Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ПЛАНУВАННЯ ПОДОРОЖЕЙ З ВИКОРИСТАННЯМ FLUTTER ТА GOOGLE MAPS API

DEVELOPMENT OF A MOBILE APPLICATION FOR TRAVEL PLANNING USING FLUTTER AND GOOGLE MAPS API

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-42
Малишевський Богдан Миколайович

Керівник:
Гульчук Юрій Миколайович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ

Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Малишевському Богдану Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка мобільного додатку для планування подорожей з використанням Flutter та Google Maps API»*
Керівник роботи: *асистент Гульчук Ю. М.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*

3. Вихідні дані до роботи: *Flutter, Google Maps API, MySQL, GitHub, методичні вказівки до виконання кваліфікаційної роботи бакалавра*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз предметної області, визначення вимог до мобільного додатку, UML-діаграми, стек технологій, реалізація застосунку, тестування системи, проєктування структури бази даних, забезпечення захисту даних*

5. Перелік графічного матеріалу: *7 рисунків*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Гульчук Ю. М.*		
Специфікація вимог до розробленої системи	*Гульчук Ю. М.*		
Розробка об'єкта проектування	*Гульчук Ю. М.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	4,42 %		
Академічна доброчесність	*Гульчук Ю. М.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	вик.
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	вик
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	вик.
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	вик.
5	Практична реалізація об'єкта проектування та розробка бази даних	до 26.04.2025 р.	вик
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	вик
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	вик

Здобувач вищої освіти Богдан МАЛИШЕВСЬКИЙ

Керівник кваліфікаційної роботи Юрій ГУЛЬЧУК

АНОТАЦІЯ

Малишевський Б. М. Розробка мобільного додатку для планування подорожей з використанням Flutter та Google Maps API. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У першому розділі даної кваліфікаційної роботи проведено аналіз предметної області, вивчено сучасні тенденції у сфері туристичних мобільних застосунків, окреслено основні потреби користувачів під час планування подорожей та сформульовано завдання. У другому розділі визначено функціональні та нефункціональні вимоги до системи, створено UML-діаграми та виконано аналіз і обґрунтування вибору технологічного стеку. Третій розділ присвячено практичній реалізації мобільного додатку: описано процес розробки ключових функцій, побудови інтерфейсу, інтеграції з картографічними сервісами, проведено модульне тестування, а також спроєктовано структуру бази даних і реалізовано базові механізми захисту даних.

У висновках узагальнено результати реалізації поставлених завдання на всіх етапах розробки.

Ключові слова: Flutter, мобільний додаток, подорож, маршрути, Google Maps API, геолокація, MySQL, планування.

ABSTRACT

Malyshevsky B. Development of a Mobile Application for Travel Planning Using Flutter and Google Maps API. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions and a list of references.

The first chapter of this bachelor's qualification project presents an analysis of the subject area, explores current trends in the field of travel mobile applications, outlines the key user needs during trip planning, and formulates the objectives. The second chapter defines the functional and non-functional requirements for the system, provides UML diagrams, and presents the analysis and justification of the chosen technology stack. The third chapter is dedicated to the practical implementation of the mobile application: it describes the development process of core features, interface design, integration with mapping services, module testing, database structure design, and the implementation of basic data protection mechanisms.

The conclusions summarize the results of fulfilling the set objectives at all stages of the development process.

Keywords: Flutter, mobile application, travel, routes, Google Maps API, geolocation, MySQL, planning.

ЗМІСТ

ВСТУП

Актуальність теми. У наш час люди все частіше подорожують самостійно, і тому зростає потреба в зручних інструментах, які допомагають швидко спланувати поїздку. Мобільні додатки стали невід'ємною частиною цього процесу, оскільки дозволяють легко знаходити маршрути, переглядати цікаві місця, будувати плани й орієнтуватися в незнайомих локаціях. Люди прагнуть мати швидкий доступ до маршрутів, туристичних об'єктів, прогнозів погоди та іншої корисної інформації без необхідності звертатися до сторонніх сервісів. Це зумовлює потребу у створенні мобільних застосунків, які поєднують зручність, функціональність та адаптивність до індивідуальних потреб користувача. Враховуючи актуальність теми, дана кваліфікаційна робота присвячена розробці мобільного додатку для планування подорожей, який враховує сучасні технологічні тенденції, вимоги до зручності користування та інтеграцію з геолокаційними сервісами.

Метою кваліфікаційної роботи бакалавра є розробити функціональний кросплатформенний мобільний додаток для планування подорожей, який дозволяє користувачам створювати маршрути, переглядати інформацію про туристичні об'єкти та інтегруватися з картографічними сервісами для покращення користувацького досвіду.

Об'єкт кваліфікаційної роботи бакалавра є мобільний застосунок як інструмент для організації та оптимізації процесу планування подорожей.

Предмет кваліфікаційної роботи є розробка мобільного застосунку для планування подорожей із використанням сучасних технологій, зокрема геолокаційних сервісів та інструментів побудови маршрутів.

Для успішної реалізації поставлених цілей необхідно:

– провести детальний аналіз предметної області, дослідити вплив пандемії та повномасштабного вторгнення;

– виявити всі функціональні та нефункціональні вимоги до мобільного додатку, для забезпечення максимальної конкурентоспроможності та попиту на ринку та розробити UML-діаграму;

– проаналізувати інструменти розробки та обрати оптимальний стек для реалізації мобільного додатка для планування подорожей;

– розробити мобільний додаток з урахуванням всіх вимог;

– детально протестувати функціонал та роботу API, налагодити систему при виявленні недоліків або помилок;

– спроєктувати базу даних, для збереження даних користувача та інформації про подорожі;

– необхідно забезпечити захист інформаційної-комп’ютерної системи для гарантування цілісності та конфіденційності даних для користувача.

РОЗДІЛ 1
АНАЛІЗ ЗАСТОСУВАННЯ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ СФЕРИ ПОДОРОЖЕЙ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасному світі люди дедалі більше прагнуть до самореалізації, особистісного зростання та розширення світогляду. Подорожі стали важливою складовою цього процесу. Вони дозволяють вийти за межі звичного середовища, познайомитися з новими культурами, подолати власні обмеження та знайти нові джерела натхнення. В умовах високої інтенсивності життя та постійного інформаційного навантаження мандрівки також виконують роль своєрідного «перезавантаження» – відновлюють емоційний баланс, додають мотивації та стимулюють внутрішній розвиток. Саме тому особисті подорожі сьогодні – не просто дозвілля, а спосіб глибшого пізнання себе і світу.

Пандемія COVID-19 стала безпрецедентним викликом для світової туристичної галузі. До 2019 року сфера міжнародного туризму демонструвала стабільне зростання, забезпечуючи понад 1,5 трлн доларів доходів на рік, створюючи мільйони робочих місць і формуючи до 10 % глобального ВВП. Проте вже на початку 2020 року через стрімке поширення коронавірусної інфекції світ зіткнувся з масовими обмеженнями на пересування, закриттям кордонів, зупинкою авіаперевезень та повною зупинкою туристичної діяльності в більшості країн.За оцінками Всесвітньої туристичної організації (UNWTO), у 2020 році кількість міжнародних туристичних прибуттів скоротилася на 20-30 %, що означало втрату понад 290-440 мільйонів поїздок у порівнянні з попереднім роком [1].

Таким чином, пандемія коронавірусу стала найглибшою кризою за всю історію існування сучасного туризму. Вона не лише спричинила значні економічні втрати, а й оголила вразливість індустрії до глобальних шоків. Це

змусило країни переглянути роль туризму у своїй економіці, посилити стратегічне планування, інвестувати в цифрову трансформацію туристичних послуг і створити механізми стійкого розвитку на випадок майбутніх глобальних загроз.

Якщо брати до уваги світові тенденції, то сфера туризму поступово відновлюється після глибокого падіння. Згідно з даними, у 2019 році у світі було зафіксовано 1,46 млрд туристів, однак у 2020 році це число впало до 0,65 млрд. Проте вже з 2021 року почалося поступове зростання, і за попередніми оцінками у 2024 році кількість туристів у світі становитиме 1,45 млрд – майже на рівні до пандемії (рис. 1.1).

Рисунок 1.1 – Статистика кількості туристів у Європі та світі [2]

У Європі ж частка туристичних потоків та доходів теж скоротилася в пандемічні роки, однак у 2021-2022 спостерігалося тимчасове домінування Європи (до 65,2 % туристів), через обмеження на міжконтинентальні подорожі. Станом на 2024 рік частка Європи стабілізувалась на рівні близько 51,7 %.

В Україні, на жаль, через повномасштабне вторгнення у 2022 році туристична галузь зазнала серйозного спаду, що лише поглибило кризу в цій

сфері, оскільки значна частина інфраструктури була зруйнована, а безпекова ситуація зробила неможливими візити як внутрішніх, так і іноземних туристів.

Для більш глибокого розуміння предметної області та оцінки можливостей сучасних цифрових інструментів у сфері туризму було проведено аналіз існуючих рішень на ринку мобільних та вебзастосунків для планування подорожей. Такий підхід дозволяє виявити ключові переваги та недоліки кожного сервісу, а також сформувати уявлення про актуальні тенденції, потреби користувачів і потенційні напрями подальшого вдосконалення подібних платформ.

TripIt – це зручний додаток для тих, хто часто подорожує і хоче мати всі деталі маршруту в одному місці (рис. 1.2) [3]. Він автоматично зчитує підтвердження бронювань із електронної пошти та створює структурований маршрут, який можна переглядати навіть офлайн. Особливо корисний для бізнес-подорожей завдяки таймлайну та функції спільного доступу до плану. Водночас у безкоштовній версії обмежений функціонал, а преміумпідписка може бути надто дорогою для звичайного користувача.

Рисунок 1.2 – Інтерфейс додатку TripIt [3]

Наступним проаналізованим додатком став Google Travel. Цей застосунок інтегрується з Gmail та Google Maps і автоматично створює подорожі на основі отриманих листів із бронюванням. Він пропонує ідеї для відвідування, ресторани та маршрути, а також має простий інтерфейс, добре знайомий користувачам Google (рис. 1.3). Це безкоштовний і досить ефективний інструмент для базового планування, але він не дозволяє тонко налаштовувати маршрут вручну, і його майбутнє викликає сумніви через звичку Google закривати сервіси [4].

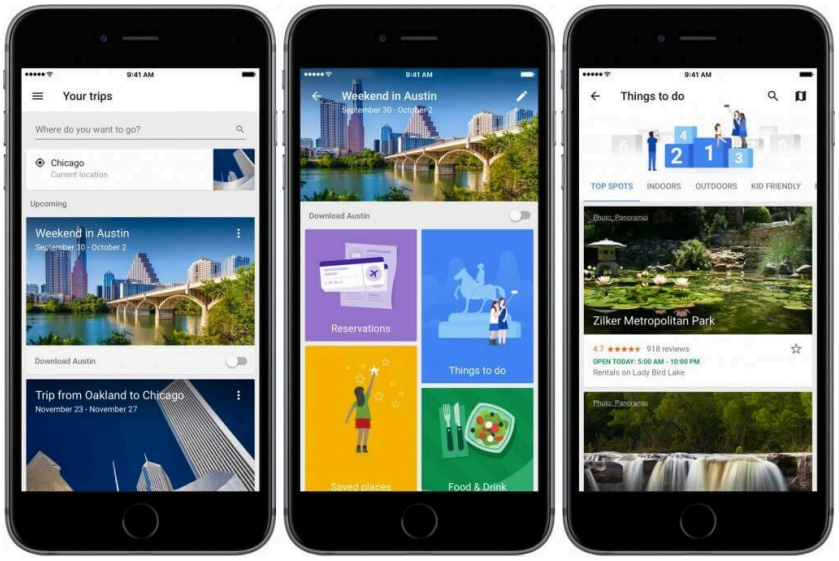

Рисунок 1.3 – Інтерфейс додатка Google Travel [4]

Roadtrippers – ідеальний вибір для тих, хто планує автоподорож. Додаток дозволяє прокладати маршрут і додавати зупинки, включаючи цікаві місця, заклади харчування, заправки та місця для ночівлі (рис. 1.4). Його фокус – не стільки на точності, скільки на тому, щоб зробити поїздку цікавішою. Але основна база даних орієнтована на США та Канаду, через це користувачам з інших регіонів, зокрема Європи чи Азії, функціональність додатка може здаватися обмеженою, що знижує його універсальність у глобальному контексті. А у безкоштовній версії можна додати лише обмежену кількість точок.

Рисунок 1.4 – Інтерфейс додатка Roadtrippers [5]

Проведений аналіз популярних застосунків для планування подорожей показав, що кожен із них має свої сильні сторони, орієнтовані на різні категорії користувачів та сценарії використання. TripIt відзначається зручністю та автоматизацією для тих, хто часто подорожує, особливо з діловою метою. Google Travel приваблює простотою та глибокою інтеграцією з іншими сервісами Google, хоч і має обмежену гнучкість. Roadtrippers, у свою чергу, найкраще підходить для автоподорожей, надаючи цікавий і насичений маршрут, однак значною мірою орієнтований на користувачів із Північної Америки. Загалом, вибір оптимального додатка залежить від типу подорожі, географічного напряму, індивідуальних потреб та вподобань користувача.

З огляду на переваги та обмеження розглянутих рішень, доцільним видається створення нового, універсального додатка для планування подорожей, який би поєднував найкращі функції аналізованих платформ. Такий сервіс міг би автоматично зчитувати бронювання з пошти, як у TripIt, інтегруватися з картами та пропонувати рекомендації, подібно до Google Travel, а також підтримувати варіанти маршрутів для автоподорожей, як у Roadtrippers.

Важливою функцією стало б формування гібридного маршруту, що охоплює авіа-, авто- та пішохідні ділянки, із можливістю ручного редагування, синхронізації з календарем і доступом в офлайн-режимі. Крім того, спільне планування в реальному часі, гнучка фільтрація локацій за інтересами, відгуками та бюджетом, а також адаптивні поради на основі штучного інтелекту зробили б такий додаток дійсно ефективним і зручним для широкого кола мандрівників.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є розробка мобільного додатку для планування подорожей з використанням технології Flutter та API сервісу Google Maps, що забезпечить користувачам зручний інтерфейс для формування маршрутів, перегляду актуальної інформації про туристичні об’єкти, бронювання та організації логістики подорожей. Реалізація такого застосунку сприятиме підвищенню ефективності та швидкості планування мандрівок, особливо в умовах зміненої туристичної ситуації, що вимагає адаптивності та гнучких цифрових рішень.

Для успішного проєктування та розробки додатка були поставленні наступні завдання:

– провести детальний аналіз предметної області, дослідити вплив пандемії та повномасштабного вторгнення;

– виявити всі функціональні та нефункціональні вимоги до мобільного додатку, для забезпечення максимальної конкурентоспроможності та попиту на ринку та розробити UML-діаграму;

– проаналізувати інструменти розробки та обрати оптимальний стек для реалізації мобільного додатка для планування подорожей;

– розробити мобільний додаток з урахуванням всіх вимог;

– детально протестувати функціонал та роботу API, налагодити систему при виявленні недоліків або помилок;

– спроєктувати базу даних, для збереження даних користувача та інформації про подорожі;

– необхідно забезпечити захист інформаційної-комп’ютерної системи для гарантування цілісності та конфіденційності даних для користувача.

Висновки до розділу 1

У сучасному світі подорожі набули нової ролі – вони стали не лише способом відпочинку, а й важливим інструментом самопізнання, розвитку особистості та відновлення емоційного балансу. Проте глобальні події останніх років, зокрема пандемія COVID-19 і повномасштабна війна в Україні, суттєво вплинули на розвиток туристичної галузі. Світовий туризм зіткнувся з безпрецедентною кризою, яка продемонструвала його вразливість до глобальних загроз і підкреслила необхідність цифрової трансформації галузі.

Незважаючи на складнощі, у 2021-2024 роках спостерігається поступове відновлення туристичних потоків. Це супроводжується зростанням попиту на цифрові інструменти, які спрощують процес планування мандрівок, дозволяють адаптуватися до змін та підвищують безпеку подорожей. Особливу актуальність у цьому контексті набувають мобільні додатки для планування подорожей, здатні інтегрувати сервіси бронювання, логістики, актуальну інформацію про маршрути й туристичні об’єкти.

У зв’язку з цим було сформульовано мету кваліфікаційної роботи – розробка мобільного додатку для планування подорожей із використанням Flutter та API Google Maps. Поставлені завдання охоплюють усі етапи розробки – від аналізу предметної області до тестування, проєктування бази даних, забезпечення безпеки даних і підготовки користувацької документації. Реалізація цього проєкту має забезпечити користувачам зручний, адаптивний та

функціональний інструмент для ефективного планування подорожей в умовах сучасних викликів.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

У процесі проєктування програмного забезпечення важливо враховувати не лише загальну ідею та функціональність майбутнього застосунку, а й конкретні технічні та користувацькі вимоги, які визначають його структуру, поведінку та надійність. Оскільки мобільний додаток для планування подорожей покликаний забезпечити комфортну взаємодію користувача з сервісами навігації, пошуку та організації маршруту, він повинен відповідати сучасним стандартам розробки, мати зручний інтерфейс та високу продуктивність. Нижче наведено перелік функціональних і нефункціональних вимог, які було сформовано на основі проведеного аналізу, вивчення потреб майбутніх користувачів та специфіки обраної платформи розробки.

Функціональні вимоги:

– авторизація та реєстрація користувачів. Користувач повинен мати змогу створити обліковий запис або увійти до існуючого через email, Google-акаунт або інші сторонні сервіси (OAuth). Це дозволяє зберігати персональні дані, історію подорожей та маршрути;

– інтерактивна мапа з підтримкою геолокації. Додаток повинен відображати карту з Google Maps API, визначати поточне місцезнаходження користувача, дозволяти масштабування, прокрутку та взаємодію з мапою;

– пошук туристичних об’єктів та сервісів. Забезпечити пошук за ключовими словами, категоріями (готелі, пам’ятки, ресторани тощо) та фільтрами (рейтинг, ціна, відстань, тип місця);

– формування та збереження маршрутів. Користувач може створити маршрут, додаючи точки на мапі, отримувати оптимальні варіанти шляху (авто,

пішки, громадський транспорт), зберігати маршрут у профілі, редагувати або видаляти його;

– перегляд інформації про об’єкти. При натисканні на об’єкт на мапі відображається детальна інформація: назва, опис, рейтинг, відгуки, фотографії, контактні дані, години роботи тощо;

– нотатки та особистий план подорожі. Користувач має змогу створювати текстові нотатки до маршруту, додавати дати та коментарі, формуючи особистий план подорожі.

– офлайн-режим. Можливість збереження мап та маршрутів для подорожей у зонах без доступу до Інтернету;

– сповіщення та підказки. Надсилання нагадувань щодо майбутніх подорожей, зміни погодних умов або інформації про ситуацію в регіоні (наприклад, небезпека, застереження);

– планувальник витрат. Можливість додавати плановані витрати на кожен етап подорожі (житло, харчування, квитки тощо) та переглядати загальний бюджет подорожі.

Після визначення основних функціональних вимог до майбутнього додатку, які охоплюють можливості планування маршруту, інтеграцію з сервісами, підтримку спільного редагування та персоналізовані рекомендації, доцільно перейти до розгляду нефункціональних вимог. Саме вони забезпечують якісні характеристики системи, такі як швидкодія, масштабованість, зручність інтерфейсу, безпека даних та стабільність роботи, що є критично важливими для комфортного та надійного використання додатка в реальних умовах подорожі.

Нефункціональні вимоги:

– кросплатформність. Додаток повинен коректно працювати на пристроях з ОС Android та iOS без втрати функціональності або якості інтерфейсу;

– продуктивність. Інтерфейс має бути швидким і чуйним. Основні функції (відкриття мапи, побудова маршруту, пошук) повинні виконуватися з мінімальними затримками;

– масштабованість. Архітектура застосунку має дозволяти подальше розширення функціоналу (додавання чатів, бронювання квитків, інтеграція з іншими сервісами);

– надійність і стійкість до збоїв. Програма не повинна аварійно завершуватись під час використання; має бути реалізована обробка помилок (наприклад, при втраті з’єднання з інтернетом);

– безпека даних. Конфіденційні дані користувачів мають передаватися через захищені канали (HTTPS), зберігатися з використанням шифрування. Забезпечити базову автентифікацію й авторизацію;

– інтуїтивно зрозумілий інтерфейс (UI/UX). Додаток має мати логічну, просту та привабливу структуру, орієнтовану на користувача без технічного досвіду;

– підтримка багатомовності. Забезпечити підтримку кількох мов, зокрема української, англійської, можливо, інших популярних мов для туристів;

– інтеграція з API сторонніх сервісів. Передбачити модульну інтеграцію з Google Maps API, сервісами бронювання (Booking.com, Airbnb), погодними сервісами або туристичними порталами.

Сукупність вищезазначених функціональних та нефункціональних вимог формує цілісний та продуманий фундамент для розробки ефективного мобільного застосунку для планування подорожей. Завдяки чіткому розмежуванню функцій, орієнтованих на зручність користувача, та технічних характеристик, що забезпечують стабільність і масштабованість, створюється надійна основа для реалізації інноваційного продукту. Такий підхід дозволяє не лише задовольнити актуальні потреби мандрівників, а й забезпечити гнучкість і готовність додатка до майбутнього розширення функціоналу відповідно до змін у туристичній галузі та зростаючих очікувань користувачів.

Для візуалізації функціональної структури програмного забезпечення було створено UML-діаграму використання (use case diagram) [6], яка наочно відображає взаємодію користувача з основними можливостями мобільного додатку (рис. 2.1). Такий підхід дозволяє чітко визначити межі системи, роль користувача та перелік сценаріїв, які реалізуються в процесі роботи з додатком.

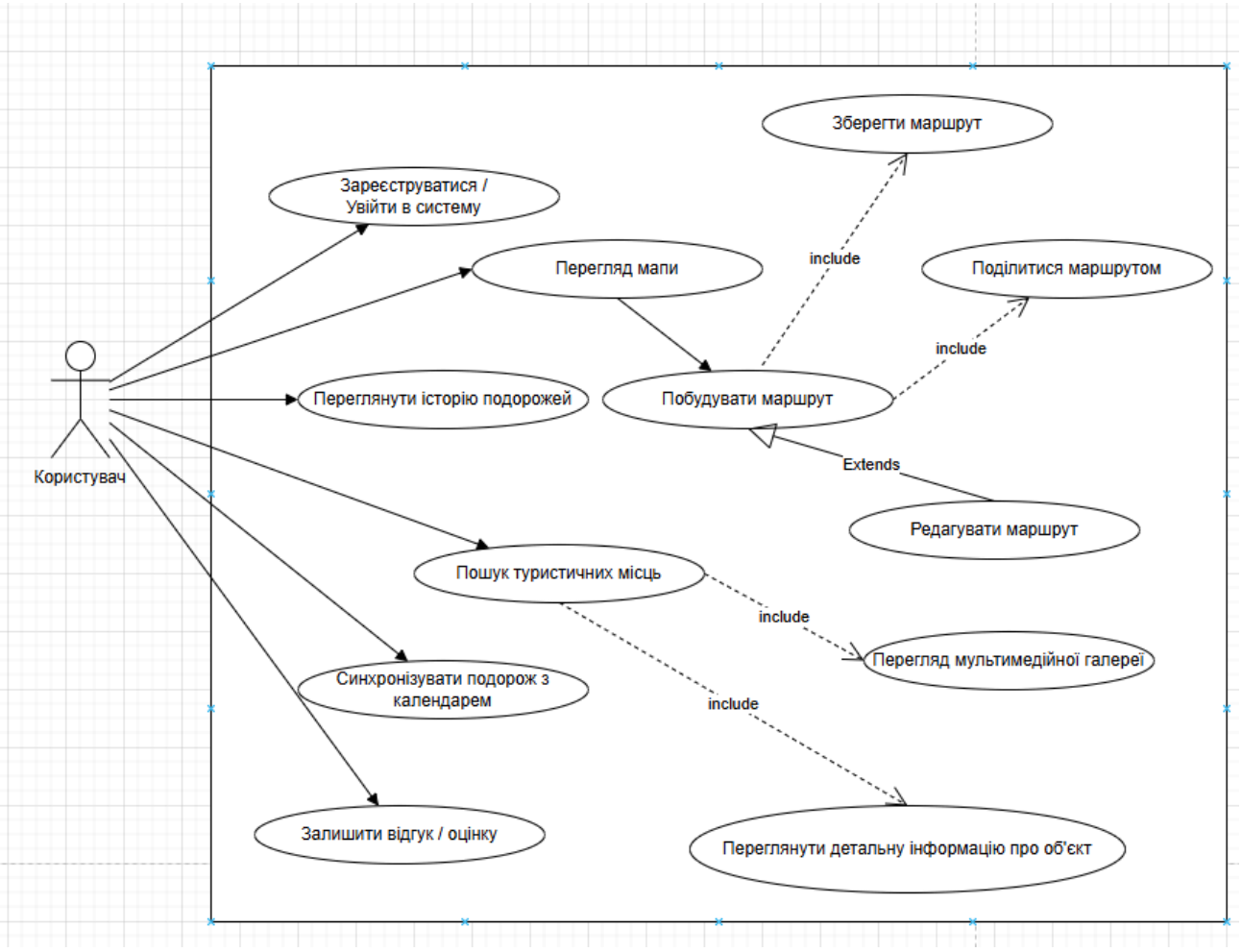

Рисунок 2.1 – Діаграма використання

Представлена діаграма використання демонструє взаємодію актора «Користувач» із ключовими функціональними можливостями мобільного додатку для планування подорожей. Основним сценарієм виступає «Побудувати маршрут», який відображає головне завдання користувача – створення

персоналізованого маршруту подорожі. Цей сценарій доповнюється іншими діями за допомогою зв’язків типу include та extend.

Після побудови діаграми варіантів використання, яка дала змогу визначити основні сценарії взаємодії користувача з системою, було розроблено UML-діаграму класів (рис. 2.2). Вона деталізує внутрішню структуру майбутнього застосунку, визначаючи ключові класи, їхні атрибути, методи та зв’язки між ними. Цей крок дозволив перейти від зовнішнього опису функціональності до формалізованого представлення логіки обробки даних і взаємодії об’єктів у системі.

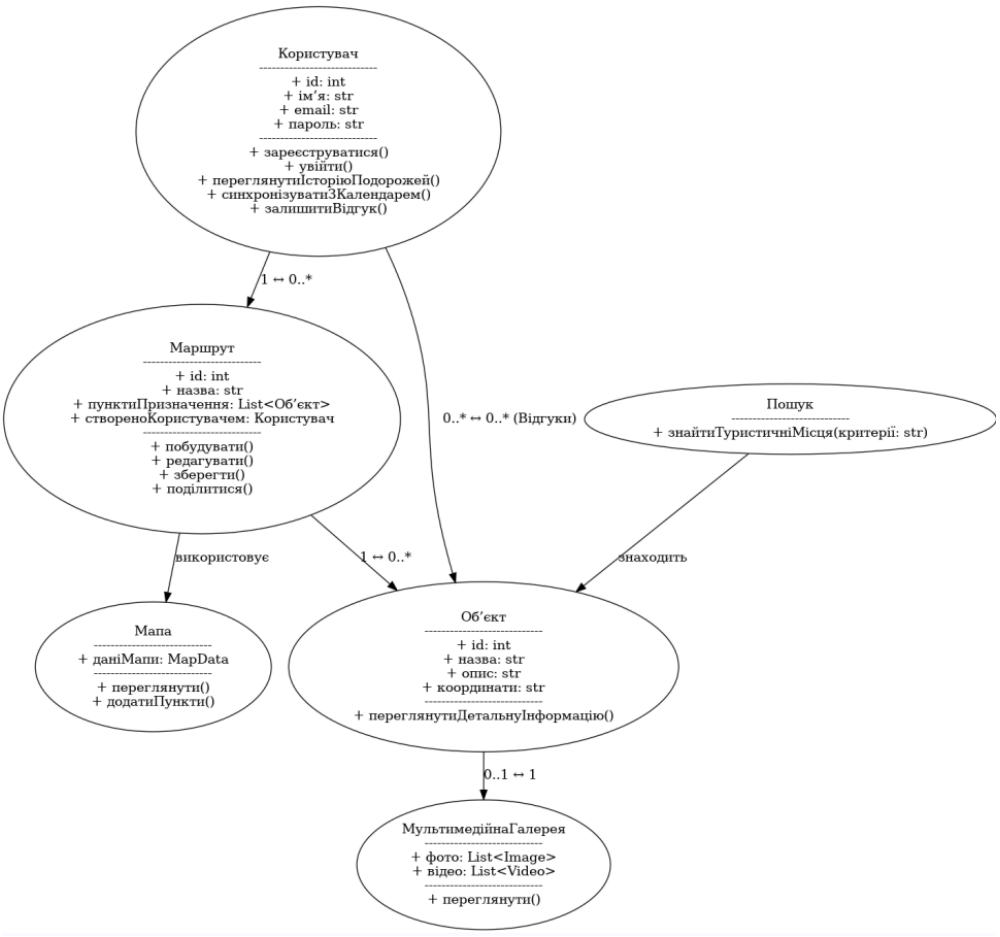

Рисунок 2.2 – Діаграма класів

Діаграма класів відображає основні сутності системи: «Користувача», «Маршрут», «Туристичний об’єкт», «Мапу», «Галерею» та модуль «Пошук».

Вона демонструє їхні атрибути, методи та взаємозв’язки. Зокрема, користувач може створювати маршрути, які складаються з туристичних об’єктів; об’єкти пов’язані з мультимедійною галереєю; маршрут відображається на мапі. Така структура забезпечує логічну організацію даних та підтримує реалізацію основної функціональності застосунку.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

На етапі розробки мобільного додатку для планування подорожей було проведено ґрунтовний аналіз доступних інструментів та технологій з метою вибору оптимального стека, який би забезпечив стабільну, зручну та масштабовану роботу застосунку на обох мобільних платформах. Основну увагу було приділено інтерфейсній частині, картографічному API та системі управління базами даних.

Для створення мобільного інтерфейсу було обрано технологію Flutter – сучасний фреймворк від компанії Google, який дозволяє розробляти кросплатформенні додатки з єдиною кодовою базою для Android та iOS [7]. Flutter забезпечує високу продуктивність за рахунок компіляції в нативний код і має велику бібліотеку віджетів, що дозволяє створювати привабливі й адаптивні інтерфейси. Крім того, Flutter має широку підтримку спільноти, регулярні оновлення та активний розвиток з боку Google, що гарантує надійність і зручність у використанні. Як альтернатива розглядалася технологія React Native, яка також дозволяє створювати кросплатформенні додатки, однак її продуктивність дещо нижча через проміжний JavaScript-шар, а створення складних інтерфейсів вимагає залучення сторонніх бібліотек. Крім того, розглядалась нативна розробка за допомогою Kotlin для Android [8] та Swift для iOS [9], але такий підхід значно ускладнив би підтримку проєкту, оскільки потребував би створення й обслуговування двох окремих кодових баз.

Наступним постало питання реалізації картографічного функціоналу. Було обрано Google Maps API, що надає доступ до потужного набору інструментів для побудови маршрутів, геолокації, перегляду об’єктів на мапі та взаємодії з ними. Google Maps API має високу точність, глобальне покриття, зручну документацію та офіційні плагіни для інтеграції з Flutter, що значно спрощує процес розробки [10]. Однією з альтернатив був Mapbox API, який також підтримує кастомізацію мап і має сучасні візуальні рішення, однак потребує складнішого налаштування та часто є платним при масштабному використанні [11]. Іншим варіантом був OpenStreetMap – є повністю відкритим і безкоштовним, проте не надає повноцінного API з готовими функціями, а отже потребує більше зусиль на реалізацію базових речей, які вже вбудовані в Google Maps [12].

Щоб забезпечити додатку необхідний функціонал необхідно зберігати дані та забезпечити їх цілісність. Щодо зберігання даних, вибір зупинився на системі управління базами даних MySQL. Ця реляційна система управління базами даних добре зарекомендувала себе у великій кількості веб та мобільних проєктів, забезпечує високу швидкодію, підтримує складні SQL-запити, зв’язки між таблицями та гарантує цілісність даних. MySQL є безкоштовною, має широку підтримку хостингів і сумісна з багатьма ORM-інструментами, що спрощує її інтеграцію з бекендом [13]. Як альтернатива розглядався PostgreSQL – ще одна потужна реляційна система управління базами даних, яка забезпечує більшу гнучкість у роботі з геоданими та складними структурами, однак для цього проєкту її переваги не були критично необхідними [14]. Також аналізувалися NoSQL-рішення, зокрема Firebase Realtime Database та Firestore, які дозволяють працювати з даними в реальному часі та мають зручну інтеграцію з мобільними застосунками. Проте їхня схема зберігання даних менш придатна для складних зв’язків між об’єктами та не дає такого рівня контролю, як традиційні реляційні системи управління базами даних. Варіант з використанням SQLite, хоча й дозволяє зберігати дані локально та легко

працювати з ними на пристрої, не задовольняв вимоги до централізованого зберігання та синхронізації даних між різними користувачами.

Під час вибору технологічного стеку для розробки мобільного додатку також було прийнято рішення щодо використання платформи для контролю версій, збереження коду та організації командної роботи. У якості такого інструменту був обраний GitHub – популярний хостинг для Git-репозиторіїв, який забезпечує зручне середовище для зберігання проєкту, відстеження змін у коді, створення гілок та об’єднання різних версій програми. GitHub відіграє важливу роль у процесі розробки, адже дозволяє не лише зберігати історію змін, а й організовувати роботу з гілками, вести документацію, створювати pull-реквести для перевірки коду та автоматизувати рутинні процеси за допомогою GitHub Actions. Це дає змогу налаштувати автоматичне тестування, перевірку синтаксису або збірку проєкту при кожному оновленні репозиторію, що значно підвищило ефективність розробки [15].

Серед альтернатив GitHub варто виділити такі сервіси, як GitLab і Bitbucket. GitLab надає схожий функціонал і особливо цінується у корпоративному середовищі завдяки потужній системі CI/CD. Bitbucket, своєю чергою, інтегрується з іншими інструментами Atlassian, такими як Jira, і також підтримує приватні репозиторії. Проте GitHub було обрано з огляду на його простоту у використанні, широку підтримку спільноти, наявність безкоштовних приватних репозиторіїв, а також хорошу інтеграцію з Flutter-проєктами.

Отже, комбінація Flutter, Google Maps API, MySQL та GitHub була обрана як найбільш збалансоване та оптимальне рішення, яке дозволяє реалізувати повний функціонал мобільного додатку з високою якістю інтерфейсу, стабільною інтеграцією з картографічними сервісами та надійним зберіганням даних. Такий стек технологій не лише повністю відповідає визначеним функціональним та нефункціональним вимогам, а й створює надійну основу для подальшого масштабування проєкту, додавання нових сервісів і інтеграцій, а також забезпечення стабільної та безпечної роботи системи в майбутньому.

Висновки до розділу 2

У результаті проведеного аналізу вимог, функціональних можливостей та доступних технологій було обґрунтовано вибір інструментів розробки мобільного додатку для планування подорожей. З урахуванням потреб користувачів та специфіки туристичної галузі було сформовано повний перелік функціональних та нефункціональних вимог, що забезпечують зручність, стабільність та ефективність роботи майбутньої системи. Для реалізації поставлених завдань було обрано оптимальний стек технологій, до якого увійшли Flutter, Google Maps API та MySQL. Такий підхід дав змогу створити кросплатформенний, гнучкий та продуктивний застосунок, здатний задовольнити актуальні запити користувачів та адаптуватися до змін у сфері цифрових туристичних сервісів. Сукупність прийнятих рішень формує надійну архітектурну основу, що дозволяє не лише реалізувати початкову версію додатка, але й розвивати його у майбутньому відповідно до потреб ринку.

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ПЛАНУВАННЯ ПОДОРОЖЕЙ

3.1 Практична реалізація об’єкта проєктування

Розробка мобільного додатку складалася з кількох послідовних етапів: ініціалізації проєкту, налаштування середовища, підключення зовнішніх сервісів, реалізації логіки користувача, створення маршруту, збереження та перегляду даних, а також налаштування взаємодії з геолокацією. Усі компоненти були реалізовані з урахуванням попередньо сформульованих функціональних і нефункціональних вимог, з особливим акцентом на кросплатформність, швидкодію та зручність інтерфейсу.

Першим кроком було створення нового проєкту за допомогою фреймворку Flutter, який дозволив реалізувати інтерфейс одразу для двох платформ – Android та iOS. Після ініціалізації було сформовано логічну структуру директорій, де lib/screens відповідає за окремі екрани додатка, lib/widgets містить багаторазові елементи інтерфейсу, lib/models – описує структуру даних, а lib/services – забезпечує взаємодію з API та локальними сервісами. У файлі pubspec.yaml (ліст. 3.1) було додано основні залежності, включно з плагінами для карт, геолокації, HTTP-запитів і базової авторизації.

Лістинг 3.1 – Файл pubspec.yaml

```
using static System.Net.Mime.MediaTypeNames;
using System.Runtime.CompilerServices;
name: travel_planner_app
description: Мобільний додаток для планування подорожей з
інтеграцією карт та маршрутів
version: 1.0.0 + 1
environment:
sdk: ">=3.0.0 <4.0.0"
dependencies:
flutter:
sdk: flutter
```

```
# Інтерфейс та навігація
cupertino_icons: ^1.0.6
  provider: ^6.1.2
  # Робота з HTTP-запитами
  http: ^0.13.6
  # Робота з геолокацією
  geolocator: ^10.1.0
  geocoding: ^2.1.0
  # Google Maps
  google_maps_flutter: ^2.5.0
  flutter_polyline_points: ^1.0.0
  # Робота з датами
  intl: ^0.18.1
  # Локальне зберігання
  shared_preferences: ^2.2.2
  # Форматування форм
  flutter_form_builder: ^9.1.1
  # Мультимедіа та галерея
  image_picker: ^1.0.5
  # Робота з повідомленнями
  flutter_local_notifications: ^16.3.0
dev_dependencies:
flutter_test:
sdk: flutter
flutter_lints: ^3.0.1
  mockito: ^5.4.4
  build_runner: ^2.4.7
flutter:
uses - material - design: true
  assets:
-assets / images /
-assets / icons /
```

кінець лістингу 3.1

Найважливішою функцією додатка є інтерактивна мапа, що відображає поточне місцезнаходження користувача, дозволяє додавати точки маршруту, переглядати об’єкти інфраструктури та формувати напрямки руху. Для цього було використано офіційний пакет google_maps_flutter, який забезпечує плавну інтеграцію з картами Google. На початковому етапі було реалізовано просту карту з центруванням на заданій географічній точці. У подальшому додано можливість взаємодії з картою: додавання маркерів при натисканні, переміщення камери та побудова маршрутів. Продемонстрований фрагмент

коду відповідає за відображення карти та інтерактивну взаємодію користувача з нею (ліст. 3.2).

Лістинг 3.2 – Відображення карти

```
GoogleMap(
  initialCameraPosition: CameraPosition(
    target: LatLng(50.4501, 30.5234),
    zoom: 12,
  ),
  markers: _markers,
  onTap: _addMarker,
)
```

кінець лістингу 3.2

Для забезпечення персоналізованого доступу було реалізовано просту систему авторизації з формами входу та реєстрації. Користувач вводить адресу електронної пошти та пароль, які надсилаються на сервер для верифікації. У перспективі така система може бути розширена за допомогою сторонніх сервісів, таких як Google Sign-In чи Firebase Auth, однак у базовому варіанті використовувався REST API та класичне збереження сесій (ліст. 3.3).

Лістинг 3.3 – Авторизацію користувача через REST API

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:shared_preferences/shared_preferences.dart';
class AuthService {
  final String baseUrl = 'https://your-backend-url.com/api';
  Future<bool> login(String email, String password) async {
    final response = await http.post(
      Uri.parse('$baseUrl/login/'),
      headers: {'Content-Type': 'application/json'},
      body: jsonEncode({
        'email': email,
        'password': password,
      }),);
    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      final prefs = await SharedPreferences.getInstance();
      await prefs.setString('token', data['token']);
```

```
      return true;
    } else {
      return false;}
  }
  Future<bool> register(String email, String password) async {
    final response = await http.post(
      Uri.parse('$baseUrl/register/'),
      headers: {'Content-Type': 'application/json'},
      body: jsonEncode({
        'email': email,
        'password': password,
      }),
    );
    return response.statusCode == 201;}
  Future<void> logout() async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.remove('token');}
  Future<bool> isLoggedIn() async {
    final prefs = await SharedPreferences.getInstance();
    return prefs.containsKey('token');}
}
```

кінець лістингу 3.3

Ключова функція додатка – створення персонального маршруту. Користувач додає точки на мапу, які формують маршрут. Ці координати передаються до Google Maps Directions API, який обчислює оптимальний шлях з урахуванням типу транспорту та дорожньої ситуації. Далі маршрут візуалізується у вигляді лінії на мапі (ліст. 3.4).

Лістинг 3.4 – Побудова маршруту з точок користувача за допомогою Google Maps Directions API

```
using static System.Net.WebRequestMethods;
using System.Collections.Generic;
using System;

import 'dart:convert';
import 'package:http/http.dart' as http;
import
'package:flutter_polyline_points/flutter_polyline_points.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';

class RouteService
```

```
{
    final String _googleApiKey = 'YOUR_GOOGLE_MAPS_API_KEY';

    Future<List<LatLng>> getRouteCoordinates(LatLng origin, LatLng
destination) async {
    final url =

'https://maps.googleapis.com/maps/api/directions/json?origin=${origi
n.latitude},${origin.longitude}&destination=${destination.latitude},
${destination.longitude}&mode=driving&key=$_googleApiKey';

    final response = await http.get(Uri.parse(url));
    final data = jsonDecode(response.body);

    if (data['status'] != 'OK') {
      throw Exception('Failed to get route');
}

final points = data['routes'][0]['overview_polyline']['points'];
final polylinePoints = PolylinePoints().decodePolyline(points);
return polylinePoints
    .map((point) => LatLng(point.latitude, point.longitude))
    .toList();
  }
}
```

кінець лістингу 3.4

Щоб забезпечити можливість повторного доступу до маршрутів, користувач має змогу зберегти маршрут у своєму профілі (ліст. 3.5). Для цього реалізовано API-запит на сервер, де дані маршруту зберігаються у базі MySQL. Кожен маршрут містить назву, список точок та додаткові параметри (наприклад, дата, тривалість, бюджет).

Лістинг 3.5 – Збереження маршруту користувача в базу через API-запит

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:shared_preferences/shared_preferences.dart';
class SaveRouteService {
  final String baseUrl = 'https://your-backend-url.com/api';
  Future<bool> saveRoute({
    required String routeName,
    required List<Map<String, dynamic>> points,
    required DateTime date,
```

```
    required Duration duration,
    required double budget,
  }) async {
    final prefs = await SharedPreferences.getInstance();
    final token = prefs.getString('token');
    final response = await http.post(
      Uri.parse('$baseUrl/routes/'),
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer $token',
      },
      body: jsonEncode({
        'name': routeName,
        'points': points,
        'date': date.toIso8601String(),
        'duration': duration.inMinutes,
        'budget': budget,
      }),
    );
    return response.statusCode == 201;
  }
}
```

кінець лістингу 3.5

Збережені маршрути доступні користувачеві через спеціальний екран. Кожен маршрут можна переглянути детальніше, включно з нотатками, фото та витратами. Інформація завантажується з сервера та виводиться у вигляді списку (ліст. 3.6).

Лістинг 3.6 – Завантаження та відображення збережених маршрутів

```
class _SavedRoutesScreenState extends State<SavedRoutesScreen> {
  List<dynamic> routes = [];
  bool isLoading = true;
  @override
  void initState()
  Future<void> fetchSavedRoutes() async {
    final prefs = await SharedPreferences.getInstance();
    final token = prefs.getString('token');
    final response = await http.get(
      Uri.parse('https://your-backend-url.com/api/routes/'),
      headers: {'Authorization': 'Bearer $token'},);
    if (response.statusCode == 200) {
      setState(() {
        routes = jsonDecode(response.body);
```

```
        isLoading = false;
      });
    } else {
      setState(() {
        isLoading = false;
      });
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Збережені маршрути')),
      body: isLoading
          ? Center(child: CircularProgressIndicator())
          : ListView.builder(
              itemCount: routes.length,
              itemBuilder: (context, index) {
                final route = routes[index];
                return Card(
                  child: ListTile(
                    title: Text(route['name']),
                    subtitle: Text('Дата: ${route['date']} | Бюджет:
${route['budget']} грн'),
                    onTap: () {
                      // TODO: перехід на екран деталізації маршруту
                    },
                  ),
                );
            );
}
```

кінець лістингу 3.6

Для точного визначення позиції користувача додаток використовує бібліотеку geolocator. Це дає змогу отримати координати в реальному часі, показувати поточне місцезнаходження та адаптувати маршрут за обставинами. Крім того, реалізовано базову систему сповіщень, яка попереджає про зміну погоди, розклад або наближення до об’єкта (ліст. 3.7).

Лістинг 3.7 – Отримання геопозиції та показу локального сповіщення

```
class LocationService {
  Future<Position> getCurrentPosition() async {
    LocationPermission permission = await
Geolocator.checkPermission();
    if (permission == LocationPermission.denied) {
```

```
      permission = await Geolocator.requestPermission();
    }
    return await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.high,
    );
  }
}
class NotificationService {
  final _notifications = FlutterLocalNotificationsPlugin();
  Future<void> show(String title, String body) async {
    const android = AndroidNotificationDetails('id', 'channel');
    const details = NotificationDetails(android: android);
    await _notifications.show(0, title, body, details);
  }
}
final location = await LocationService().getCurrentPosition();
await NotificationService().show('Нагадування', 'Ви наближаєтесь до
об’єкта');
```

кінець лістингу 3.7

Однією з особливостей розробленого мною мобільного додатку для планування подорожей є інтеграція з календарем, яка дозволяє зручно фіксувати дату початку або завершення мандрівки, встановлювати нагадування, а також синхронізувати подію з локальним календарем пристрою або Google Calendar (ліст. 3.8). Така можливість підвищує організованість користувача, дозволяючи заздалегідь спланувати інші справи навколо запланованої поїздки та отримувати автоматичні сповіщення про її наближення.

Лістинг 3.8 – Додавання події до локального календаря з використанням device_calendar

```
import 'package:device_calendar/device_calendar.dart';

final DeviceCalendarPlugin _calendarPlugin = DeviceCalendarPlugin();

Future<void> addTripToCalendar(String tripName, DateTime startDate,
DateTime endDate) async {
  var permissionsGranted = await _calendarPlugin.hasPermissions();
  if (!(permissionsGranted?.data ?? false)) {
    permissionsGranted = await _calendarPlugin.requestPermissions();
    if (!(permissionsGranted?.data ?? false)) return;
  }
```

```
  final calendarsResult = await _calendarPlugin.retrieveCalendars();
  final calendars = calendarsResult?.data;
  if (calendars == null || calendars.isEmpty) return;

  final calendarId = calendars.first.id; // використати перший
доступний календар

  final event = Event(
    calendarId,
    title: tripName,
    start: startDate,
    end: endDate,
    description: 'Запланована подорож: $tripName',
  );

  await _calendarPlugin.createOrUpdateEvent(event);
}
```

кінець лістингу 3.8

Для покращення взаємодії користувача з додатком та формування бази якісних маршрутів було реалізовано особисту систему оцінювання подорожей (ліст. 3.9). Після завершення або збереження маршруту користувач має можливість поставити оцінку у вигляді зірок (від 1 до 5), що дозволяє визначити, наскільки задовільною була поїздка. У майбутньому, за умови реалізації функції спільного доступу до маршрутів, ця система може стати основою для створення спільноти користувачів, які діляться власними мандрівками та залишають відгуки.

Оцінка маршруту зберігається в базі даних разом з інформацією про маршрут та може бути використана для сортування, фільтрації або аналітики. Це також дає змогу користувачу самостійно відстежувати, які подорожі були найвдалішими.

Лістинг 3.9 – Встановлення оцінки маршруту

```
class RatingWidget extends StatefulWidget {
  final int routeId;
  const RatingWidget({required this.routeId});
  @override
```

```
  _RatingWidgetState createState() => _RatingWidgetState();
}
class _RatingWidgetState extends State<RatingWidget> {
  int _rating = 0;
  Future<void> submitRating(int value) async {
    final prefs = await SharedPreferences.getInstance();
    final token = prefs.getString('token');
    final response = await http.post(
Uri.parse('https://your-backend-url.com/api/routes/${widget.routeId}
/rate/'),
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer $token',
      },
      body: jsonEncode({'rating': value}),
    );
    if (response.statusCode == 200) {
      setState(() {
        _rating = value;
      });
    }
  }
  @override
  Widget build(BuildContext context) {
    return Row(
      children: List.generate(5, (index) {
        return IconButton(
          icon: Icon(
            index < _rating ? Icons.star : Icons.star_border,
            color: Colors.amber,
          ),
          onPressed: () => submitRating(index + 1),
        );
      }),
    );
  }
}
```

кінець лістингу 3.9

У майбутньому функціонал геолокації та сповіщень можна істотно розширити для підвищення зручності та інформативності користувацького досвіду. Зокрема, доцільно реалізувати фонове відстеження місцезнаходження, що дозволить автоматично повідомляти користувача про наближення до запланованих точок маршруту, навіть коли застосунок згорнутий. Також

перспективним є інтегрування з погодними сервісами (наприклад, OpenWeatherMap), аби надсилати сповіщення про зміни погодних умов у регіоні подорожі.

3.2 Тестування та налагодження інформаційно-комп'ютерної системи

Однією з ключових умов створення якісного мобільного додатку є впровадження процесу тестування на всіх етапах розробки. Тестування дозволяє переконатися в правильності реалізації логіки, адекватності обробки даних та надійності інтеграцій із зовнішніми сервісами. У межах даного проєкту було зосереджено увагу на модульному тестуванні (unit testing), яке дає змогу перевірити окремі частини логіки в ізольованому середовищі.

Для тестування застосовано стандартний інструментарій, що входить до складу Flutter SDK, зокрема пакети flutter_test, test, а також mockito для створення імітованих об'єктів. Завдяки цьому вдалося протестувати функціонал без необхідності звертатися до реального сервера або API Google Maps, що дозволило скоротити час розробки та уникнути неочікуваних результатів під час інтеграції.

Одним із важливих компонентів додатка є сервіс, який відповідає за отримання даних про маршрути з Google Maps Directions API. Цей сервіс формує запит на зовнішній API, отримує відповідь у форматі JSON, після чого витягує з неї необхідну інформацію – зокрема, короткий опис маршруту (summary). У тестуванні було перевірено коректну обробку даних при викликанні Google Maps API (ліст. 3.10), так і стійкість функції до помилок (наприклад, некоректної відповіді API або статус-коду 500).

Лістинг 3.10 – Функція, яка викликає Google Maps API

```
using static System.Net.WebRequestMethods;
using System;
class RouteService
```

```
{
    Future<String> fetchRouteSummary(String origin, String
destination) async {
    final response = await http.get(Uri.parse(

'https://maps.googleapis.com/maps/api/directions/json?origin=$origin
&destination=$destination&key=API_KEY'));
    if (response.statusCode == 200) {
      final data = jsonDecode(response.body);
      return data['routes'][0]['summary'];
    } else
{
    throw Exception('Не вдалося завантажити маршрут');
}
  }
}
```

кінець лістингу 3.10

Щоб протестувати цю функцію, я створював імітацію HTTP-клієнта за допомогою бібліотеки mockito, яка дозволяє симулювати поведінку зовнішніх залежностей, зокрема – мережевих запитів до API. Це дало змогу перевірити правильність логіки взаємодії з сервером без необхідності здійснювати реальні HTTP-запити. У тестах можна задавати як успішні відповіді з очікуваними даними, так і навмисно спровоковані помилки (наприклад, код 500 або 401), щоб упевнитися, що додаток правильно реагує на різні сценарії.

Такий підхід дозволяє значно прискорити тестування, уникнути нестабільності через зовнішні сервіси, а також підвищує загальну надійність та ізольованість юніт-тестів. Це особливо важливо для перевірки функцій, пов’язаних з авторизацією, надсиланням оцінок, збереженням даних користувача тощо. Приклад одного з таких тестів, реалізованого з використанням mockito, подано у лістингу 3.11.

Лістинг 3.11 – Модульне тестування сервісу маршрутів

```
using static System.Net.WebRequestMethods;

@GenerateMocks([http.Client])
void main()
```

```
{
    group('RouteService', () {
           test('повертає назву маршруту при успішному запиті', ()
async {
            final client = MockClient();
            final service = RouteService();
                       when(client.get(any)).thenAnswer((_)async  =>
http.Response(
                jsonEncode({
                'routes': [
                  {'summary': 'Best Route via M06'}
            ]
          }),
          200));
final summary = await service.fetchRouteSummary('Kyiv', 'Lviv');
expect(summary, equals('Best Route via M06'));
    });

test('викидає виключення при помилці запиту', () {
    final client = MockClient();
    final service = RouteService();
    when(client.get(any))
        .thenAnswer((_)async => http.Response('Error', 500));

    expect(() => service.fetchRouteSummary('Kyiv', 'Lviv'),
        throwsException);
});
  });
}
}
```

кінець лістингу 3.11

Ці тести перевіряють два сценарії. Перший – коли API працює коректно і повертає очікуваний маршрут, і другий – коли API не відповідає, або відповідає з помилкою, і ми очікуємо викидання винятку.

Також слід зауважити, що у додатку використовується модель RouteModel, яка відповідає за представлення інформації про маршрут. Ця модель будується на основі JSON-об’єкта, отриманого з API. Важливо перевірити, що перетворення JSON у Dart-об’єкт працює коректно та враховує всі поля (ліст. 3.12).

Лістинг 3.12 – Клас моделі маршруту

```
using System;
class RouteModel
{
    final String name;
  final double distance;

    RouteModel({ required this.name, required this.distance});

  factory RouteModel.fromJson(Map<String, dynamic> json)
    {
        return RouteModel(
          name: json['name'],
          distance: json['distance'].toDouble(),

        );
    }
}
```

кінець лістингу 3.12

В процесі проведення тестування також використовувались юніт-тести. До прикладу юніт-тест на лістингу 3.13 перевіряє коректність роботи конструктора RouteModel.fromJson, який перетворює отримані з API або бази даних JSON-дані у внутрішню модель програми. Тест дозволяє впевнитися, що при передачі правильного JSON-об’єкта всі ключові поля (наприклад, назва маршруту та відстань) будуть коректно зчитані та збережені в об’єкті класу. Такий підхід є особливо важливим на етапі інтеграції з бекендом, оскільки гарантує стабільність структури даних і дозволяє швидко виявити зміни у форматі відповіді сервера або помилки в парсингу. Це сприяє підвищенню надійності додатка та знижує ймовірність появи логічних помилок під час роботи з маршрутами.

Лістинг 3.13 – Юніт-тест для перевірки конструктора з JSON

```
void main()
{
    test('RouteModel.fromJson повертає правильний об’єкт', () {
```

```
          final json = { 'name': 'Подорож в Карпати', 'distance':
220.5};
final route = RouteModel.fromJson(json);

expect(route.name, 'Подорож в Карпати');
expect(route.distance, 220.5);
  });
}
```

кінець лістингу 3.13

Після успішного написання всіх юніт-тестів було здійснено їх запуск за допомогою стандартної команди flutter test. Ця команда дозволяє виконати всі наявні тести у проєкті та перевірити, чи відповідає реалізація очікуваній поведінці. Результати тестування відображаються у консолі з зазначенням кількості пройдених тестів, часу виконання та можливих помилок. У межах проєкту всі тести було виконано успішно, що підтверджує коректну роботу критичних функцій, зокрема моделі маршруту та сервісу побудови маршрутів. Регулярний запуск тестів після змін у коді допомагає підтримувати стабільність системи та оперативно виявляти потенційні помилки.

Використання unit-тестів під час розробки мобільного застосунку дозволяє забезпечити стабільність логіки, скоротити час на пошук помилок та зробити код більш гнучким до змін.

3.3 Розробка бази даних

Розробляючи такий проєкт, як мобільний додаток для планування подорожей, надзвичайно важливо правильно спроєктувати базу даних, оскільки саме вона є основою для зберігання, обробки та взаємодії з усіма ключовими даними користувача. Чітка структура бази даних забезпечує цілісність інформації про маршрути, локації, користувачів, витрати та інші елементи, що є невід’ємною частиною подорожі. Грамотно розроблена схема дозволяє

ефективно працювати з запитами, зменшує дублювання даних, забезпечує швидкий доступ до інформації та полегшує масштабування системи в майбутньому. Тим більше, правильне проєктування сутностей і зв’язків між ними сприяє стабільності додатка, спрощує інтеграцію з API та дає змогу легко реалізовувати нові функції без порушення існуючої логіки.

На основі функціональних вимог мобільного додатку для планування подорожей було спроєктовано реляційну модель бази даних, представлена у вигляді схеми бази даних (рис. 3.1).

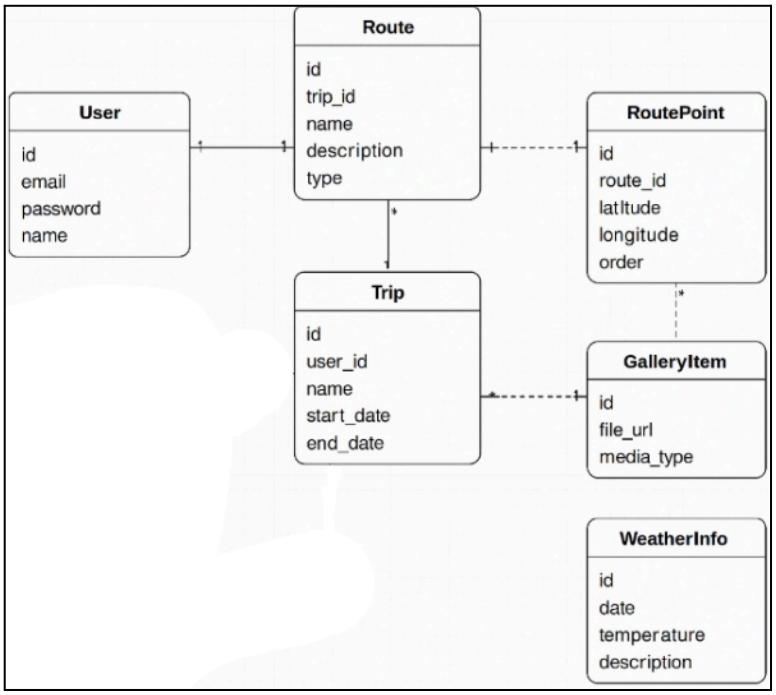

Рисунок 3.1 – Схема бази даних

Дана схема демонструє основні сутності системи, їх атрибути та зв’язки між ними. Метою цього проєктування є забезпечення логічної цілісності, нормалізації даних та підтримки масштабованості системи у процесі її використання та подальшого розвитку.

Основні моделі бази даних:

– User, сутність, що зберігає інформацію про зареєстрованих користувачів. Вона містить атрибути id (унікальний ідентифікатор), email, password та name. Один користувач може мати декілька подорожей (зв’язок 1:∞ з Trip);

– Trip, ключова сутність, що представляє подорож, створену користувачем. Атрибути: id, user_id (зовнішній ключ на User), name, start_date, end_date. Кожна подорож може мати декілька маршрутів, медіафайлів та погодних записів;

– Route, зберігає окремий маршрут у межах подорожі. Атрибути: id, trip_id (зовнішній ключ на Trip), name, description, type. Один маршрут може складатися з декількох точок (RoutePoint);

– RoutePoint, сутність, яка відповідає за збереження координат окремих точок маршруту. Атрибути: id, route_id, latitude, longitude, order. Використовується для побудови точного шляху на карті;

– GalleryItem, містить мультимедійні матеріали, пов’язані з подорожжю (наприклад, фото визначних місць, скріншоти маршруту тощо). Атрибути: id, trip_id, file_url, media_type;

– WeatherInfo, зберігає погодну інформацію, пов’язану з подорожжю або окремим днем перебування. Включає id, trip_id, date, temperature, description. Ці дані можуть використовуватись для формування прогнозів або надсилання сповіщень користувачу.

Після виокремлення основних сутностей постала потреба визначити характер взаємозв’язків між ними, адже саме ці зв’язки забезпечують цілісність моделі та відображають логіку взаємодії об’єктів у межах системи. Без чіткого встановлення типів зв’язків (один до одного, один до багатьох, багато до багатьох) неможливо сформувати повноцінну структуру бази даних, яка б адекватно відображала реальні процеси. Структурні зв’язки між сутностями:

– один користувач (User) може створити декілька подорожей (Trip), зв’язок 1:∞;

– кожна подорож може містити багато маршрутів (Route), зв’язок 1:∞;

– один маршрут складається з набору точок маршруту (RoutePoint), зв’язок 1:∞;

– кожна подорож може мати багато галерейних об’єктів (GalleryItem) та погодних записів (WeatherInfo), зв’язки 1:∞.

Проєктування бази даних дало змогу чітко структурувати інформацію, визначити взаємозв’язки між основними сутностями та закласти основу для надійного і ефективного зберігання даних. Завдяки нормалізації та продуманій логіці зв’язків було досягнуто узгодженості моделі, що сприяє стабільній роботі системи та спрощує її подальше масштабування й модифікацію.

3.4 Захист інформаційно-комп’ютерної системи

У сучасних інформаційних системах питання захисту даних користувачів є надзвичайно важливим, особливо в контексті мобільних додатків, які працюють з персональними обліковими записами, геолокацією, маршрутам подорожей та іншими чутливими даними. У процесі розробки мобільного додатку для планування подорожей були враховані основні принципи інформаційної безпеки: конфіденційність, цілісність та доступність інформації.

Одним із ключових аспектів захисту стала реалізація надійної системи автентифікації користувачів. Дані для входу, зокрема електронна пошта та пароль, проходять перевірку на стороні сервера. Для безпечного зберігання паролів застосовується хешування з використанням стійких до атак алгоритмів, таких як bcrypt, які унеможливлюють зворотне відновлення оригінального паролю навіть у разі компрометації бази даних.

Усі запити між клієнтською частиною додатку та сервером передаються виключно через захищене з’єднання HTTPS, що базується на протоколі SSL/TLS. Це дозволяє запобігти перехопленню інформації під час передачі через мережу, що особливо актуально у випадках використання відкритих Wi-Fi-мереж або мобільного інтернету в роумінгу.

Для управління доступом до даних реалізовано авторизаційний механізм, який забезпечує, щоб користувач мав доступ лише до власних маршрутів, подорожей, файлів галереї та погодних записів. Кожна дія, яка змінює стан даних (створення, редагування, видалення), супроводжується перевіркою ідентифікатора користувача (user_id), що дозволяє уникнути несанкціонованих змін або витоку інформації між акаунтами.

Крім того, для запобігання втраті або пошкодженню інформації, передбачено резервне копіювання бази даних на сервері. Регулярне створення бекапів дозволяє відновити систему в разі збою або технічної помилки, забезпечуючи високу доступність сервісу навіть у критичних ситуаціях.

Важливою частиною інформаційної безпеки є також захист від атак типу SQL-ін’єкцій. Для цього всі запити до бази даних реалізовані з використанням параметризованих SQL-операторів, або ж із застосуванням ORM-інструментів, які автоматично фільтрують вхідні дані. Це запобігає потенційним спробам змінити логіку запиту через змінні, що вводяться користувачем.

Таким чином, реалізовані механізми безпеки дозволяють гарантувати збереження персональних і маршрутних даних, запобігти несанкціонованому доступу, забезпечити захист комунікацій між клієнтом і сервером, а також підтримати стабільну роботу системи в умовах несподіваних збоїв або навмисного втручання.

Висновки до розділу 3

У результаті практичної реалізації було створено повнофункціональний мобільний додаток для планування подорожей з підтримкою геолокації, побудови маршрутів, авторизації та збереження даних. Архітектура побудована на Flutter з дотриманням принципів модульності та масштабованості. Базу даних спроєктовано з урахуванням нормалізації та чітких зв’язків між сутностями, що забезпечує стабільність і гнучкість. Проведено модульне

тестування, яке підтвердило коректність роботи основних компонентів. Особливу увагу приділено захисту даних – реалізовано автентифікацію, HTTPS, шифрування та контроль доступу.

ВИСНОВКИ

Мобільний додаток для планування особистих подорожей був успішно реалізований, що свідчить про успішне виконання всіх поставлених, на період роботи підготовки, завдань.

Було проведено детальний аналіз предметної області, розглянуто вплив пандемії COVID-19 та повномасштабного вторгнення на туристичну сферу, що дозволило глибше зрозуміти потреби сучасних користувачів і специфіку ринку. Також в процесі вивчення сучасного стану проблеми було проаналізовано існуючі рішення на ринку, оцінено переваги та недоліки кожного з них.

Було визначено повний перелік функціональних та нефункціональних вимог до мобільного додатку, що дало змогу сформувати чітке бачення продукту та забезпечити його релевантність для потенційних користувачів. Для візуалізації роботи системи було створено діаграму використання, яка відображає основні сценарії взаємодії користувача з додатком та ілюструє ключові функціональні можливості системи.

Проведено аналіз існуючих інструментів для розробки, що дозволило обрати оптимальний стек технологій для реалізації кросплатформенного мобільного застосунку. Для роботи над мобільним додатком було обрано Flutter, Google Maps API, MySQL та GitHub.

Розроблено мобільний додаток, який враховує визначені вимоги, реалізавоно ключовий функціонал та забезпечено зручний і інтуїтивно зрозумілий інтерфейс для користувачів.

Після розробки було проведене детальне тестування функціоналу, перевірено роботу API, виявлено й усунено помилки та недоліки, що дозволило покращити стабільність і якість системи.

Спроєктовано базу даних для зберігання інформації про користувачів та їхні подорожі, з урахуванням потреб масштабованості й ефективного доступу до даних.

Здійснено впровадження базових механізмів захисту інформаційної системи, що забезпечує цілісність, доступність і конфіденційність користувацьких даних відповідно до сучасних вимог безпеки.

У перспективі функціонал геолокації та системи сповіщень має значний потенціал для подальшого розвитку з метою підвищення зручності та ефективності користування додатком. Зокрема, доцільним є впровадження фонового відстеження місцезнаходження для автоматичного інформування користувача про наближення до запланованих точок маршруту навіть у фоновому режимі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. International Tourist Arrivals Could Fall by 20-30 % in 2020 – UN Tourism. UN Tourism – Bringing the world closer. URL: https://www.unwto.org/new/ internati onal-tourism-arrivals-could-fall-in-2020 (date of access: 04.02.2025).

2. Світовий туризм відновився після ковіду: як змінювалась кількість туристичних поїздок. Слово і Діло. URL: https://www.slovoidilo.ua/20 25/01/28/infografika/svit/svitovyj-turyzm-vidnovyvsya-pislya-kovidu-yak-zminyuval as-kilkist-turystychnyx-poyizdok (дата звернення: 11.02.2025).

3. TripIt – Online travel itinerary and trip planner. URL: https://www.tripit. com/web/blog/news-culture/whats-new-tripit-app-2022 (date of access: 15.02.2025).

4. Google Trips. A step in the right direction, but only a step. Revenue Hub. URL: https://revenue-hub.com/google-trips-step-right-direction-step/ (date of access: 24.02.2025).

5. Save 23 % on Roadtrippers – #1 Road Trip Planning App. URL: https://es capecampervans.com/blog/roadtrippers-trip-planning-app/ (date of access: 03.03.2025).

6. Інструкція, як будувати UML-діаграми – DOU. URL: https://dou.ua/for ums/topic/40575/ (date of access: 09.03.2025).

7. Flutter documentation. Docs – Flutter. URL: https://docs.flutter.dev/ (date of access: 13.03.2025).

8. Kotlin Docs. Kotlin Help. URL: https://kotlinlang.org/docs/home.html (date of access: 18.03.2025).

9. Swift.org. URL: https://www.swift.org/documentation/ (date of access: 22.03.2025).

10. Api Google Maps: інтеграція сервісу карт на сайт – студія-Brander. Створення і розробка інтернет-магазинів від Brander. URL: https://brander .ua/technologies/api-google-maps (дата звернення: 27.03.2025).

11. API Docs. Mapbox. URL: https://docs.mapbox.com/api/overview/ (date of access: 12.04.2025).

12. OpenStreetMap Wiki. URL: https://wiki.openstreetmap.org/ (date of access: 15.04.2025).

13. MySQL Documentation. URL: https://dev.mysql.com/doc/ (date of access: 21.04.2025).

14. PostgreSQL – Documentation. PostgreSQL – The world's most advanced open source database. URL: https://www.postgresql.org/docs/ (date of access: 25.04.2025).

15. GitHub.com Help Documentation. GitHub Docs. URL: https://docs.github.com/en (date of access: 30.04.2025).