**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет**
**імені Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

**ДОПУЩЕНО ДО ЗАХИСТУ**
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

**БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

# Деаномізація транзакцій в блокчейн

Спеціальність 122 «Комп'ютерні науки»

**122 – БКР – 401. 21910109**

***Виконав студент 4-го курсу, групи 401***
_____ ***Є.О. Грабовський***
« » червня 2023 р.

***Керівник: канд. техн. наук, доцент***
_____***Є. В. Сіденко***
« » червня 2023 р.

**Миколаїв – 2023**

# АНОТАЦІЯ

**бакалаврської кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили**

**Грабовського Єгора Олександровича**

**Тема: «Деаномізація транзакцій в блокчейн»**


Кваліфікаційна робота присвячена розробці системи деанонімізації блокчейн транзакцій із використанням мови програмування C#.

Метою даної роботи є деаномізація блокчейн транзакцій та порівняльний аналіз застосованих методів.

Об’єктом роботи є процеси деаномізація блокчейн транзакцій.

Предметом роботи є методи та алгоритми деаномізація блокчейн транзакцій.

У цій роботі було розроблено програмне забезпечення, що дозволяє деанонімізувати блокчейн транзакції та забезпечити прозорість даних. Під час розробки було виконано аналіз принципів роботи блокчейну, вивчення методів деанонімізації та обфускації даних, а також реалізація різних алгоритмів деанонімізації.

У першому розділі роботи розглянуто принципи роботи блокчейн технології, проблеми приватності та необхідність деанонімізації.

У наступних розділах описано методи деанонімізації та обфускації даних, проведений порівняльний аналіз та вибрано ефективний підхід.

Розроблено програмне забезпечення для реалізації алгоритмів деанонімізації, включаючи тестування його працездатності та ефективності.

Проект був реалізований у вигляді системи, заснованої на мові програмування C# та використовує бібліотеку NBitcoin для роботи з блокчейн транзакціями. Система має функціональність з деанонімізації та обфускації даних, забезпечуючи підвищений захист конфіденційності користувачів.



Ключові слова: блокчейн, деанонімізація, обфускація, C#, NBitcoin, прозорість даних.

# ABSTRACT

**Bachelor's qualification work**

of the student of 401 group of Petro Mohyla Black Sea National University

Hraboskyi Yehor Oleksandrovych

Title: Development of a Blockchain Transaction de-anonymization System using C# Programming Language

This thesis focuses on the development of a system for de-anonymizing blockchain transactions using the C# programming language. The aim of this work is to enhance transparency and security in the blockchain network by revealing anonymous transactions.

The object of the research is blockchain technology and the processes of anonymous transactions within the blockchain. The subject of the research is the development and implementation of blockchain transaction de-anonymization algorithms using the C# programming language.

The first chapter of the work examines the principles of blockchain technology, privacy issues, and the need for de-anonymization.

In the following chapters, various methods of data de-anonymization and obfuscation are described, and a comparative analysis is conducted to identify the most effective approach.

Software has been developed to implement de-anonymization algorithms, including testing its functionality and efficiency.

The work includes the development of software that enables the de-anonymization of blockchain transactions and ensures data transparency. The development process involved the analysis of blockchain principles, the study of de-anonymization and data obfuscation methods, and the implementation of various de-anonymization algorithms.

The project is implemented as a system based on the C# programming language and utilizing the NBitcoin library for working with blockchain transactions.



Keywords: blockchain, blockchain, deanonymization, obfuscation, C#, NBitcoin, data transparency.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет ім. Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
*(шифр і назва)*
Галузь знань **12 «Інформаційні технології»**
*(шифр і назва)*

**ЗАТВЕРДЖУЮ**

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20___ р.

# З А В Д А Н Н Я
## на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Грабовському Єгору Олександровичу

1. Тема кваліфікаційної роботи «Деаномізація транзакцій в блокчейн».

Керівник роботи Сіденко Євген Вікторович, канд. техн. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: огляд існуючих методів деаномізації, технологічні вимоги, вимоги до безпеки, вибір інструментів та технологій для реалізації програмного забезпечення.

Очікуваний результат роботи: програма по деаномізації блокечейн транзакцій у вигляді програмного забезпечення.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- розгляд аналогічних систем деанонімізації блокчейну та їхні характеристики;
- технології, що використовуються у сфері деанонімізації блокчейну;
- опис технології блокчейну та її основні принципи;

– огляд типових блокчейн мереж, які будуть використовуватися у дослідженні;
– опис формату даних транзакцій у блокчейні;
– огляд методів деанонімізації блокчейну та їх переваги та недоліки;
– вибір методу деанонімізації, який буде реалізований у роботі;
– опис архітектури системи та використання технологій (наприклад, C#, ASP.NET, Angular, MSSQL);
– розробка модулів для взаємодії з блокчейн експлорером та отримання даних для аналізу;
– розробка модулів для деанонімізації блокчейн транзакцій та аналізу отриманих результатів.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини:

– здійснити аналіз умов праці в робочому приміщенні;

– встановити необхідний рівень показників для робочого приміщення, де проводяться роботи з розробки системи;

– встановити основні принципи техніки безпеки.

7. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис |
|---|---|---|
| Спеціальна частина з охорони праці | | |
| | | |

Керівник роботи __канд. техн. наук, доцент, Сіденко Є.В._____
*(наук. ступінь, вчене звання, прізвище та ініціали)*

_____
*(підпис)*

Завдання прийнято до виконання ___Грабовський Є.О. _____
*(прізвище та ініціали)*

_____
*(підпис)*

Дата видачі завдання «____» _____ 202__ р.

**КАЛЕНДАРНИЙ ПЛАН**
**виконання бакалаврської кваліфікаційної роботи**

Тема: Деаномізація транзакцій в блокчейн_____

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1 | Подання заяви на затвердження теми та керівників БКР | 27.10.2022 | 27.10.2022 | Виконано |
| 2 | Отримання завдання на виконання БКР | 21.11.2022 | 21.11.2022 | Виконано |
| 3 | Складання календарного плану роботи на весь період виконання БКР | 05.12.2022 | 05.12.2022 | Виконано |
| 4 | Отримання завдання на переддипломну практику | 25.04.2023 | 25.04.2023 | Виконано |
| 5 | Проходження переддипломної практики, збір та аналіз матеріалів до БКР | 01.05.2023 | 14.05.2023 | Виконано |
| 6 | Розробка звіту з переддипломної практики | 14.05.2023 | 14.05.2023 | Виконано |
| 7 | Виконання БКР: аналіз алгоритмів, огляд існуючих рішень, формування задачі, розробка інформаційної системи | 10.02.2023 | 15.06.2023 | Виконано |
| 8 | Попередній захист БКР на засіданні комісії кафедри | 26.04.2023 | 16.06.2023 | Виконано |
| 9 | Доробка та остаточне оформлення БКР | 29.05.2023 | 29.05.2023 | Виконано |
| 10 | Подання БКР рецензенту | 30.05.2023 | 16.06.2023 | |
| 11 | Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту | 17.06.2023 | 17.06.2023 | |
| 12 | Захист БКР перед екзаменаційною комісією (ЕК) | 20.06.2023 | 20.06.2023 | |

Розробив студент __Грабовський Є.О._____ _____
*(прізвище, ім'я, по батькові студента)* (*підпис*)

Керівник роботи __ канд. техн. наук, доцент, Сіденко Є.В. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«___» _____ 2022 р.

# ЗМІСТ

# – ПЕРЕЛІК СКОРОЧЕНЬ

| | | |
|---|---|---|
| ASP.NET | – | active Server Pages for .NET |
| MVC | – | model, view, controller |
| SOAP | – | simple Object Access Protocol |
| REST | – | representational State Transfer |
| SQL | – | structured query language |
| HTTP | – | hypertext Transfer Protocol |
| HTML | – | HyperText Markup Language |

# ВСТУП

У сучасному цифровому світі блокчейн-технологія займає центральне місце, надаючи унікальні можливості для безпечної та прозорої передачі даних. Проте існує проблема, пов'язана з анонімністю учасників транзакцій у блокчейні. Анонімність може бути корисною для захисту конфіденційності, але в певних випадках вона може стати загрозою, коли зловмисники використовують блокчейн для сумнівних чи протизаконних операцій.

Деанонімізація блокчейн транзакцій – актуальне завдання, яке привертає увагу дослідників та розробників. Це процес ідентифікації та розкриття анонімних транзакцій з метою підвищення прозорості та безпеки у мережі блокчейн. Вирішення цього завдання вимагає застосування спеціалізованих алгоритмів та методів, а також використання різних інформаційних технологій.

Метою даної бакалаврської роботи є розробка системи деанонімізації блокчейн транзакцій із використанням мови програмування C#. В рамках роботи будуть розглянуті основні принципи функціонування блокчейну, а також вивчено методи деанонімізації та обфускації даних у контексті блокчейн транзакцій. Крім того, буде проведено аналіз та реалізацію різних алгоритмів деанонімізації, а також досліджено можливості бібліотеки NBitcoin та фреймворку ASP.NET для розробки відповідних додатків.

Результати цієї роботи будуть корисними для розробки систем та інструментів, здатних ефективно розкривати анонімні транзакції у блокчейні, покращуючи прозорість та безпеку мережі. Крім того, робота внесе свій внесок у область досліджень блокчейн технологій та деанонімізації, надаючи практичні рекомендації та рішення, що базуються на використанні мови програмування C# та пов'язаних технологій.

У наступних розділах роботи будуть представлені огляд літератури, опис методів і технологій, що використовуються, а також докладний опис розробленої системи деанонімізації блокчейн транзакцій на базі мови програмування C#.

# 1 АНАЛІЗ СУЧАСНОГО СТАНУ ЗАДАЧІ ДЕАНОМІЗАЦІЇ БЛОКЧЕЙН ТРАНЗАКЦІЙ

## 1.1 Визначення основних понять блокчейн

Блокчейн – загальноприйнятим визначенням блокчейн є цифрова розподілена база даних, яка забезпечує безпеку, надійність та прозорість в обміні даними, включаючи транзакції з криптовалютами. Блокчейн складається з блоків, які містять інформацію про транзакції та інші дані. Кожен блок містить посилання на попередній блок за допомогою хеш-функції, що забезпечує нерозривність ланцюжка блоків. Крім того, блокчейн забезпечує прозорість та надійність шляхом децентралізації - кожен учасник мережі має копію блокчейну, що дозволяє запобігти можливості втручання або підробки даних. Блокчейн є основою для ряду різноманітних застосувань, включаючи криптовалюти, цифрові контракти, системи голосування та багато іншого.

Анонімізація – це процесс, під час якого особисті дані користувачів, які зберігаються в базах даних, піддаються обробці з метою зниження ризику несанкціонованого доступу до цих даних. Це може включати видалення або заміну ідентифікаторів користувачів на анонімні значення, такі як номери або коди, які не можуть бути пов'язані з конкретною особою. Цей процес забезпечує збереження конфіденційності даних та дозволяє уникнути можливих наслідків, пов'язаних зі зловмисним використанням особистої інформації. Проте, деанонімізація не є повною гарантією захисту особистих даних, оскільки іноді можна відновити початкові ідентифікатори користувачів, використовуючи додаткову інформацію. Тому, важливо бути обережним і забезпечити максимально можливий рівень захисту особистої інформації.

Деаномізація – це зворотня сторона аномізації, Деанонімізація в блокчейні відноситься до процесу розкриття реальних ідентифікацій користувачів або учасників, які інакше є псевдонімами або анонімами в мережі блокчейн.

Транзакція – це взаємодія між двома або більше сторонами, яка включає передачу цифрових активів, таких як гроші, криптовалюта, товари або послуги. Це процес, який зазвичай відбувається в мережі, де сторони мають домовленість щодо умов транзакції. Транзакція може бути електронною або фізичною, і вона може мати різні форми, такі як купівля-продаж, передача власності, відправка платежу або контрактний обмін [1-3].

Крім цього, транзакції є важливим елементом багатьох сучасних технологій, таких як криптовалюти і блокчейн. У таких системах транзакції зберігаються в цифровому реєстрі, що називається блокчейном, і можуть бути перевірені та підтверджені учасниками мережі. Блокчейн забезпечує безпеку транзакцій, оскільки він не дозволяє їх змінювати після їх внесення в реєстр. Таким чином, транзакції можуть забезпечити безпеку та надійність у сфері електронних платежів і передачі активів [2].

Криптовалюта – це нова форма електронної валюти, яка дозволяє виконувати безпечні та анонімні транзакції в Інтернеті без посередництва фінансових установ. Криптовалюта базується на криптографічних алгоритмах, які забезпечують безпеку та конфіденційність транзакцій, а також на технології блокчейн, що дозволяє зберігати інформацію про всі транзакції в цілком безпечному та децентралізованому режимі. Криптовалюти можуть бути використані для різних цілей, включаючи покупку товарів та послуг, інвестування в криптовалютні активи, а також для спекуляції на фінансових ринках [2-4].

Приватність – це право людини на контроль за тим, як їхні особисті дані збираються, зберігаються та використовуються. У контексті блокчейн технологій, приватність означає можливість забезпечення захисту конфіденційної інформації про користувачів та їхні транзакції від несанкціонованого доступу та зловживань. У деяких випадках, наприклад у публічних блокчейнах, приватність може бути обмежена, оскільки всі дані, які передаються через мережу, є доступними для перегляду та аналізу. Однак, існують також приватні блокчейни, де доступ до деякої інформації може бути обмежений тільки для певних учасників, які мають

право на цей доступ. Таким чином, приватність є важливим аспектом у розвитку блокчейн технологій, особливо у контексті захисту персональних даних та конфіденційної інформації [3].

Децентралізація – це процес перенесення контролю та влади з централізованих структур на рівень учасників мережі, що дає можливість уникнути ризиків зловживання владою або витоку даних. У контексті блокчейн технологій, децентралізація означає, що кожен учасник мережі може перевірити інформацію, збережену в блоках, а кожен блок зберігається на комп'ютерах різних учасників мережі, що робить мережу більш безпечною та стійкою до атак ззовні. Також децентралізовані мережі забезпечують прозорість та відкритість, оскільки всі учасники мережі можуть бачити та перевіряти операції в реальному часі [4-6].

Хеш-функція – це алгоритм, який приймає на вхід будь-який текстовий рядок та перетворює його в унікальну послідовність символів фіксованої довжини, яка називається хешем. Основна властивість хеш-функції полягає в тому, що вона генерує однаковий хеш для того ж самого вхідного значення при кожному запуску. Це дозволяє перевіряти цілісність даних, оскільки навіть маленька зміна вхідних даних призведе до зміни хеш-коду [5].

У блокчейн технологіях, хеш-функції використовуються для забезпечення безпеки та цілісності даних у блоках. Кожен блок у ланцюжку блоків містить свій хеш-код, а також хеш-код попереднього блока. Це забезпечує неможливість зміни даних вже збережених блоків, оскільки будь-яка зміна даних призведе до зміни хеш-коду блоку, а отже, до порушення ланцюжка. Хеш-функції також використовуються для створення цифрових підписів та аутентифікації користувачів в системах блокчейн.

Публічний ключ – це частина криптографічного ключа, яка використовується для шифрування повідомлень або підпису цифрових підписів. У контексті блокчейн технологій, публічні ключі використовуються для ідентифікації користувачів та забезпечення безпеки транзакцій. Кожен користувач мережі

блокчейн має свій унікальний публічний ключ, який дозволяє іншим учасникам мережі перевірити його підписи та виконувати транзакції з його рахунку [6-7].

Протокол консенсусу – це набір правил, які регулюють процес прийняття рішень в мережі. Ці правила дозволяють учасникам мережі домовлятись про те, який стан мережі є правильним, та вирішувати конфлікти між різними версіями мережі. У блокчейн технологіях, протокол консенсусу є ключовим елементом, який дозволяє забезпечувати безпеку та цілісність мережі, а також гарантує, що всі учасники мережі мають однакове уявлення про стан блокчейну. Протокол консенсусу зазвичай залежить від типу блокчейну та може включати різні алгоритми, такі як Proof of Work, Proof of Stake, або інші [8].

Смарт-контракти – це програмні коди, що виконуються автоматично після виконання певних передумов або умов. У блокчейн технологіях, смарт-контракти є одним з основних інструментів для автоматизації різних бізнес-процесів та угод між учасниками мережі. Вони дозволяють зменшити витрати на проведення різноманітних операцій, підвищити їх безпеку та надійність, а також забезпечити прозорість угод між учасниками. Смарт-контракти можуть виконуватися без участі посередників, тому їх використання може допомогти підвищити ефективність та швидкість проведення операцій [9].

## 1.2 Огляд аналогів існуючих програм з проблематикою блокчейн. Аналіз публікацій.

Останні дослідження в області блокчейнів акцентують увагу на проблемах приватності та анонімності. Багато блокчейн мереж не забезпечують належного рівня конфіденційності та безпеки особистих даних, тому з’явилася потреба у технологіях, що забезпечують приватність та анонімність користувачів.

Недавні публікації, такі як A Survey of Blockchain Anonymity Techniques (2020) [10] та A Survey of Privacy in Blockchain Technologies (2021) [11] розглядають проблеми приватності в блокчейні та пропонують рішення, такі як анонімні валюти, протоколи конфіденційності та децентралізовані системи ідентифікації. У

статті Privacy-preserving blockchain protocols: An overview (2019)[12] також досліджуються проблеми приватності в блокчейні та пропонуються різні техніки, такі як анонімні міксери, протоколи конфіденційності та техніки забезпечення приватності.

Крім того, існують деякі блокчейн мережі, що забезпечують повну анонімність транзакцій та збереження приватності користувачів. Наприклад, додаток Zcash до біткойна забезпечує повну анонімність транзакцій та збереження приватності користувачів. Мережа Monero також забезпечує повну анонімність транзакцій та збереження приватності користувачів за допомогою технології Ring Confidential Transactions (RingCT) [12-14].

У світі блокчейн технологій постійно розвиваються нові методи та техніки, щоб забезпечити максимальний рівень приватності та анонімності користувачів. Проте, важливо бути уважним та обирати надійні технології.

## 1.3 Блокчейн-транзакції. Технології, методи та підходи їх деаномізації

Аналіз ланцюжка транзакцій за допомогою відповідних мережевих інструментів, який полягає у відстеженні транзакцій по мережі та накопиченні загальнодоступних відомостей про них, а також пов’язуванні їх з особистими даними користувача [15].

Аналіз протоколу та мережі, який використовує характеристики розповсюдження транзакцій з криптовалютою для визначення вихідної IP-адреси нової транзакції. На сьогодні можна обирати серед комерційних послуг та інструментів з відкритим кодом, що забезпечують програмний доступ. Для відстеження шляху переміщення транзакцій до кінцевого одержувача дозволяють провести розглянуті далі сервіси [16].

Техніки змішування: використання спеціальних сервісів або програм для змішування біткоїнів, що дозволяє ускладнити аналіз походження блокчейн транзакцій [17].

Техніки розділення: розділення великих блокчейн транзакцій на менші частини, що знижує ймовірність виявлення зв'язку між ними [18].

Використання туманної мережі: технологія, яка дозволяє створювати мережу з обмеженим доступом та забезпечує анонімність учасників [19].

Розширення приватності: додавання нових протоколів та алгоритмів до блокчейн, що дозволяє підвищити приватність транзакцій [20].

Blockchain Explorer – це один з найбільш відомих інструментів аналізу ланцюжка блоків. Він пропонує ряд можливостей для відстеження окремих транзакцій, а також надає інформацію у вигляді графіків і статистики всієї мережі. Крім того, з його допомогою можна провести аналіз стосовно руху коштівпо мережі. На головній сторінці сервісу можемо побачити діаграми, що відображають зміни цін на криптовалюту за останній день, тиждень або місяць, а також сумарний розмір непідтверджених транзакцій в байтах. Крім того, маємо можливість детального перегляду інформації, що надає нам ще більшу кількість діаграм, які побудовані за валютною статистикою, деталями блоку, інформацією про їх видобуток, мережевою активністю, кількістю активних гаманців, а також ринковими сигналами. Також тут нам одразу відомі які і ким були отримані останні блоки, їх розмір, а також список непідтверджених транзакцій та суми, які були передані. Відображені відомості також можна розглянути детальніше і дізнатися час передачі окремої транзакції в мережу, комісію, яка була сплачена за її обробку, розмір транзакції, перелік адрес і сум з витраченими і отриманими коштами.

Matbea.net – це послуга, яка дозволяє користувачам встановлювати належність біткоїн адрес. Даний інструмент надає користувачам можливість шукати інформацію по транзакціям, адресам, блокам, xPub або yPub і видає результат у вигляді детальної текстової статистики. Послуга перекладена на кілька іноземних мов та має зрозумілий інтерфейс.

ORS CryptoHound – це ще один інструмент дослідження мережі на базі штучного інтелекту, який використовується для дослідження Біткоїн та Ефіріум адрес і надає результати у вигляді списків, діаграм або таблиць. Інструмент

пропонує можливості відстеження коштів за конкретною адресою, відображення залишку на балансі, візуалізацію відношень адрес і всіх транзакцій, що проведені нею, дозволяє виконувати статистичний розрахунок вартості монет, а також формування банківських звітів. Інтерфейс інструмента представленно нижче (рисунок 1.1).

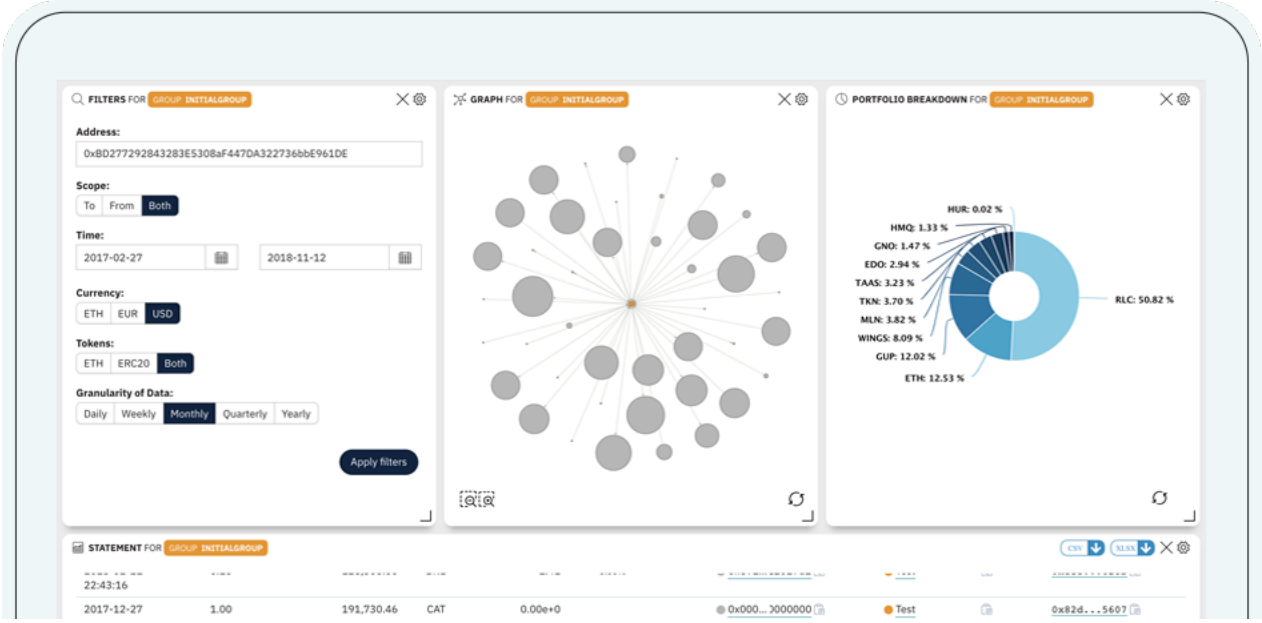


Рисунок 1.1 – ORS CryptoHound

Сервіс Glassnode – являє собою анал ітичну компанію, що займається аналітикою блокчейн мереж і надає оперативну інформацію про стан ринку, пропонуючи відображення результатів в різних категоріях. Використання подібних інструментів з відкритим кодом є досить зручним за рахунок їх доступності, однак деякі з них вимагають багато часу для дослідження окремих ділянок мережі та транзакцій і проводити такий аналіз вручну стає неефективним. До того ж, щоб вчасно виявити загрозу і попередити атаки, необхідно мати здатність передбачення методів, які можуть застосувати зловмисники в мережі.

## 1.4 Постановка задачі створення програми для деаномізації блокчейн-транзакцій

Блокчейн є потужною технологією, яка забезпечує безпеку, надійність та прозорість в обміні даними, включаючи транзакції з криптовалютами. Однак, у зв'язку зі збільшенням обсягів даних, які зберігаються в блокчейні, з'явилася необхідність зменшення або повного видалення інформації про особисті дані користувачів, щоб забезпечити їхню приватність.

Об’єктом роботи є процеси деаномізація блокчейн транзакцій.

Предметом роботи є методи та алгоритми деаномізація блокчейн транзакцій.

Метою даної роботи є деаномізація блокчейн транзакцій та порівняльний аналіз застосованих методів.

Нижче наведено перелік завдань для досягнення поставленої мети:

– огляд існуючих технологій та методів для деаномізації блокчейн транзакцій;

– вивчення застосування криптографічних методів для деаномізації транзакцій;

– розгляд можливостей застосування машинного навчання для деаномізації блокчейн даних;

– порівняння ефективності різних методів та технологій для деаномізації блокчейн транзакцій.

## 1.5 Висновки до розділу 1

У першому розділі було розглянуто базові поняття блокчейн технології та деанонімізації. Блокчейн є розподіленою базою даних, яка забезпечує безпеку, надійність та прозорість у збереженні даних. Деанонімізація є процесом зменшення або повного видалення особистих даних користувачів, що зберігаються у базах даних, з метою захисту їх приватності.

Для досягнення мети роботи, яка полягає в дослідженні методів деанонімізації блокчейн транзакцій, необхідно дослідити існуючі технології та методи деанонімізації, вивчити криптографічні методи та можливості застосування машинного навчання для деанонімізації блокчейн даних. Наступним кроком буде порівняння ефективності різних методів та технологій для деанонімізації блокчейн транзакцій та розробка рекомендацій щодо використання найбільш ефективних методів та технологій для забезпечення приватності користувачів.

# 2 МЕТОДИ ДЕАНОМІЗАЦІЯ БЛОКЧЕЙН ТРАНЗАКЦІЙ

## 2.1 Методи технології блокчейн. Популярні моделі.

Деанонімізація мережі Bitcoin чи іншої мережі - це процес ідентифікації реальних користувачів, які беруть участь у мережі. Це можна зробити, зв’язавши адреси Bitcoin з IP-адресами, які потім можна зв’язати з ідентифікаторами користувачів.

Одним із способів деанонімізації мережі Bitcoin є використання кластеризації адрес. Кластеризація адрес — це техніка, яка групує адреси Bitcoin, які, ймовірно, належать одній особі чи організації. Це можна зробити, враховуючи такі фактори, як час транзакцій, суми залучених грошей і географічне розташування адрес.

Кластеризація набуває значення, коли є одним з етапів аналізу даних, будуючи повне аналітичне рішення. Аналітику часто легше ідентифікувати групи подібних об'єктів, вивчити їх особливості і побудувати окрему модель для кожної групи, ніж створити одну загальну модель для всіх даних. Ця методика постійно використовується в маркетингу, висвітлюючи групи клієнтів, покупців, продукцію і розробляючи окрему стратегію для кожного з них [24].

Для того щоб порівняти два об'єкти, необхідно мати критерій, на основі якого буде проходити порівняння. Як правило, таким критерієм є відстань між об'єктами. Існує безліч мір відстані, нижче розглянуто декілька з них.

Манхеттенська метрика (метрика прямокутного міста, метрика L1) — метрика, запроваджена Германом Мінковським. За цією метрикою, відстань між двома точками дорівнює сумі модулів різниць їх координат [25].

У цієї метрики багато назв. Манхеттенська метрика відома як манхеттенська відстань, відстань міських кварталів, метрика прямокутного міста, метрика L1, вулична метрика або норма, метрика міського кварталу, метрика таксі, прямокутна метрика, метрика прямого кута;

Назва «манхеттенська відстань» пов'язана з вуличним плануванням Манхеттена, де вулиці перетинаються під прямими кутами.

Відстань Чебишева дорівнює максимальному відстані між відповідними координатами об'єктів. Відстань Чебишева використовують тоді, коли потрібно визначити відмінність двох об'єктів *г* і / з якої-небудь однієї координаті. Відстань Чебишева є грубою мірою відмінності, так як значна частина наявної інформації ігнорується [26].

Іноді хочеться поступово збільшувати або зменшувати вагу, пов'язану з розмірністю, для якої відповідні об'єкти дуже різні. Цього можна досягти за допомогою відстані потужності.

Квадрат евклідової відстані знаходиться як відстань між двома елементами через суму квадратів різниці значень всіх змінних. Квадрат евклідова відстані використовується для додання великих ваг найбільш віддаленим один від одного об'єктів. Особливо це важливо використовувати для стандартизованих змінних [27].

При $p = 2$ формула відстані Маньківського набирає вигляду евклідова відстані; при $p = 1$ отримуємо відстань Хеммінга.

Відстані між об'єктами, розраховані з якої-небудь з перерахованих вище формул, представляють у вигляді матриці відстаней:

Як бачимо, матриця відстаней являє собою квадратну матрицю типу "об'єкт - об'єкт" (близько $n$ ), де в якості елементів виступають відстані між об'єктами в метричному просторі. Діагональні елементи такої матриці дорівнюють нулю.

Як зазначено вище, поряд з різними видами відстаней однорідність об'єктів може бути визначена за допомогою заходів ступеня близькості (подібності). В якості запобіжного близькості (подібності) може бути використаний лінійний коефіцієнт кореляції :

Використання міри схожості в кластерному аналізі забезпечує комбінацію двох методів багатовимірного статистичного аналізу: кореляційного і кластерного аналізів. У цьому випадку в результаті отримуємо кореляційну матрицю або матрицю кореляцій.

Метод k-means використовується для кластеризації даних на основі алгоритму розділення векторного простору на задану кількість кластерів k. Алгоритм - це ітераційна процедура, в якій виконуються наступні кроки:

– вибрано кількість кластерів k;

– з початкового набору даних випадковим чином вибираються k спостережень, які будуть служити початковими центрами кластерів;

– для кожного спостереження початкової множини визначається найближчий до нього центр скупчення (відстані вимірюються в метриці Евкліда). При цьому записи «притягуються» певним центром утворюють початкові скупчення;

– обчислюються центриди, центри тяжіння скупчень. Кожен центроїд є вектором, елементи якого є середніми значеннями відповідних ознак, обчисленими з усіх записів у кластері;

– центр скупчення зміщується до його центроїда, після чого центроїд стає центром нового скупчення;

– кроки 3 і 4 ітеративно повторюються. Очевидно, що при кожній ітерації межі кластерів змінюються і змінюються їх центри. В результаті відстань між елементами всередині скупчень мінімізується, а міжсумні відстані збільшуються.

Зупинка алгоритму здійснюється тоді, коли межі кластерів і розташування центроїдів не перестають змінюватися від ітерації до ітерації, тобто при кожній ітерації в кожному кластері залишиться один і той же набір спостережень. На практиці алгоритм зазвичай знаходить набір стабільних скупчень в декількох десятках ітерацій.

Перевагою алгоритму є швидкість і простота реалізації. До недоліків можна віднести невизначеність вибору початкових центрів кластерів, а також той факт, що кількість кластерів необхідно вказати спочатку, що може вимагати якоїсь апріорної інформації про вихідні дані.

Існують методи кластеризації, які можна розглядати як похідні від k-засобів. Наприклад, метод k-медоїдів використовує медіану, а не середнє значення для

розрахунку центроїдів, що робить алгоритм більш стійким до аномальних значень в даних.

Алгоритм g-засобів (від гаусового) будує кластери, в яких розподіл даних має тенденцію до норми (Гауссіан) і знімає невизначеність вибору початкових кластерів. Алгоритм С-середнього значення використовує елементи нечіткої логіки, беручи до уваги при розрахунку центроїдів не тільки відстані, але і ступінь приналежності спостереження до набору об'єктів в кластері. Також відомий алгоритм Ллойда, який використовує не так багато векторів в якості початкового розділу,  Область векторного простору.

Ідея методу k-засобів була одночасно сформульована Уго Стейнхаусом і Стюартом Ллойдом в 1957 році.

Процес деанонімізації мережі Bitcoin з використанням кластеризації адрес включає кілька кроків.

Перший крок це збір даних - для початку необхідно зібрати дані про транзакції Bitcoin. Для цього можна використовувати публічні блокчейн-експлорери або спеціалізовані сервіси, які надають доступ до історії транзакцій. Дані про транзакції містять інформацію про входи та виходи, адреси відправників та одержувачів, а також суми переказів.

Наступний крок – це кластеризація адрес. Існує кілька алгоритмів кластеризації, які можна використовувати, наприклад:

– K-means: Це один із найпопулярніших алгоритмів кластеризації. Він поділяє адреси на задану кількість кластерів, де кожен кластер представляє групу адрес, що мають схожі характеристики, такі як суми переказів або часи транзакцій;

– DBSCAN: Це алгоритм кластеризації, який дозволяє автоматично визначати кількість кластерів та групувати адреси на основі їх густини. Адреси, що знаходяться близько один до одного в просторі ознак (наприклад, схожі суми переказів), поєднуються в один кластер;

– Ієрархічна кластеризація: Цей метод будує ієрархічну структуру кластерів, де кожна адреса може перебувати в підкластерах різного рівня. Це дозволяє отримати більш детальну інформацію про групування адрес.

Додатково до кластеризації адрес можна проаналізувати тимчасові параметри транзакцій, такі як: час між транзакціями або періодичність активності. Це може допомогти виявити закономірності або шаблони поведінки, які можуть свідчити про зв'язок між адресами.

Після кластеризації адрес, їх можна зв'язати з IP-адресами. Для цього можна використовувати різні методи, такі як моніторинг IP-адрес, що підключаються до вузлів Bitcoin. При збігу IP-адрес з певним кластером адрес можна припустити, що вони належать одному користувачеві або організації.

Зв'язок з ідентифікатором користувача: Останній крок - зв'язок адрес та IP-адрес з ідентифікаторами користувачів. Для цього може знадобитися використання зовнішніх джерел даних або додаткових методів ідентифікації користувачів, таких як аналіз поведінки або асоціація з іншими онлайн-активностями.

Важливо відзначити, що процес деанонімізації у мережі Bitcoin складний і може бути обмежений доступом до певних даних чи ресурсів. Також варто враховувати юридичні та етичні аспекти, пов'язані із проведенням таких досліджень.

Деанонімізація мережі Bitcoin становить серйозну загрозу конфіденційності користувачів. Це може бути використане для відстеження потоку коштів, ідентифікації злочинців та націленості на користувачів для переслідування чи дискримінації.

Існує кілька способів захисту від деанонімізації. Один із способів – використання Bitcoin-міксерів, які є сервісами, що змішують ваші біткойн-транзакції з транзакціями інших користувачів. Це ускладнює зв'язування ваших транзакцій із вашою особистістю.

Ще один спосіб захисту від деанонімізації – використання нової адреси Bitcoin для кожної транзакції. Це ускладнює відстеження ваших дій.

Важливо знати, що деанонімізація та анонімізація є двосторонніми процесами, і при розробці систем та технологій важливо враховувати їх обидві сторони. У той час як деанонімізація може становити загрозу для конфіденційності, анонімізація може бути важливим аспектом захисту приватності користувачів.

Деанонімізація мережі Bitcoin, а також інших мереж може бути виконана з використанням різних методів та технік, які можуть розкрити реальні ідентифікатори користувачів, що беруть участь у мережі. Це може становити загрозу для приватності користувачів та дозволяти проводити відстеження грошових потоків, ідентифікацію злочинців і навіть потенційне переслідування чи дискримінацію.

Один із таких методів деанонімізації – аналіз графіка транзакцій. Ця техніка передбачає створення графа всіх транзакцій Bitcoin, де вузли представляють адреси біткойнів, а ребра – транзакції між адресами. Аналізуючи цей граф, дослідники можуть виявляти певні шаблони, які дозволяють пов'язувати адреси із реальними особами. Наприклад, можна знайти групи адрес, які часто взаємодіють один з одним, а це може вказувати на їхню приналежність одному користувачеві або організації.

Ще одним методом деанонімізації є відстеження IP-адрес. Ця техніка включає моніторинг IP-адрес, які підключаються до вузлів Bitcoin при відправленні та отриманні транзакцій. Шляхом аналізу та зв'язування IP-адрес з конкретними транзакціями можна ідентифікувати реальні особи, які беруть участь у цих транзакціях. Наприклад, якщо певна IP-адреса використовується для надсилання та отримання значної кількості транзакцій Bitcoin, можна припустити, що вона належить активному користувачеві мережі.

Також для деанонімізації мережі Bitcoin можна використовувати зовнішні джерела інформації. Наприклад, дані із соціальних мереж можуть бути використані для зв'язування адрес Bitcoin з реальними особами. Якщо, наприклад, биткойн-адреса користувача пов'язані з обліковим записом у соціальній мережі, що містить його справжнє ім'я, можна дійти невтішного висновку про його істинної

особистості. Це може здійснюватися шляхом аналізу публічно доступної інформації та зв'язування її з відповідними адресами Bitcoin.

Важливо, що ці методи деанонімізації мають обмеження і потребують певних умов успішної ідентифікації користувачів. Наприклад, використання біткойн-міксерів (Bitcoin mixers) може утруднити зв'язування транзакцій з певними користувачами, оскільки вони заважають трасування шляху грошових коштів. Також використання нових адрес Bitcoin для кожної транзакції може знизити ймовірність успішної деанонімізації, оскільки це ускладнює аналіз та зв'язування адрес.

Також треба зазначити машинне навчання, воно відіграє важливу роль у деанонімізації мережі Bitcoin та виявленні зв'язків між адресами Bitcoin та реальними користувачами. Методи машинного навчання дозволяють аналізувати великі обсяги даних та виявляти приховані патерни, кореляції та взаємозв'язки, які можуть бути недоступними для традиційних аналітичних підходів.

Одним із прикладів використання машинного навчання у деанонімізації мережі Bitcoin є застосування класифікаторів та нейронних мереж. Для цього моделі машинного навчання навчаються на історичних даних транзакцій, щоб виявити загальні характеристики та патерни, які можуть вказувати на зв'язок між адресами Bitcoin та реальними користувачами.

Наприклад, модель може навчитися визначати, які групи адрес часто взаємодіють один з одним або які транзакції часто пов'язані з певними IP-адресами. Це дозволяє виявити потенційні зв'язки та встановити взаємозв'язки між адресами Bitcoin та реальними користувачами. Такі моделі можуть бути навчені на різних ознаках, таких як час транзакцій, суми переказів, географічне розташування адрес та інші параметри, які можуть допомогти в ідентифікації та аналізі даних.

Загалом машинне навчання надає потужний інструментарій для деанонімізації мережі Bitcoin. Воно дозволяє аналізувати дані у великому обсязі, виявляти приховані патерни та зв'язки між адресами Bitcoin та реальними користувачами. Це сприяє підвищенню ефективності розслідувань та забезпеченню

безпеки в контексті використання криптовалюти. Однак слід зазначити, що використання машинного навчання у деанонімізації мережі Bitcoin також викликає питання конфіденційності та етики, оскільки може бути використане для небажаного відстеження та порушення приватності користувачів.

Загалом деанонімізація мережі Bitcoin є складним процесом, і її успішність залежить від багатьох факторів. Користувачі, які бажають захистити свою конфіденційність, можуть вживати заходів, таких як використання біткойн-міксерів та нових адрес для кожної транзакції, щоб ускладнити деанонімізацію своїх дій.

## 2.2 Загальні технології розробки системи деаномізації блокчейн-транзакцій

Для вирішення задачі деанонімізації у блокчейні використовуються різні інформаційні технології. Однією з ключових технологій є блокчейн-платформа, яка забезпечує надійність та прозорість зберігання даних. Існують різні блокчейн-платформи, такі як Ethereum, Hyperledger Fabric, Corda та інші, які можуть бути використані у процесі розробки.

Для реалізації алгоритмів деанонімізації блокчейн транзакцій та обфускації даних можна скористатися різними мовами програмування та технологіями, наприклад мовою програмування C#, яка у поєднанні з платформою .NET надає потужні інструменти розробки додатків. Для роботи з блокчейн транзакціями можна використовувати бібліотеку NBitcoin, яка є потужним інструментом для роботи з Bitcoin транзакціями та адресами. Вона надає різні функції та класи, які дозволяють легко взаємодіяти з блокчейном Bitcoin. Нижче наведено деякі основні можливості та функції, що надаються бібліотекою NBitcoin.

NBitcoin дозволяє створювати та маніпулювати Bitcoin-транзакціями. За допомогою цієї бібліотеки можна створювати нові транзакції, додавати входи та виходи, встановлювати комісії, підписувати транзакції та надсилати їх до мережі Bitcoin. Бібліотека надає можливість генерувати нові Bitcoin-адреси, перевіряти

їхню валідність, отримувати інформацію про публічний ключ та приватний ключ, а також виконувати інші операції, пов'язані з адресами. Також вона надає функціональність для читання та обробки даних блоків та транзакцій у блокчейні Bitcoin, отримувати інформацію про блоки, транзакції, входи та виходи, а також виконувати різні операції аналізу даних. Бібліотека забезпечує доступ до криптографічних операцій, пов'язаних з Bitcoin, таких як створення та перевірка цифрових підписів, шифрування та дешифрування даних, а також хешування. Бібліотека дозволяє встановлювати з'єднання з вузлами блокчейну Bitcoin, відправляти та отримувати дані через мережу Bitcoin, а також працювати з різними блокчейн-експлорерами для отримання інформації про транзакції та блоки.

Бібліотека NBitcoin є відкритим вихідним кодом і широко використовується у спільноті розробників Bitcoin. Вона надає гнучкий та потужний інструментарій для роботи з блокчейном Bitcoin на платформі .NET з використанням мови програмування C#.

ASP.NET є потужним фреймворком для розробки веб-застосунків на платформі .NET. Він надає різні інструменти та функціональності, які допомагають розробникам створювати ефективні, масштабовані та безпечні веб-додатки. Ось деякі ключові особливості та можливості фреймворку ASP.NET:

– модель MVC: ASP.NET підтримує модель MVC, яка забезпечує поділ програми на моделі (бізнес-логіку та дані), подання (візуальне подання даних) та контролери (керування потоком даних та взаємодія з користувачем). Це дозволяє розробникам створювати добре організовані та легко супроводжувані веб-додатки;

– розробка веб-сторінок: ASP.NET надає потужний набір елементів керування (controls) та серверних компонентів, які спрощують розробку веб-сторінок. Розробники можуть використовувати готові елементи управління для створення інтерактивних інтерфейсів користувача, управління валідацією даних, працювати з формами і т.д;

– розробка веб-служб: Фреймворк ASP.NET забезпечує підтримку розробки веб-служб (Web Services) за допомогою стандартів SOAP та REST. Це

дозволяє створювати служби, які можуть обмінюватися даними та функціональністю з іншими програмами через мережу;

– аутентифікація та авторизація: ASP.NET надає інтегровану підтримку автентифікації та авторизації користувачів. Розробники можуть легко налаштувати систему автентифікації для перевірки автентичності користувачів та контролю доступу до різних частин програми;

– масштабованість та продуктивність: Фреймворк ASP.NET має потужні інструменти для оптимізації продуктивності веб-додатків. Він підтримує кешування даних, керування станом, масштабування програм на різні сервери та інші техніки, які дозволяють створювати високопродуктивні веб-додатки;

– інтеграція з іншими технологіями: ASP.NET дозволяє легко інтегрувати інші технології та інструменти, такі як бази даних (наприклад, Microsoft SQL Server), JavaScript-бібліотеки (наприклад, jQuery), хмари (наприклад, Azure) та інші.

Фреймворк ASP.NET є основою для створення різноманітних веб-застосунків, включаючи електронну комерцію, соціальні мережі, корпоративні портали та інші. Він має багатий набір інструментів та можливостей, які дозволяють розробникам створювати потужні та інтуїтивно зрозумілі веб-додатки на платформі .NET з використанням мови програмування C#.

Ще однією мовою програмування, яка дозволить Python [35] пропонує кілька бібліотек для роботи з Bitcoin-транзакціями та блокчейном Bitcoin. Однією з таких бібліотек є BitcoinLib. BitcoinLib надає простий і зручний інтерфейс для роботи з Bitcoin-транзакціями та блокчейном. За допомогою цієї бібліотеки ви можете створювати нові транзакції, підписувати їх з використанням приватних ключів, а також надсилати та отримувати транзакції через мережу Bitcoin. BitcoinLib також надає можливість працювати з гаманцями Bitcoin, створювати нові адреси та перевіряти баланси.

Ще однією популярною бібліотекою для роботи з Bitcoin у Python є python-bitcoinlib. Вона надає широкий набір функцій для роботи з Bitcoin-транзакціями,

блоками та мережею Bitcoin. Python-bitcoinlib дозволяє створювати та обробляти транзакції, працювати з мережею Bitcoin та витягувати інформацію про блоки та транзакції.

Для завдань машинного навчання, пов'язаних з аналізом транзакцій та деанонімізації даних у блокчейні, можна використовувати бібліотеку TensorFlow. TensorFlow – це потужна бібліотека з відкритим вихідним кодом, розроблена Google, яка надає широкі можливості для створення та навчання нейронних мереж. Вона має гнучку архітектуру, що дозволяє працювати з різними моделями машинного навчання, включаючи глибоке навчання. TensorFlow пропонує безліч інструментів і функцій для обробки даних, створення моделей, навчання та оцінки їх продуктивності.

Іншою популярною бібліотекою для машинного навчання Python є scikit-learn. Scikit-learn надає широкий спектр алгоритмів машинного навчання, таких як класифікація, регресія, кластеризація та багато інших. Вона має простий та інтуїтивно зрозумілий інтерфейс, що робить її дуже доступною для початківців. Scikit-learn також пропонує інструменти для обробки даних, вибору ознак, оцінки моделей і виконання інших завдань, пов'язаних з машинним навчанням.

Використання бібліотек TensorFlow та scikit-learn у Python дозволяє проводити різні аналізи даних, включаючи деанонімізацію блокчейн транзакцій. Ви можете використовувати ці бібліотеки для створення моделей машинного навчання, які навчатимуться на історичних даних транзакцій та передбачатимуть різні атрибути чи поведінку у блокчейні.

JavaScript є потужною мовою програмування, що широко застосовується для розробки веб-додатків. У контексті блокчейну, JavaScript дозволяє створювати інтерактивні інтерфейси користувача і взаємодіяти з блокчейнами Bitcoin і Ethereum через веб-інтерфейс.

Бібліотеки для роботи з блокчейном було описано нижче:

– BitcoinJS – це бібліотека, яка дозволяє працювати з Bitcoin-транзакціями та адресами у блокчейні Bitcoin. Вона надає широкий набір функцій

для роботи з Bitcoin-транзакціями, включаючи створення нових транзакцій, підпис та верифікацію транзакцій, роботу з адресами та ключами, а також можливість читання даних із блокчейну;

– Web3.js – це бібліотека, розроблена для взаємодії з блокчейном Ethereum. Вона надає зручний API для виконання операцій смарт-контрактів, відправлення транзакцій, отримання даних із блокчейну та багато іншого. Web3.js також забезпечує підтримку різних провайдерів блокчейну, таких як Infura або локальний вузол Geth або Parity.

Бібліотеки для машинного навчання:

– TensorFlow.js – це бібліотека машинного навчання, повністю реалізована на JavaScript. Вона дозволяє розробляти та виконувати моделі машинного навчання прямо у браузері або на сервері з використанням Node.js. TensorFlow.js підтримує різні типи моделей, включаючи нейронні мережі, та надає багаті можливості для обробки даних, тренування моделей та виконання прогнозування;

– Brain.js – це легковажна бібліотека для розробки нейронних мереж на JavaScript. Вона надає простий API для створення та навчання простих нейронних мереж, включаючи підтримку різних типів шарів, активаційних функцій та зворотного розповсюдження помилки. Brain.js добре підходить для вирішення завдань класифікації та прогнозування на невеликих наборах даних.

Бібліотеки для кластеризації:

– ML.js – це бібліотека машинного навчання, що включає інструменти кластеризації даних. Вона надає різні алгоритми кластеризації, включаючи популярні методи, такі як k-means, DBSCAN та ієрархічна кластеризація. ML.js забезпечує зручний API для обробки даних, виконання кластеризації та оцінки якості кластерів;

– Clusterize.js – це бібліотека, призначена для візуалізації та інтерактивної роботи з кластеризованими даними. Вона дозволяє відображати кластери на веб-сторінці та взаємодіяти з ними, надаючи функції групування, фільтрації та навігації

за даними. Clusterize.js має простий та інтуїтивно зрозумілий API, що спрощує роботу з кластеризованими даними у веб-додатках.

Для взаємодії з блокчейном Ethereum за допомогою мови програмування JavaScript використовують бібліотеку Web3.js. Web3.js надає зручний API для роботи зі смарт-контрактами у мережі Ethereum. Він дозволяє надсилати транзакції на виконання смарт-контрактів, отримувати інформацію про блоки, викликати методи смарт-контрактів та виконувати всі можливі операції, пов'язані з Ethereum. За допомогою Web3.js можна створювати децентралізовані програми (DApps), які взаємодіють із контрактами у блокчейні Ethereum.

Децентралізовані програми (DApps) - це програмні програми, які працюють на блокчейні або іншій децентралізованій мережі. На відміну від традиційних централізованих програм, які працюють на центральних серверах та контролюються однією організацією, DApps засновані на принципах децентралізації, прозорості та автономності.

Важливими особливостями DApps є:

– DApps виконуються в блокчейні або іншій децентралізованій мережі, де немає центрального вузла контролю. Натомість, вони працюють на безлічі вузлів, які взаємодіють між собою для досягнення згоди та підтримки цілісності даних;

– усі операції та дані в DApps зазвичай відкриті та доступні для перегляду всіх учасників мережі. Це дозволяє досягти високого ступеня прозорості та довіри, оскільки будь-хто може перевірити та підтвердити транзакції та стан програми;

– DApps зазвичай працюють на основі смарт-контрактів, які є програмними кодами, виконання яких автоматично виконується за певних умов. Це дозволяє DApps функціонувати автономно без необхідності централізованого керування;

– використання блокчейну та криптографії в DApps забезпечує високий рівень безпеки. Транзакції та дані захищені з використанням шифрування, а цілісність інформації підтверджується за допомогою алгоритмів консенсусу та перевірки вузлів мережі.

DApps можуть бути розроблені для різних цілей, включаючи фінансові послуги, соціальні мережі, ігри, цифрові ринки та багато іншого. Вони дозволяють користувачам взаємодіяти один з одним і проводити транзакції без посередників, забезпечуючи більшу свободу, прозорість та контроль над їхніми даними та фінансами.

Для розробки программного забезпечення для деанонімізації в блокчейні Ethereum та написання смарт-контрактів необхідно опанувати мову Solidity. Solidity – це мова програмування, спеціально розроблена для написання смарт-контрактів у блокчейні Ethereum. Він надає синтаксис та можливості для визначення структур даних, функцій та операцій, необхідних для роботи з контрактами. За допомогою Solidity можна розробляти власні смарт-контракти, що реалізують логіку деанонімізації або інші необхідні функції.

Для розробки на Solidity пропонується використовувати інструменти, такі як Remix IDE та Truffle Framework. Remix IDE - це онлайн-середовище розробки, яке надає редактор коду, компілятор та відладчик для Solidity. Вона забезпечує зручне середовище для розробки, тестування та розгортання смарт-контрактів у блокчейні Ethereum. Truffle Framework, з іншого боку, є фреймворком для розробки, тестування та управління смарт-контрактами у блокчейні Ethereum. Він надає потужні інструменти для автоматизаціїа розробки, компіляції, розгортання та тестування смарт-контрактів. Truffle також інтегрується з Remix IDE, що полегшує розробку та налагодження контрактів.

Використання JavaScript, бібліотек BitcoinJS та Web3.js, а також мови Solidity та інструментів розробки Remix IDE та Truffle Framework дозволяє вам створювати потужні системи для деанонімізації та роботи зі смарт-контрактами у блокчейні Bitcoin та Ethereum.

Всі ці моделі, методи та інформаційні технології відіграють важливу роль у вирішенні задачі деанонімізації у блокчейні та забезпечують надійний захист конфіденційності користувачів. Вони дозволяють ефективно аналізувати

транзакції, обробляти дані та взаємодіяти з блокчейном, забезпечуючи безпеку та анонімність учасників.

## 2.3 Висновки до розділу 2

Використання мови програмування C# у розробці системи деанонімізації в блокчейні надає низку переваг та можливостей. Деякі з них включають:

– бібліотеки та фреймворки: Існують різні бібліотеки та фреймворки мовою C#, такі як NBitcoin, які надають функціональність для роботи з блокчейн-транзакціями та адресами. Ці інструменти спрощують розробку та обробку даних у блокчейні;

– інтеграція з платформою .NET: Мова програмування C# тісно інтегрована з платформою .NET, що забезпечує широкі можливості для створення різних типів програм, включаючи веб-додатки та програми для роботи з блокчейном;

– робота з різними блокчейн-платформами: C# дозволяє взаємодіяти з різними блокчейн-платформами, включаючи Ethereum, Hyperledger Fabric та Corda. Це відкриває можливості для створення програм, що використовують різні блокчейн-рішення залежно від конкретних потреб проекту;

Однак при виборі методів деанонімізації важливо враховувати вимоги проекту та бажаний рівень анонімності. Методи аналізу графа транзакцій, змішування чи використання приватних блокчейнів можуть бути ефективними засобами деанонімізації в блокчейні.

Загалом використання C# у розробці системи деанонімізації в блокчейні забезпечує можливості для створення надійної та безпечної системи, яка може ефективно забезпечити анонімність користувачів при роботі з блокчейн-транзакціями.

# 3 ПРОЕКТУВАННЯ СИСТЕМИ З ВИКОРИСТАННЯМ БЛОКЧЕЙН-ТРАНЗАКЦІЙ

## 3.1 Опис вхідних даних, необхідних для функціонування системи

Вхідні дані інформаційної системи для деанонімізації блокчейн транзакцій включають інформацію про сам блокчейн, блоки і транзакції, а також історичні дані та атрибути, пов'язані з кожною транзакцією. Кожна транзакція містить різні атрибути, такі як ідентифікатор транзакції, адреса відправника, адреса одержувача, сума транзакції, дата та час проведення транзакції, комісія та інші деталі. Важливо мати доступ до повної та достовірної історії блокчейн транзакцій, щоб ефективно проводити аналіз та деанонімізацію.

Блокчейн – це основна структура даних, що використовується в інформаційних системах для зберігання та впорядкування транзакцій. Він є ланцюжком блоків, де кожен блок містить набір транзакцій або інших даних.

Кожен блок блокчейне має свій унікальний ідентифікатор, званий хеш блоку. Хеш –÷ це унікальний рядок символів, який представляє вміст блоку. Хеш генерується з використанням криптографічних хеш-функцій, таких як SHA-256 (Secure Hash Algorithm 256-bit), і служить для перевірки цілісності даних у блоці.

Крім хеша, кожен блок зберігає посилання на попередній блок у ланцюжку, утворюючи таким чином ланцюжок зв'язаних блоків. Посилання на попередній блок також називається хеш попереднього блоку. Це забезпечує цілісність блокчейна, оскільки зміна даних в одному блоці автоматично призведе до зміни хешей наступних блоків у ланцюжку, що виявляється та запобігається мережею.

Коли нова транзакція додається до блокчейну, вона спочатку перевіряється та підтверджується мережею. Потім транзакція додається до нового блоку, який створюється майнерами або учасниками мережі. Після додавання блоку в ланцюжок він стає незмінним і надійно зберігає інформацію про транзакцію.

Блокчейн забезпечує ряд переваг у порівнянні з традиційними централізованими структурами даних. По-перше, блокчейн є децентралізованою

системою, де немає центрального вузла контролю. Це означає, що дані зберігаються та керуються безліччю вузлів, що підвищує безпеку та надійність системи.

По-друге, блокчейн забезпечує прозорість та незаперечність даних. Оскільки кожен блок посилається на попередній блок і містить хеш попереднього блоку, зміна даних в одному блоці призведе до зміни хеш наступних блоків, що виявляється і відкидається мережею. Це робить блокчейн непохитним до підробки даних і забезпечує довіру до системи.

Блокчейн також має властивість незмінності даних. Після додавання блоку в ланцюжок він стає незмінним і не може бути видалено або змінено без згоди більшості учасників мережі. Це робить блокчейн надійним та стійким до цензури.

Проте блокчейн також має обмеження. Він потребує значних обчислювальних ресурсів для створення та підтримки блоків, а також великого обсягу зберігання даних. Блокчейн може бути неефективним для обробки великих обсягів транзакцій у реальному часі.

Загалом блокчейн є потужною структурою даних, яка забезпечує безпечне та надійне зберігання та впорядкування транзакцій. Він знайшов широке застосування в різних галузях, таких як фінанси, постачання, охорона здоров'я та багато іншого, і продовжує розвиватися і змінюватися для задоволення потреб різних індустрій.

Для ефективного зберігання та доступу до великих обсягів даних використовуються бази даних. База даних може бути використана для зберігання та організації інформації про блокчейн транзакції та пов'язані з ними атрибути. База даних може бути побудована на основі реляційної моделі або використовувати інші моделі даних, такі, як NoSQL або графові бази даних.

Важливим аспектом процесу деанонімізації є використання алгоритмів та моделей аналізу даних. Це можуть бути алгоритми машинного навчання, статистичні моделі чи алгоритми кластеризації. Алгоритми машинного навчання дозволяють виявляти приховані патерни та зв'язки між транзакціями, що сприяє деанонімізації та розкриттю справжньої ідентичності учасників транзакцій.

Таким чином, інформаційна система для деанонімізації блокчейн транзакцій працює з різними вхідними даними, включаючи блокчейн, блоки та транзакції, а також історичні дані та атрибути. Структури даних, такі як блокчейн та бази даних, забезпечують зберігання та організацію цих даних. Алгоритми аналізу даних дозволяють проводити деанонімізацію та розкриття ідентичності учасників транзакцій. Ці компоненти взаємодіють в інформаційній системі задля досягнення мети деанонімізації у блокчейні.

### 3.2 Формування структури системи

У процесі проектування інформаційної системи для деанонімізації блокчейн транзакцій було приділено особливу увагу деталям та обліку різних аспектів, що забезпечують високу ефективність, надійність та зручність використання системи.

В інформаційній системі для деанонімізації блокчейн транзакцій було використано клієнт-серверну архітектуру, що дозволяє ефективно обробляти запити від клієнта та надавати йому результати деанонімізації. Для взаємодії між клієнтом та сервером був використаний RESTful підхід.

REST (Representational State Transfer) є архітектурним стилем, що визначає принципи побудови розподілених систем. У REST-архітектурі сервер надає ресурси (наприклад, дані транзакцій) і клієнт взаємодіє з ними, відправляючи запити та отримуючи відповіді. RESTful API (Application Programming Interface) надає інтерфейс для доступу до ресурсів сервера за допомогою протоколу HTTP.

Використання RESTful API в інформаційній системі дозволяє клієнтській програмі відправляти запити на сервер для завантаження блокчейн даних, вибору параметрів деанонімізації та отримання результатів. Клієнт та сервер взаємодіють за допомогою стандартних HTTP-методів, таких як GET, POST, PUT та DELETE, для виконання різних операцій над ресурсами.

Переваги RESTful API включають простоту та зрозумілість інтерфейсу, стандартизацію та незалежність від платформи. RESTful API дозволяє розробляти клієнтські програми на різних платформах, таких як веб, мобільні пристрої або

настільні комп'ютери, що забезпечує гнучкість та розширюваність системи. Крім того, RESTful API має масштабованість і зручність тестування, що спрощує розробку та супровід інформаційної системи.

Однак, існують і деякі обмеження та недоліки RESTful API. Наприклад, обмеження протоколу HTTP можуть бути несумісними з деякими особливостями програми. Також, RESTful API може бути менш ефективним у разі великих обсягів даних або складних операцій, що потребують множинних запитів до сервера.

В цілому, використання RESTful API в інформаційній системі на основі клієнт-серверної архітектури забезпечує ефективну взаємодію між компонентами системи, зручність використання та розширюваність. Це дозволяє користувачам легко взаємодіяти з системою, виконувати операції з блокчейн даними та отримувати точні результати деанонімізації.

Для розробки програми була обрана мова програмування C#, яка надає широкий набір інструментів для роботи з блокчейн транзакціями та забезпечує високу продуктивність.

ASP.NET, а саме .NET Web API, був використаний для розробки бек-енду частини інформаційної системи. .NET Web API є фреймворком, призначеним для створення веб-служб, які можуть обробляти HTTP-запити та обмінюватися даними з клієнтськими додатками.

ASP.NET (Active Server Pages.NET) є фреймворком, розробленим компанією Microsoft для створення веб-додатків та веб-сервісів. Він є однією з ключових технологій у розробці веб-додатків на платформі .NET. ASP.NET надає програмістам потужні інструменти та функціональність для розробки динамічних, масштабованих та безпечних веб-додатків.

Історія ASP.NET розпочалася у 2002 році з виходом першої версії фреймворку. Вона була розроблена як еволюція попередньої технології ASP (Active Server Pages), що надає можливість створення динамічних веб-сторінок. Однак, у порівнянні з ASP, ASP.NET надає більш потужні інструменти, більш високу продуктивність та більшу гнучкість розробки веб-додатків.

Однією з ключових переваг ASP.NET є його об'єктно-орієнтована модель програмування. Фреймворк заснований на .NET, що дозволяє розробникам використовувати мови програмування, такі як C# і VB.NET, для створення веб-додатків. Це надає гнучкість та можливість вибору мови залежно від потреб проекту.

ASP.NET підтримує різні моделі розробки, включаючи MVC (Model-View-Controller) та Web Forms. MVC надає більш строгу архітектуру, розділяючи додаток на модель даних, представлення інтерфейсу користувача і контролери для обробки запитів. Web Forms, з іншого боку, надає швидший спосіб розробки веб-застосунків, абстрагуючись від складнощів веб-програмування.

ASP.NET також володіє широким набором вбудованих компонентів та бібліотек для розробки веб-додатків. Він підтримує роботу з базами даних, включає засоби для обробки введення користувача, авторизації та аутентифікації, а також надає інструменти для оптимізації продуктивності та безпеки веб-додатків.

Однією з ключових переваг ASP.NET є його масштабованість. Фреймворк надає можливості для горизонтального та вертикального масштабування веб-застосунків, дозволяючи їм обробляти великі обсяги трафіку та залишатися чуйними навіть при високих навантаженнях. Це особливо важливо для сучасних веб-застосунків, які часто стикаються з зростаючою кількістю користувачів і вимогами швидкої обробки запитів.

Однак, існують деякі недоліки ASP.NET. Одним із них є його складність для новачків. Вивчення фреймворку та його компонентів може вимагати час та зусилля. Крім того, деякі функціональності та можливості ASP.NET можуть вимагати додаткових ліцензій або платних компонентів.

В цілому, ASP.NET є потужним та гнучким фреймворком для розробки веб-додатків. Він надає розробникам безліч інструментів та функціональності для створення ефективних, безпечних та масштабованих веб-додатків. За його допомогою можна створювати різноманітні веб-сайти, електронні комерції,

корпоративні портали та інші інтерактивні веб-застосунки, що задовольняють потреби користувачів у сфері веб-розробки.

.NET Web API є фреймворком для розробки веб-служб, що використовують протокол HTTP для обміну даними між клієнтськими програмами та сервером. Він є частиною платформи .NET і надає розробникам інструменти та функціональність для створення потужних та масштабованих веб-сервісів.

Історія .NET Web API розпочалася у 2012 році з випуску версії 4.0 платформи ASP.NET. У цій версії був представлений новий фреймворк Web API, який надавав простий та ефективний спосіб створення веб-служб, що наслідують архітектурний стиль REST (Representational State Transfer). Цей стиль має на увазі використання протоколу HTTP для доступу до ресурсів і виконання операцій над ними.

.NET Web API заснований на платформі ASP.NET і включає всі основні компоненти і можливості фреймворку ASP.NET. Він підтримує різні формати даних для обміну інформацією, такі як JSON (JavaScript Object Notation) та XML (eXtensible Markup Language). Це дозволяє клієнтським програмам обмінюватися даними з веб-сервісом у зручному та зрозумілому форматі.

Однією з ключових переваг .NET Web API є його гнучкість та розширюваність. Фреймворк дозволяє розробникам вибирати найбільш підходящі методи обробки HTTP запитів і створення RESTful веб-сервісів. Він також має вбудовану підтримку маршрутизації, що спрощує визначення шляхів URL для доступу до різних ресурсів та операцій.

.NET Web API надає потужні інструменти для обробки та валідації вхідних даних, автентифікації та авторизації користувачів, а також контролю доступу до веб-сервісу. Це забезпечує безпеку та захист даних, що передаються між клієнтом та сервером.

Проте, існують деякі недоліки .NET Web API. Він може бути складним для новачків у веб-розробці, особливо для тих, хто не має досвіду роботи з фреймворком ASP.NET. Крім того, для роботи з .NET Web API потрібна наявність

платформи .NET і інфраструктури, що підтримує, що може обмежити його застосування в деяких середовищах.

В цілому, .NET Web API є потужним та гнучким фреймворком для створення веб-служб, заснованих на архітектурі REST. Він надає розробникам широкий набір інструментів та функціональності для створення ефективних та масштабованих веб-сервісів, що мають високий рівень безпеки та захисту даних. .NET Web API широко застосовується в різних галузях, включаючи веб-розробку, мобільні програми та інтеграцію систем, і продовжує розвиватися та покращуватися з кожним новим випуском платформи .NET.

Фреймворк Angular був обраний для реалізації фронт-енд частини інформаційної системи. Angular – це популярний та потужний інструмент, який полегшує розробку динамічних веб-додатків. Він надає розробникам широкий набір інструментів і функціональних можливостей для створення масштабованих та інтерактивних інтерфейсів користувача.

Однією з ключових переваг Angular є його модульна архітектура. Angular дозволяє розділити програму на окремі модулі, кожен з яких відповідає за певні функціональності або компоненти. Це сприяє покращенню організації коду, повторному використанню компонентів та полегшенню супроводу проекту.

Іншим важливим аспектом Angular є двостороннє зв'язування даних (two-way data binding). Ця функціональність дозволяє автоматично синхронізувати дані між компонентами та поданням, забезпечуючи більш динамічний і чуйний інтерфейс користувача. Також Angular надає можливості для валідації даних та управління формами, що спрощує обробку введення користувача.

Angular також забезпечує підтримку односторінкових додатків (Single Page Applications, SPA) та роутингу. За допомогою роутингового механізму можна створювати безліч сторінок і визначати їх відображення в залежності від URL. Це дозволяє створювати більш складні та багатосторінкові програми зі зручною навігацією між різними компонентами.

Однією з головних переваг Angular є його активна спільнота розробників. Angular має велику спільноту підтримки, де розробники можуть отримати допомогу, обмінятися досвідом та знайти готові рішення для своїх завдань. Це сприяє розвитку та покращенню фреймворку, а також надає безліч документації та посібників, що спрощує початок роботи з Angular.

В цілому, вибір Angular для реалізації фронт-енд частини інформаційної системи забезпечує потужний і гнучкий інструмент для створення сучасних і чуйних інтерфейсів користувача. Angular володіє широким функціональним набором, підтримується активною спільнотою розробників та надає можливості для створення модульних та масштабованих додатків.

Для обробки та аналізу блокчейн даних було обрано відповідні бібліотеки. Бібліотека NBitcoin надає потужні функціональні можливості для роботи з Bitcoin транзакціями та адресами, дозволяючи отримувати необхідну інформацію про транзакції та проводити деанонімізацію. TensorFlow та scikit-learn були обрані для реалізації машинного навчання та аналізу даних, що дозволяє ефективно обробляти та аналізувати великі обсяги транзакційних даних.

В якості бази даних для зберігання та організації даних була обрана реляційна база даних, наприклад Microsoft SQL Server. Реляційні бази даних забезпечують надійність, ефективність доступу до даних та широкий набір функцій для маніпуляції даними.

Плюси вибраних технологій та інструментів:

– клієнт-серверна архітектура забезпечує зручність та простоту використання системи, дозволяючи користувачам легко взаємодіяти з нею;

– мова програмування C# має широкі можливості для роботи з блокчейн транзакціями та забезпечує високу продуктивність системи;

– ASP.NET забезпечує гнучкість розробки веб-додатків та зручну взаємодію з блокчейном;

– Angular забезпечує гнучкість розробки веб-додатків та зручну взаємодію з блокчейном;

– NBitcoin надає потужні функціональні можливості для роботи з Bitcoin транзакціями та адресами, дозволяючи проводити деанонімізацію та отримувати необхідну інформацію про транзакції;

– TensorFlow та scikit-learn дозволяють ефективно обробляти та аналізувати великі обсяги транзакційних даних з використанням машинного навчання.

## 3.3 Висновки до розділу 3

У цьому розділі було проведено проектування інформаційної системи для деанонімізації блокчейн транзакцій. Були враховані різні аспекти, такі як ефективність, надійність та зручність використання системи.

Архітектура системи була заснована на клієнт-серверній моделі, де клієнтська програма взаємодіє з сервером для обробки запитів та отримання результатів деанонімізації. Використання мови програмування C# та фреймворку ASP.NET забезпечило широкі можливості для роботи з блокчейн транзакціями та створення зручного веб-інтерфейсу.

Для роботи з блокчейн транзакціями та аналізу даних були вибрані відповідні бібліотеки, такі як NBitcoin, TensorFlow та scikit-learn. Це дозволяє обробляти та аналізувати великі обсяги даних транзакцій, застосовувати алгоритми машинного навчання та проводити деанонімізацію.

Реляційна база даних, така як Microsoft SQL Server, була обрана для зберігання та організації даних, забезпечуючи ефективний доступ до них під час деанонімізації.

Користувальницький інтерфейс програми розроблено з урахуванням простоти та інтуїтивності. Він дозволяє завантажувати блокчейн дані, вибирати параметри для деанонімізації, відображати результати та візуалізувати їх у зручному форматі.

Проектування інформаційної системи було засноване на використанні сучасних технологій та інструментів, що забезпечує зручність використання, надійність та ефективність роботи системи.

В результаті даного проектування було створено інформаційну систему, здатну проводити деанонімізацію блокчейн транзакцій за допомогою обраних алгоритмів та інструментів.

# 4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 4.1 Засоби розробки

Для розробки програмної частини інформаційної системи були використані різноманітні засоби розробки, які забезпечили ефективну і швидку реалізацію проекту. Основними засобами, вибраними для цього процесу, були мова програмування C# та середовище розробки Rider.

Мова програмування C# (C Sharp) є однією з найпопулярніших мов програмування, що використовуються для розробки додатків на платформі .NET. Вона була створена компанією Microsoft і випущена в 2000 році як частина ініціативи по розширенню можливостей розробки програмного забезпечення для платформи Windows.

Однією з головних переваг мови C# є її потужність і ефективність. Вона пропонує багато продуктивних інструментів і можливостей, які сприяють швидкій і зручній розробці програм. C# підтримує об'єктно-орієнтовану парадигму програмування, що дозволяє організувати код у логічні структури, що забезпечують полегшення розробки, підтримку модульності і повторне використання коду.

Мова C# має сучасний синтаксис, який є читабельним і зрозумілим. Це дозволяє розробникам легко розуміти і модифікувати код, а також спрощує спільну роботу над проектом командою розробників. Крім того, мова C# підтримує різноманітні конструкції, такі як делегати, події, анонімні типи, LINQ (Language Integrated Query) і багато інших, що сприяють зручному і ефективному програмуванню.

Ще однією важливою перевагою мови C# є широкий спектр можливостей, які вона надає розробникам. Вона підтримує розробку різноманітних типів додатків, включаючи веб-додатки, мобільні додатки, десктопні програми, сервіси та інше. Це

дозволяє розробникам працювати над різноманітними проектами і використовувати мову C# для рішення різних завдань.

У контексті розробки інформаційної системи для деанонімізації було обрано мову програмування C# з урахуванням її потужності, ефективності та широкого спектру можливостей. C# дозволив зручно і швидко реалізувати логіку деанонімізації, працювати з базою даних та забезпечити взаємодію з іншими компонентами системи.

Середовище розробки Rider, розроблене компанією JetBrains, було обрано для розробки інформаційної системи з причини його потужності, функціональності та зручного інтерфейсу. Rider є однією з найпопулярніших інтегрованих середовищ розробки для мови C# і надає широкі можливості для комфортної розробки програмного забезпечення.

Однією з головних переваг Rider є його потужність. Він пропонує розширені функціональні можливості, які допомагають розробникам писати код швидше і ефективніше. Наприклад, Rider надає автодоповнення коду, що полегшує написання правильного синтаксису і прискорює процес програмування. Він також включає інструменти для аналізу коду, які допомагають виявляти помилки і покращувати якість програмного забезпечення.

Ще одною перевагою Rider є вбудований налагоджувач, який дозволяє розробникам відстежувати виконання програми, зупиняти його на певних кроках, аналізувати значення змінних і виявляти помилки. Це спрощує процес налагодження програм та допомагає виявляти та виправляти проблеми з кодом.

Окрім цього, Rider підтримує інтеграцію з різними системами контролю версій, такими як Git, SVN і Mercurial. Це дозволяє команді розробників зручно працювати над проектом, зберігати версії коду і відстежувати зміни.

Загальний інтерфейс Rider є зручним і інтуїтивно зрозумілим. Він має добре організовану структуру, що дозволяє легко навігувати по проекту, переглядати його структуру і швидко знаходити потрібні файли та компоненти.

Завдяки всім цим функціональним можливостям і зручному інтерфейсу, Rider стає потужним інструментом для розробки програмного забезпечення на мові C#. Використання цього середовища дозволило забезпечити зручну, продуктивну та якісну розробку програмної частини інформаційної системи.

Для розробки веб-інтерфейсу проекту був використаний фреймворк Angular. Angular є потужним інструментом для створення односторінкових додатків, який надає широкі можливості для розробки веб-інтерфейсу. Він пропонує компонентний підхід до побудови інтерфейсу, що дозволяє розбити його на незалежні компоненти і забезпечує легкість управління та модульність коду. Angular також надає потужні інструменти для обробки запитів до сервера, маршрутизації та валідації даних.

Для забезпечення взаємодії з базою даних була використана бібліотека Entity Framework. Entity Framework є об'єктно-реляційним маппером (ORM), який дозволяє легко взаємодіяти з базою даних з використанням об'єктно-орієнтованого підходу. За допомогою Entity Framework можна виконувати операції з даними, такі як створення, читання, оновлення і видалення, а також виконувати складні запити до бази даних. Використання Entity Framework спрощує роботу з даними і дозволяє швидко реалізувати потрібні функції.

ML.Net є бібліотекою машинного навчання для платформи .NET, яка надає можливості з розробки моделей машинного навчання та їх використання в проекті. За допомогою ML.Net можна створювати моделі класифікації, регресії, кластеризації та інші, що дозволяє реалізувати аналітичні функції і передбачення в системі.

Для роботи з блокчейном та обробки транзакцій була використана бібліотека NBitcoin. NBitcoin є потужним інструментом для розробки додатків, пов'язаних з блокчейном та криптовалютами, на платформі .NET. Вона надає зручний API для створення, підписування та перевірки транзакцій, роботи з адресами та ключами, а також для отримання інформації про блоки, транзакції та інші дані блокчейну.

Використання NBitcoin дозволяє забезпечити надійну та безпечну обробку блокчейн даних в системі.

Для розробки серверної частини проекту був використаний фреймворк ASP.NET Web API. ASP.NET Web API є розширенням фреймворку ASP.NET, яке дозволяє легко створювати веб-сервіси і реалізовувати API для взаємодії з клієнтськими додатками. Він надає потужні інструменти для обробки HTTP-запитів, маршрутизації, серіалізації та десеріалізації даних, аутентифікації і авторизації користувачів, а також для роботи з базою даних. Використання ASP.NET Web API дозволило реалізувати зручний та ефективний спосіб взаємодії між клієнтом та сервером в системі.

В процесі розробки було використано систему контролю версій Git. Git є однією з найпопулярніших і надійних систем контролю версій, яка дозволяє відстежувати зміни в коді, зберігати його історію та співпрацювати з іншими розробниками. Використання Git дозволило зручно працювати з кодом, створювати гілки для розробки різних функціональностей, вносити та об'єднувати зміни, а також вирішувати конфлікти при одночасній роботі над проектом. Git також надає можливість відновлення попередніх версій коду в разі потреби.

Крім того, для забезпечення зручного розгортання та керування залежностями був використаний пакетний менеджер NuGet. NuGet дозволяє легко встановлювати, оновлювати та керувати бібліотеками та компонентами, необхідними для проекту. Використання NuGet спрощує процес установки та підтримки залежностей, забезпечує актуальні версії бібліотек і полегшує роботу з ними.

### 4.2 Вимоги до технічного забезпечення

Для належної роботи даного програмного продукту необхідно мати певну технічну і програмну конфігурацію. Розглянемо детальніше вимоги до технічного забезпечення.

Додаток складається з двох частин - клієнтської та бекенд-частин. Клієнтська частина, яка реалізована з використанням фреймворка Angular, може бути запущена на будь-якій платформі завдяки його кросплатформенності. Тим самим, для запуску клієнтської частини немає обмежень щодо операційної системи. Бекенд-частина, написана на мові C# та використовується фреймворк .NET, також є кросплатформенною, що дозволяє запускати її на різних операційних системах.

Програмний продукт підтримує різні архітектури процесорів, включаючи x64 та ARM. Таким чином, можна використовувати будь-який сумісний процесор для запуску системи, залежно від доступних можливостей та вимог проекту.

Оперативна пам'ять: Для належної роботи системи рекомендується мати не менше 8 ГБ оперативної пам'яті. Це дозволить забезпечити ефективну виконавчу діяльність, швидку обробку запитів та зручну роботу з програмним продуктом.

Для збереження програмного забезпечення, баз даних та інших ресурсів необхідний достатній обсяг вільного дискового простору. Рекомендується мати не менше 10 ГБ вільного дискового простору для ефективної роботи системи.

Для роботи системи необхідно мати локально скачану біткоїн-мережу або її частину. Розмір біткоїн-мережі може варіюватися від 1 ГБ до 140 ГБ або навіть більше, залежно від обсягу транзакцій та інших факторів. Наявність відповідного дискового простору для збереження біткоїн-мережі є необхідною умовою для коректної роботи системи.

Для користувачів системи рекомендується використовувати останні версії сучасних веб-браузерів, таких як Google Chrome, Mozilla Firefox або Microsoft Edge. Це забезпечить найкращу сумісність з програмним продуктом та забезпечить безпеку при використанні.

Для повноцінного користування системою необхідно мати ком’ютерну мишку, клавіатуру та монітор. Ці елементи комп'ютерної периферії є основними для взаємодії з програмним продуктом та забезпечують комфортну роботу з системою.

Дотримання цих вимог до технічного забезпечення допоможе забезпечити ефективну розробку, тестування та роботу інформаційної системи з високою продуктивністю та надійністю.

### 4.3 Розробка backend частини програмного застосунку

Створимо застосунок з використанням фреймвору ASP.Net API (рисунок 4.1).

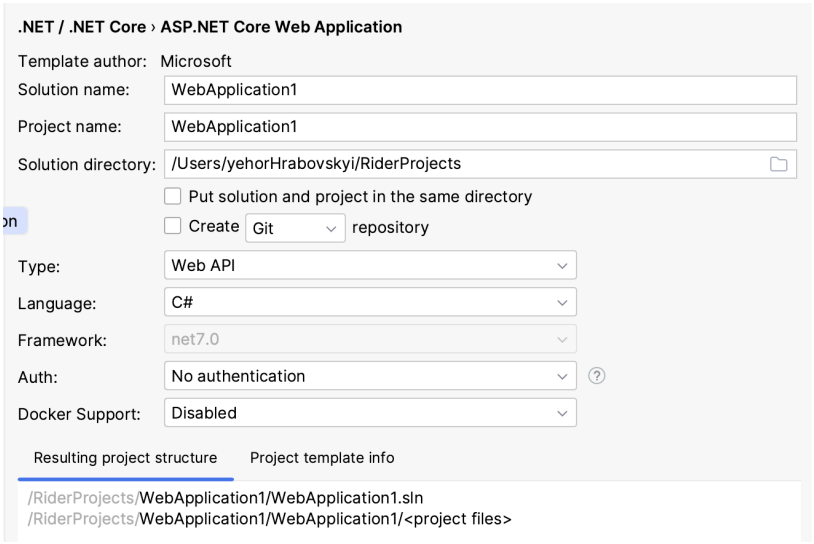


Рисунок 4.1 – Меню створення додатку

Далі треба визначити структуру проекту (рисунок 4.2):

– domain рівень відповідає за зберігання data-класів, які представляють найменші структурні одиниці проекту. Він включає в себе моделі даних, об'єкти-значення та інші сутності, які використовуються в проекті.

– application рівень відповідає за зберігання бізнес-логіки проекту. Він включає в себе сервіси, менеджери, правила бізнесу та інші компоненти, які реалізують функціонал додатку. Цей рівень використовує моделі даних з рівня Domain для виконання бізнес-операцій.

– clustering рівень відповідає за бізнес-логіку кластеризації. Він є незалежним проектом, який може бути відключений без впливу на решту системи.

В цьому проекті знаходяться алгоритми та компоненти, які виконують кластеризацію даних за певними критеріями.

– infrastructure/IOC рівень відповідає за підключення всіх залежностей в проекті. Він з'єднує рівень Application та Clustering, надаючи необхідні компоненти для їх спільної роботи. В цьому проекті можуть бути реалізовані контейнери залежностей (наприклад, IOC-контейнер), налаштування підключення до бази даних та інші компоненти інфраструктури.

– persistence рівень відповідає за роботу з базою даних та логіку збереження даних. Він включає в себе репозиторії, які забезпечують доступ до даних і виконують операції збереження, оновлення та видалення даних в базі.

– API рівень є вхідним рівнем системи і представляє інтерфейс для взаємодії з зовнішніми системами чи користувачами. Він може бути реалізований як веб-сервіс, REST API, GraphQL або будь-який інший механізм комунікації. Цей рівень використовує функціонал з рівня Application для обробки запитів та надання відповідей.

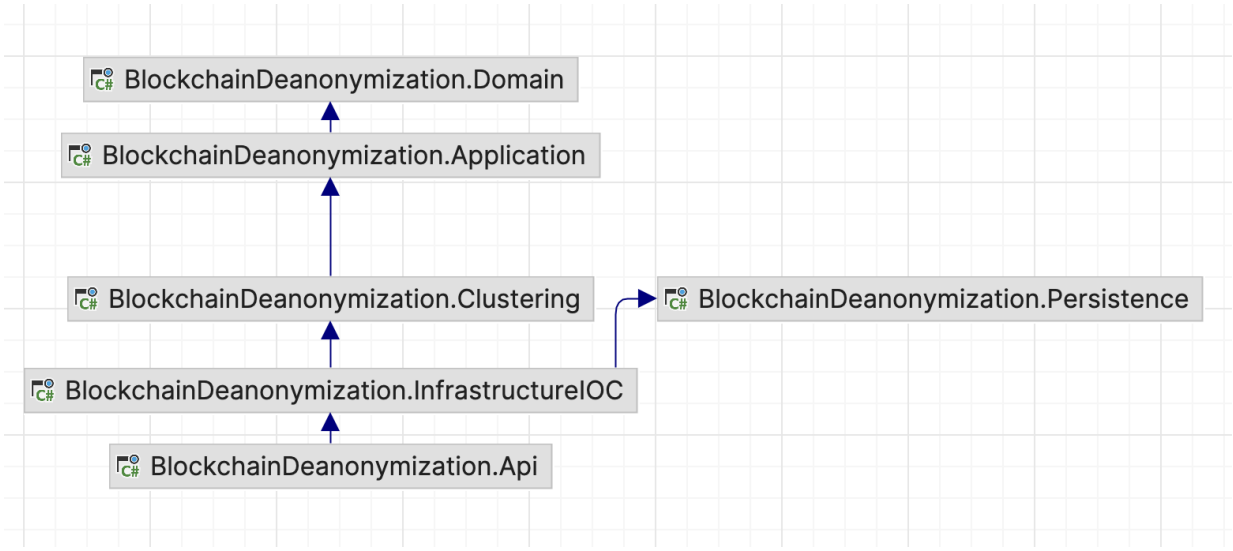


Рисунок 4.2 – Діаграма проектів

Далі треба написати контролер, який буде вхідною точкою проекта. Так як задача передбачає лише отримання кластеризованих даних, то контролер буде

один: TransactionController. Цей клас повинен мати лише один ендроінт – ClusterTransactions, який буде повертати кластеризовані транзакії з кінця блокчейну(перший блок створено 3 грудня 2009 року).

Контролер TransactionController відповідає за обробку HTTP-запитів, пов'язаних з транзакціями в контексті деанонімізації блокчейну. Цей контролер використовується для взаємодії з клієнтами та надання їм необхідних послуг.

При створенні екземпляру контролера в конструктор передається залежність ITransactionService, яка відповідає за логіку обробки транзакцій.

Метод ClusterTransactions (рисунок 4.3) є обробником POST-запитів на маршрут "/transaction/cluster". Цей метод приймає об'єкт GetClustersRequest, який містить необхідні параметри для кластеризації транзакцій.

Внутрішній об'єкт _transactionService викликає метод GetClusteredTransaction, який виконує кластеризацію транзакцій на основі заданої кількості транзакцій та кількості кластерів. Результат кластеризації зберігається у змінній clusterLabels. Далі метод повертає результат у відповідь HTTP з кодом 200 (OK). Результат кластеризації перетворюється на JSON-рядок з використанням бібліотеки Newtonsoft.Json та форматується з відступами (Formatting.Indented).

```
[HttpPost("cluster")]
 public IActionResult ClusterTransactions(GetClustersRequest request)
 {
     var clusterLabels =
_transactionService.GetClusteredTransaction(request.TransactionCount, request.ClusterCount);

     return Ok(JsonConvert.SerializeObject(clusterLabels, Formatting.Indented));
 }
```

Рисунок 4.3 – Метод ClusterTransactions

BitcoinService є компонентом системи, який забезпечує функціональні можливості для завантаження та аналізу транзакцій з блокчейну Bitcoin з метою їх класифікації за кластерами. Цей клас включає різні методи та використовує додаткові компоненти для досягнення цілей.

Конструктор класу BitcoinService отримує об'єкт RpcSettings, який містить налаштування для підключення до локального вузла Bitcoin. Ці налаштування включають URL-адресу вузла, користувача та пароль. Після ініціалізації RpcSettings, він використовується для встановлення з'єднання з вузлом Bitcoin через протокол RPC.

Метод LoadTransactionData використовує RpcSettings для встановлення з'єднання з вузлом Bitcoin та отримання даних про транзакції з блокчейну. Він приймає кількість транзакцій, які потрібно завантажити, та послідовно отримує транзакції з кожного блоку, щоб отримати повну інформацію.

Спочатку метод перевіряє, чи в базі даних вже є достатня кількість транзакцій. Якщо так, він повертає збережені транзакції з кінця списку. В іншому випадку, він встановлює з'єднання з вузлом Bitcoin та послідовно завантажує транзакції з кожного блоку.

Для кожної завантаженої транзакції виконується аналіз, і дані про неї зберігаються у відповідному форматі TransactionData. Ці дані включають характеристики транзакції, такі як загальна вартість виведення коштів (TotalOut), розмір транзакції (Size) та інформацію про вхідний та вихідний гаманці (InputWallet та OutputWallet). Додатково, використовується допоміжний метод ClusteringHelper для отримання додаткової інформації про гаманці.

Після завантаження і аналізу транзакцій їх дані додаються до списку transactions. Після цього метод повертає цей список, що містить повну інформацію про вказану кількість транзакцій з останніх блоків.

Метод InitializeMLContext створює новий об'єкт MLContext, який використовується для роботи з бібліотекою ML.NET. MLContext є основним об'єктом, який управляє процесом навчання моделей машинного навчання та

виконання прогнозів. Цей метод гарантує належну ініціалізацію MLContext перед його використанням у інших методах.

Метод TrainModel (рисунок 4.4) використовує MLContext та IDataView для навчання моделі кластеризації. Перед навчанням моделі виконується підготовка даних шляхом конкатенації різних атрибутів транзакцій, нормалізації ознак та додавання відповідних перетворень до конвеєра ML.NET. Потім до конвеєра додається алгоритм кластеризації KMeans з вказаною кількістю кластерів. Після навчання модель повертається у вигляді об'єкта ITransformer.

```
  public ITransformer TrainModel(MLContext mlContext, IDataView dataView, int
numberOfClusters)
    {
        var pipeline = mlContext.Transforms
            .Concatenate("Features", "InputAndOutputWallet", "TotalOut", "Size")
            .Append(mlContext.Transforms.NormalizeMinMax("Features"))
            .Append(mlContext.Transforms.Concatenate("Features", "InputAndOutputWallet",
"TotalOut", "Size"))
            .AppendCacheCheckpoint(mlContext)
            .Append(mlContext.Clustering.Trainers.KMeans("Features", numberOfClusters:
numberOfClusters));

        return pipeline.Fit(dataView);;
    }
```

Рисунок 4.4 – TrainModel

Метод MakePredictions (рисунок 4.5) використовує навчену модель для прогнозування кластерів для нових наборів даних, які передаються через IDataView. Він виконує трансформацію даних з використанням моделі та повертає прогнозовані кластери для кожного набору даних.

```
  public IEnumerable<PredictionCluster> MakePredictions(MLContext mlContext, ITransformer
model, IDataView dataView)
    {
        var predictions = model.Transform(dataView);
        var clusterLabels = mlContext.Data.CreateEnumerable<PredictionCluster>(predictions,
reuseRowObject: false);

        return clusterLabels;
    }
```

Рисунок 4.5 – MakePrediction

Компоненти, які використовуються в BitcoinService, включають RPCClient з бібліотеки NBitcoin для взаємодії з блокчейном Bitcoin через протокол RPC. Використання RPCClient дозволяє встановити з'єднання з вузлом Bitcoin та виконувати запити до нього для отримання інформації про блоки та транзакції. Також використовується бібліотека ML.NET для реалізації функціоналу машинного навчання та алгоритмів кластеризації. ML.NET надає зручний спосіб побудови та навчання моделей машинного навчання для аналізу даних.

Загалом, BitcoinService є ключовим компонентом, який забезпечує функціональність для завантаження та аналізу транзакцій з блокчейну Bitcoin з використанням машинного навчання та алгоритмів кластеризації. Він використовує високофункціональні компоненти, такі як RPCClient та ML.NET, для досягнення цілей аналізу та класифікації транзакцій за кластерами.

### 4.4 Розробка frontend частини програмного застосунку

Розробка frontend частини програмного застосунку використовує Angular, який є одним з популярних фреймворків для розробки веб-додатків. В коді представлений компонент DiagramComponent, який відповідає за візуалізацію даних та взаємодію з backend.

Для того, щоб ініцілювати проект треба Node.Js, він включає в себе пакетний менеджер npm, який буде використовуватися для установки залежностей проекту.

Після встановлення Node.Js треба ініцілювати проект, в якому можна писати код (рисунок 4.6).

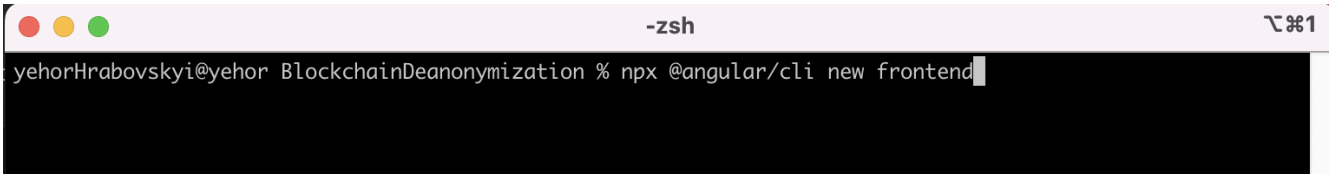


Рисунок 4.6 – Команда створення фронтенд частини додатку

Ця команда створить новий каталог з назвою "frontend" та встановить необхідні пакети та залежності для Angular.

Після ініціалізації треба створити компонент (рисунок 4.7), в якому треба буде писати код зв’язанний з відображенням результатів кластеризації.

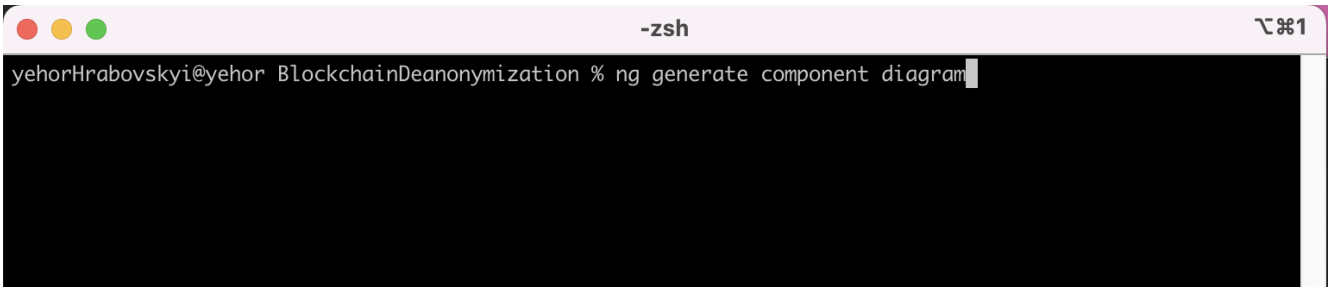


Рисунок 4.7 – Команда створення компоненту

В компоненті DiagramComponent (рисунок 4.8) використовується декоратор @ViewChild для отримання посилання на елемент chartCanvas у шаблоні. Це дозволяє отримати доступ до HTML-елемента <canvas> та його контексту для подальшої візуалізації графіку.

```
export class DiagramComponent implements AfterViewInit {
  @ViewChild('chartCanvas') chartCanvas!: ElementRef<HTMLCanvasElement>;
  chart!: Chart;
  blockHeight: number = 0;
  clusterCount: number = 0;
}
```

Рисунок 4.8 – Клас DiagramComponent

Функція ngAfterViewInit (рисунок 4.9) викликається після того, як компонент та його відображення вже проініціалізовані. У даному випадку, вона викликає функцію createChart для створення графіку за допомогою бібліотеки Chart.js.

```
ngAfterViewInit() {
   this.createChart();
 }
```

Рисунок 4.9 – Метод ngAfterViewInit

Функція fetchData (рисунок 4.10) виконує HTTP POST-запит до сервера за допомогою HttpClient. Вона передає дані blockHeight та clusterCount як частину запиту. Отримані дані обробляються у функції subscribe, де вони передаються до функції updateChart для оновлення графіку з використанням отриманих даних.

```
fetchData() {
    const requestPayload = {
      blockHeight: this.blockHeight,
      clusterCount: this.clusterCount
    };

    this.http.post<any[]>('http://localhost:5049/Transaction/cluster',
requestPayload).subscribe(
      data => {
        // Process the retrieved clustered data
        console.log('Clustered data:', data);
        this.updateChart(data);
      },
      error => {
        console.error('Error fetching clustered data:', error);
      }
    );
  }
```

Рисунок 4.10 – Метод fetchData

У функції createChart (рисунок 4.11) відбувається створення нового графіку за допомогою бібліотеки Chart.js. Вона реєструє необхідні модулі та плагіни, отримує доступ до контексту рендерингу 2D за допомогою getContext('2d') та ініціалізує новий об'єкт Chart з використанням цього контексту. У цьому об'єкті задаються тип графіку, дані, опції щодо масштабування, масштаби осей, плагіни для зумування та панорамування тощо.

```
createChart() {
    Chart.register(...registerables);
    Chart.register(zoomPlugin);

    const canvas = this.chartCanvas.nativeElement;
    const ctx = canvas?.getContext('2d');
    if (!ctx) {
      console.error('Failed to get 2D rendering context');
      return;
    }
    this.chart = new Chart(ctx, {
      type: 'scatter',
      data: {
        datasets: [{
          label: 'Clustered Data',
          data: [],
          backgroundColor: 'rgba(75, 192, 192, 0.6)',
          borderColor: 'rgba(75, 192, 192, 1)',
          pointRadius: 5, // Adjust the size of the dots
        }]
      },
      options: {
        responsive: true,
        scales: {
          x: {
            type: 'linear',
            position: 'bottom',
            min: 0, // Set the initial minimum value of the x-axis scale
            max: 100, // Set the initial maximum value of the x-axis scale
          },
          y: {
            type: 'linear',
            position: 'left',
            min: 0, // Set the initial minimum value of the y-axis scale
            max: 200, // Set the initial maximum value of the y-axis scale
          }
        },
        plugins: {
          zoom: {
            zoom: {
              wheel: {
                enabled: true, // Enable zooming using the mouse wheel
              },
              pinch: {
                enabled: true, // Enable zooming using pinch gestures (for
touch devices)
              },
              mode: 'xy', // Enable zooming on both the x-axis and y-axis
            },
            pan: {
              enabled: true, // Enable panning
              mode: 'xy', // Enable panning on both the x-axis and y-axis
            },
          },
        },
```

Рисунок 4.11 – Метод createChart

Функція updateChart (рисунок 4.12) використовує отримані дані для оновлення графіку. Вона групує дані по кластерах, генерує кольори для кожного кластера, створює набори даних та оновлює датасети графіку.

```
updateChart(data: any[]) {
    const clusters = data.map(item => item.PredictedClusterId);

    // Get unique cluster IDs
    const uniqueClusters = [...new Set(clusters)].sort((a, b) => a - b);

    // Generate colors for each cluster
    const clusterColors = this.generateClusterColors(uniqueClusters.length);

    // Prepare datasets
    const datasets = uniqueClusters.map((clusterId, index) => {
      const clusterData = data.filter(item => item.PredictedClusterId ===
clusterId);
      const features = clusterData.map(item => item.Features);
      return {
        label: `Cluster ${clusterId}`,
        data: features,
        backgroundColor: clusterColors[index],
        borderColor: clusterColors[index],
        pointRadius: 2, // Adjust the size of the dots
      };
    });

    // Update chart datasets
    this.chart.data.datasets = datasets;
    this.chart.update();
  }
```

Рисунок 4.12 – Метод updateChart

Функція generateClusterColors (рисунок 4.13) генерує випадкові кольори для кластерів на графіку.

```
  generateClusterColors(numClusters: number): string[] {
    // Generate random colors for clusters
    const colors = [];
    for (let i = 0; i < numClusters; i++) {
      const color = `rgba(${this.getRandomColorValue()},
${this.getRandomColorValue()}, ${this.getRandomColorValue()}, 0.6)`;
      colors.push(color);
    }
    return colors;
  }
```

Рисунок 4.13 – Метод generateClusterColors

У шаблоні компонента DiagramComponent використовуються HTML-елемент <canvas> для відображення графіку, два <input> поля для введення значень blockHeight та clusterCount, та кнопка для виклику функції fetchData (рисунок 4.14).

```
<canvas #chartCanvas></canvas>
<label for="blockHeight">Block Height:</label>
<input type="number" [(ngModel)]="blockHeight" id="blockHeight"
placeholder="Enter the block height">

<label for="clusterCount">Cluster Count:</label>
<input type="number" [(ngModel)]="clusterCount" id="clusterCount"
placeholder="Enter the number of clusters">

<button (click)="fetchData()">Fetch Clustered Data</button>
```

Рисунок 4.14 – HTML сторінка додатку

В файлі app.module.ts (рисунок 4.15) оголошений модуль AppModule, який включає компонент DiagramComponent та необхідні модулі для роботи з Angular та HTTP-запитами (рисунок 4.15).

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { DiagramComponent } from './components/diagram/diagram.component';
import {AppRoutingModule} from "./app-routing-module";
import {HttpClientModule} from "@angular/common/http";
import {FormsModule} from "@angular/forms";

@NgModule({
  declarations: [
    AppComponent,
    DiagramComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Рисунок 4.15 – модуль AppModule

## 4.5 Висновки до розділу 4

У цьому розділі було розглянуто різні аспекти розробки системи, що включає взаємодію з мережею Bitcoin та використання нейромереж для аналізу даних.

Взаємодія з мережею Bitcoin була реалізована за допомогою бібліотек NBitcoin та NBitcoin.RPC. Ці бібліотеки надають зручний доступ до функціоналу Bitcoin, такого як отримання блоків, хешів блоків, отримання даних про транзакції тощо. Використання клієнта RPC дозволяє взаємодіяти з вузлом Bitcoin за допомогою HTTP-запитів.

Для реалізації кластеризації транзакцій було використано ML.NET - бібліотеку машинного навчання для платформи .NET. ML.NET надає зручні інструменти для побудови, тренування та застосування моделей машинного навчання. Була створена модель для кластеризації транзакцій на основі алгоритму K-means.

На фронтенді було використано Angular - популярний фреймворк для розробки веб-додатків з використанням TypeScript. За допомогою Angular було створено веб-інтерфейс, який надає зручну взаємодію з системою. Було створено компоненти для завантаження фотографій та відображення діаграми кластеризації.

У результаті розроблено повноцінну систему, яка поєднує в собі взаємодію з мережею Bitcoin, аналіз транзакцій за допомогою нейромереж та веб-інтерфейс для зручного користування. Ця система може бути використана для подальшого дослідження та аналізу даних Bitcoin з використанням машинного навчання.

# ВИСНОВКИ

У рамках даної бакалаврської кваліфікаційної роботи було проведено дослідження та розробку системи, що включає в себе взаємодію з мережею Bitcoin та застосування нейромереж для аналізу даних.

Під час розробки системи взаємодії з мережею Bitcoin було використано бібліотеки NBitcoin та NBitcoin.RPC, які забезпечують зручний доступ до функціоналу Bitcoin. Це дозволило отримувати дані про блоки та транзакції, а також виконувати різні операції з мережею Bitcoin.

Для аналізу даних було використано нейромережі з використанням бібліотеки ML.NET для платформи .NET. Була створена модель нейромережі для кластеризації транзакцій, що дозволяє групувати транзакції за схожими характеристиками. Це надає можливість отримати інформацію про зв'язки між різними групами транзакцій та спростити подальший аналіз даних.

На фронтенді було використано Angular, який забезпечив зручний інтерфейс для користувачів. Було розроблено компоненти для завантаження фотографій та відображення діаграми кластеризації. Це надає користувачам зручний спосіб взаємодії з системою та отримання результатів аналізу даних.

У результаті роботи було створено повноцінну систему, яка поєднує в собі взаємодію з мережею Bitcoin та аналіз даних з використанням нейромереж. Система може бути використана для подальшого дослідження та аналізу даних Bitcoin, що сприяє розвитку області криптовалют та машинного навчання.

У процесі розробки системи було використано сучасні технології та інструменти, які забезпечили зручну та ефективну реалізацію поставлених завдань. Результати дослідження та розробки вказують на потенціал застосування нейромереж та аналізу даних для вивчення та розуміння мережі Bitcoin.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Pagani, "QUIC Bitcoin: Fast and Secure Peer-to-Peer Payments and Payment Channels," 2022 IEEE Future Networks World Forum (FNWF), Montreal, QC, Canada, 2022, pp. 578-584, doi: 10.1109/FNWF55208.2022.00107.

2. H. Henderi, Q. Aini, I. Sembiring, P. A. Sunarya, V. Tashya Devana and F. P. Oganda, "A Bitcoin Blockchain-Based Educational Digital Assets Management System," 2022 IEEE Creative Communication and Innovative Technology (ICCIT), Tangerang, Indonesia, 2022, pp. 1-6, doi: 10.1109/ICCIT55355.2022.10118739.

3. S. Suhaimi, L. Y. Hui, N. B. Shafee, H. Hashim, S. N. Huda and M. Mohd Azlishah Othman, "Financial Technologies for Accepting Transaction Using Cryptocurrency," 2022 International Conference on Digital Transformation and Intelligence (ICDI), Kuching, Sarawak, Malaysia, 2022, pp. 209-213, doi: 10.1109/ICDI57181.2022.10007372.

4. C. Suthaputchakun and Y. Cao, "Blockchain-based Secure Ambulance-to-Everything Communications in Emergency Rescue Operations," 2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN), Barcelona, Spain, 2022, pp. 174-179, doi: 10.1109/ICUFN55119.2022.9829619.

5. S. Bartolucci and S. Fiorentino, "Blockchain and Smart Contracts as New Governance Tools for the Sharing Economy," 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C), Stuttgart, Germany, 2021, pp. 118-119, doi: 10.1109/ICSA-C52384.2021.00030.

6. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Maniatis, P. (2018). Hyperledger Fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference (pp. 30:1-30:15). Association for Computing Machinery.

7. Zheng, Z., Xie, S., Dai, H. N., Chen, X., & Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. In IEEE International Congress on Big Data (pp. 557-564). IEEE.

8. J. M. Montes, C. E. Ramirez, M. C. Gutierrez and V. M. Larios, "Smart Contracts for supply chain applicable to Smart Cities daily operations," 2019 IEEE International Smart Cities Conference (ISC2), Casablanca, Morocco, 2019, pp. 565-570, doi: 10.1109/ISC246665.2019.9071650.

9. Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. IEEE Transactions on Information Theory, 22(6), 644-654. doi: 10.1109/TIT.1976.1055638

10. Biryukov, A., Feher, D., & Pustogarov, I. (2020). A Survey of Blockchain Anonymity Techniques. Proceedings on Privacy Enhancing Technologies, 2020(1), 1-27. doi: 10.2478/popets-2020-0001

11. Singh, N., & Kumar, N. (2021). A Survey of Privacy in Blockchain Technologies. Journal of Information Security and Applications, 59, 102765. doi: 10.1016/j.jisa.2021.102765

12. Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2019). An Overview of Privacy-Preserving Blockchain Protocols. IEEE Access, 7, 114456-114476. doi: 10.1109/ACCESS.2019.2932707

13. Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., & Virza, M. (2014). Zerocash: Decentralized Anonymous Payments from Bitcoin. In Proceedings of the 2014 IEEE Symposium on Security and Privacy (pp. 459-474). doi: 10.1109/SP.2014.36

14. J. Duan, L. Gu and S. Zheng, "ARCT: An Efficient Aggregating Ring Confidential Transaction Protocol in Blockchain," in IEEE Access, vol. 8, pp. 198118-198130, 2020, doi: 10.1109/ACCESS.2020.3034333.

15. Androulaki, E., Karame, G. O., Roeschlin, M., Scherer, T., & Capkun, S. (2013). Evaluating User Privacy in Bitcoin. In International Conference on Financial Cryptography and Data Security (pp. 34-51). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-39884-1_3

16. Biryukov, A., Khovratovich, D., & Pustogarov, I. (2014). Deanonymisation of Clients in Bitcoin P2P Network. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (pp. 15-29). doi: 10.1145/2660267.2660379

17. Ron, D., & Shamir, A. (2013). Quantitative Analysis of the Full Bitcoin Transaction Graph. In International Conference on Financial Cryptography and Data Security (pp. 6-24). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-39884-1_2

18. Möser, M., Böhme, R., & Breuker, D. (2013). An Inquiry into Money Laundering Tools in the Bitcoin Ecosystem. In Proceedings of the 2013 APWG eCrime Researchers Summit (pp. 1-18). Retrieved from https://pdfs.semanticscholar.org/972d/4b2b9c3b7d7df205d9e47875432049b882c1.pdf

19. Koshy, P., Koshy, D., & McDaniel, P. (2014). An Analysis of Anonymity in the Bitcoin System. In Security and Privacy in Social Networks (pp. 197-223). Cham: Springer. doi: 10.1007/978-3-319-11379-1_9

20. X. Li, C. Xu and Q. Zhao, "Shellproof: More Efficient Zero-Knowledge Proofs for Confidential Transactions in Blockchain," 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Toronto, ON, Canada, 2020, pp. 1-5, doi: 10.1109/ICBC48266.2020.9169437.

21. J. Han, M. Kamber, and J. Pei, "Data Mining: Concepts and Techniques," 3rd ed. Burlington, MA, USA: Morgan Kaufmann Publishers, 2011.

22. H. Minkowski, "Die Grundgleichungen für die elektromagnetischen Vorgänge in bewegten Körpern," Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, vol. 53, pp. 53-111, 1908.

23. P. Chebyshev, "Über die Annäherung der gegebenen Funktionen durch Polynome bester Approximation," Mathematische Annalen, vol. 58, no. 3, pp. 337-362, 1904.

24. D. Zhang, L. Liu, and G. Lu, "A Comparative Study of Similarity Measures for Content-Based Image Retrieval," in Proceedings of the 2003 Pacific Rim Conference on Multimedia, Singapore, 2003, pp. 288-296.

25. J. MacQueen, "Some methods for classification and analysis of multivariate observations," in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 1967, pp. 281-297.

26. J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm," Journal of the Royal Statistical Society: Series C (Applied Statistics), vol. 28, no. 1, pp. 100-108, 1979.

27. R. Hayes, "Deanonymization of Clients in Bitcoin P2P Network," in Proceedings of the 2013 APWG eCrime Researchers Summit, San Francisco, CA, USA, 2013, pp. 1-10.

# ДОДАТОК А

## Код програмної реалізації

```
using System.Net;
using BlockchainDeanonymization.Api.Settings;
using BlockchainDeanonymization.Application.Interfaces.Services;
using BlockchainDeanonymization.Clustering.Data;
using BlockchainDeanonymization.Domain.Models;
using BlockchainDeanonymization.Persistence;
using Microsoft.ML;
using NBitcoin;
using NBitcoin.RPC;

namespace BlockchainDeanonymization.Clustering;

public class BitcoinService : ICryptocurrencyService
{
    private readonly RpcSettings _rpcSettings;
    private readonly BlockchainDeanonymizationDatabaseContext _dbContext;

    public BitcoinService(RpcSettings rpcSettings,
BlockchainDeanonymizationDatabaseContext dbContext)
    {
        _rpcSettings = rpcSettings;
        _dbContext = dbContext;

        _rpcSettings.Password = "245245";
        _rpcSettings.Url = "http://localhost:8332/";
        _rpcSettings.User = "yehor";
    }

    public IEnumerable<TransactionData> LoadTransactionData(int transactionCount)
    {
        var transactions = new List<TransactionData>();

        var rpcClient = new RPCClient(new NetworkCredential(_rpcSettings.User,
_rpcSettings.Password), new Uri(_rpcSettings.Url));

        var sizeOfBlockchain = rpcClient.GetBlockCount();

        var existingTransactions = _dbContext.TransactionData.ToList();

        if (existingTransactions.Count >= transactionCount)
        {
            return existingTransactions.TakeLast(transactionCount);
        }

        var transactionsToAdd = FindNewTransactions(rpcClient, sizeOfBlockchain,
existingTransactions, transactionCount - existingTransactions.Count);

        _dbContext.TransactionData.AddRange(transactionsToAdd);
        _dbContext.SaveChanges();

        transactions.AddRange(transactionsToAdd);

        return transactions;
    }
```

```
    private List<TransactionData> FindNewTransactions(RPCClient rpcClient, int
sizeOfBlockchain, List<TransactionData> existingTransactions, int count)
    {
        var transactionsToAdd = new List<TransactionData>();
        var id = 0;

        while (transactionsToAdd.Count < count && sizeOfBlockchain > 0)
        {
            var blockHash = rpcClient.GetBlockHash(sizeOfBlockchain);
            Block? block = null;

            try
            {
                block = rpcClient.GetBlock(blockHash);
            }
            catch (Exception e)
            {
                // Handle exception
            }

            var newTransactions = block?.Transactions.Select(t => new
TransactionData
                                    {
                                        Features = new float[]
                                        {
                                        },
                                        TotalOut =
(float)t.TotalOut.ToDecimal(MoneyUnit.BTC),
                                        Size = t.GetVirtualSize(),
                                        InputWallet =
ClusteringHelper.GetInputWalletInfo(t),
                                        OutputWallet =
ClusteringHelper.GetOutputWalletInfo(t),
                                        InputAndOutputWallet =
ClusteringHelper.GetInputAndOuputWalletInfo(t),
                                    }).ToList()
                                    ?? new List<TransactionData>();

            foreach (var transaction in newTransactions)
            {
                var existingTransaction = existingTransactions.FirstOrDefault(t =>
t.Id == transaction.Id);
                if (existingTransaction != null)
                {
                    continue;
                }

                transactionsToAdd.Add(transaction);
                if (transactionsToAdd.Count >= count)
                {
                    break;
                }
            }

            sizeOfBlockchain--;
        }

        return transactionsToAdd;
    }

    public MLContext InitializeMLContext()
```

```
    {
        return new MLContext();
    }

    public ITransformer TrainModel(MLContext mlContext, IDataView dataView, int
numberOfClusters)
    {
        var pipeline = mlContext.Transforms
            .Concatenate("Features", "InputAndOutputWallet", "TotalOut", "Size")
            .Append(mlContext.Transforms.NormalizeMinMax("Features"))
            .Append(mlContext.Transforms.Concatenate("Features",
"InputAndOutputWallet", "TotalOut", "Size"))
            .AppendCacheCheckpoint(mlContext)
            .Append(mlContext.Clustering.Trainers.KMeans("Features",
numberOfClusters: numberOfClusters));

        return pipeline.Fit(dataView);
    }

    public IEnumerable<PredictionCluster> MakePredictions(MLContext mlContext,
ITransformer model, IDataView dataView)
    {
        var predictions = model.Transform(dataView);
        var clusterLabels =
mlContext.Data.CreateEnumerable<PredictionCluster>(predictions, reuseRowObject:
false);

        return clusterLabels;
    }
}
```

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет**
**імені Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

## Спеціальний розділ

# Охорона праці

на тему:

## Деаномізація транзакцій в блокчейн

Спеціальність 122 «Комп'ютерні науки»

**122 – БКР – 401. 21910109**

***Виконав студент 4-го курсу, групи 401***
_____ ***Є.О. Грабовський***
« » червня 2023 р.

***Консультант: канд. техн. наук, доцент***
_____***А. О. Алексєєва***
« » червня 2023 р.

**Миколаїв – 2023**

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

| | | |
|---|---|---|
| ВДТ | – | візуальний дисплейний термінал |
| ЕОМ | – | електронна обчислювальна машина |
| ПК | – | персональний комп’ютер |
| ПЕОМ | – | персональні електронні обчислювальні машини |
| НПАОП | – | нормативно-правові акти про охорону праці |
| НАПБ | – | нормативний акт з пожежної безпеки |
| ДСН | – | державні санітарні норми |
| ДСанПІН | – | державні санітарні норми, правила, гігієнічні нормативи |
| КПО | – | коефіцієнт природного освітлення |

# ВСТУП

Нині інформаційні технології, особливо персональні комп'ютери, стали невід'ємною частиною сучасних підприємств. Вони значно спрощують робочі процеси, але важливо пам'ятати, що їх використання може негативно впливати на здоров'я користувачів. Ця проблема вимагає покращення умов роботи та застосування профілактичних заходів для запобігання негативним наслідкам.

Інтенсивне використання комп'ютера може викликати різноманітні проблеми, проте дотримання правил роботи з ним може суттєво знизити негативний вплив на спину, зір, суглоби та вплив електромагнітного випромінювання.

Під час роботи з комп'ютером користувач повністю залежить від положення екрана. Крім того, постійне оновлення зображення і низька частота оновлення можуть викликати мерехтіння, що негативно позначається на очах і внутрішніх м'язах очей, призводячи до перенапруги зору і розвитку короткозорості. Тривала робота з комп'ютером вимагає підвищеної концентрації, що може викликати головний біль, дратівливість, нервову напругу та стрес. Низька частота оновлення або повільна робота комп'ютера можуть призвести до нервових розладів.

Мета цього розділу полягає в аналізі умов роботи на робочому місці або у виробничому приміщенні розробки системи.

Відповідно до мети даного розділу, визначено такі завдання:

- Проаналізувати умови роботи у робочому приміщенні.
- Визначити достатній рівень показників для робочого приміщення, де проводять роботи з розробки системи.
- Визначити основні правила техніки безпеки під час роботи з екранними пристроями.

# 1 ВИМОГИ ДО БЕЗПЕКИ ПРАЦІ ТА ВІДПОЧИНКУ ПРИ ВИКОНАННІ РОБІТ НА РОБОЧОМУ МІСЦІ

Конституція України гарантує кожній особі право на належні, безпечні та здорові умови праці. У цьому розділі ми розглянемо основні показники належних умов праці та безпеки на робочому місці розробника-програміста в офісних приміщеннях. На робочому місці розробника-програміста необхідно дотримуватись вимог з безпеки праці, які містяться в нормативно-правових актах, зокрема в НПАОП 0.00-7.15-18 «Вимоги з безпеки та охорони здоров'я працівників при роботі з екранними пристроями» [1], затверджених наказом Міністерства соціальної політики України від 14.02.2018 № 207 та зареєстрованих у Міністерстві юстиції України 25.04.2018 за № 508/31960.

Документ НПАОП 0.00-7.15-18 [1] є заміною для НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації обчислювальних машин», затвердженого наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 № 6.

Вимоги цього документа розроблені на основі Директиви 90/270/ЄЕС від 29 травня 1990 року, яка стосується мінімальних вимог безпеки та здоров'я під час роботи з екранними пристроями (п'ята основна директива у рамках статті 16 Директиви 89/391/ЄЕС) [2].

## 1.1 Загальні положення

Згідно з НПАОП 0.00-7.15-18 [1], були визначені такі загальні положення:

- вимоги до безпеки та охорони здоров'я під час роботи з екранними пристроями поширюються на всіх суб'єктів господарювання, незалежно від форми власності, організаційно-правової форми та видів діяльності;
- встановлені вимоги не обмежують право роботодавця встановлювати власні, суворіші вимоги, за умови їх відповідності чинному законодавству.

Вимоги не поширюються на:

- робочі місця, що надаються для навчання в освітніх закладах;
- робочі місця працівників, які керують транспортними засобами (оператори транспортних засобів, водії, пілоти);
- робочі місця працівників, які виконують ремонт та налагодження екранних пристроїв;
- обчислювальні пристрої з невеликими пристроями індикації даних та випромінювання (калькулятори, касові апарати та інші);
- портативні системи обробки даних при їхньому тимчасовому використанні;
- цифрові (дисплейні) друкарські машини;
- смартфони, планшети та мобільні засоби телефонії.

## 1.2 Безпека робочих місць працівників з екранними пристроями

Вимоги щодо безпеки та охорони здоров'я на робочих місцях працівників з екранними пристроями, встановлені в НПАОП 0.00-7.15-18 [1], включають такі положення:

Робочі місця мають бути спроектовані таким чином, щоб працівники мали достатньо простору для зміни робочого стану та вільного переміщення.

Всі види випромінювання, крім видимого спектру екранних пристроїв, повинні бути зведені до рівня, що не перевищує максимально допустимого значення.

Робоче місце має бути організоване з урахуванням ергономічних, психофізіологічних та антропологічних вимог, враховуючи характер виконуваних робіт.

Висвітлення робочого місця має відповідати вимогам Державних санітарних правил та норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин (ДСанПІН 3.3.2.007-98) [3] та

забезпечувати відповідний контраст між екраном та навколишнім середовищем.

Робоча поверхня столу повинна бути достатньою для розміщення екрану, клавіатури, документів і необхідного обладнання, а також мати низьку здатність поверхні, що відображає.

Робоче крісло має бути стійким і дозволяти працівникові вільно переміщатися та займати комфортне становище. Сидіння та спинка сидіння повинні бути регульованими по висоті та нахилу, а при необхідності може використовуватися підставка для ніг.

Мінімальні вимоги безпеки при роботі з екранними пристроями, встановлені в НПАОП 0.00-7.15-18 [1] та Директиві ЄС 90/270/ЄЕС [2], включають такі положення:

- підтримання чистоти та порядку на робочому місці шляхом щоденного прибирання та очищення робочих місць та екранних пристроїв;
- відключення екранних пристроїв від електричної мережі після закінчення роботи;
- негайне відключення всіх електронних приладів від електричної мережі у разі аварійних ситуацій;
- забороняється виконувати технічні роботи з екранними пристроями під час роботи на робочому місці працівника;
- заборонено самостійно проводити технічні роботи та зміни у конструкції екранних пристроїв;
- заборонено працювати з несправними екранними пристроями та пристроями, що сигналізують про несправність;
- при роботі з екранними пристроями, пов'язаною з нервово-емоційною напругою, приміщення повинні відповідати вимогам оптимальних умов мікроклімату відповідно до ДСН 3.3.6.042-99 [4] та НПАОП 0.00-7.15-18 [1].

Додатково, відповідно до Директиви ЄС 90/270/ЄЕС [2] та НПАОП 0.00-7.15-18 [1], встановлені такі мінімальні вимоги безпеки до екранних пристроїв:

- екранні пристрої не повинні загрожувати працівникам;
- символи на екранах мають бути чіткими та мати відповідний розмір;
- між символами та рядками символів має бути належна відстань;
- зображення на екрані має бути стабільним, без мерехтіння;
- яскравість і контрастність символів повинні легко налаштовуватися працівником та швидко адаптуватися до навколишніх умов;
- екран не повинен відображатись, щоб уникнути дискомфорту для працівника;
- всі види випромінювання, крім видимої частини електромагнітного спектра, мають бути зведені до безпечного рівня охорони здоров'я працівників;
- екран повинен бути легко поворотним та похилим залежно від потреб працівника;
- при необхідності може використовуватись регульований стіл або підставка для екрана;
- клавіатура повинна бути відокремлена від екранного пристрою та мати функції регулювання висоти, щоб забезпечити зручну робочу позицію та уникнути втоми рук;
- поверхня клавіатури не повинна відображатись, а клавіші та їх розташування повинні полегшувати роботу та мати чіткі позначення;
- пристрої, що входять до робочої станції, не повинні виділяти надлишкове тепло, щоб не створювати дискомфорт для працівника;
- роботодавець повинен забезпечити програмне забезпечення, яке відповідає завданням та кваліфікації працівника, а також адаптоване до його особливостей.

## 1.3 Вимоги щодо режиму праці та відпочинку

При розробці програмного забезпечення для реалізації програмних застосунків для деаномізації блокчейн-транзакцій необхідно дотримуватись вимог до режимів праці і відпочинку, які регулюються Державними санітарними правилами і нормами роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98 (п.5) [3]. Організація праці, пов'язана з використанням ВДТ ЕОМ і ПЕОМ, повинна передбачати регламентовані перерви для відпочинку з метою збереження здоров'я працюючих, запобігання професійним захворюванням та підтримки працездатності.

Під час виконання різних видів трудової діяльності протягом дня, включаючи роботу з ВДТ ЕОМ і ПЕОМ, рекомендується передбачати додаткові короткі перерви до появи ознак втоми та зниження працездатності.

Основною роботою з ВДТ ЕОМ і ПЕОМ слід вважати ту, що займає не менше 50% часу протягом робочої зміни:

– обідні перерви – для відпочинку та вживанню їжі;

– перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

Тривалість обідньої перерви визначається відповідно до чинного законодавства про працю та Правил внутрішнього трудового розпорядку на підприємстві (організації, установі).

Встановлення внутрішньозмінних режимів праці і відпочинку для роботи з ВДТ ЕОМ і ПЕОМ здійснюється з урахуванням характеру трудової діяльності, рівня напруженості та складності праці, з диференціацією для кожної професії.

За характером трудової діяльності виділено три професійні групи згідно з діючим класифікатором професій (ДК 003:2010) [5]:

1. Розробники програм (інженери-програмісти) займаються в основному роботою з відеотерміналом та документацією, виконуючи інтенсивний обмін інформацією з ЕОМ і часто приймають рішення. Їх робота характеризується високим рівнем розумової творчої праці, що супроводжується напруженою роботою очей, концентрацією уваги, нервово-емоційним напруженням, необхідністю утримання фіксованої робочої пози, обмеженим рухом і періодичним навантаженням на кисті верхніх кінцівок. Робота відбувається в режимі активної взаємодії з ЕОМ, часто в обмежені строки, що вимагає періодичного пошуку помилок та швидкого виконання завдань.

2. Оператори електронно-обчислювальних машин виконують роботу, пов'язану з обліком інформації, що надходить з ВДТ за запитом або передається через нього. Їх робота включає перерви різної тривалості, коли вони займаються іншими завданнями. Ця робота характеризується напруженням зору, невеликими фізичними навантаженнями та середнім рівнем нервового напруження. Вона виконується у вільному темпі.

3. Оператор комп'ютерного набору виконує рутинну роботу з обробки документів та введенням даних за допомогою клавіатури. Робота характеризується фізичним напруженням на кисті верхніх кінцівок через швидке введення даних та навантаження на зір на екрані дисплея. Вона також включає періодичні зміни фокусу зору між документами і екраном. Оператор відчуває загальну недостатню фізичну активність і нервово-емоційне напруження.

При роботі з ЕОМ протягом 8-годинної робочої зміни встановлюються внутрішньозмінні режими праці та відпочинку, залежно від характеру праці. Для розробників програм, які використовують ЕОМ, рекомендується надавати 15-хвилинну регламентовану перерву для відпочинку кожну годину роботи з відеодисплейним терміналом (ВДТ).

– для операторів із застосування ЕОМ слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожною години роботи за ВДТ.

У випадках, коли регламентовані перерви не можуть бути застосовані з причин виробничого характеру, рекомендується обмежувати безперервний робочий час з відеодисплейним терміналом (ВДТ) до 4 годин. При 12-годинній робочій зміні регламентовані перерви повинні бути встановлені протягом перших 8 годин роботи, подібно до перерв при 8-годинній робочій зміні, і протягом останніх 4 годин роботи, незалежно від характеру трудової діяльності, рекомендується вводити 15-хвилинні перерви кожну годину.

Для зниження негативного впливу монотонності рекомендується використовувати чергування між операціями, пов'язаними з обробкою тексту і числових даних, що дозволяє змінювати характер виконуваної роботи. Також рекомендується чергувати введення даних та редагування текстів.

Для поліпшення фізичного та психічного стану, запобігання втомі та покращення кровообігу можна використовувати перерви, під час яких рекомендується виконувати комплекс фізичних вправ. Це допомагає зменшити нервово-емоційне напруження, втомлення зору, а також протидіяти негативним ефектам недостатньої фізичної активності.

У деяких випадках, коли працюючі з ВДТ ЕОМ і ПЕОМ виявляють хронічні проблеми зі зором, незважаючи на дотримання санітарно-гігієнічних норм щодо режимів праці і відпочинку та використання засобів захисту очей, можуть застосовуватися індивідуальні підходи. Це може включати обмеження часу, витраченого на роботу з ВДТ, зміну характеру роботи або чергування з іншими видами діяльності, які не пов'язані з ВДТ.

Активний період відпочинку включає в себе виконання комплексу фізичних вправ, спрямованих на зняття нервового напруження, розслаблення м'язів, відновлення функцій фізіологічних систем, які під час робочого

процесу зазнають навантаження. Цей період також сприяє зняттю втоми очей, поліпшенню кровообігу в мозку та підвищенню працездатності.

# 2 ШКІДЛИВІ ВИРОБНИЧІ ФАКТОРИ НА РОБОЧОМУ МІСЦІ

## 2.1 Характеристика шкідливих факторів на робочому місці

Інтенсивне використання персональних комп'ютерів з візуальними дисплейними терміналами є основною причиною багатьох хвороб. Робота користувача за комп'ютером відбувається в одноманітній позі, що обмежує загальну м'язову активність, при цьому навантажується зір, виникає значне напруження органів зору та психологічне напруження через наступні фізичні фактори:

– збільшене навантаження на очі;

– випромінювання електромагнітних хвиль у наднизькочастотному, низькочастотному та середньочастотному діапазонах (5 Гц – 400 кГц);

– довготривале сидяче положення;

– перевантаження суглобів рук;

– комп'ютер є серйозним джерелом алергенів через підвищений рівень електростатики;

– тривале нервове та емоційне напруження (вплив комп'ютера на психічне здоров'я).

Під час роботи за комп'ютером очі постійно фокусуються в одному напрямку, а також зазнають постійного переключення від яскравих об'єктів з позитивним контрастом до темних з негативним контрастом. Протягом восьмигодинного робочого дня користувач звертає погляд на екран приблизно 30000 разів, що створює перевантаження для очей і ускладнює їх адаптацію. Ці особливості призводять до напруження м'язів та сприйняття світла очима, що спричиняє такі симптоми, як блискіт у очах, біль у очах, напруга у підборівнах, розмитість контурів та нечіткість зображення [6].

Щоб зменшити втому очей, важливо дотримуватися наступних рекомендацій: забезпечити правильне освітлення, використовуючи природне світло, а при необхідності - штучне, уникати освітлення, яке освітлює екран

монітора, щоб уникнути відблисків, дотримуватися режиму праці та виконувати зорову гімнастику.

Серйозною проблемою є електромагнітне випромінювання, яке виникає внаслідок роботи комп'ютера. Монітор з електронно-променевою трубкою створює змінне електромагнітне випромінювання, яке розповсюджується у різних напрямках, а використання високої напруги призводить до формування електростатичного поля навколо монітора. Сильне електромагнітне випромінювання є характерною рисою застарілої технології, тому рекомендується перейти на сучасне обладнання, наприклад, рідкокристалічний монітор, та обмежувати час роботи за екраном, роблячи короткі перерви під час роботи.

Неменше серйозною проблемою, що впливає на працездатність та здоров'я, є малорухливий спосіб життя та тривале сидіння. Це може призвести до розвитку запущеного стану остеохондрозу та стиснення нервової системи через спотворення хребта. Небезпечним є те, що концентрація на роботі за монітором пригнічує вчасне сприйняття сигналів болю, які є тривожними симптомами для організму. Щоб зменшити шкідливий вплив тривалого сидіння, важливо обрати відповідні меблі для робочого місця. Крісло повинно мати ролики, регулюватись за висотою сидіння та нахилом спинки, мати підлокітники і можливість обертатись навколо своєї осі. Стіл, в свою чергу, має мати висувну дошку для клавіатури та простір для розміщення інших пристроїв. В перервах варто проводити гімнастичні розминки, щоб зняти статичне напруження м'язів спини та рук.

Робота з комп'ютером за допомогою миші та клавіатури може призводити до неправильного положення рук при вводі даних. Це може спричинити стиснення нервів в зап'ястному каналі і біль в зап'ястях (карпальний тунельний синдром). Захворювання рук та кистей спостерігаються у користувачів комп'ютерів у 7-12 разів частіше, ніж у інших людей. Щоб уникнути синдрому карпального тунелю, важливо дотримуватися

простих рекомендацій щодо організації робочого місця та режиму праці, а також робити короткі перерви кожну годину для виконання вправ для рук.

Монітор комп'ютера притягує пил з повітря через електростатичні заряди, які утворюються на його поверхні. Цей пил може містити багато мікроорганізмів та алергенних часток. Робота за комп'ютером у неприбраному приміщенні може викликати алергічні реакції. Щоб запобігти цьому, рекомендується дотримуватися наступних заходів: забезпечувати чистоту на робочому місці та підтримувати оптимальний мікроклімат в приміщенні шляхом вологого прибирання та провітрювання.

Робота з комп'ютером суттєво навантажує нервову систему, що часто призводить до нервово-психічних порушень та стресу. У працівників, що працюють за комп'ютерами, частіше спостерігаються такі розлади, як тривога, пригніченість та дратівливість, а також можуть виникати проблеми зі сном, втрата апетиту та інші захворювання.

Для зменшення навантаження на нервову систему рекомендується проводити час на прогулянках на свіжому повітрі, спілкуватися з людьми особисто і дотримуватися режиму праці і відпочинку, зокрема, регулярно робити перерви під час роботи з комп'ютером.

## 2.2 Вимоги до освітлення, шуму та вібрації на робочих місцях програмістів-розробників

Відомо, що тривала робота з комп'ютером та документами при недостатній освітленості може спричинити значне напруження очей. Тому вимоги до освітлення мають велике значення. Вимоги до освітлення зазначені у ДБН В.2.5-28-2018 "Природне і штучне освітлення", які були затверджені Міністерством регіонального розвитку, будівництва та житлово-комунального господарства України наказом від 03.10.2018 р. № 264 [7].

Відповідно до цих вимог, природне освітлення в приміщенні з комп'ютером повинно здійснюватися через вікна, переважно орієнтовані на

північ або північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не менше 1,5%. Штучне освітлення повинно бути системою загального рівномірного освітлення, а також можуть використовуватися світильники місцевого освітлення, якщо необхідно. Освітленість на робочих столах і в зоні розміщення документів повинна бути в межах 300-500 лк. Важливими параметрами є $E_{мін}$ (мінімальний рівень освітленості) і $K_п$ (коефіцієнт пульсації світлового потоку), який не повинен перевищувати 20%. При встановленні світильників місцевого освітлення потрібно уникати відблисків на поверхні екрану, а освітленість екрану не повинна перевищувати 300 лк.

Вимірювання з використанням люксметра MASTECH MS6610 показали, що рівень природної освітленості на поверхні, де розташований комп'ютер програміста, складає 385 лк. Порівняно з освітленістю тієї ж поверхні відкритим небосхилом, яка становить 20000 лк, це відповідає коефіцієнту природної освітленості (КПО) 1,925%, що відповідає нормам. Крім того, були проведені вимірювання при комбінованій системі освітлення, де рівень освітленості склав 428 лк, що відповідає КПО 2,14% і також відповідає нормам.

Вимоги, які забезпечують захист користувачів комп'ютера від шуму та вібрації, визначені в документах ДСанПіН 3.3.2.007-98 "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин" [3], ДСН 3.3.6-037-99 "Санітарні норми виробничого шуму, ультразвуку та інфразвуку" [8] і ДСН 3.3.6-039-99 "Державні санітарні норми виробничої загальної та локальної вібрації" [9].

При роботі з комп'ютером джерелами шуму є різні компоненти, такі як жорсткий диск, вентилятор блока живлення ПК, вентилятор процесора (кулер), швидкісні CD-ROM та DVD-ROM приводи, механічні сканери та рухомі частини принтера.

У досліджуваному приміщенні є лише 2 робочі місця для програмістів-розробників, які оснащені моноблоками з рідкокристалічними моніторами.

Рівень шуму, що випромінюється моноблоками під час простою і під час роботи, не перевищує 27 дБА.

Згідно методикам [11], за наявності декількох джерел шуму з однаковим рівнем інтенсивності $LLi$ загальний рівень шуму визначають за формулою 2.1:

$$LL = LLi + 10\lg(nn); \quad (2.1)$$

Ми маємо всього два джерела, отже загальний рівень шуму буде визначатися так:

$$LL = 27 + 10\lg(2) = 30; \quad (2.2)$$

Розраховане значення середнього рівня шуму 2.2 не перевищує гранично допустимий рівень шуму для робочого місця програміста, яке за нормами ДСН 3.3.6-037-99 [8] складає 50 дБА.

Оцінка рівня вібраційної безпеки проводиться безпосередньо на робочих місцях, де використовуються комп'ютери, в процесі трудової діяльності. Згідно ДСН 3.3.6-039-99 [9], категорія вібрацій згідно з санітарними нормами та критерієм оцінки – 3 та тип "в", що означає комфорт. Таким чином, характеристика умов праці відносно вібрацій на робочих місцях працівників розумової праці і персоналу, яка не займається фізичною працею, відповідає санітарним нормам. Робоче місце повністю відповідає вимогам щодо вібраційної безпеки.

# 3 ПОЖЕЖНА БЕЗПЕКА РОБОЧИХ МІСЦЬ ПРОГРАМІСТІВ-РОЗРОБНИКІВ

Давайте розглянемо дії розробника-програміста, що регламентуються Правилами пожежної безпеки України (НАПБ А.01.001-2014), Правилами пожежної безпеки України від 30.12.2014 № 1417 [10] та Кодексом цивільного захисту України (№ 5403-VI) [11]

У разі надзвичайної ситуації необхідно негайно припинити роботу, відключити комп'ютер або оргтехніку від електромережі та повідомити посібник про подію. Заборонено допускати сторонніх осіб у небезпечну зону.

Якщо є ознаки пожежі або горіння, необхідно повідомити керівництво та негайно зателефонувати до оперативно-рятувальної служби за номерами '101' або '112'. У повідомленні вказати адресу та місце пожежі, наявність людей та своє прізвище.

Слід активувати систему оповіщення про пожежу та спробувати загасити (локалізувати) пожежу власними засобами пожежогасіння. Слід пам'ятати, що електротехнічні пристрої, що знаходяться під напругою, слід гасити тільки після їх відключення від електромережі та за допомогою вуглекислотних або порошкових вогнегасників.

Потім необхідно розпочати евакуацію людей із будівлі у безпечне місце відповідно до плану евакуації та залучити інших людей до цього процесу.

Будівництво та проведення заходів щодо пожежної безпеки визначаються відповідно до Правил пожежної безпеки України (НАПБ А.01.001-2014) [10]. Шляхи евакуації працівників у разі пожежі (переходи, аварійні виходи) визначаються відповідно до ДБН В.1.1-7:2016 57 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [12]. Засоби виявлення загорянь та пожеж використовуються відповідно до ДБН В.2.5-56:2014

«Системи протипожежного захисту» [13], а вибір та експлуатація вогнегасників регулюється «Правилами експлуатації та типовими нормами належності вогнегасників», наказ МВС України від 11.01. 25 [14].

Відповідно до вимог Правил пожежної безпеки України, першочерговими засобами пожежогасіння є вогнегасники. В офісних та громадських будинках на кожному поверсі має бути не менше двох портативних вогнегасників з масою заряду вогнегасної речовини 5 кг або більше. Крім того, в офісних приміщеннях з оргтехнікою на кожні 20 м2 площі має бути розміщений газовий вогнегасник з масою заряду вогнегасної речовини 3 кг або більше, таких як ВВК-1,4 або ВВК-2. Слід пам'ятати про особливості використання газових вогнегасників, тому що при їх використанні в приміщенні може знижуватися вміст кисню нижче за безпечний рівень.

# ВИСНОВКИ

В процесі аналізу умов праці на підприємствах було з'ясовано, що робота з електронними пристроями має певні негативні наслідки для працівників. Використання екранних дисплеїв може спричиняти втому, запаморочення, біль в очах, спині, напруженість та стрес, а також порушення роботи суглобів.

Крім того, на підприємствах існують інші шкідливі умови, такі як шум і вібрація, які також негативно впливають на трудову діяльність людини. У зв'язку з цим, підприємства мають дотримуватись усіх вимог та норм охорони праці, щоб мінімізувати шкідливі наслідки для здоров'я працівників.

Ефективний режим роботи та належний відпочинок є важливими аспектами забезпечення безпеки та здоров'я на робочому місці. Регулярні перерви під час роботи дозволяють відновити фізичну та психологічну енергію працівників і запобігають виникненню надмірної втоми. Рекомендується включати короткі періоди активного відпочинку, такі як розтяжки, прогулянки або вправи, що сприяють розслабленню м'язів та покращенню кровообігу.

Крім того, важливо дотримуватись правильного режиму праці та відпочинку, встановлюючи оптимальні графіки роботи та визначаючи адекватний час на відпочинок. Це допомагає зберегти високу працездатність протягом робочого дня та попереджує ризик виникнення втоми, помилок та нещасних випадків.

Виконання поставлених вимог і нормативів з охорони праці є обов'язковим для підприємств та власників, оскільки це допомагає забезпечити безпеку та здоров'я працюючих людей. Такий підхід дозволяє попередити негативні наслідки, пов'язані з трудовою діяльністю.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями : Наказ М-ва соц. політики України від 14.02.2018 р. № 207. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 20.05.2023).
2. Про мінімальні вимоги безпеки та здоров’я при роботі з екранними пристроями : Директива від 29.05.1990 р. № 90/270/ЄЕС.
3. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин : Правила від 10.12.1998 р. № 7. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення: 20.05.2023).
4. Постанова : Санітарні норми мікроклімату виробн. приміщень ДСН 3.3.6.042-99 від 01.12.1999 р. № 42. URL: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 20.05.2023).
5. Класифікатор професій ДК 003:2010 : Наказ від 28.07.2010 р. № 327. URL: https://zakon.rada.gov.ua/rada/show/va327609-10#Text (дата звернення: 20.05.2023).
6. Демчан І. С. «Шкідливі фактори при роботі з комп’ютерами». URL: https://www.slideshare.net/d_iruna/ss-48672800#:~:text=%D0%9E%D1%81%D0%BD%D0%BE%D0% (дата звернення: 21.05.2023) .
7. Про затвердження ДБН В.2.5-28:2018 "Природне і штучне освітлення" : Наказ від 03.10.2018 р. № 264. URL: https://ips.ligazakon.net/document/FN047999 (дата звернення: 21.05.2023).
8. Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99 : Постанова від 01.12.1999 р. № 37. URL: https://zakon.rada.gov.ua/rada/show/va037282-99#Text (дата звернення: 21.05.2023).

9. Державні санітарні норми виробничої загальної та локальної вібрації ДСН 3.3.6.039-99 : Норми від 01.12.1999 р. № 39. URL: https://zakon.rada.gov.ua/rada/show/va039282-99#Text (дата звернення: 21.05.2023).
10. Про затвердження Правил пожежної безпеки в Україні : Наказ М-ва внутр. справ України від 30.12.2014 р. № 1417 : станом на 7 квіт. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/z0252-15#Text (дата звернення: 21.05.2023).
11. Кодекс цивільного захисту України : Кодекс України від 02.10.2012 р. № 5403-VI : станом на 31 берез. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/5403-17#Text (дата звернення: 21.05.2023).
12. Пожежна безпека об`єктів будівництва. Загальні вимоги : ДБН від 31.10.2016 р. № В.1.1-7:2016.
13. Системи протипожежного захисту. Зі зміною № 1 : ДБН від 13.11.2014 р. № В.2.5-56:2014.
14. Про затвердження Правил експлуатації та типових норм належності вогнегасників : Наказ М-ва внутр. справ України від 15.01.2018 р. № 25 : станом на 15 лип. 2022 р. URL: https://zakon.rada.gov.ua/laws/show/z0225-18#Text (дата звернення: 21.05.2023).