Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: "Розробка захищеної онлайн-платформи для продажів на основі технології blockchain"

Виконав: студент (ка) IV курсу, групи СБ-42

Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Станіславський Максим Володимирович
підпис (прізвище та ініціали)

Керівник Козак Р. О.
підпис (прізвище та ініціали)

Нормоконтроль Дроздова Т. В..
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Станіславському Максиму Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка захищеної онлайн-платформи для продажів на основі технології blockchain

Керівник роботи Козак Руслан Орестович, к.т.н. доцент кафедри кібербезпеки
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «02» 05 2025 року № 4/7-361

2. Термін подання студентом завершеної роботи 17.06.2025

3. Вихідні дані до роботи Вимоги до безпеки онлайн-магазину

4. Зміст роботи (перелік питань, які потрібно розробити)
Проаналізувати технологію blockchain
Проаналізувати захист централізованих та децентралізованих вебзастосунків
Розробити та протестувати онлайн-платформу для продажів на основі технології blockchain
Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Тема, мета, задачі. Актуальність дослідження та стан проблеми безпеки в електронній комерції. Загрози кібербезпеки в сучасних онлайн-магазина. Проблеми централізованих платформ та потреба у децентралізації. Принципи роботи блокчейну Ethereum. Смарт-контракти: визначення та особливості. Цикл життя транзакції в Ethereum. Архітектура розробленого онлайн-магазину (Next.js+Magento 2). Взаємодія користувача з MetaMask та смарт-контракта Верифікація та розгортання смарт-контракту. Прототип взаємодії магазину з контрактом у вигляді схеми. Висновки та подальші перспективи розвитку системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Мариненко С.Ю., к. т. н., доцент кафедри інжиніринг МТ		

7. Дата видачі завдання 29.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.01 – 30.01	Виконано
2.	Підбір джерел для аналізу блокчейну	31.01 – 05.02	Виконано
3.	Опрацювання джерел в галузі дослідження	06.02 – 20.02	Виконано
4.	Провести аналіз методів захисту децентралізованих вебдодатків	22.02 – 12.03	Виконано
5.	Налаштування hardhat	15.03-25.03	Виконано
6.	Налаштування безпеки онлайн-платформи для продажів	25.02 – 10.04	Виконано
7.	Оформлення розділу «Аналіз ризиків безпеки централізованих та децентралізованих вебзастосунків»	10.02 – 05.03	Виконано
8.	Оформлення розділу «Дослідження технологій Ethereum та смарт-контрактів»	26.03 – 11.04	Виконано
9.	Оформлення розділу «Розробка та тестування прототипу децентралізованого вебдодатку»	12.04-20.04	Виконано
10.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	26.04 – 07.05	Виконано
11.	Оформлення кваліфікаційної роботи	02.05 – 07.06	Виконано
12.	Нормоконтроль	10.06 – 11.06	Виконано
13.	Перевірка на плагіат	12.06 – 13.06	Виконано
14.	Попередній захист кваліфікаційної роботи	16.06 – 17.06	Виконано
15.	Захист кваліфікаційної роботи	27.06.2025	

Студент _____ Станіславський М. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Козак Р. О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка безпечного онлайн-магазину на основі технології blockchain // Кваліфікаційна робота ОР «Бакалавр» // Станіславський Максим Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБ-42 // Тернопіль, 2025 // С. 56, рис. – 14, табл. – - , кресл. – -, додат. – 3.

КЛЮЧОВІ СЛОВА: Blockchain, Ecommerce, Ethereum, Security, Cybersecurity

У роботі проаналізовано сучасні загрози безпеці електронної комерції та обґрунтовано актуальність використання Blockchain для їх мінімізації. Розглянуто принципи роботи блокчейну Ethereum, смарт-контракти на мові Solidity і засоби інтеграції блокчейну з веб-застосунками (Node.js, Web3 тощо). Реалізовано прототип онлайн-магазину з підключенням до мережі Ethereum: розроблено смарт-контракти для забезпечення безпечних транзакцій (оплата товарів криптовалютою, реєстрація/логін користувачів). Результати тестування прототипу продемонстрували, що використання блокчейн-технології дозволяє підвищити захищеність онлайн-магазину: забезпечується цілісність і прозорість даних транзакцій, унеможливлюється несанкціоноване коригування інформації про замовлення, зменшується ймовірність шахрайства.

ABSTRACT

Development of a secure online store based on blockchain technology // Qualification work for bachelor's degree // Stanislavskyi Maksym // Ternopil National Technical University named after Ivan Puluj, Faculty of Computer Information Systems and Software Engineering, Department of Cyber Security, group SB-42 // Ternopil, 2025 // P. - 56, fig. - 14, table -, drawings -, appendix - 3.

KEYWORDS: Blockchain, Ecommerce, Ethereum, Security, Cybersecurity

The paper analyzes the current threats to e-commerce security and substantiates the relevance of using Blockchain to minimize them. The principles of the Ethereum blockchain, smart contracts in the Solidity language, and means of integrating the blockchain with web applications (Node.js, Web3, etc.) are considered. A prototype of an online store connected to the Ethereum network was implemented: smart contracts were developed to ensure secure transactions (payment for goods with cryptocurrency, order verification). The results of testing the prototype demonstrated that the use of blockchain technology can improve the security of the online store: it ensures the integrity and transparency of transaction data, prevents unauthorized changes to order information, and reduces the likelihood of fraud.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

PoW	–	Proof of Work
PoS	–	Proof of Stake
API	–	Application Programming Interface
ETH	–	Ethereum

ВСТУП

Актуальність теми. Сфера електронної комерції стрімко розвивається, але водночас супроводжується зростанням кількості кіберзагроз і методів для шахрайства. За оцінками дослідників, втрати інтернет-магазинів від онлайн-шахрайства щороку обчислюються десятками мільярдів доларів. Відповідно до статистики, Mastercard/Juniper (2024) оцінює глобальні втрати від ecommerce шахрайства $41 млрд у 2022 році [1]. Традиційні централізовані платформи електронної комерції мають вразливі місця, пов’язані з ризиком компрометації даних користувачів, атаками на платіжні системи та людським фактором (наприклад, несанкціонований доступ адміністраторів або помилки, що призводять до витоків даних). Це відкриває нові можливості для захисту даних і фінансових операцій: децентралізована та незмінна структура блокчейну підвищує безпеку системи електронної комерції, захищаючи цілісність транзакцій і знижуючи ризики шахрайства, несанкціонованого доступу та витоків даних. Існуючі традиційні системи електронної комерції, що базуються на централізованих рішеннях, мають значні вразливості. Зокрема, такі платформи стикаються з ризиками компрометації персональних даних користувачів, атаками типу «людина посередині», витоками платіжної інформації, шахрайськими діями з боку адміністраторів і недобросовісних працівників. Крім цього, існують і технологічні вразливості: недостатнє шифрування даних, неналежний контроль доступу та проблеми з автентифікацією користувачів. Ці фактори суттєво підривають довіру користувачів до онлайн-магазинів і створюють серйозні фінансові та репутаційні втрати для компаній, які працюють у сфері e-commerce.

Мета і задачі дослідження. Метою даної роботи є розробка прототипу безпечного онлайн-магазину з використанням технології блокчейн (Ethereum), що забезпечує підвищений рівень захисту транзакцій та даних користувачів. Для досягнення поставленої мети в роботі необхідно вирішити задачі, такі як: аналіз сучасних загроз та вразливостей в електронній комерції; вивчення технології блокчейн на базі платформи Ethereum та з’ясувати яким чином вони можуть бути

використані для підвищення безпеки; реалізувати прототип онлайн-магазину з виконанням основної бізнес-логіки магазину, такої як реєстрація користувача та здійснення оплати за товар; забезпечити кібербезпеку розробленого рішення.

Об'єкт дослідження. Об'єктом дослідження є системи електронної комерції та процеси забезпечення їхньої кібербезпеки, зокрема процеси обробки онлайн-транзакцій в інтернет-магазинах і захисту конфіденційних даних користувачів. Іншими словами, досліджується типовий онлайн-магазин як цілісна система, що включає клієнтську частину, серверну інфраструктуру та платіжні засоби, на які потенційно можуть здійснюватися атаки.

Предмет дослідження. Предметом дослідження є методи та засоби використання технології блокчейн для підвищення рівня безпеки онлайн-магазину. Зокрема, це застосування платформи Ethereum та смарт-контрактів для захисту транзакцій і даних: механізми децентралізоване проведення оплати через блокчейн, верифікація автентичності та цілісності даних без залучення довірених третіх сторін. Предмет охоплює також інтеграцію блокчейн-компонентів із традиційною архітектурою веб-застосунків (фронтенд/бекенд) та питання безпечної взаємодії між ними.

Практичне значення одержаних результатів. Запропоноване дослідження має високе практичне значення, оскільки результатом роботи є діючий прототип онлайн-магазину, що наочно демонструє можливості впровадження технології блокчейн для кібербезпеки платформ електронної комерції. Цей прототип може стати основою для реального впровадження у бізнес-практику компаній, які зацікавлені в підвищенні безпеки своїх інтернет-магазинів. Використання таких технологій дозволить значно скоротити ризики шахрайства, мінімізувати можливості несанкціонованого доступу до персональних та фінансових даних, а також підвищити рівень довіри користувачів завдяки прозорості й захисту транзакцій. Крім того, впровадження таких рішень допоможе компаніям відповідати сучасним вимогам регуляторних органів щодо захисту даних та фінансових операцій, що позитивно вплине на їхню конкурентоспроможність і стійкість на ринку.

РОЗДІЛ 1 АНАЛІЗ РИЗИКІВ КІБЕРБЕЗПЕКИ ЦЕНТРАЛІЗОВАНИХ ТА ДЕЦЕНТРАЛІЗОВАНИХ ВЕБЗАСТОСУНКІВ

1.1 Вразливості централізованих систем онлайн-магазинів

За даними останніх досліджень, на сферу електронної комерції припадає до 32% успішних кібератак щорічно. Наслідками можуть бути фінансові втрати, витоки даних клієнтів та підрив довіри до компанії. Традиційні архітектури онлайн-магазинів є централізованими: всі дані та логіка зберігаються на одному сервері або кластері під контролем однієї організації. Загрози для інших типів систем проаналізовано в [2, 3, 4, 5]. Розглянемо основні загрози для безпеки та вразливості традиційних (централізованих) систем. Це створює декілька критичних вразливостей.

Крадіжка персональних даних. Шахрай може через вразливості в системі безпеки або за допомогою методів соціальної інженерії отримати несанкціонований доступ до конфіденційної інформації користувачів (електронна скриньки, пароль, адреса) з метою її викрадення. Наприклад, у результаті кібератаки на eBay у 2014 році зловмисники отримали доступ до зашифрованих паролів і особистої інформації 145 мільйонів користувачів (імена, електронні адреси, фізичні адреси, телефони та дати народження) [6]. У цьому випадку фінансова інформація (номери кредитних карток) не була викрадена, але обсяг скомпрометованих персональних даних зробив інцидент одним із найбільших в історії. Наслідком для компанії була втрата довіри клієнтів, шкода репутації та відтік користувачів, а також юридична відповідальність у вигляді судових позовів зі сторони громадськості та підвищеного нагляду з боку як громадськості, так і державних структур.

Однією з найбільш поширених атак на централізовані онлайн-магазини є атака типу «людина посередині» (Man-in-the-middle, MitM). Ця атака дозволяє зловмисникам перехоплювати й змінювати дані, які передаються між клієнтом та сервером. Через використання ненадійних або погано захищених протоколів передачі даних (наприклад, незашифрованого HTTP замість HTTPS) хакери

можуть отримати доступ до конфіденційних даних, таких як паролі, номери кредитних карток та персональна інформація користувачів. Захист від таких атак вимагає використання надійних методів шифрування та регулярних аудитів безпеки мережевих з'єднань.

Шахрайство з оплатою. У сфері електронної комерції найпоширеніший сценарій – використання викрадених платіжних даних (номери кредитних карток, реквізити рахунків) для здійснення несанкціонованих покупок.

Рисунок 1.1 – Типи шахрайств з кредитними карточками

Зловмисники можуть отримати ці дані з витоків (зломів сайтів, про що йшлося вище) або через фішингові атаки, а потім здійснювати покупки від імені жертв. Інша форма – так званий “friendly fraud” (дружнє шахрайство), коли реальний покупець після отримання товару оскаржує транзакцію, удаючи, що не здійснював покупку, аби отримати компенсацію і фактично привласнити товар без оплати. Також зустрічаються випадки шахрайства з боку продавців – створення фальшивих інтернет-магазинів, що приймають оплату, але не відправляють. Масштаби платіжного шахрайства в електронній комерції дуже великі. Як показує статистика, в Європі частка шахрайств саме в середовищі електронної оплати через картку становила близько 84% від усіх фінансових махінацій з картками у 2021 році, що підкреслює домінування онлайн-шахрайств [7].

Зловживання привілейованим доступом. Привілейований доступ – це розширені права в системі, які мають адміністратори або інші уповноважені особи. Зловживання такими привілеями розглядається як серйозна кіберзагроза, адже внутрішній порушник може використати свої права для заподіяння витоку

особистих даних користувачів, платіжних даних тощо. На відміну від зовнішніх хакерів, інсайдер уже володіє законним доступом, тому його шкідливі дії важче виявити. За статистикою, майже кожен п’ятий випадок витоку даних у світі спричинений саме діями персоналу [8].

1.2 Застосування блокчейна для кіберзахисту вебзастосунків

1.2.1 Поняття та принцип роботи блокчейна

Блокчейн (англ. blockchain) – це розподілений реєстр транзакцій, що зберігається одночасно на багатьох вузлах. Ключова особливість – незмінність даних: інформацію, записану в блокчейн, надзвичайно складно (практично неможливо) підробити чи видалити завдяки криптографічним зв’язкам між блоками. На відміну від централізованої БД, що знаходиться в єдиному місці, блокчейн розподіляє дані між багатьма учасниками мережі, що суттєво ускладнює.

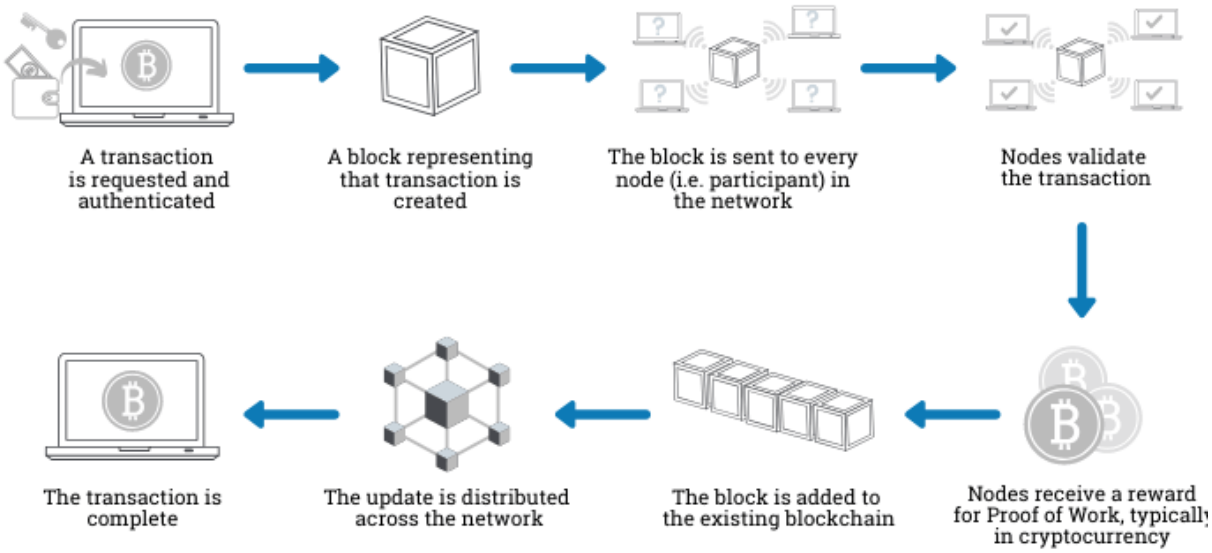

Рисунок 1.2 – Приклад роботи блокчейна

Ключову роль у безпеці блокчейну відіграє хешування – перетворення будь-яких вхідних даних за допомогою математичного алгоритму (хеш-функції) в унікальний рядок фіксованої довжини. Кожен блок у блокчейні має свій

криптографічний хеш, який обчислюється на основі його вмісту. Більш того, блок містить хеш попереднього блоку, утворюючи ланцюжок: будь-яка зміна даних у блоці змінить його хеш і розірве ланцюг зв'язків. Завдяки цьому забезпечується цілісність.

Дані в блокчейні групуються в блоки. Кожен блок зазвичай містить список підтверджених транзакцій, свій унікальний криптографічний підпис (хеш) та посилання на хеш попереднього блоку. Така структура "блок + хеш попередника" і формує безперервний ланцюг блоків.

Принцип роботи блокчейна базується на двох основних концепціях: блоках та криптографії. Кожен блок містить набір транзакцій, часову мітку (timestamp), унікальний криптографічний хеш блоку і хеш попереднього блоку. Саме це посилання на попередній блок формує ланцюжок блоків (звідси й назва – blockchain), що робить структуру блокчейна безперервною та стійкою до маніпуляцій.

Криптографічна безпека блокчейна забезпечується хеш-функціями, такими як SHA-256, які генерують унікальний хеш-фрагмент для будь-якого обсягу вхідних даних. Це дозволяє легко перевірити цілісність інформації: будь-яка зміна даних в одному блоці негайно змінює його хеш, а це, у свою чергу, змінює хеші усіх наступних блоків. Тому несанкціоновані зміни в мережі відразу помітні учасникам і є дуже дорогими з точки зору ресурсів.

Розподілений характер блокчейна означає, що кожен учасник мережі зберігає повну копію всього ланцюга блоків. Це означає, що немає єдиної точки відмови: навіть якщо деякі вузли вийдуть з ладу або будуть атаковані, мережа продовжить функціонувати. У розподіленій мережі критично важливо, щоб усі вузли погоджувалися з єдиним станом реєстру. Це досягається за допомогою алгоритму консенсусу (Proof of Work та Proof of Stake) – набору правил, які визначають, як учасники підтверджують валідність нових блоків і транзакцій. Алгоритми консенсусу гарантують, що всі вузли мережі мають однакову копію даних і довіряють її цілісності. Найпоширенішими типами консенсусу є Proof of Work (PoW) та Proof of Stake (PoS).

Proof of Work (PoW) – це оригінальний механізм, використаний у Bitcoin, де так звані майнери виконують складні обчислення для перевірки транзакцій і додавання нового блоку. Майнери змагаються у вирішенні криптографічної задачі; перший, хто знайде рішення (хеш з необхідними властивостями), отримує право додати блок у ланцюг і винагороду у вигляді криптовалюти.

Proof of Stake – альтернативний механізм, запропонований для підвищення ефективності. У системі PoS відсутнє майнінгове змагання; замість цього право створити новий блок надається вузлам-валідаторам пропорційно до кількості монет, яку вони тримають та «заморожують» як заставу (stake).

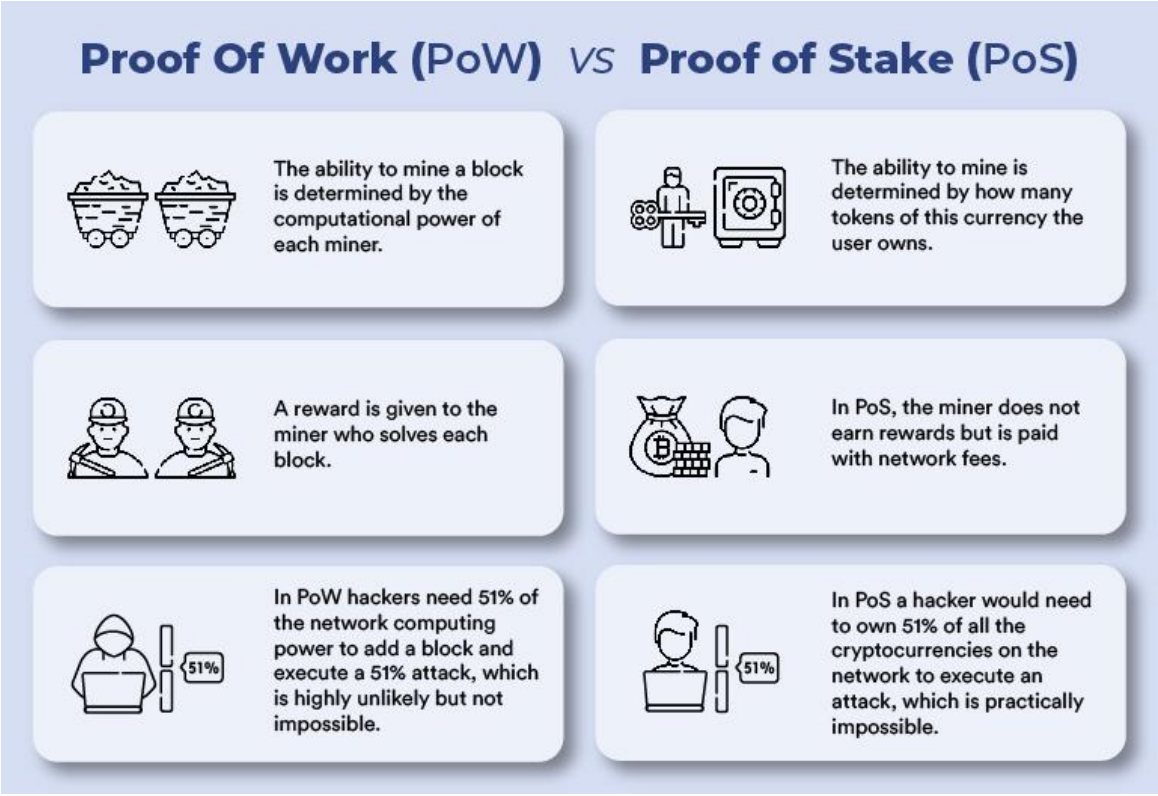

Рисунок 1.3 – Порівняння PoW та PoS як алгоритмів консенсусу

Найважливіше, що є у блокчейні, це вузли (ноди). Вузли є основою децентралізованої мережі: вони зберігають повні копії блокчейну, ретранслюють транзакції і блоки, а також беруть участь у перевірці та підтвердженні нових блоків. Деякі вузли (наприклад, майнери або валідатори) активно додають нові блоки, інші – підтримують актуальну копію ланцюга і перевіряють коректність даних. Важливо, що будь-хто може стати вузлом, встановивши програмне

забезпечення і приєднавшись до мережі – завдяки цьому блокчейн є відкритою і довірчою системою без центрального керівника.

Суттєвою особливістю блокчейну є публічність і прозорість даних. У публічних мережах, таких як Bitcoin або Ethereum, всі транзакції доступні будь-якому учаснику мережі, що дозволяє здійснювати ефективний аудит і забезпечувати повну прозорість фінансових і операційних процесів.

В цілому, використання технології блокчейн дозволяє значно підвищити рівень безпеки даних і транзакцій, зменшити ризики несанкціонованого доступу, шахрайства та корупції, а також забезпечити прозорість і довіру між учасниками мережі.

1.2.2 Методи використання блокчейна для кіберзахисту вебзастосунків

Одна з ключових переваг блокчейну – незмінність записів. Після того як дані записано до блокчейну, їх практично неможливо змінити або видалити без сліду. Ця властивість особливо цінна для захисту критично важливих даних вебзастосунків, таких як журнали доступу, історія фінансових транзакцій, налаштування конфігурацій або контрольні хеші файлів. Наприклад, адміністратор вебдодатка, що має доступ до бази даних, не зможе «приховати» свою діяльність або стерти компрометуючі записи, якщо ці дані заздалегідь були хешовані та збережені у блокчейні. Це робить блокчейн ефективним інструментом для виявлення несанкціонованих змін та аудиту безпеки. Для вебзастосунків це надає потужний інструмент контролю цілісності: критично важливі дані або хеші цих даних можна зберігати в блокчейні. Якщо зловмисник отримає доступ до бази даних застосунку і спробує змінити або видалити інформацію, розподілений реєстр одразу виявить невідповідність – адже оригінальний запис залишається в блокчейні. Незмінність даних гарантує, що історія змін не може бути переписана, а всі учасники можуть криптографічно перевірити автентичність даних у будь-який момент. Це стосується як даних користувачів, так і, наприклад, важливих налаштувань чи довідників – будь-якої інформації, де критично знати, що вона не була підроблена. Через цю перевагу,

уряд Естонії впровадив власний блокчейн (KSI) для захисту цілісності державних даних. З цьому рішенні “історію не може переписати ніхто – ані хакери, ані адміністратори, навіть саме керівництво – і автентичність електронних даних може бути математично доведена” [9].

Найбільший ризик для персональних даних – неправомірне використання або модифікація персональних даних інсайдером. Блокчейн працює з публічними адресами, які не містять прив’язки до особистих даних користувача. Таким чином, у разі атаки на мережу або витоку даних, зловмисник не отримає персоніфікованої інформації. У контексті вебзастосунків це дозволяє реалізувати автентифікацію без паролів – наприклад, через цифровий підпис транзакції гаманцем користувача. Сайт може перевірити, що запит дійсно підписаний власником публічного ключа, не зберігаючи жодної чутливої інформації (логінів, паролів, токенів) у своїй базі даних.

На рахунок шахрайства, то усі підтверджені транзакції в блокчейні зберігаються в загальному реєстрі, доступному для перегляду учасниками (у публічних блокчейнах – всім охочим). Це означає, що існує повний аудит-трейл фінансових операцій: кожен переказ можна відстежити до його джерела. Така прозорість ускладнює життя шахраям – спроби приховати “чорні” платежі чи провести несанкціоновані перекази стають більш помітними. Наприклад, у Bitcoin усі транзакції публічні, що дозволяє аналітикам виявляти підозрілі моделі переказів. У приватних або консорціумних блокчейнах (наприклад, міжбанківських) учасники мають спільний журнал транзакцій, і жодна сторона не може нишком відредагувати цифри у своїй копії – будь-яка розбіжність буде видима іншим. Це створює атмосферу взаємної довіри: контрагенти знають, що дані про платежі достовірні та узгоджені між усіма сторонами, а підробка виключена на технічному рівні. До того ж, при виникненні спорів або підозр у шахрайстві, блокчейн дає змогу швидко перевірити історію – адже вона перевірена і не може бути змінена.

Окрім цього, публічність та прозорість блокчейну є серйозною перевагою в контексті боротьби з фінансовим шахрайством. Усі підтверджені транзакції в блокчейні зберігаються в незмінному реєстрі, що доступний для перегляду

учасниками мережі. У публічних блокчейнах, як-от Ethereum або Bitcoin, кожна операція має цифрову історію: з якого гаманця, коли і на яку суму вона була здійснена. Це дає змогу здійснювати ретроспективний аналіз транзакцій, автоматично виявляти підозрілі схеми (наприклад, «розбивання» платежів на дрібні частини, перекази між пов’язаними адресами) та вчасно реагувати на шахрайство.

1.3 Прозорість і контроль

Однією з ключових проблем у сфері електронної комерції є недостатній рівень прозорості дій, які здійснюються з боку постачальників послуг, а також обмеженість контролю, який має кінцевий користувач над своїми власними транзакціями та даними. У централізованих системах клієнти зазвичай покладаються на обіцянки компаній щодо безпеки та добросовісності, проте ці обіцянки не завжди підкріплені технічними гарантіями. Це створює значний простір для шахрайства, маніпуляцій з боку адміністрації або навіть навмисного приховування інформації про транзакції чи змін умов обслуговування.

У цьому контексті блокчейн-технології, зокрема Ethereum, відкривають новий рівень прозорості завдяки децентралізованому, незмінному та публічно доступному реєстру транзакцій. Кожна дія – від реєстрації користувача до здійснення покупки – фіксується у вигляді транзакції, яка зберігається в блоках та доступна для перегляду будь-яким учасником мережі. Це означає, що навіть адміністрація онлайн-магазину не має можливості змінити чи приховати записану подію без згоди мережі, оскільки консенсусний механізм Ethereum не допускає односторонніх змін.

Користувачі можуть самостійно перевіряти хеші своїх транзакцій, ідентифікатори замовлень, статус оплати та інші критично важливі параметри, що зберігаються у смарт-контрактах. Наприклад, якщо в магазині було заявлено, що товар буде відправлений після підтвердження транзакції, то сам факт підтвердження можна перевірити через оглядач блоків (наприклад, Etherscan), не звертаючись до техпідтримки чи адміністрації.

Крім того, прозорість у блокчейні сприяє формуванню довіри до бренду. Клієнти можуть бути впевнені, що система не приховує дані про знижки, не змінює умови акцій або не блокує частину функціоналу на власний розсуд. Це особливо актуально у B2C-сегменті, де довіра до продавця має безпосередній вплив на рівень продажів.

Ще одним важливим аспектом є підвищення захищеності споживача від шахрайства. Завдяки відкритому коду смарт-контрактів можна перевірити, що функції не містять прихованих механізмів переказу коштів третім особам або коду, який може змінювати логіку обробки оплати.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ETHEREUM ТА СМАРТ-КОНТРАКТІВ

2.1 Поняття та принцип роботи технології Ethereum

Ethereum – це блокчейн-платформа другого покоління, яка розширює можливості технології Bitcoin, додаючи підтримку повноцінних програм на блокчейні. Ethereum забезпечує розподілену обчислювальну платформу для різноманітних децентралізованих застосунків та має вбудовану криптовалюту Ether (ETH), але на відміну від Bitcoin, що підтримує лише передачу власних монет, Ethereum дозволяє створювати безліч інших токенів і децентралізованих сервісів поверх свого блокчейну.

Ethereum функціонує як глобальний розподілений комп'ютер, копії якого запускаються на тисячах вузлів (комп'ютерів) по всьому світу. Кожен вузол мережі зберігає повну історію блокчейну та актуальний глобальний стан системи – інформацію про всі акаунти, баланси та дані смарт-контрактів. Мережа працює за принципом спільного виконання: всі вузли узгоджуються щодо послідовності транзакцій і однаково відтворюють обчислення, щоб дійти до одного й того ж нового стану після кожного блоку. Такий підхід реалізує детерміновану станову машину: на вході береться попередній стан системи та набір нових транзакцій, а на виході – новий стан після їх виконання. Правила переходу від старого стану до нового визначає алгоритм Ethereum і вбудована в кожен вузол Ethereum Virtual Machine (EVM). Усі чесні вузли виконують транзакції однаково, тому блокчейн Ethereum гарантує, що після додавання нового блоку вся мережа узгоджується на однаковому результаті обчислень (новому глобальному стані).

2.1.1 Транзакції в Ethereum

Транзакція в Ethereum – це підписаний цифровим ключем запит, який ініціюється користувачем через зовнішній обліковий запис і призводить до зміни стану блокчейн-системи. Такий запит може включати переказ криптовалюти

ETH, створення нового смарт-контракту або виклик функції вже розгорнутого контракту. Кожна транзакція містить технічні параметри, такі як відправник, отримувач, передана сума, лічильник nonce, вхідні дані для контракту, а також обов'язкові обмеження на використання обчислювальних ресурсів – gas limit та вартість за одиницю газу. Після створення транзакції вона проходить через етап підпису, транслюється в мережу, де тимчасово зберігається у транзакційному пулі до моменту, коли буде обрана валідатором для включення до блоку. У момент включення виконується розрахунок вартості за використані ресурси, і якщо транзакція була оброблена успішно, її ефекти стають частиною оновленого стану блокчейну. Якщо ж ресурсу недостатньо або трапляється помилка, транзакція відхиляється, а витрачений газ не повертається. Саме поняття газу дозволяє уникнути зловживань та забезпечити обмеження на складність обчислень, що є критично важливим з міркувань безпеки та стабільності мережі.

2.1.2 Блоки як основа структури

У блокчейні Ethereum блоки відіграють фундаментальну роль, оскільки вони слугують контейнерами для групи транзакцій, які разом утворюють хронологічно пов'язаний ланцюг подій. Кожен блок містить посилання на попередній, що забезпечує незмінність записаної історії та стійкість до фальсифікацій. У рамках механізму консенсусу Proof-of-Stake блоки створюються валідаторами, які обираються згідно із заздалегідь визначеним алгоритмом, що враховує ставку і випадковість. У блоці, окрім транзакцій, зберігаються також службові дані, зокрема хеш попереднього блоку, стан мережі, цифрові підтвердження голосування інших валідаторів, і навіть елементи, пов'язані з безпековими діями, такими як виключення зловмисників або підтвердження виходу з мережі. Розмір блоку визначається максимальною кількістю газу, яку може вмістити блок, що впливає на кількість і складність транзакцій. Для Ethereum це обмеження динамічне, і в сучасній реалізації допускається блок до 30 мільйонів одиниць газу. У стандартних умовах новий блок створюється приблизно кожні 12 секунд, що гарантує стабільне оновлення

стану мережі. Усі ці елементи дозволяють Ethereum не лише фіксувати та зберігати транзакції, а й забезпечувати їх криптографічну захищеність та незмінність, що є основою для довіри у децентралізованому середовищі.

2.1.3 Ethereum Virtual Machine (EVM)

EVM – це ізольоване віртуальне середовище виконання коду, однакове на всіх вузлах Ethereum. По суті, EVM можна уявити як єдиний віртуальний процесор, розподілений між усіма учасниками мережі. Кожен повний вузол Ethereum запускає EVM для обробки транзакцій і виконання смарт-контрактів, гарантуючи при цьому однаковий результат виконання на всіх комп'ютерах мережі.

EVM є машиною 256-бітної архітектури: під час виконання контракту EVM оперує стеком, пам'яттю та лічильником команд, а також відстежує витрачений газ (операційний ліміт). Вихідний код смарт-контракту компілюється у байт-код EVM (набір низькорівневих інструкцій), який зберігається в блокчейні. Коли транзакція викликає виконання контракту, EVM починає зчитувати його байт-код (немов з віртуальної ПЗП) та послідовно виконує інструкції, змінюючи стан віртуальної машини і за потреби звертаючись до даних контракту у постійній пам'яті (сховищі). Важливою рисою EVM є ізоляція виконуваного коду: смарт-контракт виконується у "пісочниці" без прямого доступу до файлової системи чи мережі зовнішнього світу. Це підвищує безпеку, оскільки навіть зловмисний або помилковий контракт не може вийти за межі визначених правил.

Для досягнення консенсусу щодо нових блоків Ethereum сьогодні використовує механізм Proof-of-Stake. Згідно зі статистикою, тисячі валідаторів по черзі пропонують блоки (кожні ~12 секунд формується черговий блок), перевіряють транзакції в них та голосують за їх валідність [10]. Валідатори повинні мати внесок (стейк) у 32 ETH, і за чесну роботу отримують винагороду в ETH, а у випадку зловживань ризикують втратити заставу. Така модель замінила колишній майнінг на основі Proof-of-Work і значно підвищила енергоефективність та швидкодію мережі Ethereum.

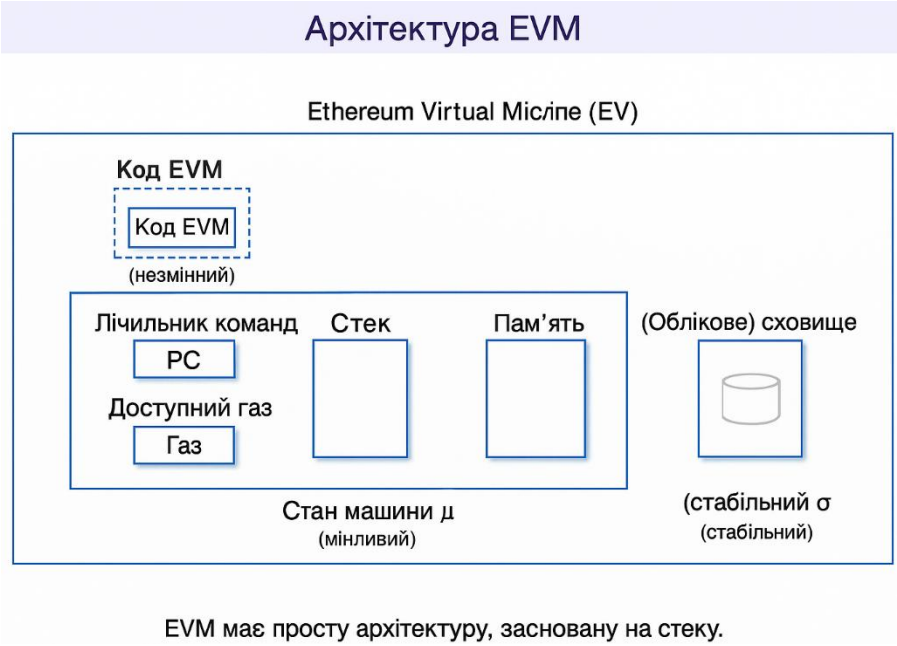

Рисунок 2.1 – Зображення архітектури EVM (Ethereum Virtual Machine)

Важливим аспектом EVM є її повна детермінованість. Це означає, що кожна транзакція або виклик контракту завжди приводить до одного й того ж результату незалежно від місця виконання або часу. Ця властивість є фундаментальною для забезпечення консенсусу в мережі Ethereum, оскільки всі вузли повинні отримати однаковий результат при виконанні одного й того ж коду.

2.1.4 Gas Limit та Gas Price в Ethereum

Газ (gas) – це одиниця виміру обчислювальної роботи в Ethereum. Кожна операція, яку виконує EVM (наприклад, додавання чисел, зберігання даних у сховище, відправка токенів), має фіксовану «цінність» у газі. Коли користувач відправляє транзакцію, він задає, скільки газу готовий витратити (параметр Gas Limit) і яку ціну заплатити за одиницю газу (Gas Price). Загальна плата за виконання транзакції визначається як:

$$P_{ETH} = G \times (B + P) \quad (1.1)$$

Газ виконує декілька важливих функцій одночасно. По-перше, він є платою за використання ресурсів мережі: складніші транзакції дорожчі, тому що споживають більше обчислень і пам’яті. Це створює економічний стимул оптимізувати смарт-контракти та перешкоджає зловживанням – ніхто не зможе завантажити мережу безкоштовно безмежним циклом, оскільки за кожну інструкцію доводиться платити. По-друге, механізм газу служить для обмеження виконання: коли EVM виконує контракт, він зменшує лічильник газу відповідно до виконуваних операцій. Якщо газ, виділений під транзакцію (gas limit), вичерпується до завершення контракту, виконання негайно припиняється, всі зміни відкочуються, а невитрачений газ не повертається відправнику (натомість сплачується як комісія за частково виконану роботу). Таким чином, розробники мають писати ефективний код, а користувач – встановлювати достатній ліміт газу, щоб транзакція довела виконання до кінця.

Ціна на газ формується відповідно до поточного стану мережі: при високому навантаженні ціна газу може зростати, а при низькому – знижуватися. Користувачі, які встановлюють вищу Gas Price, мають більшу ймовірність того, що їх транзакція буде швидко включена до наступного блоку, оскільки майнери або валідатори перш за все включають транзакції з найвищою ціною газу.

Оптимізація використання газу є важливим завданням для розробників смарт-контрактів, оскільки це впливає на вартість транзакцій для кінцевих користувачів. Контракти з добре оптимізованим кодом, який ефективно використовує ресурси EVM, зменшують загальні витрати користувачів і підвищують привабливість застосунку для широкого кола клієнтів. Тому важливо використовувати профільовані інструменти для аналізу і оптимізації споживання газу при розробці смарт-контрактів.

2.2 Поняття та принцип роботи технології смарт-контрактів

2.2.1 Технологія смарт-контрактів Ethereum

Смарт-контракт – це програмний код, який автоматично виконується в умовах блокчейну. У мережі Ethereum смарт-контракт має власну адресу і складається з набору функцій (коду) та даних (свого стану). Він не контролюється жодним окремим користувачем: після розгортання контракт «живе» у розподіленій мережі, і інші рахунки взаємодіють із ним шляхом відправки транзакцій, що викликають функції контракту. Таким чином смарт-контракт може задавати умови транзакцій або передачі коштів і автоматично примушувати їх виконання за цієї програми. Це забезпечує прозорість і надійність – усі учасники бачать програмну логіку контракту та її виконання.

Смарт-контракти складаються з двох основних компонентів: програмного коду, який визначає логіку і правила виконання контракту, та постійного сховища даних, яке містить стан цього контракту (наприклад, баланс рахунку, список користувачів, стан угоди). Код контракту пишеться спеціалізованими мовами програмування, такими як Solidity, яка спеціально створена для платформи Ethereum. Solidity дозволяє описувати складні бізнес-процеси і автоматизувати їх без участі посередників або центральних органів управління.

Принцип роботи смарт-контракту в мережі Ethereum базується на децентралізованій обчислювальній моделі Ethereum Virtual Machine (EVM). Коли користувач ініціює транзакцію, яка викликає функцію смарт-контракту, транзакція потрапляє до блокчейну та проходить процес валідації усіма вузлами мережі. EVM кожного вузла мережі виконує контрактний код у точності однаково, що забезпечує досягнення консенсусу щодо правильності результату виконання. Таким чином, будь-яка транзакція, що впливає на стан контракту, є абсолютно прозорою, неможливою для підробки або зміни заднім числом.

Ключовими властивостями смарт-контрактів є незмінність, автономність та прозорість. Після того як контракт розгорнуто у мережі, його неможливо модифікувати. Це забезпечує довіру серед учасників транзакцій, адже всі дії

виконуються строго згідно з описаними у контракті умовами. Завдяки автономності виконання смарт-контрактів немає потреби у довірених третіх сторонах, а їхня прозорість дозволяє всім учасникам переглядати логіку роботи контрактів та історію взаємодій з ними.

2.2.2 Взаємодія Ethereum та смарт-контрактів

Виконання смарт-контрактів у Ethereum забезпечується Ethereum Virtual Machine (EVM), що працює на всіх вузлах мережі. Кожна транзакція, яка ініціює смарт-контракт (створює його або викликає функцію), у мережі обробляється наступним чином: EVM на кожному вузлі виконує збережений байткод контракту для розрахунку нового стану або результатів операції. Газ ураховується за кожен зроблений крок обчислень, і якщо газ витрачено повністю, транзакція відкочується (втрати становлять лише сплачені комісії). Таким чином забезпечується консенсусне виконання – усі вузли перевіряють і фіксують результат запуску контракту в своєму локальному стані блокчейну.

Виконання смарт-контрактів у Ethereum забезпечується Ethereum Virtual Machine (EVM) – децентралізованою віртуальною машиною, що працює на всіх вузлах мережі. Кожна транзакція, яка ініціює смарт-контракт (створює його або викликає функцію), у мережі обробляється наступним чином: EVM на кожному вузлі виконує збережений байткод контракту для розрахунку нового стану або результатів операції. Газ ураховується за кожен зроблений крок обчислень, і якщо газ витрачено повністю, транзакція відкочується (втрати становлять лише сплачені комісії). Таким чином забезпечується консенсусне виконання – усі вузли перевіряють і фіксують результат запуску контракту в своєму локальному стані блокчейну. Смарт-контракти на Ethereum пишуться високорівневими мовами, які компілюються в байткод EVM. Найпоширеніша з них – Solidity, яка розроблена спеціально для Ethereum і нагадує синтаксис C++/JavaScript.

Переваги такого підходу полягають у високому ступені децентралізації та незмінності. Смарт-контракти, розміщені на Ethereum, отримують вигоду від безпеки та надійності блокчейн-мережі. Їхня логіка виконується прозоро (будь-

хто може побачити код) і автоматично, тому зменшуються ризики шахрайства та потреба в посередниках. Однак існують і обмеження: через обмежену пропускну здатність блокчейну Ethereum, висока активність приводить до зростання комісій за газ і затримок обробки. Крім того, незмінність коду означає, що після розгортання будь-які виправлення помилок потребують додаткових механізмів оновлення (наприклад, шаблони проксі-контрактів). В цілому, Ethereum залишається провідною платформою для смарт-контрактів, поєднуючи розвинену технічну інфраструктуру (EVM, мови програмування, інструменти розробки) зі значною спільнотою і підтримкою з боку Ethereum Foundation.

Окрім цього, смарт-контракти в Ethereum можуть взаємодіяти між собою, що дозволяє створювати складні ієрархічні системи та сервіси. Це дає можливість побудови багаторівневих децентралізованих додатків (DApps), які можуть включати як фінансові сервіси, так і нефінансові застосунки, такі як ідентифікація користувачів, управління правами доступу, логістика та голосування. Завдяки інтеграції з оракулами, смарт-контракти можуть також взаємодіяти із зовнішніми даними (наприклад, курси валют, прогнози погоди або результати спортивних змагань), що значно розширює їхні можливості і дозволяє автоматизувати процеси, які залежать від реальних подій.

Важливо підкреслити, що написання оптимізованих смарт-контрактів є критично важливим аспектом їх розробки. Оскільки кожна операція в Ethereum коштує певну кількість газу, складні і неоптимізовані контракти можуть суттєво підвищити витрати на транзакції, що негативно впливатиме на користувачів і ефективність усього додатку. Тому розробники мають приділяти особливу увагу оптимізації коду, мінімізуючи кількість необхідних операцій та економлячи ресурси мережі.

2.2.3 Безпека смарт-контрактів

Безпека смарт-контрактів є одним із критично важливих аспектів під час розробки будь-якого застосунку в екосистемі Ethereum. Через незмінну природу коду після його публікації, будь-яка помилка або недоліки логіки можуть

призвести до фатальних наслідків, включаючи втрату коштів, блокування доступу до ресурсів або втрату контролю над логікою управління. Історія Ethereum вже має приклади масштабних інцидентів, таких як злам The DAO або вразливості в мультисиг-гаманцях Parity, які підтверджують важливість належного захисту контрактної логіки на всіх етапах розробки.

Основними загрозами для смарт-контрактів є неконтрольований доступ до функцій, повторні виклики, перезапис змінних стану, конкуренція між транзакціями, а також логічні помилки у визначенні прав доступу. У децентралізованому середовищі немає централізованого механізму відкату змін чи підтримки постфактум, тому відповідальність за стабільність функціонування повністю покладається на розробника.

Надійна архітектура контракту повинна передбачати ретельне розмежування прав доступу, контроль умов виконання функцій, уникнення довільних викликів зовнішніх контрактів без перевірок та обмеження можливостей перевантаження мережі через надмірну складність обчислень. Для цього застосовуються шаблони розробки на зразок власника контракту, ролевого доступу або модульності, які дозволяють чітко структурувати логіку.

Невід’ємною складовою процесу розробки є багатоетапне тестування контракту. Важливо не лише покривати стандартні сценарії, але й враховувати граничні умови, атаки через сторонні контракти, а також ситуації з недостачею газу або спробами перевантаження сховищ. Автоматизовані засоби аналізу коду, зокрема Slither, MythX, Echidna чи Foundry, допомагають виявити вразливості ще до публікації контракту, однак вони не замінюють формальну верифікацію або зовнішній аудит.

Однією з найбільш просунутих технік у сфері забезпечення безпеки є застосування формальних методів верифікації, які дозволяють математично довести правильність реалізації певної логіки. У такий спосіб можна гарантувати, що контракт дійсно відповідає заявленим інваріантам, наприклад, збереженню балансу, однозначності викликів або відсутності доступу до функцій з боку сторонніх акторів.

Таким чином, безпека смарт-контрактів – це не одноразова дія, а постійний процес, що охоплює грамотну архітектуру, практики безпечного кодування, комплексне тестування та незалежну перевірку з боку фахівців. Саме цей підхід дозволяє зменшити ризики при розгортанні контрактів у незмінному, публічному і незахищеному середовищі Ethereum.

2.3 Безпечне зберігання критичних даних у блокчейні

Публічні блокчейн-платформи, на зразок Ethereum, не є конфіденційними за своєю природою – всі транзакції та їхній вміст видно кожному учаснику мережі. З цієї причини зберігання персональних або комерційно важливих даних у відкритому вигляді є неприйнятним із погляду захисту інформації. Однак це не означає, що блокчейн не може використовуватися для зберігання критично важливої інформації. Існують перевірені підходи до безпечної організації такої архітектури.

По-перше, дані перед збереженням у блокчейні піддаються криптографічній обробці. Це може бути хешування (наприклад, через keccak256) або шифрування асиметричними ключами (наприклад, за допомогою відкритого ключа користувача з MetaMask). Таким чином, навіть якщо дані потраплять у руки сторонніх осіб, вони не зможуть бути прочитані чи підроблені.

По-друге, застосовується підхід роздільного зберігання: самі дані (наприклад, опис товару, ПІБ покупця, адреса доставки) зберігаються у зовнішньому децентралізованому сховищі, наприклад IPFS, тоді як у блокчейні зберігається лише контрольна сума (CID) або хеш посилання на ці дані. Така модель дозволяє захистити вміст, забезпечуючи при цьому можливість верифікації автентичності інформації.

Крім того, можливе використання алгоритмів цифрового підпису. Коли користувач надсилає свої персональні дані для опрацювання замовлення, ці дані можуть бути підписані його особистим ключем, що дозволить у майбутньому довести автентичність цієї інформації, не розкриваючи її змісту.

Іншим поширеним механізмом є використання zero-knowledge proof (ZKP) – протоколів, що дозволяють перевірити істинність твердження без розкриття самих даних. Наприклад, замість того щоб передавати точну адресу доставки, система може підтвердити, що вона входить у дозволений географічний регіон, не розкриваючи повну адресу.

Завдяки цим підходам Ethereum може бути використаний не лише як фінансова система, але і як платформа для зберігання та обробки критичних даних із дотриманням принципів конфіденційності, цілісності та контролю доступу, які є основними у галузі кібербезпеки.

РОЗДІЛ 3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОТОТИПУ ДЕЦЕНТРАЛІЗОВАНОГО ВЕБДОДАТКУ

3.1 Розробка децентралізованого вебдодатку

3.1.1 Архітектура системи (Next.js + Magento 2)

Для створення безпечного, масштабованого та функціонального онлайн-магазину було реалізовано розподілену архітектуру, в якій різні компоненти системи відповідають за окремі аспекти обробки запитів. Система є стійкою до типових веб-атак (XSS, CSRF та ін.), відповідає сучасним вимогам щодо конфіденційності, доступності й цілісності інформації та легко адаптується до розширення функціоналу або збільшення навантаження.

Веб-фреймворк Next.js використовується переважно як фронтенд-фреймворк, який надає зручний, швидкий і безпечний інтерфейс для користувачів. Він дозволяє розробляти веб-додатки на основі React. Він відповідальний за наступні аспекти: виведення сторінок (з використанням як серверного рендерингу та клієнтського), взаємодія з користувачем (React-компоненти, інтеграція з Web3-гаманцями), оформлення замовлення (клієнтські компоненти з отримання даних від Magento 2), захист даних (екранування введення, Content Security Policy, перевірка форм). Ізоляція логіки представлення даних у Next.js дозволяє краще контролювати точки доступу, запобігає XSS-атакам та дозволяє реалізувати безпечні механізми аутентифікації через Web3 (без зберігання паролів на сервері).

Magento 2 використовується як потужний ecommerce-двигун, який бере на себе функції обробки бізнес-логіки, збереження даних, управління користувачами, кошиком, замовленнями і каталогом товарів. Magento дозволяє гнучко налаштовувати продуктову структуру, керувати кошиком, знижками, доставкою, оплатою, а також зберігати історію замовлень. Комунікація між Next.js та Magento 2 відбувається за допомогою GraphQL API, що дозволяє реалізувати лише ті запити, які дійсно потрібні клієнтській стороні. Наприклад,

замість отримання повного JSON-об'єкта товару, Next.js може надіслати запит лише на назву, зображення та ціну. Це значно знижує обсяг переданої інформації та підвищує продуктивність інтерфейсу.

3.1.2 Налаштування середовища у hardhat

Hardhat – це локальне середовище розробки смарт-контрактів на базі блокчейну Ethereum. Воно складається з кількох компонентів для редагування, компіляції, налагодження та розгортання смарт-контрактів і децентралізованих додатків (dApps), які разом утворюють повноцінне середовище розробки.

Для того, щоб розпочати локальну розробку, мені знадобилось ввести наступні команди:

Лістинг 3.1 – Команди для запуску проекту у hardhat

```
npx hardhat init
npx hardhat clean
npx hardhat compile
npx hardhat node
```

Після запуску цих команд, середовище розробки запустило JSON-RPC сервер для взаємодії з середовищем розробки та створило 20 гаманців з 10 тис. одиниць Ethereum на кожному. Ці гаманці будуть у подальшому використовуватись з метою тестування функціоналу.

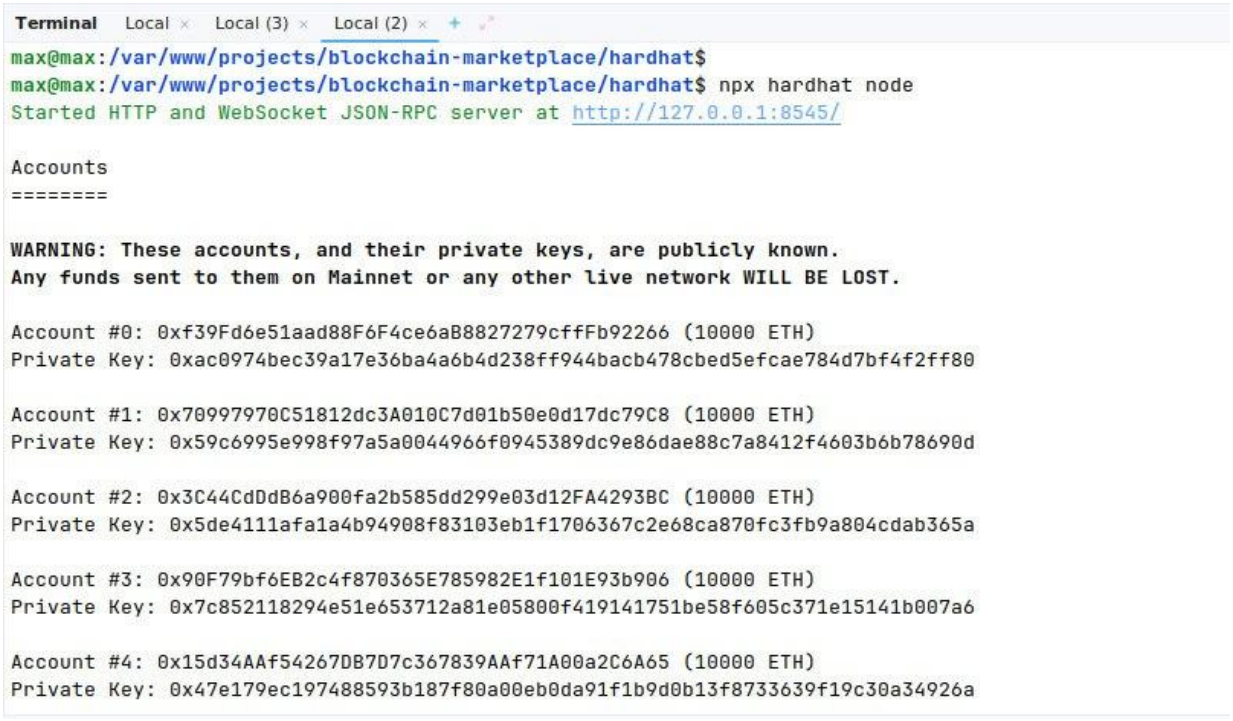

Рисунок 3.1 – Локальна адреса середовища розробки та ключі гаманців

Важливо також згадати, що мережа Hardhat автоматично обробляє майнінг: за замовчуванням після кожної транзакції він одразу створює новий блок. Це означає, що розробнику не доводиться вручну чекати декількох підтверджень блоку під час тестування. Процес розгортання та взаємодії зі смарт-контрактом буде розглянутий у наступних підрозділах

3.1.3 Розробка смарт-контракту мовою програмування Solidity

Solidity – об'єктно-орієнтована та предметно-орієнтована мова програмування для створення смарт-контрактів на платформі Ethereum. При розробці смарт-контракту були дотримані рекомендації щодо безпеки розробки [11], які включають:

- Застосування бібліотек OpenZeppelin для стандартних контрактів (Ownable, ERC-стандарти)
- Використання msg.sender замість tx.origin при перевірці авторизації та обмеження.
- Правильне призначення модифікаторів видимості для функцій

Смарт-контракт реалізує базову логіку обробки замовлення з урахуванням вимог безпеки до перевірки даних та переказу коштів (див. Додаток А). Його основна мета – перевірити, чи замовлення створене довіреною особою, та у разі успішної перевірки передати отриману оплату на адресу скарбниці. Ключова функція контракту – це placeOrder (див. лістинг 3.2)

Лістинг 3.2 – Тіло функції placeOrder

```
function placeOrder(
    bytes32 requestHash,
    uint256 price
) external payable {
    bytes32 hash = keccak256(abi.encode(trustedSigner, price));

    if (hash != requestHash) {
        revert ValidationFailed("Invalid hash");
    }

    (bool success, ) = treasury.call {
        value: price
    }("");

    if (success == false) {
        revert ValidationFailed("Invalid payment");
    }
}
```

Перш за все, у тілі функції сам контракт самостійно генерує еталонне значення хешу за допомогою функції keccak256. Ця криптографічна хеш-функція приймає адреси довіреного підписника та переданої суми замовлення і повертає 32-байтний хеш. Функція заснована на алгоритмі Keccak (SHA-3), і в Ethereum її застосовують у багатьох внутрішніх механізмах, включаючи адресацію, підпис транзакцій, зберігання паролів, хешування структур тощо. Однією з важливих властивостей keccak256 є лавинний ефект (незначна зміна вхідних даних – значна зміна хешу). Завдяки цьому вона забезпечує надійний захист від підміни чи повторного використання даних. Щоб дані могли бути передані у keccak256, вони мають бути попередньо закодовані у бінарному форматі, який розуміє віртуальна машина Ethereum. Для цього застосовується функція abi.encode. Вона використовується для серіалізації (перетворення у байтовий вигляд) довільної кількості аргументів у формат, який відповідає

правилам ABI (Application Binary Interface) – тобто стандарту, за яким контракти взаємодіють між собою та ззовні.

Якщо хеші не співпадають, то транзакція миттєво скасовується. Цей метод забезпечує контроль цілісності інформації.

3.1.4 Взаємодія клієнтської частини зі смарт-контрактом при оплаті замовлення

Однією з ключових цілей системи є забезпечення безпечного способу оплати замовлення, а також захисту персональних даних користувача, які можуть супроводжувати транзакцію (див. Додаток Б). Бібліотека web3.js відіграє ключову роль у процесі взаємодії між клієнтським інтерфейсом та смарт-контрактом, розгорнутим у мережі Ethereum. Вона дає змогу веб-застосунку напряму спілкуватися з блокчейном, використовуючи функціональність, яку надає гаманець користувача, зокрема MetaMask. За її допомогою можна підключитися до мережі Ethereum, ініціалізувати смарт-контракт за його адресою та інтерфейсом (ABI), викликати його функції, відправляти транзакції і обробляти їх результати. Ця бібліотека забезпечує повноцінний міст між інтерфейсом та смарт-контрактом, дозволяючи виконувати всі необхідні операції у межах децентралізованої моделі без звернення до традиційного сервера.

Спершу, при оформлення замовлення, користувач вводить адресу доставки. Для здійснення оплати через смарт-контракт, нам потрібно взнати адресу гаманця користувача. Одним з найзручніших онлайн-гаманців є MetaMask. MetaMask – це криптовалютний гаманець, доступний у вигляді розширення для браузера та мобільного додатка. Він дозволяє зберігати, переказувати та взаємодіяти з криптовалютами, такими як Ethereum та токени, що базуються на Ethereum, включаючи NFT. MetaMask є некастодіальним гаманцем, тобто користувачі самостійно контролюють свої приватні ключі та кошти. На отримання адреси гаманця потрібно отримати дозвіл від користувача. Ми використовуємо функцію, зображену у лістингу 3.3, для того, щоб отримати адресу користувача.

Лістинг 3.3 – Код для отримання адреси гаманця користувача

```
    export async function getCurrentUserAddress(): Promise<string>
    {
        const result: Partial<string[]> | null | undefined = await
window.ethereum.request({ method: 'eth_requestAccounts' });

        if (!result || !result[0]) {
            throw new Error('No address found');
        }

        return result[0];
    }
```

Другим кроком, щоб забезпечити конфіденційність адреси доставки у середовищі децентралізованої інфраструктури, було реалізовано механізм асиметричного шифрування цієї інформації публічним ключем користувача, отриманим через MetaMask. Завдяки цьому тільки сам користувач і служба доставки зможуть розшифрувати ці дані – ніхто інший, включно з розробниками або власниками сервера, не матиме доступу до них. Задля забезпечення конфіденційності цієї інформації ми будемо використовувати асиметричний криптографічний алгоритм x25519-xsalsa20-poly1305, процес шифрування зображений у лістингу 3.4.

Лістинг 3.4 – Процес шифрування

```
    export   async   function   encryptDataWithMetamaskKey(address:
string, data: string) {
        const publicKey = await getEncryptionPublicKey(address);
        if (!publicKey) {
            throw new Error('No public key found');
        }

        const encrypted = encrypt({
            publicKey,
            data,
            version: 'x25519-xsalsa20-poly1305',
        });

        return '0x' + Buffer.from(JSON.stringify(encrypted), 'utf-
8').toString('hex');
    }
```

Цей підхід дозволяє безпечно шифрувати дані на за допомогою публічного ключа гаманця користувача і при цьому лише власник відповідного приватного

ключа зможе розшифрувати повідомлення. Алгоритм x25519-xsalsa20-poly1305 є комбінацією трьох криптографічних примітивів, що забезпечує конфіденційність, цілісність і достовірність даних:

- x25519 – алгоритм для обміну ключами на основі еліптичної кривої Curve25519. Він використовується для встановлення спільного секрету між відправником і одержувачем без потреби в симетричному ключі, що передається мережею. Це забезпечує приватність навіть у разі компрометації ключа в майбутньому.
- XSalsa20 – потужний і дуже швидкий потоковий шифр, який генерує псевдовипадковий потік байтів (key stream) на основі секретного ключа та nonce (одноразового числа). Його перевага – висока продуктивність і криптостійкість, навіть для великих обсягів даних.
- Poly1305 – алгоритм аутентифікації повідомлень (MAC), який додає до зашифрованих даних т.зв. тег автентичності. Це дозволяє перевірити, чи не були дані змінені або підроблені під час передавання.

На третьому етапі виконується формування даних для передачі у смарт-контракт. Зокрема, необхідно отримати список замовлених товарів, вартість замовлення та контрольну суму (хеш) замовлення. Ці дані отримуються з серверної функції (див. Додаток В). Контрольна сума обчислюється за допомогою криптографічної хеш-функції keccak256 на основі фіксованих параметрів: довірений підписник та сума замовлення.

Лістингу 3.5 – Створення контрольної суми

```
    const signer = await getAdminSigner(web3);
    const hash = web3.utils.keccak256(
        web3.eth.abi.encodeParameters(['address',       'uint256'],
[signer.address, data.grand_total.value])
    );
```

Останнім кроком ми ініціалізовуємо контракт для бібліотеки, та викликаємо сам контракт, передаючи йому публічну адресу гаманця користувача та робимо конвертацію USD в Ether за наступною формулою (ціна за одиницю ETH тут взята сталою:

$$Total_{ETH} = Total_{USD} \div 2548 \quad (3.1)$$

У випадку успішної транзакції, ми виводимо її хеш. Якщо сталася помилка – повідомляємо користувача про це.

3.2 Тестування вебдодатку

Спочатку користувач заходить на головну сторінку магазину, тут він може подивись на найпопулярніші товари у магазині.

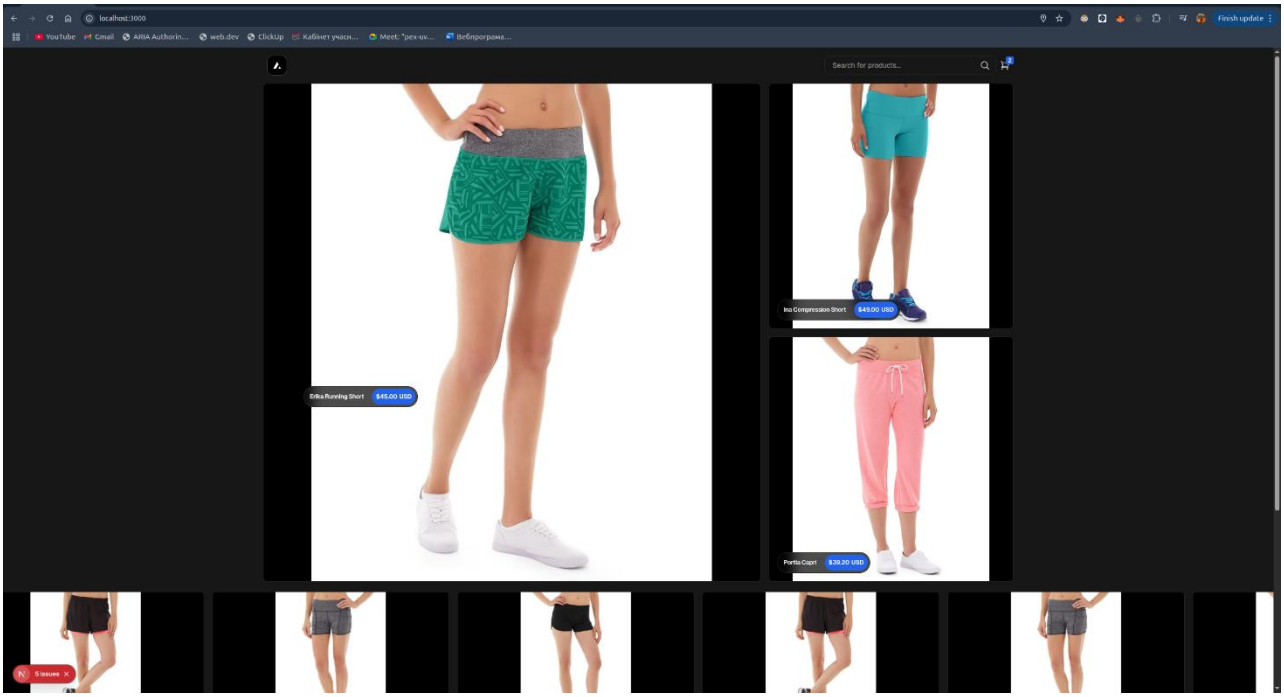

Рисунок 3.2 – Головна сторінка магазину

Далі користувач може обрати будь-який товар і натиснути на нього, перейшовши на сторінку самого товару. На цій сторінці користувач може вибрати необхідні опції товару (колір, розмір і т.п.), кількість товару. Щоб додати товару кошик, потрібно натиснути на кнопку “Add to Cart”.

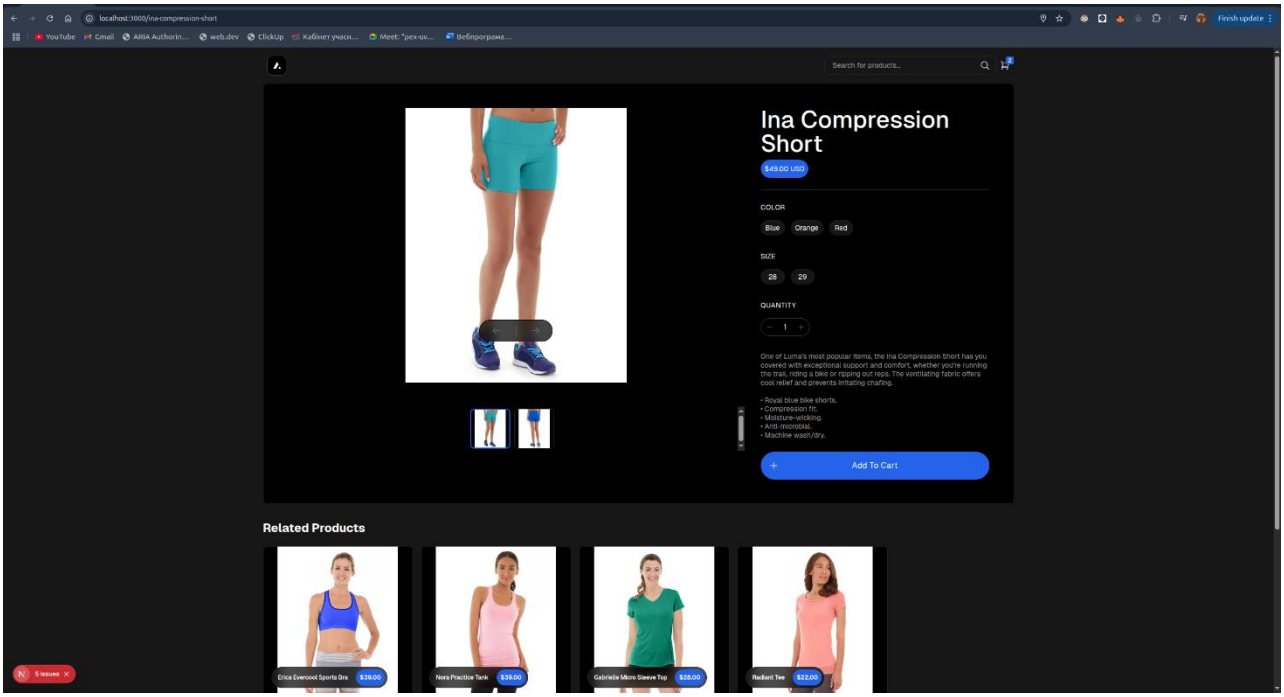

Рисунок 3.3 – Сторінка продукта, вибір опцій

У карті користувач може видалити попередньо додані продукти, подивитись на загальну ціну замовлених продуктів. Користувачу потрібно натиснути на кнопку “Proceed to Checkout”, щоб оформити замовлення.

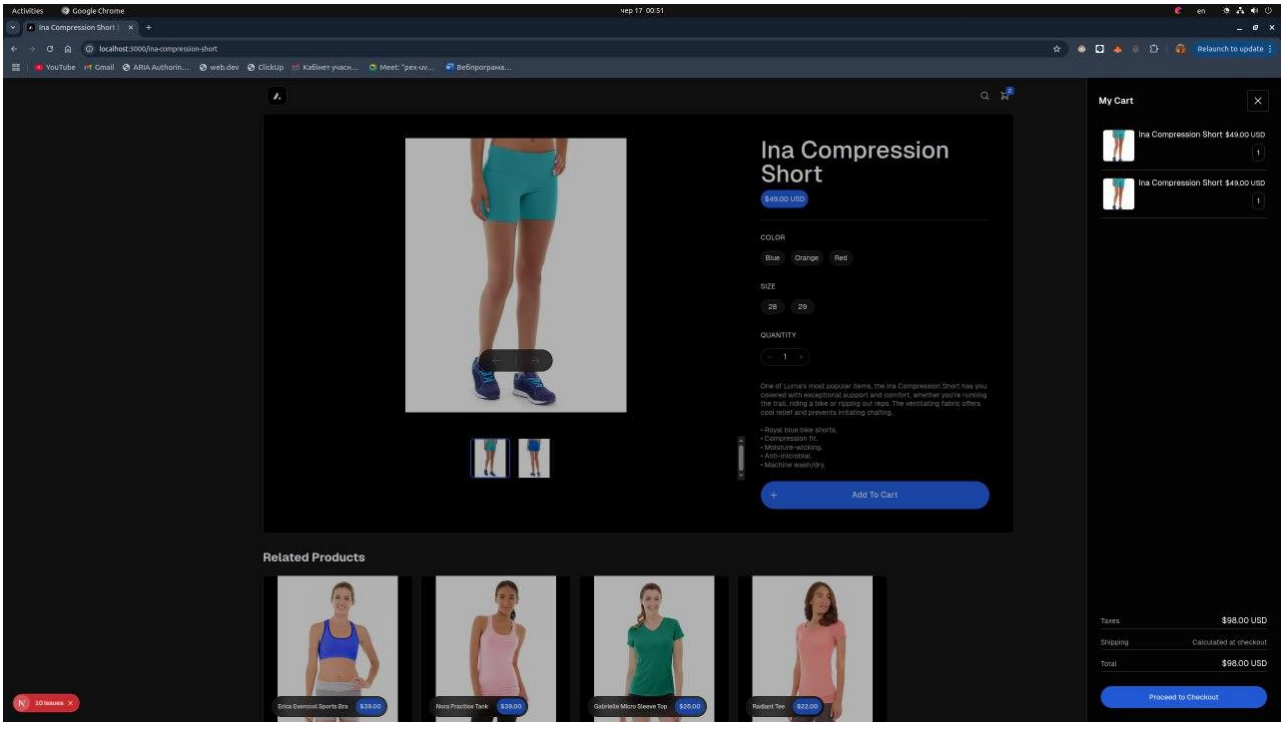

Рисунок 3.4 – Кошик та продукти у ньому

На цій сторінці користувачу треба перевірити всі деталі замовлення.

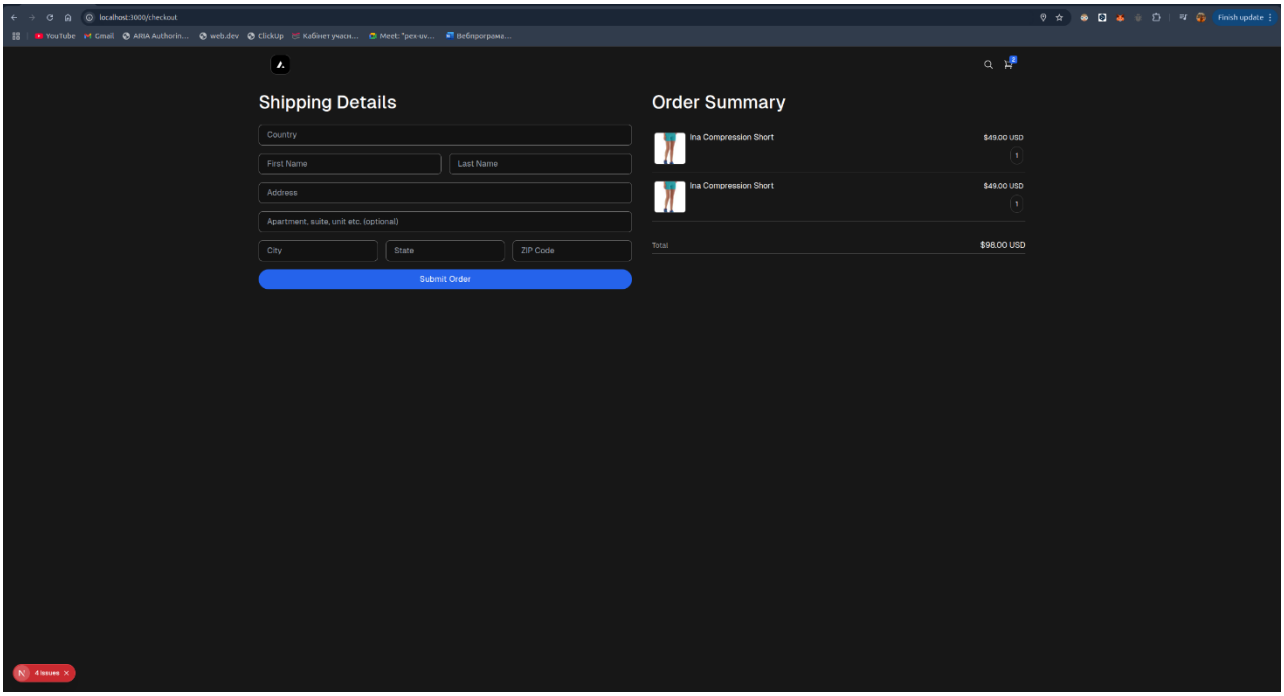

Рисунок 3.5 – Сторінка оплати

Після заповнення деталей замовлення, користувачу необхідно натиснути кнопку “Submit Order”.

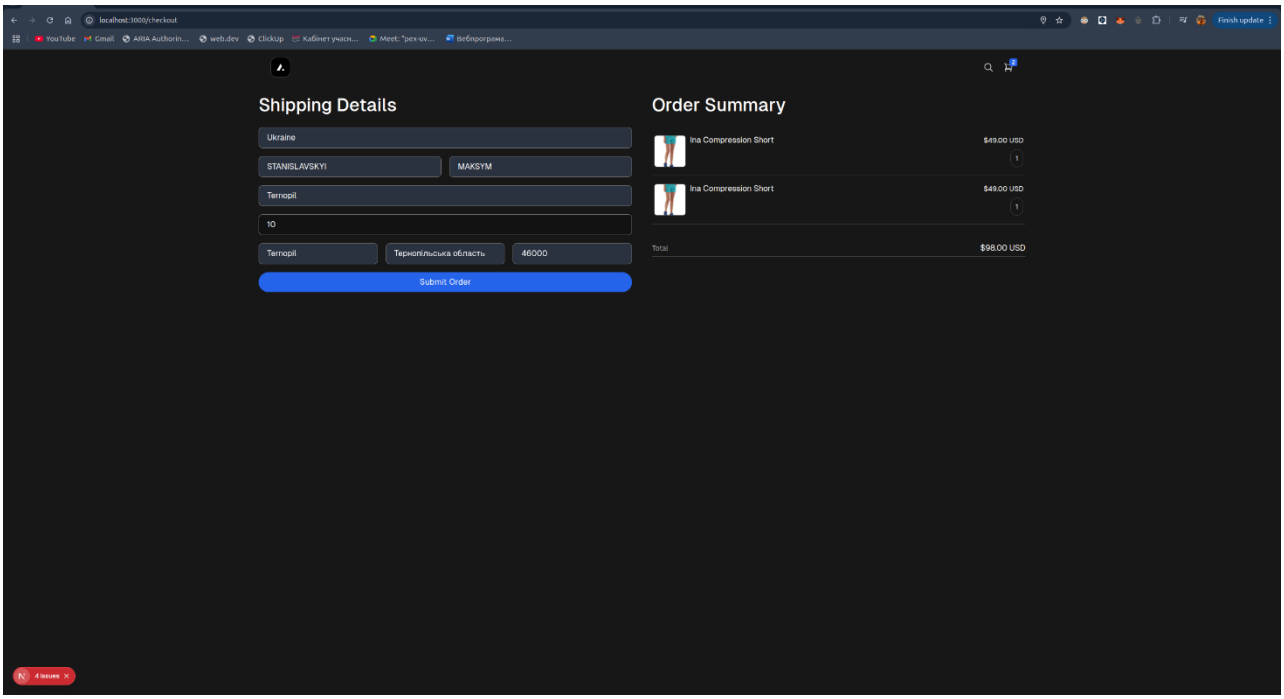

Рисунок 3.6 – Сторінка оплати з введеними деталями замовлення

Коли кнопка була натиснута, деталі заповлення повинні бути зашифрованими за допомогою публічного ключа гаманця. Для створення та

тестування гаманця, був вибраний криптогаманець MetaMask, у якого є також доволі широкий API для взаємодії зі смарт-контрактами, проведення транзакцій та ін. Для надання публічного ключа для шифрування, натискаємо кнопку “Provide”.

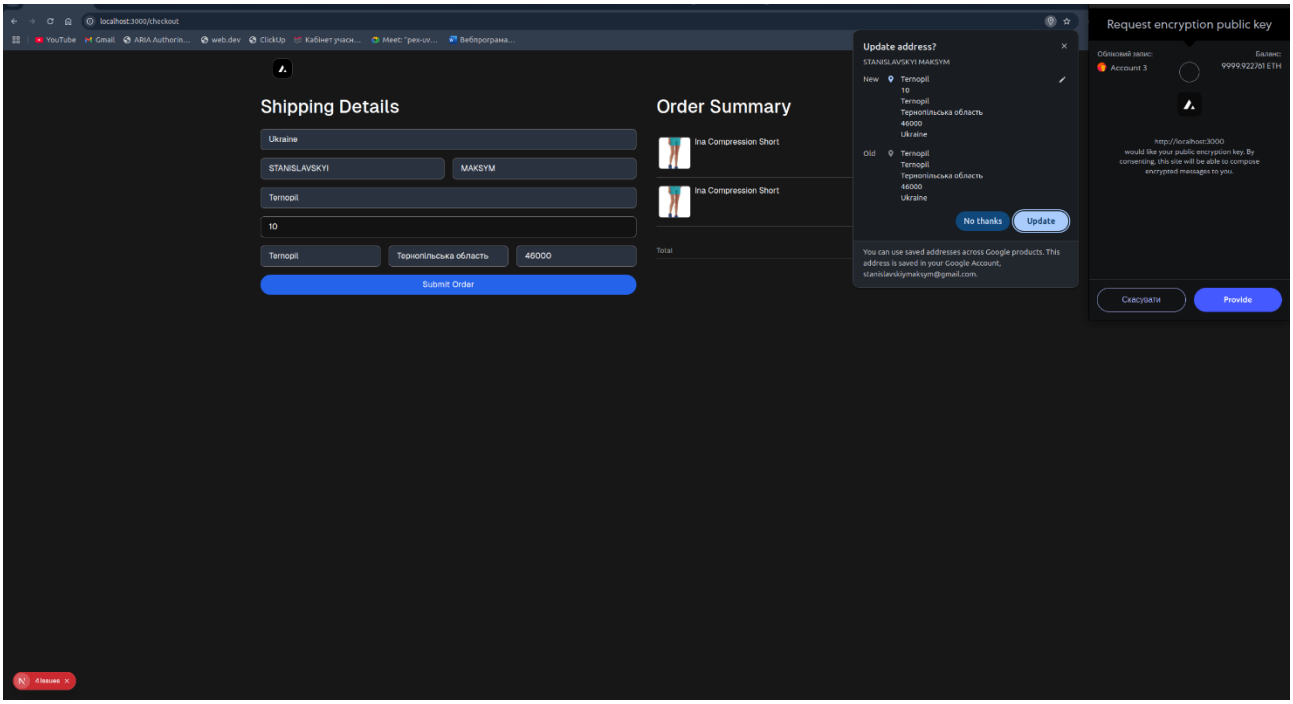

Рисунок 3.7 – Отримання публічного ключа гаманця через MetaMask

Після шифрування публічних даних, клієнтська сторона звертається до смарт-контракту для оплати замовлення, підтверджуємо його за допомогою кнопки “Підтвердити”.

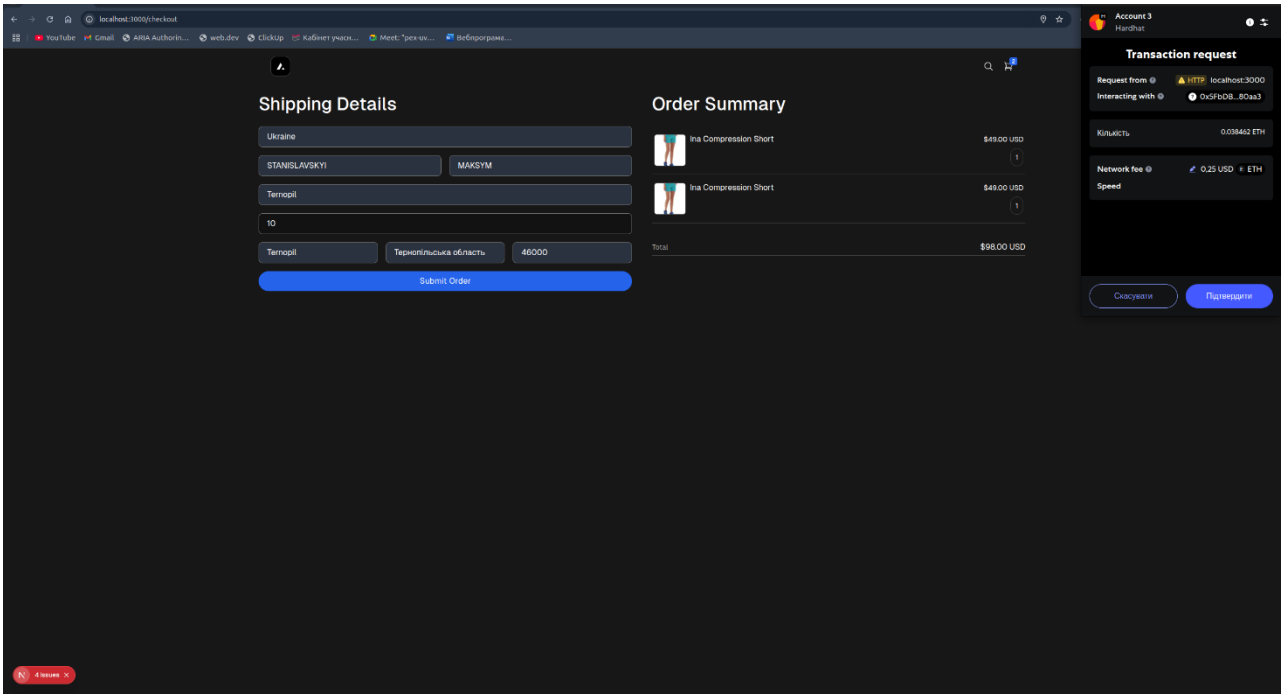

Рисунок 3.8 – Підтвердження транзакції для оплати товару

Після успішного виконання транзакції, ми отримаємо повідомлення з хешом транзакції.

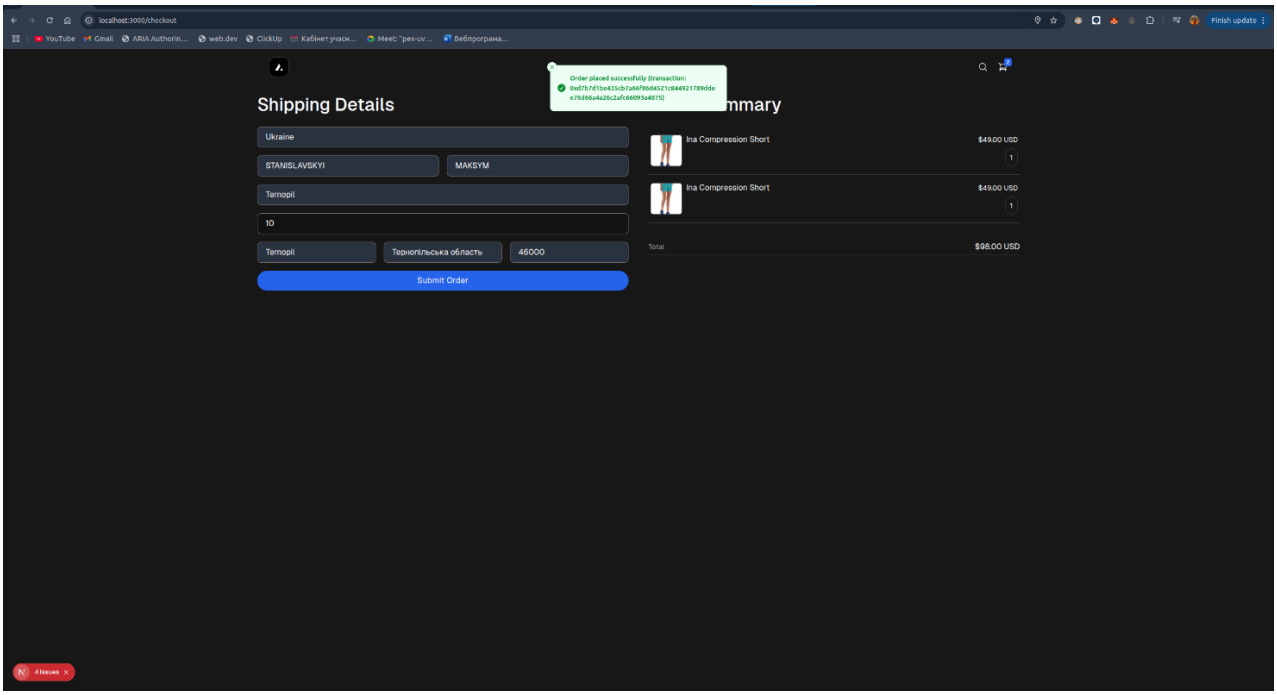

Рисунок 3.9 – Повідомлення про успішну оплату

Після успішної транзакції у розділі “Недавні” у додатку MetaMask з’явиться нова транзакція та баланс, відповідно, зміниться.

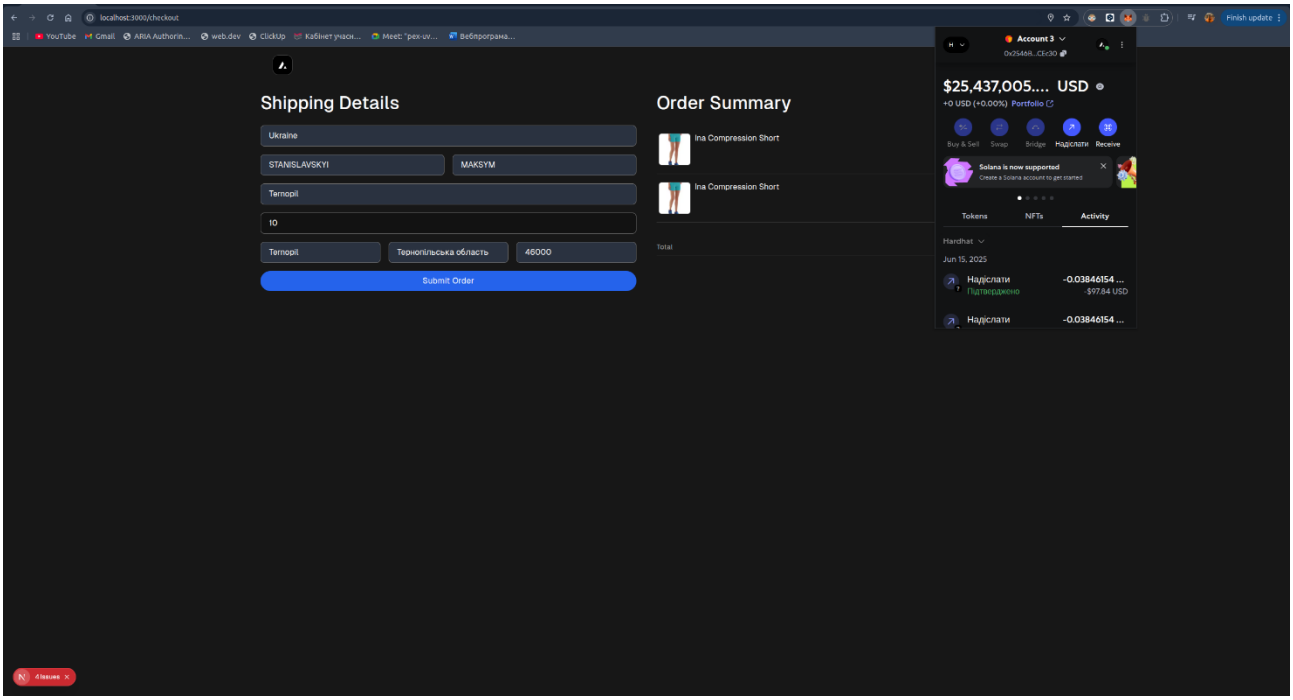

Рисунок 3.10 – Баланс гаманця, недавні проведені транзакції

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Безпека життєдіяльності при вибуху автозаправочної станції

Автозаправні станції є об'єктами підвищеної небезпеки, оскільки вони зберігають і реалізують легкозаймисті речовини, такі як бензин, дизельне паливо та скраплений газ. У разі виникнення аварійної ситуації, наприклад, порушення герметичності резервуарів, витоку парів пального або займання на території АЗС, може відбутися вибух, який несе серйозну загрозу для життя та здоров’я людей, що перебувають неподалік, зокрема в офісах, розташованих у межах вибухонебезпечної зони.

Під час вибуху формуються кілька небезпечних факторів одночасно: ударна хвиля, розліт уламків, пожежа та утворення токсичних продуктів згоряння. Ударна хвиля здатна спричинити механічне руйнування офісних будівель, вибити вікна, обвалити перегородки, завдати людям травм через уламки скла та конструкцій. Якщо внаслідок вибуху виникає пожежа, то температура в зоні займання досягає кількох сотень градусів, що робить перебування в приміщенні небезпечним через ризик опіків та задимлення. Важливою небезпекою є також токсичний дим, який у закритих просторах офісів швидко знижує концентрацію кисню, ускладнює дихання, викликає запаморочення, втрату свідомості й може призвести до отруєння.

Ситуацію ускладнює можливість паніки серед працівників, що призводить до неорганізованої евакуації, травмування та скупчення людей у вузьких проходах. Повторні вибухи через пошкодження інших резервуарів або газових систем на АЗС можуть стати додатковим фактором ризику для тих, хто не залишив небезпечну зону вчасно.

Запобігти загрозі або мінімізувати її наслідки можна лише за умови належної підготовки персоналу та обладнання офісного приміщення. Ще до виникнення надзвичайної ситуації мають бути розроблені плани евакуації, проведені навчання з безпеки, встановлені пожежні сповіщувачі та вогнегасники. Офісні працівники повинні бути поінформовані про розташування

аварійних виходів, алгоритм дій у разі вибуху та порядок надання першої медичної допомоги. Під час загрози або безпосередньо після вибуху важливо діяти швидко й злагоджено: залишити приміщення через найближчий безпечний вихід, попередити інших, не користуватись ліфтами, прикривати обличчя вологою тканиною в разі задимлення та прямувати до місця збору поза зоною ураження.

Після евакуації необхідно перевірити наявність усіх працівників і повідомити рятувальні служби про можливу наявність постраждалих або осіб, які залишились у будівлі. Повернення до приміщення можливе лише після дозволу відповідних аварійних служб.

Забезпечення життєдіяльності при вибуху автозаправної станції залежить від рівня підготовленості, організації простору та обізнаності працівників з правилами поведінки у надзвичайній ситуації. Комплексне поєднання інженерних, організаційних та освітніх заходів є запорукою зменшення людських втрат та матеріальних збитків під час таких інцидентів.

4.2 Роль робочого графіка і перерв у зниженні психоемоційного напруження

Організація робочого часу і правильне ергономічне облаштування праці суттєво впливають на психоемоційний стан працівників. Українське законодавство (КЗпП України) передбачає обідню перерву тривалістю до двох годин – не пізніше ніж через 4 години від початку зміни.

Часті короткі перерви протягом дня мають статистично доведений позитивний ефект: мікроперерви (до 10 хвилин) підвищують енергійність, знижують втому. Працівники, які мають змогу регулярно «перезавантажувати» увагу, демонструють кращу ефективність. Оптимально – 5–15 хвилин відпочинку щогодини.

Операторам комп’ютерів обов’язкові регламентовані перерви: 10–15 хвилин після кожної години або двох годин безперервної роботи за ПК. Ці норми

закріплені наказами Мінсоцполітики України. Вони спрямовані на профілактику втоми зору, хребта і психоемоційного вигорання.

Ергономіка праці передбачає хронометраж: аналіз часу, витраченого на різні типи діяльності. Це дозволяє оптимізувати графік, виявити найбільш навантажені періоди і правильно планувати фази відпочинку. При фізично важкій або монотонній роботі варто робити перерви кожні 30–60 хвилин.

Ігнорування перерв і тривала безперервна праця спричиняє зниження продуктивності, накопичення помилок і професійне вигорання – синдром, визнаний ВООЗ як професійне явище. Симптоми: емоційне виснаження, цинізм і низька результативність. Превентивні заходи – нормовані робочі зміни, гнучкий графік, психологічна підтримка, регулярний відпочинок.

Належна організація праці запобігає професійним захворюванням. За даними українських фахівців, до 70% остеохондрозу викликано сидячою роботою. Вчасні перерви, правильна постава і фізкультхвилинки суттєво зменшують ці ризики. Таким чином, ефективне управління часом праці та відпочинку – це не тільки гігієнічна вимога, а ключ до довготривалої продуктивності і здоров’я. Економія на перервах – це ілюзія ефективності, яка призводить до перевантаження, помилок, зниження продуктивності та зростання витрат на медичне обслуговування.

ВИСНОВКИ

У результаті проведеного дослідження було досягнуто поставлену мету – розроблено прототип безпечного онлайн-магазину на основі технології блокчейн (Ethereum). Робота починалася з аналізу актуальних загроз і вразливостей у галузі електронної комерції. Встановлено, що традиційні централізовані платформи електронної комерції мають суттєві вразливості, пов’язані з ризиком витоків персональних даних користувачів, несанкціонованими платежами, внутрішніми атаками та помилками людського фактора. Аналіз міжнародних досліджень підтвердив критичність проблеми, вказавши на масштабні фінансові втрати компаній від шахрайських дій.

Вирішення проблем безпеки запропоновано через інтеграцію децентралізованої технології блокчейн, зокрема платформи Ethereum, яка характеризується прозорістю, незмінністю та криптографічним захистом даних. Дослідження технології блокчейн та смарт-контрактів підтвердило її здатність ефективно захищати онлайн-транзакції та конфіденційні дані користувачів. Використання блокчейн-технології дозволяє знизити ризики, пов'язані з шахрайством та несанкціонованими змінами даних, завдяки особливостям роботи розподіленого реєстру та автоматизованого виконання смарт-контрактів.

Практична частина роботи полягала в реалізації діючого прототипу онлайн-магазину, який базується на інтеграції Next.js для фронтенду та Magento 2 для бекенд-інфраструктури. Для взаємодії з блокчейном була застосована бібліотека web3.js та криптогаманець MetaMask, що забезпечило криптографічне шифрування даних і транзакцій. Здійснено розробку та налаштування смарт-контрактів, які відповідають за безпечну обробку замовлень, верифікацію автентичності платежів і збереження незмінних даних у блокчейні. Це дозволяє уникнути підробок транзакцій та гарантувати цілісність фінансових операцій.
Тестування розробленого рішення підтвердило його працездатність та високу ефективність у забезпеченні кібербезпеки. Кожна транзакція, здійснена через розроблену систему, є прозорою та перевіряється всіма учасниками блокчейн-мережі, що значно знижує ймовірність шахрайських дій. Впровадження даного

рішення дозволяє забезпечити конфіденційність користувачів шляхом шифрування персональних даних публічним ключем гаманця користувача, що гарантує, що лише сам користувач може отримати доступ до власних даних.

Практичне значення проведеного дослідження полягає у можливості інтеграції запропонованих технологій у реальні комерційні рішення. Це дозволить компаніям суттєво підвищити рівень безпеки своїх систем електронної комерції, знизити ризики шахрайських дій і витоків інформації, а також підвищити довіру користувачів завдяки прозорості й незмінності транзакцій. Запропонована модель може бути масштабована та адаптована під конкретні бізнес-потреби, що робить її перспективною для подальшого розвитку та впровадження на практиці.

Отже, у ході роботи було продемонстровано, що використання блокчейн-технологій і смарт-контрактів є ефективним засобом для вирішення ключових проблем безпеки електронної комерції. Результати дослідження свідчать про значні переваги такого підходу перед традиційними централізованими рішеннями. Таким чином, можна рекомендувати розглянуту технологію як перспективний напрям для подальших досліджень та впровадження в комерційну діяльність компаній, зацікавлених у підвищенні рівня кібербезпеки своїх платформ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ecommerce fraud trends and statistics merchants need to know in 2024. *Mastercard. URL:* https://b2b.mastercard.com/news-and-insights/blog/ecommerce-fraud-trends-and-statistics-merchants-need-to-know-in-2024

2. Zagorodna N., Stadnyk M., Lypa B., Gavrylov M., Kozak R. Network Attack Detection Using Machine Learning Methods //Challenges to National Defence in Contemporary Geopolitical Situation, 2022, 2022(1), pp. 55–61

3. Lechachenko, T., Kozak, R., Skorenkyy, Y., Kramar, O., & Karelina, O. (2023). Cybersecurity aspects of smart manufacturing transition to Industry 5.0 model. CEUR Workshop Proceedings, 3628, 325-329. Proceedings of the 3rd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2023), Ternopil, Ukraine, November 22-24, 2023.

4. Skorenkyy, Y., Zolotyy, R., Fedak, S., Kramar, O., & Kozak, R. (2023). Digital twin implementation in transition of smart manufacturing to Industry 5.0 practices. Proceedings of the 1st International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2023), Ternopil, Ukraine, June 14-16, 2023, 12-23.

5. Revniuk, O. A., Zagorodna, N. V., Kozak, R. O., Karpinski, M. P., & Flud, L. O. (2024). The improvement of web-application SDL process to prevent insecure design vulnerabilities. Applied Aspects of Information Technology, 7(2), 162–174.

6. Card fraud in Europe declines significantly. *European Central Bank*. URL: https://www.ecb.europa.eu/press/pr/date/2023/html/ecb.pr230526~f09bc3c664.en.html

7. KSI blockchain. *e-estonia.com*. URL: https://e-estonia.com/solutions/cyber-security/ksi-blockchai.

8. Ethereum vs Bitcoin: What’s the Difference?. *Cryptohopper*. URL: https://www.cryptohopper.com/news/ethereum-vs-bitcoin-what-s-the-difference-8239

9. 6 Найкращих практик безпеки смарт-контрактів Solidity. *Alchemy.* URL: https://www.alchemy.com/overviews/smart-contract-security-best-practices

10. A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum Whitepaper. *Buterin*. URL: https://ethereum.org/en/whitepaper/

11. Introduction to Ethereum Smart Contracts. *ConsenSys.* URL: https://consensys.io/developers/docs/smart-contracts/

12. Smart Contract Security Best Practices. *Ethereum Foundation*. URL: https://ethereum.org/en/developers/docs/smart-contracts/security/

13. What Are Smart Contracts on Ethereum? *Binance Academy*. URL: https://academy.binance.com/en/articles/what-are-smart-contracts

14. Accounts, Transactions, and Gas. *Ethereum Foundation.* URL: https://ethereum.org/en/developers/docs/gas/

15. Smart Contract Security Top 10. *OWASP Foundation*. URL: https://owasp.org/www-project-smart-contract-security-top-10/

16. Revniuk O., Zagorodna N., Kozak R., Yavorskyy B. (2025) Development of an information system for the quantitative assessment of web application security based on the OWASP ASVS standard. Scientific Journal of TNTU (Tern.), vol 118, no 2, pp. 56–65.

17. Kotsiuba, I., Velykzhanin, A., Biloborodov, O., Skarga-Bandurova, I., Biloborodova, T., Yanovich, Y., & Zhygulin, V. (2018, December). Blockchain evolution: from bitcoin to forensic in smart grids. In 2018 IEEE international conference on big data (big data) (pp. 3100-3106). IEEE.

18. Shevchuk, R., Lishchynskyy, I., Ciura, M., Lyzun, M., Kozak, R., & Kasianchuk, M. (2025). Application of Blockchain Technology in Emergency Management Systems: A Bibliometric Analysis. Applied Sciences, 15(10), 5405.

19. Babakov, R. M., et al. "Internet of Things for Industry and Human Application. Vol. 3." (2019): 1-917.

20. Охорона праці. / За ред. В.П.Кучерявого. – Львів: Оріяна – Нова, 2007.

21. Запорожець О.І. Безпека життєдіяльності. Підручник, 2-е видання, Центр учбової літератури, 2020. 448 с.

22. ДСТУ ISO 6309:2007 „Протипожежний захист. Знаки безпеки”.

23. Закон України „Про охорону праці” (від 21 листопада 2002 року, з змінами і доповненнями).

Додаток А Лістинг смарт-контракту

```
pragma solidity ^0.8.10;


import
"@openzeppelin/contracts/utils/cryptography/SignatureChecker.sol";
import
"@openzeppelin/contracts/utils/cryptography/MessageHashUtils.sol";
import "hardhat/console.sol";


contract OrderManager {
    error ValidationFailed(string message);


    address payable private treasury;
    address private trustedSigner;


    constructor(address _trustedSigner, address payable _treasury)
{
        trustedSigner = _trustedSigner;
        treasury = _treasury;
    }


    function placeOrder(
        bytes32 requestHash,
        uint256 price
    ) external payable {
        bytes32 hash = keccak256(abi.encode(trustedSigner,
price));


        if (hash != requestHash) {
            revert ValidationFailed("Invalid hash");
        }


        (bool success, ) = treasury.call{value: price}("");


        if (success == false) {
            revert ValidationFailed("Invalid payment");
```

```
            }
        }
    }
```

Додаток Б Лістинг коду для обробки замовлення на клієнтській частині

```
const handleSubmit = async (e: SyntheticEvent<HTMLFormElement>) =>
{
        e.preventDefault();

        const formData = new FormData(e.currentTarget);
        const data = Object.fromEntries(formData.entries());

        const address = await getCurrentUserAddress();

        const shippingData = await encryptDataWithMetamaskKey(
            address,
            JSON.stringify(data)
        );

        const contractData = await getContractData(shippingData);

        if (!contractData.data) {
            throw new Error('Contract data is not valid');
        }

        const contract = new
web3.eth.Contract(OrderManagerArtifact.abi,
"0x5FbDB2315678afecb367f032d93F642f64180aa3");

        try {
            // @ts-ignore
            const result = await contract.methods.placeOrder(
                contractData.data.messageHash,
                contractData.data.grand_total.value
            )
                .send({
                    from: address,
                    value: web3.utils.toWei(
                        contractData.data.grand_total.value /
2548,
```

```
                        'ether'
                    )
                });


            const receipt = await
web3.eth.getTransactionReceipt(result.transactionHash);


            toast.success(`Order placed successfully (transaction:
${receipt.transactionHash})`);
        } catch (error: any) {
            toast.error('Something went wrong during the
transaction process')
        }
    }
```

Додаток В Лістинг коду для отримання даних замовлення

```
export async function getContractData(shippingData: string):
Promise<{ data?: ContactData, message?: string }> {
    const web3 = new Web3(process.env.HARDHAT_ENDPOINT);

    const cookieFiles = await cookies();
    const cartId = cookieFiles.get('cart_id');

    if (!cartId?.value) {
        return {
            data: undefined,
            message: 'Cart ID not found'
        }
    }

    const cart = await getCartById(cartId.value);

    if (!cart) {
        return {
            data: undefined,
            message: 'Cart has expired'
        }
    }

    const data = {
        shippingData,
        grand_total: cart.prices.grand_total,
        items: cart.itemsV2.items.map(item => ({
            id: item.id,
            name: item.product.name,
            quantity: item.quantity
        }))
    }

    const signer = await getAdminSigner(web3);
```

```
  const hash = web3.utils.keccak256(
        web3.eth.abi.encodeParameters(['address', 'uint256'],
[signer.address, data.grand_total.value])
    );

    return {
        data: {
            ...data,
            messageHash: hash
        },
        message: undefined
    }
}
```