МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Факультет Комп'ютерних наук
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів тестування систем на основі технології блокчейн

Виконала:
студент 2 курсу, групи ІПЗм-21-1
Шишло О.В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-наукова
Керівник доц., к.т.н. Кириченко І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2023

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної Інженерії

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Шишло Ользі Володимирівні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів тестування систем на основі технології блокчейн

затверджена наказом університету від « 29 » березня 2023 р. № 302 Ст

2. Термін подання студентом роботи до екзаменаційної комісії «15» травня 2023 р.

3. Вихідні дані до роботи *описаний процес тестування систем на основі блокчейну, тест-план, пояснювальна записка.*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз предметної галузі і постановка задачі, дослідження методів тестування систем на основі технології блокчейн, вивчення можливостей їх використання під час проектування процесу тестування систем на основі блокчейн-технології, висновки, перелік джерел посилань.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	30.01.2023	Виконано
2	Постановка задачі	06.02.2023	Виконано
3	Дослідження методів тестування систем на основі технології блокчейн	13.02.2023	Виконано
4	Планування експериментальної частини дослідження	10.03.2023	Виконано
6	Процес тестування системи на основі блокчейн технології	10.04.2023	Виконано
7	Аналіз результатів дослідження	02.05.2023	Виконано
8	Підготовка пояснювальної записки	13.05.2023	Виконано
9	Перевірка роботи на антиплагіат	09.05.2023	Виконано
10	Нормоконтроль	11.05.2023	Виконано
11	Рецензування	12.05.2023	Виконано
12	Підготовка презентації та доповіді	12.05.2023	Виконано
13	Попередній захист	13.05.2023	Виконано
14	Занесення роботи в електронний архів	13.05.2023	Виконано
15	Допуск до захисту в зав. кафедрою	15.05.2023	Виконано

Дата видачі завдання 23.01.2023 р.

Студент _____ Шишло О. В.
(підпис)

Керівник роботи _____ доц., к.т.н. Кириченко І. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Звіт до кваліфікаційної роботи містить: 100 с., 18 рис., 8 табл., 17 джерел, 6 додатків.

БЛОКЧЕЙН, ІНСТРУМЕНТИ, МЕТОДИ, СИСТЕМА, СМАРТ-КОНТРАКТИ, ТЕСТ, ТЕСТ-ПЛАН, ТЕСТУВАННЯ

Об’єктом дослідження є методи тестування систем на основі технології блокчейну, планування тестування.

Метою роботи є дослідження та визначення ефективних методів тестування систем, в основі яких лежить технологія блокчейн з ціллю оптимізації процесу забезпечення якості даних систем, шляхом якісного планування процесу тестування.

У результаті роботи було здійснено аналіз предметної галузі, поставлена задача для дослідження, проведено дослідження методів тестування систем на основі блокчейну та розроблено тест-план на основі результатів дослідження.

BLOCKCHAIN, METHODS, SMART-CONTRACTS, SYSTEM, TEST, TEST-PLAN, TESTING, TOOLS

The object of research is methods and tools for testing systems based on blockchain technology, test planning.

The purpose of this work is to identify effective methods for testing systems based on blockchain technology in order to optimize the process of ensuring the quality of these systems by means of high-quality planning of the testing process.

As a result of the work, the subject area was analyzed, the research task was set, the methods of testing blockchain-based systems were studied, and a test plan was developed based on the results of the study.

Я, Шишло Ольга Володимирівна, студентка гр. ІПЗм-21-1, здобувачка вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів тестування систем на основі технології блокчейн», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

В наших реаліях людина не може абсолютною мірою покладатися та довіряти іншим малознайомим їй людям. Особливо, коли справи стосуються персональної інформації. Для вирішення цієї потреби багато систем використовують ті чи інші алгоритми шифрування або ж технології задля забезпечення незмінності, достовірності та конфіденційності переданих чи отриманих даних. На сьогодні найпопулярнішою з них є технологія блокчейн. Це технологія, яка може бути використана для захисту, зберігання а також управління даними в децентралізованому та криптографічному форматі. Наразі вона повністю відповідає запитам користувачів стосовно безпеки їх даних.

Проте неможливо довіряти системам, що не були протестовані фахівцями належним чином. Саме тому одним з найважливіших етапів розробки систем є їх тестування. І системи, що використовують блокчейн технологію, не виняток Дане дослідження буде актуальним для QA-спеціалістів, що займаються плануванням процесу тестування.

На жаль, існує тенденція, що досить високий рівень покриття тестами зазвичай мають лише системи, що безпосередньо стосуються грошових переказів. Хоча дану технологію використовують досить багато різних галузей та сфер.

Аналіз предметної галузі надасть змогу визначити «підґрунтя» для проведення дослідження. Після чого необхідно сформувати завдання самого дослідження, результатом якого стане рішення щодо популяризації та спрощення процесу тестування систем на основі блокчейн технології. А також в ході роботи необхідно проаналізувати процес планування тестування, дослідити методи тестування систем на основі блокчейн технології.

Об’єктом дослідження є якість та ефективність процесу тестування систем на основі технології блокчейн.

Предметом дослідження є методи тестування систем на основі технології блокчейн.

Методами дослідження є проведення процесу тестування системи на основі блокчейн-технології та вимірювання критичності та вартості знайдених багів.

Метою роботи є визначення ефективних методів тестування систем, в основі яких лежить технологія блокчейн з ціллю оптимізації процесу забезпечення якості даних систем, шляхом якісного планування процесу тестування.

Для досягнення поставленої мети необхідно вирішити наступні питання:

– провести аналіз базових та специфічних для блокчейн-систем методів тестування;

– обрати систему на основі блокчейн технології, яка буде тестуватись з використанням обраних методів тестування;

– розробити тест-план для процесу тестування обраної блокчейн-системи;

– протестувати обрану блокчейн-систему за розробленим тест-планом;

– проаналізувати результати дослідження та надати рекомендації з використання методів тестування для систем на основі технології блокчейн.

Отримані результати дослідження можуть бути використані під час планування процесу тестування різноманітних систем на основі технології блокчейн.

Результати даної кваліфікаційної роботи було представлено на «7th International Conference on Computational Linguistics and Intelligent Systems» CoLinS-2023 (Scopus) April 20–21, 2023 [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної галузі

Блокчейн це децентралізований, розподілений і цифровий реєстр, який використовується для документування транзакцій (блоків) на багатьох комп'ютерах або ж, інакше кажучи, вузлах, так, що запис не може бути змінений «заднім числом» без зміни всіх наступних блоків і без консенсусу мережі [2].

Ключові характеристики, які роблять блокчейн унікальним:

– децентралізованість;
– технологія базується на основі протоколу консенсусу;
– незмінність;
– прозорість;
– цензура.

Розглянемо детальніше кожну з характеристик даної технології.

Децентралізованість. Транзакції між мережами не підпорядковуються якомусь конкретному центральному органу, і жодна конкретна організація не може претендувати на ексклюзивний контроль над даними або ж процесами.

Так як технологія заснована на протоколі консенсусу, кожна транзакція вимагатиме згоди всіх відповідних сторін для того, щоб бути дійсною.

Блокчейн підтримує незмінність, таким чином неможливо стерти або замінити вже записані дані [3]. Завдяки цій характеристиці технологія запобігає фальсифікації даних у мережі, що забезпечує довіру до кожної з транзакцій.

Прозорість. Історія кожної транзакції записується і є доступною для будь-якої сторони, яка бере участь в аудиті. Таким чином є можливість відстежувати термін служби активів.

Технологія блокчейн вільна від цензури, оскільки не контролюється жодною стороною [4]. Саме тому влада, в тому числі уряд, не може перервати роботу мережі.

Віра Будхі в своїй статті описала загальні недоліки блокчейн технології [5], до них можна віднести наступні характеристики:

– низькі швидкість і продуктивність;
– висока вартість впровадження;
– модифікація даних.

Блокчейн значно повільніший за традиційну базу даних, оскільки виконує більше операцій. Блокчейн верифікує криптографічне підписання транзакцій [6]. Також слід зазначити, що він покладається на механізм консенсусу для підтвердження транзакцій. Деякі з цих механізмів, такі як підтвердження роботи, мають низьку пропускну здатність транзакцій [7], наприклад, механізм підтвердження виконання роботи («proof-of-work»). Існує також надмірність, коли мережа вимагає, щоб кожен вузол відігравав важливу роль у перевірці та зберіганні кожної транзакції.

Блокчейн дорожчий в порівнянні з традиційною базою даних. Окрім того, підприємства потребують відповідного планування та виконання, щоб інтегрувати блокчейн у свій бізнес-процес [8].

Модифікація даних. Технологія блокчейн не дозволяє легко модифікувати дані після того, як вони були записані, а вимагає переписування кодів у всіх блоках, що займає багато часу і коштує дорого. Недоліком цієї функції є складність виправлення помилок та внесення певних необхідних корективів.

Не зважаючи на недоліки, використання технології блокчейн дозволяє користувачам:

– проводити транзакції без посередника;
– створювати нові форми цифрових активів;
– вирішувати проблему подвійного витрачання;
– забезпечувати незмінність інформації;
– стежити за власними даними та активами.

Таким чином дана технологія може вирішити одну з найважливіших проблем в комерційних відносинах – проблему довіри та витоку даних між їх учасниками.

Розглянемо головні компоненти блокчейну (див. рис. 1), а саме:

– вузловий додаток;
– спільний реєстр;
– алгоритм консенсусу;
– віртуальна машина.

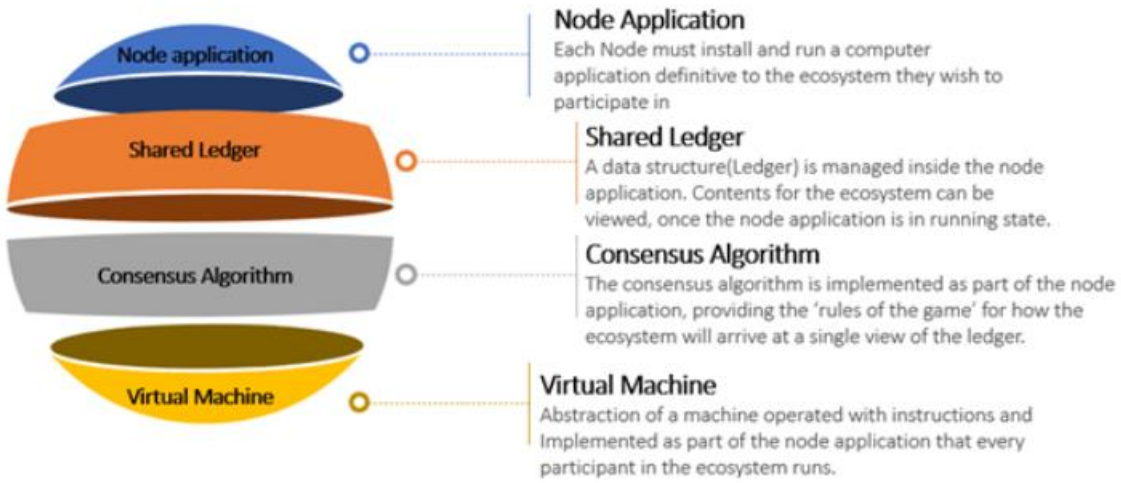

Рисунок 1 – Компоненти блокчейну (https://www.guru99.com/)

Вузловий додаток – кожен вузол повинен встановити та запустити комп'ютерний додаток, що відповідає екосистемі, в якій він бажає брати участь.

Структура даних (реєстр) управляється всередині програми вузла. Вміст екосистеми можна переглянути, як тільки додаток вузла буде запущено.

Алгоритм консенсусу реалізований як частина вузлового додатку, забезпечуючи «правила гри» для того, як екосистема прийде до єдиного вигляду реєстру.

Віртуальна машина – абстракція машини, що працює за допомогою інструкцій і реалізована як частина вузлового додатку, який запускає кожен учасник екосистеми.

Наведена нижче наступна схема допоможе проілюструвати, як працює транзакція в технології блокчейн (див. рис. 2).

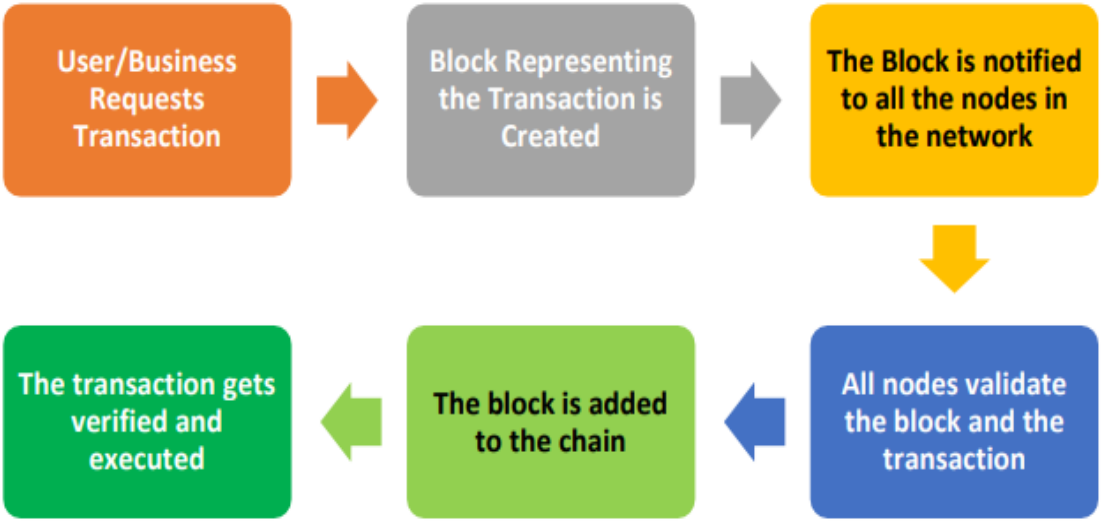

Рисунок 2 – Схема роботи технології блокчейн (https://www.guru99.com/)

Коли користувач відправляє запит на транзакцію, створюється відповідний блок, що представляє дану транзакцію. Після чого даний блок повідомляється всім вузлам мережі. Тільки після того, як блок та відповідна йому транзакція були підтверджені всіма вузлами мережі, блок може бути доданий до ланцюга.

Блокчейн був розроблений спочатку для цифрових валют, таких як біткоїн, але пізніше його переваги призвели до використання для запису не тільки фінансових транзакцій, але й практично всього, що має вагому цінність, від охорони здоров'я до банківської справи та роздрібної торгівлі.

На сьогоднішній день технологія блокчейн має дуже широку сферу використання. Проте належним чином тестуються зазвичай лише системи, що безпосередньо пов’язані з грошовими переказами.

1.2 Постановка завдання дослідження

Відповідно до теми роботи, задачею цього дослідження є аналіз та порівняння методів, що використовуються для тестування систем, в основі яких лежить технологія блокчейн. Виходячи з усього перерахованого вище, в рамках дослідження необхідно вирішити наступні завдання:

– розглянути етапи планування процесу тестування;

– провести аналіз базових та специфічних для блокчейн-систем методів тестування;

– обрати систему на основі блокчейн технології, яка буде тестуватись з використанням обраних методів тестування;

– розробити тест-план для процесу тестування обраної блокчейн-системи;

– протестувати обрану блокчейн-систему за розробленим тест-планом;

– проаналізувати результати дослідження та надати рекомендації з використання методів тестування для систем на основі технології блокчейн.

Після проведення теоретичного аналізу його результати мають бути підтверджені за допомогою експерименту. Для цього має бути визначено фактори впливу на вартість та вагу багів, а також розроблено деталізований тест-план.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ТЕСТУВАННЯ СИСТЕМ НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

2.1 Планування процесу тестування

У зв'язку зі зміною технологій тестування додатків, розроблених на основі технології блокчейн, пов'язане з численними проблемами. Щоб цьому запобігти, слід якісно спланувати весь процес [9].

Тестування систем на основі блокчейн технології складається з наступних етапів [10]:

– ініціація;
– проектування;
– тестування;
– підготовка звітів.

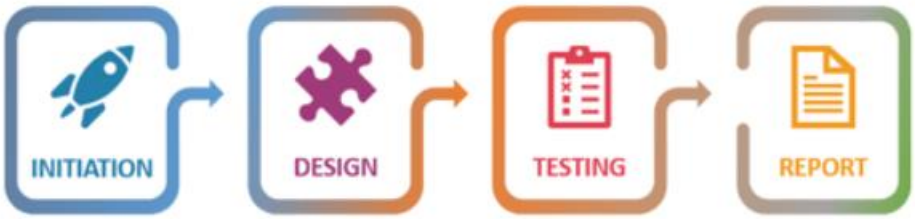

Рисунок 3 – Етапи тестування систем на основі блокчейн технології (https://www.guru99.com/)

Як зображено на рисунку 3, першим етапом є етап ініціації. Він передбачає розуміння архітектури блокчейну, а саме розуміння та аналіз як бізнес-вимог, так і функціональних. Тобто опис поведінки додатку і те, як користувач буде з ним взаємодіяти. Також на даному етапі необхідно розробити повну стратегію тестування. Під час розробки стратегії слід детально описати підхід до тестування додатку.

Наступним кроком є проектування. Цей етап складається з декількох фаз:

– створення тест-стратегії та/або тест-плану;

– написання тест-кейсів: команда QA спеціалістів створює тест-кейси з відповідними кроками для кожної з перевірок (бажано, щоб вони перевірялись бізнес аналітиками);

– підготовка тестових даних: необхідно підготувати тестові дані на основі бізнес-вимог до продукту, вони можуть бути, як створені вручну, так і за допомогою різноманітних інструментів;

– налаштування середовища: в рамках цієї фази передбачається налаштування середовища, на якому буде проводитись тестування;

– метрики продуктивності: на цьому кроці необхідно визначити показники для представлення інформації щодо продуктивності цілої системи або ж її компонентів.

Особливу увагу на етапі проектування слід звернути на створення таких тестових артефактів, як тест-стратегія та тест-план.

Тест-стратегія – високорівневий документ, що містить опис рівнів тестування і підходів до тестування в межах цих рівнів. Діє на рівні компанії або програми (одного або більше проектів).

Тест-план – документ, що описує засоби, підходи, графік робіт і ресурси, необхідні для проведення тестування. Крім того, визначає інструменти тестування, функціональність, яку потрібно протестувати, розподіл ролей в команді, тестове оточення, техніки тест-дизайну, що використовуватимуться, а також критерії початку та закінчення тестування та ризики.

Якісно спроектований тест план повинен містити в собі наступну інформацію:

– об’єкт тестування – що саме буде тестуватись, середовище тестування, додаткове обладнання та додатки;

– повний перелік функціоналу, який має бути перевірений;

– повний перелік методів тестування, які повинні бути використані, їх застосування стосовно об’єкта тестування;

– перелік інструментів, що повинні бути використані під час процесу тестування;

– логічно описана структура перевірок: підготовчі роботи, безпосередньо процес тестування, аналіз отриманої інформації стосовно запланованих фаз розробки проекту.

Також слід зазначити, що існує 2 типи тест-планів:

– майстер тест-план;

– деталізований тест-план.

Майстер тест-план вважається більш статичним, адже містить у своїй структурі високорівневі дані, що не піддаються постійному використанню в процесі здійснення перевірки контролю якості.

Що ж стосується деталізованого тест-плану, то в ньому можна знайти більш конкретну інформацію про стратегію оптимального тестування, повний розклад виконаних робіт. Тобто, цей вид плану можна вважати більш «живим» джерелом тестових маніпуляцій, який постійно піддається редагуванню і в ньому можна знайти більш реальний стан речей під час тестування продукту, що розробляється.

2.2 Методи тестування систем на основі технології блокчейн

Традиційне тестування включає в себе наступні типи [11]:

– функціональне тестування;

– нефункціональне тестування;

– тестування продуктивності;

– тестування безпеки;

– тестування інтеграції.

На додаток до традиційного тестування та валідації слід мати спеціалізовані можливості, такі як:

– тестування смарт-контрактів;

– тестування однорангових/вузлових мереж;

– власні математичні та криптографічні навички;

– провідні в галузі інструменти.

Поєднаємо класичні методи тестування зі специфічними методами тестування систем, що базуються на технології блокчейн (див. рис. 4).

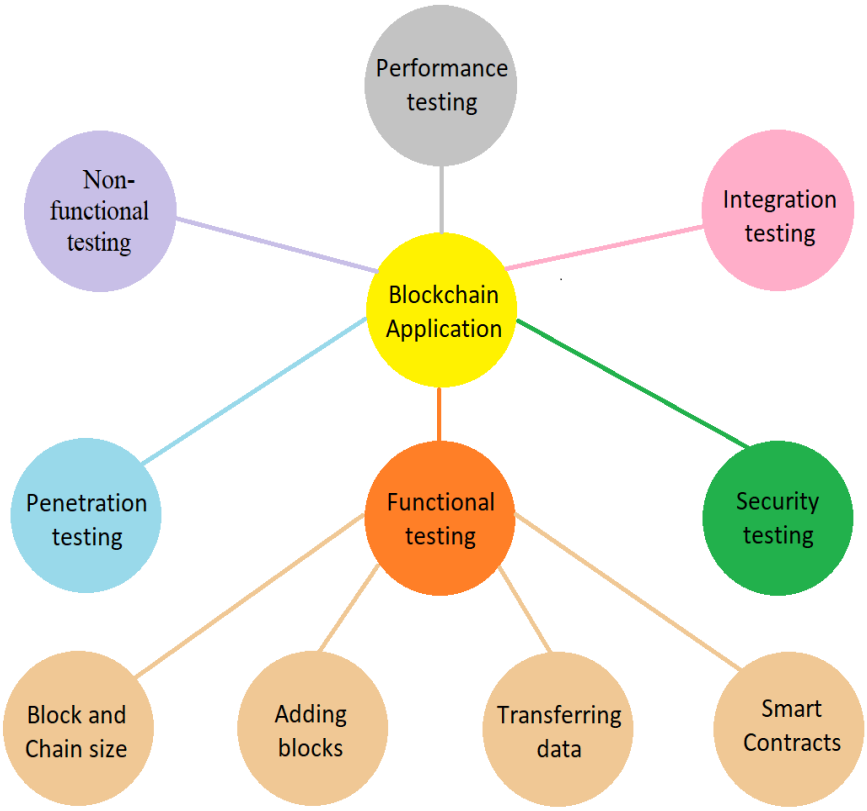

Рисунок 4 – Методи тестування блокчейн додатків

Функціональне тестування відіграє життєво важливу роль в оцінці бізнес-обставин та ефективності сценаріїв використання. Нижче наведені ключові моменти функціонального тестування в блокчейн-додатках [12]:

– перевірка розміру блоку та ланцюга;
– перевірка можливості передачі даних;
– перевірка можливості додавання блоків;
– перевірка смарт-контрактів.

Розмір блоку: Кожен блок в блокчейні має розмір пам'яті в мегабайтах, і був зменшений до 1 МБ з 36 МБ з міркувань безпеки. Тестувальники повинні зосередитися на тому, які дані про транзакції необхідно обрати, які методи шифрування слід використовувати для з'єднання цих блоків, і подібні складні сценарії.

Передача даних: Втрата даних під час передачі між блоками повинна бути протестована, оскільки основна архітектура ланцюжка блоків обертається навколо транзакцій і безпеки даних.

Додавання блоку: Блоки, які додаються в ланцюжок, повинні бути ретельно оцінені, оскільки після додавання в ланцюжок вони не можуть бути змінені.

Смарт-контракт: Переконання в тому, що сторони, які беруть участь в транзакціях, дотримуються правил смарт-контракту забезпечить безперебійне функціонування блокчейн-додатку.

Тестування вузлів: Всі різноманітні вузли, присутні в мережі, повинні тестуватися незалежно один від одного. Це забезпечить їх безперебійного функціонування.

Що ж стосується інтеграційного тестування, оскільки блокчейн – це екосистема, яка складається з різних компонентів, тому всі вони повинні бути пов'язані між собою. Крім того, дуже важливо, щоб різні API, пов'язані з цими компонентами, були протестовані на сумісність один з одним.

Тестування продуктивності в блокчейні є важливим з точки зору кількості транзакцій і розміру транзакції, що тестується на продуктивність блоку, або додатку, що готується до розгортання у виробництво. Інші важливі та залежні параметри включають мережу, послідовність транзакцій на кожному вузлі, швидкість обробки транзакцій, користувальницький і системний інтерфейс, а також відповіді, які вимагаються від смарт-контрактів.

Тестування розміру мережі та її здатності обробляти транзакції є критично важливим, оскільки дозволяє виявити вузькі місця в апаратному і програмному забезпеченні до розгортання, а також витрати, понесені на запуск програми в хмарі, або будь-якому іншому відповідному середовищі. Наскрізні сценарії розглядаються для забезпечення загальної продуктивності середовища блокчейн.

Також слід зазначити, що одна з головних цілей тестування – це визначення того, чи є додаток вразливим до атак, та чи є системи авторизації та аутентифікації надійними.

Тестування безпеки додатково розглядає інші важливі аспекти, наприклад, конфіденційність, цілісність, відсутність відмови в обслуговуванні, доступність тощо. Тестування безпеки виявляється важливим у випадку злому рівня ідентифікації блокчейн-додатку.

Транзакції, які знаходяться в процесі виконання під час виявлення злому шару ідентифікації, не можуть бути негайно зупинені. Таким чином, тестування безпеки повинно проводитися з метою виявлення всіх потенційних зломів шару ідентифікації.

Тестування безпеки додатків на основі блокчейну також включає в себе виклики або тестування таких компонентів, як методи підпису гаманців, приватні ключі, консенсус алгоритм консенсусу та залежність від платформи додатку.

Іншим важливим фактором для тестування безпеки є те, що шахрайські транзакції є незворотні, оскільки в технології блокчейн відміна транзакції майже неможлива.

Наступним методом тестування, який повинен бути включений в процес тестування, є тестування на проникнення [13].

Кібератаки та шахрайські хакерські атаки надзвичайно складно здійснити в системах на основі блокчейну, коли смарт-контракти працюють коректно [14]. Однак будь-яка помилка в складних математичних і програмних правилах може стати гарною можливістю для неавторизованих користувачів зі зловмисними намірами.

Тестування на проникнення в блокчейн дозволяє QA-інженерам створювати і виконувати тестові кейси, які імітують поведінку кібератак, щоб побачити, чи будуть заблоковані дії будь-якого несанкціонованого користувача в системі. Цей метод включає в себе тестування, яке імітує реальні сценарії зі спробами входу в систему через мережеві сервіси, бездротові мережі, мережі соціальної інженерії та блокчейн-додатки.

Тестування та забезпечення якості, безумовно, є дуже важливим і критичним аспектом для будь-якого впровадження блокчейну. Наслідки будь-якого інциденту в режимі реального часу можуть бути серйозними і призвести до значних

фінансових втрат, в залежності від галузі або застосування. Ця точка зору зосереджена на функціональних аспектах тестування зразка блокчейн-додатку та ключових областях, які необхідно тестувати для таких додатків, включаючи рекомендації щодо фреймворків для автоматизації тестування, що можуть допомогти виконати ці тести ефективно.

2.3 Інструменти для тестування систем на основі блокчейну

Визначення правильного інструменту для тестування блокчейн-додатків є життєво важливим кроком для ефективного та успішного тестування.

До критичних фаз підходу до тестування систем на основі блокчейну можна віднести наступні (див. рис. 5):

– першим кроком необхідно визначити платформу;

– після цього слід визначити критичні точки тестування;

– наступним кроком є визначення правильного інструменту тестування

– останній крок – це розгортка інструменту, тестування та стабілізація додатку.

Рисунок 5 – Критичні фази підходу до тестування (https://www.guru99.com/)

Нижче наведено кілька інструментів, які допомагають у тестуванні блокчейн-додатків та гарантують, що вони функціонують належним чином [15]:

– Ethereum Tester;

– Ganache (Testrpc);

– Hyperledger Composer;

– Exonum Testkit;

– BitcoinJ;

– Populus;

– Manticore;
– Corda Testing Tools;
– Embark Framework;
– Truffle.

Інструмент «Ethereum», одна з найбільш використовуваних платформ для створення блокчейн-додатків, має безліч інструментів, які можуть полегшити як розробку, так і тестування додатків.

«Ethereum Tester» надійний для інтеграції з Web3, API, Smart Contracts, Backend та деяких інших блокчейн-тестів. Тестові мережі (Ropsten, Kovan, і Rinkeby) імітують виробництво, подібне до блокчейну (де знаходиться ваш реальний ефір і токени). Це допомагає як розробникам, так і тестувальникам моделювати. На рисунку 6 можна побачити головну сторінку графічної версії інструменту.

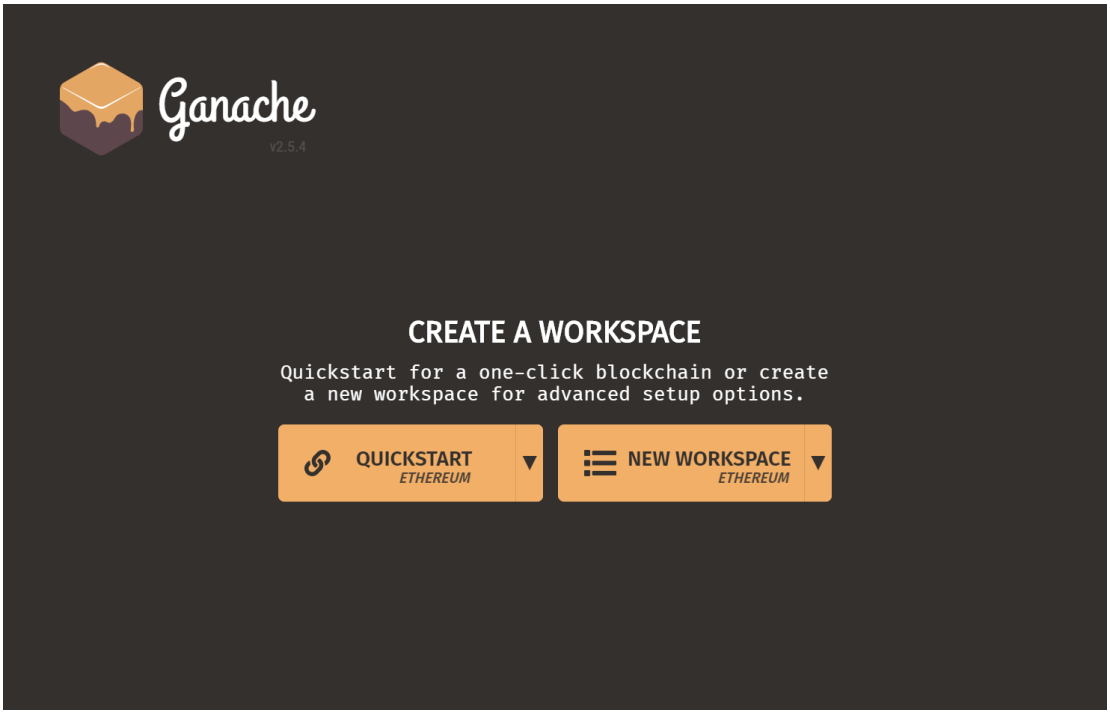

Рисунок 6 – Головний екран графічної версії інструменту «Ganache» (https://trufflesuite.com/ganache/)

Інструмент «Ganache (Testrpc)», раніше відомий як «Testrpc», це широко використовувана бібліотека для локального тестування контрактів Ethereum. Він створює симуляцію блокчейну, яка дозволяє будь-кому використовувати кілька облікових записів для тестування. Оскільки результати тестування отримані на основі симуляції, а не реальної події, вони будуть задовільними, але не ідеальними.

«Hyperledger Composer» – це інструмент з відкритим вихідним кодом, який дозволяє моделювати і тестувати вашу блокчейн мережу за допомогою мінімального набору інструментів, таких як Docker і браузер. Він полегшує розробку блокчейну за допомогою мови моделювання, інтерфейсу користувача та CLI. Це дозволяє автоматизоване системне тестування, інтерактивне тестування та автоматизоване модульне тестування. За допомогою цього інструменту ми можемо виконувати в основному три види тестування: інтерактивне тестування, автоматизоване модульне та системне тестування.

Тестування діяльності всього сервісу блокчейн-додатку – це спеціалізація «Exonum Testkit». Інструмент дозволяє проводити тестування API і виконання транзакцій без необхідності роботи мережі та алгоритму консенсусу.

«BitcoinJ» – клієнтська бібліотека Bitcoin з відкритим вихідним кодом, побудована за допомогою Java та реалізує мережевий протокол Bitcoin. Вона може підтримувати гаманець, відправляти/отримувати транзакції, не потребуючи локальної копії Bitcoin Core, та має багато інших розширених можливостей. Хоча вона розроблена на JAVA, її можна використовувати з будь-якої JVM-сумісної мови, такої як Python, JAVAScript тощо.

«Populous» – це пірингова платформа для виставлення рахунків. Вона використовує технологію блокчейн розподіленого реєстру, щоб забезпечити глобальну торгову платформу для фінансування рахунків-фактур. Фінансування рахунків-фактур – це форма фінансування, яка миттєво розблоковує грошові кошти, пов'язані з неоплаченими рахунками-фактурами. Власники бізнесу дозволяють покупцям рахунків-фактур купувати рахунки-фактури зі знижкою, щоб розблокувати свої грошові кошти швидше.

Інструмент «Manticore» використовується для проведення тестування безпеки блокчейн-додатків. «Manticore» – це інструмент символьного виконання для аналізу бінарних файлів та смарт-контрактів.

«Corda» – це розподілений реєстр з відкритим вихідним кодом на основі блокчейну . Він має вбудовану функцію тестування, яка допоможе з:

– написанням контрактних тестів;
– тестуванням інтеграції;
– написанням потокових тестів;
– навантажувальним тестуванням.

«Embark» – це фреймворк для створення, тестування та розгортання блокчейн додатків. Він дозволяє розробляти та розгортати децентралізовані додатки. Децентралізований додаток використовує одну або декілька децентралізованих технологій.

Наразі «Embark» інтегрується з децентралізованими сховищами (IPFS), блокчейнами EVM (Ethereum) та децентралізованими комунікаційними платформами децентралізованої комунікації (Whisper та Orbit). Для розгортання підтримується Swarm.

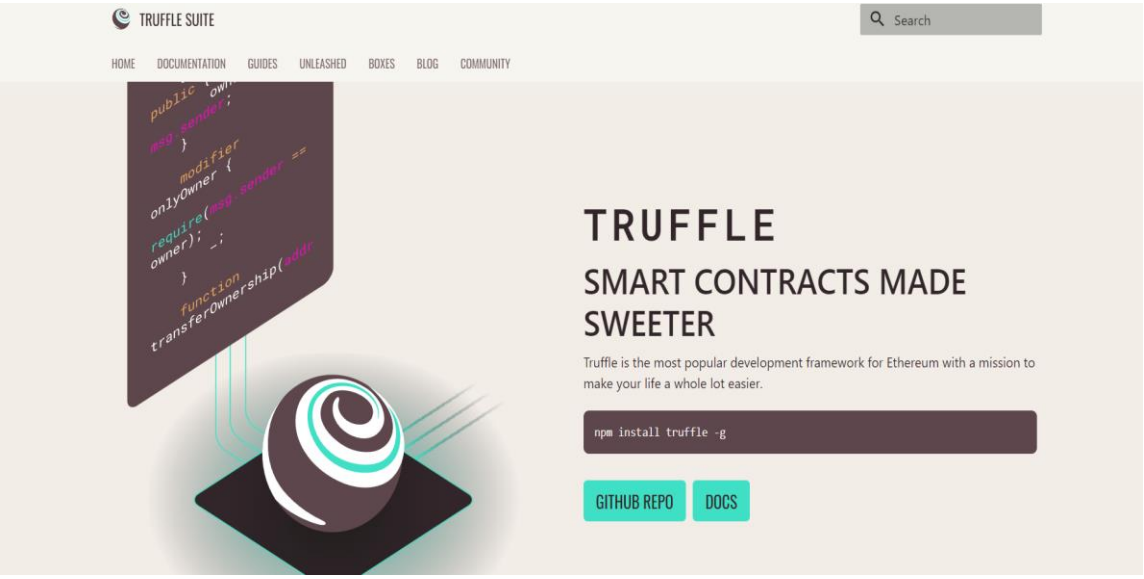

Рисунок 7 – Головний екран графічної версії інструменту «Truffle» (https://trufflesuite.com/)

«Truffle» – це добре знайомий інструмент для розробників Ethereum з різноманітними функціями тестування, включаючи автоматизоване тестування контрактів (див. рис. 7). Цей фреймворк має функціональність, що виходить за рамки простого тестування, і його варто додати в набір інструментів для тестування.

Також слід зазначити, що «Truffle» – це середовище розробки, фреймворк для тестування та конвеєр активів для блокчейнів з використанням віртуальної машини Ethereum (EVM) світового класу, що має на меті полегшити життя розробника. Truffle вважається найпопулярнішим інструментом для розробки блокчейн-додатків з більш ніж 1,5 мільйонами завантажень за весь час існування.

Вибір допоміжних інструментів під час тестування також є досить важливим моментом. Зазвичай, цей вибір обґрунтовується на етапі планування процесу тестування, адже він впливає, як на ефективність, швидкодію та на сумарну вартість даного процесу.

2.4 Аналіз факторів впливу на вартість та критичність багу в системі на основі блокчейн технології

В нашому житті кожна помилка має свою ціну. Аналогічно і кожен баг має власну вартість. Це можуть бути, як прямі витрати його виправлення, так і непрямі, наприклад, втрачений час на розробку, зіпсовані відносини з клієнтами, витік конфіденційної інформації і, як результат, втрачений бізнес.

Розглянемо нижче основні фактори, які впливають на вартість багу:

– етап тестування, на якому баг був знайдений;
– складність фіксу з технічної точки зору;
– час, витрачений на фікс;
– складність перевірки фіксу;
– час, витрачений на перевірку фіксу;
– необхідність додаткових релізів.

Кент Бек писав у своїй книзі, що більшість дефектів в кінцевому підсумку коштують більше, ніж коштувало б їхнє запобігання [16].

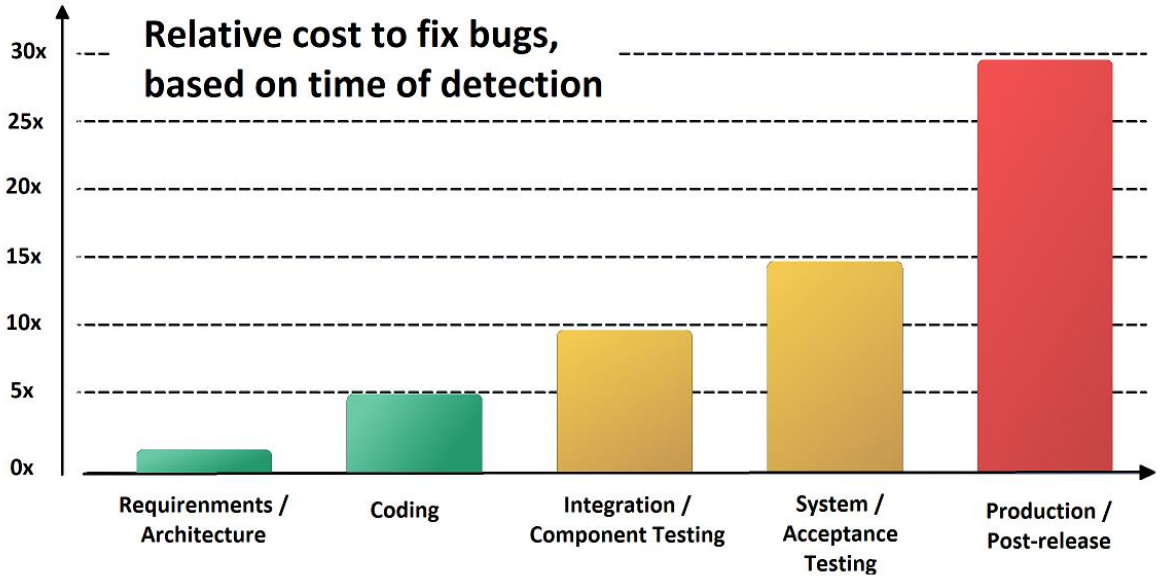

Рисунок 8 – Відносна вартість виправлення помилок залежно від часу виявлення

На рисунку 8 наведено відносну вартість бага від етапу на якому він був знайдений.

Також слід зазначити, що час, необхідний на фікс та його перевірку на пряму залежить від рівня навичок спеціалістів, що будуть задіяні в даному процесі, та від їх складності з технічної точки зору.

Необхідність додаткових релізів має залежність від етапу тестування, на якому баг був знайдений та локалізований.

Нижче наведено основні фактори, що впливають на критичність або ж вагу бага:

– серйозність бага;
– пріоритет бага;
– потенційні ризики після фіксу;
– потенційні ризики, якщо проблема не була усунена.

Серйозність бага або ж його критичність з технічної точки зору та з точки зору бізнес-логіки характеризує рівень впливу бага на загальну функціональність системи

Пріоритет бага характеризує на скільки швидко він має бути усунений, адже дана проблема може впливати на загальне враження користувачів від системи та на бізнес компанії-замовника.

Слід зазначити, що потенційні ризики слід розглядати невід'ємно один від одного під час вирішення, чи варто включати фікс бага в нову версію системи. Якщо ж потенційні ризики після фіксу є досить високими, а ризики, якщо фікс не був усунений, значно нижчі, то вартість такого фіксу може бути збільшена вдвічі або ж втричі. Наприклад, якщо фікс може породити більш серйозні та пріоритетні баги, на усунення яких буде витрачено набагато більше часу, проте проблема не є вагомою, якщо її не усунути.

Потенційні ризики також залежать від етапу тестування, на якому була ідентифікована проблема. Тому на наступних етапах дослідження особливу увагу слід приділити саме цьому фактору, адже він впливатиме, як на вагу, так і на вартість багів.

3 ПЛАНУВАННЯ ЕКСПЕРИМЕНТАЛЬНОЇ ЧАСТИНИ ДОСЛІДЖЕННЯ

3.1 План експерименту

Експериментальна частина дослідження спрямована на перевірку ефективності розробленого тест-плану для системи в основі якої лежить технологія блокчейн.

Експеримент складатиметься з наступних етапів:

– опис програмної системи на основі блокчейн технології, яка буде тестуватись в рамках дослідження;

– розробка тест-плану для процесу тестування обраної системи;

– тестування системи за розробленим тест-планом;

– аналіз отриманих результатів процесу тестування.

Під час розробки тест-плану будуть враховані отримані результати теоретичної частини дослідження.

На етапі планування експериментальної частини дослідження вкрай важливо зазначити систему оцінювання рівня впливу на вагу та вартість конкретної знайденої проблеми.

3.2 Визначення системи оцінювання вартості багу

Нижче описано систему оцінювання факторів впливу на вартість багу.

Складність перевірки фіксу пропонується оцінювати за такою градацією:

– «High»: дана оцінка говорить про необхідність високого рівня кваліфікації QA спеціаліста, що займатиметься перевіркою;

– «Medium»: дана оцінка говорить про необхідність середнього рівня кваліфікації QA спеціаліста, що займатиметься перевіркою;

– «Low»: дана оцінка говорить про те, що даною перевіркою може зайнятись QA спеціаліст з мінімальним досвідом та рівнем кваліфікації.

Час на перевірку фікса пропонується оцінювати в «чистих» годинах. Тобто буде враховуватись час за умови, що QA спеціаліст займатиметься лише цією задачею.

Необхідність додаткових релізів пропонується оцінювати наступними позначками:

– «No»: додаткові релізи не потрібні;

– «Stg»: потрібен додатковий реліз на стейджинг;

– «Prod»:потрібен додатковий реліз на продакшн.

Для зазначення етапу тестування на якому було ідентифіковано проблему пропонується використовувати наступні позначення:

– «Unit»: етап юніт-тестування;

– «Integration»: етап інтеграційного тестування;

– «System»: етап системного тестування;

– «UAT»: етап приймального тестування.

Так як оцінку складності фікса з технічної точки зору та час, необхідний на усунення проблеми, мають надавати саме розробники, ці фактори впливу не будуть описані в рамках експериментальної частини.

3.3 Визначення системи оцінювання ваги багу

Нижче описано систему оцінювання факторів впливу на вартість багу.

Серйозність багу пропонується оцінювати за такою градацією:

– «Blocker»: баг блокує частину функціоналу таким чином, що подальша робота з системою стає неможливою;

– «Critical»: баг порушує бізнес-логіку основного функціоналу системи, стабільно відтворюються та робить неможливим використання основних функцій.

– «Major»: баг ускладнює роботу основного функціоналу, але не порушує її. Або ж баг порушує роботу додаткового функціоналу;

– «Minor»: баг ускладнює роботу додаткового функціоналу системи, але не порушує його, відтворюється не стабільно або має очевидні «обхідні шляхи»;

– «Trivial»: баг не впливає на основний та додатковий функціонал системи, але погіршує загальне враження користувачів від роботи з системою.

Пріоритет багу пропонується оцінювати за такою градацією:

– «Top»: якщо баг досить негативно впливає на продукт або бізнес компанії-замовника;

– «High»: якщо баг має бути усунутий в першу чергу;

– «Normal»: якщо баг має бути усунутий у другу чергу (в робочому порядку);

– «Low»: якщо баг має бути усунутий в останню чергу за наявності ресурсів та часу.

Потенційні ризики пропонується оцінювати за такою градацією:

– «High»: високий рівень ризику;

– «Medium»: ризики можуть бути, але незначні;

– «Low»: низький рівень ризику.

Після того, як було визначено план експерименту та введено системи оцінювання ваги та вартості багів, можна переходити, власне, до самого експерименту.

4 ОПИС ПРОГРАМНОЇ СИСТЕМИ НА ОСНОВІ БЛОКЧЕЙН ТЕХНОЛОГІЇ

Існує невтішна тенденція, що системи, в основі яких лежить технологія блокчейн, та які не пов'язані напряму з платіжними транзакціями, мають не достатній рівень покриття тестуванням. Саме тому для експериментальної частини дослідження було обрано систему для голосування.

На високому рівні проста система голосування складається з організатора, машини для голосування та голосу (див. рис. 9).

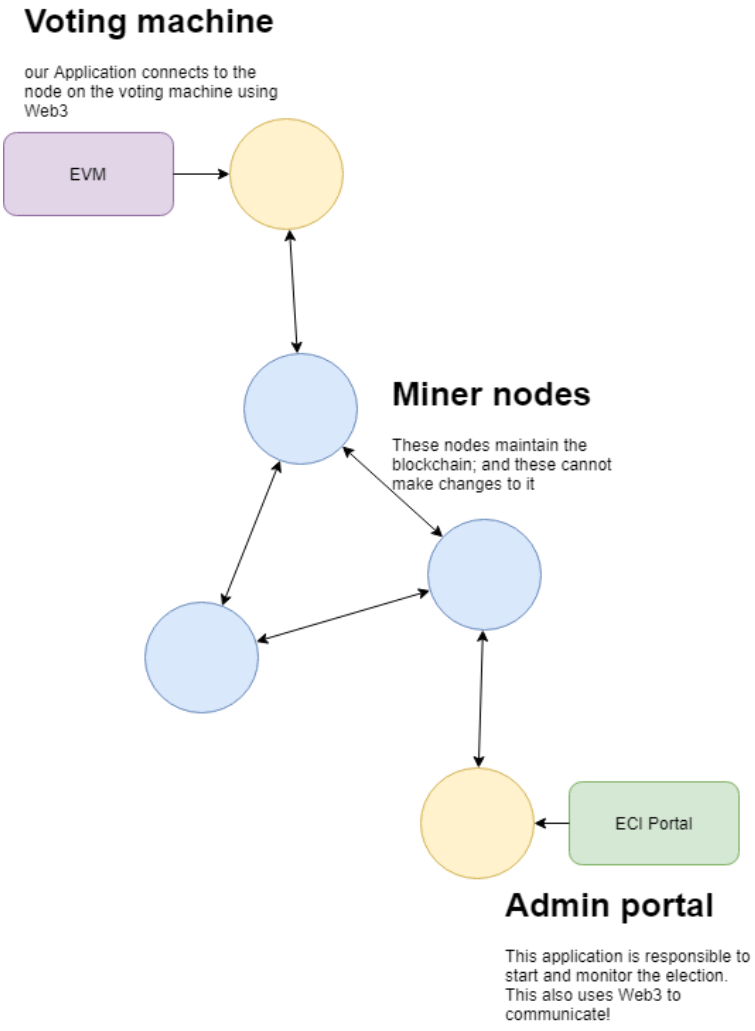

Рисунок 9 – Структура системи

Зрештою, дана система зводиться до децентралізованого веб-додатку (див. рис. 10), компонентами якого є:

– смарт-контракти;

– фронт-енд частина для виборчої комісії (ECI) та для машини для голосування;

– служба аутентифікації виборців.

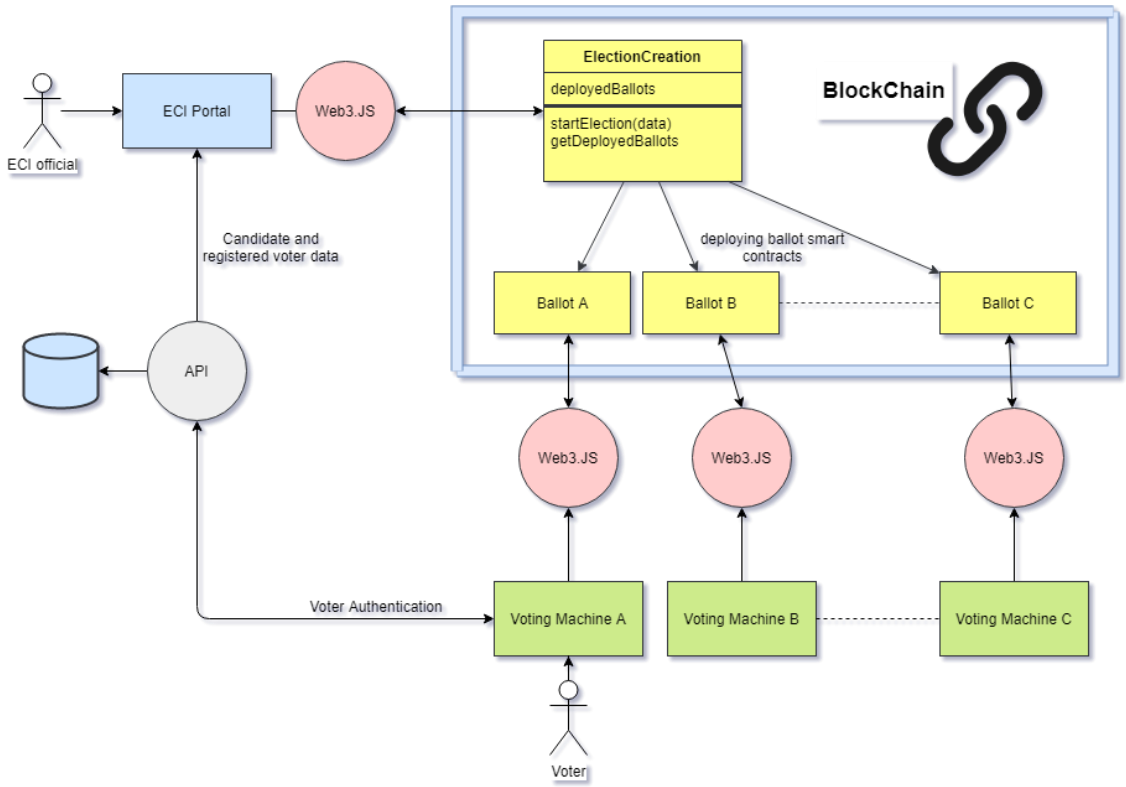

Рисунок 10 – Архітектура системи

Система включає два смарт-контракти:

– смарт-контракт «Ballot»;

– смарт-контракт про створення виборчого бюлетеня.

Смарт-контракт Ballot – це елемент системи, завдяки якому реєструються голоси (див. рис. 11). Кожен виборчий округ матиме екземпляр Ballot, розгорнутий на блокчейні. В свою чергу він складається з переліку даних про кандидатів для відповідного округу. Бюлетень складається з функції голосування, яка може бути викликана клієнтом для збільшення кількості голосів за кандидата. Оскільки дана

функція змінює стан контракту, вона є платною і тому записується в блокчейн. Кожен голос має часовий параметр, який визначає тривалість дійсності голосування. Доступ до цього контракту матиме виборча машина для голосування та портал ECI для моніторингу та підрахунку голосів.

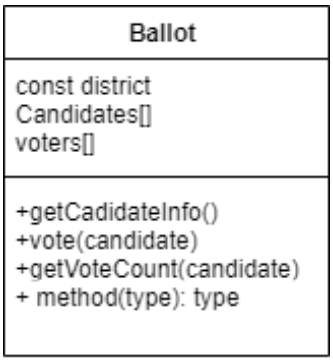

Рисунок 11 – Смарт-контракт «Ballot»

Смарт-контракт про створення виборчого бюлетеня – це елемент системи, який використовується для розміщення виборчих бюлетенів на блокчейні (див. рис. 12). Підключитися до цього контракту може лише представник КВУ, який керує клієнтом порталу КВУ. Цей контракт приймає на вхід базу даних усіх кандидатів відповідних округів і розміщує бюлетені для кожного округу на блокчейні. Даний контракт також має функцію, яка повертає адреси розміщених бюлетенів, щоб вони могли бути використані машинами для голосування пізніше.

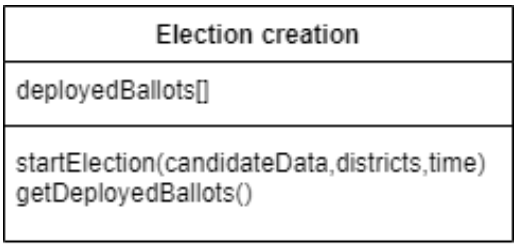

Рисунок 12 – Смарт-контракт «Election creation»

Дана система має два веб-клієнти, а саме:

– портал ECI;
– машина для голосування.

Обидва клієнти потребують зв'язку за протоколами http та rpc, тому використовують api/бібліотеки, такі як XMLHttpRequest та web3.js.

ECI-портал виконує кілька функцій. В першу чергу він повинен реєструвати виборців, які мають право голосу. Для цього за допомогою даного функціоналу створюється таблиця, що складається з необхідних даних про виборців. Ці дані включають хеш ідентифікатора та відбитків пальців виборця, а також його виборчий округ. Для кожного зареєстрованого виборця ці дані збираються та зберігаються в базі даних.

Для створення бази даних кандидатів відповідних округів додається ще одна функція. Це дозволить створити таблицю даних про кандидатів, яка буде використана для завантаження бюлетенів пізніше.

Також ECI-портал дозволяє посадовій особі ОВК розпочати вибори, визначивши кандидатів та час проведення виборів. Це дозволить використовувати інтелектуальну систему створення виборів, яка буде розгортати виборчі бюлетені.

Адреси розгорнутих бюлетенів зберігаються в окремій таблиці для подальшого використання. Портал використовує адресу, що зберігається в таблиці, для доступу до бюлетенів з метою моніторингу голосування, таким чином, оголошуючи переможця виборів.

Інтерфейс машини для голосування спочатку перевіряє право голосу виборця. Для цього машина вводить ідентифікатор виборця та підпис з відбитками пальців виборця. Ці дані хешуються і порівнюються з базою даних виборців для аутентифікації. Виборці, які мають право голосу, отримують доступ до бюлетеня, де вони можуть віддати свій голос. Інтерфейс простий і містить інструкції кількома мовами.

Взаємодія з бюлетенем здійснюється за допомогою протоколу web3 ipc, який підключається до вузла на комп'ютері. Веб-додаток взаємодіє зі смарт-контрактом бюлетеня за допомогою web3js і таким чином надсилає голос у вигляді транзакції.

Також слід сказати про сервіс автентифікації виборців. Вся взаємодія з базою даних побудована на REST API, створеному за допомогою додатку NodeJS. API взаємодіє з клієнтами та сервером mysql, розміщеним в Інтернеті.

База даних містить 3 таблиці:

– таблиця, що зберігає дані про кандидатів;

– таблиця, що зберігає дані про виборців;

– таблиця, що зберігає адреси контрактів.

Блокчейн реалізований за допомогою програми Geth. Geth – це реалізація протоколу Ethereum на основі Go. В даній системі було створено приватну p2p-мережу з усіх машин для голосування, майнерів та ECI-порталу.

Рисунок 13 – Графічний інтерфейс сторінки «Add Candidate»

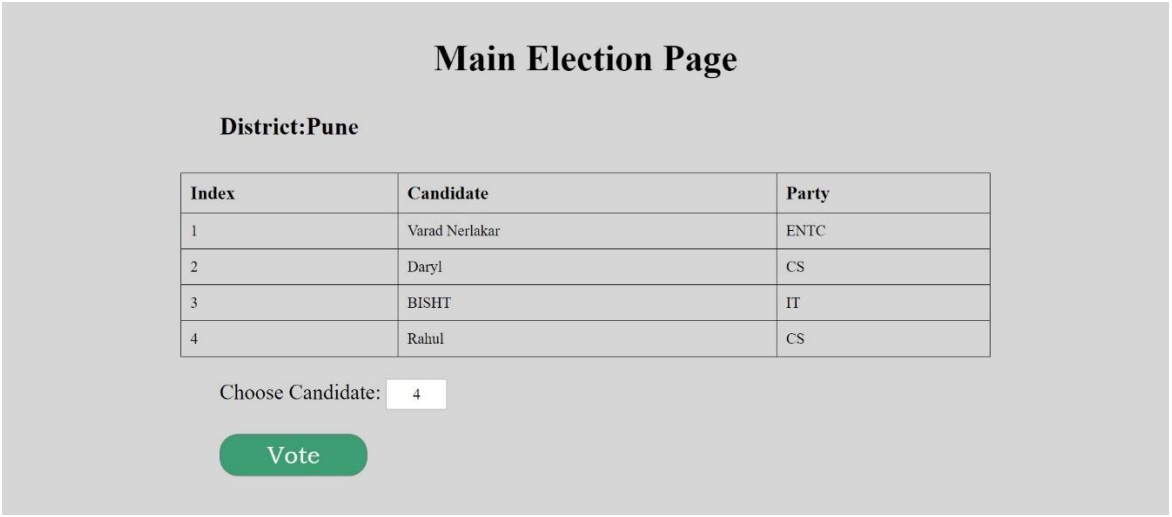

Рисунок 14 – Графічний інтерфейс сторінки «Main Election»

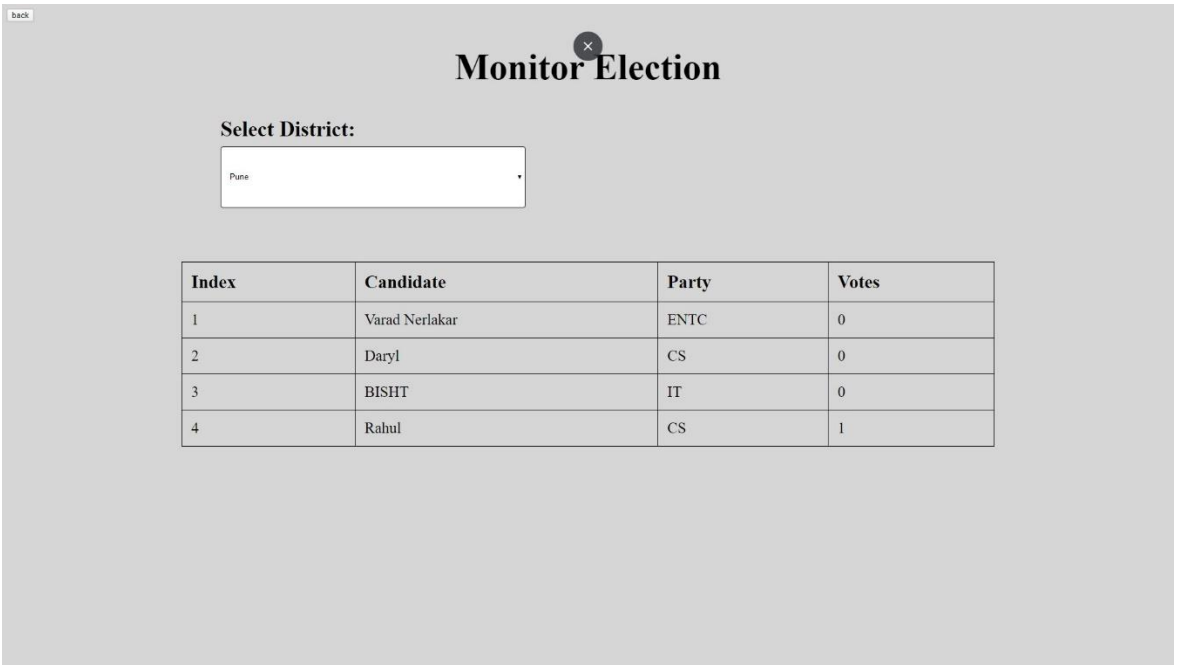

Рисунок 15 – Графічний інтерфейс сторінки «Monitor Election»

Система має досить простий та зрозумілий інтерфейс (див. рис. 13 – 15). Проте інтерфейс системи не буде розглядатись в рамках даного дослідження, адже головний акцент дослідження спрямований саме на функціонал системи.

5 ПРОЦЕС ТЕСТУВАННЯ СИСТЕМИ НА ОСНОВІ БЛОКЧЕЙН ТЕХНОЛОГІЇ

В попередніх розділах було проаналізовано методи тестування систем на основі блокчейн технології, фактори, що впливають на вартість та вагу багів, описано та обрано систему, в основі якої лежить технологія блокчейн. Таким чином було отримано всю необхідні дані, що дозволяють перейти до наступного етапу дослідження, а саме до проектування процесу тестування.

Даний процес описано за допомогою деталізованого тест-плану (див. дод. Б). Під час його написання особливу увагу потрібно приділити функціональному тестуванню та тестуванню смарт-контрактів на рівні юніт-тестування та інтеграційного тестування, а також тестуванню безпеки та тестуванню на проникність – на рівні системного та приймального тестування.

В наступних підрозділах наведено детальний опис багів, які було знайдено на кожному з етапів тестування системи.

5.1 Юніт-тестування

В рамках даного етапу тестування було виконано наступні перевірки:

– перевірки розмірів блоків в рамках функціонального тестування;

– перевірки можливості додавання блоків в рамках функціонального тестування;

– перевірки роботи смарт-контрактів.

В результаті вищевказаних перевірок було знайдено наступні баги:

– баг-1: максимальний розмір блоку менше 1 МБ. Проблема була знайдена під час перевірки розміру блоку в рамках функціонального тестування;

– баг-2: блок додається некоректно. Проблема була виявлена під час перевірки можливості додавання блоків в рамках функціонального тестування;

– баг-3: дані виборця завантажуються поза ланцюжком. Проблема була знайдена під час перевірки роботи смарт-контрактів.

У таблиці 1 наведено оцінку кожного з факторів впливу на вартість вищеописаних багів.

Таблиця 1 – Оцінка факторів впливу на вартість знайдених багів під час юніт-тестування

Фактори впливу	Баг-1	Баг-2	Баг-3
Складність перевірки фіксу	Low	Medium	Medium
Час, необхідний на перевірку фіксу (в годинах)	0.25	0.5	0.5
Необхідність додаткових релізів	No	No	No
Етап тестування	Unit	Unit	Unit

Баг-1 не є складним в перевірці, тому ним може зайнятись спеціаліст з мінімальним досвідом, що значно зменшить вартість самого багу. Також на його перевірку необхідна досить мала кількість часу.

До перевірки фіксів по багу-2 та багу-3 має бути залучений спеціаліст з середнім рівнем кваліфікації, який має достатній рівень досвіду для передбачення всіх кейсів, які слід перевірити, щоб переконатись, що фікс не породив нові проблеми. За цієї умови буде використана мала кількість часу, необхідного для перевірки.

На даному етапі тестування немає необхідності в додаткових релізах, що значно зменшує вартість багів. Адже залучення інших спеціалістів, таких як реліз менеджер, та витрати часу на реліз не потрібні.

У таблиці 2 наведено оцінку кожного з факторів впливу на вагу вищеописаних багів.

Таблиця 2 – Оцінка факторів вагу на вартість знайдених багів під час юніт-тестування

Фактори впливу	Баг-1	Баг-2	Баг-3
Серйозність	Minor	Blocker	Blocker
Пріоритет	Normal	Top	Top

Кінець таблиці 2

Фактори впливу	Баг-1	Баг-2	Баг-3
Рівень потенційних ризиків після фіксу	Low	Medium	Medium
Рівень потенційних ризиків, якщо проблему не буде усунуто	Low	High	High

Баг-1 не порушує роботу системи за звичайних умов використання, проте може ускладнити її, якщо об’єм даних користувача буде максимальним. Саме тому він має бути виправлений в робочому порядку. Рівень потенційних ризиків після усунення проблеми є досить низьким. Рівень потенційних ризиків, якщо проблему не буде виправлено також є мінімальним, адже баг-1 не порушує роботу системи за звичайних умов використання і вірогідність того, що користувач завантажуватиме максимальний об’єм даних є досить низькою.

Баг-2 та баг-3 блокують частину функціоналу, адже додавання блоків є вкрай важливою та базовою частиною функціоналу, так як весь ланцюжок будуватиметься на основі вже доданих блоків та змінити їх буде неможливо. Тому вони мають бути усунені щонайшвидше. Якщо ж говорити про рівні ризиків, то потенційні ризики після фіксу можуть бути, проте рівень ризиків, якщо проблему не буде виправлено, значно більший, адже проблеми стосуються роботи системи в цілому.

5.2 Інтеграційне тестування

В рамках даного етапу тестування було виконано наступні перевірки:

– перевірка розміру ланцюжка в рамках функціонального тестування;

– перевірка можливості передачі даних в рамках функціонального тестування;

– перевірка смарт-контрактів.

В результаті вищевказаних перевірок було знайдено наступні баги:

– баг-1: розмір ланцюжка менше заданого. Неможливо додати новий блок. Ця проблема була знайдена під час перевірки розміру ланцюжка в рамках функціонального тестування;

– баг-2: частина даних втрачається під час передачі. Ця проблема була виявлена під час перевірки можливості передачі даних в рамках функціонального тестування;

– баг-3: дані виборця завантажуються поза ланцюжком. Ця проблема була виявлена під час перевірки смарт-контрактів.

У таблиці 3 наведено оцінку кожного з факторів впливу на вартість вищеописаних багів.

Таблиця 3 – Оцінка факторів впливу на вартість знайдених багів під час інтеграційного тестування

Фактори впливу	Баг-1	Баг-2	Баг-3
Складність перевірки фіксу	Medium	Medium	Medium
Час, необхідний на перевірку фіксу (в годинах)	0.5	1	0.5
Необхідність додаткових релізів	No	No	No
Етап тестування	Integration	Integration	Integration

До перевірки фіксів по багу-1, багу-2 та багу-3 має бути залучений спеціаліст з середнім рівнем кваліфікації, який має достатній рівень досвіду для передбачення всіх кейсів, які слід перевірити, щоб переконатись, що фікс не породив нові проблеми. За цієї умови буде використана відносно мала кількість часу, необхідного для перевірки.

На даному етапі тестування немає необхідності в додаткових релізах, що значно зменшує вартість багів. Адже залучення інших спеціалістів, таких як реліз менеджер, та витрати часу на реліз не потрібні.

У таблиці 4 наведено оцінку кожного з факторів впливу на вагу вищеописаних багів.

Таблиця 4 – Оцінка факторів вагу на вартість знайдених багів під час інтеграційного тестування

Фактори впливу	Баг-1	Баг-2	Баг-3
Серйозність	Critical	Critical	Blocker
Пріоритет	Top	High	Top
Рівень потенційних ризиків після фіксу	Medium	Medium	Medium
Рівень потенційних ризиків, якщо проблему не буде усунуто	High	High	High

Баг-1 порушує роботу основного функціоналу. Саме тому він має бути виправлений якнайшвидше. Потенційні ризики після фіксу можуть бути, проте рівень ризиків, якщо проблему не буде виправлено, значно більший, адже проблеми стосуються роботи системи в цілому.

Баг-2 порушує роботу основного функціоналу, адже втрата даних користувачів є недопустимою під час роботи системи. Саме тому він має бути виправлений в першу чергу. Потенційні ризики після фіксу можуть бути, проте рівень ризиків, якщо проблему не буде виправлено, значно більший, адже проблеми стосуються безпеки даних користувача.

Баг-3 блокує роботу основного функціоналу. Саме тому він має бути виправлений якнайшвидше. Потенційні ризики після фіксу можуть бути, проте рівень ризиків, якщо проблему не буде виправлено, значно більший, адже проблеми стосуються роботи системи в цілому.

5.3 Системне тестування

В рамках даного етапу тестування було виконано наступні перевірки:

– перевірка можливості передачі даних в рамках функціонального тестування;

– перевірка смарт-контрактів;

– перевірка безпеки системи в рамках тестування безпеки;

– перевірка безпеки системи в рамках тестування на проникнення.

В результаті вищевказаних перевірок було знайдено наступні баги:

– баг-1: частина даних втрачається під час передачі. Ця проблема була знайдена під час перевірки можливості передачі даних в рамках функціонального тестування;

– баг-2: дані виборця завантажуються поза ланцюжком. Ця проблема була виявлена під час перевірки смарт-контрактів;

– баг-3: існує можливість розшифровки даних. Ця проблема була виявлена під час перевірки безпеки системи в рамках тестування безпеки;

– баг-4: неавторизований користувач в системі не блокується. Ця проблема була виявлена під час перевірки безпеки системи в рамках тестування на проникнення.

У таблиці 5 наведено оцінку кожного з факторів впливу на вартість вищеописаних багів.

Таблиця 5 – Оцінка факторів впливу на вартість знайдених багів під час інтеграційного тестування

Фактори впливу	Баг-1	Баг-2	Баг-3	Баг-4
Складність перевірки фіксу	Medium	Medium	High	High
Час, необхідний на перевірку фіксу (в годинах)	1	0.5	2	2
Необхідність додаткових релізів	Stg	Stg	Stg	Stg
Етап тестування	System	System	System	System

До перевірки фіксів по багу-1 та багу-2 має бути залучений спеціаліст з середнім рівнем кваліфікації, який має достатній рівень досвіду для передбачення всіх кейсів, які слід перевірити, щоб переконатись, що фікс не породив нові проблеми. За цієї умови буде використана відносно мала кількість часу, необхідного для перевірки.

До перевірки фіксів по багу-3 та багу-4 має бути залучений спеціаліст з високим рівнем кваліфікації, який має багатий досвід для передбачення всіх кейсів, які слід перевірити, щоб переконатись, що фікс не породив нові проблеми. Проте навіть за цієї умови буде використана досить велика кількість часу, необхідного для перевірки.

На даному етапі тестування є необхідність в додаткових релізах стейджингу. Для цього мають бути залучені інші спеціалісти за необхідністю. Також слід зазначити, що будь-який реліз потребує певного часу на його проведення.

У таблиці 6 наведено оцінку кожного з факторів впливу на вагу вищеописаних багів.

Таблиця 6 – Оцінка факторів вагу на вартість знайдених багів під час інтеграційного тестування

Фактори впливу	Баг-1	Баг-2	Баг-3	Баг-4
Серйозність	Critical	Blocker	Critical	Critical
Пріоритет	High	Top	High	High
Рівень потенційних ризиків після фіксу	Medium	Medium	High	Medium
Рівень потенційних ризиків, якщо проблему не буде усунуто	High	High	High	High

Баг-1 порушує роботу основного функціоналу, адже втрата даних користувачів є недопустимою під час роботи системи. Саме тому він має бути виправлений в першу чергу. Потенційні ризики після фіксу можуть бути, проте

рівень ризиків, якщо проблему не буде виправлено, значно більший, адже проблеми стосуються безпеки даних користувача.

Баг-2 блокує роботу основного функціоналу. Саме тому він має бути виправлений якнайшвидше. Потенційні ризики після фіксу можуть бути, проте рівень ризиків, якщо проблему не буде виправлено, значно більший, адже проблеми стосуються роботи системи в цілому.

Баг-3 порушує роботу основного функціоналу, адже конфіденційність даних користувачів є обов'язковою під час роботи системи. Саме тому він має бути виправлений в першу чергу. Рівень, як потенційних ризиків після фіксу так і потенційних ризиків, якщо проблему не буде виправлено, досить висока, адже проблема стосуються конфіденційності даних.

Баг-4 порушує роботу основного функціоналу, адже безпека системи вкрай важлива. Саме тому він має бути виправлений в першу чергу. Потенційні ризики після фіксу можуть бути, проте рівень ризиків, якщо проблему не буде виправлено, значно більший.

5.4 Приймальне тестування

В рамках даного етапу тестування було виконано наступні перевірки:

– перевірка можливості передачі даних в рамках функціонального тестування;

– перевірка смарт-контрактів;

– перевірка безпеки системи в рамках тестування безпеки;

– перевірка безпеки системи в рамках тестування на проникнення.

В результаті вищевказаних перевірок було знайдено наступні баги:

– баг-1: частина даних втрачається під час передачі. Ця проблема була знайдена під час перевірки можливості передачі даних в рамках функціонального тестування;

– баг-2: дані виборця завантажуються поза ланцюжком. Ця проблема була виявлена під час перевірки смарт-контрактів;

– баг-3: існує можливість розшифровки даних. Ця проблема була виявлена під час перевірки безпеки системи в рамках тестування безпеки;

– баг-4: неавторизований користувач в системі не блокується. Ця проблема була виявлена під час перевірки безпеки системи в рамках тестування на проникнення.

У таблиці 7 наведено оцінку кожного з факторів впливу на вартість вищеописаних багів.

Таблиця 7 – Оцінка факторів впливу на вартість знайдених багів під час інтеграційного тестування

Фактори впливу	Баг-1	Баг-2	Баг-3	Баг-4
Складність перевірки фіксу	Medium	Medium	High	High
Час, необхідний на перевірку фіксу (в годинах)	1,5	1	2,5	2,5
Необхідність додаткових релізів	Stg, Prod	Stg, Prod	Stg, Prod	Stg, Prod
Етап тестування	UAT	UAT	UAT	UAT

До перевірки фіксів по багу-1 та багу-2 має бути залучений спеціаліст з середнім рівнем кваліфікації, який має достатній рівень досвіду для передбачення всіх кейсів, які слід перевірити, щоб переконатись, що фікс не породив нові проблеми. За цієї умови буде використана відносно мала кількість часу, необхідного для перевірки на стейджингу, але також є необхідність перевірки проблему на продакшені.

До перевірки фіксів по багу-3 та багу-4 має бути залучений спеціаліст з високим рівнем кваліфікації, який має багатий досвід для передбачення всіх кейсів, які слід перевірити, щоб переконатись, що фікс не породив нові проблеми. Проте навіть за цієї умови буде використана досить велика кількість часу, необхідного

для перевірки на стейджингу, але також є необхідність перевірки проблему на продакшені.

На даному етапі тестування є необхідність в додаткових релізах стейджингу, а потім продакшену. Для цього мають бути залучені інші спеціалісти за необхідністю. До релізу продакшену має бути залучена вся QA-команда для проведення смоук-тестування, щоб переконатись, що релізна версія системи є стабільною. Також слід зазначити, що будь-який реліз потребує певного часу на його проведення.

У таблиці 8 наведено оцінку кожного з факторів впливу на вагу вищеописаних багів.

Таблиця 8 – Оцінка факторів вагу на вартість знайдених багів під час інтеграційного тестування

Фактори впливу	Баг-1	Баг-2	Баг-3	Баг-4
Серйозність	Critical	Blocker	Critical	Critical
Пріоритет	High	Top	High	High
Рівень потенційних ризиків після фіксу	Medium	Medium	High	Medium
Рівень потенційних ризиків, якщо проблему не буде усунуто	High	High	High	High

Баг-1 порушує роботу основного функціоналу, адже втрата даних користувачів є недопустимою під час роботи системи. Саме тому він має бути виправлений в першу чергу. Потенційні ризики після фіксу можуть бути, проте рівень ризиків, якщо проблему не буде виправлено, значно більший, адже проблеми стосуються безпеки даних користувача.

Баг-2 блокує роботу основного функціоналу. Саме тому він має бути виправлений якнайшвидше. Потенційні ризики після фіксу можуть бути, проте

рівень ризиків, якщо проблему не буде виправлено, значно більший, адже проблеми стосуються роботи системи в цілому.

Баг-3 порушує роботу основного функціоналу, адже конфіденційність даних користувачів є обов'язковою під час роботи системи. Саме тому він має бути виправлений в першу чергу. Рівень, як потенційних ризиків після фіксу так і потенційних ризиків, якщо проблему не буде виправлено, досить висока, адже проблема стосуються конфіденційності даних.

Баг-4 порушує роботу основного функціоналу, адже безпека системи вкрай важлива. Саме тому він має бути виправлений в першу чергу. Потенційні ризики після фіксу можуть бути, проте рівень ризиків, якщо проблему не буде виправлено, значно більший.

Під час тестування системи також було використано один з інструментів, а саме Truffle.

Вибір інструмента був обумовлений такими факторами, як:

– доступність;
– сумісність з мовою програмування Solidity;
– зручність використання.

Для того, щоб скористатись даним інструментом в IntelliJ IDEA, необхідно просто завантажити плагін «Truffle» (див. рис. 16).

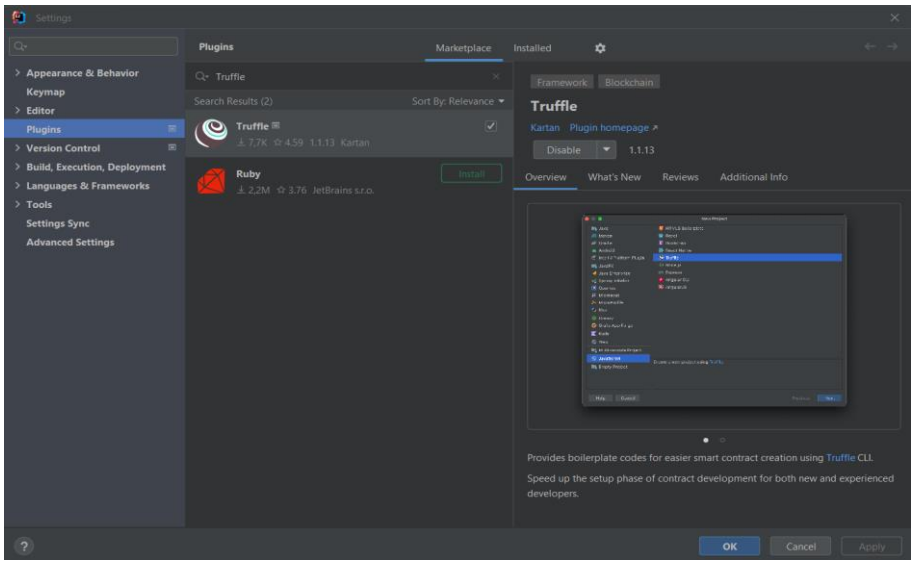

Рисунок 16 – Встановлення інструменту «Truffle» в IntelliJ IDEA

Наступним кроком необхідно створити папку «test» в проекті та додати в неї, безпосередньо, файл, в якому будуть розроблені авто-тести. Після цього можна перейти до написання тестів.

Нижче наведено програмну реалізацію одного з тестів.

```
import "truffle/Assert.sol";
import "../contracts/election.sol";

contract TestElectioncreation {

    Electioncreation election;

    function beforeEach() public {
        election = new Electioncreation();
    }

    function testStartelec() public {
        BallotManager ballotManager = new BallotManager();
        string[][] memory candidates = new string [][] (2);
        candidates[0] = new string[](2);
        candidates[0][0] = "Alice";
        candidates[0][1] = "Bob";
        candidates[1] = new string[](2);
        candidates[1][0] = "Charlie";
        candidates[1][1] = "Dave";

        string[][] memory party = new string[][](2);
        party[0] = new string[](2);
        party[0][0] = "Party1";
        party[0][1] = "Party2";
        party[1] = new string[](2);
        party[1][0] = "Party3";
        party[1][1] = "Party4";

        string[] memory district = new string[](2);
        district[0] = "District1";
        district[1] = "District2";

        uint hour = 8;

        ballotManager.startelec(candidates, party, district, hour);

        address[] memory deployedBallots =
ballotManager.getsDeployedBallots();
        uint expectedLength = 2;
        Assert.equal(deployedBallots.length, expectedLength, "Unexpected
number of deployed ballots");

        for (uint i = 0; i < deployedBallots.length; i++) {
            Ballot ballot = Ballot(deployedBallots[i]);
            string[] memory expectedCandidates = candidates[i];
            Assert.equal(ballot.getCandidates(), expectedCandidates,
"Unexpected candidates in ballot");
```

```
            string[] memory expectedParty = party[i];
            Assert.equal(ballot.getParty(),    expectedParty,    "Unexpected
party in ballot");
            string memory expectedDistrict = district[i];
            Assert.equal(ballot.getDistrict(),            expectedDistrict,
"Unexpected district in ballot");
            address expectedElectionCommissioner = msg.sender;
            Assert.equal(ballot.getElectionCommissioner(),
expectedElectionCommissioner, "Unexpected election commissioner in ballot");
            uint expectedHour = hour;
            Assert.equal(ballot.getVotingEndHour(),           expectedHour,
"Unexpected voting end hour in ballot");
        }
    }
```

Цей тест створює новий екземпляр «BallotManager» і викликає функцію startelec() з деякими вибірковими даними.

Після чого викликається функція getsDeployedBallots() для отримання адрес розгорнутих бюлетенів і перевіряє, чи розгорнуто очікувану кількість бюлетенів.

Наступним кроком перебираються всі розгорнуті бюлетені та перевіряються, чи відповідають їх властивості очікуваним значенням на основі вхідних даних, отриманих у startelec().

```
PS D:\Voting_system\Blockchain_Based_Voting_system-master> truffle test
Using network 'test'.


Compiling your contracts...
===========================
```

Рисунок 17 – Запуск авто-тесту за допомогою інструмента «Truffle»

Щоб запустити виконання тесту, необхідно використати команду «truffle test» (див. рис. 17).

```
  1 passing (6s)
  0 failing
```

Рисунок 18 – Запуск авто-тесту за допомогою інструмента «Truffle»

На рисунку 18 можна побачити, що тест був пройдено успішно.

6 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Проаналізувавши результати експерименту, описаного в попередньому розділі, можна зробити наступні висновки.

Коректне тестування системи на етапі юніт-тестування може суттєво зменшити кількість та серйозність ризиків, в тому числі і для безпеки даних користувачів. Якщо критична помилка не буде знайдена на цьому етапі, то вартість помилки буде зростати в залежності від того, на якому з наступних етапів тестування вона буде знайдена.

Варто також зазначити, що критичними з точки зору потенційних ризиків є помилки в блоках або ланцюжках блоків. Адже згідно з технологією блокчейн, блоки даних не можуть бути змінені, а ланцюжки блоків будуються на основі попередніх блоків. Тому одна помилка може призвести до значних проблем для всієї системи, особливо коли йдеться про безпеку даних.

Що стосується інтеграційного тестування, то проблеми із взаємодією компонентів системи також мають значний вплив на безпеку та надійність системи. Саме тому інтеграційне тестування відіграє значну роль у процесі тестування. Ціна помилки на цьому етапі також буде набагато нижчою, ніж на одному з наступних етапів. Однак потенційні ризики після виправлення помилки набагато нижчі, ніж потенційні ризики, якби помилка не була знайдена на цьому етапі.

Після інтеграційного тестування було проведено тестування системи. Проаналізувавши результати тестування на цьому етапі, можна зробити висновок, що ціна помилки значно зростає.

Що стосується тестування безпеки та тестування на проникнення, то їх слід проводити під час тестування системи, щоб переконатися, що система працює стабільно та відповідає заданим вимогам безпеки даних.

Одним з останніх етапів є приймальне тестування. На цьому етапі вартість і вага помилки є найвищою. Адже найбільші ризики, як після виправлення, так і якщо баг, який безпосередньо впливає на безпеку та надійність системи, не буде знайдений під час приймального тестування.

Підсумовуючи результати на кожному етапі експерименту, можна зробити висновок, що особливу увагу варто приділити модульному та інтеграційному тестуванню. Адже саме з цього моменту вартість помилки починає зростати в геометричній прогресії. Втім, системне та приймальне тестування також відіграють значну роль у безпеці системи на основі блокчейну.

Необхідно виконувати функціональне тестування на кожному етапі тестування. Особливу увагу слід приділити тестуванню смарт-контрактів. Кількість і серйозність ризиків можна значно зменшити, правильно оцінивши систему на етапі модульного та інтеграційного тестування. Тестування на безпеку та проникнення також повинно бути виконано в рамках тестування системи.

Так як більшість допоміжних інструментів, що були розглянуто в теоретичній частині, платні, було прийнято рішення використати найдоступніший з них. Через недостатність фінансових можливостей, порівняння інструментів не було проведено в даному дослідженні. Проте вони можуть бути розглянуті в подальшому.

Результати дослідження були застосовані на етапі планування тестування, а саме під час розробки плану тестування.

ВИСНОВКИ

Тестування та забезпечення якості це дуже важливий етап для будь-якої системи. Тому що наслідки будь-якого інциденту в режимі реального часу можуть бути серйозними, а також можуть призвести до значних фінансових втрат або втрат даних, в залежності від галузі та застосування. Ця точка зору зосереджена на функціональних аспектах тестування блокчейн-систем та їх ключових областях, які необхідно тестувати, включаючи рекомендації щодо використання інструментів для забезпечення ефективності виконання тестів.

Визначення правильного підходу до тестування систем з використанням блокчейн технології є важливим кроком для ефективного та успішного тестування.

В ході виконання кваліфікаційної роботи була досліджена ефективність процесу тестування блокчейн-систем в залежності від різних методів тестування.

Для цього було:

– розглянуто підхід до планування процесу тестування;

– розглянуто базові методи тестування, а також методи тестування, які є специфічними для блокчейн-систем;

– розглянуто допоміжні інструменти для тестування систем на основі блокчейн технології;

– проаналізовано фактори впливу на вартість та критичність багів в системах на основі блокчейн технології;

– визначено метрики оцінювання вартості та ваги багу, які були використані під час проведення експерименту;

– проаналізовано структуру блокчейн-системи, яка була протестована під час проведення експерименту;

– спроектовано процес тестування обраної блокчейн-системи за допомогою тест-плану;

– протестовано обрану блокчейн-систему за розробленим тест-планом;

– оцінено вартість та вагу багів за допомогою визначених метрик;

– проаналізовано отримані результати дослідження.

Мета завдання досягнута за рахунок використання отриманих результатів дослідження під час написання деталізованого тест-плану.

В даній роботі було удосконалено підхід до процесу тестування, а саме написання тест-плану, з урахуванням вибору методів тестування для забезпечення фактичної якості систем, в основі яких лежить технологія блокчейн.

Для підтвердження отриманих результатів і зроблених висновків або для їх спростування потрібні подальші дослідження. Для подальшого дослідження слід розглянути більш складні підходи до планування процесу тестування, та комплексні метрики визначення ефективності та загальної вартості даного процесу.

Кваліфікаційна робота пройшла апробацію на науковій конференції «7th International Conference on Computational Linguistics and Intelligent Systems» CoLinS–2023 (Scopus) April 20–21, 2023 (матеріали статті наведені у додатку В).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kyrychenko, I., Shyshlo O., Shanidze, N. Minimizing Security Risks and Improving System Reliability in Blockchain Applications: A Testing Method Analysis. 7th International Conference on Computational Linguistics and Intelligent Systems (Scopus), April 20–21, 2023, Kharkiv, Ukraine.

2. Sharonova, N., Kyrychenko, I., Tereshchenko, G. Application of big data methods in E-learning systems. 5th International Conference on Computational Linguistics and Intelligent Systems (Scopus), April 22–23, 2021, Kharkiv, Ukraine.

3. ADAM HAYES. Blockchain Facts: What Is It, How It Works, and How It Can Be Used [Електронний ресурс] / ADAM HAYES. – 2022. – Режим доступу до ресурсу: https://www.investopedia.com/terms/b/blockchain.asp

4. Privacy-Protected Blockchain System / Junho Jeong, Qikai Zhong, Haibo Mi та ін.]. // 20th IEEE International Conference on Mobile Data Management (MDM). – 2019

5. A Study on Prevention and Automatic Recovery of Blockchain Networks Against Persist ent Censorship Attacks / Jae-Seok Kim, Jin-Myeong Shin, Seok-Hwan Choi, Yoon-Ho Choi. // IEEE Access. – 2022. – С. 110770 – 110784

6. Veera Budhi. Advantages And Disadvantages Of Blockchain Technology [Електронний ресурс] / Veera Budhi // Forbes. – 2022. – Режим доступу до ресурсу: https://www.forbes.com/sites/forbestechcouncil/2022/10/20/advantages-and-disadvantages-of-blockchain-technology/

7. A High-Speed Data Retrieval Model on Blockchain / Jingang Yu, Yongkang Hou, Shu Li, Zhifeng Wen. // 11th International Conference of Information and Communication Technology (ICTech)). – 2022

8. Shi Yan. Analysis on Blockchain Consensus Mechanism Based on Proof of Work and Proof of Stake / Shi Yan. // 2022 International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI). – 2022

9. Gruzdo I., Kyrychenko I., Tereshchenko G., Shanidze N. Metrics Applicable for Evaluating Software at The Design Stage. 5th International Conference on

Computational Linguistics and Intelligent Systems (Scopus), April 22–23, 2021, Kharkiv, Ukraine.

10. Lina Wang. Application of Blockchain in Life Cycle Cost Management of Weapon Equipment / Lina Wang, Yanhong Zhang, Ying Ren. // 2020 2nd International Conference on Applied Machine Learning (ICAML). – 2020;

11. Barbara Thompson. Blockchain Testing Tutorial [Електронний ресурс] / Barbara Thompson. – 2022. – Режим доступу до ресурсу: https://www.guru99.com/blockchain-testing.html

12. Software Testing News. The need for Blockchain Testing [Електронний ресурс] / Software Testing News – Режим доступу до ресурсу: https://www.softwaretestingnews.co.uk/the-need-for-blockchain-testing/

13. Systematic Literature Review on Penetration Testing for Mobile Cloud Computing Applications / Ahmad Salah Al-Ahmad, Hasan Kahtan, Fadhl Hujainah, Hamid A. Jalab. // IEEE Access. – 2019. – C. 173524 – 173540

14. Nedas Matulevicius. Systematic Literature Review on Penetration Testing for Mobile Cloud Computing Applications / Nedas Matulevicius, Lucas C. Cordeiro. // 2021 XI Brazilian Symposium on Computing Systems Engineering (SBESC). – 2021

15. Pascal Akunne. The complete guide to blockchain testing [Електронний ресурс] / Pascal Akunne. – 2021. – Режим доступу до ресурсу: https://blog.logrocket.com/complete-guide-blockchain-testing/#:~:text=Blockchain%20testing%20is%20the%20systematic,require%20special%20tools%20to%20test;

16. Kent Beck. Extreme Programming Explained: Embrace Change / Kent Beck., 2004. – 39 с. – (2nd ed).