Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» грудня 2024 р.

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**Модель криптографічного захисту даних користувачів у web-застосунках**»

Спеціальність 123 – Комп'ютерна інженерія.
Галузь знань: 12 – Інформаційні технології.
Освітня програма «Комп'ютерна інженерія».

Захищено на засіданні
Екзаменаційної комісії № 42
протокол № __ від __.12.2024 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
Студент групи КІ– 61
Веремієнко Богдан Олександрович_____

Керівник:
к.е.н, доцент закладу вищої освіти
Чуб Ольга Ігорівна_____

Рецензент: д.т.н, професор кафедри теоретичної та прикладної інформатики (Харківського національного університету імені В.Н. Каразіна)
ФРОЛОВ В'ячеслав Вікторович_____

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи магістра складається зі вступу, п’ятьох розділів, висновків, списку використаних джерел. Загальний обсяг роботи складає 69 сторінок, з них основного тексту 58 сторінок, містить 28 рисунків, 3 таблиці та 27 найменувань за списком використаних джерел.

Кваліфікаційна робота магістра присвячена дослідженню та реалізації моделі криптографічного захисту даних у веб-застосунках із використанням технології блокчейн. **Мета роботи** – аналіз сучасних криптографічних методів, особливостей інтеграції блокчейну для забезпечення безпеки даних та розробка децентралізованого веб-застосунку з авторизацією через криптовалютний гаманець.

Об’єктом дослідження є методи криптографічного захисту даних у веб-застосунках. **Предметом дослідження** виступає інтеграція блокчейну та смарт-контрактів для забезпечення надійності й анонімності користувачів у децентралізованих системах.

Результати роботи підтвердили доцільність використання блокчейну для створення безпечних, анонімних та прозорих веб-застосунків. Запропоновані рішення можуть знайти застосування у системах захисту даних, електронній комерції та децентралізованих платформах.

Ключові слова: криптографія, захист даних, блокчейн, смарт-контракти, децентралізація, криптогаманець, веб-застосунок, анонімність, безпека транзакцій, децентралізовані платформи, алгоритм консенсусу.

ABSTRACT

The explanatory note to the master's qualification work consists of an introduction, five chapters, conclusions, and a list of references. The total volume of the work is 60 pages, including 28 figures, 3 tables, and 27 references.

The master's thesis is dedicated to the study and implementation of a cryptographic data protection model in web applications using blockchain technology. **The purpose of the work** is to analyze modern cryptographic methods, explore the features of blockchain integration for data security, and develop a decentralized web application with cryptocurrency wallet-based authorization.

The object of the study is cryptographic data protection methods in web applications. **The subject of the study** is the integration of blockchain and smart contracts to ensure user reliability and anonymity in decentralized systems.

The results of the study confirmed the feasibility of using blockchain to create secure, anonymous, and transparent web applications. The proposed solutions can be applied in data protection systems, e-commerce, and decentralized platforms.

Keywords: cryptography, data protection, blockchain, smart contracts, decentralization, crypto wallet, web application, anonymity, transaction security, decentralized platforms, consensus algorithm.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

UI	–	User Interface, інтерфейс користувача;
UX	–	User Experience, досвід користувача;
IPFS	–	InterPlanetary File System, децентралізована система зберігання файлів;
HTTPS	–	HyperText Transfer Protocol Secure, протокол захищеної передачі даних;
DApps	–	Decentralized Application, децентралізований застосунок;
NFT	–	Non-Fungible Token, унікальний цифровий актив;
SHA	–	Secure Hash Algorithm, алгоритм гешування;
AES	–	Advanced Encryption Standard, алгоритм симетричного шифрування;
RSA	–	Rivest–Shamir–Adleman, алгоритм асиметричного шифрування;

ВСТУП

У сучасному світі захист інформації у веб-застосунках стає дедалі важливішим через зростання цифровізації, використання хмарних технологій і збільшення кількості кіберзагроз. Особливої актуальності набуває застосування криптографії як основного інструменту для забезпечення конфіденційності, цілісності та автентичності даних. Серед сучасних технологій криптографічного захисту особливе місце займає блокчейн, що завдяки своїй децентралізованій природі та прозорості дозволяє підвищити рівень безпеки даних у веб-середовищі.

Актуальність теми обумовлена необхідністю створення надійних моделей захисту даних у веб-застосунках, що поєднують ефективність криптографічних методів та переваги блокчейн-технології. Використання блокчейну для зберігання, управління доступом і забезпечення транзакційної безпеки пропонує нові перспективи в галузі інформаційної безпеки.

Об’єктом дослідження є криптографічний захист даних у веб-застосунках. Предметом дослідження виступає інтеграція блокчейн-технології для забезпечення високого рівня захисту даних.

Метою роботи є розробка моделі криптографічного захисту даних у веб-застосунках із використанням блокчейн-технології, що забезпечить підвищену безпеку даних та мінімізацію ризиків.

Для досягнення мети були поставлені такі завдання:

1. Дослідити теоретичні основи криптографічного захисту даних, включаючи класифікацію методів та сучасні тенденції у криптографії.
2. Провести аналіз криптографічних методів, що використовуються у веб-застосунках, з акцентом на їх ефективність у централізованих та децентралізованих системах.
3. Розглянути технологію блокчейн як інструмент для підвищення рівня захисту даних та дослідити її ключові компоненти, такі як смарт-контракти та криптовалютні гаманці.

4. Розробити практичний веб-застосунок, що інтегрує блокчейн для забезпечення безпеки даних, та продемонструвати його функціональні можливості.

Наукова новизна роботи полягає у використанні блокчейну для побудови моделі захисту даних у веб-застосунках, яка поєднує переваги децентралізованих систем та традиційних криптографічних методів. Практична значущість дослідження полягає в можливості впровадження розробленого рішення в реальні системи для забезпечення високого рівня захисту даних користувачів.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ КРИПТОГРАФІЧНОГО ЗАХИСТУ ДАНИХ

1.1 Основні поняття та принципи

Криптографія є фундаментальною дисципліною в інформаційній безпеці, яка забезпечує конфіденційність, цілісність та достовірність даних. Основна мета криптографії — захист інформації від несанкціонованого доступу шляхом її перетворення у вигляд, який стає незрозумілим для сторонніх осіб. Таке перетворення відбувається за допомогою криптографічних алгоритмів, що використовують спеціальні ключі для шифрування та дешифрування повідомлень.

Перш за все, основні поняття криптографії включають в себе терміни «шифрування», «дешифрування», «ключ» і «криптографічний алгоритм». Шифрування — це процес перетворення відкритого тексту в зашифрований, що називається шифротекстом. Зворотний процес — дешифрування — полягає у відновленні відкритого тексту з шифротексту за допомогою криптографічного ключа. Ключі, які можуть бути симетричними (однаковими для шифрування і дешифрування) або асиметричними (відрізняються для обох операцій), є центральними елементами криптографічних методів, оскільки саме вони визначають рівень захищеності інформації.

Існує два основних принципи криптографії: симетричне та асиметричне шифрування. У симетричному шифруванні для шифрування і дешифрування використовується один і той самий ключ, що вимагає його безпечного обміну між сторонами. Приклади таких алгоритмів включають DES, AES і Blowfish. Асиметричне шифрування передбачає використання пари ключів: відкритого (public) і закритого (private). Відкритий ключ використовується для шифрування даних, а закритий — для дешифрування, що вирішує проблему обміну ключами

та дозволяє використовувати відкритий ключ як загальнодоступний. RSA та алгоритм Ель-Гамаля є популярними прикладами асиметричних методів.

На додаток до основних методів шифрування, сучасна криптографія включає такі важливі поняття, як хешування, електронний цифровий підпис і криптографія з відкритим ключем. Хешування — це процес обчислення унікального «відбитка» для повідомлення, що дозволяє перевірити його цілісність, але не дозволяє відновити оригінальне повідомлення. Алгоритми хешування, такі як SHA-256 і MD5, використовуються для забезпечення перевірки даних і зменшення ризику несанкціонованої зміни [11].

Електронний цифровий підпис (ЕЦП) служить для підтвердження автентичності й цілісності документа, а також для ідентифікації автора. Це дозволяє уникати фальсифікації та гарантує, що повідомлення походить саме від особи, яка його підписала. ЕЦП зазвичай базується на алгоритмах асиметричного шифрування: автор використовує свій закритий ключ для підписання документа, а отримувач може перевірити підпис, використовуючи відкритий ключ автора.

Одним з важливих принципів сучасної криптографії є також концепція стійкості алгоритмів до атак. Застосування криптографічного алгоритму повинно гарантувати, що будь-яка спроба розкрити ключ або шифротекст потребує надмірних ресурсів і часу. Принцип стійкості ґрунтується на припущенні, що обчислювальна складність злому перевищує доступні потужності зловмисника, навіть з урахуванням сучасних технологій [21].

Таким чином, основні поняття та принципи криптографії, зокрема шифрування, дешифрування, ключі, симетричне і асиметричне шифрування, хешування та електронний цифровий підпис, становлять основу систем криптографічного захисту даних. Ці принципи, застосовувані з належною обережністю, забезпечують високий рівень безпеки інформації та дозволяють захищати її від несанкціонованого доступу, маніпуляцій та фальсифікацій.

1.2 Класифікація криптографічних методів

Криптографічні методи є основним інструментом забезпечення захисту інформації в сучасному цифровому середовищі. Їхня класифікація базується на різних підходах до шифрування, управління ключами, а також на завданнях, які вони вирішують. Загальна класифікація криптографічних алгоритмів наведена на рисунку 1.1.

Рисунок 1.1 — Класифікація криптографічних алгоритмів

Залежно від принципу дії, криптографічні методи можна розділити на кілька основних категорій: безключові (хеш-функції), одноключові (симетричне шифрування) та двоключові (асиметричне шифрування і електронний підпис). Розглянемо ці методи детальніше [22, 24].

Безключові методи не потребують використання ключів для шифрування або дешифрування даних. Основним представником цієї категорії є хеш-функції. Хеш-функції виконують перетворення вхідних даних довільної довжини у

вихідні дані фіксованого розміру, які називаються дайджестом або хешем. Основна ідея таких методів полягає в тому, щоб забезпечити однозначну відповідність між вхідними даними та їхнім дайджестом. Наприклад, навіть незначна зміна вхідного повідомлення призводить до повної зміни хешу.

Ключові властивості хеш-функцій:

- Односторонність. Неможливо відновити початкові дані, знаючи тільки хеш.
- Стійкість до колізій. Надзвичайно важко знайти два різних повідомлення, які мають однаковий хеш.
- Швидкодія. Хешування виконується швидко навіть для великих обсягів даних.

Популярні приклади хеш-функцій: MD5, SHA-1, SHA-256 та SHA-3. Сфера застосування хеш-функцій включає перевірку цілісності файлів, зберігання паролів у базах даних, створення цифрових підписів та забезпечення роботи криптовалютних мереж, таких як Bitcoin. Наприклад, у блокчейні хеш-функції використовуються для обчислення ідентифікаторів блоків та створення ланцюжка блоків.

Симетричні методи шифрування базуються на використанні одного ключа для обох процесів: шифрування та дешифрування. Візуальне відображення наведено на рисунку 1.2. Ця категорія методів є однією з найстаріших і широко використовуваних у криптографії. Симетричні алгоритми можуть бути блочними або потоковими.

- Блочні алгоритми. Дані розбиваються на блоки фіксованого розміру, кожен з яких шифрується окремо. До блочних алгоритмів належать DES (Data Encryption Standard), AES (Advanced Encryption Standard) і Blowfish. Наприклад, AES є сучасним стандартом шифрування, який широко застосовується у мережевій безпеці (наприклад, у протоколі HTTPS), у файлових системах та в програмах для захисту даних.
- Потокові алгоритми. Дані шифруються по одному біту або байту в режимі реального часу. Потокове шифрування забезпечує високу швидкість

роботи, що робить його ідеальним для передачі потокового відео чи аудіо. Прикладом є алгоритм RC4, який раніше використовувався у мережевих протоколах WEP і SSL.

Рисунок 1.2 — Симетричне шифрування

Основним недоліком симетричних методів є проблема безпечного обміну ключами між сторонами, особливо в умовах відкритих мереж. Однак ці методи забезпечують високу швидкість шифрування та стійкість до багатьох атак. Алгоритми симетричного шифрування відзначаються високою швидкістю, низькими вимогами до обчислювальної потужності та мінімальним впливом на швидкість інтернету.

Асиметричні методи є більш сучасними і складними. Вони використовують пару ключів: відкритий та закритий ключ. Основна ідея полягає в тому, що дані, зашифровані відкритим ключем, можуть бути дешифровані тільки закритим ключем, увага на рисунок 1.3.

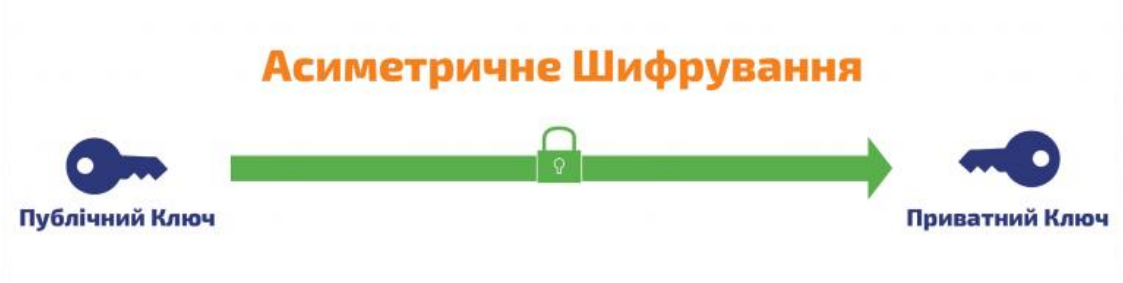

Рисунок 1.3 — Асиметричне шифрування

Асиметричне шифрування використовується для захисту конфіденційної інформації, зокрема під час обміну ключами для симетричних алгоритмів. Найвідомішими алгоритмами є RSA, DSA та ElGamal. Алгоритм RSA, наприклад, базується на труднощах факторизації великих чисел. Його застосування включає захищені електронні транзакції, електронну пошту та шифрування файлів.

Переваги асиметричного шифрування:

- Відсутність необхідності безпечної передачі ключа.
- Можливість створення цифрових підписів.

Недоліком є значно більша швидкість обчислень порівняно із симетричними методами.

Цифровий підпис є важливим засобом верифікації та забезпечення цілісності даних. Процес створення цифрового підпису включає хешування повідомлення та шифрування хешу закритим ключем. Отриманий підпис перевіряється за допомогою відкритого ключа. Процес підписання та процес верифікації наведено на рисунку 1.4.

Цифрові підписи активно використовуються в електронних документах, онлайн-банкінгу та блокчейнах. Наприклад, у криптовалютах цифрові підписи забезпечують підтвердження транзакцій без розкриття приватної інформації користувача.

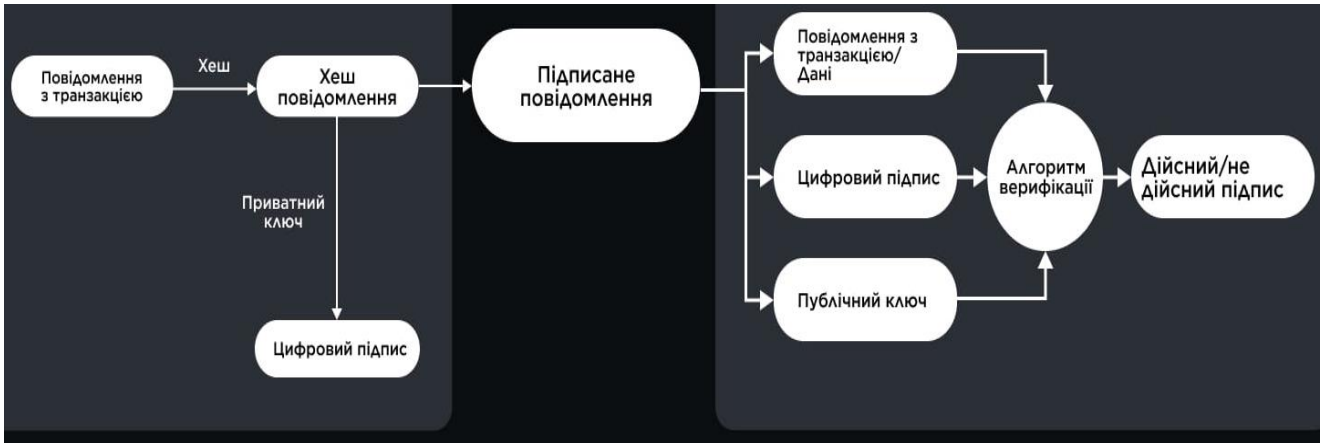

Рисунок 1.4 — Процес підписання та процес верифікації

1.3 Сучасні тенденції у криптографії

Криптографія є ключовим компонентом забезпечення безпеки даних у цифрову епоху. Сучасний розвиток технологій, зростання обчислювальних потужностей і поява нових загроз спонукають до швидкої еволюції криптографічних методів. Основні тенденції у цій галузі пов’язані з вдосконаленням існуючих алгоритмів, розробкою нових технологій та адаптацією до сучасних викликів, таких як квантові обчислення, кіберзлочинність і глобалізація цифрових комунікацій.

Однією з найважливіших тенденцій є розвиток квантової криптографії. Традиційні криптографічні алгоритми, такі як RSA або ECC (криптографія на еліптичних кривих), засновані на складності математичних задач, таких як факторизація великих чисел чи обчислення дискретного логарифма. Проте квантові комп’ютери здатні вирішувати ці задачі значно швидше, що може поставити під загрозу безпеку сучасних систем шифрування. У відповідь на це вчені розробляють квантово-стійкі алгоритми, які базуються на задачах, складність яких зберігається навіть за умов використання квантових обчислень. До таких методів належать, наприклад, криптографія на базі решіток, теорії кодування або многочленних функцій.

Ще однією ключовою тенденцією є поширення методів гомоморфного шифрування. Ця технологія дозволяє виконувати обчислення над зашифрованими даними без їх розшифрування. Це відкриває великі перспективи для забезпечення конфіденційності в хмарних обчисленнях, де дані часто передаються і зберігаються на віддалених серверах. Завдяки гомоморфному шифруванню користувачі можуть бути впевнені, що їхня інформація залишається захищеною навіть під час обробки.

Не менш важливою тенденцією є інтеграція криптографії з технологіями блокчейну. Блокчейн використовує асиметричне шифрування та хешування для забезпечення прозорості, автентичності та незмінності даних. Завдяки цьому, криптографія стала основою для створення децентралізованих фінансових

систем, цифрових активів і смарт-контрактів. Крім того, блокчейн пропонує нові можливості для розробки безпечних протоколів і систем ідентифікації.

Сучасні виклики також стимулюють удосконалення систем автентифікації. Паролі, як традиційний метод захисту, дедалі частіше замінюються на більш надійні способи, такі як біометрична автентифікація, багатофакторна перевірка та криптографічні токени. Наприклад, протоколи FIDO (Fast IDentity Online) використовують асиметричне шифрування для створення безпечного механізму ідентифікації без збереження паролів.

Зростання обсягу даних та підключених пристроїв у мережі також вимагає вдосконалення методів шифрування для Інтернету речей (IoT). Ці пристрої часто мають обмежені обчислювальні ресурси, тому актуальними є легковагові криптографічні алгоритми. Вони дозволяють забезпечити безпеку передачі даних, автентичність пристроїв та захист від атак, таких як "людина посередині" (Man-in-the-Middle). Окремої уваги заслуговує зростаюча роль машинного навчання у криптографії. З одного боку, алгоритми машинного навчання застосовуються для виявлення уразливостей у криптографічних системах і прогнозування можливих атак. З іншого боку, вони використовуються для розробки адаптивних систем захисту, які здатні змінювати свої параметри залежно від поведінки загроз [27].

Сучасні тенденції у криптографії відображають постійний розвиток цієї галузі, спрямований на забезпечення надійного захисту даних у мінливих умовах цифрового світу. Квантова криптографія, гомоморфне шифрування, інтеграція з блокчейном, легковагові алгоритми для IoT та машинне навчання — це лише частина тих інновацій, які формують майбутнє криптографії та гарантують її відповідність сучасним викликам.

Висновки за розділом 1

У першому розділі було розглянуто теоретичні аспекти криптографічного захисту даних, що є основою для розуміння механізмів, які забезпечують

конфіденційність, цілісність та автентичність інформації в сучасному цифровому середовищі. Детальний аналіз основних понять, класифікації криптографічних методів та сучасних тенденцій у цій сфері дозволив сформувати цілісне уявлення про поточний стан і перспективи розвитку криптографії.

Було визначено, що криптографія є наукою про шифрування даних, яка включає методи захисту інформації за допомогою математичних алгоритмів. Основними функціями криптографії є забезпечення конфіденційності, цілісності, автентичності та незаперечності даних. У цьому контексті підкреслюється важливість використання ключів, які є основним елементом криптографічних систем. Особливу увагу було приділено концепціям симетричного та асиметричного шифрування, які є базовими для більшості сучасних криптографічних методів.

Також розглянуто класифікацію криптографічних алгоритмів. Основними групами є безключові методи (хешування), одноключові методи (симетричне шифрування) та двоключові методи (асиметричне шифрування і електронний підпис). Кожен із цих підходів має свої переваги та обмеження, які роблять їх придатними для різних завдань. Наприклад, симетричне шифрування забезпечує високу швидкість роботи, що робить його ідеальним для захисту великих обсягів даних, тоді як асиметричні методи є незамінними для забезпечення безпечної передачі ключів. Також було акцентовано увагу на електронному підписі, який забезпечує автентичність і цілісність електронних документів.

Досліджено сучасні тенденції у криптографії, які відповідають викликам сьогодення. Зокрема, розвиток квантової криптографії спрямований на протидію загрозам, що виникають із появою квантових комп’ютерів. Гомоморфне шифрування відкриває нові можливості для обробки зашифрованих даних у хмарних системах, забезпечуючи при цьому конфіденційність. Інтеграція криптографії з блокчейном сприяє розвитку децентралізованих систем, які є надійними та прозорими. Легковагові алгоритми для Інтернету речей дозволяють забезпечити безпеку даних у пристроях із обмеженими ресурсами.

Важливу роль відіграє також застосування машинного навчання для виявлення уразливостей та розробки адаптивних захисних систем.

Таким чином, криптографія сьогодні є не лише важливим елементом безпеки інформаційних систем, але й сферою, яка активно розвивається, пристосовуючись до нових викликів і технологічних змін. Отримані знання та класифікація криптографічних методів стануть основою для подальшого дослідження та реалізації практичних рішень у галузі захисту даних. Розуміння сучасних тенденцій дозволяє спрогнозувати напрями розвитку криптографії та оцінити її перспективи в умовах цифрової трансформації суспільства.

РОЗДІЛ 2
АНАЛІЗ КРИПТОГРАФІЧНИХ МЕТОДІВ У WEB-ЗАСТОСУНКАХ

2.1 Вимоги до захисту інформації у веб-середовищі

У сучасному світі веб-застосунки відіграють ключову роль у багатьох аспектах людського життя, включаючи комунікацію, фінансові операції, обмін даними та доступ до сервісів. Разом із цим зростає важливість забезпечення безпеки інформації, яка обробляється, зберігається та передається у веб-середовищі.

Однією з ключових вимог до безпеки веб-застосунків є забезпечення конфіденційності даних. Це означає, що інформація, яка передається між клієнтом і сервером або зберігається на стороні сервера, має бути захищена від несанкціонованого доступу. Для досягнення цього використовуються методи шифрування, що забезпечують, аби навіть у випадку перехоплення дані залишалися недоступними для сторонніх осіб. Наприклад, використання протоколу HTTPS, який забражено на рисунку 2.1.

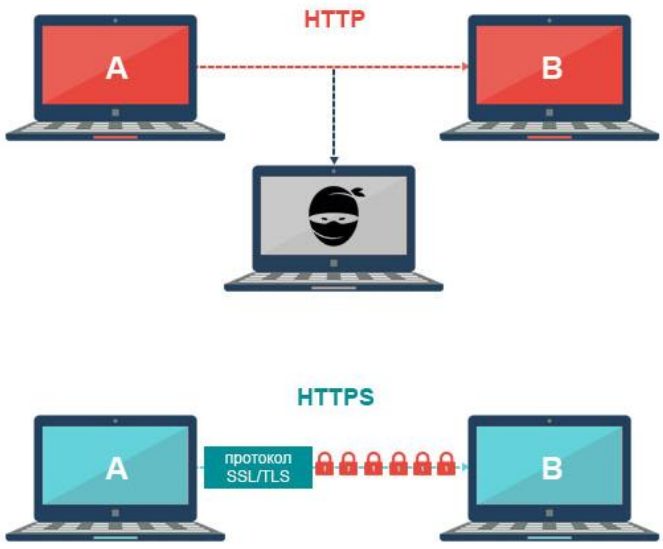

Рисунок 2.1 — Принцип роботи протоколу HTTPS

Цей протокол базується на SSL/TLS-шифруванні, стало стандартом для захисту даних у веб-трафіку. Вимоги до захисту інформації у веб-застосунках визначаються сучасними кіберзагрозами, правовими стандартами та зростаючими очікуваннями користувачів щодо конфіденційності їхніх даних. Наступною важливою вимогою є цілісність даних, що передбачає захист від їхньої модифікації. У веб-застосунках цілісність забезпечується використанням криптографічних хеш-функцій, які генерують унікальний цифровий відбиток даних. Цей відбиток порівнюється на стороні отримувача, що дозволяє визначити, чи були дані змінені під час передачі. Застосування цього підходу особливо важливе для фінансових транзакцій, де будь-яка зміна даних може призвести до значних збитків.

Третьою важливою вимогою є аутентифікація користувачів та систем. А саме важливо розуміти відмінності між ідентифікацією, аутентифікацією та авторизацією. Дуже добре пояснення наведено на рисунку 2.2.

Рисунок 2.2 — Відмінності між ідентифікацією, аутентифікацією та авторизацією

У веб-середовищі необхідно переконатися, що користувачі є саме тими, за кого себе видають, а також, що сервери, до яких вони звертаються, є надійними.

Для цього використовуються методи асиметричного шифрування, такі як сертифікати цифрового підпису, що дозволяють перевірити автентичність серверів. Крім того, для користувачів застосовуються багатофакторні методи автентифікації, які поєднують паролі, біометричні дані та одноразові коди доступу [3, 13, 14].

Протидія відмовам у доступі також є важливою складовою безпеки веб-застосунків. Атаки, спрямовані на порушення доступності сервісу, такі як DDoS-атаки, можуть завдати значної шкоди як бізнесу, так і користувачам. Для захисту від таких загроз застосовуються спеціалізовані сервіси та алгоритми, які розподіляють навантаження або блокують шкідливий трафік.

Окремо варто підкреслити вимогу до прозорості та відповідності стандартам. У сучасному світі багато країн і регіонів ухвалюють закони, що регулюють зберігання і обробку персональних даних, такі як GDPR у Європейському Союзі. Це вимагає від розробників веб-застосунків дотримання чітких вимог щодо обробки, зберігання та передачі даних. Застосування криптографічних методів є необхідною умовою для забезпечення відповідності таким регламентам.

Додатково, важливим аспектом є захист даних у стані спокою, тобто коли дані зберігаються на сервері чи пристрої. Для цього використовуються методи симетричного шифрування, які забезпечують збереження інформації навіть у випадку фізичного доступу до носія даних. Наприклад, шифрування баз даних з використанням алгоритмів AES (Advanced Encryption Standard) є стандартною практикою для багатьох веб-сервісів.

Сучасні вимоги до захисту інформації у веб-середовищі також враховують виклики, пов’язані з мобільністю та хмарними обчисленнями. У хмарних системах дані часто зберігаються на віддалених серверах, що ускладнює їхній захист. У таких випадках використовуються технології кінцевого шифрування, які гарантують, що дані залишаються захищеними навіть від адміністраторів хмарного сервісу.

Також можна зазначити, що вимоги до захисту інформації у веб-застосунках є багатогранними та охоплюють широкий спектр заходів. Вони спрямовані на забезпечення конфіденційності, цілісності, автентичності та доступності інформації, а також на виконання правових і етичних норм. Усе це робить криптографію ключовим компонентом безпеки в сучасному веб-середовищі та основою для розвитку інноваційних рішень у сфері інформаційних технологій.

2.2 Відмінності криптографічного захисту у централізованих та децентралізованих системах

Криптографія є незамінним інструментом для забезпечення безпеки в різних типах систем. Однією з важливих проблем є різниця у підходах до криптографічного захисту в централізованих та децентралізованих системах. Обидва підходи мають свої особливості, переваги та недоліки, а також різні вимоги до безпеки та управління даними. Для кращого розуміння важливо розглянути, як організовані ці системи, і в чому полягають основні відмінності в їхньому криптографічному захисті.

Централізовані системи — це такі, у яких всі дані зберігаються на одному сервері або в межах одного централізованого середовища. У цьому випадку існує єдиний центр управління доступом, а користувачі підключаються до цього сервера для отримання послуг або ресурсів. Наприклад, це можуть бути традиційні веб-сервіси або хмарні платформи, де дані зберігаються на серверах провайдерів. У централізованих системах криптографічний захист часто реалізується через використання симетричного шифрування для захисту переданих даних між клієнтом і сервером, а також через асиметричне шифрування для автентифікації та забезпечення конфіденційності на рівні з'єднання (наприклад, SSL/TLS протоколи для HTTPS-з'єднань). Основною перевагою централізованої архітектури є контрольований доступ і можливість застосовувати комплексні методи моніторингу та управління безпекою. У таких

системах адміністратор або організація має повний контроль над управлінням криптографічними ключами та політиками безпеки. Наприклад, сервер може використовувати центральний ключ для шифрування всіх даних на стороні сервера. Це спрощує управління безпекою, оскільки всі ключі зберігаються в одному місці, і є можливість централізовано оновлювати або відмовляти доступ до них. Однак у централізованих системах існують і суттєві ризики. Основним з них є єдину точку відмови (single point of failure), тобто можливість повного припинення роботи системи у разі компрометації центрального сервера. Якщо хакер отримає доступ до централізованої бази даних, де зберігаються ключі або інша конфіденційна інформація, це може призвести до масштабних наслідків, таких як втрата цілісності даних або їх викрадення [16, 17].

Децентралізовані системи, на відміну від централізованих, не мають єдиного контролюючого органу чи сервера. Замість цього, дані зберігаються на численних учасниках мережі, кожен з яких має рівні права доступу до інформації. Прикладом таких систем можуть бути блокчейн-мережі, в яких транзакції і дані не зберігаються в одному місці, а розподіляються між багатьма учасниками. Така модель застосовується в криптовалютах, таких як біткойн, де кожен учасник мережі має копію всієї транзакційної історії. У децентралізованих системах криптографічний захист реалізується з особливою увагою до розподілу даних і автентифікації користувачів без централізованого контролю. Для забезпечення безпеки використовуються методи асиметричного шифрування, які дозволяють користувачам підписувати транзакції або дані, не потребуючи для цього стороннього центру сертифікації. Ключова роль у таких системах належить відкритим і закритим ключам. Наприклад, в блокчейні кожен учасник має пару ключів, де відкритий ключ використовується для перевірки підпису, а закритий — для підпису своїх транзакцій. Однією з основних переваг децентралізованих систем є відсутність єдиної точки відмови. Якщо один учасник мережі або вузол зазнає компрометації, це не призведе до зупинки всієї системи. Блокчейн-мережі, як приклад, забезпечують високий рівень захисту від атак на рівні інфраструктури, оскільки для компрометації мережі зловмиснику

потрібно контролювати більшість учасників (так звані атаки 51%). Проте, у децентралізованих системах є й певні труднощі. Однією з них є складність управління криптографічними ключами. Оскільки немає центрального органу, який би контролював ключі, кожен учасник відповідальний за зберігання своїх приватних ключів. У разі втрати приватного ключа користувач може втратити доступ до своїх даних або коштів. Це також збільшує ризик атак через фішинг або інші методи соціальної інженерії. Також децентралізовані системи часто мають вищі вимоги до обчислювальних ресурсів і більш складні механізми консенсусу. Наприклад, в блокчейн-мережах транзакції повинні бути підтверджені і записані на всіх учасниках мережі, що може займати значний час і ресурси.

Однією з головних відмінностей у криптографічному захисті між централізованими та децентралізованими системами є структура управління ключами. У централізованих системах ключі зазвичай зберігаються в одному місці і керуються єдиним адміністратором, що забезпечує зручне управління, але одночасно створює єдину точку відмови. У децентралізованих системах ключі розподілені серед усіх учасників мережі, що знижує ризики централізованих атак, але ускладнює управління та підвищує вимоги до обчислювальних ресурсів. Ще однією важливою відмінністю є механізми консенсусу. У централізованих системах, як правило, одна особа або організація приймає рішення щодо доступу і безпеки, тоді як у децентралізованих системах рішення приймаються через консенсус між учасниками, що створює додаткову складність у забезпеченні правильності та цілісності інформації [26].

Вибір між централізованою та децентралізованою архітектурою для криптографічного захисту залежить від конкретних потреб системи. Централізовані системи підходять для бізнесу, де важлива швидкість і ефективність обробки даних, тоді як децентралізовані системи забезпечують більшу безпеку в умовах високого рівня довіри до сторонніх учасників в розподіленому зберіганні інформації. Для розуміння, який саме тип архітектури

потрібен для web-застосунку було розроблено зведену таблицю 2.1. Загалом, можна побачити, наскільки децентралізовані системи кращі.

Таблиця 2.1.

Порівняння централізованих та децентралізованих систем

	Централізована система	***Децентралізована система***
Мережеві та апаратні ресурси	Обслуговуються та контролюються однією організацією	Належать всім учасникам мережі та спільно ними використовуються, обслуговування створює певні складнощі, тому що ніхто не володіє ними повністю
Компоненти рішення	Обслуговуються та контролюються однією особою	Кожен учасник має точну копію одного розподіленого реєстру
Дані	Обслуговуються та контролюються однією особою	Додаються лише за умови групового консенсусу
Управління	Контролюється центральним органом	Ніхто не володіє даними, ними володіють усі одночасно
Єдина точка відмови	Так	Ні
Відмовостійкість	Низька	Неймовірно висока
Безпека	Обслуговуються та контролюються однією особою	Збільшується зі збільшенням числа учасників мережі
Продуктивність	Обслуговуються та контролюються однією особою	Зменшується зі збільшенням кількості учасників мережі
Приклад	ERP-система	Блокчейн

Висновки за розділом 2

У другому розділі було здійснено аналіз криптографічних методів у веб-застосунках, зокрема визначено ключові вимоги до захисту інформації, розглянуто відмінності між централізованими та децентралізованими системами, а також оцінено виклики та обмеження сучасних криптографічних підходів.

Результати аналізу підтверджують, що захист інформації у веб-середовищі є багатовимірним завданням, яке вимагає врахування різних факторів: від забезпечення конфіденційності переданих даних до гарантування автентичності і цілісності інформації. Основними вимогами до криптографічного захисту у веб-застосунках є ефективне управління ключами, використання надійних алгоритмів шифрування, стійких до сучасних атак, а також забезпечення продуктивності системи навіть за умов підвищених навантажень. У цьому контексті важливу роль відіграють протоколи, такі як HTTPS, що дозволяють реалізувати комплексний захист за допомогою TLS.

Дослідження відмінностей між централізованими та децентралізованими системами показало, що кожен із підходів має свої сильні сторони і недоліки. Централізовані системи, такі як традиційні сервери, забезпечують високу швидкість обробки даних і спрощене управління ключами, однак вони є вразливими через наявність єдиної точки відмови. Децентралізовані системи, навпаки, мають переваги у вигляді більшого рівня безпеки та відсутності централізованого контролю, проте стикаються з проблемами складності управління ключами, вищими витратами ресурсів і необхідністю узгодження між учасниками мережі. Ці системи особливо ефективні у сферах, де важливим є збереження довіри між учасниками та уникнення монополії контролю над даними.

Виявлені виклики криптографічного захисту демонструють необхідність балансування між безпекою, продуктивністю та зручністю використання системи. Виклики, такі як розвиток квантових обчислень, нові види атак і загрози

втрати ключів, вимагають постійного вдосконалення алгоритмів шифрування та інфраструктури захисту. Одним із перспективних напрямків є впровадження квантово-стійких алгоритмів, які можуть забезпечити стійкість до атак із використанням квантових комп'ютерів.

Загалом, аналіз показав, що ефективна криптографічна система у веб-застосунках має враховувати особливості обраної архітектури, вимоги до швидкості обробки даних і забезпечення конфіденційності користувачів. Вибір методів шифрування та способів управління ключами повинен базуватися на специфічних потребах застосунку, характері даних та рівні довіри до сторонніх учасників. Централізовані системи залишаються популярними для комерційних застосувань, тоді як децентралізовані моделі, такі як блокчейн, відкривають нові горизонти для створення безпечних та довірених платформ.

РОЗДІЛ 3
БЛОКЧЕЙН ЯК ІНСТРУМЕНТ КРИПТОГРАФІЧНОГО ЗАХИСТУ

3.1. Основи та особливості технології блокчейн

Блокчейн — це революційна технологія, яка відіграє ключову роль у сучасному цифровому світі, забезпечуючи надійність, прозорість і безпеку даних. Її суть полягає у створенні децентралізованого середовища для запису, зберігання та передачі інформації. На відміну від традиційних баз даних, які управляються централізовано, блокчейн побудований за принципом розподіленого реєстру, де дані зберігаються в усіх вузлах мережі, а кожен запис захищений криптографічними методами.

Ця технологія набуває популярності завдяки своїй здатності вирішувати проблеми довіри між сторонами, які не мають прямого контакту. Наприклад, у фінансових операціях блокчейн забезпечує прозорість, що знижує ризик шахрайства, а в логістиці дозволяє відстежувати переміщення товарів у режимі реального часу. Але, перш ніж зрозуміти повну картину, важливо звернути увагу на базове поняття, яке лежить в основі блокчейну — реєстр.

Реєстр є формою систематизації та обліку будь-якої інформації. Так, реєстр у його початковому вигляді був покладений в основу комерційної діяльності ще в давні часи та використовувався для фіксування та зберігання різної інформації, переважно про гроші чи майно. Спочатку для запису використовувались глиняні таблички, потім — папірус, пергамент і папір. Проте переломним моментом стало впровадження комп’ютерної техніки, що спочатку використовувалася для перетворення інформації з паперу в цифровий код. На цей час алгоритми уможливили створення цифрових розподілених реєстрів, які мають можливості та властивості, що виходять далеко за межі традиційних паперових та електронних реєстрів [1, 4, 9].

Розподілений реєстр становить базу даних, що розподілена між декількома мережевими вузлами (нодами), кожен з яких отримує дані з інших вузлів і

зберігає повну копію реєстру. Водночас такі вузли оновлюються незалежно один від одного. Ключовою особливістю розподіленого реєстру є децентралізація, тобто відсутність єдиного центру зберігання та реєстрації даних. Водночас інформація в усіх вузлах розподіленого реєстру має бути валідна та актуальна, що є можливим лише через досягнення згоди між усіма вузлами такого реєстру. Кожен вузол складає та записує оновлення реєстру незалежно від інших вузлів. Потім вузли голосують за оновлення, щоб упевнитися, що більшість вузлів погоджується з кінцевим варіантом. Досягнення згоди щодо однієї з копій реєстру називається консенсусом, цей процес виконується автоматично за допомогою алгоритму консенсусу. Щойно консенсус досягнутий, розподілений реєстр оновлюється, і остання погоджена версія реєстру зберігається в кожному вузлі. Приклад загальної структури розподіленого реєстру відображено на рисунку 3.1.

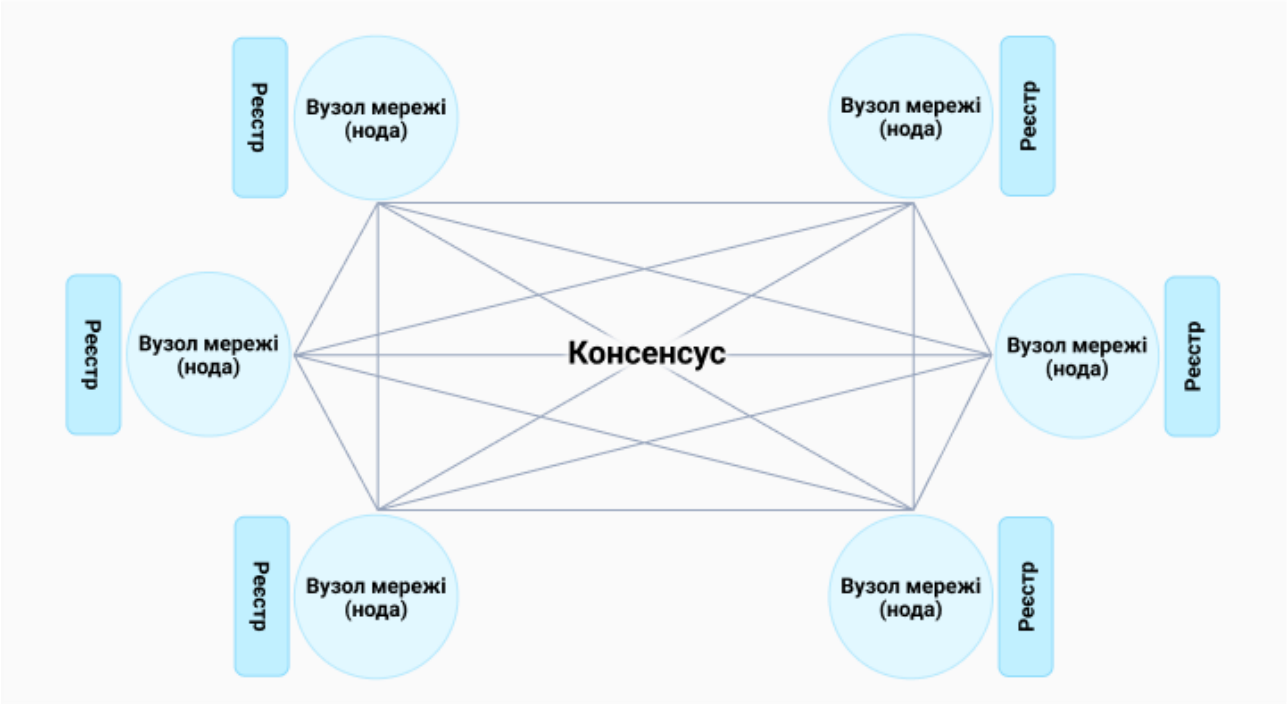

Рисунок 3.1 — Приклад загальної структури розподіленого реєстру

Блокчейн є одним із видів розподіленого реєстру, в якому для досягнення консенсусу між мережевими вузлами використовується послідовність блоків. Блоки організовані у хронологічній послідовності, об’єднані один із одним та захищені криптографічними методами, що можна побачити на рисунку 3.2. Кожен такий блок містить хеш-код, обчислений із попереднього блоку, та корисне навантаження. В якості корисного навантаження може виступати інформація про транзакції, операції, укладені договори, внесення в реєстр даних про фізичну особу, суб’єкт підприємницької діяльності, майно тощо. Іншими словами, корисним навантаженням може бути практично будь-яка інформація. За своєю сутністю блокчейн є реєстром записів, що постійно поповнюється, до якого можна лише додавати дані, але при цьому не можна видаляти чи змінювати дані, збережені у попередніх блоках.

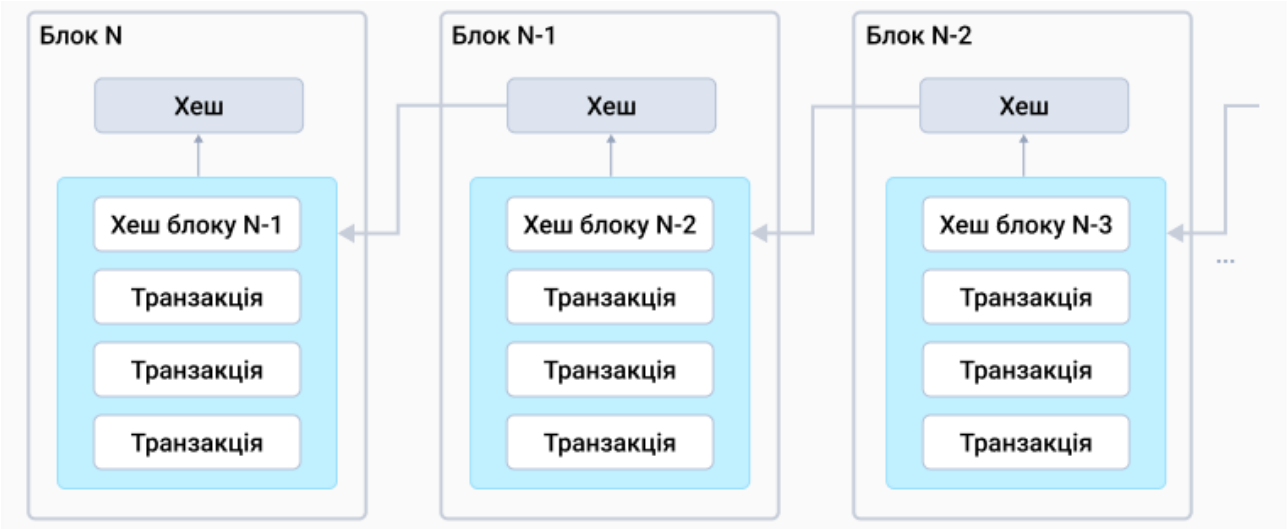

Рисунок 3.2 — Приклад загальної структури та організації блоків

У світі блокчейн-технологій існує кілька різних типів децентралізованих або розподілених мереж, кожен з яких має свої особливості та призначення. Основними типами блокчейн-сетей є публічний, приватний, гібридний і консорціумний блокчейн.

Публічні блокчейни не потребують дозволу для приєднання до мережі, що дозволяє будь-кому стати учасником. У таких мережах всі учасники мають рівні права на доступ до інформації, її редагування та перевірку. Цей тип блокчейну зазвичай використовують для криптовалют, таких як Bitcoin, Ethereum, Litecoin, де кожен може брати участь у процесі майнінгу або здійснювати транзакції без попереднього дозволу. Публічний блокчейн є повністю відкритим і прозорим, що робить його ідеальним для застосувань, де важлива дистрибуція ідеї довіри серед численних учасників [15].

Приватні блокчейни, іноді їх називають керованими або консорціумними, контролюються однією організацією або групою організацій. Власник мережі визначає, хто може приєднатися до мережі і які права доступу має кожен учасник. Такий тип блокчейну є частково децентралізованим, оскільки обмеження доступу зменшують рівень дистрибуції. Приватні блокчейни зазвичай використовуються в бізнес-середовищі, де є потреба в захисті чутливої інформації. Яскравим прикладом є Ripple, платформа для обміну цифровою валютою, яка надає більш контрольований доступ до даних порівняно з публічними системами.

Гібридний блокчейн поєднує в собі властивості як публічних, так і приватних блокчейнів. У таких мережах організації можуть встановлювати як публічні, так і приватні механізми доступу до даних. Це дає змогу контролювати доступ до певної інформації, при цьому зберігаючи відкритість інших даних для публіки. Гібридні блокчейни часто використовують смарт-контракти для підтвердження приватних транзакцій у публічному реєстрі. Такі рішення дозволяють, наприклад, зберігати цифрові валюти в публічному доступі, одночасно захищаючи доступ до банківської інформації чи інших чутливих даних.

Блокчейн-консорціуми являють собою мережі, які управляються групою організацій. Кожен учасник консорціуму має чітко визначену роль у функціонуванні блокчейну і відповідальність за прийняття рішень щодо доступу до даних. Такі мережі часто створюються для забезпечення співпраці між

кількома організаціями, що мають спільні інтереси. Блокчейн-консорціуми часто використовуються в бізнесі для зниження витрат, збільшення ефективності та покращення співпраці між учасниками. Яскравим прикладом є Global Shipping Business Network — некомерційний блокчейн-консорціум, що спеціалізується на цифровізації судноплавства та співпраці між операторами морських перевезень.

Вибір типу блокчейн-мережі залежить від конкретних цілей і вимог бізнесу. Публічні блокчейни є найкращими для відкритих транзакцій, де важлива дистрибуція і прозорість. Приватні блокчейни підходять для закритих середовищ, де потрібен контроль над доступом до інформації. Гібридні блокчейни дозволяють поєднати переваги обох типів, зберігаючи контроль за критично важливою інформацією, але одночасно даючи доступ до певних даних широкій аудиторії. Блокчейн-консорціуми ж є ідеальним вибором для організацій, що бажають спільно управляти мережею, знижуючи витрати і підвищуючи ефективність взаємодії. Таким чином, кожен з цих типів має свої унікальні властивості, які відповідають специфічним потребам користувачів і організацій, що їх використовують.

3.2. Використання блокчейну для забезпечення безпеки даних

Блокчейн має ряд основоположних принципів, які визначають його ефективність і затребуваність у сучасному цифровому світі. Перш за все, це технологія, що функціонує на основі розподіленого реєстру. Вона здатна працювати доти, доки існує хоча б один активний вузол мережі. Це означає, що навіть у випадку збоїв або відключень частини мережі, блокчейн зберігає свою функціональність і надійність.

Важливою особливістю є відсутність єдиного центру контролю. Усі учасники мережі мають доступ до повної історії транзакцій, проте жоден із них не володіє абсолютним контролем над системою. Це забезпечує децентралізований характер мережі, де кожен вузол відіграє однаково важливу

роль. Така структура виключає можливість маніпуляцій із боку однієї організації або окремої особи.

Блокчейн також відзначається високим рівнем прозорості. Усі транзакції, що здійснюються в мережі, є доступними для перевірки будь-яким користувачем. Це сприяє створенню довірчого середовища, оскільки інформація про кожну операцію може бути перевірена на достовірність. Завдяки цьому блокчейн стає незамінним інструментом для систем, де критично важливими є прозорість і довіра. Ще однією ключовою перевагою є незмінність даних. Інформація, записана в блокчейні, підтверджується багатьма мережевими вузлами, що робить її захищеною від видалення чи зміни. Це забезпечує збереження цілісності записів, а також підвищує рівень захищеності даних користувачів.

Технологія блокчейн поєднує відкритість із безпекою завдяки використанню сучасних криптографічних методів. Високий ступінь надійності досягається завдяки хешуванню та алгоритмам консенсусу, що дозволяє уникнути шахрайства та несанкціонованого доступу до даних. Блокчейн також виключає необхідність посередників у транзакціях. Усі операції здійснюються безпосередньо між учасниками мережі, автоматично перевіряються та підтверджуються мережею вузлів. Це значно скорочує витрати на комісії та час на виконання транзакцій, що є важливим фактором для бізнесу та фінансових систем. Таким чином, блокчейн забезпечує ефективність, безпеку та прозорість у зберіганні, обліку й передачі інформації. Його унікальні властивості роблять цю технологію затребуваною в різних сферах, таких як фінанси, логістика, охорона здоров'я, управління даними тощо. Блокчейн є перспективним механізмом, який здатний трансформувати традиційні підходи до обміну інформацією, створюючи нові можливості для бізнесу та суспільства [18].

Термін "блокчейн-протоколи" відноситься до різних типів платформ, що використовуються для розробки додатків на основі блокчейн-технології. Кожен протокол адаптує основні принципи блокчейну під конкретні сфери або завдання. Вибір протоколу залежить від потреб організації або розробників, які

прагнуть створити надійні, ефективні та масштабовані рішення. Розглянемо кілька найбільш відомих блокчейн-протоколів.

Hyperledger Fabric — це проект з відкритим вихідним кодом, спрямований на створення інструментів і бібліотек для підприємств. Цей протокол дозволяє компаніям швидко і ефективно створювати приватні блокчейн-додатки. Однією з головних особливостей Hyperledger Fabric є його модульність, що дозволяє адаптувати платформу під конкретні потреби. Вона надає можливості для ідентифікації користувачів та контролю доступу, що робить її особливо корисною для таких сфер, як відстеження ланцюга поставок, торгівельне фінансування, програми лояльності, а також безготівкові розрахунки по фінансових активах. Hyperledger Fabric широко використовується в тих випадках, коли необхідно забезпечити приватність і конфіденційність даних, але при цьому зберегти доступність та прозорість процесів для відповідних учасників мережі.

Ethereum — це децентралізована блокчейн-платформа з відкритим вихідним кодом, що використовується для створення публічних блокчейн-додатків. Ethereum дозволяє створювати децентралізовані додатки (dApps) на основі смарт-контрактів, що автоматизують виконання певних умов без потреби в посередниках. Ethereum Enterprise — це версія платформи, орієнтована на комерційне використання та призначена для корпоративного середовища. Вона підтримує створення приватних або гібридних мереж, що дозволяють організаціям контролювати доступ до даних і транзакцій, що важливо для бізнес-процесів у великих корпораціях.

Corda — це блокчейн-проект з відкритим вихідним кодом, розроблений спеціально для бізнесу. Corda дозволяє створювати сумісні блокчейн-мережі, в яких транзакції мають гарантовану конфіденційність. Це робить його ідеальним вибором для фінансових установ, де важливо зберігати приватність даних і при цьому забезпечувати швидкість і безпеку транзакцій. Corda також підтримує використання смарт-контрактів для автоматизації фінансових операцій, що дозволяє компаніям здійснювати угоди без необхідності звертатися до

посередників. Технологія використовується переважно у фінансовому секторі, для здійснення міжбанківських розрахунків та в інших секторах, де важливі швидкість і безпека обміну даними.

Quorum — це модифікація протоколу Ethereum з відкритим вихідним кодом, що орієнтована на використання в приватних блокчейн-мережах. Quorum підтримує конфіденційність транзакцій, що робить його особливо корисним для корпоративних і фінансових застосувань, де важливий контроль доступу і захист даних. Він також підходить для блокчейн-консорціумів, де кожен учасник володіє частиною мережі та має доступ до певних даних, що дозволяє здійснювати транзакції та зберігати важливу інформацію в межах групи учасників. Протокол дозволяє використовувати можливості Ethereum для створення приватних мереж з високим рівнем конфіденційності, але зберігаючи переваги децентралізації і прозорості, характерні для публічних блокчейнів.

Кожен з цих блокчейн-протоколів, які зображені на рисунку 3.3, має свої переваги та обмеження, тому вибір залежить від конкретних потреб організацій. Hyperledger Fabric і Quorum більше підходять для корпоративних і приватних мереж з високими вимогами до конфіденційності, тоді як Ethereum і Corda пропонують більшу відкритість і сумісність для побудови децентралізованих додатків, що орієнтовані на масштабування та інноваційність.

Рисунок 3.3 — Розглянуті блокчейн-протоколи

3.3. Криптовалютні гаманці та їх роль у авторизації

Криптовалютні гаманці є важливою частиною блокчейн-екосистеми, оскільки вони дозволяють користувачам зберігати та управляти своїми цифровими активами, а також забезпечують механізми для автентифікації та авторизації в блокчейн-системах. Криптовалютні гаманці не лише виконують роль сховища криптовалюти, але й стають важливими інструментами для забезпечення безпеки та захисту даних користувачів у процесах авторизації на різних платформах, включаючи децентралізовані додатки (dApps).

Криптовалютний гаманець — це програмне забезпечення або апаратний пристрій, що дозволяє користувачам зберігати, відправляти та отримувати криптовалюту. У контексті блокчейн-екосистеми гаманці виконують важливу роль, забезпечуючи безпеку транзакцій завдяки використанню криптографії для захисту доступу до активів. Гаманці використовують пару ключів: приватний та публічний. Приватний ключ — це секретний код, який надає доступ до коштів користувача та дозволяє підписувати транзакції, тоді як публічний ключ — це відкритий ідентифікатор, що дозволяє отримувати кошти.

Існують різні типи криптовалютних гаманців, зокрема:

- Програмні гаманці — це програмне забезпечення, яке може бути встановлене на комп'ютері або мобільному пристрої. Вони можуть бути гарячими (підключеними до інтернету) або холодними (офлайн).
- Апаратні гаманці — фізичні пристрої, які зберігають приватні ключі в оффлайн-режимі, що робить їх більш захищеними від зовнішніх атак.
- Онлайн-гаманці — гаманці, що знаходяться в інтернеті, і дозволяють швидко доступатися до активів через браузер. Однак вони менш захищені через постійну взаємодію з мережею.

З розвитком технологій блокчейн криптовалютні гаманці почали виконувати не тільки функції зберігання та управління цифровими активами, але й важливу роль у забезпеченні безпеки авторизації та аутентифікації користувачів у різних системах. Вони дозволяють користувачам здійснювати

вхід в децентралізовані платформи, взаємодіяти з блокчейн-додатками, а також підтверджувати свою особу при здійсненні різноманітних операцій. У традиційних системах авторизації користувачі зазвичай надають свої логін та пароль для доступу до платформи. Однак у блокчейн-системах, де важливе значення має збереження конфіденційності та анонімності, криптовалютні гаманці використовуються як основний інструмент для аутентифікації без необхідності створення традиційних облікових записів.

Основним принципом такої авторизації є використання публічного ключа для ідентифікації користувача, а приватного ключа для підписання транзакцій та підтвердження доступу до ресурсу або платформи. Коли користувач підключає свій гаманець до блокчейн-додатку, система перевіряє публічний ключ користувача та запитує підписування за допомогою приватного ключа для підтвердження, що транзакція або запит дійсно ініційовані тим користувачем, який володіє відповідними цифровими активами.

Приклади використання криптовалютних гаманців для авторизації

1. У децентралізованих додатках (dApps), які працюють на основі блокчейн-технології, криптовалютні гаманці служать засобом входу в систему. Це дозволяє уникнути необхідності створювати традиційний обліковий запис з паролем і логіном. Для авторизації користувач підключає свій гаманець, і платформа перевіряє підписану транзакцію, яка підтверджує його права на виконання певних операцій.
2. Відома система одностороннього доступу до платформи через криптовалютний гаманець використовується в багатьох блокчейн-платформах, таких як Ethereum та Bitcoin. Користувач підключається до платформи за допомогою своєї криптовалюти, що дає йому змогу анонімно взаємодіяти з іншими учасниками або виконувати операції без необхідності реєстрації.
3. В екосистемі Web3 криптовалютні гаманці виступають як основний інструмент для авторизації користувачів, де вони забезпечують доступ до різноманітних онлайн-сервісів, таких як платформи для DeFi

(децентралізованих фінансів) або NFT (незамінних токенів). Користувач може підключити гаманець, наприклад, MetaMask, для доступу до різних додатків, здійснення покупок або участі в криптовалютних транзакціях.

Серед основних переваг криптовалютних гаманців для авторизації можна виділити кілька ключових аспектів, які забезпечують їхню популярність і ефективність у сучасних блокчейн-системах. Усі ці переваги відображено на рисунку 3.4.

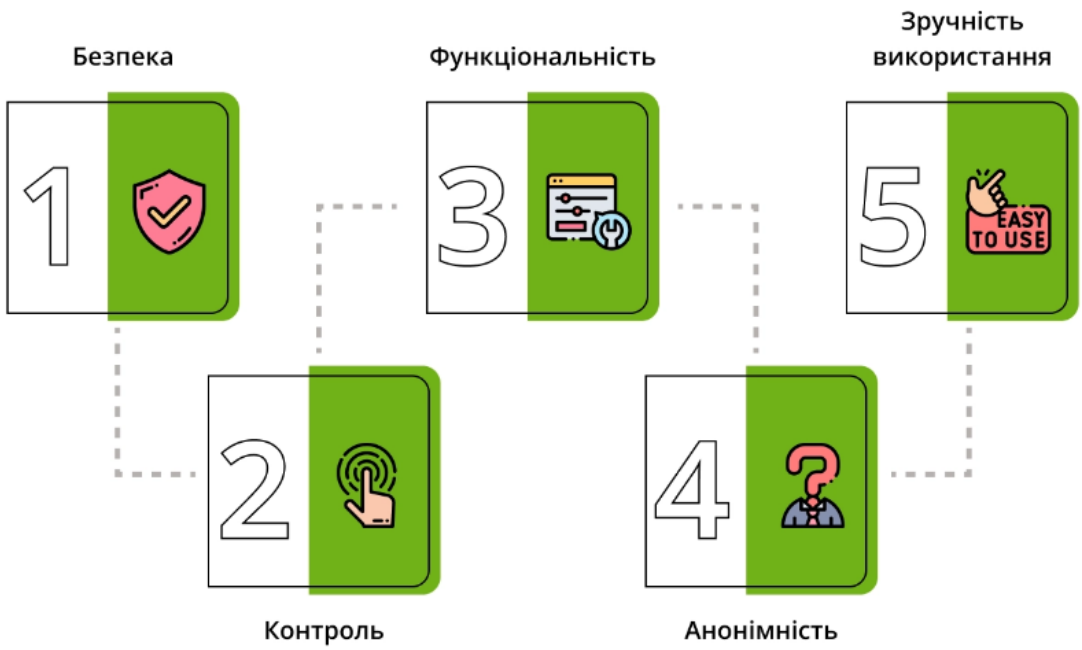

Рисунок 3.4 — Основні переваги криптовалютних гаманців

По-перше, безпека є однією з найважливіших характеристик криптовалютних гаманців. Доступ до гаманця можливий тільки через приватний ключ, що гарантує надійний захист від несанкціонованого доступу. Якщо хтось намагатиметься отримати доступ до акаунта на платформі, навіть за умови, що він має доступ до самого облікового запису, він не зможе здійснити жодних операцій без приватного ключа. Таким чином, висока ступінь безпеки забезпечує захист від крадіжки та шахрайства, що робить криптовалютні гаманці надійним інструментом для зберігання та управління цифровими активами.

По-друге, контроль над власними даними є важливою перевагою криптовалютних гаманців. Користувач повністю контролює свої активи, не покладаючись на сторонні організації або центральні банки. Це означає, що лише власник гаманця має право на підписання транзакцій і зміну даних. Також це дозволяє користувачам здійснювати транзакції без посередників, що зменшує ризики та витрати, а також підвищує ефективність використання.

Ще однією важливою перевагою є функціональність криптовалютних гаманців. Вони дозволяють не тільки зберігати криптовалюти, а й здійснювати платежі, інвестувати, підписувати угоди через смарт-контракти та взаємодіяти з різними блокчейн-додатками. Гаманці стають універсальним інструментом для доступу до децентралізованих платформ, що відкриває нові можливості для бізнесу та користувачів. Анонімність є ще однією важливою перевагою використання криптовалютних гаманців для авторизації. Оскільки процес авторизації не вимагає надання особистих даних, таких як ім’я, адреса чи електронна пошта, користувач може зберігати свою анонімність. Це особливо важливо для тих, хто прагне зберегти конфіденційність при здійсненні фінансових операцій або при взаємодії з децентралізованими платформами, де важлива приватність. І, звісно, важливою перевагою є зручність використання. Криптовалютні гаманці забезпечують простоту та швидкість доступу до платформ, що значно полегшує процес авторизації та взаємодії з блокчейн-системами. Користувачі можуть швидко підключатися до додатків, підтверджувати транзакції та здійснювати операції без необхідності проходити складні процедури перевірки або реєстрації, що робить процес більш ефективним та доступним для широкого кола користувачів.

Висновки за розділом 3

У третьому розділі дослідження розглянуто основи та особливості технології блокчейн, її використання для забезпечення безпеки даних, а також

роль криптовалютних гаманців у процесах авторизації та захисту даних користувачів. Технологія блокчейн, завдяки своїй децентралізованій природі, надає можливість забезпечення високого рівня безпеки даних через використання криптографічних методів, що дозволяє знижувати ризики фальсифікацій і зловживань. Одним із важливих аспектів, що розглядається в розділі, є роль криптовалютних гаманців, які не тільки дозволяють зберігати і управляти цифровими активами, але й виступають важливим інструментом для забезпечення авторизації та аутентифікації в блокчейн-системах.

Завдяки криптовалютним гаманцям, користувачі можуть здійснювати аутентифікацію без необхідності створення традиційних облікових записів з логінами та паролями, що забезпечує високий рівень конфіденційності та анонімності. Публічні та приватні ключі, що використовуються в гаманцях, дозволяють гарантувати безпеку транзакцій і захист від несанкціонованого доступу. Важливим є також те, що гаманці дозволяють контролювати власні дані і здійснювати операції без посередників, що забезпечує не тільки безпеку, але й ефективність використання.

Розглянуті переваги криптовалютних гаманців для авторизації включають безпеку, контроль над даними, функціональність, анонімність і зручність використання. Всі ці аспекти роблять криптовалютні гаманці незамінним інструментом для взаємодії з блокчейн-системами та децентралізованими платформами. У висновку можна зазначити, що з розвитком блокчейн-технологій роль криптовалютних гаманців буде лише зростати, а їх впровадження в різні сфери діяльності дозволить значно підвищити рівень безпеки та ефективності цифрових операцій.

Цей розділ підкреслює важливість криптовалютних гаманців не тільки як інструментів зберігання активів, а й як основних засобів для забезпечення безпеки та аутентифікації в сучасних блокчейн-екосистемах.

РОЗДІЛ 4
ІНТЕГРАЦІЯ БЛОКЧЕЙНУ В СИСТЕМИ ЗАХИСТУ ДАНИХ

4.1. Особливості інтеграції блокчейну у веб-застосунки

Для успішної інтеграції блокчейну у веб-застосунки необхідно забезпечити гармонійне поєднання децентралізованої природи блокчейну та звичної для користувачів логіки роботи сучасних веб-додатків. Одним із ключових викликів є оптимізація обробки транзакцій. У традиційних веб-додатках операції виконуються миттєво на сервері, тоді як у блокчейн-системах транзакції потребують часу на підтвердження в мережі (залежно від її завантаження та типу блокчейну). Це зумовлює необхідність впровадження механізмів відкладеної обробки або показу проміжного стану, щоб забезпечити комфорт користувачів.

Ще одним важливим аспектом є управління приватними ключами користувачів. Веб-застосунок має бути налаштований так, щоб мінімізувати ризики витоку або втрати приватних ключів. Один із найкращих підходів — інтеграція гаманців, таких як MetaMask або Trust Wallet, що дозволяють користувачам зберігати ключі локально. Крім того, можуть використовуватися рішення на основі криптографії з розподілом секретів (наприклад, Shamir's Secret Sharing) для підвищення безпеки.

Для інтеграції блокчейну у веб-застосунки також важливо враховувати питання масштабованості. Блокчейн-системи мають обмежену пропускну здатність порівняно з традиційними базами даних. Щоб уникнути перевантажень, можливе застосування шарів масштабування, таких як Layer 2-протоколи (наприклад, Optimism або zk-Rollups), що дозволяють зменшити навантаження на основний блокчейн, зберігаючи його безпеку [5, 6].

Інтеграція смарт-контрактів є ще однією суттєвою частиною цього процесу. Смарт-контракти забезпечують автоматизацію бізнес-логіки веб-застосунків. Наприклад, у системах бронювання квитків вони можуть

автоматично обробляти оплату та перевіряти доступність місць у режимі реального часу. Однак розробка смарт-контрактів потребує ретельного тестування на предмет уразливостей, оскільки помилки в коді можуть призвести до втрати коштів.

Окремо варто відзначити роль інтеграції децентралізованих сховищ, таких як IPFS або Arweave, для зберігання даних, які не можна ефективно розмістити безпосередньо в блокчейні. Це дозволяє створювати веб-застосунки, які одночасно забезпечують децентралізоване зберігання великих обсягів даних і доступ до них через блокчейн для перевірки достовірності.

Нарешті, інтеграція блокчейну потребує уважного підходу до UX/UI дизайну. Для користувачів, які не знайомі з криптовалютними технологіями, інтерфейси повинні бути максимально інтуїтивними та простими, зокрема містити підказки щодо використання криптографічних гаманців, підпису транзакцій та перевірки балансу.

4.2. Використання смарт-контрактів для захисту даних

Смарт-контракти можна назвати втіленням мрії про чесність у бізнесі, оскільки вони забезпечують автоматичне виконання умов угоди всіма сторонами без необхідності залучення посередників чи гарантів. У цьому розділі ми розглянемо, що таке смарт-контракти, їхні основні типи та як створити такий контракт самостійно.

Ще у 1994 році Нік Сабо, відомий криптограф, правознавець і дослідник у сфері комп'ютерних технологій (деякі вважають, що саме він є творцем Bitcoin, який приховується під псевдонімом Сатоші Накамото), запропонував ідею використання децентралізованого реєстру для реалізації самовиконуваних контрактів, які отримали назву "смарт-контракти". "Я називаю ці контракти "розумними", тому що вони набагато функціональніші за звичайні паперові угоди. Смарт-контракт — це набір зобов’язань, прописаних у цифровій формі,

разом із протоколами, що регулюють виконання цих зобов’язань", — так Сабо описував концепцію смарт-контрактів.

Якщо проаналізувати це визначення, можна зрозуміти, що смарт-контракт — це автоматизований комп'ютерний алгоритм, умови виконання якого закодовані у вигляді програмного коду. Цей код зберігається в децентралізованій мережі, наприклад, у блокчейні. У ньому містяться дані та інструкції, які виконують транзакції відповідно до умов договору, забезпечуючи їхню прозорість і незворотність. При цьому смарт-контракти виключають необхідність залучення банків, державних органів чи інших третіх сторін для підтвердження дотримання умов угоди [7].

Якщо не заглиблюватися у технічні деталі, смарт-контракти можна порівняти з автоматизованими механізмами для обміну цінностями. Наприклад, вони дозволяють обмінювати фіатні гроші на біткоїни, цифрові товари, такі як відеоігри, чи навіть фізичні активи, як-от золото. Принцип їхньої роботи схожий на функціонування звичайного кавового автомата, який виконує обмін грошей на чашку кави без участі продавця. У цьому випадку "цінностями" виступають гроші покупця та кава, а умовою обміну є внесення визначеної суми (ціни кави) в автомат. Механізм автомата приймає ресурси від обох сторін і автоматично виконує обмін.

Смарт-контракт працює за схожим принципом, але його функціонал реалізується через програмний код. Цей код містить інструкції та правила, які виконуються за заздалегідь визначеною логікою "якщо → тоді":

- Якщо смарт-контракт отримує об'єкт А (наприклад, 1 біткоїн), тоді користувач, який передав цей об'єкт, отримує об'єкт В (наприклад, 16 ефіру).
- Якщо контракт отримує об'єкт В, тоді запускається функція Х (наприклад, відтворення відео чи доступ до цифрового продукту).

- Якщо контракт отримує об'єкт С, тоді користувач, який передав об'єкт С, отримує доступ до додатку чи іншого ресурсу.

Такі контракти можуть бути створені та розгорнуті на різних блокчейн-платформах, таких як Ethereum, або Hyperledger Fabric. Кожна платформа має свої особливості, але всі смарт-контракти мають спільну структуру: код, що визначає функції контракту, та дані, які представляють його стан. Ці компоненти зберігаються на певній адресі у блокчейні. Смарт-контракти також є своєрідними обліковими записами в блокчейні. Вони мають такі ж права, як і звичайні облікові записи користувачів, але управляються не людьми, а власним кодом. Завдяки цьому користувачі блокчейна можуть взаємодіяти зі смарт-контрактами, надсилаючи їм токени для виконання транзакцій, обміну криптовалюти або запуску інших функцій. Загалом схема роботи смарт-контракту зображена на рисунку 4.1. Така автоматизація робить смарт-контракти ефективним інструментом, що дозволяє усунути посередників, знизити ризики людської помилки та забезпечити прозорість і надійність угод [10].

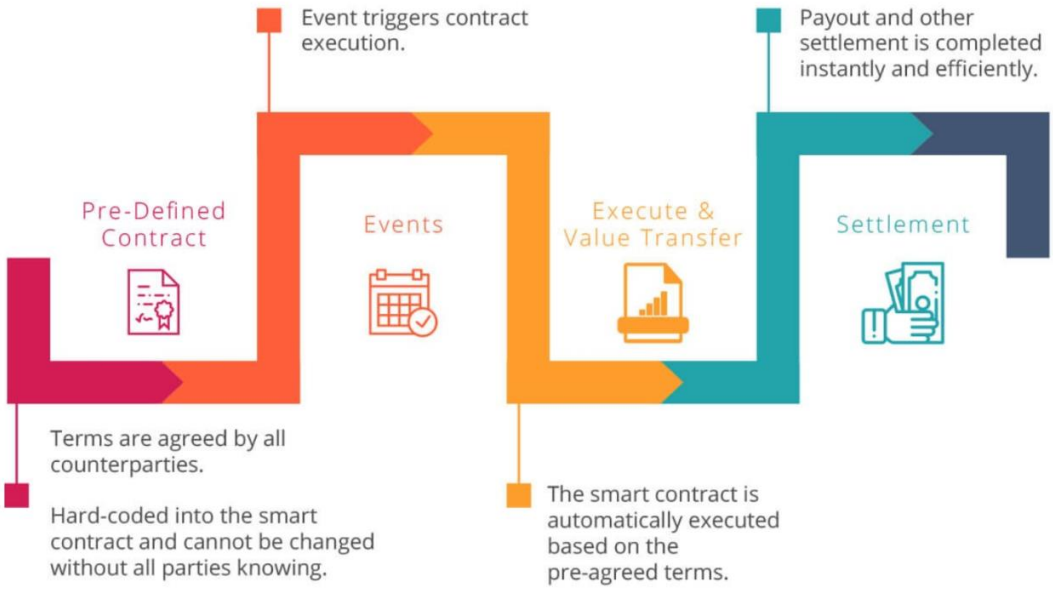

Рисунок 4.1 — Схема роботи смарт-контракту

Кількість переваг, які можуть отримати звичайні користувачі, бізнес та державні організації завдяки використанню смарт-контрактів і блокчейну, є вражаючою. Ось деякі з ключових аспектів:

1. Однією з основних переваг децентралізованих блокчейн-систем є абсолютна прозорість. Усі транзакції в межах блокчейну є відкритими для перевірки та недоступними для несанкціонованого редагування. Завдяки цьому користувачі та бізнес можуть бути впевнені, що інформація, зафіксована в смарт-контракті, залишиться незмінною та захищеною від несанкціонованих змін чи шахрайських дій.

2. Смарт-контракти дозволяють автоматизувати численні ручні бізнес-процеси, такі як укладення угод та контроль їх виконання. Крім того, відсутність посередників (наприклад, юристів, банків чи брокерів) значно зменшує витрати, що особливо важливо для малого та середнього бізнесу.

3. Традиційні договори потребують значних витрат часу для підготовки, підписання та подальшого моніторингу. Натомість смарт-контракти, створені одного разу, автоматично виконують необхідні дії, як-от випуск NFT або оформлення транзакцій. Це суттєво прискорює бізнес-процеси.

4. Смарт-контракти усувають людський фактор, що сприяє зміцненню довіри між сторонами угоди. Виключення можливості порушення умов договору автоматично усуває конфлікти між контрагентами, дозволяючи зосередитися на вирішенні будь-яких технічних питань.

5. Блокчейн забезпечує надійне збереження інформації завдяки її дублюванню на численних вузлах мережі. Це дозволяє уникнути втрат даних через технічні збої або кібератаки. Дані залишатимуться доступними, поки працює сама блокчейн-система.

6. Правильно розроблені смарт-контракти забезпечують захист від несанкціонованого доступу до даних, знижуючи ризики шахрайських дій.

Хоча фішингові атаки все ще можливі, розвиток технологій поступово знижує цю вразливість.

7. Смарт-контракти використовують сучасні методи шифрування, що робить їх одним із найбезпечніших способів укладання угод. Завдяки цьому вони стають важливим інструментом для бізнесу та державних структур, які працюють із конфіденційними даними.

Попри численні переваги, смарт-контракти мають і певні обмеження, які слід враховувати під час їх впровадження:

1. Хоча смарт-контракти мають високий рівень захисту, технологія все ще є відносно новою. Це відкриває можливості для хакерів, які шукають вразливості в коді. Для мінімізації ризиків необхідно проводити детальне тестування та аудит коду, залучаючи як внутрішніх фахівців, так і незалежних експертів.
2. Автоматизація процесів і висока швидкість роботи смарт-контрактів можуть посилювати негативні наслідки в разі помилок у коді або некоректного функціонування системи. Наприклад, помилки можуть спричинити значні фінансові втрати або створити інші труднощі в масштабованості.
3. Смарт-контракти практично неможливо змінити після їхнього розгортання в блокчейні. Ця характеристика забезпечує захист даних, але ускладнює внесення виправлень чи оновлень у код навіть за потреби зміни мінімальних деталей.

Попри ці недоліки, смарт-контракти залишаються важливим етапом розвитку сучасних технологій, які сприяють автоматизації, прозорості та безпеці процесів у різних сферах. Якщо порівнювати смарт контракт та звичайний паперовий контракт, до якого ми усі звикли, то отримаємо зведену таблицю 4.1.

Таблиця 4.1.

Порівняння смарт-контракту зі звичайним контрактом

	Смарт-контракт	***Звичайний контракт***
Носій	Алгоритм або програма	Паперовий документ
Мова виконання	Мови програмування: Solidity, JavaScript	Юридичний, канцелярит
Підстави для виконання	Умови, прописані у смарт-контракті	Закони, нормативи, договори, бажання учасників
Зміни	Після ініціалізації контракту, його неможливо змінити	Можна вносити в будь-який момент
Виконання	Прописані в смарт-контракті умови виконуються автоматично всіма учасниками	Кожен учасник сам вирішує, виконувати контракт чи ні, як і за яких умов виконувати
Наслідки невиконання	Штрафи і санкції прописані в смарт-контракті, тому покарання відбувається автоматично	Щоб домогтися виконання або покарати порушника, потрібно звернутися до суду
Посередники	Ті, які прописані в законі	Нотаріуси, юристи, чиновники, банки, продавці, довірені особи, поручителі та інші
Створення	Потрібна допомога програміста	Потрібна допомога юриста, чиновника, нотаріуса
Носії цінності	Цифрові гроші та сертифікати	Документи, готівка та цифрові гроші

4.3. Процес розробки смарт-контракту

Розібравшись із концепцією, призначенням та перевагами смарт-контрактів, можна перейти до питання їхньої розробки для застосування у своєму проєкті чи бізнесі. Ось ключові етапи цього процесу:

Першим і найважливішим рішенням під час розробки смарт-контракту є вибір відповідної блокчейн-платформи. Якщо раніше основним вибором був Ethereum та кілька менш відомих платформ, то сьогодні ринок пропонує набагато більше варіантів. Ethereum залишається лідером за кількістю впроваджених смарт-контрактів та децентралізованих додатків (Dapps), але має значні недоліки: низька швидкість, високі витрати на транзакції та проблеми з масштабуванням [19]. Через це дедалі більше розробників обирають альтернативні платформи, такі як Polygon, Polkadot, Cardano, Solana, Binance Smart Chain (BSC), Tezos чи Hyperledger. Шість популярних платформ для створення смарт-контрактів представлено на рисунку 4.2.

Як і у випадку з вибором блокчейнів, для розробки смарт-контрактів доступний широкий набір інструментів. Вони поділяються на різні категорії залежно від завдань, які мають виконувати. Деякі інструменти є критично важливими, наприклад, мова програмування, без якої створити смарт-контракт просто неможливо. Інші є додатковими і використовуються для полегшення чи оптимізації роботи. Нижче наведено перелік найбільш популярних і затребуваних інструментів, які активно застосовуються розробниками у сфері створення смарт-контрактів.

	Execution environment	Smart contract language	Turing completeness	Permission type	Consensus
Ethereum	EVM	Solidity	✓	Public	PoW (PoS expected)
Polkadot	PEE	Depends on a selected chain	Depends on a selected chain	Depends on a selected chain	NPoS
Hyperledger	Docker	JavaScript, Go		Private	CFT
Tezos	Tezos VM	Michelson	✓✓	Public	PoS
Stellar	Docker	Net, Scala, C++, Go	✗	Consortium	FBA (SCP)
Solana	LLVM	C, Rust	✓	Public	PoH and PoS

Рисунок 4.2 — Популярні платформи для створення смарт-контракту

Процес створення смарт-контракту включає написання його програмного коду. Розробник може писати контракт із нуля, але це не обов'язково. Щоб спростити роботу, часто використовують готові шаблони або бібліотеки з відкритим кодом. Для кожної блокчейн-платформи існує свій набір інструментів для розробки. Нижче представлено допоміжну таблицю 4.2, що може значно допомогти при розробці смарт-контракту. Наприклад, для Ethereum дуже популярною є бібліотека OpenZeppelin, яка надає готові рішення для безпечної реалізації контрактів, таких як токени або механізми голосування. OpenZeppelin також забезпечує сумісність із найновішими стандартами, що дозволяє скоротити час на тестування і гарантує безпеку коду. Якщо ж обрана платформа Hyperledger, вона пропонує власний набір інструментів, спеціально розроблених для потреб корпоративних блокчейнів. Ці інструменти допомагають адаптувати смарт-контракти до складних бізнес-процесів, інтегруючи їх у вже існуючу IT-інфраструктуру. Загалом використання готових бібліотек та інструментів має безліч переваг. Вони не тільки прискорюють процес розробки, але й знижують ризики помилок, пов'язаних із написанням коду вручну. Крім того, такі ресурси дозволяють розробникам зосередитися на унікальних функціях контракту, а не на базовій реалізації, яка вже опрацьована в численних перевірених рішеннях.

Таблиця 4.2.

Основні інструменти при розробці смарт-контракту

	Опис	***Інструмент***
Мови програмування	Необхідні для створення смарт-контрактів. Вибір мови майже завжди залежить від обраного блокчейна	Solidity, C++, Go, Rust, JavaScript, C, Vyper, DAML
Інтегровані середовища розробки	Цей інструментарій допомагає писати і тестувати код. Вони можуть бути мережевими або настільними	Remix, SettleMint, EthFiddle, Visual Studio Code

Продовження табл. 4.2.

Фреймворки	Використовуються для більш ефективного тестування і розгортання смарт-контрактів. Вибір фреймворку майже завжди залежить від уподобань і досвіду розробника	Hardhat, Truffle, Brownie, DappTools, ApeWorX
Тестові мережі	Необхідні для поміщення смарт-контракту в контрольоване середовище і визначення того, чи поводиться він так, як очікувалося	Rinkeby, Goerli, Kovan, Hyperledger Umbra, Ropsten
Криптовалютні гаманці	Гаманці з підтримкою смарт-контрактів надають доступ до Dapps і Web3 для тестування протоколу та іншого	Metamask, TrustWallet, Coinbase

З огляду на велику кількість зламів смарт-контрактів, тестування є одним з найважливіших етапів у їх розробці. Оскільки смарт-контракти є програмним забезпеченням з відкритим кодом, будь-хто може переглянути їх код і виявити потенційні уразливості, які можна використати для злому контракту. Крім того, після публікації смарт-контракту на блокчейні його вже не можна буде змінити, що робить тестування останньою можливістю виправити всі можливі помилки та недоліки перед запуском. Для полегшення тестування більшість блокчейнів мають тестові мережі, що дозволяють провести перевірку смарт-контракту без ризику втрати грошей, даних або репутації. Вибір тестової мережі залежить від конкретного блокчейна. Наприклад, для Ethereum популярними тестовими мережами є Ropsten і Rinkeby, а для Hyperledger Fabric необхідна тестова мережа Hyperledger Umbra [25].

Останнім етапом є розгортання смарт-контракту на основній мережі блокчейну. Після цього контракт стає доступним для користувачів і більше не підлягає змінам. Щоб оновити смарт-контракт, потрібно створити та запустити

нову версію, при цьому стара залишатиметься в мережі, поки блокчейн функціонує. Так, наприклад, на Ethereum зараз працює кілька версій Uniswap, що функціонують паралельно.

Висновки за розділом 4

Інтеграція блокчейну в системи захисту даних є важливим кроком на шляху до створення більш безпечних, прозорих та ефективних веб-застосунків. З огляду на широке поширення діджиталізації та зростаючі загрози кібербезпеці, блокчейн виступає як потужний інструмент, який здатен забезпечити захист даних від несанкціонованого доступу та фальсифікацій. Система, побудована на блокчейні, характеризується децентралізацією, що дозволяє усунути ризик централізованих атак і значно підвищити стійкість до зломів. Важливим елементом для досягнення високого рівня безпеки є використання смарт-контрактів, які автоматизують процеси верифікації та забезпечення виконання умов без необхідності довіряти третім сторонам. Завдяки своїй прозорості, незмінності та автоматичному виконанню, смарт-контракти дозволяють уникнути помилок і маніпуляцій, що може значно знизити ймовірність шахрайства та зловживань.

Особливості інтеграції блокчейну в веб-застосунки свідчать про значний потенціал цієї технології для підвищення рівня безпеки. Вибір правильної платформи для розробки та відповідних інструментів є критичним для досягнення бажаного результату. Наприклад, платформи Ethereum і Hyperledger надають різні можливості для реалізації смарт-контрактів, орієнтуючись на різні типи користувачів і бізнес-потреби. Застосування таких інструментів, як OpenZeppelin для Ethereum або набори інструментів для Hyperledger, значно спрощують процес розробки і тестування смарт-контрактів.

Процес розробки смарт-контрактів складається з кількох етапів, починаючи від вибору платформи і мови програмування до тестування та розгортання контракту в основній мережі. Завдяки розвитку технологій, зокрема використанню тестових мереж, розробники можуть знизити ризики помилок і забезпечити безпеку контрактів до їх публікації. Оскільки смарт-контракти є незмінними після їх публікації, тестування перед впровадженням є критично важливим етапом для мінімізації потенційних уразливостей і помилок. В майбутньому можна очікувати подальше вдосконалення цих технологій та їх більш широке впровадження в сфері захисту даних.

РОЗДІЛ 5
ПРОГРАМНА РЕАЛІЗАЦІЯ

Розроблений веб-застосунок являє собою інноваційну платформу для бронювання та придбання квитків на заходи різних типів, включаючи концерти, театральні вистави, спортивні матчі та інші події. Основна особливість платформи — інтеграція з технологіями Web 3.0, що забезпечує авторизацію та здійснення транзакцій через криптовалютний гаманець MetaMask. Це дозволяє забезпечити високий рівень прозорості, безпеки та децентралізації при взаємодії користувачів із системою.

Основний функціонал веб-застосунку включає наступні можливості:

- Користувачі мають змогу переглядати детальну інформацію про заходи, зокрема назву, дату, місце проведення та доступні квитки.
- Інтерфейс дозволяє користувачам інтерактивно обирати місця у схемі залу для бронювання чи покупки.
- Використання криптовалютного гаманця для входу в систему надає можливість уникати традиційних реєстраційних форм і забезпечує безпечний доступ до особистих даних.
- Здійснення транзакцій із використанням криптовалюти гарантує прозорість розрахунків та знижує ризики шахрайства.

Цей застосунок є не лише зручним інструментом для користувачів, але й перспективним рішенням для організаторів заходів, які прагнуть автоматизувати продаж квитків, мінімізувати витрати на посередників та підвищити довіру до своїх послуг.

5.1. Стек технологій та інструменти

Розробка веб-застосунку базується на сучасному технологічному стеку, що забезпечує ефективну інтеграцію з блокчейном і надає зручний користувацький інтерфейс.

Основні компоненти технологічного стеку:

- Solidity — мова програмування для створення смарт-контрактів, що забезпечує основну логіку роботи застосунку, зокрема механізми оплати та авторизації через криптовалютний гаманець.
- Javascript — використовується для побудови інтерактивних елементів інтерфейсу (React.js) і тестування функціоналу системи.
- Hardhat — фреймворк для розробки, тестування та деплою смарт-контрактів, що значно спрощує взаємодію з блокчейн-середовищем.
- Ethers.js — бібліотека для інтеграції з блокчейном, яка дозволяє виконувати транзакції, взаємодіяти зі смарт-контрактами та працювати з криптогаманцями.
- React.js — фронтенд-фреймворк для створення швидкого, адаптивного та інтуїтивно зрозумілого користувацького інтерфейсу.
- MetaMask — криптовалютний гаманець, інтегрований у платформу, використовується для авторизації користувачів та проведення транзакцій у мережі блокчейн.

Цей стек технологій забезпечує злагоджену роботу всіх компонентів системи, гарантує високу продуктивність, безпеку даних і масштабованість застосунку [2, 8, 12].

5.2. Вимоги до початкового налаштування

Для запуску веб-застосунку необхідно виконати початкове налаштування середовища, що забезпечить коректну роботу всіх компонентів системи.

1. Веб-браузер: Для доступу до застосунку рекомендується використовувати сучасний браузер, наприклад, Google Chrome, який підтримує інтеграцію з розширеннями для криптовалютних гаманців.
2. MetaMask: Потрібно встановити розширення MetaMask у браузері. Воно слугуватиме криптовалютним гаманцем для взаємодії з блокчейном, виконання транзакцій і авторизації користувачів.

3. Node.js: Необхідно встановити платформу Node.js, бажано останню стабільну версію (LTS). Вона необхідна для запуску фронтенд-серверу та роботи інструментів розробки.
4. Термінал: Використання терміналу потрібне для встановлення залежностей, тестування та запуску локального сервера.

Ці базові налаштування створюють середовище для роботи з проєктом, забезпечуючи інтеграцію між користувацьким інтерфейсом, смарт-контрактами та блокчейн-мережею.

5.3. Налаштування та запуск

Процес розпочинається з налаштування криптогаманця. Для цього використовуємо гаманець MetaMask, який необхідно встановити у вигляді розширення для браузера Google Chrome. Після встановлення розширення потрібно увійти до свого криптогаманця, як на рисунку 5.1, або створити новий обліковий запис, якщо його ще немає.

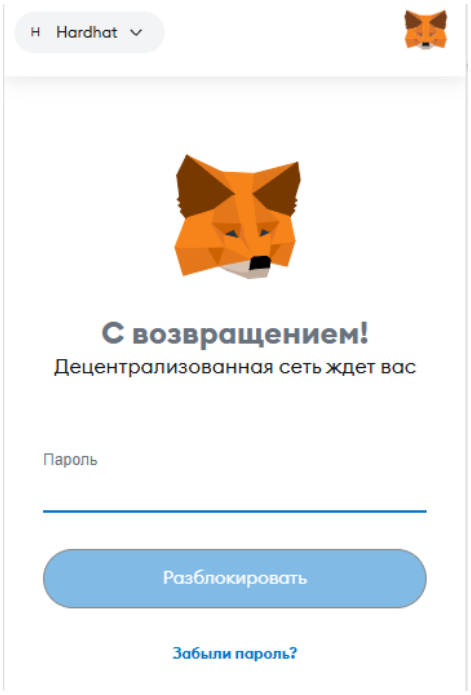

Рисунок 5.1 — Вхід до гаманця MetaMask

Після успішного входу до криптогаманця відкривається головне вікно MetaMask (рисунок 5.2). У цьому інтерфейсі відображаються основні дані гаманця: баланс криптовалюти, доступні мережі, а також інструменти для управління активами. У ньому можна змінювати мережі, підключати гаманець до децентралізованих додатків, відправляти та отримувати криптовалюту, а також налаштовувати додаткові параметри залежно від потреб.

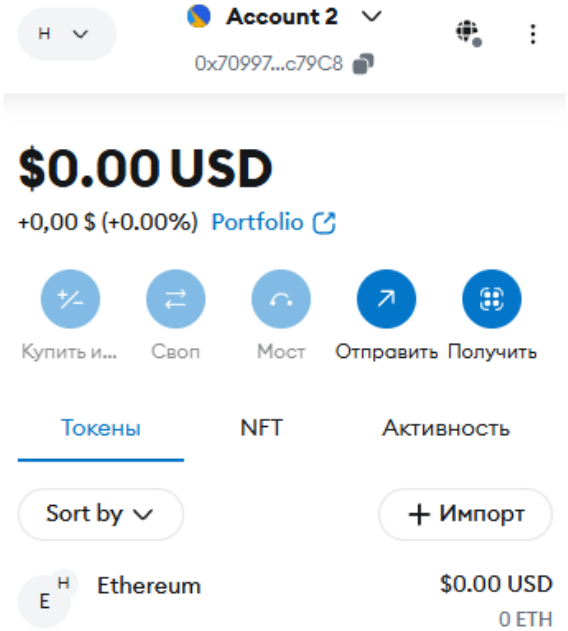

Рисунок 5.2 — Успішна авторизація криптогаманця

На цьому етапі робота з криптогаманцем завершується, і увага переміщується на запуск локального серверу застосунку (рисунок 5.3). Для цього у терміналі здійснюється перехід до директорії проєкту, де розташовані всі необхідні файли та залежності.

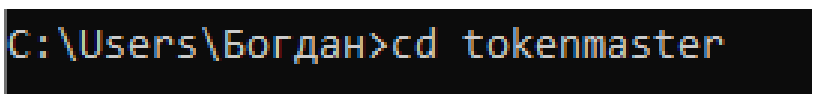

Рисунок 5.3 — Команда входу до директорії проєкту

Після переходу до директорії проєкту необхідно встановити всі потрібні залежності. Для цього виконується відповідна команда у терміналі (рисунок 5.4). Після успішного встановлення залежностей запускаються тести, які перевіряють коректність роботи смарт-контрактів та функціональність застосунку (рисунок 5.5).

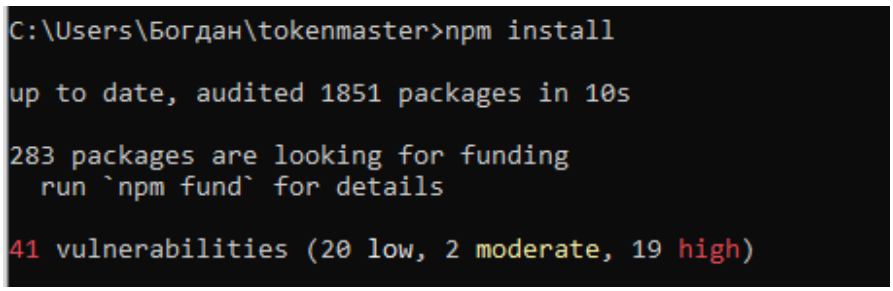

Рисунок 5.4 — Команда встановлення залежностей

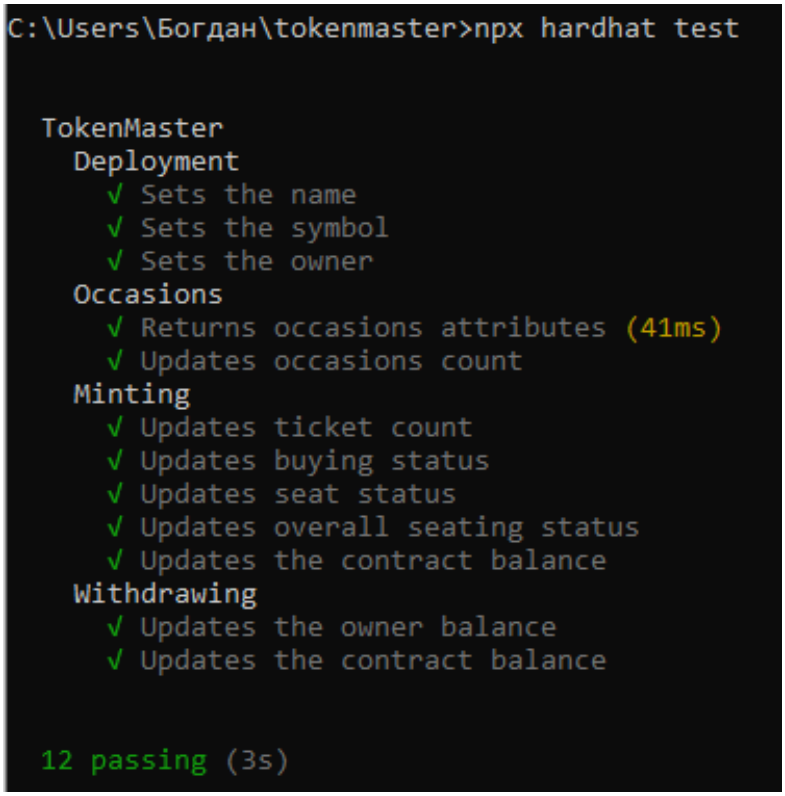

Рисунок 5.4 — Команда запуску тестів

Після успішного проходження всіх тестів можна перейти до наступного етапу. Необхідно запустити вузол Hardhat, який забезпечить локальне середовище для взаємодії зі смарт-контрактами та перевірки роботи застосунку. Це виконується за допомогою відповідної команди у терміналі, перебуваючи в директорії проєкту (рисунок 5.6).

```
C:\Users\Богдан\tokenmaster>npx hardhat node
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/
```

Рисунок 5.6 — Команда запуску вузла Hardhat

Для продовження необхідно запустити сценарій розгортання. У новому терміналі, залишаючись у директорії проєкту, виконується відповідна команда (рисунок 5.7), яка розгортає смарт-контракти на локальному вузлі Hardhat. Цей крок дозволяє перевірити коректність роботи контрактів у тестовому середовищі.

```
C:\Users\Богдан\tokenmaster>npx hardhat run ./scripts/deploy.js --network localhost
Deployed TokenMaster Contract at: 0x5FbDB2315678afecb367f032d93F642f64180aa3

Listed Event 1: UFC Miami
Listed Event 2: ETH Tokyo
Listed Event 3: ETH Privacy Hackathon
Listed Event 4: Dallas Mavericks vs. San Antonio Spurs
Listed Event 5: ETH Global Toronto
```

Рисунок 5.7 — Команда розгортання функціоналу проекту

На завершальному етапі запускається користувацький інтерфейс застосунку. У терміналі виконується команда для запуску фронтенд-серверу (рисунок 5.8), що забезпечує доступ до інтерфейсу через веб-браузер. Це дозволяє взаємодіяти з функціоналом застосунку та перевірити його роботу у зручному візуальному середовищі.

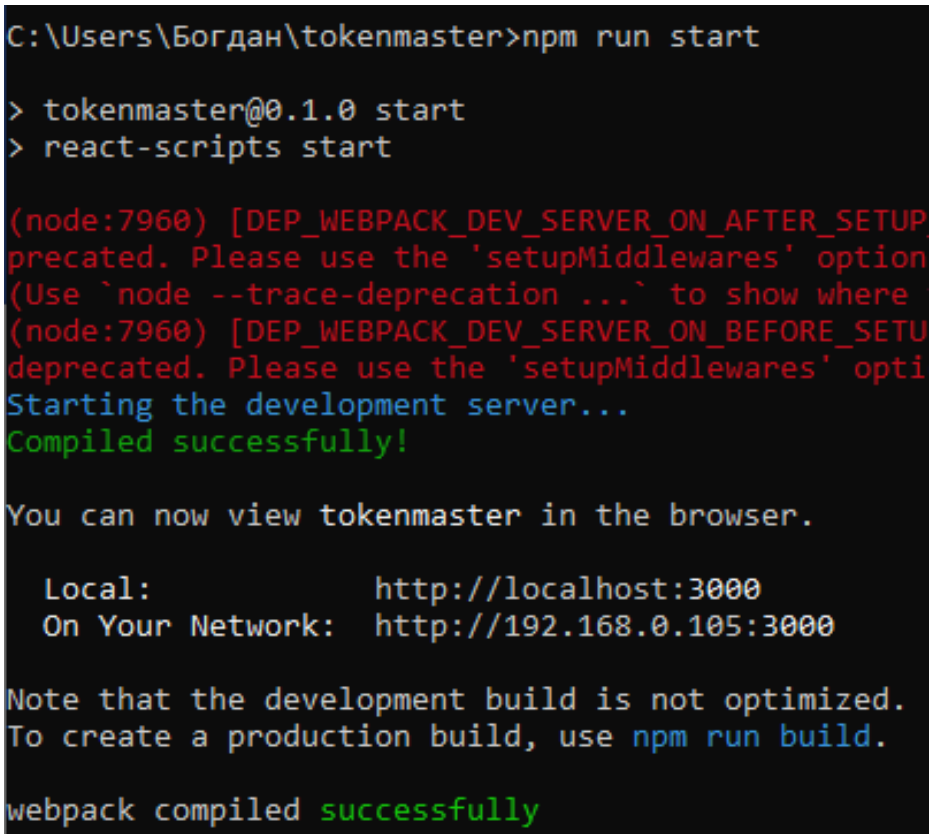

Рисунок 5.8 — Команда запуску інтерфейсу веб-застосунку

Веб-застосунок успішно запущено та готовий до використання. Його функціонал можна перевірити через браузер, забезпечуючи доступ до всіх інтегрованих можливостей.

5.4. Демонстрація роботи застосунку

Робота користувача із веб-застосунком починається з авторизації за допомогою криптогаманця. Для цього необхідно натиснути кнопку «Connect», яка забезпечує інтеграцію з обраним гаманцем, наприклад, MetaMask (рисунок 5.9).

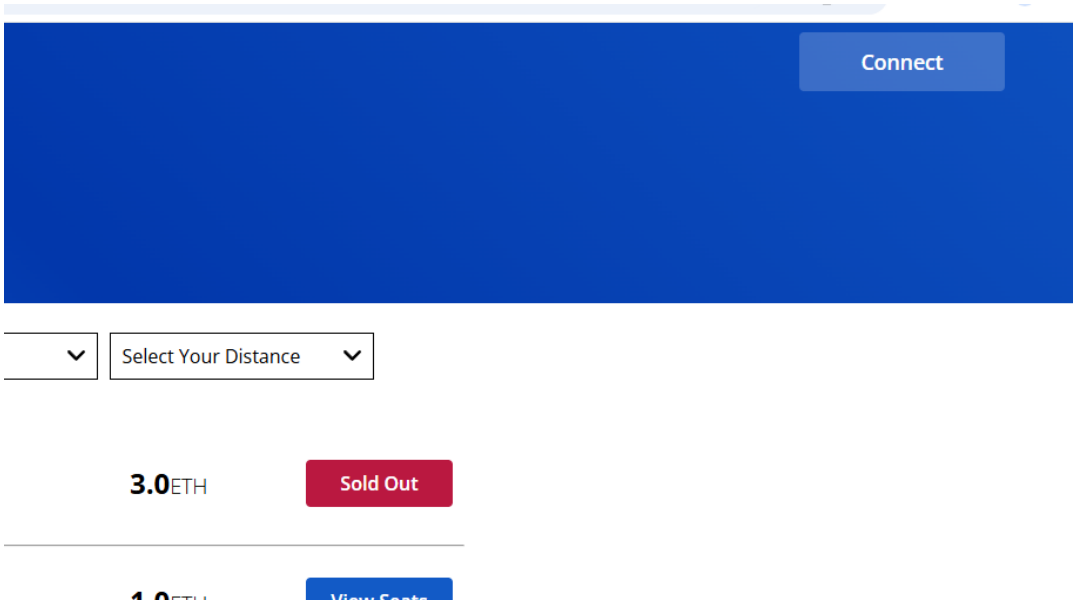

Рисунок 5.9 — Авторизація через криптогаманець

У разі успішної авторизації на екрані відобразиться адреса підключеного криптогаманця (рисунок 5.10).

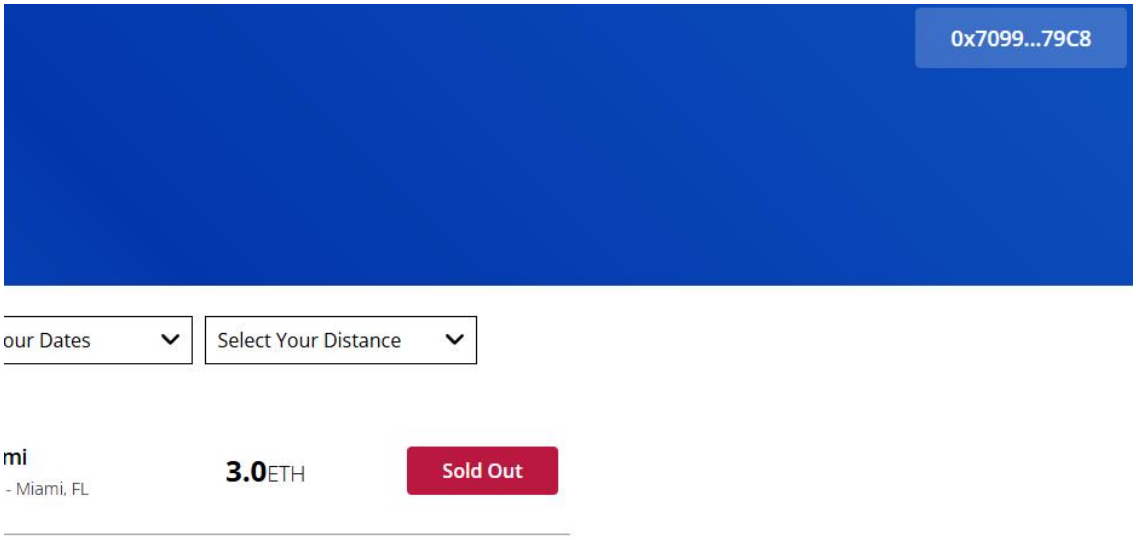

Рисунок 5.10 — Успішна авторизація

Веб-застосунок пропонує чотири основні категорії для продажу квитків на заходи: концерти, спортивні події, мистецькі вистави та театральні постановки.

Для зручності користувачів передбачена окрема категорія для інших заходів. Крім того, реалізовано функцію пошуку, яка дозволяє швидко знайти потрібний захід за ключовими словами (рисунок 5.11).

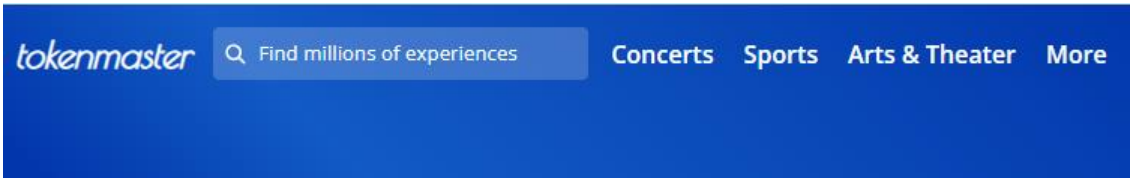

Рисунок 5.11 — Пошук подій за категоріями

Для зручності пошуку заходів у веб-застосунку передбачено систему фільтрів. Вони дозволяють обрати заходи за такими критеріями, як дата проведення, жанр або тривалість. Це допомагає користувачам швидко знайти подію, що відповідає їхнім уподобанням (рисунок 5.12).

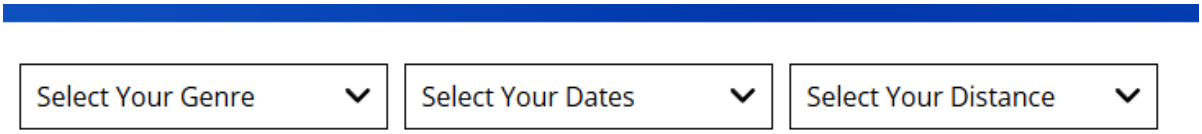

Рисунок 5.12 — Пошук за фільтрами

Загальний вигляд застосунку представляє зручний інтерфейс, у якому користувач може легко орієнтуватися. Основні категорії розташовані у вигляді окремих вкладок або кнопок, пошук та фільтри розміщені у верхній частині сторінки для швидкого доступу. Дизайн оптимізований для інтуїтивного використання та комфортної взаємодії з функціоналом (рисунок 5.13).

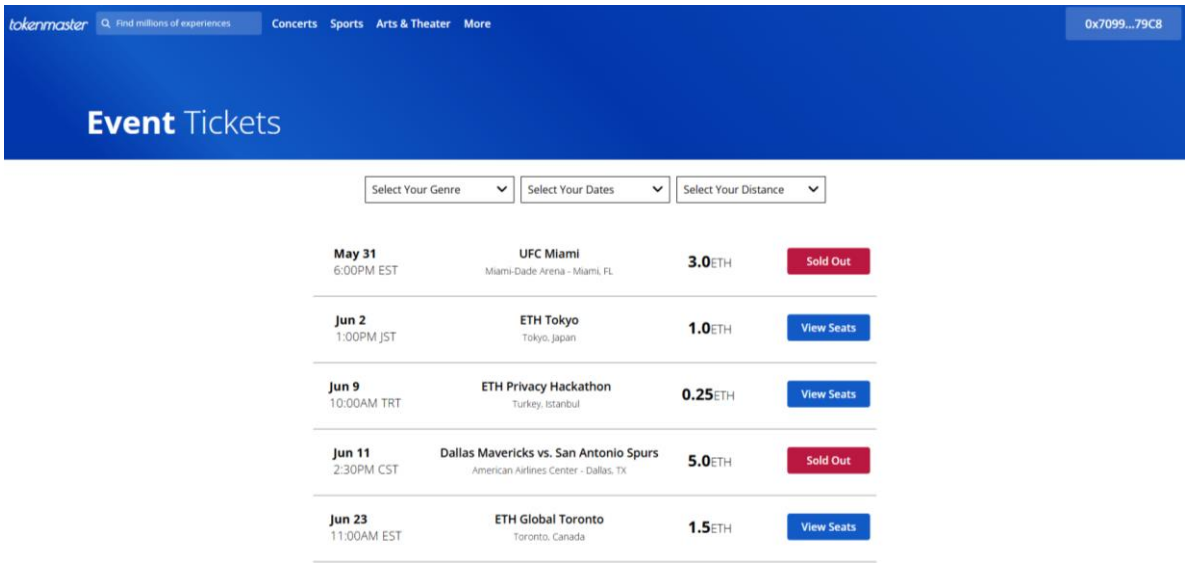

Рисунок 5.13 — Загальний вигляд веб-застосунку

Для проведення тестування виконується сценарій придбання квитків на конференцію компанії Ethereum, що запланована у місті Tokyo. Спочатку натискається кнопка «View Seats», після чого відкривається карта із розташуванням місць у залі (рисунок 5.14). На цьому етапі обирається вільне місце, яке є найбільш зручним для користувача, шляхом натискання на нього.

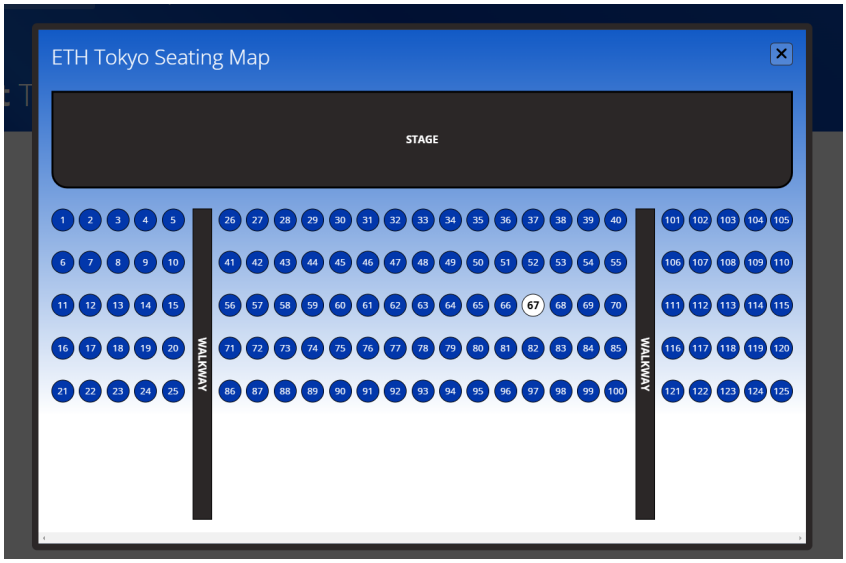

Рисунок 5.14 — Карта місць

Після вибору місця відкривається вікно криптогаманця для здійснення оплати (рисунок 5.15). У цьому вікні відображається сума, необхідна для оплати квитка, а також комісія мережі. Перевіривши зазначену інформацію, можна або відхилити транзакцію, або підтвердити її для завершення процесу покупки.

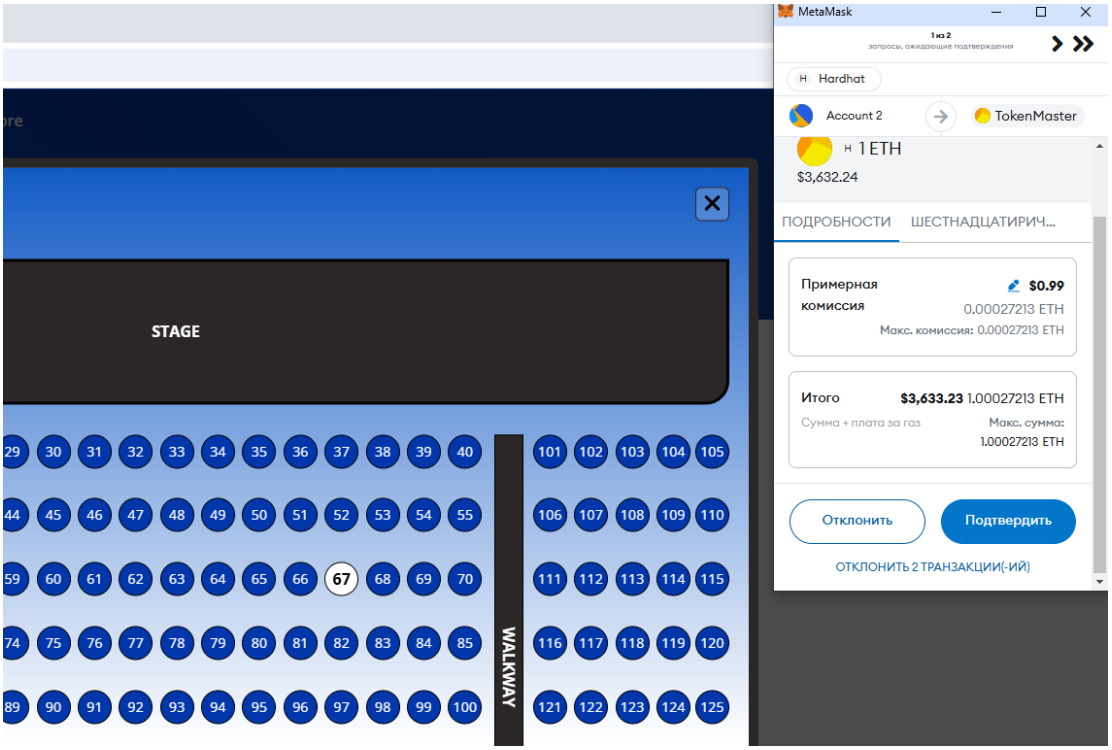

Рисунок 5.15 — Оплата квитка

Після підтвердження транзакції обране місце змінює свій статус на зайняте, що відображається у карті розташування місць. Одночасно у криптогаманці з'являється запис про успішно проведену транзакцію, що підтверджує завершення процесу покупки (рисунок 5.16).

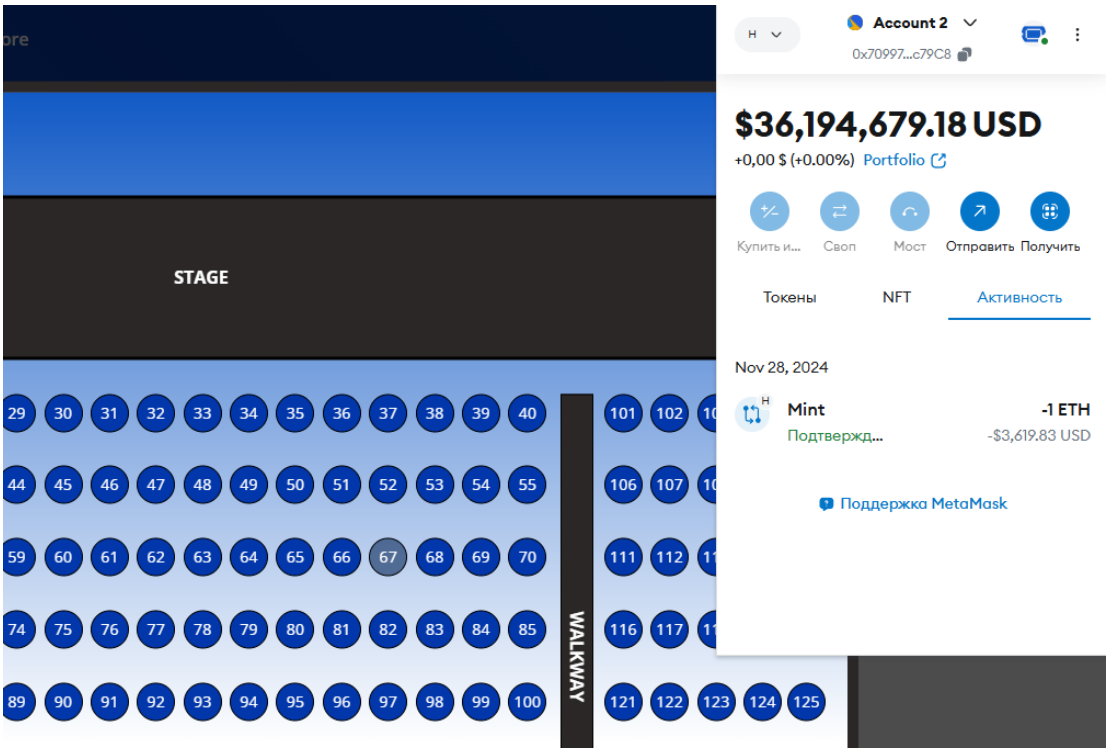

Рисунок 5.16 — Успішне бронювання місця

Таким чином, транзакція була виконана безпечно та анонімно, без необхідності розкривати особисті дані. Це гарантує високий рівень безпеки та конфіденційності під час взаємодії користувача з веб-застосунком [20, 23].

Висновки за розділом 5

У п’ятому розділі було детально описано процес програмної реалізації веб-застосунку, що забезпечує безпечну та зручну роботу з продажем квитків на основі технології блокчейн. Було визначено стек технологій, необхідних для створення застосунку, а також інструменти, які використовувалися для його налаштування, тестування та розгортання. Реалізація розпочалася з підготовки середовища розробки, включаючи встановлення всіх необхідних залежностей, розгортання локального сервера та налаштування криптогаманця MetaMask для взаємодії з блокчейном. Використання Hardhat для написання та тестування смарт-контрактів забезпечило стабільність і безпеку програмного коду. Успішне

тестування підтвердило готовність застосунку до розгортання в реальному середовищі.

На прикладі демонстрації було показано функціонал системи, починаючи від авторизації через криптогаманець до здійснення транзакції. Особливий акцент зроблено на безпеці користувачів, адже всі операції виконуються анонімно, без збереження персональних даних. Зручність використання веб-застосунку підвищується завдяки впровадженим категоріям подій, функції пошуку та фільтрам, що дозволяють швидко знаходити потрібні заходи та купувати квитки.

У результаті, створений застосунок демонструє потенціал технології блокчейн у сучасних веб-системах, поєднуючи високу безпеку, конфіденційність і зручність користування. Цей підхід може стати основою для розвитку нових рішень у сфері електронної комерції та забезпечення захисту даних.

ВИСНОВКИ

У кваліфікаційній роботі досліджено моделі криптографічного захисту даних у веб-застосунках із використанням блокчейн-технологій. У першому розділі проведено теоретичний аналіз основ криптографії, її принципів та сучасних тенденцій, що визначають напрямки розвитку інформаційної безпеки.

Другий розділ присвячено аналізу криптографічних методів у веб-застосунках, порівнянню централізованих і децентралізованих систем захисту, а також вимогам до захисту інформації в мережі.

У третьому розділі розглянуто технологію блокчейн як ефективний інструмент для забезпечення безпеки даних, досліджено її основи, застосування для захисту інформації та роль криптовалютних гаманців у процесах авторизації.

Четвертий розділ присвячено інтеграції блокчейну в системи захисту даних, зокрема використанню смарт-контрактів для управління доступом до інформації та забезпечення автоматизації процесів. Також описано ключові аспекти їх розробки.

У п’ятому розділі представлено програмну реалізацію дослідженого рішення: описано стек технологій, процес налаштування середовища, тестування та запуску застосунку. Веб-застосунок демонструє авторизацію через криптогаманець, пошук і фільтрацію подій, а також безпечну оплату через блокчейн-транзакції.

Результати роботи підтвердили ефективність блокчейну для забезпечення безпеки та конфіденційності даних у веб-застосунках. Розроблений застосунок ілюструє практичні можливості технології для створення сучасних безпечних цифрових платформ і може слугувати основою для подальших розробок у сфері захисту інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Antonopoulos A. M. Mastering Bitcoin: Programming the Open Blockchain. O'Reilly Media, Inc., 2017. – 411 p.
2. Antonopoulos A. M., Wood G. Mastering Ethereum: Building Smart Contracts and DApps. O'Reilly Media, Inc., 2018. – 424 p.
3. Aumasson J. Serious Cryptography: A Practical Introduction to Modern Encryption. No Starch Press, 2017. – 312 p.
4. Bauerle N. What is the Blockchain and How Does It Work? In Blockchain: Developing Disruptors. Packt Publishing, 2018.
5. Bashir I. Mastering Blockchain: Deeper Insights into Decentralization, Cryptography, Bitcoin, and Popular Blockchain Frameworks. Packt Publishing, 2018.
6. Bichova I. B., Cherednichenko V. V. Особливості криптографічного захисту ділової документації. Перспективи управлінської діяльності суб'єктів господарювання в контексті економічної безпеки. Черкаси, 2017.
7. Burniske C., Tatar J. Cryptoassets: The Innovative Investor's Guide to Bitcoin and Beyond. McGraw-Hill Education, 2017.
8. Buterin V. A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum Foundation, 2014. URL: https://ethereum.org/whitepaper/
9. Calcaterra C. Blockchain and Cryptocurrency Regulation. Bloomsbury Professional, 2018.
10. Dacko M., Vovk V.M., Monastyrsky M.A. Перспективи впровадження технології BLOCKCHAIN у банківську систему України. Формування ринкової економіки в Україні. Львів, 2017.
11. Drescher D. Blockchain Basics: A Non-Technical Introduction in 25 Steps. Apress, 2017.
12. Ferguson N., Schneier B., Kohno T. Cryptography Engineering: Design Principles and Practical Applications. Wiley, 2010.

13. Francuk V.M. Захист інформаційних ресурсів: криптографічні та стеганографічні методи захисту даних. НПУ імені М.П. Драгоманова, 2012.
14. Glinchuk L., Yatsyuk S., Kuzmych O., Bahniuk N., Chernyashchuk N. Requirements Analysis and Methodology of Selection of Topics for Studying the Basics of Cryptographic Protection of Information. COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION, 2020. https://doi.org/10.36910/6775-2524-0560-2020-41-03
15. Iansiti M., Lakhani K.R. The Truth About Blockchain. Harvard Business Review, 2017.
16. Kogut Yu. Технології блокчейн та криптовалюта. Ризики та кібербезпека. Консалтингова компанія Сідкон.
17. Костюченко В.М., Малюк О.С. Блокчейн як інструмент забезпечення прозорості державних закупівель. Державне управління: удосконалення та розвиток, 2019.
18. Kravchenko P., Skryabin B., Dubinina O. Блокчейн і децентралізовані системи. Промарт, 2019.
19. Kovalenko V.V., Serheyeva O.S. Перспективи використання технології блокчейн в державному управлінні в Україні. Публічне урядування, 2018.
20. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. URL: https://bitcoin.org/bitcoin.pdf
21. Narayanan A., Bonneau J., Felten E., Miller A., Goldfeder S. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press, 2016.
22. Tapscott D., Tapscott A. Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World. Penguin, 2016.
23. Тарнавський Ю.А. Технології захисту інформації. КПІ ім. Ігоря Сікорського, 2018.
24. Тапскотт Д., Тапскотт А. Блокчейн-революція. Видавництво Літопис, 2019.

25. Спасітєлєва С.О., Бурячок В.Л. Перспективи розвитку додатків блокчейн в Україні. Кібербезпека: освіта, наука, техніка, 2018.
26. Zibin Zheng, Shaoan Xie, Hong-Ning Dai. Blockchain Challenges and Opportunities: A Survey. Int. J. Web and Grid Services, 2018.
27. Олійник О.В. Організаційно-правові засади захисту інформаційних ресурсів України. Інститут законодавства Верховної Ради України, 2006.

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123– Комп'ютерна інженерія
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
в.о. завідувача кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«12» вересня 2024року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Веремієнко Богдана Олександровича
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«Модель криптографічного захисту даних користувачів у web-застосунках»**
керівник роботи Чуб Ольга Ігорівна, к.е.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету ***№ 4101-5/3657 від 12 листопада 2024 року***

2. Строк подання студентом роботи ***30 листопада 2024 року***

3. Перелік питань, які потрібно розробити

1. Дослідження основних методів криптографії.
2. Аналіз сучасних криптографічних стандартів для захисту даних у web-застосунках.
3. Огляд вразливостей і загроз безпеці даних у web-застосунках.
4. Вивчення механізмів шифрування та захисту даних під час передачі через мережу.
5. Вибір криптографічних технологій для реалізації моделі захисту.
6. Розробка моделі криптографічного захисту даних.
7. Тестування та оцінка ефективності розробленої моделі.
8. Використання блокчейну для захисту персональних даних.
9. Аналіз програмних рішень для впровадження криптографічного захисту у web-застосунки.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Затвердження теми роботи	05.09.2024 — 15.09.2024
2	Аналіз та пошук методичної літератури	16.09.2024 — 01.10.2024
3	Реалізація теоретичного матеріалу	01.10.2024 — 15.10.2024
4	Розробка структури майбутньої програми	16.10.2024 — 24.10.2024
5	Реалізація графічного модулю, який включає в себе інтерфейс для роботи з користувачем	16.10.2024 — 17.11.2024
6	Оформлення пояснювальної записки	17.11.2024 — 20.11.2024
7	Перед захист кваліфікаційної роботи	Листопад 2024
8	Представлення кваліфікаційної роботи керівнику та рецензенту	28.11.2024 - 30.11.2024

5. Дата видачі завдання ***05 вересня 2024року.***

Студент **Веремієнко Б. О.** _____
ініціали, прізвище підпис

Керівник роботи **Чуб О. І.** _____
ініціали, прізвище підпис

Додаток Б

Технічне завдання
на розробку програмного виробу
«Модель криптографічного захисту даних користувачів у web-застосунках»

Назва розділу	Назва та зміст підрозділу
1. Вступ	1.1. Назва програмного виробу: Модель криптографічного захисту даних користувачів у web-застосунках. 1.2. Галузь застосування: інформаційна безпека, веб-розробка, децентралізовані системи.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – «Комп'ютерна інженерія». 2.2. Завдання на кваліфікаційну роботу студента (магістра) із наказом №4101-5/3657 12.11.2024.
3. Призначення розробки	3.1. Мета розробки програмного виробу: створення моделі криптографічного захисту даних із використанням блокчейн-технології для забезпечення анонімності, прозорості та надійності у веб-застосунках. 3.2. Призначення програмного виробу: забезпечення безпеки даних користувачів у децентралізованих веб-системах через інтеграцію криптографії та смарт-контрактів. 3.3. Вихідні дані для розробки: методи криптографії, технології блокчейн, протоколи авторизації через криптогаманці, основи веб-розробки.
4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: реалізація авторизації користувачів через криптовалютний гаманець, механізм зберігання транзакцій та даних у блокчейні, використання смарт-контрактів для управління даними, забезпечення конфіденційності та анонімності користувачів. 4.2. Вимоги до надійності : захист даних від витоків і несанкціонованого доступу, відповідність стандартам кібербезпеки. 4.3. Вимоги до умов експлуатації : працездатність у популярних веб-браузерах, підтримка сучасних операційних систем. 4.4. Вимоги до складу параметрів технічних засобів: сумісність із сучасними криптогаманцями, такими як MetaMask і WalletConnect, швидкодія та низьке споживання ресурсів серверу. 4.5. Вимоги до інформаційної та програмної сумісності: сумісність із бібліотеками Ethers.js, Web3.js, та Solidity, інтеграція з протоколами блокчейн-мережі Ethereum. 4.6. Вимоги до маркіровки та упаковки не пред’являються. 4.7. Вимоги до транспортування та зберігання не пред’являються. 4.8. Спеціальні вимоги до програмного виробу не пред’являються.
5. Вимоги до програмної документації.	Програмною документацією до виробу «Модель криптографічного захисту даних користувачів у web-застосунках» вважати:

	1) Справжнє Технічне завдання на розробку програмного виробу; 2) Програму та методику випробувань розробленого програмного виробу.
6. Техніко-економічні показники	1) Орієнтовна оцінка ефективності: підвищення безпеки даних у веб-застосунках, скорочення витрат на інтеграцію традиційних механізмів захисту даних. 2) Терміни та витрати коштів: Терміни розробки програмного виробу можуть становити до 6 місяців. Витрати на розробку включатимуть заробітну плату розробників, придбання необхідного обладнання та програмного забезпечення. Вартість розробки може сягати від $50,000 до $100,000. 3) Порівняння з аналогами: більша безпека завдяки децентралізації та прозорості.
7. Стадії та етапи розробки	1) Аналіз вимог і дослідження: на цій стадії визначаються вимоги до програмного виробу, а також проводиться аналіз ринку та конкурентів. 2) Проектування: на цьому етапі розробляється детальний план роботи, складається технічне завдання та визначається архітектура програмного продукту. 3) Розробка: це етап, на якому програмісти пишуть код та тестують програмний продукт. 4) Тестування: на цій стадії проводяться тестування програмного виробу для виявлення та виправлення помилок та недоліків. 5) Випробування та валідація: на цьому етапі перевіряється, чи задовольняє програмний продукт вимоги та специфікації, які були визначені на початкових етапах розробки. 6) Впровадження та підтримка: це завершальний етап, на якому вирішуються питання пов'язані з розгортанням програмного виробу в робоче середовище, його підтримкою та забезпеченням безперебійної роботи.
8. Порядок контролю та приймання	1) Перевірку ходу розробки програмного виробу Керівнику робіт виконувати 1 раз на 3 тижні. 2) Випробування програмного виробу відповідно до Програми та методики випробувань провести на базі комп’ютерного класу. 3) Захист розробленого програмного виробу провести на засіданні ДЕК. 4) Пояснювальну записку подати на паперових носіях в одному екземплярі, в електронному вигляді – на CD-диску в одному екземплярі.

Виконавець _____ Веремієнко Б. О. Замовник _____ Чуб О. І.

Додаток В

Програма та методика випробувань програмного виробу «Модель криптографічного захисту даних користувачів у web-застосунках»

1. Об'єкт випробувань

1.1 Найменування програмного виробу: Модель криптографічного захисту даних користувачів у web-застосунках.

1.2 Область застосування: захист даних у веб-застосунках із використанням блокчейн-технологій та криптографічних методів.

1.3 Умовне призначення розробки: використання в системах авторизації через криптовалютні гаманці, забезпечення безпечного зберігання та передачі конфіденційних даних, реалізація функцій смарт-контрактів для управління доступом до даних.

2. Мета випробувань

Підтвердження відповідності функціональних та інших характеристик розробленого програмного виробу вимогам, зазначеним у Технічному завданні.

3. Загальні положення

3.1 Підстави щодо випробувань

Підставою для проведення випробувань є: 1) Наказ ХНУ про затвердження тем кваліфікаційних робіт магістрів; 2) Навчальний план дисципліни.

3.2 Місце та тривалість випробувань

Приймальні випробування проводяться на базі комп'ютерного класу кафедри.

Підрозділ 3.3. «Обсяг випробувань»

Приймальні випробування проводяться відповідно до цієї Програми та методики випробувань і включають перевірку: працездатності програмного забезпечення, інтеграції з криптовалютними гаманцями, реалізації шифрування даних, функціонування смарт-контрактів.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться за участю Замовника, Виконавця та членів комісії, призначених для прийому курсової роботи.

4. Вимоги до програмного виробу

Вимогами вважається: забезпечення інтерактивного інтерфейсу для авторизації користувачів, сумісність із популярними криптовалютними гаманцями (MetaMask, Trust Wallet), надійність зберігання та шифрування даних за допомогою AES або іншого алгоритму, мінімальні затримки під час взаємодії із блокчейном, відповідність стандартам безпеки.

5. Вимоги до програмної документації

Програмною документацією до виробу «Модель криптографічного захисту даних користувачів у web-застосунках» вважати:

1) Справжнє Технічне завдання на розробку програмного виробу;

2) Програму та методику випробувань розробленого програмного виробу.

6. Засоби та порядок випробувань

6.1 Засоби випробувань

Для проведення випробувань програмного виробу необхідно забезпечити доступ до відповідного апаратного та програмного забезпечення. Основними технічними засобами є

комп'ютер із мінімальними характеристиками: оперативна пам’ять 4 ГБ, процесор із двома ядрами (2.5 ГГц) та стабільне підключення до Інтернету зі швидкістю не менше 10 Мбіт/с. Операційна система може бути Windows або Linux.
Також потрібно встановити криптовалютний гаманець MetaMask як браузерне розширення або додаток, а для взаємодії з блокчейном — надати доступ до тестової мережі, наприклад Ethereum Goerli. Крім того, слід забезпечити встановлення всіх необхідних бібліотек і програмних залежностей для роботи програмного виробу.

6.2 Порядок проведення випробувань

1) Якість програмної документації перевіряється на відповідність вимогам ДЕСТ 19.301-79 ЄСПД «Програма та методика випробувань».

2) Програма працює відповідно до умов експлуатації ОС Windows 10.

3) Для роботи необхідне встановлення зазначених розширень у репозиторії GitHub.

4) Порядок проведення випробувань:

-Для завантаження програмного виробу і ознайомлення з вихідним кодом можна скористатися репозиторієм на платформі GitHub: https://github.com/novitskiiy/tokenmaster

-Інструкції з інсталяції та використання програми наведені у файлі README.md, розміщеному в цьому репозиторії.

-Після успішного налаштування здійснити авторизацію через кнопку «Connect»;

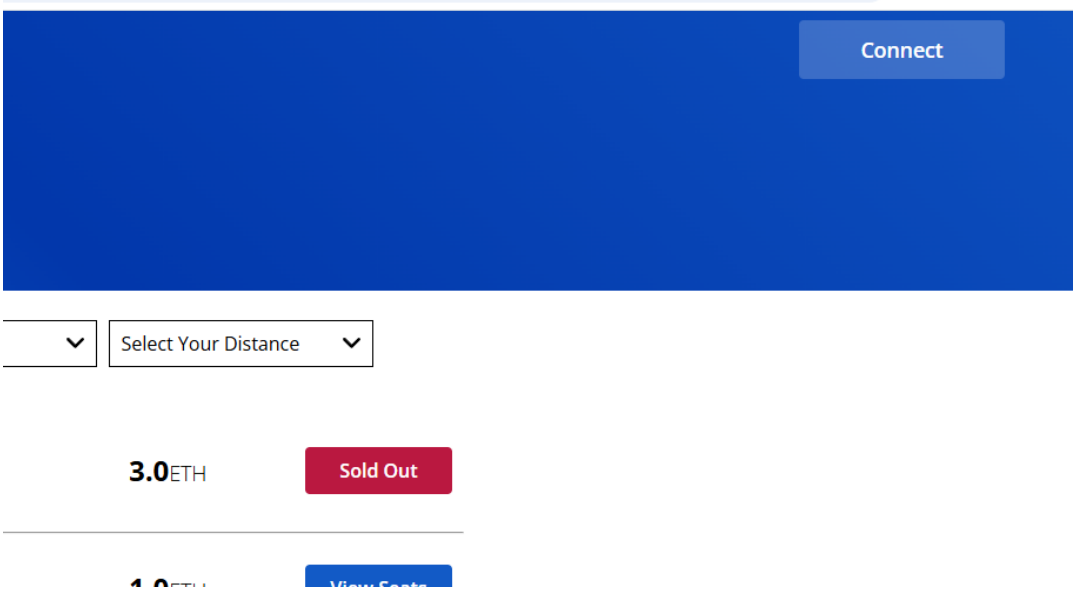

Рисунок В.1 — Авторизація

-Після авторизації обрати довільну подію, для перегляду вільних місць, та натиснути «View Seats»;

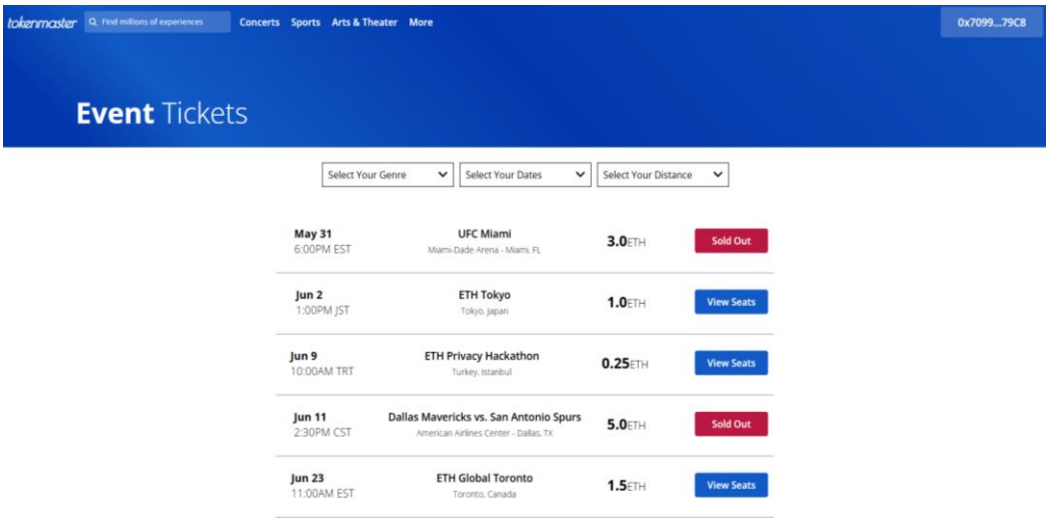

Рисунок В.2 — Вибір заходу

-Обрати будь-яке вільне місце;

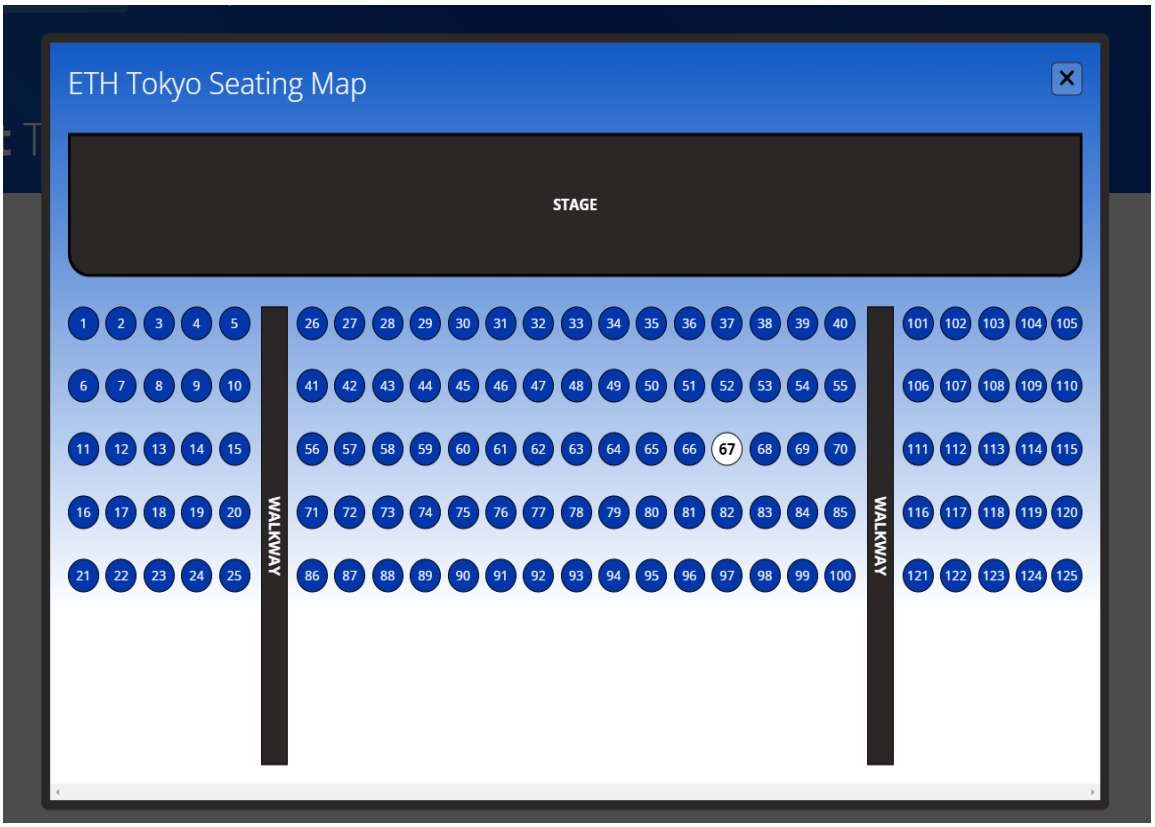

Рисунок В.3 — Карта розташування місць

-Здійснити оплату квитка через криптогаманець;

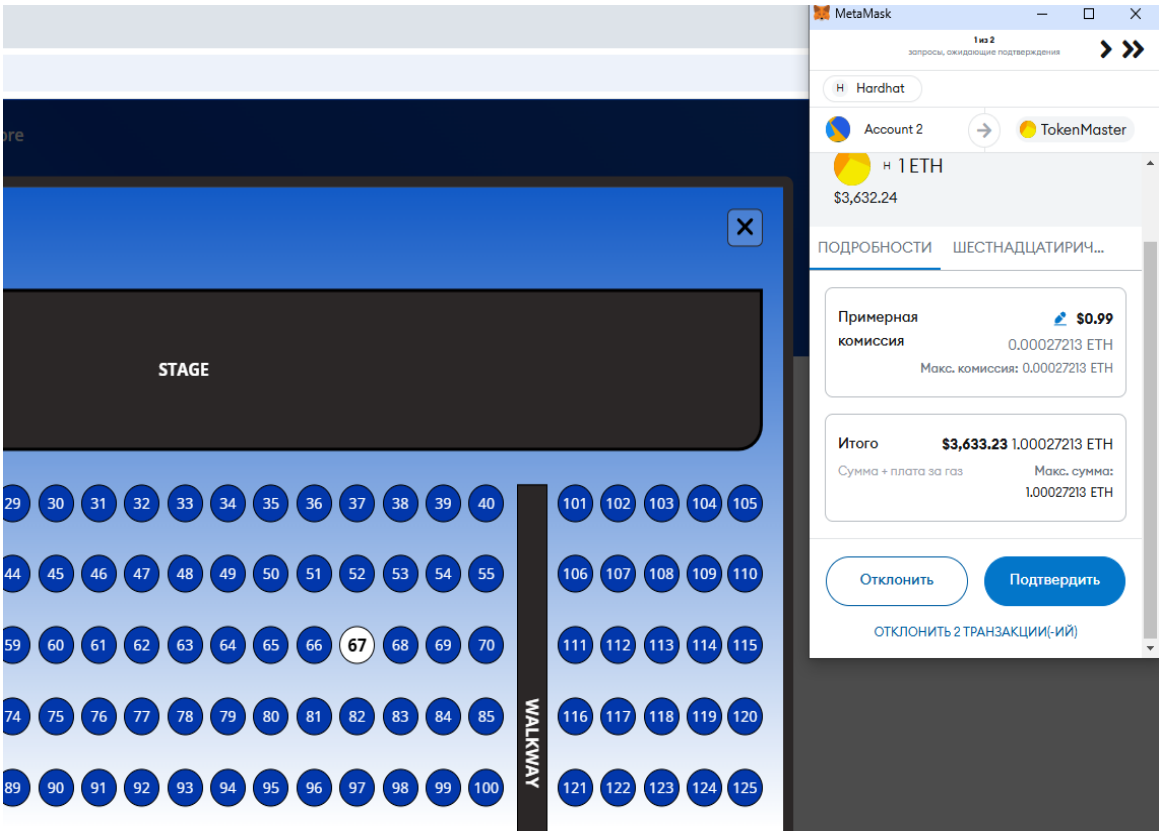

Рисунок В.4 — Процес здійснення транзакції

-Переконатися транзакція пройшла успішно та обране місце на карті розташування виділено забронюванним;

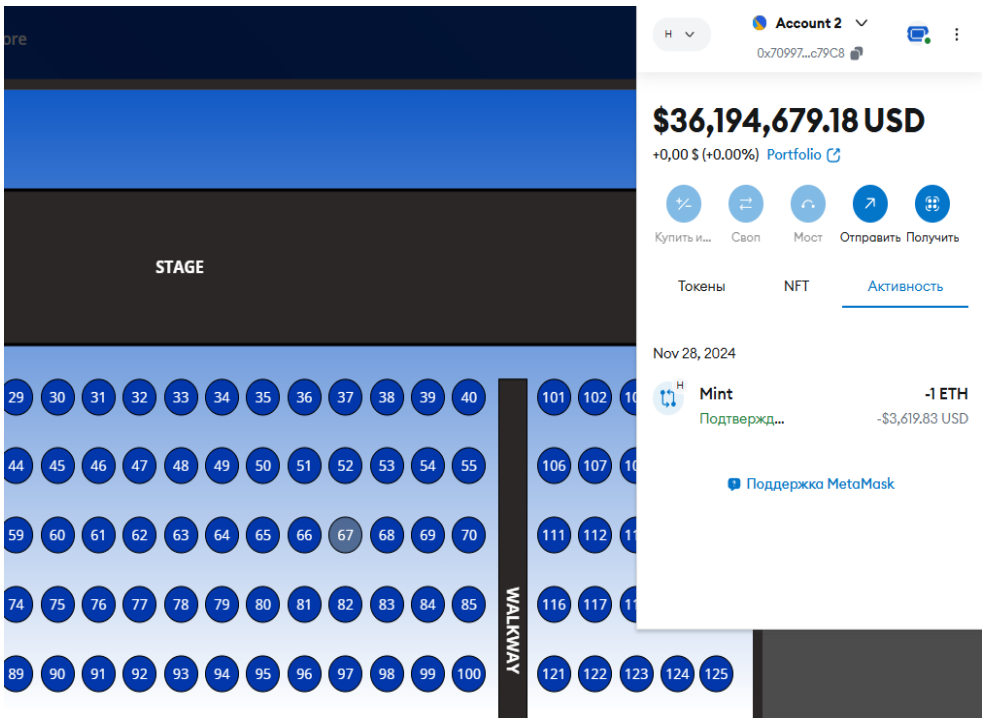

Рисунок В.5 — Виконана транзакція

5) Для проведення випробувань пропонується тест, опис якого міститься у додатку.

Тест вважається пройденим, якщо не виявлено жодних помилок компілятором нашої IDE.

6) Програмний виріб вважається таким, що пройшов випробування в цілому, якщо транзакція пройдена успішно, а також обране місце на подію заброньовано.

Виконавець  Веремієнко Б. О.

Додаток Г

Лістинг 1

```
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract TokenMaster is ERC721 {
    address public owner;
    uint256 public totalOccasions;
    uint256 public totalSupply;

    struct Occasion {
        uint256 id;
        string name;
        uint256 cost;
        uint256 tickets;
        uint256 maxTickets;
        string date;
        string time;
        string location;
    }

    mapping(uint256 => Occasion) occasions;
    mapping(uint256 => mapping(address => bool)) public
hasBought;
    mapping(uint256 => mapping(uint256 => address)) public
seatTaken;
    mapping(uint256 => uint256[]) seatsTaken;

    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    constructor(
        string memory _name,
        string memory _symbol
    ) ERC721(_name, _symbol) {
        owner = msg.sender;
    }

    function list(
        string memory _name,
        uint256 _cost,
        uint256 _maxTickets,
        string memory _date,
        string memory _time,
```

```
        string memory _location
    ) public onlyOwner {
        totalOccasions++;
        occasions[totalOccasions] = Occasion(
            totalOccasions,
            _name,
            _cost,
            _maxTickets,
            _maxTickets,
            _date,
            _time,
            _location
        );
    }

    function mint(uint256 _id, uint256 _seat) public payable
{
        // Require that _id is not 0 or less than total
occasions...
        require(_id != 0);
        require(_id <= totalOccasions);

        // Require that ETH sent is greater than cost...
        require(msg.value >= occasions[_id].cost);

        // Require that the seat is not taken, and the seat
exists...
        require(seatTaken[_id][_seat] == address(0));
        require(_seat <= occasions[_id].maxTickets);

        occasions[_id].tickets -= 1; // <-- Update ticket
count

        hasBought[_id][msg.sender] = true; // <-- Update
buying status
        seatTaken[_id][_seat] = msg.sender; // <-- Assign
seat

        seatsTaken[_id].push(_seat); // <-- Update seats
currently taken

        totalSupply++;

        _safeMint(msg.sender, totalSupply);
    }

    function getOccasion(uint256 _id) public view returns
(Occasion memory) {
        return occasions[_id];
    }
```

```
    function getSeatsTaken(uint256 _id) public view returns
(uint256[] memory) {
        return seatsTaken[_id];
    }

    function withdraw() public onlyOwner {
        (bool success, ) = owner.call{value:
address(this).balance}("");
        require(success);
    }
}
```

Лістинг 2

```
import { useEffect, useState } from 'react'
import { ethers } from 'ethers'

// Components
import Navigation from './components/Navigation'
import Sort from './components/Sort'
import Card from './components/Card'
import SeatChart from './components/SeatChart'

// ABIs
import TokenMaster from './abis/TokenMaster.json'

// Config
import config from './config.json'

function App() {
  const [provider, setProvider] = useState(null)
  const [account, setAccount] = useState(null)

  const [tokenMaster, setTokenMaster] = useState(null)
  const [occasions, setOccasions] = useState([])

  const [occasion, setOccasion] = useState({})
  const [toggle, setToggle] = useState(false)

  const loadBlockchainData = async () => {
    const provider = new
ethers.providers.Web3Provider(window.ethereum)
    setProvider(provider)

    const network = await provider.getNetwork()
    const tokenMaster = new
ethers.Contract(config[network.chainId].TokenMaster.address,
TokenMaster, provider)
    setTokenMaster(tokenMaster)
```

```
    const totalOccasions = await tokenMaster.totalOccasions()
    const occasions = []

    for (var i = 1; i <= totalOccasions; i++) {
      const occasion = await tokenMaster.getOccasion(i)
      occasions.push(occasion)
    }

    setOccasions(occasions)

    window.ethereum.on('accountsChanged', async () => {
      const accounts = await window.ethereum.request({
method: 'eth_requestAccounts' })
      const account = ethers.utils.getAddress(accounts[0])
      setAccount(account)
    })
  }

  useEffect(() => {
    loadBlockchainData()
  }, [])

  return (
    <div>
      <header>
        <Navigation account={account} setAccount={setAccount}
/>

        <h2 className="header__title"><strong>Event</strong>
Tickets</h2>
      </header>

      <Sort />

      <div className='cards'>
        {occasions.map((occasion, index) => (
          <Card
            occasion={occasion}
            id={index + 1}
            tokenMaster={tokenMaster}
            provider={provider}
            account={account}
            toggle={toggle}
            setToggle={setToggle}
            setOccasion={setOccasion}
            key={index}
          />
        ))}
      </div>

      {toggle && (
        <SeatChart
```

```
          occasion={occasion}
          tokenMaster={tokenMaster}
          provider={provider}
          setToggle={setToggle}
        />
      )}
    </div>
  );
}

export default App;
```

Лістинг 3

```
const hre = require("hardhat")

const tokens = (n) => {
  return ethers.utils.parseUnits(n.toString(), 'ether')
}

async function main() {
  // Setup accounts & variables
  const [deployer] = await ethers.getSigners()
  const NAME = "TokenMaster"
  const SYMBOL = "TM"

  // Deploy contract
  const TokenMaster = await
ethers.getContractFactory("TokenMaster")
  const tokenMaster = await TokenMaster.deploy(NAME, SYMBOL)
  await tokenMaster.deployed()

  console.log(`Deployed TokenMaster Contract at:
${tokenMaster.address}\n`)

  // List 6 events
  const occasions = [
    {
      name: "UFC Miami",
      cost: tokens(3),
      tickets: 0,
      date: "May 31",
      time: "6:00PM EST",
      location: "Miami-Dade Arena - Miami, FL"
    },
    {
      name: "ETH Tokyo",
      cost: tokens(1),
      tickets: 125,
      date: "Jun 2",
```

```
      time: "1:00PM JST",
      location: "Tokyo, Japan"
    },
    {
      name: "ETH Privacy Hackathon",
      cost: tokens(0.25),
      tickets: 200,
      date: "Jun 9",
      time: "10:00AM TRT",
      location: "Turkey, Istanbul"
    },
    {
      name: "Dallas Mavericks vs. San Antonio Spurs",
      cost: tokens(5),
      tickets: 0,
      date: "Jun 11",
      time: "2:30PM CST",
      location: "American Airlines Center - Dallas, TX"
    },
    {
      name: "ETH Global Toronto",
      cost: tokens(1.5),
      tickets: 125,
      date: "Jun 23",
      time: "11:00AM EST",
      location: "Toronto, Canada"
    }
  ]

  for (var i = 0; i < 5; i++) {
    const transaction = await
tokenMaster.connect(deployer).list(
      occasions[i].name,
      occasions[i].cost,
      occasions[i].tickets,
      occasions[i].date,
      occasions[i].time,
      occasions[i].location,
    )

    await transaction.wait()

    console.log(`Listed Event ${i + 1}:
${occasions[i].name}`)
  }
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

Лістинг 4

```
import { useEffect, useState } from 'react'
import Seat from './Seat'
import close from '../assets/close.svg'

const SeatChart = ({ occasion, tokenMaster, provider,
setToggle }) => {
  const [seatsTaken, setSeatsTaken] = useState(false)
  const [hasSold, setHasSold] = useState(false)

  const getSeatsTaken = async () => {
    const seatsTaken = await
tokenMaster.getSeatsTaken(occasion.id)
    setSeatsTaken(seatsTaken)
  }

  const buyHandler = async (_seat) => {
    setHasSold(false)

    const signer = await provider.getSigner()
    const transaction = await
tokenMaster.connect(signer).mint(occasion.id, _seat, { value:
occasion.cost })
    await transaction.wait()

    setHasSold(true)
  }

  useEffect(() => {
    getSeatsTaken()
  }, [hasSold])

  return (
    <div className="occasion">
      <div className="occasion__seating">
        <h1>{occasion.name} Seating Map</h1>

        <button onClick={() => setToggle(false)}
className="occasion__close">
          <img src={close} alt="Close" />
        </button>

        <div className="occasion__stage">
          <strong>STAGE</strong>
        </div>

        {seatsTaken && Array(25).fill(1).map((e, i) =>
          <Seat
            i={i}
            step={1}
```

```
            columnStart={0}
            maxColumns={5}
            rowStart={2}
            maxRows={5}
            seatsTaken={seatsTaken}
            buyHandler={buyHandler}
            key={i}
          />
        )}

        <div className="occasion__spacer--1 ">
          <strong>WALKWAY</strong>
        </div>

        {seatsTaken && Array(Number(occasion.maxTickets) -
50).fill(1).map((e, i) =>
          <Seat
            i={i}
            step={26}
            columnStart={6}
            maxColumns={15}
            rowStart={2}
            maxRows={15}
            seatsTaken={seatsTaken}
            buyHandler={buyHandler}
            key={i}
          />
        )}

        <div className="occasion__spacer--2">
          <strong>WALKWAY</strong>
        </div>

        {seatsTaken && Array(25).fill(1).map((e, i) =>
          <Seat
            i={i}
            step={(Number(occasion.maxTickets) - 24)}
            columnStart={22}
            maxColumns={5}
            rowStart={2}
            maxRows={5}
            seatsTaken={seatsTaken}
            buyHandler={buyHandler}
            key={i}
          />
        )}
      </div>
    </div >
  );
}

export default SeatChart;
```