МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ПОЛТАВСЬКИЙ ПОЛІТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Циклова комісія дисциплін програмної інженерії

ЗВІТ

з технологічної практики

«Розробка і супроводження програмного продукту»

на тему Мобільний додаток для підтримки водіїв під час відряджень

Виконав: здобувач освіти 4 курсу,
групи _____ 45
спеціальності 121
Інженерія програмного забезпечення
Любченка О.В.
(прізвище та ініціали)

Керівник Кривцова О. П.
(підпис) (прізвище та ініціали)

Полтава – 2024

ЗМІСТ

ВСТУП 3

1. ПОСТАНОВКА ЗАВДАННЯ 5

1.1. Основні вимоги до продукту 6

1.2. Вимоги до інтерфейсу 7

2. ПЛАНУВАННЯ СИСТЕМИ 9

2.1. Архітектура системи 10

2.2. Тестування 10

2.3. Інструкція з використання системи 11

ВИСНОВКИ 12

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 14

ДОДАТОК А. ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ПРОГРАМНИХ ПРОДУКТІВ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ 15

ДОДАТОК Б. UML ДІАГРАМА ПРЕЦЕДЕНТІВ 16

ДОДАТОК В. ПРОТОТИП ІНТЕРФЕЙСУ 17

ДОДАТОК Г. UML ДІАГРАМА КЛАСІВ 18

ДОДАТОК Ґ. ER ДІАГРАМА 19

ДОДАТОК Д. ВИХІДНІ КОДИ 20

ДОДАТОК Е. РЕЗУЛЬТАТИ ТЕСТУВАННЯ 23

ДОДАТОК Є. ЗНІМКИ ЕКРАНУ 25

ВСТУП

Місцем моєї технологічної практики була компанія Акціонерне товариство «Укртранснафта».

Компанія є єдиним оператором системи магістральних нафтопроводів України з 2001 року, що надає послуги з транспортування нафти трубопровідним транспортом на нафтопереробні підприємства (НПЗ) України та транзитом до країн Східної та Центральної Європи, а також нафтопродуктів з використанням української ділянки магістрального нафтопродуктопроводу «Самара-Західний напрямок» як в прямому, так і реверсному напрямку.

Основним напрямком її діяльності є оперування нафтотранспортної системи України, яка включає 18 нафтопроводів загальною протяжністю 4569 км, 51 нафтоперекачувальну станцію (НПС), 11 резервуарних парків загальною місткістю 1010 тисяч кубометрів. Роботу НПС забезпечують 176 насосних агрегатів загальною потужністю електродвигунів 356,9 тисяч кВт.

В ході практики я ознайомився з роботою відділу аналітики комп’ютерних систем цеху електрозв’язку «Глинсько-Розбишівська» ЛВДС «Глинсько-Розбишівська», де переді мною було поставлено завдання розробити мобільний застосунок для допомоги водіям підприємства у пошуку інфраструктури для них та автомобілів під час відрядження.

Для виконання поставленого завдання мені стали в нагоді знання, вміння й навички з дисциплін «Автоматизація інформаційних систем» та «Бази даних».

Під час практики я працював із низкою сучасних інструментів та технологій, які суттєво розширили мої знання та навички в галузі розробки програмного забезпечення.

Серед використаних інструментів була потужна і зручна IDE Microsoft Visual Studio Code. Цей інструмент забезпечує ефективне середовище

розробки завдяки своїй гнучкості, підтримці багатьох мов програмування та широкому вибору плагінів, які оптимізують робочий процес.

Також я освоїв Google Firebase Command Line Interface (CLI), що дозволило мені працювати з хмарними сервісами Firebase безпосередньо через командний рядок. Це значно пришвидшує процес розробки додатків.

Для управління ресурсами в хмарному середовищі я використовував Google Cloud Console, яка надає інтуїтивно зрозумілий інтерфейс для адміністрування проєктів та відстеження використання ресурсів. Google Maps Platform допомогла мені інтегрувати інтерактивні карти та геолокаційні сервіси у додаток, що значно підвищило його функціональність.

Під час практики я також працював із Figma – сучасним інструментом для дизайну інтерфейсів. Завдяки Figma я створив прототипи додатків, які дозволили наочно уявити кінцевий продукт ще до початку кодування.

Особливу увагу я приділив вивченню мови програмування Dart, яка є основою для розробки на Flutter. Ця мова вирізняється простотою та високою продуктивністю, що дозволяє створювати швидкі та ефективні додатки.

Фреймворк Flutter став ключовим елементом моєї практики. Завдяки Flutter я зміг розробляти кросплатформенні додатки з багатим функціоналом і сучасним дизайном. Цей фреймворк забезпечує високу швидкість розробки та дозволяє створювати додатки для Windows та Android з єдиною базою коду.

Я також вивчив роботу з базою даних Firebase та її компонентами, такими як Firebase Firestore, Firebase Authentication та Firebase Realtime Database. Firestore забезпечує зручне зберігання даних із можливістю масштабування, Firebase Authentication – простий спосіб реалізації автентифікації користувачів, а Realtime Database дозволяє обмінюватися даними в реальному часі, що є важливим для створення інтерактивних додатків.

1. ПОСТАНОВКА ЗАВДАННЯ

Описаний проєкт має стати корисним інструментом для водіїв підприємства, які перебувають у відрядженнях. Мета програми – забезпечити зручний доступ до важливої інфраструктури, такої як автомобільні заправки, готелі та технічні сервіси, з якими організація має укладені домовленості. Цей додаток стане ефективним засобом оптимізації витрат часу та ресурсів під час пересування водіїв.

Розробка проєкту передбачає створення двох ключових компонентів: клієнтського застосунку для операційної системи Android та десктопного застосунку для Windows. Клієнтський застосунок надає користувачам можливість переглядати мітки інфраструктурних об’єктів на мапі. Використання інтеграції з Google Maps дозволяє водіям швидко отримувати додаткову інформацію про розташування об’єкта, а також прокладати маршрути до обраної локації. Зручність такого функціоналу забезпечує оптимізацію роботи, підвищуючи ефективність та точність пересування.

Десктопний застосунок виконує іншу важливу функцію – він слугує інструментом управління базою даних. Завдяки йому адміністратори можуть контролювати інформацію про об’єкти інфраструктури, вносити необхідні зміни, оновлювати дані та слідкувати за актуальністю інформації, що відображається у клієнтському застосунку. Це сприяє впевненості у тому, що водії завжди отримують найсвіжішу та найбільш достовірну інформацію.

Середовище розробки для створення обох компонентів обрано Visual Studio Code, а мовою програмування – Dart. Такий вибір пояснюється зручністю роботи у Visual Studio Code, який є потужним інструментом для написання коду, та перевагами Dart, що забезпечує високу продуктивність і простоту створення міжплатформних рішень.

Кінцевий термін розробки збігається з завершенням навчальної практики, що визначає досить обмежені часові рамки. Однак це також стимулює ефективне використання часу та ресурсів, підвищуючи продуктивність роботи над проєктом. Завершення проєкту передбачає

передачу вихідного коду організації для подальшого розвитку та вдосконалення системи. Це дозволить підприємству адаптувати застосунок до нових потреб або додати функціональність, яка може виникнути в майбутньому.

Фінальна версія мобільного застосунку має стати незамінним компаньйоном для водіїв, допомагаючи їм у виконанні завдань, підвищуючи ефективність їхньої роботи та забезпечуючи зручний доступ до важливої інформації в дорозі. Інтуїтивно зрозумілий інтерфейс, актуальність даних і функціональна зручність стануть основними перевагами цього програмного забезпечення, що сприятиме його успішному впровадженню в практику діяльності підприємства.

1.1. Основні вимоги до продукту

ФУНКЦІОНАЛЬНІ ВИМОГИ:

- Відтворення інфраструктурних точок на інтерактивній карті;
- опція перегляду докладної інформації про обраний об’єкт;
- інтеграція з Google Maps для доступу до детальної інформації про локацію за її координатами;
- можливість прокладення маршруту до потрібного об’єкта через Google Maps;
- реалізація системи користувацьких профілів для захисту даних і обмеження доступу;
- функціональність для спілкування з адміністратором у разі виникнення критичних ситуацій.

НЕФУНКЦІОНАЛЬНІ ВИМОГИ:

- Використання кросплатформного фреймворку Flutter.

ПЕРЕЛІК РЕАЛІЗОВАНИХ ЗА ЧАС ПРАКТИКИ ФУНКЦІОНАЛЬНИХ ВИМОГ

- Відтворення інфраструктурних точок на інтерактивній карті;
- опція перегляду докладної інформації про обраний об’єкт;

— інтеграція з Google Maps для доступу до детальної інформації про локацію за її координатами;
— можливість прокладення маршруту до потрібного об’єкта через Google Maps;
— реалізація системи користувацьких профілів для захисту даних і обмеження доступу.

ПЕРЕЛІК НЕРЕАЛІЗОВАНОЇ ФУНКЦІОНАЛЬНОСТІ

— Функціональність для спілкування з адміністратором у разі виникнення критичних ситуацій. Причина: надмірна складність створення функціональності для відтворення push-сповіщень, що є ключовим для забезпечення безпеки водіїв.

1.2. Вимоги до інтерфейсу

Інтерфейс мобільного клієнта, розробленого для Android, має повністю відповідати стандарту Material Design. Цей стандарт був обраний через його широке розповсюдження та рекомендації Google для створення зручних, інтуїтивних і гармонійних інтерфейсів. Візуальний стиль Material Design передбачає використання чітких ліній, м'яких анімацій і динамічних ефектів, які забезпечують естетично привабливий і функціональний вигляд програми.

Для платформи Windows інтерфейс реалізується відповідно до принципів Fluent Design. Fluent Design System пропонує користувачам візуальний стиль, який акцентує увагу на прозорості, світлі, глибоких відтінках та анімації. Це дозволяє створювати сучасний інтерфейс, що виглядає природно і гармонійно в екосистемі Windows, забезпечуючи безперебійну інтеграцію програми з операційною системою.

Для досягнення цих цілей був складений перелік ключових вимог до зовнішнього вигляду програми. Було створено прототипи інтерфейсів із використанням сучасного інструменту Figma, що дозволяє ефективно моделювати основні елементи інтерфейсу та візуалізувати ідеї.

Під час розробки особливу увагу було приділено рішенням, які забезпечують зручність користування та покращують загальний досвід користувача. Серед таких рішень варто виділити оптимальне розташування елементів керування, зручну навігацію між екранами, адаптивний дизайн для різних розмірів екранів і створення інтуїтивно зрозумілих підказок. Усі ці аспекти спрямовані на підвищення ефективності та приємності взаємодії користувача з програмою.

Зовнішній вигляд програми характеризується сучасністю і функціональністю. Головний екран містить ключові елементи керування, розташовані таким чином, щоб забезпечити легкий доступ до основних функцій. Деталі інтерфейсу, включаючи макети вікон та розташування елементів, представлені у додатку В у вигляді скріншотів, що супроводжуються посиланнями на відповідні зображення.

2. ПЛАНУВАННЯ СИСТЕМИ

Для реалізації проєкту була обрана каскадна модель життєвого циклу. Ця модель передбачає поетапне виконання всіх процесів розробки у суворо визначеній послідовності, що дозволяє мінімізувати ризики і забезпечити структурований підхід до роботи. Каскадна модель включає кілька ключових етапів, кожен з яких завершується перед початком наступного, що дозволяє створювати детальну документацію та підтримувати чіткий контроль над процесом розробки.

Першим етапом було узгодження вимог. На цьому етапі визначались цілі проєкту, потреби користувачів і технічні вимоги до системи. Усі ці аспекти були зафіксовані у формі технічного завдання, яке стало основою для подальшої роботи.

Далі відбувалося проєктування. Цей етап включав розробку архітектури системи, створення прототипів інтерфейсу та деталізацію функціональних компонентів. Завдяки ретельному проєктуванню вдалося уникнути змін під час наступних етапів.

Етап розробки передбачав безпосереднє написання коду відповідно до створеного дизайну та технічного завдання. Для контролю версій коду використовувався GitHub. Крім того, це сприяло збереженню змін і спрощенню інтеграції компонентів.

Після завершення розробки виконувалося тестування. На цьому етапі система перевірялася на наявність помилок, тестувалися функціональні можливості, стабільність роботи та відповідність вимогам. Тестування дозволило забезпечити високий рівень якості програмного забезпечення перед його фінальною передачею.

Останній етап – передача вихідних кодів. Уся робота була систематизована, і підприємство отримало кінцевий продукт разом із супровідною документацією, що включала інструкції з використання.

2.1. Архітектура системи

Архітектура проекту побудована на основі Flutter, що дозволяє створювати кросплатформні мобільні додатки. Основні компоненти включають Firebase для аутентифікації та зберігання даних, Google Maps для відображення карт, Geolocator для визначення поточного місцезнаходження користувача, та URL Launcher для відкриття зовнішніх посилань.

Проект використовує патерн "Provider" для управління станом, що дозволяє легко передавати дані між різними частинами додатку. Крім того, використовується патерн "Factory" для створення об'єктів моделі з даних, отриманих з Firestore.

Система складається з наступних підсистем і компонентів:

— Аутентифікація: Використовує Firebase Auth для входу користувачів;
— відображення карт: Використовує Google Maps для відображення карт та маркерів;
— управління маркерами: Використовує Firestore для зберігання та завантаження маркерів;
— визначення місцезнаходження: Використовує Geolocator для отримання поточного місцезнаходження користувача;
— інтерфейс користувача: Використовує Flutter для побудови UI, включаючи екрани аутентифікації та карти.

Ця архітектура забезпечує модульність, що дозволяє легко додавати нові функції та підтримувати існуючі.

2.2. Тестування

У процесі розробки програмного забезпечення було виконано unit тестування, що є важливим етапом забезпечення якості коду. Для проведення тестів використовувалися пакети test та flutter_test, які входять до екосистеми Flutter і дозволяють ефективно тестувати функціональність окремих модулів застосунку.

Unit тестування охопило ключові модулі програми, такі як:

— Модуль роботи з маркерами: У цьому модулі було перевірено коректність створення, збереження та відображення маркерів на мапі. Особливу увагу приділили перевірці відповідності даних, що зберігаються у маркерах, їхньому подальшому використанню в інтерфейсі користувача;

— модуль автентифікації: Було протестовано основні сценарії, включаючи, вхід у систему та вихід з неї;

— модуль мапи: Тестування цього модуля охопило функції взаємодії з мапою, включаючи масштабування, переміщення та інтеграцію з маркерами. Тестувалося, як мапа обробляє події користувача та чи правильно вона синхронізується з іншими компонентами застосунку.

Результати тестування підтвердили стабільність і правильність роботи всіх перевірених модулів. Використання пакетів test та flutter_test дозволило швидко виявляти й виправляти помилки на ранніх етапах розробки, забезпечуючи високу якість продукту та його відповідність заданим вимогам. Результати тестування наведено в додатку Е.

2.3. Інструкція з використання системи

Завантажити клієнтський додаток компаньйону можна з релізів репозиторію (Рисунок Д.1), або магазину програм APKPure, за назвою пакету «com.ukrtransnafta».

Після встановлення та запуску програми необхідно виконати вхід на відповідному екрані (Рисунок Є.1) за допомогою електнонної пошти та паролю, отриманих від адміністратора.

Після успішного входу стає можливим перейти до місцязнаходження користувача за допомогою «Плаваючої кнопки дії» на екрані мапи (Рисунок Є.2). Також змінити тип маркерів, що шукаються на «Панелі навігацій», перейти до місцязнаходження, чи побудови маршруту до маркера за допомогою «Плаваючих кнопок дії», після натискання на маркер, а також вийти з облікового запису кнопкою в «Панелі програми».

ВИСНОВКИ

Під час проходження технологічної практики в акціонерному товаристві «Укртранснафта» переді мною було поставлено завдання розробити мобільний додаток для допомоги водіям підприємства під час відряджень. Основна мета проєкту полягала у створенні інструменту, який би забезпечив швидкий доступ до інфраструктурних об’єктів, таких як заправки, готелі та технічні сервіси, а також спрощував планування маршрутів і підвищував ефективність пересування.

У ході роботи було створено мобільний застосунок для операційної системи Android і десктопний застосунок для Windows. Мобільний застосунок забезпечує відображення важливих об’єктів на інтерактивній карті з можливістю перегляду детальної інформації та прокладання маршруту. Десктопний застосунок виконує функцію управління базою даних інфраструктурних об’єктів, що дозволяє адміністраторам оновлювати та коригувати інформацію, доступну водіям.

Практика дала змогу значно розширити мої технічні навички. Я вдосконалив знання з мови Dart і фреймворку Flutter, що дозволило створити кросплатформенний додаток. Використання Google Firebase і його компонентів, таких як Firestore, Authentication і Realtime Database, допомогло освоїти роботу з хмарними технологіями. Інтеграція Google Maps Platform підвищила функціональність застосунку, а створення прототипів у Figma надало цінний досвід у дизайні інтерфейсів.

У процесі розробки особливу увагу було приділено тестуванню продукту. Було проведено юніт-тестування основних компонентів, а також функціональне тестування для перевірки роботи інтегрованих сервісів. Це дозволило виявити й усунути низку помилок, забезпечивши стабільну роботу програми.

Не всі аспекти проєкту вдалося реалізувати у повному обсязі. Наприклад, функціональність для спілкування з адміністратором у критичних ситуаціях залишилася на етапі проєктування через складність

реалізації push-сповіщень у відведені терміни. Водночас базовий функціонал був успішно впроваджений, і підприємство планує використовувати додаток у своїй діяльності.

Практика стала для мене можливістю закріпити теоретичні знання з таких дисциплін, як «Автоматизація інформаційних систем» і «Бази даних», та отримати практичний досвід роботи в реальних умовах. Я також здобув навички управління проєктом, використовуючи інструмент GitHub.

Можливість працювати над практичним завданням для реального замовника стала неоціненним досвідом, який сприяв моєму професійному зростанню.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Figma Learn – Help Center [Електронний ресурс] – Режим доступу до ресурсу: https://help.figma.com/
2. Firebase Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs
3. Docs Flutter [Електронний ресурс] – Режим доступу до ресурсу: https://docs.flutter.dev/
4. Dart documentation [Електронний ресурс] – Режим доступу до ресурсу: https://dart.dev/guides
5. GitHub Docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.github.com/en
6. Documentation for Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs
7. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ПРОХОДЖЕННЯ ТЕХНОЛОГІЧНОЇ ПРАКТИКИ ДЛЯ ЗДОБУВАЧІВ ОСВІТИ СПЕЦІАЛІЗАЦІЇ 121 ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СПЕЦІАЛЬНОСТІ «РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ» [Текст]: МЕТОДИЧНІ РЕКОМЕНДАЦІЇ / Відокремлений структурний підрозділ «Полтавський політехнічний фаховий коледж Національного технічного університету «Харківський політехнічний інститут»; [уклад.: О. В. Бабич, О. В. Бабич]. – Полтава: ВСП «ППФК НТУ «ХПІ», 2023. – 38 с.

ДОДАТОК А. ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ПРОГРАМНИХ ПРОДУКТІВ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ

Порівняння було виконано з сервісами електронних мап та їхнім API, оскільки на ринку програмного забезпечення, у відкритому доступі, відсутні аналоги створеної програми, оскільки вони використовуються лише в межах підприємства.

	Google Maps	Open Street Map	Компаньйон УТФ
Нативність	Так	Ні	Так
Інтуїтивно-зрозумілий інтерфейс користувача	Так	Так	Так
Наявність української мови	Так	Так	Так
Автентифікація з використанням соціальних облікових записів	Так	Ні	Так
Наявність довідкових матеріалів та документації	Так	Так	Так
Можливості з налагодження та розширення функціональності	Так	Так	Так
Вартість ліцензії	Платна	Безкоштовна	Відсутня

ДОДАТОК Б. UML ДІАГРАМА ПРЕЦЕДЕНТІВ

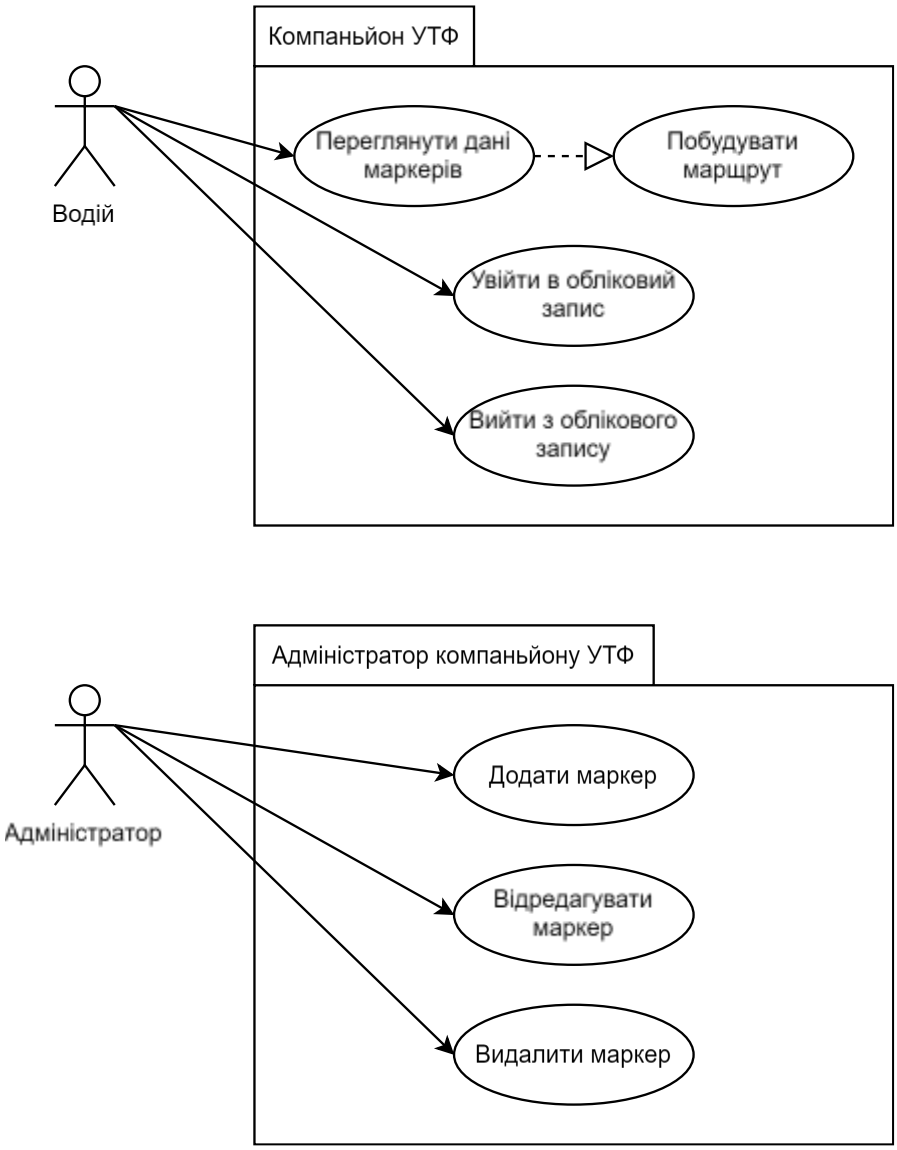

Рисунок Б.1 – Діаграма прецедентів

ДОДАТОК В. ПРОТОТИП ІНТЕРФЕЙСУ

Рисунок В.1 – Екран входу

Рисунок В.2 – Екран мапи

ДОДАТОК Г. UML ДІАГРАМА КЛАСІВ

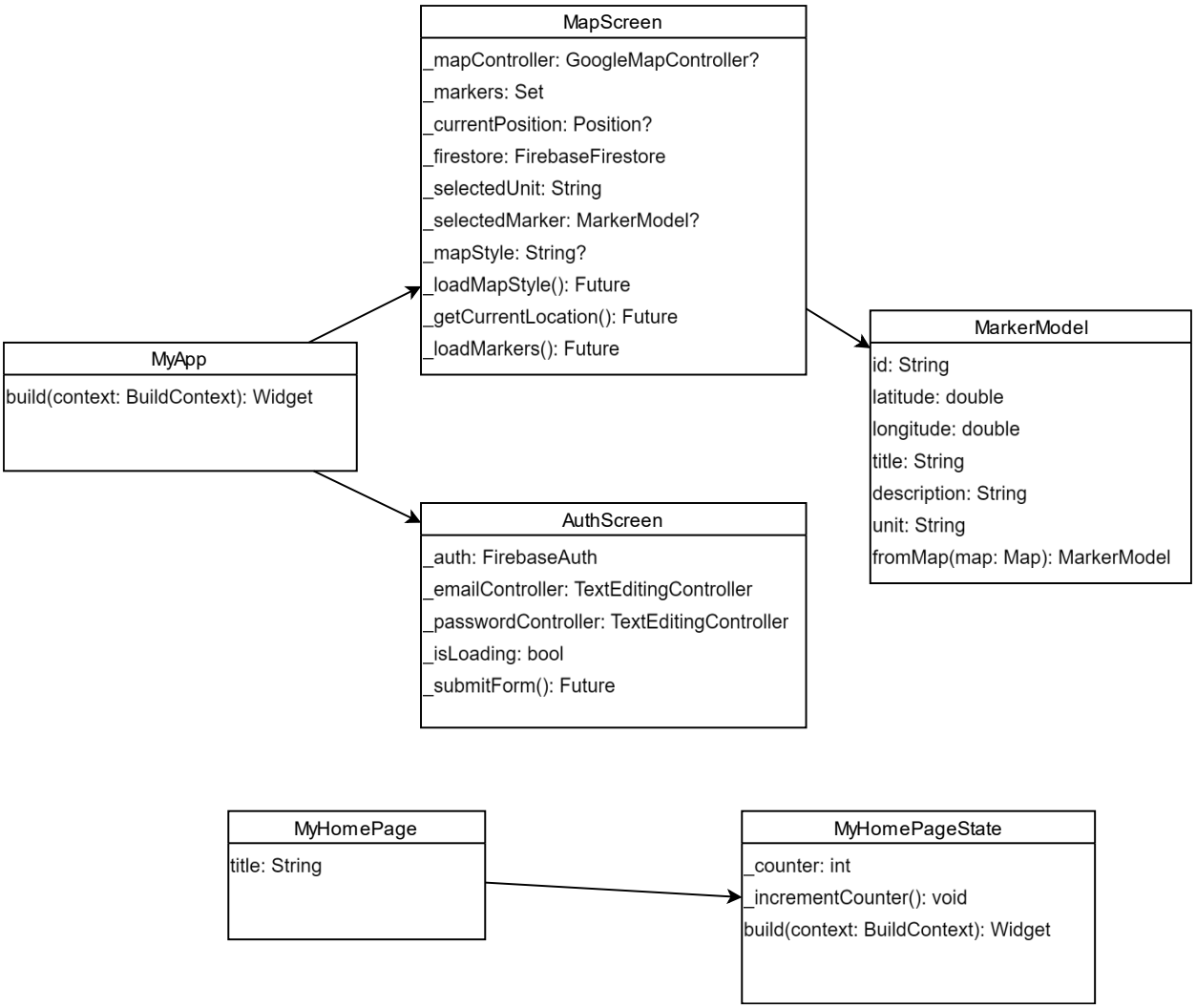

Рисунок Г.1 – Діаграма класів

ДОДАТОК Ґ. ER ДІАГРАМА

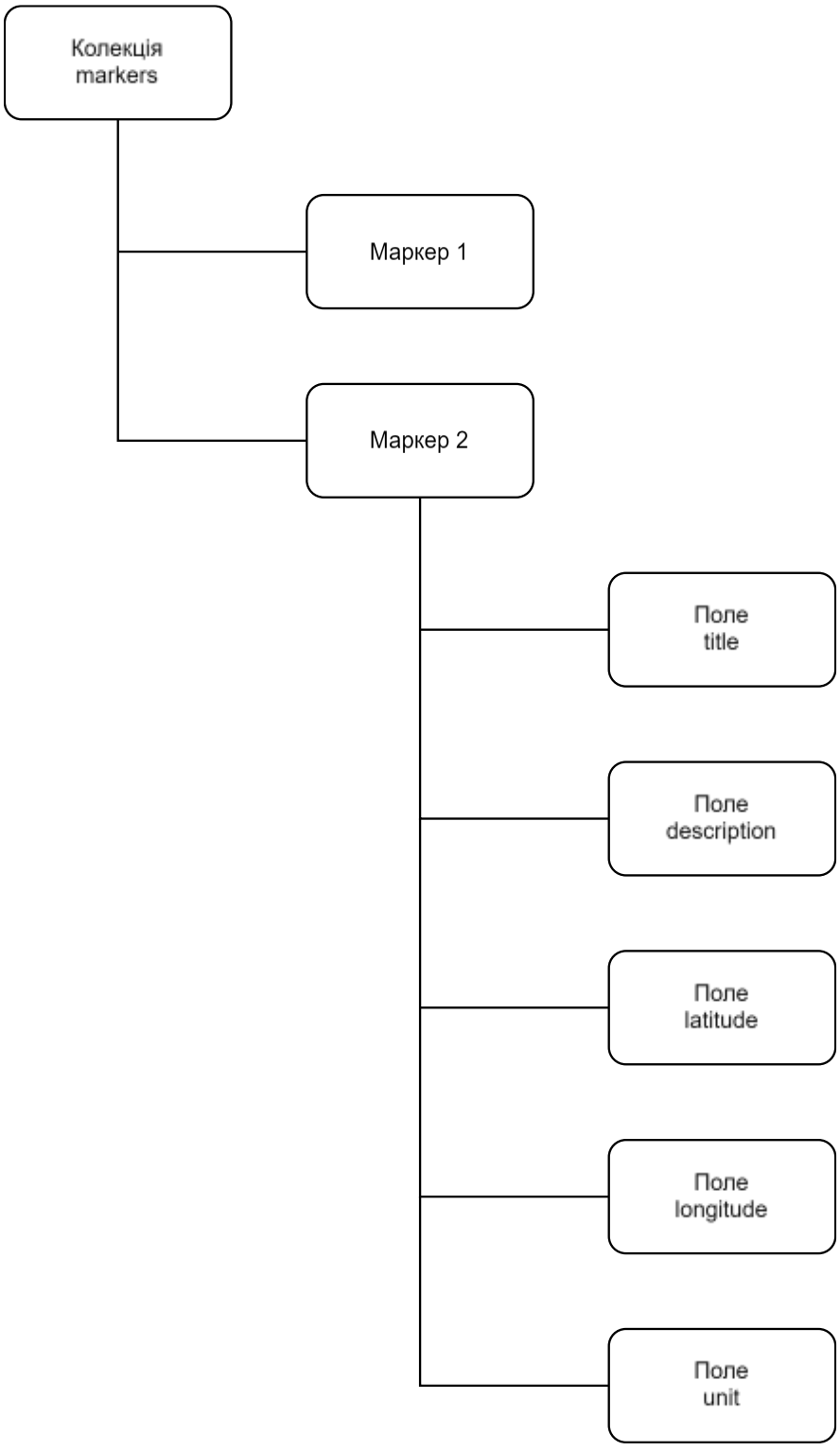

Рисунок Ґ.1 – Структура бази даних

ДОДАТОК Д. ВИХІДНІ КОДИ

```dart
//----------------------------------------
//-  Copyright (c) 2024. Liubchenko Oleh  -
//----------------------------------------

import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'firebase_options.dart';
import 'screens/auth_screen.dart';
import 'screens/map_screen.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Map App',
      theme: ThemeData(
        useMaterial3: true,
        colorScheme: ColorScheme.fromSeed(
          seedColor: Colors.teal,
          brightness: Brightness.light,
        ),
      ),
      darkTheme: ThemeData(
        useMaterial3: true,
        colorScheme: ColorScheme.fromSeed(
          seedColor: Colors.teal,
          brightness: Brightness.dark,
        ),
      ),
      themeMode: ThemeMode.system,
      home: StreamBuilder(
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (ctx, snapshot) {
          if (snapshot.hasData) {
            return MapScreen();
          }
          return AuthScreen();
        },
      ),
```

```
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ),
    );
  }
```

Повний код програми доступний на веб-сервісі GitHub за посиланням, що зображене на рисунку Д.1.

Рисунок Д.1 – Посилання на репозиторій

ДОДАТОК Е. РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Тестування компоненту маркерів:

```
import 'package:flutter_test/flutter_test.dart';
import 'package:ukrtransnafta/models/marker.dart';

void main() {
  group('MarkerModel', () {
    test('fromMap creates a valid MarkerModel', () {
      final map = {
        'id': '1',
        'latitude': 50.0,
        'longitude': 30.0,
        'title': 'Test Marker',
        'description': 'This is a test marker',
        'unit': 'gas',
      };

      final marker = MarkerModel.fromMap(map);

      expect(marker.id, '1');
      expect(marker.latitude, 50.0);
      expect(marker.longitude, 30.0);
      expect(marker.title, 'Test Marker');
      expect(marker.description, 'This is a test marker');
      expect(marker.unit, 'gas');
    });
  });
}
```

Результат:

```
PS D:\Work folder\4-PPPC-Practic-TPE> flutter test
00:02 +1: All tests passed!
PS D:\Work folder\4-PPPC-Practic-TPE>
```

Тестування компоненту автентифікації:

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:ukrtransnafta/screens/auth_screen.dart';

void main() {
  testWidgets('AuthScreen has email and password fields',
(WidgetTester tester) async {
    await tester.pumpWidget(MaterialApp(home: AuthScreen()));

    expect(find.byType(TextField), findsNWidgets(2));
    expect(find.text('Електронна пошта'), findsOneWidget);
```

```
    expect(find.text('Пароль'), findsOneWidget);
  });

  testWidgets('AuthScreen has a login button', (WidgetTester tester)
async {
    await tester.pumpWidget(MaterialApp(home: AuthScreen()));

    expect(find.text('Увійти'), findsOneWidget);
  });
}
```

Результат:

```
PS D:\Work folder\4-PPPC-Practic-TPE> flutter test
00:08 +1: All tests passed!
PS D:\Work folder\4-PPPC-Practic-TPE>
```

Тестування компоненту мапи:

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:ukrtransnafta/screens/map_screen.dart';

void main() {
  testWidgets('MapScreen has a GoogleMap widget', (WidgetTester
tester) async {
    await tester.pumpWidget(MaterialApp(home: MapScreen()));

    expect(find.byType(GoogleMap), findsOneWidget);
  });

  testWidgets('MapScreen has a bottom navigation bar', (WidgetTester
tester) async {
    await tester.pumpWidget(MaterialApp(home: MapScreen()));

    expect(find.byType(NavigationBar), findsOneWidget);
  });
}
```

Результат:

```
PS D:\Work folder\4-PPPC-Practic-TPE> flutter test
00:11 +1: All tests passed!
PS D:\Work folder\4-PPPC-Practic-TPE>
```

ДОДАТОК Є. ЗНІМКИ ЕКРАНУ

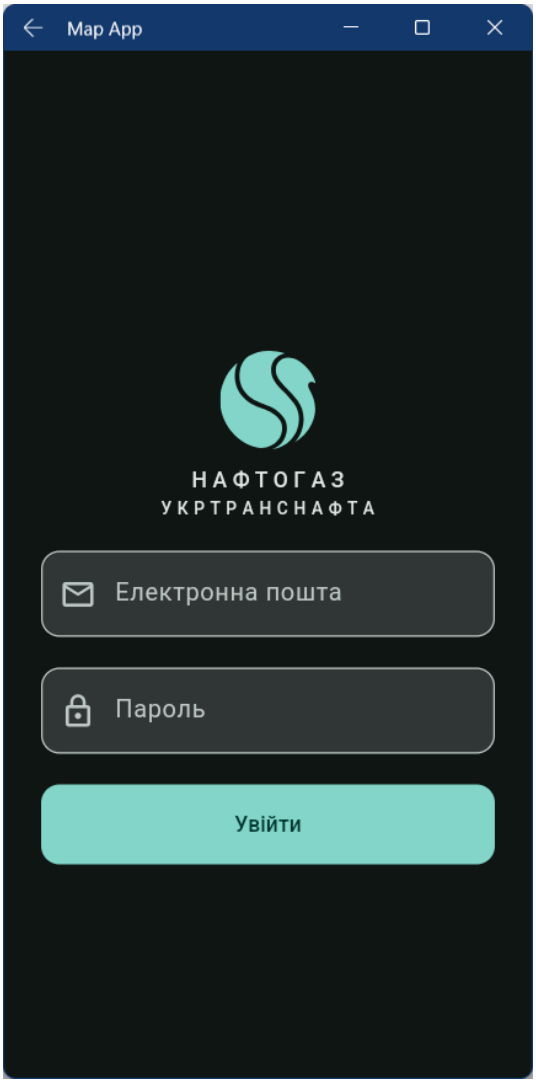

Рисунок Є.1 – Екран входу

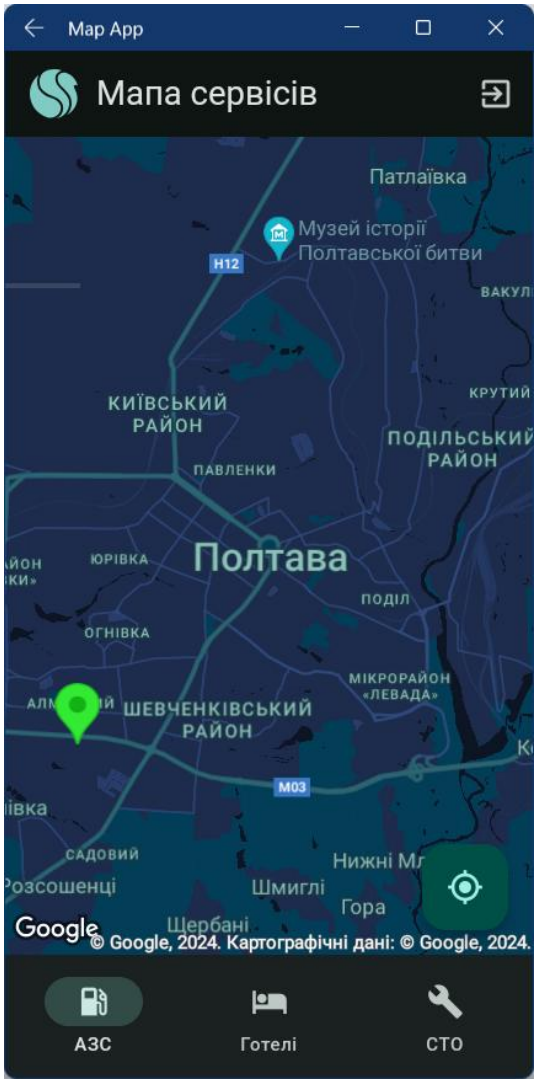

Рисунок Є.2 – Екран мапи

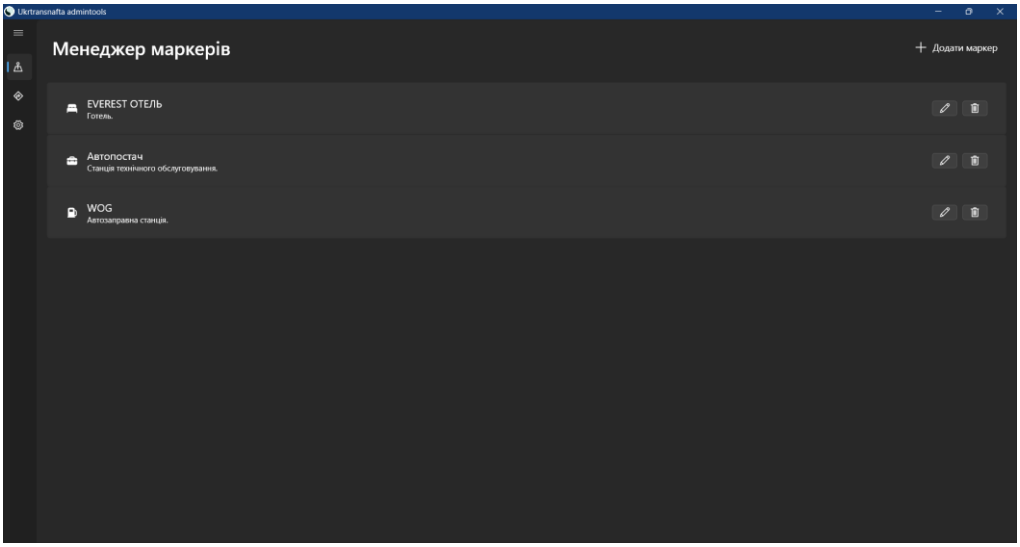

Рисунок Є.3 – Екран списку маркерів в панелі адміністратора

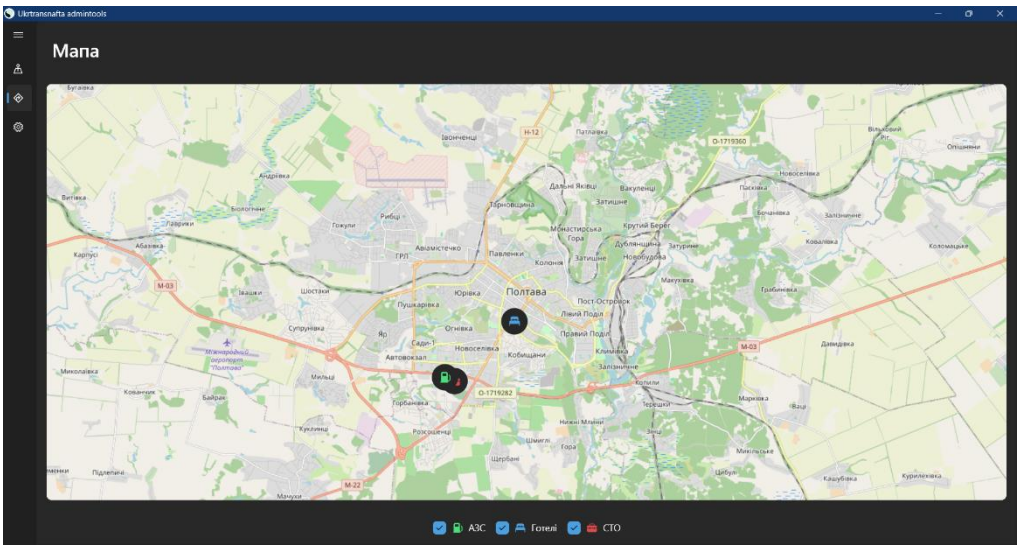

Рисунок Є.4 – Екран мапи в панелі адміністратора

Рисунок Є.5 – Екран налаштувань в панелі адміністратора