ДОДАТОК А

Слайди презентації

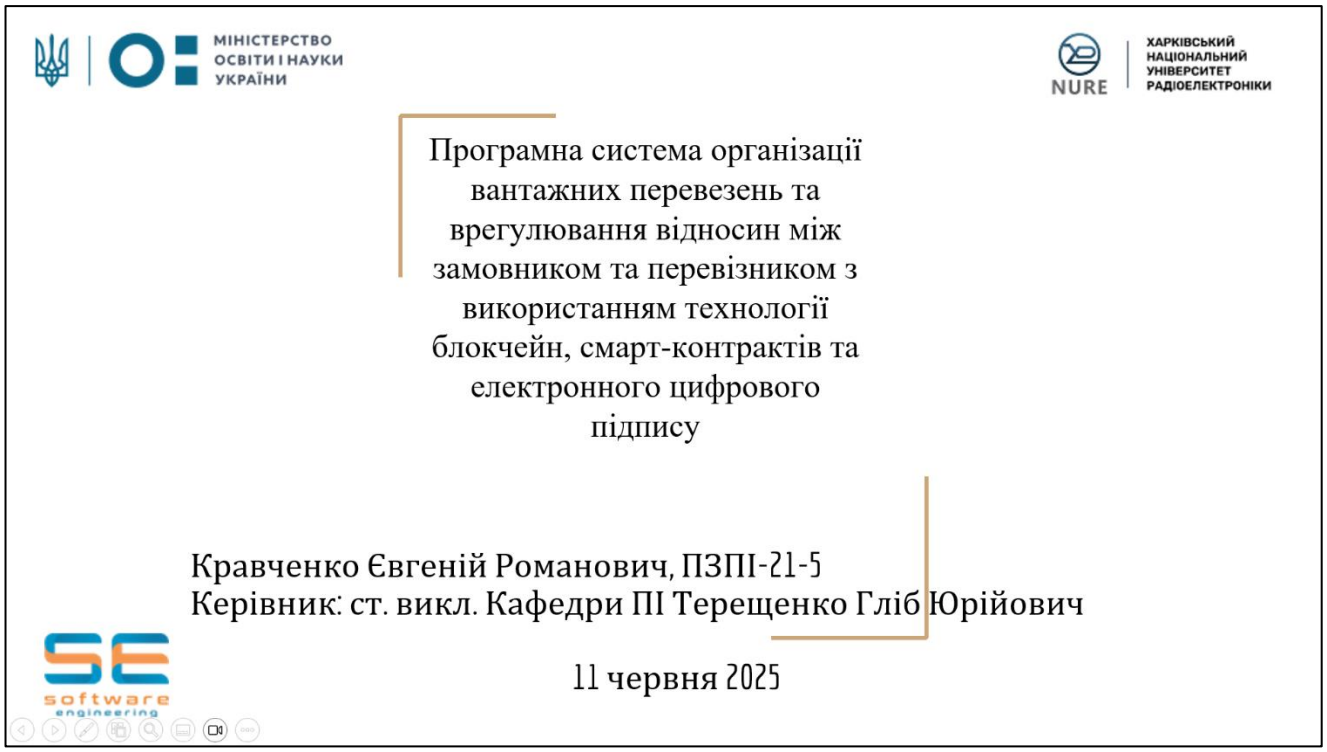


Рисунок А.1 – Слайд № 1

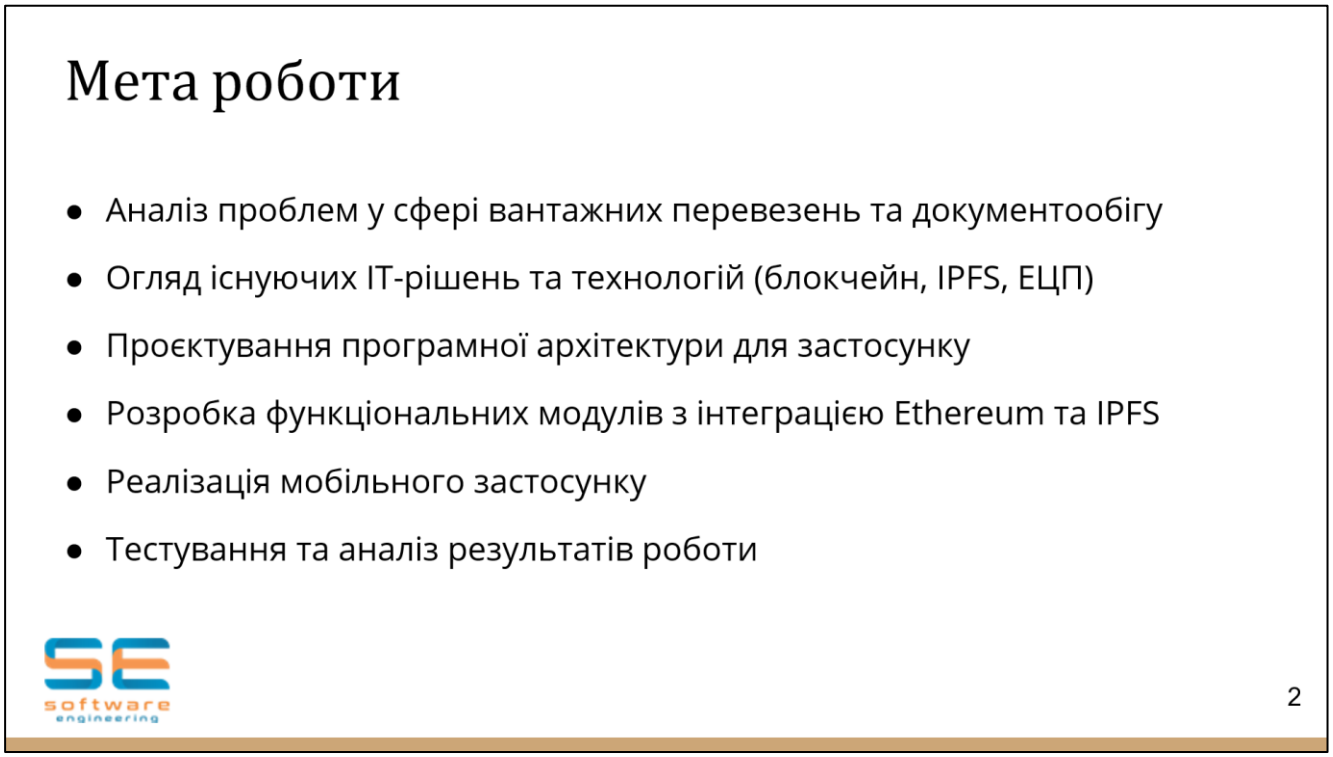


Рисунок А.1 – Слайд № 2

Аналіз проблеми (аналіз існуючих рішень)

Cargo Release by GSBN

- орієнтоване на морські перевезення
- обмежене у використанні для невеликих компаній та приватних підприємців

3

Рисунок А.3 – Слайд № 3

Постановка задачі та опис системи

Проблеми

- Недостатня прозорість
- Складність документообігу та ризики шахрайства
- Паперовий документообіг
- Централізовані рішення можуть піддаватися атакам

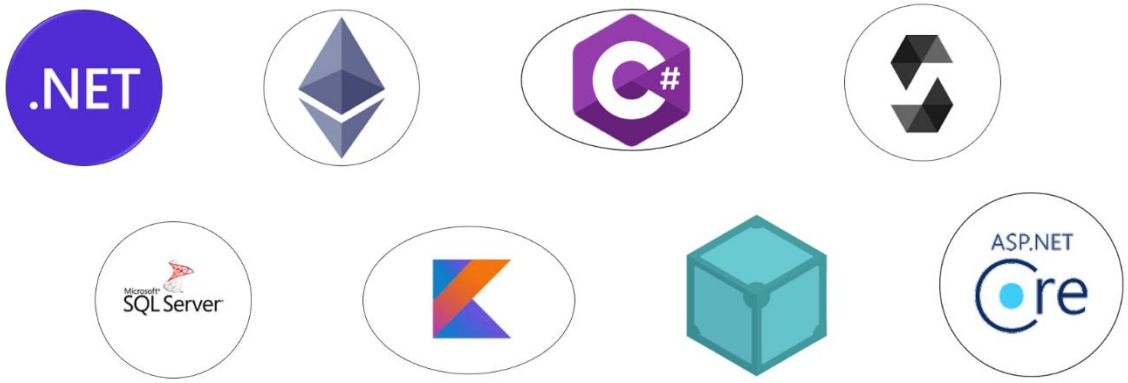
Очікування

- прозора взаємодія між учасниками
- електронний документообіг
- Можливість підписати документи ЕЦП
- Децентралізоване сховище для критичних даних
- Автоматизація укладання та контролю угоди
- Підвищення довіри між учасниками

4

Рисунок А.4 – Слайд № 4

Вибір технологій розробки

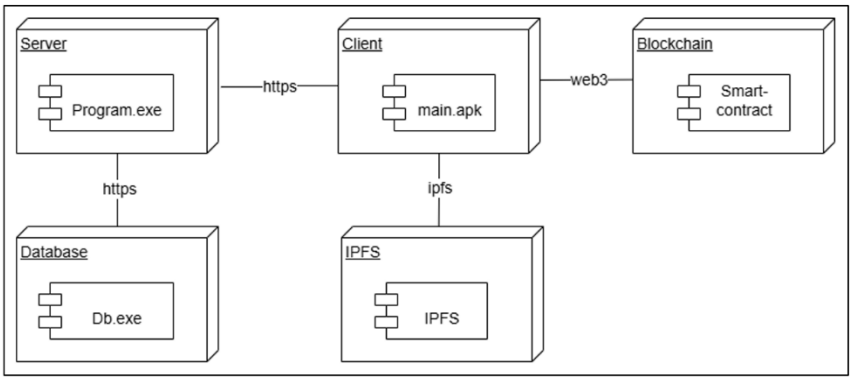


5

Рисунок А.5 – Слайд № 5

Архітектура створенного програмного забезпечення

UML- діаграма розгортання

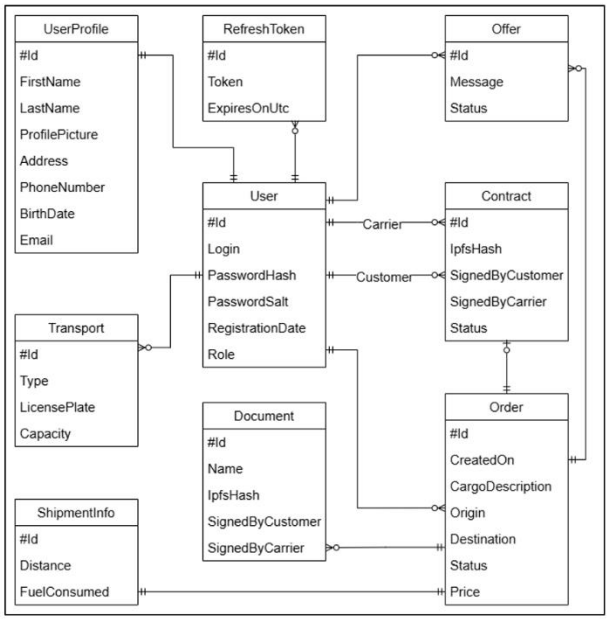


6

Рисунок А.6 – Слайд № 6

Архітектура створенного програмного забезпечення

ER-модель бази даних



7

Рисунок А.7 – Слайд № 7

Процес розробки

- Аналіз предметної галузі
- Постановка задачі та формування вимог
- Проєктування архітектури системи
- Розробка серверної частини
- Розробка клієнтської частини
- Інтеграція блокчейну та IPFS
- Тестування

8

Рисунок А.8 – Слайд № 8

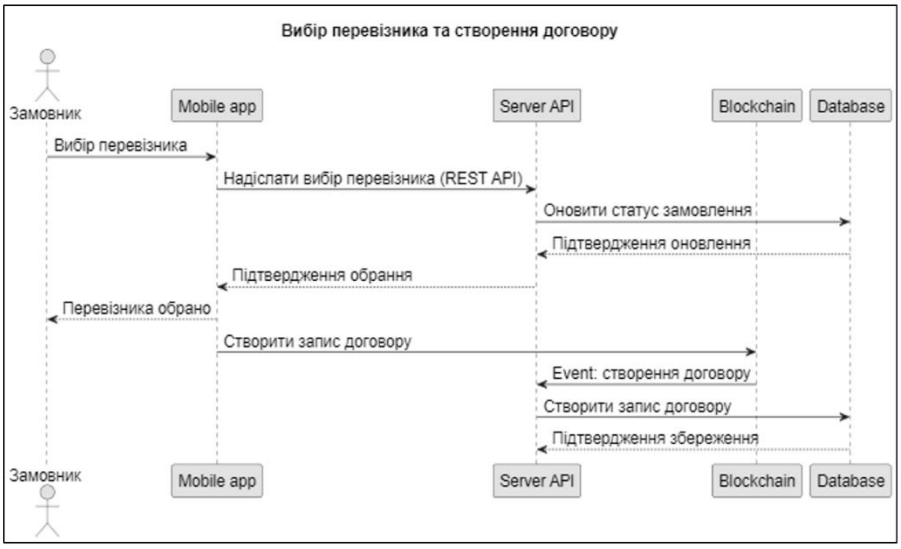
Обрані мови програмування та фреймвоки

- Сервер – C#, .NET 8, ASP.NET Core Web API, EF Core
- База даних – MS SQL Server
- Клієнтська частина – Kotlin, Retrofit, Jetpack Compose, Web3J
- Блокчейн – Ethereum, Solidity
- Файлове сховище - IPFS

9

Рисунок А.9 – Слайд № 9

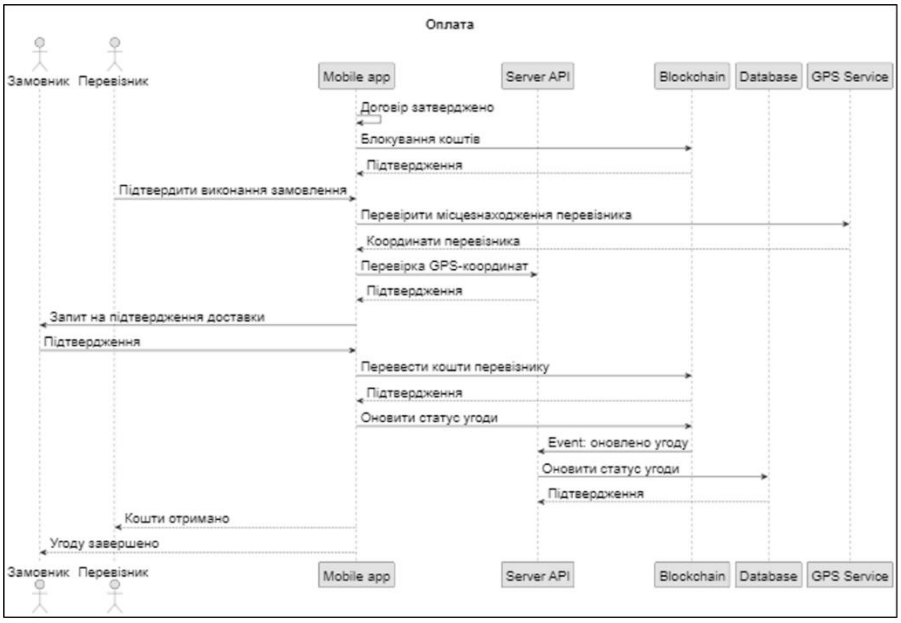
Дизайн системи



10

Рисунок А.10 – Слайд № 10

Дизайн системи



11

Рисунок А.11 – Слайд № 11

Приклад реалізації

```
using DiCargoHubApi.DAL.DbContexts;
using Microsoft.EntityFrameworkCore;

namespace DiCargoHubApi.ApiInfrastructure.Extensions;
0 references
public static class WebApplicationExtensions
{
    1 reference
    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.");
        }
    }
}
```

12

Рисунок А.12 – Слайд № 12

Приклад реалізації

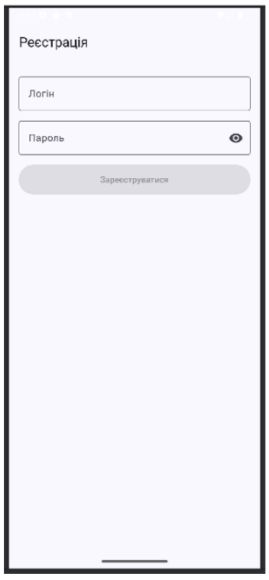
```
namespace DiCargoHubApi.DAL.Contracts;
17 references
public interface IUnitOfWork : IDisposable
{
    1 reference
    void Commit();

    16 references
    Task CommitAsync();

    15 references
    IRepository<TEntity> GetRepository<TEntity>()
        where TEntity : BaseEntity;

    1 reference
    Lazy<IRepository<TEntity>> GetLazyRepository<TEntity>()
        where TEntity : BaseEntity;

    1 reference
    IDbContextTransaction BeginTransaction();

    1 reference
    Task<IDbContextTransaction> BeginTransactionAsync();

    2 references
    Task ExecuteInTransactionAsync(Func<Task> action);

    1 reference
    int ExecuteSqlRaw(string sql, params object[] parameters);
}
```

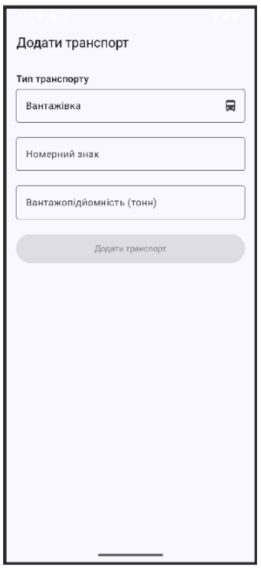
```
public interface IRepository<TEntity>
    where TEntity : BaseEntity
{
    void Add(TEntity entity);

    Task AddAsync(TEntity entity);

    1 reference
    TEntity? GetById(Guid id);

    15 references
    Task<TEntity?> GetByIdAsync(Guid id);

    3 references
    TEntity? Get(Expression<Func<TEntity, bool>> predicate);

    1 reference
    IQueryable<TEntity> GetList(Expression<Func<TEntity, bool>> predicate);

    6 references
    Task<List<TEntity>> GetListAsync(Expression<Func<TEntity, bool>> predicate);

    8 references
    IQueryable<TEntity> GetAll();

    void Update(TEntity entity);

    void UpdateRange(params TEntity[] entities);

    void Remove(TEntity entity);
}
```

13

Рисунок А.13 – Слайд № 13

Інтерфейс користувача





14

Рисунок А.14 – Слайд № 14

Тестування

Підходи

- Модульне тестування
- Інтеграційне тестування
- Функціональне тестування
- Навантажувальне тестування
- Unit-тести

Інструменти

- xUnit
- Postman
- Hardhat
- JMeter
- Android emulator

15

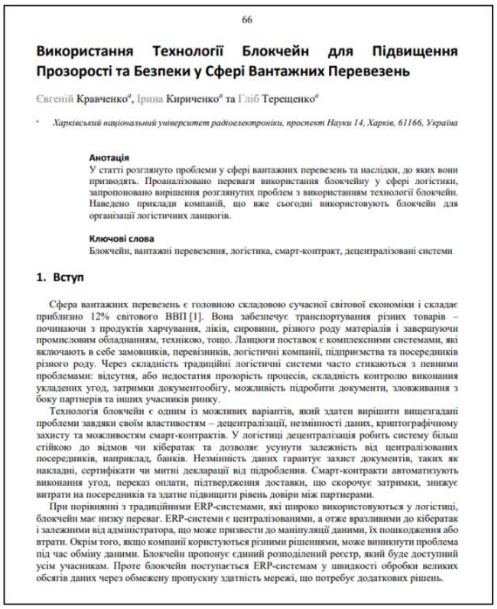
Рисунок А.15 – Слайд № 15

Публікація результатів

16

Рисунок А.16 – Слайд № 16

Публікація результатів



66

Використання Технології Блокчейн для Підвищення Прозорості та Безпеки у Сфері Вантажних Перевезень

Євгеній Кравченко[a], Ірина Кириченко[a] та Гліб Терещенко[a]

[a] *Харківський національний університет радіоелектроніки, проспект Науки 14, Харків, 61166, Україна*

Анотація

У статті розглянуто проблеми у сфері вантажних перевезень та наслідки, до яких вони призводять. Проаналізовано переваги використання блокчейну у сфері логістики, запропоновано вирішення розглянутих проблем з використанням технології блокчейн. Наведено приклади компаній, що вже сьогодні використовують блокчейн для організації логістичних ланцюгів.

Ключові слова

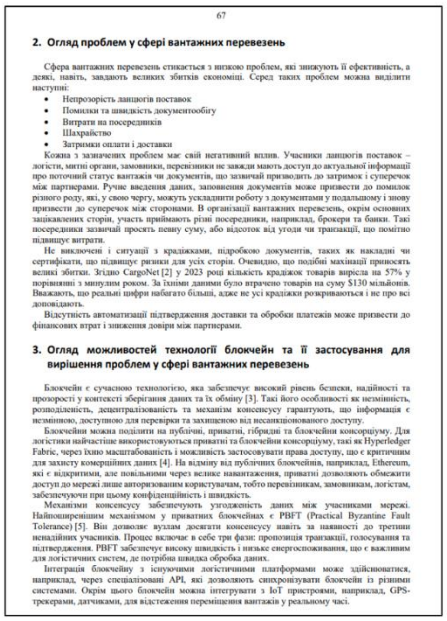
Блокчейн, вантажні перевезення, логістика, смарт-контракт, децентралізовані системи

1. Вступ

Сфера вантажних перевезень є головною складовою сучасної світової економіки і складає приблизно 12% світового ВВП [1]. Вона забезпечує транспортування різних товарів – починаючи з продуктів харчування, ліків, сировини, різного роду матеріалів і завершуючи промисловим обладнанням, технікою, тощо. Ланцюги поставок є комплексними системами, які включають в себе замовників, перевізників, логістичні компанії, підприємства та посередників різного роду. Через складність традиційні логістичні системи часто стикаються з певними проблемами: відсутня, або недостатня прозорість процесів, складність контролю виконання укладених угод, затримки документообігу, можливість підробити документи, зловживання з боку партнерів та інших учасників ринку.

Технологія блокчейн є одним із можливих варіантів, який здатен вирішити вищезгадані проблеми завдяки своїм властивостям – децентралізації, незмінності даних, криптографічному захисту та можливостям смарт-контрактів. У логістиці децентралізація робить систему більш стійкою до відмов чи кібератак та дозволяє усунути залежність від централізованих посередників, наприклад, банків. Незмінність даних гарантує захист документів, таких як накладні, сертифікати чи митні декларації від підроблення. Смарт-контракти автоматизують виконання угод, переказ оплати, підтвердження доставки, що скорочує затримки, знижує витрати на посередників та здатне підвищити рівень довіри між партнерами.

При порівнянні з традиційними ERP-системами, які широко використовуються у логістиці, блокчейн має низку переваг. ERP-системи є централізованими, а отже вразливими до кібератак і залежними від адміністратора, що може призвести до маніпуляції даними, їх пошкодження або втрати. Окрім того, якщо компанії користуються різними рішеннями, може виникнути проблема під час обміну даними. Блокчейн пропонує єдиний розподілений реєстр, який буде доступний усім учасникам. Проте блокчейн поступається ERP-системам у швидкості обробки великих обсягів даних через обмежену пропускну здатність мережі, що потребує додаткових рішень.



67

2. Огляд проблем у сфері вантажних перевезень

Сфера вантажних перевезень стикається з низкою проблем, які знижують її ефективність, а деякі, навіть, завдають великих збитків економіці. Серед таких проблем можна виділити наступні:

- Непрозорість ланцюгів поставок
- Помилки та швидкість документообігу
- Витрати на посередників
- Шахрайство
- Затримки оплати і доставки

Кожна з зазначених проблем має свій негативний вплив. Учасники ланцюгів поставок – логісти, митні органи, замовники, перевізники не завжди мають доступ до актуальної інформації про поточний статус вантажів чи документів, що зазвичай призводить до затримок і суперечок між партнерами. Ручне введення даних, заповнення документів може призвести до помилок різного роду, які, у свою чергу, можуть ускладнити роботу з документами у подальшому і знову призвести до суперечок між сторонами. В організації вантажних перевезень, окрім основних зацікавлених сторін, участь приймають різні посередники, наприклад, брокери та банки. Такі посередники зазвичай просять певну суму, або відсоток від угоди чи транзакції, що помітно підвищує витрати.

Не виключені і ситуації з крадіжками, підробкою документів, таких як накладні чи сертифікати, що підвищує ризики для усіх сторін. Очевидно, що подібні махінації призводять великі збитки. Згідно CargoNet [2] у 2023 році кількість крадіжок товарів виросла на 57% у порівнянні з минулим роком. За їхніми даними було втрачено товарів на суму $130 мільйонів. Вважають, що реальні цифри набагато більші, адже не усі крадіжки розкриваються і не про всі доповідають.

Відсутність автоматизації підтвердження доставки та обробки платежів може призвести до фінансових втрат і зниження довіри між партнерами.

3. Огляд можливостей технології блокчейн та її застосування для вирішення проблем у сфері вантажних перевезень

Блокчейн є сучасною технологією, яка забезпечує високий рівень безпеки, надійності та прозорості у контексті зберігання даних та їх обміну [3]. Такі його особливості як незмінність, розподіленість, децентралізованість та механізм консенсусу гарантують, що інформація є незмінною, доступною для перевірки та захищеною від несанкціонованого доступу.

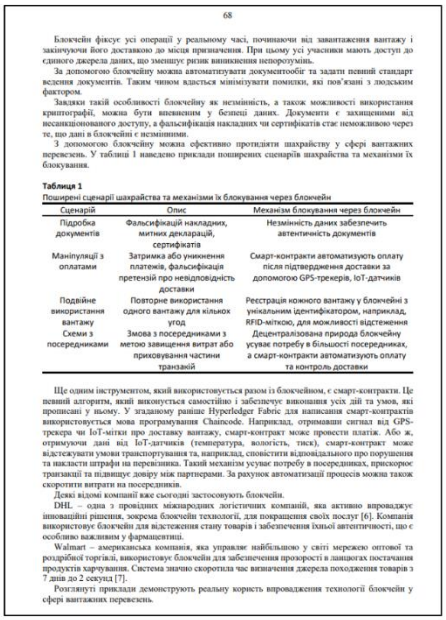
Блокчейни можна поділити на публічні, приватні, гібридні та блокчейни консорціуму. Для логістики найчастіше використовуються приватні та блокчейни консорціуму, такі як Hyperledger Fabric, через їхню масштабованість і можливість застосовувати права доступу, що є критичним для захисту комерційних даних [4]. На відміну від публічних блокчейнів, наприклад, Ethereum, які є відкритими, але повільними через велике навантаження, приватні дозволяють обмежити доступ до мережі лише авторизованим користувачам, тобто перевізникам, замовникам, логістам, забезпечуючи при цьому конфіденційність і швидкість.

Механізми консенсусу забезпечують узгодженість даних між учасниками мережі. Найпоширенішим механізмом у приватних блокчейнах є PBFT (Practical Byzantine Fault Tolerance) [5]. Він дозволяє вузлам досягати консенсусу навіть за наявності до третини ненадійних учасників. Процес включає в себе три фази: пропозиція транзакції, голосування та підтвердження. PBFT забезпечує високу швидкість і низьке енергоспоживання, що є важливим для логістичних систем, де потрібна швидка обробка даних.

Інтеграція блокчейну з існуючими логістичними платформами може здійснюватися, наприклад, через спеціалізовані API, які дозволяють синхронізувати блокчейн із різними системами. Окрім цього блокчейн можна інтегрувати з IoT пристроями, наприклад, GPS-трекерами, датчиками, для відстеження переміщення вантажів у реальному часі.

Рисунок А.17 – Слайд № 17

Публікація результатів



68

Блокчейн фіксує усі операції у реальному часі, починаючи від завантаження вантажу і закінчуючи його доставкою до місця призначення. При цьому усі учасники мають доступ до єдиного джерела даних, що зменшує ризик виникнення непорозумінь.

За допомогою блокчейну можна автоматизувати документообіг та задати певний стандарт ведення документів. Таким чином вдається мінімізувати помилки, які пов'язані з людським фактором.

Завдяки такій особливості блокчейну як незмінність, а також можливості використання криптографії, можна бути впевненим у безпеці даних. Документи є захищеними від несанкціонованого доступу, а фальсифікація накладних чи сертифікатів стає неможливою через те, що дані в блокчейні є незмінними.

З допомогою блокчейну можна ефективно протидіяти шахрайству у сфері вантажних перевезень. У таблиці 1 наведено приклади поширених сценаріїв шахрайства та механізми їх блокування.

Таблиця 1

Поширені сценарії шахрайства та механізми їх блокування через блокчейн

Сценарій	Опис	Механізм блокування через блокчейн
Підробка документів	Фальсифікація накладних, митних декларацій, сертифікатів	Незмінність даних забезпечить автентичність документів
Маніпуляції з оплатами	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
Схеми з посередниками	Змова з посередниками з метою завищення витрат або приховування частини транзакцій	Децентралізована природа блокчейну усуває потребу в більшості посередників, а смарт-контракти автоматизують оплату та контроль доставки

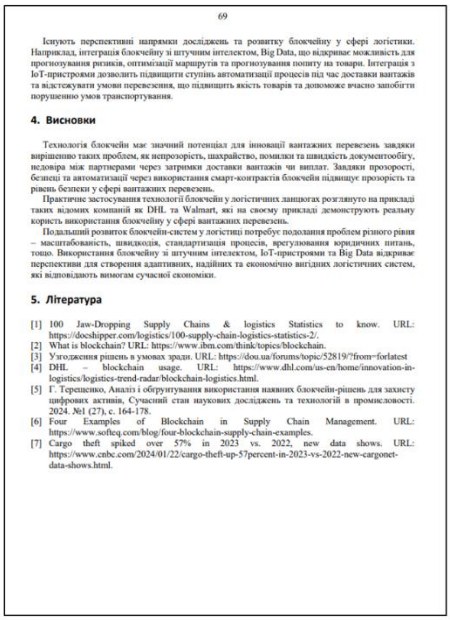
Ще одним інструментом, який використовується разом із блокчейном, є смарт-контракти. Це певний алгоритм, який виконується самостійно і забезпечує виконання усіх дій та умов, які прописані у ньому. У згаданому раніше Hyperledger Fabric для написання смарт-контрактів використовується мова програмування Chaincode. Наприклад, отримавши сигнал від GPS-трекера чи IoT-мітки про доставку вантажу, смарт-контракт може провести платіж. Або ж, отримуючи дані від IoT-датчиків (температура, вологість, тиск), смарт-контракт може відстежувати умови транспортування та, наприклад, сповістити відповідального про порушення та накласти штрафи на перевізника. Такий механізм усуває потребу в посередниках, прискорює транзакції та підвищує довіру між партнерами. За рахунок автоматизації процесів можна також скоротити витрати на посередників.

Деякі відомі компанії вже сьогодні застосовують блокчейн.

DHL – одна з провідних міжнародних логістичних компаній, яка активно впроваджує інноваційні рішення, зокрема блокчейн технології, для покращення своїх послуг [6]. Компанія використовує блокчейн для відстеження стану товарів і забезпечення їхньої автентичності, що є особливо важливим у фармацевтиці.

Walmart – американська компанія, яка управляє найбільшою у світі мережею оптової та роздрібної торгівлі, використовує блокчейн для забезпечення прозорості в ланцюгах постачання продуктів харчування. Система значно скоротила час визначення джерела походження товарів з 7 днів до 2 секунд [7].

Розглянуті приклади демонструють реальну користь впровадження технології блокчейн у сфері вантажних перевезень.



69

Існують перспективні напрямки досліджень та розвитку блокчейну у сфері логістики. Наприклад, інтеграція блокчейну зі штучним інтелектом, Big Data, що відкриває можливість для прогнозування ризиків, оптимізації маршрутів та прогнозування попиту на товари. Інтеграція з IoT-пристроями дозволить підвищити ступінь автоматизації процесів під час доставки вантажів та відстежувати умови перевезення, що підвищить якість товарів та допоможе вчасно запобігти порушенню умов транспортування.

4. Висновки

Технологія блокчейн має значний потенціал для інновацій вантажних перевезень завдяки вирішенню таких проблем, як непрозорість, шахрайство, помилки та швидкість документообігу, недовіра між партнерами через затримки доставки вантажів чи виплат. Завдяки прозорості, безпеці та автоматизації через використання смарт-контрактів блокчейн підвищує прозорість та рівень безпеки у сфері вантажних перевезень.

Практичне застосування технології блокчейн у логістичних ланцюгах розглянуто на прикладі таких відомих компаній як DHL та Walmart, які на своєму прикладі демонструють реальну користь використання блокчейну у сфері вантажних перевезень.

Подальший розвиток блокчейн-систем у логістиці потребує подолання проблем різного рівня – масштабованість, швидкодія, стандартизація процесів, врегулювання юридичних питань, тощо. Використання блокчейну зі штучним інтелектом, IoT-пристроями та Big Data відкриває перспективи для створення адаптивних, надійних та економічно вигідних логістичних систем, які відповідають вимогам сучасної економіки.

5. Література

[1] 100 Jaw-Dropping Supply Chains & logistics Statistics to know. URL: https://docshipper.com/logistics/100-supply-chain-logistics-statistics-2/.
[2] What is blockchain? URL: https://www.ibm.com/think/topics/blockchain.
[3] Угоджування рішень в умовах зради. URL: https://dou.ua/forums/topic/52819/?from=forlatest
[4] DHL – blockchain usage. URL: https://www.dhl.com/us-en/home/innovation-in-logistics/logistics-trend-radar/blockchain-logistics.html.
[5] Г. Терещенко, Аналіз і обґрунтування використання наявних блокчейн-рішень для захисту цифрових активів, Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27), с. 164-178.
[6] Four Examples of Blockchain in Supply Chain Management. URL: https://www.softeq.com/blog/four-blockchain-supply-chain-examples.
[7] 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.

Рисунок А.18 – Слайд № 18

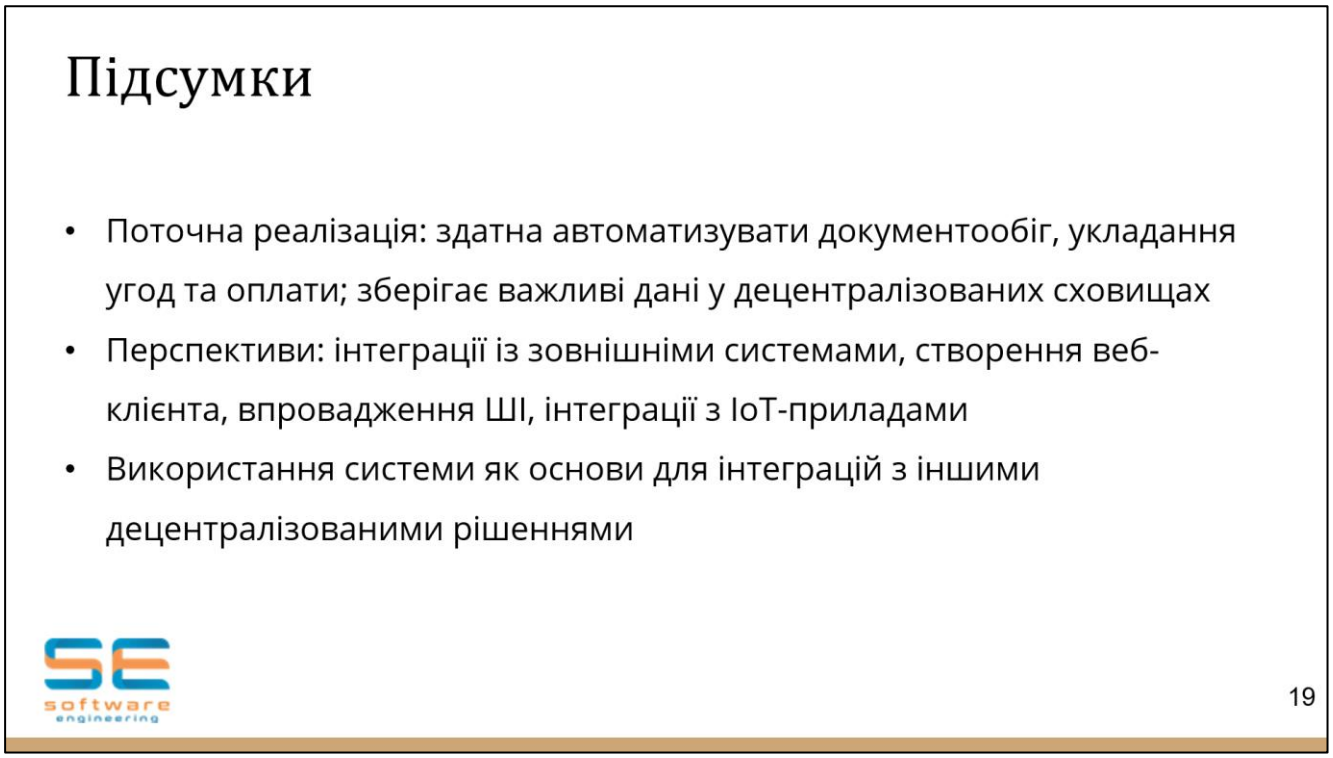


Рисунок А.19 – Слайд № 19

ДОДАТОК Б

Специфікація програмного продукту

1 ВСТУП

1.1 Огляд продукту

Сьогодні сфера вантажних перевезень стикається з деякими проблемами. Середи них проблеми з довірою між партнерами, безпека документообігу, можливі шахрайства. Рішенням даних проблем може стати програмна система, що здатна забезпечити прозору взаємодію між замовником і перевізником шляхом фіксації угод через блокчейн, збереження документів у IPFS та їх підписання за допомогою електронного цифрового підпису.

1.2 Мета

Метою є 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.

1.3 Межі

Система охоплює реєстрацію та авторизацію користувачів, інструменти управління та пошуку замовлень, підписання документів за допомогою ЕЦП, інтеграцію із блокчейном та систему відгуків та рейтингів.

Система не має iOS-застосунку чи веб-застосунку, не має модуля бухгалтерії, аналітики, інтеграцій з ERP.

1.4 Короткий опис

Проєкт включає у себе мобільний застосунок для ОС Android, серверну частину, елементи взаємодії з блокчейном та IPFS і підтримку ЕЦП.

1.5 Означення та абревіатури

- ЕЦП – електронний цифровий підпис;
- IPFS – InterPlanetary File System, розподілене файлове сховище;
- Ethereum – платформа для створення децентралізованих застосунків і підтримкою смарт-контрактів;
- Смарт-контракт – програмний код, що виконується;
- ПЗ – програмне забезпечення;
- API – Application Programming Interface, або ж інтерфейс програмування додатків; спосіб взаємодії програмних засобів між собою;
- ERP – Enterprise Resource Planning; система, що інтегрується та управляє ключовими бізнес процесами компанії; наприклад, фінансами, виробництвом, продажами, тощо;
- ОС – операційна система.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Програмна система має на меті стати платформою для взаємодії замовників і перевізників. При цьому вона повинна підвищити прозорість операцій, автоматизувати укладання угод, проведення оплати і зменшити ризик шахрайства.

2.2 Функції продукту

До основних функцій продукту відносяться наступні:

- 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), підписання документів електронним цифровим підписом.

2.3 Характеристики користувачів

Основними користувачами системи виступають замовники та перевізники. Очікується, що обидва користувача мають базові навички користування мобільним телефоном. Вони уміють розбиратися з мобільним інтерфейсом та розуміють, яку

інформацію необхідно заповнювати у відповідні поля.

2.4 Загальні обмеження

Мобільний застосунок працює на пристроях з версією Android 6.0 (API 23) і вище. Сервер функціонує на платформі ASP.NET Core 8.0. У якості бази даних використовується СУБД Microsoft SQL Server 2022. У якості блокчейну виступає Ethereum. Взаємодія з IPFS та блокчейном відбувається зі сторони клієнта. Інформація, що записується на блокчейн, також записується у базу даних для того, щоб пришвидшити отримання даних під час, наприклад, пошуку.

2.5 Припущення й залежності

Усі учасники обов’язково мають доступ до мережі Інтернет. І замовник, і перевізник мають електронний цифровий підпис. Сторонні сервіси, тобто GPS та блокчейн, доступні та мають стабільний зв’язок.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Інтерфейс користувача реалізовано у вигляді мобільного застосунку. Для реалізації можна використовувати Jetpack Compose. Мобільний застосунок надає доступ до основного функціоналу системи. Інтерфейс змінюється відповідно до того, ким є користувач (замовником чи перевізником). Дизайн повинен бути мінімалістичний та зручний.

3.1.2 Апаратний інтерфейс

Мобільний застосунок підтримується на пристроях з версією Android 6.0 та вище, тобто версією API 23 та вище. Сервер розгортається на серверах із підтримкою .NET 8. IPFS використовується як розподілене сховище документів.

3.1.3 Програмний інтерфейс

Взаємодія між клієнтською та серверною частинами відбувається через RESTful API. Запити оброблюються у форматі JSON. Сервер надає кінцеві точки для реєстрації, аутентифікації, роботи із замовленнями, пропозиціями на перевезення та рейтингами.

3.1.4 Комунікаційний протокол

Обмін даними між сервером та клієнтом відбувається через протокол HTTPS. Для взаємодії з блокчейном та смарт-контрактами використовується Web3 бібліотека. Клієнт взаємодіє з IPFS напряму через публічні шлюзи чи локальний вузол.

3.1.5 Обмеження пам’яті

Мобільний застосунок працює стабільно на пристроях із мінімум 2 гігабайтами оперативної пам’яті. Сервер здатен забезпечити обробку до 500 сесій одночасно. Документи, які зберігаються у IPFS, не перевищують розмір 10

мегабайт, щоб не навантажувати мережу.

3.1.6 Операції

Мобільний застосунок підтримує пуш-повідомлення. GPS координати перевізника можна отримати у реальному часі.

3.1.7 Функції продукту

До основних функцій продукту відносяться наступні:

- 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), підписання документів електронним цифровим підписом.

3.1.8 Припущення і залежності

Усі учасники обов’язково мають доступ до мережі Інтернет. І замовник, і перевізник мають електронний цифровий підпис. Сторонні сервіси, тобто GPS та блокчейн, доступні та мають стабільний зв’язок.

3.2 Атрибути програмного продукту

3.2.1 Надійність

Система працює стабільно при звичайному навантаження. Передбачено механізм обробки помилок.

3.2.2 Доступність

Мобільний застосунок доступний користувачам у будь-який час. Сервер доступний 99% часу

3.2.3 Безпека

Доступ до програмної системи надається лишу авторизованим користувачам. Використовувати JWT токени для авторизації. Використовувати ЕЦП для підписання документів. Для з’єднань використовувати протокол HTTPS. Користувача надавати доступ до функціоналу за ролями.

3.2.4 Супроводжуваність

Під час проєктування програмної системи врахувати потребу внесення змін та додавання нового функціоналу у майбутньому. Забезпечити можливість легко вносити зміни та розширювати систему.

3.2.5 Переносимість

Серверна частина здатна функціонувати на серверах з ОС Windows та Linux, або бути запущеною у Docker контейнері. Мобільний застосунок може функціонувати на пристроях з ОС Android.

3.2.6 Продуктивність

Система повинна витримувати навантаження 1000 користувачів одночасно. Середній час очікування відповіді від сервера не перевищує 1 секунди.

3.3 Вимоги до бази даних

Для збереження основних даних використовувати реляційну СУБД (Microsoft SQL Server 2022). Важливу інформацію зберігати на блокчейні для забезпечення її цілісності. Ці дані також зберігати у базу даних для швидкого отримання під час виконання пошуку. Документи зберігати у децентралізованій файловій системі.

ДОДАТОК В

Фрагменти коду програмної реалізації

В.1 Фрагмент коду реалізації «UnitOfWork»

```
public class UnitOfWork : IUnitOfWork
{
    private readonly DiCargoHubApiDbContext _dbContext;
    private Dictionary<Type, object> _repositories;
    private bool _disposed = false;

    public UnitOfWork(DiCargoHubApiDbContext dbContext)
    {
        _dbContext = dbContext;
        _repositories = [];
    }

    public virtual IDbContextTransaction BeginTransaction()
    {
        return _dbContext.Database.BeginTransaction();
    }

    public virtual async Task<IDbContextTransaction>
BeginTransactionAsync()
    {
        return await _dbContext.Database.BeginTransactionAsync();
    }

    public virtual async Task ExecuteInTransactionAsync(Func<Task>
action)
    {
        await using var tx = await
_dbContext.Database.BeginTransactionAsync();
        try
        {
            await action();
            await _dbContext.SaveChangesAsync();
            await tx.CommitAsync();
        }
        catch
        {
            await tx.RollbackAsync();
            throw;
        }
    }

    public virtual void Commit()
    {
        _dbContext.SaveChanges();
    }

    public async virtual Task CommitAsync()
    {
        await _dbContext.SaveChangesAsync();
```

```
        }

        public virtual void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool disposing)
        {
            if (!_disposed)
            {
                if (disposing)
                {
                    _dbContext.Dispose();
                }

                _disposed = true;
            }
        }

        public virtual Lazy<IRepository<TEntity>>
GetLazyRepository<TEntity>()
            where TEntity : BaseEntity
        {
            return new
Lazy<IRepository<TEntity>>(GetRepository<TEntity>);
        }

        public virtual IRepository<TEntity> GetRepository<TEntity>()
            where TEntity : BaseEntity
        {
            if (_repositories.ContainsKey(typeof(TEntity)))
            {
                return
(IRepository<TEntity>)_repositories[typeof(TEntity)];
            }

            var repository = new Repository<TEntity>(_dbContext);
            _repositories.Add(typeof(TEntity), repository);

            return repository;
        }

        public virtual int ExecuteSqlRaw(string sql, params object[]
parameters)
        {
            return _dbContext.Database.ExecuteSqlRaw(sql, parameters);
        }
    }
```

В.2 Фрагменти коду реалізації глобального обробника помилок

```
    public static class ErrorHandler
    {
```

```
        private static readonly JsonSerializerOptions
_jsonSerializerOptions =
            new()
            {
                PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
                DefaultIgnoreCondition =
JsonIgnoreCondition.WhenWritingNull,
            };

        public static IApplicationBuilder UseErrorHandler(
            this IApplicationBuilder applicationBuilder)
        {
            return applicationBuilder.Use(HandleError);
        }

        private static async Task HandleError(
            HttpContext httpContext,
            Func<Task> next)
        {
            try
            {
                await next();
            }
            catch (Exception ex)
            {
                string message;
                object data;

                switch (ex)
                {
                    case UnauthorizedAccessException:
                        httpContext.Response.StatusCode =
StatusCodes.Status401Unauthorized;
                        message = ex.Message;
                        data = ex.Data;
                        break;
                    case Exception:
                        httpContext.Response.StatusCode =
StatusCodes.Status400BadRequest;
                        message = ex.Message;
                        data = ex.Data;
                        break;
                    default:
                        httpContext.Response.StatusCode =
StatusCodes.Status400BadRequest;
                        message = ex.Message;
                        data = ex.ToString();
                        break;
                }

                var error = new
                {
                    Message = message,
                    Error = data
                };

                var errorJson = JsonSerializer.Serialize(
                    error,
```

```
                    _jsonSerializerOptions);

                await httpContext.Response.WriteAsync(errorJson);
            }
        }
    }
```

В.3 Фрагменти коду реалізації смарт-контракту «ContractRegistry»

```
    // SPDX-License-Identifier: MIT
    pragma solidity ^0.8.0;

    contract ContractRegistry {
        enum ContractStatus { Pending, Active, Completed, Cancelled }

        struct ContractDetail {
            address customer;
            address carrier;
            uint256 orderId;
            ContractStatus status;
            bool signedByCustomer;
            bool signedByCarrier;
            string[] documentHashes;
        }

        uint256 private _nextContractId = 1;
        mapping(uint256 => ContractDetail) private _contracts;

        event ContractCreated(
            uint256 indexed contractId,
            address indexed customer,
            address indexed carrier,
            uint256 orderId
        );

        event ContractSigned(
            uint256 indexed contractId,
            address indexed signer
        );

        event DocumentMetadataAdded(
            uint256 indexed contractId,
            string ipfsHash
        );

        event ContractCompleted(
            uint256 indexed contractId
        );

        function createContract(address _carrier, uint256 _orderId)
external returns (uint256) {
            require(_carrier != address(0), "Carrier address cannot be
zero");
            require(_carrier != msg.sender, "Carrier and Customer cannot
be same");
```

```
            uint256 newId = _nextContractId++;
            ContractDetail storage detail = _contracts[newId];

            detail.customer = msg.sender;
            detail.carrier = _carrier;
            detail.orderId = _orderId;
            detail.status = ContractStatus.Pending;
            detail.signedByCustomer = false;
            detail.signedByCarrier = false;

            emit ContractCreated(newId, msg.sender, _carrier, _orderId);
            return newId;
        }

        function signContract(uint256 _contractId) external {
            ContractDetail storage detail = _contracts[_contractId];
            require(detail.customer != address(0), "Contract does not
exist");
            require(detail.status == ContractStatus.Pending, "Contract
not pending");

            if (msg.sender == detail.customer) {
                require(!detail.signedByCustomer, "Customer already
signed");
                detail.signedByCustomer = true;
            } else if (msg.sender == detail.carrier) {
                require(!detail.signedByCarrier, "Carrier already
signed");
                detail.signedByCarrier = true;
            } else {
                revert("Only Customer or Carrier can sign");
            }

            emit ContractSigned(_contractId, msg.sender);

            if (detail.signedByCustomer && detail.signedByCarrier) {
                detail.status = ContractStatus.Active;
            }
        }

        function addDocumentMetadata(uint256 _contractId, string calldata
_ipfsHash) external {
            ContractDetail storage detail = _contracts[_contractId];
            require(detail.customer != address(0), "Contract does not
exist");
            require(
                msg.sender == detail.customer || msg.sender ==
detail.carrier,
                "Only participants can add document metadata"
            );
            require(bytes(_ipfsHash).length > 0, "IPFS hash cannot be
empty");

            detail.documentHashes.push(_ipfsHash);
            emit DocumentMetadataAdded(_contractId, _ipfsHash);
        }
```

```
        function completeContract(uint256 _contractId) external {
            ContractDetail storage detail = _contracts[_contractId];
            require(detail.customer != address(0), "Contract does not
exist");
            require(detail.status == ContractStatus.Active, "Contract not
active");
            require(msg.sender == detail.carrier, "Only carrier can
complete");

            detail.status = ContractStatus.Completed;
            emit ContractCompleted(_contractId);
        }

        function getContract(uint256 _contractId)
            external
            view
            returns (
                address customer,
                address carrier,
                uint256 orderId,
                ContractStatus status,
                bool signedByCustomer,
                bool signedByCarrier,
                string[] memory documentHashes
            )
        {
            ContractDetail storage detail = _contracts[_contractId];
            require(detail.customer != address(0), "Contract does not
exist");

            return (
                detail.customer,
                detail.carrier,
                detail.orderId,
                detail.status,
                detail.signedByCustomer,
                detail.signedByCarrier,
                detail.documentHashes
            );
        }

        function isContractActive(uint256 _contractId) external view
returns (bool) {
            ContractDetail storage detail = _contracts[_contractId];
            require(detail.customer != address(0), "Contract does not
exist");
            return (detail.status == ContractStatus.Active);
        }

        function getParticipants(uint256 _contractId) external view
returns (address, address) {
            ContractDetail storage detail = _contracts[_contractId];
            require(detail.customer != address(0), "Contract does not
exist");
            return (detail.customer, detail.carrier);
        }
    }
```

В.4 Фрагменти коду реалізації смарт-контракту «PaymentEscrow»

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

interface IContractRegistry {
    function isContractActive(uint256 contractId) external view
returns (bool);
    function getParticipants(uint256 contractId) external view
returns (address customer, address carrier);
}

contract PaymentEscrow {
    IContractRegistry public registry;

    struct Escrow {
        uint256 amount;
        bool    deposited;
        bool    released;
        bool    refunded;
    }

    mapping(uint256 => Escrow) public escrows;

    event FundsDeposited(uint256 indexed contractId, address indexed
customer, uint256 amount);
    event FundsReleased(uint256 indexed contractId, address indexed
carrier, uint256 amount);
    event FundsRefunded(uint256 indexed contractId, address indexed
customer, uint256 amount);

    constructor(address _registry) {
        require(_registry != address(0), "Registry address cannot be
zero");
        registry = IContractRegistry(_registry);
    }

    function depositFunds(uint256 _contractId) external payable {
        require(msg.value > 0, "Must deposit > 0");
        require(!escrows[_contractId].deposited, "Funds already
deposited");

        bool active = registry.isContractActive(_contractId);
        require(active, "Contract is not active");

        escrows[_contractId] = Escrow({
            amount: msg.value,
            deposited: true,
            released: false,
            refunded: false
        });

        (address customer, ) = registry.getParticipants(_contractId);
        emit FundsDeposited(_contractId, customer, msg.value);
    }
```

```
        function releaseFunds(uint256 _contractId) external {
            Escrow storage e = escrows[_contractId];
            require(e.deposited, "No funds deposited");
            require(!e.released, "Funds already released");
            require(!e.refunded, "Funds already refunded");

            bool active = registry.isContractActive(_contractId);
            require(!active, "Contract not yet completed or still
active");

            (, address carrier) = registry.getParticipants(_contractId);
            require(msg.sender == carrier, "Only carrier can release
funds");

            uint256 amountToPay = e.amount;
            e.released = true;
            (bool sent, ) = carrier.call{ value: amountToPay }("");
            require(sent, "Transfer to carrier failed");

            emit FundsReleased(_contractId, carrier, amountToPay);
        }

        function refundFunds(uint256 _contractId) external {
            Escrow storage e = escrows[_contractId];
            require(e.deposited, "No funds deposited");
            require(!e.released, "Funds already released");
            require(!e.refunded, "Funds already refunded");

            bool active = registry.isContractActive(_contractId);
            require(!active, "Contract still active");

            (address customer, ) = registry.getParticipants(_contractId);
            require(msg.sender == customer, "Only customer can refund");

            uint256 amountToRefund = e.amount;
            e.refunded = true;
            (bool sent, ) = customer.call{ value: amountToRefund }("");
            require(sent, "Refund to customer failed");

            emit FundsRefunded(_contractId, customer, amountToRefund);
        }

        function getEscrowInfo(uint256 _contractId)
            external
            view
            returns (
                uint256 amount,
                bool deposited,
                bool released,
                bool refunded
            )
        {
            Escrow storage e = escrows[_contractId];
            return (e.amount, e.deposited, e.released, e.refunded);
        }
    }
```

ДОДАТОК Г

Тези І Міжнародної науково-технічної конференції «Сучасні інформаційні технології та системи штучного інтелекту» MIT@AIS-2025

Використання Технології Блокчейн для Підвищення Прозорості та Безпеки у Сфері Вантажних Перевезень

Євгеній Кравченко[a], Ірина Кириченко[a] та Гліб Терещенко[a]

[a] *Харківський національний університет радіоелектроніки, проспект Науки 14, Харків, 61166, Україна*

Анотація

У статті розглянуто проблеми у сфері вантажних перевезень та наслідки, до яких вони призводять. Проаналізовано переваги використання блокчейну у сфері логістики, запропоновано вирішення розглянутих проблем з використанням технології блокчейн. Наведено приклади компаній, що вже сьогодні використовують блокчейн для організації логістичних ланцюгів.

Ключові слова

Блокчейн, вантажні перевезення, логістика, смарт-контракт, децентралізовані системи

1. Вступ

Сфера вантажних перевезень є головною складовою сучасної світової економіки і складає приблизно 12% світового ВВП [1]. Вона забезпечує транспортування різних товарів – починаючи з продуктів харчування, ліків, сировини, різного роду матеріалів і завершуючи промисловим обладнанням, технікою, тощо. Ланцюги поставок є комплексними системами, які включають в себе замовників, перевізників, логістичні компанії, підприємства та посередників різного роду. Через складність традиційні логістичні системи часто стикаються з певними проблемами: відсутня, або недостатня прозорість процесів, складність контролю виконання укладених угод, затримки документообігу, можливість підробити документи, зловживання з боку партнерів та інших учасників ринку.

Технологія блокчейн є одним із можливих варіантів, який здатен вирішити вищезгадані проблеми завдяки своїм властивостям – децентралізації, незмінності даних, криптографічному захисту та можливостям смарт-контрактів. У логістиці децентралізація робить систему більш стійкою до відмов чи кібератак та дозволяє усунути залежність від централізованих посередників, наприклад, банків. Незмінність даних гарантує захист документів, таких як накладні, сертифікати чи митні декларації від підроблення. Смарт-контракти автоматизують виконання угод, переказ оплати, підтвердження доставки, що скорочує затримки, знижує витрати на посередників та здатне підвищити рівень довіри між партнерами.

При порівнянні з традиційними ERP-системами, які широко використовуються у логістиці, блокчейн має низку переваг. ERP-системи є централізованими, а отже вразливими до кібератак і залежними від адміністратора, що може призвести до маніпуляції даними, їх пошкодження або втрати. Окрім того, якщо компанії користуються різними рішеннями, може виникнути проблема під час обміну даними. Блокчейн пропонує єдиний розподілений реєстр, який буде доступний усім учасникам. Проте блокчейн поступається ERP-системам у швидкості обробки великих обсягів даних через обмежену пропускну здатність мережі, що потребує додаткових рішень.

MIT@AIS'2025s: 1st International Scientific and Practical Conference "Modern Information Technologies and Artificial Intelligence Systems", May 19–22, 2025, Kharkiv-Yaremche, Ukraine
EMAIL: yevhenii.kravchenko@nure.ua (Yevhenii Kravchenko); iryna.kyrychenko@nure.ua (Iryna Kyrychenko); hlib.tereshchenko@nure.ua (Glib Tereshchenko)
ORCID: 0009-0003-6796-4509 (Yevhenii Kravchenko); 0000-0002-7686-6439 (Iryna Kyrychenko); 0000-0001-8731-2135 (Glib Tereshchenko)

2. Огляд проблем у сфері вантажних перевезень

Сфера вантажних перевезень стикається з низкою проблем, які знижують її ефективність, а деякі, навіть, завдають великих збитків економіці. Серед таких проблем можна виділити наступні:

- Непрозорість ланцюгів поставок
- Помилки та швидкість документообігу
- Витрати на посередників
- Шахрайство
- Затримки оплати і доставки

Кожна з зазначених проблем має свій негативний вплив. Учасники ланцюгів поставок – логісти, митні органи, замовники, перевізники не завжди мають доступ до актуальної інформації про поточний статус вантажів чи документів, що зазвичай призводить до затримок і суперечок між партнерами. Ручне введення даних, заповнення документів може призвести до помилок різного роду, які, у свою чергу, можуть ускладнити роботу з документами у подальшому і знову призвести до суперечок між сторонами. В організації вантажних перевезень, окрім основних зацікавлених сторін, участь приймають різні посередники, наприклад, брокери та банки. Такі посередники зазвичай просять певну суму, або відсоток від угоди чи транзакції, що помітно підвищує витрати.

Не виключені і ситуації з крадіжками, підробкою документів, таких як накладні чи сертифікати, що підвищує ризики для усіх сторін. Очевидно, що подібні махінації приносять великі збитки. Згідно CargoNet [2] у 2023 році кількість крадіжок товарів вирісла на 57% у порівнянні з минулим роком. За їхніми даними було втрачено товарів на суму $130 мільйонів. Вважають, що реальні цифри набагато більші, адже не усі крадіжки розкриваються і не про всі доповідають.

Відсутність автоматизації підтвердження доставки та обробки платежів може призвести до фінансових втрат і зниження довіри між партнерами.

3. Огляд можливостей технології блокчейн та її застосування для вирішення проблем у сфері вантажних перевезень

Блокчейн є сучасною технологією, яка забезпечує високий рівень безпеки, надійності та прозорості у контексті зберігання даних та їх обміну [3]. Такі його особливості як незмінність, розподіленість, децентралізованість та механізм консенсусу гарантують, що інформація є незмінною, доступною для перевірки та захищеною від несанкціонованого доступу.

Блокчейни можна поділити на публічні, приватні, гібридні та блокчейни консорціуму. Для логістики найчастіше використовуються приватні та блокчейни консорціуму, такі як Hyperledger Fabric, через їхню масштабованість і можливість застосовувати права доступу, що є критичним для захисту комерційних даних [4]. На відміну від публічних блокчейнів, наприклад, Ethereum, які є відкритими, але повільними через велике навантаження, приватні дозволяють обмежити доступ до мережі лише авторизованим користувачам, тобто перевізникам, замовникам, логістам, забезпечуючи при цьому конфіденційність і швидкість.

Механізми консенсусу забезпечують узгодженість даних між учасниками мережі. Найпоширенішим механізмом у приватних блокчейнах є PBFT (Practical Byzantine Fault Tolerance) [5]. Він дозволяє вузлам досягати консенсусу навіть за наявності до третини ненадійних учасників. Процес включає в себе три фази: пропозиція транзакції, голосування та підтвердження. PBFT забезпечує високу швидкість і низьке енергоспоживання, що є важливим для логістичних систем, де потрібна швидка обробка даних.

Інтеграція блокчейну з існуючими логістичними платформами може здійснюватися, наприклад, через спеціалізовані API, які дозволяють синхронізувати блокчейн із різними системами. Окрім цього блокчейн можна інтегрувати з IoT пристроями, наприклад, GPS-трекерами, датчиками, для відстеження переміщення вантажів у реальному часі.

Блокчейн фіксує усі операції у реальному часі, починаючи від завантаження вантажу і закінчуючи його доставкою до місця призначення. При цьому усі учасники мають доступ до єдиного джерела даних, що зменшує ризик виникнення непорозумінь.

За допомогою блокчейну можна автоматизувати документообіг та задати певний стандарт ведення документів. Таким чином вдасться мінімізувати помилки, які пов'язані з людським фактором.

Завдяки такій особливості блокчейну як незмінність, а також можливості використання криптографії, можна бути впевненим у безпеці даних. Документи є захищеними від несанкціонованого доступу, а фальсифікація накладних чи сертифікатів стає неможливою через те, що дані в блокчейні є незмінними.

З допомогою блокчейну можна ефективно протидіяти шахрайству у сфері вантажних перевезень. У таблиці 1 наведено приклади поширених сценаріїв шахрайства та механізми їх блокування.

Таблиця 1

Поширені сценарії шахрайства та механізми їх блокування через блокчейн

Сценарій	Опис	Механізм блокування через блокчейн
Підробка документів	Фальсифікацій накладних, митних декларацій, сертифікатів	Незмінність даних забезпечить автентичність документів
Маніпуляції з оплатами	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
Схеми з посередниками	Змова з посередниками з метою завищення витрат або приховування частини транзакій	Децентралізована природа блокчейну усуває потребу в більшості посередниках, а смарт-контракти автоматизують оплату та контроль доставки

Ще одним інструментом, який використовується разом із блокчейном, є смарт-контракти. Це певний алгоритм, який виконується самостійно і забезпечує виконання усіх дій та умов, які прописані у ньому. У згаданому раніше Hyperledger Fabric для написання смарт-контрактів використовується мова програмування Chaincode. Наприклад, отримавши сигнал від GPS-трекера чи IoT-мітки про доставку вантажу, смарт-контракт може провести платіж. Або ж, отримуючи дані від IoT-датчиків (температура, вологість, тиск), смарт-контракт може відстежувати умови транспортування та, наприклад, сповістити відповідального про порушення та накласти штрафи на перевізника. Такий механізм усуває потребу в посередниках, прискорює транзакції та підвищує довіру між партнерами. За рахунок автоматизації процесів можна також скоротити витрати на посередників.

Деякі відомі компанії вже сьогодні застосовують блокчейн.

DHL – одна з провідних міжнародних логістичних компаній, яка активно впроваджує інноваційні рішення, зокрема блокчейн технології, для покращення своїх послуг [6]. Компанія використовує блокчейн для відстеження стану товарів і забезпечення їхньої автентичності, що є особливо важливим у фармацевтиці.

Walmart – американська компанія, яка управляє найбільшою у світі мережею оптової та роздрібної торгівлі, використовує блокчейн для забезпечення прозорості в ланцюгах постачання продуктів харчування. Система значно скоротила час визначення джерела походження товарів з 7 днів до 2 секунд [7].

Розглянуті приклади демонструють реальну користь впровадження технології блокчейн у сфері вантажних перевезень.

Існують перспективні напрямки досліджень та розвитку блокчейну у сфері логістики. Наприклад, інтеграція блокчейну зі штучним інтелектом, Big Data, що відкриває можливість для прогнозування ризиків, оптимізації маршрутів та прогнозування попиту на товари. Інтеграція з IoT-пристроями дозволить підвищити ступінь автоматизації процесів під час доставки вантажів та відстежувати умови перевезення, що підвищить якість товарів та допоможе вчасно запобігти порушенню умов транспортування.

4. Висновки

Технологія блокчейн має значний потенціал для інновації вантажних перевезень завдяки вирішенню таких проблем, як непрозорість, шахрайство, помилки та швидкість документообігу, недовіра між партнерами через затримки доставки вантажів чи виплат. Завдяки прозорості, безпеці та автоматизації через використання смарт-контрактів блокчейн підвищує прозорість та рівень безпеки у сфері вантажних перевезень.

Практичне застосування технології блокчейн у логістичних ланцюгах розглянуто на прикладі таких відомих компаній як DHL та Walmart, які на своєму прикладі демонструють реальну користь використання блокчейну у сфері вантажних перевезень.

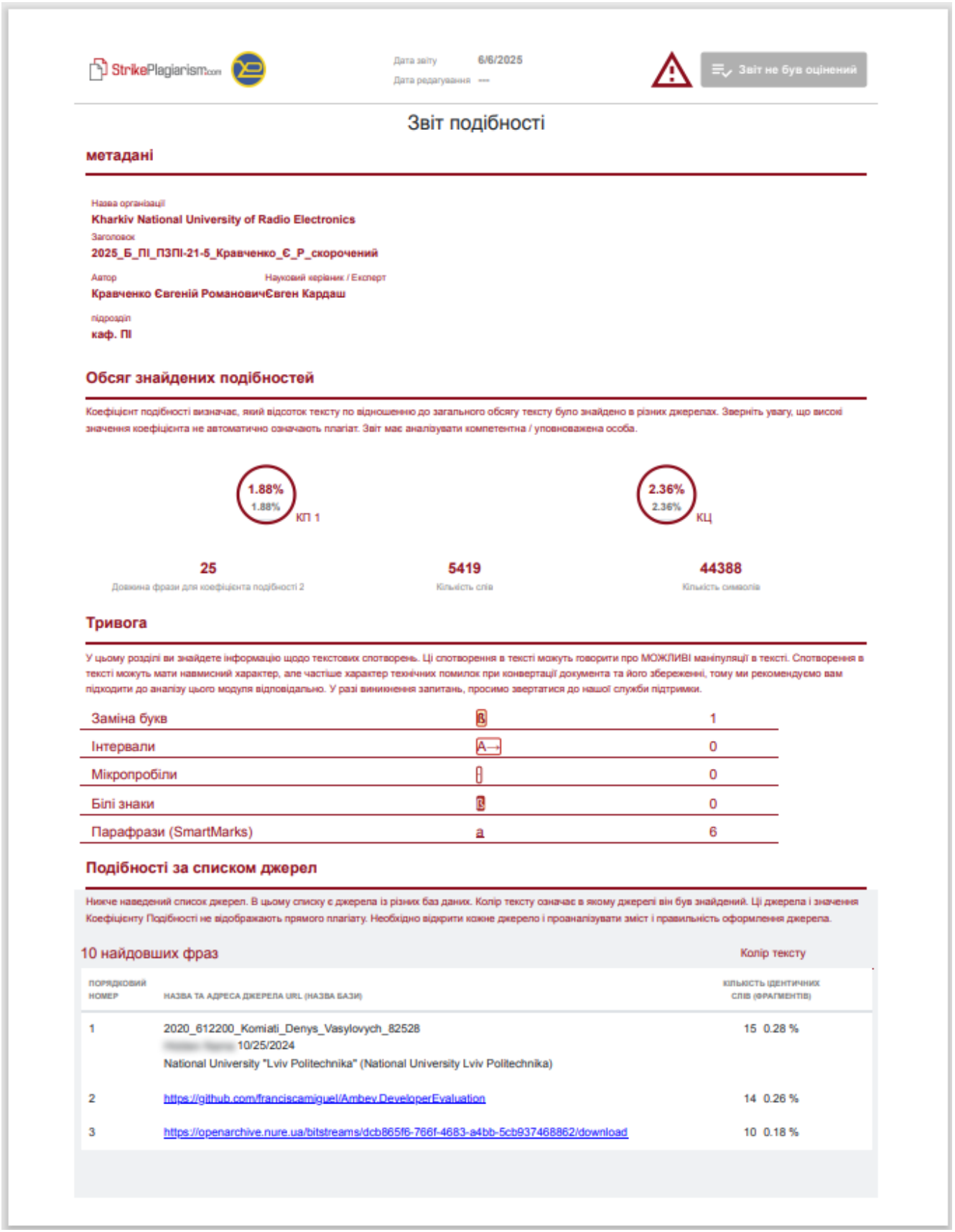
Подальший розвиток блокчейн-систем у логістиці потребує подолання проблем різного рівня – масштабованість, швидкодія, стандартизація процесів, врегулювання юридичних питань, тощо. Використання блокчейну зі штучним інтелектом, IoT-пристроями та Big Data відкриває перспективи для створення адаптивних, надійних та економічно вигідних логістичних систем, які відповідають вимогам сучасної економіки.

5. Література

[1] 100 Jaw-Dropping Supply Chains & logistics Statistics to know. URL: https://docshipper.com/logistics/100-supply-chain-logistics-statistics-2/.
[2] What is blockchain? URL: https://www.ibm.com/think/topics/blockchain.
[3] Узгодження рішень в умовах зради. URL: https://dou.ua/forums/topic/52819/?from=forlatest
[4] DHL – blockchain usage. URL: https://www.dhl.com/us-en/home/innovation-in-logistics/logistics-trend-radar/blockchain-logistics.html.
[5] Г. Терещенко, Аналіз і обґрунтування використання наявних блокчейн-рішень для захисту цифрових активів, Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27), с. 164-178.
[6] Four Examples of Blockchain in Supply Chain Management. URL: https://www.softeq.com/blog/four-blockchain-supply-chain-examples.
[7] 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.

ДОДАТОК Д

Звіт з результатами перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/6/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ-21-5_Кравченко_Є_Р_скорочений
Автор
Кравченко Євгеній Романович
Науковий керівник / Експерт
Євген Кардаш
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

1.88% 1.88% КП 1

2.36% 2.36% КЦ

25	5419	44388
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	B	1
Інтервали	A→	0
Мікропробіли	¶	0
Білі знаки	B	0
Парафрази (SmartMarks)	a	6

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2020_612200_Komiati_Denys_Vasylovych_82528 [redacted] 10/25/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	15 0.28 %
2	https://github.com/franciscamiguel/Ambev.DeveloperEvaluation	14 0.26 %
3	https://openarchive.nure.ua/bitstreams/dcb865f6-766f-4683-a4bb-5cb937468862/download	10 0.18 %

4	2020_612200_Komiati_Denys_Vasylovych_82528 [redacted] 10/25/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	10 0.18 %
5	2020_612200_Komiati_Denys_Vasylovych_82528 [redacted] 10/25/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	9 0.17 %
6	Програмне забезпечення біржі фріланcу [redacted] 6/19/2022 National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби")	9 0.17 %
7	Програмне забезпечення біржі фріланcу [redacted] 6/19/2022 National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби")	9 0.17 %
8	2020_612200_Komiati_Denys_Vasylovych_82528 [redacted] 10/25/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	7 0.13 %
9	2020_612200_Komiati_Denys_Vasylovych_82528 [redacted] 10/25/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	7 0.13 %
10	https://codereview.stackexchange.com/questions/171162/generic-repository-and-generic-service	6 0.11 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (1.33 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2020_612200_Komiati_Denys_Vasylovych_82528 [redacted] 10/25/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	48 (5) 0.89 %
2	Програмне забезпечення біржі фріланcу [redacted] 6/19/2022 National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби")	18 (2) 0.33 %
3	Вплив архітектурних особливостей на вибір між Grpc та RESTFUL API для мікросервісної архітектури [redacted] 12/2/2024 Kharkiv National University of Economics named after S.Kuznets (KNUE) (KNUE)	6 (1) 0.11 %

з Інтернету (0.55 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://github.com/franciscamiguel/Ambev.DeveloperEvaluation	14 (1) 0.26 %
2	https://openarchive.nure.ua/bitstreams/dcb865f6-766f-4683-a4bb-5cb937468862/download	10 (1) 0.18 %
3	https://codereview.stackexchange.com/questions/171162/generic-repository-and-generic-service	6 (1) 0.11 %

Список прийнятих фрагментів (немає прийнятих фрагментів)