Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система організації вантажних перевезень та врегулювання відносин між замовником та перевізником з використанням технології блокчейн, смарт-контрактів та електронного цифрового підпису
(тема)

Виконав:
здобувач 4 року навчання,
групи ПЗПІ-21-5

Євгеній КРАВЧЕНКО
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Гліб ТЕРЕЩЕНКО
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Кравченку Євгенію Романовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система організації вантажних перевезень та врегулювання відносин між замовником та перевізником з використанням технології блокчейн, смарт-контрактів та електронного цифрового підпису затверджена наказом по університету від 19 травня 2025р. № 397Ст
2. Термін подання студентом роботи до екзаменаційної комісії 10 червня 2025 р.
3. Вихідні дані до роботи У програмній системі передбачити: авторизацію та реєстрацію користувачів; управління профілем користувача; інструменти керування замовленнями на перевезення вантажів; пошук замовлень та управління пропозиціями на виконання замовлень; автоматизацію укладання договорів, контроль виконання угод, автоматизацію оплати у разі успішного виконання угоди; систему відгуків та рейтингу; ведення електронного документообігу, тобто можливість додавати супровідні документи та підписувати документи електронним цифровим підписом. Використовувати: СУБД Microsoft SQL Server 2022; мови програмування C#, Kotlin, Solidity; платформи .NET, Ethereum; середовища розробки Microsoft Visual Studio 2022,

Android Studio.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, постановка задачі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	14.04.2025 – 20.04.2025	*виконано*
2	Створення специфікації ПЗ	21.04.2025 – 27.04.2025	*виконано*
3	Проектування ПЗ	28.04.2025 – 11.05.2025	*виконано*
4	Розробка ПЗ	12.05.2025 – 18.05.2025	*виконано*
5	Тестування ПЗ	19.05.2025 – 21.05.2025	*виконано*
6	Оформлення пояснювальної записки	22.05.2025 – 29.05.2025	*виконано*
7	Нормоконтроль, рецензування	30.05.2025 – 02.06.2025	*виконано*
8	Підготовка презентації та доповіді	03.06.2025 – 05.06.2025	*виконано*
9	Попередній захист	05.06.2025 – 09.06.2025	*виконано*
10	Здача роботи у електронний архів	08.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	10.06.2025	*виконано*

Дата видачі завдання «04» «квітня» 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Гліб ТЕРЕЩЕНКО
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 75 стор., 33 рис., 1 табл., 5 додатків, 17 джерел.

БЛОКЧЕЙН, ВАНТАЖНІ ПЕРЕВЕЗЕННЯ, ЕЛЕКТРОННИЙ ЦИФРОВИЙ ПІДПИС, СМАРТ-КОНТРАКТИ, C#, ETHEREUM, KOTLIN

Об’єкт розробки – програмна система організації вантажних перевезень та врегулювання взаємовідносин між замовником і перевізником.

Мета розробки – підвищити прозорість, безпеку та ефективність взаємодії учасників логістичного процесу, створити систему для організації перевезень вантажів, яка спрощує пошук замовників для перевізників і навпаки, забезпечує прозорість та надійність угод і автоматизує ключові аспекти співпраці між сторонами у сфері вантажних перевезень шляхом впровадження технології блокчейн, смарт-контрактів та електронного цифрового підпису.

Метод рішення – середовища розробки Microsoft Visual Studio 2022, Android Studio, платформи .NET Core та Ethereum, мови програмування C#, Kotlin та Solidity, фреймворк ASP.NET Core та СУБД MS SQL.

У результаті розробки створено клієнт-серверний застосунок, що забезпечує зручну взаємодію між учасниками вантажних перевезень, підтримує автоматизовану фіксацію угод із використанням блокчейну та смарт-контрактів і підписання договорів із використанням електронного цифрового підпису.

ABSTRACT

BLOCKCHAIN, FREIGHT TRANSPORTATION, ELECTRONIC DIGITAL SIGNATURE, SMART CONTRACTS, C#, ETHEREUM, KOTLIN

The object of development is a software system for organizing freight transportation and regulating the relationship between the customer and the carrier.

The purpose of the development is to increase transparency, security and efficiency of interaction between participants in the logistics process, to create a system for organizing cargo transportation that simplifies the search of carriers for customers, and vice versa, ensures transparency and reliability of transactions, and automates key aspects of cooperation between the parties in the field of cargo transportation through the introduction of blockchain technology, smart contracts and electronic digital signatures.

Solution method – Microsoft Visual Studio 2022, Android Studio, .NET Core and Ethereum platforms, C#, Kotlin and Solidity programming languages, ASP.NET Core framework and MS SQL database.

As a result of the development, a client-server application was created that provides convenient interaction between participants in freight transportation, supports automated recording of transactions using blockchain and smart contracts, and signing of contracts using an electronic digital signature.

ЗМІСТ

ВСТУП

Сфера вантажних перевезень є однією з провідних галузей сучасної економіки. Вона забезпечує транспортування товарів різного призначення – від продуктів харчування, медикаментів, матеріалів, сировини до промислового обладнання та техніки. За даними досліджень, логістичні процеси та пов'язана з ними інфраструктура становлять близько 12% світового ВВП [1]. Ланцюги постачання є комплексними системами, які включають в себе велику кількість учасників: замовники, перевізники, логістичні компанії, митні та банківські установи, контролюючі органи, тощо.

На сьогоднішній день традиційні логістичні системи, часто ERP-системи, стикаються з низкою проблем, таких як відсутність прозорості в угодах, затримки в документообігу, низький рівень довіри між учасниками, можливість підробки документів, складність контролю за виконанням угод та ризик зловживань. Якщо компанії довгий час співпрацюють, то між ними виникає довіра, чого не можна сказати про нових гравців у сфері вантажних перевезень чи людей, в яких немає постійного перевізника чи клієнта.

Одним із можливих способів вирішення вищезгаданих проблем є використання технології блокчейн. Її головними перевагами є децентралізація, незмінність даних, криптографічний захист та можливість автоматизації бізнес-процесів за допомогою смарт-контрактів [2]. Блокчейн дозволяє побудувати прозору й захищену систему для організації взаємодії між сторонами у сфері вантажних перевезень. Смарт-контракти здатні забезпечити автоматичне виконання угод, фіксувати факти доставки та ініціювати оплати, що значно підвищує ефективність і безпеку процесів. Крім того, використання електронного цифрового підпису та ведення документообігу з використанням блокчейну унеможливлює внесення несанкціонованих або невідомих іншим сторонам змін у документах.

Актуальність розробки зумовлена необхідністю створення клієнт-серверного застосунку. Його основне завдання – спростити організацію вантажних перевезень та врегулювання взаємовідносин між замовником і перевізником.

Мета роботи – розробка мобільного програмного застосунку та серверу, які здатні забезпечити безпечну, прозору й автоматизовану організацію вантажних перевезень із використанням технології блокчейн, смарт-контрактів та електронного цифрового підпису.

Результати роботи можна використати для створення чи вдосконалення платформ для організації вантажних перевезень та у діяльності підприємств, які займаються вантажоперевезеннями. Окрім цього, існують перспективні напрямки подальшого розвитку. Інтеграція зі штучним інтелектом дозволить прогнозувати ризики, оптимізувати маршрути та прогнозувати попит на товари. Інтеграція з різними IoT-пристроями, як от датчики температури, вологості чи тиску, дозволить відстежувати умови транспортування, що підвищить ступінь автоматизації процесів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сфера вантажних перевезень грає велику роль у житті кожного із нас. Існує постійна необхідність доставляти товари з одного місця в інше. Це можуть бути продукти харчування, медикаменти, матеріали, сировина чи якесь обладнання. Масштаби перевезень також різняться. Іноді треба доставити вантаж у сусіднє місто, а іноді – на інший край світу.

Історія розвитку вантажних перевезень тісно пов’язана з розвитком цивілізації. Потреба транспортувати товари на далекі відстані існує з давніх часів.

Сучасна сфера вантажних перевезень охоплює різні види транспорту. Найпоширеніший тип у більшості країн – це автотранспорт. Автомобілі здатні забезпечити гнучкість маршрутів та доставку вантажу «від дверей до дверей». При цьому вони можуть перевозити менші обсяги вантажу у порівнянні з іншими типами транспорту.

Залізничний транспорт також є дуже ефективним та економічним для перевезення великих партій вантажів на великі відстані. Однак він потребує спеціальної інфраструктури, що не завжди дозволяє доставити товар до кінцевого покупця.

Основним способом доставки товарів між континентами є морські перевезення. Вони мають низьку вартість на одиницю вантажу при великих обсягах. З іншої сторони час доставки досить тривалий та існує залежність від погодних умов. Морський транспорт потребує спеціальної інфраструктури так само, як залізничний.

Авіаційні перевезення слугують для швидкої доставки вантажів на великі відстані. Їхня вартість значно вища, але вони є незамінними для доставки у віддалені регіони або термінових вантажів.

Сьогодні досить часто використовуються інтермодальні перевезення. Такий метод поєднує у собі декілька видів транспортних засобів. Він дозволяє зекономити час та кошти.

Вантажні перевезення це також і про відносини, які є важливою складовою будь якої діяльності. Велика кількість сторін приймає участь для транспортування вантажів. Кожна з них виконує свою роль:

- замовники – фізичні чи юридичні особи, які хочуть доставити вантаж та створюють попит на перевезення;
- перевізники – компанії або приватні особи, які займаються транспортуванням вантажу;
- логістичні оператори – займаються організацією процесу перевезення, підбирають маршрути;
- митні органи – контролюють міжнародні перевезення, перевіряють документи та слідкують за дотриманням законодавства;
- фінансові та страхові установи – здійснюють супровід платежів та страхуванням ризиків.

Ефективність всього процесу залежить від того, наскільки злагодженою є взаємодія між усіма сторонами. Час доставки, відповідність усіх документів, довіра до партнерів, безпека товарів грають важливу роль у галузі перевезень. У цей же час більшість рішень у цій сфері базуються на централізованих системах. Кожна сторона має власні інформаційні системи, тому для обміну даними необхідно користуватися електронною поштою, паперовими документами чи створювати API з цією метою. Це звичайно призводить до затримок.

Вантажні перевезення регулюються низкою міжнародних і національних нормативних актів та документів. Серед основних можна виділити наступні:

- CMR – транспортний документ, який найбільш широко використовується при міжнародних перевезеннях автомобільним транспортом і діє на підставі Конвенції про договір міжнародного автомобільного перевезення вантажів [3];
- Incoterms – набір правил, виданих Міжнародною торгівельною палатою (ICC), які визначають обов’язки продавців та покупців у сфері міжнародної торгівлі [4];

- e-CMR – приклад міжнародного стандарту електронного документообігу, є електронною версією документу CMR [5];
- коносамент – документ, що видається морським перевізником вантажу його відправнику, що засвідчує прийняття вантажу до перевезення і містить зобов’язання доставити вантаж до пункту призначення і передати його одержувачу [6];
- авіаційна накладна – документ, що підтверджує наявність договору між вантажовідправником і перевізником (авіалінією або агентом авіакомпанії) про перевезення вантажу по авіалініям перевізника [7].

У випадку міжнародної торгівлі для кожного вантажу необхідно пройти митний контроль. Зазвичай це включає в себе певний пакет документів. Деякі з них були щойно перелічені. Якщо з якимось документом виникне проблема, чи він загубиться, то вантаж можуть затримати на тривалий час. Пакет документів потрібен і для перевезень у межах однієї країни. І в цьому випадку проблеми з документами здатні викликати затримки, хоча і не такі довгі. Для попередження подібних ситуацій необхідно забезпечити надійний механізм для зберігання, перевірки та доступу до документів у цифровому вигляді.

Рішенням цих викликів може стати технологія блокчейн [8, 9]. Такі її характеристики як децентралізація, незмінність даних, криптографічний захист у парі зі смарт-контрактами, які дають змогу автоматизувати процеси, та децентралізованою системою збереження файлів дають можливість адаптувати сферу вантажних перевезень до наявних вимог.

Сьогодні існує велика кількість рішень для організації документообігу та взаємодії між учасниками у сфері вантажних перевезень. До них відносяться корпоративні системи управління перевезеннями, платформи для обміну документами та власні рішення великих компаній. Прикладом таких платформ є «Tocan TMS» та «Oracle SCM». Вони дозволяють планувати маршрути, відстежувати транспорт у реальному часі та вести облік вантажів та документів. Однак ці рішення є централізованими системами, а отже мають ризики. Серед проблем можна виділити вразливість до шахрайства, підробки документів та

обмежена прозорість та довіра. У таблиці 1.1 наведено поширені сценарії шахрайства та механізми їх блокування за допомогою блокчейну.

Таблиця 1.1 – Поширені сценарії шахрайства та механізми їх блокування через блокчейн

Сценарій	Опис	Механізм блокування через блокчейн
Маніпуляції з оплатою	Затримка або уникнення платежів, фальсифікація претензій про невідповідність доставки	Смарт-контракти автоматизують оплату після підтвердження доставки за допомогою GPS-трекерів, IoT-датчиків
Подвійне використання вантажу	Повторне використання одного вантажу для кількох угод	Реєстрація кожного вантажу у блокчейні з унікальним ідентифікатором, наприклад, RFID-міткою, для можливості відстеження
Підробка документів	Фальсифікацій накладних, митних декларацій, сертифікатів	Незмінність даних забезпечить автентичність документів
Схеми з посередниками	Змова з посередниками з метою завищення витрат або приховування частини транзакцій	Децентралізована природа блокчейну усуває потребу в більшості посередниках, а смарт-контракти автоматизують оплату та контроль доставки

Чудовим прикладом використання технології блокчейн для перевезення вантажів є система «Cargo Release» компанії Global Shipping Business Network [10]. GSBN позиціонує себе як нейтральний, некомерційний консорціум, який забезпечує доступну та сталу світову торгівлю без паперових документів [11]. Їхній сервіс ефективно вирішує одну із проблем у морській логістиці – повільний і уразливий до помилок та шахрайства документообіг. Для отримання вантажу в порту необхідно надати відповідний документ – коносамент. Цей процес займає

багато часу, включає численні перевірки та призводить до затримок у випадку втрати документів чи виявлення помилки або факту шахрайства. «Cargo Release» цифровізує даний процес за допомогою блокчейну, значно пришвидшує час обробки документів, підвищує безпеку та робить процес більш прозорим для усіх учасників. Компанія стверджує, що сервіс скорочує час розвантаження з декількох днів до декількох годин. Станом на 2025 рік сервіс доступний у 24 портах світу і з його допомогою було реалізовано півтора мільйона доставок.

1.2 Виявлення та вирішення проблем

У ході аналізу предметної галузі було виявлено низку проблем у сфері вантажних перевезень. Розглянемо причини проблем, їх наслідки та варіанти вирішення.

Використання різних інформаційних систем, локальних рішень чи паперового документообігу призводить до зниження узгодженості між учасниками, збільшення часу, який витрачається на завантаження та розвантаження, помилках у маршрутах та документах. Дану проблему допоможе вирішити створення єдиного сервісу із використанням технології блокчейн у критичних точках для підвищення прозорості та надійності.

Використання паперових документів призводить до затримок під час обробки та передачі документів. Також є вірогідність просто загубити їх. Як наслідок подовжується загальний час транспортування, що може призвести до прострочення термінів та накопичення вантажів на складах. Рішенням може стати використання цифрового документообігу з можливістю підписувати документи за допомогою електронного цифрового підпису.

Ще однією проблемою, пов’язаною із використанням паперового документообігу та централізованих рішень, є можливість шахрайства та підробки документи. Як наслідок зростає недовіра між партнерами, компанії втрачають репутацію, виникають юридичні спори, які можуть затягнутися на довгий період та потягнути за собою великі витрати. Згідно CargoNet [12] у 2023 році кількість крадіжок товарів виросла на 57% у порівнянні з минулим роком. За їхніми даними

було втрачено товарів на суму $130 мільйонів. Для вирішення даної проблеми можна зберігати документи у розподіленій файловій системі зі збереженням гешу документу у блокчейні, що гарантує незмінність.

Пошук нових партнерів для клієнтів, які не мають постійного замовника чи перевізника, може стати проблемним через відсутність довіри між ними. Як наслідок учасники процесу починають вимагати один від одного гарантії, проводити додаткові аудити, звертатися до посередників, що збільшує витрати. Сервіс із використанням технології блокчейн у критичних точках здатен підвищити рівень прозорості і надійності. Смарт-контракти можна використати для автоматизації таких процесів, як дотримання виконання угод. Таким чином можна підняти рівень довіри між партнерами.

Технологія блокчейн може стати рішенням більшості розглянутих проблем. Вона здатна забезпечити децентралізовану, надійну та прозору систему для взаємодій у сфері вантажних перевезень. Смарт-контракти здатні забезпечити автоматичне виконання умов угод. Зі свого боку електронний цифровий підпис допоможе вести електронний документообіг.

1.3 Постановка задачі

Метою розробки є створення інформаційної системи, яка здатна задовільнити наступні вимоги:

- забезпечити взаємодію між замовником та перевізником з мінімізацією кількості посередників;
- автоматизувати укладання договорів та контроль їх виконання;
- реалізувати цифровий документообігу зі збереженням документів у розподіленій файловій системі із записом кешу документу у блокчейні та можливістю підписати документи електронним цифровим підписом;
- підвищити рівень довіри до малих і середніх перевізників, а також до клієнтів, які не мають постійного перевізника, за рахунок відгуків та автоматизації виконання угод;

- забезпечити масштабованість системи для її подальшого розвитку та виходу на інші ринки;
- забезпечити користувачам зручний доступ до функціоналу системи на мобільних пристроях.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Програмна система для організації вантажних перевезень і врегулювання відносин між замовником і перевізником має бути надійною та відповідати сучасним стандартам безпеки. Основна мета застосунку – забезпечити прозорість, безпечність та автоматизацію ключових процесів у сфері вантажних перевезень.

Система будується за клієнт-серверною архітектурою. Вона включає в себе серверну частину та мобільний застосунку у якості клієнтської частини. Також застосунок має інтеграцію з розподіленою файловою системою, блокчейном та смарт-контрактами.

Серверна частина – веб-сервіс з REST API-архітектурою [13]. Вона відповідає за обробку запитів клієнтів через HTTP-запити, реалізує бізнес-логіку програмної системи. Сервер має бути масштабованим, забезпечувати логування та відповідати сучасним вимогам безпеки.

Клієнтська частина – мобільний застосунок для платформи Android [14]. Вона має забезпечити користувачам зручний доступ до функціоналу системи, підтримувати отримання push-повідомлень. Мінімальна версія, що підтримується API 23 (Android 6.0 Marshmallow).

У системі передбачається використання технології блокчейн та смарт-контрактів для забезпечення прозорості укладання угод, автоматизації оплати та контролю виконання угод.

Передбачається можливість ведення електронного документообігу. Документи зберігати у розподіленій файловій системі, при цьому зберігати кеш файлу у блокчейні для забезпечення незмінності документа. Надати користувачу можливість підписувати документи за допомогою електронного цифрового підпису. Можна використати існуючий сервіс для підписів, або власну реалізацію на основі алгоритмів RSA/ECDSA [15].

Функціональні вимоги до системи:

– реєстрація та авторизація користувачів;

- управління профілем користувача – редагування особистої інформації, перегляд історії замовлень та виконаних перевезень;
- управління замовленнями на перевезення вантажів – можливість створення заявок на перевезення із зазначенням необхідних параметрів;
- пошук замовлень на перевезення вантажів за різними критеріями;
- надання та управління пропозиціями на виконання замовлень;
- автоматизація укладання договорів, контроль виконання угоди;
- автоматизація оплати – у разі успішного виконання угоди оплата здійснюється автоматично через смарт-контракт;
- відстеження вантажу по GPS для підтвердження виконання угоди;
- розрахунок приблизних витрат палива на маршрут;
- система відгуків та рейтингу;
- ведення електронного документообігу, а саме можливість додавати супровідні документи (ТТН, акти) та підписання документів електронним цифровим підписом.

Нефункціональні вимоги:

- підтримка навантаження до 1000 одночасних користувачів;
- час відгуку API не більше 1 секунди на більшість запитів;
- обмін даними між клієнтом і сервером через захищене з’єднання (HTTPS);
- збереження чутливих даних у зашифрованому вигляді;
- доступ до функціоналу системи за ролями;
- система повинна бути гнучкою та масштабованою.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML-проєктування ПЗ

На основі вимог до програмної системи визначено основних користувачів системи та їхні бізнес-можливості. Їх зображено на UML-діаграмі прецедентів (див. рис. 3.1).

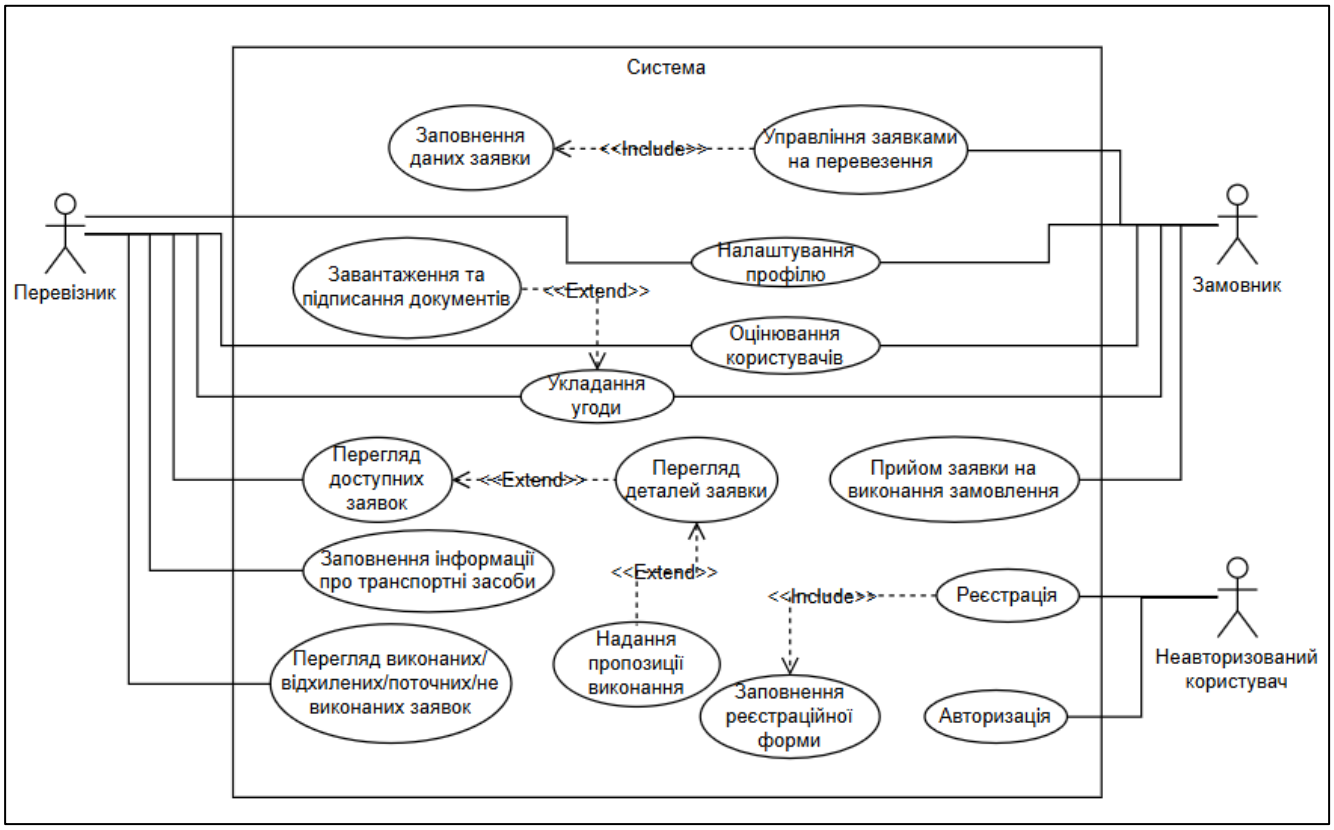

Рисунок 3.1 – UML діаграма прецедентів програмної системи (рисунок виконаний самостійно)

Розглянувши діаграму бачимо, що основними користувачами системи є:

– неавторизований користувач;
– замовник;
– перевізник.

До сценаріїв взаємодії неавторизованого користувача з системою входять:

– можливість авторизуватися у системі;

- можливість зареєструватися у системі; реєстрація у системі включає у себе заповнення реєстраційної форми.

До сценаріїв взаємодії замовника з системою входять:

- управління заявками на перевезення; створення та редагування заявок включає в себе обов’язкове заповнення даних заявки;
- можливість налаштувати профіль користувача;
- можливість укласти угоду; за потреби завантажити супровідні документи та підписати їх за допомогою електронного цифрового підпису;
- можливість оцінювати інших користувачів;
- можливість прийняти заявку на виконання замовлення серед запропонованих.

До сценаріїв взаємодії перевізника з системою входять:

- можливість переглядати доступні заявки на перевезення вантажу;
- можливість переглядати деталі заявки на перевезення вантажу;
- можливість надати пропозицію виконання заявки на перевезення вантажу;
- можливість заповнювати інформацію про транспортні засоби;
- перегляд заявок та їх статусів;
- можливість укласти угоду; за потреби завантажити супровідні документи та підписати їх за допомогою електронного цифрового підпису;
- можливість налаштовувати профіль користувача;
- можливість оцінювати інших користувачів.

На UML-діаграмі послідовностей «Створення замовлення» (див. рис. 3.2) зображено сценарій, який показує взаємодію користувачів із системою та компонентів системи між собою під час створення замовлення на перевезення вантажу.

Замовник створює замовлення та заповнює необхідні дані. Мобільний застосунок надсилає дані замовлення на сервер. Далі інформація зберігаються у базу даних, а замовник отримує відповідне повідомлення.

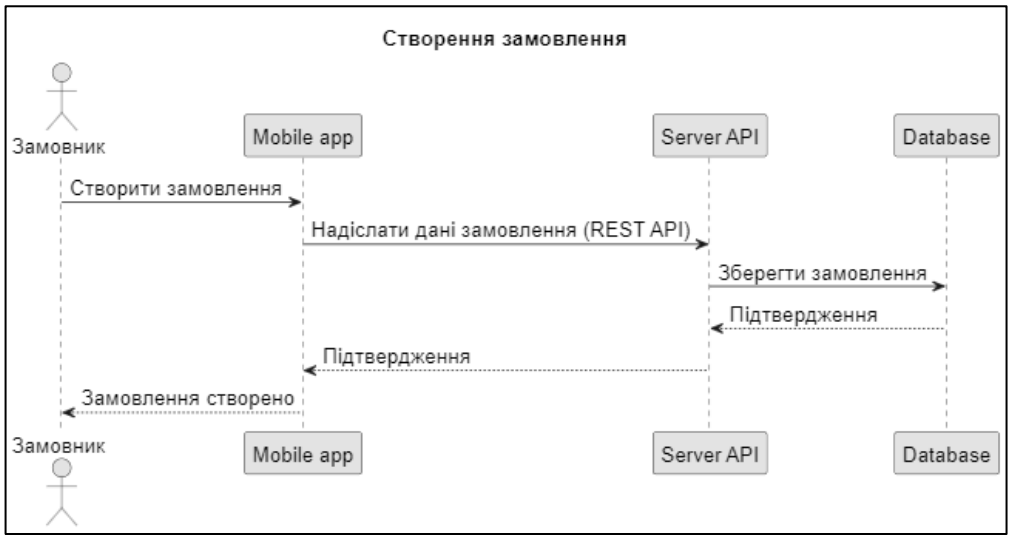

Рисунок 3.2 – UML-діаграма послідовностей «Створення замовлення» (рисунок виконаний самостійно)

На UML-діаграмі послідовностей «Створення пропозиції» (див. рис. 3.3) можна побачити сценарій, який відображає ще один етап взаємодії користувача з системою.

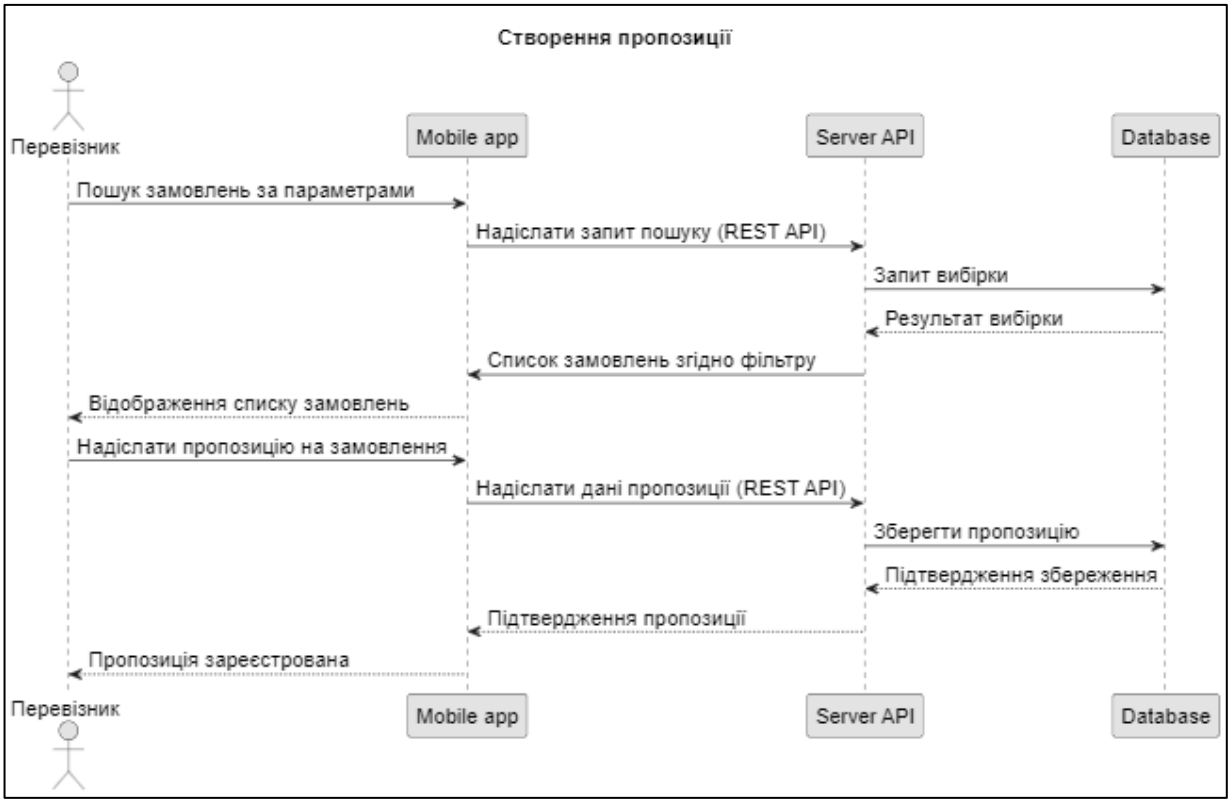

Рисунок 3.3 – UML-діаграма послідовностей «Створення пропозиції» (рисунок виконаний самостійно)

Перевізник здійснює пошук замовлень за параметрами та отримує список відфільтрованих замовлень. Далі він обирає те, що йому до вподоби, та заповнює дані пропозиції. Після цього пропозиція зберігається у базу даних, а перевізник бачить відповідне повідомлення.

На UML-діаграмі послідовностей «Вибір перевізника та створення договору» (див. рис. 3.4) можна побачити як відбувається вибір перевізника та створення угоди. Після перегляду наданих до замовлення пропозицій замовник обирає одну з них. Далі відповідна інформація зберігається у базі даних. Отримавши підтвердження з серверу, мобільний застосунок викликає смарт-контракт для запису угоди на блокчейн. В цей час смарт-контракт генерує подію, на яку реагує сервер та зберігає запис угоди у базу даних.

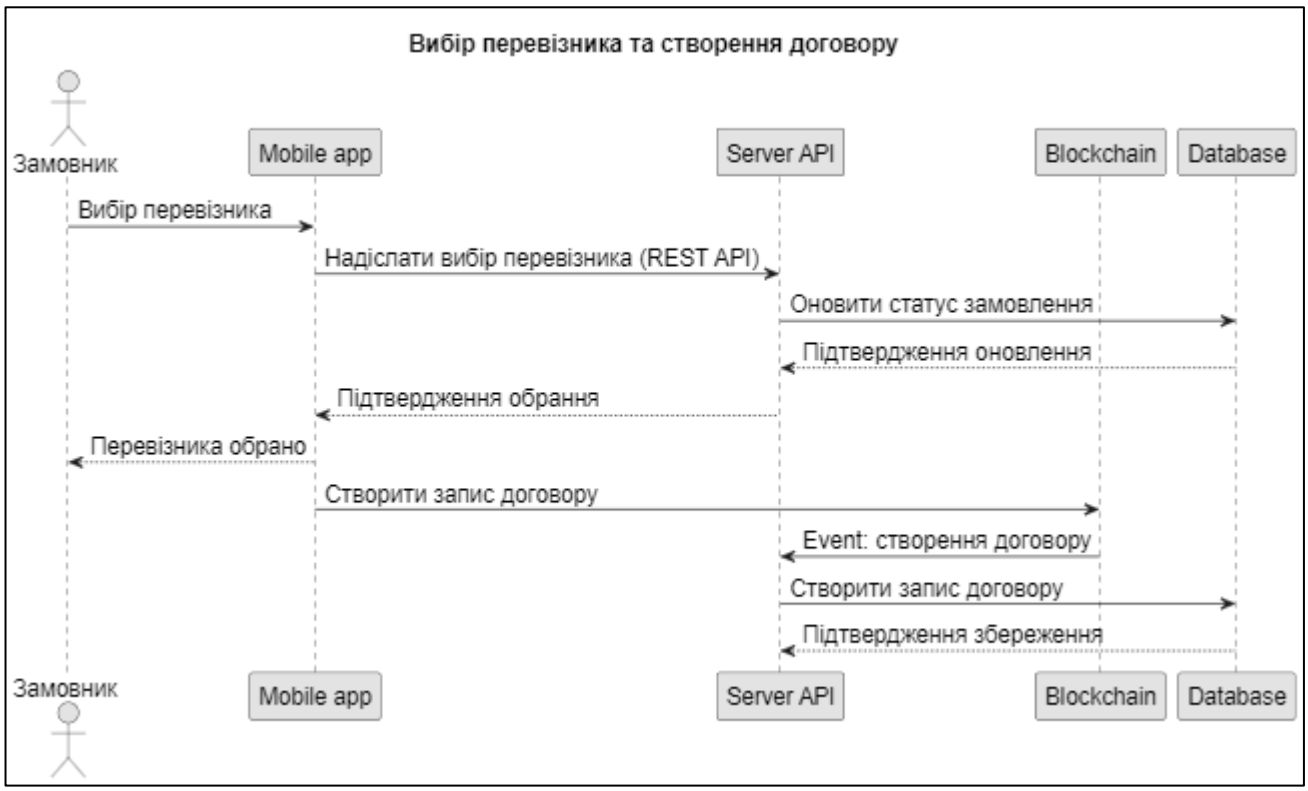

Рисунок 3.4 – UML-діаграма послідовностей «Вибір перевізника та створення договору» (рисунок виконаний самостійно)

На UML-діаграмі послідовностей «Завантаження документа» (див. рис. 3.5) бачимо процес завантаження документа.

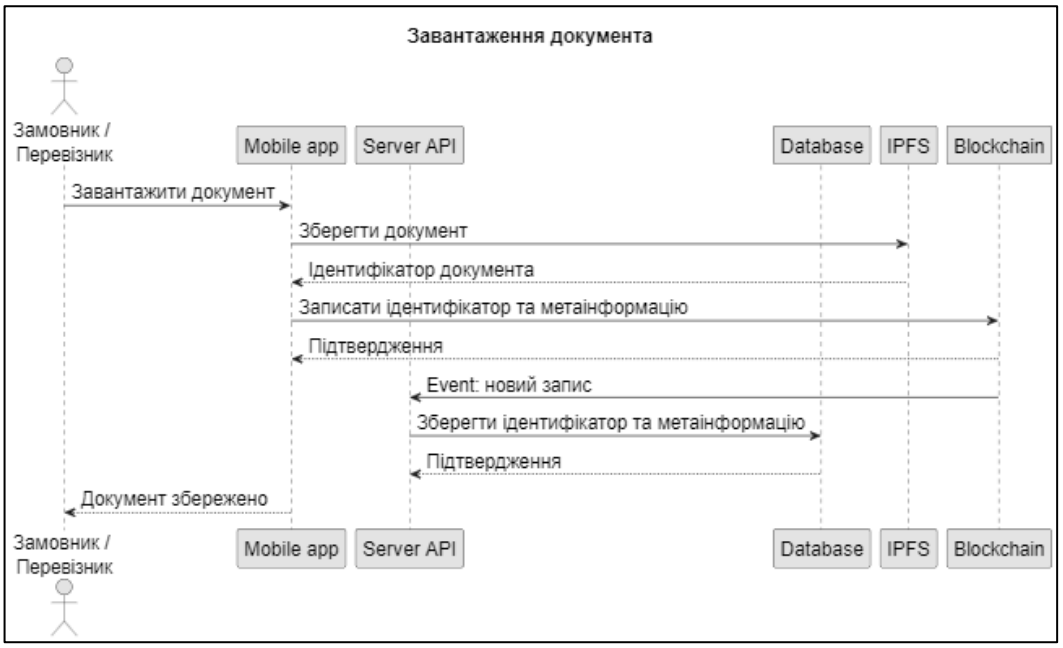

Рисунок 3.5 – UML-діаграма послідовностей «Завантаження документа» (рисунок виконаний самостійно)

На UML-діаграмі послідовностей «Підписання документа» (див. рис. 3.6) зображено процес підписання документа.

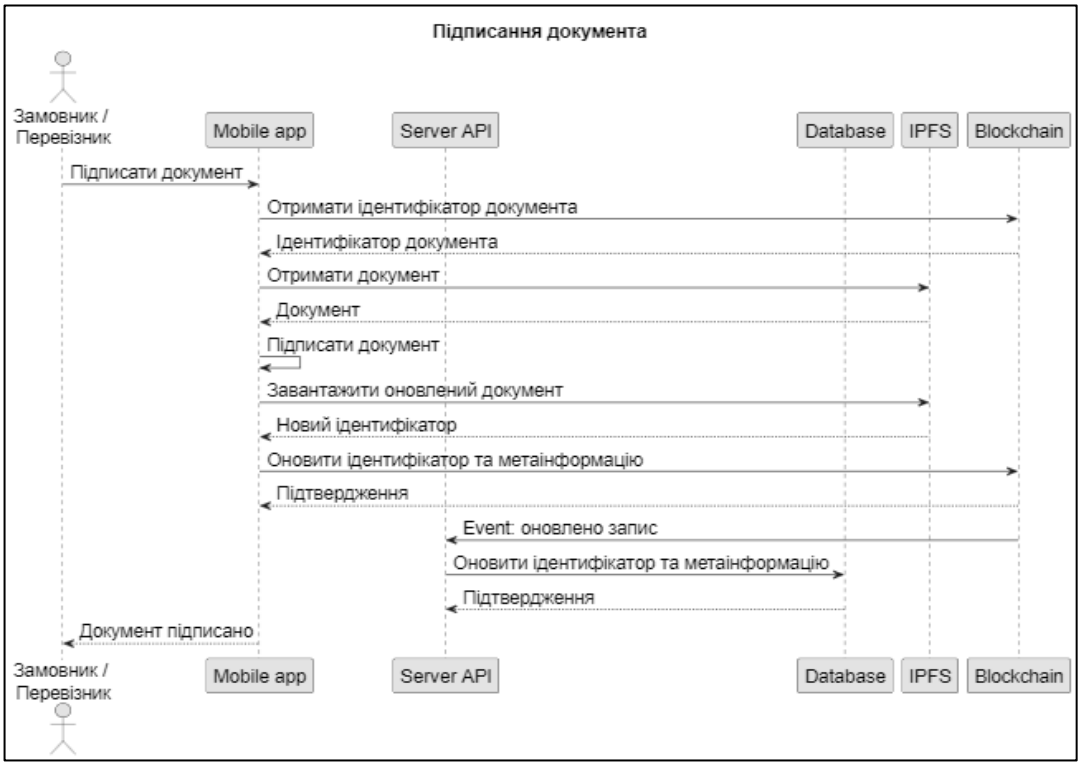

Рисунок 3.6 – UML-діаграма послідовностей «Підписання документа» (рисунок виконаний самостійно)

Спочатку користувач ініціює підписання. Далі мобільний застосунок робить запит для отримання ідентифікатора файлу з блокчейну. Наступний крок – отримати файл з IPFS. Після цього документ підписується на клієнтській стороні. Далі оновлений файл завантажується у IPFS а нові дані про документ записуються на блокчейн. Смарт-контракт генерує подію і сервер оновлює інформацію у базі даних.

На UML-діаграмі послідовностей «Оплата» (див. рис. 3.7) бачимо як у системі відбувається оплата.

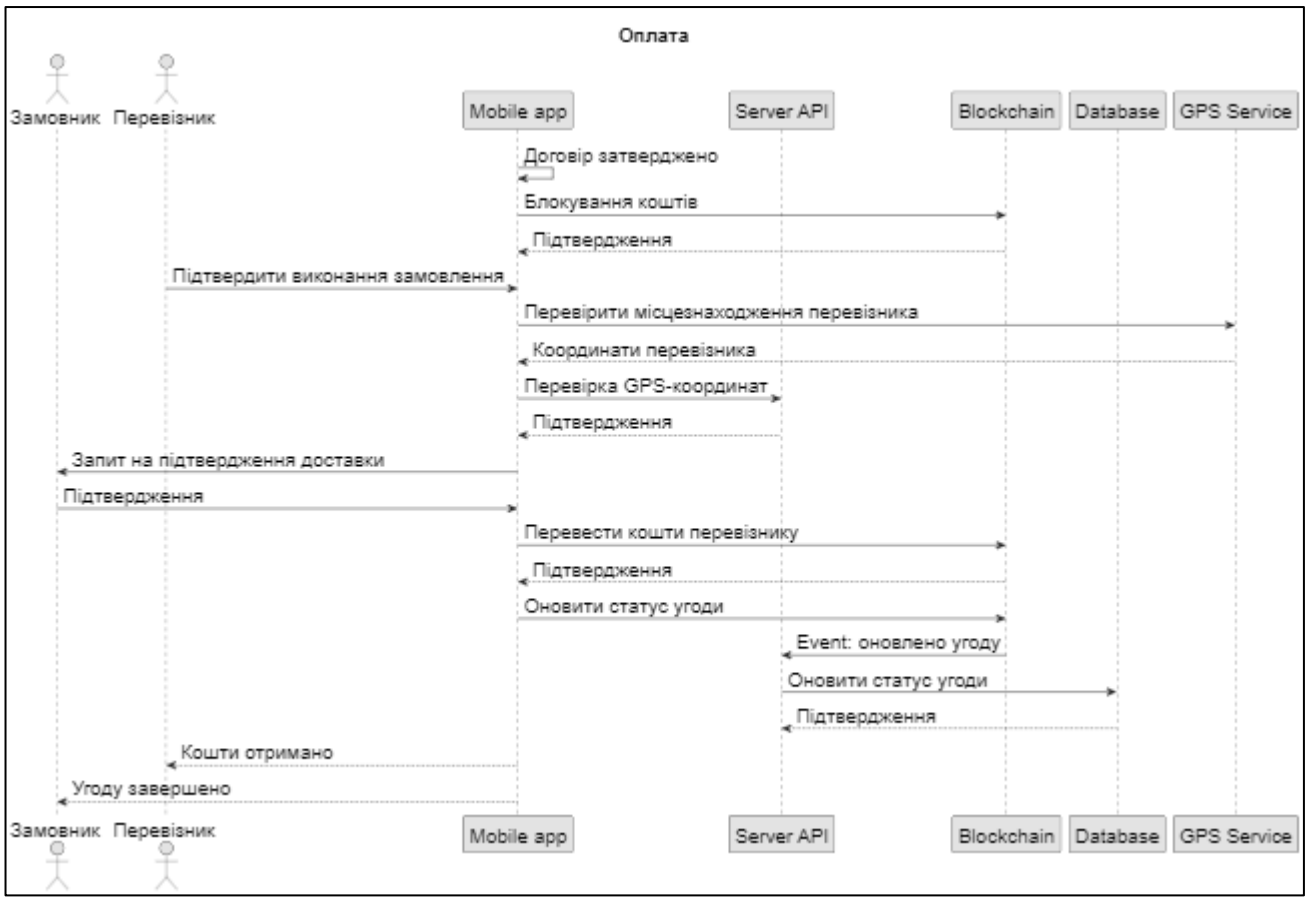

Рисунок 3.7 - UML-діаграма послідовностей «Оплата» (рисунок виконаний самостійно)

Після затвердження договору за допомогою смарт-контракту з рахунку замовника знімаються кошти. Через певний час перевізник підтверджує завершення замовлення. Далі система перевіряє GPS-координати і очікує на підтвердження від замовника. Після завершення усіх перевірок кошти

надсилаються на рахунок перевізника. У кінці відбувається оновлення статусу замовлення.

3.2 Проєктування архітектури ПЗ

Програмна система для організації вантажних перевезень представляє собою клієнт-серверний застосунок з інтеграцією блокчейну та розподіленої файлової системи. Обрана архітектура гарантує масштабованість, безпеку, прозорість процесів та ефективну взаємодію між учасниками. UML-діаграма розгортання (див. рис. 3.8) показує основні частини системи та їх взаємодію між собою. Загальна структура системи складається з п’яти основних частин: сервер, база даних, клієнт (мобільний застосунок), блокчейн та розподілене сховище файлів (IPFS).

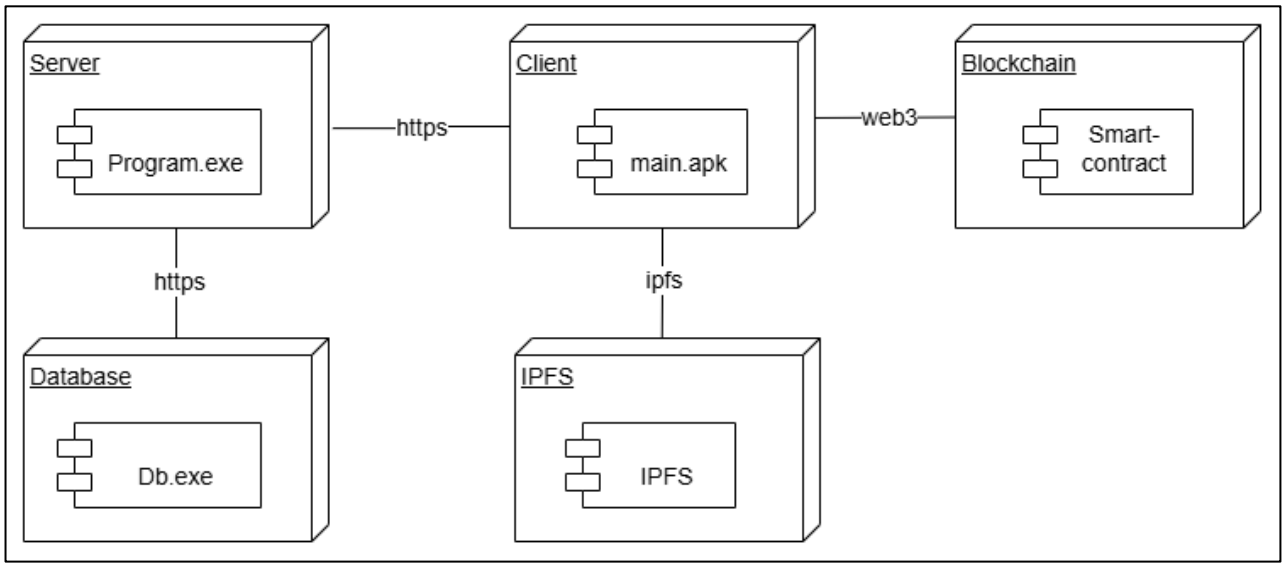

Рисунок 3.8 – UML-діаграма розгортання програмної системи (рисунок виконано самостійно)

Сервер є центральним елементом програмної системи. Він відповідає за обробку HTTP-запитів від клієнтів, авторизацію та аутентифікацію користувачів, реалізує основну бізнес-логіку та організовує взаємодію між іншими частими системи. Для серверу обрано багатошарову архітектуру. Вона забезпечує високий рівень абстрактності та надає можливість легко змінювати наявний функціонал і

додавати новий. На початку розробки багатошарову архітектуру легше підтримувати, у неї легше вносити зміни, ніж у архітектуру мікросервісів. Одночасно з цим при виникненні потреби програма може бути досить легко переписана на архітектуру мікросервісів за умови, що вона створена з дотриманням принципів ООП та SOLID [16].

База даних виконує роль сховища даних. Для системи обрано реляційну СУБД. Такий тип бази найкраще підходить для систем, де необхідно зберігати дані об’єктів та зв’язки між ними. Для взаємодії з сервером використовується протокол HTTPS.

Клієнтська частина – це мобільний застосунок основна задача якого надати користувачам доступ до основного функціоналу системи. Для безпечного обміну запитами з сервером використовується протокол HTTPS. Для клієнту обрано архітектуру MVVM. Вона забезпечує розділення відповідальності, зручно тестується, підтримує масштабування та легко поєднується з іншими компонентами.

Основна задача блокчейну – забезпечувати прозорість процесів та незмінність важливих даних. Клієнт взаємодії з блокчейном через протокол Web3.

Розподілене файлове сховище використовується для зберігання документів. Завдяки децентралізованій природі таке сховище здатне забезпечити незмінність файлів, що є важливим елементом у програмній системі. Клієнт взаємодіє зі сховищем через протокол IPFS.

Розглянута архітектура поєднує у собі централізовану систему з децентралізованими елементами, що дозволяє досягти високої продуктивності системи і при цьому підвищити рівень її надійності і прозорості.

3.3 Проєктування структури зберігання даних

На ER-діаграмі (див. рис. 3.9) зображено основні сутності системи, їхні поля та зв’язки між ними.

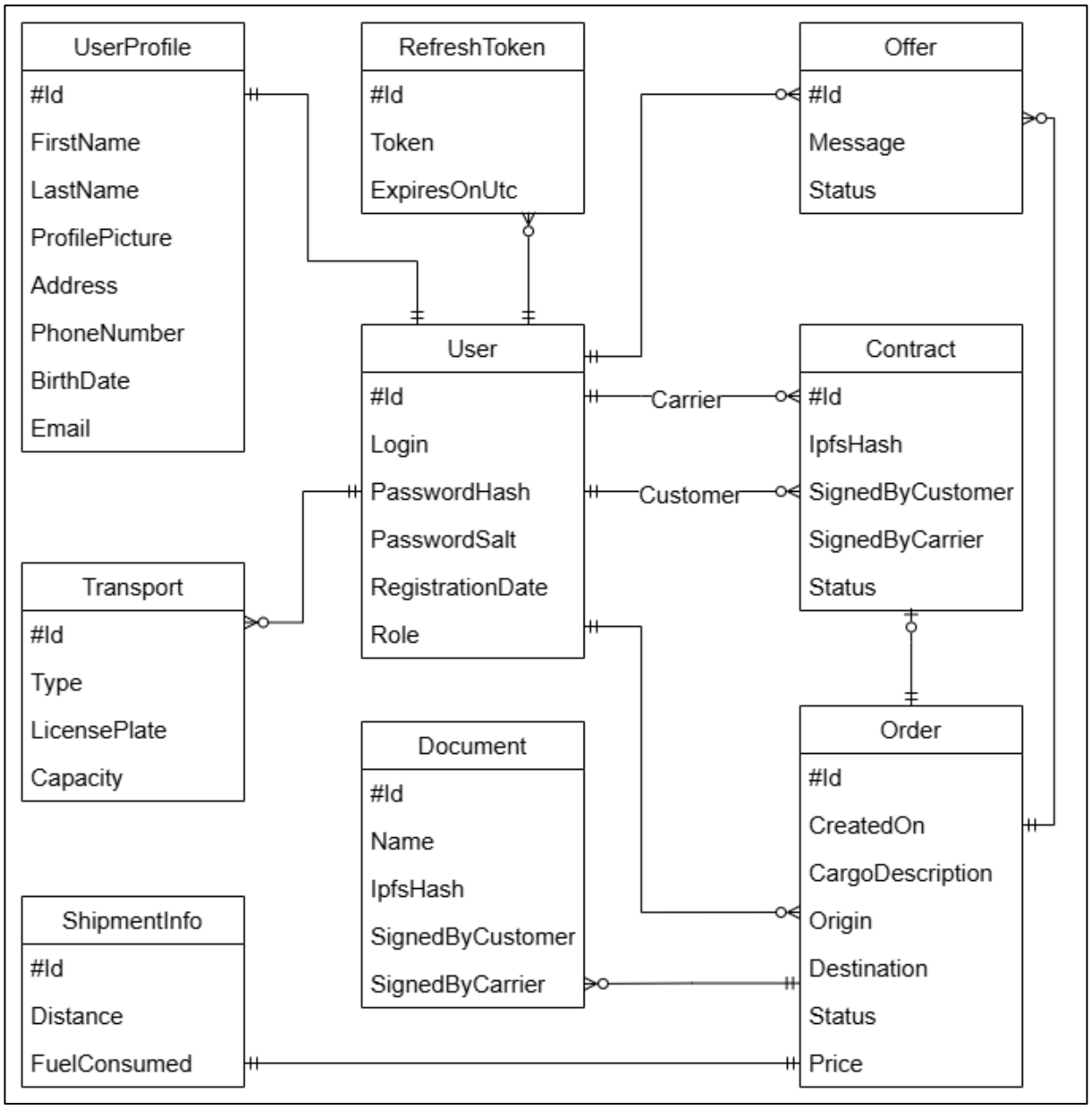

Рисунок 3.9 – ER-модель даних програмної системи (рисунок виконаний самостійно)

На діаграмі можна побачити наступні сутності:

– користувач («User») – представляє користувача системи, містить дані автентифікації, роль та дату реєстрації;
– профіль користувача («UserProfile») – містить додаткову інформацію про користувача системи, а саме: ім’я, прізвище, номер телефону, адресу, дату народження та email;

- токен оновлення («RefreshToken») – допоміжна сутність для оновлення токену авторизації користувача;
- замовлення («Order») – заявка на перевезення вантажу, яку створює замовник, містить опис вантажу, ціну та статус;
- пропозиція («Offer») – пропозиція, яку надає перевізник, на виконання певного замовлення, містить коментар від перевізника та статус;
- договір («Contract») – угода між замовником і перевізником;
- документ («Document») – супровідний документ до замовлення;
- інформація про доставку («ShipmentInfo») – додаткова інформація про замовлення, містить дистанцію маршруту та кількість витраченого палива;
- транспортний засіб («Transport») – містить інформацію про транспортний засіб, зареєстрований перевізником, а саме: тип, номер та вантажопідйомність.

Розглянемо зв’язки між сутностями:

- користувач та профіль користувача мають відношення «один-до-одного»;
- користувач може мати 0 або багато замовлень водночас замовлення може мати лише одного користувача – відношення «один-до-багатьох»;
- користувач може мати 0 або багато пропозицій на виконання водночас пропозиція на виконання може мати лише одного користувача – відношення «один-до-багатьох»;
- користувач може мати 0 або багато транспортних засобів водночас транспортний засіб може мати лише одного користувача – відношення «один-до-багатьох»;
- користувач може мати 0 або багато токенів оновлення водночас токен відноситься лише до одного користувача – відношення «один-до-багатьох»;
- користувач як замовник може мати 0 або багато договорів водночас договір може мати лише одного замовника – відношення «один-до-багатьох»;

- користувач як перевізник може мати 0 або багато договорів водночас договір може мати лише одного перевізника – відношення «один-до-багатьох»;
- замовлення може мати 0 або багато пропозицій водночас пропозиція може мати лише одне замовлення – відношення «один-до-багатьох»;
- замовлення може мати 0 або один контракт водночас контракт може мати лише одне замовлення – відношення «один-до-одного»;
- замовлення може мати 0 або багато документів водночас документ може мати лише одне замовлення – відношення «один-до-багатьох»;
- замовлення може мати один запис інформації про доставку водночас запис інформації про доставку може мати лише одне замовлення – відношення «один-до-одного».

3.4 Приклади найцікавіших алгоритмів та методів

Система відгуків та рейтингу є важливою частиною програмної системи, адже вона має вплив на відношення користувачів один до одного. Чим вищий рейтинг у користувача, тим більша ймовірність, що з ним будуть співпрацювати інші люди. Саме тому важливо правильно розрахувати цей самий рейтинг. Він розраховується за наступною формулою

$$R = 0.4 * \frac{1}{n}\sum_{i=1}^{n} r_i + 0.6 * \frac{m}{k} \quad (3.1)$$

де R – рейтинг користувача;

n – кількість завершених замовлень, які мають оцінку;

r_i – оцінка користувача і-го замовлення;

m – кількість успішно завершених замовлень;

k – всього замовлень користувача.

З формули 3.1 бачимо, що у формуванні рейтингу користувача приймає участь середнє арифметичне з певної кількості останніх оцінок, які були

проставлені іншими користувачами, а також відсоток успішно завершених замовлень.

Для перевізника важливо розуміти приблизні витрати палива на маршрут під час вибору замовлення. Програмна система розраховує цю величину за такою формулою

$$P = \frac{s}{100} * \frac{1}{n} \sum_{i=1}^{n} c_i * p \quad (3.2)$$

де P – орієнтована вартість палива;

n – кількість завершених замовлень, які мають статистику по витраченому паливу;

c_i – витрати палива на 100 кілометрів за i-те замовлення;

p – актуальна ціна палива.

Окрім перевізника ця інформація може стати в нагоді і замовнику. Розраховуючи вартість палива на маршрут, можна одразу пропонувати замовнику встановити рекомендовану суму за доставку, на яку погодяться водії.

Беручи до уваги, що в кожного транспортного засобу витрати палива відрізняються, у формулі враховуються витрати палива для транспорту відповідного користувача. Якщо перевізник ще невідомий або в нього не вистачає даних для розрахунку вартості, для розрахунку використовуються дані з останніх замовлень, які були виконані на відповідному типі транспортного засобу.

3.5 Створення UI/UX

У програмній системі, що розроблюється, інтерфейсом для взаємодії користувача із системою виступає мобільний застосунок. Дизайн має бути адаптивним, мінімалістичним та інтуїтивно зрозумілим. Враховуючи функціональні вимоги до системи, описані у розділі 3, необхідно забезпечити мобільний застосунок наступними сторінками:

– сторінка реєстрації – призначена для створення нового облікового запису, містить форму, в яку користувач вписує свої дані;

- сторінка авторизації – призначена для того, щоб користувач міг увійти в систему;
- сторінка профілю користувача – відображає рейтинг, особисту інформацію користувача, дозволяє редагувати її тут, а для перевізників також є можливість вказати свої транспортні засоби;
- головна сторінка – наповнення змінюється в залежності від того, ким є користувач; для замовника відображає список створених заяв та їхні статуси; для перевізника відображає список доступних заявок з короткою інформацією про вантаж і маршрут;
- сторінка створення / редагування замовлення – доступна лише замовнику; містить форму для заповнення інформації нового замовлення чи редагування даних існуючого;
- сторінка для перегляду деталей замовлення – містить повну інформацію по замовленню; з цієї сторінки перевізник може відправити пропозицію на виконання, а замовник – прийняти її;
- сторінка договору – відображає умови угоди та коротку інформацію про вантаж; звідси можна підписати договір;
- сторінка документів – дозволяє переглядати супровідні до замовлення документи та підписувати їх за потреби;
- модальне вікно для оцінки користувача – слугує для оцінки користувача після виконання замовлення.

Мобільний застосунок має підтримувати темну та світлу тему інтерфейсу. Він повинен мати інтуїтивно зрозумілу навігацію та анімації переходів. Також застосунок повинен підтримувати українську й англійську локалізації та мати можливість надсилати пуш-повідомлення.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Для реалізації програмної системи було обрано клієнт-серверну архітектуру. До основних частин застосунку відносяться серверна частина, клієнтська частина, база даних, блокчейн та IPFS. Це поєднання дозволяє створити централізовану систему, яка містить децентралізовані елементи. Такий підхід допомагає досягти високої продуктивності і при цьому підвищити рівень надійності та прозорості системи.

Серверна частина реалізована у вигляді веб сервісу з REST API-архітектурою. REST базується на стандартних HTTP методах і є незалежним від платформи та мови програмування. Він простий у використанні, не вимагає додаткових обгорток поверх протоколу HTTP та має широку підтримку серед багатьох інструментів.

Сервер представляє собою ASP.NET Core Web API застосунок, написаний на платформі .NET 8 та мові програмування C#. ASP.NET Core – це фреймворк для створення веб-застосунків на платформі .NET. Він продуктивний, кросплатформний та масштабований. З його допомогою можна писати як невеликі, так і високонавантажені системи. Мова програмування C# – це об’єктно-орієнтована мова програмування. Вона має потужну стандартну бібліотеку, яка дозволяє працювати з файлами, шифруванням, HTTP-запитами, тощо. Окрім того C# підтримує асинхронність, що є критично важливим для забезпечення ефективної обробки запитів REST-сервісів.

Серверна частина має багатошарову архітектуру. Вона поділена на такі шари: Domain, Data Access Layer (DAL), Business Logic Layer (BLL) та Web API. Ця архітектура забезпечує високий рівень абстрактності та надає можливість легко змінювати наявний функціонал чи додавати новий.

«Domain» містить сутності, які відображають модель предметної області. Усі сутності спадкують абстрактний базовий клас «BaseEntity».

«DAL» слугує для забезпечення взаємодії з базою даних. Було використано ORM Entity Framework Core. Він дає змогу працювати з базою даних за допомогою C# об’єктів без потреби писати SQL-запити та захищає від SQL-ін’єкцій. Окрім

того ORM дозволяє не прив’язуватися до конкретної СУБД і за потреби легко замінити її. Для створення бази даних та застосування змін до її структури використовується підхід «code first». Конфігурація сутностей здійснюється за допомогою реалізації інтерфейсу «IEntityTypeConfiguration<T>». Міграції відстежуються та застосовуються до бази даних автоматично під час запуску серверу. За це відповідає метод, що розташовано нижче.

```
public static void ApplyMigrations(this WebApplication app)
{
    using var scope = app.Services.CreateScope();
    var dbContext =
scope.ServiceProvider.GetRequiredService<DiCargoHubApiDbContext>();

    var pendingMigrations =
dbContext.Database.GetPendingMigrations();
    if (pendingMigrations.Any())
    {
        Console.WriteLine("Applying pending migrations...");
        dbContext.Database.Migrate();
        Console.WriteLine("Migrations applied successfully.");
    }
    else
    {
        Console.WriteLine("No pending migrations found.");
    }
}
```

У «DAL» використано шаблони «Repository» та «UnitOfWork», які спрощують роботу з базою даних та підвищують гнучкість архітектури.

У системі реалізовано базовий репозиторій, який містить типові методи для проведення операцій із будь-якою сутністю. Фрагмент реалізації наведено нижче.

```
public class Repository<TEntity> : IRepository<TEntity>
    where TEntity : BaseEntity
{
    private readonly DiCargoHubApiDbContext _dbContext;
    private readonly DbSet<TEntity> _dbSet;

    public Repository(DiCargoHubApiDbContext dbContext)
    {
        _dbContext = dbContext;
        _dbSet = _dbContext.Set<TEntity>();
    }

    public virtual async Task AddAsync(TEntity entity)
    {
```

```
            await _dbSet.AddAsync(entity);
        }

        public virtual TEntity? Get(Expression<Func<TEntity, bool>>
predicate)
        {
            return _dbSet.FirstOrDefault(predicate);
        }

        public virtual IQueryable<TEntity> GetAll()
        {
            return _dbSet.AsQueryable();
        }

        public async virtual Task<TEntity?> GetByIdAsync(Guid id)
        {
            return await _dbSet.FindAsync(id);
        }

        public async Task<List<TEntity>>
GetListAsync(Expression<Func<TEntity, bool>> predicate)
        {
            return await _dbSet
                .Where(predicate)
                .ToListAsync();
        }

        public virtual void Remove(TEntity entity)
        {
            _dbSet.Remove(entity);
        }

        public virtual void Update(TEntity entity)
        {
            _dbSet.Update(entity);
        }

        public virtual void UpdateRange(params TEntity[] entities)
        {
            _dbSet.UpdateRange(entities);
        }
    }
```

«UnitOfWork» забезпечує цілісність та атомарність операцій з базою даних. Він дозволяє виконувати кілька операцій над різними сутностями в межах однієї транзакції. Фрагмент реалізації наведено у додатку В.

«BLL» містить бізнес-логіку проєкту. Цей шар містить сервіси, кожен з яких відповідає за обробку певної частини логіки застосунку. Кожен з сервісів реалізує відповідний інтерфейс з метою дотримання принципу інверсії залежностей.

«Web API» є зовнішнім інтерфейсом між серверною логікою та мобільним клієнтом. Обмін даними між клієнтом і сервером відбувається у JSON форматі. Контролери відповідають за маршрутизацію HTTP-запитів. Вони перевіряють наявність доступу у користувача, валідність вхідних даних та виклики відповідних сервісів. JWT-токени використовуються для авторизації користувачів у системі. Також у «Web API» реалізовано глобальний обробник помилок. Реалізація наведена у додатку В.

Бібліотека «Autofac» використовується системою як контейнер впровадження залежностей.

Конфігурація проєкту зберігається у файлі «appsettings.json». Він містить налаштування авторизації та рядок підключення до БД.

У якості бази даних обрано СУБД Microsoft SQL Server. Вона здатна забезпечити високу продуктивність, безпеку даних та масштабованість. Окрім того Microsoft SQL Server чудово інтегрується з іншими продуктами Microsoft, зокрема платформою .NET.

IPFS використовується системою у якості децентралізованого сховища документів. Застосування цієї технології забезпечує незмінність та стійкість до відмов збереження важливих файлів.

Для реалізації смарт-контрактів у програмній системі обрано платформу Ethereum. Дана платформа є однією з найпоширеніших та має велику кількість користувачів. Вона підтримує велику кількість бібліотек та зручних інструментів, що полегшує розробку. Ethereum використовує мову програмування Solidity для написання смарт-контрактів. У межах проєкту створено два смарт-контракти: «ContractRegistry» та «PaymentEscrow». Перший смарт-контракт відповідає за створення, підписання та завершення контрактів. Також він відповідає за додавання метаданих документів і має декілька методів для отримання інформації. Другий смарт-контракт керує оплатою. Через нього відбувається зняття коштів з рахунку замовника та зарахування оплати на рахунок перевізника. Ще замовник може повернути кошти, якщо контракт було скасовано. Фрагменти коду реалізації смарт-контрактів наведено у додатку В.

Клієнтська частина програмної система – це мобільний застосунок для платформи Android. Вона побудована за принципами архітектури MVVM. Даний тип архітектури підтримує розділення відповідальності та полегшує тестування.

Під час розробки мобільного застосунку було використано такі бібліотеки:

– «Jetpack Compose» для побудови інтерфейсу;
– «Room» для зберігання даних локально;
– «Hilt» для механізму ін’єкції залежностей;
– «Web3j» та «WalletConnect» для взаємодії зі смарт-контрактами.

Підписання документів реалізовано на клієнтській стороні.

На рисунках 4.1 та 4.2 зображено сторінку реєстрації та сторінку додавання транспортного засобу відповідно. Вони демонструють основну ідею створеного дизайну, тобто простоту та мінімалізм.

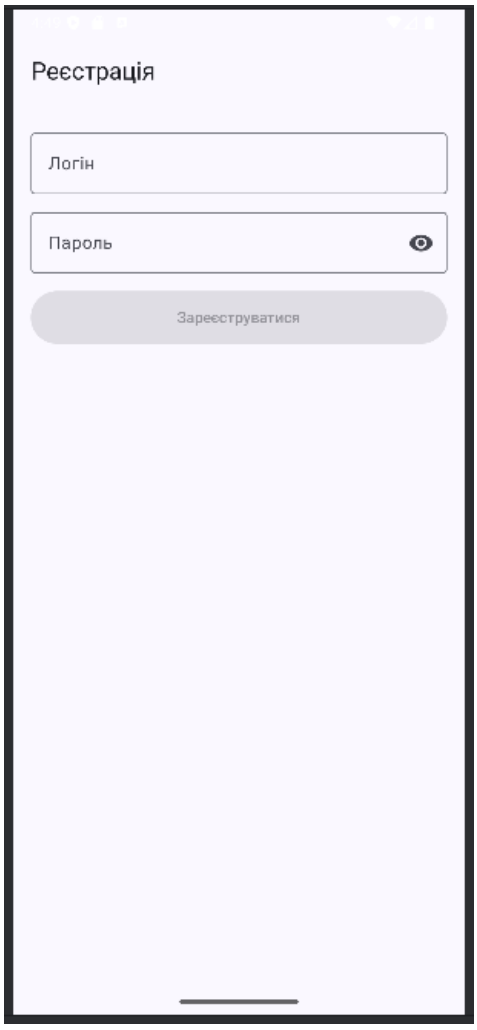

Рисунок 4.1 – Сторінка реєстрації (рисунок виконаний самостійно)

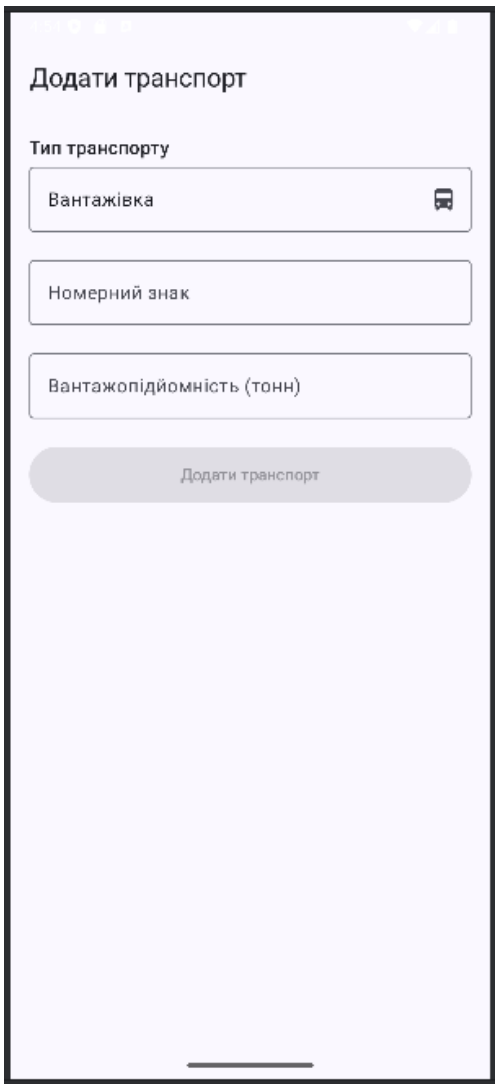

Рисунок 4.2 – Сторінка додавання транспорту (рисунок виконано самостійно)

Сервер та база даних створеної програмної системи розгортаються у Docker-контейнерах. Мобільний застосунок запускається на смартфонах користувачів.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для тестування розробленого програмного забезпечення було використано декілька інструментів та типів тестування.

За допомогою функціонального тестування було перевірено основні сценарії використання розробленої системи:

– реєстрація та авторизація користувачів;

– управління замовленнями на перевезення;

– пошук замовлень та надання пропозицій на їх виконання;

– робота з електронними документами.

Для перевірки правильної роботи створених смарт-контрактів було розгорнуто тестову блокчейн мережу за допомогою Hardhat.

У результаті проведеного тестування підтверджено, що основні сценарії взаємодії користувача з системою було реалізовано правильно.

Для автоматизації тестування розробленого REST API було використано Postman. На рисунку 5.1 зображено тестування кінцевої точки, що надає список наданих пропозицій до певного замовлення на перевезення.

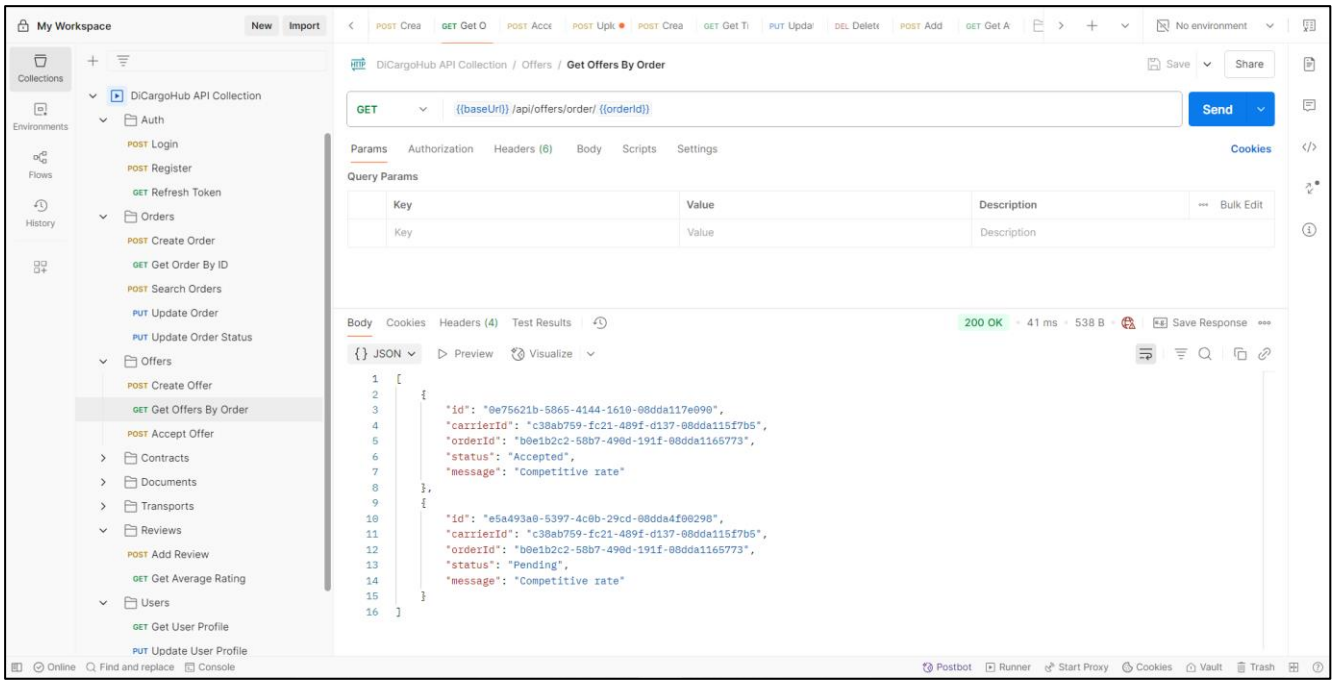

Рисунок 5.1 Тестування REST API за допомогою Postman

У результаті тестування REST API було успішно перевірено розроблені кінцеві точки та правильність обробки запитів серверною частиною.

Також проведено навантажувальне тестування сервера для перевірки продуктивності та стабільності. Для цього було використано програму Apache JMeter. Були змодельовані сценарії одночасної роботи з системою 1000 користувачів та перевірено середній час очікування відповіді від сервера. На рисунку 5.2 зображено результати проведеного тестування.

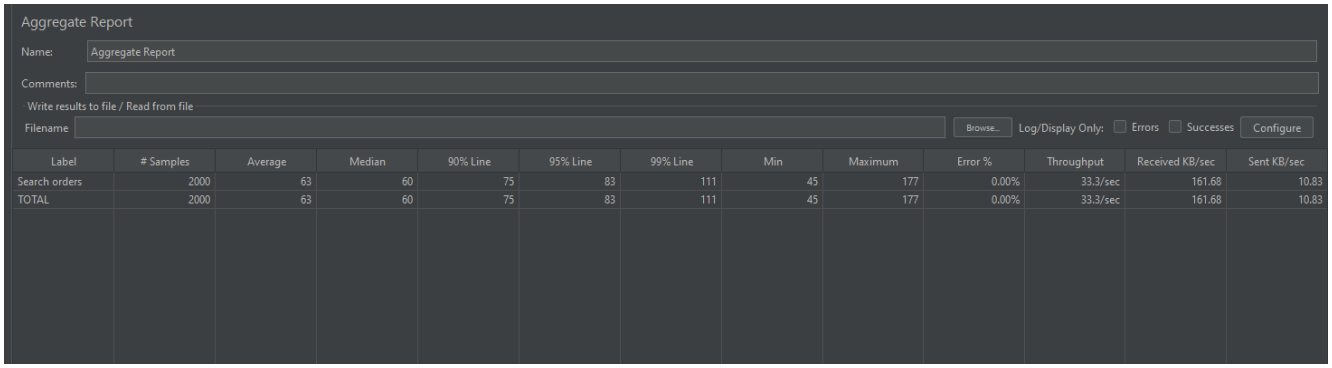

Рисунок 5.2 – Результат проведеного навантажувального тестування сервера

Для проведення навантажувального тестування було обрано кінцеву точку пошуку запитів на перевезення. Вона забезпечує найбільше навантаження на сервер, адже здійснює фільтрацію великої кількості даних за декількома параметрами. З результатів бачимо, що сервер здатен забезпечити стабільну роботу при навантаженні 33.3 запити на секунду. При цьому усі запити були успішно виконані а середній час очікування відповіді склав 63 мс, що задовольняє вимоги до сисеми.

Для тестування бізнес-логіки, яка виконує розрахунки, було написано unit-тести. Для цього було використано бібліотеку xUnit. За допомогою unit-тестування було перевірено правильність розрахунку рейтингу користувача та витрат на пальне. На рисунку 5.3 зображено результати виконання тестів.

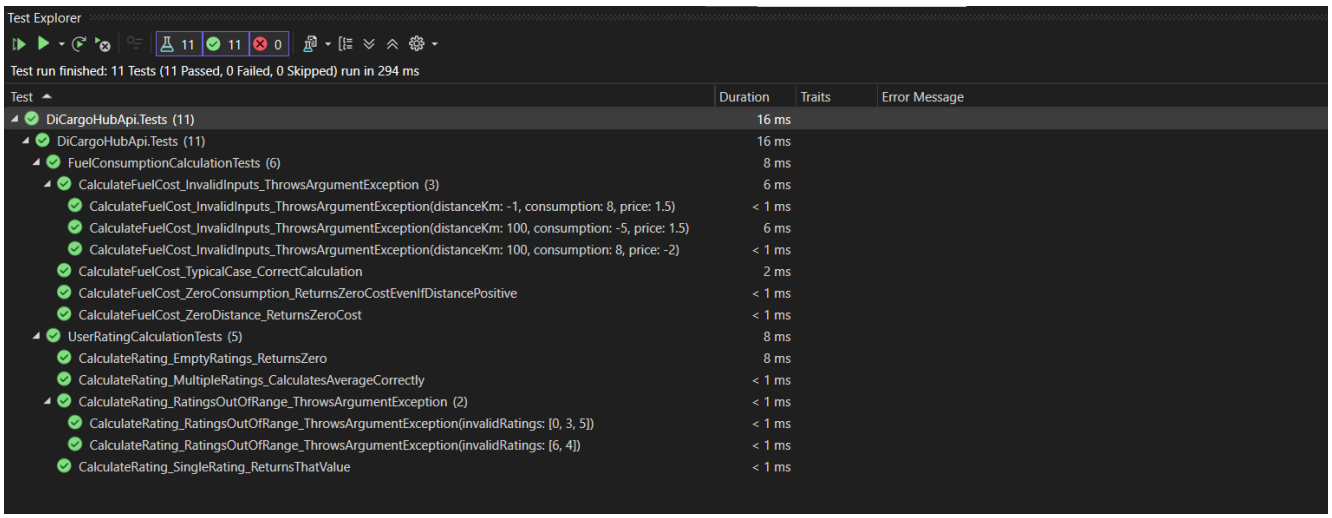

Рисунок 5.3 – Результати виконання unit-тестів

За допомогою unit-тестів було перевірено низку сценаріїв. Для розрахунку рейтингу користувача перевірено такі випадки:

– відсутність оцінок
– наявність однієї оцінки
– наявність декількох оцінок
– наявність оцінки, яка виходить поза межами можливих значень.

Для розрахунку витрат на пальне перевірено наступні випадки:

– нульова відстань;
– нульове споживання палива;
– типовий сценарій;
– від’ємна відстань;
– від’ємне споживання палива.

За результатами виконання unit-тестів можна стверджувати про правильність реалізації бізнес-логіки.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи проведено аналіз предметної галузі, досліджено розвиток перевезень вантажів та зазначено важливість логістики у сучасному світі. Було проаналізовано аналоги, визначено основні виклики та проблеми, які трапляються у сучасних логістичних системах. Використання партнерами різних інформаційних систем для організації роботи, затримки в документообігу, відсутність довіри між учасниками ринку, можливість шахраювання чи підробки документів, які використовуються у сфері логістики, мають негативний вплив на галузь. На основі отриманих даних було сформульовано вимоги до системи в цілому, окремі вимоги до її частин та поставлено задачі.

В рамках роботи було проведено проєктування, розробка та тестування програмної системи для організації вантажних перевезень та врегулювання відносин між замовниками та перевізниками із використанням технології блокчейн, смарт-контрактів та електронного цифрового підпису. Створена система автоматизує ключові процеси взаємодії сторін, підвищує прозорість та безпеку перевезень, а також спрощує документообіг у сфері вантажних перевезень.

У результаті проведеної роботи створено програмну систему, яка відповідає сучасним стандартам безпеки, гнучкості та масштабованості. Отримані результати можуть бути корисними для створення чи вдосконалення платформ для організації вантажних перевезень або у діяльності підприємств, які займаються вантажоперевезеннями. Розроблену систему можна розвивати в подальшому, наприклад, інтегрувати сторонні сервіси, штучний інтелект та різні IoT-пристрої.

Апробація отриманих результатів відбулася у вигляді представлення матеріалів на І Міжнародній науково-технічній конференції «Сучасні інформаційні технології та системи штучного інтелекту» MIT@AIS-2025 (див. дод. Г), що засвідчило актуальність і практичну цінність проведеного дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Стаття «100 Jaw-Dropping Supply Chains & logistics Statistics to know» – URL: https://docshipper.com/logistics/100-supply-chain-logistics-statistics-2/ (Дата звернення 10.05.2025).

2. Г.Ю. Терещенко, Н.В. Шаронова, І.В. Кириченко, Проблеми і Перспективи Практичного Застосування Інформаційної Технології Blockchain в Smart-Контрактах, Інтелектуальні системи та інформаційні технологі (ISIT-2019). – Матеріали Міжн. Наук.-практ. Конф. – Одеса, 19 − 24 серпня 2019 р. – С. 214–219.

3. «Що таке CMR? Для чого потрібна товаро-транспортна накладна CMR (ЦМР)?» – URL: https://trans-atlas.com.ua/ua/article/91 (Дата звернення 03.06.2025).

4. «Know Your Incoterms» – URL: https://www.trade.gov/know-your-incoterms Дата звернення (04.05.2025).

5. «eCMR: know better the dematerialized consignment note» – URL: https://www.dashdoc.com/en/blog/ecmr-definition-advantages Дата звернення (04.05.2025).

6. «Документи для митного оформлення. Транспортний документ – морський коносамент» – URL: https://ua-broker.com/shcho-potribno-znaty/dokumenti-dlya-mitnogo-oformlennya/transportniy-dokument-morskiy-konosament/ Дата звернення (04.05.2025).

7. «Документи для митного оформлення. Транспортний документ – авіанакладна (AWB)» – URL: https://ua-broker.com/shcho-potribno-znaty/dokumenti-dlya-mitnogo-oformlennya/transportniy-dokument-avianakladna-awb/ Дата звернення (04.05.2025).

8. Tereshchenko, Glib, and Iryna Kyrychenko. 2024. “Analysis and Justification of the Use of Existing Blockchain Solutions for the Protection of Digital Assets”. INNOVATIVE TECHNOLOGIES AND SCIENTIFIC SOLUTIONS FOR INDUSTRIES, no. 1 (27) (July):164-78. https://doi.org/10.30837/ITSSI.2024.27.164.

9. Кравченко Є. Р., Кириченко І. В., Терещенко Г. Ю. Використання Технології Блокчейн для Підвищення Прозорості та Безпеки у Сфері Вантажних Перевезень. Сучасні інформаційні технології та системи штучного інтелекту:

матеріали 1ї Міжнародної науково-практичної конференції. Частина 1. : Міжнар. наук. конф., м. Харків - Яремче, 19–22 трав. 2025 р. Харків, 2025. С. 66–69.

10. «Cargo Release» by GSBN – URL: https://www.gsbn.trade/cargo-release (Дата звернення 04.06.2025).

11. «About GSBN» – URL: https://www.gsbn.trade/about-gsbn Дата звернення (05.06.2025).

12. Стаття «Cargo theft spiked over 57% in 2023 vs. 2022, new data shows» - URL: https://www.cnbc.com/2024/01/22/cargo-theft-up-57percent-in-2023-vs-2022-new-cargonet-data-shows.html (Дата звернення 05.06.2025).

13. Kanjilal J. ASP.NET Web API: Build RESTful web applications and services on the .NET framework. Packt Publishing, 2013. 224 с.

14. Android Programming: The Big Nerd Ranch Guide / C. Stewart et al. Pearson Education, Limited, 2022. 640 с.

15. Стаття «ECDSA vs RSA. What and why?» – URL: https://tejnaren07.medium.com/ecdsa-vs-rsa-what-and-why-e83c4c3b501a (Дата звернення 13.05.2025).

16. Martin R. C. Agile Software Development, Principles, Patterns, and Practices. 2-ге вид. Prentice Hall, 2002. 529 с.

17. Посилання на власний GitHub репозиторій – URL: https://github.com/NureKravchenkoYevhenii/2025_B_PI_PZPI-21-5_Kravchenko_Y_R (Дата звернення 05.06.2025).