НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.9

«ПОГОДЖЕНО»

Декан факультету
інформаційних технологій

Болбот І.М., д.т.н., професор

_____

____ _____ 2024 р.

«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»

Завідувач кафедри комп'ютерних наук

Голуб Б.Л., к.т.н., доцент

_____

____ _____ 2024 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Система моніторингу фінансових транзакцій на базі Ethereum та EVM сумісних блокчейнів в контенті фінансової аналітики_____

Спеціальність _____122 «Комп'ютерні науки»_____

_____
(код і назва)

Освітня програма _____Комп'ютерно еколого-економічний моніторинг_

_____
(назва)

Орієнтація освітньої програми _____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_____ _____ _____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

__кандидат технічних наук, доцент___ _____ _Сватко Віталій Володимирович__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ **____Войтович Сергій Вікторович_____**
(підпис) (ПІБ студента)

КИЇВ-2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

UI – User Interface, інтерфейсь користувача

ML – Machine learning, машинне навчання

IDE – Integrated Drive Electronics, інтегроване середовище розробки

EVM – Ethereum Virtual Machine, віртуальна машина Ethereum

OLAP – Online Analytical Processing, аналітична обробка в реальному часі

SSAS – SQL Server Analysis Service

SSRS – SQL Server Reporting Service

SSIS – SQL Server Integration Service

API – Application Programming Interface, ітерфейс програмування застосунків

REST – Representational State Transfer, передача репрезентативного стану

HTTP – HyperText Transfer Protocol

1. ВСТУП

За останній час значного розвитку набирає система блокчейн, кількість користувачів з кожним роком збільшується, а стабільність, надійність та безпека мережі покращується. Кількість оброблених транзакцій стрімко наближається до кількості фінансових операцій таких платіжних систем як VISA або Mastercard. Багато країн світу почали роботу над проектуванням законодавчих актів та норм, що дали б змогу інтегрувати блокчейн технології та криптовалюти в світ фінансових операцій, зробити використання криптовалютних платежів доступними, прозорими та безпечними. Деякі країни світу навіть використовують криптовалюти як фінансовий резерв. Україна не виняток і в нас теж відбуваються певні роботи над регуляцією обігу криптовалют, створення системи оподаткування та можливості інтеграції криптовалют в платіжні сервіси.

1.1 АКТУАЛЬНІСТЬ.

Ринковий обіг однієї лише криптовалюти Bincoin станом на 16 листопада 2024 складає більше 64 мільярдів доларів США за 24 години. Так і технологічні гіганти як Tesla вже мали досвід впровадження оплати за товари в криптовалюті. Деякі українські компанії, такі як Foxtrot, WOG та інші теж мали досвід провадження платежів використовуючи криптовалюти. Все більше фізичних осіб та підприємців задаються питанням впровадження криптовалютних платежів, проте на даних момент законом чітко не регламентовано як саме працювати та декларувати криптовалюти. Проте ні для кого не секрет що будь-які платежі приносять або дохід державі у вигляді податків, або відбуваються в тіні. Для всіх платежів існують системи моніторингу, такі як банківські системи, податкові системи та інші. Ці системи дають змогу як підприємцям так і фізичним особам легко слідкувати за обігом своїх активів, мати змогу вчасно та коректно декларувати їх

2. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 ОБ’ЄКТ ДОСЛІДЖЕННЯ

Об’єкт дослідження – Ethereum блокчейн. Система реалізована як єдина децентралізована віртуальна машина. Дозволяє створення та виконання децентралізованих онлайн сервісів на базі розуміних контрактів. Децентралізація забезпечується одноранговістю окремих компонентів мережі, тобто кожен учасник мережі має однакові права та обов’язки в системі. Також важливою частиною є політика валідації транзакцій. Кожен учасник мережі може валідувати блоки транзакцій та отримувати за це винагороду, кожен інший учасник мережі може валідувати роботу іншого учасника та підтверджувати правильність. Для доступу до процесу валідації потрібно внести депозит у розмірі 32 ETH. У разі виявлення помилки валідації в учасника мережі, його депозит не повертається і буде розподілений між учасниками мережі, таким чином забезпечується цілісність, стабільність та безпека мережі. Загалом блокчейн являється послідовністю блоків, кожен блок має свій унікальний хеш, кожен наступний блок включає хеш попереднього, і вираховує хеш базуючись на власних даних включаючи хеш попереднього блоку, таким чином реалізовується захист від внесення змін в блокчейн заднім числом.

2.2 ПРЕДМЕТ ДОСЛІДЖЕННЯ

Предмет дослідження – Транзакція в Ethereum блокчейні. Транзакція – це найменша модульна одиниця роботи в системі Ethereum блокчейн. Кожна транзакція може бути або виконана повністю, або невиконана зовсім, дуже схоже до принципів виконання транзакцій в реляційних базах даних. Алгоритм виконання транзакції описується програмним кодом «розмуного контракту», розуміний контракт – програмний код що зберігається в блокчейні, для розгортання програмного коду в блокчейні також використовуються транзакції.

Кожна транзакція має свою складність яка вираховується в одиницях GAS. GAS залежить від складності необхідних обчислень, об’ємів читання та запису що необхідні для виконання.

Операція	GAS	Опис
ADD, SUB	3	Базові арифметичні операції
MUL	5	Операція множення
DIV, MOD	5	Операція ділення та остачі від ділення
EXP	10 + 50/byte	Операція експоненти
LT, GT, EQ, ISZERO	3	Порівняльні операції
AND, OR, XOR, NOT	3	Логічні операції
SHA3	30 + 6/word	Keccak-256 хеш функція
SLOAD	2100 (може варіюватись)	Читання значення зі блокчейну
SSTORE	Варіантивно	Запис в блокчейн
MLOAD	3	Читання пам’яті
MSTORE, MSTORE8	3	Запис в пам’ять
PUSH1, PUSH32	3	Запис 1-32 байтів в стек
JUMP, JUMPI	8	Стрибок до іншого програмного рядка
CALL	700 + внутрішній GAS	Виклик зовнішнього коду
CALLCODE, DELEGATECALL	700 + внутрішній GAS	Делегованих виклик або Proxy
RETURN	0	Повернення результату
REVERT	0	Зупинка виконання, відхилення змінів стану

		блокчейну

Таблиця 1.1 – визначення складності транзакцій в блокчейні.

SSTORE – запис не нульового значення в пустий слот сховища 21000 GAS. Зміна існуючого значення на не нульове – 5000 GAS. Очищення значення на нульове – 20000 GAS повертається.

2.3 МЕТА ДОСЛІДЖЕННЯ

Мета дослідження – долідити транзакції на предмет наявності патернів, алгоритмів та закономірностей які можна використати для визначення транзакції як фінансової та використовувати в системі моніторингу фінансових транзакцій. Таким чином для проведення аналізу даних, побудови звітностей та формулювання гіпозез використовується технологія OLAP, в той час як для визначення нових патернів використовуються технології Data Mining а для застосування цих патернів та гіпотез в системі використовується технологія Machine Learning.

2.4 НАУКОВА НОВИЗНА

Наукова новизна – для досягнення поставленої мети вперше було досліджено транзакції в Ethereum блокчейні, на предмет наявності патернів та закономірностей, що дають змогу класифікувати транзакції за типом, а саме визначати фінансові операції. Також вперше було розроблено програмний модуль, алгоритм якого базуючись на побудованій Machine Learning моделі передбачав тип транзакції, відштовхуючись від даних транзакції, а саме кількості GAS що потребувала транзакція для обробки, а також типі конкретної криптовалюти.

2.5 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Etherem блокчейн, система що дає можливість обробки транзакцій шляхом виконання розуміних контрактів на децентралізованій віртуальній машині.

Основні ролі в Ethereum блокчейні:

Роль	Основні операції	Опис
Користувач	• Відправка транзакцій • Взаємодія з розумними контрактами	Індивідуальні особи або програми, які взаємодіють з блокчейном
Валідатори	• Забезпечують безпеку мережі • Валідують транзакції та додають блоки в блокчейн	Пропонують, а також валідують нові блоки
Розробник	• Створення розумних контрактів використовуючи мову Solidity • Розробка програмних рішень для покращення роботи мережі вцілому	Індивідуальні особи або команди що будують децентралізовані застосунки, розумні контракти або блокчейн інфраструктуру

Оператори Нод	• Забезпечують децентралізацію • Забезпечують доступність в різних регіонах • Забезпечують надійність шляхом резервного копіювання • Валідують транзакції в блоках на предмет помилки.	Запущені клієнти Ethereum мережі для підтримки копій блокчейну
Децентралізовані застосунки	• Забезпечують розвиток блокчейн екосистеми. • Забезпечують гнучкість при взаємодії з блокчейном.	Застосунки що складаються з одного або більше розумних контрактів, що розгорнуті в блокчейні та виконуються в рамках обробки транзакції.

Таблиця 2.1 – ролі та компонетни в Ethereum блокчейні

Як працює мережа – при взаємодії з системою блокчейн, користувач або програма ініціює транзакцію, формується певний набір даних для обробки транзакції, які складаються з трьох основних компонентів: метадані, кеш та дані запиту. Метадані характеризують транзакцію що дає можливість в подальшому легше індексувати, валідувати, приорітезовувати та виконувати транзакцію. Кеш –

дані що були доступні шляхом виконання попередніх транзакцій та можуть бути корисними при виконанні поточних, не є обов'язковими, проте можуть позитивно вплинути на ціну за виконання транзакції. І власне дані запиту – дані що включають в себе адресу розумного контракту та дані що потребує розумний контракт для виконня. Ці дані підписуються цифровим підписом використовуючи приватний ключ. Після підписання транзакції цифровий підпис також включається в транзакцію.

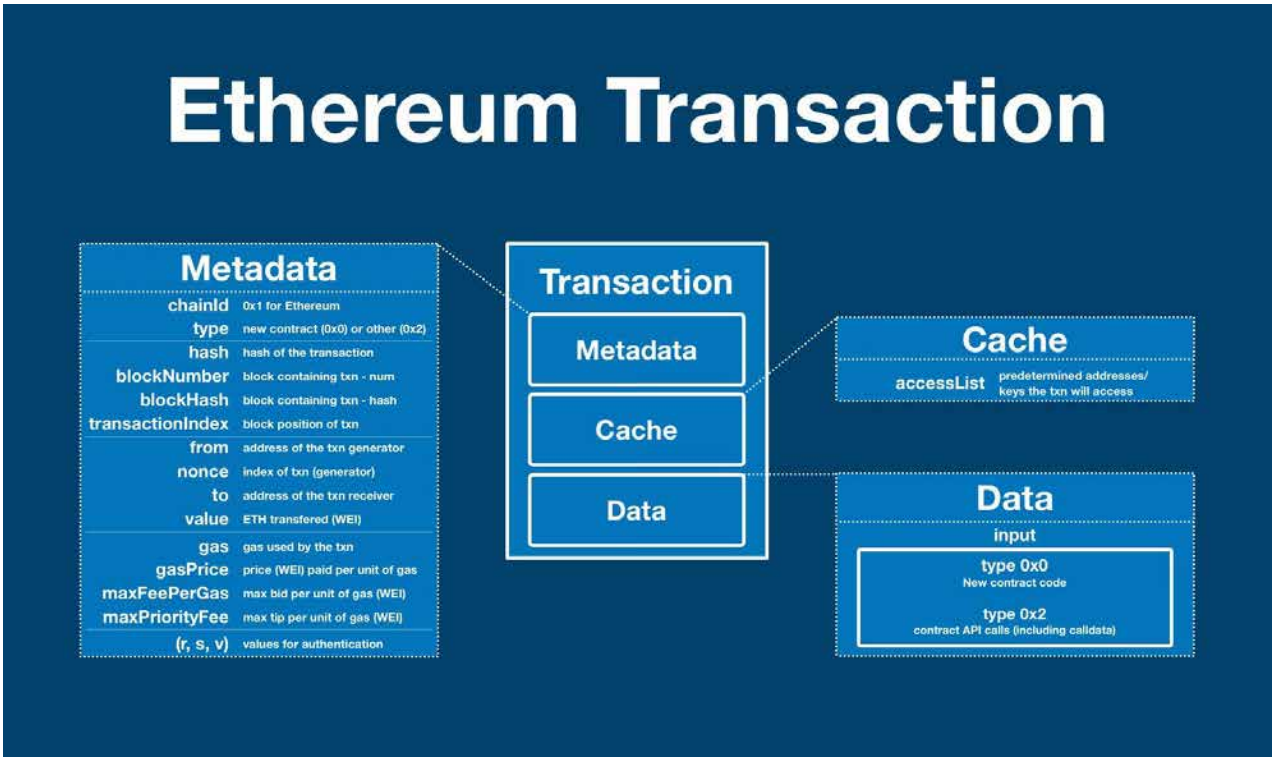


Рис. 2.1 – структура транзакції в Ethereum блокчейні

Для підписання транзакції використовується криптографія еліптичної кривої, яка передбачає наявність публічного та приватного ключа. Частіше за все використовуєтьс для шифрування веб трафіку асиметричним способом. Таким чином в блокчейні, маючи доступ до приватних ключів гаманця забезпечує повний доступ до гаманця.

Після формування та підписання транзакції – вона відправляється на будь-яку доступну ноду. Нода в свою чергу поширює цю транзакцію в мережу, відправляючи її в «пул транзакцій» який в свою чергу бачать інші учасники мережі.

Після потрапляння транзакції в мережу, валідатори та оператори нод за допомогою криптографії перевіряють цифровий підпис на предмет того що він був зроблений приватним ключем, що являється парним ключем для публічного ключа гаманця що ініціював і відправив транзакцію в мережу, таким чином нівелюється можливість відправки транзакції під видом іншого користувача.

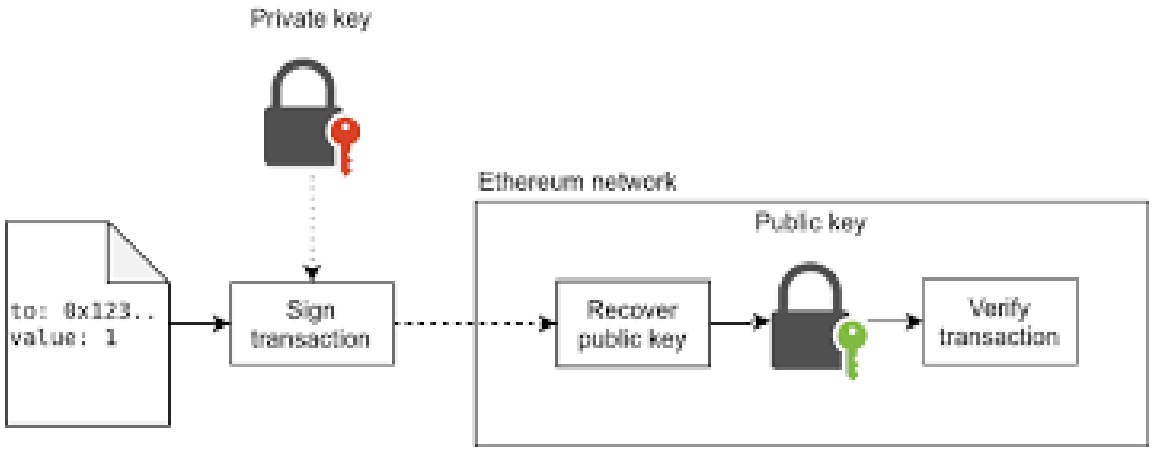


Рис. 2.2 – процес ініціації та відправки транзакції в мережу Ethereum

Валідатори вибирають транзакції з «пулу транзакцій». Метадані транзакції включають ціну за виконання, таким чином транзакції вибираються шляхом максимізації прибутку. Далі обробка транзакції, яка включає перевірку метаданих, перевірку балансу необхідного для сплати за виконання транзакції та переказів в рамках транзакції, також йде перевірка на те чи вказана ціна та розмір GAS відповідає обсягу робіт що потребується для виконання. Транзакції що не відповідають цим вимогам – відхиляються мережею. Далі транзакції додаються в новостворений блок, інші транзакцію також включаються в блок доки не

досягається обмеження по кількості транзакцій або по об’єму робіт який необхідно виконати, тобто сумарний GAS.

Виконання транзакції передбачає собою виконання розумного контракту – програмного коду розгорнутого в блокчейні. Програмний код може включати перевірки, читання та запис в сховище, а також взаємодію з іншими розумними контрактами. При виникненні передбаченої або непередбаченої помилки транзакція відхиляється повністтю, включаючи зміни в сховищі, проте так як обробка вже відбувалась, хоч і не завершилась успішно – плата за обробку знімається з користувача який відправив транзакцію, плата включає в себе обсяг робіт який було виконано до виникнення помилки.

Після створення та формування нового блоку – валідатор відправляє іншим валідаторам для перевірки на правильність, як тільки блок досягає консенсусу – він додається в блокчейн. При виявленні помилок або фальсифікацій, депозит що був внесений валідатором частково спалюється і частково розподіляється між іншими валідаторами.

Після включення транзакції блок а блоку в блокчейн формуються деталі транзакції, який включає в себе основну інформацію таку як:

- Статус транзакції – статус може бути успішний або не успішний.
- GAS що був використаний для виконання транзакції
- Логи – перелік подій що виникли під час виконання розумного контракту
- Хеш транзакції та деталі блоку в рамках якого вона була виконана

Деталі транзакції можуть бути переглянути на безпосередньо в блокчейні або на таких оглядачах блокчейну як до прикладу Etherscan, що надають можливість зручного перегляду розширеної інформації про транзакції та блоки.

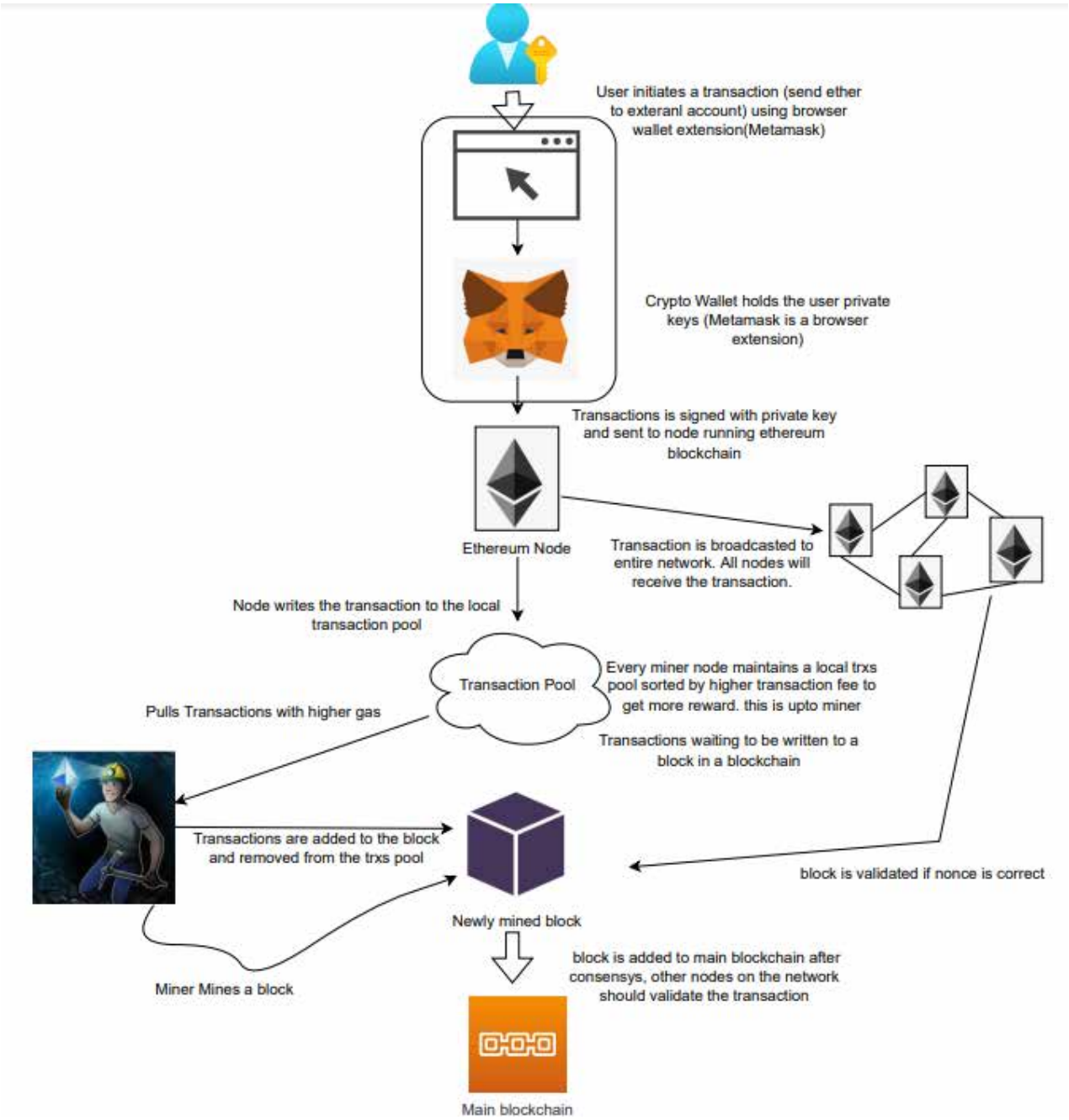


Рис. 2.3 – загальна схема роботи Ethereum мережі.

2.6 ПОСТАНОВКА ЗАВДАННЯ

В рамках магістерської роботи потрібно дослідити Ethereum блокчейн, визначити патерни та закономірності для виявлення фінансових транзакцій. Дослідити можливість та спроектувати систему яка дасть змогу проводити

моніторинг фінансових транзакцій в Ethereum блокчейні. Система повинна відповідати на наступні запитання:

- Який сумарний обіг криптовалют був для конкретного гаманця за визначений проміжок часу
- Який сумарний обіг криптовалют був в конкретному регіоні за визначений проміжок часу
- Яка найпопулярніша криптовалюта в розрізі конкретного гаманця
- Яка найпопулярніша криптовалюта в розрізі визначеного регіону

2.7 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Досліджуючи питання фінансового моніторингу в рамках Ethereum блокчейн я наткнувся на такі існуючі рішення:

DeBank – веб-сайт який дає можливість переглянути наявні активи на гаманці.

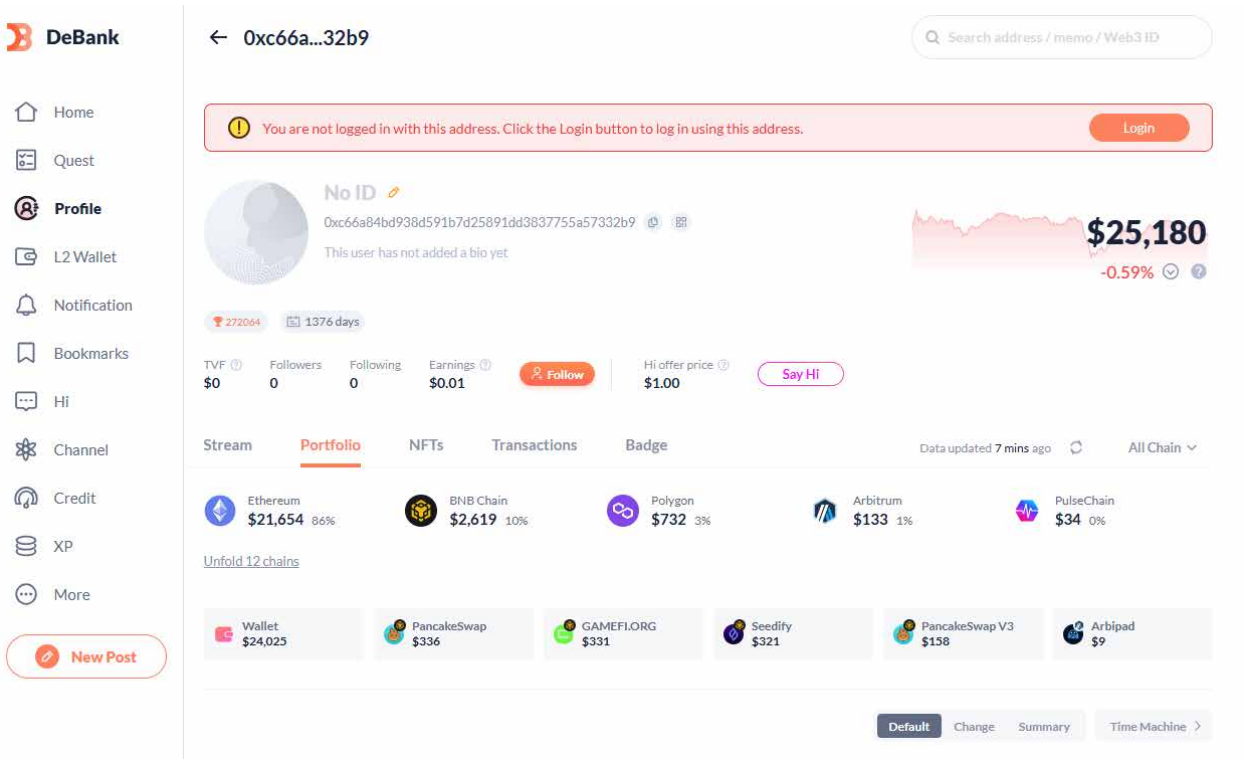


Рис. 2.4 – інтерфейс веб-застосунку Debank

DeBank – дає можливість швидко та зручно переглядати наявні активи, дізнаватись їхню актуальну ціну, підтримує інші EVM сумісні блокчейн мережі. Проте він не дає можливість переглядати аналітичні та статистичні дані в історичній перспективі. Крім того наявність активів ніяк не вказує на фінансові транзакції і на дату та час коли цей актив був куплений/проданий. Таким чином це рішення більше підходить для деталізованого огляду поточного стану гаманця, ніж для фінансового моніторингу.

DexTools – веб застосунок який має доволі широкий функціонал. Це сервіс дає можливість в режимі реального часу бачити нові торгові пари, які відразу характеризуються пристутністю на децентралізованих біржах, наявністю безпекового аудиту коду розумного контракту, кількістю ліквідності та можливими вразливостями в коді розумного контракту.

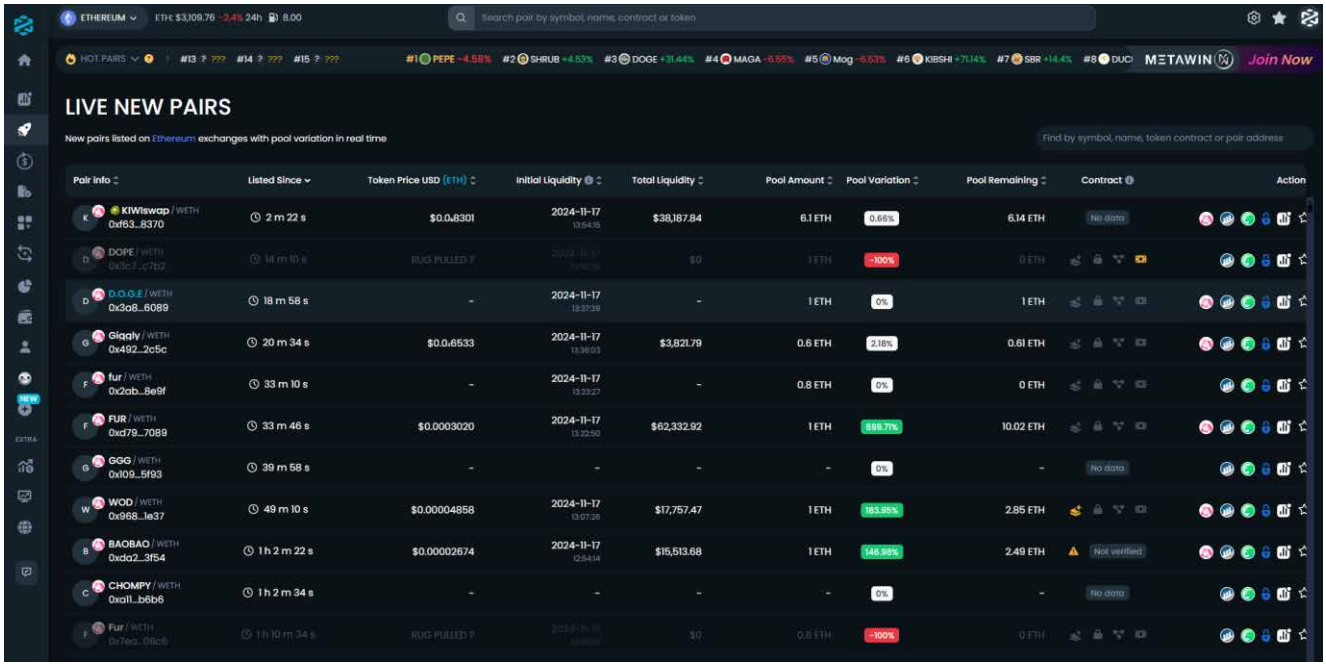


Рис. 2.5 – веб інтерфейс сервісу DexTools (торгові пари в режимі реального часу)

Сервіс також дає доволі обширну аналітику в історичній перспективі в розрізі торгових пар в розрізі приросту ціни за 24 години, в розрізі об’єму торгів, в розрізі зниження ціни та інше.

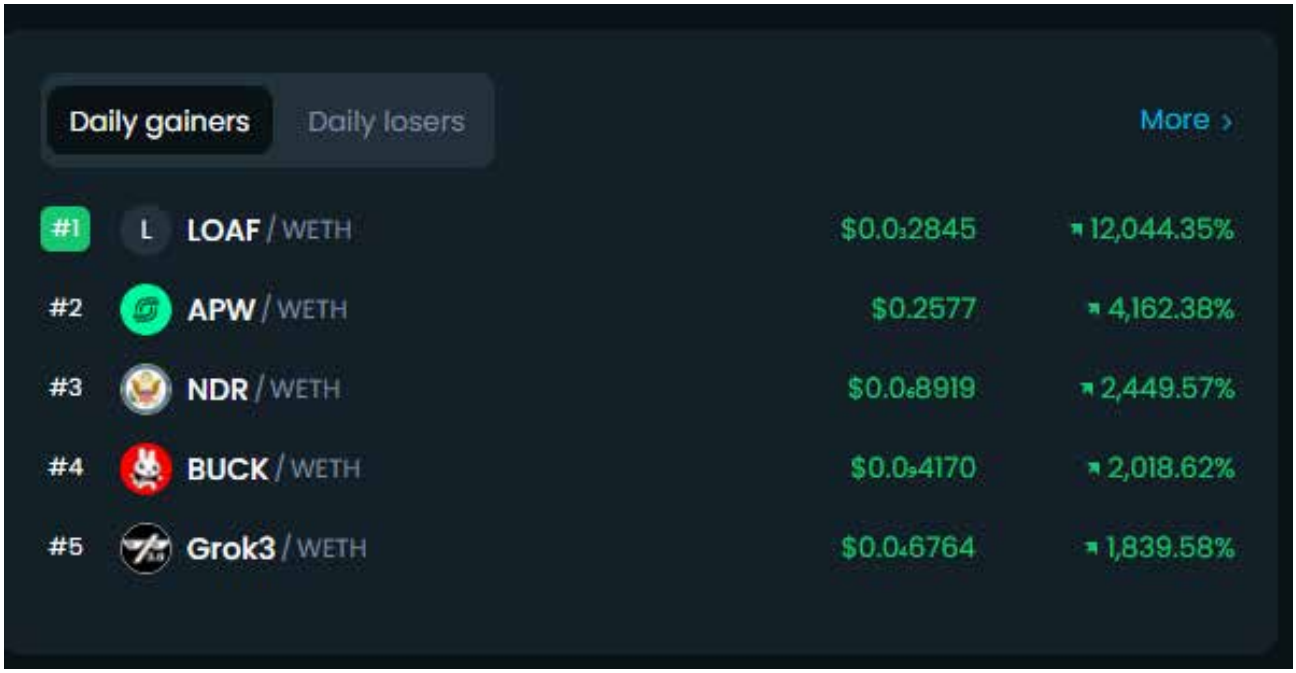


Рис. 2.6 – веб інтерфейс сервісу DexTools (лідери приросту за 24 години)

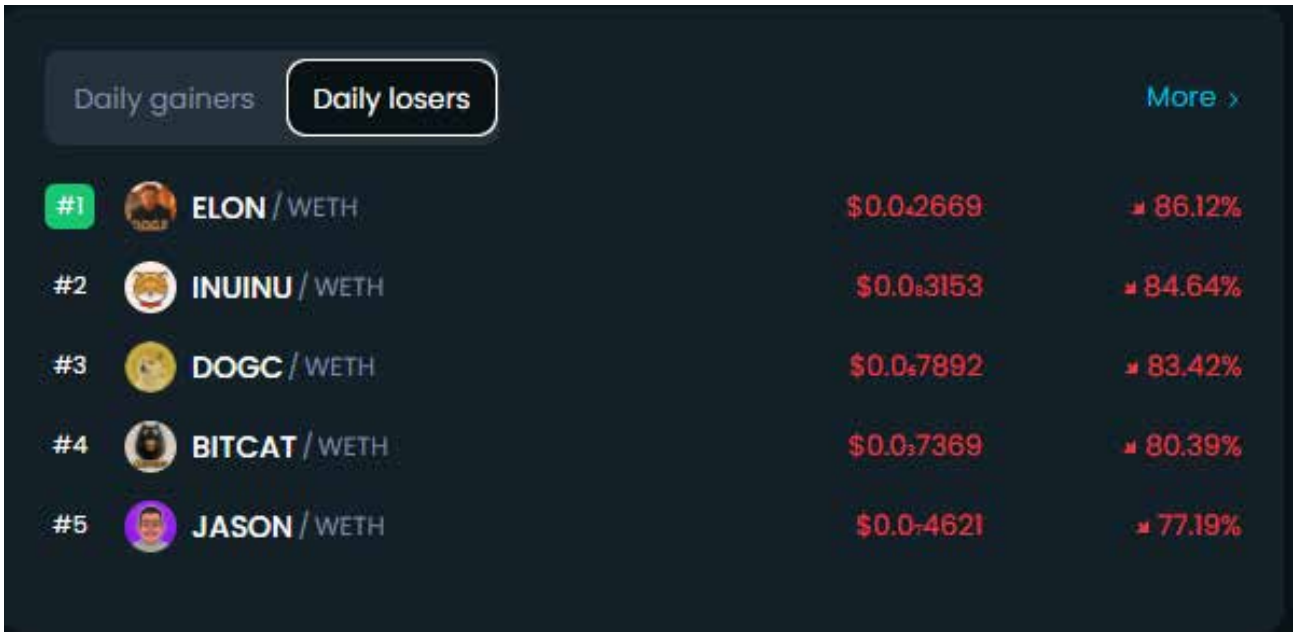


Рис 2.7 – веб інтерфейс сервісу DexTool (лідери за зниженням ціни за 24 години)

Проте наявність доволі обширного функціоналу не покриває задачі фінансового моніторингу в розрізі конкретного гаманця. Це рішення може бути використано як приклад, на основі якого можна робити систему моніторингу, адже кількість інформація яку збирає цей сервіс і кількість аналітики яку він надає дійсно вражає, проте його основна ціль – дати як можна більше корисної інформації в розрізі торгових пар, в той час як для нас важливим є статистика і аналітика в розрізі гаманця.

3. МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Під час моделювання системи я керувався функціональним та об’єктно орієнтованим підходом. Ці підходи мають різні принципи, кожен має свої переваги та недоліки, але обидва мають за мету одну ціль – показати систему в розрізі її основних компонентів. Для початку розлянемо принципи, переваги та недоліки обох

3.1 ОСНОВНІ ПРИНЦИПИ МОДЕЛЮВАННЯ

Функціональний підхід основні принципи:

- Орієнтується на обробку даних через функції, які є математичним перетворенням, тобто описує систему з точки зору одиниці роботи «функції»
- Дані є незмінними, тобто функції не змінюють передані аргументи, а натомість створюють нові дані на основі вхідних
- Уникає побічних ефектів, функції залежать тільки від вхідних даних, і не впливають на зовнішній стан при поверненні результату

Функціональний підхід в розрізі моделювання:

- Система розбивається на функції, кожна з яких виконує певну задачу

- Дані передаються між функціями що робить потік обчислень явним і передбачуваним
- Складні задачі вирішуються через комбінування простих функцій
- Наприклад, для моделювання фінансової системи функції можуть представляти трансформації транзакцій, балансів тощо.

Переваги функціонального підходу:

- Простота тестування – кожна функція може бути протестована ізольовано
- Легко відслідкувати потік даних
- Зрозумілість і передбачуваність
- Зручність при роботі з великими обчислювальними задачами

Недоліки функціонального підходу:

- Важко моделювати складні системи, де є багато взаємодій об'єктів або станів.
- Менш природний для опису реального світу, де об'єкти змінюють свої стани.
- Для початківців може бути складно зрозуміти концепції, як-от імунітет даних чи композиція.

Об’єктно орієнтований підхід, основні принципи:

- Орієнтований на моделювання світу через об'єкти – екземпляри класів, які об'єднують дані (властивості) та поведінку (методи).
- Визначає такі концепції, як інкапсуляція, успадкування та поліморфізм.
- Акцент на стані об'єкта, який може змінюватися протягом життєвого циклу системи.

Об’єктно орієнтований підхід в розрізі моделювання:

- Система розбивається на об’єкти, що відповідають реальним сутностям або абстракціям.
- Об’єкти взаємодіють між собою через методи, змінюючи свої стани.
- Наприклад, для моделювання банківської системи можна створити класи для акаунтів, транзакцій, клієнтів.

Переваги об’єктно орієнтованого підходу:

- Природний спосіб опису реального світу (наприклад, об'єкт = автомобіль із властивостями «модель», «швидкість» і методами «рухатися», «зупинитися»).
- Сприяє повторному використанню коду через спадкування.
- Зручний для роботи з великими та складними системами, що мають багато взаємодій.

Недоліки об’єктно орієнтованого підходу:

- Складність для новачків через концепції, як-от поліморфізм чи інкапсуляція.
- Може спричиняти надмірну абстракцію, що ускладнює код.
- Важче відстежувати потік виконання, оскільки стан розподілений між багатьма об'єктами.

Порівняння підходів у використанні:

Аспект	Функціональний підхід	Об’єктно орієнтований підхід
Моделювання даних	Набір функцій, які обробляють дані	Об’єкти, що інкапсулюють дані й поведінку
Управління станом	Стан передається через	Стан зберігається та

	функції	змінюється в об'єктах
Повторне використання	Функції можна використовувати в композиціях	Реалізується через успадкування та класи
Простота тестування	Легко тестувати функції	Потрібно враховувати стан об'єктів
Природність моделювання	Менш інтуїтивний для реального світу	Легко співвіднести з реальними сутностями

Таблиця 3.1 – порівняння функціонального та об’єктно орієнтованого підходів

Таким чином для моделювання системи, можна використовувати обидва, проте варто зауважити, що для моделювання системи моніторингу фінансових транзакцій вцілому, варто використовувати об’єктно орієнтований підхід, так як він дає можливість враховувати взаємодії між сутностями, а також дає можливість багаторазового використання коду, враховуючи спадкування.

З іншої сторони функціональний підхід варто використовувати для моделювання функціоналу обробки даних. Це дає можливість краще зрозуміти основні операції які потрібно виконати, легко їх протестувати та мати чіткий потік даних.

3.2 ОСНОВНІ СКЛАДОВІ СИСТЕМИ

Transaction – сутність яка описує транзакцію в системі Ethereum блокчейн, містить в собі детально інформацію про те звідки, куди, коли і в якій кількості було відправлено.

User – сутність яка описує роль користувача Ethereum блокчейн, містить в собі інформацію про користувача, а саме адресу гаманця, а також додаткову інформацію для ідентифікації особи.

Token – сутність яка описує конкретну криптовалюту, містить інформацію про адресу розумного контракту в блокчейні, а також додаткову інформацію для можливості коректної обробки числових значень.

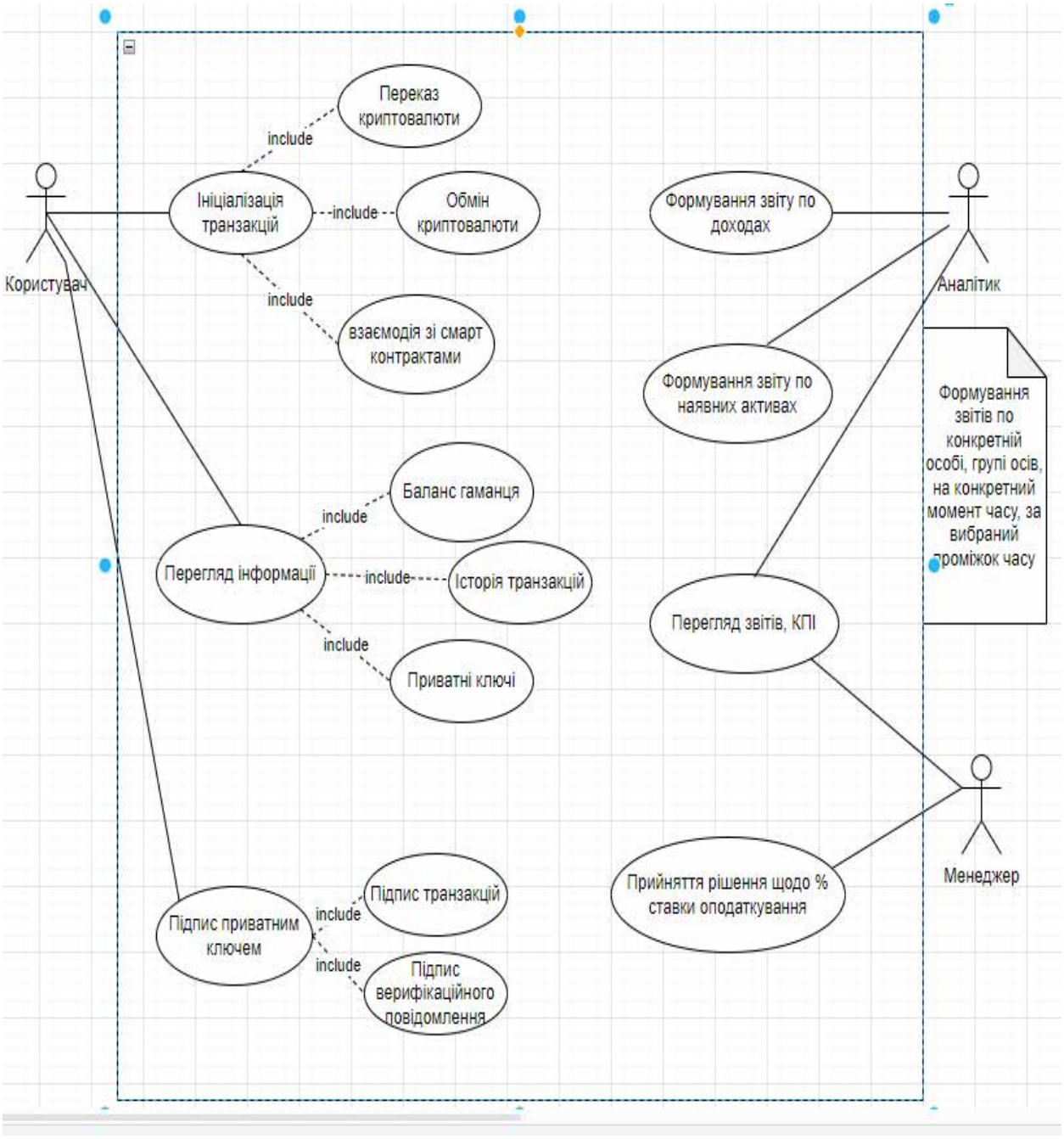


Рис 3.1 – діаграма прецидентів системи.

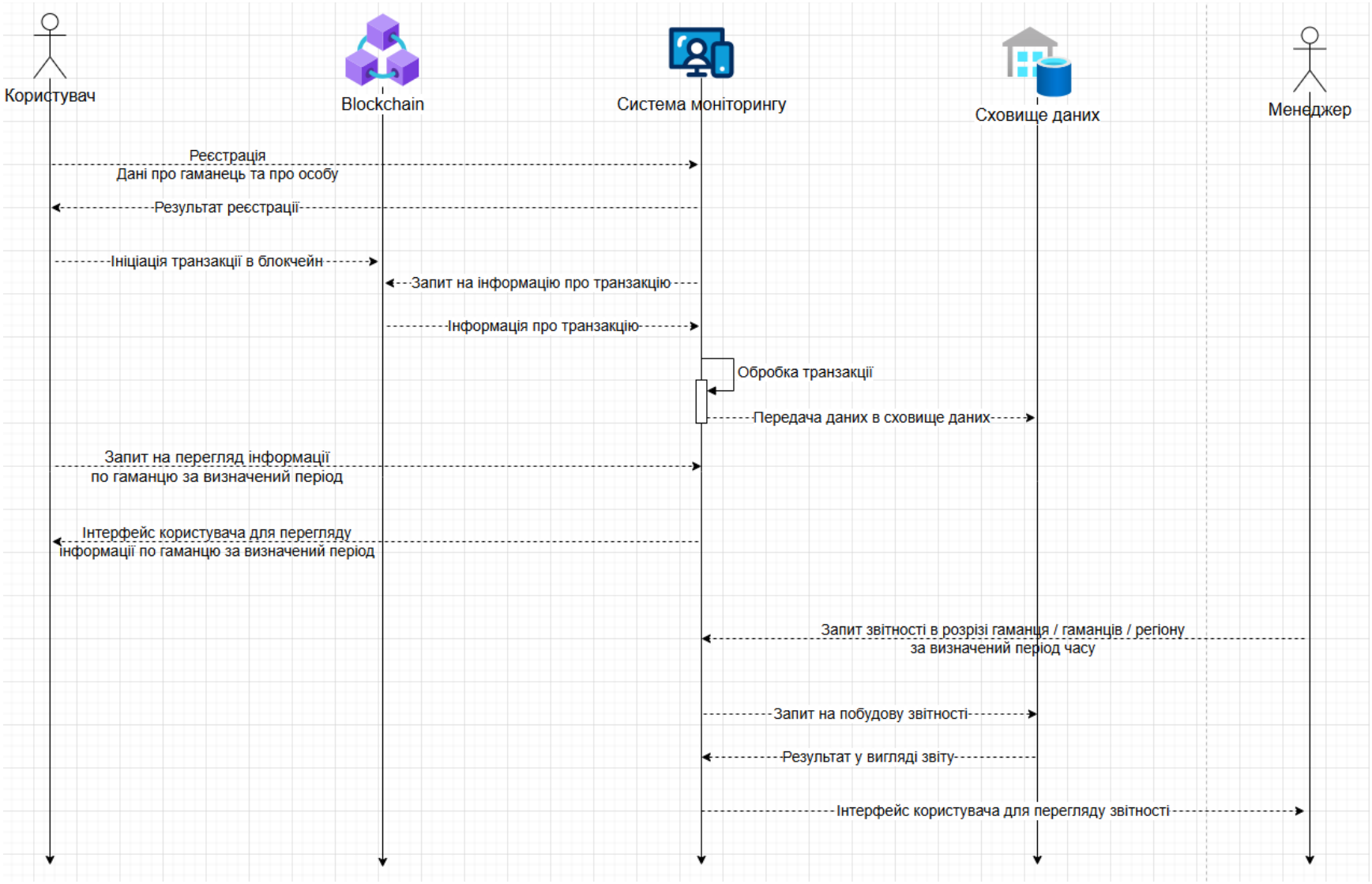


Рис 3.2 – Діаграма послідовності системи (високорівнева)

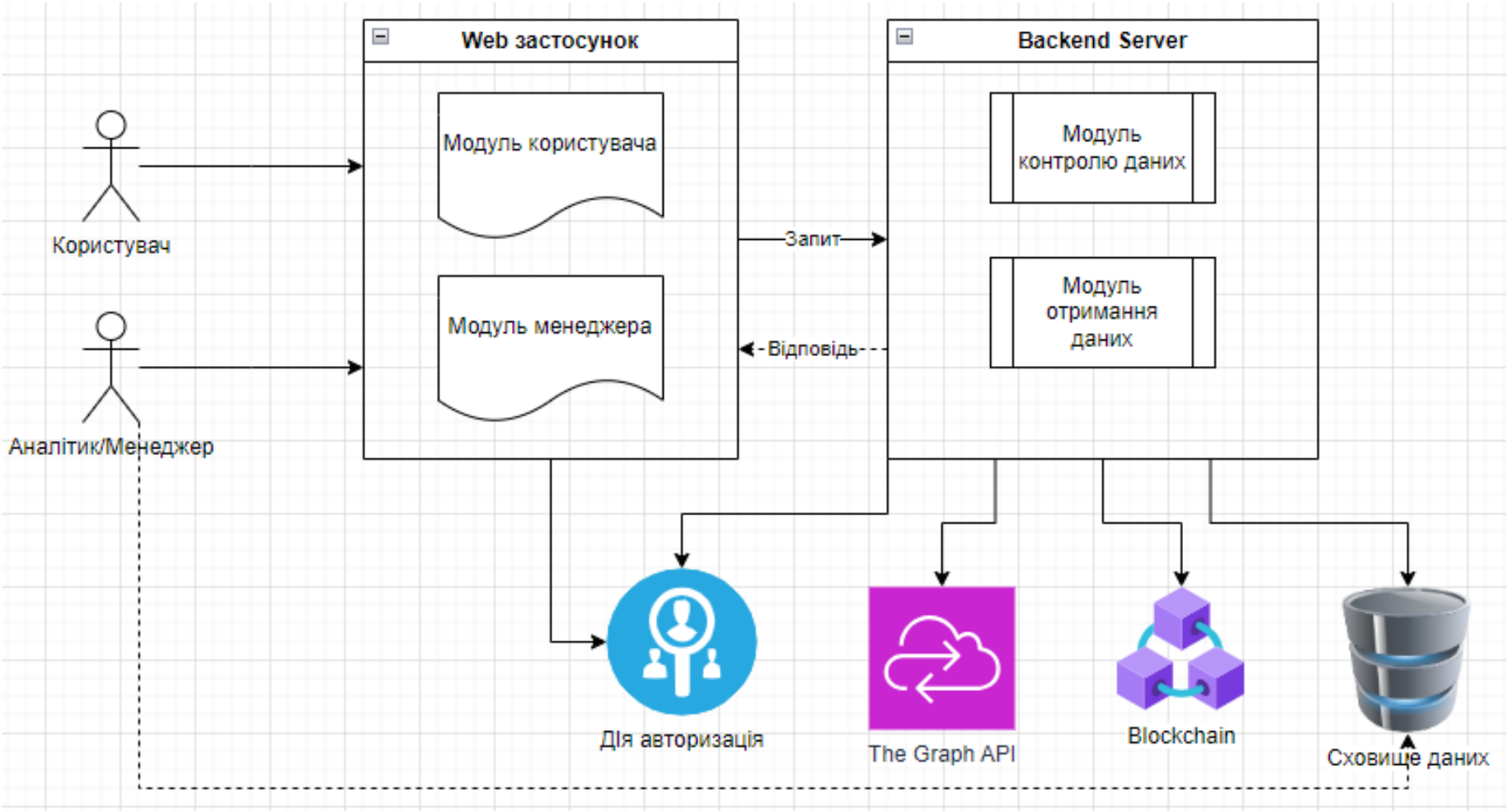


Рис 3.3 – Топологія системи

4. РОЗРОБКА СИСТЕМИ

Під час розробки системи потрібно було зпроектувати архітектуру системи, визначитись з інструментами та засобами для реалізації та взаємодії окремих програмних модулів між собою.

Розглянемо основні програмні модулі та сторонні сервіси які потрібно зпроектувати та реалізувати, або забезпечити інтеграцію з ними:

- **Клієнтський веб застосунок, або інтерфейс користувача** – цей модуль напряму взаємодіє лише з веб сервером, використовуючи REST API як підхід до комунікації клієнт-сервер. Також клієнтський застосунок має реалізовувати інтеграцію з криптовалютним гаманцем, для можливості генерації цифрового підпису, який в подальшому сервер повинен валідувати.
- **Веб сервер** – основний програмний модуль, що відповідає за коректну обробку інформаційного потоку, передачі даних в сховище даних, обробку запитів як звичайних користувачів так і менеджерів, а також, інтеграцію з сторонніми сервісами для отримання додаткової інформації яка потрібна для деталізованої статистики або аналізу.
- **Ethereum блокчейн** – існуюча мережа, на базі якої проводиться фінансовий моніторинг. Система не підлягає будь-якому втручанню, а використовується лише в рамках інтеграції для читання інформації.
- **Сховище даних** – сховище даних на базі **OLAP** з підтримкою програмних модулів **SSAS, SSIS, SSRS.** Це модуль потрібен для зберігання великих об’ємів даних та проведення аналізу даних, будування звітностей, тощо.
- **Moralis** – сторонній сервіс, який дає більш зручний доступ до інформації в блокчейні. Сервіс індексує інформацію в блокчейні на своїй стороні, а також надає функціонал «Webhook» для сповіщення системи про певну подію в блокчейні.

- **CoinMarketCap** – сторонній сервіс, який надає деталізовану інформацію про ціни на криптовалюти, з цим сервісом взаємодіє веб сервер, для отримання необхідної інформації
- **Binance API** - сторонній веб сервіс, який також надає інформацію на ціни про криптовалюти, з ним взаємодіє веб сервер для отримання інформації ще на етапі виявлення закономірностей і патернів.
- **Machine Learning модель** – програмний модуль який відповідає за прогнозування типу транзакції відштовхуючись від вхідних даних

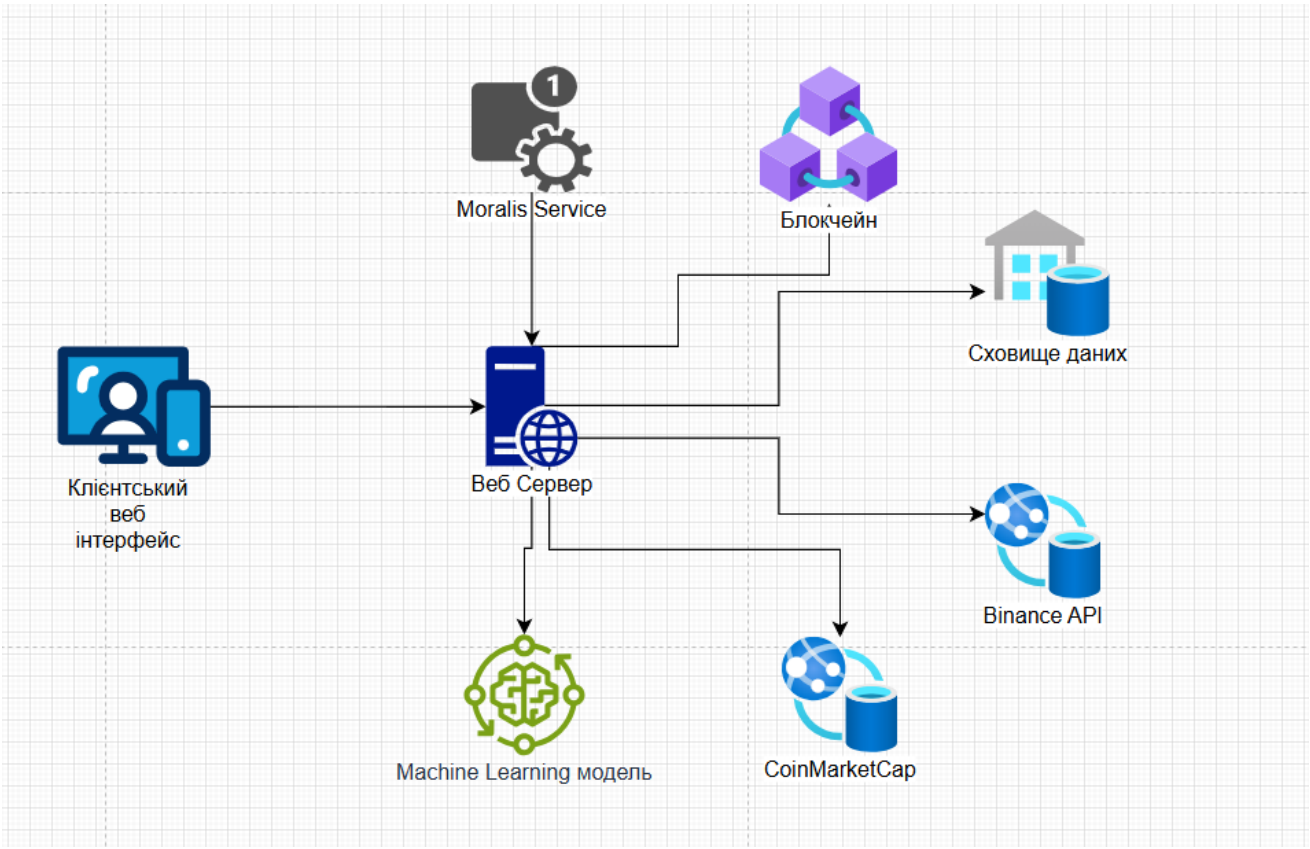


Рис 4.1 – Діграма взаємодії між програмними модулями.

Таким чином для реалізації системи потрібно визначитись з іструментами та засобами. Так як кожен модуль виконує окремий тип задача – підповідно кожен потребує власних інструментів і засобів для розробки.

Розглянемо інструменти та засоби які були використані для розробки кожного окремого програмного модуля.

Клієнтський веб інтерфейс – для реалізації клієнтського веб інтерфейсу я використовував React JS, та мову програмування Typescript. React JS являється легким Javascript фреймворком, він дає можливість швидко та зручно будувати веб інтерфейс користувача, легко інтегруватись з веб сервером для комунікації через REST API, а також на його базі є багато UI бібліотек які вже реалізують основні компоненти інтерфейсу такі як кнопки, поля вводу, таблиці, тощо. Я для своїх задач обрав бібліотеку Material UI. Ця бібліотека має велику кількість готових компонентів, які легко адаптуються і видозмінюються, проте містять в собі весь необхідний функціонал, який часто використовується, як от валідація полів вводу, адаптація полів виводу під розмір екрану, тощо.

З точки зору інструментів розробки я обрав Visual Studio Code – легка IDE, яка дає можливість працювати з програмним кодом на різних мовах, містить інтеграцію з GitHub, а також розширення для інтеграції з іншими сторонніми сервісами і системами як наприклад AWS, Azure, Copilot.

Веб сервер – для реалізації серверної частини я обрав мову програмування C# на базі .NET 6. На даний момент найновіша версія .NET являється восьма, проте шосту я обрав не просто так, у зв’язку з моїм бажанням розгорнути веб сервер на AWS платформі я ознайомився з документацією сервісу, і дізнався що Lambda функції підтримують виконання .NET 6 і нижче. Тому на даний момент потрібно було відштовхуватись від цього. Загалом C# являється об’єктно орієнтованою мовою програмування, що є доволі зручним для проектування системи базуючись на результатах моделювання, особливо що стосується взаємодії між сутностями. В якості засобів взаємодії з базою даних та сховищем даних я обрав Entity Framework – ORM система яка дає можливість описувати сутності

бази даних у вигляді коду використовуючи об’єктно орієнтований підхід, визначати взаємодії між сутностями в програмному коду а також керувати міграціями структури бази даних засобами коду.

В якості оперативної бази даних я обрав PostgreSQL – реляційна база даних, має досить високий рівень продуктивності і являється дешевшою за MS SQL Server у зв’язку з відсутністю необхідності придбання ліцензії на використання

В якості інструментів я обрав Visual Studio – як нативний інструмент від Microsoft, ця IDE дає можливість легко орієнтуватись в коді, зручно маніпулювати версіями залежностей, версіями .NET, а також має можливість інтеграції з базами даних що часто пришвидшує розробку і нівелює необхідність додаткового ПЗ для взаємодії з базою даних.

Сховище даних – для реалізації сховища даних було використано SQL Server Analytical Services. Ця система дає змогу будувати гіперкуб з необхідними вимірами, а також з додатковими модулями такими як SQL Server Integration Services та SQL Server Reporting Services налаштовувати потік даних від операційної бази даних та в подальшому побудову репортів, KPI та застосовувати підходи Data Mining для виявлення нових закономірностей.

В якості інструментів було використано SQL Server, SQL Server Management Tools а також Visual Studio

Machine Learning модель – цей програмний модуль було розроблено використовуючи мову програмування Python. Мова Python являється однією з найбільш популярний в сфері Machine Learning та Data Mining. На базі цієї мови є багато готових рішень для алгоритмів машинного навчання, аналізу даних, візуалізації даних та інше. Ще одна перевага Python в тому що мова передбачає використання функціонального підходу, що являється ключовим в проектуванні

ML моделей оскільки потребує багато математичний обчислень та зрозумілого потоку даних.

В якості інструментів я обрав все той же Visual Studio, а також для деяких задач Jupiter Notebook. Jupiter Notebook дає можливість створення середовища під конкретну задачу, це середовище можна налаштувати для використання конкретної версії Python, а також конкретної версії залежностей, і усуває потенційні конфлікти версій що є необхідним для використання конкретних бібліотек .

4.1 РОЗРОБКА АВТОРИЗАЦІЇ ТА РЕЄСТРАЦІЇ

Для авторизації та реєстрації користувача потрібно зпроектувати 2 частини:

- Авторизація через гаманець та підтвердження власності гаманця
- Авторизація через портал Дія та отримання персональних даних користувача

Для авторизації через гаманець необхідно отримати на серверній частині адресу гаманця а також цифровий підпис гаманця, для підтвердження власності гаманця. З сторони клієнтського веб інтерфейсу необхідно реалізувати інтеграцію з криптовалютним гаманцем для отримання цифрового підпису.

Таким чином потік виконання буде виглядати наступним чином – клієнтський веб інтерфейс передає серверу запит на авторизацію, для цього надається адреса гаманця. Сервер генерує випадковий рядок символів та зберігає в оперативну базу даних пару адреса гаманця та випадковий рядок символів і у відповідь надсилає клієнтському веб інтерфейсу цей рядок символів.

Клієнтський інтерфейс передає запит підпису рядка символів на криптовалютний гаманець, далі користувач підтверджує підпис вже в рамках гаманця, після чого результат у вигляді цифрового підпису повертається клієнту.

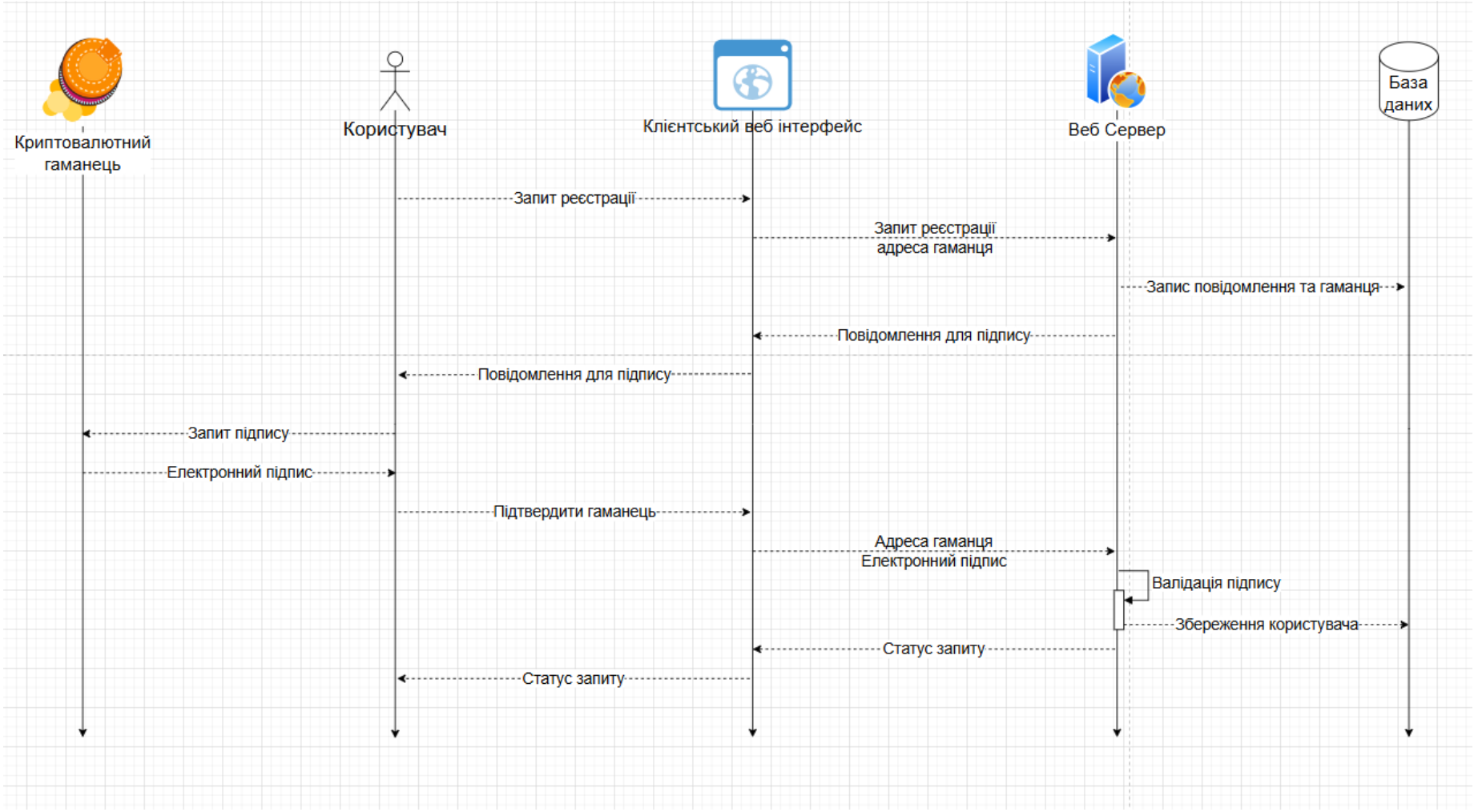


Рис. 4.2 – Діаграма послідовності авторизації/реєстрації користувача через гаманець

Клієнтський веб інтерфейс передає веб серверу цифровий підпис а також адресу гаманця. Сервер дістає з оперативної бази даних повідомлення для відповідного гаманця, яке було збережене на попередньому етапі, далі використовуючи криптографічні алгоритми повідомлення хешується за допомогою адреси гаманця – публічний ключ, та співставляється з електронним підписом який було отримано. У разі успішного проходження валідації на веб інтерфейс користувача повертається результат у вигляді успішного статусу та авторизаційного токена, у інакшому випадку – інформація про помилку.

Для авторизація через портал Дія потрібно реалізувати OAuth2 з OIDC підходом.

OAuth2 – це відкритий стандарт авторизації який дозволяє користувачам відкривати приватні дані що зберігаються на одному сервісі іншому сервісу, без необхідності вводу авторизаційних даних на сайті для якого потрібно розкрити персональні дані. Посуті OAuth2 стандарт забезпечує процес авторизації користувача.

OIDC – це надбудова над OAuth2 яка дає можливість автентифікації користувача.

Автентифікація та авторизація: Автентифікація відповідає на питання хто являється користувачем, надає певний ідентифікатор користувача що вказує на конкретного користувача в системі, в той час як авторизація забезпечує механізми доступу, тобто дозволяє або обмежує доступ до конкретних ресурсів системи або сервісу.

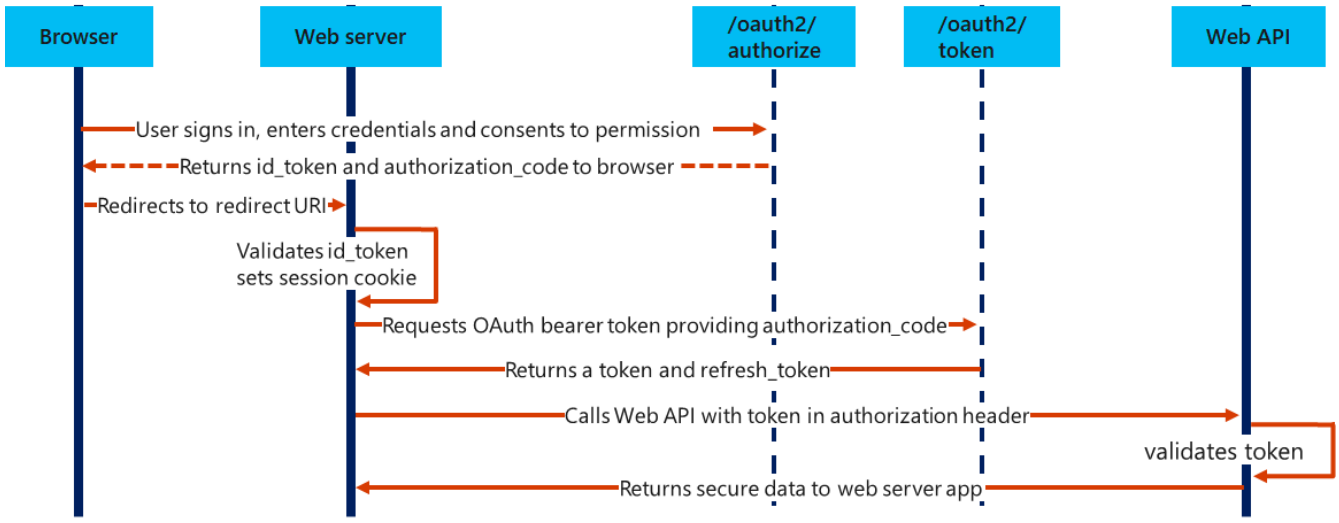


Рис 4.3 – OIDC протокол авторизації

Нажаль в процесі розробки системи виникли складнощі з інтеграцією з порталом Дія. Для цього необхідно робити запити та отримувати доступи, а сам портал наразі не має відкритого протоколу авторизації як це є наприклад в Google.

Проте частина проектування завершена і це дасть можливість імплементувати програмний продукт використовуючи поточні напрацювання та досягти бажаного результату.

4.2 РОЗРОБКА СИНХРОНІЗАЦІЇ З БЛОКЧЕЙНОМ

Для синхронізації з блокчейном я використав сторонній сервіс Moralis. Цей сервіс проводить індексацію блокчейна на своїй стороні, зберігає дані у більш оптимальному для читання вигляді і дає можливість запитувати інформацію використовуючи різні фільтри.

В своїй роботі я звернув увагу на функціонал оповіщення про події в блокчейні на базі «Webhook». Це підхід який передбачає надсилання на API веб сервера інформації про якусь подію в блокчейні при її виникненні.

Це значно оптимізує функціонал системи, адже не потрібно мати свою операційну ноду з швидким доступом до всього блокчейну, а також не потрібно перебирати всі транзакції в блокчейні та фільтрувати ті які цікавлять, натомість можна визначити критерії та події які цікавлять і отримувати сповіщення про них через точку входу API.

Для реалізації даного функціоналу потрібно було визначити які саме транзакції мене цікавлять та як правильно задати їх в системі Moralis. Для цього я дослідив ля цього я дослідив ERC-20 стандарт токенів, інтерфейс взаємодії з розумним контрактом, а також події які генерує цей контракт.

Таким чином було визначено що кожен переказ токенів супроводжується генерацією події Transfer.

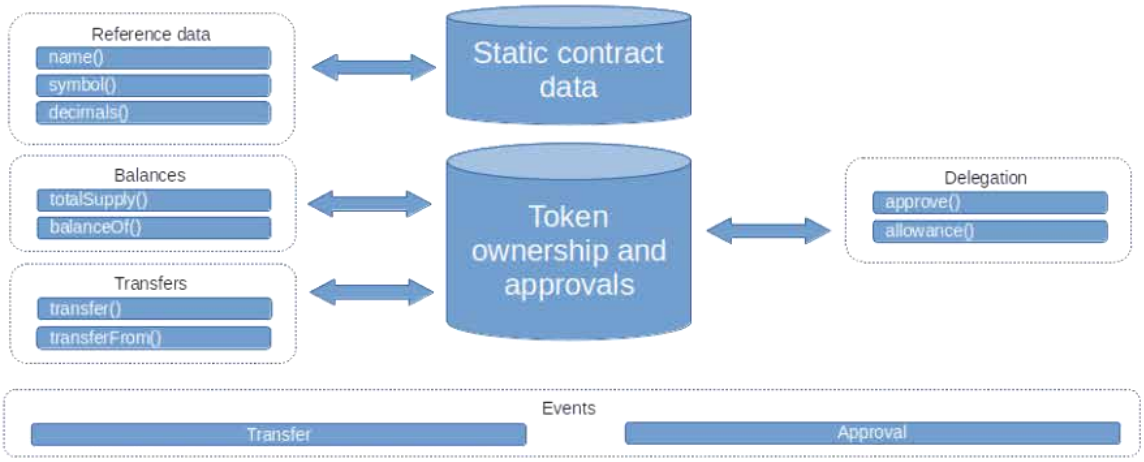


Рис. 4.4 – принцип роботи криптовалютних переказів на базі ERC-20 токенів

Далі для визначення того щоб наконфігурувати Webhook потрібно задати адресу або адреси розумного контракту, адресу або адреси гаманців для яких цікавлять сповіщення, а також інтерфейс взаємодії з розумним контрактом для визначення події що цікавить.

Для визначення інтерфейсу події використовується ABI розумного контракту, його можна знайти на сервісі Etherscan використовуючи адресу розумного контракту.

Рис. 4.5 – вигляд ABI розуміного контракту

Сам інтерфейс – це набір даних у JSON форматі які описують функції, дані та події, параметри функції, типи даних та помилок. Далі передавши цей інтерфейс в сервіс Moralis ми можемо визначити подію Transfer як ту на яку ми хочемо підписатись.

Після цього сервіс Moralis буде відправляти запит на API адресу, яка була вказана як точка доступу для Webhook, і передавати інформацію стосовно події яка виникла. В моєму випадку ця інформація включає в себе:

- Від кого був здійснений переказ
- Кому був здійснений переказ
- Яка криптовалюта була задіяна в переказі
- Яка кількість криптовалюти була задіяна в переказі

- Дата та час виконання транзакції в рамках якої був здійснений переказ
- GAS – тобто складність транзакції
- Ціна на GAS в момент обробки транзакції
- Хеш транзакції для отримання додаткової інформації безпосередньо з блокчейна за необхідності.

В рамках даного опису може виникнути питання, як гарантувати що на відкриту точку доступу API буде приходити лише сповіщення від сервісу Moralis, адже теоретично будь-хто знаючи цю адресу може відправити свій запит і тим самим повідомити систему про подію якої насправді не відбулося.

Відповідь на це питання знову дає криптографія. Тут все доволі просто. При реєстрації на платформі розробник повинен отримати секретний ключ. Це секретний ключ використовується сервісом Moralis при відправці сповіщень наступним чином. Дані що передаються у вигляді JSON рядка, доповнюються секретним ключем і хешуються, результат цього хешування передається разом з запитом в HTTP заголовку.

Веб сервер на своїй стороні, коли приймає запит від Moralis сервісу, зобов'язаний зробити аналогічну процедуру, а саме використати секретний ключ, згенерувати хеш використовуючи вищезгаданий алгоритм і перевірити чи хеш отриманий в HTTP заголовку співпадає з тим який отриманий на API точці доступу.

У разі якщо хеші співпадають система вважає запит авторизованим і приступає до обробки даних та збереженню необхідної інформації в базу даних.

У разі якщо хеші не співпадають – система повертає помилку, вказуючи на неможливість обробки запиту.

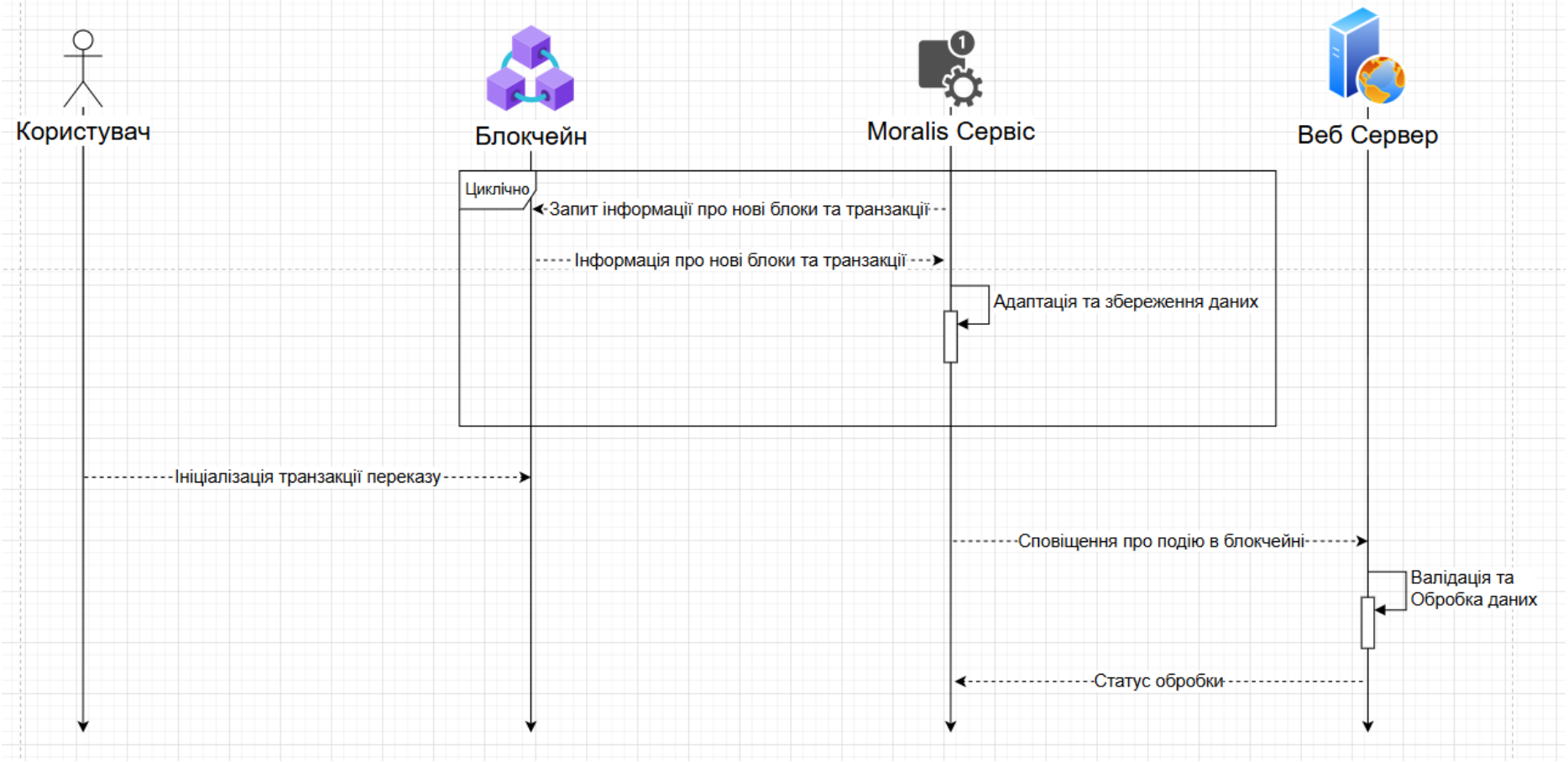


Рис. 4.6 – Діаграма послідовності, інтеграція веб серверу з сервісом Moralis.

4.3 РОЗРОБКА СТРУКТУРИ БАЗИ ДАНИХ

Під час розробки структури оперативної бази даних, було визначено та побудовано сутності та зв’язки якими оперує система.

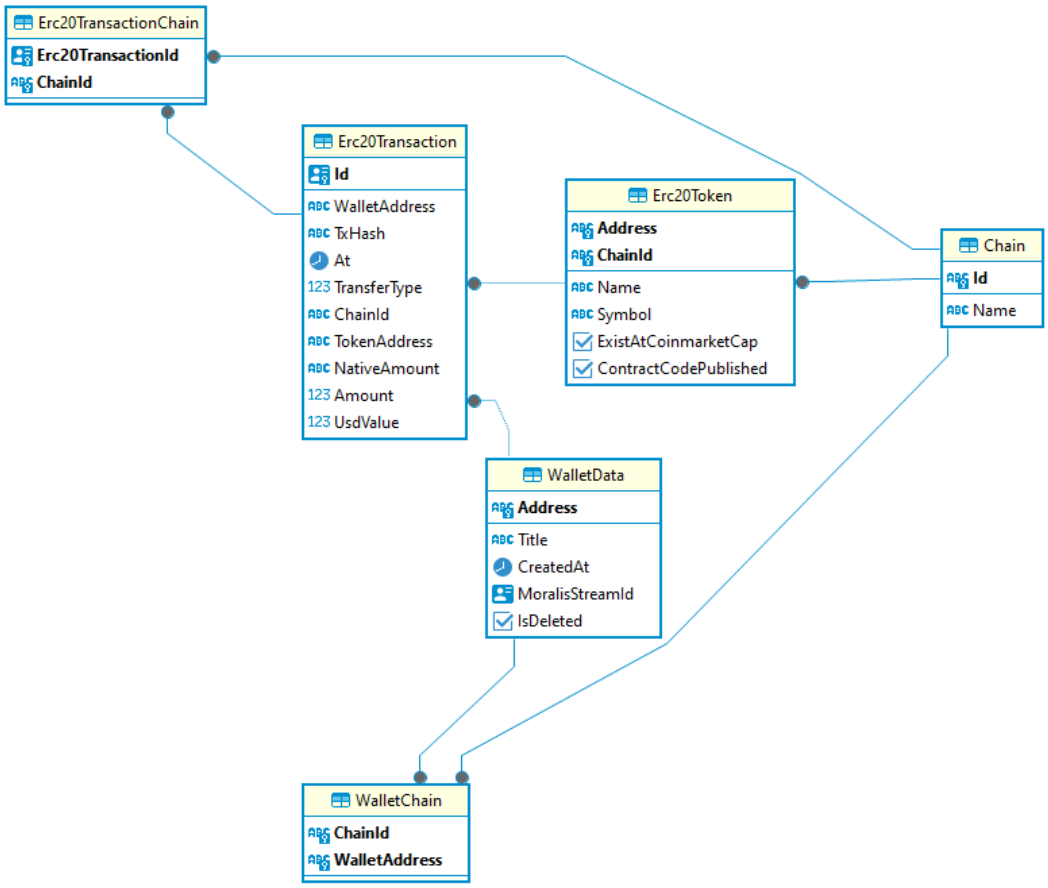


Рис. 4.7 – ER діаграма оперативної бази даних

Розглянувши діаграму на рисунку 4.7 можемо визначити 4 головних сутності:

- WalletData – таблиця яка відповідає за дані гаманця, ключем в даній таблиці виступає поле Address, яке являє собою адресу гаманця і є унікальним в розрізі блокчейну.

- Erc20Transaction – таблиця яка описує інформацію про транзакцію, містить в собі необхідні для подальшого аналізу поля, а також хеш транзакції для змоги отримати додаткову інформацію безпосередньо з блокчейну за потреби
- Chain – таблиця яка містить в собі інформацію про мережу, наша система здатна проводити моніторинг не тільки в Ethereum, але в Ethereum сумісних блокчейнах, тому є необхідність зберігати і мережу також
- Erc20Token – таблиця містить інформацію про конкретну криптовалюту в конкретній мережі.

Також на діаграмі можна побачити додаткові допоміжні таблиця, які в свою чергу реалізують зв'язок багато до багатьох. Це таблиці WalletChain, а також Erc20TransactionChain.

4.4 РОЗРОБКА СХОВИЩА ДАНИХ

Маючи за джерело даних мою оперативну базу даних я приступив до проектування сховища даних. Спочатку я спроектував гіперкуб, далі використовуючи SQL Server Analysis Service та Visual Studio яка дає можливість безпосередньої взаємодії з SSAS – я побудував цей гіперкуб, визначив таблицю фактів і таблиці вимірів.

В моєму гіперкубі було 3 виміри:

- Wallet_Dim – дає можливість отримати інформацію в розрізі гаманця, а також міста та регіона.
- Currency_Dim – дає можливість отримати інформацію в розрізі конкретної криптовалюти.
- Date_Dim – дає можливість отримати інформацію в розрізі проміжку часу

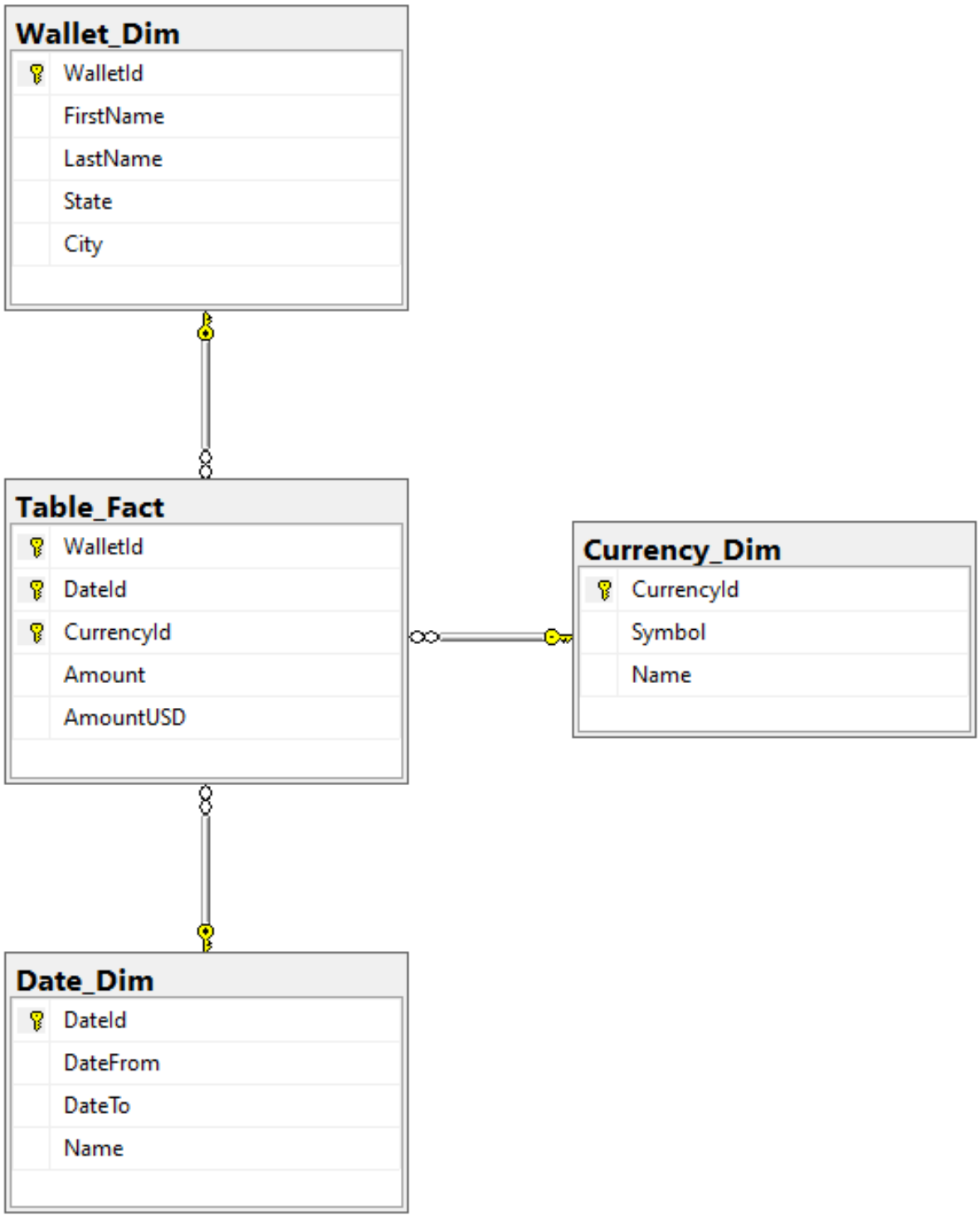


Рис 4.8 – OLAP гіперкуб

Поквартальний звіт доходів по гаманцях

Quartal 1

		GRT		LINK		RNDR		SHIB		USDC		USDT		
		Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount
Bill	Gates									5806		5806	55244	55244
Elon	Mask	88		600	560	74		1863	170911924	31147		31147	53222	53222
Harry	Potter	404		2774				3331	310438157				1238048	1238048
Jack	Sparrow	610		4016		917		495		200		200	10696	10696
Jef	Bethos												55235	55235
Jhoe	Biden	14754		103161	19679	2621		20031	16634				257031	257031
Mike	Tyson												1289	1289
Sam	Megison	3986		27251				297587	221648	347241	32000159455			
Tom	Kruz	44100		250910										
Tony	Stark											7986	7986	3500	3500

Quartal 2

		DYDX		GRT		RNDR		SHIB		USDC		USDT		
		Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount
Bill	Gates										4377	4377	24625	24625
Elon	Mask								1255	165121055	5102	5102	69203	69203
Harry	Potter								5499	740120891			745980	745980
Jack	Sparrow				74	735							15001	15001
Jhoe	Biden	31594		15873	21936	192080							3073334	3073334
Mike	Tyson												357	357
Sam	Megison				12657	121117	21844	10992	14977	1999585544				
Tom	Kruz				3186	22326								
Tony	Stark										19496	19496		

Quartal 3

Next Page

		GRT		LINK		RNDR		SHIB		USDC		USDT		
		Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount
Bill	Gates										35116	35116	125539	125539
Elon	Mask								177	24074783	24930	24930	96761	96761
Harry	Potter	4		50	1893	244	20	13	1412	193600216	68572	68572	1314774	1314774
Jack	Sparrow												19487	19487
Jef	Bethos										336	336	1275	1275
Jhoe	Biden				23363	2965							962342	962342
Mike	Tyson												436	436
Oleh	Prochayka						8647	4578						
Sam	Megison	7265		81996			246	161	8264	1127217859				
Tom	Kruz	1142		10874	11359	1508			488	59406381				
Tony	Stark										11000	11000		

Quartal 4

		DYDX		GRT		LINK		RNDR		SHIB		USDC		USDT		WETH		
		Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount	USD	Amount
Bill	Gates												8838	8838	163893	163893	32321	15
Boris	Brejcha						1381	98										
Chak	Noris						510	51										
Elon	Mask						432	28			1001	105887814	10728	10728	108208	108208		
Harry	Potter						674	42			8986	992031893			2349893	2349893		
Jack	Sparrow	1002		436	500	3914												
Jef	Bethos														5900	5900		
Jhoe	Biden	4563		1485	14411	94268									331378	331378		
Mike	Tyson														2180	2180		
Oleh	Prochayka								3034	853								
Sam	Megison				6609	43540			428708	121034	123623	13171192306						
Tom	Kruz				49404	323974	88569	6284	5887	1640								
Tony	Stark												4000	4000				

Рис. 4.9 – звітність поквартального доходу по гаманцях в розрізі криптовалют.

Наступним кроком я налаштував побудову звітностей які включали в себе обсяг доходу в розрізі гаманців та криптовалют поквартально, а також звітностей які включали в себе інформацію по доходах, в розрізі регіонів і також поквартально.

Рис. 4.10 звітність поквартального доходу по регіонах.

5. РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

Маючи достатню кількість даних потрібно було дослідити наявні транзакції на предмет наявності патернів та закономірностей. Для виявлення таких патернів ідеальним інструментом буде Data Minig. Для початку я адаптував наявні дані для подальшої роботи, також отримав додаткову інформацію по кожній транзакції і підготував набір даних який виглядав наступним чином:

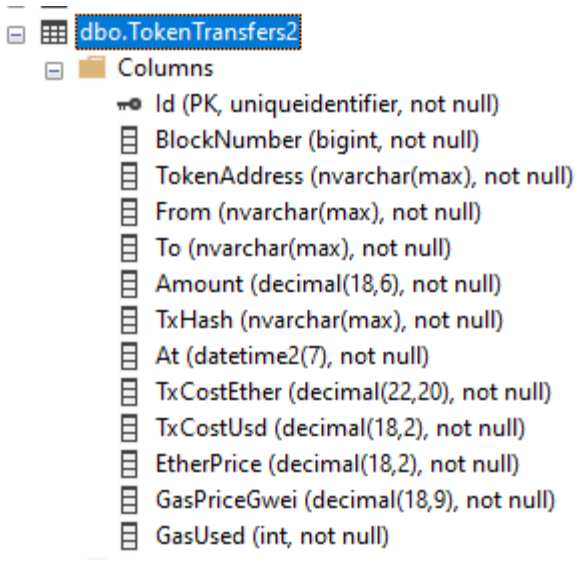


Рис. 5.1 – структура даних для задач Data Mining та Machine learning.

5.1 ПОВЕРХНЕВИЙ АНАЛІЗ ДАНИХ

Наступним кроком я провів поверхневий огляд даних ProfileReport бібліотеку для мови програмування Python.

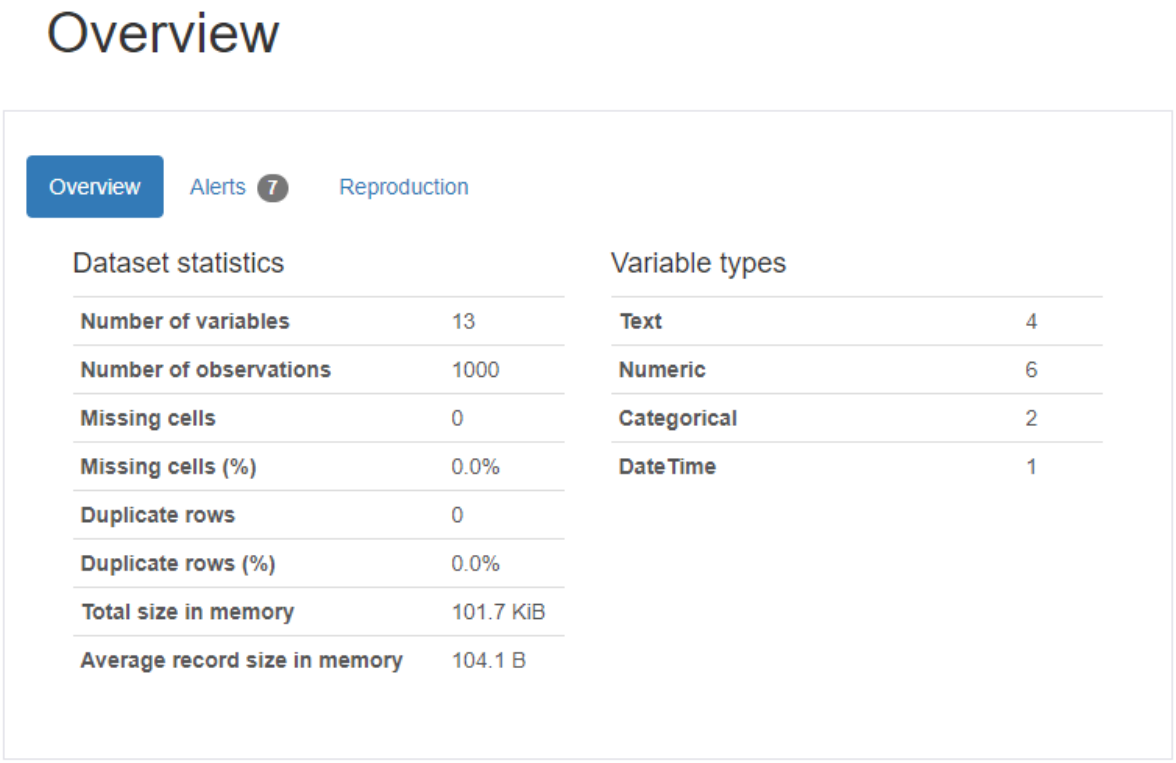


Рис 5.2 – загальний огляд набору даних засобами ProfileReport

Далі я провів огляд даних по окремо для кожного поля. Більшість полів таких як до прикладу BlockNumber, TxHash, At не несли в собі жодного логічного сенсу, адже були скоріше операційними ніж інформативними. Проте були й колонки які виявились цікавими, як от EtherPrice, GasPriceGwei а також GasUsed

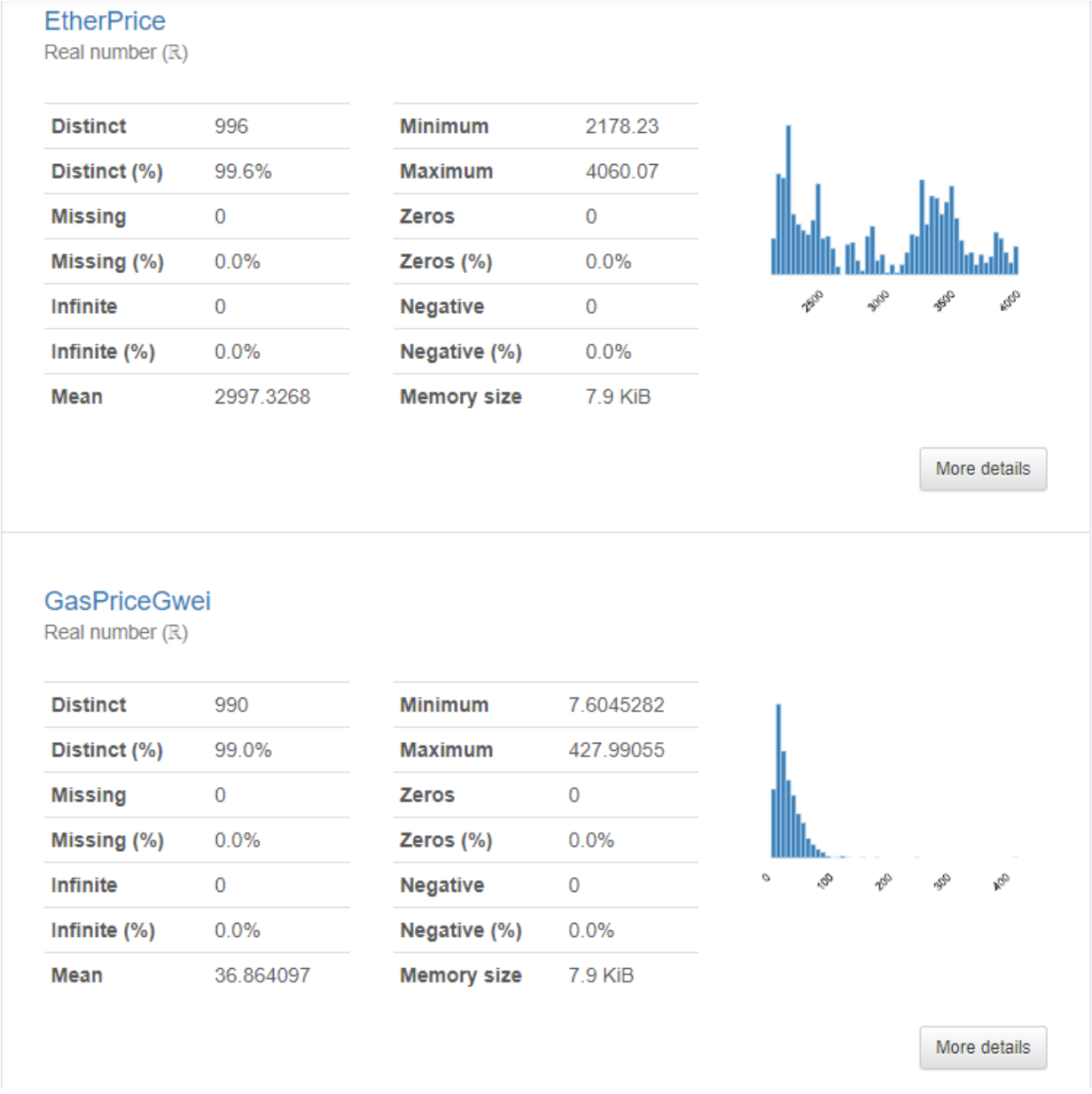


Рис 5.3 – Профайлінг даних. Поля EtherPrice та GaspPriceGwei.

Як видно з рисунку 4.13 більшість транзакцій було виконано при низькому показнику GasPriceGwei, в той час як Ether Price не має впливу на кількість транзакцій.

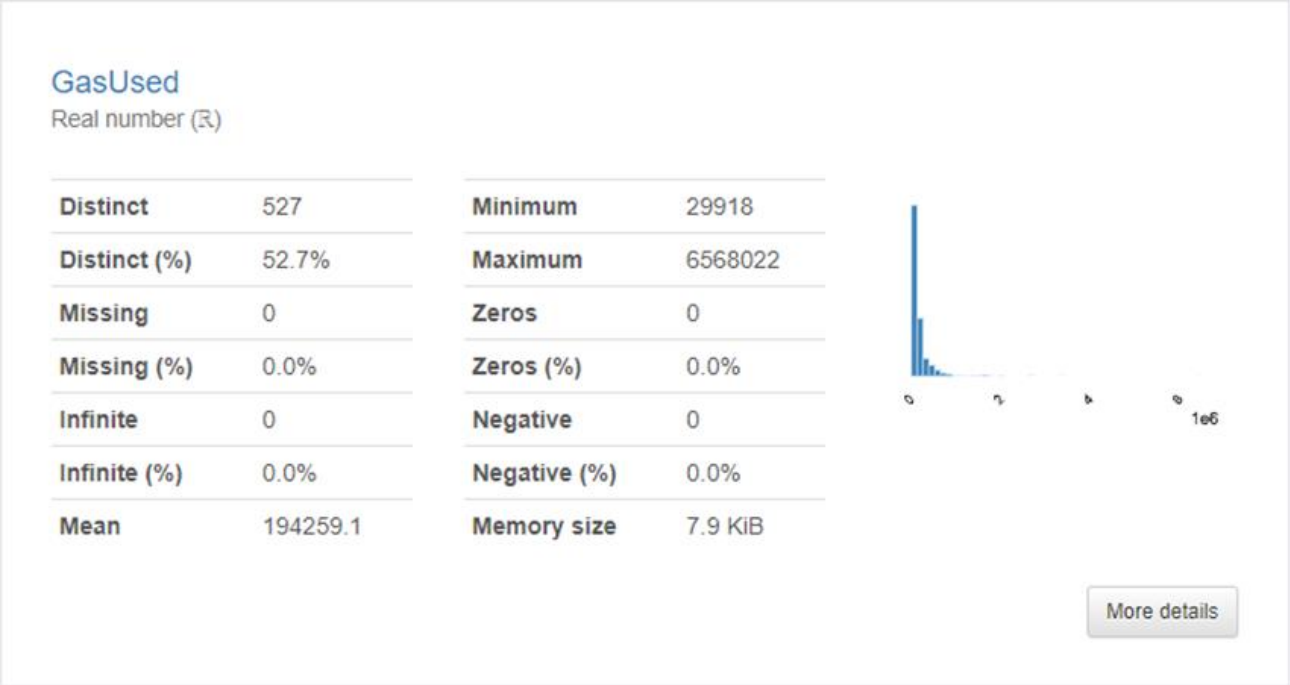
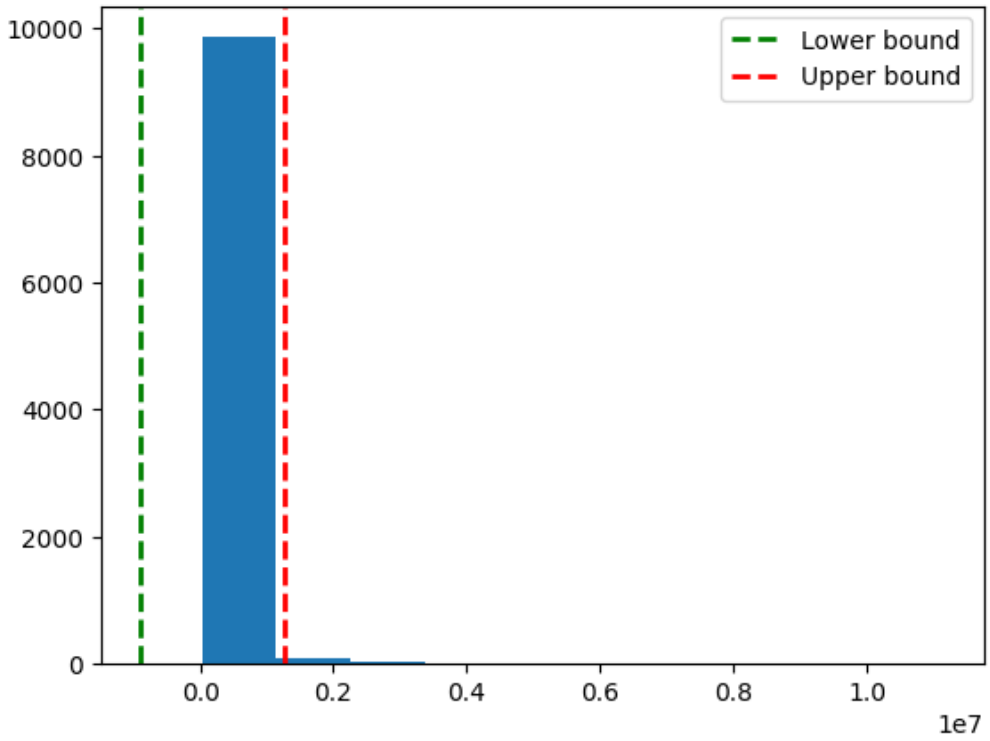


Рис 5.4 – Профайлінг даних. Поле GasUsed.

Як видно з рисунку 4.14 – більшість транзакцій, які включають в себе переказ коштів, не являються складними, так як GasUsed описує складність транзакції, або кількість необхідної роботи для обробки транзакції, можемо констатувати що більшість переказів не являються складними транзакціями.

Далі я перевірив аномалії та викиди для поля GasUsed використовуючи медіану та стандартну девіацію наступним чином:

- Мінімальний поріг – mean – 3 * std де mean – медіана значень по GasUsed, std – стандартна девіація значень по GasUsed
- Максимальний поріг – mean + 3*std де mean – медіана значень по GasUsed, std – стандартна девіація значень по GasUsed



	TxHash	GasUsed
16	0x8889dbb72641934fdbde98558fe7a0e680d6f4f96213...	2734142
114	0x1344208708ab96eb40de866da7f177edc37b1b257b1c...	1626716
434	0xcb59b6438adb705250dfa85ccaa2fdb1b60b15760fe9...	1779029
449	0xbe9af1f987a5c993c0de28ffd2626abcd74bfa34ffc7...	1543884
464	0x0cf9d4cc04ac4f82c606e7e1e3a1efa38e43b206af2e...	3397996
...	...	...
9271	0x75c02428a140d98a4c58c44d58f95b0131b49a93653a...	1758995
9294	0x5edadb0f56438300f50417570e57155d6213681bc8f5...	1800908
9370	0x8183d1bc67194b50d0bdb631e16b437879b09900e68c...	1609473
9429	0x6d236c0ed18cb8bba4c1a474744f36cea0de2ebba31e...	2788168
9442	0x135bfde45e6896971eafab28723ed3a1c83393d7548b...	1532924

[118 rows x 2 columns]

Рис. 5.5 – аномалії та викиди для колонки GasUsed

Наступним кроком я детальніше ознайомився з транзакціями які були в списку аномалій та викидів. Перевіривши кожну транзакцію окремо, я виявив що в рамках однієї транзакції було виконано велику кількість переказів. Це було здебільшого виконання якогось окремого розумного контракту, і скоріше за все ці транзакції були виконані в рамках внутрішніх операції криптовалютної біржі для хеджування активів між гаманцями.

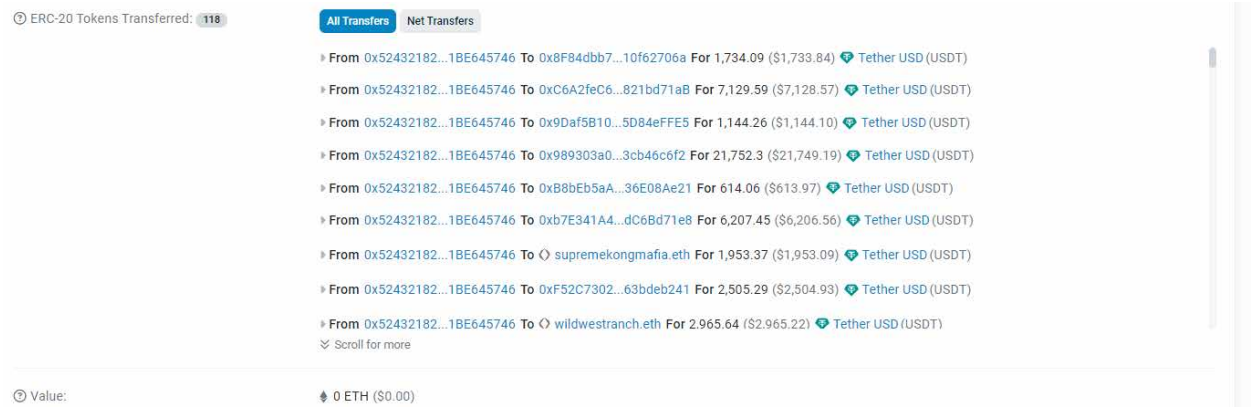


Рис. 5.6 – аналіз транзакції зі списку аномалій та викидів.

5.2 КЛАСТЕРИЗАЦІЯ ТА МЕТОД АСОЦІАТИВНИХ ПРАВИЛ

Далі було проведено кластерний аналіз засобами Python та використовуючи алгоритм K-середніх.

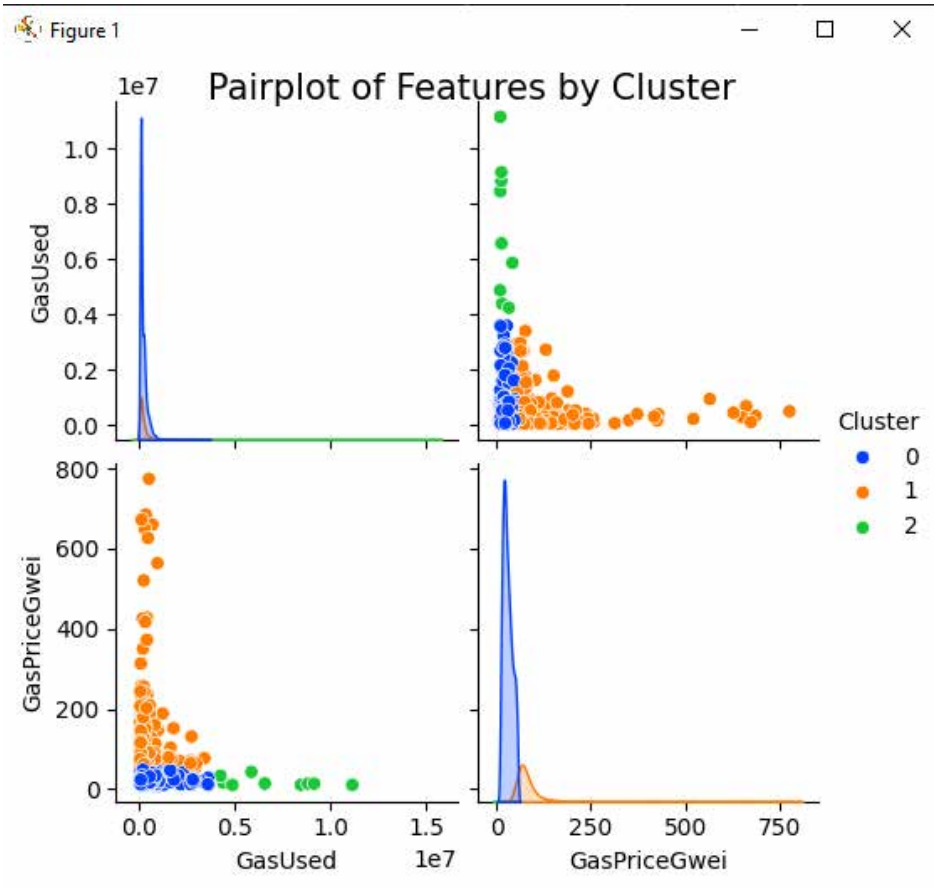


Рис. 5.7 – 2D представлення кластерного аналізу за змінною GasUsed

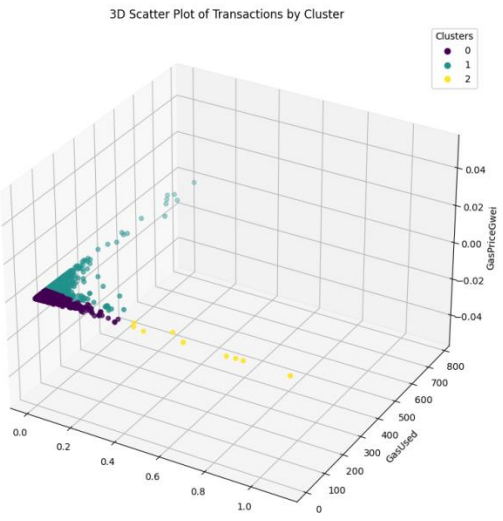


Рис 5.8 – 3D представлення кластерного аналізу за змінною GasUsed.

Розібравши детальніше транзакції характерні кожному кластеру я виділив, що кожен за полем GasUsed, тобто за складністтю виконання, транзакції поділились на ті які мають один переказ, ті які мають до десяти переказів в рамках однієї транзакції, і ті які мають десятки а то й сотні переказів в рамках однієї транзакції.

Для моєї системи важливо відслідковувати саме транзакції які мають 1 переказ. Але сервіс Moralis не надавав інформації про кількість переказів в транзакції, і цю інформацію потрібно було отримувати додатковим запитом з блокчейну. Цей додатковий запит являється доволі ресурсозатратним, тому було вирішено побудувати Machine Learning модель яка буде прогнозувати кількість переказів в рамках однієї транзакції базуючись на вхідних даних.

Для більшого розуміння даних я також побудував теплову карту асоціативних правил, що дало розуміння про наявність залежності додатково між криптовалютами та кількостями транзакцій.

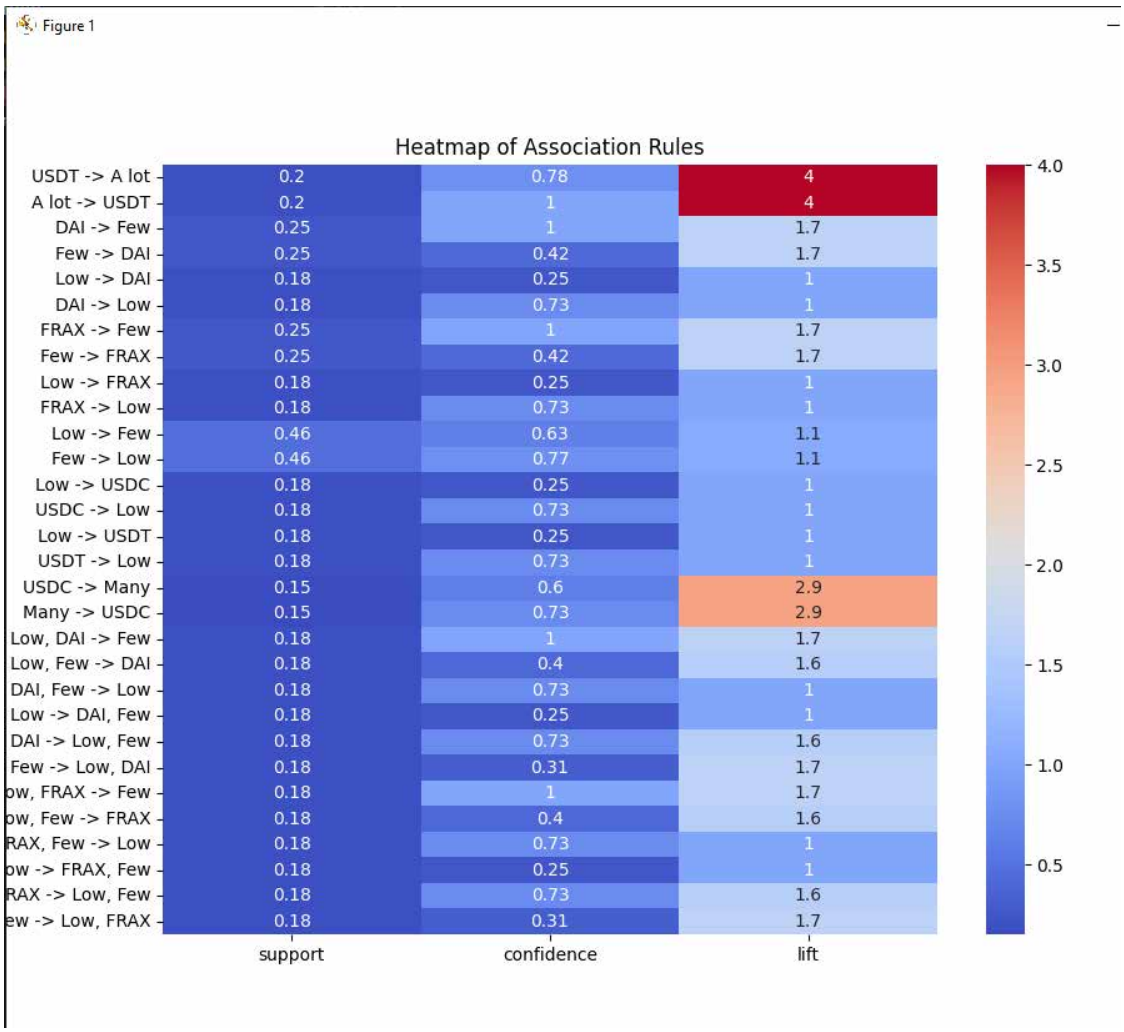


Рис. 5.9 – теслова мапа асоціативних правил

5.3 ПОБУДОВА ML МОДЕЛІ

Для побудови ML моделі я підготував 2 набори даних, для тренування та для тестування. Кожен з них складався з 10,000 записів, для кожної транзакції я також отримав безпосередньо з блокчейна інформацію про кількість операцій переказу.

Наступним кроком потрібно було провести масштабування кількісних ознак та масштабування якісних ознак.

Для масштабування я використовував алгоритми Standart Scaler, MinMax Scaler та Robust Scaler. Для кодування даних я використовував алгоритми OneHotEncoding, LabelEncoding і TargetEncoding.

Даля масштабування я визначив наступні необхідні кількісні ознаки

- GasPriceGwei
- GasUsed
- TransfersCount

Для кодування я виділив якісну ознаку CryptoCurrency що визначає назву криптовалюти.

В результаті я виконав кодування та масштабування даних різними алгоритмами і зкомбінував результати отримавши 6 наборів даних для навчання.

Наступним кроком я визначив алгоритми машинного навчання які хотів би використати в своїй роботі:

- Градієнтне посилення
- Лінійна регресія
- Асамблевий регресор

Використовуючи 6 підготовлений на попередньому кроці наборів даних, а також 3 алгоритми машинного навчання я натренував 18 ML моделей, які відрізнялись алгоритмами та вхідним набором даних. Для кожної окремої ML моделі я провів тестування і отримав наступні результати представлені в таблиці 4.1.

Результати виконання тестування ML моделей показали, що найгірше з передбаченням кількості переказів в транзакції справляється лінійна регресія, в

той час як найкраще справляється алгоритм градієнтного посилення, особливо в комбінації з MinMax алгоритмом для масштабування кількісних значень та Label Encoding для кодування якісних значень. Таким чином, модель машинного навчання була побудована і натренована для подальшого використання в системі.

Алгоритм машинного навчання	Алгоритм масштабування	Алгоритм кодування	Точність
Лінійна регресія	MinMax Scaler	Label Encoder	88.1%
	MinMax Scaler	OneHot Encoder	88.1%
	MinMax Scaler	Target Encoder	87.9%
	Standart Scaler	Label Encoder	88.1%
	Standart Scaler	Target Encoder	87.9%
	Standart Scaler	OneHot Encoder	88.1%
Градієнтне посилення	MinMax Scaler	Label Encoder	99.7%
	MinMax Scaler	OneHot Encoder	99.7%
	MinMax Scaler	Target Encoder	99.4%
	Standart Scaler	Label Encoder	99.7%
	Standart Scaler	Target Encoder	99.4%
	Standart Scaler	OneHot Encoder	99.7%
Асамблевий регресор	MinMax Scaler	Label Encoder	94.4%
	MinMax Scaler	OneHot Encoder	93%
	MinMax Scaler	Target Encoder	91%
	Standart Scaler	Label Encoder	94%
	Standart Scaler	Target Encoder	91.5%
	Standart Scaler	OneHot Encoder	93.5%

Таблиця 5.1 – порівняння результатів тестування ML моделей

6. ВИСНОВКИ

Під час виконання роботи було розроблено систему економічного моніторингу фінансових транзакцій на базі Ethereum та EVM сумісних блокчейнів. Розробка системи включала в себе розробку таких програмних модулів як:

- Клієнтський веб інтерфейс
- Веб сервер
- Сховище даних
- Machine Learning модель

Окрім безпосередньо розробки програмних модулів було також виконано інтеграцію веб сервера з такими сторонніми ресурсами як:

- Ethereum блокчейн
- Moralis
- Binance Api
- CoinMarketCap

Також на стороні клієнтського веб інтерфейсу було реалізовано інтеграцію з криптогаманцем Ethereum «Metamask»

Для реалізації системи було використано наступні технології:

- C# .NET 6 – для реалізації серверної частини
- React JS та Typescript – для реалізації клієнтського веб інтерфейсу
- SQL – для роботи з оперативною базою даних
- OLAP технології та MDX Query для взаємодії з гіперкубом

- Python для аналізу, кластеризації та побудови ML моделі
- Solidity – мова програмування розумних контрактів, для дослідження інтерфейсу переказів криптовалют в блокчейні

Проведені дослідження показали, що в Ethereum блокчейні є безліч різних транзакцій, будь-яка операції по зміні стану блокчейну – транзакція, і далеко не кожна з таких операцій змінює баланс криптовалют. Проте під час дослідження було виявлено певні закономірності і патерни, частина з них базувалась на технічному аспекті роботи мережі, інша частина була отримана за допомогою методів Data Mining.

В рамках дослідження було виявлено, що транзакції які передбачають переказ криптовалют мають спільні риси, а саме адресу розумного контракту конкретної криптовалюти, а також за результатами виконання транзакції – події типу «Transfer».

Як показав аналіз транзакцій з вибірки, деякі транзакції включали в собі більше одного переказу криптовалют, кластерний аналіз показав 3 типи таких транзакцій. Подальший аналіз конкретних транзакцій показав, що ці транзакції не являються фінансовими та їх не потрібно включати в моніторинг. Тому було розроблено модель машинного навчання яка на основі наявних вхідних даних може прогнозувати кількість переказів в транзакції з точністтю 99.7%

В подальшому система повинна бути доопрацьована інтеграцією з державини порталами Дія або BankID для можливості співставлення криптовалютного гаманця і конкретної людини, це дасть змогу використовувати систему для моніторингу фінансової активності в Ethereum та EVM сумісних блокчейнах користувачів, і буде корисною для забезпечення юридичного регулювання криптовалют в країні.

7. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Ethereum transaction | Wiki.js [Електронний ресурс] – https://inevitableeth.com/home/ethereum/blockchain/transaction

Transaction Flow on Ethereum blockchain [Електронний ресурс] – https://medium.com/block6/how-transaction-flow-on-the-ethereum-blockchain-8130f3f42a6a

Ethereum Virtual Machine (EVM) [Електронний ресурс] – https://ethereum.org/uk/developers/docs/evm/

Introduction to smart contracts [Електронний ресурс] – https://ethereum.org/en/developers/docs/smart-contracts/

DeBank [Електронний ресурс] – https://debank.com/

DexTools [Електронний ресурс] – https://www.dextools.io/

Multidimentional Model Databases (SSAS) [Електронний ресурс] – https://learn.microsoft.com/uk-ua/analysis-services/multidimensional-models/multidimensional-model-databases-ssas?view=asallproducts-allversions

SQL Server Reporting Service (SSRS) [Електронний ресурс] – https://learn.microsoft.com/ru-ru/sql/reporting-services/reports/reporting-services-reports-ssrs?view=sql-server-ver16

SQL Server Integration Service (SSIS) [Електронний ресурс] – https://learn.microsoft.com/ru-ru/sql/integration-services/sql-server-integration-services?view=sql-server-ver16

OLAP [Електронний ресурс] – https://uk.wikipedia.org/wiki/OLAP

React JS [Електронний ресурс] – https://react.dev/learn

Getting Started EF Core [Електронний ресурс] – https://learn.microsoft.com/en-us/ef/core/get-started/overview/first-app?tabs=netcore-cli

Materail UI [Електронний ресурс] – https://mui.com/material-ui/

What is Webhook ? [Електронний ресурс] – https://www.redhat.com/en/topics/automation/what-is-a-webhook

Об’єктно орієнтоване моделювання [Електронний ресурс] – https://uk.wikipedia.org/wiki/Об%27єктно-орієнтоване_моделювання

Typescript [Електронний ресурс] – https://www.typescriptlang.org/

Welcome to AWS Documentation [Електронний ресурс] – https://docs.aws.amazon.com/

Moralis Api Documentation [Електронний ресурс] – https://docs.moralis.com/

Binance APIs [Електронний ресурс] – https://www.binance.com/en/binance-api

CoinMarketCap API Documentation [Електронний ресурс] – https://coinmarketcap.com/api/documentation/v1/

The Basic MDX Query [Електронний ресурс] – https://learn.microsoft.com/en-us/analysis-services/multidimensional-models/mdx/mdx-query-the-basic-query?view=asallproducts-allversions