Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Метод та засоби управління децентралізованими фінансами за домогою технології Blockchain»

Виконала: студентка 2 курсу,
групи 2КІ-20м напряму підготовки
(спеціальності)
123 — «Комп'ютерна інженерія»
_____Білоус Г.О.

Керівник: к.т.н.,

_____Богомолов С.В.

«____» _____ 2021 р.

Опонент: к.т.н., доц., зав. каф. МБІС

_____ Карпінець В.В.

«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«____» _____ 2021 р.

Вінниця ВНТУ - 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень магістр
Спеціальність 123 — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
_____ проф., д.т.н. О.Д. Азаров

«___»_____2021 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Білоус Ганні Олександрівні

1 Тема роботи «Метод та засоби управління децентралізованими фінансами за допомогою технології Blockchain»керівник роботи Богомолов Сергій Віталійович, к.т.н.,_затверджені наказом вищого навчального закладу від 24.12.2021 р. №227

2 Строк подання студентом роботи 15.12.2021 р.

3 Вихідні дані до роботи: кінцева кількість токенів, включаючи виганороду за користування 12 000. Швидкість проведення транзакції не більше 1хв. Мінімальна комісія за транзакцію 0.002 bnb.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити). Вступ. Метод управління децентралізованими фінансами за допомогою технології Blockchain. Засоби реалізації технології Блокчейн. Розробка програмного засобу управління децентралізованими фінансами. Розрахунок економічної доцільності створення програмного засобу управління децентралізованими фінансами.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема хешів блоків блокчейну, структурна схема програми.

6 Консультанти розділів роботи представлені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1,2,3	Богомолов С. В., к.т.н.		
4	Лесько О. Й., к. е. н., професор		

7 Дата видачі завдання 01.09.2021 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2 – Календарний план

№	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	01.09.21	
2	Огляд методу та технології blockchain	02.09-07.09.21	
3	Аналіз та вибір засобів для реалізації програмного забезпечення	08.09-11.09.21	
4	Розробка смарт-контракту	11.09-09.10.21	
5	Налаштування взаємодії смарт-контракту із blockhchain Ethereum	15.10-20.10.21	
6	Розробка програми розпізнавання програми децентралізованого управління фінансами	10.10-29.10.21	
7	Розробка інтерфейсу та тестування програми децентралізованого управління фінансами	29.10-08.11.21	
8	Підготовка матеріалів та опис розробки засобів децентралізованого управління фінансами	10.11-14.11.21	
9	Розрахунок економічної частини роботи	16.11-28.11.21	
10	Оформлення пояснювальної записки та ілюстративного матеріалу	28.11-02.12.21	
11	Аналіз виконання роботи, висновки, додатки	02.12-08.12.21	
12	Перевірка якості виконання магістерської роботи та усунення недоліків	14.12.21	

Студентка _____ Білоус Г. О.
Керівник роботи _____ Богомолов С. В.

АНОТАЦІЯ

УДК 004.9

Білоус Г.О. Метод управіння децентралізованими фінансами за допомогою технології Blockchain. Магістерська кваліфікаційна робота зі спеціальності 123 — комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця: ВНТУ, 2021, 113 с.

На укр.мові. Бібліогр.: 52 назв, рис. 24, табл. 11.

Дана магістерська робота присв'ячена децентралізованій системі управління фінасами за допомогою технології blockchain. Поняття «блокчейн» почало активно обговорюватися зі зростанням популярності цифрової валюти. Дану технологію вже використовують не тільки у сфері фінансів, але і для обробки та зберігання персональних даних, ідентифікації, особистості та маркетингу.

Вважається, що ця технологія здатна стати справжнім проривом у сфері фінансів та захищених баз даних. В роботі було проведено аналіз методу децентралізованого управління фінансами та запропоновано нову структуру смарт-контраку, який дозволяє зільшити швидкість проведених транзакцій та отримувати винагороду за коритсування програмою.

В даній роботі були виконані економічні розрахунки, які дозволяють визначити доцільность розробки нового програмного продукту.

Графічна частина складається з 4 плакатів із результатами еспрементальних досліджень.

Ключові слова: блокчейн, цифрова валюта, децентралізовані фінанси, blockchain Etherium, смарт-контракти

ANNOTATION

Bilous H. Decentralized finance management method with Blockchain technology. Master's thesis in specialty 123 — computer engineering, educational program — computer engineering. Vinnytsia: VNTU, 2021, 113 p.

In Ukrainian. Bibliogr .: 52 titles, fig. 24, table. 11.

This master's thesis is devoted to a decentralized financial management system using blockchain technology. The concept of "blockchain" began to be actively discussed with the growing popularity of digital currency. This technology is already used not only in the field of finance, but also for the processing and storage of personal data, identification, identity and marketing. It is believed that this technology can be a real breakthrough in finance and secure databases. The paper analyzes the method of decentralized financial management and proposes a new structure of smart contract, which allows to increase the speed of transactions and receive a reward for using the program.

In this master's thesis economic calculations were performed to determine the feasibility of developing a new software product.

The graphic part consists of 4 posters with the results of experimental research.

Keywords: blockchain, digital currency, decentralized finance, blockchain Etherium, smart contracts

ЗМІСТ

					08-23.МКР.016.00.000 ПЗ		
Змн.	*Лист*	*№ докум.*	*Підпис*	*Дата*			
Розробив		*Білоус Г. О.*			*Метод та засоби управління децентралізованими фінансами за допомогою технології Blockchain*	*Літ.*	*Арк.* 6 / *Аркушів* 108
Керівник		*Богомолов С.В.*					
Рецензент		*Карпінець В.В.*				*ВНТУ, гр. 2КІ-20м*	
Н. Контроль		*Швець С.І.*					
Затверджую		*Азаров О.Д.*					

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ — програмне забезпечення,

BFT — Byzantine Fault Tolerance,

GUI — Graphical User Interface

MVC — Model View Controller,

P2P — Peer-to-Peer network,

PoET — Proof of Elapsed Time consensus protocol,

PoW — Proof of Work consensus protocol,

TEE — Trusted Execution Environ

ВСТУП

Блокчейн можна назвати без перебільшення багатоцільовим проектом. Для розробників програмного забезпечення блокчейн є найцікавішим середовищем з появи мови Java в 1995 року. Для бізнесу блокчейн став потужним каталізатором реінжинірингу бізнес-операцій та зовнішніх зв'язків. Підприємцям блокчейн дозволяє створювати стартапи, не боячись розпочинати нову справу з невеликою кількістю клієнтів.

Блокчейн – це не просто об'єкт, продукт, тенденція або певна можливість. Блокчейн складається з частин, деякі з яких працюють разом, інші – самостійно і незалежно. Завдяки модульності блокчейн має безліч варіантів використання.

Ідея блокчейна проста, але потужна – вона полягає насамперед у новаторському підході. Йдеться тут не лише про те, як створити найкращу мережу, банк чи забезпечення більш якісного обслуговування. Розвиток блокчейна залежить від операцій, які виконують користувачі, тим самим, впливаючи на на його технічні характеристики. Поширення блокчейна відбувається поступово, починаючи з розробників та стартапів. Активне використання почалось із сфер, пов'язані з 1Т-бізнесом, а за ними – компаній, які відкрили для себе величезний потенціал блокчейну.

Щоб дістатися епату загального поширення технології, потрібно збільшими користувачів додатків блокчейна. У довгостроковій перспективі більшість користувачів не знатимуть чи розумітимуть, що у програмному забезпеченні чи сервісі, яким вони користуються, є блокчейн. На даний момент, люди оцінюють можливості програми за її перевагами та зручністю, а не за рахунок того, що вона працює на певній базі даних і заснована на тій чи іншій технології.

Так само як економіка Інтернету, блокчейн має увесь потенціал створити нову економічну систему, в якій транзакції та підпсання ключових договорів будуть здійснюватись без участі посередництва банківських чи нотаріальних установ. Криптотехнологічна економіка заснована на децентралізованій довірі як у політичному плані, так і в плані цифрової

архітектури. Блокчейн може забезпечити відкритий доступ доступ до звісності та зменшити час на укладення договорів чи проведення фінансових транзакій.

Так само, як інтернет став нішею поширення та обміну інформацією – так само блокчейн може стати провідною технологією для передачі цінностей. Але, як і у випадку з будь-якою перспективою, потрібен час, щоб технологія набула загального поширення. Знадобляться мільйони людей, які знаються на технології блокчейну, мільйони бізнес-лідерів і мільйони активних користувачів. Під час виконання даної магістерської роботи було створено не тільки прогаму, яка забезпечує ефективне управління децентралізованими фінасами, але і заохочує користувачів до активного її використання.

Об’єктом дослідження є процес децеталізованого надійного збереження даних в розподілених мережах.

Предметом дослідження є метод і засоби організації децентралізованих транзакцій, зокрема фінансових між користувачами мережі

Мета роботи — вдосконалення методів децентралізованих транзакцій з використанням технології blockchain.

Для досягнення цієї мети потрібно вирішити такі задачі:

— проаналізувати метод та засоби управління фінансами за допомогою технології blockchain;

— виявити та обрати найбільш ефективні засоби для реалізації управління децентралізованими фінансами;

— розробити спосіб винагороди користувачів за здійснення транзакцій та користування програмним продуктом;

— обрати інструментарій для розробки програмного продукту;

— розробити та дослідити ефективність створеного прикладного програмного продукту.

Наукова новизна роботи — вдосконалено методику розподілу винагород в розподіленій системі blockhain Etherium, що дозволило збільшити привабливість застосування використання blockhain сервісу

Практична цінність роботи полягає в створенні децентралізованої системи управління фінансами, що дозволяє заощадити фінанси за рахунок відсутності комісій за проведення транзакцій банківськими установами та спростити громіздкі процеси переказу та зарахування коштів, а також схвалення та підтвердження операцій. Сферою застосування результатів роботи є фінансові системи, маркетинг.

Методи дослідження магістерської роботи: використовувався системний аналіз, який застосовується для дослідження найбільш ефективних способів зберігання фінансів, об'єктно-орієнтовані методи проектування, які використовувані як основна методологія побудови зв’язаних блоків інформації (blockchain); методи теорії обчислювальних систем для побудови математичних моделей, які застосовуються при пошуку найбільш оптимального шляху передачі інформації в блоках.

Апробація результатів роботи здійснена під час доповіді на 50-й науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії ВНТУ.

Матеріали магістерської роботи опубліковані у [3].

1 МЕТОД УПРАВЛІННЯ ДЕЦЕНТРАЛІЗОВАНИМИ ФІНАНСАМИ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ BLOCKCHAIN

Після переходу в онлайн простір зростає ризик що будь-яка стороння людина зможе здійснити транзакцію без відома власника валюти. Впровадження блокчейна збільшує швидкість обміну, зменшує тимчасові витрати, покращує якість, надійність та доступність послуг. Ключова особливість методу полягає у децентралізації системи. Якщо базу даних, розташовану на єдиному сервері, зламати теоретично можна, за умови застосування будь-яких існуючих засобів захисту, з блокчейнами жоден з цих методів не спрацює. Простими словами – немає що зламувати. Залишається лише варіант спроби крадіжки особистих ключів окремих користувачів. Одна з причин впровадження блокчейнів у фінансову сферу – безпека [1] При використанні технології блокчейн для розробки програмного забезпечення, яке слугуватиме для управління децентралізованими фінансами, насамперед загострюють увагу на тому, що у системі немає єдиного сервера. Усі ланцюжки із даними блокчейна розподілені між користувачами. Блокчейн також називають технологією розподілених реєстрів, адже весь ланцюжок даних угод із усією інформацією від початку внесення цінностей зберігається комп'ютерах безлічі незалежних користувачів. Навіть якщо один або кілька комп'ютерів дадуть збій, інформація не зникне. На відміну від звичайних баз даних, змінити чи видалити ці записи не можна, можна лише додати нові. Тобто, для того, щоб здійснити транзакцію без відома конкретної людини, доведеться відредагувати інформацію від моменту реєстру цінності до останнього її переказу на десятках безлічі незалежних носіїв [1].

1.1 Метод децентралізації управління

Системи які представляють з себе сукупності пов'язаних елементів, що взаємодіють між собою, потребують управління. Причому, дана проблема стосується будь-яких систем — від програмних та апаратних технологічних

комплексів до соціальної організації товариств. Інакше, функціональність пов'язаних елементів, запланована при проектуванні може бути під загрозою. Причиною тому те, що більшість систем не здатні до ефективної самоорганізації. З цією управлінською проблематикою людська цивілізація стикалася протягом усієї своєї історії (рисунок 1.2).

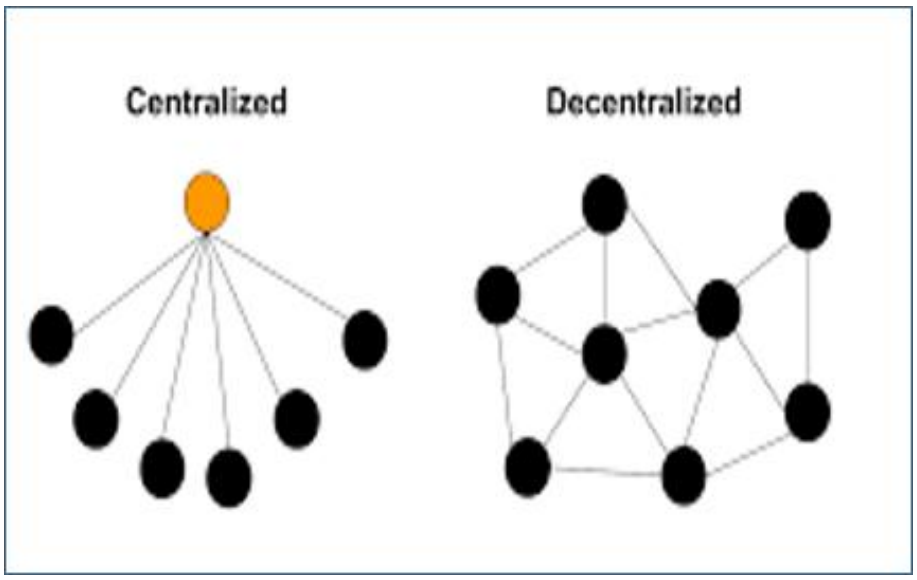


Рисунок 1.2 — Різниця централізованої та децентралізованої моделі управління

Варіанти управління системами, можна розділити на дві основні форми — децентралізовану і централізовану.

Рання форма управління соціумом природно склалася ще за часів первісних людей. У ті часи племінні групи мали підпорядковувались строгій управлінській ієрархії. Але якщо розглядати управління усім населенням, можна говорити суто про децентралізацію. Більше того, у більшості випадків, кожна група являла собою окреме ізольоване управління, тому сукупну популяцію виду Homo Sapiens складно уявити єдиною, хоч і децентралізованою системою. На великих відстанях, були відсутні управлінські зв'язки між групами, а взаємодія, мала переважно деструктивний характер. Зазвичай стики груп були спрямовані на знищення або ж асиміляцію слабких груп сильнішими. З розвитком соціальних взаємовідносин, між групами почали формуватись стійкі зв'язки. Зрештою,

це породило складні ієрархічні системи з домінуючими елементами управління.

Як тільки сукупна чисельність груп, взаємодіючих усередині ієрархії, стає відносно великою, система починає набувати рис централізованої моделі. Тобто, із збільшенням кількості соціальних об'єднань, люди створили поняття «держава». На чолі цього об'єднання встав одноосібний правитель, виборний чи спадковий. Така форма влаштування організації виявилася цілком життєздатною, оскільки дожила до наших днів, хоч з часом зазнавши певних модифікацій. Таким чином, можна констатувати, що вимушена форма соціальних об'єднань із децентралізованим управлінням, на той час, еволюціонувала в більш прогресивний централізований метод. Таким чином, централізація породила багато можливостей. Наприклад — можливість ресурсної консолідації, здійснювати проектів на державному рівні, ведення загарбницьких війни, здійснення масштабного будівництва. Історія розвитку таких могутніх держав, як Єгипет або Вавилон є яскравими прикладами.

Можна зробити висновок, що централізація це єдиний правильний і найбільш ефективний спосіб управління системами. Проте вже середньовічний. Віднедавна, із розвитком технологій та всесвітньої комунікації, яка не вимагала очних зустрічей, почали з'являтися інші управлінські форми. В даному випадку не йдеться про еволюцію управлінських принципів. У деяких випадках політичні обставини та глобальну залежність економік країн одна від одної, просто не дозволяють створювати централізовані управлінські моделі, які були б достатньо ефективними для вирішення сучасних проблем. Історія знає чимало прикладів, коли децентралізація у вигляді феодальної роздробленості країн призводила до їх ослаблення, а часом і загибелі. Дані приклади говорять про неоднозначність твердження, що одна з форм управління системами краще за іншу. Безумовно, у обох форм управління є свої плюси та мінуси. Перенесемо проведений аналіз від форм соціального устрою до технологічних систем.

Схожість соціальної та технологічної форми управління базується на загальних принципах, які зводяться до застосування сукупності методів впливу суб'єкта на об'єкт. Розглянемо управління структурою глобальної комп'ютерної мережі інтернет, як технологічний приклад. як тільки інтернет став загальнодоступним, люди стали активно використовувати його для організації різних комерційних, соціальних та державних сервісів. Цікаво, що сам по собі інтернет — це децентралізована структура, хоч і має ієрархічну природу нижніх рівнях використання. Кінцевий користувач підключається до мережі через свого провайдера який будучи невеликою організацією, має лише один зовнішній канал.

Чим більший суб'єкт мережі, тим більше зв'язків він має з іншими великими суб'єктами. Дані зв'язки можуть здійснюватись як через пункти обміну мережевим трафіком так і за допомогою прямих з'єднань. Найбільші мережеві оператори володіють власною інфраструктурою магістральних каналів, яка може бути розміщена по всьому світу. До того ж, вказані оператори забезпечують найбільш значну пропускну здатність для даних, що передаються. Разом з тим, інтернет не має єдиної «точки відмови». Тобто відключення навіть досить великого, одного учасника системи, не призведе до зупинки роботи мережі в цілому (за винятком того сегмента, на якому був повністю замкнутий великий вузол, випавший з мережі). Незважаючи на це, елементи замкненого сегменту можуть повернутися в онлайн перейшовши на резервні канали зв'язку. Тим часом, вся решта мережі збереже працездатність.

Саме такий підхід як відсутність точки відмови є однією із найважливіших переваг децентралізованих систем. Фактично, для того щоб знищити інтернет, необхідно відключити усі його робочі вузли. Така задача собою представляє неймовірну організаційну та технологічну складність, яка межує з практичною неможливістю виконання задуманого. Тобто можна говорити про теоретичну невразливість мережі, яка побудованої на основі розподіленої топології, яка не містить в собі жодного керуючого центру. Якщо розглянути ситуацію детальніше, звернувшись на рівень сервісів що

побудовані на базі мережі інтернет, то легко бачити, що переважна більшість вибірки побудована на технології «клієнт — сервер» (рисонок 1.3), тобто технології централізованої.

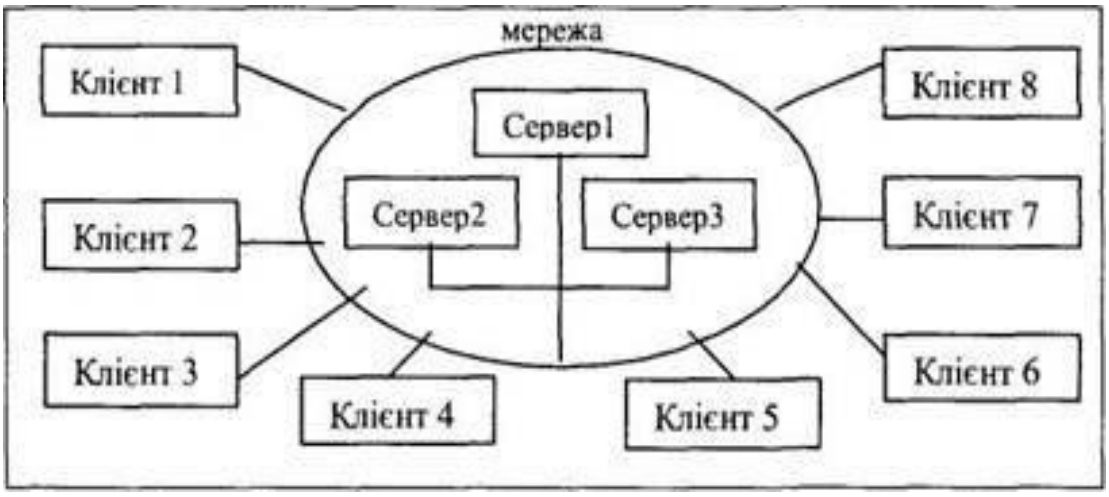


Рисунок 1.3 — Схематичне зображення технології клієнт — сервер

В сучасному світі, люди давно звикли користуватись різноманітними інтернет-сервісами. Системи хмарного зберігання даних (наприклад, фотографій, документів або відео-файлів), підтримка сервісів електронної пошти, доступ до системи «банк — клієнт» для здійснення платежів, керування власними рахунками, бронювання готелів та авіаквитків, торгові платформи для заключення та реалізації угод на фінансових ринках а також, багато іншого — усі ці сервіси побудовані на основі централізованої інфраструктури. При використанні кожної системи, яка має в основі централізовану інфраструктуру, клієнту для отримання доступу до ресурсів чи послуг, необхідно відвідати спеціальний сайт постачальника конкретної послуги, ввести свій пароль та логін (або інші персональні дані) та підключитися до центрального сервера, де зберігаються дані клієнта, його активи тощо. Однак у випадку відключення центрального серверу постачальника послуг з якоїсь причини, можливість користування конкретною послугою буде недоступна. Доведеться чекати, поки сервер усуне технічні неполадки та відновить працездатність. В даному випадку було розглянути головну проблему централізованих систем — наявність «точки відмови».

Відмова в обслуговуванні може бути результатом дії різних факторів: помилок програмного забезпечення, технологічних проблем, пов'язаних із несправністю технічного обладнання, зловживань обов'язками та правами всередині структури самого постачальника послуг, зовнішніх хакерських атак зловмисників чи дії комп’ютерних вірусів. Не останню роль можуть грати результати репресивного впливу державних силових або регулятивних структур з боку юрисдикції, в якій постачальник послуг фізично розташований. Усі вищеперераховані фактори, результатом впливу яких може виникнути відмова в обслуговуванні, змушують замислитися над тим, як можна організаційно чи технологічно уникнути подібних ситуацій. Виникнення технології блокчейн стало відповіддю це питання. Технологія заснована на побудові децентралізованої системи для зберігання та обміну даними, яка виключає усі негативні фактори, які можуть природно виникнути при використанні централізації сервісів. На зміну мережевої топології «зірка» (рисунок 1.4).

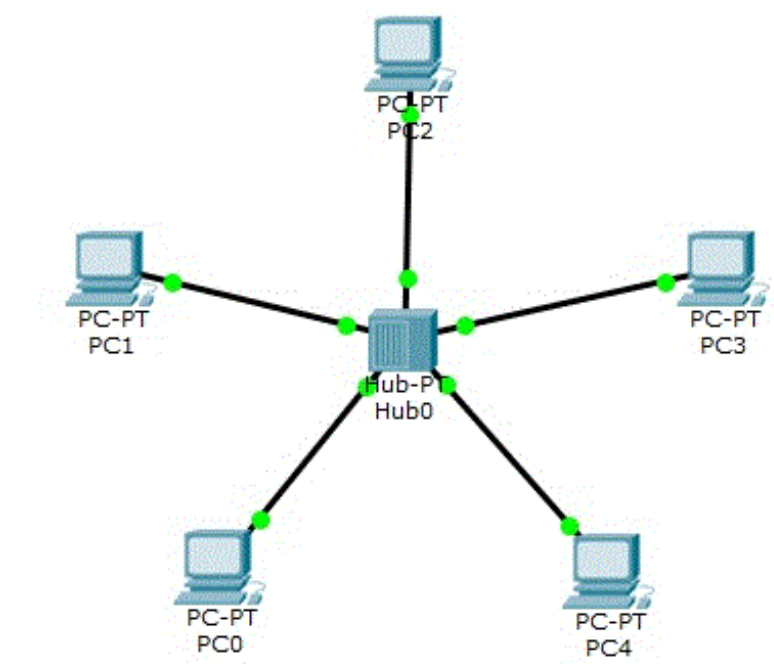


Рисунок 1.4 — Схема топології зірка

У ній промені від усіх вузлів-користувачів сходяться до вузлу — серверу — центральної точки в обов'язковому порядку, прийшла форма організації мережі, в якій поняття «центральний сервер» відсутнє як таке, а

вся взаємодія здійснюється безпосередньо між вузлами-клієнтами. Такі мережі ще називають «пірінгові» або «однорангові». Усі вузли в подібній мережі є здебільшого рівноправними, кожен із нових членів мережі може виконувати як серверні, так і клієнтські функції. Децентралізована топологія мережі, побудована таким чином, усуває фактор «точки відмови». При цьому підвищується рівень надійності та працездатності системи майже до граничної величини.

Однак, розглянемо доволі закономірне питання: якщо сервери в мережі як такі відсутні, то яким чином у подібній системі зберігаються загальні дані, як вони захищені від несанкціонованого доступу, як вони захищені від перезапису чи модифікації і як вони поширюються через мережу? Яким чином подібні системи розвиваються та обслуговуються, якщо всі учасники мережі мають однаковий пріоритет обслуговування та однакові права? Методологія технології блокчейн вирішує більшість із поставлених питань. Дані реплікуються для кожного вузла системи. Захист від несанкціонованого доступу до даних або від несанкціонованої зміни інформації забезпечується завдяки математичним алгоритмам асиметричної криптографії. Система повноцінно функціонує виходя заданого набору правил, із якими погоджуються кожен із учасників системи. В разі якщо необхідно внести будь-які зміни, рішення ухвалюється голосуванням усіх учасників системи. Рішення вважається прийнятим, якщо більш ніж 51% включно усіх голосів прийняли обраний варіант.

Слід зазначити, адміністрування децентралізованих систем значно складніше, за адміністрування централізованих, де рішення приймається одноголосно, або кількома особами на верхівці ієрархії. На даний момент вирішено далеко не всі проблеми, які можуть виникнути під час управління децентралізованими системами. У наступних розділах дане питання неодноразово підніматиметься. Детальне обговорення даної проблематики здійснюватиметься із технічних точок зору у подальших розділах.

1.2 Особливості задач, які можна вирішити за допомогою технології блокчейн

Міжнародні структури намагаються поставити технологію блокчейн на службу прозорості економіки. У 2017 році групою волонтерів було створено сайт при штабі ООН у Нью-Йорку. З моменту роботи сайту дана група закликала до спільної роботи інших співробітників організації. Незабаром кількість людей, працюючих над проєктом налічувала 85 учасників із різних країн світу. Наразі колектив співпрацює з урядами низки європейських країн, наприклад урядом Норвегії, та веде декілька пілотних проектів із запровадження у сфері містобудування технології блокчейн. При Міжнародному банку відкрито нову лабораторію, основне завдання якої — застосування технології блокчейн для створення стійких реєстрів власності і надійних цифрових ідентифікаторів особи. Міжамериканський банк розвитку [1] США та медіалабораторія Массачусетського інституту працюють над програмою, що допоможе латиноамериканським фермерам отримувати кредит на основі записів про покупки сільгосптехніки та промислового інвентарю, підтверджені блокчейн-реєстрами.

Неурядові та благодійні організації, такі як Фонд Рокфеллера та Всесвітній економічний форум, також вивчають потенціал технології блокчейн.

Децентралізована система зберігання даних може вирішити проблему дефіциту соціального капіталу. Яскравим прикладом якої є проблема табору Азрак, згадана вище в роботі. Проект ООН, в рамках якого було створено єдиний каталог транзакцій, що не виходив за межі спільноти, допоміг людям створити взаємну довіру та платформу для обміну цінностями.

Проблема недовіри та підозрілості стара як світ, проте тепер, володіючи новим, безпрецедентно потужним інструментом, що направлений на її вирішення, спільнотам легше накопичувати соціальний капітал. Для країн, які розвиваються, дана можливість особливо важлива, оскільки це дозволяє економіці країни наблизитись до моделі економіки першого світу. Ряд звичайних громадян зможуть скористатися економічними

можливостями, яких їм не дано від народження Наприклад, власники нерухомості з невисоким доходом зможуть брати кредит на покращення житлових умов; дрібні вуличні торговці матимуть доступ до страхових програм.

Однак потенціал блокчейна помітний не тільки як засіб для розвитку країн, або для застосування у гуманітарних місіях. До цього часу економічна взаємодія між суб'єктами здійснювалася виключно при залученні реєстраційних служб, банків та інших формальних установ. Дані служби виступають як довірені третій сторони та фіксують проведення операцій. Такий підхід що дозволяє забезпечити довіру економічній системі, обмінюватися матеріальними та нематеріальними цінностями, зменшуючи при цьому ризик шахрайства та вибудовувати повноцінне суспільство. Однак, проблема в тому, що дані установи виступають виключно як посередники і до того ж, мають надто велику силу, вирішуючи, кого впускати чи не впускати у поле фінансової взаємодії. До того ж, даний механізм не надто надійний.

Яскравим прикладом цьому є криза 2008 року, коли банки відверто порушили зобов'язання вести прозорий облік. До того ж, послуги усіх перелічених установ не безкоштовні. Це дозволяє їм встановлювати надмірно високі комісії або відсотки за кредитами. Неефективність та нерентабельність посередників нерідко призводить до відмови від укладення угод та транзакцій. Знайшовши спосіб уникнути послуг банківських установ виникає можливість не тільки до заощадження коштів, а і до створення нових моделей підприємницької діяльності.

Поява інтернету вже позбавила людей частини необов’язкових посередників, блокчейн ж продовжує цю тенденцію. Ще десять років тому було важко собі уявити, що можливо комфортно себе почувати в машині незнайомця, з яким щойно вперше сконтактував через смартфон?

Однак сервіси на кшталт Uber та BlaBlaCar допомогли суспільству подолати бар'єр недовіри завдяки вбудованій системі рейтингу водіїв та

пасажирів. Це стало можливим лише з розвитком цифрових комунікацій та соціальних мереж.

Наведений приклад показує — якщо технологія допомагає вирішити проблему довіри та забезпечити почуття безпеки, люди цілком готові і здатні до прямої взаємодії із незнайомими людьми. Усе це відкриває шлях до економічної моделі peer-to-peer (P2P) (рисунок 1.5).

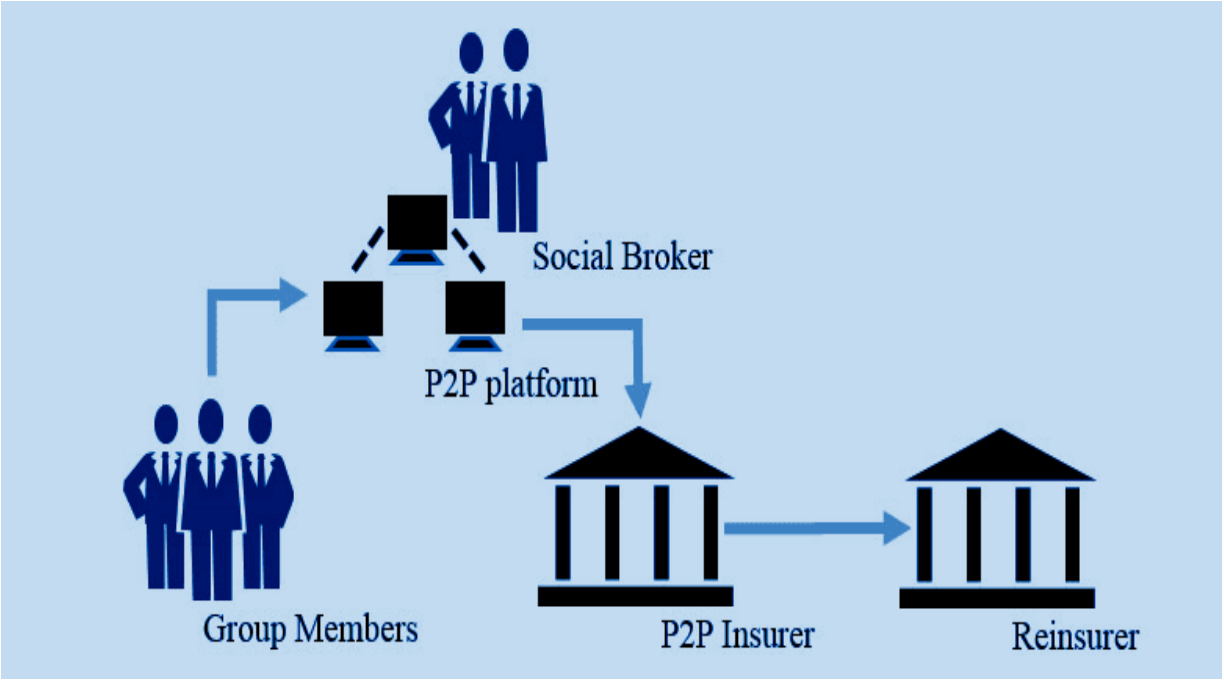


Рисунок 1.5 — Модель peer-to-peer

Технологія блокчейн ставить логічне питання — навіщо зупинятися на Uber? Можна уникнути користування послугами компанії, якщо вона забирає собі 25 відсотків із кожної поїздки та зловживає доступом до інформації про пасажирів [7] Якскравим прикладом повнісю децентралізованої системи ізраїльська платформа Commuterz що базується на основі блокчейну. У цьому випадку платформа не має конкретних власників. Вона використовує програмне забезпечення із відкритим кодом, який може завантажити собі кожен. Усі користувачі платформи ведуть розрахунки у цифровій валюті, не витрачаючи кошти на те, щоб оплатити 25% комісії за обслуговування посередництва уявною фірмою.

Поклавши управління соціальним капіталом на децентралізовану мережу, замість низки уповноважених посередників, створивши нові цифрові

токени та активи, відкривається можливість змінити саму природу громадської організації. Створюються нові підходи до співпраці та взаємодії, що перетворить багато галузей та організацій.

Потенціал блокчейна цілком підходить для описаного типу взаємодії суспільства. Як приклад, коротка вибірка результатів, які можна отримати у найближчому майбутньому:

— невразливі реєстри власності, за допомогою яких можна довести право на володіння будинком, автомобілем та іншими активами;

— децентралізовані обчислювальні системи та сховища даних, що витіснять корпоративні хмарні обчислення та веб-хостинг, для підтримки яких вистачить потужностей звичайного персонального комп'ютера;

— децентралізація змі та контенту, що дозволить музикантам, художникам та усім виробникам унікального контенту розпоряджатися та керувати своїм «цифровим активом»;

— миттєві, прямі, захищені міжбанківські операції, що дозволить вивільнити трильйони доларів на міжбанківському ринку, що зараз витрачаються на проведення транзакцій через десятки спеціалізованих установ протягом двох — семи робочих днів;

— ланцюги постачання на основі блокчейну — використання постачальниками загальної інформаційної платформи значно підвищить прозорість та ефективність усіх операцій з виробництва товару;

— децентралізований «інтернет речей», де будь-які пристрої можуть обмінюватися інформацією без посередників, що призведе до великих змін у сфері логістики, а також створення децентралізованих енергомереж;

— цифрові посвідчення особи, які можна отримати без участі державних структур та бюрократичних інстанцій.

Багато великих корпорацій також бачать у новій технології можливість звільнити потоки капіталу і направити їх у вигідніше русло. У будь-якому випадку багато галузей та підприємств вважають за потрібне поекспериментувати з технологією блокчейн, щоб зрозуміти, до чого приведе її розвиток.

У банківській сфері — галузі, яку цифрові валюти загрожують звести нанівець, починають усвідомлювати, що блокчейн-протоколи можуть значно спростити громіздкі процеси переказу та зарахування коштів, а також схвалення та підтвердження операцій. Використовуючи надійний загальнодоступний реєстр, який можна оновлювати одночасно в режимі реального часу, банки істотно знизили вартість транзакцій і вивільнили капітал для нових вкладень. Це чудова новина для інвестиційних банків, таких як Goldman Sachs, але потенційна загроза для депозитарних банків на кшталт State Street або клірингових компаній на кшталт Deposit Trust та Clearing Corporation, чия бізнес-модель базується на виконанні посередницьких функцій. Проте всі види фінансових організацій зараз потребують досліджувати нову технологію.

Наприклад, науково-технічна лабораторія R3 CEV отримала 107 мільйонів доларів від ста найбільших фінансових компаній світу на розробку технології розподіленого реєстру [10]. Запропонована командою R3 платформа Corda побудована на базі блокчейну, хоч це й не відображено у назві. Вона має відповідати моделям та правилам банківської справи, при цьому здешевлюючи захищені міжбанківські операції на трильйони доларів.

Втім, не треба думати, що нова сфера вже захоплена міжнародними корпораціями. Цього разу інвесторів спокусила нова форма залучення інвестицій — ICO, або первинна пропозиція монет (токенів) — суть якої полягає у продажі інвесторам фіксованої кількості одиниць цифрової валюти, отриманих шляхом разової або прискореної емісії. Цей інструмент краудфандінгу заснований на технології блокчейн. Стартапи, що виникли на хвилі інтересу до ICO, просувають нові децентралізовані програми, які можуть зробити революцію в будь-якій сфері від реклами до медицини. У їх основі лежать спеціальні токени, які служать як залучення інвестицій, так створення мережі користувачів. Механізм частково нагадує роботу краудфандингових сайтів (наприклад, Kickstarter), однак у цьому випадку покупець токена має шанс швидко заробити на вторинному ринку. На момент написання цих рядків рекордна сума, отримана від попереднього

продажу токенів під час проведення ICO, становить 257 мільйонів доларів. Саме стільки заробила мережа лабораторій Protocol Labs, випустивши токен під назвою Filecoin для просування нового протоколу децентралізованої мережі IPFS. Ідея проекту зводиться до встановлення користувачами спеціального програмного забезпечення, яке дозволяє зберігати дані на жорстких дисках комп'ютерів. За наданий простір на диску вони отримують токени, які потім можна обміняти на біткоїни або інші криптовалютні одиниці. Цілком ймовірно, що багато ICO порушують протоколи безпеки і рано чи пізно мильна бульбашка лусне, причому постраждають ні в чому не винні інвестори. Однак у новому цифровому бумі є щось освіжаюче, демократичне. Цілі когорти дрібних інвесторів отримали доступ до перших стадій капіталовкладень — привілеї, як правило, венчурних інвесторів та інших професіоналів фінансової сфери.

1.3 Хешування даних

Інструмент хешування даних є невід'ємною та важливою частиною технології блокчейн. Хешування використовується для формування адресації в блокчейн-системах, для створення цифрового електронного підпису повідомлень. Перш ніж розглядати елементи блокчейн-систем, потрібно розглянути як працює процедура хешування даних.

Хешування — це метод перетворення набору даних довільного розміру в стандартизований рядок сталої довжини. Виконується хешування за допомогою спеціального алгоритму. Якщо розглянути певний набір даних, (наприклад усі знімки конференцій проведених в університеті), то можна створити цифровий відбиток даних, довжиною в десять символів. При проведенні хешування необхідно обрати алгоритм перетворення вхідних даних і використовувати алгоритм без зміни інших даних довільного розміру, отримуючи на виході стандартний рядок в десять символів. У такому разі, можна стверджувати що при процедурі хешування використовується «детермінований алгоритм», адже саме такі алгоритми завжди видають

наперед визначений результат. В кінці процедури, отриманий результат є унікальним відображенням вхідних даних, які були перетворені.

Для забезпечення виконання даної умови необхідно створити такий алгоритм перетворення, який за жодних обставин не допустить отримання однакового результату перетворення для різних вхідних наборів даних. Тобто не виникне жодних так званих колізій. При цьому найменша зміна впорядкованого набору вхідних даних, включно із зміною єдиного біта, повинна видозмінювати отриманий результуючий хеш на виході на абсолютно новий вихідний код. Прикладом роботи одного з найпростіших алгоритмів хешування є алгоритм SHA-1. В роботі даного алгоритму прообразами хешів являються два варіанти написання англійського слова «децентралізація», при цьому в другому слові змінено лише одну літеру (рисунок 1.6).

Рисунок 1.6 — Результат хешування слова, зміненого на одну літеру

Дивлячись на отримані результати легко бачити, що другий хеш немає нічого спільного з першим, незважаючи на те, що різниця у вихідних прообразах мінімальна. З першого погляду може виникнути питання — навіщо взагалі це потрібно? Насправді хешування — виключно корисна функція, яка доволі широко застосовується у комп'ютерних технологіях. Розглянемо ситуацію, коли необхідно передати незахищеним каналом зв'язку значний обсяг даних. При цьому, при передачі інформації з тих чи інших причин можуть виникати перешкоди та спотворення. Постає питання: чи дійшов до кінцевого отримувача набір даних у вихідному вигляді? Якщо

буде проводитись порівняння кожного біту вихідної інформації з отриманим, можна із впевненістю говорити про вдалу передачу даних, без помилок та втрати даних.

Тепер виникає ряд проблем — якщо під час передачі дані втрутився хтось сторонній і навмисно спотворить інформацію? Якщо обсяг інформації складає гігабайти даних? Процес порівняння двох гігантських інформаційних блоків може забрати значний час. Чи не простіше до блоку даних, що передається, додати короткий унікальний «цифровий відбиток», створений за допомогою загальновідомого алгоритму хешування? В такому випадку, при отриманні унікального коду можна ще раз запустити алгоритм перетворення, надавши на вхід отримані дані, а потім просто порівняти результуючий хеш з тим, який був доданий до даних, які передавались. Якщо результуючі хеш-коди збігаються, то можна зробити висновок, що передача інформації пройшла без спотворень. Таким чином, маємо на руках дані, повністю аналогічні вихідним. Так здійснюється перевірка цілісності даних. Популярним варіантом використання алгоритму подібної перевірки є отримання так званої «контрольної суми». Розрахунок контрольної суми базується на алгоритмі хешування вхідної сукупності даних (рисунок 1.7).

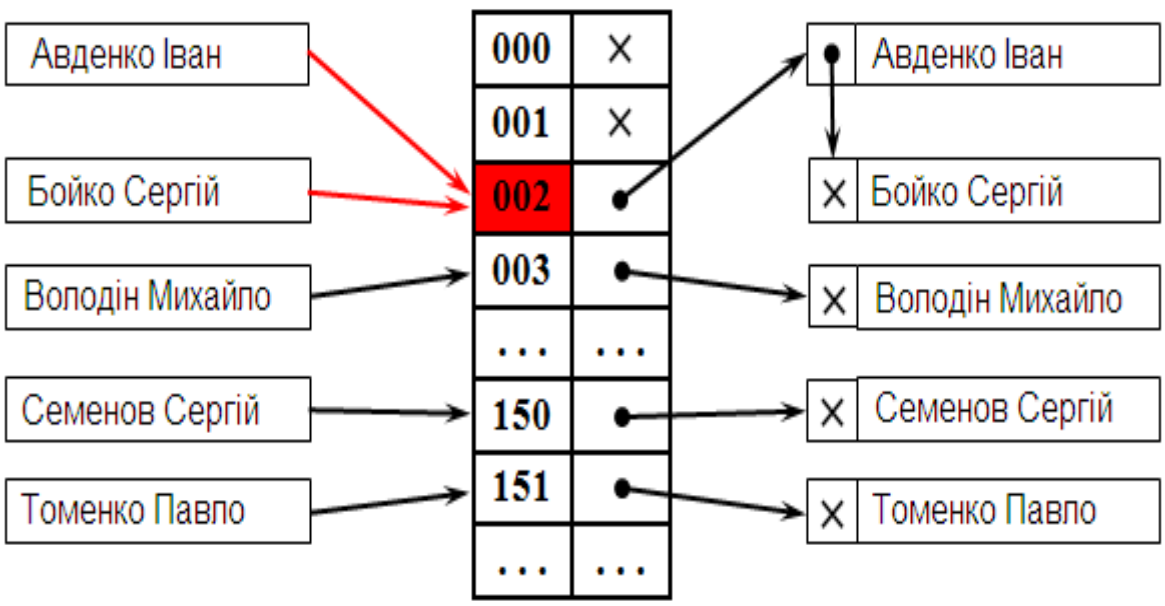


Рисунок 1.7 — Схематичне зображення алгоритму хешування даних

Як приклад роботи алгоритмів хешування розглянемо найбільш популярний алгоритм хешування, який базуються на технології блокчейн. Наприклад Ethereum (SHA-3) або Bitcoin (SHA-256). Дані алгоритми складаються з заданої кількості ітерацій, в кожній з яких з виконуються логічні операції з наступного набору:

— конкатанація (склеювання двох блоків даних, при якому другий стає продовженням першого. наприклад, конкатанація «ppp» та «ннн» дає «pppннн»);

— додавання (звичайне арифметичне додавання для двох чи більше чисел);

— кон'юнкція, або логічне і (результат операції буде істинним (1), у випадку коли обидва біти є одиницями, в іншому випадку результат вважається хибним (0));

— диз'юнкція, або логічне або (результат операції буде істинним (1), якщо хоча б один із аргументів є істинним (1), в іншому випадку результат вважається хибним (0));

— логічне виключне або, xor (результатом операції для двох біт буде істина (1), тільки якщо один з аргументів буде істинним (1), а другий помилковим (0), інакше результат буде помилковим (0));

— логічне заперечення, not (побітова інверсія, результат одинарної операції, при якій результуючий біт завжди буде протилежний за значенням, тобто одиниці перетворюються на нулі, а нулі на 1);

— побітові зрушення (значення бітів здвигаються до сусідніх регістрів за напрямом зсуву, наприклад, для блоку «01110001» результатом логічного зсуву вліво буде «11100010»), також побітові зрушення можуть бути логічними (коли останній біт у напрямку зсуву втрачається, а перший стає нулем) і циклічними (коли останній біт у напрямку стає місцем першого). у наведеному вище прикладі розглядається циклічний зсув.

Розмірковуючи логічно, ми приходимо до розуміння, що майже неможливо перетворити малий блок даних (хеш-код) для отримання початкового блоку великого обсягу інформації. Нарешті, технологія

хешування активно використовується для прискорення пошуку необхідних даних. Для цього формуються так звані «хеш-таблиці». Хеш-таблиці містять хеші різноманітних інформаційних блоків. Вони сортуються у певному порядку, таким чином, що при здійсненні пошуку можна швидко знайти дані, звернувшись до їх хеш-кодів. Таким чином, здійснюється звернення одразу до потрібного розділу, замість масштабного пошуку по всій базі.

Для того щоб уявити собі проблематику, пов'язану з криптостійкістю найпопулярніших алгоритмів хешування, оцінимо розраховані показники різноманіття варіантів хешів та ймовірностей знаходження колізій для них. Співвідношення між розрядністю (розміром) хешу n і числом можливих виходів (варіантів генерацій хешу) становить 2 ступенем n. Якщо середня довжина хеша в основних популярних блокчейн-проектах становить 256 біт, це означає кількість виходів, що дорівнює 2256 або приблизно 1,2×1077, тобто значення, наближене до кількості атомів у Всесвіті.

1.4 Принципи роботи блокчейну

Цінність моделі функціонування блокчейн — мереж полягає в комбінуванні різних інструментів, технологій та принципів, які, будучи певним чином поєднаними, формують логічну та захищену структуру для розподіленого зберігання даних. Що ж є блокчейн? Фактично його можна порівняти з великою книгою бухгалтерського обліку, на сторінках якої записуються фінансові операції, що проводяться між сторонами. Тільки ця книга складено так, що кожен запис, який до нього потрапляє, не може бути згодом ніяким чином змінена чи видалена — цьому будуть перешкоджати серйозні криптографічні алгоритми, інтегровані в технологію. Самі ж дані зберігаються не в якомусь конкретному місці, що має статус керуючого центру, а копіюються та синхронізуються, або, інакше кажучи, реплікуються між усіма учасниками системи. вузлами мережі.

Таким чином, навіть якщо хтось захоче поміняти збережені у дані, то інші учасники системи просто не приймуть до уваги ці зміни, оскільки вони були проведені всупереч прийнятим у системі правилам. Як же влаштовано

таку «бухгалтерську книгу»? Її «сторінки» називаються блоками. Так само, як і сторінки в звичайній книзі, блоки йдуть один за одним другом у строгому пронумерованому порядку. Однак якщо звичайну сторінку можна з книги вилучити або за бажання перемістити в інше місце, а то й зовсім викинути, то з блоками так обійтися не вдасться. Усі блоки жорстко зчеплені між собою спеціальними криптографічними «замками», зламати які, навіть теоретично, винятково складно. Звідси, власне, і назва технології — блок-чейн(від англійського blockchain — «ланцюжок блоків») Щоб стати надійним сховищем даних, будь-яка блокчейн-структура повинна задовольняти наступним критеріям:

— мати децентралізовану технологічну основу, тобто вміти поширювати між усіма вузлами мережі необхідні дані та підтримувати їх актуальний стан через процеси реплікації та синхронізації;

— підтримувати нерозривний зв'язок між блоками даних шляхом формування в кожному новому блоці посилання на попередній по відношенню до нього блок;

— вміти ефективно кодувати масиви даних в унікальні інформаційні блоки стандартного розміру, інакше кажучи, хешувати дані;

— застосовувати виключно стійкі до злому криптографічні алгоритми, необхідні захисту записуваних блоків даних;

— використовувати елементи спеціального підрозділу математики (теорії ігор) — для того, щоб усі вузли системи дотримувалися встановлених правила та досягали загального консенсусу при створенні нових блоків та записи у яких даних.

Усі перераховані вище твердження складають п'ять основних «стовпів», на яких базується технологія блокчейн. Надалі ми розглянемо кожен із них детальніше.

Основні принципи, на яких базується блокчейн, такі:

— розподілений регістр 2.0, побудований за принципом книги обліку та розподілений між усіма учасниками;

— децентралізація та відмова від посередництва: блокчейн не контролюється жодним центральним органом, у цій довірчій системі відносин між двома учасниками немає третіх осіб;

— консенсус — факт прийняття транзакції чи відмовитися від неї є результатом розподіленого консенсусу, а чи не рішення якогось централізованого інституту;

— незмінність та стійкість: неможливо змінити або знищити записи;

— розподілена довіра та прозорість — поділяються дані, операції та консенсус.

1.5 Структура блокчейну

Блокчейн був визначений як ланцюжок блоків, у якому кожна транзакція шифрується для того, щоб утворити наступний блок. Кожна слідуюча за нею, транзакція, своєю чергою, шифрується з урахуванням попереднього блоку. Саме такий підхід і призвів до виникнення самого поняття ланцюжка блоків, тобто блокчейну. Біткойн-адреса представлена у форматі АБСМ [11] за допомогою спеціалізованого кодування 58 буквено-цифрових символів. Перелік доступних символів складають цифри, великі та малі літери, за винятком літер і цифр I, I, O та O, які засновник технології Сатосі Накамото виключив, оскільки у деяких шрифтах вони виглядають однаково. Перша створена адреса мала вигляд: 1A1zP1eP5QGefi2DMPTfTL5SLmv7Divfna44. Біткойн-адреса — це єдина інформація, необхідна для утворення або підтвердження транзакцій. Для прийому не потрібно програмного забезпечення біткойн, достатньо повідомити адресу, і тільки платник відповідатиме за відправлення транзакції в решту мережі. Для створення та організації роботи блокчейну необхідні реєстр, алгоритм для перевірки транзакцій реєстр (рядок блоків, наприклад біткойн), шифрування з ключами для захисту угоди та однорангова мережа.

Для прикладу розглянемо блокчейн біткойн, та опишемо його функціонування, розбивши на чотири етапи:

— етап 1 — два учасники погоджують умови транзакції (передачу грошей, активи, фінансові документи тощо);

— етап 2 — журнал сканується членами мережі, і за допомогою аналізу його хронології члени мережі засвідчуються, що продавець справді має заявлені активи або фонди, які він продає;

— етап 3 — якщо все гаразд, транзакція підтверджується та додається до останнього блоку ланцюга;

— етап 4 — журнал поширюється для кожного із учасників мережі, розповсюдженість якого є гарантією захищеності даних та для фальсифікації чи видалення хоча б однієї транзакції необхідно змінити дані журналів для всіх членів (вузлів) мережі.

Таким чином, для отримання статусу достовірності, кожна угода має бути підписана асиметричною криптографією типу закритий ключ/ відкритий ключ. Отже, для здійснення транзакції в блокчейні типу біткойн необхідні три види інформації: особистий ключ дебетового адресу, загальний ключ кредитованого адресу, сума транзакції.

1.6 Технологічний розподіл технології блокчейн

Технологію блокчейн, за своїм алгоритмом, можна поділити на 3 види:

— блокчейн 1.0 — даний вид блокчейну використовується переважно в криптовалюті. прикладами можуть бути ethereum, litecoin, bitcoin;

— блокчейн 2.0 — це розумні контракти (smart contracts), який використовується в широкому класі фінансових додатків, що виконують роботу з ф'ючерсами, облігаціями, акціями, заставними та іншими видами фінансових активів, та у даній роботі буде розглянуто саме цей вид блокчейна;

— блокчейн 3.0 — інші додатки, що засновані на даній технології та використовуються поза фінансовою сферою.

1.7 Блоки

Кожен блок, що знаходиться в ланцюжку, містить в собі заголовок та список транзакцій. У заголовку міститься хеш попереднього блоку, хеш транзакцій, унікальний власний хеш та інша додаткова інформація. Оскільки у кожному блоці записаний унікальний хеш попереднього блоку то практично неможливо змінити будь-яку інформацію блоку, не змінивши при цьому записи всього ланцюжку унікальних хеш-кодів, починаючи з першого блоку. Не можна навіть видалити транзакцію чи вставити її між уже проведеними транзакціями. Блоки утворюють собою лінійну послідовність (ланцюжок). Звідси і походить слово назва технології — блокчейн. Блоки додаються в ланцюжок рівномірно, незалежно від кількості проведених транзакцій. Проміжок часу додавання нових блоків специфічна для кожного виду блокчейну. Наприклад час запису нового блоку в ланцюжок для Bitcoin складає 10 хвилин, для Ефіріуму 17 секунд.

1.8 Смартконтракти

Коли йдеться про фінансові питання, сучасній людині досить важко обійтися без посередників у їх вирішенні, чи то купівля будинку, страхування майна чи придбання цінного паперу. Значно спростити та здешевити цю процедуру можуть допомогти розумні контракти. Розумний договір — це програмний алгоритм, здатний виконувати функції паперового документа. Коли відбувається певна подія, розумний контракт виконує певні пункти договору, прописані у ньому. Важливою особливістю розумних контрактів є те, що вони знаходяться не на сервері, а безпосередньо в ланцюжку блоків, а отже мають надійний криптографічний захист. Розумні контракти стали другим ступенем у розвитку технологій блокчейн. Після успіху Bitcoin з’явилося безліч альтернативних криптовалют, які загалом копіювали його код, але при цьому вносили ряд додаткових можливостей, що залучають користувачів. Однією з таких функцій стали розумні контракти. Яскравим

прикладом криптовалютного проекту, що реалізує цю особливість, може бути Ethereum, створений нашим співвітчизником Віталієм Бутеріним.

На сайті Ethereum підкреслюється, що платформа може бути використана для «кодифікації, децентралізації, забезпечення безпеки та торгівлі практично всім — доменними іменами, фінансовими активами, краудфандінгом, управління компаніями та інтелектуальною власністю». Ethereum має ряд конкурентів, таких як NEM, Hyperledger Fabric, але «золотим стандартом» у світі смарт-контрактів на сьогоднішній день є він. Розумні контракти мають низку переваг і недоліків. До плюсів даного винаходу можна віднести надійність, незмінність, прискорення обміну активами, скорочення витрат за рахунок відсутності третіх осіб під час угод, зручність. Проте розумні контракти мають певні ризики. Перший — можливість помилки програміста, що призведе до невірному виконання умов договору. Крім цього, розумні контракти запускаються при настанні певних подій, і їх дію не можна скасувати у разі непередбачених ситуацій, отже, вони не мають достатньої гнучкості. Іншим недоліком смарт-контрактів є нерозуміння їх принципів роботи широкими масами, що суттєво обмежує коло застосування.

1.9 Механізми досягнення консенсусу

Найбільш цінною ланкою блокчейну являються алгоритми досягнення консенсусу. Саме ці алгоритми забезпечують надійність технології. На даний момент існують 3 основні механізми досягнення узгодження.

1.9.1 Proof of work

Доказ роботи (proof of work). Proof of work являється протоколом захисту системи Якщо хтось забажає записати блок в існуючу базу даних ланцюга, то йому необхідно виконати певне складне завдання, побудоване за принципом односторонньої функції. Процес обчислення займає тривалий термін, у той час як сторона, що приймає, швидко перевіряє отриманий результат. Перед надсиланням повідомлення, до заголовка додавалася деяка

позначка, підтвердити валідність якої можна лише повним перебором. Перевірка обчислень на стороні приймається швидко — за рахунок одноразового обчислення SHA-1 з заздалегідь підготовленою міткою. На даний момент саме алгоритм доказу роботи заслужив найбільший авторитет серед інших механізмів створення надійних систем. Справа в тому, що саме він здатний протистояти «атакам Сівіли», суть яких полягає в тому, що зловмисник створює безліч підроблених учасників і таким чином схиляє консенсус у свій бік (рисунок 1.7).

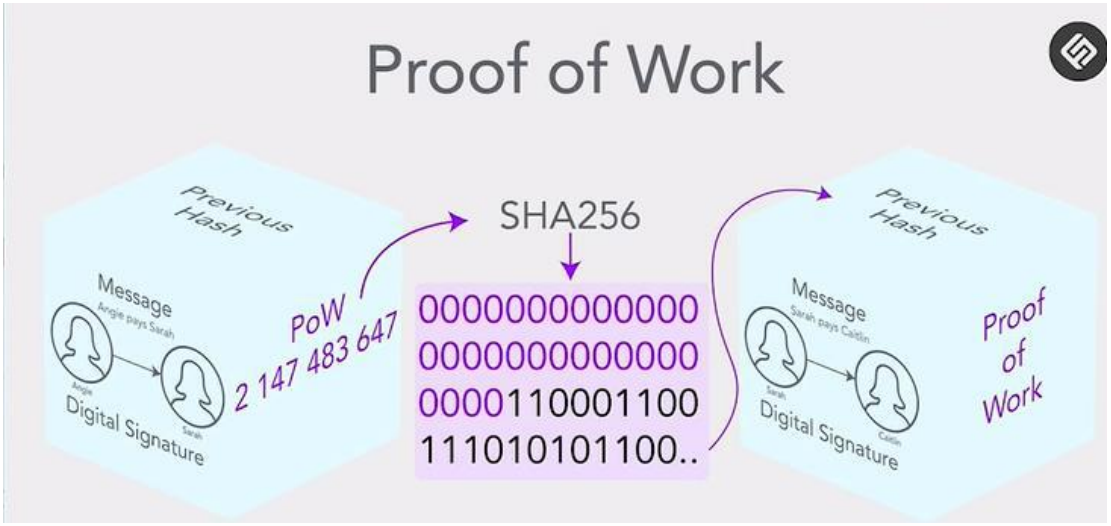


Рисунок 1.7 — Схематичне зображення алгоритму перевірки proof-of-work

Проведення подібної атаки ускладнює алгоритм доказу виконання роботи, так як для її виконання шахраю доведеться витратити колосальну обчислювальну потужність. Також більшість Блокчейнів стягують комісію за участь у консенсусі, отже «атака Сівіли» стане дуже дорогою операцією. Нерідко алгоритм proof-of-work піддається критиці через надмірну енерговитратність, але поки що це єдиний засіб протистояння втручанням у систему подібного роду.

1.9.2 Proof-of-stake

Proof-of-stake (доказ частки) — це протокол захисту, створений як альтернатива протоколу proof-of-work. Для запису нового блоку в базу даних, необхідно підтвердити існування певної суми на рахунку. При створенні наступного блоку, із найбільш високою ймовірністю, системою буде обрано

учасника із великою кількістю коштів на рахунку. Ймовірність такого вибору залежить від потужності процесорів устаткування. Щоб підірвати надійність системи, учасник повинен володіти понад 50% усіх засобів системи, що являється надзвичайно затратним методом. У порівнянні з proof-of-work, протокол proof-of-stake має більше переваг (рисунок 1.8).

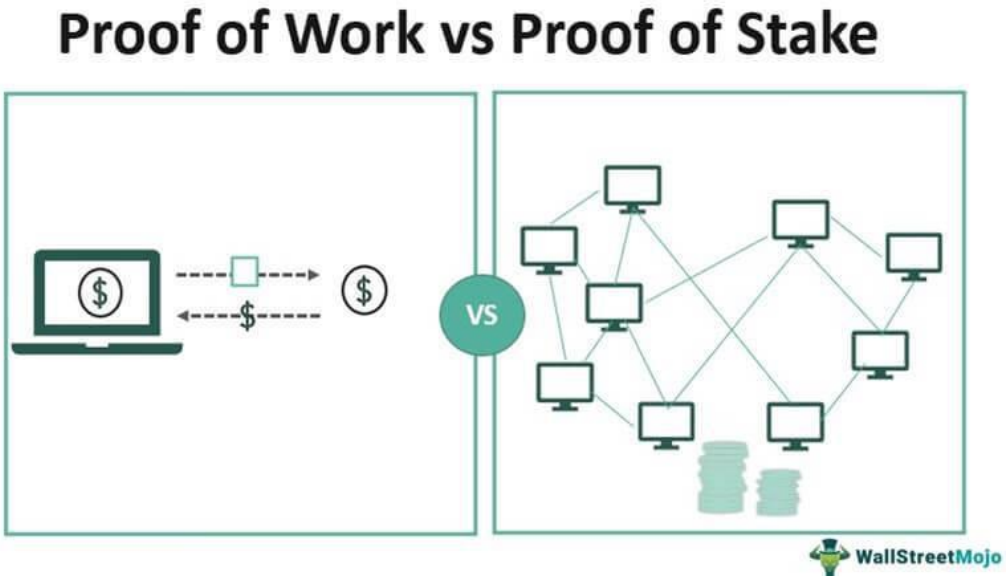


Рисунок 1.8 — Порівняння методу proof–of–work і proof–of–stake

Головна перевага методу — менші витрати часу (немає потреби у тривалих обчисленнях), проте це позбавляє можливих проблем. Також немає доказів ефективності захисту від ризиків, що виникає в криптовалютах. Два суттєві плюси цього протоколу — атака на систему коштує дуже дорого, і якщо якийсь учасник її все ж таки проведе, то сам істотно від цього постраждає, оскільки порушить стійкість системи. Аргументи проти — спосіб дає мотивацію накопичувати кошти окремих рахунках, що ставить під питання децентралізацію; у разі утворення невеликої кількості учасників, що зосередили у своїх руках більшість коштів, ця група може нав'язати свої умови функціонування системи.

1.9.3 Delegated-proof-of-stake

Delegated-proof-of-stake — удосконалена версія протоколу захисту proof-of-stake, специфіка якої полягає в тому, що блоки породжуються зумовленим безліччю користувачів системи (101 делегат), які отримують

винагороду за свій обов'язок та караються за зловмисну поведінку (такі як участь у подвійному витраченні коштів). Список користувачів, які підходять для підписання блоків, періодично змінюється відповідно до певних правил; наприклад, у Slasher делегати обираються виходячи з їх частки та історії блокчейну. Делегати можуть отримувати голоси всіх користувачів, сила голосу залежить від частки валюти у голосуючого. Delegated-proof-of-stake має ті ж переваги та недоліки, що й proof-of-stake.

У даному розділі було проаналізовано управління децентралізованими фінансами, а саме — управління децентралізованими фінансами за допомогою технології blockcahin. Наведено детальне описання структури технології, описання особливостей залач, які можуть бути вирішені із використанням даної технології, принципи та алгоритми створення програм, які базуються на технології blockchain. Основним підходом до управління фінансами пропонується використати метод децентралізованого управління, за основу взяти вид блокчейну Ehterium, алгоритм хешування SHA-1. Саме такий набір методів дозволяє підвищити продуктивність програмного забезпечення, збільшити швидкість транзакцій та зменшити витрами на комісію за перекази.

2 ЗАСОБИ РЕАЛІЗАЦІЇ ТЕХНОЛОГІЇ БЛОКЧЕЙН

2.1 Кроссплатформенний засіб для створення програмних застосунків React

Постійна потреба в покращенні користувацького досвіду тримає розробників на ногах і змушує їх адаптуватися до нових технологій і тенденцій, які забезпечать кращі результати. Однією з таких технологій, яка привертає увагу всіх завдяки своїм вражаючим характеристикам, є React JS (рисунок 2.1). Його популярність можна оцінити тим фактом, що навіть великі програми, такі як Facebook, WhatsApp, Netflix та Instagram, використовують React JS. Отже, що робить React JS провідною технологією для розробки мобільних додатків та веб-сторінок?

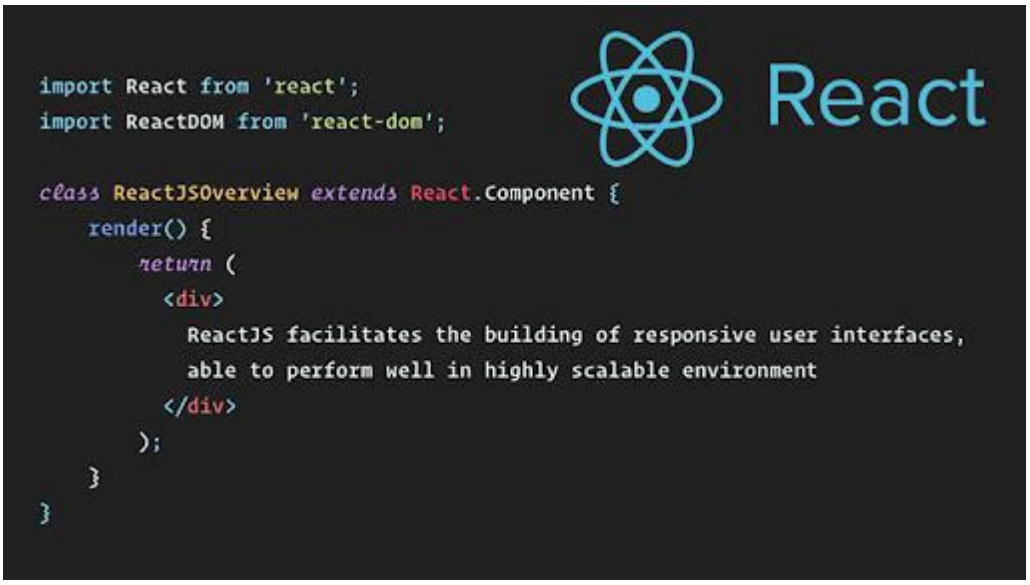


Рисунок 2.1 — Приклад програмного коду, написаного мовою React JS

Розглянемо ряд причин. Насамперед — технологія легка в опануванні. Перш ніж почати використовувати технологію для розробки веб-додатків та мобільних додатків, потрібно вивчити основи технології та зрозуміти, як вона функціонує, щоб використовувати усі її функції. React JS — це не повнофункціональний фреймворк, а бібліотека графічного інтерфейсу JavaScript з відкритим вихідним кодом, який зосереджений на одній конкретній справі та спрямований на ефективне виконання завдання. Це View у шаблоні Model-View-Controller (MVC). Будь-якому розробнику JavaScript було б легко осягнути основи React JS і почати розробляти веб-

додатки використовуючи при цьому React JS протягом кількох днів після опанування документів та навчальних посібників.

2.1.1 Написання компонентів за допомогою JSX

Прості в написанні компоненти React JS використовує JSX, який є додатковим розширенням синтаксису JavaScript. Написання компонентів стає набагато простіше з JSX, оскільки він дозволяє змішувати HTML з JavaScript (рисунок 2.2). Завдяки меншій кількості правил, яких він дотримується, можна писати код швидше та прозоріше, в порівнянні з іншими сучаними технологіями. JSX може бути не найпопулярнішим розширенням синтаксису, яке існує. Але він може виявитися досить ефективним для розробки спеціальних компонентів або великих програм.

2.1.2 Можливість повторного використання компонентів

React JS підтримує можливість повторного використання компонентів. Можливість повторного використання компонентів є великою перевагою для розробників. React JS дозволяє повторно використовувати компоненти (рисунок 2.2), розроблені для іншої програми, що реалізує аналогічну функціональність.

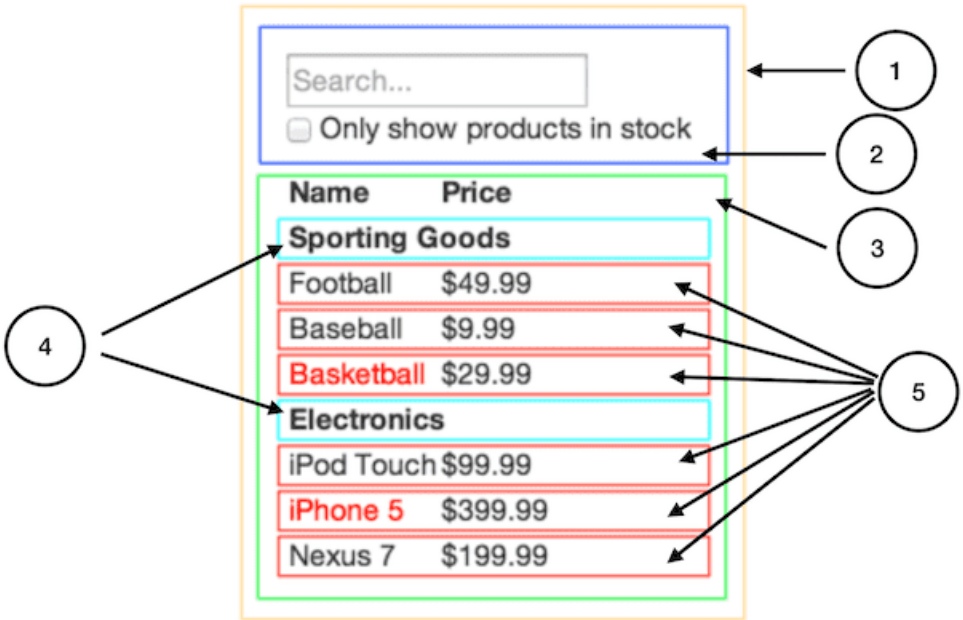


Рисунок 2.2 — Приклад розбиття користувацього інтерфейсу на компоненти, які можна повторно використовувати

Це допоможе зменшити витрати та зусилля на розробку, забезпечуючи безперебійну і майже бездоганну роботу компонентів. Коли React зустрічає такий елемент, він збирає всі JSX-атрибути та дочірні елементи в один об'єкт і передає їх нашому компоненту. Цей об'єкт називається пропси (props). Компоненти можуть посилатися на інші компоненти у поверненому дереві. Це дозволяє нам використовувати ту саму абстракцію (компоненти) на будь-якому рівні нашої програми. Неважливо, чи було написано код кнопки, форми або цілого екрану — всі вони, як правило, є компонентами в React-додатках.

В результаті зростає швидкість розробки додатків, що економить дорогоцінний час як розробників, так і клієнтів.

Не допустимо пропускати стрічки, так рисунок на наступній стор., якщо не вміщується, але викладення тексту можливе тут далі.

2.1.3 Ізоморфні програми

Технологія дозволяє створювати ізоморфні програми. Ізоморфні програми (або ізоморфний JavaScript) дають змогу використовувати один і той самий код як для клієнтських, так і для серверних компонентів програми. Це один із підходів до розробки додатків, який забезпечує перевагу швидкості під час візуалізації на сервері. Це також розширює можливості пошукових систем індексувати сторінки веб-сайту та пропонує кращий досвід для користувачів. За допомогою React JS можна створювати компоненти, які безперебійно працюють як на стороні сервера, так і на стороні клієнта.

2.1.4 Віртуальний DOM

React JS краще працює завдяки Virtual DOM оновлення об’єктної моделі документа або DOM часто стає причиною зниження продуктивності у разі розробки веб-додатків. У React JS можна легко уникнути цієї проблеми, оскільки можна використати віртуальний DOM. React JS дозволяє створювати віртуальний DOM і розміщувати його в пам’яті. Перевага такої

дії полягає в тому, що щоразу, коли відбувається будь-яка зміна фактичної DOM, віртуальна DOM змінюється миттєво, оскільки вона знаходиться в пам’яті (рисунок 2.3).

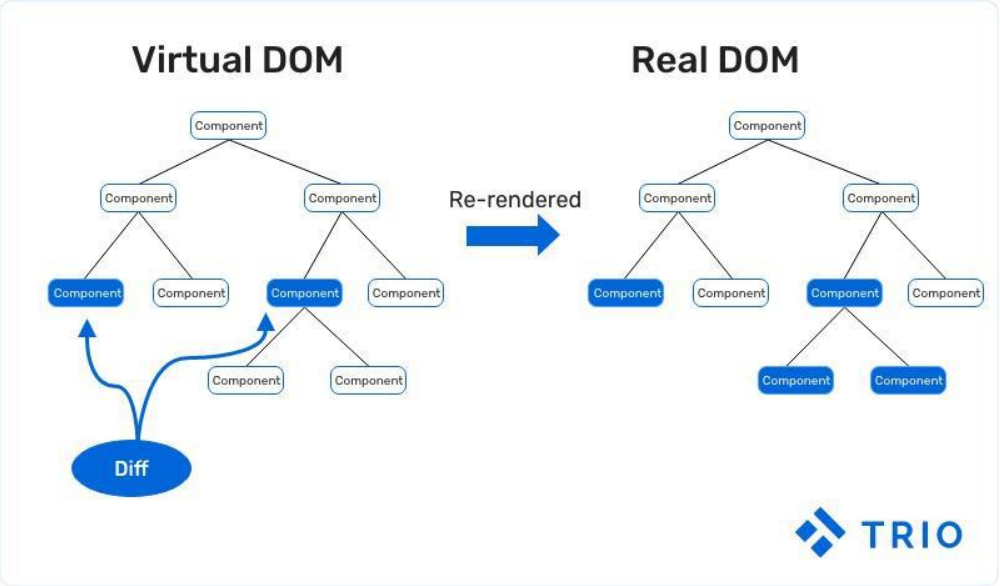


Рисунок 2.3 — Відмінність віртуального DOM та реального DOM

Таким чином, оновлення DOM не виконуються на регулярній основі, і швидкість роботи програми жодним чином не загрожує DOM.

2.1.5 SEO оптимізація

Технологія чудово підходить для SEO. Найчастіше розробникам доводиться стикатися з проблемою того, що JavaScript-фреймворки зовсім не дружні до пошукових систем. Це призводить до того, що програми, розроблені на JS, у більшості випадків не є оптимізованими для SEO. Але за допомогою React JS можна успішно подолати цю перешкоду. React JS дозволяє створювати інтерфейси користувача, які можна переглядати в різних пошукових системах.

Надає інструменти розробника прямо з коробки. React JS був розроблений з урахуванням потреб інженерів-програмістів. Ось чому React JS постачається з набором інструментів, який приносить багато балів брауні від розробників. React Developer Tools розроблено як розширення для розробників Chrome. Це дозволяє спостерігати за ієрархією реактивних компонентів і перевіряти поточний стан і параметри компонента.

2.1.6 Бібліотеки JavaScript

React пропонує усі переваги бібліотеки JavaScript. Однією з основних причин вибрати React JS для розробки веб- та мобільних додатків є бібліотека JavaScript. Як розробник, ви не обмежені нав’язуванням будь-яких шаблонів, шаблонів або складної архітектури, які можуть стати перешкодою в процесі створення вашої програми. Це дозволяє розробнику створити програму так, як він її уявляєте. Саме тому React JS неухильно набирає популярності серед розробників додатків.

Якщо розробник, який займається односторінковими додатками і необхідно створити швидкі, зручні та чуйні програми, розробка додатків React, безумовно, є гідним вибором. Якщо потрібно розробити масштабні програми з даними, які часто змінюються, React JS може допомогти успішно впоратися з цим завданням. Враховуючи переваги цієї технології, немає причин, чому React JS не буде вітатися розробниками в гонитві за створення сучасних додатків.

2.2 Засіб для програмування блокчейн Ethereum Solidity

На серпень 2021 року у блокчейні Ethereum налічується майже 166 мільйонів унікальних адрес. Розглянемо простий приклад, щоб зрозуміти, що таке адреси в Ethereum (і блокчейне в цілому). Для цього розглянемо уявний сценарій. Певна персона вперше знаходиться у відпустці в Греції та вирішує відправити поштовий лист своєму другові Олександру Олександровичу, зі своїми враженнями та натяком, що він обов'язково має його відвідати. Він відносить лист на пошту, листоноша запитує: «Куди відправити?», на що персона відповідаєте: «Моєму другу, Олександру Олександровичу».

Співробітник за стійкою обов'язково скаже, що — він не знає, хто цей ваш друг і він не знає, де він живе. Оскільки персоні відома адреса Олександра, він напишете її на листівці та передасть її співробітнику поштового відділення. Потім поштова компанія надішле листівку, використовуючи свою кур'єрську мережу та відділення по всьому світу.

Листівка передаватиметься між поштовими відділеннями та перевізниками. Коли лист прибуде до Вінниці, листоноша візьме його і опустить до поштової скриньки Олександра Олександровича — це його домашня адреса.

Історія з листівкою може бути дуже спрощеним прикладом. Адже кожен знає, як надіслати листа та перевірити свою поштову скриньку, щоб подивитися, що йому приніс кур'єр. Адреса Ethereum має схожі характеристики з поштовими адресами. Завдяки використанню криптографії з відкритим ключем.

2.2.1 Унікальність адреси

У реальному світі поштова адреса пов'язана з поштовою скринькою, а вірніше вона пов'язана з її географічним розташуванням. Навіть якщо у світі існує кілька Олександрів Олександровичів, але є тільки один Олександр Олександрович, який проживає за адресою «UA, 121552, Вінниця, вул.Хмельницьке шоссе, д. 17». Поштова адреса — це як етикетка на поштовій скриньці, яка відповідає імені людини та місцю, де знаходиться ця поштова скринька. Ethereum адреса — це останні 20 байт хеш (хеш функція keccak-256) відкритого ключа. Оскільки функції хешування детерміновані, це означає, що для різних вхідних даних буде різний хеш, при цьому для одних і тих самих вхідних даних хеш функція повертатиме завжди однаковий хеш. Тож для унікального закритого ключа створюється унікальний хеш.

2.2.2 Секретні ключі

Ключ поштової скриньки не тільки унікальний, але й особистий та секретний. Унікальність вже було розглянуто, тепер розглянемо поняття «особистий» та «секретний».

Особистий — тільки ви володієте ключем, який відкриває вашу поштову скриньку. Ви зберігаєте ключ у таємниці, прикріпивши його до кільця для ключів разом із іншими ключами.

Секретний (закритий) — ви і тільки ви знаєте, для чого може бути використаний цей фізичний ключ. Якщо я дам вам свій набір ключів, ви не

знатимете, яким саме ключем вам відкрити мою поштову скриньку. Аналогічно в Ethereum (рисунок 2.4), закритий ключ зберігається у вашому гаманці.

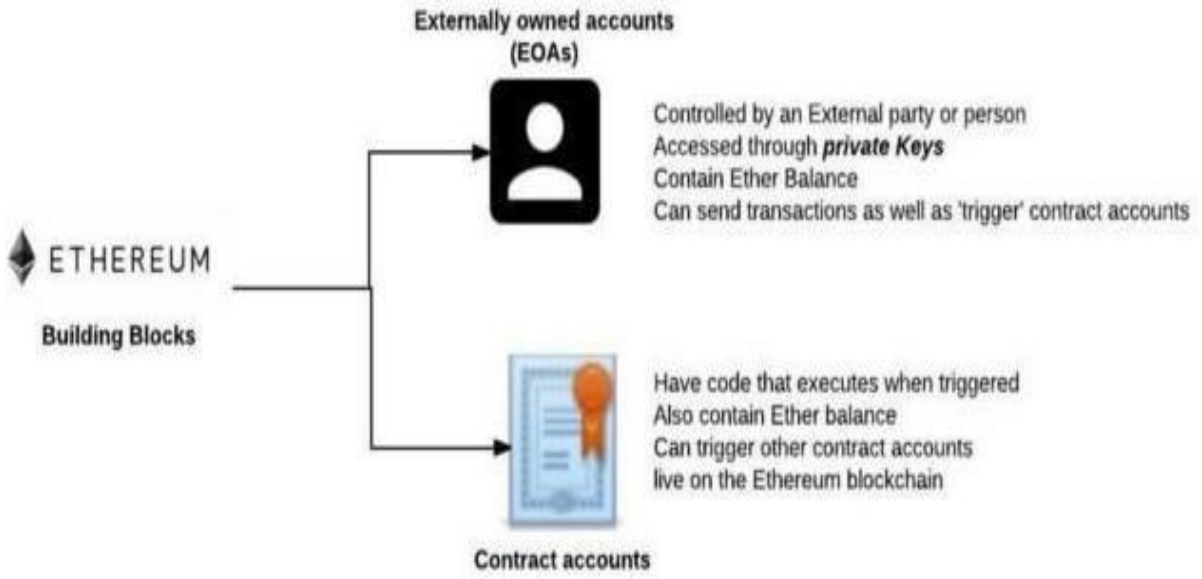


Рисунок 2.4 — Види адрес в Ethereum: Externally owned accounts, contract accounts

Тільки ви повинні знати його та ніколи не передавати його стороннім особам У реальному світі можна відкрити поштову скриньку унікальним фізичним ключем. Поштова скринька має вбудований замок, до якого прив'язаний унікальний секретний ключ для його відкриття. В Ethereum можна використовувати свій обліковий запис з унікальним закритим ключем.

У світі криптографії «особистий» та «закритий» ключ є взаємозамінними термінами. Відкритий ключ є похідним від закритого ключа, тому вони пов'язані між собою. Закриті ключі Ethereum дозволяють відправляти ether шляхом підписання транзакцій. Єдиним винятком є смарт-контракти, як побачимо пізніше. Для доступу до ключів існують різні типи адрес. Ethereum адреса — являє собою адресата повідомлення.

Адреса в платіжній частині транзакції Ethereum — це те саме, що і рахунок одержувача при банківському переказі. External owned accounts (облікові записи, що належать зовнішнім користувачам, EOA), який контролюються закритими ключами.

Закритий ключ дає контроль над ether на рахунку та над процесами аутентифікації, необхідної рахунку при взаємодії зі смарт-контрактами.

Закриті ключі використовуються для створення цифрових підписів, які потрібні для транзакцій із витрачання будь-яких коштів на рахунку. Contract accounts (облікові записи смарт-контрактів, CA) — самоврядні своїм кодом. На відміну від EOA, у смарт-контрактів немає відкритих чи закритих ключів. Смарт-контракти підтримуються не закритим ключем, а властивим кодом. Можна сказати, що вони «володіють собою».

2.3 Технологія створення смарт-контракту в Solidity

Розглянемо детальніше бібліотеки у Solidity.

«Бібліотеки можна розглядати як неявні базові смарт-контракти для смарт-контрактів, які їх використовують» з документації мови Solidity Бібліотека в Solidity — це тип смарт-контракту, що містить код, що багаторазово використовується. Після розгортання в блокчейні (розгортається лише один раз) йому присвоюється певна адреса, а його властивості/методи можуть багаторазово використовуватись іншими смарт-контрактами в мережі Ethereum. Вони дозволяють вести розробку більш модульним методом. Іноді корисно думати про бібліотеку як шматок коду, який можна викликати з будь-якого смарт-контракту без необхідності його повторного розгортання.

2.3.1 Переваги бібліотек Solidity

Однак бібліотеки в Solidity не обмежуються лише однією можливістю повторного використання. Ось деякі інші переваги:

— зручність використання;

— функції бібліотеки можуть використовуватися багатьма смарт-контрактами;

— якщо у вас багато смарт-контрактів, які містять деякий загальний для кожного смарт-контракту код, ви можете винести цей загальний код до бібліотеки;

— використання коду у вигляді бібліотеки заощадить вам gas (комісія за створення власного смарт-контракту), адже витрата gas'a також залежить від розміру смарт-контракту;

— використання базового смарт-контракту замість бібліотеки для поділу загального коду не заощадить gas, тому що Solidity успадкування працює шляхом копіювання коду;

— бібліотеки Solidity можна використовувати для додавання функцій до типів даних.

Наприклад, подумайте про бібліотеки як про стандартні бібліотеки або пакети, які можна використовувати в інших мовах програмування для виконання складних математичних операцій над числами.

2.3.2 Обмеження бібліотек Solidity

У Solidity бібліотеки не мають свого стану і, отже, мають такі обмеження: див. 7.7 Переліки в ДСТУ

— мають сховища (тому що неспроможні мати змінних змінних стану);

— можуть зберігати ether (тому що неспроможні мати функцію відкату);

— не дозволяє використовувати payable функції (оскільки вони не можуть зберігати ether);

— не можуть ні успадковувати, ні бути успадкованими;

— не можуть бути знищеними (немає функції selfdestruct() з версії 0.4.20).

Бібліотеки не призначені для зміни стану смарт-контракту, вони повинні використовуватися тільки для виконання простих операцій на основі вхідних даних та повернення результату.

Однак бібліотеки дозволяють заощадити значну кількість символів (отже, не забруднювати блокчейн кодом, що повторюється), оскільки один і той же код не потрібно розгортати знову і знову. Різні смарт-контракти на Ethereum можуть просто покладатися на ту саму вже розгорнуту бібліотеку.

Той факт, що кілька смарт-контрактів залежать від повторюваного фрагмента коду, може зробити середовище більш безпечним. Уявіть собі, що існує не тільки добре перевірений код для ваших проектів (наприклад заготівлі кодів OpenZeppelin), але й можливість покладатися на вже розгорнутий у блокчейні бібліотечний код, який вже використовують інші смарт-контракти. Це, безумовно, допомогло б ефективно побудувати архітектуру програми.

2.4 Розгортання Solidity

Розгортання бібліотеки трохи відрізняється від розгортання звичайного смарт-контракту. Ось два сценарії. Вбудована бібліотека — якщо смарт-контракт використовує бібліотеку, яка має лише внутрішні функції, то EVM вбудовує бібліотеку в смарт-контракт. Замість використання delegatecall для виклику функції, він просто використовує оператор JUMP (звичайний виклик методу). Цей сценарій не потребує окремого розгортання бібліотеки. Пов'язана бібліотека — з іншого боку, так як бібліотека містить публічні (public) або зовнішні (external) функції, її необхідно розгорнути як окремий смарт-контракт. При розгортанні бібліотеки генерується унікальна адреса в блокчейні. Ця адреса повинна бути пов'язана з смарт-контрактом.

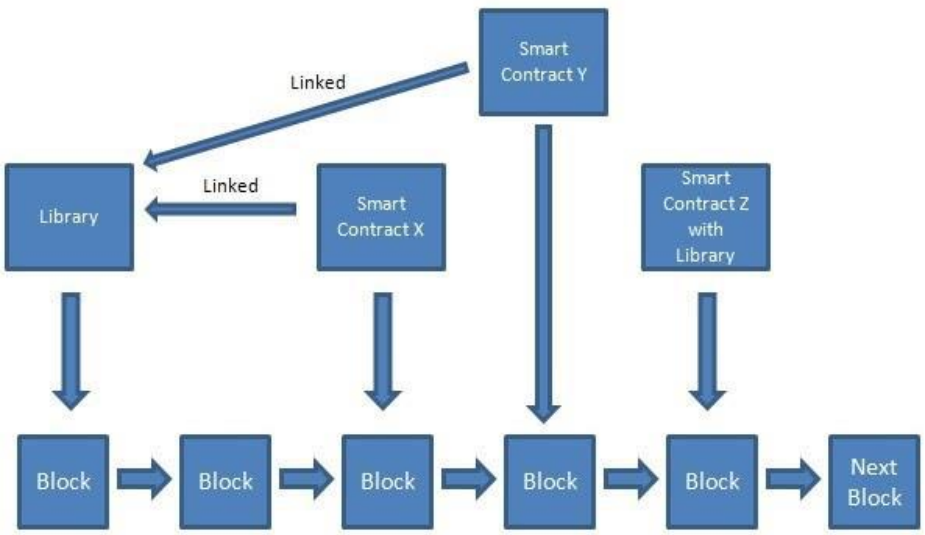


Рисунок 2.5 — Варіанти розгортання бібліотек Solidity

Для того, щоб програма була більш ергономічною та зручною для використання, використаємо метод пов'язаної бібліотеки. Такми чином, отримавши усі доступні функції для створення власного смарт-контракту та заощадивши при цьому час на написанні стандартних функцій обробки методів проведення транзакцій.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБОБУ УПРАВЛІННЯ ДЕЦЕНТРАЛІЗОВАНИМИ ФІНАНСАМИ

Управління децентралізованими фінансами включає в себе ряд операцій, таких як здійснення транзакцій, виведення та внесення коштів, зарахування коштів на депозит, нарахування винагород корисувачам системи, та здійснюється за ряд послідовних дій. У даному розділі магістерської роботи виконується розробка програмних засобів для управління децентралізованими фінансами та здійснення усіх вищеперерахованих функцій.

3.1 Постановка задачі

Програмною реалізацією є найпростіший блокчейн, який реалізує функції обчислення хеша попереднього блоку, створення нового блоку та додавання його в ланцюжок блоків, а також перевірку цілісності ланцюжка блоків. Звичайно, основна складність створення блоку в блокчейн системі це реалізація алгоритму proof-of-work. Однак на даний момент ця задача є важкою, але принаймні буде показана загальна структура побудови ланцюжка блоків.

Ідея полягає в тому, щоб створити послідовність транзакцій, блок кожної з яких міститиме ім'я відправника, суму переказу, ім'я одержувача та хеш попереднього блоку. Спочатку створюється перший блок у ланцюжку, так званий генезис блок (genesis block). Після цього за ним записуються інші блоки, причому перед додаванням кожного з них обчислюється хеш попереднього блоку.

Також реалізовано процедуру перевірки цілісності блоків. Суть її полягає в наступному: кожному за блоків, починаючи з другого, визначається попередній блок, попереднього блоку обчислюється хеш і порівнюється з полем «hash» поточного блоку. Якщо вони збігаються, блок успішно проходить перевірку на цілісність. Алгоритми хешування мають різну надійність. Багато в чому ступінь довіри до блокчейну визначається вибором

алгоритму хешування. У наведеній програмній реалізації використано алгоритм SHA-1. Існують інші, не менш популярні алгоритми, наприклад, SHA-256, SHA-2 (використовується в Bitcoin), MD5, які можуть використовуватися для аналогічних цілей. Однак високонадійні алгоритми витрачають на роботу куди більший час, а в даному прикладі їх використання немає сенсу, тому і був обраний алгоритм SHA-1.

3.2 Створення токена ERC-20

При реалізації технології блокчейн можна вибрати створення монети або токена. Монета має власний блокчейн, тоді як токен побудований на вже існуючій мережі. Криптовалюти покладаються на блокчейни через свою безпеку та децентралізований характер.

Створення токена вимагає менше зусиль та витрат, ніж створення криптовалют. Для розробки монети потрібна команда розробників та експертів. Токен все ще потребує технічних знань, але його можна створити, не витрачаючи роки на розробку, беручи за основу технологію інших блокчейнів, таких як Ethereum, Binance Smart Chain, Solana і Polygon.

Вибір жетона або монети буде змінюватися залежно від вибору галаштувань та корисності, яка необхідна проекту. Загалом, витрати залежать від необхідної роботи, як-от зовнішніх розробників та часу.

Ethereum і Binance Smart Chain — популярні блокчейни для створення цифрових валют. Можна використовувати встановлений код для створення токенів самостійно або заплатити за використання служби створення монет. Бічні ланцюги є ще одним популярним вибором, оскільки вони забезпечують більше налаштування з основними перевагами блокчейну.

Перш ніж створити власний токен, потрібно розглянути його корисність, токеноміку та юридичний статус. Після — вибір блокчейну, механізму консенсусу та архітектури необхідні для етапу розробки. Далі необхідно розглянути аудит проекту та остаточну юридичну перевірку. Для створення власного проекту, було обрано створення токену на основі Ethereum за допомогою використання смарт-контрактів E20.

Повна назва токена ERC — «Ethereum Request for Comments». Число 20 це випадковий ідентифікаційний номер пропозиції. Токени ERC-20 найбільш популярні в мережі Ефіріум. Вони використовуються для оплати різних функцій, їх називають службові токени. Ще вони використовуються для оплати товарів та послуг різних мереж.

Для створення токену ERC-20 використовуємо середовище розробки — Remix для написання Смарт-Контракту для нашого токену ERC-20. Remix — це онлайн-інструмент, який дозволяє написати смарт контракт мовою Solidity, почнемо його створення (рисунок 3.1).

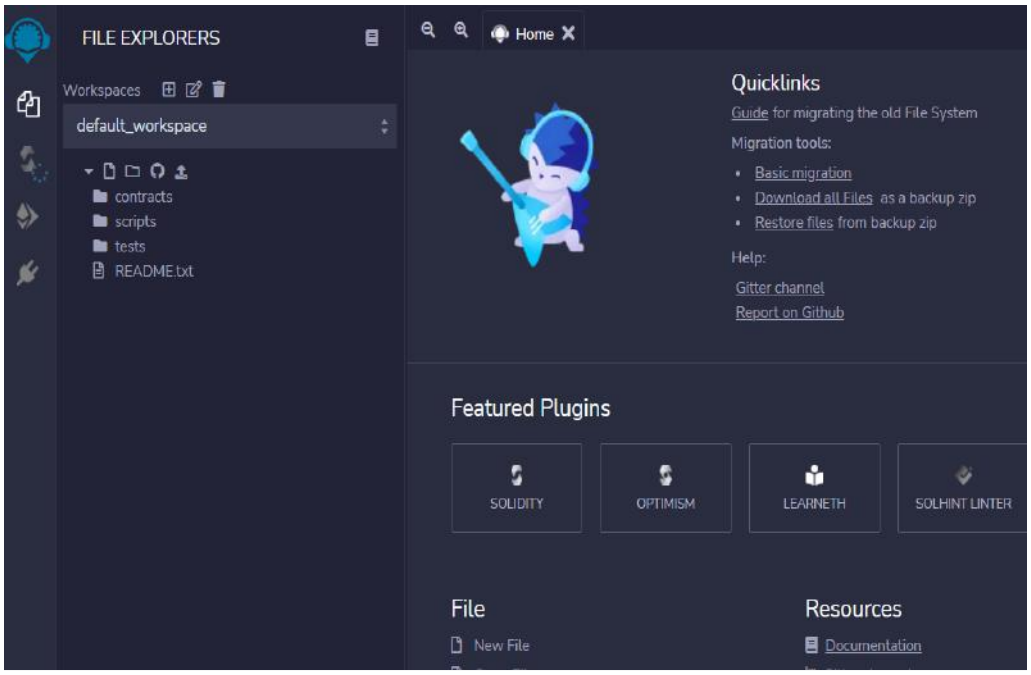


Рисунок 3.1 — Варіанти розгортання бібліотек Solidity

Перейдемо до налаштувань токену. При налаштуванні токену існує стандарт, якого необхідно дотримуватись. Стандарт ERC20 складається з 3 обов'язкових правил і 6 обов'язкових правил. Необхідно включити в токен смарт-контракту певні функції, що відповідає вимогам ERC20. Якщо не включили обов’язкові функції, неможливо буде запустити смарт-контракт.

Обов'язкові правила наступні:

— total supply. відповідає за загальну емісію криптовалют, також унеможливлює генерацію нових токенів, якщо досягнуто ліміту;

— balance0f. виконує визначення початкового числа токенів, що приписуються конкретній адресі. найчастіше це адреса, яка належить творцям ico, та протоколом описані два способи переміщення цифрових валют, з яких токени розподіляються між учасниками системи, і навіть здійснюються фінансові операції;

— transfer — дозволяє передавати токени людині, яка вклала гроші в проект під час ico;

— transfer form — відповідає проведення операцій між учасниками системи;

— approve. призначений для перевірки smart-контракту — за допомогою цієї функції перевіряється, чи контракт може виконувати дистрибуцію токенів;

— allowance — використовується для перевірки балансу відправника цифрових валютних одиниць.

Створюємо головну функцію, в якій будуть прописані усі основні правила та налашування (рисунок 3.2). Не обов'язкові правила:

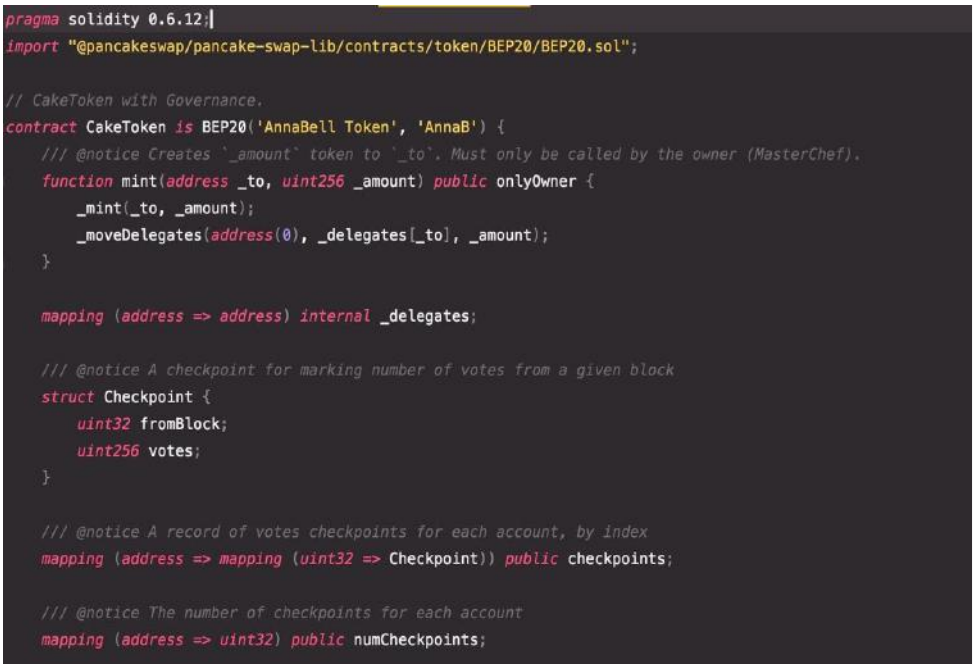
— name — повна назва токена;

— decimal — кількість знаків після коми, так як потрібно вказувати кількість знаків, якщо хочете мати можливість зараховувати нецілу кількість токенів;

— symbol — позначення токена для бірж, на яких він використовуватиметься.

Набір перерахованих функцій дає можливість біржам та операторам сховищ валюти формувати загальну кодову базу, яка взаємодіє з будь-яким Smart-контрактом, заснованим на ERC20.

Для того, щоб захистити проект від шахрайства, реалізуємо функцію передачі прав власності токена іншому контракту.



```
pragma solidity 0.6.12;
import "@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol";

// CakeToken with Governance.
contract CakeToken is BEP20('AnnaBell Token', 'AnnaB') {
    /// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
    function mint(address _to, uint256 _amount) public onlyOwner {
        _mint(_to, _amount);
        _moveDelegates(address(0), _delegates[_to], _amount);
    }

    mapping (address => address) internal _delegates;

    /// @notice A checkpoint for marking number of votes from a given block
    struct Checkpoint {
        uint32 fromBlock;
        uint256 votes;
    }

    /// @notice A record of votes checkpoints for each account, by index
    mapping (address => mapping (uint32 => Checkpoint)) public checkpoints;

    /// @notice The number of checkpoints for each account
    mapping (address => uint32) public numCheckpoints;
```

Рисунок 3.2 — Основні налаштування токену

Дана функція реалізує основну функцію захисту, а саме – унеможливлює створення токенів власноруч для продажу. Також це не дасть підірвати цінність володіння токенами за рахунок різкого збільшення кількості токенів при тих же інвестиціях зовні (рисунок 3.3).

```
function _delegate(address delegator, address delegatee)
    internal
{
    address currentDelegate = _delegates[delegator];
    uint256 delegatorBalance = balanceOf(delegator); // balance of underlying CAKEs (not scaled);
    _delegates[delegator] = delegatee;

    emit DelegateChanged(delegator, currentDelegate, delegatee);

    _moveDelegates(currentDelegate, delegatee, delegatorBalance);
}

function _moveDelegates(address srcRep, address dstRep, uint256 amount) internal {
    if (srcRep != dstRep && amount > 0) {
        if (srcRep != address(0)) {
            // decrease old representative
            uint32 srcRepNum = numCheckpoints[srcRep];
            uint256 srcRepOld = srcRepNum > 0 ? checkpoints[srcRep][srcRepNum - 1].votes : 0;
            uint256 srcRepNew = srcRepOld.sub(amount);
            _writeCheckpoint(srcRep, srcRepNum, srcRepOld, srcRepNew);
        }

        if (dstRep != address(0)) {
            // increase new representative
            uint32 dstRepNum = numCheckpoints[dstRep];
            uint256 dstRepOld = dstRepNum > 0 ? checkpoints[dstRep][dstRepNum - 1].votes : 0;
            uint256 dstRepNew = dstRepOld.add(amount);
            _writeCheckpoint(dstRep, dstRepNum, dstRepOld, dstRepNew);
        }
```

Рисунок 3.3 — Основні налаштування токену

Як вхідні параметри функції передаються адреса контракту, яким буде здійснюватись перестраховка. В тілі функції здійснюється передача прав перевірки загальної кількості токенів та типу контракту, за допомогою якого використовується реалізація смарт-контранту.

Будучи незалежною стороною, потрібно прописати усі права та властивості, які незалежний смарт-контракт може здійснювати із токенами проекту. Відкриваємо NewToken.sol, ставимо замість змінних у <дужках> потрібні значення:

— totalSupply — загальна кількість токенів;

— adress — адреса контракту, якому будуть передаватись права для перевірки;

— name — повна назва контракту для перевірки, в нашому випадку — IBEP 20;

— symbol — символ для бірж.

Тепер ці дані нам необхідно внести до смарт-контракту, доступного за посиланням. Завантажуємо обидва .sol файли. Переходимо до редактора remix (рисунок 3.4).

```
contract MasterChef is Ownable {
    using SafeMath for uint256;
    using SafeBEP20 for IBEP20;

    // Info of each user.
    struct UserInfo {
        uint256 amount;     // How many LP tokens the user has provided.
        uint256 rewardDebt; // Reward debt. See explanation below.
    }

    // Info of each pool.
    struct PoolInfo {
        IBEP20 lpToken;           // Address of LP token contract.
        uint256 allocPoint;       // How many allocation points assigned to this pool. CAKEs to distribute per block.
        uint256 lastRewardBlock;  // Last block number that CAKEs distribution occurs.
        uint256 accCakePerShare; // Accumulated CAKEs per share, times 1e12. See below.
    }

    // The CAKE TOKEN!
    CakeToken public cake;
    // The SYRUP TOKEN!
    SyrupBar public syrup;
    // Dev address.
    address public devaddr;
    // CAKE tokens created per block.
    uint256 public cakePerBlock;
    // Bonus muliplier for early cake makers.
    uint256 public BONUS_MULTIPLIER = 1;
```

Рисунок 3.4 — Налаштування контракту для перевірки

Розглянемо реалізацію смарт-контракту за допомогою блокчейну Ethereum. Смарт-контракти Bitcoin обмежені у можливостях, у той час як Ethereum був спроектований з урахуванням потреб розробників програмного забезпечення і активно застосовується для розподіленої роботи програм на основі блокчейн технології, зокрема віртуальної машини Turing Complete. Програмний код контракту буде написаний спеціальною мовою для створення смарт-контрактів Solidity, що багато в чому схоже на JavaScript. Транзакцію до Ethereum можна реалізувати кількома способами. У Ethereum можна робити три речі:

— перекласти ETH іншому користувачеві;

— створити смарт-контракт і записати його в блокчейн;

— виконати смарт-контракт.

Смарт-контракт — це лише код, який можна виконати, здійснивши транзакцію на його адресу. Коли здійснюється передача ефіру, інформація про транзакцію записується в блокчейн Ethereum корисувачем системи. Коли розробник додає або виконує код смарт-контракту, одразу ж до системи записується новий блок. Після збереження блоку, автоматично виконується код програми.

3.3 Розрахунок комісії

За кожну операцію треба сплатити комісію. Ця комісія йде у нагороду корисувачам системи, чиї комп'ютери займаються додаванням блоків та виконанням коду смарт-контрактів. Одиниця винагороди в Ethereum називається gas. Що таке gas? Gas (газ або бензин) — це одиниця сплати комісії в Ethereum. Наприклад, переведення з гаманця на гаманець коштує 21000 gas. Вважається ціна газу в gwei — ефіро-копійках. Gwei = 0.000000001 ETH. Якщо з ціною переказ коштів все зрозуміло, то вартість запису або виконання смарт-контракту залежить від його складності — чим більше операцій, то більше газу потрібно для його виконання. В Ethereum розробник сам призначаєте розмір комісії за операцію

Коли ж ставиться на чергу якась із транзакцій, необхідно вказати:

— адресу одержувача;

— суму eth для перекладу (можливо 0) максимум газу, який може бути витрачений на виконання операції;

— ціну за газ у величині gwei;

— офіційний гаманець ethereum (прописати скільки ефіру відправити, куди і ціну за газ).

Комісія обчислюється з кількості газу та його вартості у gwei. В інтерфейсі гаманця зазвичай є повзунок, через який можна встановити ціну за газ у діапазоні від 1 до 60 Gwei. Чим дорожче встановлено ціну газу — тим швидше відбудеться транзакція.

Розглянемо приклад розрахування комісії за використання gas. Переказ ETH коштує 21000 gas. Якщо розробник встановить ціну 1 gas = 40 Gwei то користувач заплатить комісію за переказ 0,00084 ETH.

3.4 Створення Smart-контракту.

У даному розділі описано деталі створення ключового контракту програми, а саме контракту, який відповідає за винагороду. Всі користувачі які володіють нашими токенами, можуть внести їх до контракту, та отримати постійну винагороду у виді створеного токену (рисунок 3.5).

```
contract MasterChef is Ownable {
    using SafeMath for uint256;
    using SafeBEP20 for IBEP20;

    // Info of each user.
    struct UserInfo {
        uint256 amount;     // How many LP tokens the user has provided.
        uint256 rewardDebt; // Reward debt. See explanation below.
    }

    // Info of each pool.
    struct PoolInfo {
        IBEP20 lpToken;           // Address of LP token contract.
        uint256 allocPoint;       // How many allocation points assigned to this pool. CAKEs to distribute per
        uint256 lastRewardBlock;  // Last block number that CAKEs distribution occurs.
        uint256 accCakePerShare; // Accumulated CAKEs per share, times 1e12. See below.
    }

    // The CAKE TOKEN!
    CakeToken public cake;
    // The SYRUP TOKEN!
    SyrupBar public syrup;
    // Dev address.
    address public devaddr;
    // CAKE tokens created per block.
```

Рисунок 3.5 — Опис контракту винагород

Для винагороди використовується токен, створений у розділі 3.1. Створимо функцію, яка буде відповідати за внесення токенів до смарт-контракту.

```
function enterStaking(uint256 _amount) public {
    PoolInfo storage pool = poolInfo[0];
    UserInfo storage user = userInfo[0][msg.sender];
    updatePool(0);
    if (user.amount > 0) {
        uint256 pending = user.amount.mul(pool.accCakePerShare).div(1e12).sub(user.rewardDebt);
        if(pending > 0) {
            safeCakeTransfer(msg.sender, pending);
        }
    }
    if(_amount > 0) {
        pool.lpToken.safeTransferFrom(address(msg.sender), address(this), _amount);
        user.amount = user.amount.add(_amount);
    }
    user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12);

    syrup.mint(msg.sender, _amount);
    emit Deposit(msg.sender, 0, _amount);
}
```

Рисунок 3.6 — Опис функції внесення

На рисунку 3.6 зображено текст функції депозиту, яка отримує на вхід число токенів, які мають бути внесені до смарт-контракту. Коли користувач виконує дану транзакцію до блокчейну, окрім внесення токенів, функція також врахує кількість попередніх винагород. Після проведення транзакції токени поступлять на рахунок користувача. Усе це буде виконано в рамках однієї транзакції до блокчейну. З кожним оброблений блоком в блокчейні, смарт-контракт автоматично додає винагороду користувачам, які внесли свої токени до смарт-контракту.

Щоб отримати внесені токени назад, користувач має викликати функцію leaveStaking(). Функція потребує на вхід число токенів, які користувач планує перенести зі смарт-контракту до персонального гаманц (рисунок 3.7). Окрім часу перенесення, контракт також вираховує повну кількість токенів, які користувач має отримати. Після виклику функції вони

також будуть перенесені до персонального гаманцю в рамках однієї транзакції.

```
function leaveStaking(uint256 _amount) public {
    PoolInfo storage pool = poolInfo[0];
    UserInfo storage user = userInfo[0][msg.sender];
    require(user.amount >= _amount, "withdraw: not good");
    updatePool(0);
    uint256 pending = user.amount.mul(pool.accCakePerShare).div(1e12).sub(user.rewardDebt);
    if(pending > 0) {
        safeCakeTransfer(msg.sender, pending);
    }
    if(_amount > 0) {
        user.amount = user.amount.sub(_amount);
        pool.lpToken.safeTransfer(address(msg.sender), _amount);
    }
    user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12);

    syrup.burn(msg.sender, _amount);
    emit Withdraw(msg.sender, 0, _amount);
}
```

Рисунок.3.7 — Повернення токенів зі смарт-контракту

Після створення контракту,необхідно змінити права власника токену, на адрес контракту. Дана процедура необхідна для правильної роботи контракту винагород. Також це збільший безпечність токену від неправомірних дій, які можуть вплинути на його економічну цінність.

Після розробки графічного інтерфейсу програма матиме вигляд, зображений на рисунку 3.8.

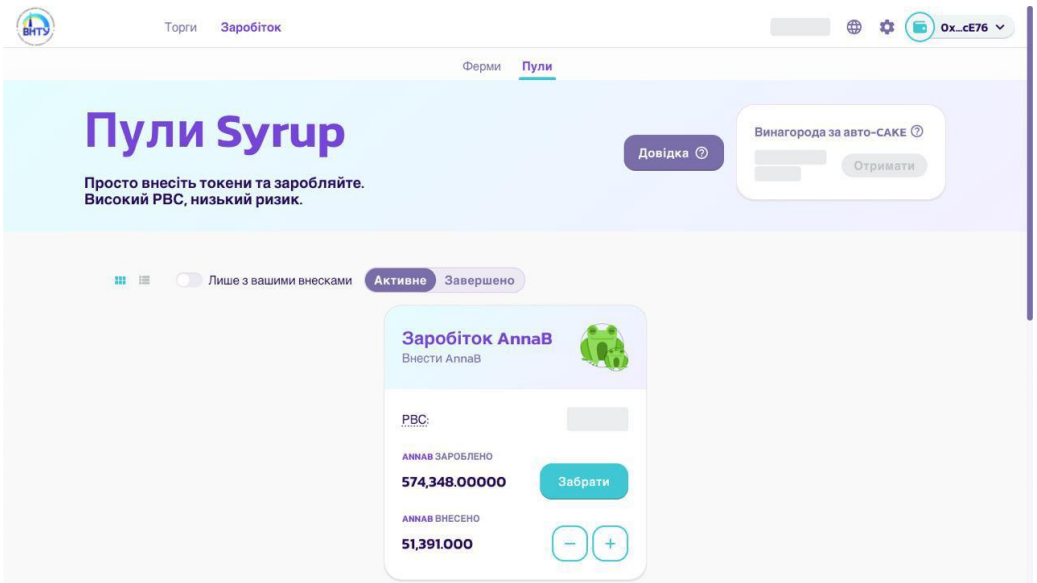


Рисунок 3.8 — Інтерфейс корисувача програми

У даному розділі було запропоновано новий спосіб створення смарт-контракту, який доповнює існуючі програміні рішення на основі блокчейну, удосконалюючи технологію та роблячи його використання більш універсальним. Оскільки деякі користувачі та власники бізнесу хотіли б отримати переваги блокчейну, але технологія є не досить поширеною, було створено програму, яка заохочує користувачів до її використання (див. додаток Г, Д). Програма буде корисною для випадків, коли користувачі та власники бізнесу потребують як публічної звітності, так і миттєвого опрацювання транзакцій.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ДОЦІЛЬНОСТІ СТВОРЕННЯ ПРОГРАМНОГО ЗАСОБУ УПРАВЛІННЯ ДЕЦЕНТРАЛІЗОВАНИМИ ФІНАНСАМИ

Дослідження завжди дорогі. Ці витрати на виробництво та реалізацію товарів необхідно постійно зменшувати, оскільки це прогрес будь-якого виробництва. На основі економічних розрахунків [49] можна продемонструвати рентабельність та ефективність впровадження результатів досліджень у виробництво, тобто комерціалізація наукових досліджень. Дана магістерська кваліфікаційна робота відноситься до розряду прикладної науково-технічної роботи. Прогнозується виведення науково-технічної розробки на ринок із залученням потенційним інвестора. Дану послідовність приведено на рисунку 4.1.

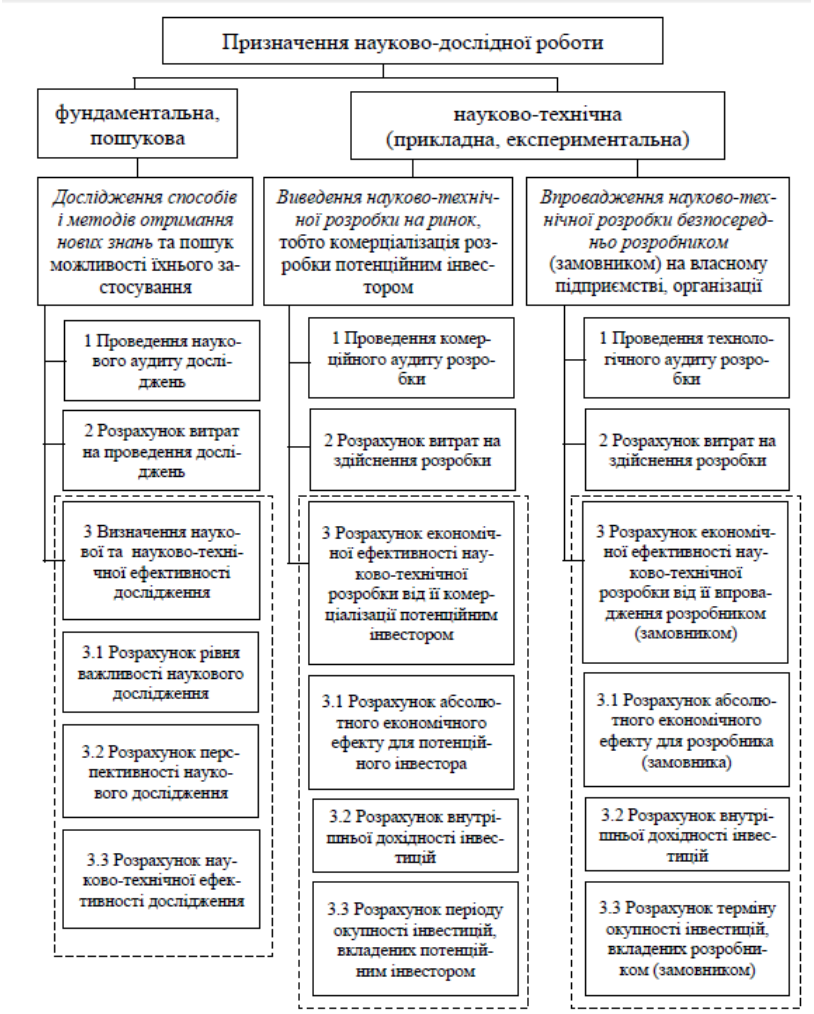


Рисунок 4.1 — Складові економічної частини магістерської кваліфікаційної роботи

Економічна частина цієї магістерської роботи буде поділена на такі елементи. Усі подальші економічні розрахунки будуть розглянуті у згаданих розділах економічної частини.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою оцінки потенціалу комерційного розвитку є оцінка потенціалу комерційного розвитку, що випливає з науково-технічних досліджень. За результатами оцінки робляться висновки про напрямки (особливості) організації в майбутньому її впровадження з урахуванням встановленої оцінки. Комерційний потенціал інвестицій буде оцінюватись відповідно до дванадцяти критеріїв, наведених у таблиці 4.1.

Таблиця 4.1 — Оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Багато аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
Практична здійсненність					
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні.	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси.Джерела фінансування є	Потрібні незначніфінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років.Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років.Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років.Термін окупності інвестицій менше 3-х років

Закінчення таблиці 4.1

12	Необхідно регламентні документи та велика кількість дозвільних документів на виробництво продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

На основі таблиці різні експерти, у нашому випадку викладачі кафедри ОТ визначають різні результати. Результати цієї оцінки комерційного потенціалу узагальнено у таблиці 4.2.

Таблиця 4.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	1. Семеренко В. П., к.т.н., доц. кафедри ОТ	2. Крупельницький Л.В., к.т.н., доц. кафедри ОТ	3. Черняк О.І., к.т.н., доц. кафедри ОТ
	Бали:		
1.Технічна здійсненність концепції	3	3	4
2.Ринкові переваги (наявність аналогів)	3	2	4
3. Ринкові переваги (ціна продукту)	4	2	3
4.Ринкові переваги (технічні властивості)	3	2	2
5. Ринкові переваги (експлуатаційні витрати)	3	4	3
6. Ринкові перспективи (розмір ринку)	2	3	3
7. Ринкові перспективи (конкуренція)	4	3	3
8. Практична здійсненність (наявність фахівців)	3	4	4
9. Практична здійсненність (наявність фінансів)	2	4	2

Закінчення таблиці 4.2

10.Практична здійсненність (необхідність нових матеріалів)	2	3	4
11. Практична здійсненність (термін реалізації)	4	3	3
12. Практична здійсненність (розробка документів)	2	3	3
Сума балів	$СБ_1 = 37$	$СБ_2 = 36$	$СБ_3 = 38$
Середньо-арифметична сума балів $СБ_С$	$СБ_С = \frac{\sum_1^3 СБ_i}{3} = \frac{37 + 36 + 38}{3} = 37$		

Відповідно до таблиці 4.2, а також відповідно до рекомендацій, наведених у таблиці 4.3, можна зробити висновок про рівень потенціалу комерційного розвитку [50].

Таблиця 4.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 — 10	Низький
11 — 20	Нижче середнього
21 — 30	Середній
31 — 40	Вище середнього
41 — 48	Високий

З урахуванням середніх арифметичних балів $СБ_С$ = 37, які були визначені експертами, можна зробити висновок, що рівень комерційного потенціалу цієї розробки буде вище середнього.

Програмний ресурс MOONLIFT було використано для порівняння властивостей. Це програмне забезпечення працює більш швидко, але не використовує заохочення для користувачів.

Нова розробка, навпаки, є вузькоспрямованою і тому має більшу швидкість. Окрім того, розроблюваний інтерфейс не перевантажений зайвою інформацією, що значно полегшує роботу. Розповсюдження аналога

відбувається лише в рамках передплати, тоді як плата за використання програми, яка є продуктом розробки, є одноразовою.

Також, ще один аналог розробленого програмного забезпечення — MoonLift, який призначений для обміну ексклюзивних цифрових валют, але даний сервіс є не надто пишириним та популярним. Якщо ж здійснити порівняння із створеним програмним продуктом, то MoonLift має певні переваги — більш розширений функціонал, та ексклюзивні канали для обміну цифрових валют, що робить роботу додатку більш швидкою.

Ну і, звісно ж, є певний недолік — в базовій, безкоштовній версії аналогі в значній мірі обмежений функціонал, в той час, як в даній розробці магістерської кваліфікаційної роботи він безкоштовний та доступний у вільному доступі. Порівння розробки з її аналогами приведено в таблиці 4.4.

Таблиця 4.4 — Порівняння характеристик розробки із аналогом

Показники	Розробка	Аналог1	Аналог2
Функціонал	9	8	9
Швидкодія	9	9	7
Надійність	8	9	8
Метод розповсюдження	8	7	6
Інтерфейс, простота використання	7	6	7

Продукт буде просуватися за допомогою реклами в соціальних мережах, пошукових системах та багатьох інших джерелах Інтернету. Використовуючи аналітику цих сервісів, можна буде націлити рекламу на цільову групу захисників інформації.

Продукт також може бути використаний в інформаційній безпеці для розмежування доступу, банківської справи, соціальних мереж, криміналістики, комп’ютерних ігор та інших сфер.

Новизна дослідження полягає в тому, що буде створено новий тип смарт-контракту, який дозволить користувачам отримувати винагороду за користування сервісом.

Виходячи з результатів цього порівняння, можна з упевненістю сказати, що новий дизайн є конкурентоспроможним, оскільки в деяких аспектах в ньому переважає один з найкращих аналогів на ринку. Даний рівень було досягнуто за рахунок покращення та/або розширення функціональних можливостей нової науково-технічної розробки порівняно з аналогічними розробками, існуючими в цей час на ринку.

4.3 Розрахунок витрат на здійснення науково-дослідної роботи

У магістерській роботі розглядається програмне забезпечення для управління децентралізованими фінансами за допомогою технології Blockchain, тому значну частину витрат складають витрати на розробку, а не на виробництво та відтворення [50]. Відповідно, є певна специфіка розрахунків.

4.3.1 Витрати на оплату праці

Основна заробітна плата розробників, що працюють над проектом, визначена у формулі 4.1:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де k — кількість посад дослідників, залучених до процесу досліджень;

$М_{ni}$ — місячний посадовий оклад конкретного дослідника, грн;

$Т_р$ — середня кількість робочих днів в місяці, $Т_р$ = 21…23 дні; обрано 22 дні;

t_i— кількість днів роботи конкретного дослідника, дн.

Над створенням розробки працював менеджер проекту та інженер програмного забезпечення, тому ми виконаємо для данх працівників усі необхідні розрахунки, та після чого вносимо їх до таблиці 4.5:

$$З_{о.к.} = \frac{19250 \cdot 8}{22} = 7000(грн),$$

$$З_{о.в.} = \frac{12100 \cdot 44}{22} = 24200\ (грн).$$

Таблиця 4.5 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	19250	875	8	7000
Старший інженер-програміст	12100	550	44	24000
Всього				31000

Витрати на основну заробітну плату робітників за відповідними найменуваннями робіт відсутні, тобто $З_р = 0$. Додаткова винагорода ($З_{дод.}$) усіх розробників та працівників, які брали участь у цьому етапі роботи, обчислюється як 10 ... 12% від суми основної заробітної плати дослідників та робітників за формулою 4.2:

$$З_{дод.} = (З_о + З_р) \cdot \frac{Н_{дод.}}{100\%}, \qquad (4.2)$$

де $Н_{дод.}$ — норма нарахування додаткової заробітної плати.

$$З_{дод.к.} = \frac{10 \cdot 7000}{100} = 700\ (грн),$$

$$З_{дод.в.} = \frac{10 \cdot 24000}{100} = 2400\ (грн),$$

$$З_{дод} = З_{дод.к.} + З_{дод.в.} = 3100 \text{ (грн)}.$$

4.3.2 Відрахування на соціальні заходи

Заробітна плата робітників відсутня, тому $З_р = 0$. Нарахування на заробітну плату дослідників та нарахування на заробітну плату працівників, які брали участь у цьому етапі роботи, розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за наступною формулою 4.3:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (4.3)$$

де $Н_{зп}$ — норма нарахування на заробітну плату.

$$З_н = (31000 + 0 + 1300) \cdot \frac{22\%}{100\%} = 7106 \text{ (грн.)}$$

4.3.3 Сировина та матеріали

Витрати на матеріали (М), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за наступною формулою 4.4:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.4)$$

де $Н_j$ — кількість матеріалу j-го виду, шт.;

n — кількість видів матеріалу.

$Ц_j$ — ціна матеріалу j-го виду, грн;

$К_j$ — коефіцієнт транспортних витрат, $К_j$ = (1,1…1,15), обираємо $К_j$ 1,15;

$В_j$ — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів j-го найменування, грн/кг.;

Результати розрахунків занесено до таблиці 4.6.

Таблиця 4.6 — Витрати на матеріали

Найменування комплектуючих	Ціна за 1 шт., грн	Кількість матеріалу, шт..	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Ліцензія для середовища програмуваня	674,00	1	0	0	775,1
Дизайн пректу	800,00	1	0	0	920
Карта пам’яті	650,00	1	0	0,00	747,50
Всього (з урахуванням транспортних витрат)					2442,6

4.3.4 Розрахунок витрат на комплектуючі

Оскільки кінцевий продукт, який ми створюємо — це програмний інструмент, це не спричиняє жодних витрат на компоненти та $К_в = 0$.

4.3.5 Спецустаткування для наукових (експериментальних) робіт

Спецустаткування для проведення експериментальних робіт по створенню програмного продукту по розпізнаванню особи за зображенням її обличчя не має потреби залучати.

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

Програмне забезпечення для створення програмного продукту по розпізнаванню особи за зображенням її обличчя використовується таке, що є у вільному розповсюдженні, тому витрати на придбання такого забезпечення відсутні. Це мова програмування ReactJS та програмні продукти із бібліотек із відкритим кодом Solidity.

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути

розраховані з використанням прямолінійного методу амортизації за формулою 4.5:

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12,} \quad (4.5)$$

Таблиця 4.7 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців.	Амортизаційні відрахування, грн
ЕОМ	13000	2	2	1083,33
Приміщення	150000	20	2	1250
Всього				2333,33

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($В_е$) розраховують за формулою 4.6:

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (4.6)$$

де W_{yi} — встановлена потужність обладнання на певному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), $Ц_е = 4,62$ [51];

$К_{впі}$ — коефіцієнт, що враховує використання потужності, $К_{впі}$ <1; обираємо $К_{впі}$ = 0,7;

η_i — коефіцієнт корисної дії обладнання, η_i<1; обираємо η_i= 0,73.

$$В_е = \sum_{i=1}^{1} \frac{0,12 \cdot 352 \cdot 4,62 \cdot 0,7}{0,8} = 178,07 \text{ (грн.)}$$

Проведені розрахунки необхідно звести до таблиці 4.8.

Таблиця 4.8 — Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
ЕОМ	0,12	352	178,07
Всього			178,07

4.3.9 Службові відрядження

Під час розробки програмного забезпечення відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, магістрів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень, не плануються.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» не плануються, так як у цьому не має потреби.

4.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 4.7:

$$I_в = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$I_{в.к.} = 7000 \cdot \frac{50\%}{100\%} = 3500\ (\text{грн.}),$$

$$I_{в.в.} = 24000 \cdot \frac{50\%}{100\%} = 12000 \text{ (грн.)},$$

$$I_{в} = I_{в.к.} + I_{в.в.} = 15500 \text{ (грн.)}.$$

4.3.12 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою 4.8:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Беремо норму нарахування 100%.

$$В_{нзв.в.} = 7000 \cdot \frac{100\%}{100\%} = 7000 \text{ (грн.)},$$

$$В_{нзв.в.} = 24000 \cdot \frac{100\%}{100\%} = 24000 \text{ (грн.)},$$

$$В_{нзв} = В_{нзв.к.} + В_{нзв.в.} = 28000 \text{ (грн.)}.$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою 4.9:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.9)$$

У нашому випадку:

$З_р = 0, К_в = 0, В_{спец} = 0, В_{прг} = 0, В_{св} = 0, В_{сп} = 0,$ тому отримаємо:

$$В_{заг} = 31000 + 3100 + 7106 + 2442{,}6 + 2333{,}33 + 178{,}07 + 15500 + 310 = 61\,970 \text{ (грн)}.$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою 4.10:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.10)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи; обираємо $\eta = 0{,}5$.

$$ЗВ = \frac{61970}{0{,}5} = 123940(грн).$$

4.4 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

Для всіх наведених випадків можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_і$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки [52], розраховується за формулою 4.11:

$$\Delta П_і = (\pm \Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_і \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right) \quad (4.11)$$

де $\pm\Delta Ц_о$— зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

N — основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ — основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_о = Ц_б \pm \Delta Ц_о$;

$Ц_б$ — основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

ΔN — зміна основного кількісного показника від впровадження результа-тів науково-технічної розробки в аналізованому році;

Λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість становить 20%, а коефіцієнт Λ=0,8333;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ=0,2...0,5. Обраємо $\rho = 0{,}38$;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році ϑ=18%.

В результаті впровадження результатів наукових розробок поліпшується якість програмного забезпечення, що дозволяє подорожчати за його впровадження, а кількість потенційних користувачів ресурсу збільшиться — у перший рік — на 120 одиниць, на другий рік — ще на 220 одиниць, на третій рік — ще 310 штук.

Ми прогнозуємо щорічний приріст чистого прибутку компанії від впровадження результатів наукових розробок щодо вихідного стану. Збільшення чистого прибутку підприємства $\Delta П_i$за перший рік складе:

$$\Delta П_1 = [650 \cdot 0 + (2800 + 1400) \cdot 120] \cdot 0,8333 \cdot 0{,}38 \cdot \left(1 - \frac{18\%}{100\%}\right) = 130\,866{,}76 \text{ (грн)}.$$

Збільшення чистого прибутку компанії $\Delta П_i$ на другий рік (порівняно з базовим, тобто роком, що передує впровадженню результатів наукових досліджень) складе:

$$\Delta П_2 = [650 \cdot 0 + (2800 + 1400) \cdot (120 + 220)] \cdot 0,8333 \cdot 0{,}38 \cdot \left(1 - \frac{18\%}{100\%}\right) = 370\,789{,}17 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_i$ на третій рік складе:

$$\Delta П_3 = [650 \cdot 0 + (2800 + 1400) \cdot (120 + 220 + 310)] \cdot 0,8333 \cdot 0{,}38 \cdot \left(1 - \frac{18\%}{100\%}\right) = 708\,861{,}64 \text{ (грн)}.$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого

впровадження та комерціалізації науково-технічної розробки за формуою 4.12:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.12)$$

де $\Delta П_i$— збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,15. Обираємо $\tau = 0{,}12$;

t— період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{130\,866{,}76}{(1+0{,}12)^1} + \frac{370\,789{,}17}{(1+0{,}12)^2} + \frac{708\,861{,}64}{(1+0{,}12)^3} = 116\,845{,}32 + 295\,590{,}72 + 504\,553{,}71$$
$$= 916\,998{,}75 \text{ (грн.)}$$

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу 4.13:

$$PV = k_{інв} \cdot ЗВ, \quad (4.13)$$

де $k_{інв}$— коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2...5, але може бути і більшим. Обираємо $k_{інв}$ = 3;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 3 \cdot 123940 = 371820 \text{ (грн.)}$$

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохіддля потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме (формула 4.14):

$$Е_{абс} = ПП - PV \quad (4.14)$$

де ПП — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV — теперішня вартість початкових інвестицій, грн.

$$Е_{абс} = 916998{,}75 - 371820 = 545178{,}75 \text{ (грн.)}$$

Внутрішня економічна дохідність інвестицій $Е_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою 4.15:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (4.15)$$

де $Е_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн;

PV — теперішня вартість початкових інвестицій, грн;

$Т_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$Е_в = \sqrt[3]{1 + \frac{545\,178{,}75}{371\,820}} - 1 = 0{,}35$$

Далі визначають бар'єрну ставку дисконтування τ_{min}, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$визначається за формулою 4.16:

$$\tau_{\text{мін}} = d + f, \qquad (4.16)$$

де d— середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d = 0,9...0,12. Обираємо $d = 0{,}11$;

f — показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою. Обираємо $f = 0{,}38$;

$$\tau_{\text{мін}} = d + f = 0{,}11 + 0{,}38 = 0{,}49\%$$

Величина $E_в > \tau_{\text{мін}}$, отже інвестор може бути зацікавлений у фінансуванні цього дослідження.

Далі розраховуємо період окупності інвестицій $T_{ок}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки (формула 4.17):

$$T_{ок} = \frac{1}{E_в}, \qquad (4.17)$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{0{,}35} = 2{,}86 \text{ року}$$

Оскільки $T_{ок}$= 2,86 року тоді розвиток доречний.

ВИСНОВКИ

Однією із сучасних технологій, яка ще не набула широкого розповсюдження, але здатна змінити уявлення про управління фінансами стала технологія blockchain. Дана технологія дозволяє здійснювати управління фінансами чи будь-якими іншими данними, що зберігаються у середовищі, побудованому за допломогою блокчейну. Незважаючи на розміщення блоків в загальнодоступній мережі інтернет, шифрація доступу до кожного з них дозволяє містити в безпеці дані, що зберігаються в них. Сам ланцюжок блоків може вільно передаватися будь-якому користувачеві Інтернету без ризику втрати вмісту. В даній магістерській кваліфікаційній роботі розглянуто технологію blockchain для управління децентралізованими фінансами та розроблено програму для управління цифровими фінасами.

У першому розділі магістерської роботи було виконано аналіз методу управління децентралізованими фінансами за допомогою технології blockchain. Було детально розглянуто структуру технології, принципи розробки програмного забезпечення із використанням blockchain, переваги та недоліки централізованого та децентралізованого методів управління, алгоритми шифрування даних, методи хешування інформації.

У другому розділі магістерської роботи було обрано сучасний та ефективний інструментарій для розробки програмного продукту. Для розробки програмного засобу було обрано програмне забезпечення Microsoft Visual Studio Code. Для розробки програми використовувалась мова програмування React JS та бібліотека Solidity. Розробка здійснювалась із дотриманнями вимог стандарту ERC-20 та алгоритму SHA-1.

У третьому розділі магістерської роботи було розділі було розроблено новий спосіб створення смарт-контрактів, який доповнює існуючі програміні рішення на основі блокчейну та удосконалює програмні засоби, зо використовують технологію, роблячи впровадження блокчейку більш ефективним та універсальним. Оскільки деякі користувачі та власники бізнесу хотіли б отримати переваги блокчейну, але технологія є не досить

поширеною, було створено програму, яка заохочує користувачів до її використання. На основі вибраних засобів було розроблено та досліджено ефективність створеного програмного засобу.

Тестування розробленої системи, підтвердило правильність роботи програми та ефективність обраного підходу, за рахунок збільшення швидкості транзакцій та зменшення розміру комісії за корисування коштами. Розроблений програмний продукт може використовуватися для випадків, коли користувачі та власники бізнесу потребують і публічної звітності і миттєвого опрацювання транзакцій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bitcoin and Cryptocurrencies: Law Enforcement Investigative Guide [Електронний ресурс]. Режим доступу: http://www.iacpcybercenter.org/ wp-content/uploads/2018/03/Bitcoin.pdf.

2. Mt. Gox [Електронний ресурс]. Режим доступу: https://en. bitcoin.it/wiki/Mt._Gox.

3. Богомолов С. В. Технологія Blockchain / С. В. Богомолов, Г. О. Білоус // Молодь в науці: Дослідження, проблеми, перспективи (МН-2022). [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/view/14111. Дата звернення: Листопад 21, 2021.

4. Silk Road: A Cautionary Tale about Online Anonymity [Електронний ресурс]. Режим доступу: https://medium.com/@marcell74/ the-silk-road-a-real-thriller-and-the-truth-about-the-anonymity-of-bitcoin-198b519ca397.

5. Elliptic: Detect And Prevent Criminal Activity In Cryptocurrency [Електронний ресурс]. Режим доступу: https://www.elliptic.co/.

6. Ethereum Project [Електронний ресурс]. Режим доступу: https: //www.ethereum.org/.

7. CoinJoin [Електронний ресурс]. Режим доступу: https://en. bitcoin.it/wiki/CoinJoin.

8. CoinJoin Security Research [Електронний ресурс]. Режим доступу: https://pdfs.semanticscholar.org/0325/0ef8dfa44bb26a43df0e7e846324286e35e5.pdf.

9. What’s a Sybil Attack & How Do Blockchains Mitigate Them? [Електронний ресурс]. Режим доступу: https://coincentral.com/ sybil-attack-blockchain/.

10. Monero: Privacy in the blockchain [Електронний ресурс]. Режим доступу: https://eprint.iacr.org/2018/535.pdf.

11. CryptoNote [Електронний ресурс]. Режим доступу: https:// cryptonote.org/.

11. Monero Ring Attack: Recreating Zero Mixin Transaction Effect [Електронний ресурс]. Режим доступу: https://eprint.iacr.org/2018/348. pdf.

12. A Traceability Analysis of Monero’s Blockchain [Електронний 80 ресурс]. Режим доступу: https://eprint.iacr.org/2017/338.pdf.

13. On the (Im)possibility of Obfuscating Programs [Електронний ресурс]. Режим доступу: https://www.iacr.org/archive/crypto2001/ 21390001.pdf.

14. Indistinguishability Code Obfuscation research [Електронний ресурс]. — Режим доступу: https://eprint.iacr.org/2013/451.pdf.

15. Intel Software Guard Extensions [Електронний ресурс]. Режим доступу: https://software.intel.com/sgx.

16. Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contract Execution [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/1804.05141.

17. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system,"2008, [Електронний ресурс]. Режим доступу: https://bitcoin.org/bitcoin.pdf.

18. Proof of work[Електронний ресурс]. Режим доступу: https://en. bitcoin.it/wiki/Proof_of_work.

19. N. van Saberhagen, "Cryptonote v 2. 0,"2013. Режим доступу: https://downloads.getmonero.org/whitepaper_annotated.pdf. S. Meiklejohn et al., "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names,"USENIX ;login:, 2013.

20. A. Kumar, C. Fischer, S. Tople, and P. Saxena, "A Traceability Analysis of Monero’s Blockchain,"IACR Cryptology ePrint Archive, vol. 2017, p. 338, 2017.

21. A. Miller, M. M¨oser, K. Lee, and A. Narayanan, "An Empirical Analysis of Linkability in the Monero Blockchain,"arXiv preprintarXiv:1704.04299, 2017.

22. RingCT [Електронний ресурс]. Режим доступу: https://ww. getmonero.org/resources/moneropedia/ringCT.html.

23. A note on fees [Електронний ресурс]. Режим доступу: https: //ww.getmonero.org/2017/12/11/A-note-on-fees.html.

24. Monero Avg. Transaction Fee historical chart [Електронний ресурс]. Режим доступу: https://bitinfocharts.com/comparison/ 81 monero-transactionfees.html.

25. What is a SYN flood attack [Електронний ресурс]. Режим доступу: https://www.imperva.com/learn/application-security/syn-flood/.

26. Kovri [Електронний ресурс]. Режим доступу: https://www. getmonero.org/resources/moneropedia/kovri.html.

27. Проект невидимый интернет (I2P) [Електронний ресурс]. Режим доступу: https://geti2p.net/ru/about/intro.

28. Hyperledger Sawtooth [Електронний ресурс]. Режим доступу: https://sawtooth.hyperledger.org/docs/core/releases/1.0/introduction.html.

29. Corda Network [Електронний ресурс]. Режим доступу: https: //www.r3.com/.

30. Hyperledger Fabric [Електронний ресурс]. Режим доступу: https: //www.hyperledger.org/projects/fabric.

31. Hyperledger Burrow [Електронний ресурс]. Режим доступу: https://www.hyperledger.org/projects/hyperledger-burrow.

32. What is Nakamoto Consensus? Complete Beginner’s Guide [Електронний ресурс]. Режим доступу: https://blockonomi.com/ nakamoto-consensus/.

33. Byzantine Fault Tolerance [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Byzantine_fault.

34. Hash Functions, Merkle Trees and Radix Tries. Things to Know before Tackling ”Merkle Tries” [Електронний ресурс]. Режим доступу: https://medium.com/orbs-network/ designing-a-state-database-blockchains-part-i-the-basics-90d814d6973b.

35. Secp256k1[Електронний ресурс]. Режим доступу: https://en. bitcoin.it/wiki/Secp256k1.

36. Gossip protocol[Електронний ресурс]. Режим доступу: https:// en.wikipedia.org/wiki/Gossip_protocol.

37. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [Електронний ресурс]. Режим доступу: https://tools.ietf.org/html/rfc5280. 82

38. Everything You Need to Know About SSL Certificates [Електронний ресурс]. — Режим доступу: https://www.verisign.com/ en_US/website-presence/online/ssl-certificates/index.xhtml

39. Apache Kafka [Електронний ресурс]. Режим доступу: https:// kafka.apache.org/.

40. Lightning Network: Scalable, Instant Bitcoin/Blockchain Transactions [Електронний ресурс]. Режим доступу: https: //lightning.network/.

41. Evaluating User Privacy in Bitcoin [Електронний ресурс]. — Режим доступу: https://eprint.iacr.org/2012/596.

42. The components of the Wired Spanning Forest are recurrent [Електронний ресурс]. Режим доступу: https://link.springer.com/article/ 10.1007%2Fs00440-002-0236-0. 43. About Tor project [Електронний ресурс] . — Режим доступу: https: //www.torproject.org/.

ДОДАТОК А

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

_____проф., д.т.н. О. Д. Азаров

«___» _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Метод та засоби управління децентралізованими фінансами за допомогою технології Blockchain»

08-23.МКР.016.00.000 ТЗ

Науковий керівник: к.т.н.

_____ Богомолов С.В.

Магістрант групи 2КІ-20м

_____ Білоус Г.О.

Вінниця 2021

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність даного дослідження визначається необхідністю вирішення основних проблем зберігання та управління фінансами, які б мали відкритий публічний доступ до звісності та мали б підвищену швидкість виконання транзакцій. Особливо актуальним дане дослідження є у зв’язку з необхідністю забезпечення захисту від несанкціонованого доступу до фінансів, або будь-якої іншої цінної інформації, яка потребує бути захищеною.

1.2 Наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення МКР

2.1 Мета магістерської роботи полягає у підвищенні ефективності та надійності методів управління децентралізованими фінансами.

2.2 Призначення розробки — виконання магістерської кваліфікаційної роботи.

3 Вихідні дані для виконання МКР

Виконати розробку програмного забезпечення для управління децентралізованими фінансами на основі технології blockchain та провести його тестування. Лістинги програми представити в додатках до роботи.

4 Вимоги до виконання МКР

МКР повинна задовольняти такі вимоги:

— запропонувати нові підходи для реалізації управління децентралізованими фінансами;

— розробити алгоритм побудови смартконтракту, який дозволяє здійснювати фінансові транзакції та надавати винагороду за використання токенів;

— вхідні дані — 10 000 токенів для виконання транзакцій та подальшого використання;

— результат роботи, а саме правильного здійснення виконання транзакцій та нарахування винагород за корисування рзроблюваною програмою.

5 Етапи МКР та очікувані результати

Етапи МКР та очікувані результати наведені в таблиці А.1

Таблиця А.1 — Етапи виконання роботи

№	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	03.09.21	08.09.21	Вступ
2	Аналіз літературних джерел технології Blockchain	10.09.21	21.09.21	розділ 1
3	Розробка технічного завдання	20.09.2021	21.09.21	Технічне завдання
3	Розробка структури системи управління децентралізованими фінансами	22.09.21	03.10.21	Розділ 2, розробка структури
4	Розробка програми, проектування програмного продукту	05.10.21	27.10.21	Розділ 3, розробка програми
5	Практична реалізація, результати.	04.11.21	17.11.21	Розділ 3
6	Розробка економічної частини	17.11.21	29.11.21	Розділ 4
7	Оформлення пояснювальної записки	01.12.21	16.12.21	ПЗ, презентація

6 Матеріали, що подаються до захисту МКР: пояснювальна записка МКР, ілюстративні та графічні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив рецензента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів розрахункової та графічної документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР

Вимоги викладені в ДСТУ 3008-2015, та положенні про МКР ВНТУ 2021.

Технічне завдання до виконання отримала _____ Білоус Г. О.

ДОДАТОК Б

Лістинг створення токену

```
pragma solidity 0.6.12;

import "@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol";

// CakeToken with Governance.

contract CakeToken is BEP20('AnnaBell Token', 'AnnaB') {

    /// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).

    function mint(address _to, uint256 _amount) public onlyOwner {

        _mint(_to, _amount);

        _moveDelegates(address(0), _delegates[_to], _amount);

    }

    mapping (address => address) internal _delegates;

    /// @notice A checkpoint for marking number of votes from a given block

    struct Checkpoint {

        uint32 fromBlock;

        uint256 votes;

    }

    /// @notice A record of votes checkpoints for each account, by index

    mapping (address => mapping (uint32 => Checkpoint)) public checkpoints;

    /// @notice The number of checkpoints for each account

    mapping (address => uint32) public numCheckpoints;

    /// @notice The EIP-712 typehash for the contract's domain

    bytes32 public constant DOMAIN_TYPEHASH = keccak256("EIP712Domain(string name,uint256 chainId,address verifyingContract)");
```

```
        /// @notice The EIP-712 typehash for the delegation struct used by the
contract

        bytes32 public constant DELEGATION_TYPEHASH =
keccak256("Delegation(address delegatee,uint256 nonce,uint256 expiry)");

        /// @notice A record of states for signing / validating signatures

        mapping (address => uint) public nonces;

          /// @notice An event thats emitted when an account changes its delegate

        event DelegateChanged(address indexed delegator, address indexed
fromDelegate, address indexed toDelegate);

        /// @notice An event thats emitted when a delegate account's vote balance
changes

        event DelegateVotesChanged(address indexed delegate, uint
previousBalance, uint newBalance);

        /**

         * @notice Delegate votes from `msg.sender` to `delegatee`

         * @param delegator The address to get delegatee for

         */

        function delegates(address delegator)

            external

            view

            returns (address)

        {

            return _delegates[delegator];

        }

       /**

        * @notice Delegate votes from `msg.sender` to `delegatee`

        * @param delegatee The address to delegate votes to

        */

        function delegate(address delegatee) external {
```

```
    return _delegate(msg.sender, delegatee);
}
/**
 * @notice Delegates votes from signatory to `delegatee`
 * @param delegatee The address to delegate votes to
 * @param nonce The contract state required to match the signature
 * @param expiry The time at which to expire the signature
 * @param v The recovery byte of the signature
 * @param r Half of the ECDSA signature pair
 * @param s Half of the ECDSA signature pair
 */
function delegateBySig(
    address delegatee,
    uint nonce,
    uint expiry,
    uint8 v,
    bytes32 r,
    bytes32 s
)
    external
{
    bytes32 domainSeparator = keccak256(
        abi.encode(
            DOMAIN_TYPEHASH,
            keccak256(bytes(name())),
            getChainId(),
            address(this)
```

```
                )
            );
            bytes32 structHash = keccak256(
                abi.encode(
                    DELEGATION_TYPEHASH,
                    delegatee,
                    nonce,
                    expiry
                )
            );
            bytes32 digest = keccak256(
                abi.encodePacked(
                    "\x19\x01",
                    domainSeparator,
                    structHash
                )
            );
            address signatory = ecrecover(digest, v, r, s);
            require(signatory != address(0), "CAKE::delegateBySig: invalid signature");
            require(nonce == nonces[signatory]++, "CAKE::delegateBySig: invalid nonce");
            require(now <= expiry, "CAKE::delegateBySig: signature expired");
            return _delegate(signatory, delegatee);
        }
        /**
         * @notice Gets the current votes balance for `account`
         * @param account The address to get votes balance
```

```
     * @return The number of current votes for `account`
     */
    function getCurrentVotes(address account)
        external
        view
        returns (uint256)
    {
        uint32 nCheckpoints = numCheckpoints[account];
        return nCheckpoints > 0 ? checkpoints[account][nCheckpoints - 1].votes : 0;
    }

    /**
     * @notice Determine the prior number of votes for an account as of a block number
     * @dev Block number must be a finalized block or else this function will revert to prevent misinformation.
     * @param account The address of the account to check
     * @param blockNumber The block number to get the vote balance at
     * @return The number of votes the account had as of the given block
     */
    function getPriorVotes(address account, uint blockNumber)
        external
        view
        returns (uint256)
    {
        require(blockNumber < block.number, "CAKE::getPriorVotes: not yet determined");
```

```
        uint32 nCheckpoints = numCheckpoints[account];
        if (nCheckpoints == 0) {
            return 0;
        }
        // First check most recent balance
        if (checkpoints[account][nCheckpoints - 1].fromBlock <=
blockNumber) {
            return checkpoints[account][nCheckpoints - 1].votes;
        }
        // Next check implicit zero balance
        if (checkpoints[account][0].fromBlock > blockNumber) {
            return 0;
        }
        uint32 lower = 0;
        uint32 upper = nCheckpoints - 1;
        while (upper > lower) {
            uint32 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
            Checkpoint memory cp = checkpoints[account][center];
            if (cp.fromBlock == blockNumber) {
                return cp.votes;
            } else if (cp.fromBlock < blockNumber) {
                lower = center;
            } else {
                upper = center - 1;
            }
        }
        return checkpoints[account][lower].votes;
```

```
    }

    function _delegate(address delegator, address delegatee)

        internal

    {

        address currentDelegate = _delegates[delegator];

        uint256 delegatorBalance = balanceOf(delegator); // balance of
underlying CAKEs (not scaled);

        _delegates[delegator] = delegatee;

        emit DelegateChanged(delegator, currentDelegate, delegatee);

        _moveDelegates(currentDelegate, delegatee, delegatorBalance);

    }

    function _moveDelegates(address srcRep, address dstRep, uint256
amount) internal {

        if (srcRep != dstRep && amount > 0) {

            if (srcRep != address(0)) {

                // decrease old representative

                uint32 srcRepNum = numCheckpoints[srcRep];

                uint256 srcRepOld = srcRepNum > 0 ?
checkpoints[srcRep][srcRepNum - 1].votes : 0;

                uint256 srcRepNew = srcRepOld.sub(amount);

                _writeCheckpoint(srcRep, srcRepNum, srcRepOld, srcRepNew);

            }

            if (dstRep != address(0)) {

                // increase new representative

                uint32 dstRepNum = numCheckpoints[dstRep];

                uint256 dstRepOld = dstRepNum > 0 ?
checkpoints[dstRep][dstRepNum - 1].votes : 0;

                uint256 dstRepNew = dstRepOld.add(amount);
```

```
                _writeCheckpoint(dstRep, dstRepNum, dstRepOld, dstRepNew);
            }
        }
    }
    function _writeCheckpoint(
        address delegatee,
        uint32 nCheckpoints,
        uint256 oldVotes,
        uint256 newVotes
    )
        internal
    {
        uint32 blockNumber = safe32(block.number,
"CAKE::_writeCheckpoint: block number exceeds 32 bits");


        if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints -
1].fromBlock == blockNumber) {
            checkpoints[delegatee][nCheckpoints - 1].votes = newVotes;
        } else {
            checkpoints[delegatee][nCheckpoints] = Checkpoint(blockNumber,
newVotes);
            numCheckpoints[delegatee] = nCheckpoints + 1;
        }
        emit DelegateVotesChanged(delegatee, oldVotes, newVotes);
    }
    function safe32(uint n, string memory errorMessage) internal pure returns
(uint32) {
        require(n < 2**32, errorMessage);
```

```
        return uint32(n);
    }
    function getChainId() internal pure returns (uint) {
        uint256 chainId;
        assembly { chainId := chainid() }
        return chainId;
    }
}
```

ДОДАТОК В

Лістинг смарт-контракту

```
pragma solidity 0.6.12;

import '@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol';
import '@pancakeswap/pancake-swap-lib/contracts/token/BEP20/IBEP20.sol';
import '@pancakeswap/pancake-swap-lib/contracts/token/BEP20/SafeBEP20.sol';
import '@pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol';

import "./CakeToken.sol";
import "./SyrupBar.sol";

// import "@nomiclabs/buidler/console.sol";

interface IMigratorChef {
    // Perform LP token migration from legacy PancakeSwap to CakeSwap.
    // Take the current LP token address and return the new LP token address.
    // Migrator should have full access to the caller's LP token.
    // Return the new LP token address.
    //
    // XXX Migrator must have allowance access to PancakeSwap LP tokens.
    // CakeSwap must mint EXACTLY the same amount of CakeSwap LP tokens or
    // else something bad will happen. Traditional PancakeSwap does not
    // do that so be careful!
    function migrate(IBEP20 token) external returns (IBEP20);
}

// MasterChef is the master of Cake. He can make Cake and he is a fair guy.
//
// Note that it's ownable and the owner wields tremendous power. The ownership
// will be transferred to a governance smart contract once CAKE is sufficiently
// distributed and the community can show to govern itself.
//
// Have fun reading it. Hopefully it's bug-free. God bless.
contract MasterChef is Ownable {
    using SafeMath for uint256;
    using SafeBEP20 for IBEP20;

    // Info of each user.
```

```
    struct UserInfo {
        uint256 amount;     // How many LP tokens the user has provided.
        uint256 rewardDebt; // Reward debt. See explanation below.
    }

    // Info of each pool.
    struct PoolInfo {
        IBEP20 lpToken;           // Address of LP token contract.
        uint256 allocPoint;       // How many allocation points assigned to this pool. CAKEs to distribute per block.
        uint256 lastRewardBlock;  // Last block number that CAKEs distribution occurs.
        uint256 accCakePerShare; // Accumulated CAKEs per share, times 1e12. See below.
    }

    // The CAKE TOKEN!
    CakeToken public cake;
    // The SYRUP TOKEN!
    SyrupBar public syrup;
    // Dev address.
    address public devaddr;
    // CAKE tokens created per block.
    uint256 public cakePerBlock;
    // Bonus muliplier for early cake makers.
    uint256 public BONUS_MULTIPLIER = 1;
    // The migrator contract. It has a lot of power. Can only be set through governance (owner).
    IMigratorChef public migrator;

    // Info of each pool.
    PoolInfo[] public poolInfo;
    // Info of each user that stakes LP tokens.
    mapping (uint256 => mapping (address => UserInfo)) public userInfo;
    // Total allocation points. Must be the sum of all allocation points in all pools.
    uint256 public totalAllocPoint = 0;
    // The block number when CAKE mining starts.
    uint256 public startBlock;

    event Deposit(address indexed user, uint256 indexed pid, uint256 amount);
    event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);
    event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);
```

```
        constructor(
            CakeToken _cake,
            SyrupBar _syrup,
            address _devaddr,
            uint256 _cakePerBlock,
            uint256 _startBlock
        ) public {
            cake = _cake;
            syrup = _syrup;
            devaddr = _devaddr;
            cakePerBlock = _cakePerBlock;
            startBlock = _startBlock;

            // staking pool
            poolInfo.push(PoolInfo({
                lpToken: _cake,
                allocPoint: 1000,
                lastRewardBlock: startBlock,
                accCakePerShare: 0
            }));

            totalAllocPoint = 1000;

        }

        function updateMultiplier(uint256 multiplierNumber) public onlyOwner {
            BONUS_MULTIPLIER = multiplierNumber;
        }

        function poolLength() external view returns (uint256) {
            return poolInfo.length;
        }

        // Add a new lp to the pool. Can only be called by the owner.
        // XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.
        function add(uint256 _allocPoint, IBEP20 _lpToken, bool _withUpdate) public onlyOwner {
            if (_withUpdate) {
                massUpdatePools();
            }
            uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
            totalAllocPoint = totalAllocPoint.add(_allocPoint);
            poolInfo.push(PoolInfo({
```

```
            lpToken: _lpToken,
            allocPoint: _allocPoint,
            lastRewardBlock: lastRewardBlock,
            accCakePerShare: 0
        }));
        updateStakingPool();
    }

    // Update the given pool's CAKE allocation point. Can only be called by the owner.
    function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate) public onlyOwner {
        if (_withUpdate) {
            massUpdatePools();
        }
        uint256 prevAllocPoint = poolInfo[_pid].allocPoint;
        poolInfo[_pid].allocPoint = _allocPoint;
        if (prevAllocPoint != _allocPoint) {
            totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint);
            updateStakingPool();
        }
    }

    function updateStakingPool() internal {
        uint256 length = poolInfo.length;
        uint256 points = 0;
        for (uint256 pid = 1; pid < length; ++pid) {
            points = points.add(poolInfo[pid].allocPoint);
        }
        if (points != 0) {
            points = points.div(3);
            totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points);
            poolInfo[0].allocPoint = points;
        }
    }

    // Set the migrator contract. Can only be called by the owner.
    function setMigrator(IMigratorChef _migrator) public onlyOwner {
        migrator = _migrator;
    }

    // Migrate lp token to another lp contract. Can be called by anyone. We trust that migrator contract is good.
    function migrate(uint256 _pid) public {
```

```
        require(address(migrator) != address(0), "migrate: no migrator");
        PoolInfo storage pool = poolInfo[_pid];
        IBEP20 lpToken = pool.lpToken;
        uint256 bal = lpToken.balanceOf(address(this));
        lpToken.safeApprove(address(migrator), bal);
        IBEP20 newLpToken = migrator.migrate(lpToken);
        require(bal == newLpToken.balanceOf(address(this)), "migrate: bad");
        pool.lpToken = newLpToken;
    }

    // Return reward multiplier over the given _from to _to block.
    function getMultiplier(uint256 _from, uint256 _to) public view returns
(uint256) {
        return _to.sub(_from).mul(BONUS_MULTIPLIER);
    }

    // View function to see pending CAKEs on frontend.
    function pendingCake(uint256 _pid, address _user) external view returns
(uint256) {
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][_user];
        uint256 accCakePerShare = pool.accCakePerShare;
        uint256 lpSupply = pool.lpToken.balanceOf(address(this));
        if (block.number > pool.lpLastRewardBlock && lpSupply != 0) {
            uint256 multiplier = getMultiplier(pool.lastRewardBlock,
block.number);
            uint256 cakeReward =
multiplier.mul(cakePerBlock).mul(pool.allocPoint).div(totalAllocPoint);
            accCakePerShare =
accCakePerShare.add(cakeReward.mul(1e12).div(lpSupply));
        }
        return
user.amount.mul(accCakePerShare).div(1e12).sub(user.rewardDebt);
    }

    // Update reward variables for all pools. Be careful of gas spending!
    function massUpdatePools() public {
        uint256 length = poolInfo.length;
        for (uint256 pid = 0; pid < length; ++pid) {
            updatePool(pid);
        }
    }


    // Update reward variables of the given pool to be up-to-date.
    function updatePool(uint256 _pid) public {
```

```
        PoolInfo storage pool = poolInfo[_pid];
        if (block.number <= pool.lastRewardBlock) {
            return;
        }
        uint256 lpSupply = pool.lpToken.balanceOf(address(this));
        if (lpSupply == 0) {
            pool.lastRewardBlock = block.number;
            return;
        }
        uint256 multiplier = getMultiplier(pool.lastRewardBlock,
block.number);
        uint256 cakeReward =
multiplier.mul(cakePerBlock).mul(pool.allocPoint).div(totalAllocPoint);
        cake.mint(devaddr, cakeReward.div(10));
        cake.mint(address(syrup), cakeReward);
        pool.accCakePerShare =
pool.accCakePerShare.add(cakeReward.mul(1e12).div(lpSupply));
        pool.lastRewardBlock = block.number;
    }

    // Deposit LP tokens to MasterChef for CAKE allocation.
    function deposit(uint256 _pid, uint256 _amount) public {

        require (_pid != 0, 'deposit CAKE by staking');

        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][msg.sender];
        updatePool(_pid);
        if (user.amount > 0) {
            uint256 pending =
user.amount.mul(pool.accCakePerShare).div(1e12).sub(user.rewardDebt);
            if(pending > 0) {
                safeCakeTransfer(msg.sender, pending);
            }
        }
        if (_amount > 0) {
            pool.lpToken.safeTransferFrom(address(msg.sender), address(this),
_amount);
            user.amount = user.amount.add(_amount);
        }
        user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12);
        emit Deposit(msg.sender, _pid, _amount);
    }

    // Withdraw LP tokens from MasterChef.
    function withdraw(uint256 _pid, uint256 _amount) public {
```

```
        require (_pid != 0, 'withdraw CAKE by unstaking');
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][msg.sender];
        require(user.amount >= _amount, "withdraw: not good");

        updatePool(_pid);
        uint256 pending =
user.amount.mul(pool.accCakePerShare).div(1e12).sub(user.rewardDebt);
        if(pending > 0) {
            safeCakeTransfer(msg.sender, pending);
        }
        if(_amount > 0) {
            user.amount = user.amount.sub(_amount);
            pool.lpToken.safeTransfer(address(msg.sender), _amount);
        }
        user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12);
        emit Withdraw(msg.sender, _pid, _amount);
    }

    // Stake CAKE tokens to MasterChef
    function enterStaking(uint256 _amount) public {
        PoolInfo storage pool = poolInfo[0];
        UserInfo storage user = userInfo[0][msg.sender];
        updatePool(0);
        if (user.amount > 0) {
            uint256 pending =
user.amount.mul(pool.accCakePerShare).div(1e12).sub(user.rewardDebt);
            if(pending > 0) {
                safeCakeTransfer(msg.sender, pending);
            }
        }
        if(_amount > 0) {
            pool.lpToken.safeTransferFrom(address(msg.sender), address(this),
_amount);
            user.amount = user.amount.add(_amount);
        }
        user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12);

        syrup.mint(msg.sender, _amount);
        emit Deposit(msg.sender, 0, _amount);
    }

    // Withdraw CAKE tokens from STAKING.
    function leaveStaking(uint256 _amount) public {
        PoolInfo storage pool = poolInfo[0];
```

```
        UserInfo storage user = userInfo[0][msg.sender];
        require(user.amount >= _amount, "withdraw: not good");
        updatePool(0);
        uint256 pending =
user.amount.mul(pool.accCakePerShare).div(1e12).sub(user.rewardDebt);
        if(pending > 0) {
            safeCakeTransfer(msg.sender, pending);
        }
        if(_amount > 0) {
            user.amount = user.amount.sub(_amount);
            pool.lpToken.safeTransfer(address(msg.sender), _amount);
        }
        user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12);

        syrup.burn(msg.sender, _amount);
        emit Withdraw(msg.sender, 0, _amount);
    }

    // Withdraw without caring about rewards. EMERGENCY ONLY.
    function emergencyWithdraw(uint256 _pid) public {
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][msg.sender];
        pool.lpToken.safeTransfer(address(msg.sender), user.amount);
        emit EmergencyWithdraw(msg.sender, _pid, user.amount);
        user.amount = 0;
        user.rewardDebt = 0;
    }

    // Safe cake transfer function, just in case if rounding error causes pool to
not have enough CAKEs.
    function safeCakeTransfer(address _to, uint256 _amount) internal {
        syrup.safeCakeTransfer(_to, _amount);
    }

    // Update dev address by the previous dev.
    function dev(address _devaddr) public {
        require(msg.sender == devaddr, "dev: wut?");
        devaddr = _devaddr;
    }
}
```

ДОДАТОК Г

Результат внесення депозиту

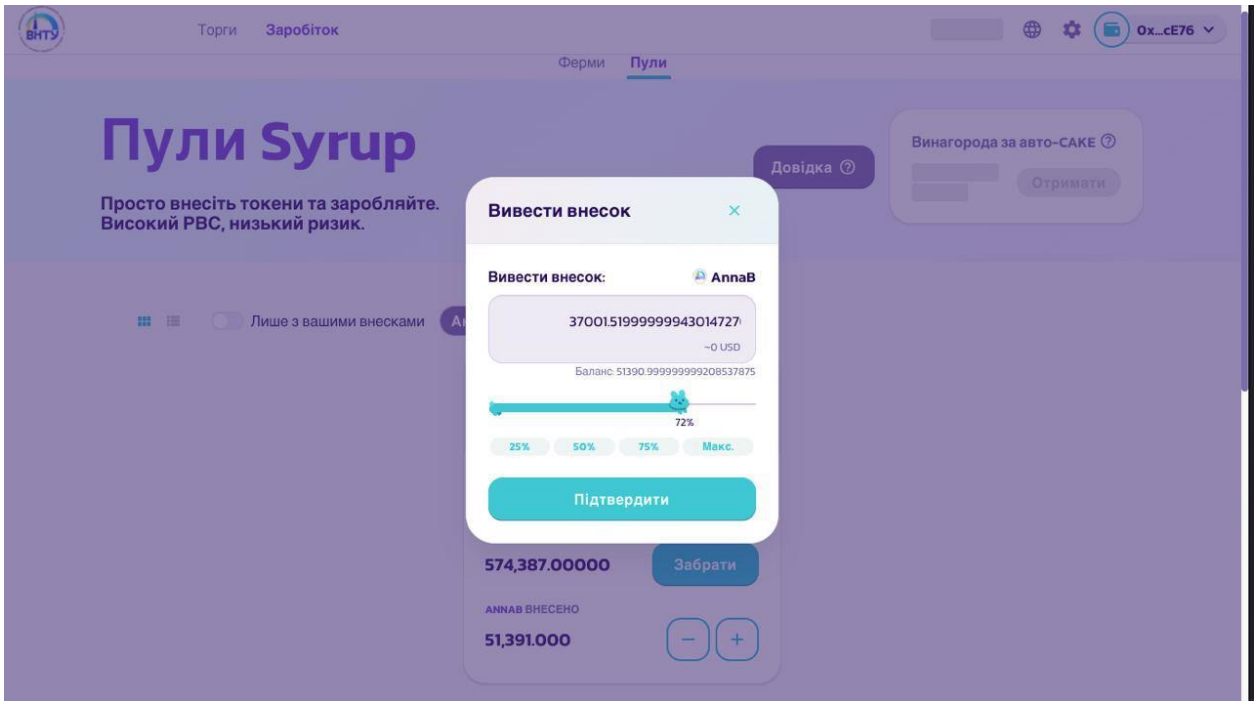


Рисунок Г.1 — Результат внесення депозиту

ДОДАТОК Д

Результат виведення коштів на гаманець

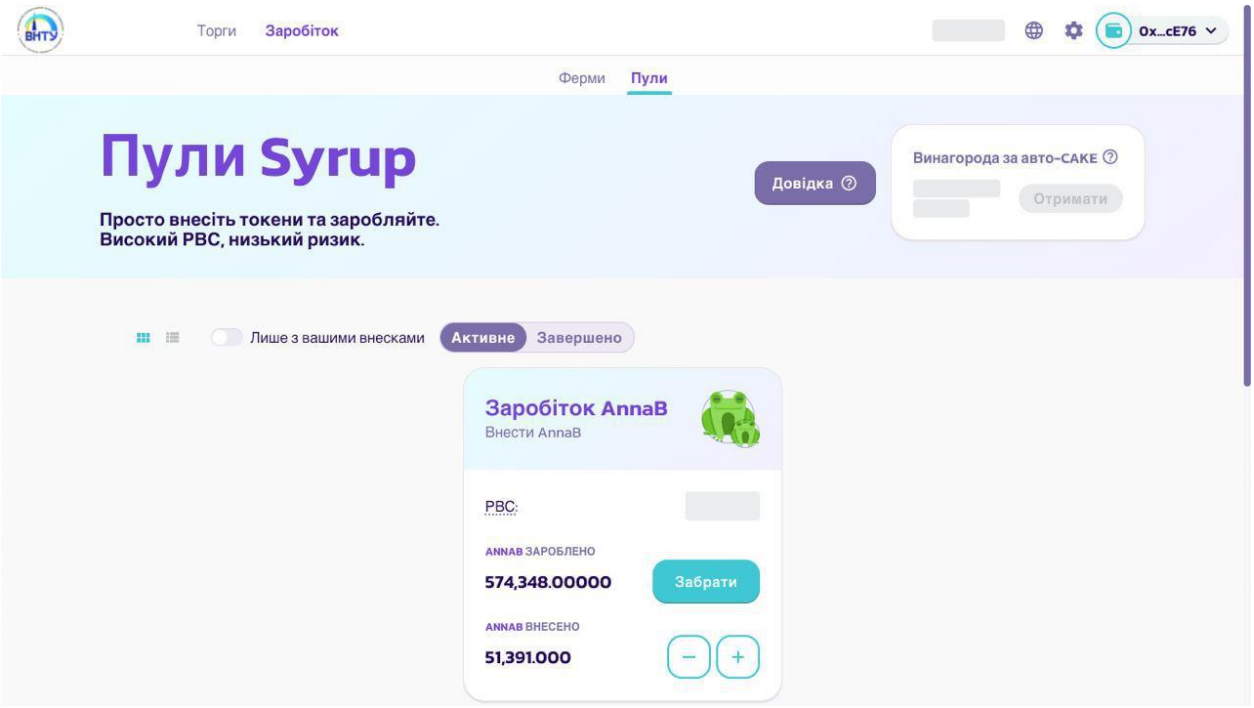


Рисунок Д.1 — Результат веведення коштів на гаманець

ДОДАТОК Е

Структурна схема програми

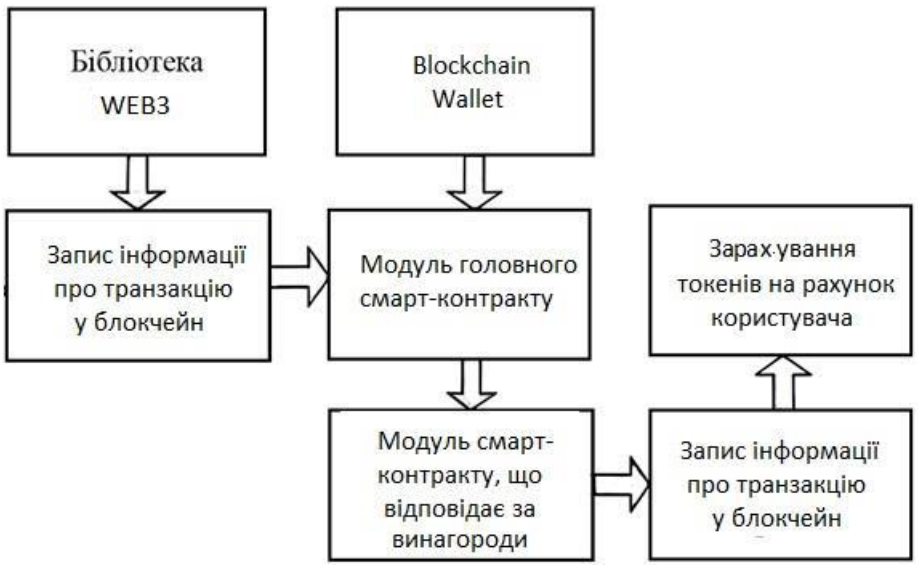


Рисунок Е.1 — Структурна схема програми

ДОДАТОК Ж

Схема хешів блоків блокчейну

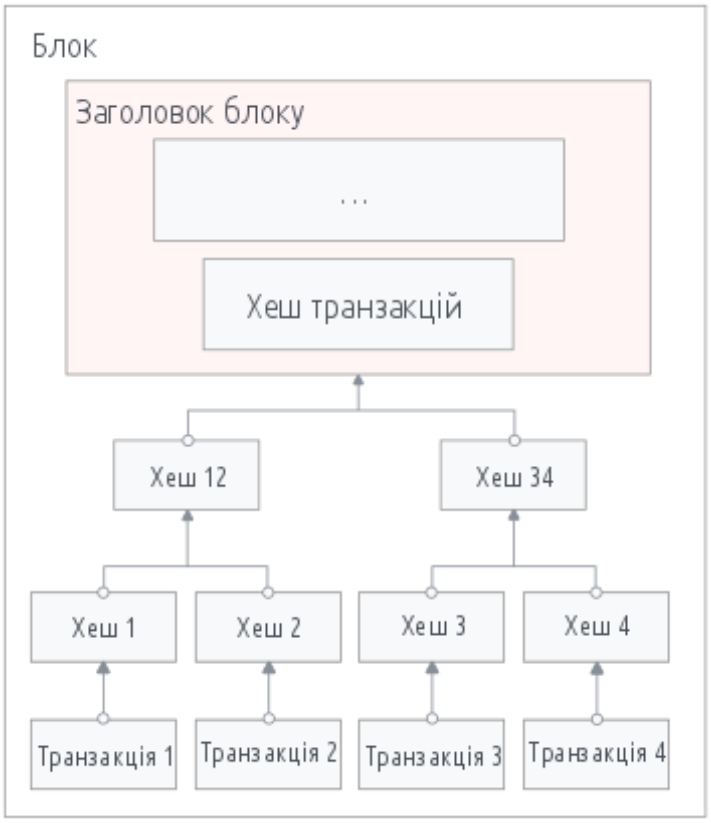


Рисунок Ж.1 — Схема хешів блоків блокчейну

ДОДАТОК И

Протокол перевірки навчальної (кваліфікаційної) роботи

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Метод та засоби управління децентралізованими фінансами за допомогою технології Blockchain

Тип роботи: магістерська кваліфікаційна робота

(кваліфікаційна роботи, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет (інститут), навчальна група)

Науковий керівник Богомолов С.В. доцент

(прізвище, ініціали, посада)

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	98,4
Тривога/Білі знаки	/	Схожість	1,6

Аналіз звіту подібності (відмінити подібне)

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності і відсутності самостійності її автора. Робот направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Білоус Г. О.

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

Ступінь оригінальності роботи відповідає вимогам, що висуваються до МКР

Особа, відповідальна за перевірку_____ Захарченко С.М.

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали)