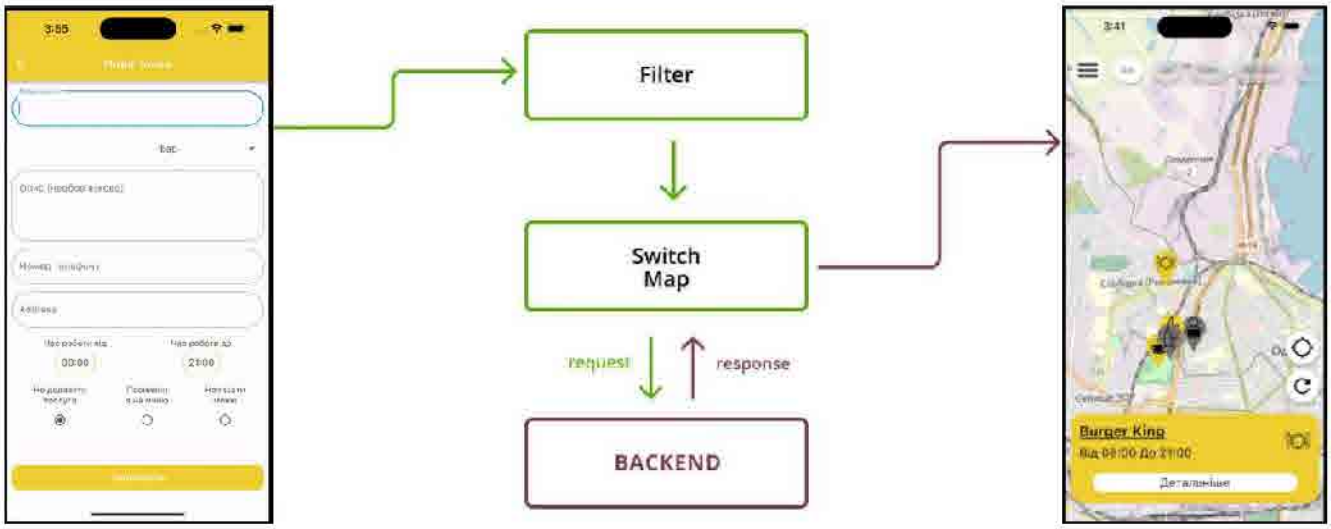


Рисунок 1.37. Загальна схема процесу фільтрації геоданих в додатку

Місцерозташування пункту незламності на карті і позицію самого користувача можливо відображати чи віднайти на сітці шляхом опрацювання двох чисел, котрі є горизонтальними й вертикальними координатами розміщення.

Основні особливості із ініціалізацією серверу при реалізації коду наступні:

- створюється екземпляр Express-додатку const app = express();
- застосовується middleware cors, котрий дозволить крос-доменні запити;
- шляхом додавання заголовку Access-Control-Allow-Origin:* до відповідей сервера.
- створюються роути для обробки запитів. Роут /points обробляє GET-запити і повертає список пунктів незламності за поміччю citiesHandler(). Роут /points/:id/shelters/ повертає пункти незламності, в котрих є електроенергія, за поміччю shelterListHandler(id).

При відсутності параметрів запиту відсутні чи їх некоректності, сервер відправляє відповідні статуси помилок й повідомлення.

Handler є функцією (модулем), котрий виконує обробку певної операції чи запиту. У даному випадку shelterListHandler є handler, котрий обробляє запит на отримання списку пунктів незламності, в котрих є електроенергія.

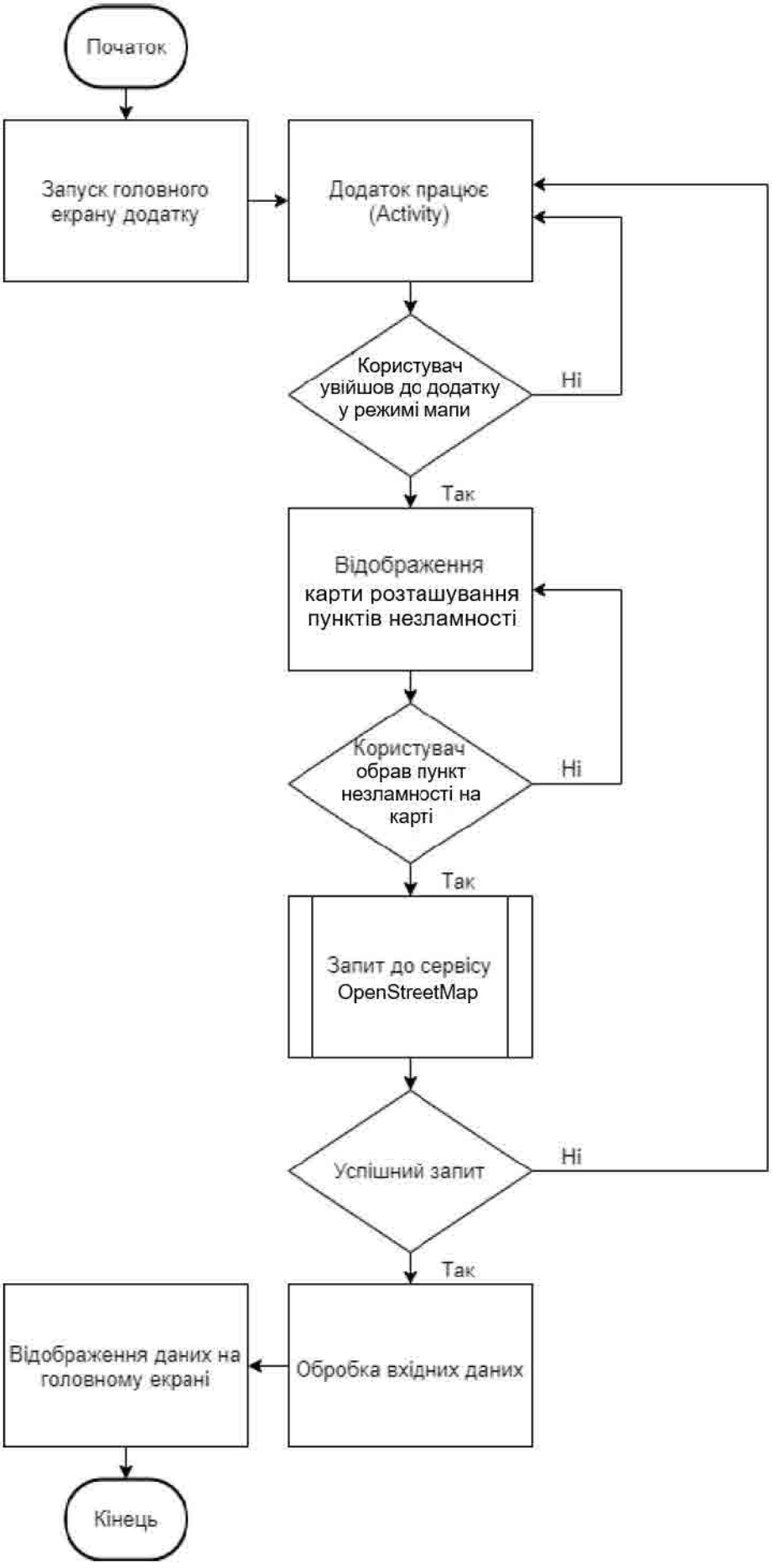


Рисунок 1.38. БСА обробки дій користувача в додатку

Він приймає ідентифікатор пункту незламності як вхідний параметр, знаходить відповідне значення і фільтрує список закладів, аби повернути результат знайденого пункту незламності (рис.1.37). За поміччю методу find фільтрується список пунктів незламності data.points, аби віднайти місце із відповідним ідентифікатором. За поміччю методу filter фільтрується список закладів із використанням data.shelters, аби віднайти усі пункти незламності, в котрих є електроенергія (shelter.PointId === id). Повертається об'єкт, що вміщує знайдений пункт незламності, як об'єкт типу { point: tPoint; shelterList: Array<tShelter> }.

1.5 Реалізація додатку і його тестування

1.5.1 Створення екрану входу до додатку

БСА обробки дій користувача в додатку наведено на рис.1.38. При першому запуску додатку відображається екран входу і реєстрування (рис.1.39) із можливістю входу в режим перегляду карти без необхідності реєстрування.

Рисунок 1.39. Екран входу і реєстрування в мобільному додатку

Екран входу до додатку дозволить обрати одну із трьох мов інтерфейса (українську, російську, англійську). В режимі перегляду без реєстрування гість системи може продивитись карту, де позначені активні пункти незламності і визначити, де зараз є електроживлення, продивись перелік додаткових послуг. В режимі перегляду із входом за поміччю профілю Google користувач системи може працювати із картою, де додавати в режимі адміністратора нові пункти незламності, вказувати перелік додаткових послуг. Після вибору пункту меню “Користувач” в меню додатку із головного екрану буде відбуватись перехід на екран виходу із акаунту (рис.1.40).

Рисунок 1.40. Екран виходу із акаунту в мобільному додатку

1.5.2 Створення головного екрану додатку

Після авторизації користувач потрапляє на головний екран гео-інформаційного мобільного додатку для ідентифікації пунктів незламності (рис.1.41), де має доступ до головного функціоналу. Навігаційна панель із лівого боку чи меню відображає такі пункти:

– мапа – відображає карту за поміччю сервісу OpenStreetMap;
– користувач – відображає екран входу/виходу із акаунту;
– обране – відображає додані користувачем в список пункти незламності;
– допомога.

При виборі пункту «Мапа» на карті відображаються тільки ті пункти незламності, котрі були додані в систему.

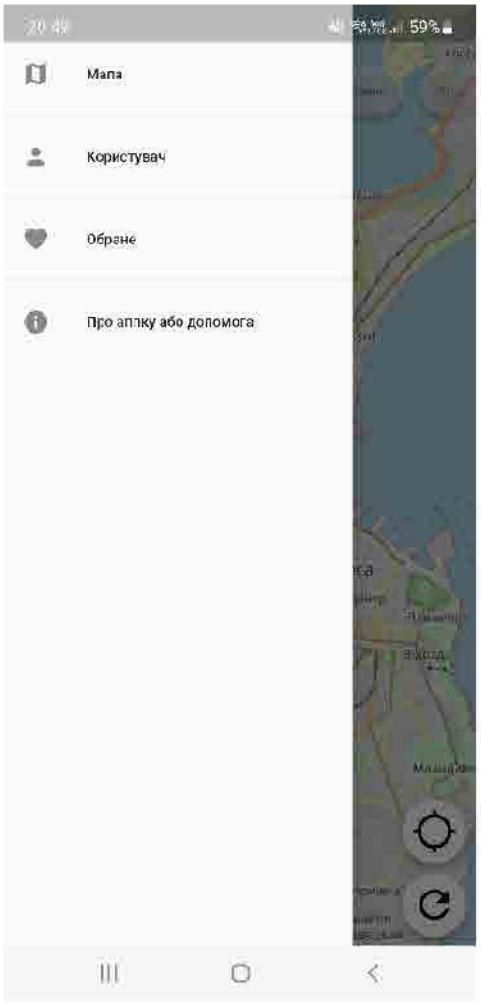


Рисунок 1.41. Меню на головному екрані додатку

1.5.3 Створення екрану мапи додатку

На екрані «Мапа» відображаються й заклади, в котрих є електроживлення, й заклади, в котрих електроживлення нема, що позначаються сірим кольором геоміток на карті (рис.1.42).

В верхній панелі додатку розташовано 6 кнопок: ”Усі”, ”Бар”, ”Кафе”, ”Ресторан”, ”Пункт незламності”, ”Коворкінг”. Це дозволить реалізувати фільтрацію пунктів на карті і відображувати тільки обрані чи усі заклади міста, що

були додані в додаток на карту.

В нижній правій частині візуального інтерфейса мобільного додатку розташовані стандартні кнопки ГІС-додатку: Дозволити визначення і відстежування гео-координат мобільного пристрою користувача і оновлення зображення карти.

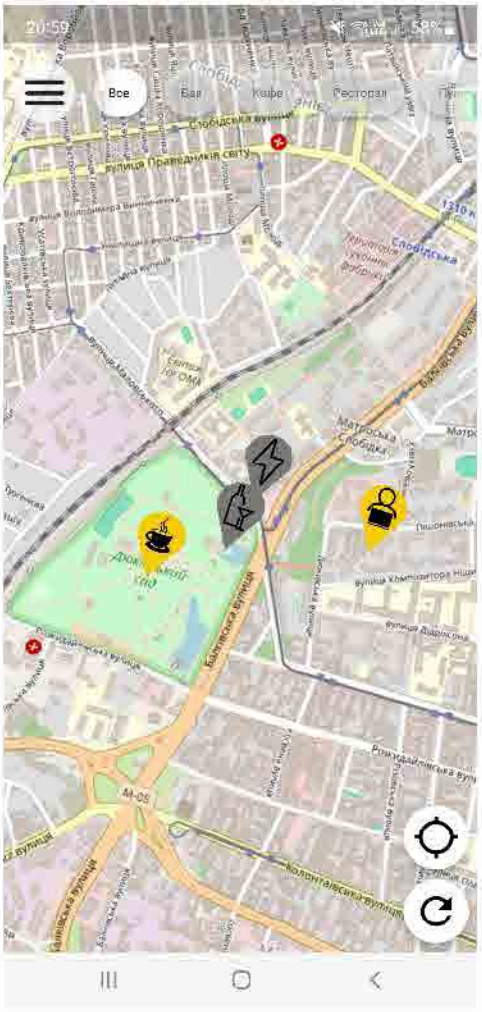


Рисунок 1.42. Екран мапи із позначенням пунктів незламності додатку

1.5.4 Створення екрану перегляду інформації про заклад

Екран перегляду інформації про громадський заклад викликається при натисненні геомітки, розташованої на мапі й натиснення кнопки “Детальніше” (рис. 1.43). При цьому відображується графік роботи громадського закладу, список додаткових послуг (зокрема меню, вартість додаткових послуг). По кольору інформаційного екрану, що відкрився (жовтий чи сірий), можливо визначити статус пункту незламності, але саме – наявність електроживлення (рис.1.44).

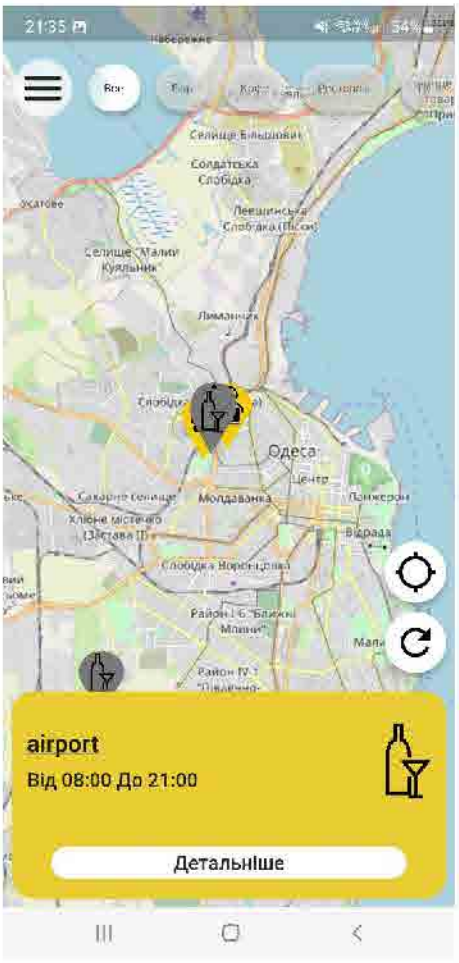


Рисунок 1.43. Панель для виклику інформації про громадський заклад в додатку

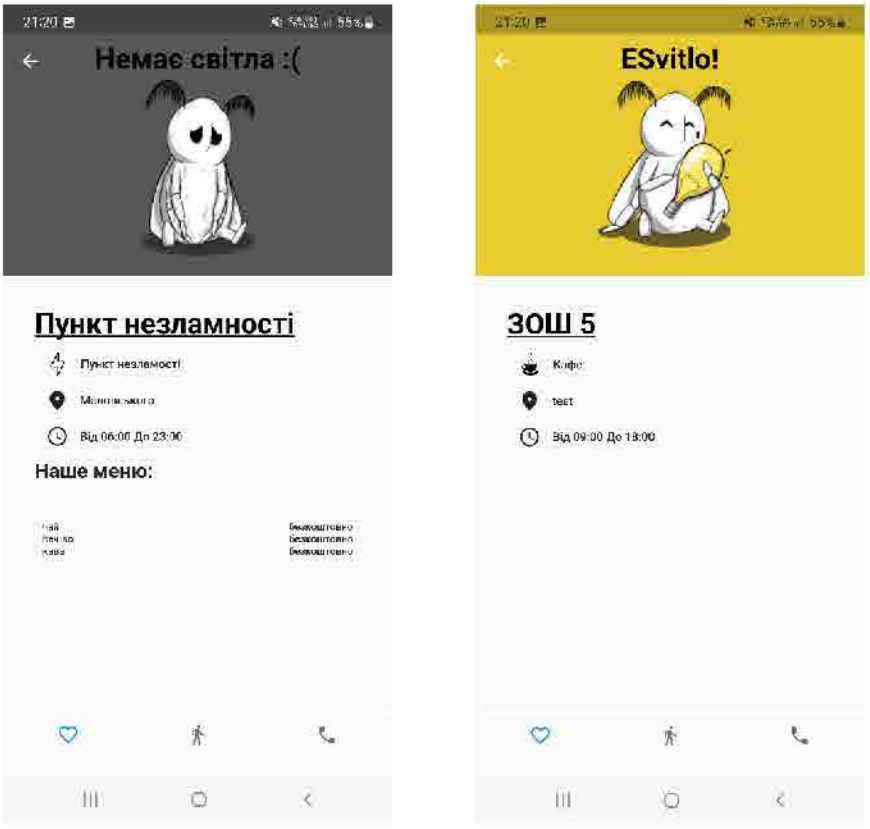


Рисунок 1.44. Екран перегляду детальної інформації про заклад в додатку

На екрані перегляду інформації про громадський заклад передбачено спроможність натиснути на “серці” і додати громадський заклад до обраного (рис. 1.45).

Рисунок 1.45. Додавання громадського закладу до обраного в додатку

1.5.5 Додавання нового пункту незламності в режимі адміністратора

При вході до додатку в режимі адміністратора (за відповідним профілем Google) передбачено додавання на карту нових точок – пунктів незламності, в якості котрих спроможні бути різні громадські заклади, котрі дали згоду на під’єднання до системи и надають відповідні послуги.

В режимі адміністратору з’являється спроможність після вибору геомітки із позначкою громадського закладу на карті ввести назву закладу, обрати тип закладу (Бар, Кафе, Ресторан, Пункт незламності, Коворкінг), вказати номер телефону і адресу, обрати час роботи закладу, зробити посилання чи створити меню із додатковими платними/безоплатними послугами (рис. 1.46).

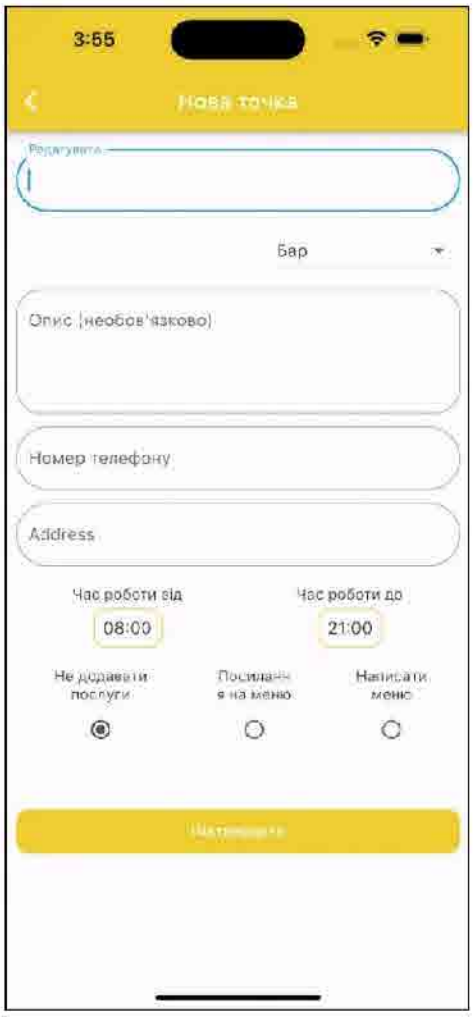


Рисунок. 1.46. Екран додавання нового закладу в мобільному додатку

Етапи реалізації мобільного додатку для ідентифікації пунктів незламності наведені схематично на рис.1.47. Код програми Backend-частини програмного застосунку, складеного мовою Dart, наведено в Додатку АЛЕ. Під час тестування програмного застосунку перевірено усі його можливості, зокрема пошук й ідентифікація пунктів незламності на карті, перегляд докладної інформації про громадський заклад, додавання пунктів незламності до панелі обраного, аутентифікація користувача і його вхід без авторизації (рис.1.39-1.46).

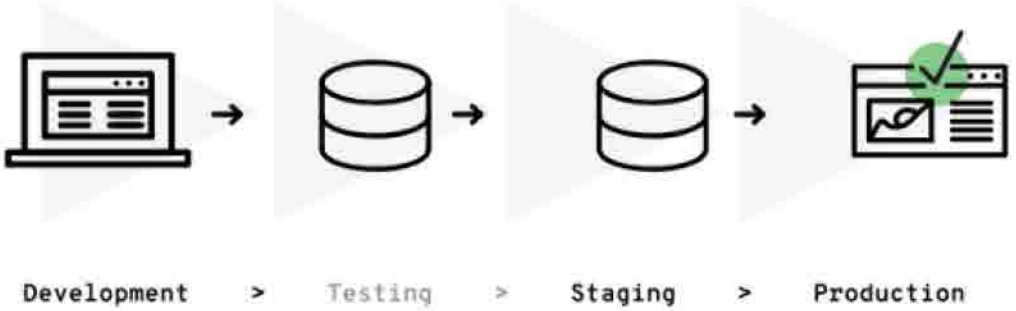


Рисунок. 1.47. Етапи реалізації програмного додатку

2 РОЗДІЛ ОХОРОНИ ПРАЦІ І ТЕХНІКИ БЕЗПЕКИ

Охорона здоров'я працівників, забезпечення безпечних умов праці, ліквідація професійних захворювань й виробничого травматизму складають одну із головних турбот нашої держави.

Трудові права громадян охороняються законом. Захист трудових прав здійснюється державними органами, але так само професійними спілками

Відповідно до Конституції України, громадянам забезпечується рівноправність в області праці, незалежно від національності й раси. Жінці у Україні надані рівні із чоловіком права на працю, оплату праці й соціальне забезпечення.

Випускною роботою передбачалася розроблення гео-інформаційного мобільного додатку для ідентифікації пунктів незламності.

Розроблення програмного застосунку проводилася за поміччю ПК. Робота може кваліфікуватися як робота програміста.

Вибір технічних засобів забезпечення безпеки повинен здійснюватися на основі вивчення особливостей кожного виявленого небезпечного й шкідливого виробничого фактору й зони його дії – так званої небезпечної зони.

2.1 Аналіз небезпечних і шкідливих чинників, що впливають на працівника

Аналіз умов праці, технологічних процесів, апаратури й обладнання із точки зору можливості виникнення появи небезпечних факторів, виділення шкідливих виробничих речовин. На основі такого аналізу визначаються небезпечні ділянки виробництва, можливі аварійні ситуації, розробляються заходи відносно їх усунення чи обмеження наслідків.

В розділі охорона праці дипломного проекту розглядається питання створення безпечних й здорових умов праці для програміста із застосування персонального комп'ютера. Аналіз умов праці показує, що на працівників спроможні негативно впливати наступні фізичні і психофізіологічні фактори:

– підвищені чи знижені температура, вологість повітря робочої зони;

– недостатня освітленість робочого місця;
– підвищений рівень шуму на робочому місці;
– підвищені іонізація повітря і рівень електромагнітних хвиль;
– нервово-психічні і фізичні перевантаження.

2.2 Розроблення заходів із охорони праці

2.2.1 Мікроклімат робочої зони

Робота за енерговитратами відноситься до категорії легких робіт 1а, 1б, тому повинні дотримуватися наступні вимоги згідно ДСанПіН 3.3.2.-007-98.

Таблиця 2.1 Норми мікроклімату для приміщень із ВДТ ЕОМ і ПЕОМ

Пора року	Категорія робіт	Температура повітря, С, не більше	Відносна вологість повітря %	Швидкість руху повітря, м/с
Холодна	Легка-1а	22-24	40-60	0,1
	Легка-1б	21-23	40-60	0,1
Тепла	Легка-1а	23-25	40-60	0,1
	Легка-1б	22-24	40-60	0,1

Рівні позитивних й негативних іонів в повітрі приміщень із ВДТ мають відповідати санітарно-гігієнічним нормам № 2152-80.

Таблиця 2.2 Рівні позитивних й негативних іонів в повітрі

Рівні	Число іонів у 1 $см^3$ повітря	Число іонів у 1 $см^3$ повітря
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

Для підтримки у приміщенні нормального, що відповідає гігієнічним вимогам, складу повітря, видалення із нього шкідливих речовин використовують вентиляцію. При природній вентиляції (за поміччю вікон) повітря надходить в приміщення й видаляється внаслідок різниці температур. Але вона має низку недоліків. Тому в приміщенні застосовується штучна, загально обмінна вентиляція,

котра очищає повітря й направляє його до робочого місця. Повітря, перед його споживанням можливо піддати обробці: підігріти, зволожити, охолодити тощо.

2.2.2 Виробниче освітлення

Освітлення приміщення має природне і штучне походження. Природне освітлення подається крізь віконні прорізі, бокове. Для штучного освітлення в приміщенні використовуються люмінесцентні лампи, котрі у порівнянні із лампами розжарювання мають ряд істотних переваг. Так за спектральним складом світла вони близькі до природного світла, мають підвищену світлову віддачу, триваліший термін служби. Норма освітленості на робочих місцях складає 300-500лк.

- Випадковий дотик до струмоведучих частин, в результаті ведення робіт поблизу чи на цих частинах;
- Несправність захисних засобів, якими потерпілий доторкався до струмоведучих частин;

2.2.3 Електробезпека

Значення сили струму, що проходить крізь організм людини, залежить від напруги, під якою перебуває людина й від опору ділянки тіла, до якого прикладена ця напруга. Джерелом живлячої напруги є мережа змінного струму із напругою 229В, на яку поширюється ГОСТ 25861-83. Основними причинами електротравматизму є:

- напругою, не відключеною;
- несподіване виникнення напруги крізь ушкодження ізоляції там, де у нормальних умовах його бути не повинно;
- контакт струмопровідного устаткування із проводом, що перебуває під напругою.

Для попередження поразок електричним струмом необхідно чітко й в повному обсязі виконувати правила провадження робіт й правил технічної експлуатації. Необхідно виключити спроможність доступу оператора до частин устаткування, що працює під небезпечною напругою, до неізольованим частинам,

призначеним для роботи при малій напрузі й не підключеним до захисного заземлення, але так само підводити електроживлення до ПЕОМ від розетки за поміччю спеціальної вилки із заземлюючим контактом.

2.2.4 Вимоги до приміщень

Робочі місця із ВДТ повинні, як правило, розміщуватися у окремих приміщеннях. В випадку розміщення робочих місць в спеціальних залах чи приміщеннях із джерелами небезпечних виробничих факторів вони повинні розміщуватися в повністю ізольованих кабінетах із природним освітленням і організованим повітрообміном. Площа, на якій розташовується одне робоче місце із ВДТ, повинна становити не менше як 6,0 м^2, об'єм приміщення – не менше як 20,0м^3.

Поверхня підлоги має бути рівною, без вибоїн, неслизькою, зручною для очищення і вологого прибирання, мати антистатичні властивості.

Забороняється застосовувати для оздоблення інтер'єру полімерні матеріали, що виділяють в повітря шкідливі хімічні речовини.

Усі виробничі, але так само допоміжні приміщення – коридори, східці, проходи – повинні утримуватися у чистоті й порядку у відповідності до санітарних правил для підприємств.

Розміщення робочих місць із ВДТ ЕОМ й ПЕОМ в підвальних приміщеннях, на цокольних поверхах заборонено. При приміщеннях із ВДТ мають бути обладнанні побутові приміщення для відпочинку, кімната психологічного розвантаження.

2.2.5 Організація робочого місця

Обладнання й організація робочого місця із ВДТ мають забезпечувати відповідність конструкцій всіх елементів робочого місця і їх взаємного розміщення, ергономічним вимогам, із урахуванням характеру й особливостей трудової діяльності (ДСанПіН 3.3.2.-007-98).

Конструкція робочого місця й взаємне розміщення всіх його елементів (сидіння, органи керування, засобу відображення інформації) відповідають

антропометричним, фізіологічним й психологічним вимогам, але так само характеру роботи. Конструкція робочих меблів дає спроможність забезпечувати спроможність індивідуального регулювання їх відповідно до потреб працівника для підтримки зручної пози. Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей, на відстані близько 70 см, що укладається у припустимі рамки від 60 до 90 см. Частота мерехтіння екрана дорівнює 100 Гц, що відповідає умові більше 70 Гц.

Для зниження нервово-емоційного напруження, стомлювання, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно впроваджувати виконання комплексу вправ, котрі наведені в Державних санітарних правилах й нормах роботи із візуальними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98.

2.3 Пожежна безпека

Під пожежною безпекою розуміють систему державних й суспільних заходів, спрямованих на охорону від вогню людей й власності. Пожежна безпека приміщень, що мають електричні мережі, регламентується ГОСТ 12.1.033-81, ГОСТ 12.1.004-85. Робота оператора ЕОМ повинна вестися у приміщенні, що відповідає категорії Д пожежної безпеки (негорючі речовини й матеріали у холодному стані).

Пожежна безпека об'єкта забезпечується:

- Системою запобігання пожежі;
- Системою протипожежного захисту;
- Організаційно-технічними заходами.

Усі приміщення повинні бути забезпечені первинними засобами пожежогасіння: пожежним водопостачанням (пожежні крани ПК), пожежні щити із набором пожежного інструменту, вуглекислотними чи порошковими вогнегасниками. В випадку виникнення пожежі необхідно відключити електроживлення, викликати по телефону 101 пожежну команду, евакуювати людей із приміщення відповідно до плану евакуації й приступити до ліквідації.

ВИСНОВКИ

В даній випускній роботі виконано розробку гео-інформаційного мобільного додатку для ідентифікації пунктів незламності. Розроблений програмний Android-додаток eSvitlo! розміщений в магазині додатків Google Play для вільного скачування всіма охочими користувачами й має на меті надати користувачам інформацію про найближчі громадські заклади (кафе, бари, бізнес-центри, коворкінг-центри й т.д.) в м.Одеса і допомогти їм віднайти точки із найбільш доречними умовами й набором додаткових послуг, але перш за все – із наявністю електроенергії в даному закладі.

Метою виконання даної кваліфікаційної роботи так само був аналіз й порівняння існуючих технологій розроблення мобільних додатків, зокрема фреймворків, сервісів, технологій геоінформаційних систем.

Створений мобільний додаток має візуальний сенсорний інтерфейс й призначений для роботи на Android-пристроях. Основне призначення додатка полягає саме в допомозі користувачам віднайти заклади на карті м.Одеси, у котрих є доступ до мережі електроживлення. В режимі адміністратора додаток дозволить додавати нові точки (громадські заклади) на карту. Усі режими роботи додатку протестовані не тільки розробником даного додатку, але й багатьма користувачами, котрі скачали даний додаток із Google Play.

Застосунок використовує архітектурний підхід, базуючись на клієнт-серверній моделі. Застосунок реалізовано із використанням мови програмування Dart і фрейм-ворка Flutter. Так само було використано OpenStreetMap для інтеграції географічної карти м.Одеса. BackEnd-частина додатку побудована на базі технологій Google Firebase. Для збереження даних застосовується Firebase Realtime Database, що забезпечує ефективне і масштабоване збереження інформації. Для забезпечення аутентифікації користувачів додатку застосовано Firebase Auth. Для збирання метаданих застосовано Firebase analytics, але для збирання помилок – firebase crashlytics.

Створений мобільний додаток може значно допомогти у складний час для нашої країни.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Кузьміних У.О. Управління версіями програмних засобів проекту: Навчальний посібник – КПІ ім. Ігоря Сікорського, 2023.
2. Дункан Е., Ларссон М., Ляшенко М. Бази даних: від основ SQL до проектування баз даних. Київ: Видавництво "Дія", 2022. – 320 с.
3. Володимир Гайдаржи, Ігор Ізварін. Бази даних у інформаційних системах. / Університет «Україна», 2018, – 113-114с.
4. Створення додатку на Flutter: перші кроки: [Веб-сайт]. URL: https://dou.ua/lenta/articles/flutter-first-steps/.
5. Polienova V.A., Fedenko V.A., Tytenko, S.V. Educational mobile application based on concept maps: Modern engineering and innovative technologies, issue 23, part 1, 2022. pp. 12-18.
6. Rap Payne. Beginning App Development with Flutter: Create Cross-Platform Mobile Apps 1st ed. Edition, 2019. 74 p.
7. Mariot Tsitoara: Beginning Git and GitHub: A Comprehesive Guide to Version Control, Project Management, and Teamwork for the New Developer. https://cutt.ly/nnoZHPn
8. Dart Documentation: [Веб-сайт]. https://dart.dev/guides/ (англійською мовою).
9. Flutter Documentation: [Веб-сайт]. https://docs.flutter.dev/ (англійською мовою).
10. How to upload data to Firebase Firestore Cloud Database: [Веб-сайт]. https://medium.com/@devesu/how-to-upload-data-to-firebase-firestore-cloud-database-63543d7b34c5
11. Firebase: [Веб-сайт]. https://firebase.google.com/
12. Firebase Realtime database [Веб-сайт]. https://firebase.google.com/docs/database
13. Web Design [Веб-сайт]. https://webdesign.tutsplus.com.
14. Flutter FAQ [Веб-сайт]. https://flutter.dev/docs/resources/faq

ДОДАТОК А

Модулі main.dart, map_screen.dart, detail_screen.dart з кодом мовою Dart

```
import 'dart:ui';

import 'package:e_svitlo/%20info_screen/info_screen.dart';
import 'package:e_svitlo/dictionary/flutter_delegate.dart';
import 'package:e_svitlo/favorites_screen/favorites_screen.dart';
import 'package:e_svitlo/firebase_options.dart';
import 'package:e_svitlo/login_screen/login_screen.dart';
import 'package:e_svitlo/login_screen/models/login_cubit.dart';
import 'package:e_svitlo/map_screen/map_screen.dart';
import 'package:e_svitlo/map_screen/models/markers_cubit.dart';
import 'package:e_svitlo/my_branches/my_branches_screen.dart';
import 'package:e_svitlo/splash_screen/splash_screen.dart';
import 'package:e_svitlo/utils/bloc_observer.dart';
import 'package:e_svitlo/utils/crashlytics_util.dart';
import 'package:e_svitlo/utils/language_provider.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_crashlytics/firebase_crashlytics.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:provider/provider.dart';
import 'package:shared_preferences/shared_preferences.dart';

void main() async {
 WidgetsFlutterBinding.ensureInitialized();
 await Firebase.initializeApp(
    name: 'eSvitlo', options: DefaultFirebaseOptions.currentPlatform);
 final preference = await SharedPreferences.getInstance();
 late String lang;
 try {
  lang = preference.getString('language') ?? 'uk';
  FlutterError.onError = (errorDetails) {
   FirebaseCrashlytics.instance.recordFlutterFatalError(errorDetails);
  };
  PlatformDispatcher.instance.onError = (error, stack) {
   FirebaseCrashlytics.instance.recordError(error, stack, fatal: true);
   return true;
  };
  Bloc.observer = SimpleBlocObserver();
 } catch (e) {
  debugPrint(e.toString());
  CrashlyticsUtil.sendError(e);
  lang = 'uk';
 }
 runApp(
  MultiProvider(
   providers: [
    BlocProvider<LoginCubit>(
     create: (BuildContext context) => LoginCubit()..init(),
    ),
    BlocProvider<MarkersCubit>(
     create: (BuildContext context) => MarkersCubit()..getBranches(),
    ),
    ChangeNotifierProvider(
     create: (BuildContext context) =>
       LanguageProvider(preference)..changeLanguage(lang),
```

```
      ),
    ],
    child: MaterialApp(
      routes: {
        Routes.mapScreen: (_) => const MapScreen(),
        Routes.login: (_) => const LoginScreen(),
        Routes.myBranches: (_) => const MyBranchesScreen(),
        Routes.favorite: (_) => const FavoritesScreen(),
        Routes.info: (_) => const InfoScreen()
      },
      locale: const Locale('uk'),
      supportedLocales: FlutterDictionaryDelegate.getSupportedLocales,
      localizationsDelegates:
          FlutterDictionaryDelegate.getLocalizationDelegates,
      home: const SplashScreen(),
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primaryColor: const Color(0xFFEACC33),
        accentColor: const Color(0xFF575757),
        appBarTheme: const AppBarTheme(
          color: Color(0xFFEACC33),
        ),
        floatingActionButtonTheme: const FloatingActionButtonThemeData(
            backgroundColor: Colors.white,
            foregroundColor: Colors.black,
            iconSize: 40.0),
      ),
    ),
  ),
 );
}

class Routes {
 static const String mapScreen = 'map-screen';
 static const String login = 'login-screen';
 static const String myBranches = 'my-branches-screen';
 static const String favorite = 'favorite-screen';
 static const String info = 'info-screen';
}


import 'dart:io';

import 'package:e_svitlo/dictionary/flutter_dictionary.dart';
import 'package:e_svitlo/edit_screen/edit_screen.dart';
import 'package:e_svitlo/global_widgets/main_layout.dart';
import 'package:e_svitlo/login_screen/models/login_cubit.dart';
import 'package:e_svitlo/map_screen/models/branch_model.dart';
import 'package:e_svitlo/map_screen/models/markers_cubit.dart';
import 'package:e_svitlo/map_screen/widgets/branch_marker_widget.dart';
import 'package:e_svitlo/utils/language_provider.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_map/flutter_map.dart';
import 'package:latlong2/latlong.dart';
import 'package:provider/provider.dart';

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

 @override
 State<MapScreen> createState() => _MapScreenState();
}
```

```
class _MapScreenState extends State<MapScreen> {
 final MapController controller = MapController();
 bool chooseCoordinateMode = false;
 List<CircleMarker>? circleMarkers;
 Marker? myMarker;
 BranchModel? activeBranch;

 @override
 Widget build(BuildContext context) {
  context.read<MarkersCubit>().getBranches();
  context.read<LoginCubit>().update();
  final language = Provider.of<LanguageProvider>(context).language;
  return BlocBuilder<MarkersCubit, MarkersState>(
   builder: (context, state) {
    return MainLayout(
     current: ScreensEnum.map,
     activeBranchForInfo: activeBranch,
     selectCoordinateMode: chooseCoordinateMode,
     title: chooseCoordinateMode
       ? language.home.choosePlace
       : language.home.title,
     controller: controller,
     onTapAddBranch: () {
      setState(() {
       chooseCoordinateMode = !chooseCoordinateMode;
      });
     },
     getLocation: (location) {
      setState(() {
       circleMarkers = <CircleMarker>[
        CircleMarker(
          point: LatLng(location.latitude!, location.longitude!),
          color: Colors.blue.withOpacity(0.7),
          borderStrokeWidth: 2,
          useRadiusInMeter: true,
          radius: location.accuracy ?? 10),
       ];
       myMarker = Marker(
        point: LatLng(location.latitude!, location.longitude!),
        builder: (context) => const Icon(
         Icons.my_location,
         color: Colors.lightBlueAccent,
        ),
       );
      });
     },
     body: FlutterMap(
      mapController: controller,
      options: MapOptions(
       center: LatLng(46.482952, 30.712481),
       zoom: 12.0,
       maxZoom: 18.0,
       //  swPanBoundary: LatLng(46.320524, 30.532936),
       //  nePanBoundary: LatLng(46.604408, 30.905424),
       minZoom: 3,
       onTap: chooseCoordinateMode
         ? (position, latLng) {
          showQuestionDialog(context, () async {
           Navigator.of(context).pushReplacement(
            MaterialPageRoute(
             builder: (_) => BlocProvider.value(
              value: context.read<MarkersCubit>(),
              child: EditScreen(
```

```
                    newBranch: (branch) async {
                     await context
                        .read<MarkersCubit>()
                        .addBranch(
                          branch,
                          (branchId) => context
                              .read<LoginCubit>()
                              .addBranch(branchId),
                        );
                     context.read<MarkersCubit>().getBranches();
                    },
                    latLng: latLng,
                   ),
                  ),
                 ),
                );
               });
              }
             : (position, latLng) {
               setState(() {
                activeBranch = null;
               });
              }),
        children: [
         TileLayer(
          urlTemplate: 'https://tile.openstreetmap.org/{z}/{x}/{y}.png',
          userAgentPackageName: 'dev.fleaflet.flutter_map.example',
         ),
         if (state is MarkersSuccess && !chooseCoordinateMode)
          MarkerLayer(
           markers: [
            ...state.branchesSorted
              .map(
               (e) => Marker(
                point: e.latLng,
                builder: (context) => BranchMarkerWidget(
                  branchModel: e,
                  onTap: () {
                   setState(() {
                    activeBranch = e;
                   });
                  }),
                width: 60.0,
                height: 120.0,
               ),
              )
              .toList(),
            if (myMarker != null) myMarker!
           ],
          ),
         if (circleMarkers != null)
          CircleLayer(
           circles: circleMarkers!,
          )
        ],
       ),
      );
     },
    );
}

void showQuestionDialog(BuildContext context, Function() onYes) {
 final language = FlutterDictionary.instance.language.popUp;
 if (Platform.isIOS) {
```

```
    showCupertinoModalPopup(
      context: context,
      builder: (context) => CupertinoAlertDialog(
       title: Text(language.here),
       actions: [
        CupertinoDialogAction(
         onPressed: Navigator.of(context).pop,
         child: Text(language.no),
        ),
        CupertinoDialogAction(
         onPressed: () {
          onYes();
         },
         child: Text(language.yes),
        ),
       ],
      ),
    );
   } else {
    showDialog(
      context: context,
      builder: (context) => AlertDialog(
       title: Text(language.here),
       actions: [
        InkWell(
         onTap: Navigator.of(context).pop,
         child: Text(language.no),
        ),
        InkWell(
         onTap: () => onYes(),
         child: Text(language.yes),
        ),
       ],
      ),
    );
   }
  }
}
```

```
import 'package:e_svitlo/dictionary/flutter_dictionary.dart';
import 'package:e_svitlo/login_screen/models/login_cubit.dart';
import 'package:e_svitlo/map_screen/models/branch_model.dart';
import 'package:e_svitlo/map_screen/models/markers_cubit.dart';
import 'package:e_svitlo/utils/item_type_util.dart';
import 'package:flutter/material.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:map_launcher/map_launcher.dart';
import 'package:provider/provider.dart';
import 'package:url_launcher/url_launcher.dart';

class DetailScreen extends StatelessWidget {
  DetailScreen({Key? key, required this.branch}) : super(key: key);
  final BranchModel branch;
  final language = FlutterDictionary.instance.language;

  @override
  Widget build(BuildContext context) {
    final ThemeData theme = Theme.of(context);
    final cubitLogin = Provider.of<LoginCubit>(context);
    final markerCubit = context.read<MarkersCubit>();
    final String to =
        '${language.popUp.to} ${_zeroAdder(branch.to!.hour)}:${_zeroAdder(branch.to!.minute)}';
```

```
final String from =
    '${language.popUp.from} ${_zeroAdder(branch.from!.hour)}:${_zeroAdder(branch.from!.minute)}';
return Scaffold(
  body: CustomScrollView(
    slivers: [
      SliverAppBar(
        pinned: true,
        elevation: 4.0,
        expandedHeight: 256.0,
        backgroundColor: branch.electroStatus
            ? Theme.of(context).primaryColor
            : Theme.of(context).accentColor,
        flexibleSpace: SafeArea(
          child: FlexibleSpaceBar(
            titlePadding: const EdgeInsets.only(top: 8.0),
            title: Align(
              alignment: Alignment.topCenter,
              child: Text(
                branch.electroStatus
                    ? 'ESvitlo!'
                    : language.detail.noElectro,
                style: Theme.of(context).textTheme.headline4?.copyWith(
                      fontWeight: FontWeight.bold,
                      color: Colors.black,
                    ),
              ),
            ),
            background: Align(
              alignment: Alignment.center,
              child: Padding(
                padding: const EdgeInsets.only(bottom: 8.0),
                child: Image.asset(
                  branch.electroStatus
                      ? 'assets/png/happy_motylok.png'
                      : 'assets/png/sad_motilok.png',
                ),
              ),
            ),
            expandedTitleScale: 1,
            collapseMode: CollapseMode.none,
          ),
        ),
      ),
      SliverList(
        delegate: SliverChildListDelegate(
          [
            Padding(
              padding: const EdgeInsets.all(33.0),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: <Widget>[
                  Text(
                    branch.name,
                    style:
                        Theme.of(context).textTheme.headline4?.copyWith(
                              fontWeight: FontWeight.bold,
                              decoration: TextDecoration.underline,
                              color: Colors.black,
                            ),
                  ),
                  Padding(
                    padding: const EdgeInsets.all(12.0),
                    child: Column(
                      children: [
                        _getRow(
                          trailing: SizedBox(
                            width: 25.0,
                            height: 25.0,
```

```
          child: iconTypeUtil(branch.type),
        ),
        (markerCubit.state as MarkersSuccess)
            .types[branch.type],
        context,
      ),
      const SizedBox(height: 14.0),
      _getRow(
        icon: Icons.location_on_rounded,
        branch.address,
        context,
      ),
      const SizedBox(height: 14.0),
      _getRow(
        icon: Icons.access_time_rounded,
        '$from $to',
        context,
      ),
    ],
  ),
),
if (branch.menu != null || branch.urlMenu != null)
  Text(
    language.detail.menu,
    style: theme.textTheme.titleLarge?.copyWith(
      fontWeight: FontWeight.w700,
      fontSize: 24.0,
    ),
  ),
if (branch.menu != null && branch.menu!.isNotEmpty)
  Padding(
    padding: const EdgeInsets.all(8.0),
    child: SizedBox(
      height: branch.menu!.length * 40.0,
      child: ListView.builder(
        physics: const NeverScrollableScrollPhysics(),
        itemBuilder: (ctx, index) {
          return Row(
            mainAxisAlignment:
                MainAxisAlignment.spaceBetween,
            children: [
              Text(
                branch.menu!.keys.toList()[index],
                style: theme.textTheme.titleMedium
                    ?.copyWith(
                        fontWeight: FontWeight.w500,
                        fontSize: 12.0),
              ),
              Text(
                branch.menu!.values.toList()[index],
                style: theme.textTheme.titleMedium
                    ?.copyWith(
                        fontWeight: FontWeight.w500,
                        fontSize: 12.0),
              ),
            ],
          );
        },
        itemCount: branch.menu!.length,
      ),
    ),
  ),
if (branch.urlMenu != null)
  Padding(
    padding: const EdgeInsets.all(16.0),
    child: Center(
      child: Image.network(
        branch.urlMenu!,
```

```
                    errorBuilder: (_, __, ___) =>
                        const Text('oops'),
                    frameBuilder: (_, __, ___, ____) =>
                        const CircularProgressIndicator.adaptive(),
                  ),
                ),
              )
            ],
          ),
        ),
      ),
    ],
  ),
  bottomNavigationBar: BottomNavigationBar(
    items: [
      if (cubitLogin.state is LoginSuccess)
        BottomNavigationBarItem(
          label: '',
          icon: IconButton(
            onPressed: () => cubitLogin.changeFavorite(branch.id),
            icon: Icon(
              (cubitLogin.state as LoginSuccess)
                      .model
                      .favoritesId
                      .contains(branch.id)
                  ? Icons.favorite_rounded
                  : Icons.favorite_outline_outlined,
            ),
          ),
        ),
      BottomNavigationBarItem(
        label: '',
        icon: GestureDetector(
          onTap: () => _openMapsSheet(context),
          child: const Icon(Icons.directions_walk_rounded),
        ),
      ),
      BottomNavigationBarItem(
        label: '',
        icon: GestureDetector(
          onTap: _call,
          child: const Icon(Icons.phone),
        ),
      ),
    ],
  ),
 );
}

Widget _getRow(
  String text,
  BuildContext context, {
  IconData? icon,
  Widget? trailng,
}) {
  return Row(
    children: <Widget>[
      trailng ?? Icon(icon!),
      const SizedBox(width: 12.0),
      Text(
        text,
        style: Theme.of(context).textTheme.bodyLarge,
      ),
    ],
  );
}
```

```
  void _openMapsSheet(context) async {
    try {
      final Coords coords =
          Coords(branch.latLng.latitude, branch.latLng.longitude);
      final title = branch.name;
      final availableMaps = await MapLauncher.installedMaps;

      showModalBottomSheet(
        context: context,
        builder: (BuildContext context) {
          return SafeArea(
            child: SingleChildScrollView(
              child: Wrap(
                children: <Widget>[
                  for (var map in availableMaps)
                    ListTile(
                      onTap: () => map.showMarker(
                        coords: coords,
                        title: title,
                      ),
                      title: Text(map.mapName),
                      leading: SvgPicture.asset(
                        map.icon,
                        height: 30.0,
                        width: 30.0,
                      ),
                    ),
                ],
              ),
            ),
          );
        },
      );
    } catch (e) {
      debugPrint(e.toString());
    }
  }

  Future<void> _call() async {
    await launchUrl(Uri(scheme: 'tel', path: branch.phone));
  }

  String _zeroAdder(int val) {
    if (val < 10) return '0$val';
    return val.toString();
  }
}
```

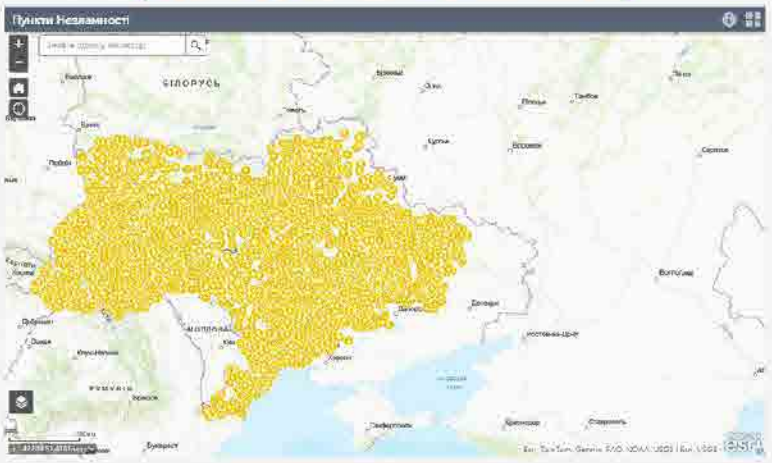
ДОДАТОК Б

Слайди мультимедійної презентації

Розробка геоінформаційного мобільного додатку для ідентифікації пунктів незламності

Розробник: Вєлков І.В, ОТФК ОНТУ Керівник роботи: к.т.н. Іванова Л.В., ОТФК ОНТУ

Розташування офіційних пунктів незламності в Україні (веб-додаток Nezlamnist)

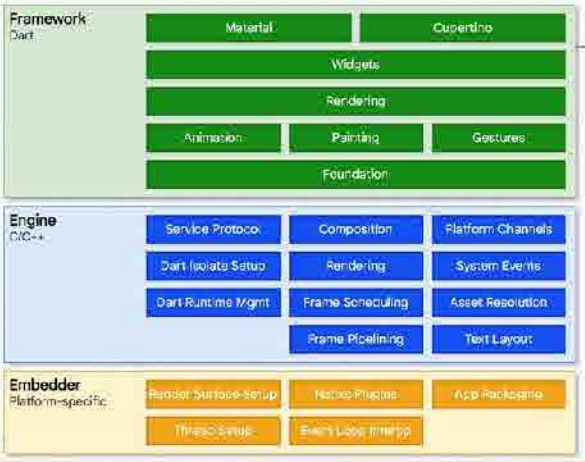


Адреси Пунктів Незламності доступні в офіційному чат-боті в Telegram та viber

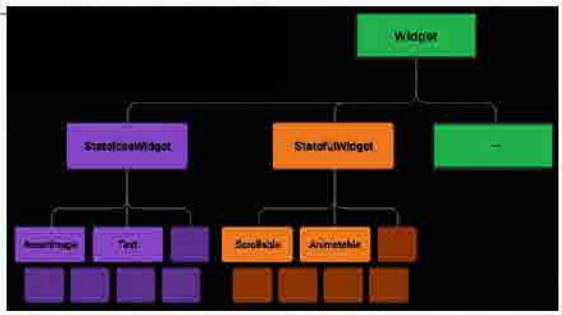
Порівняння крос-платформних фреймворків

	React Native	Flutter	Xamarin	Kotlin Multiplatform Mobile
Мова програмування	JavaScript та React	Dart	C# та .Net	Kotlin
Перший реліз	2015	2017	2011	2017
Розробник платформи	Facebook та спільнота	Google та спільнота	Microsoft	JetBrains
Спільнота	Невелика	Достатня спільнота, активний розвиток	Мала спільнота	Велика
Відкритий вихідний код	Так	Так	Так	Ні
Інструменти побудови UI	Компоненти Native та декларативний UI	Віджети та декларативний UI	Xamarin.Forms	Платформний UI
Продуктивність роботи	Висока	Висока	Середня через Xamarin.Forms	Висока

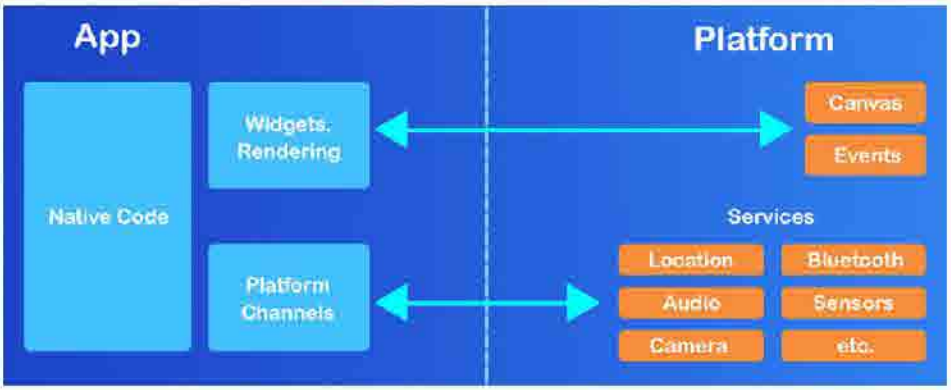
Компоненти фреймворку Flutter



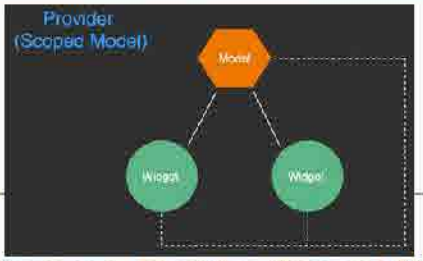
Ієрархія класів Flutter



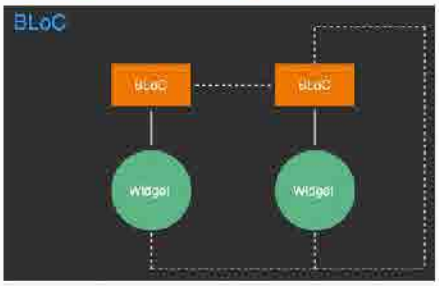
Взаємодія додатку користувача з платформою Flutter



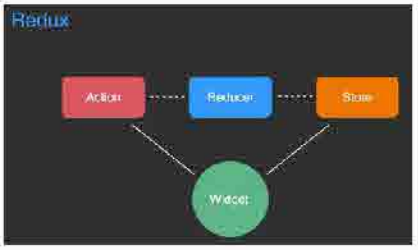
Stateful Widget у Flutter



Тип архітектури Provider (Scoped Model)



Тип архітектури BLoC

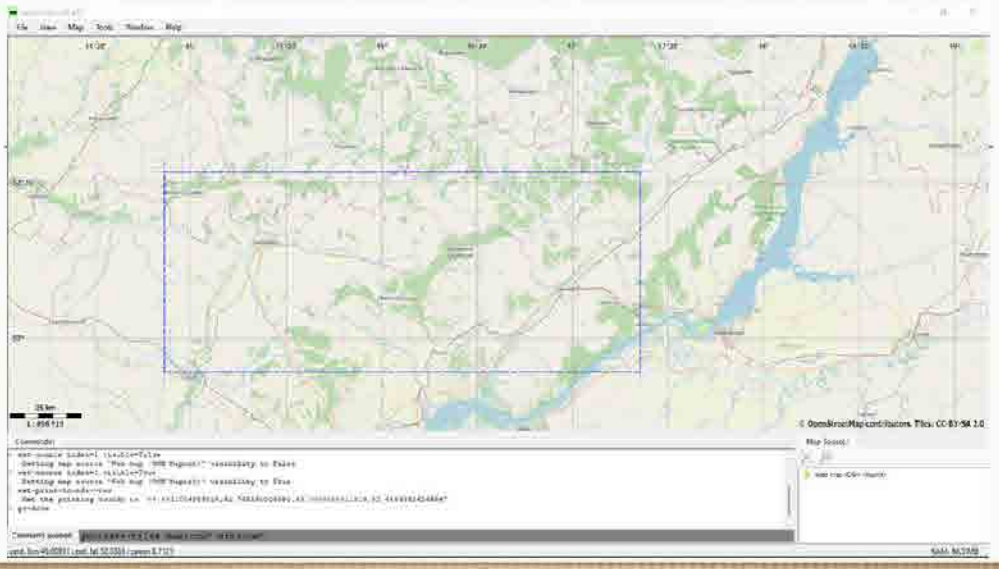


Тип архітектури Redux

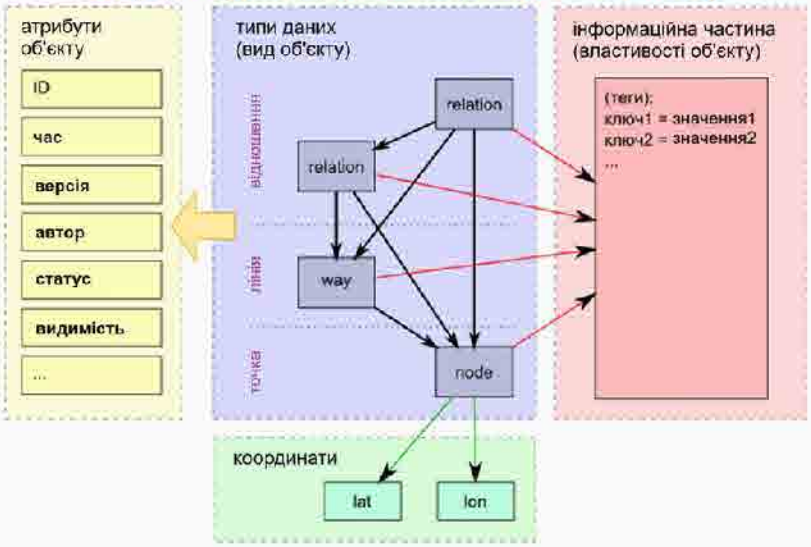
Порівняння геоінформаційних сервісів

	MapKit	GoogleMaps	MapBox	OpenStreetMap
Нативний фреймворк	Так	Ні	Ні	Ні
Відкритий вихідний код	Ні	Ні	Так, частинами	Так
Розробник платформи	Apple	Google	MapBox	Стів Кост та вільні розробники
Стилізація карти	Ні	Так	Так	Так
Кешування карти	Немає, потрібно розробляти самому	Немає, не можливо розробити через умови використання	Існує	Існує
Побудова маршрутів офлайн	Ні	Ні	Так	Так

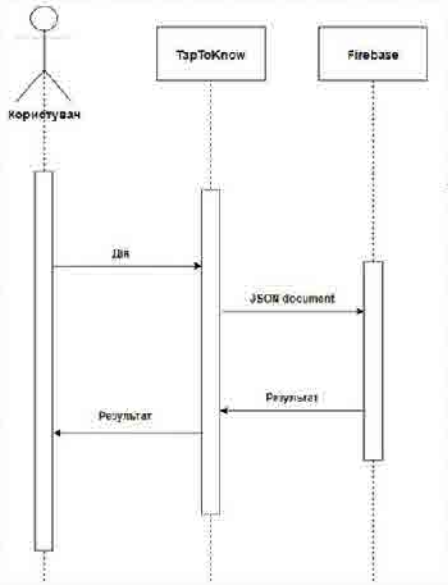
Застосування сервісу OpenStreetMap



Структура даних OpenStreetMap

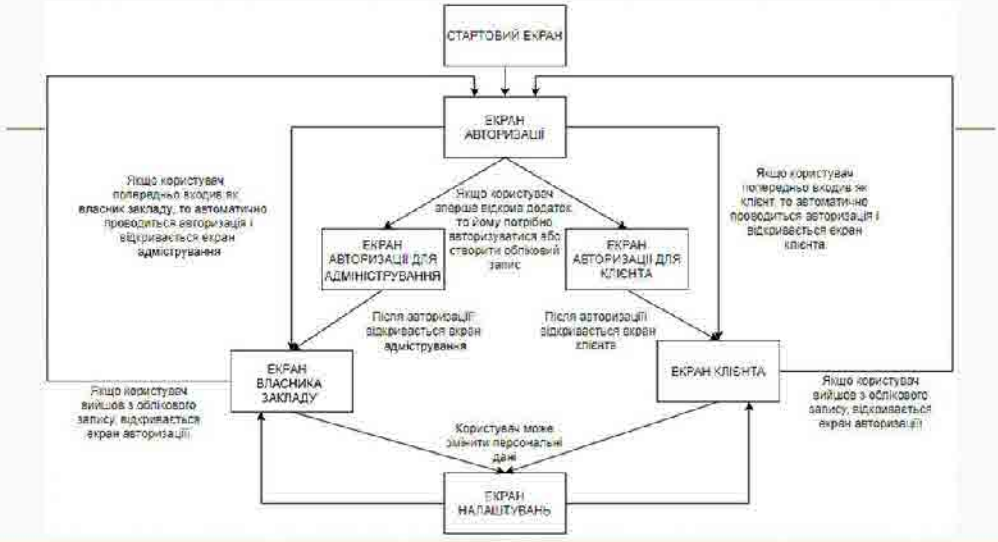


Узагальнена схема варіантів використання мобільного додатку



Діаграма послідовності взаємодії користувача з додатком

Узагальнена структурна схема мобільного додатку



Архітектурна модель API додатку

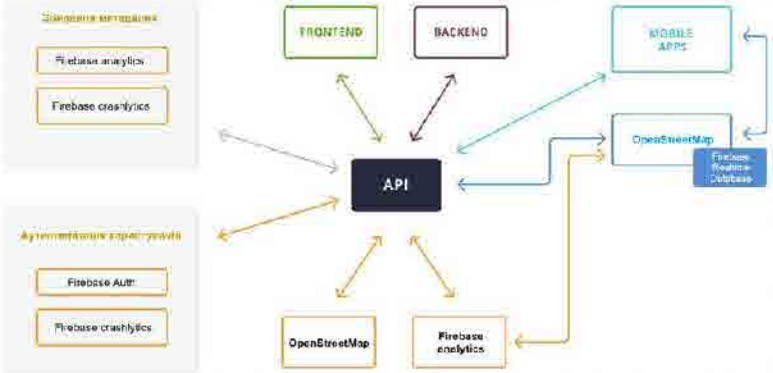
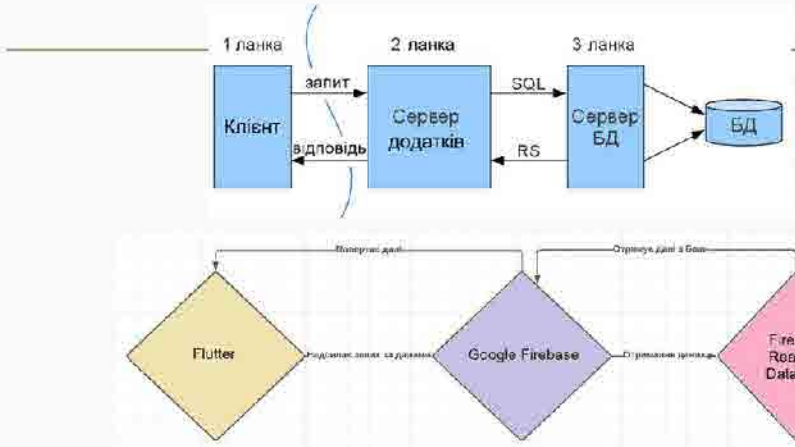
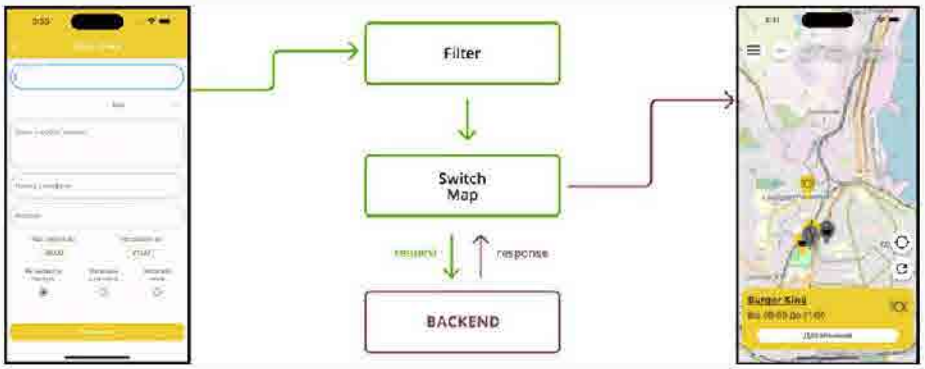


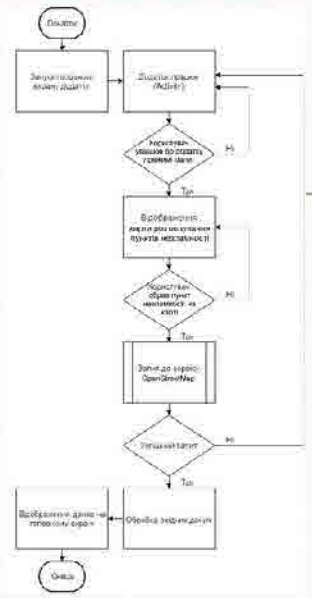
Схема клієнт-серверної взаємодії мобільного додатку



Загальна схема процесу фільтрації геоданих у додатку



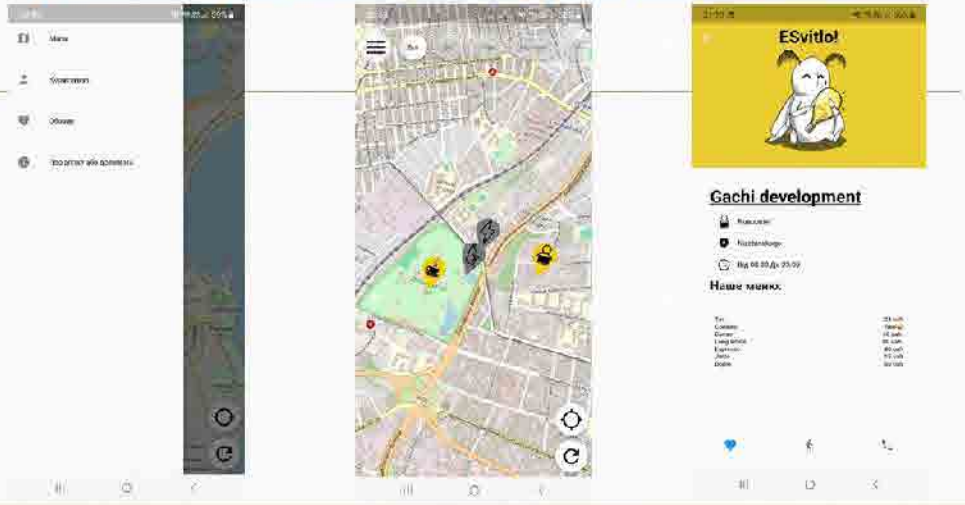
БСА обробки дій користувача у додатку



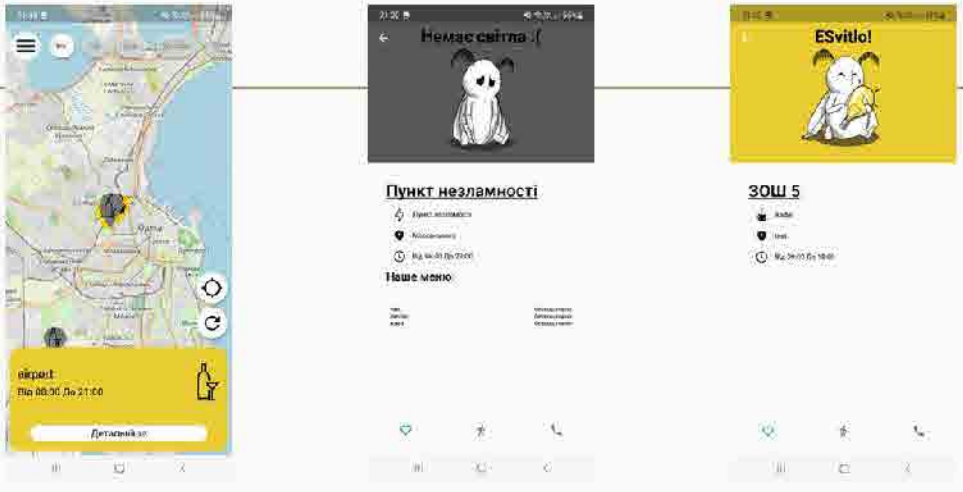
Екран входу та реєстрації у мобільному додатку. Екран виходу з акаунту у мобільному додатку

Екран додавання нового громадського закладу на карту у режимі адміністратора

Меню на головному екрані додатку.
Екран мапи з позначенням пунктів незламності додатку.
Додавання громадського закладу до обраного у додатку



Панель для виклику інформації про громадський заклад у додатку.
Екран перегляду детальної інформації про заклад у додатку



Відокремлений структурний підрозділ

Одеський технічний фаховий коледж ОНТУ

ВІДГУК

Керівника про кваліфікаційну роботу бакалавра

Вєлкова Івана Васильовича

(прізвище, ім`я та по батькові)

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Спеціальність *123 «Комп'ютерна інженерія»*

Тема кваліфікаційної роботи

«Розробка геоінформаційного мобільного додатку для ідентифікації пунктів незламності»

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) Обсяг і якість виконання роботи (розрахунково-пояснювальної записки)

Пояснювальна записка виконана якісно, у достатньому обсязі, відповідно до індивідуального завдання та теми дипломного проекту, розділи пояснювальної записки відповідають етапам рішення завдання, поставленого у дипломному проекті

Презентація виконана якісно, у достатньому обсязі. Презентація наочно демонструє результати роботи.

б) Самостійність роботи над кваліфікаційною роботою

Студент самостійно обрала напрям та тематику кваліфікаційної роботи. Провів аналіз існуючих рішень і зробив необхідні висновки для реалізації проекту. Виявив навички самостійно опрацьовувати новий матеріал та виконувати пошук необхідної літератури та інших джерел інформації

в) Теоретична підготовка бакалавра

відповідає вимогам, що надаються до бакалавра зі спеціальності «Комп'ютерна інженерія»

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва

У дипломному проекті розглянута та реалізована сучасна тема створення мобільного додатку для інформаційної підтримки щодо наявних пунктів незламності Одеського регіону описані принципи розробки мобільного додатку у сучасному середовищі програмування та процес підбору інструментів та технологій для розробки графічного дизайну та ергономіки . Обґрунтовано вибір програмних засобів для реалізації програмного забезпечення, проведено його тестування

Загальна оцінка 5(відмінно)

Прізвище, ім`я, по батькові Іванова Лілія Вікторівна

Місто роботи і посада керівника проекту_ВСП «Одеський технічний фаховий коледж ОНТУ» к.т.н., зав. кафедрою Комп'ютерної інженерії

Підпис

«____» 2024р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу здобувача (здобувачки) освіти

відділення комп'ютерних систем

Вєлкова Івана Васильовича

(прізвище, ім`я та по батькові)

Спеціальність 123 Комп'ютерна інженерія

Освітня програма Комп'ютерна інженерія

Керівник кваліфікаційної роботи Іванова Лілія Вікторівна

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи *«Розробка геоінформаційного мобільного додатку для ідентифікації пунктів незламності»*

Обсяг розрахунково-пояснювальної записки 81 сторінок

Обсяг графічної (презентаційної) частини 20 аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаного кваліфікаційної роботи завданню

кваліфікаційна робота у повному обсязі відповідає темі та завданню

б) характеристика виконання кожного розділу кваліфікаційної роботи

Кваліфікаційна робота складається з розділів: Передмова. Аналітичний огляд існуючих рішень. Обгрунтування вибору технологій для створення геоінформаційних систем та мобільних додатків. Архітектура та компоненти мобільного додатку. Розробка елементів додатку та інтерфейсу користувача. Реалізація додатку та його тестування. Розділ охорони праці. Висновок. Перелік використаних джерел інформації. Кожен розділ присвячено одному з етапів виконання завдання кваліфікаційної роботи та містить необхідну інформацію щодо результатів виконаної роботи.

в) оцінка якості виконання пояснювальної записки та графічної частини кваліфікаційної роботи

Пояснювальна записка виконана якісно, у достатньому обсязі, відповідно до індивідуального завдання та теми дипломного проекту, розділи пояснювальної записки відповідають етапам рішення завдання, поставленого у кваліфікаційній роботі. Презентація виконана якісно, у достатньому обсязі. Презентація наочно демонструє результати роботи.

г) перелік позитивних якостей кваліфікаційної роботи

1. Актуальна тематика
2. Сучасні технології реалізації програмного продукту
3. Якісне подання результатів роботи

д) основні недоліки кваліфікаційної роботи

1. Можливо варту було б зробити інтерфейс застосунку більш зручним для користувача.
2. Варто було б передбачити контроль наявності електроживлення також за допомогою бази даних відключень ДТЕК.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові рецензента к.т.н. Селіванова Алла Віталіївна

Місце роботи і посада рецензента Одеський національний технологічний університет, декан факультету комп'ютерної інженерії, програмування та кіберзахисту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ ФАКУЛЬТЕТ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ, ПРОГРАМУВАННЯ ТА КІБЕРЗАХИСТУ

Підпис:

«17» червня 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григорівна Краснокутська

ID перевірки:
1016269382

Дата перевірки:
21.05.2024 18:11:16 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
21.05.2024 20:39:14 EEST

ID користувача:
100011688

Назва документа: **2БКС-28_Іван Вєлков**

Кількість сторінок: **62** Кількість слів: **11381** Кількість символів: **84206** Розмір файлу: **3.70 MB** ID файлу: **1016060252**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

31.5%
Схожість

Найбільша схожість: **9.16%** з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/58551/1/Kuzmenko_bakalavr.p...

31.5% Джерела з Інтернету | **158** ..... Сторінка 64

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Підозріле форматування | **13 сторінок**

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Вєлков Іван Васильович,
здобувач освіти гр. 2БКС-28, та

Іванова Лілія Вікторівна,
керівник випускної роботи,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної роботи бакалавра на тему:

«Розробка геоінформаційного мобільного додатку для ідентифікації пунктів незламності» (автор роботи – Вєлков І.І., керівник роботи – Іванова Л.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Вєлков І.В. /

Керівник _____ / Іванова Л.В. /

«13» червня 2024 р.