Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 - «Інженерія програмного забезпечення»

На тему: Розробка інформаційно-аналітичної системи управління рухом міського громадського транспорту

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-22-2
_____ / І.М. Брюховецький /

Керівник кваліфікаційної роботи _____ / А.М. Гриценко /

Завідувач кафедри _____ / А.М. Стрюк /

Кривий Ріг

2026

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Освітньо-кваліфікаційний рівень: бакалавр

Спеціальність: 121 - Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А.М. Стрюк

«___»_____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-22-2 Брюховецькому Ігору Миколайовичу

1. Тема: Розробка інформаційно-аналітичної системи управління рухом громадського транспорту затверджено наказом по КНУ №285 від «28» травня 2026 р.

2. Термін подання студентом закінченого проекту «06» червня 2026 р.

3. Вихідні дані по роботі: *Пояснювальна записка: 113 сторінки, 25 рисунків, 2 додатки, 25 використаних у роботі джерел*

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): Аналіз існуючих програмних рішень в галузі управління рухом громадського транспорту. Проектування та розробка інформаційно аналітичної системи. Тестування системи управління.

5. Перелік графічного демонстраційного матеріалу: Загальна схема організації руху міського громадського транспорту. Порівняння таблиця існуючих інформаційних систем громадського транспорту. Загальний алгоритм роботи інформаційно-аналітичної системи. Архітектура системи. Структура бази даних. Діаграми використання. Приклади роботи системи управління (інтерфейси користувача).

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Формулювання мети та задач роботи	18.02.2026-20.02.2026
2	Аналіз інформаційних джерел	21.02.2026-09.03.2026
3	Визначення вимог до програмного забезпечення	10.03.2026-18.03.2026
4	Розробка функціональної схеми	19.03.2026-23.03.2026
5	Розробка алгоритмів	24.03.2026-31.03.2026
6	Розробка бази даних	01.04.2026-10.04.2026
7	Розробка програмного забезпечення	10.04.2026-13.05.2026
8	Тестування програмного забезпечення	14.05.2026-20.05.2026
9	Оформлення пояснювальної записки	21.05.2026-31.06.2026
10	Розробка демонстраційних матеріалів	01.06.2026-10.06.2026

Дата видачі завдання: « 18 » лютого 2026 р.

Студент _____ / І.М. Брюховецький /

Керівник роботи _____ / А.М. Гриценко /

ЗМІСТ

РЕФЕРАТ

ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА, МІСЬКИЙ ГРОМАДСЬКИЙ ТРАНСПОРТ, GPS-МОНІТОРИНГ, ДИСПЕТЧЕРСЬКЕ УПРАВЛІННЯ, ВЕБЗАСТОСУНОК, МОБІЛЬНИЙ ЗАСТОСУНОК, ІМІТАЦІЯ РУХУ, АНАЛІТИКА.

Пояснювальна записка: 113 с., 21 табл., 25 рис., 3 дод., 25 джерел.

Об’єктом розробки є процеси моніторингу та диспетчерського управління рухом міського громадського транспорту.

Метою роботи є розробка інформаційно-аналітичної системи, що забезпечує ведення транспортних даних, отримання GPS-координат, відображення транспортних засобів на карті, контроль рейсів і формування аналітичних показників.

У роботі проаналізовано предметну область та існуючі транспортні сервіси, сформовано вимоги, розроблено функціональну й архітектурну схеми, алгоритми обробки координат та реляційну базу даних. Реалізовано серверну частину, вебзастосунок, мобільний застосунок водія, інформаційно-аналітичну підсистему та модуль імітації руху.

Система підтримує керування маршрутами, зупинками, водіями, транспортними засобами й рейсами, передавання GPS-координат, відображення транспорту на карті, визначення запізнень, відхилень від маршруту, тривалих зупинок і втрати зв’язку. Аналітична підсистема формує узагальнені показники виконання рейсів і зареєстрованих подій.

Проведене функціональне, інтеграційне та системне тестування підтвердило працездатність основних компонентів і відповідність реалізованих функцій сформованим вимогам. Модуль імітації дозволив перевірити роботу системи без використання реального транспортного засобу. Результати розробки можуть бути використані як основа для подальшої інтеграції з реальними GPS-трекерами та міськими транспортними сервісами.

ABSTRACT

INFORMATION AND ANALYTICAL SYSTEM, URBAN PUBLIC TRANSPORT, GPS MONITORING, DISPATCH MANAGEMENT, WEB APPLICATION, MOBILE APPLICATION, TRAFFIC SIMULATION, ANALYTICS.

Explanatory note: 124 pages, 21 tables, 25 figures, 3 appendices, 25 references.

The object of development is the monitoring and dispatch management of urban public transport.

The aim is to develop a system for transport data management, GPS coordinate acquisition, vehicle visualization, trip monitoring and analytical indicator generation.

The work includes domain analysis, requirements definition, system architecture, algorithms and database design. A server application, web application, driver mobile application, analytical subsystem and traffic simulation module were implemented.

The system supports route and trip management, GPS data transmission, vehicle visualization and detection of delays, route deviations, prolonged stops and connection loss. The analytical subsystem generates summarized trip and event indicators.

Functional, integration and system testing confirmed the operability of the main components. The simulation module made it possible to verify the system without using a real vehicle. The developed prototype can serve as a basis for integration with real GPS trackers and municipal transport services.

The system can be used by transport companies, dispatch centers and local authorities.

ВСТУП

Міський громадський транспорт є важливою складовою транспортної інфраструктури, від стабільної роботи якої залежить мобільність населення та доступність основних міських об’єктів. Зростання інтенсивності перевезень підвищує вимоги до дотримання розкладів, оперативності диспетчерського управління та якості інформування пасажирів.

Традиційний диспетчерський контроль часто ґрунтується на планових графіках, телефонному зв’язку з водіями та ручному опрацюванні даних. Такий підхід ускладнює своєчасне виявлення затримок, відхилень від маршруту та тривалих зупинок, а також обмежує можливості аналізу виконаних рейсів.

Одним із напрямів удосконалення транспортного управління є використання інтелектуальних транспортних систем, які поєднують інформаційні, телекомунікаційні та навігаційні технології. Європейська комісія розглядає їх упровадження як засіб підвищення ефективності, безпеки та екологічності транспорту [1].

Використання глобальних навігаційних супутникових систем дає змогу отримувати координати транспортних засобів, відображати їх на електронній карті, порівнювати фактичний рух із маршрутом і розкладом та зберігати історію рейсів. Застосування відстеження в реальному часі дозволяє транспортним операторам підвищувати ефективність обслуговування пасажирів [2].

Актуальність теми полягає в необхідності створення системи, яка забезпечуватиме централізоване ведення даних про маршрути, зупинки, транспортні засоби, водіїв і рейси, а також моніторинг руху та формування аналітичних показників. Диспетчер зможе контролювати транспорт на карті та виявляти проблемні ситуації, адміністратор — керувати користувачами й довідковими даними, водій — передавати GPS-координати через мобільний застосунок, а пасажир — переглядати актуальну інформацію про рух.

Об’єктом розробки є процеси інформаційного забезпечення, моніторингу та диспетчерського управління рухом міського громадського транспорту.

Предметом розробки є методи та програмні засоби збирання, передавання, зберігання, обробки й візуалізації даних про рух транспортних засобів.

Метою кваліфікаційної роботи є розробка інформаційно-аналітичної системи управління рухом міського громадського транспорту, яка забезпечує ведення транспортних даних, отримання GPS-координат, відображення положення транспортних засобів, контроль рейсів і формування аналітичної інформації.

Для досягнення поставленої мети необхідно виконати такі завдання:

– дослідити особливості організації та диспетчерського управління громадським транспортом;

– проаналізувати існуючі інформаційні системи та визначити їхні переваги й недоліки;

– сформувати вимоги та сценарії використання системи;

– розробити функціональну схему, архітектуру й основні алгоритми;

– спроєктувати базу даних;

– реалізувати серверну частину та програмний API;

– розробити вебзастосунок для адміністратора, диспетчера й пасажира;

– розробити мобільний застосунок водія для передавання GPS-координат;

– реалізувати модуль імітації руху;

– провести тестування розробленого програмного забезпечення.

Під час виконання роботи використовуються методи аналізу предметної області, об’єктно-орієнтованого проєктування, моделювання програмних систем, проєктування реляційних баз даних і функціонального тестування. Для перевірки підсистеми моніторингу застосовується імітаційне моделювання руху транспортного засобу.

Практичне значення роботи полягає у створенні програмного прототипу для автоматизації диспетчерського контролю. Система забезпечуватиме централізоване зберігання транспортних даних, відображення місцеположення транспортних засобів, контроль виконання рейсів, виявлення відхилень і формування аналітичних звітів.

Подальший розвиток системи може передбачати інтеграцію з реальним GPS-обладнанням, сервісами прогнозування часу прибуття, електронними інформаційними табло та міськими платформами відкритих транспортних даних.

Галуззю застосування результатів роботи є діяльність транспортних підприємств, диспетчерських центрів, органів місцевого самоврядування та розробників інформаційних сервісів громадського транспорту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Особливості організації руху міського громадського транспорту

Міський громадський транспорт забезпечує регулярне перевезення населення між житловими районами та основними об’єктами міської інфраструктури. У межах цієї роботи розглядається наземний громадський транспорт, рух якого здійснюється за визначеними маршрутами та розкладами.

Відповідно до Закону України «Про автомобільний транспорт», міський автобусний маршрут загального користування призначений для регулярних пасажирських перевезень у межах населеного пункту [3]. Організація таких перевезень передбачає формування маршрутної мережі, визначення зупинок і розкладів, призначення транспортних засобів та водіїв, а також контроль виконання рейсів.

Маршрут визначає послідовність руху між початковою, проміжними та кінцевою зупинками. Для нього задаються номер, назва, напрямок, довжина, орієнтовна тривалість і перелік зупинок. Відомості про схему маршруту, розклад та умови перевезень фіксуються у відповідній документації [4]. В інформаційній системі ці дані доцільно зберігати в електронному вигляді.

Кожна зупинка повинна мати унікальний ідентифікатор, назву та географічні координати. Координати використовуються для відображення маршруту на карті, визначення положення транспортного засобу та фіксації прибуття до контрольної зони.

Розклад містить запланований час початку рейсу, проходження зупинок і завершення поїздки. Конкретне виконання маршруту транспортним засобом розглядається як рейс, що пов’язується з маршрутом, водієм, транспортним засобом і фактичними даними руху.

Фактичний рух може відрізнятися від запланованого через затори, аварії, ремонтні роботи, погодні умови або технічні несправності. Тому для оперативного управління необхідно отримувати актуальні дані про місцеположення транспорту та порівнювати їх із маршрутом і розкладом.

Контроль за виконанням рейсів здійснює диспетчер. Він відстежує положення транспортних засобів, затримки, тривалі зупинки та відхилення від маршруту. Традиційний контроль за допомогою телефонного або радіозв'язку не забезпечує постійного отримання точних даних і потребує ручної реєстрації інформації.

Для автоматизації моніторингу використовуються глобальні навігаційні супутникові системи. Мобільний застосунок водія або бортове обладнання періодично передає серверу координати, час вимірювання, швидкість та ідентифікатор активного рейсу. Сервер перевіряє й зберігає дані, відображає транспорт на карті та визначає відхилення від запланованого руху.

Для стандартизованого подання транспортних даних застосовується специфікація GTFS. Її статична частина описує маршрути, рейси, зупинки, розклади та календарі роботи [5]. GTFS Realtime доповнює ці відомості актуальними позиціями транспортних засобів, змінами рейсів і повідомленнями про порушення руху [6]. Основні принципи GTFS доцільно врахувати під час проєктування бази даних системи.

У процесі організації руху взаємодіють організатор перевезень, транспортне підприємство, адміністратор системи, диспетчер, водій і пасажир. Їхні основні функції наведено в таблиці 1.1.

Для функціонування інформаційно-аналітичної системи необхідно використовувати планові, оперативні та аналітичні дані. До планових належать маршрути, зупинки, розклади, транспортні засоби, водії та заплановані рейси. Оперативні дані формуються під час руху та включають координати, час їх отримання, швидкість, статус рейсу й зафіксовані відхилення. Аналітичні дані утворюються після обробки накопиченої

інформації та можуть містити середню затримку, фактичну тривалість рейсу, кількість виконаних поїздок та інші показники.

Таблиця 1.1 – Учасники процесу організації руху громадського транспорту

Учасник	Основні функції
Організатор перевезень	Формування маршрутної мережі, визначення умов перевезень і контроль якості транспортного обслуговування
Транспортне підприємство	Надання транспортних засобів, призначення водіїв, організація виконання рейсів
Адміністратор системи	Керування користувачами, ролями, маршрутами, зупинками та іншими довідковими даними
Диспетчер	Контроль руху транспортних засобів, виявлення відхилень та опрацювання подій
Водій	Виконання призначеного рейсу та передавання даних про поточне місцеположення
Пасажир	Перегляд маршрутів, зупинок, розкладів і поточного положення транспорту

Загальну взаємодію учасників і програмних компонентів доцільно подати у вигляді схеми, наведеної на рисунку 1.1.

На схемі необхідно відобразити транспортне підприємство, адміністратора, диспетчера, водія та пасажира. Водій через мобільний застосунок передає GPS-координати серверній частині. Сервер зберігає дані у базі, порівнює фактичний рух із маршрутом і розкладом та передає результати до вебзастосунку. Диспетчер отримує оперативні дані й повідомлення про відхилення, адміністратор керує довідковою інформацією, а пасажир переглядає доступні відомості про рух транспорту.

Під час проєктування системи необхідно враховувати, що якість GPS-даних залежить від умов приймання супутникового сигналу та стабільності мобільного зв’язку. Координати можуть надходити із затримкою, похибкою або повторюватися. Тому серверна частина повинна перевіряти час

отримання, допустимий діапазон координат і належність точки до зони маршруту.

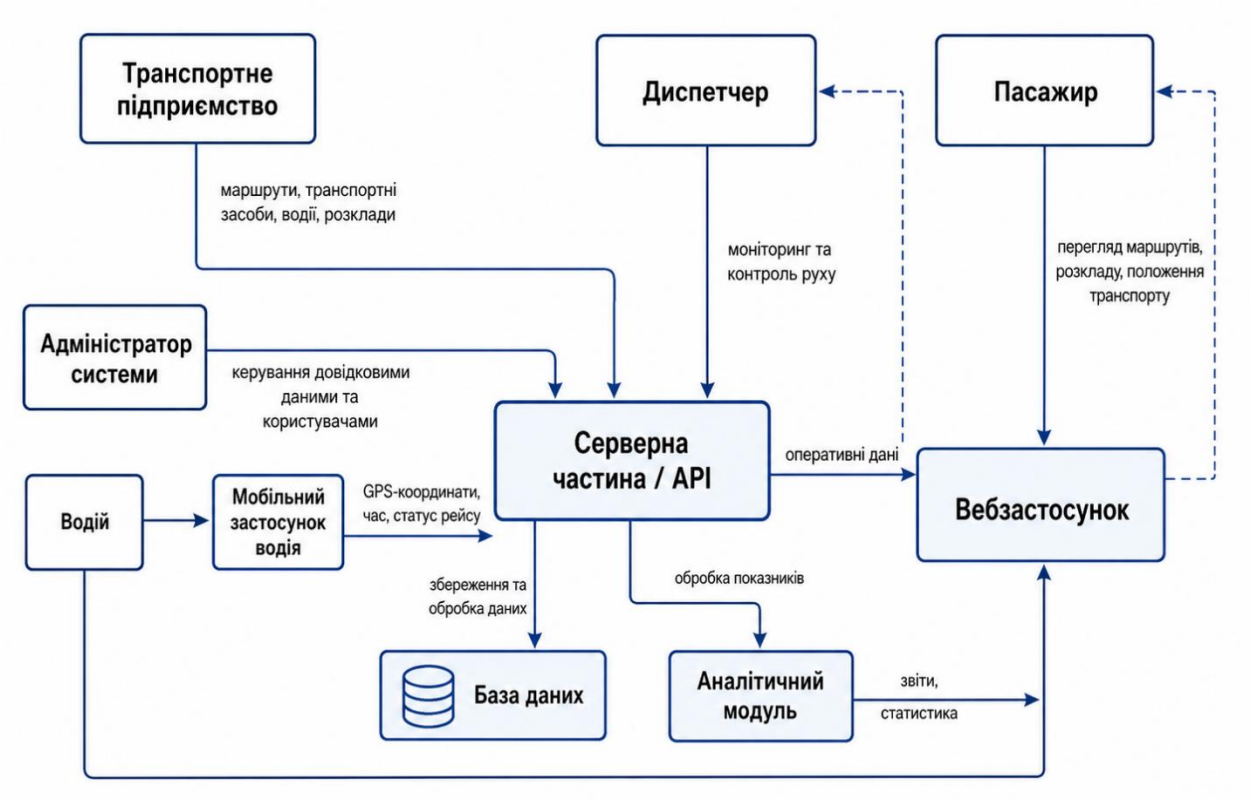


Рисунок 1.1 – Загальна схема організації руху міського громадського транспорту

Таким чином, організація руху міського громадського транспорту поєднує планування маршрутів і розкладів, призначення транспортних засобів та водіїв, виконання рейсів, диспетчерський контроль і надання інформації пасажирам. Автоматизація цих процесів потребує централізованого зберігання даних, отримання GPS-координат і порівняння фактичних показників із плановими. Зазначені особливості визначають основні вимоги до інформаційно-аналітичної системи, що розробляється.

1.2 Аналіз проблем управління рухом громадського транспорту

Ефективність міського громадського транспорту залежить від регулярності руху, дотримання розкладу, оперативності диспетчерського

контролю та доступності інформації для пасажирів. На виконання рейсів впливають затори, аварії, ремонтні роботи, погодні умови, технічні несправності й нерівномірний пасажиропотік.

Основними проблемами є запізнення, порушення інтервалів і відхилення від маршруту. Для їх виявлення необхідно отримувати актуальні GPS-координати, порівнювати фактичний рух із запланованим і формувати повідомлення лише після підтвердження відхилення кількома послідовними точками.

Якість моніторингу залежить від стабільності GPS-сигналу та мобільного з’єднання. Координати можуть надходити із затримкою або тимчасово не передаватися, тому система повинна зберігати час останнього оновлення, відрізняти тривалу зупинку від втрати зв’язку та повідомляти диспетчера про відсутність даних.

Традиційний телефонний або радіозв’язок не забезпечує постійного контролю всіх транспортних засобів. Електронна карта й автоматичне передавання координат зменшують кількість ручних операцій та дозволяють швидко виявляти проблемні рейси.

Розрізненість даних про маршрути, розклади, водіїв, транспортні засоби та рейси ускладнює аналіз. Централізована база даних дає змогу пов’язати координати з конкретним рейсом і порівнювати фактичні показники з плановими.

Накопичення історії руху необхідне для розрахунку кількості виконаних рейсів, середнього запізнення, тривалості поїздок, випадків втрати зв’язку та відхилень від маршруту. Подання результатів у вигляді таблиць і діаграм спрощує виявлення повторюваних проблем.

Важливим є також інформування пасажирів. Статичний розклад не враховує затримки та зміни руху, тому європейська політика міської мобільності передбачає розвиток доступу до актуальних транспортних даних [7]. Інформація в реальному часі допомагає точніше планувати поїздки [8], але

пасажирський інтерфейс не повинен містити службових або персональних відомостей.

Основні проблеми управління рухом та засоби їх вирішення наведено в таблиці 1.2.

Таблиця 1.2 – Проблеми управління рухом міського громадського транспорту

Проблема	Наслідки	Засоби вирішення в системі
Відхилення від розкладу	Збільшення часу очікування, порушення пересадок, нерегулярність руху	Порівняння планового та фактичного часу, фіксація запізнень
Нерівномірні інтервали	Одночасне прибуття кількох транспортних засобів і тривалі перерви	Аналіз координат та інтервалів між транспортними засобами
Відхилення від маршруту	Пропуск зупинок, порушення схеми руху	Порівняння GPS-координат із геометрією маршруту
Тривала зупинка	Затримка рейсу, можливість технічної несправності	Контроль зміни координат і тривалості перебування в одній зоні
Втрата зв’язку	Відсутність актуального положення транспорту	Контроль часу останнього оновлення та повідомлення диспетчера
Розрізненість даних	Складність контролю й аналізу виконаних рейсів	Використання централізованої бази даних
Ручний диспетчерський контроль	Високе навантаження на диспетчера, ризик помилок	Електронна карта, автоматичне визначення подій і сповіщення
Відсутність історичної аналітики	Неможливість оцінити регулярність та знайти повторювані проблеми	Збереження історії руху, формування показників і звітів
Недостатнє інформування пасажирів	Невизначеність часу очікування та планування поїздки	Відображення маршрутів, розкладів і поточного положення транспорту

Розроблювана система повинна не лише відображати транспорт на карті, а й визначати активний рейс, порівнювати фактичний рух із плановим, реєструвати проблемні події та зберігати результати для подальшого аналізу. Для тестування цих функцій передбачається імітатор, який відтворює нормальний рух, запізнення, тривалу зупинку, відхилення від маршруту та втрату зв’язку.

Таким чином, основними проблемами управління міським громадським транспортом є відхилення від розкладу, порушення регулярності, недостатня оперативність диспетчерського контролю, нестабільність передавання GPS-даних, розрізненість інформації та відсутність засобів історичного аналізу. Усунення зазначених недоліків потребує комплексної системи, яка об’єднує планові, оперативні та аналітичні дані й надає окремі програмні засоби адміністратору, диспетчеру, водію та пасажиру.

1.3 Аналіз існуючих інформаційних систем

Для визначення функціональних можливостей розроблюваної інформаційно-аналітичної системи доцільно проаналізувати наявні програмні рішення у сфері міського громадського транспорту. Такі системи відрізняються призначенням, набором функцій і категоріями користувачів. Одні з них орієнтовані переважно на пасажирів і забезпечують планування поїздок, інші можуть використовуватися для відображення транспортних даних, отриманих від міських операторів.

Для аналізу обрано сервіси EasyWay, Moovit, Google Maps із підтримкою Google Transit та міський застосунок «Київ Цифровий». Ці рішення містять функції перегляду маршрутів, зупинок, розкладів, прогнозованого часу прибуття або поточного положення транспорту.

EasyWay є сервісом для перегляду маршрутів міського громадського транспорту. Він підтримує пошук маршруту між заданими точками, відображення зупинок і перегляд транспорту, який проходить через вибрану

зупинку [9]. Сервіс доступний у вигляді вебверсії та мобільного застосунку й підтримує низку міст України.

Інтерфейс EasyWay побудований навколо електронної карти. На ній відображаються лінії маршрутів, зупинки та, за наявності відповідних даних, поточне положення транспортних засобів. Користувач може вибрати маршрут і переглянути послідовність його проходження. Такий спосіб подання є наочним і може бути врахований під час реалізації пасажирської частини розроблюваної системи.

Moovit є міжнародною платформою міської мобільності. Сервіс забезпечує планування поїздок, перегляд маршрутів і зупинок, а також отримання прогнозованого часу прибуття транспорту. Користувачам можуть надаватися повідомлення про затримки, ремонтні роботи та інші зміни транспортного обслуговування [10].

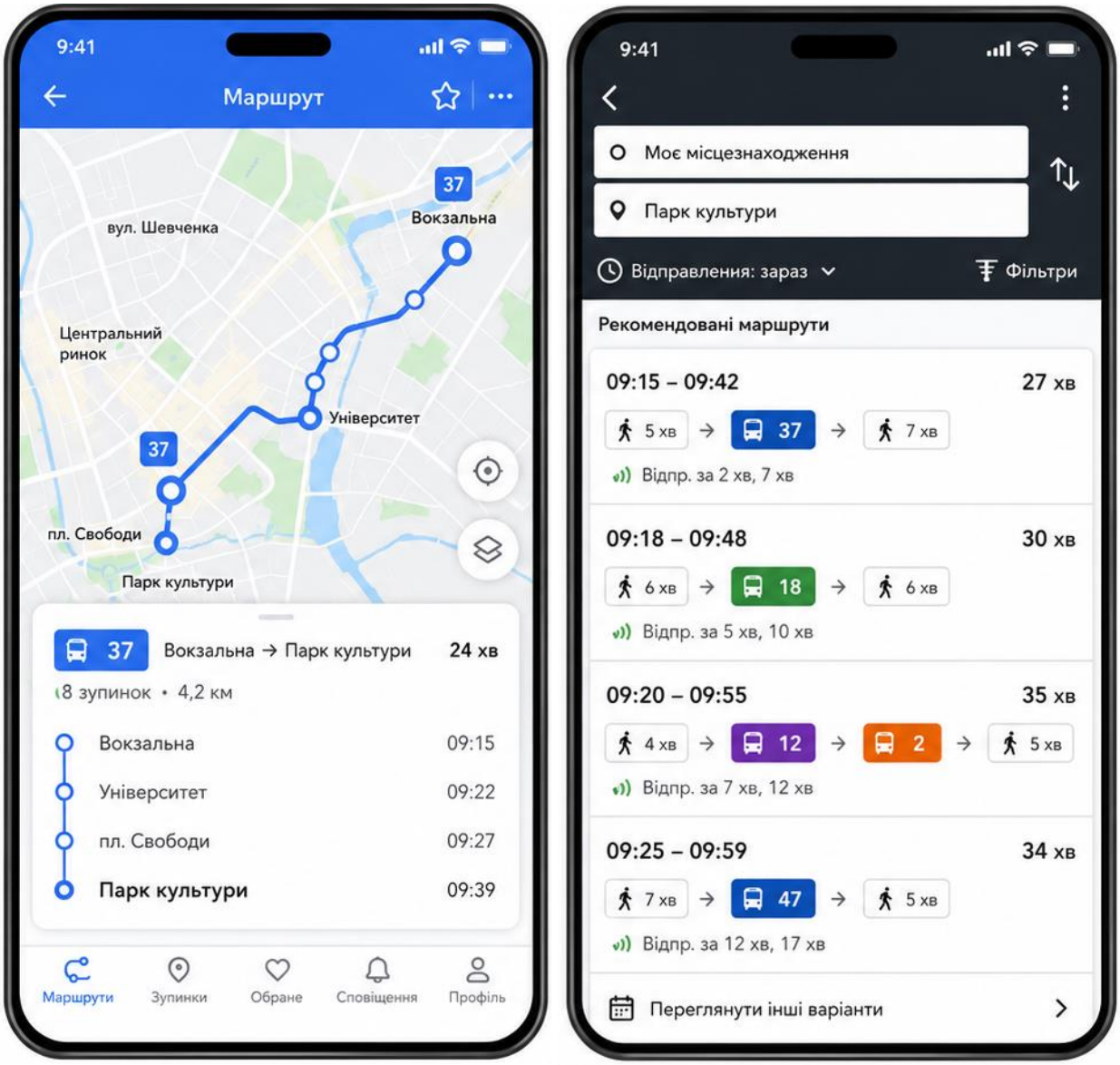
Особливістю Moovit є підтримка мультимодальних маршрутів. Система може поєднувати піше пересування з автобусом, трамваєм, метрополітеном та іншими доступними видами транспорту. Для кожного варіанта відображаються орієнтовна тривалість поїздки, кількість пересадок і час відправлення.

Приклади інтерфейсів EasyWay та Moovit наведено на рисунку 1.2.

Google Maps є універсальним картографічним сервісом, який підтримує побудову автомобільних, пішохідних, велосипедних і транспортних маршрутів. Інформація про громадський транспорт інтегрується за допомогою Google Transit. Залежно від доступності даних користувач може переглядати маршрути, зупинки, запланований або актуальний час відправлення [11].

Перевагою Google Maps є поєднання даних громадського транспорту із загальною картою міста, пошуком адрес і об’єктів. Під час планування поїздки система відображає декілька варіантів маршруту, час у дорозі, пересадки та піші ділянки. Водночас Google Maps не є внутрішньою системою транспортного підприємства, а лише відображає дані, які надаються транспортними операторами.

«Київ Цифровий» є міським мобільним застосунком, який об’єднує транспортні та інші електронні послуги столиці. У транспортній частині застосунку користувач може переглядати маршрути, отримувати інформацію про прибуття міського транспорту та користуватися цифровими транспортними сервісами [12].



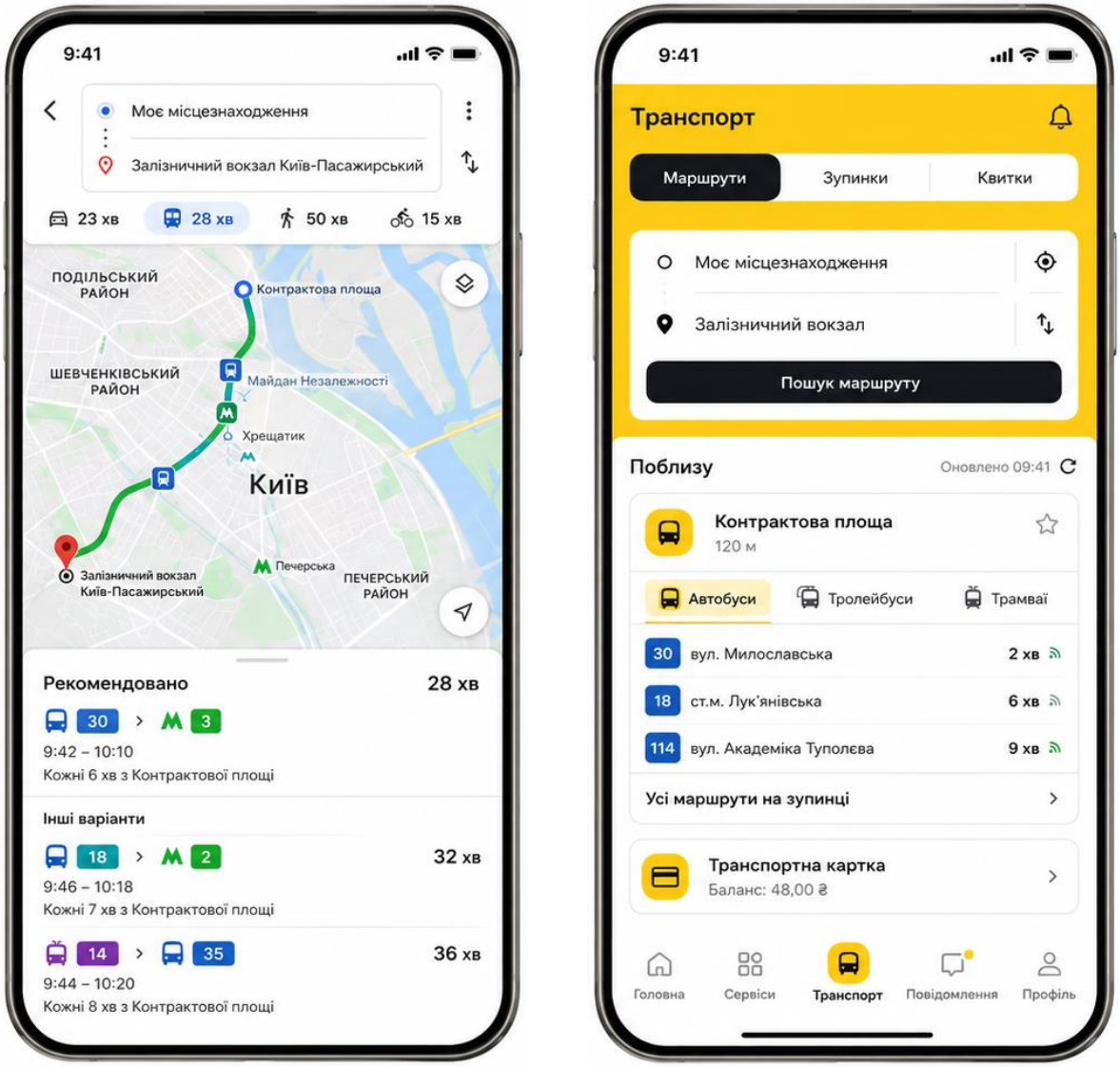
а) інтерфейс EasyWay; б) інтерфейс Moovit

Рисунок 1.2 – Інтерфейси сервісів планування поїздок громадським транспортом

Перевагою «Києва Цифрового» є інтеграція декількох міських послуг в одному інтерфейсі. Транспортна інформація подається у компактному вигляді,

пристосованому до використання на мобільному пристрої. Такий підхід доцільно врахувати під час розроблення пасажирського інтерфейсу системи.

Приклади інтерфейсів Google Maps та «Київ Цифровий» наведено на рисунку 1.3.



а) маршрут громадським транспортом у Google Maps

б) транспортний розділ застосунку «Київ Цифровий»

Рисунок 1.3 – Інтерфейси картографічного та міського транспортного сервісів

Наведені приклади демонструють, що для пасажирського інтерфейсу найбільш важливими є електронна карта, пошук маршруту, відображення

зупинок, час прибуття та повідомлення про зміни руху. При цьому службові функції транспортного підприємства зазвичай приховані від пасажира або реалізовані в окремих програмних продуктах.

Результати порівняння розглянутих систем наведено в таблиці 1.3. Знак «+» означає наявність функції у загальнодоступному інтерфейсі, знак «–» — її відсутність, а знак «±» — залежність функції від міста та доступності даних транспортного оператора.

Порівняння показує, що розглянуті системи добре виконують функції пасажирського інформування. Вони дають змогу переглядати маршрути, зупинки, прогнозований час прибуття та, за наявності відповідних даних, поточне положення транспортних засобів.

Загальнодоступні інтерфейси розглянутих систем переважно орієнтовані на пасажирів і не охоплюють внутрішнє управління транспортним підприємством. У них відсутні функції обліку водіїв і транспортних засобів, призначення рейсів, контролю передавання GPS-координат та аналізу історії руху.

У розроблюваній системі доцільно поєднати переваги аналогів: картографічне відображення маршрутів і транспорту, повідомлення про затримки та простий інтерфейс для кожної категорії користувачів. Вебзастосунок забезпечуватиме роботу адміністратора, диспетчера й пасажира, а мобільний застосунок водія — передавання GPS-координат.

Серверна частина зберігатиме маршрути, зупинки, розклади, транспортні засоби, водіїв, рейси та історію руху. Прив’язка координат до конкретного рейсу дозволить контролювати виконання маршруту, визначати відхилення від розкладу та формувати аналітичні показники. Для тестування цих функцій передбачено модуль імітації руху.

Таблиця 1.3 – Порівняння існуючих інформаційних систем громадського транспорту

Критерій	EasyWay	Moovit	Google Maps	Київ Цифровий	Розроб-лювана система
Перегляд маршрутів і зупинок	+	+	+	+	+
Побудова пасажирського маршруту	+	+	+	±	+
Відображення транспорту на карті	±	±	±	±	+
Дані про прибуття в реальному часі	±	+	±	+	+
Повідомлення про зміни руху	±	+	±	+	+
Керування маршрутами та зупинками	–	–	–	–	+
Облік водіїв і транспортних засобів	–	–	–	–	+
Призначення транспортного засобу на рейс	–	–	–	–	+
Передавання GPS-даних водієм	–	–	–	–	+
Диспетчерський контроль рейсів	–	–	–	–	+
Виявлення відхилень від графіка	–	–	–	–	+
Історична аналітика за рейсами	–	–	–	–	+
Розмежування доступу за ролями	–	–	–	–	+
Імітація руху для тестування	–	–	–	–	+

Отже, для завдань транспортного підприємства необхідна спеціалізована система, яка поєднує диспетчерський контроль, GPS-моніторинг, рольовий доступ, збереження історії рейсів і формування аналітичних звітів.

1.4 Обґрунтування розробки та постановка задачі

Проведений аналіз показав, що ефективне управління міським громадським транспортом потребує централізованого зберігання та узгодження даних про маршрути, зупинки, розклади, транспортні засоби, водіїв, рейси й координати руху.

Основними проблемами є відхилення від розкладу та маршруту, нерівномірні інтервали, тривалі зупинки, втрата зв’язку й несвоєчасне отримання інформації диспетчером. Розрізнене зберігання відомостей також ускладнює контроль фактичного виконання рейсів і подальший аналіз.

Наявні пасажирські сервіси забезпечують перегляд маршрутів, зупинок і актуальної транспортної інформації [9–12], але не охоплюють повний цикл внутрішнього управління транспортним підприємством. Тому доцільною є розробка спеціалізованої системи, що поєднує планування, GPS-моніторинг, диспетчерський контроль та аналітику.

Адміністратор повинен керувати користувачами й довідковими даними, диспетчер — створювати та контролювати рейси, водій — передавати GPS-координати через мобільний застосунок, а пасажир — переглядати маршрути, розклади та положення транспорту.

Система має клієнт-серверну архітектуру й складається з вебзастосунку, мобільного застосунку водія, серверної частини, бази даних, аналітичного модуля та імітатора руху. Сервер обробляє запити, перевіряє права доступу, зберігає GPS-дані та пов’язує їх із конкретним рейсом, водієм і транспортним засобом.

Модуль імітації руху призначений для перевірки нормального руху, запізнення, тривалої зупинки, відхилення від маршруту та втрати зв’язку без використання реального транспортного засобу.

Об’єктом розробки є процеси моніторингу та диспетчерського управління міським громадським транспортом.

Предметом розробки є програмні засоби збирання, передавання, зберігання, обробки й візуалізації даних про рух транспортних засобів.

Метою роботи є розробка інформаційно-аналітичної системи, яка забезпечує ведення транспортних даних, отримання GPS-координат, контроль виконання рейсів, відображення транспорту на карті та формування аналітичних показників.

Для досягнення мети необхідно:

– сформувати вимоги та сценарії використання;

– розробити функціональну й архітектурну схеми;

– спроєктувати алгоритми та базу даних;

– реалізувати серверну частину й API;

– розробити веб- і мобільний застосунки;

– створити модуль імітації руху;

– реалізувати аналітичні функції;

– провести функціональне та інтеграційне тестування.

Результатом роботи має стати програмний прототип, що забезпечує повний базовий цикл: від формування маршрутів і рейсів до отримання координат, диспетчерського контролю та аналізу історії руху.

Отже, результати аналізу підтверджують доцільність створення спеціалізованої інформаційно-аналітичної системи та визначають основу для її подальшого проєктування й реалізації.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

2.1 Формування вимог і сценаріїв використання

Проєктування інформаційно-аналітичної системи управління рухом міського громадського транспорту розпочинається з визначення її функціонального призначення, категорій користувачів і вимог до програмного забезпечення. Сформовані вимоги визначають перелік функцій, які повинна підтримувати система, обмеження її роботи та очікуваний результат взаємодії користувачів із програмними компонентами.

Розроблювана система призначена для централізованого ведення транспортних даних, моніторингу поточного положення транспортних засобів, контролю виконання рейсів і формування аналітичної інформації. Вона повинна поєднувати вебзастосунок, мобільний застосунок водія, серверну частину та базу даних.

Основними категоріями користувачів системи є адміністратор, диспетчер, водій і пасажир. Кожна роль має окремі повноваження та працює лише з тією інформацією, яка необхідна для виконання відповідних завдань.

Адміністратор відповідає за налаштування системи та ведення довідкових даних. Він створює облікові записи користувачів, призначає ролі, додає маршрути, зупинки, транспортні засоби та водіїв, а також редагує або деактивує наявні записи.

Диспетчер здійснює оперативний контроль руху. Він переглядає активні рейси, положення транспортних засобів на карті, відомості про запізнення, втрату зв’язку та інші події. Диспетчер також може створювати рейси, призначати на них транспортні засоби й водіїв та змінювати статус виконання рейсу.

Водій взаємодіє із системою через мобільний застосунок. Після авторизації він переглядає призначений рейс, запускає його виконання та

передає GPS-координати транспортного засобу. Після завершення маршруту водій закінчує рейс, а передавання координат припиняється.

Пасажир користується відкритою частиною вебзастосунку. Він може переглядати маршрути, зупинки, розклади та поточне положення транспортних засобів. Пасажир не має доступу до персональних даних водіїв, службових повідомлень і функцій редагування.

Розподіл основних можливостей між ролями наведено в таблиці 2.1.

Таблиця 2.1 – Функції користувачів інформаційно-аналітичної системи

Функція	Адміністратор	Диспетчер	Водій	Пасажир
Авторизація в системі	+	+	+	–
Керування користувачами та ролями	+	–	–	–
Керування маршрутами й зупинками	+	Перегляд	–	Перегляд
Керування транспортними засобами	+	Перегляд	–	–
Керування водіями	+	Перегляд	–	–
Створення та призначення рейсів	±	+	–	–
Запуск і завершення рейсу	–	±	+	–
Передавання GPS-координат	–	–	+	–
Перегляд транспорту на карті	+	+	Власний рейс	+
Перегляд повідомлень про відхилення	+	+	Власний рейс	–
Перегляд аналітичних звітів	+	+	–	–
Перегляд розкладів	+	+	+	+

Позначення «±» у таблиці означає, що функція може бути доступною залежно від прийнятого розподілу повноважень. Наприклад, адміністратор може створювати рейси під час початкового налаштування, однак оперативне призначення доцільно виконувати диспетчеру.

Функціональні вимоги описують можливості, які система повинна надавати користувачам. Їх доцільно згрупувати за окремими функціональними підсистемами.

Підсистема автентифікації та розмежування доступу повинна забезпечувати:

– вхід користувача за логіном і паролем;

– визначення ролі після успішної авторизації;

– обмеження доступу до функцій відповідно до ролі;

– завершення активного сеансу;

– блокування доступу неактивних облікових записів.

Підсистема адміністрування повинна забезпечувати:

– створення, редагування та деактивацію облікових записів;

– призначення ролей користувачам;

– ведення даних про водіїв і транспортні засоби;

– створення та редагування маршрутів;

– додавання зупинок і визначення порядку їх проходження;

– створення та редагування розкладів.

Підсистема управління рейсами повинна забезпечувати:

– створення рейсу на основі маршруту та розкладу;

– призначення водія і транспортного засобу;

– визначення запланованого часу початку й завершення;

– зміну стану рейсу;

– перегляд активних, завершених і скасованих рейсів;

– збереження фактичного часу початку та завершення.

Підсистема GPS-моніторингу повинна забезпечувати:

– отримання координат від мобільного застосунку водія;

– перевірку коректності координат і часу їх формування;

– збереження координат із прив’язкою до активного рейсу;

– визначення часу останнього оновлення;

– відображення транспортного засобу на карті;

– виявлення тривалої відсутності нових координат;

– визначення можливого відхилення від маршруту.

Підсистема диспетчерського контролю повинна забезпечувати:

– перегляд активних рейсів;

– відображення транспортних засобів на карті;

– фільтрацію рейсів за маршрутом, статусом і транспортним засобом;

– відображення запізнень та інших проблемних подій;

– перегляд інформації про конкретний рейс;

– реєстрацію коментаря диспетчера щодо події.

Інформаційно-аналітична підсистема повинна забезпечувати:

– зберігання історії виконання рейсів;

– визначення планової та фактичної тривалості рейсу;

– розрахунок відхилення від розкладу;

– підрахунок кількості виконаних, скасованих і проблемних рейсів;

– формування звітів за маршрутом, транспортним засобом і часовим періодом;

– подання показників у вигляді таблиць і діаграм.

Пасажирська частина повинна забезпечувати:

– перегляд списку маршрутів;

– перегляд послідовності зупинок;

– перегляд розкладу;

– відображення транспортних засобів на карті;

– перегляд часу останнього оновлення даних;

– отримання повідомлень про затримку або тимчасову зміну руху.

Для тестування підсистеми моніторингу необхідно передбачити програмний імітатор руху. Він повинен виконувати роль джерела тестових

GPS-даних і передавати їх через той самий серверний інтерфейс, що й мобільний застосунок. Імітатор має підтримувати нормальний рух уздовж маршруту, зміну швидкості, зупинку, відхилення від маршруту та припинення передавання координат.

Сформовані функціональні вимоги наведено в узагальненому вигляді в таблиці 2.2.

Таблиця 2.2 – Основні функціональні вимоги до системи

Код	Вимога	Основний користувач
FR-01	Система повинна забезпечувати авторизацію та розмежування доступу	Адміністратор, диспетчер, водій
FR-02	Система повинна забезпечувати ведення маршрутів і зупинок	Адміністратор
FR-03	Система повинна забезпечувати облік водіїв і транспортних засобів	Адміністратор
FR-04	Система повинна забезпечувати створення та призначення рейсів	Диспетчер
FR-05	Мобільний застосунок повинен передавати GPS-координати активного рейсу	Водій
FR-06	Система повинна відображати транспортні засоби на карті	Диспетчер, пасажир
FR-07	Система повинна визначати відхилення від графіка й маршруту	Диспетчер
FR-08	Система повинна зберігати історію рейсів і координат	Адміністратор, диспетчер
FR-09	Система повинна формувати аналітичні показники та звіти	Адміністратор, диспетчер
FR-10	Система повинна надавати пасажиру відкриту транспортну інформацію	Пасажир
FR-11	Система повинна підтримувати імітацію руху для тестування	Розробник, диспетчер

Крім функціональних вимог, під час проєктування необхідно врахувати нефункціональні вимоги. Вони визначають характеристики якості,

продуктивності, надійності та зручності використання програмного забезпечення.

До основних нефункціональних вимог належать:

– вебінтерфейс повинен коректно працювати в актуальних версіях основних браузерів;

– мобільний застосунок повинен отримувати координати лише після надання користувачем відповідного дозволу;

– доступ до службових функцій повинен надаватися лише авторизованим користувачам;

– передавання облікових і координатних даних повинно виконуватися через захищене з’єднання;

– паролі не повинні зберігатися у відкритому вигляді;

– система повинна перевіряти вхідні дані на клієнтському та серверному рівнях;

– час відкриття основних сторінок за нормального навантаження не повинен створювати помітних затримок для користувача;

– сервер повинен коректно обробляти повторні, пропущені або несвоєчасно отримані координати;

– інтерфейс повинен бути зрозумілим для користувачів без спеціальної технічної підготовки;

– програмна архітектура повинна допускати подальше додавання нових маршрутів, транспортних засобів і користувачів.

Основні сценарії використання описують послідовність дій користувача та реакцію системи. Центральним сценарієм для адміністратора є налаштування довідкових даних. Адміністратор авторизується, створює маршрут, додає до нього зупинки, реєструє транспортний засіб і обліковий запис водія. Після перевірки даних вони стають доступними диспетчеру.

Основним сценарієм диспетчера є формування та контроль рейсу. Диспетчер вибирає маршрут і розклад, призначає водія та транспортний засіб,

після чого створює рейс. Під час його виконання диспетчер переглядає положення транспорту на карті та отримує повідомлення про відхилення.

Сценарій роботи водія починається з авторизації в мобільному застосунку. Водій вибирає призначений рейс і натискає кнопку початку. Застосунок отримує дозвіл на використання геолокації та періодично передає координати. Після завершення маршруту водій закінчує рейс.

Для пасажира основним сценарієм є пошук транспортної інформації. Пасажир відкриває вебзастосунок, вибирає маршрут або зупинку та переглядає розклад і поточне положення транспортних засобів. Авторизація для виконання цього сценарію не є обов’язковою.

Взаємодію користувачів із системою доцільно подати у вигляді UML-діаграми варіантів використання, наведеної на рисунку 2.1.

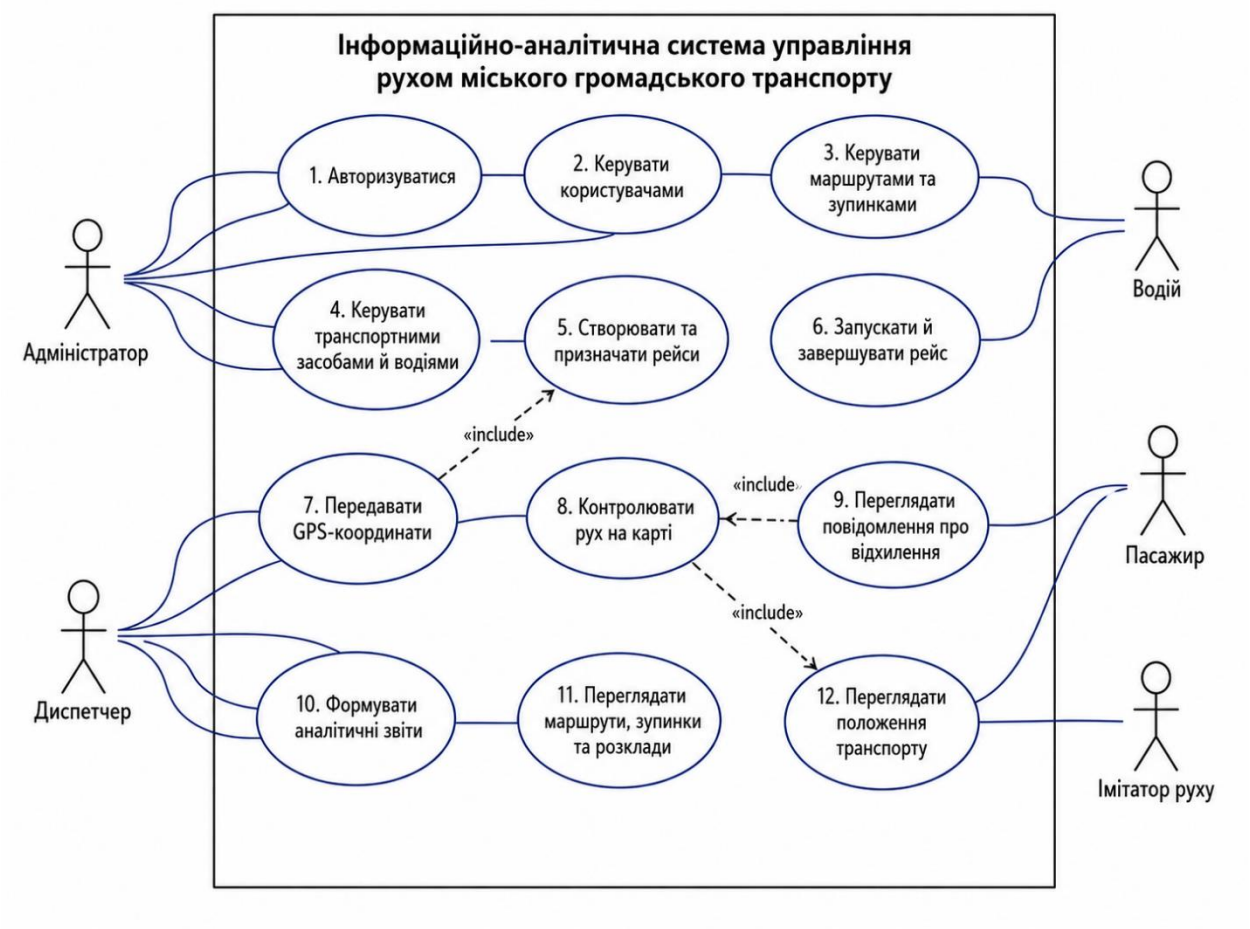


Рисунок 2.1 – Діаграма варіантів використання інформаційно-аналітичної системи

Для адміністратора необхідно показати зв’язки з функціями керування користувачами, маршрутами, зупинками, транспортними засобами й водіями. Диспетчера слід пов’язати зі створенням рейсів, контролем руху, переглядом відхилень і звітів. Водій взаємодіє з функціями авторизації, запуску рейсу та передавання координат. Пасажир переглядає маршрути, розклади й положення транспорту.

За потреби до діаграми можна додати актора «Імітатор руху». Він взаємодіятиме з варіантом використання «Передавати GPS-координати». Це покаже, що під час тестування координати можуть надходити не лише від мобільного застосунку, а й від програмного модуля імітації.

Сформовані вимоги та сценарії визначають межі розроблюваної системи. До її складу входять засоби адміністрування, диспетчерського контролю, мобільного передавання GPS-даних, пасажирського інформування та аналітики. Функції оплати проїзду, технічної діагностики транспортного засобу, автоматичного керування рухом і повноцінного прогнозування пасажиропотоку до першої версії системи не включаються.

Таким чином, визначені функціональні та нефункціональні вимоги охоплюють основні процеси управління рухом міського громадського транспорту. Сценарії використання встановлюють порядок взаємодії адміністратора, диспетчера, водія та пасажира з програмним забезпеченням і є основою для подальшого проєктування функціональної схеми, архітектури, алгоритмів і бази даних системи.

2.2 Розробка функціональної та архітектурної схем

На основі сформованих вимог і сценаріїв використання необхідно визначити склад основних функціональних підсистем та спосіб їх взаємодії. Розроблювана інформаційно-аналітична система повинна забезпечувати роботу адміністратора, диспетчера, водія та пасажира, а також підтримувати отримання, зберігання й обробку даних про рух транспортних засобів.

Функціональна схема відображає основні процеси, що виконуються в системі, та інформаційні зв’язки між її складовими. До основних функціональних підсистем належать:

– підсистема автентифікації та розмежування доступу;

– підсистема адміністрування довідкових даних;

– підсистема управління маршрутами, зупинками та розкладами;

– підсистема управління транспортними засобами, водіями та рейсами;

– підсистема отримання GPS-координат;

– підсистема диспетчерського моніторингу;

– інформаційно-аналітична підсистема;

– пасажирська інформаційна підсистема;

– модуль імітації руху.

Підсистема автентифікації забезпечує перевірку облікових даних користувача та визначення його ролі. Після успішного входу система формує набір доступних функцій відповідно до повноважень адміністратора, диспетчера або водія. Пасажирська частина може бути доступною без обов’язкової авторизації, оскільки вона містить лише відкриту транспортну інформацію.

Підсистема адміністрування призначена для ведення облікових записів користувачів і довідкових даних. Адміністратор створює та редагує маршрути, зупинки, транспортні засоби, відомості про водіїв і розклади. Ці дані використовуються іншими функціональними підсистемами під час формування рейсів і відображення транспортної інформації.

Підсистема управління рейсами забезпечує створення транспортного завдання на основі вибраного маршруту та розкладу. До рейсу призначаються водій і транспортний засіб, після чого зберігаються заплановані параметри його виконання. Під час руху рейс переходить між станами «запланований», «активний», «завершений» або «скасований».

Підсистема отримання GPS-координат приймає дані від мобільного застосунку водія або модуля імітації руху. Кожне повідомлення повинно

містити ідентифікатор активного рейсу, географічні координати, час їх отримання та, за можливості, швидкість руху. Сервер перевіряє формат даних, визначає їх належність до активного рейсу та зберігає у базі даних.

Підсистема диспетчерського моніторингу використовує отримані координати для відображення транспортних засобів на електронній карті. Для кожного активного рейсу диспетчер може переглянути маршрут, транспортний засіб, водія, час останнього оновлення та поточний стан. За потреби відображаються повідомлення про запізнення, відхилення від маршруту, тривалу зупинку або втрату зв’язку.

Інформаційно-аналітична підсистема обробляє історичні дані про рейси та координати. Вона визначає фактичну тривалість руху, величину відхилення від розкладу, кількість виконаних і скасованих рейсів, а також інші показники. Результати можуть бути подані у вигляді таблиць, діаграм і звітів за вибраний часовий період.

Пасажирська підсистема надає відкриті відомості про маршрути, зупинки, розклади та поточне положення транспорту. При цьому службова інформація, персональні дані водіїв і внутрішні повідомлення диспетчера не повинні бути доступними пасажиру.

Модуль імітації руху призначений для тестування системи. Він послідовно формує координати вздовж обраного маршруту та передає їх серверу через той самий програмний інтерфейс, що й мобільний застосунок. Це дозволяє перевіряти відображення транспорту на карті та реакцію системи на різні сценарії без використання реального транспортного засобу.

Функціональну структуру системи наведено на рисунку 2.2.

Після визначення функціонального складу необхідно розробити архітектурну схему програмного забезпечення. Для системи обрано клієнт-серверну архітектуру, оскільки вона дозволяє розділити користувацькі інтерфейси, бізнес-логіку та зберігання даних.

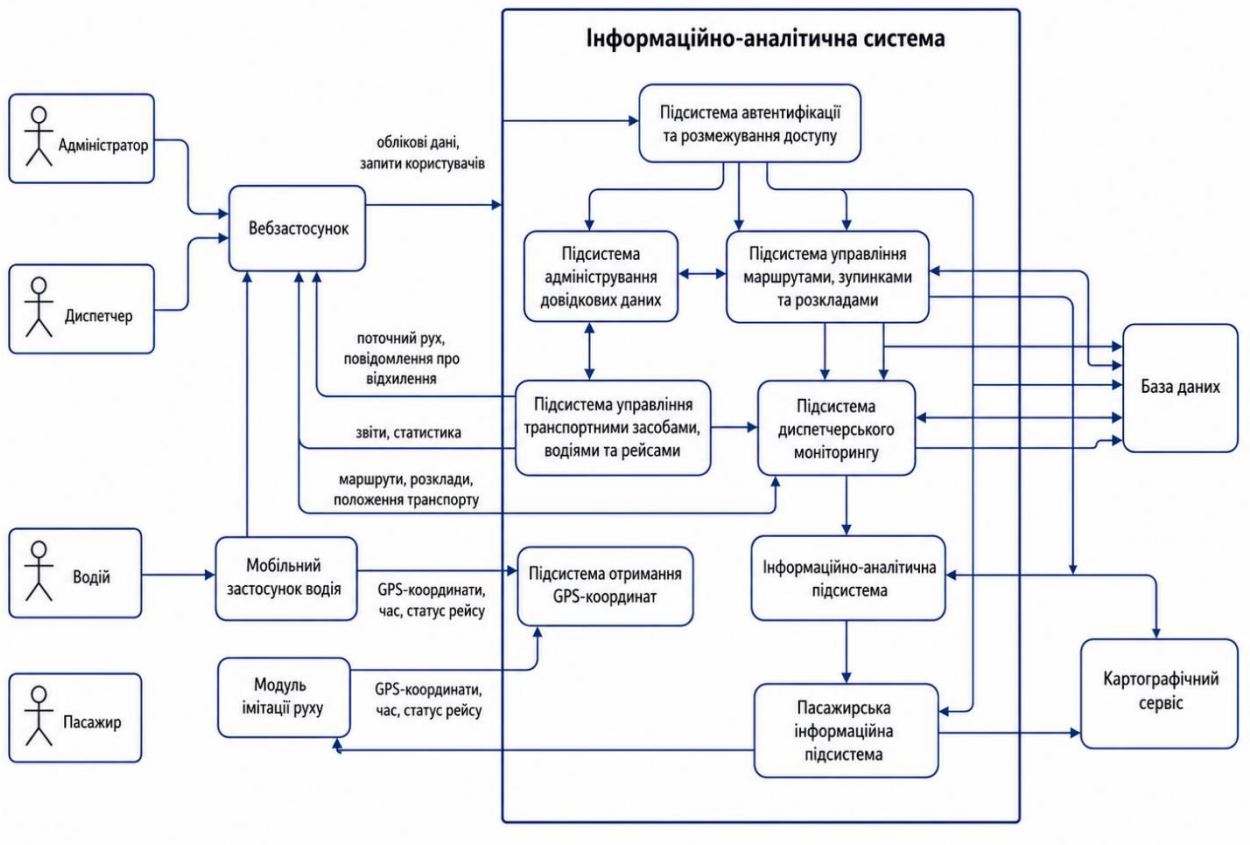


Рисунок 2.2 – Функціональна схема інформаційно-аналітичної системи

До складу програмної архітектури входять:

– вебклієнт;

– мобільний клієнт;

– серверний застосунок;

– база даних;

– зовнішній картографічний сервіс;

– модуль імітації руху.

Вебклієнт забезпечує взаємодію адміністратора, диспетчера та пасажира із системою. Він відповідає за відображення сторінок, форм, таблиць, карт і діаграм. Вебклієнт не повинен безпосередньо звертатися до бази даних. Усі операції виконуються через серверний API.

Для адміністратора вебклієнт надає сторінки керування користувачами та транспортними довідниками. Для диспетчера реалізується карта активних рейсів, список подій і засоби перегляду аналітичних показників. Пасажирська

частина містить відкритий перегляд маршрутів, зупинок, розкладів і положення транспорту.

Мобільний клієнт використовується водієм. Його функціональність повинна бути компактною, щоб не відволікати водія під час роботи. Основними екранами є авторизація, перелік призначених рейсів, відомості про вибраний рейс, кнопки початку та завершення руху, а також індикатор стану GPS і передавання даних.

Мобільний застосунок отримує координати за допомогою засобів геолокації пристрою. Під час активного рейсу він періодично формує повідомлення й надсилає його серверному API. Якщо з’єднання тимчасово відсутнє, застосунок повинен відобразити відповідний стан. За можливості координати можуть тимчасово зберігатися локально та передаватися після відновлення зв’язку.

Серверний застосунок є центральним компонентом архітектури. Він приймає HTTP-запити від клієнтів, виконує автентифікацію, перевіряє права доступу, обробляє вхідні дані та взаємодіє з базою даних. На серверному рівні реалізується основна бізнес-логіка системи.

Серверну частину доцільно поділити на логічні модулі:

– модуль автентифікації та авторизації;

– модуль керування користувачами;

– модуль маршрутів і зупинок;

– модуль транспортних засобів і водіїв;

– модуль рейсів і розкладів;

– модуль приймання GPS-координат;

– модуль виявлення подій;

– модуль аналітики та звітності.

Такий поділ спрощує підтримку програмного коду та дозволяє змінювати окремі частини системи без суттєвого впливу на інші компоненти.

База даних призначена для централізованого зберігання інформації. У ній містяться облікові записи, ролі, маршрути, зупинки, транспортні засоби,

водії, розклади, рейси, GPS-координати, події та сформовані аналітичні показники.

Взаємодія з базою даних здійснюється лише серверною частиною. Такий підхід підвищує безпеку та забезпечує єдині правила перевірки даних. Клієнтські застосунки отримують лише ту інформацію, яка дозволена відповідній ролі.

Картографічний сервіс використовується для відображення електронної карти, маршрутів, зупинок і транспортних засобів. Вебклієнт може завантажувати картографічні шари через відповідний програмний інтерфейс, а координати й транспортні дані отримувати із серверної частини.

Модуль імітації руху є окремим тестовим клієнтом. Він не повинен мати прямого доступу до бази даних. Координати передаються через серверний API, що дозволяє перевірити весь ланцюг обробки даних у тих самих умовах, що й під час роботи мобільного застосунку.

Для оперативного оновлення положення транспорту може використовуватися періодичне опитування серверного API або двостороннє з’єднання між вебклієнтом і сервером. У першому випадку клієнт надсилає запити через визначений проміжок часу. У другому випадку сервер сам передає оновлення після отримання нових координат. Остаточний механізм залежатиме від обраного технологічного стека.

Архітектурну схему системи наведено на рисунку 2.3.

Між клієнтськими застосунками та серверною частиною необхідно показати обмін даними через захищені HTTP-запити. Між сервером і базою даних відображається читання та збереження інформації. Картографічний сервіс використовується для завантаження карти й географічних даних.

Обрана архітектура має низку переваг. Централізована серверна логіка забезпечує однакові правила обробки даних для веб- і мобільного клієнтів. Розмежування компонентів підвищує безпеку та спрощує тестування. Додавання нових клієнтських застосунків у майбутньому не потребуватиме зміни структури бази даних, якщо вони використовуватимуть наявний API.

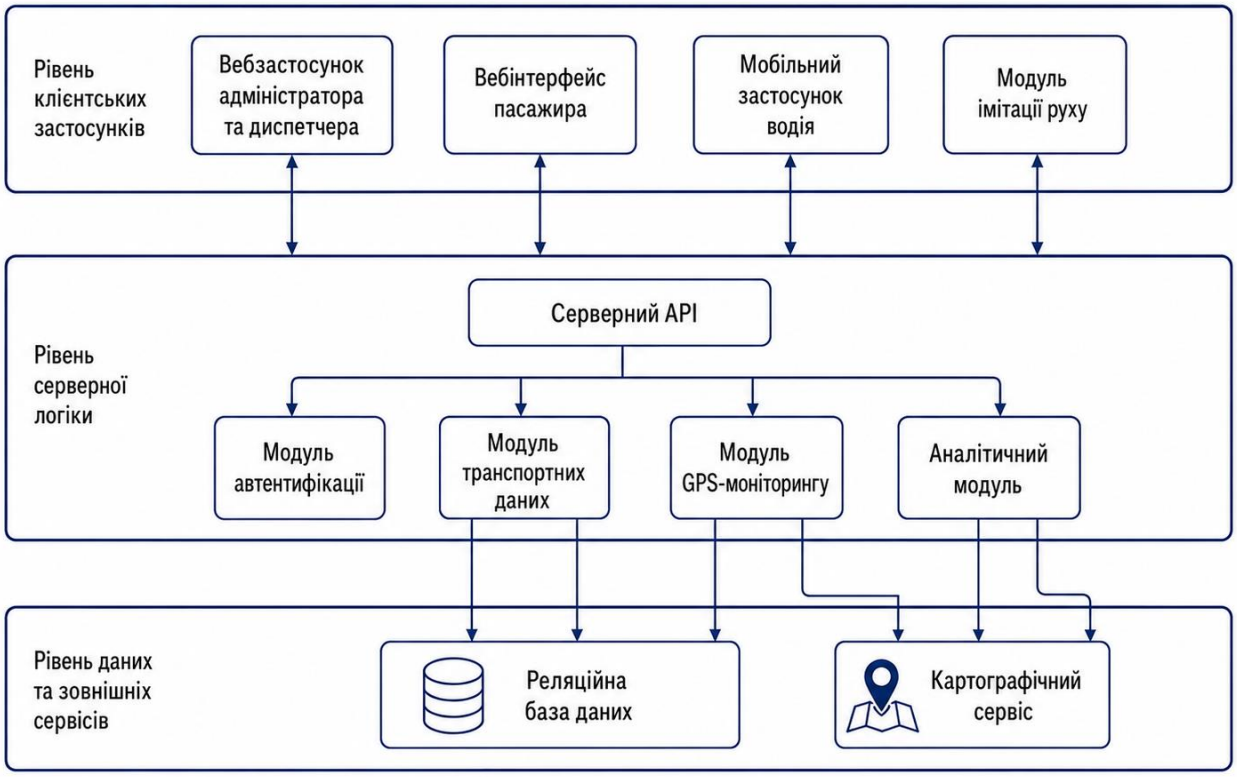


Рисунок 2.3 – Архітектурна схема інформаційно-аналітичної системи

Недоліком клієнт-серверного підходу є залежність від доступності сервера та мережевого з’єднання. У разі відсутності інтернету мобільний застосунок не може одразу передати координати, а диспетчер не отримує актуального положення транспорту. Для зменшення впливу цієї проблеми необхідно контролювати час останнього оновлення та передбачити повідомлення про втрату зв’язку.

Таким чином, функціональна схема визначає склад підсистем і потоки інформації, а архітектурна схема встановлює програмні компоненти та спосіб їх взаємодії. Клієнт-серверна архітектура забезпечує централізоване опрацювання даних, підтримку декількох категорій користувачів і можливість подальшого розширення інформаційно-аналітичної системи.

2.3 Розробка алгоритмів роботи системи

Після визначення функціональної структури та архітектури інформаційно-аналітичної системи необхідно розробити алгоритми виконання

її основних процесів. Алгоритми встановлюють послідовність дій користувачів і програмних компонентів, порядок обробки даних та умови переходу між окремими станами системи.

Основними процесами, для яких необхідно розробити алгоритми, є:

– взаємодія користувачів із системою;

– отримання та перевірка GPS-координат;

– визначення відхилення від маршруту й розкладу;

– формування диспетчерських подій;

– імітація руху транспортного засобу.

Загальний алгоритм роботи системи охоплює взаємодію клієнтських застосунків із серверною частиною та базою даних. Послідовність дій залежить від ролі користувача.

На початку роботи користувач відкриває веб- або мобільний застосунок. Для адміністратора, диспетчера та водія виконується авторизація. Облікові дані передаються серверному API, після чого сервер знаходить відповідний обліковий запис, перевіряє пароль і стан користувача.

Якщо облікові дані введено неправильно або обліковий запис деактивовано, система повертає повідомлення про помилку. У разі успішної авторизації визначається роль користувача та формується набір доступних функцій.

Адміністратор отримує доступ до керування користувачами, маршрутами, зупинками, транспортними засобами, водіями та розкладами. Дані, введені або змінені адміністратором, проходять серверну перевірку та зберігаються в базі даних.

Диспетчер створює та призначає рейси, переглядає активні транспортні засоби на карті та контролює повідомлення про відхилення. Серверна частина надає йому актуальні координати, статуси рейсів і результати обробки транспортних даних.

Водій після авторизації в мобільному застосунку вибирає призначений рейс і запускає його виконання. Застосунок починає отримувати координати

мобільного пристрою та передавати їх серверу. Передавання координат продовжується до завершення рейсу.

Пасажир використовує відкриту частину вебзастосунку. Він вибирає маршрут або зупинку, після чого система відображає розклад, послідовність зупинок і поточне положення транспортних засобів.

Загальний алгоритм роботи інформаційно-аналітичної системи наведено на рисунку 2.4 [15].

Центральним процесом системи є отримання та обробка GPS-координат транспортного засобу. Джерелом даних може бути мобільний застосунок водія або модуль імітації руху. Для обох джерел використовується однакова структура повідомлення та один серверний маршрут API.

Повідомлення про місцеположення повинно містити:

– ідентифікатор рейсу;

– ідентифікатор транспортного засобу;

– географічну широту;

– географічну довготу;

– час визначення координат;

– швидкість руху, якщо вона доступна;

– ознаку джерела даних.

Після надходження повідомлення сервер перевіряє авторизаційні дані відправника та наявність активного рейсу. Якщо рейс не знайдено, завершено або скасовано, координата не додається до історії руху.

На наступному етапі перевіряється формат отриманих даних. Значення широти повинно перебувати в діапазоні від мінус 90 до 90 градусів, а довготи — від мінус 180 до 180 градусів. Також перевіряється відповідність транспортного засобу призначеному рейсу та наявність часової позначки.

Сервер повинен враховувати можливість повторного надсилання даних. Якщо координата з однаковим ідентифікатором рейсу та часом уже існує в базі даних, повторний запис не створюється. Дані, які надійшли із суттєвим запізненням, можуть бути позначені як несвоєчасні.

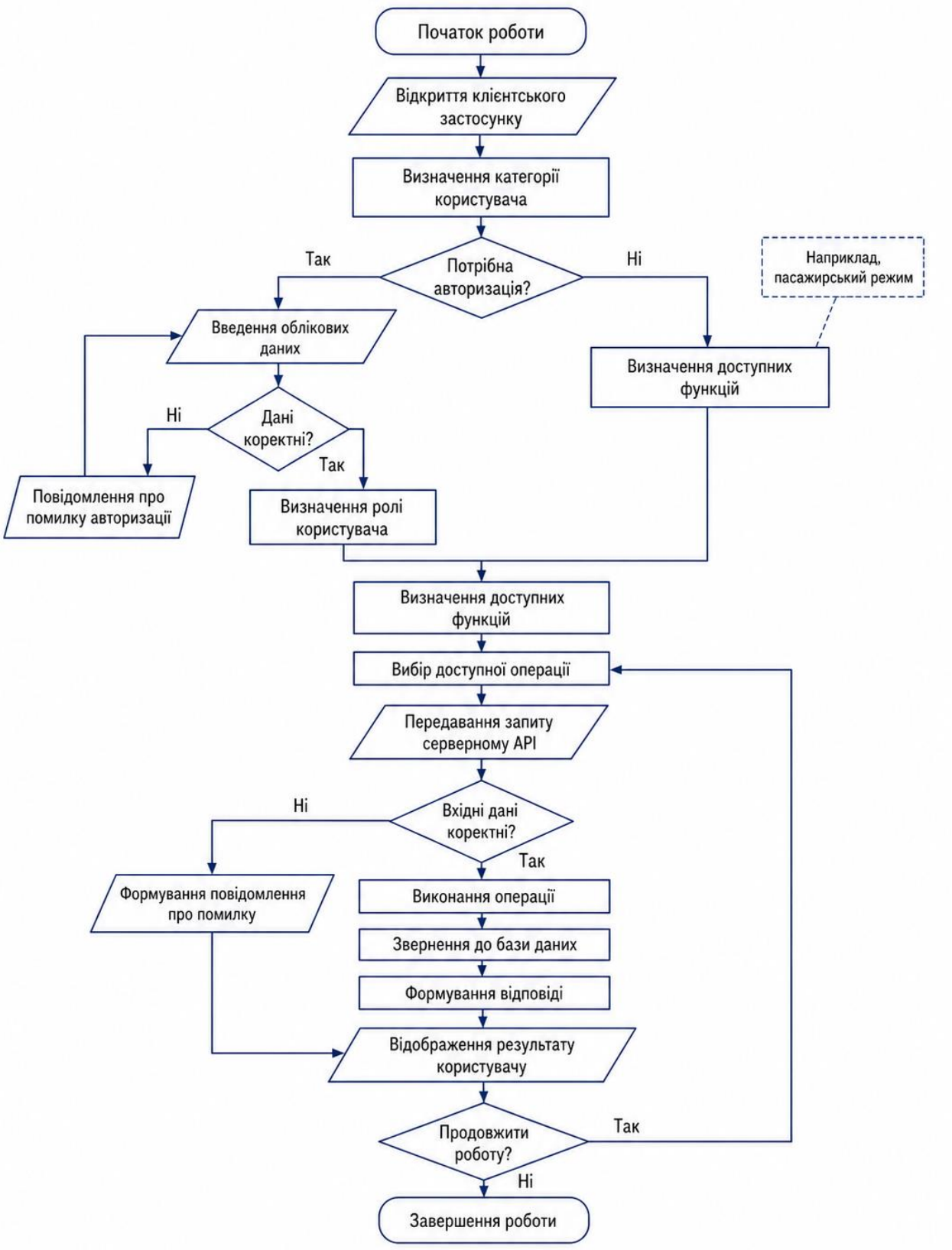


Рисунок 2.4 – Загальний алгоритм роботи інформаційно-аналітичної системи

Після успішної перевірки координата зберігається в базі даних. Одночасно оновлюється поточне положення транспортного засобу та час останнього зв’язку. Це дозволяє отримувати актуальний стан рейсу без опрацювання всієї історії координат.

Після збереження даних виконується визначення положення транспортного засобу відносно маршруту. Поточна точка порівнюється з геометрією маршруту. Якщо відстань до найближчої його ділянки не перевищує допустимого значення, транспортний засіб вважається таким, що рухається за встановленим маршрутом.

Якщо допустиму відстань перевищено, система не повинна одразу створювати критичне повідомлення, оскільки окрема GPS-точка може містити похибку. Подія «Відхилення від маршруту» формується лише в тому випадку, якщо декілька послідовних координат знаходяться за межами допустимої зони або транспортний засіб перебуває поза маршрутом протягом визначеного часу.

Для контролю виконання розкладу координати також порівнюються з положенням зупинок. Якщо транспортний засіб потрапляє до заданої контрольної зони зупинки, фіксується фактичний час її проходження. Отримане значення порівнюється із запланованим часом.

Відхилення від розкладу визначається за формулою:

$$\Delta t = t_{\text{факт}} - t_{\text{план}} ,$$

де Δt — відхилення від розкладу; $t_{\text{факт}}$ — фактичний час проходження зупинки; $t_{\text{план}}$ — запланований час проходження зупинки.

Якщо значення Δt додатне, транспортний засіб запізнюється. Від’ємне значення означає випередження графіка. Якщо абсолютне значення відхилення не перевищує встановленого допустимого інтервалу, рейс вважається таким, що виконується за розкладом.

Окремо контролюється тривалість відсутності нових координат. Якщо з моменту останнього повідомлення минув установлений час, система формує

подію «Втрата зв’язку». Після надходження нової коректної координати статус зв’язку відновлюється.

Також перевіряється тривала відсутність переміщення. Якщо декілька послідовних координат перебувають у межах невеликого радіуса, а транспортний засіб не знаходиться в зоні зупинки, система формує повідомлення про можливу тривалу зупинку.

Алгоритм обробки GPS-координат і визначення відхилень наведено на рисунку 2.5.

Для зменшення кількості помилкових подій система повинна використовувати налаштовувані параметри. До них належать допустима відстань від маршруту, радіус контрольної зони зупинки, допустиме запізнення, максимальний час без координат і тривалість нерухомого стану.

Для перевірки роботи підсистеми моніторингу без використання реального транспортного засобу передбачено модуль імітації руху. Він формує послідовність координат уздовж вибраного маршруту та передає їх серверному API у тому самому форматі, що й мобільний застосунок.

Перед початком імітації вибираються маршрут, рейс і режим руху. Після цього завантажується впорядкована послідовність координат маршруту. Для кожної ділянки може бути задано орієнтовну швидкість або час переходу до наступної точки.

У нормальному режимі імітатор послідовно переходить між точками маршруту. Через визначений проміжок часу формується повідомлення, яке містить координати, часову позначку та ідентифікатор рейсу. Повідомлення надсилається серверному API, після чого модуль переходить до наступної точки.

У зоні зупинки може передбачатися коротка затримка, що імітує посадку та висадку пасажирів. Швидкість руху може бути сталою або змінюватися залежно від ділянки маршруту.

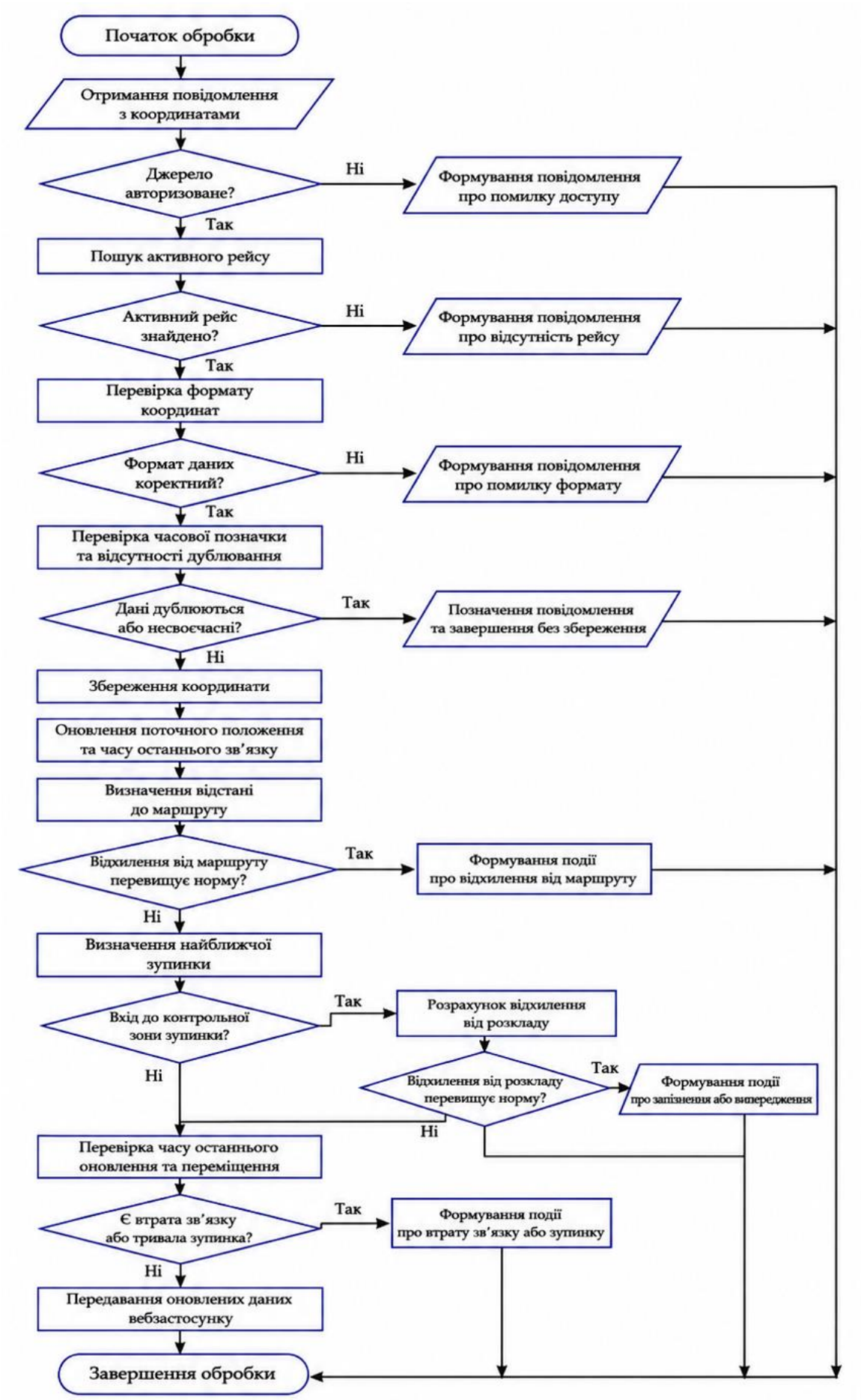


Рисунок 2.5 – Алгоритм обробки GPS-координат і визначення відхилень

Для перевірки проблемних сценаріїв модуль імітації повинен підтримувати такі режими:

– нормальний рух;

– рух зі зниженою швидкістю;

– запізнення на визначений час;

– тривала зупинка;

– відхилення від маршруту;

– тимчасове припинення передавання координат.

У режимі відхилення використовується окрема послідовність точок або до координат маршруту додається зміщення. У режимі втрати зв’язку координати можуть продовжувати змінюватися локально, але протягом установленого часу не передаються серверу.

Алгоритм роботи модуля імітації руху наведено на рисунку 2.6.

Під час оформлення блок-схем необхідно дотримуватися єдиного напрямку виконання алгоритму, переважно зверху вниз. Виходи з умов повинні мати підписи «так» і «ні». Лінії потоку не повинні перетинати блоки, а текст усередині символів має стисло описувати одну операцію або перевірку.

Для переходу між віддаленими частинами великої схеми можуть використовуватися з’єднувачі. Це дозволяє уникнути довгих перехресних ліній і підвищити читабельність рисунка.

Наведені блок-схеми узагальнюють логіку виконання основних процесів системи та встановлюють послідовність переходів між окремими етапами обробки даних. Їх використання дає змогу ще до початку програмної реалізації перевірити повноту передбачених сценаріїв, виявити можливі помилки в логіці та визначити точки взаємодії між клієнтськими застосунками, серверною частиною і базою даних. Окреме подання алгоритмів також спрощує подальше тестування, оскільки кожен із них може бути перевірений за незалежним набором вхідних даних і очікуваних результатів.

Рисунок 2.6 – Алгоритм імітації руху транспортного засобу

Таким чином, розроблені алгоритми охоплюють основні процеси інформаційно-аналітичної системи: взаємодію користувачів із програмними компонентами, отримання та перевірку GPS-координат, визначення відхилень і тестову імітацію руху. Вони є основою для подальшої реалізації серверних модулів, програмного API та логіки клієнтських застосунків.

2.4 Проєктування бази даних та інтерфейсу

Для забезпечення роботи інформаційно-аналітичної системи необхідно спроєктувати структуру даних і користувацькі інтерфейси. База даних повинна зберігати відомості про користувачів, водіїв, транспортні засоби, маршрути, зупинки, розклади, рейси, GPS-координати та події руху. Інтерфейс повинен надавати кожній категорії користувачів доступ лише до тих функцій, які відповідають її ролі.

Під час проєктування бази даних використано реляційну модель. Вона подає інформацію у вигляді взаємопов’язаних відношень, які на практиці реалізуються як таблиці. Основні принципи реляційного підходу були сформульовані Е. Коддом і передбачають логічне подання даних незалежно від способу їх фізичного зберігання [16]. Для розроблюваної системи цей підхід є доцільним, оскільки предметна область містить структуровані сутності та чітко визначені зв’язки між користувачами, маршрутами, зупинками, транспортними засобами, рейсами й координатами.

На етапі концептуального проєктування необхідно визначити основні сутності, їх атрибути та зв’язки. Побудова ER-моделі дає змогу відобразити структуру предметної області до вибору конкретної системи керування базами даних і зменшити ймовірність появи суперечливих або дубльованих даних [17].

Під час формування таблиць враховано принципи нормалізації. Повторювані групи відомостей виділяються в окремі сутності, а зв’язки між ними встановлюються за допомогою зовнішніх ключів. Наприклад, інформація про маршрут не дублюється в кожному рейсі, а зберігається один

раз у таблиці Routes та пов’язується з таблицею Trips через поле route_id. Це зменшує надлишковість і спрощує оновлення даних [17].

Основними сутностями системи є User, Role, Driver, Vehicle, Route, Stop, RouteStop, Schedule, Trip, GpsPoint і Event.

Сутність User призначена для зберігання облікових даних користувачів. Вона пов’язується із сутністю Role, яка визначає набір прав доступу. Для водія додатково використовується окрема сутність Driver, що містить службові відомості та пов’язується з обліковим записом користувача.

Сутність Vehicle містить інформацію про транспортний засіб: реєстраційний номер, бортовий номер, тип, місткість, технічний стан і статус доступності. Один транспортний засіб може використовуватися в багатьох рейсах, але кожен конкретний рейс пов’язується лише з одним транспортним засобом.

Сутність Route описує маршрут громадського транспорту. Вона містить номер, назву, напрямок, довжину, вид транспорту та стан активності. Оскільки один маршрут включає багато зупинок, а одна зупинка може входити до складу багатьох маршрутів, між сутностями Route і Stop реалізується зв’язок «багато до багатьох».

Для представлення цього зв’язку використовується проміжна сутність RouteStop. Вона зберігає порядок проходження зупинки, напрямок руху, орієнтовну відстань від початку маршруту та плановий часовий зсув відносно початку рейсу.

Сутність Schedule призначена для зберігання розкладів. Вона пов’язується з маршрутом та містить тип дня, час початку, період дії й інші параметри планового руху. Для одного маршруту можуть створюватися окремі розклади для робочих, вихідних і святкових днів.

Сутність Trip представляє конкретне виконання маршруту. Вона поєднує маршрут, розклад, водія та транспортний засіб. У ній зберігаються плановий і фактичний час початку та завершення рейсу, а також його поточний статус.

Координати руху зберігаються в сутності GpsPoint. Кожен запис пов’язаний із конкретним рейсом і містить широту, довготу, час вимірювання, швидкість та джерело даних. Джерелом може бути мобільний застосунок водія або модуль імітації руху.

Сутність Event використовується для реєстрації подій, що виникають під час виконання рейсу. До них належать запізнення, випередження розкладу, відхилення від маршруту, тривала зупинка, втрата зв’язку та відновлення передавання даних. Подія пов’язується з рейсом і містить тип, час формування, опис, рівень важливості та стан опрацювання.

Концептуальну модель бази даних наведено на рисунку 2.7.

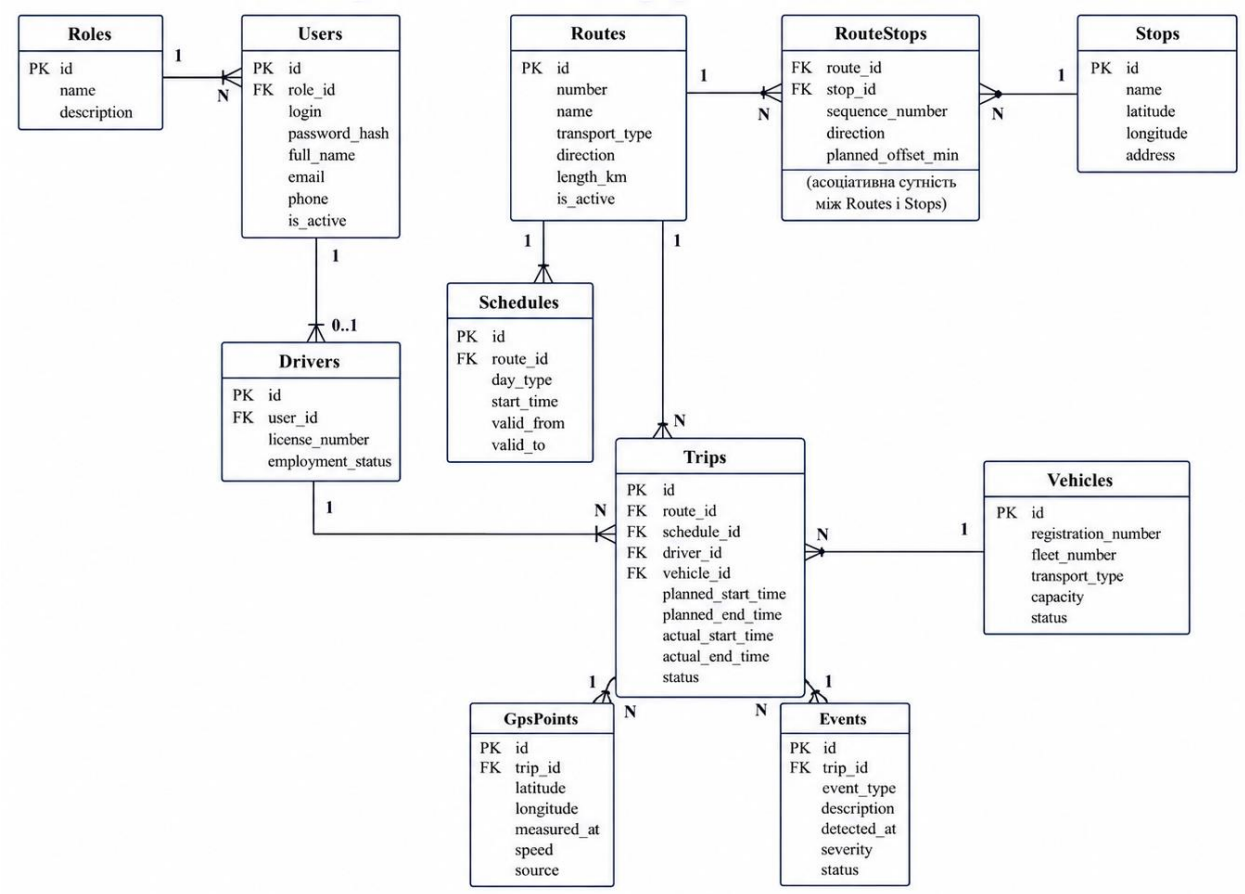


Рисунок 2.7 – ER-діаграма бази даних інформаційно-аналітичної системи

Для забезпечення цілісності даних у таблицях використовуються первинні та зовнішні ключі. Первинний ключ однозначно ідентифікує запис, а зовнішній ключ установлює зв’язок між таблицями. Використання зовнішніх

ключів забезпечує посилальну цілісність і не дозволяє створювати записи, пов’язані з неіснуючими об’єктами [18].

Крім ключів, на рівні бази даних доцільно застосовувати обмеження унікальності, обов’язковості та допустимих значень. Наприклад, логін користувача, бортовий номер транспортного засобу та номер посвідчення водія повинні бути унікальними. Поля, без яких запис не має змісту, позначаються як обов’язкові. Перевірка лише на рівні клієнтського або серверного застосунку не гарантує цілісності всіх збережених даних, тому основні обмеження дублюються в СУБД [18].

Видалення записів, які вже використовуються в рейсах або історичних даних, доцільно обмежити. Для користувачів, маршрутів, водіїв і транспортних засобів замість фізичного видалення застосовується ознака активності. Це дозволяє зберегти історичну цілісність даних про завершені рейси.

Структуру основних таблиць наведено в таблицях 2.3–2.8. Для скорочення обсягу пояснювальної записки допоміжні поля аудиту, зокрема дата створення, дата оновлення та ідентифікатор користувача, який виконав зміну, не повторюються в кожній таблиці, але передбачаються під час програмної реалізації.

Таблиця 2.3 – Структура таблиці Drivers

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор водія
user_id	UUID	Зовнішній ключ на поле id таблиці Users
license_number	VARCHAR(50)	Унікальний номер посвідчення водія
employment_status	VARCHAR(30)	Поточний статус працевлаштування водія

Таблиця 2.4 – Структура таблиці Roles

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор ролі
name	VARCHAR(50)	Унікальна назва ролі користувача
description	VARCHAR(255)	Опис повноважень і призначення ролі

Таблиця 2.5 – Структура таблиці Users

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор користувача
role_id	UUID	Зовнішній ключ на поле id таблиці Roles
login	VARCHAR(100)	Унікальний логін користувача
password_hash	VARCHAR(255)	Хеш пароля користувача
full_name	VARCHAR(150)	Повне ім’я користувача
email	VARCHAR(150)	Адреса електронної пошти
phone	VARCHAR(20)	Контактний номер телефону
is_active	BOOLEAN	Ознака активності облікового запису

Таблиця 2.6 – Структура таблиці Vehicles

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор транспортного засобу
registration_number	VARCHAR(20)	Державний реєстраційний номер
fleet_number	VARCHAR(20)	Унікальний бортовий номер
transport_type	VARCHAR(30)	Тип транспортного засобу
capacity	INTEGER	Загальна пасажиромісткість
status	VARCHAR(30)	Поточний стан або доступність транспортного засобу

Таблиця 2.7 – Структура таблиці Routes

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор маршруту
number	VARCHAR(20)	Номер маршруту
name	VARCHAR(150)	Назва або опис маршруту
transport_type	VARCHAR(30)	Вид громадського транспорту
direction	VARCHAR(30)	Напрямок руху маршруту
length_km	DECIMAL(8,2)	Загальна довжина маршруту в кілометрах
is_active	BOOLEAN	Ознака активності маршруту

Таблиця 2.8 – Структура таблиці Stops

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор зупинки
name	VARCHAR(150)	Назва зупинки
latitude	DECIMAL(9,6)	Географічна широта зупинки
longitude	DECIMAL(9,6)	Географічна довгота зупинки
address	VARCHAR(255)	Адреса або додатковий опис розташування

Таблиця 2.9 – Структура таблиці Schedules

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор розкладу
route_id	UUID	Зовнішній ключ на поле id таблиці Routes
day_type	VARCHAR(30)	Тип дня: робочий, вихідний або святковий
start_time	TIME	Плановий час початку руху
valid_from	DATE	Дата початку дії розкладу
valid_to	DATE	Дата завершення дії розкладу

Таблиця 2.10 – Структура таблиці RouteStops

Поле	Тип даних	Опис
route_id	UUID	Зовнішній ключ на поле id таблиці Routes
stop_id	UUID	Зовнішній ключ на поле id таблиці Stops
sequence_number	INTEGER	Порядковий номер зупинки в межах маршруту
direction	VARCHAR(30)	Напрямок руху, для якого використовується зупинка
planned_offset_min	INTEGER	Плановий час проходження від початку рейсу, хв

Для таблиці RouteStops доцільно застосувати складений первинний ключ за полями route_id, stop_id, direction і sequence_number або створити для цієї комбінації обмеження унікальності.

Таблиця 2.11 – Структура таблиці GpsPoints

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор GPS-точки
trip_id	UUID	Зовнішній ключ на поле id таблиці Trips
latitude	DECIMAL(9,6)	Географічна широта транспортного засобу
longitude	DECIMAL(9,6)	Географічна довгота транспортного засобу
measured_at	TIMESTAMP	Дата й час визначення координат
speed	DECIMAL(6,2)	Швидкість транспортного засобу, км/год
source	VARCHAR(30)	Джерело координат: мобільний застосунок або імітатор

Таблиця 2.13 – Структура таблиці Trips

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор рейсу
route_id	UUID	Зовнішній ключ на поле id таблиці Routes
schedule_id	UUID	Зовнішній ключ на поле id таблиці Schedules
driver_id	UUID	Зовнішній ключ на поле id таблиці Drivers
vehicle_id	UUID	Зовнішній ключ на поле id таблиці Vehicles
planned_start_time	TIMESTAMP	Заплановані дата й час початку рейсу
planned_end_time	TIMESTAMP	Заплановані дата й час завершення рейсу
actual_start_time	TIMESTAMP	Фактичні дата й час початку рейсу
actual_end_time	TIMESTAMP	Фактичні дата й час завершення рейсу
status	VARCHAR(30)	Поточний статус рейсу

Таблиця 2.13 – Структура таблиці Events

Поле	Тип даних	Опис
id	UUID	Первинний ключ, унікальний ідентифікатор події
trip_id	UUID	Зовнішній ключ на поле id таблиці Trips
event_type	VARCHAR(50)	Тип зареєстрованої події
description	TEXT	Детальний опис події
detected_at	TIMESTAMP	Дата й час виявлення події
severity	VARCHAR(20)	Рівень важливості події
status	VARCHAR(30)	Поточний стан опрацювання події

Під час практичної реалізації для полів статусів доцільно використовувати обмежений перелік допустимих значень. Наприклад, статус рейсу може набувати значень «запланований», «активний», «завершений» або «скасований», а статус транспортного засобу — «доступний», «на маршруті», «на обслуговуванні» або «несправний». Це зменшує кількість помилкових і неоднозначних записів та спрощує подальшу фільтрацію даних.

Для таблиць Trips, GpsPoints і Events важливим є збереження хронологічної послідовності даних. Саме за часовими позначками система визначатиме фактичний перебіг рейсу, час останнього зв’язку, затримки та момент виникнення подій. Тому поля planned_start_time, actual_start_time, measured_at і detected_at повинні зберігатися в єдиному часовому форматі з урахуванням часового поясу системи.

Окрему увагу слід приділити індексуванню зовнішніх ключів і часових полів. Індекси за trip_id, route_id, vehicle_id, status і measured_at прискорять вибірку активних рейсів, завантаження історії координат та формування аналітичних звітів. Водночас надмірна кількість індексів може збільшити витрати під час запису даних, тому їх остаточний склад доцільно визначати після аналізу типових запитів і результатів тестування продуктивності.

Наведена структура відповідає ER-діаграмі та забезпечує реалізацію визначених зв’язків між сутностями. Таблиця RouteStops виконує роль асоціативної сутності між маршрутами та зупинками. Таблиця Trips є центральною для оперативної частини системи, оскільки пов’язує маршрут, розклад, водія та транспортний засіб. Таблиці GpsPoints і Events зберігають фактичні дані, сформовані під час виконання рейсу.

3 РОЗРОБКА ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

3.1 Вибір засобів і технологій розробки

Для реалізації інформаційно-аналітичної системи необхідно вибрати технології, які забезпечують створення вебзастосунку, мобільного застосунку водія, серверної частини, бази даних і картографічного модуля. Під час вибору враховано відповідність клієнт-серверній архітектурі, підтримку роботи з GPS-координатами, можливість розмежування доступу, інтеграцію з реляційною базою даних та зручність подальшого розширення системи.

Для розроблення серверної частини обрано платформу ASP.NET Core і мову програмування C#. ASP.NET Core підтримує створення вебсервісів і REST API, автентифікацію користувачів, розмежування доступу та обмін даними між веб- і мобільними клієнтами [19]. Серверна частина виконуватиме основну бізнес-логіку системи: роботу з маршрутами, рейсами, транспортними засобами, GPS-координатами та аналітичними показниками.

Для доступу до бази даних використовується Entity Framework Core. Цей засіб дає змогу працювати з таблицями через класи C#, підтримує зв’язки між сутностями та міграції структури бази даних. Його використання зменшує обсяг типового коду та спрощує підтримку моделі даних.

Як систему керування базами даних обрано PostgreSQL. Вона підтримує реляційну модель, транзакції, первинні й зовнішні ключі, індекси та складні SQL-запити [20]. PostgreSQL відповідає спроєктованій структурі даних і може використовуватися для зберігання як довідкової інформації, так і великої кількості GPS-точок.

Для створення вебзастосунку обрано React у поєднанні з TypeScript. React дозволяє будувати інтерфейс із повторно використовуваних компонентів, що є зручним для реалізації таблиць, форм, карток, панелей фільтрації та диспетчерської карти [21]. TypeScript забезпечує статичну типізацію та допомагає узгодити структури даних вебклієнта з моделями серверного API.

Вебзастосунок використовуватиметься адміністратором, диспетчером і пасажиром. Для адміністратора передбачаються сторінки керування довідковими даними, для диспетчера — карта активних рейсів і повідомлення про відхилення, а для пасажира — перегляд маршрутів, зупинок і положення транспорту.

Мобільний застосунок водія пропонується реалізувати за допомогою React Native та Expo. Такий вибір дає змогу застосувати компонентний підхід React і створити мобільний клієнт для отримання та передавання GPS-координат. Основними функціями застосунку будуть авторизація, вибір рейсу, запуск і завершення руху та передавання місцеположення транспортного засобу.

Для відображення маршрутів, зупинок і транспортних засобів на електронній карті обрано бібліотеку Leaflet та картографічні дані OpenStreetMap. Leaflet підтримує маркери, лінії маршрутів, шари карти та обробку подій користувача. Використання відкритих картографічних засобів дозволяє реалізувати базові функції без обов’язкового підключення комерційного сервісу.

Для оперативного оновлення положення транспортних засобів може використовуватися SignalR. Після отримання нової GPS-координати сервер передаватиме оновлення підключеному диспетчерському вебклієнту. Це зменшує необхідність у постійному повторному завантаженні даних.

Для керування версіями програмного коду використовується Git. Серверна частина та PostgreSQL можуть запускатися в контейнерах Docker, що забезпечує однакове середовище розроблення й тестування.

Обраний технологічний стек наведено в таблиці 3.1.

Для узагальнення взаємозв’язку між обраними засобами розробки їх доцільно подати у вигляді багаторівневої схеми. Клієнтський рівень охоплює технології створення веб- і мобільного інтерфейсів, серверний рівень — засоби реалізації програмного API, бізнес-логіки та оперативного обміну даними, а рівень даних і зовнішніх сервісів — систему керування базою даних

та картографічні компоненти. Узагальнену структуру технологічного стека інформаційно-аналітичної системи наведено на рисунку 3.1.

Таблиця 3.1 – Засоби реалізації інформаційно-аналітичної системи

Компонент системи	Обрана технологія	Основні причини вибору
Вебзастосунок	React, TypeScript	Компонентна структура, статична типізація, підтримка динамічних інтерфейсів
Серверна частина	ASP.NET Core, C#	Створення API, автентифікація, розмежування доступу, кросплатформність
Доступ до даних	Entity Framework Core	Робота через об’єктні моделі, підтримка міграцій і зв’язків
База даних	PostgreSQL	Реляційна модель, транзакції, індекси та обмеження цілісності
Мобільний застосунок	React Native, Expo	Підтримка мобільних платформ і доступ до геолокації
Картографічний модуль	Leaflet, OpenStreetMap	Відкрита карта, підтримка маршрутів, зупинок і маркерів
Оперативне оновлення	SignalR	Передавання оновлень від сервера до вебклієнта
Засоби розроблення	Git, Docker	Керування версіями та відтворюване середовище запуску

Наведений технологічний стек забезпечує логічне розділення системи на клієнтський, серверний і рівень даних. Веб- та мобільний застосунки відповідають за взаємодію з користувачами, серверна частина реалізує бізнес-логіку й обробку запитів, а PostgreSQL забезпечує централізоване зберігання інформації. Такий поділ спрощує розроблення, тестування та подальше супроводження програмного забезпечення.

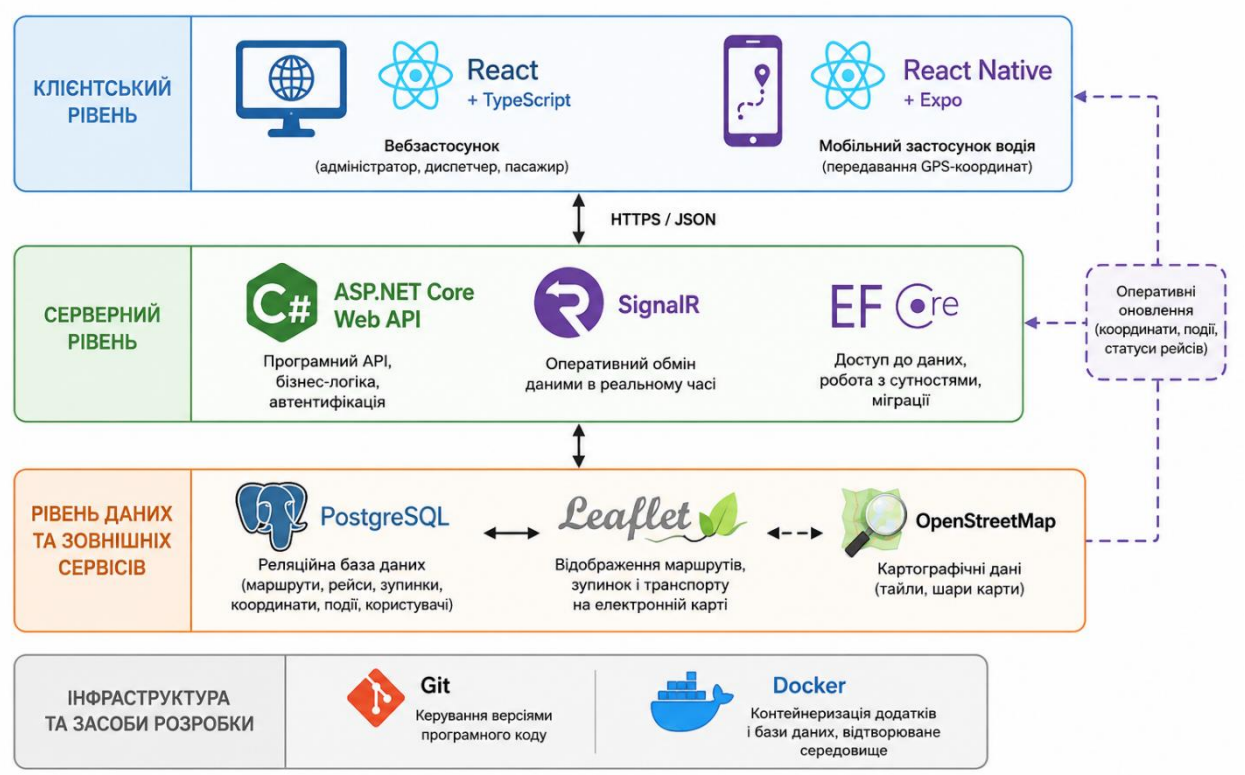


Рисунок 3.1 – Технологічний стек інформаційно-аналітичної системи

Важливою перевагою обраних технологій є їх взаємна сумісність. React і React Native використовують подібний компонентний підхід, ASP.NET Core надає єдиний API для всіх клієнтів, Entity Framework Core забезпечує зв’язок серверної логіки з реляційною базою даних, а SignalR дозволяє оперативно передавати оновлення про координати, події та статуси рейсів. Це дає змогу реалізувати всі основні функції системи в межах узгодженої архітектури.

Використання Git і Docker додатково підвищує керованість процесу розробки. Git забезпечує контроль змін програмного коду, а Docker спрощує налаштування однакового середовища для серверного застосунку та бази даних. У результаті обраний набір засобів відповідає функціональним вимогам системи та створює основу для її подальшого розширення і розгортання.

3.2 Реалізація серверної частини та бази даних

Серверна частина інформаційно-аналітичної системи реалізована на платформі ASP.NET Core із використанням мови програмування C#. Вона є

центральним компонентом системи та забезпечує взаємодію вебзастосунку, мобільного застосунку водія, модуля імітації руху та реляційної бази даних PostgreSQL.

Основними функціями серверної частини є:

– автентифікація користувачів і перевірка прав доступу;

– опрацювання запитів клієнтських застосунків;

– керування маршрутами, зупинками, водіями та транспортними засобами;

– створення й супроводження рейсів;

– приймання та перевірка GPS-координат;

– визначення подій і відхилень під час руху;

– формування даних для диспетчерського та аналітичного інтерфейсів;

– читання та збереження інформації в базі даних.

Для спрощення підтримки програмного коду серверний застосунок поділено на логічні рівні. Рівень API приймає HTTP-запити та формує відповіді. Рівень бізнес-логіки виконує перевірки й операції предметної області. Рівень доступу до даних забезпечує роботу з PostgreSQL через Entity Framework Core.

Загальну структуру серверної частини наведено на рисунку 3.2.

Контролери є точками входу до серверного API. Вони приймають запити клієнтів, перевіряють основні параметри та передають виконання відповідному сервісу. Наприклад, контролер маршрутів опрацьовує операції отримання, створення та редагування маршрутів, а контролер рейсів — операції планування, запуску й завершення рейсів.

Бізнес-логіка не розміщується безпосередньо в контролерах. Для кожної предметної області використовуються окремі серверні сервіси. Такий підхід дає змогу повторно використовувати операції, незалежно тестувати їх і зменшити зв’язність між частинами застосунку.

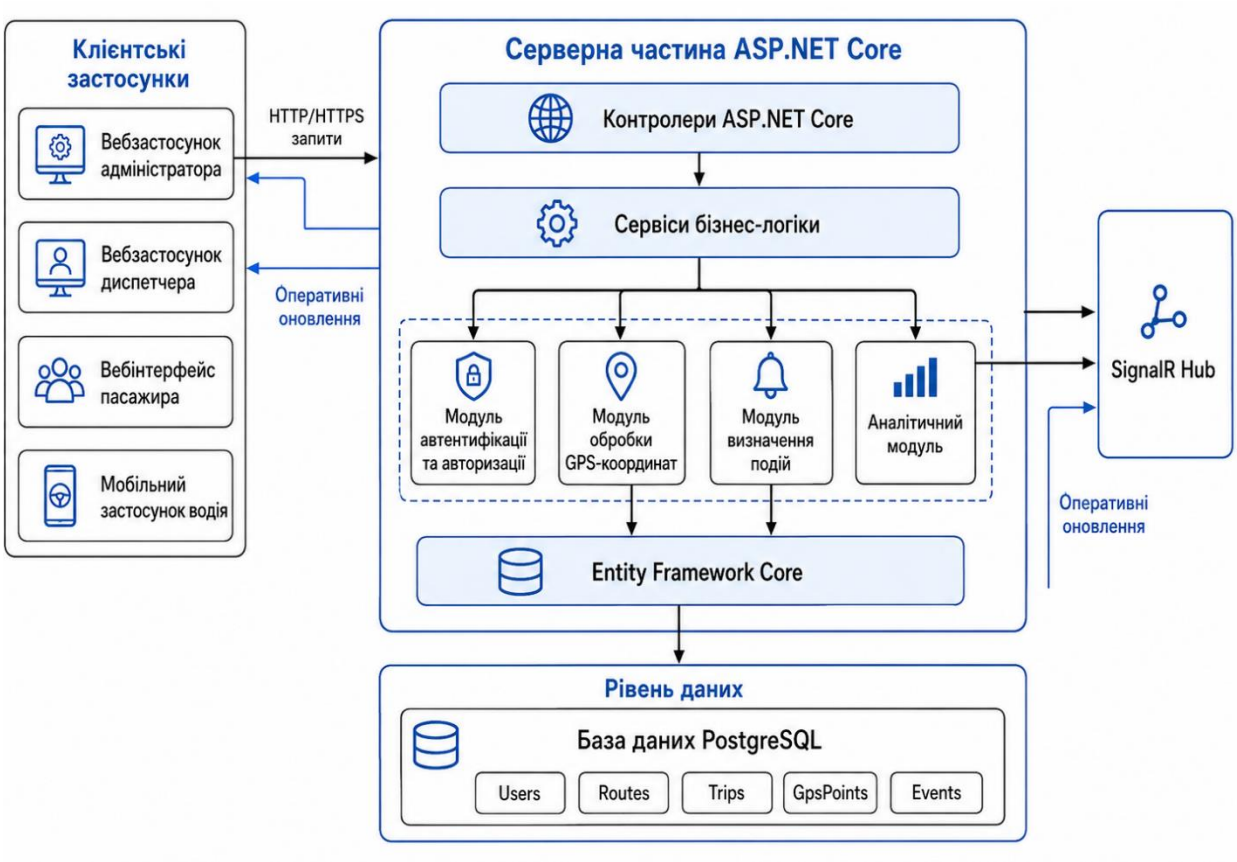


Рисунок 3.2 – Структура серверної частини інформаційно-аналітичної системи

До основних сервісів належать:

– AuthService — перевірка облікових даних і формування токенів доступу;

– UserService — керування обліковими записами й ролями;

– RouteService — робота з маршрутами, зупинками та їх послідовністю;

– VehicleService — робота з транспортними засобами;

– DriverService — робота з даними водіїв;

– TripService — створення, запуск, завершення та скасування рейсів;

– GpsService — перевірка та збереження координат;

– EventService — формування подій про відхилення;

– AnalyticsService — розрахунок показників і формування звітних даних.

Для передавання даних між клієнтом і сервером використовуються окремі моделі запитів і відповідей. Це дозволяє не передавати клієнту

внутрішні поля сутностей бази даних і встановлювати окремі правила перевірки для кожної операції.

Наприклад, запит на створення рейсу містить ідентифікатори маршруту, розкладу, водія, транспортного засобу та запланований час. Перед збереженням сервер перевіряє існування пов’язаних записів, активність маршруту, доступність транспортного засобу та відсутність часових конфліктів.

Послідовність обробки типового запиту наведено на рисунку 3.3.

Для автентифікації користувачів використовується токенний підхід. Після успішної перевірки логіна й пароля сервер формує токен, що містить ідентифікатор користувача та його роль. Під час доступу до захищених ресурсів сервер перевіряє дійсність токена та необхідні повноваження.

Паролі не зберігаються в базі даних у відкритому вигляді. Для кожного пароля формується криптографічний хеш. Це знижує ризик розкриття облікових даних у разі несанкціонованого доступу до таблиці користувачів.

Окремим компонентом серверної частини є модуль приймання GPS-координат. Мобільний застосунок або імітатор передає ідентифікатор рейсу, широту, довготу, час вимірювання та швидкість. Сервер перевіряє активність рейсу, допустимість координат, відповідність транспортного засобу та відсутність дублювання.

Після перевірки GPS-точка зберігається в таблиці GpsPoints. Одночасно оновлюється поточний стан рейсу, визначається відстань до маршруту та найближчої зупинки. Якщо виявлено порушення встановлених параметрів, створюється запис у таблиці Events.

Після отримання нової координати сервер за допомогою SignalR передає диспетчерському вебклієнту оновлене положення транспортного засобу. Завдяки цьому маркер на карті може змінювати положення без повного перезавантаження сторінки.

Для роботи з PostgreSQL використовується Entity Framework Core. Кожній таблиці бази даних відповідає клас сутності, а зв’язки між ними

задаються через навігаційні властивості та конфігурацію моделей. Основні сутності відповідають ER-діаграмі, наведеній на рисунку 2.7.

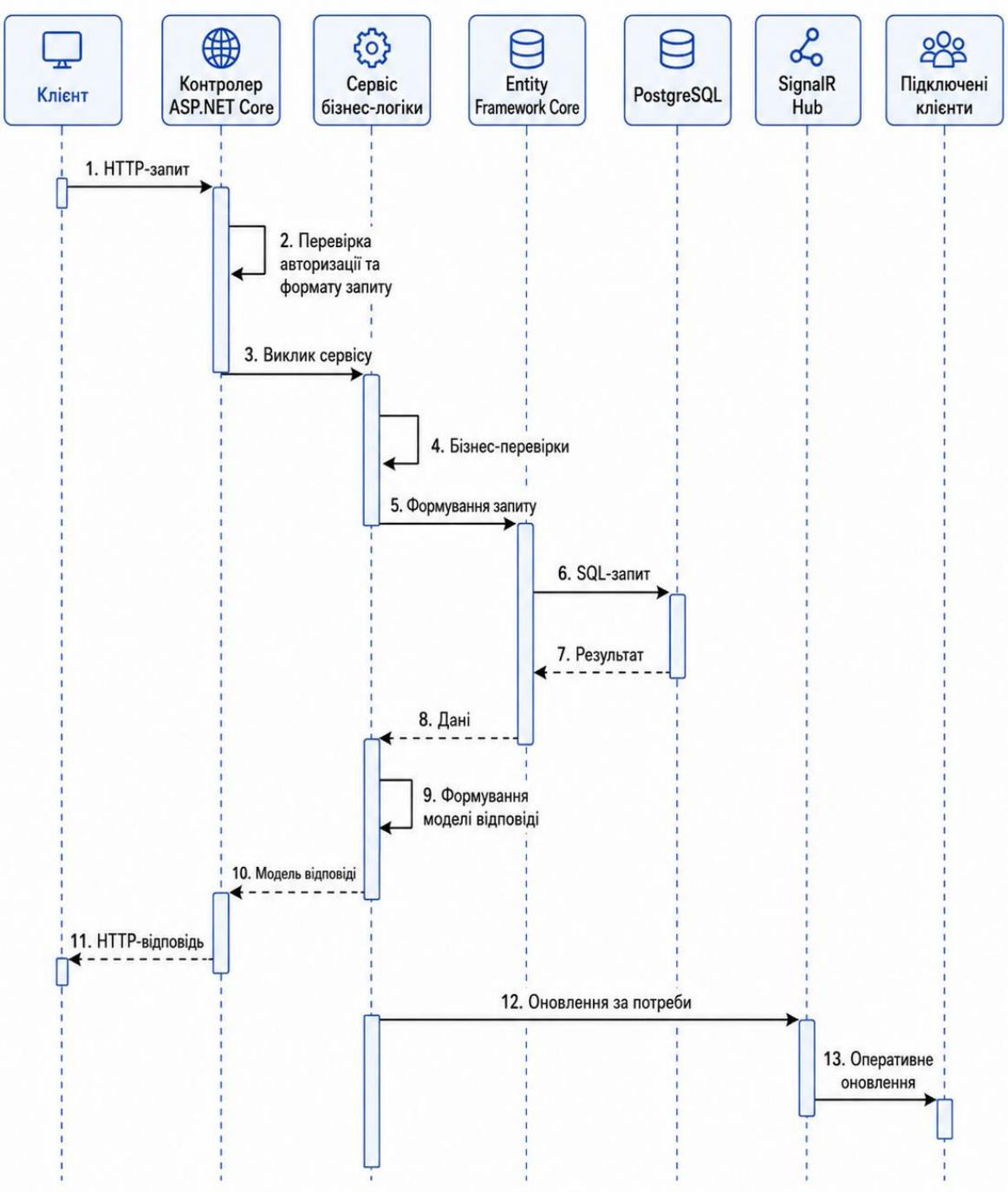


Рисунок 3.3 – Послідовність обробки запиту серверною частиною

Центральним класом рівня даних є контекст бази даних. Він містить набори сутностей Users, Roles, Drivers, Vehicles, Routes, Stops, RouteStops,

Schedules, Trips, GpsPoints і Events. Через контекст сервер виконує читання, додавання, оновлення та збереження записів.

Схему взаємодії серверних компонентів із базою даних наведено на рисунку 3.4.

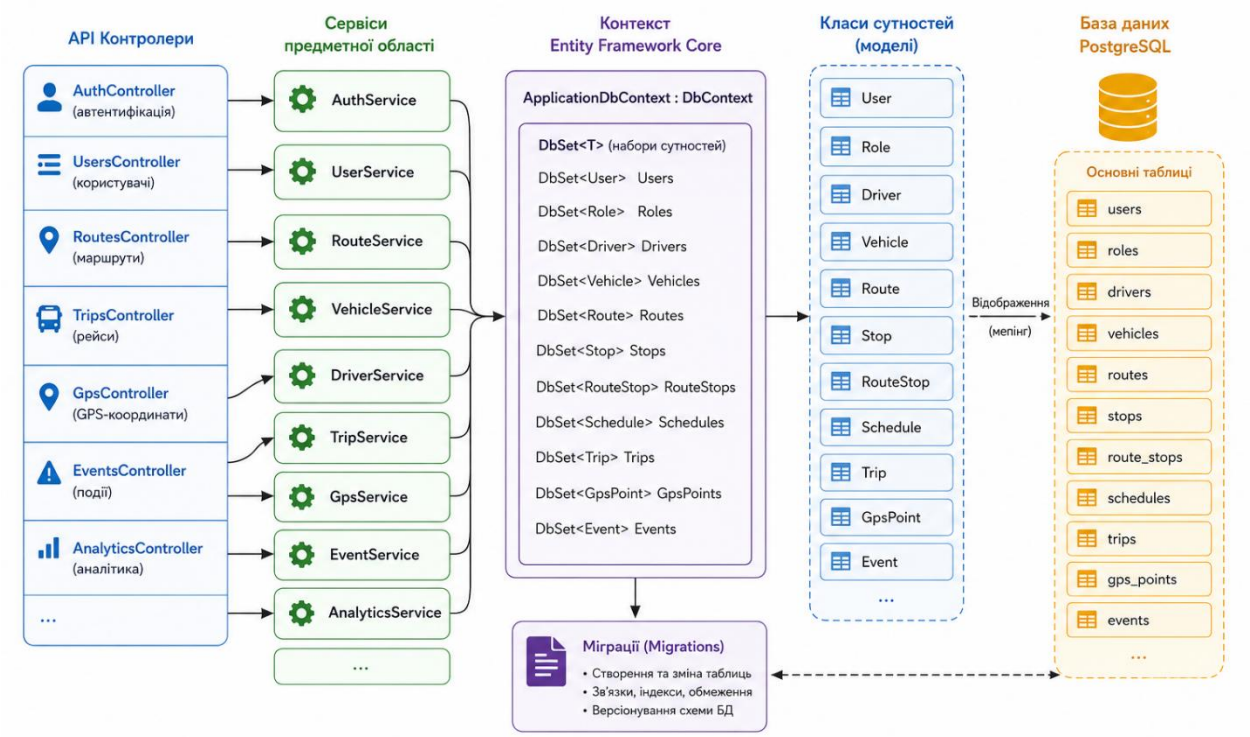


Рисунок 3.4 – Реалізація доступу серверної частини до бази даних

Структура бази даних створюється та змінюється за допомогою міграцій Entity Framework Core. Кожна міграція містить операції створення або зміни таблиць, ключів, індексів і обмежень. Це дозволяє відтворити однакову структуру бази даних у середовищі розроблення та тестування.

Для підтримання цілісності використовуються первинні й зовнішні ключі, обмеження унікальності та обов'язковості полів. Наприклад, логін користувача, бортовий номер транспортного засобу та номер посвідчення водія не повинні дублюватися.

Перед збереженням дані перевіряються на серверному рівні. Додатково основні обмеження реалізуються в базі даних. Такий подвійний контроль не

дозволяє зберігати некоректні записи навіть у випадку помилки клієнтського застосунку.

Для прискорення запитів створюються індекси за зовнішніми ключами, статусами та часовими полями. Особливе значення має індекс таблиці GpsPoints за полями trip_id і measured_at, оскільки історія руху завантажується для конкретного рейсу в хронологічному порядку.

Операції, які змінюють декілька пов’язаних записів, виконуються в межах транзакції. Наприклад, запуск рейсу передбачає зміну його статусу, фіксацію фактичного часу початку та переведення транспортного засобу в стан виконання рейсу. Якщо одна з операцій не виконується, зміни скасовуються.

Таким чином, реалізована серверна частина забезпечує централізоване виконання бізнес-логіки та єдині правила взаємодії з базою даних. Поділ на контролери, сервіси та рівень доступу до даних спрощує супроводження системи. Використання PostgreSQL, Entity Framework Core, міграцій, індексів і транзакцій забезпечує цілісне та послідовне зберігання транспортної інформації.

3.3 Реалізація веб-застосунку

Веб-застосунок інформаційно-аналітичної системи реалізовано з використанням React і TypeScript. Він забезпечує роботу адміністратора, диспетчера та пасажира й взаємодіє із серверною частиною через програмний API.

Основними завданнями веб-застосунку є:

– авторизація користувачів;

– відображення функцій відповідно до ролі;

– керування маршрутами, зупинками, водіями та транспортними засобами;

– створення й перегляд рейсів;

– відображення транспорту на електронній карті;

– перегляд повідомлень про відхилення;

– формування аналітичних таблиць і діаграм;

– надання пасажирам відкритої транспортної інформації.

Структуру веб-клієнта поділено на сторінки, компоненти, сервіси роботи з API, моделі даних і засоби керування станом. Такий поділ дозволяє повторно використовувати елементи інтерфейсу та спрощує подальше розширення системи.

До основних груп компонентів належать:

– навігаційні компоненти;

– таблиці та форми введення;

– картографічні компоненти;

– інформаційні картки;

– засоби фільтрації;

– модальні вікна;

– повідомлення про помилки й успішне виконання операцій.

Після запуску веб-застосунку користувач переходить на сторінку авторизації. На ній розміщено поля введення логіна й пароля та кнопку входу. Після відправлення форми дані передаються серверному API. У разі успішної перевірки сервер повертає токен доступу та роль користувача.

Отриманий токен використовується під час виконання захищених запитів. Якщо токен відсутній або недійсний, користувач перенаправляється на сторінку входу. Для пасажирського інтерфейсу авторизація не є обов’язковою.

Приклад сторінки авторизації наведено на рисунку 3.5.

Після входу користувач потрапляє до головного інтерфейсу, структура якого залежить від його ролі. У лівій або верхній частині сторінки розміщується навігаційне меню, а в центральній частині — робоча область.

Адміністратор отримує доступ до сторінок керування користувачами, маршрутами, зупинками, транспортними засобами, водіями та розкладами.

Для представлення довідкових даних використовуються таблиці з пошуком, фільтрацією та сортуванням.

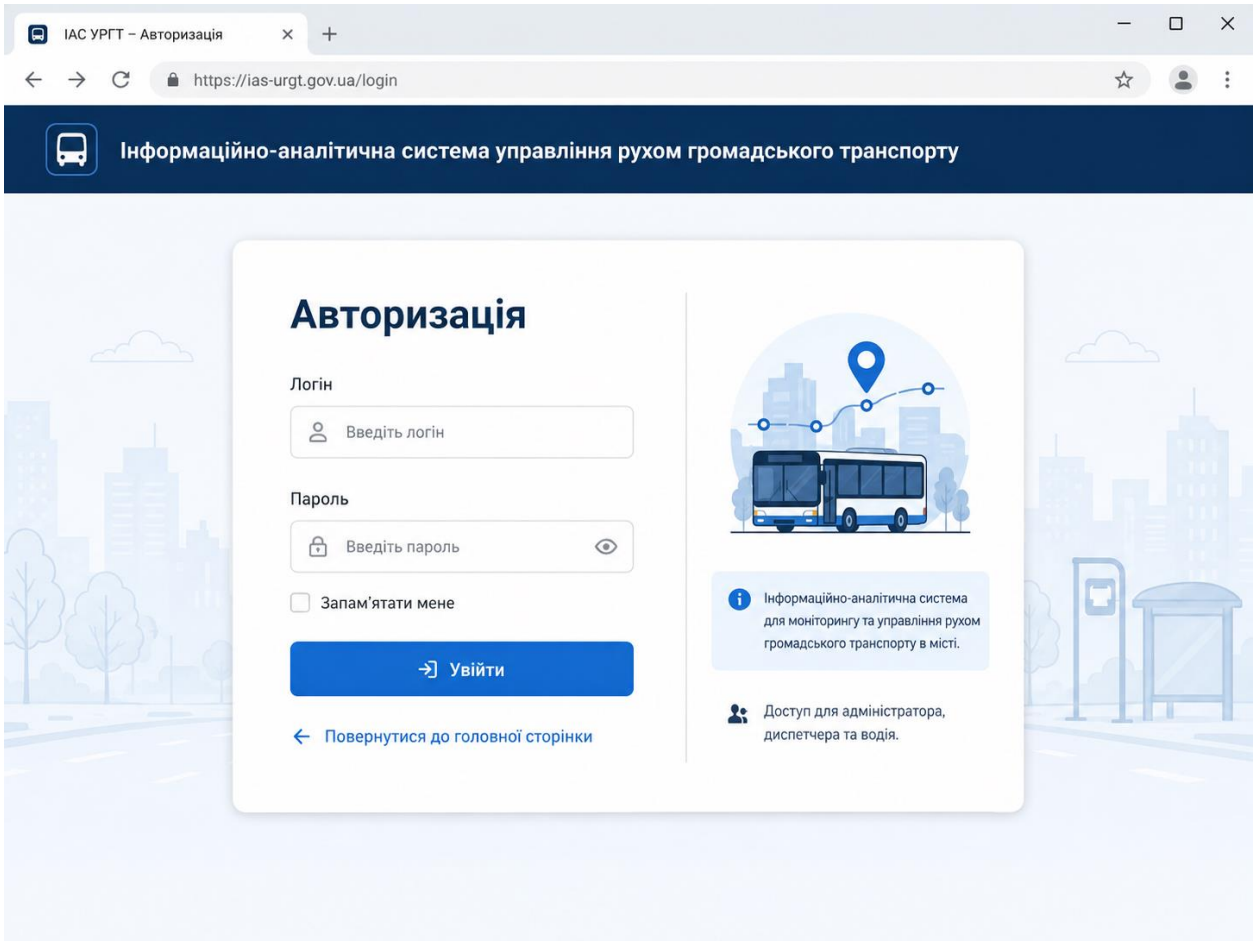


Рисунок 3.5 – Сторінка авторизації у веб-застосунку

Кожна таблиця містить основні відомості про записи та кнопки доступних дій. Наприклад, сторінка маршрутів відображає номер, назву, тип транспорту, напрямок і статус. Адміністратор може створити новий маршрут, відредагувати наявний або змінити його стан.

Форми додавання й редагування реалізовано у вигляді окремих сторінок або модальних вікон. Перед відправленням даних виконується клієнтська перевірка обов’язкових полів. Після цього введені значення передаються серверу, де проходять повторну перевірку.

Приклад адміністративної сторінки наведено на рисунку 3.6.

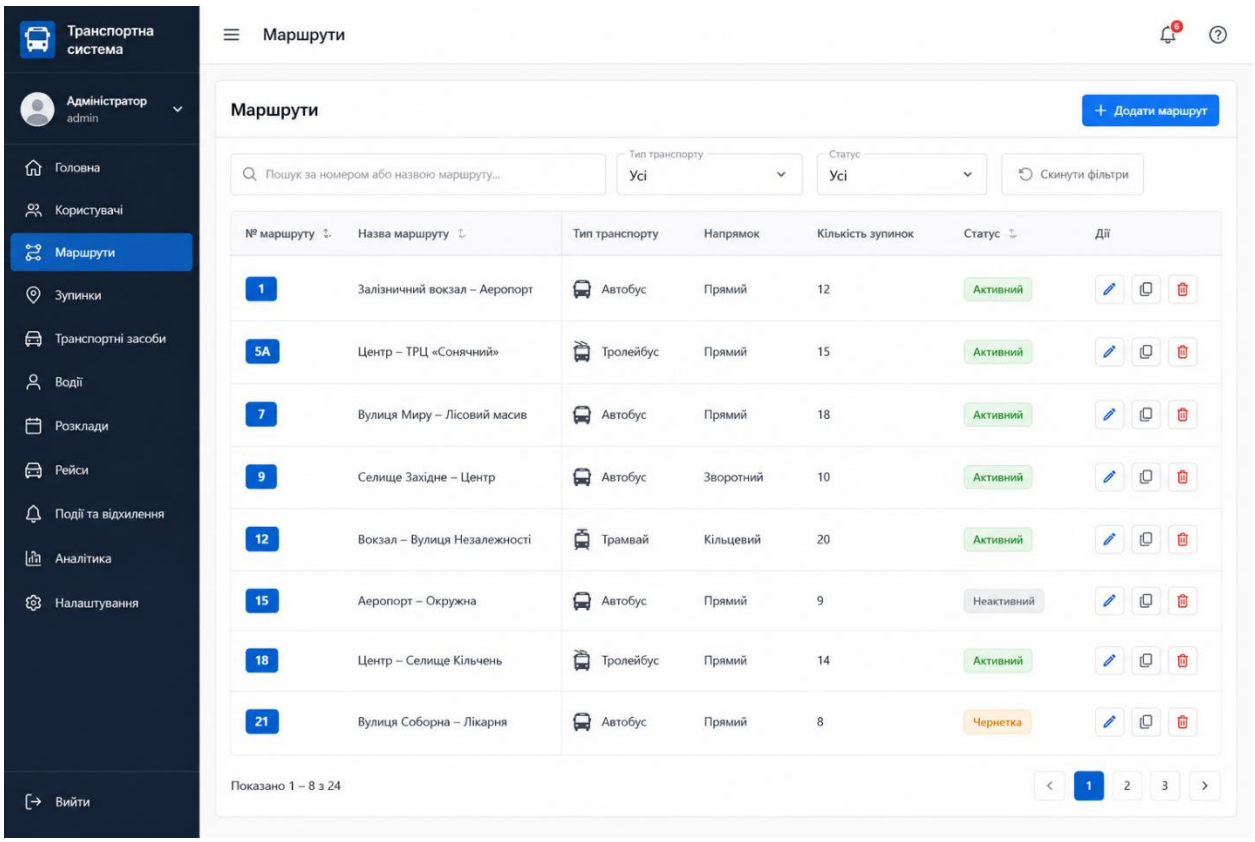


Рисунок 3.6 – Сторінка керування маршрутами у веб-застосунку

Інтерфейс диспетчера орієнтований на оперативний контроль руху. Центральним елементом сторінки є електронна карта, на якій відображаються активні транспортні засоби, маршрути й зупинки.

Кожен транспортний засіб подається на карті у вигляді маркера. Після вибору маркера відображаються номер маршруту, бортовий номер, водій, статус рейсу, швидкість і час останнього оновлення координат.

Для побудови карти використано бібліотеку Leaflet і картографічні шари OpenStreetMap. Геометрія маршруту відображається у вигляді лінії, а зупинки — окремими позначками.

Поруч із картою розміщується список активних рейсів. Для кожного рейсу відображаються маршрут, транспортний засіб, водій, час початку та поточний стан. Диспетчер може фільтрувати записи за маршрутом, статусом або транспортним засобом.

Рейси, для яких виявлено відхилення, виділяються візуально. Додатково відображається короткий текст повідомлення, наприклад «Запізнення», «Відхилення від маршруту», «Тривала зупинка» або «Втрата зв’язку».

Оперативне оновлення координат здійснюється через SignalR. Після отримання нової GPS-точки сервер передає її підключеним клієнтам, а маркер транспортного засобу змінює положення без перезавантаження сторінки.

Приклад диспетчерського інтерфейсу наведено на рисунку 3.7.

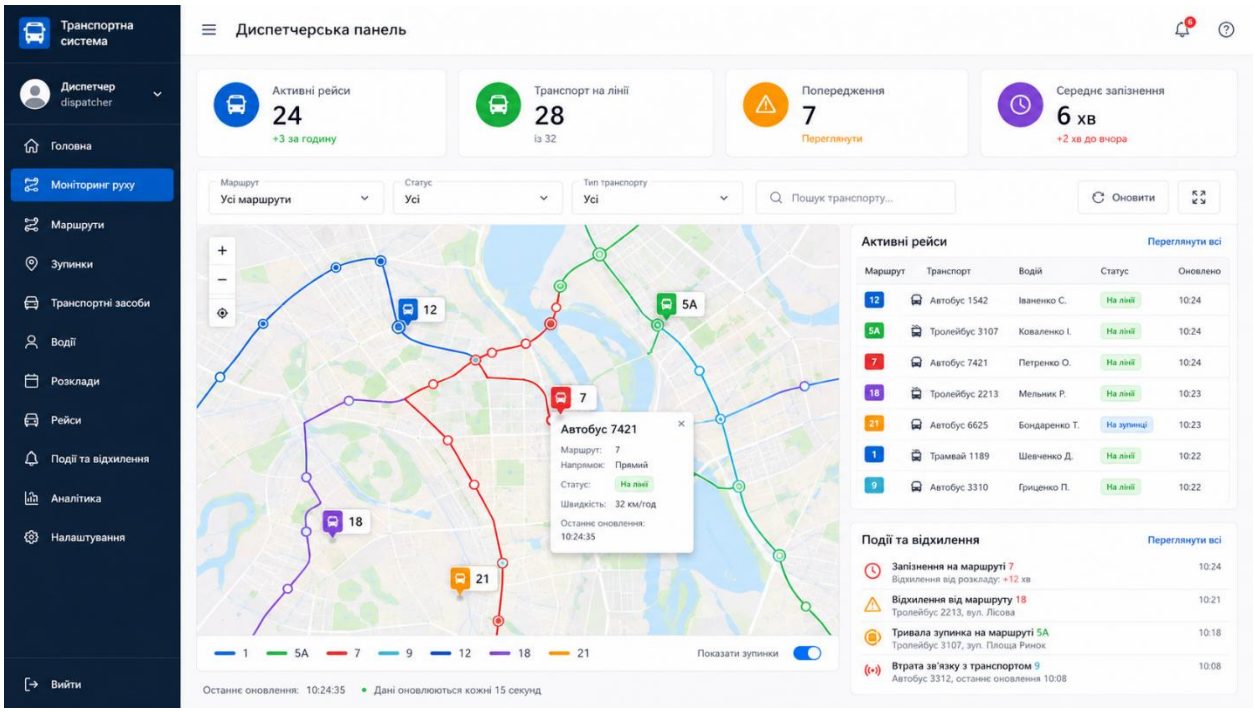


Рисунок 3.7 – Диспетчерська панель моніторингу руху

Для перегляду історії руху реалізовано окрему сторінку рейсу. На ній відображаються загальні відомості, фактичний час початку й завершення, маршрут, список подій та послідовність GPS-координат.

Історична траєкторія може відображатися на карті у вигляді лінії. Це дозволяє визначити ділянки відхилення та порівняти фактичний рух із запланованим маршрутом.

Аналітична частина веб-застосунку містить фільтри за часовим періодом, маршрутом і транспортним засобом. Після вибору параметрів сервер повертає розраховані показники.

До основних показників належать:

– кількість запланованих рейсів;

– кількість виконаних і скасованих рейсів;

– середнє запізнення;

– середня фактична тривалість рейсу;

– кількість випадків втрати зв’язку;

– кількість відхилень від маршруту.

Отримані дані подаються у вигляді інформаційних карток, таблиць і діаграм. Графічне подання дозволяє швидко оцінити стан маршрутів і виявити повторювані проблеми.

Пасажирський інтерфейс має спрощену структуру. Користувач може вибрати маршрут, переглянути послідовність зупинок і побачити транспортні засоби на карті.

У відкритому інтерфейсі не відображаються персональні дані водіїв, службові коментарі та внутрішні події транспортного підприємства. Пасажиру доступні лише номер маршруту, напрямок, зупинки, розклад, поточне положення транспорту й час оновлення даних.

Приклад пасажирського інтерфейсу наведено на рисунку 3.8.

Для взаємодії із сервером у вебзастосунку створено окремий сервісний рівень. Кожна група запитів має відповідний модуль, наприклад для користувачів, маршрутів, рейсів, координат і аналітики.

Такий підхід дозволяє не розміщувати код HTTP-запитів безпосередньо в компонентах інтерфейсу. Компонент викликає функцію сервісу й отримує готові дані або повідомлення про помилку.

Під час виконання запитів користувачеві відображається індикатор завантаження. У разі помилки система показує зрозуміле повідомлення.

Наприклад, якщо сервер недоступний, користувач отримує повідомлення про неможливість завантажити дані, а не лише технічний код помилки.

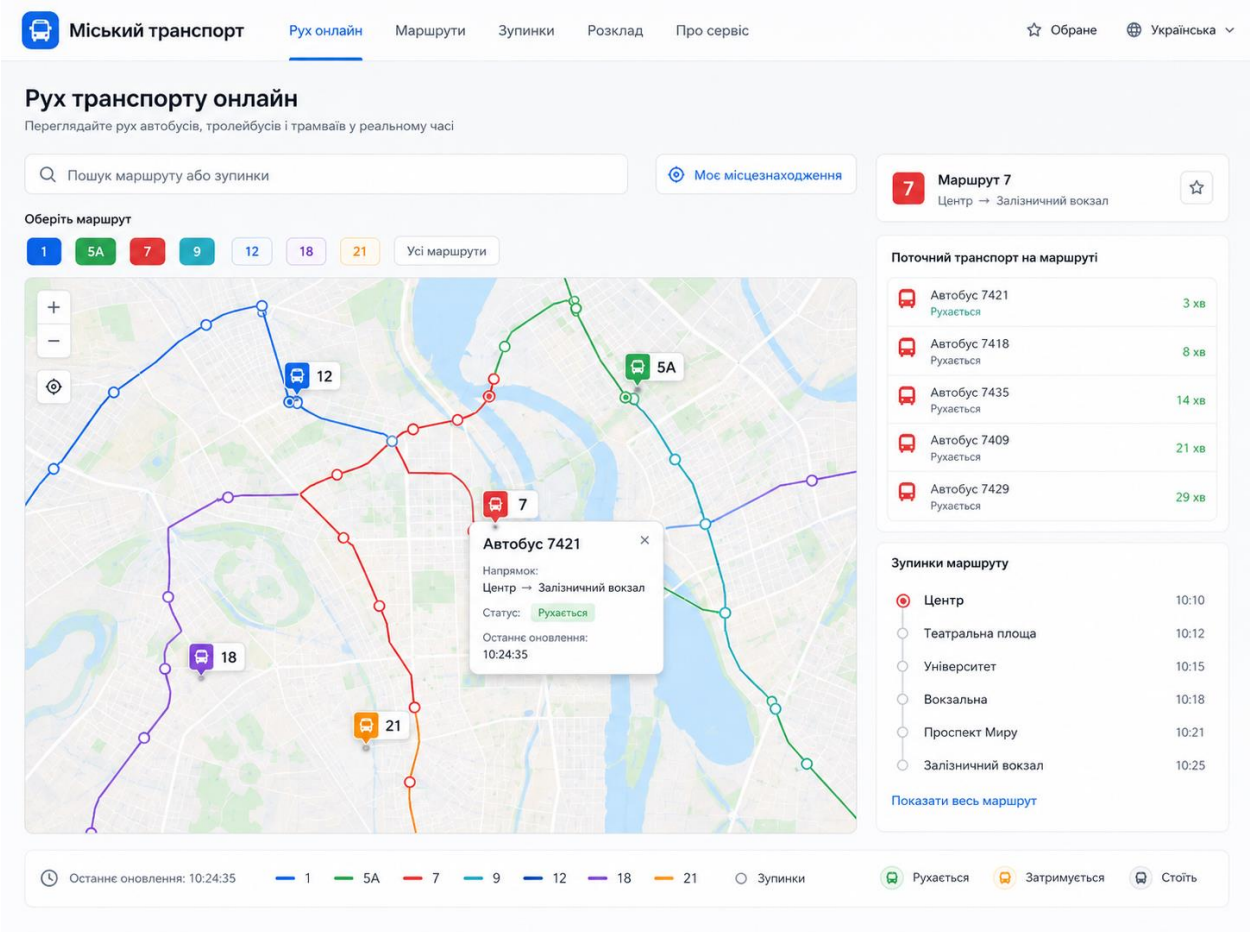


Рисунок 3.8 – Пасажирська сторінка перегляду руху транспорту

Особливу увагу приділено підтвердженню критичних операцій. Перед скасуванням рейсу, деактивацією користувача або зміною статусу транспортного засобу відкривається додаткове діалогове вікно.

Інтерфейс адаптується до різних розмірів екрана. На вузьких екранах навігаційне меню згортається, таблиці можуть прокручуватися горизонтально, а інформаційні блоки розміщуються один під одним.

Для диспетчерської панелі основним залишається використання настільного комп’ютера або ноутбука, оскільки одночасний перегляд карти, списку рейсів і подій потребує достатньої площі екрана.

Усі сторінки мають єдиний стиль оформлення. Для заголовків, кнопок, таблиць, форм і повідомлень використовуються спільні компоненти. Статуси доповнюються текстовими позначеннями та піктограмами, тому інформація не залежить лише від кольору.

Таким чином, реалізований вебзастосунок забезпечує окремі інтерфейси для адміністратора, диспетчера та пасажира. Компонентна структура React, типізація TypeScript, інтеграція з серверним API, Leaflet і SignalR дозволяють виконувати операції керування, відображати транспорт на карті та оперативно оновлювати дані про рух.

3.4 Реалізація мобільного застосунку та імітатора руху

Мобільний застосунок водія призначений для отримання GPS-координат транспортного засобу та їх передавання серверній частині під час виконання рейсу. Для його реалізації використано React Native — засіб створення мобільних застосунків на основі компонентного підходу React [22]. Це дозволяє організувати інтерфейс у вигляді окремих екранів і повторно використовуваних компонентів.

Функціональність мобільного застосунку навмисно обмежено основними операціями, необхідними водієві. Це зменшує кількість елементів керування та не перевантажує інтерфейс під час виконання рейсу.

До основних функцій застосунку належать:

– авторизація водія;

– перегляд призначених рейсів;

– вибір поточного рейсу;

– запуск і завершення виконання рейсу;

– отримання координат мобільного пристрою;

– передавання координат серверному API;

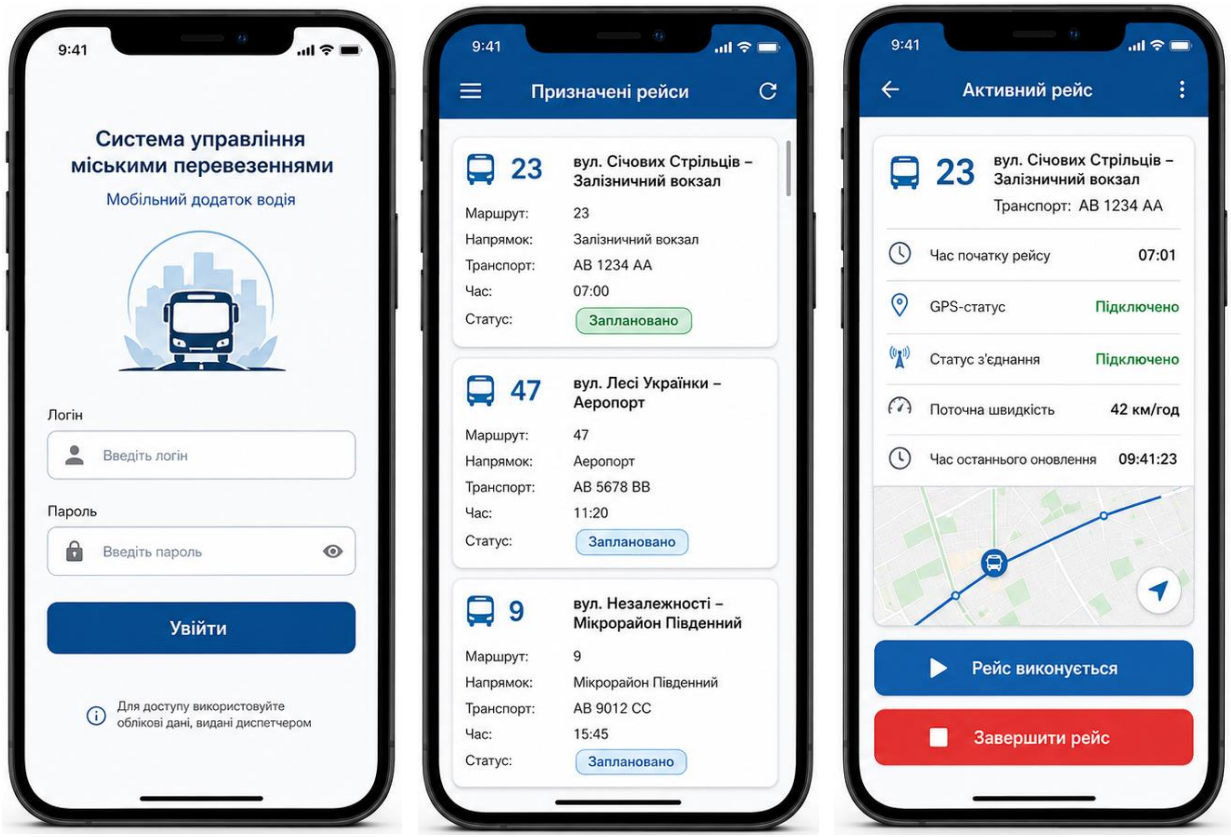
– відображення стану GPS і мережевого з’єднання;

– повідомлення про помилки передавання даних.

Після запуску застосунку водій переходить на сторінку авторизації. Він вводить логін і пароль, після чого застосунок надсилає запит серверу. У разі успішної перевірки сервер повертає токен доступу, який використовується для подальших захищених запитів.

Після авторизації водієві відображається список призначених рейсів. Для кожного рейсу зазначаються номер маршруту, транспортний засіб, напрямок, запланований час початку та поточний статус. Водій може вибрати лише той рейс, який призначено його обліковому запису.

Основні екрани мобільного застосунку наведено на рисунку 3.9.



а) авторизація водія; б) список призначених рейсів; в) екран активного рейсу

Рисунок 3.9 – Основні екрани мобільного застосунку водія

Перед початком руху застосунок перевіряє наявність дозволу на доступ до геолокації. Отримання поточного місцеположення та відстеження змін координат реалізується за допомогою засобів Expo Location [23]. Якщо дозвіл

не надано, користувачеві відображається відповідне повідомлення. Передавання координат розпочинається лише після підтвердження доступу та натискання кнопки запуску рейсу.

На екрані активного рейсу відображаються номер маршруту, бортовий номер транспортного засобу, час початку, поточна швидкість, стан GPS та час останнього успішного передавання координат. Основними елементами керування є кнопки запуску та завершення рейсу.

Послідовність передавання GPS-даних наведено на рисунку 3.10.

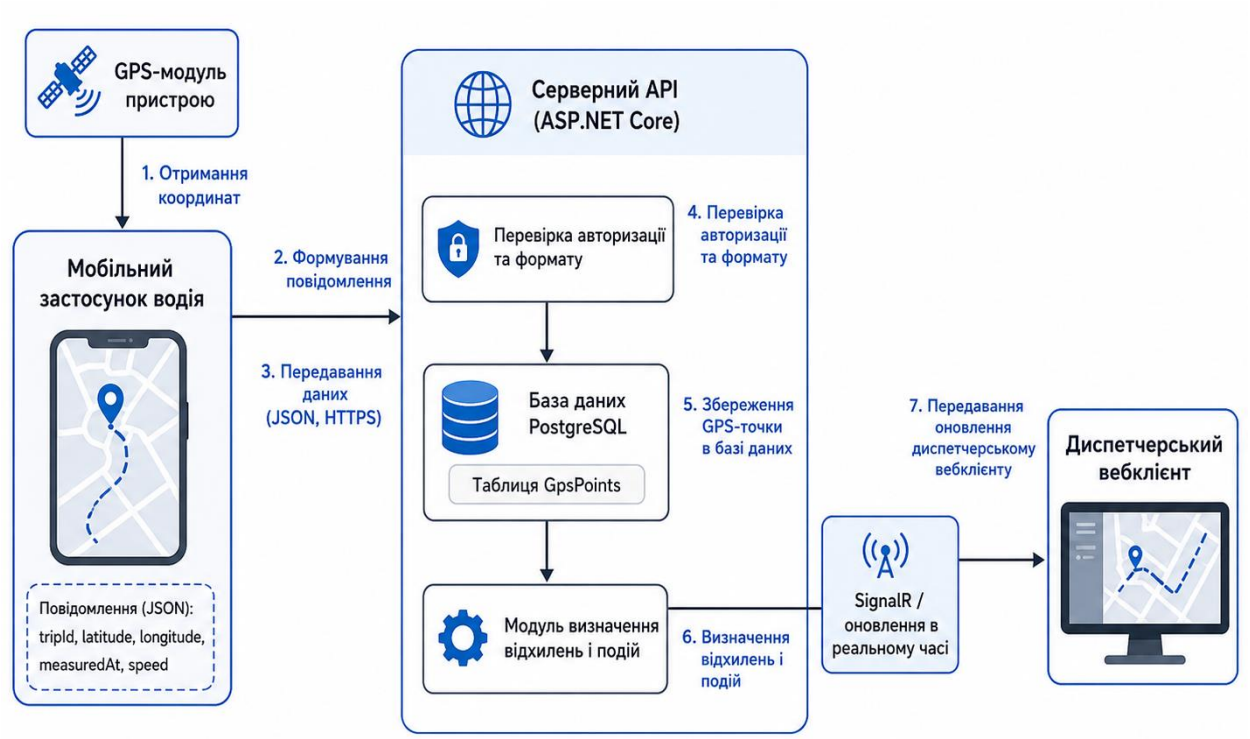


Рисунок 3.10 – Схема передавання GPS-координат мобільним застосунком

Частота оновлення координат повинна забезпечувати достатню точність відображення руху та не створювати надмірного навантаження на сервер і мобільний пристрій. У тестовій версії координати можуть передаватися через фіксований інтервал, наприклад один раз на 10–15 секунд.

У разі відсутності інтернет-з’єднання застосунок відображає стан помилки передавання. За потреби координати можуть тимчасово зберігатися в локальному буфері та надсилатися після відновлення зв’язку. При цьому

зберігається початковий час вимірювання, щоб сервер міг відрізнити актуальні та затримані дані.

Під час завершення рейсу застосунок припиняє отримання координат і надсилає серверу запит на зміну статусу рейсу. Сервер фіксує фактичний час завершення та оновлює стан транспортного засобу.

Для перевірки підсистеми GPS-моніторингу без постійного використання реального мобільного пристрою реалізовано програмний імітатор руху. Він виконує роль тестового клієнта та передає координати через той самий серверний маршрут API, що й мобільний застосунок.

Імітатор отримує координати маршруту у вигляді впорядкованої послідовності точок. Після вибору рейсу користувач задає швидкість, частоту оновлення та режим роботи. Модуль послідовно переходить між точками маршруту й формує тестові GPS-повідомлення.

До основних режимів імітації належать:

– нормальний рух;

– рух зі зниженою швидкістю;

– запізнення;

– тривала зупинка;

– відхилення від маршруту;

– тимчасова втрата зв’язку.

У нормальному режимі координати передаються послідовно вздовж геометрії маршруту. У режимі зниженої швидкості перехід між точками виконується повільніше. Для моделювання тривалої зупинки одна й та сама координата передається протягом заданого періоду.

Відхилення від маршруту моделюється шляхом додавання зміщення до координат або використання окремої послідовності точок. У режимі втрати зв’язку передавання даних тимчасово припиняється, після чого може бути автоматично відновлене.

Інтерфейс модуля імітації руху наведено на рисунку 3.11.

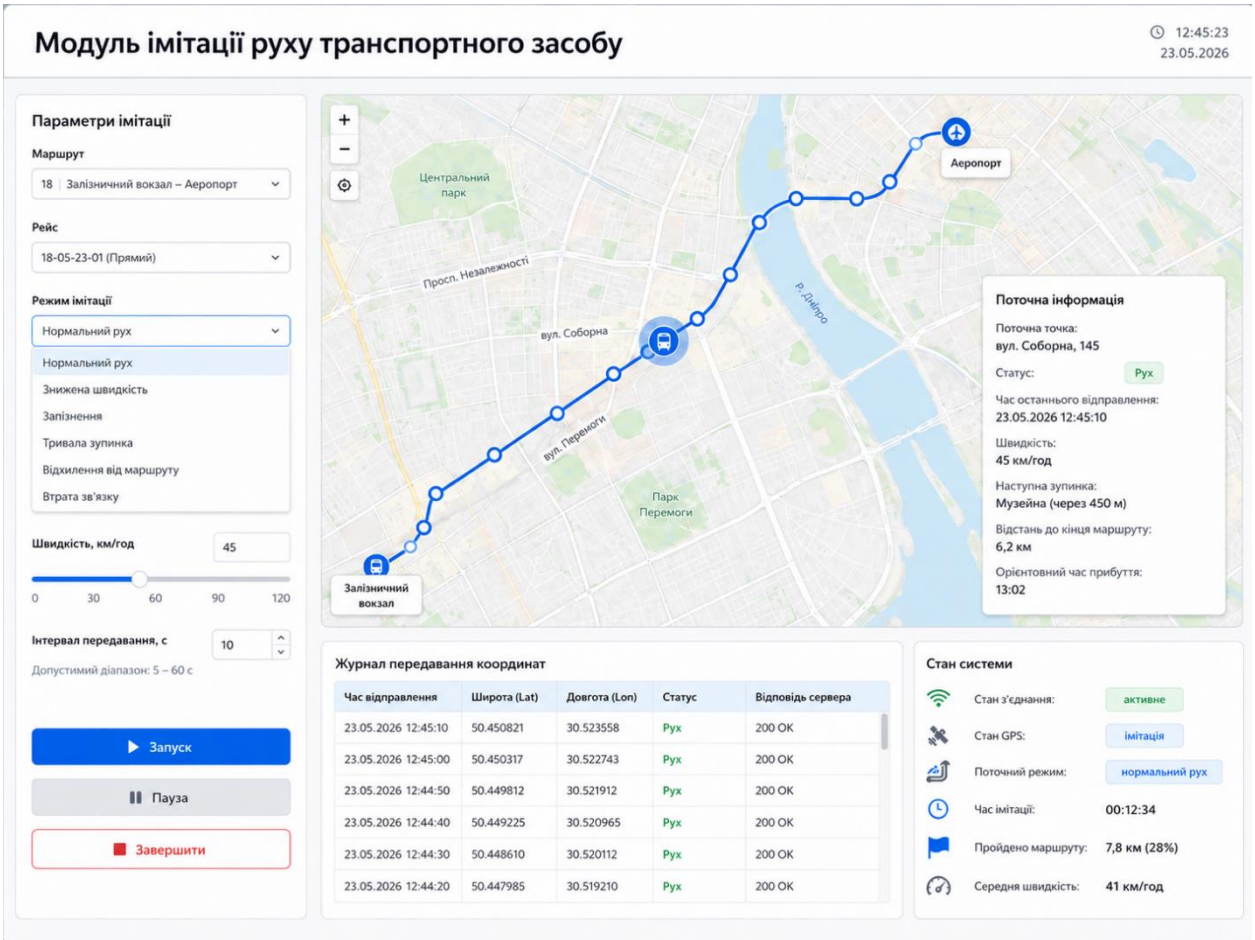


Рисунок 3.11 – Інтерфейс модуля імітації руху транспортного засобу

Імітатор дозволяє перевірити повний цикл обробки GPS-даних: від формування координати до її відображення на диспетчерській карті. Також за його допомогою можна протестувати формування подій про відхилення, затримки, зупинки та втрату зв’язку.

Для уникнення впливу тестових даних на реальні рейси записи імітатора мають позначатися окремим значенням поля source. Це дає змогу фільтрувати їх під час формування звітів або використовувати окреме тестове середовище.

Мобільний застосунок та імітатор використовують однаковий формат повідомлень і один серверний API. Завдяки цьому серверна логіка перевірки, збереження та обробки координат не залежить від джерела даних.

Таким чином, мобільний застосунок забезпечує передавання фактичних координат водієм під час виконання рейсу, а модуль імітації дає змогу перевіряти систему в контрольованих умовах. Спільний програмний

інтерфейс забезпечує однакову обробку реальних і тестових GPS-даних та спрощує функціональне тестування інформаційно-аналітичної системи.

3.5 Реалізація інформаційно-аналітичної підсистеми

Інформаційно-аналітична підсистема призначена для обробки накопичених даних про виконання рейсів і формування показників, необхідних диспетчеру та адміністратору. Джерелами інформації є таблиці Trips, GpsPoints, Events, Routes і Vehicles, які містять відомості про рейси, координати транспортних засобів, зафіксовані події, маршрути та рухомий склад.

Аналітичну логіку реалізовано в окремому серверному модулі AnalyticsService. Він виконує вибірку даних із бази, застосовує задані фільтри, групує записи та розраховує узагальнені показники. До них належать кількість запланованих, виконаних і скасованих рейсів, середня тривалість поїздки, середнє відхилення від розкладу, кількість випадків втрати зв’язку та відхилення від маршруту.

Для формування звіту користувач може задати часовий період, маршрут, транспортний засіб або статус рейсу. Після цього сервер опрацьовує відповідну вибірку та формує модель відповіді для вебзастосунку. Отримані результати відображаються у вигляді інформаційних карток, таблиць, діаграм і звітів.

Функціональну схему інформаційно-аналітичної підсистеми наведено на рисунку 3.12.

Наведена схема показує, що підсистема не створює окремих дублікатів оперативної інформації, а використовує дані, уже збережені в основній базі. Це забезпечує узгодженість результатів і виключає необхідність підтримувати додаткове сховище для аналітики.

Повний набір аналітичних даних доступний адміністратору та диспетчеру. Пасажирський інтерфейс не містить внутрішніх службових показників, історії подій і детальної інформації про виконання рейсів. Такий

розподіл відповідає визначеним ролям користувачів і забезпечує захист службової інформації.

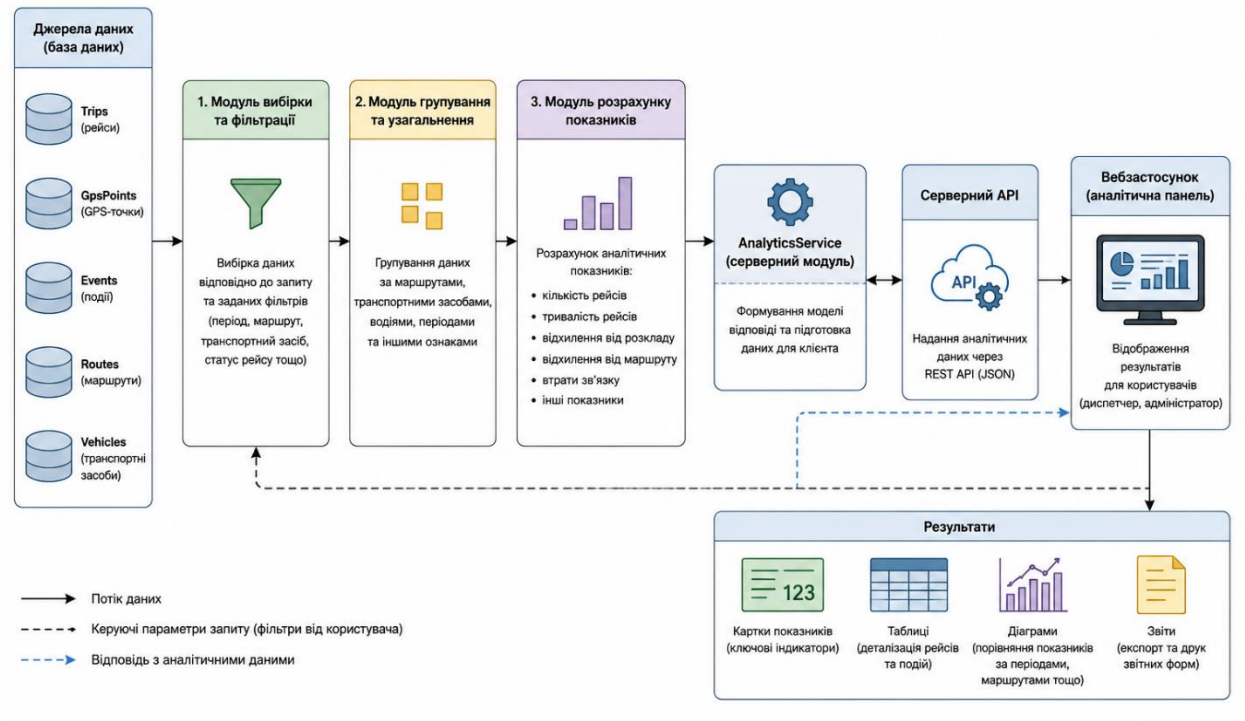


Рисунок 3.12 – Функціональна схема інформаційно-аналітичної підсистеми

Коректність розрахунків перевіряється як на даних фактично виконаних рейсів, так і на тестових записах, сформованих модулем імітації руху. Це дозволяє перевірити обчислення тривалості рейсів, кількості подій, запізнень і відхилень для різних сценаріїв роботи системи.

Таким чином, реалізована інформаційно-аналітична підсистема забезпечує централізоване формування показників на основі оперативних і довідкових даних. Її використання спрощує контроль виконання рейсів, прискорює пошук проблемних маршрутів і створює основу для подальшого розширення функцій звітності та аналізу.

4 ТЕСТУВАННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ РОЗРОБКИ

4.1 Організація та методика тестування

Тестування інформаційно-аналітичної системи проводиться з метою перевірки відповідності реалізованого програмного забезпечення сформованим вимогам. Під час тестування оцінюється робота серверної частини, бази даних, вебзастосунку, мобільного застосунку водія та модуля імітації руху.

Організація тестування базується на загальних положеннях стандартів ISO/IEC/IEEE 29119, які визначають основні поняття та процеси тестування програмного забезпечення [24; 25].

Основними завданнями тестування є:

– перевірка правильності виконання функцій системи;

– перевірка взаємодії між програмними компонентами;

– перевірка розмежування прав доступу;

– перевірка збереження та обробки GPS-координат;

– перевірка визначення відхилень від маршруту й розкладу;

– оцінювання коректності користувацьких інтерфейсів.

У межах роботи застосовуються функціональне, інтеграційне та системне тестування. Функціональне тестування перевіряє окремі операції з позиції користувача. Інтеграційне тестування використовується для перевірки взаємодії веб- і мобільного клієнтів із серверним API та базою даних. Системне тестування охоплює повний цикл роботи системи.

Тестове середовище включає серверний застосунок ASP.NET Core, тестову базу даних PostgreSQL, вебзастосунок, мобільний застосунок або його емулятор і модуль імітації руху. База даних заповнюється контрольними відомостями про користувачів, маршрути, транспортні засоби, водіїв і рейси.

Для перевірки GPS-моніторингу модуль імітації відтворює нормальний рух, запізнення, тривалу зупинку, відхилення від маршруту та втрату зв’язку.

Це дозволяє перевірити реакцію серверної частини та диспетчерського інтерфейсу без використання реального транспортного засобу.

Перевірки оформлюються у вигляді тест-кейсів. Структуру тест-кейсу наведено в таблиці 4.1.

Таблиця 4.1 – Структура тест-кейсу

Поле	Опис
Ідентифікатор	Унікальний номер перевірки
Назва	Функція або сценарій, що перевіряється
Початкові умови	Стан системи перед виконанням тесту
Послідовність дій	Кроки користувача або тестового модуля
Очікуваний результат	Правильна реакція системи
Фактичний результат	Результат виконання перевірки
Статус	Успішно або неуспішно

Під час позитивного тестування використовуються коректні вхідні дані. Наприклад, перевіряється успішна авторизація, створення рейсу та збереження координати активного транспортного засобу.

Негативне тестування включає неправильний пароль, відсутність обов’язкових полів, недопустимі координати, передавання даних для завершеного рейсу та спроби доступу до функцій, які не відповідають ролі користувача.

Тест вважається успішним, якщо фактичний результат відповідає очікуваному, дані коректно зберігаються в базі, а помилкові дії не порушують працездатність і цілісність системи. Після виправлення виявленої помилки відповідний тест виконується повторно.

Таким чином, обрана методика охоплює перевірку основних функцій, взаємодії компонентів і повних користувацьких сценаріїв. Використання контрольних даних та імітатора руху забезпечує повторюваність тестування й дозволяє оцінити результати розробки.

4.2 Функціональне тестування системи

Функціональне тестування виконувалося для перевірки основних операцій інформаційно-аналітичної системи відповідно до вимог, сформованих у підрозділі 2.1. Перевірки охоплювали авторизацію користувачів, розмежування доступу, роботу з транспортними даними, створення рейсів, приймання GPS-координат, відображення транспорту на карті та формування аналітичних показників.

Тестування проводилося в середовищі, що включало серверний застосунок ASP.NET Core, тестову базу даних PostgreSQL, вебзастосунок, мобільний клієнт та модуль імітації руху. Для виконання перевірок було підготовлено облікові записи адміністратора, диспетчера й водія, а також контрольні записи маршрутів, зупинок, транспортних засобів і рейсів.

Спочатку перевірено механізм авторизації. Користувач із правильними обліковими даними отримував доступ до системи відповідно до своєї ролі. У разі введення неправильного пароля сервер відхиляв запит і повертав повідомлення про помилку. Також було перевірено, що водій не може отримати доступ до адміністративних сторінок, а пасажир не має доступу до службових даних.

Операції керування маршрутами, зупинками, водіями та транспортними засобами перевірялися шляхом створення, редагування та деактивації записів. Під час введення неповних або некоректних даних система не виконувала збереження та відображала відповідне повідомлення.

Під час тестування рейсів диспетчер створював новий рейс, вибирав маршрут, розклад, водія і транспортний засіб. Сервер перевіряв наявність пов’язаних записів та їх доступність. Після запуску рейсу його статус змінювався на активний, а після завершення фіксувався фактичний час закінчення.

Для перевірки GPS-моніторингу використовувався мобільний застосунок та імітатор руху. Коректні координати активного рейсу зберігалися

в таблиці GpsPoints і відображалися на диспетчерській карті. Координати з недопустимими значеннями широти або довготи відхилялися сервером.

Окремо перевірено формування подій. У разі тривалої відсутності координат створювалася подія втрати зв’язку. Відхилення від маршруту та перевищення допустимого запізнення також фіксувалися в таблиці Events і відображалися диспетчеру.

Результати основних функціональних перевірок наведено в таблиці 4.2.

Таблиця 4.2 – Результати функціонального тестування системи

№	Функція	Тестовий сценарій	Очікуваний результат	Результат
1	2	3	4	5
1	Авторизація	Введення правильного логіна й пароля	Вхід до системи та визначення ролі	Успішно
2	Авторизація	Введення неправильного пароля	Відмова у вході та повідомлення про помилку	Успішно
3	Розмежування доступу	Водій відкриває адміністративну сторінку	Доступ заборонено	Успішно
4	Керування маршрутами	Створення маршруту з коректними даними	Запис збережено в базі даних	Успішно
5	Перевірка форми	Створення маршруту без номера	Дані не збережено, показано помилку	Успішно
6	Створення рейсу	Призначення маршруту, водія і транспортного засобу	Рейс створено зі статусом «запланований»	Успішно
7	Запуск рейсу	Водій запускає призначений рейс	Статус змінено на «активний»	Успішно
8	Передавання GPS	Надсилання коректної координати	Координату збережено та показано на карті	Успішно

Продовження табл. 4.2

1	2	3	4	5
9	Перевірка GPS	Надсилання недопустимої координати	Запит відхилено, запис не створено	Успішно
10	Втрата зв’язку	Відсутність координат понад допустимий час	Створено відповідну подію	Успішно
11	Відхилення від маршруту	Передавання кількох точок поза маршрутом	Сформовано повідомлення диспетчеру	Успішно
12	Аналітика	Формування звіту за маршрутом і періодом	Відображено узагальнені показники	Успішно

Фактичні результати відповідали очікуваним для всіх наведених сценаріїв. Серверна частина коректно перевіряла вхідні дані, а помилкові операції не призводили до створення некоректних записів у базі даних.

Під час перевірки взаємодії компонентів нові GPS-координати передавалися диспетчерському вебклієнту, після чого положення транспортного засобу на карті оновлювалося. Дані про завершені рейси використовувалися для розрахунку тривалості руху, кількості подій та відхилень.

Таким чином, функціональне тестування підтвердило працездатність основних можливостей системи. Реалізовані операції авторизації, адміністрування, керування рейсами, GPS-моніторингу та аналітики відповідають сформованим вимогам.

4.3 Тестування руху транспортних засобів

Тестування руху транспортних засобів виконувалося з метою перевірки правильності приймання GPS-координат, відображення положення транспорту на карті та формування подій у разі відхилення від установлених

параметрів. Для проведення перевірок використано модуль імітації руху, який передавав серверній частині координати у форматі, аналогічному до мобільного застосунку водія.

Перед початком тестування в системі було створено маршрут із послідовністю зупинок, транспортний засіб, водія та активний рейс. В імітаторі задавалися маршрут, швидкість руху, інтервал передавання координат і режим роботи.

У нормальному режимі модуль послідовно формував координати вздовж заданого маршруту. Сервер перевіряв отримані значення, зберігав їх у таблиці GpsPoints і передавав оновлене положення диспетчерському вебклієнту. Маркер транспортного засобу на карті змінював положення відповідно до отриманих координат.

Під час проходження контрольних зон зупинок система фіксувала фактичний час їх досягнення. Отримані значення порівнювалися із запланованим розкладом. Якщо різниця не перевищувала допустимого значення, рейс відображався як такий, що виконується за графіком.

Приклад відображення нормального руху транспортного засобу наведено на рисунку 4.1.

Для перевірки реакції системи на проблемні ситуації були змодельовані такі сценарії:

– рух зі зниженою швидкістю;

– тривала зупинка;

– відхилення від маршруту;

– тимчасова втрата зв’язку;

– передавання некоректних координат.

У режимі зниженої швидкості імітатор збільшував час проходження між точками маршруту. Після входження транспортного засобу до контрольної зони зупинки система визначала фактичний час прибуття та обчислювала запізнення.

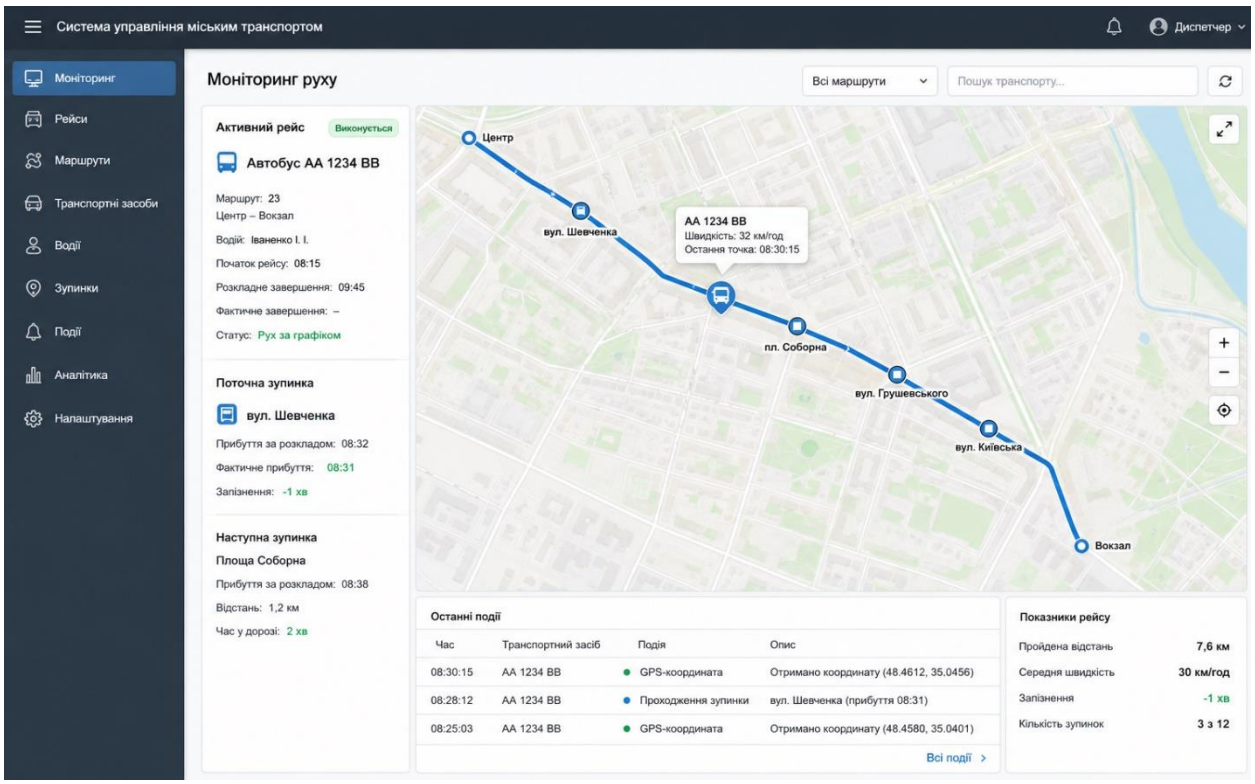


Рисунок 4.1 – Відображення руху транспортного засобу за маршрутом

Для тестування тривалої зупинки однакова або близька координата передавалася протягом установленого періоду. Якщо транспортний засіб не перебував у зоні запланованої зупинки, система формувала подію про відсутність руху.

Відхилення від маршруту моделювалося передаванням кількох послідовних координат за межами допустимої зони. Окрема точка поза маршрутом не спричиняла негайного повідомлення, що дозволяло врахувати можливу похибку GPS. Подія створювалася лише після підтвердження відхилення декількома координатами.

Приклад відображення проблемного сценарію наведено на рисунку 4.2.

Під час перевірок серверна частина коректно розрізняла нормальний рух і проблемні стани. Координати з недопустимими значеннями не зберігалися, а події формувалися лише після виконання заданих умов.

Результати тестування руху наведено в таблиці 4.3.

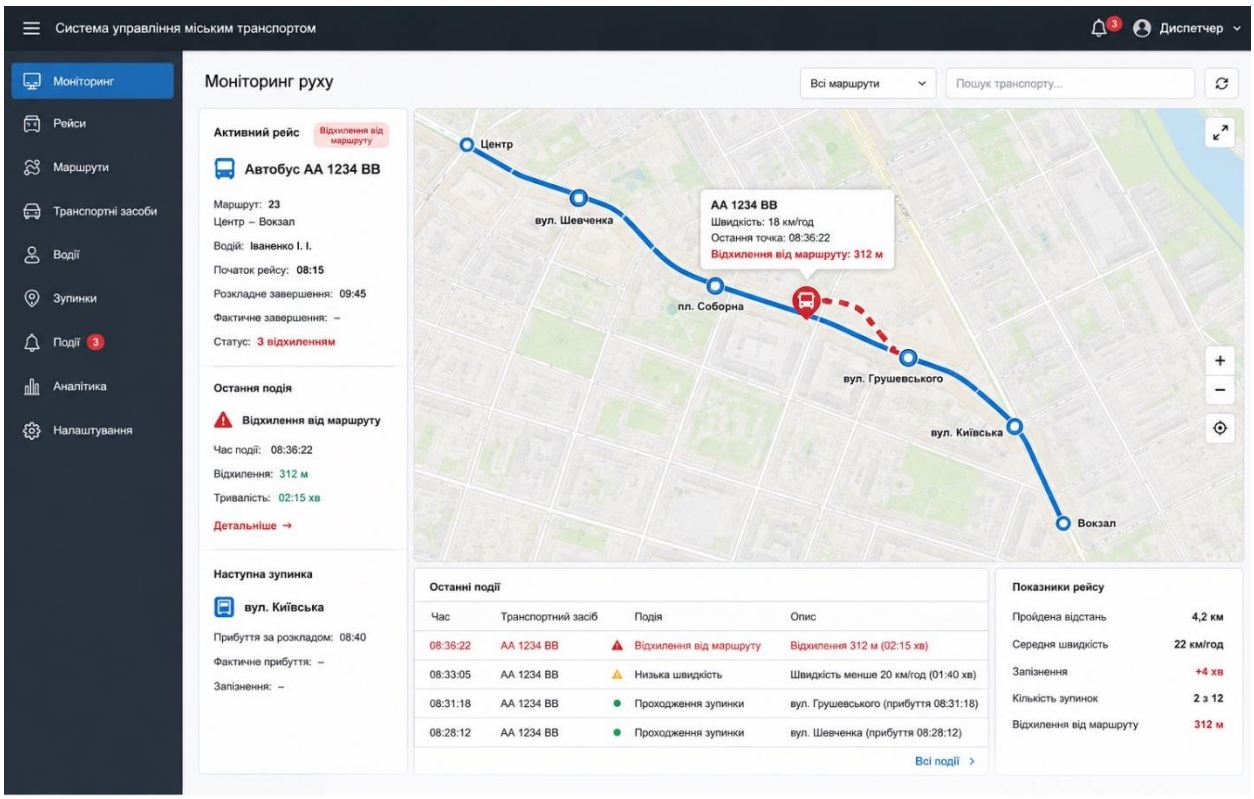


Рисунок 4.2 – Виявлення відхилення транспортного засобу від маршруту

Таблиця 4.3 – Результати тестування руху транспортних засобів

№	Сценарій	Очікуваний результат	Фактичний результат
1	Нормальний рух	Координати зберігаються, маркер рухається за маршрутом	Відповідає
2	Рух зі зниженою швидкістю	Визначається запізнення відносно розкладу	Відповідає
3	Тривала зупинка	Формується подія про відсутність руху	Відповідає
4	Відхилення від маршруту	Після кількох точок поза маршрутом створюється подія	Відповідає
5	Втрата зв’язку	Після перевищення часу очікування формується повідомлення	Відповідає
6	Некоректні координати	Дані відхиляються та не зберігаються	Відповідає
7	Відновлення зв’язку	Нові координати приймаються, положення оновлюється	Відповідає

Отже, тестування підтвердило правильність роботи підсистеми GPS-моніторингу. Модуль імітації дозволив відтворити основні сценарії руху та перевірити весь цикл обробки даних: від формування координат до їх відображення на карті й реєстрації диспетчерських подій.

4.4 Аналіз результатів і обмежень системи

Проведене тестування підтвердило працездатність основних компонентів інформаційно-аналітичної системи. Реалізовано авторизацію користувачів, розмежування доступу, керування маршрутами, зупинками, транспортними засобами, водіями та рейсами. Серверна частина коректно приймає GPS-координати, перевіряє їх формат, зберігає в базі даних і передає оновлене положення транспорту диспетчерському вебклієнту.

Модуль імітації руху дозволив перевірити систему за різних сценаріїв. Під час нормального руху координати послідовно відображалися на карті, а рейс виконувався без формування попереджень. У режимах запізнення, тривалої зупинки, відхилення від маршруту та втрати зв’язку система створювала відповідні події та відображала їх диспетчеру.

Функціональне тестування також підтвердило правильність розмежування доступу. Адміністратор отримував можливість керувати довідковими даними, диспетчер — створювати й контролювати рейси, водій — запускати рейс і передавати координати, а пасажир — переглядати відкриту інформацію про маршрути та рух транспорту.

Інформаційно-аналітична підсистема забезпечила формування основних показників, зокрема кількості виконаних і скасованих рейсів, середньої тривалості поїздки, кількості запізнень, відхилень від маршруту та випадків втрати зв’язку. Отримані дані можуть використовуватися для оцінювання регулярності руху та виявлення проблемних маршрутів.

Для узагальнення результатів розробки виконано зіставлення реалізованих функцій із вимогами, сформованими в підрозділі 2.1. Результати наведено в таблиці 4.4.

Таблиця 4.4 – Відповідність реалізованої системи основним функціональним вимогам

Код вимоги	Зміст вимоги	Результат реалізації	Статус
FR-01	Авторизація та розмежування доступу	Реалізовано токенну авторизацію та доступ відповідно до ролі	Виконано
FR-02	Керування маршрутами й зупинками	Реалізовано створення, редагування, перегляд і деактивацію записів	Виконано
FR-03	Облік водіїв і транспортних засобів	Реалізовано відповідні довідники та зв’язок із рейсами	Виконано
FR-04	Створення та призначення рейсів	Реалізовано формування рейсу з вибором маршруту, водія й транспортного засобу	Виконано
FR-05	Передавання GPS-координат	Реалізовано мобільний застосунок водія та модуль імітації руху	Виконано
FR-06	Відображення транспорту на карті	Реалізовано диспетчерський і пасажирський картографічні інтерфейси	Виконано
FR-07	Визначення відхилень	Реалізовано виявлення запізнення, відхилення від маршруту та втрати зв’язку	Виконано
FR-08	Збереження історії руху	GPS-точки й події зберігаються з прив’язкою до конкретного рейсу	Виконано
FR-09	Формування аналітичних показників	Реалізовано базові показники, таблиці та діаграми	Виконано
FR-10	Пасажирський доступ до транспортної інформації	Реалізовано перегляд маршрутів, зупинок, розкладів і положення транспорту	Виконано
FR-11	Імітація руху для тестування	Реалізовано режими нормального руху, зупинки, відхилення та втрати зв’язку	Виконано

Наведене зіставлення підтвердило реалізацію всіх основних функціональних вимог. До її переваг належать централізоване зберігання

транспортних даних, поєднання планової та фактичної інформації про рейси, відображення транспорту на електронній карті, автоматичне виявлення проблемних ситуацій, розмежування доступу за ролями, підтримка імітації руху та формування аналітичних показників.

Водночас система має окремі обмеження. Точність визначення положення транспортного засобу залежить від якості GPS-сигналу, характеристик мобільного пристрою та умов міської забудови.

Передавання даних також залежить від стабільності інтернет-з’єднання. За відсутності мережі диспетчер тимчасово не отримує актуальних координат. Надалі це обмеження може бути частково усунене локальним збереженням GPS-точок та їх передаванням після відновлення зв’язку.

Прогнозування часу прибуття виконується за спрощеним алгоритмом і не враховує дорожні затори, погодні умови та пасажиропотік. Імітатор руху відтворює основні тестові сценарії, проте не повністю моделює реальні зміни швидкості, щільність транспортного потоку та тривалість зупинок. Аналітична підсистема формує базові показники, але не підтримує автоматичну оптимізацію розкладів і прогнозування пасажиропотоку.

Перед практичним впровадженням необхідно провести навантажувальне тестування, перевірити мобільний застосунок на різних пристроях і виконати апробацію на реальному маршруті. Подальший розвиток може передбачати інтеграцію з GPS-трекерами, зовнішніми транспортними сервісами, електронними табло, системами оплати проїзду та стандартами обміну транспортними даними.

Таким чином, результати тестування підтвердили відповідність системи поставленим вимогам і можливість її використання як програмного прототипу для моніторингу міського громадського транспорту. Виявлені обмеження визначають напрями подальшого вдосконалення та підготовки системи до практичного застосування.

ВИСНОВКИ

У кваліфікаційній роботі розроблено інформаційно-аналітичну систему управління рухом міського громадського транспорту яка забезпечує централізоване ведення транспортних даних, отримання GPS-координат, відображення транспортних засобів на карті, контроль виконання рейсів і формування аналітичної інформації.

Під час аналізу предметної області розглянуто організацію міських пасажирських перевезень, структуру маршрутної мережі, роль розкладів, зупинок, транспортних засобів, водіїв і диспетчерського контролю. Визначено основні проблеми: відхилення від розкладу та маршруту, нерівномірність руху, тривалі зупинки, втрата зв’язку й розрізненість транспортних даних.

Аналіз існуючих пасажирських сервісів показав, що вони забезпечують перегляд маршрутів, зупинок і актуального положення транспорту, але не охоплюють повний цикл внутрішнього управління транспортним підприємством. Це підтвердило доцільність розробки спеціалізованої системи, яка поєднує диспетчерський контроль, GPS-моніторинг, ведення рейсів, рольовий доступ і аналітику.

На етапі проєктування сформовано функціональні й нефункціональні вимоги, визначено ролі адміністратора, диспетчера, водія та пасажира, розроблено сценарії їх взаємодії. Запропоновано архітектуру, до складу якої входять веб-застосунок, мобільний застосунок водія, серверна частина, база даних, аналітичний модуль, картографічний сервіс та імітатор руху.

Розроблено алгоритми авторизації, обробки GPS-координат, визначення відхилень від маршруту й розкладу, формування подій та імітації руху. Спроєктовано реляційну базу даних для зберігання відомостей про, маршрути, зупинки, транспортні засоби, водіїв, розклади, рейси, координати й події. Прив’язка GPS-точок до конкретного рейсу забезпечує відновлення історії руху та формування аналітичних показників.

Для реалізації використано React і TypeScript для веб-застосунку, React Native та Expo для мобільного клієнта, ASP.NET Core і C# для серверної частини, PostgreSQL для зберігання даних, Leaflet та OpenStreetMap для роботи з картою, а SignalR — для оперативного оновлення координат.

Веб-застосунок забезпечує окремі інтерфейси для адміністратора, диспетчера та пасажира. Адміністратор керує довідковими даними, диспетчер контролює активні рейси та події на карті, а пасажир переглядає маршрути, зупинки, розклади й поточне положення транспорту. Мобільний застосунок водія підтримує авторизацію, вибір рейсу, запуск і завершення поїздки та передавання GPS-координат.

Для перевірки системи створено модуль імітації руху, який відтворює нормальний рух, запізнення, тривалу зупинку, відхилення від маршруту та втрату зв’язку. Інформаційно-аналітична підсистема формує кількість виконаних і скасованих рейсів, середню тривалість поїздки, відхилення від розкладу та кількість проблемних подій.

Проведено функціональне, інтеграційне та системне тестування. Перевірено авторизацію, розмежування доступу, роботу з довідковими даними, створення рейсів, приймання GPS-координат, відображення транспорту на карті, формування подій та аналітичних показників. Результати підтвердили реалізацію основних функціональних вимог.

Практичне значення полягає у створенні програмного прототипу, для автоматизації диспетчерського контролю транспортного підприємства. Система централізує інформацію, зменшує кількість ручних операцій і підвищує оперативність отримання даних про рух.

Подальший розвиток передбачає інтеграцію з реальними GPS-трекерами, розширення аналітики та підключення зовнішніх міських сервісів.

Отже, мету кваліфікаційної роботи досягнуто, а поставлені завдання виконано. Розроблена система підтверджує можливість комплексного використання інформаційних технологій для моніторингу та управління рухом міського громадського транспорту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Intelligent Transport Systems Directive and Action Plan [Електронний ресурс] // European Commission. Directorate-General for Mobility and Transport. URL: https://transport.ec.europa.eu/transport-themes/smart-mobility/road/its-directive-and-action-plan_en (дата звернення: 05.06.2026).

2. Intelligent Transport Systems for Sustainable Mobility [Електронний ресурс] / United Nations Economic Commission for Europe. Geneva : United Nations, 2024. URL: https://unece.org/sites/default/files/2024-06/ITS%20for%20sustainable%20Mobility_E_pdf_web.pdf (дата звернення: 05.06.2026).

3. Про автомобільний транспорт : Закон України від 05.04.2001 № 2344-III : станом на 5 черв. 2026 р. // Законодавство України : база даних / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/2344-14 (дата звернення: 05.06.2026).

4. Про затвердження Правил надання послуг пасажирського автомобільного транспорту : Постанова Кабінету Міністрів України від 18.02.1997 № 176 : станом на 5 черв. 2026 р. // Законодавство України : база даних / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/176-97-%D0%BF (дата звернення: 05.06.2026).

5. GTFS Schedule Reference // General Transit Feed Specification : website. URL: https://gtfs.org/documentation/schedule/reference/ (date of access: 05.06.2026).

6. GTFS Realtime Reference // General Transit Feed Specification : website. URL: https://gtfs.org/documentation/realtime/reference/ (date of access: 05.06.2026).

7. The New EU Urban Mobility Framework : Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions, COM(2021) 811 final.

Brussels, 14.12.2021. URL: https://transport.ec.europa.eu/system/files/2021-12/com_2021_811_the-new-eu-urban-mobility.pdf (date of access: 05.06.2026).

8. Creating a common European mobility data space // European Commission : website. URL: https://transport.ec.europa.eu/transport-themes/smart-mobility/creating-common-european-mobility-data-space_en (date of access: 05.06.2026).

9. • EasyWay – маршрути громадського транспорту України : вебсайт. URL: https://www.eway.in.ua/ (дата звернення: 05.06.2026).

10. Moovit App Features: One App, All Your Urban Mobility Options // Moovit : website. URL: https://moovit.com/features/ (date of access: 05.06.2026).

11. About Google Transit // Google Transit Partners Help : website. URL: https://support.google.com/transitpartners/answer/1111471 (date of access: 05.06.2026).

12. У міського застосунку «Київ Цифровий» з’явилася англійська версія // Київська міська рада : офіційний інтернет-портал. 25.10.2022. URL: https://kyivcity.gov.ua/news/u_miskogo_zastosunku_kiv_tsifroviy_zyavilas_angliyska_versiya__An_English_version_of_the_Kyiv_Digital_city_app_is_now_available/ (дата звернення: 05.06.2026).

13. ISO/IEC/IEEE 29148:2018. Systems and software engineering — Life cycle processes — Requirements engineering. Geneva : International Organization for Standardization, 2018.

14. Object Management Group. OMG Unified Modeling Language Version 2.5.1 [Електронний ресурс]. URL: https://www.omg.org/spec/UML/2.5.1 (дата звернення: 05.06.2026).

15. ДСТУ ISO 5807:2016. Обробляння інформації. Символи та угоди щодо документації стосовно даних, програм та системних блок-схем, схем мережевих програм та схем системних ресурсів (ISO 5807:1985, IDT). Чинний від 2016-10-10. Київ : ДП «УкрНДНЦ», 2016.

16. Codd E. F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. 1970. Vol. 13, no. 6. P. 377–387. DOI: 10.1145/362384.362685.

17. Elmasri R., Navathe S. B. Fundamentals of Database Systems. 7th ed. Harlow : Pearson Education, 2016. 1272 p.

18. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York : McGraw-Hill Education, 2020. 1376 p.

19. ASP.NET Core documentation // Microsoft Learn : website. URL: https://learn.microsoft.com/en-us/aspnet/core/ (date of access: 06.06.2026).

20. PostgreSQL Documentation // PostgreSQL : website. URL: https://www.postgresql.org/docs/current/ (date of access: 06.06.2026).

21. React Documentation // React : website. URL: https://react.dev/ (date of access: 06.06.2026).

22. React Native Documentation // React Native : website. URL: https://reactnative.dev/ (date of access: 06.06.2026).

23. Expo Location // Expo Documentation : website. URL: https://docs.expo.dev/versions/latest/sdk/location/ (date of access: 06.06.2026).

24. ISO/IEC/IEEE 29119-1:2022. Software and systems engineering — Software testing — Part 1: General concepts. 2nd ed. Geneva : International Organization for Standardization, 2022.

25. ДСТУ ISO/IEC/IEEE 29119-1:2017. Інженерія систем і програмних засобів. Тестування програмних засобів. Частина 1. Поняття та визначення (ISO/IEC/IEEE 29119-1:2013, IDT). Чинний від 2019-01-01. Київ : ДП «УкрНДНЦ», 2017.

Додаток А

Лістинг основних програмних модулів

Лістинг А.1 – Модель даних рейсу

```
namespace TransportMonitoring.Domain.Entities;

public enum TripStatus
{
    Planned,
    Active,
    Completed,
    Cancelled
}

public sealed class Trip
{
    public Guid Id { get; set; }

    public Guid RouteId { get; set; }
    public Route Route { get; set; } = null!;

    public Guid? ScheduleId { get; set; }
    public Schedule? Schedule { get; set; }

    public Guid DriverId { get; set; }
    public Driver Driver { get; set; } = null!;

    public Guid VehicleId { get; set; }
    public Vehicle Vehicle { get; set; } = null!;

    public DateTime PlannedStartTime { get; set; }
    public DateTime PlannedEndTime { get; set; }

    public DateTime? ActualStartTime { get; set; }
    public DateTime? ActualEndTime { get; set; }

    public TripStatus Status { get; set; } = TripStatus.Planned;

    public ICollection<GpsPoint> GpsPoints { get; set; } =
        new List<GpsPoint>();

    public ICollection<TransportEvent> Events { get; set; } =
        new List<TransportEvent>();
}
```

Клас Trip описує конкретне виконання маршруту та встановлює зв’язок між маршрутом, водієм, транспортним засобом, розкладом, GPS-точками й подіями.

Лістинг А.2 – Модель GPS-точки

```
namespace TransportMonitoring.Domain.Entities;

public enum GpsSource
{
    MobileApplication,
    Simulator
}

public sealed class GpsPoint
{
    public Guid Id { get; set; }

    public Guid TripId { get; set; }
    public Trip Trip { get; set; } = null!;

    public decimal Latitude { get; set; }
    public decimal Longitude { get; set; }

    public decimal? Speed { get; set; }

    public DateTime MeasuredAt { get; set; }
    public DateTime ReceivedAt { get; set; }

    public GpsSource Source { get; set; }
}
```

Поле MeasuredAt містить час визначення координати пристроєм, а ReceivedAt — час її отримання серверною частиною. Це дозволяє виявляти затримане передавання даних.

Лістинг А.3 – Контекст бази даних

```
using Microsoft.EntityFrameworkCore;
using TransportMonitoring.Domain.Entities;

namespace TransportMonitoring.Infrastructure;

public sealed class TransportDbContext : DbContext
{
    public TransportDbContext(
        DbContextOptions<TransportDbContext> options)
        : base(options)
    {
    }

    public DbSet<Role> Roles => Set<Role>();
    public DbSet<User> Users => Set<User>();
    public DbSet<Driver> Drivers => Set<Driver>();
    public DbSet<Vehicle> Vehicles => Set<Vehicle>();
    public DbSet<Route> Routes => Set<Route>();
```

```
    public DbSet<Stop> Stops => Set<Stop>();
    public DbSet<RouteStop> RouteStops => Set<RouteStop>();
    public DbSet<Schedule> Schedules => Set<Schedule>();
    public DbSet<Trip> Trips => Set<Trip>();
    public DbSet<GpsPoint> GpsPoints => Set<GpsPoint>();
    public DbSet<TransportEvent> Events => Set<TransportEvent>();

    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<User>()
            .HasIndex(x => x.Login)
            .IsUnique();

        modelBuilder.Entity<Vehicle>()
            .HasIndex(x => x.FleetNumber)
            .IsUnique();

        modelBuilder.Entity<Driver>()
            .HasIndex(x => x.LicenseNumber)
            .IsUnique();

        modelBuilder.Entity<RouteStop>()
            .HasKey(x => new
            {
                x.RouteId,
                x.StopId,
                x.Direction,
                x.SequenceNumber
            });

        modelBuilder.Entity<GpsPoint>()
            .HasIndex(x => new
            {
                x.TripId,
                x.MeasuredAt
            });

        modelBuilder.Entity<GpsPoint>()
            .Property(x => x.Latitude)
            .HasPrecision(9, 6);

        modelBuilder.Entity<GpsPoint>()
            .Property(x => x.Longitude)
            .HasPrecision(9, 6);

        modelBuilder.Entity<Trip>()
            .Property(x => x.Status)
            .HasConversion<string>();

        modelBuilder.Entity<GpsPoint>()
```

```
            .Property(x => x.Source)
            .HasConversion<string>();
    }
}
```

Контекст визначає таблиці бази даних, унікальні обмеження, складений ключ RouteStops та індекс для хронологічного отримання координат рейсу.

Лістинг А.4 – Модель запиту на передавання GPS-координати

```
using System.ComponentModel.DataAnnotations;

namespace TransportMonitoring.Application.Gps;

public sealed class CreateGpsPointRequest
{
    [Required]
    public Guid TripId { get; init; }

    [Range(-90.0, 90.0)]
    public decimal Latitude { get; init; }

    [Range(-180.0, 180.0)]
    public decimal Longitude { get; init; }

    [Range(0.0, 250.0)]
    public decimal? Speed { get; init; }

    [Required]
    public DateTime MeasuredAt { get; init; }

    [Required]
    public string Source { get; init; } = string.Empty;
}
```

Атрибути валідації обмежують допустимі значення широти, довготи та швидкості ще до виконання основної серверної логіки.

Лістинг А.5 – Контролер приймання GPS-координат

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using TransportMonitoring.Application.Gps;

namespace TransportMonitoring.Api.Controllers;

[ApiController]
[Route("api/gps-points")]
[Authorize(Roles = "Driver,Administrator")]
public sealed class GpsPointsController : ControllerBase
{
    private readonly IGpsService _gpsService;

    public GpsPointsController(IGpsService gpsService)
```

```
    {
        _gpsService = gpsService;
    }

    [HttpPost]
    public async Task<IActionResult> Create(
        CreateGpsPointRequest request,
        CancellationToken cancellationToken)
    {
        if (!ModelState.IsValid)
        {
            return ValidationProblem(ModelState);
        }

        var result = await _gpsService.AddPointAsync(
            request,
            User,
            cancellationToken);

        if (!result.IsSuccess)
        {
            return BadRequest(new
            {
                message = result.ErrorMessage
            });
        }

        return Ok(new
        {
            message = "GPS-координату успішно збережено",
            pointId = result.PointId
        });
    }
}
```

Контролер приймає HTTP-запит, перевіряє модель і передає виконання операції сервісу GPS-моніторингу.

Лістинг А.6 – Сервіс обробки GPS-координат

```
using System.Security.Claims;
using Microsoft.AspNetCore.SignalR;
using Microsoft.EntityFrameworkCore;
using TransportMonitoring.Api.Hubs;
using TransportMonitoring.Domain.Entities;
using TransportMonitoring.Infrastructure;

namespace TransportMonitoring.Application.Gps;

public sealed class GpsService : IGpsService
{
    private readonly TransportDbContext _dbContext;
    private readonly IDeviationService _deviationService;
```

```
    private readonly IHubContext<TransportHub> _hubContext;

    public GpsService(
        TransportDbContext dbContext,
        IDeviationService deviationService,
        IHubContext<TransportHub> hubContext)
    {
        _dbContext = dbContext;
        _deviationService = deviationService;
        _hubContext = hubContext;
    }

    public async Task<GpsOperationResult> AddPointAsync(
        CreateGpsPointRequest request,
        ClaimsPrincipal user,
        CancellationToken cancellationToken)
    {
        var trip = await _dbContext.Trips
            .Include(x => x.Route)
            .ThenInclude(x => x.RouteStops)
            .ThenInclude(x => x.Stop)
            .FirstOrDefaultAsync(
                x => x.Id == request.TripId,
                cancellationToken);

        if (trip is null)
        {
            return GpsOperationResult.Failure(
                "Рейс не знайдено");
        }

        if (trip.Status != TripStatus.Active)
        {
            return GpsOperationResult.Failure(
                "GPS-дані можна передавати лише для активного
рейсу");
        }

        var duplicateExists = await _dbContext.GpsPoints
            .AnyAsync(
                x => x.TripId == request.TripId &&
                     x.MeasuredAt == request.MeasuredAt,
                cancellationToken);

        if (duplicateExists)
        {
            return GpsOperationResult.Failure(
                "GPS-точку з такою часовою позначкою вже
збережено");
        }

        if (request.MeasuredAt >
            DateTime.UtcNow.AddMinutes(2))
```

```
        {
            return GpsOperationResult.Failure(
                "Час вимірювання координати є некоректним");
        }

        var point = new GpsPoint
        {
            Id = Guid.NewGuid(),
            TripId = request.TripId,
            Latitude = request.Latitude,
            Longitude = request.Longitude,
            Speed = request.Speed,
            MeasuredAt = request.MeasuredAt,
            ReceivedAt = DateTime.UtcNow,
            Source = Enum.Parse<GpsSource>(
                request.Source,
                ignoreCase: true)
        };

        _dbContext.GpsPoints.Add(point);
        await _dbContext.SaveChangesAsync(cancellationToken);

        await _deviationService.ProcessPointAsync(
            trip,
            point,
            cancellationToken);

        await _hubContext.Clients
            .Group($"trip-{trip.Id}")
            .SendAsync(
                "VehiclePositionUpdated",
                new
                {
                    tripId = trip.Id,
                    latitude = point.Latitude,
                    longitude = point.Longitude,
                    speed = point.Speed,
                    measuredAt = point.MeasuredAt
                },
                cancellationToken);

        return GpsOperationResult.Success(point.Id);
    }
}
```

Сервіс перевіряє існування й активність рейсу, запобігає дублюванню GPS-точок, зберігає координату та передає її до модуля визначення відхилень і диспетчерського інтерфейсу.

Лістинг А.7 – Розрахунок відстані між координатами

```
namespace TransportMonitoring.Application.Common;

public static class GeoDistance
{
    private const double EarthRadiusMeters = 6_371_000;

    public static double CalculateMeters(
        double latitude1,
        double longitude1,
        double latitude2,
        double longitude2)
    {
        var lat1 = DegreesToRadians(latitude1);
        var lat2 = DegreesToRadians(latitude2);
        var deltaLat = DegreesToRadians(latitude2 - latitude1);
        var deltaLon = DegreesToRadians(longitude2 - longitude1);

        var a =
            Math.Sin(deltaLat / 2) * Math.Sin(deltaLat / 2) +
            Math.Cos(lat1) * Math.Cos(lat2) *
            Math.Sin(deltaLon / 2) * Math.Sin(deltaLon / 2);

        var c = 2 * Math.Atan2(
            Math.Sqrt(a),
            Math.Sqrt(1 - a));

        return EarthRadiusMeters * c;
    }

    private static double DegreesToRadians(double degrees)
    {
        return degrees * Math.PI / 180.0;
    }
}
```

Функція використовує формулу гаверсинусів для визначення наближеної відстані між двома географічними точками.

Лістинг А.8 – Визначення відхилення від маршруту

```
using Microsoft.EntityFrameworkCore;
using TransportMonitoring.Application.Common;
using TransportMonitoring.Domain.Entities;
using TransportMonitoring.Infrastructure;

namespace TransportMonitoring.Application.Events;

public sealed class DeviationService : IDeviationService
{
    private const double AllowedDistanceMeters = 120;
    private const int RequiredOutsidePoints = 3;
```

```
private readonly TransportDbContext _dbContext;

public DeviationService(TransportDbContext dbContext)
{
    _dbContext = dbContext;
}

public async Task ProcessPointAsync(
    Trip trip,
    GpsPoint currentPoint,
    CancellationToken cancellationToken)
{
    var routePoints = trip.Route.RouteStops
        .OrderBy(x => x.SequenceNumber)
        .Select(x => new
        {
            Latitude = (double)x.Stop.Latitude,
            Longitude = (double)x.Stop.Longitude
        })
        .ToList();

    if (routePoints.Count == 0)
    {
        return;
    }

    var minimumDistance = routePoints
        .Select(x => GeoDistance.CalculateMeters(
            (double)currentPoint.Latitude,
            (double)currentPoint.Longitude,
            x.Latitude,
            x.Longitude))
        .Min();

    if (minimumDistance <= AllowedDistanceMeters)
    {
        return;
    }

    var previousPoints = await _dbContext.GpsPoints
        .Where(x => x.TripId == trip.Id)
        .OrderByDescending(x => x.MeasuredAt)
        .Take(RequiredOutsidePoints)
        .ToListAsync(cancellationToken);

    if (previousPoints.Count < RequiredOutsidePoints)
    {
        return;
    }

    var allPointsOutside = previousPoints.All(point =>
        routePoints.Min(routePoint =>
            GeoDistance.CalculateMeters(
```

```
                    (double)point.Latitude,
                    (double)point.Longitude,
                    routePoint.Latitude,
                    routePoint.Longitude))
                > AllowedDistanceMeters);

        if (!allPointsOutside)
        {
            return;
        }

        var activeEventExists = await _dbContext.Events
            .AnyAsync(
                x => x.TripId == trip.Id &&
                     x.EventType == "RouteDeviation" &&
                     x.Status == "Active",
                cancellationToken);

        if (activeEventExists)
        {
            return;
        }

        var transportEvent = new TransportEvent
        {
            Id = Guid.NewGuid(),
            TripId = trip.Id,
            EventType = "RouteDeviation",
            Description =
                $"Транспортний засіб відхилився від маршруту " +
                $"на {minimumDistance:F0} м",
            Severity = "Warning",
            Status = "Active",
            DetectedAt = DateTime.UtcNow
        };

        _dbContext.Events.Add(transportEvent);
        await _dbContext.SaveChangesAsync(cancellationToken);
    }
}
```

Подія формується лише після отримання кількох послідовних точок за межами допустимої зони, що зменшує кількість помилкових попереджень.

Лістинг А.9 – SignalR Hub для оперативного оновлення даних

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.SignalR;

namespace TransportMonitoring.Api.Hubs;

[Authorize]
public sealed class TransportHub : Hub
```

```
{
    public async Task SubscribeToTrip(Guid tripId)
    {
        await Groups.AddToGroupAsync(
            Context.ConnectionId,
            $"trip-{tripId}");
    }

    public async Task UnsubscribeFromTrip(Guid tripId)
    {
        await Groups.RemoveFromGroupAsync(
            Context.ConnectionId,
            $"trip-{tripId}");
    }

    public async Task SubscribeToDispatcherUpdates()
    {
        await Groups.AddToGroupAsync(
            Context.ConnectionId,
            "dispatchers");
    }
}
```

Підключені диспетчерські клієнти можуть підписуватися на оновлення конкретного рейсу та отримувати координати без перезавантаження сторінки.

Лістинг А.10 – Отримання GPS-координат у мобільному застосунку

```
import * as Location from "expo-location";

export interface GpsPointRequest {
  tripId: string;
  latitude: number;
  longitude: number;
  speed: number | null;
  measuredAt: string;
  source: "MobileApplication";
}

export async function startLocationTracking(
  tripId: string,
  accessToken: string
): Promise<Location.LocationSubscription> {
  const permission =
    await Location.requestForegroundPermissionsAsync();

  if (permission.status !== "granted") {
    throw new Error(
      "Не надано дозвіл на використання геолокації"
    );
  }
```

```
  return Location.watchPositionAsync(
    {
      accuracy: Location.Accuracy.High,
      timeInterval: 10_000,
      distanceInterval: 10
    },
    async (location) => {
      const request: GpsPointRequest = {
        tripId,
        latitude: location.coords.latitude,
        longitude: location.coords.longitude,
        speed: location.coords.speed === null
          ? null
          : location.coords.speed * 3.6,
        measuredAt: new Date(
          location.timestamp
        ).toISOString(),
        source: "MobileApplication"
      };

      const response = await fetch(
        "https://server.example/api/gps-points",
        {
          method: "POST",
          headers: {
            "Content-Type": "application/json",
            Authorization: `Bearer ${accessToken}`
          },
          body: JSON.stringify(request)
        }
      );

      if (!response.ok) {
        console.error(
          "Не вдалося передати GPS-координату"
        );
      }
    }
  );
}
```

Мобільний застосунок отримує координати через Expo Location і передає їх серверному API через заданий часовий або просторовий інтервал.

Лістинг А.11 – Зупинення передавання координат

```
import * as Location from "expo-location";

let activeSubscription:
  Location.LocationSubscription | null = null;

export function saveLocationSubscription(
  subscription: Location.LocationSubscription
```

```
): void {
  activeSubscription = subscription;
}

export function stopLocationTracking(): void {
  if (activeSubscription !== null) {
    activeSubscription.remove();
    activeSubscription = null;
  }
}
```

Після завершення рейсу підписка на оновлення місцеположення видаляється, що припиняє використання GPS і передавання даних.

Лістинг А.12 – Основний цикл імітатора руху

```
using System.Net.Http.Headers;
using System.Net.Http.Json;

namespace TransportMonitoring.Simulator;

public sealed class MovementSimulator
{
    private readonly HttpClient _httpClient;

    public MovementSimulator(
        HttpClient httpClient,
        string accessToken)
    {
        _httpClient = httpClient;

        _httpClient.DefaultRequestHeaders.Authorization =
            new AuthenticationHeaderValue(
                "Bearer",
                accessToken);
    }

    public async Task RunAsync(
        Guid tripId,
        IReadOnlyList<RouteCoordinate> route,
        SimulationOptions options,
        CancellationToken cancellationToken)
    {
        for (var index = 0; index < route.Count; index++)
        {
            cancellationToken.ThrowIfCancellationRequested();

            var coordinate = route[index];

            if (options.Mode == SimulationMode.ConnectionLoss &&
                index >= options.ErrorStartPoint &&
                index < options.ErrorEndPoint)
            {
```

```
                await Task.Delay(
                    options.SendInterval,
                    cancellationToken);

                continue;
            }

            if (options.Mode == SimulationMode.RouteDeviation &&
                index >= options.ErrorStartPoint &&
                index < options.ErrorEndPoint)
            {
                coordinate = coordinate with
                {
                    Latitude = coordinate.Latitude + 0.003,
                    Longitude = coordinate.Longitude + 0.003
                };
            }

            var request = new
            {
                tripId,
                latitude = coordinate.Latitude,
                longitude = coordinate.Longitude,
                speed = options.SpeedKmPerHour,
                measuredAt = DateTime.UtcNow,
                source = "Simulator"
            };

            var response = await _httpClient.PostAsJsonAsync(
                "api/gps-points",
                request,
                cancellationToken);

            response.EnsureSuccessStatusCode();

            var delay = options.Mode ==
                        SimulationMode.LongStop &&
                        index == options.ErrorStartPoint
                ? options.LongStopDuration
                : options.SendInterval;

            await Task.Delay(delay, cancellationToken);
        }
    }
}
```

Імітатор послідовно передає координати маршруту та змінює свою поведінку залежно від вибраного режиму тестування.

Лістинг А.13 – Параметри імітації руху

```
namespace TransportMonitoring.Simulator;
```

```
public enum SimulationMode
{
    Normal,
    SlowMovement,
    LongStop,
    RouteDeviation,
    ConnectionLoss
}

public sealed record RouteCoordinate(
    double Latitude,
    double Longitude);

public sealed class SimulationOptions
{
    public SimulationMode Mode { get; init; } =
        SimulationMode.Normal;

    public double SpeedKmPerHour { get; init; } = 40;

    public TimeSpan SendInterval { get; init; } =
        TimeSpan.FromSeconds(10);

    public TimeSpan LongStopDuration { get; init; } =
        TimeSpan.FromMinutes(3);

    public int ErrorStartPoint { get; init; } = 10;
    public int ErrorEndPoint { get; init; } = 15;
}
```

Параметри визначають швидкість, інтервал передавання та ділянку маршруту, на якій відтворюється проблемна ситуація.

Лістинг А.14 – Формування базових аналітичних показників

```
using Microsoft.EntityFrameworkCore;
using TransportMonitoring.Domain.Entities;
using TransportMonitoring.Infrastructure;

namespace TransportMonitoring.Application.Analytics;

public sealed class AnalyticsService
{
    private readonly TransportDbContext _dbContext;

    public AnalyticsService(
        TransportDbContext dbContext)
    {
        _dbContext = dbContext;
    }

    public async Task<AnalyticsResult> GetSummaryAsync(
        DateTime dateFrom,
```

```
        DateTime dateTo,
        Guid? routeId,
        CancellationToken cancellationToken)
    {
        var tripsQuery = _dbContext.Trips
            .AsNoTracking()
            .Where(x =>
                x.PlannedStartTime >= dateFrom &&
                x.PlannedStartTime <= dateTo);

        if (routeId.HasValue)
        {
            tripsQuery = tripsQuery.Where(
                x => x.RouteId == routeId.Value);
        }

        var trips = await tripsQuery
            .ToListAsync(cancellationToken);

        var tripIds = trips
            .Select(x => x.Id)
            .ToList();

        var events = await _dbContext.Events
            .AsNoTracking()
            .Where(x => tripIds.Contains(x.TripId))
            .ToListAsync(cancellationToken);

        var completedTrips = trips
            .Where(x =>
                x.Status == TripStatus.Completed &&
                x.ActualStartTime.HasValue &&
                x.ActualEndTime.HasValue)
            .ToList();

        var averageDuration = completedTrips.Count == 0
            ? 0
            : completedTrips.Average(x =>
                (x.ActualEndTime!.Value -
                 x.ActualStartTime!.Value).TotalMinutes);

        return new AnalyticsResult
        {
            PlannedTrips = trips.Count,
            CompletedTrips = completedTrips.Count,
            CancelledTrips = trips.Count(x =>
                x.Status == TripStatus.Cancelled),
            AverageDurationMinutes =
                Math.Round(averageDuration, 1),
            RouteDeviationCount = events.Count(x =>
                x.EventType == "RouteDeviation"),
            ConnectionLossCount = events.Count(x =>
                x.EventType == "ConnectionLoss")
```

```
            };
        }
    }
```

Додаток Б

Скріншоти інтерфейсів програмних рішень

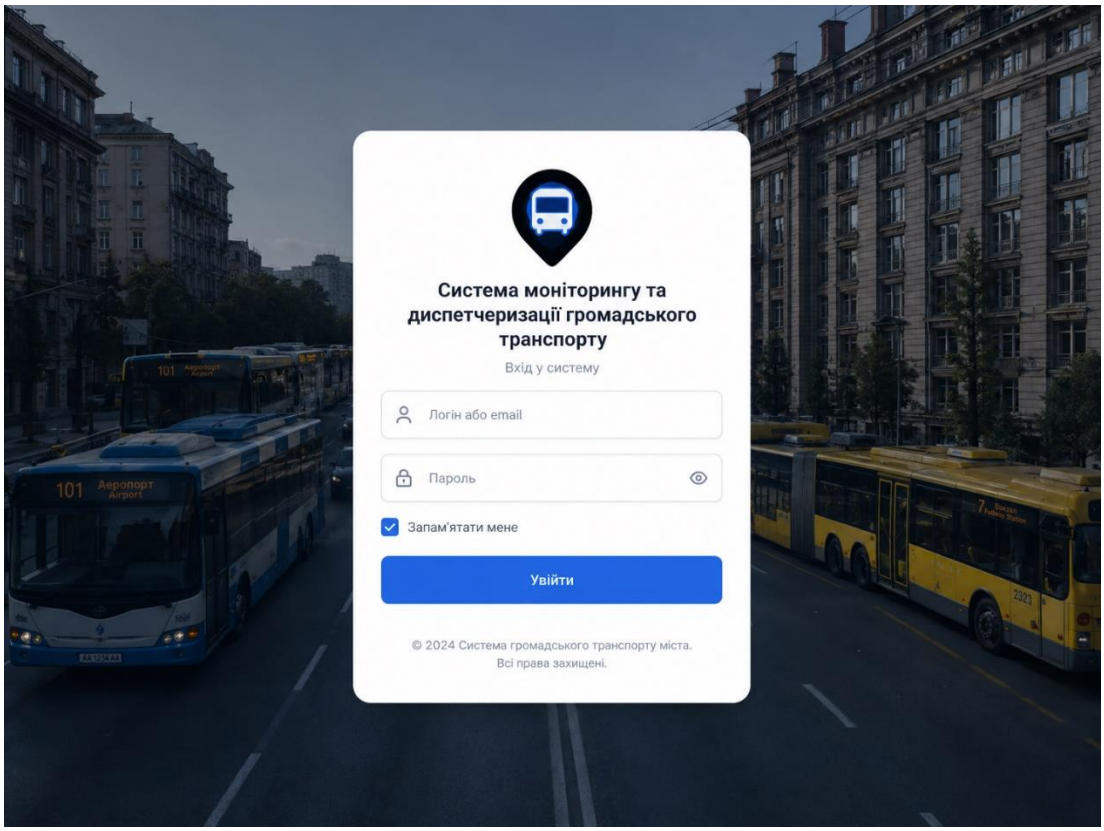


Рисунок Б.1 – Сторінка авторизації користувача в інформаційно-аналітичній системі моніторингу громадського транспорту

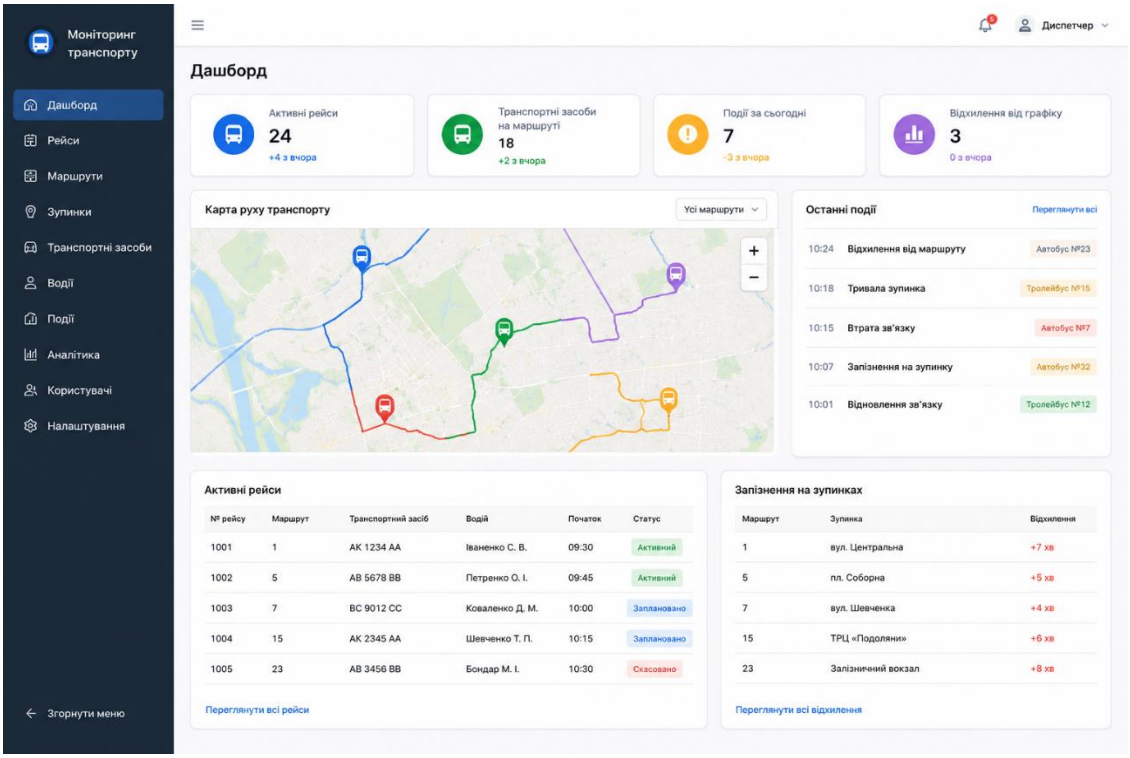


Рисунок Б.2 – Диспетчерська панель моніторингу руху громадського транспорту

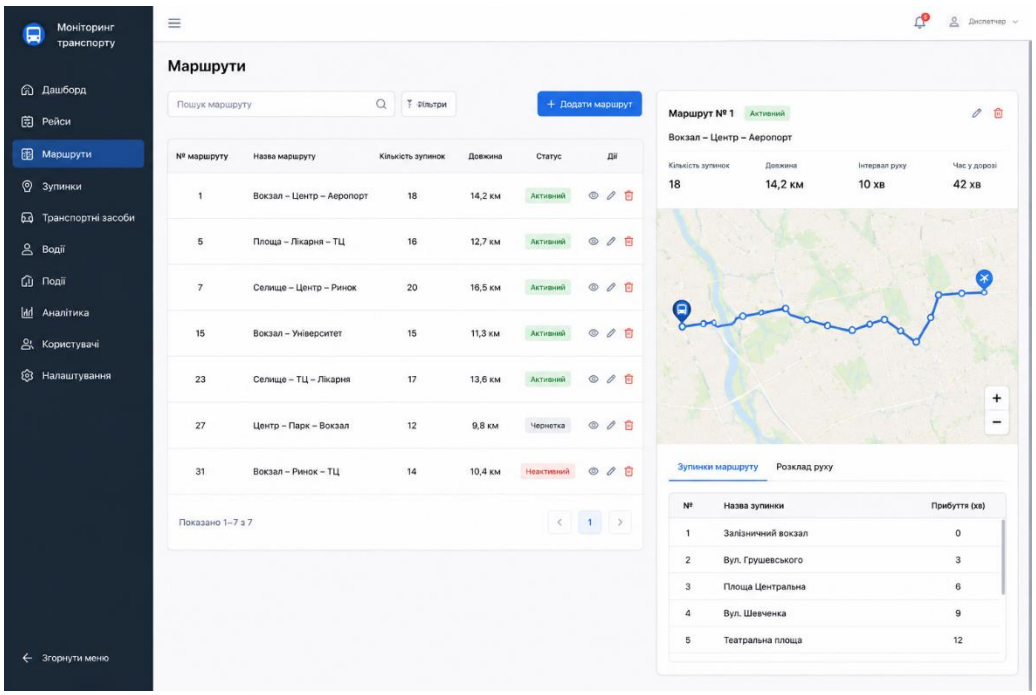


Рисунок Б.3 – Сторінка керування маршрутами у веб-застосунку інформаційно-аналітичної системи

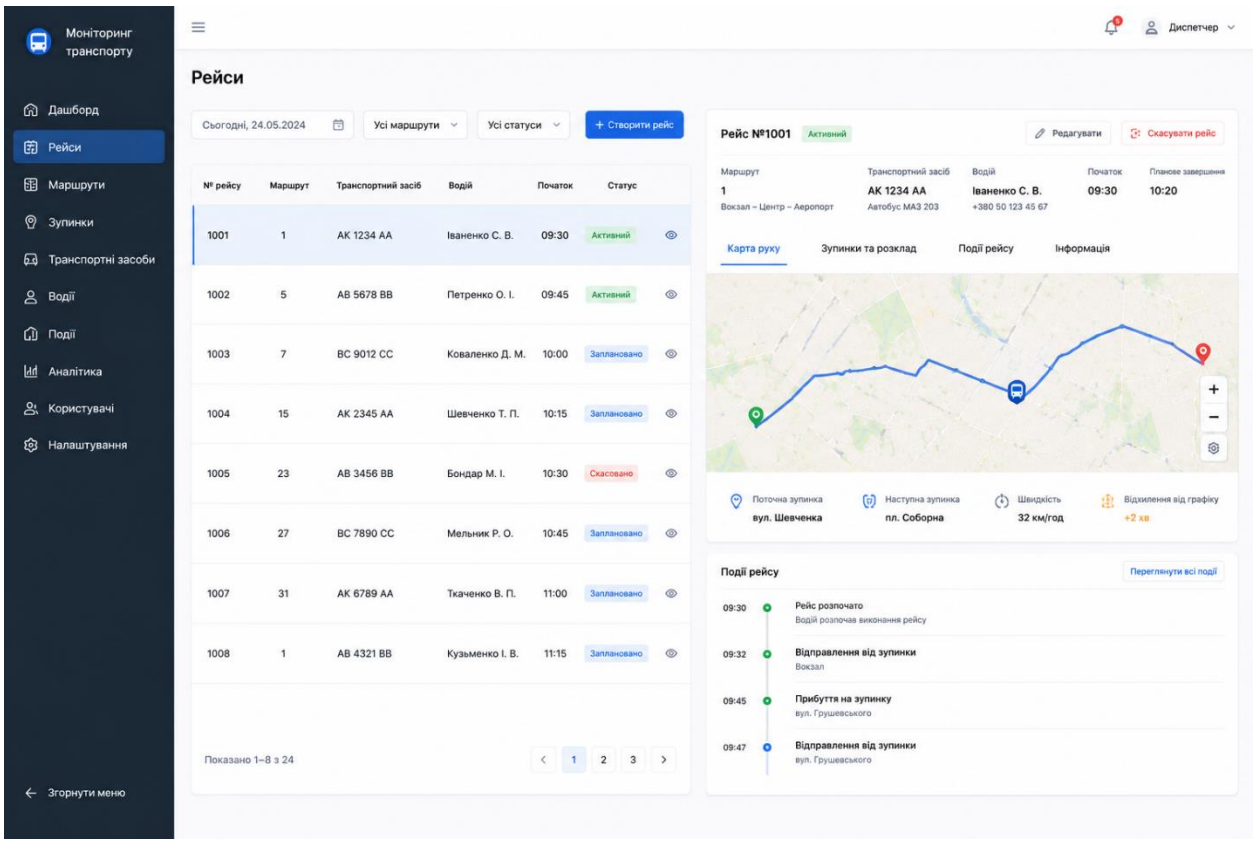


Рисунок Б.4 – Сторінка керування рейсами та перегляду поточного руху транспортного засобу

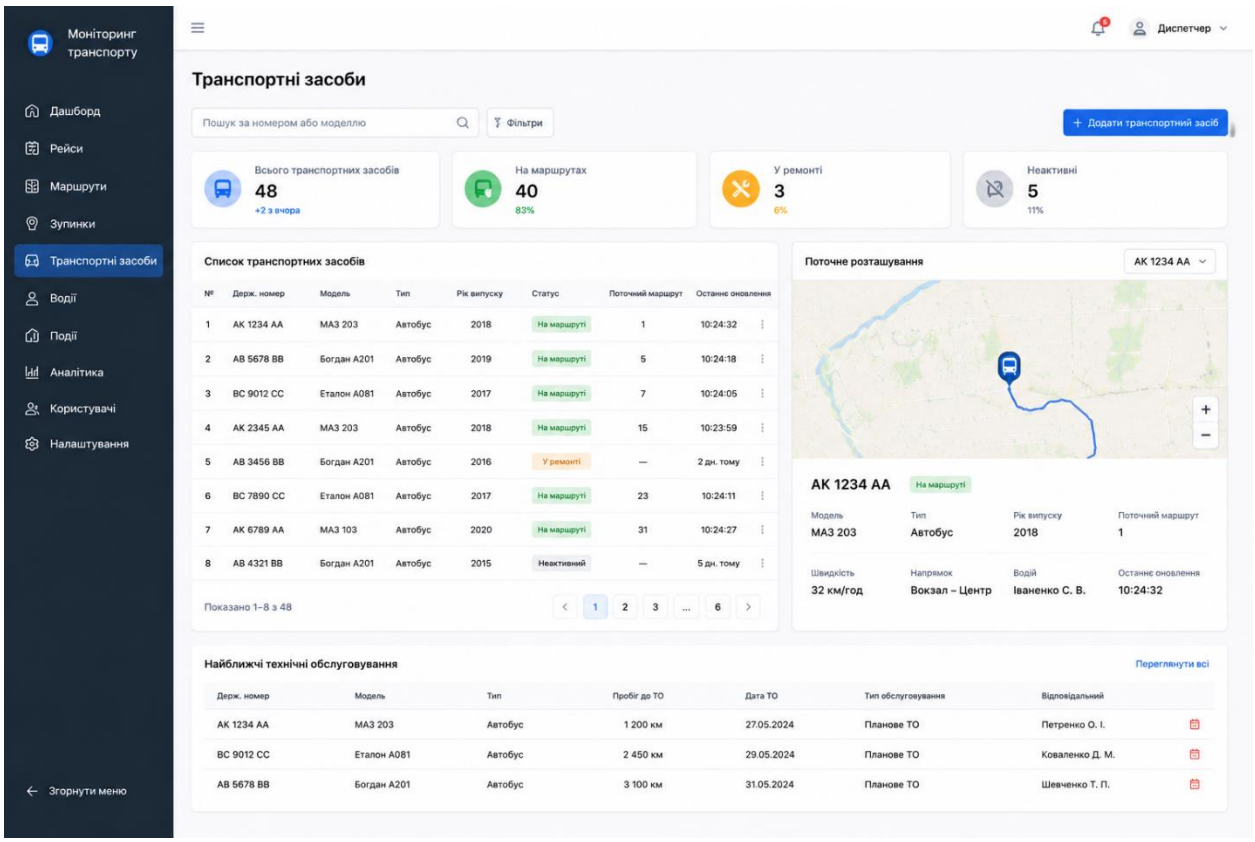


Рисунок Б.5 – Сторінка керування транспортними засобами у веб-застосунку інформаційно-аналітичної системи

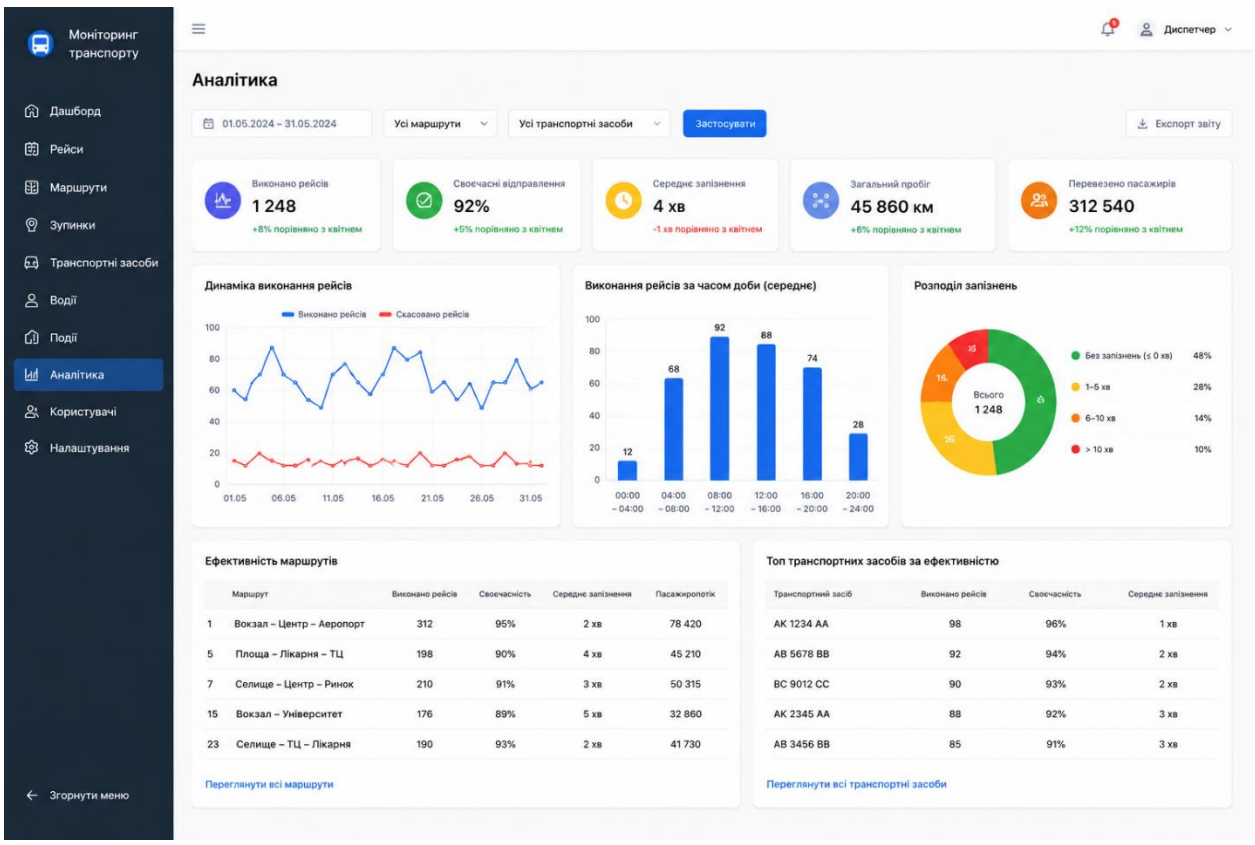


Рисунок Б.6 – Сторінка аналітичної підсистеми з показниками роботи громадського транспорту

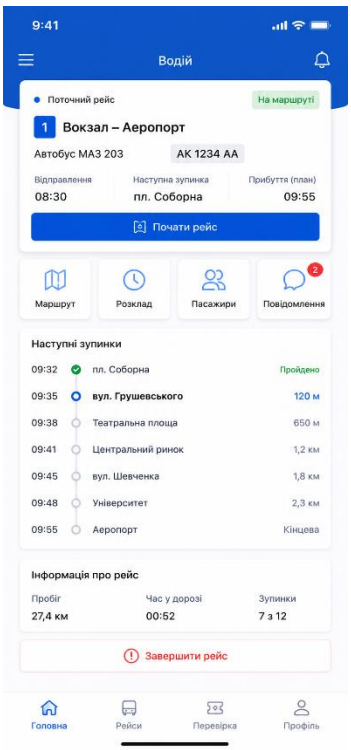


Рисунок Б.7 – Головний екран мобільного застосунку водія з інформацією про поточний рейс

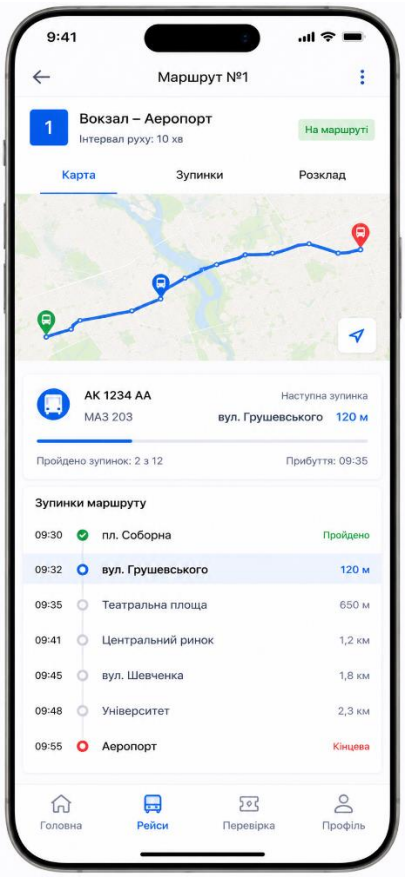


Рисунок Б.9 – Екран мобільного застосунку водія з картою маршруту та переліком зупинок поточного рейсу