Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

Магістр
(освітній ступінь)

на тему: Сервіс управління механізмом авторських прав на мультимедійні файли (на основі технології Blockchain)

Виконав: студент 6 курсу, групи СНм-61
спеціальності
122 «Комп'ютерні науки»
(шифр і назва спеціальності (напряму підготовки))
(підпис) Копанецький О.Р. (прізвище та ініціали)
Керівник (підпис) доц. Гром'як Р.С. (прізвище та ініціали)
Нормоконтроль (підпис) доц. Мацюк О.В. (прізвище та ініціали)
Рецензент (підпис) проф. Пастух О.А. (прізвище та ініціали)

м. Тернопіль – 2019

АНОТАЦІЯ

Сервіс управління механізмом авторських прав на мультимедійні файли (на основі технології Blockchain) // Дипломна робота освітнього рівня «Магістр» // Копанецький Олександр Романович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра комп'ютерних наук, група СНм–61 // Тернопіль, 2019 // С. – , рис. –*11*, табл.– *5*, слайдів – *17* , додат. – *5* , бібліогр. – *14* .

Ключові слова: BLOCKCHAIN, PYTHON, БАЗА ДАНИХ, МЕТАДАНІ, СЦЕНАРІЙ ВИКОРИСТАННЯ, СМАРТ-КОНТРАКТ

Основними завданнями дипломної роботи є розробка сервісу управління механізмом авторських прав на мультимедійні файли та його програмна реалізація.

У першому розділі дипломної роботи проведено аналіз предметної області дослідження, наведено основні характеристики технології Blockchain, зокрема для використання у медіа сфері. Описано можливості смарт-контракту як способу обміну цифровими цінностями (даними) в Blockchain. Розглянуто особливості та моделі архітектури клієнт-сервер.

У другому розділі описано основні системні вимоги до сервісу, шаблони, розроблені для опису вимог, а також самі вимоги за сценаріями «Музика і метадані» та «Аудіо-фрагменти і метадані». Запропоновано варіанти використання, які є засобом опису функціональних вимог до сервісу.

У третьому розділі побудовано та описано діаграми варіантів використання, компонентів, діяльності та послідовності. Дані діаграми в різних аспектах ілюструють склад і поведінку розроблюваних програмних елементів сервісу. Також розроблено функціональну частину сервісу, наведено окремі лістинги програмного коду.

У четвертому розділі дипломної роботи наведено основні можливості програмного забезпечення, яке використовується для створення сервісу, зокрема мови програмування Python та системи керування базами даних SQLite.

П’ятий розділ дипломної роботи присвячений обґрунтування економічної ефективності проведеного дослідження.

У шостому розділі висвітлені важливі питання охорони праці та безпеки в надзвичайних ситуаціях.

Сьомий розділ описує екологічні питання, які виникли при розробці.

Мета дослідження: розробка програмного продукту, який призначений для управління механізмом авторських прав на мультимедійні файли.

Об’єкт дослідження: процес управління авторськими правами на мультимедійні файли.

Предмет дослідження: сценарії впровадження технології Blockchain для реалізації механізму авторських прав на мультимедійні файли.

ANNOTATION

A service of copyright control mechanism for multimedia files (based on Blockchain technology) // Kopanetskii Oleksandr // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2019 // P. - , Fig. - *11* , Table - *5* , Slide - *17* , References - *14* .

Keywords: BLOCKCHAIN, PYTHON, DATABASE, METADATA, USE CASE, SMART CONTRACT

Thesis deals with the copyright management service for media files and software implementation of it.

In the first section of the thesis the analysis of the subject area of the research was conducted, the main features of Blockchain technology, use in the media. Possibilities of smart contract as a way of sharing digital values (data) in Blockchain are described. Features and models of client-server architecture are considered.

The second section describes the basic system requirements for the service, the templates developed to describe the requirements, as well as the requirements for the "Music and metadata" and "Audio snippets and metadata" use cases. Uses cases are offered that are a means of describing the functional requirements of the service.

In the third section describes builds and describes diagrams of use cases, components, activities, and sequences. These diagrams in various aspects illustrate the composition and behavior of the developed software elements of the service. A functional part of the service was also developed, as well as individual code listings.

The fourth section of the thesis describes basic features of the software used to create the service, including Python programming language and SQLite database management system.

Background includes the following tasks:

– investigated the situation in the field of copyright observance of music records;

– block analyzed the main features of Blockchain technology for recording, validation and distribution of audio fragments;

–different types of client-server interaction architecture were analyzed;

– use cases of implementation of Blockchain technology for registration and authentication of audio material for further distribution are considered;

– software service is designed and implemented.

Actuality of the theme. The major problems of the music industry include transparency, clarity and the operation of the licensing mechanism. Since the start of music distribution on the Internet, the music industry has been looking for ways to monetize digital music. Existing outdated copyright databases and the license fee collection system make it difficult to obtain music from legitimate sources.

By using Blockchain technology and smart contracts to create a common and truly decentralized database of copyrights for music, it is possible to provide an instant and complete transparent transfer of license fees, including distribution of funds to co-authors, producers, technical partners, publishers and publishers. The most important innovation is that smart contracts can provide an automatic allocation of funds for each payment of a music recording according to the specified conditions, and the account of each participant instantly reflects the receipt of funds.

The purpose of the research is development of a software product to control the copyright mechanism for multimedia files.

Object of research – the process of managing the copyright of multimedia files.

Subject of research – use-cases of Blockchain technology implementation for realization a multimedia file copyright mechanism.

Implementation of the developed software service will improve the level of copyright protection of media files and ensure transparency and accessibility of legal information.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

API – (Application Programming Interface) – набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) або операційною системою для використання у зовнішніх програмних продуктах;

IP - (Internet Protocol) – протокол мережевого рівня для передавання датаграм між мережами;

FTP – (File Transfer Protocol) – протокол передачі файлів, дає можливість абоненту обмінюватися двійковими і текстовими файлами з будь-яким комп'ютером мережі, що підтримує протокол FTP;

HTTP – (Hyper Text Transfer Protocol) – протокол передачі даних, що використовується в комп'ютерних мережах;

P2P-network – peer-to-peer network (тимчасова мережа);

БД – база даних;

ПК – персональний комп'ютер.

ЗМІСТ

ВСТУП

Актуальність теми. Індустрія інформаційних технологій в наш час розвивається дуже стрімко, і нові технології з'являються так часто, що фахівці часто не встигають зреагувати на їх появу. Так сталося і з Blockchain – розроблена в рамках криптовалюти Bitcoin, вперше випущеної ще в 2009-му році, яка останнім часом набрала популярності. Технологію Blockchain намагаються застосувати в різних областях. Зокрема, вона може сприяти раціоналізації механізму авторських прав і забезпечити прозорість і чесність оплати праці музикантів і інших правовласників. До головних проблем музичної індустрії відносяться прозорість, ясність і функціонування механізму поширення ліцензій. З моменту початку розповсюдження музики в мережі Інтернет, музична індустрія веде пошуки шляхів монетизації цифрових музичних записів. Основні відомості, необхідні для констатації, хто написав і виконав музику, яку слухає аудиторія, і кому вона зараз належить, часто не беруться до уваги. Наявність цієї інформації і її достовірність обов'язкові для забезпечення отримання грошових коштів власниками і продавцями. Існуючі застарілі бази даних авторських прав і система збору ліцензійного мита в цілому ускладнює отримання музики з легітимних джерел.

Завдяки використанню технології Blockchain і смарт-контрактів для створення загальної і по-справжньому децентралізованої бази даних авторських прав на музичні записи існує можливість забезпечити миттєве і повністю прозоре перерахування ліцензійного мита, включаючи розподіл коштів співавторам, продюсерам, технічним партнерам, видавництвам і лейблам. Найважливіше полягає в тому, що смарт-контракти можуть забезпечити при кожній оплаті музичного запису автоматичний розподіл коштів відповідно до зазначених умов, а на рахунку кожного з учасників миттєво відображається надходження коштів.

Зв'язок із науковими програмами, планами, темами. Дипломна робота виконана відповідно до наукової тематики Тернопільського національного технічного університету імені Івана Пулюя, кафедри комп'ютерних наук.

Мета і задачі дослідження. Мета роботи полягає у розробці програмного продукту, який призначений для управління механізмом авторських прав на мультимедійні файли.

Для досягнення вказаної мети, в роботі поставлено та розв'язано наступні задачі:

– досліджено ситуацію в сфері дотримання авторських прав на музичні записи;

– проаналізовано основні можливості технології Blockchain щодо реєстрації, перевірки і поширення аудіо-фрагментів;

– проведено аналіз різних видів архітектури взаємодії «клієнт-сервер»;

– розглянуті сценарії впровадження технології Blockchain для реєстрації та аутентифікації аудіоматеріалу для подальшого розповсюдження;

– спроектовано та реалізовано програмний сервіс.

Об'єкт дослідження: процес управління авторськими правами на мультимедійні файли.

Предмет дослідження: сценарії впровадження технології Blockchain для реалізації механізму авторських прав на мультимедійні файли.

Методи дослідження. Метод теоретичного дослідження та експериментальний з використання персонального комп'ютера. Методика дослідження базується на теоретичних і прикладних результатах, досягнутих у комп'ютерних науках.

Наукова новизна отриманих результатів. Наукова новизна полягає у вирішенні науково-практичної задачі створення сервісу для управління механізмом авторських прав на мультимедійні файли, при цьому одержано наступні результати:

– розроблено шаблони сценаріїв «Музика і метадані» та «Аудіо-фрагменти і метадані»;

– запропоновано варіанти використання, які є засобом опису функціональних вимог до сервісу;

– сформульовано основні системні вимоги;

– розроблено програмний засіб.

Практичне значення одержаних результатів. Впровадження розробленого сервісу дозволить покращити рівень захисту авторських прав на мультимедійні файли та забезпечити прозорість і доступність правової інформації.

Публікації. Результати дослідження апробовано на VII науково-технічній конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя (11-12 грудня 2019р.) у вигляді опублікованих тез.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика технології Blockchain

Технологічна модель Blockchain була розроблена в рамках криптовалюти Bitcoin. Bitcoin-Blockchain – це відкрита, децентралізована веб-система обліку, яка містить всі коли-небудь вчинені Bitcoin-транзакції. Ідея протоколу Blockchain полягає в тому, щоб забезпечити можливість прозорого проведення транзакцій між двома незнайомими сторонами без залучення центральної інстанції, які підтверджуються. Незважаючи на те, що Blockchain була розроблена для підтримки Bitcoin, дана концепція може бути визначена і використана і окремо від Bitcoin.

У загальному випадку, Blockchain – це база даних, в яку записуються факти і повні копії якої містяться на всіх комп'ютерах, об'єднаних в P2P-мережу. База даних хронологічно лінійно розширюється. Факти, які заносяться в базу, можуть бути різними, наприклад, грошові транзакції або підписи вмісту. Члени мережі – анонімні суб'єкти, що називаються вузлами мережі. Новий вузол, який приєднується до мережі, зобов'язаний завантажити повну копію Blockchain. Всі комунікації всередині мережі використовують інструменти криптографії для точної і безпечної ідентифікації відправника і одержувача. У разі, якщо вузол хоче додати факт в базу, в мережі повинен бути сформований консенсус, який визначає, де буде розміщений факт. Даний консенсус називається блоком.

Розглядаючи це визначення більш детально можна визначити, які нововведення несе за собою технологія Blockchain [4].

Самі по собі P2P-мережі аж ніяк не новинка, вони існують вже довгий час. P2P-мережі – це розподілені системи, яким необхідно вирішити одну дуже складну проблему інформатики – вирішення конфліктів (the resolution of conflicts), або узгодження (reconciliation). На той випадок, коли два несумісні факти надходять одночасно, система повинна мати правила, що визначають, який з цих фактів вважати дійсним. У той час як в централізованих системах (за

рахунок застосування централізованого консенсусу (тобто є тільки одна база даних, яка визначає дійсність транзакції)) і реляційних базах даних (за рахунок посилкової цілісності) ця проблема вирішена, механізми, що застосовуються в таких системах, не можуть бути застосовані в P2P-мережах. Щоб гарантувати цілісність всередині P2P-мережі, необхідно ввести систему консенсусу – спосіб, щоб всі вузли мережі погодилися з порядком фактів. Blockchain пропонує для цього алгоритм «докази роботи» (proof-of-work consensus), що використовує блоки. Цей децентралізований консенсус руйнує стару парадигму централізованого консенсусу.

Децентралізована схема, на якій заснований протокол Bitcoin, переносить авторитет і довіру на децентралізовану мережу і надає її вузлам можливість безперервно і послідовно включати свої транзакції в загальний блок і створювати унікальний «ланцюжок» – Blockchain.

Блоки – це спеціальний спосіб організації фактів в мережі недовірених користувачів (network of non-trusted peers). В їх основі лежить проста ідея: згрупувати факти в блоки, скласти з блоків єдиний ланцюжок, який буде реплікований по всіх вузлах мережі. Цей ланцюжок блоків видно всім вузлам в мережі. Кожен блок посилається на попередній, так може бути простежено походження кожного факту. Криптографія (в особі хеш-кодів) використовується для забезпечення безпеки аутентифікації джерела транзакції і усуває необхідність центрального посередника. Комбінація криптографії та технології Blockchain гарантує, що жодна з транзакцій не буде записана двічі. Факти, ще не додані в блок, називаються очікуваними (pending).

Майнінгом називається процес підтримки діяльності розподіленої платформи і «пошуку» блоків. Стандартний вузол не займається майнінгом, він лише отримує блоки від інших вузлів. Перетворення звичайного вузла у вузол-майнер – добровільне. Процес майнінгу вимагає від комп'ютера-вузла великої обчислювальної потужності. Майнінг полягає у виборі транзакцій (фактів) з числа тих, що очікують і подальшому отриманні хеш-значення сформованого блоку транзакцій. Блок вважається успішно підтвердженим, якщо його хеш-значення починається з певної кількості нулів, проте, велику частину часу

отримані хеш-значення не є успішними, блок трохи змінюється і хеш-значення генерується заново. Успішно підтверджений блок включається в ланцюг і стає видимим для всіх вузлів мережі. Кількість нулів «успішного» хеш-значення вибирається таким чином, щоб кожен блок з'являвся через приблизно однакові проміжки часу. Це дуже складний процес, що важко піддається візуальному поданню. Його можна порівняти з викиданням величезної кількості гральних кісток, поки в будь-якій спробі не випадуть всі шістки. Завдяки складності даного процесу, шанс «знайти» успішне хеш-значення є вкрай низьким, а отже, запобігається можливість шахрайства (ніхто не в змозі контролювати, які саме транзакції (факти) пройдуть майнінг) і підвищується безпека мережі. Винятком є ситуація, коли шахрай володіє більш ніж половиною вузлів мережі [5].

Популярним міфом про Blockchain, який забезпечив їй безліч прихильників, так і противників, є її анонімність. З точки зору Bitcoin (і криптовалюти в цілому), Blockchain – це ефективний інструмент для побудови розподіленої децентралізованої економіки. У той же час, існує багато можливостей для різного роду шахрайської діяльності, наприклад, відмивання грошей.

Однак, анонімність – це єдиний спосіб захисту персональних даних користувачів. У той час, як традиційні банківські моделі надають захист даних за рахунок участі третьої особи, а в Blockchain всі транзакції (факти) видно всім вузлам. Кожен вузол може бачити відправника і одержувача, проте відсутня інформація, яка пов'язувала б транзакцію з реальними людьми.

Щоб дотримуватися цього принципу, наприклад, в Bitcoin рекомендується проводити кожну транзакцію з нової Bitcoin-адреси. Кількість адрес, якими може володіти користувач, не обмежена, так що дуже складно визначити, кому яка адреса належить. Однак, якщо особу відправника встановлюється, можна дуже легко простежити подальші транзакції, проведені з даної адреси.

Потенційно, блокчейн усуває будь-яких посередників при грошовому переказі, користувачеві доступний графічний інтерфейс користувача (GUI) через власний персональний комп'ютер або інший пристрій. Аналогічним функціоналом зараз оперують так звані "гаманці" для різноманітних

криптовалют на ринку.

За останні кілька років було зроблено чимало спроб змінити ситуацію: прихильники підвищення рівня анонімності запропонували нові системи анонімізації (такі як TRR (Transaction Remote Release), засновану на методах шифрування, що використовуються в платформі Tor; технологія доказів з нульовим розголошенням (Zero- Knowledge Proof), коли одна зі сторін доводить наявність у неї потрібної інформації, не розголошуючи її змісту); ті, хто вважає анонімність слабким місцем даної технології, шукали методи розпізнавання відправників транзакцій (для Bitcoin створений Bitcoin Big Bang, покликаний боротися з відмиванням грошей) [7].

Використання такої децентралізованої системи має свої переваги і недоліки. До переваг застосування Blockchain можна віднести:

– підвищує рівень захисту від підробок за рахунок відслідковування ланцюжків походження;

– скорочує або допомагає зменшити витрати IT-інфраструктури;

– допомагає захистити величезну кількість транзакцій;

– скорочує витрати на проведення транзакцій за рахунок їх прямого проведення;

– ускладнює виникнення безконтролю за рахунок прозорості;

– надає більш просту верифікацію контрольних точок даних;

– надає можливість контролю переказів усередині системи.

Крім того, слід зазначити такі недоліки:

– сильна залежність від надійності криптографічних функцій;

– ускладнене регулювання через відсутність центральних адресатів;

– з ростом кількості транзакцій зростає обсяг Blockchain, і вона займає все більше місця на комп'ютерах-вузлах;

– необхідна система постійного стимулювання (вузлів-майнерів), щоб підтримувати Blockchain-інфраструктуру.

Інформація що зберігається в блокчейні існує за принципом загальнодоступної та постійно обновлюваної бази даних. Фактично, це новий

принцип використання мережі, що має свої переваги та недоліки. База даних блокчейну не зберігається на єдиному носії – це означає що всі його записи є дійсно загальнодоступними та легко верифікованими. Не існує жодної централізованої та основної копії яку б могли викрасти та зламати. Інформація поширюється через тисячі комп'ютерів одночасно, уся інформація блокчейну доступна одразу всім учасникам інтернету. Технологія Блокчейн, так само як й інтернет, має свої вбудовані механізми захисту.

В основу технології Blockchain – роботу всіх механізмів закладено використання таких технологій та методів роботи і шифрування даних:

– асиметричні алгоритми шифрування або «асиметричні криптосистеми» (пари "приватних" та "публічних" ключів);

– хеш-функції або "хешування" даних (функції MD та SHA);

– хеш-таблиці для запису результатів хешування – операцій в блоках транзакцій (використання хеш-дерева типу "Дерево Меркла");

– смарт-контракти (англ. Smart Contracts) – метод передачі даних (цифрових цінностей) від однієї особи до іншої;

– токени та реалізація механізму Proof of concept (POC) – доказ концепції, як методу верифікації події (затвердження угоди) в системі.

Шляхом зберігання однакових блоків інформації по всій мережі, блокчейн:

– не може бути контрольованим єдиною організацією;

– не має єдиного "вразливого" центру для атак хакерів;

– постійно оновлює інформацію так, що не існує застарілих або більш нових копій даних.

Незважаючи на істотні плюси використання технології Blockchain, застосування її в кожному окремому проекті повинно бути добре обдуманим і зваженим рішенням. У великій кількості випадків завдання, покладені в рамках проекту на Blockchain, можуть бути набагато більш надійно і ефективно виконані реляційними базами даних, такими як Oracle або MySQL. Якщо таке можливо в проекті, то використання реляційних баз даних є кращим, тому що вони пройшли крізь багато років розробки, мають одні з найбільш добре відтестованих і оптимізованих вихідних кодів в світі. Звичайно, це не робить

Blockchain марною технологією. Однак, перш ніж починати проект, який базується на ній, необхідно мати абсолютно чітку ідею, як і з якої причини в проекті використовується дана технологія.

1.2 Смарт-контракти як спосіб обміну цифровими цінностями (даними) в блокчейні

Усі криптовалюти успішно застосовуються для проведення щоденних валютних транзакцій. Але, ті ж самі мережі, де вони реалізуються, можливо використовувати в цілях розподіленої роботи програмного забезпечення та розповсюдження даних за принципами анонімності та простих договорів. Для даних цілей в блокчейні створюється спеціальний об'єкт – смарт-контракт. Такі програми записуються в мережі та залишаються дійсними назавжди. Крім того, у всіх учасників мережі залишається копія проведеної операції. При цьому роботу контракту можна зіставляти з управлінням грошовими операціями: створенням аукціону, гри з грошовою винагородою, парі, тощо. Також, смарт-контракти відмінно пасують для автоматизації бухгалтерського обліку: контракт може записувати у собі, від кого і скільки прийшло грошей та планувати на цій основі фінансові прогнози на прибутки та втрати підприємства. Усі учасники мережі мають доступ до кількості операцій та їх призначення – блокчейн захищає від несанкціонованих та прихованих спекуляцій

Смарт-контракт – це програмний код, який містить інформацію про транзакцію (або, простіше, угоду) у форматі “якщо … тоді …”. Наприклад, “Якщо користувачем X буде занесено в систему 100 Вс, тоді він отримає 10 токенів N від користувача Y”. Токен – це внутрішня валюта компанії або особи, яка випускається та розповсюджується за методом залучення інвестицій. За дану валюту, в кожному конкретному випадку, інвестор може отримати різноманітні блага, що пропонує ініціатор компанії. Процедура проводиться на манер первинної публічної пропозиції акцій, за виключенням відсутності юридичних прав або державного регулювання поширення ICO (Initial Coin Offering, укр. “Первинне розміщення монет”, маючи на увазі новий підвид цифрової валюти в

котру вкладено певний процент прав та бонусів інвесторами проекту).

Якщо X та Y виконують свої зобов'язання, то кожен з них отримує попередньо визначений ресурс. Якщо хтось з учасників даної процедури спробує уникнути виконання своїх зобов'язань, то угода буде вважатися не завершеною і сторони залишаться при своїх інтересах та майнових правах.

Принцип роботи смарт-контрактів достатньо прозорий: актив потрапляє у програму і вона сама слідкує за виконанням умов угоди. Коли вони будуть виконані, то компанія (продавець) отримає крипто валюту у встановленому еквіваленті до товару, що перейде у власність інвестора (покупця). Основними атрибутами будь-якого смарт-контракту є:

– підписанти – сторони домовленості, що приймають оговорені умови (для цього використовуються адреси облікових записів та цифрова сигнатура, або цифровий мульти-підпис, якщо підписантів контракту більше ніж два);

– предмет домовленості – власне, ресурси для обміну (значення). При цьому, вони мають бути частиною системи, в рамках якої реалізується контракт. Якщо X та Y бажають скласти домовленість в мережі, то X повинен мати оговорену суму на рахунку, а Y – розмістити обмовлену кількість токенів в межах платформи;

– умови домовленості – математично підтверджений опис умов (станів та функцій даного смарт контракту), за яких контракт буде вважатися можливим для виконання та функціональним. [9]

Графічне порівняння смарт-контракту та звичайного паперового контракту наведено на рис. 1.1. Звичайний контракт не підкріплено жодним іншим доказом аніж папером, смарт-контракт – точно перевірена та прозора одиниця в складі мережі блокчейн.

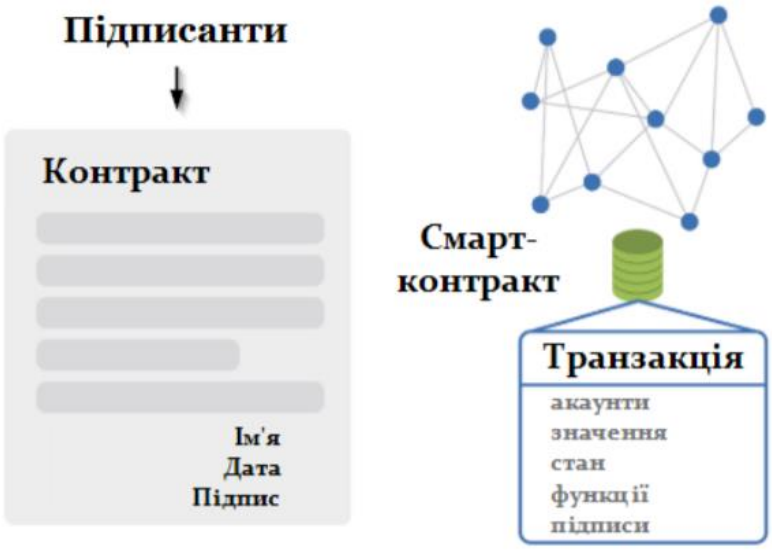


Рисунок 1.1 – Звичайний контракт та смарт-контракт в спрощеному вигляді

Основні переваги смарт-контрактів [10]. Доказ концепції (англ. Proof of concept, PoC) є реалізацією методу або ідеєю демонстрування здійсненності або демонстрації можливості виконання домовленості в принципі з метою перевірки того, що певна угода (інформація та дані, в широкому розумінні) мають практичний потенціал або достовірність та здійсненність. Доказ в такому випадку не обов'язково має бути повним або затвердженим в повному обсязі.
Одним з найголовніших критеріїв успіху смарт-контрактів є проведення угод без залучення третьої сторони (зазвичай, в реальному світі вони виступають гарантами виконання угоди та усіх умов домовленості). Смартконтракти працюють за іншим принципом: лиш тільки реальні складові угоди потраплять в єдиний простір-платформу для обміну та мають відповідні електронні підписи сторін угоди – домовленість вважається закритою та відбувається автоматичний обмін цінностями.

Друга перевага – безпека та конфіденційність угод. Усі контракти зберігаються в блокчейні у зашифрованому вигляді. Про умови та предмет угоди знають тільки сторони домовленості, а зробити несанкціоновані зміни у програмний код виявляється неможливим на практиці.

Третьою перевагою смарт-контрактів є зниження витрат часу та матеріальних благ на проведення угод. Якщо усі умови домовленості виконані, то користувачі здійснюють обмін активами миттєво. Але, не дивлячись на всі переваги, смарт-контракти мають і ряд правил, що мають бути виконаними для їх реалізації. Обов'язковими умовами в даному випадку виступають:

– наявність децентралізованої інформаційної середи (блокчейн), в межах якої буде знаходитись достатня кількість клієнтів для отримання та виконання запитів у всесвітньому цифровому просторі;

– наявність автоматичної бази даних (хеш-таблиці) для проведення транзакцій (укладання контрактів) в мережі;

– наявність спеціальних інструментів та алгоритмів для виконання смарт-контрактів (хеш-функції), що б задовольняли вимогам безпечного розрахунку та однозначного визначення складових угоди, запису даних в БД;

– використання методів асиметричного шифрування (закритого та відкритого ключів) за допомогою яких генеруються цифрові сигнатури клієнтів мережі;

– математично доведена цілісність по Тюрінгу (можливість реалізації в системі будь-якої обчислюваної функції, що не порушує законів логіки даної системи).

Смарт-контракт – це, перш за все, програма. І, як і будь-яка програма, в ньому присутні певні недоліки:

– складність самостійного впровадження смарт-контракту;

– висока залежність від людського фактору (робота смарт-контракту та його безпечність ціликом залежать від правильності написаного програмного коду);

– недостатня адаптивність (дані, що вже занесено в блокчейн, неможливо змінити);

– погане масштабування. При одночасному запуску декількох контрактів пропускна здатність системи знижується пропорційно.

Також, знову слід зазначити, що використання смарт-контракту, як і здійснення будь-якої операції у блокчейні мають бути перевірені учасниками мережі.

Користувач завжди сплачує за розрахунки (перевірку хешу) незалежно від того, чи була успішною ваша операція, чи ні. Навіть якщо операція нездійсненна, майнери мають перевірити та виконати таку транзакцію (розрахувати) і саме за це має сплачувати ініціатор. Механізм дуже схожий до того, що відбувається з будь-якими банківськими операціями. Користувач може переглянути прикріплені 'чайові' (gas limit × gas price) до утвореної транзакції з конвертацією по курсу. Саме ця сума буде винагородженням для майнерів, що роблять роботу з розрахунків та розміщують такі операції в блоках та захищають постійність блокчейну. [10]

1.3 Використання Blockchain у медіасфері

1.3.1 Можливості у боротьбі з піратством

Реальність така, що блокчейн на сьогоднішній день не готовий викорінити піратство. Більш того, будь-які цифрові носії, створені для споживання людиною, можуть бути захоплені і дубльовані. Піратство поведінкове, і немає достатніх доказів того, чому люди роблять це, згідно з доповіддю британського органу дослідження авторських прав. Дослідники виявили, що велика частина знань про несанкціонований обмін файлами відноситься до музики, вказуючи на те, що причини, за якими люди незаконно обмінюються музикою, не обов'язково збігаються з іншими формами контенту. Наприклад, люди можуть ділитися музичними файлами незаконно, так як це робиться в їх співтоваристві. З іншого боку, відсутність доступу в деяких країнах може послужити мотивацією для того, щоб люди могли ділитися телевізійними програмами в Інтернеті. Неможливість може бути причиною інших форм контенту, особливо у випадку ігор та програмного забезпечення. Ідея про те, що певний контент не має фінансової цінності, також може бути злочинною.

Знову ж, жодне з перерахованих вище не є остаточним і повним, що

означає, що немає гарантії, що проект blockchain, побудований у відповідь на будь-яку з вищевказаних причин, за якими люди можуть незаконно ділитися медіаконтентом, зупинить піратство цього конкретного виду контенту. У звіті британського органу дослідження авторських прав вказується, що як галузі, так і законодавці мають потребу в надійних доказах того, чому люди незаконно використовують будь-який медіаконтент, щоб домогтися успіху. Наявні дані свідчать про те, що блокчейн може - в кращому випадку - бути тільки компонентом більш широкої ініціативи по боротьбі з піратством. Іншими словами, блокчейн може в якійсь мірі допомогти в боротьбі з піратством. Малоймовірно, що вона знищить піратство в поодинці.

Деякі експерти вважають, що неможливо або надзвичайно складно викрити будь-медіа-контент незаконно, якщо весь інтернет буде побудований на технології blockchain. Це може статися коли-небудь, але в даний час тут є деякі практичні підходи до того, як blockchain може зіграти свою роль в боротьбі з піратством.

1.3.2 Полегшення контролю за контентом

Vevue (потокова служба блокової послідовності, https://www.vevue.com/blockchain) працює над технологічним кодовим інтелектуальним контрактом спостереження, який зможе відстежувати життєвий цикл будь-якого контенту. Якщо хтось копіює контент, це відстежується технологією будь-якими можливими способами, включаючи відео або запис екрану, наша платформа зможе ідентифікувати власника пристрою / системи, в яких останній раз відтворювався контент, стверджує засновник служби Томас Олсон.

Процес відстеження та ідентифікації скопійованого контенту є складним і виходить за рамки ланцюжка. Vevue розробляє інтелектуальний обчислювальний засіб, який очікує патент, для виконання завдання захисту контенту. Vevue відмовилася надати детальну інформацію про так званому унікальному процесі відстеження та ідентифікації контенту, оскільки це комерційна таємниця, яка ще не має повної правового захисту. Така служба

може бути частиною того, що допомагає блокувати боротьбу з піратством.

Blockchain діє тільки як запис, проти якого відслідковуються інші технології - наприклад, інтелектуальний обчислювальний засіб - перехресно перевіряється, а інтелектуальний контракт грунтується на записах інших технологій в системі, щоб ініціювати дію.

Справді, опис того, як працює технологія Vevue, схоже на те, як працює комбінація судових водяних знаків і блокчейнів.

1.3.3 Платформа для полювання «за головами»

CustosTech (південноафриканська компанія, https://www.custostech.com) використовує блок біткоіна, щоб допомогти власникам інтелектуальної власності боротися з піратством. CustosTech розробила запатентовану технологію судово-медичної експертизи, яка дозволяє впровадити грошову винагороду - біткойн, в даному випадку - пов'язаний з унікальним серійним номером в мультимедійні файли, такі як відео, аудіо, електронні книги і аудіокниги, під час кодування обробити.

Цифровий водяний знак, технологія криміналістичного водяного маркування, існує вже певний час, і існує безліч компаній, що пропонують цифрові послуги водяного знака практично для будь-якого виду медіаконтенту - навіть контенту, що надається через надприбуткові додатки. Судовий водяний знак (унікальний серійний номер) зазвичай розміщується або у випадкових точках, або по всій довжині частини медіаконтенту, і він непомітний для одержувача вмісту. Спосіб, яким він вбудований по довжині файлу, також ускладнює його видалення. Більш того, унікальний порядковий номер служить для ідентифікації отримувача медіафайлу, і на нього не впливають зміна розміру, масштабування, перекодування, запис або будь-які інші зміни, яким можуть бути підданий мультимедійний контент. Тому в разі підозрілого піратського файлу унікальний серійний номер можна витягти, щоб визначити юридичного одержувача файлу, а, отже, і походження піратського контенту.

Як, ймовірно, можна сказати, тільки творці медіа-простору стимулюють пошук піратських медіафайлів. Однак CustosTech використовує глобальний

характер Blockchain і Bitcoin для стимулювання середнього споживача до пошуку медіафайлів, які могли бути незаконно поширені у всьому світі. Будь-хто може приєднатися до програми полювання за біткоїнами CustosTech, подавши заявку на сайт. Якщо особа наймається, «мисливець за головами» отримує інструмент під назвою «Приватний» для використання в скринінгу контенту, захищеного CustosTech для піратства. Як тільки «мисливець за головами» виявляє піратський контент, він витягує біткоїн, прикріплений до цього файлу, в свій власний гаманець біткоїнів. Процес вилучення відправляє унікальний серійний номер, який показує походження піратського контенту, CustosTech і власнику контенту для необхідного дії щодо порушення авторських прав - наприклад, юридична дія, скасування рахунків і «чорний список». Щоб це було доцільно, CustosTech вимагає, щоб вартість дії, здійсненої проти порушника, була більше, ніж вартість вбудованого біткоїна.

Блок-ланцюжок служить тільки системою стимулювання, що дозволяє технології базового водяного знаку давати більш високі результати.

1.3.4 Залучення споживачів з грошовими стимулами

Це може бути складно реалізувати в просторі відео на вимогу. Однак ігрова індустрія може отримати з цього вигоду. В ігровому просторі для мобільних пристроїв вже є практика, відповідно до якої користувачі заохочуються до перегляду реклами з віртуальними кредитами в грі. Rawg (платформа для виявлення ігрових автоматів, https://rawg.io) прагне допомогти ігровим індустріям скоротити піратство, створивши систему blockchain, яка винагороджує гравців за допомогою токенів за досягнення в грі.

Звичайно, Rawg не почала боротися з піратством. Rawg - це, по суті, платформа для виявлення ігрових рухів на різних пристроях, яка збирає інформацію про всі ігри, якими володіє геймер на всіх своїх ігрових пристроях в одному місці, щоб поліпшити досвід геймера. Це допомагає геймерам отримати більш повне уявлення про те, як вони грають в ігри на різних ігрових платформах і тим самим отримують рекомендації. Наприклад, на основі переваг геймера на Xbox, Rawg може рекомендувати аналогічні PlayStation або настільні

ігри. Геймер також може стежити за іншими гравцями схожими інтересами, щоб побачити, що вони грають. Перш ніж вирушити в Blockchain, Rawg вже побудував співтовариство геймерів, які вже синхронізувалися з іграми 50,000.

Rawg робить це ще на один крок краще, надаючи більше можливостей гравцям, дозволяючи їм заробляти токени - які можна обміняти на фіати - на досягнення в грі. Це по своїй суті відштовхує піратство, в деякому роді, тому що геймери можуть синхронізувати досягнення, отримані з непіратском ігор на офіційних платформах. Якщо геймери знають, що вони можуть окупити гроші, витрачені на покупку гри, і, можливо, заробити більше грошей, вони можуть бути менш зацікавлені в піратських іграх, згідно Rawg.

Ряд інших стартапів, включаючи Crycash, Enjin, Dmarket і багато інших, також намагаються допомогти гравцям монетизувати свій ігровий час. Rawg наведено тому, що у нього вже є робоча система, яка може використовувати блокування для запобігання піратства в масштабі.

1.3.5 Blockchain для управління правами

Ще один засіб, Dot Blockchain Media (https://verifi.media), який розробляє новий формат файлів - дублюється «.BC» - для уніфікації управління правами в музичній індустрії, також використовує криміналістичне водяне маркування для вбудовування адрес блок-ланцюжків в музичні файли. На відміну від CustosTech, Dot Blockchain не хоче вистежувати піратів музичних файлів - принаймні, не безпосередньо. Замість цього він просто хоче, щоб вся музична індустрія реєструвала музичну інформацію, від виконавця до продюсера і іншу комерційну інформацію про незмінній системі, яка є блоковим ланцюгом.

Компанія вже випустила музичний файл «.BC» - STOLAR «Forget and feel», який розповсюджується FUGA. Пісня доступна на iTunes, Google, Spotify і інших музичних платформах, а інформацію про водяні знаки можна прочитати за допомогою програми Digimarc Discover.

1.4 Архітектура клієнт-сервер

Архітектура клієнт-сервер – архітектура комп'ютерної мережі, в якій безліч клієнтів (віддалених процесорів) запитують і отримують доступ до послуг від сервера. Клієнт і сервер взаємодіють між собою за допомогою різних мережевих протоколів, таких як IP протокол, HTTP протокол, FTP та інші. Вибір протоколу взаємодії залежить від сфери його застосування. Комп'ютери клієнтської сторони надають інтерфейс, що дозволяє користувачеві комп'ютера запитувати доступ до послуг сервера і виводити результати, отримані від сервера. Сервер очікує, поки надійдуть запити від клієнтів, і потім відповідає на них. В ідеалі, сервер надає стандартизований зрозумілий інтерфейс клієнтам, щоб клієнт не був обізнаний про специфіку системи (тобто апаратного і програмного забезпечення), яка надає послуги.

Ця обчислювальна модель особливо ефективна в тому випадку, якщо у клієнтів і сервера є чітко визначені завдання, які вони зазвичай виконують. Безліч клієнтів може звертатися до даних на сервері одночасно, і в той же час комп'ютери клієнтів можуть виконувати інші дії. Так як серверний і клієнтський комп'ютер вважаються інтелектуальними пристроями, модель клієнт-сервер повністю відрізняється від старої мейнфрейм моделі, в якій центральний мейнфрейм-комп'ютер виконував всі завдання для пов'язаних з ним «німих» терміналів. Отже, клієнтська частина системи, побудованої в рамках даної парадигми, запитує послуги, а сервер – їх представляє. Під послугами в даному випадку розуміються різні ресурси: дані, файли, об'єкти, час процесора, пристрої відображення і т.д. Для такої системи типові такі властивості:

– запит на послуги являє собою команду, що потрібно виконати від сервера, і найчастіше абстрактний; сервер сам визначає, якими засобами його виконати;

– ідеальне клієнт-серверне програмне забезпечення незалежно від операційної системи або апаратної платформи;

– розміщення клієнтів і серверів часто зрозуміле для користувача;

– сервер може стати клієнтом, а клієнт – сервером;

– система клієнт-сервер може бути масштабована горизонтально, шляхом додавання або вилучення клієнтських станцій, з незначним впливом на продуктивність чи вертикально, шляхом міграції на більші і швидкі серверні машини або мульти сервери;

– клієнт-серверну взаємодію починає клієнт, сервер лише відповідає на запити клієнта.

Концепція клієнт-сервер спрямована, перш за все, на поділ навантаження між учасниками процесу взаємодії, а також на розмежування коду замовника послуг (тобто клієнта) і постачальника (тобто сервера).

Прийнято виділяти два види архітектури взаємодії клієнт-сервер: дворівневу архітектуру і багаторівневу, яку іноді ще трирівневою, проте це окремий випадок. На рисунку 1.2 показано схему дворівневої архітектури.

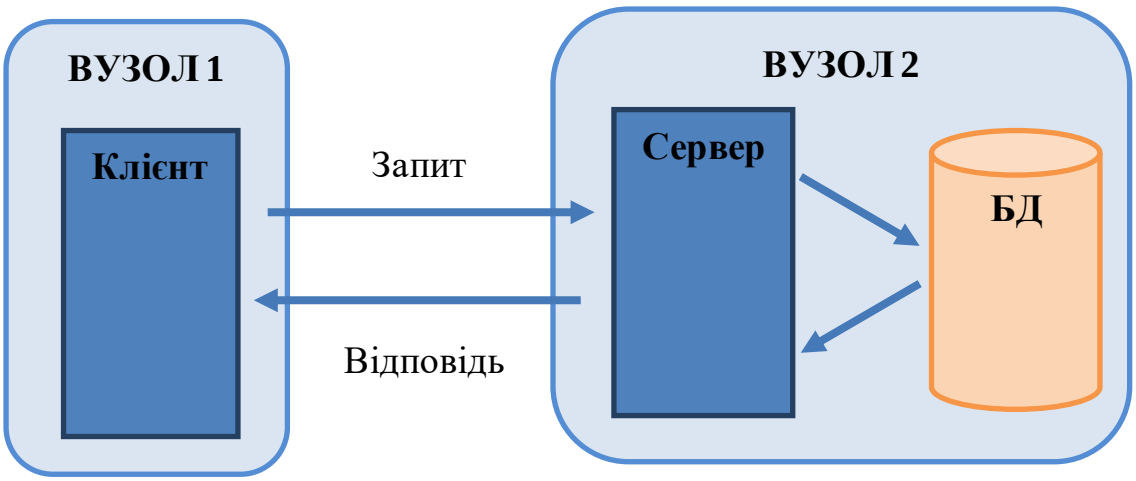


Рисунок 1.2 – Дворівнева архітектура клієнт-сервер

В рамках дворівневої архітектури три базові компоненти додатка розподіляються між вузлами (клієнтом і сервером) різними способами.

Залежно від способу розподілу компонентів виділяють різні моделі дворівневої архітектури (див. рисунок 1.3).

Дворівнева архітектура передбачає поділ логіки, даних і роботи програми між клієнтськими і серверними пристроями. Всю або майже всю логіку і дані додатків можна знайти на сервері. Клієнт звертається до сервера для виконання

конкретних, специфічних для додатка завдань. Процедура розгляду заяв в рамках даної архітектури відбувається на одній машині, без використання сторонніх ресурсів, що призводить до посилення вимог до продуктивності сервера, проте забезпечує надійність.

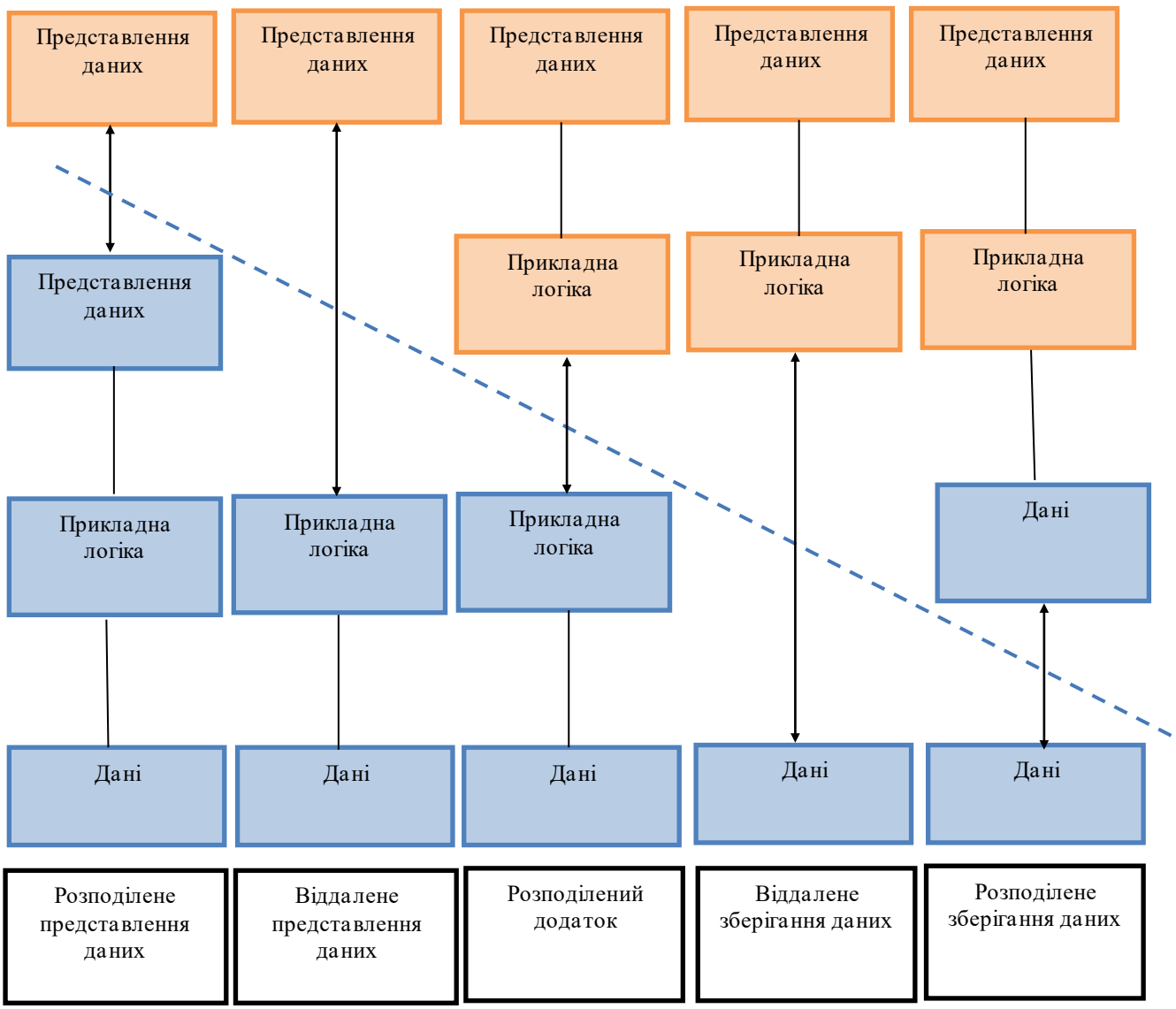


Рисунок 1.3 – Моделі дворівневої архітектури

Багаторівнева архітектура заснована на моделі сервера додатків, де додаток розділений на дві і більше частин (рисунок 1.4).

Кожна з цих частин може виконуватися на окремій машині. Частини програми взаємодіють між собою за допомогою повідомлень заздалегідь визначеного формату. Тобто, запит користувача обробляється декількома серверами, що дозволяє знизити навантаження на сервер завдяки розподілу

операцій. Однак, надійність такого підходу нижча, ніж дворівнева архітектура.

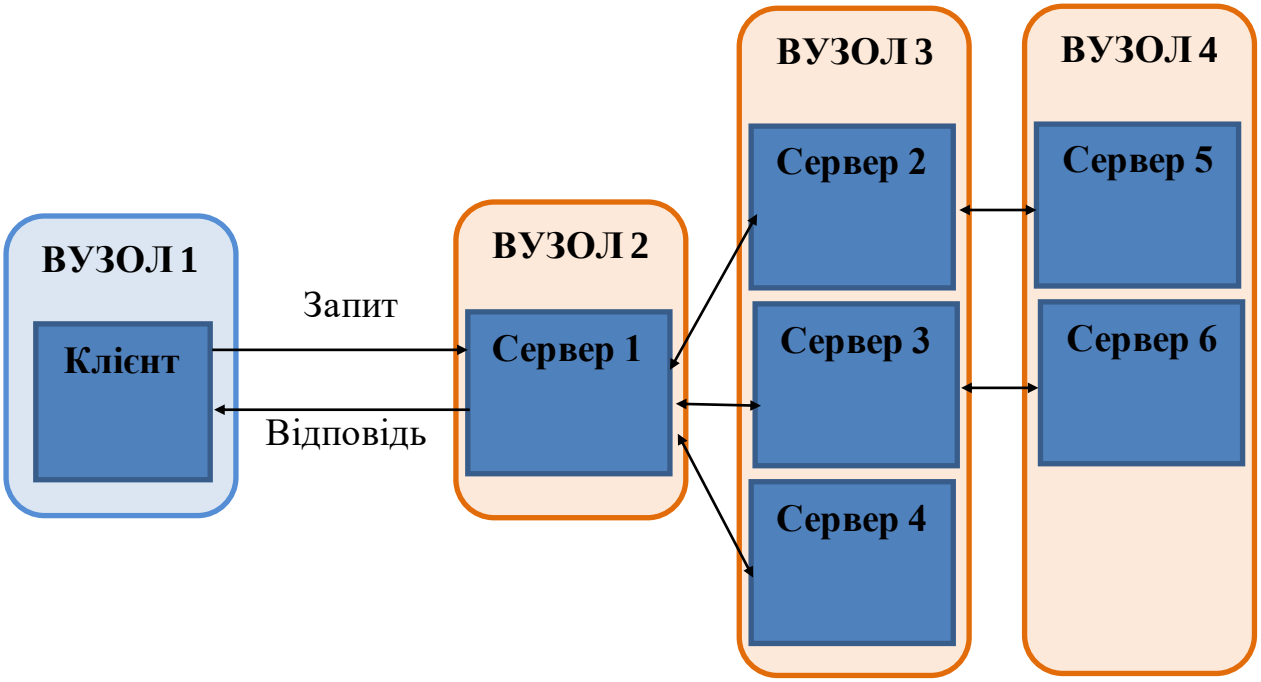


Рисунок 1.4 – Багаторівнева архітектура

До переваг клієнт-серверної архітектури відносять поділ коду клієнтської і серверної частин, знижені вимоги до комп'ютерів клієнтів (за рахунок виконання більшості обчислювальних операцій на сервері), гнучкість.

Як недоліки даної архітектури слід зазначити високу вартість серверного обладнання (значно перевищує вартість клієнтського), необхідність обслуговування сервера спеціальним співробітником, при недостатній потужності серверного обладнання робота системи дуже помітно погіршується: збільшується час очікування відповіді від сервера, потужності сервера не вистачає для задоволення запитів всіх клієнтів.

Вибір даної архітектури для розроблюваних додатків є досить природним і по суті єдино можливим, виходячи з ідей і вимог. До того ж, дана архітектура дозволяє розширювати і здійснювати модифікації функціональної частини програми практично непомітно для кінцевих користувачів, що в процесі експлуатації буде дуже актуально, а в разі використання тонкого клієнта (тобто роботи з додатком через браузер) – повністю приховати внутрішні зміни від користувачів.

1.5 Висновки до першого розділу

У першому розділі дипломної роботи проведено аналіз предметної області дослідження, наведено основні характеристики технології Blockchain, зокрема її використання у медіа сфері, для боротьби з піратством, полегшення контролю за контентом та управління правами користувачів. Описано можливості смарт-контракту як способу обміну цифровими цінностями (даними) в блокчейні. Розглянуто особливості та моделі архітектури клієнт-сервер.

2 СИСТЕМНИЙ АНАЛІЗ МЕХАНІЗМУ АВТОРСЬКИХ ПРАВ НА МУЛЬТИМЕДІЙНІ ФАЙЛИ

Існує безліч реалізацій технології Blockchain, на основі яких може бути побудований сервіс. У даній роботі була використана технологія Multichain [6], так як вона дозволяє створювати приватні Blockchain, що дуже зручно для дослідницьких цілей. У цьому сенсі Multichain залишає вибір за кінцевим користувачем і дозволяє йому контролювати приватність і публічність ланцюга, цільовий час блоків, способи взаємодії сторін, вибір цих сторін, максимальний розмір блоку і метадані, які можна включати в транзакції. Крім того, для даної технології існує оболонка для виклику API-команд в програмному коді для мови Python [7, 8]. Multichain підтримує велику кількість різних API-команд, необхідних для реалізації сервісу.

Однак при продовженні розвитку розробки можливий перехід на іншу технологію, де зручніше реалізований механізм реєстрації активів.

2.1 Опис основних системних вимог

На даному етапі роботи були сформульовані вимоги з позиції різних груп кінцевих користувачів. Далі наведені шаблони, розроблені для опису вимог, а також самі вимоги до систем за сценарієм «Музика і метадані» та «Аудіо-фрагменти і метадані». В якості першого рівня абстракції вимог до системи були обрані призначені для користувача історії, так як їх легко сформулювати спільно із замовником, а потім перетворити в варіанти використання і точніше описати систему. [9]

Для користувацьких історій був розроблений наступний шаблон:

У якості <роль в рамках проектованої системи> я хочу <опис бажаних можливостей>, щоб <мета>.

Кожній користувацькій історії присвоюється унікальний ідентифікатор. Ідентифікатор будується за такими принципами:

1. Кожен ідентифікатор починається з абревіатури US (user story)

користувацька історія.

2. За нею слфдує позначення сценарію для якого діє користувацька історія: ММ – для сценарію «Музика і метадані», АМ – для сценарію «Аудіо-фрагменти і метадані».

3. Далі вказується роль суб'єкта, який бажає використовувати систему: N – користувач, R – правовласник.

4. Останнім вказується порядковий номер користувацької історії.

Приклади позначень і розшифровка:

US MMR00 – користувацька історія для сценарію «Музика і метадані» від правовласника №00;

US MMN00 – користувацька історія для сценарію «Музика і метадані» від користувача №00;

US AMN00 – користувацька історія для сценарію «Аудіо-фрагменти і метадані» від користувача №00.

Для сценарію «Музика і метадані» були розроблені наступні користувацькі історії.

З боку правовласника «я хочу ...»

US MMR01 реєструвати записи своїх композицій, щоб захистити свої права на них.

US MMR02 прикріплювати до своїх зареєстрованих композицій метадані, щоб розширити інформацію про композиції.

US MMR03 читати метадані, додані до моїх композицій іншими користувачами і правовласниками, щоб аналізувати і вдосконалювати свою творчість.

US MMR04 вказувати співавторів, партнерів та ін., щоб полегшити процедуру розподілу прибутку від продажу пісні.

US MMR05 продавати свої зареєстровані записи, щоб заробляти гроші.

US MMR06 завантажувати свої записи на сервер, щоб продавати їх в цифровому форматі.

US MMR07 щоб проводилася аутентифікація записів, щоб уникнути дублікатів та плагіату.

US MMR08 проводити пошук зареєстрованих записів за різними параметрами (назва, альбом, виконавець, тощо), щоб дізнатися чи додати інформацію до запису.

З боку користувача «я хочу ...»

US MMN01 додавати метадані до будь-якого зареєстрованого запису, щоб висловити свою думку і допомогти музикантам.

US MMN02 купувати зареєстровані записи, щоб оплатити працю музикантів та інших задіяних осіб і зробити можливою їх подальшу роботу.

US MMN03 читати метадані, додані до будь-якого з зареєстрованих записів, щоб дізнатися думки інших користувачів і додаткову інформацію про записи.

US MMN04 проводити аутентифікацію записів, щоб дізнатися джерело (в даному випадку, правовласника).

US MMN05 діяти анонімно, щоб ніхто не дізнався моїх уподобань.

US MMN06 діяти під псевдонімом, щоб отримувати пропозиції, виключаючи можливість того, що мої уподобання вкажуть на мою реальну особистість.

US MMN07 діяти із зазначенням моєї реальної особистості, щоб отримувати кращі пропозиції, в тому числі і про різноманітні сервіси.

US MMN08 проводити пошук зареєстрованих записів за різними параметрами (назва, альбом, виконавець, тощо), щоб дізнатися інформацію про записи або купити запис.

Для сценарію «Аудіо-фрагменти і метадані» були розроблені наступні призначені користувацькі історії.

З боку ккористувача «я хочу ...»

USAMN01 реєструвати аудіо-фрагменти, щоб забезпечити можливість їх подальшого використання.

US AMN02 ініціювати перевірку аудіо-фрагментів на маніпуляції, щоб переконатися в достовірності фрагмента.

US AMN03 прикріплювати метадані до будь-якого зареєстрованого аудіо-фрагменту, щоб висловити свою думку.

US AMN04 читати метадані, додані до будь-якого фрагменту іншими користувачами, щоб дізнатися додаткову інформацію й інші думки про фрагмент.

US AMN05 щоб система автоматично проводила перевірку реєстрованих фрагментів на маніпуляції, щоб підтвердити або поставити під сумнів автентичність фрагменту.

US AMN06 завантажити будь-який з зареєстрованих фрагментів, щоб використовувати його в подальшому.

US AMN07 щоб проводилася аутентифікація фрагментів, щоб уникнути дублікатів та дізнатися джерело.

US AMN08 проводити пошук фрагментів за різними параметрами (назва, дата, тощо), щоб завантажити фрагмент для подальшого експерименту, ініціювати перевірку фрагмента на маніпуляції, прочитати або прикріпити метадані.

2.2 Варіанти використання

Варіанти використання в даному випадку виступають в якості засобу опису функціональних вимог до сервісу. Їх легко виділити з розроблених для користувача історій, вони чітко описують потреби кінцевих користувачів, які повинна задовольняти система. Для опису варіантів використання було створено наступний шаблон:

1. Ідентифікатор.
2. Актор (роль суб'єкта в системі).
3. Пріоритет (від 1 – низький до 10 – високий).
4. Частота використання (від 1 – вкрай рідко до 10 – дуже часто).
5. Батьківська користувацька історія (користувацька історія, яку реалізує даний варіант використання).
6. Передумови (опис початкового стану, причини)
7. Мета/результат.
8. Порядок дій (приклад або алгоритм)

9. Післяумови.

Кажному варіанту використання співставляється унікальний ідентифікатор. Ідентифікатор будується за такими принципами:

– кожен ідентифікатор починається з абревіатури UC (use case) – варіант використання;

– за нею йде позначення сценарію, для якого діє варіант використання: ММ – для сценарію «Музики і метадані», АМ – для сценарію «Аудіо-фрагменти і метадані».

– останнім вказується порядковий номер варіанта використання.

Приклади позначень і розшифровка наведені нижче:

UCMM00 – варіант використання для сценарію «Музика і метадані» №00;

UC AM00 – варіант використання для сценарію «Аудіо-фрагменти і метадані» №00.

Варіанти використання (функціональні вимоги) для сценарію «Музика і метадані» виділені на основі раніше описаних користувацьких історій, представлені нижче.

1. UC MM01 Зареєструвати правовласника

Актор	Правовласник
Пріоритет	1
Частота використання	6
Передумови	Правовласник хоче використовувати функції системи.
Мета/результат	Створити рахунок у системі, щоб отримати доступ до функціональності системи.
Порядок дій	Натиснути «Зареєструвати правовласника». Ввести реєстраційні дані Натиснути «ОК»
Післяумови	Правовласник зареєстрований і може бути аутентифікований в системі.

2. UC MM02 аутентифікувати правовласника

Актор	Правовласник
Пріоритет	1
Частота використання	8
Батьківська призначена для користувача історія	US MMR01, US MMR02, US MMR03, US MMR04, US MMR06, US MMR07, US MMR08
Передумови	У правовласника є акаунт в системі.
Мета/результат	Бути аутентифікованим в системі, щоб використовувати функції системи.
Порядок дій	Ввести ім’я користувача. Ввести пароль. Натиснути «ОК».
Післяумови	Правовласник аутентифікований в системі і може з нею взаємодіяти.

3. UC MM03 Зареєструвати запис

Актор	Правовласник
Пріоритет	1
Частота використання	7
Батьківська призначена для користувача історія	US MMR01
Передумови	Запис існує, належить правовласнику, правовласник аутентифікований в системі.
Мета/результат	Зареєструвати запис в системі, щоб захистити свої права на запис і мати можливість його продажу.
Порядок дій	Натиснути «Зареєструвати запис». Вибрати запис у вікні вибору, натиснути «ОК». Згенерувати хеш-значення для файлу. Провести аутентифікацію запису. Провести опис, вказати учасників. Завантажити запис на сервер. Зберегти метадані.

Післяумови	Запис зареєстрований.

4. UC MM04 Прикріпити метадані

Актор	Правовласник
Пріоритет	2
Частота використання	6
Батьківська призначена для користувача історія	US MMR01, US MMR02, US MMR04
Передумови	Запис зареєстровано в системі
Мета/результат	Прикріпити метадані до запису, щоб доповнити інформацію про запис.
Порядок дій	Натиснути «Додати метадані». Ввести данні. Натиснути «ОК».
Післяумови	Метадані прикріплені до запису, збережені їх видно для всіх користувачів і правовласників.

5. UC MM05 Прочитати метадані

Актор	Правовласник
Пріоритет	2
Частота використання	6
Батьківська призначена для користувача історія	US MMR03
Передумови	Користувач (-і) або правовласник (-и) прикріпив (-и) метадані до запису.
Мета/результат	Прочитати метадані, прикріплені до будь-якої з зареєстрованих записів, щоб дізнатися думки інших користувачів/правовласників; прочитати метадані до своїх записів, щоб аналізувати і вдосконалювати свою творчість.
Порядок дій	Знайти композицію Відкрити інформацію про записи. Прочитати метадані.

Післяумови	Метадані прочитані.

6. UC MM06 Вказати учасників

Актор	Правовласник
Пріоритет	1
Частота використання	7
Батьківська призначена для користувача історія	US MMR01, US MMR04
Передумови	Іде реєстрація запису.
Мета/результат	Вказати учасників, щоб спростити розподіл прибутку від продажу запису між учасниками.
Порядок дій	Ввести учасників у відповідних полях у вікні. Натиснути «ОК».
Післяумови	Учасники вказані і збережені.

7. UC MM07 Продати запис

Актор	Система, правовласник
Пріоритет	2
Частота використання	8
Батьківська призначена для користувача історія	US MMR05
Передумови	Запис зареєстрований в системі, учасники вказані.
Мета/результат	Продавати записи, щоб заробити грошей і зробити можливою подальшу творчість.
Порядок дій	Чекати, поки користувач натисне кнопку «Купити запис». Виконати транзакцію. Якщо транзакція виконана, надати користувачеві можливість завантажити запис.
Післяумови	Транзакція виконана, прибуток розподілений між зазначеними учасниками.

8. UC MM08 Завантажити запис на сервер

Актор	Система

Пріоритет	2
Частота використання	7
Батьківська призначена для користувача історія	US MMR01, US MMR06
Передумови	Іде реєстрація запису, проведена його аутентифікація (запис не був зареєстрований раніше).
Мета/результат	Завантажити запис на сервер, щоб уможливити подальший продаж запису.
Порядок дій	Завантажити обраний користувачем запис. Підтвердити завантаження.
Післяумови	Запис завантажено на сервер, посилання для подальшого завантаження збережено.

9. UC MM09 аутентифікувати запис

Актор	Система
Пріоритет	2
Частота використання	7
Батьківська призначена для користувача історія	US MMR01, US MMR07, US MMN04
Передумови	а) йде реєстрація запису, хеш-значення для запису отримано; б) файл для аутентифікації обраний, хеш-значення для нього отримано.
Мета/результат	Аутентифікувати запис, щоб уникнути плагіату і дублікатів; щоб дізнатися джерело (в даному випадку – правовласника).
Порядок дій	Порівняти отримане хеш-значення зі значеннями вже зареєстрованих композицій: а) якщо збіг знайдено, скасувати реєстрацію; б) якщо збіг не знайдено, показати інформацію про записи

Післяумови	а) прийнято рішення, чи можливо продовження реєстрації; б) виведена інформація про запис або повідомлення, що такий запис не зареєстрований.

10. UC MM10 Отримати хеш-значення

Актор	Система
Пріоритет	1
Частота використання	7
Батьківська призначена для користувача історія	US MMR01, US MMR06, US MMR07, US MMN04
Передумови	Йде реєстрація/аутентифікація запису, запис обрано.
Мета/результат	Отримати хеш-значення для запису, щоб його зареєструвати і/або аутентифікувати.
Порядок дій	Виконати алгоритм генерації хеш-значення для запису. Зберегти отримане хеш-значення.
Післяумови	Хеш-значення отримано, збережено і може бути використано в подальшому.

11. UC MM11 Знайти запис

Актор	Правовласник
Пріоритет	2
Частота використання	7
Батьківська призначена для користувача історія	US MMR02, US MMR03, US MMR08
Передумови	Правовласник аутентифікований в системі.
Мета/результат	Знайти запис, щоб дізнатися інформацію про запис і/або прикріпити метадані.
Порядок дій	Ввести параметр пошуку в поле пошуку. Натиснути «Знайти».

Післяумови	Результати пошуку виведені.

12. UC MM12 Зареєструвати користувача

Актор	Користувач
Пріоритет	1
Частота використання	6
Передумови	Суб'єкт хоче використовувати систему.
Мета/результат	Створити рахунок у системі, щоб отримати доступ до її функцій.
Порядок дій	Натиснути «Зареєструвати користувача». Ввести обов'язкові дані. Ввести додаткові (особисті) дані (щоб діяти із зазначенням реальної особистості). Натиснути «ОК».
Післяумови	Суб'єкт зареєстрований в системі і може бути в ній аутентифікований.

13. UC MM13 аутентифікація користувача

Актор	Користувач
Пріоритет	1
Частота використання	8
Батьківська призначена для користувача історія	US MMN01, US MMN02, US MMN03, US MMN04, US MMN06, US MMN07, US MMN08
Передумови	У користувача є акаунт в системі.
Мета/результат	Бути аутентифіконим в системі, щоб використовувати її функції.
Порядок дій	Ввести ім'я користувача. Ввести пароль. Натиснути «ОК».
Післяумови	Користувач аутентифікований в системі і може з нею взаємодіяти.

14. UC MM14 Купити запис

Актор	Користувач
Пріоритет	2
Частота використання	7
Батьківська призначена для користувача історія	US MMN02
Передумови	Запис обрано.
Мета/результат	Купити запис, щоб оплатити роботу музикантів і ін. учасників і заохотити їх подальшу роботу.
Порядок дій	Знайти запис. Натиснути «Купити запис». Підтвердити транзакцію. Якщо транзакція виконана, підтвердити завантаження запису.
Післяумови	Користувач оплатив запис і завантажив його з сервера.

15. UC MM15 Прочитати метадані

Актор	Користувач
Пріоритет	2
Частота використання	6
Батьківська призначена для користувача історія	US MMN03
Передумови	Метадані прикріплені і збережені, видно для всіх користувачів.
Мета/результат	Прочитати метадані, прикріплені користувачами або правовласниками до будь-якого із зареєстрованих записів, щоб дізнатися інші думки і додаткову інформацію про запис.
Порядок дій	Знайти запис. Відкрити інформацію про запис. Прочитати метадані
Післяумови	Метадані, прикріплені до вибраного запису,

прочитані.

16. UC MM16 Прикріпити метадані

Актор	Користувач
Пріоритет	2
Частота використання	5
Батьківська призначена для користувача історія	US MMN01
Передумови	Запис зареєстровано в системі.
Мета/результат	Прикріпити метадані до вибраної композиції, щоб висловити свою думку.
Порядок дій	Натиснути «Прикріпити метадані». Ввести текст. Натиснути «ОК».
Післяумови	Метадані прикріплені, збережені і видні для інших користувачів і правовласників.

17. UC MM17 Знайти запис

Актор	Користувач
Пріоритет	2
Частота використання	7
Батьківська призначена для користувача історія	US MMN01, US MMN02, US MMN03, US MMN08
Передумови	Користувач аутентифікований в системі або діє анонімно.
Мета/результат	Знайти запис, щоб дізнатися інформацію про нього і/або прикріпити метадані до запису, та/або купити запис.
Порядок дій	Ввести параметр пошуку в поле пошуку. Натиснути «Знайти».
Післяумови	Результати пошуку видно для користувача.

18. UC MM18 Діяти анонімно

Актор	Користувач

Пріоритет	2
Частота використання	Не визначена
Батьківська призначена для користувача історія	US MMN05, US MMN01, US MMN03, US MMN08
Передумови	Система дає можливість діяти анонімно.
Мета/результат	Діяти анонімно, щоб неможливо було встановити зв'язок між діями користувача, а також між дією і реальною особою користувача.
Порядок дій	Частково використовувати функціонал системи без реєстрації/аутентифікації.
Післяумови	Ніякі дії користувача не можуть бути пов'язані між собою або з реальною особою користувача.

19. UC MM18 Діяти під псевдонімом

Актор	Користувач
Пріоритет	2
Частота використання	Не визначена
Батьківська призначена для користувача історія	US MMN06, US MMN01, USMMN02, US MMN03, US MMN08
Передумови	Система надає можливість діяти під псевдонімом.
Мета/результат	Діяти під псевдонімом, щоб отримувати пропозиції по записах, але виключити можливість встановлення зв'язку дій користувача з його реальною особою.
Порядок дій	При реєстрації не вводити особисті дані. Пройти аутентифікацію в системі.
Післяумови	Ніякі дії користувача не можуть бути пов'язані з його реальною особою.

20. UC MM18 Діяти із зазначенням реальної особистості

Актор	Користувач
Пріоритет	2

Частота використання	Не визначена
Батьківська призначена для користувача історія	US MMN07, US MMN01, US MMN02, US MMN03, US MMN08
Передумови	Система надає можливість діяти під реальним ім’ям.
Мета/результат	Діяти з вказанням реальної особистості, щоб отримувати кращі пропозиції.
Порядок дій	Ввести особисті дані при реєстрації. Пройти аутентифікацію в системі.
Післяумови	Всі дії користувача пов’язані з його реальною особистістю, користувач отримує пропозиції згідно його дій.

Варіанти використання (функціональні вимоги) для сценарію «Аудіо-фрагменти і метадані», виділені на основі розроблених раніше призначених для користувача історій, представлені нижче.

1. UC AM01 Зареєструвати користувача

Актор	Користувач
Пріоритет	1
Частота використання	6
Передумови	Суб’єкт хоче використовувати функції системи.
Мета/результат	Створити рахунок у системі, щоб отримати доступ до функцій системи.
Порядок дій	Натиснути «Зареєструвати користувача». Ввести дані для реєстрації. Натиснути «ОК».
Післяумови	Користувач зареєстрований в системі і може бути в ній аутентифікований.

2. UC AM02 аутентифікувати користувача

Актор	Користувач
Пріоритет	1

Частота використання	8
Батьківська призначена для користувача історія	US AMN01, US AMN02, US AMN03, US AMN04, US AMN06, US AMN07, US AMN08
Передумови	Користувач зареєстрований в системі.
Мета/результат	Бути аутентіфікованим в системе, щоб використовувати функції системи.
Порядок дій	Ввести ім’я користувача Ввести пароль Натиснути «ОК»
Післяумови	Користувач аутентифікований в системі і може з нею взаємодіяти.

3. UC AM03 Зареєструвати аудіо фрагмент

Актор	Користувач
Пріоритет	1
Частота використання	7
Батьківська призначена для користувача історія	US AMN01
Передумови	Аудіо-фрагмент записаний.
Мета/результат	Зареєструвати аудіо-фрагмент, щоб він міг бути знайдений, аутентифікований і використаний в подальшому.
Порядок дій	Натиснути «Зареєструвати аудіо фрагмент». Вибрати файл у вікні вибору файлу, натиснути «ОК». Згенерувати хеш-значення для аудіо-фрагмента. Аутентифікувати аудіо-фрагмент. Це місце потребує опису фрагмента. Завантажити фрагмент на сервер. Провести перевірку на маніпуляції зберегти метадані

Післяумови	Аудіо-фрагмент зареєстрований.

4. UC AM04 Ініціювати перевірку на маніпуляції

Актор	Користувач
Пріоритет	1
Частота використання	5
Батьківська призначена для користувача історія	US AMN02
Передумови	Аудіо фрагмент зареєстрований в системі.
Мета/результат	Ініціювати перевірку аудіо-фрагмента на автентичність щоб перевірити маніпуляції, фрагмента.
Порядок дій	Знайти фрагмент. Натиснути «Перевірити фрагмент». Очікувати результатів перевірки.
Післяумови	Аудіо-фрагмент перевірений, результати перевірки видно користувачеві.

5. UC AM05 Прикріпити метадані

Актор	Користувач
Пріоритет	2
Частота використання	6
Батьківська призначена для користувача історія	US AMN01, US AMN03
Передумови	а) аудіо-фрагмент зареєстрований; б) іде реєстрація аудіо-фрагмента.
Мета/результат	Прикріпити метадані до будь-якого з зареєстрованих аудіо-фрагментів, щоб викласти свою думку або доповнити інформацію про фрагменті.
Порядок дій	Натиснути «Прикріпити метадані». Ввести текст. Натиснути «ОК».

Післяумови	Метадані прикріплені і збережені, видно всім користувачам.

6. UC AM06 Прочитати метадані

Актор	Користувач
Пріоритет	2
Частота використання	6
Батьківська призначена для користувача історія	US AMN04
Передумови	Метадані прикріплені і збережені.
Мета/результат	Прочитати метадані, прикріплені до будь-якого зареєстрованого аудіо-фрагменту, щоб дізнатися думки інших користувачів і додаткову інформацію про фрагмент.
Порядок дій	Знайти аудіо-фрагмент Відкрити інформацію про фрагмент Прочитати метадані
Післяумови	Метадані, прикріплені до заданого фрагменту, прочитані.

7. UC AM07 Автоматично проводити перевірку аудіо-фрагмента на маніпуляції

Актор	Система
Пріоритет	1
Частота використання	7
Батьківська призначена для користувача історія	US AMN01, US AMN05
Передумови	Іде реєстрація аудіо фрагмента, фрагмент завантажений на сервер.
Мета/результат	Провести перевірку аудіо-фрагмента на маніпуляції, щоб виявити і позначити підозрілі фрагменти.
Порядок дій	Запустити програму перевірки для аудіо-

	фрагмента Отримати і обробити результати. Маркувати, якщо фрагмент визнаний підозрілим.
Післяумови	Підозрілі фрагменти марковані.

8. UC AM08 Завантажити аудіо-фрагмент

Актор	Користувач
Пріоритет	1
Частота використання	7
Батьківська призначена для користувача історія	US AMN06
Передумови	Аудіо-фрагмент зареєстрований.
Мета/результат	Завантажити будь-який зареєстрований аудіо-фрагмент, щоб використовувати його в подальшому.
Порядок дій	Натиснути «Завантажити аудіо-фрагмент». Редагувати шлях завантаження (опціонально). Підтвердити завантаження
Післяумови	Аудіо-фрагмент завантажений.

9. UC AM09 аутентифікувати аудіо-фрагмент

Актор	Користувач
Пріоритет	1
Частота використання	7
Батьківська призначена для користувача історія	US AMN01, US AMN07
Передумови	а) йде реєстрація аудіо фрагмента, хеш значення для нього отримано; б) хеш-значення для аудіо фрагмента отримано.
Мета/результат	Аутентифікувати аудіо-фрагмент, щоб запобігти повторному завантаженню або дізнатися джерело.
Порядок дій	Порівняти хеш-значення аудіо-фрагмента зі значеннями вже зареєстрованих фрагментів:

	а) якщо збіг знайдено, скасувати реєстрацію аудіо-фрагмента б) якщо збіг не знайдено, вивести інформацію про аудіо-фрагмент.
Післяумови	а) аудіо-фрагмент аутентифікований, прийнято рішення про продовження процесу реєстрації. б) виведена інформація про аудіо-фрагмент, якщо збігів, не знайдено – повідомлення про відсутність збігів.

10. UC AM10 Отримати хеш-значення

Актор	Система
Пріоритет	1
Частота використання	8
Батьківська призначена для користувача історія	US AMN01, US AMN07
Передумови	а) аудіо-фрагмент обраний; б) Пароль введений.
Мета/результат	Отримати хеш-значення, щоб забезпечити подальшу роботу системи.
Порядок дій	Виконати алгоритм генерації хеш-значення. Зберегти отримане значення.
Післяумови	Хеш-значення отримано і збережено для подальшого використання.

11. UC AM11 Знайти аудіо-фрагмент

Актор	Користувач
Пріоритет	2
Частота використання	6
Батьківська призначена для користувача історія	US AMN08
Передумови	Користувач аутентифікований в системі.
Мета/результат	Знайти аудіо-фрагмент, щоб

	переглянути/прикріпити метадані та/або завантажити аудіо-фрагмент.
Порядок дій	Ввести параметр пошуку в поле пошуку. Натиснути «Знайти».
Післяумови	Результати пошуку видно користувачеві.

12. UC AM12 Завантажити аудіо-фрагмент на сервер

Актор	Система
Пріоритет	1
Частота використання	7
Батьківська призначена для користувача історія	US AMN01
Передумови	Йде реєстрація аудіо-фрагмента, фрагмент обраний, отримано хеш-значення.
Мета/результат	Завантажити аудіо-фрагмент на сервер, щоб забезпечити можливість завантаження і подальшого використання іншими користувачами.
Порядок дій	Завантажити обраний аудіо-фрагмент на сервер. Підтвердити завантаження
Післяумови	Аудіо фрагмент завантажено на сервер, реєстрація триває.

2.3 Висновки до другого розділу

У другому розділі дипломної роботи описано основні системні вимоги до сервісу, шаблони, розроблені для опису вимог, а також самі вимоги за сценаріями «Музика і метадані» та «Аудіо-фрагменти і метадані». Запропоновано варіанти використання, які є засобом опису функціональних вимог до сервісу.

3 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СЕРВІСУ

3.1 Діаграми компонентів

На етапі побудови архітектури сервісу на основі визначених раніше вимог і з урахуванням обраних технологій були розроблені діаграми варіантів використання, компонентів, діяльності та послідовності. Дані діаграми в різних аспектах ілюструють склад і поведінку розроблюваних систем. У цьому розділі розглядається лише кілька прикладів різних видів діаграм, повний перелік діаграм знаходиться в додатках. Діаграма компонентів відображає розкладання проектованої системи на структурні компоненти і відносини між ними. Як фізичні компоненти розглядаються файли, бібліотеки, модулі, пакети і т.д. Для сценарію «Музика і метадані» була розроблена структура компонентів, зображена на рисунку 3.1.

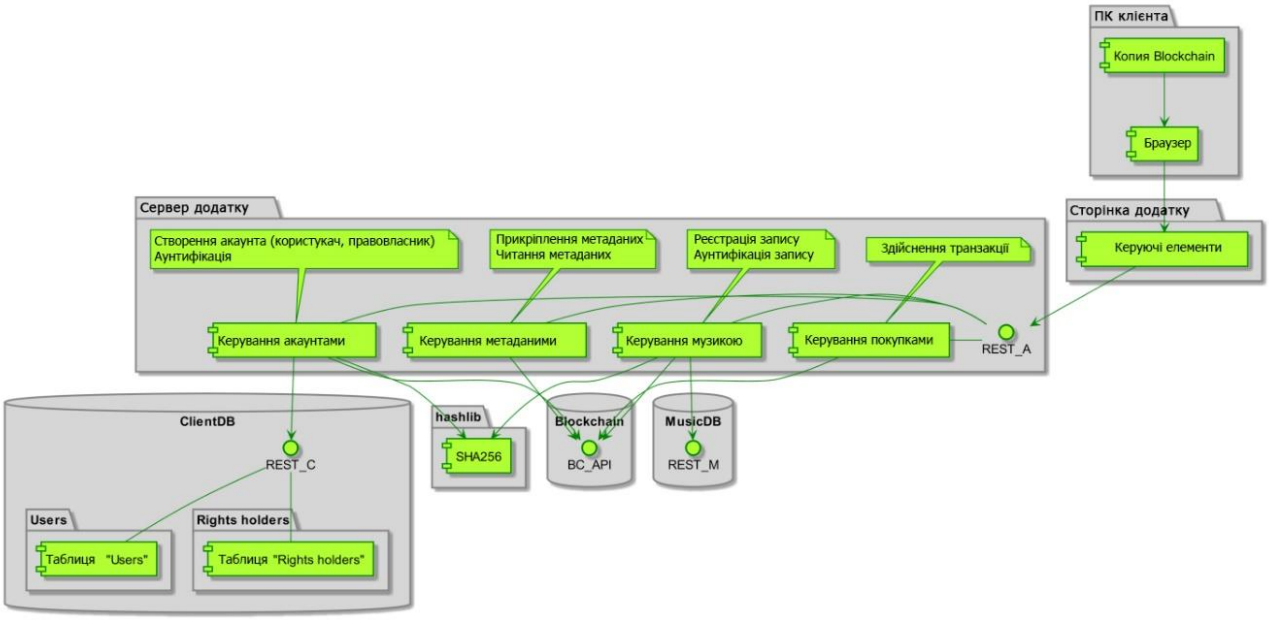


Рисунок 3.1 – Діаграма компонентів для сценарію «Музика і метадані»

ПК Клієнта повинен для доступу до інтернет-сторінки додатка мати встановлений браузер. Також на ПК Клієнта міститься копія Blockchain, яка

автоматично скачується при реєстрації і оновлюється при вході в систему.

Сторінка додатку містить керуючі елементи, що дозволяють взаємодіяти з системою і користуватися її функціоналом, відправляючи команди на сервер. Сервер додатку виконує функції відповідно до отриманих запитів.

Функціонал програми умовно розбитий на 4 частини:

– управління акаунтами (створення аккаунта користувача і правовласника, аутентифікація користувачів і правовласників у системі);

– управління музикою (реєстрація записів (доступна тільки для правовласників), аутентифікація записів);

– управління метаданими (прикріплення і читання метаданих);

– управління покупками (здійснення транзакцій).

База даних ClientDB містить дані про зареєстрованих користувачів (таблиця Users) і правовласників (таблиця Rights holders), доступ до бази даних здійснюється за допомогою SQL-запитів з коду серверної частини системи. Таблиця даних про користувачів містить наступні поля:

– Ім’я користувача;

– Хеш-значення пароля;

– Ім’я;

– Прізвище;

– Дата народження.

Таблиця даних про правовласника містить наступні поля:

– Ім’я користувача;

– Хеш-значення пароля;

– Опис.

База даних MusicDB містить дані про зареєстровані правовласниками записи і прикріплені метадані. Доступ до бази даних здійснюється за допомогою SQL-запитів з коду серверної частини системи.

Таблиця даних про музичні записи містить наступні поля:

– Назва музичного твору;

– Автори;

– Виконавці;
– Назва альбому;
– Рік виходу;
– Хеш-значення файлу запису;
– Посилання на файл запису;
– ID-активу в Blockchain.

Таблиця метаданих містить наступні поля:

– Хеш-значення запису, до якого прикріплені метадані;
– Текст (тіло метаданих);
– Хеш-значення тіла метаданих;
– ID-активу в Blockchain.

Бібліотека hashlib використовується для виклику конструктора алгоритму генерації хеш-значення SHA256. У Blockchain зберігається інформація про зареєстровані правовласниками записи, прикріплені метадані та проведені транзакції. Доступ до Blockchain здійснюється за допомогою JSON-RPC команд з коду серверної частини системи. Для сценарію «Аудіо-фрагменти і метадані» була розроблена структура компонентів зображена на рисунку 3.2.

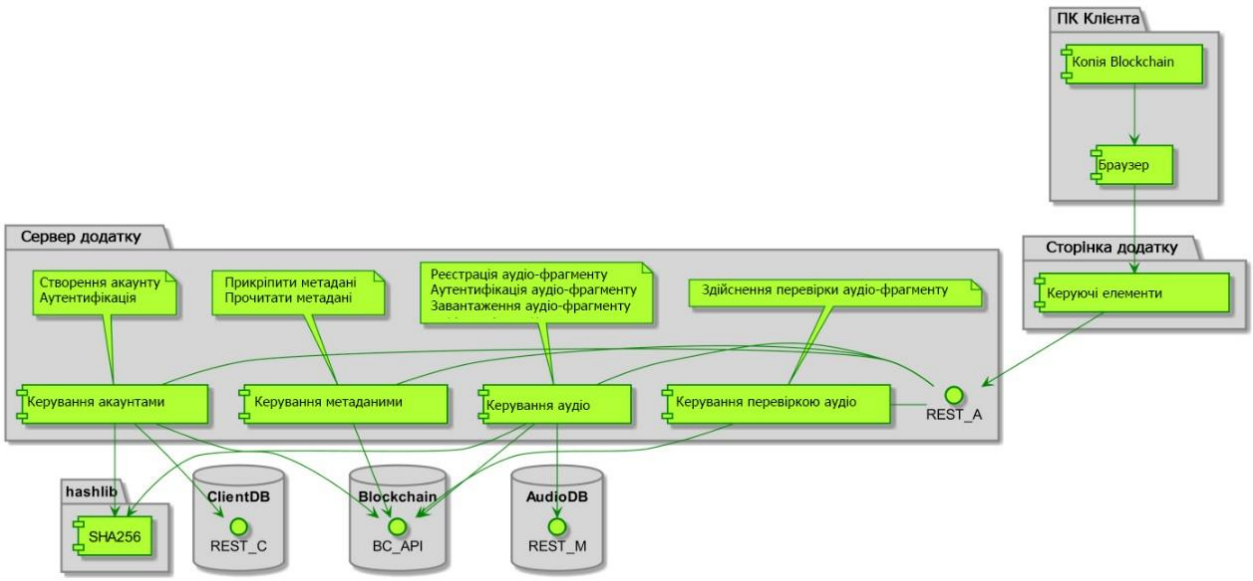


Рисунок 3.2 – Діаграма компонентів для сценарію «Аудіо-фрагменти і метадані»

ПК Клієнта повинен для доступу до інтернет-сторінки додатку мати встановлений браузер. Також на ПК Клієнта міститься копія Blockchain, яка автоматично скачується при реєстрації і оновлюється при вході в систему.

Сторінка додатку містить керуючі елементи, що дозволяють взаємодіяти з системою і користуватися її функціоналом, відправляючи команди на сервер.

Сервер додатку виконує функції відповідно до отриманих запитів. Функціонал програми умовно розбитий на 4 частини:

– керування обліковими записами (створення акаунта користувача, аутентифікація зареєстрованих користувачів в системі);

– управління аудіо-фрагментами (реєстрація аудіо-фрагментів, аутентифікація аудіо-фрагментів, завантаження аудіо-фрагментів);

– управління метаданими (прикріплення і читання метаданих);

– управління перевіркою аудіо-фрагментів (здійснення перевірки аудіо-фрагментів на наявність маніпуляцій, даний компонент надається сторонньою організацією).

База даних ClientDB містить дані про зареєстрованих користувачів, доступ до бази даних здійснюється за допомогою SQL-запитів з коду серверної частини системи. Таблиця даних про користувачів містить наступні поля:

– Ім’я користувача;

– Хеш-значення пароля.

База даних AudioDB містить дані про зареєстровані аудіо-фрагмент. Доступ до бази даних здійснюється за допомогою SQL-запитів з коду серверної частини системи. Таблиця даних про аудіо-фрагменти містить наступні поля:

– Назва аудіо-фрагмента;

– Дата запису;

– Хеш-значення аудіо-фрагмента;

– ID-активу в Blockchain.

Таблиця метаданих містить наступні поля:

– Хеш-значення аудіо-фрагмента, до якого прикріплені метадані;

– Текст (тіло метаданих);

– Хеш-значення тіла метаданих;

– ID-активу в Blockchain.

Бібліотека hashlib використовується для виклику конструктора алгоритму генерації хеш-значення SHA256.

У Blockchain зберігається інформація про зареєстровані аудіо-фрагменти, прикріплені метадані, проведені перевірки на маніпуляції і загрузки аудіо-фрагментів. Доступ до Blockchain здійснюється за допомогою JSON-RPC команд з коду серверної частини системи.

3.2 Діаграми діяльності і послідовності

Діаграма діяльності відображає послідовність дій, необхідних для досягнення певної мети, тобто показує специфіку поведінки системи в певній ситуації. Діаграми діяльності, розроблені для моделювання процесу виконання операцій в системах (для обох сценаріїв), знаходяться в додатку Б.

Як приклад буде розглянута діаграма, що описує процес реєстрації користувача.

На рисунку 3.3 зображено процес реєстрації користувача в системі.

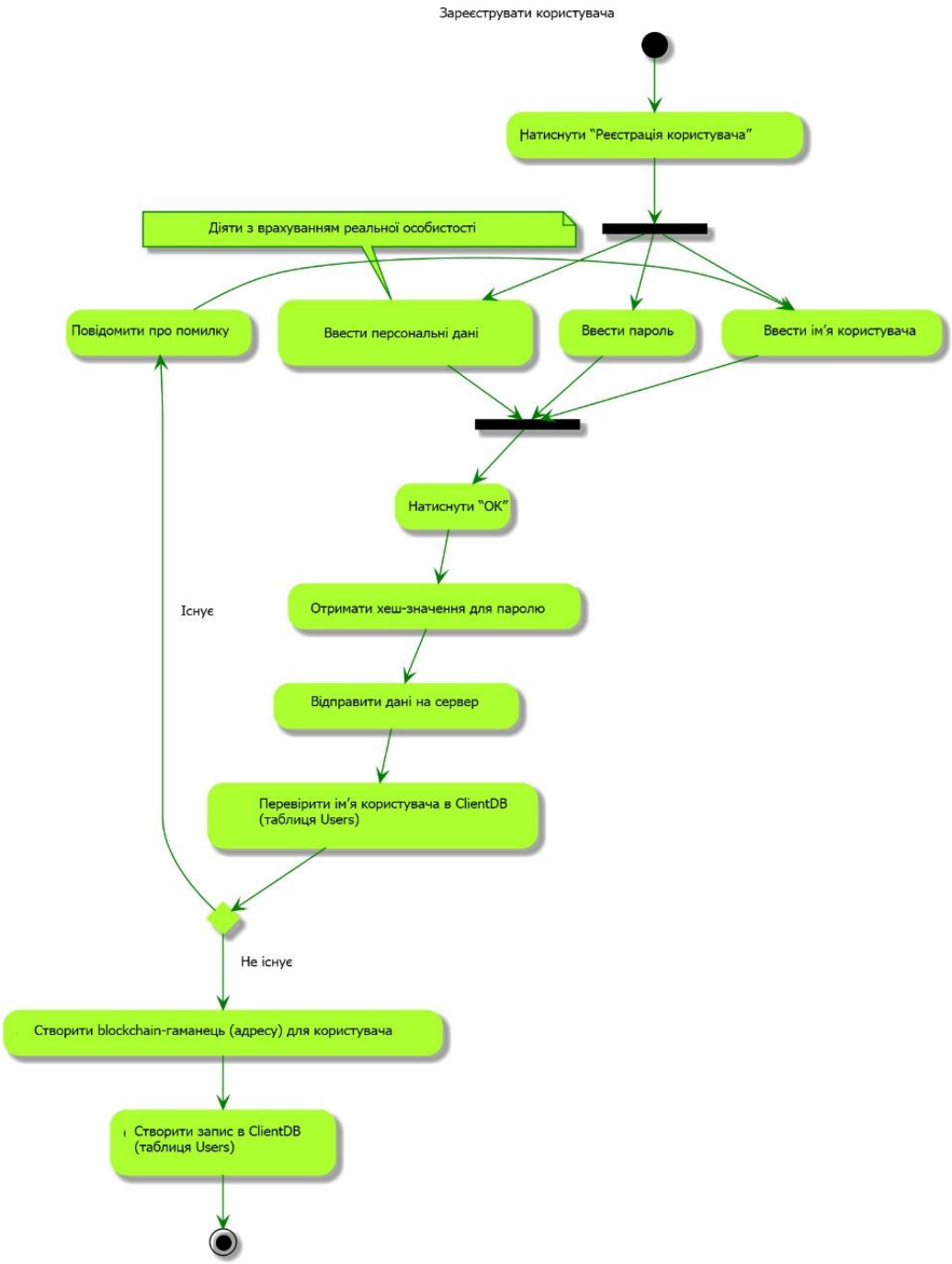


Рисунок 3.3 – Реєстрація користувача (діаграма діяльності)

Для реєстрації користувач вводить бажане ім'я користувача, пароль і особисті дані користувача (за бажанням). Для безпечної передачі і зберігання пароля формується його хеш-значення. Далі проводиться перевірка доступності імені користувача (тобто ведеться пошук такого імені користувача серед вже існуючих імен в базі даних). Якщо користувач з таким ім'ям вже зареєстрований, пропонується ввести інше ім'я користувача. Потім генерується blockchain-адреса для користувача, створюється запис в базі даних користувачів.

Процес аутентифікації користувача наведена на рисунку 3.4.

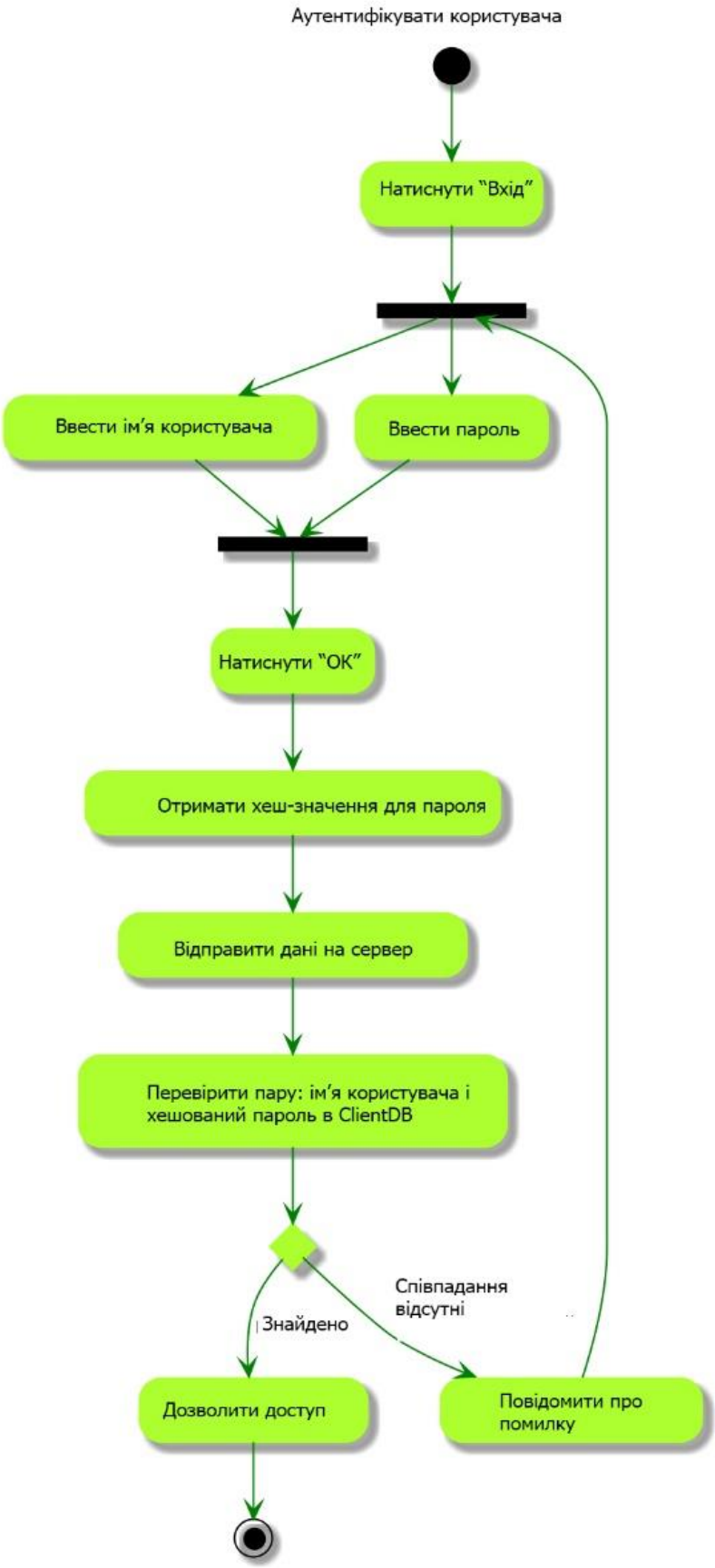


Рисунок 3.4 – Аутентифікація користувача (діаграма діяльності)

На рисунку 3.5 зображено процес реєстрації аудіо фрагмента.

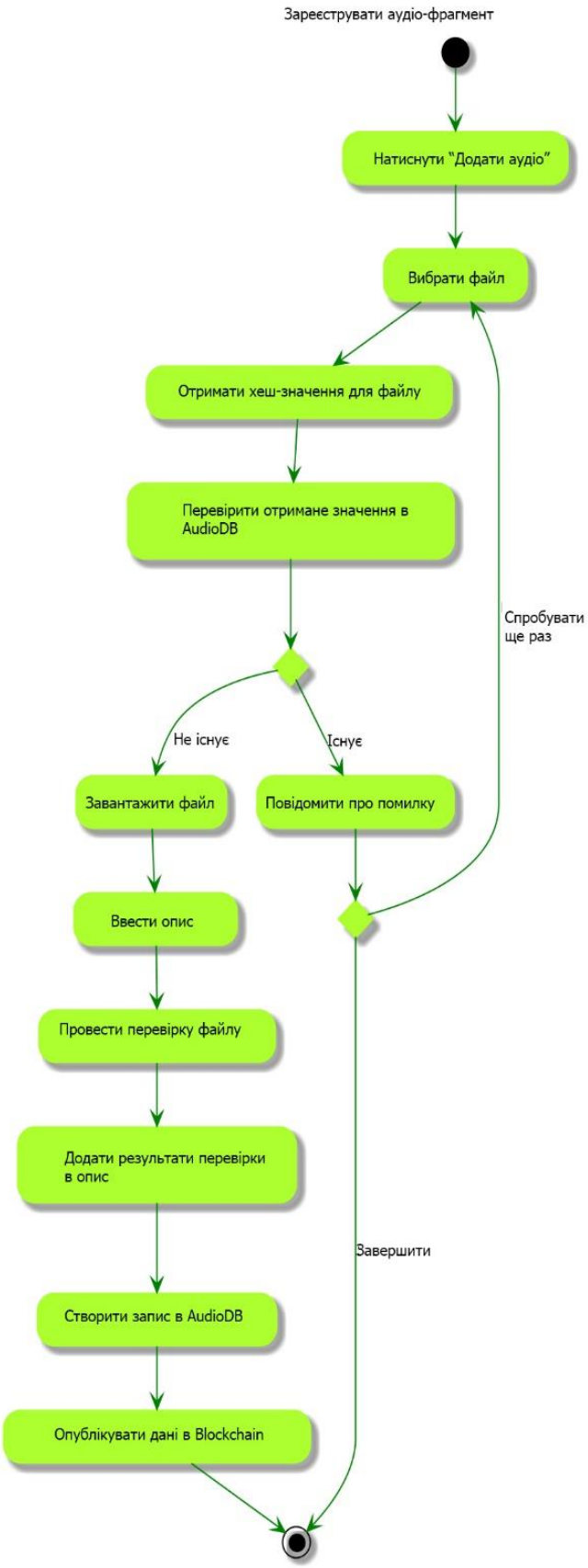


Рисунок 3.5 – Реєстрація аудіофрагмента (діаграма діяльності)

На рисунку 3.6 зображено процес завантаження аудіофрагмента.

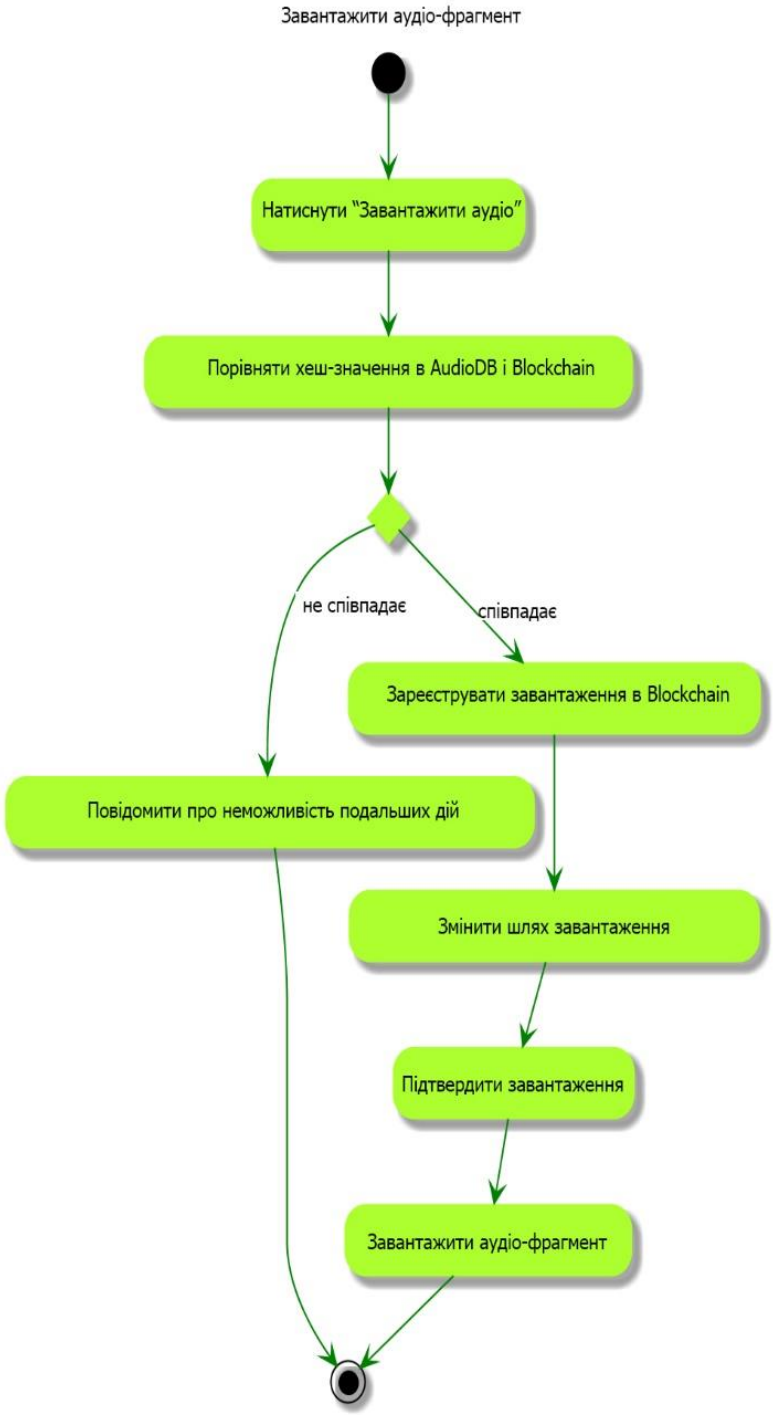


Рисунок 3.6 – Завантаження аудіо фрагмента (діаграма діяльності)

Діаграми послідовності призначаються для моделювання взаємодії об'єктів системи (в часі) і обміну повідомленнями між об'єктами. Дані діаграми показують не тільки алгоритм дій, а й унаочнюють «спілкування» елементів системи між собою. Як приклад розглядається діаграма, що ілюструє процес

реєстрації користувача в системі (рисунок 3.7).

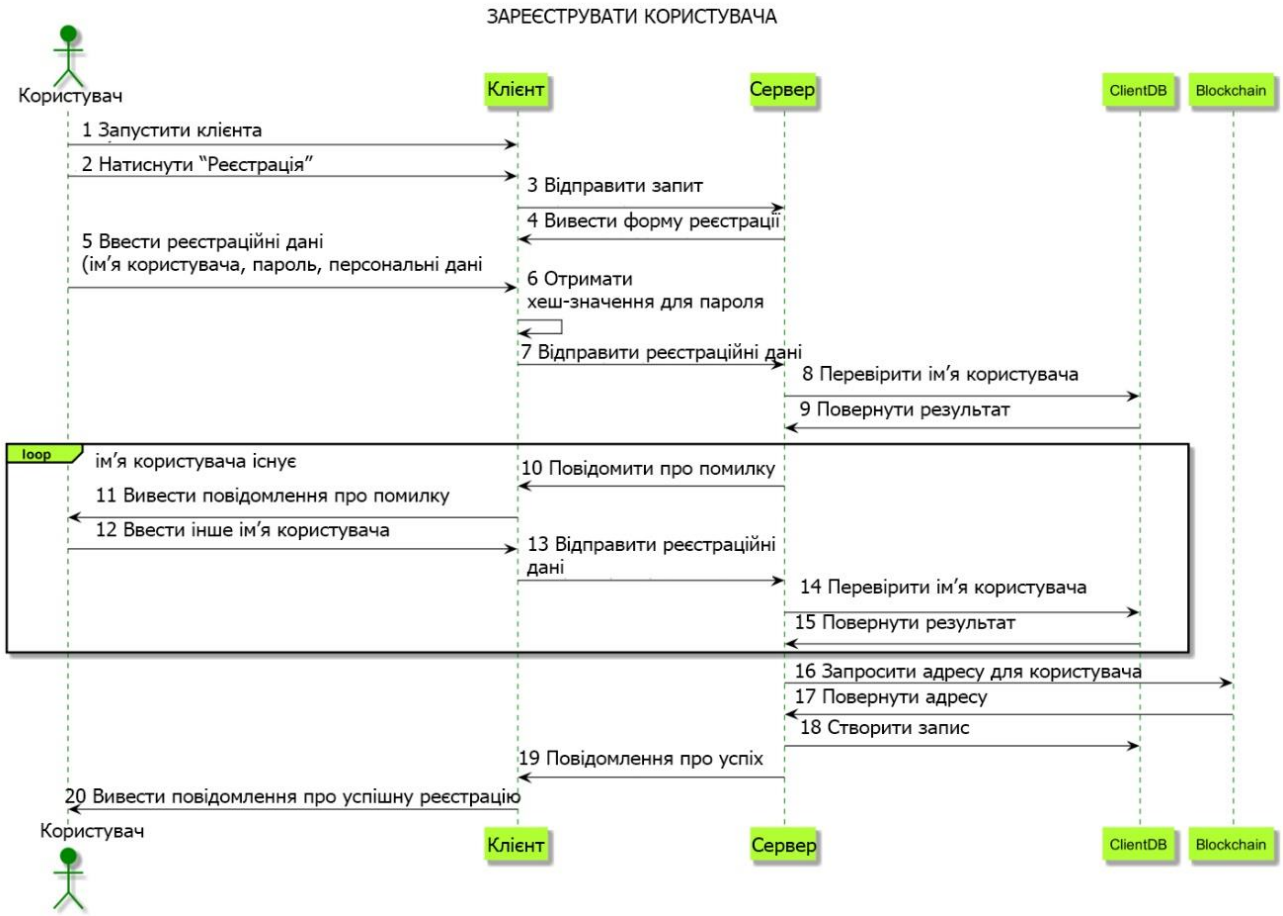


Рисунок 3.7 – Реєстрація користувача (діаграма послідовності)

Користувач запускає клієнта, натискає на кнопку реєстрації. Клієнт відправляє запит реєстрації на сервер, виводиться форма реєстрації. Користувач заповнює поля форми (ім'я користувача, пароль, особисті дані (опціонально)). Генерується хеш-значення для безпечної передачі і зберігання пароля. Реєстраційні дані передаються на сервер. Далі надсилається запит до бази даних на перевірку імені користувача. Якщо користувач з таким ім'ям вже зареєстрований в системі, результат пошуку буде не порожнім, сервер відправить клієнту повідомлення про недоступність імені користувача, клієнт виведе повідомлення для користувача, після чого користувач зможе ввести інше ім'я і процес буде виконаний заново. Далі для користувача генерується blockchain-адреса, створюється запис в базі даних користувачів. Сервер повідомляє клієнту про успішну реєстрацію користувача. Клієнт виводить

повідомлення для користувача.

Діаграми послідовностіі, розроблені для моделювання процесу виконання операцій в системах (для обох сценаріїв), знаходяться в додатку В.

3.3 Розробка функціональної частини сервісу

В рамках даного етапу роботи були розроблені прототипи частини функціоналу системи для сценарію «Музика і метадані». Розробка здійснювалася на мові програмування Python з використанням технології Multichain, бази даних SQLite, бібліотеки для шифрування hashlib, «оболонки» для виклику JSON-RPC команд з коду програми Savoir. Для тестування клієнт-серверної взаємодії була написана тестова клієнтська і серверна частина системи.

Для перевірки роботи функцій була створена приватна Blockchain, тестові бази даних для зберігання інформації про клієнтів (‘clients.db’) і записів (‘music.db’).

Для доступу до Blockchain необхідно в серверній частині системи створити інтерфейс за допомогою бібліотеки Savoir так як показано у лістингу 3.1.

Лістинг 3.1 – Створення інтерфейсу на стороні сервера

```
# Multichain-дані з multichain.conf
rpcuser = ‘multichainrpc’
rpcpasswd = ‘A9gSx6vxLuqHM84axcX5mwq69MB3e3h7iQrSdQ9ECQUd’
rpchost = ‘10 .129.8.165 ‘
rpcport = ‘7220’ chainname = ‘tchain’
# Створення інтерфейсу для API-команд
api = Savoir (rpcuser, rpcpasswd, rpchost, rpcport, chainname)
```

Після цього можна використовувати створений інтерфейс для того, щоб відправляти команди Blockchain.

Щоб забезпечити безпеку передачі пароля, а також для реєстрації і аутентифікації користувачів, правовласників і записів, використовується хешування.

Для доступу до баз даних необхідно створити на серверній частині системи

підключення до бази даних. В даному прикладі (лістинг 3.2) представлено створення підключення і курсора для бази даних клієнтів, підключення і курсор для бази даних із записами створюється аналогічно.

Лістинг 3.2 – Створення підключення і курсора для бази даних клієнтів

```
# Створення підключення до бази даних
conn_client = sqlite3.connect ( 'client.db')
# створення курсора для підключення
cur_client = conn_client.cursor ()
```

Для встановлення зв'язку між клієнтом і сервером були використані сокети. Приклад створення сокета клієнта показано у лістингу 3.3.

Лістинг 3.3 – Створення сокета клієнта

```
# Дані з'єднання
HOST = 'localhost'
PORT = 9090
# Створення сокета клієнта
sock = socket.socket ()
# підключення до сервера
sock.connect ((HOST, PORT))
```

Приклад створення сокета сервера показано у лістингу 3.4.

Лістинг 3.4 – Створення сокета сервера

```
# Дані з'єднання
HOST = ''
PORT = 9090
# Створення сокета сервера
sock = socket.socket ()
# Зв'язування сокета з хостом і портом
sock.bind ((HOST, PORT))
# Визначення обсягу черги вхідних підключень
sock.listen (10)
     while True:
     # Обробляти вхідні підключення
     conn, addr = sock.accept ()
     print ( 'Connected by', addr) while True:
     # Поки клієнт не закрив з'єднання
     # отримувати дані від клієнта
data = sock.recv (1024)
```

Взаємодія між клієнтом і сервером відбувається шляхом відправки команд і параметрів від клієнта серверу і відповідей сервера клієнту. Формування запиту від клієнта і отримання відповіді від сервера відбувається наступним так як показано у лістингу 3.5.

Лістинг 3.5 – Формування запиту від клієнта і отримання відповіді від сервера

```
    # Формування словника з назвою викликається методу і переданими параметрами і переклад його в рядок
    str_json = json.dumps ({ 'method': <назва методу "," ім'я параметра>: <значення параметра>})
    # Новий рядок в байти
    data = str_json.encode ()
    # відправка даних на сервер
    sock.send (data)
    # Отримання відповіді від сервера
    data = sock.recv (1024)
    # Переклад отриманих байт в рядок
    data = data.decode ()
```

Обробка запиту клієнта і формування відповіді клієнтові відбувається наступним чином (лістинг 3.6).

Лістинг 3.6 – Обробка запиту клієнта і формування відповіді клієнтові

```
    # Отримання даних від клієнта
    data = sock.recv (1024)
    # Переклад отриманих байт в рядок
    data = data.decode ()
    # Новий рядок в словник
    dict_json = json.loads (data)
    # Отримати назву методу і параметри зі словника і викликати відповідну функцію
    # Формування відповіді користувачеві в залежності від результатів виконання функції
    # Відправка відповіді користувачу
    conn.send (answer)
```

У прототипі були реалізовані функції реєстрації користувача і правовласника, аутентифікації користувача і правовласника, реєстрації і аутентифікації записів. Як приклад написання коду системи буде розглянуто код функції реєстрації користувача для клієнтської і серверної частин системи.

Приклад реалізації функції реєстрації користувача представлено у лістингу 3.7.

Лістинг 3.7 – Реалізація функції реєстрації користувача

```
# На стороні клієнта
username = str (input ( ‘Enter username:’))
# введення імені користувача
# Формування і відправка запиту на перевірку доступності імені користувача на сервер
# Отримання відповіді сервера
# обробка відповіді сервера
password = string (input ( ‘Enter password:’))
# введення пароля
pass_hash = sha256.update (password)
# хешування пароля
while True:
# запит на введення персональних даних
answer = input ( ‘Do you want to enter your personal data? [y / n] \ n’)if answer == ‘y’:
# введення персональних даних
first_name = string (input ( ‘Enter your firstname: \ n ‘))
name: \ n ‘))
last_name = string (input ( ‘Enter your last
birthday = input ( ‘Enter your birthday DD.MM.YYYY): \ n ‘)
break
elif answer == ‘n’: # відмову від введення персональних даних first_name = None
last_name = None birthday = None break
# Формування і відправка запиту на сервер
# отримання відповіді від сервера
if data == ‘Fail’:
# якщо реєстрація не виконана
return False
else:
# якщо реєстрація виконана print (data)
# print blockchain address return True
# На стороні сервера
# Перед реєстрацією здійснюється перевірка імені користувача
# Отримання даних від клієнта
# Створення з’єднання з базою даних і курсора cur_client.execute ( "SELECT * FROM users WHERE
username =? ", username)
# пошук імені користувача в базі
check = cur_client.fetchone ()
# отримання рядки результатів
conn_client.close ()
# закриття з’єднання
if check! = None: # якщо ім’я користувача вже існує return
False
else:
return True
```

Продовження лістингу 3.7

```
    # Продовження реєстрації при успішному проходженні перевірки
    # отримання інших реєстраційних даних від клієнта
    address = api.getnewaddress ()
    # отримання blockchain-адреси
    для користувача
    insert = (username, pass_hash, address, first_name, last_name,
birthday) # формування набору даних для занесення в базу даних
    # Підключення до бази даних, створення курсора
    cur_client.execute ( "INSERT INTO users
    VALUES (?,?,?,?,?,?) ", Insert)
    # додавання даних користувачів в базу
    conn_client.commit ()
    # підтвердження додавання записи
    conn_client.close ()
    # закриття з'єднання
```

У лістингу 3.8 наведено фрагмент коду функції сервера

Лістинг 3.8 – Функції сервера (фрагмент)

```
    #
    # Account Management
    #
    def check_username(username, type):
    conn_client = sqlite3.connect('client.db') # connect to data
base
    with user's and rights holder's data
    cur_client = conn_client.cursor() # create cursor for
connection
    to data base with user's and rights holder's data
    if type == 'u':
    cur_client.execute("SELECT * FROM users WHERE username=?",
    username) # search for username in data base
    elif type == 'rh':
    cur_client.execute("SELECT * FROM rights_holders WHERE
    username=?", username) # search for username in data base
    check = cur_client.fetchone() # get result row
    conn_client.close()
    if check != None: # username already exists
    return False
    else:
    return True
    def reg_user (username, pass_hash, first_name, last_name,
birthday)
    address = api.getnewaddress() # get multichain address for
user
    insert = (username, pass_hash, address, first_name, last_name,
    birthday)
    try:
    conn_client = sqlite3.connect('client.db') # connect to data
    base with user's and rights holder's data
```

Подальші приклади реалізації функцій клієнта та сервера знаходяться в додатках Е та Ж відповідно.

3.4 Висновки до третього розділу

У третього розділі дипломної роботи побудовано та описано діаграми варіантів використання, компонентів, діяльності та послідовності. Дані діаграми в різних аспектах ілюструють склад і поведінку розроблюваних програмних елементів сервісу. Також розроблено функціональну частину сервісу, наведено окремі лістинги програмного коду.

4 СПЕЦІАЛЬНА ЧАСТИНА

4.1 Мова програмування Python

Python – високорівнева мова програмування, найчастіше використовується в якості скрипт-мови разом з відповідним інтерпретатором, що робить виконання коду трохи повільніше, ніж у компільованих мов, однак спрощує налагодження [10]. Головна мета дизайну Python – легко читається вихідний код і підвищення продуктивності програміста. Python має простий синтаксис і використовує відступи в якості головного структуруючого елементу. Незважаючи на деяку мінімалістичність синтаксису ядра, Python є дуже гнучким і широко застосовним за рахунок стандартної бібліотеки, що включає в себе широкий набір корисних функцій. Крім того, дана мова програмування підтримує кілька парадигм програмування (об’єктно і аспектно-орієнтоване, структурне, функціональне та імперативне).

Python проста у використанні, та водночас повноцінна мова програмування, що надає набагато більше засобів для структурування і підтримки великих програм, ніж shell. З іншого боку, вона краще за C обробляє помилки, і, будучи мовою дуже високого рівня, має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники, ефективна реалізація яких на C потребує значних витрат часу.

Завдяки більш загальним типам даних, Python застосовують до більш широкого кола задач, ніж Awk і навіть Perl, у той ж час багато речей на мові Python робляться настільки ж просто.

Python дозволяє розбивати програми на модулі, що потім можуть бути використані в інших програмах. Python поставляється з великою бібліотекою стандартних модулів, які можна використовувати як основу для нових програм або як приклади при вивченні мови. Стандартні модулі надають засоби для роботи з файлами, системними викликами, мережними з'єднаннями і навіть інтерфейсами до різних графічних бібліотек.

Python - інтерпретована мова, що дозволяє заощадити значну кількість

часу, що зазвичай витрачається на компіляцію. Інтерпретатор можна використовувати інтерактивно, що дозволяє експериментувати з можливостями мови, писати шаблони програм або тестувати функції при розробці “знизу-вверх”. Він також зручний як настільний калькулятор. Python дозволяє писати дуже компактні й зручні для читання програми. Програми, написані мовою Python, звичайно значно коротші еквівалента на C або C++ з декількох причин:

– типи даних високого рівня дозволять виразити складні операції однією інструкцією;

– групування інструкцій виконується за допомогою відступів замість фігурних дужок;

– немає необхідності в оголошенні змінних.

Python розширювана мова [11]: знання C дозволяє додавати нові функції, що вбудовуються, або модулі для виконання критичних операцій з максимальною швидкістю або написання інтерфейсу до комерційних бібліотек, доступним тільки у двійковій формі. Інтерпретатор мови Python може бути вбудований у програму, написану на C, і використовувати його як розширення або командну мову для цієї програми. Python використовується в даний час десятками тисяч програмістів в усьому світі, і число людей, що використовують його, швидко зростає, подвоюється і потроюється щороку. Python приваблює користувачів з ряду причин. Він використовується для розробки програм і дозволяє провести розробку набагато швидше, ніж традиційні мови типу C, C++ або Java . Ця мова працює однаково добре на Windows, UNIX, Macintosh, і OS/2, може використовуватися, для легкої розробки як малих додатків чи сценаріїв, так і для розгортання великих програм. Python пропонує доступ до могутнього і легкого у використанні комплекту 29 інструментальних засобів графічного інтерфейсу користувача. Традиційні машинні мови типу C і Pascal мають ряд характеристик, наприклад, сувора типізація, базові типи, складні (і звичайно довгі) цикли, і потреба у великих кількостях кодів для виконання відносно малих задач. Java досить новий, але розділяє більшість характеристик, включених у цей перелік. Програмісти, знайомі з традиційними мовами погодяться, що відсутність суворої типізації полегшує роботу з Python.

Архітектурними особливостями цієї мови є:

– динамічна типізація (не потрібно визначати тип змінної в момент оголошення, змінна зв’язується з ним при присвоєнні їй значення);

– повна інтроспекція (тип і структура об’єкта може бути визначена безпосередньо під час виконання програми);

– автоматичне управлення пам’яттю (збір сміття, відсутність витоків);

– механізм обробки виключень;

– підтримка багатопотокових обчислень.

Крім цього, у Python є ще деякі приємні особливості, такі як інтеграція C/C++, вбудована підтримка Unicode в рядках, кросплатформність (відмінності у функціонуванні програм можуть виникнути лише в рідкісних випадках, але їх легко передбачити завдяки наявності докладної документації).

Основні відмінності Python від інших мов програмування:

– керування пам'яттю тут цілком автоматичнее, тобто не потрібно хвилюватися щодо розподілу або звільнення пам'яті. Немає загрози “небезпечного посилання”. Java - єдина мова, що пропонує таку концепцію;

– типи зв'язані з об'єктами, а не зі змінними. Це означає, що змінній може бути призначене значення будь-якого типу, і що (наприклад) масив може містити об'єкти різних типів. Традиційні мови не надають такої можливості;

– операції звичайно виконуються в більш високому рівні абстракції. Це частково результат того, як написана мова, і частково результат розширеної стандартної бібліотеки кодів, що поставляється разом з Python.

Ці та інші особливості Python роблять розгортання додатків надзвичайно швидким. Продуктивність створеного додатку залежить від його особливостей. Звичайно, для чисельного алгоритму, що виконує звичайну арифметику цілого числа в циклі 'for', неважливо, на якій мові він написаний. Але для “середнього” додатка, збільшення продуктивності може бути просто дивовижним. Один недолік Python, у порівнянні з найбільш традиційними мовами, полягає в тому, що це - не цілком компільована мова; замість цього, вона частково транслює програму до внутрішньої форми байт-коду, і цей байт-код виконується

інтерпретатором Python. Однак, у перспективі – сучасні комп'ютери мають так багато обчислювального потенціалу, який не використовується, що для 90% додатків швидкодія зв'язана з вибором мови. Java теж компілюється в байт-код, але в даний час працює повільніше ніж Python у більшості випадків. Крім того, дуже просто об'єднати Python з модулями, написаними на C або C++, які можна використовувати, щоб збільшити швидкість роботи програм в критичних ділянках.

Вибір на користь даної мови програмування для написання прототипу функціоналу системи був зроблений через гнучкість, відносну простоту вивчення та можливості використання для вирішення різних завдань, а також наявності безлічі бібліотек, що дозволяють працювати з іншими технологіями, задіяними в проекті [11].

4.2 Система управління базами даних SQLite

SQLite – це вбудована в процес бібліотека, яка реалізує автономний, безсерверний, що не вимагає спеціальної попередньої настройки, транзакційний SQL-движок. Код SQLite переданий в суспільне надбання і, отже, може бути безоплатно використаний з будь-якою метою, як комерційною, так і особистою. SQLite –широко поширена база даних в світі, яка використовується в незліченній кількості додатків, включаючи кілька гучних проектів (Google Chrome, Safari, Garena, Adobe Photoshop Lightroom і ін.). [12]

Слово «вбудовується» в даному контексті означає, що SQLite не використовує парадигму клієнт-сервер і, отже, не має окремого серверного процесу (і взагалі кажучи, не є окремим працюючим процесом, а надає програмі бібліотеку, з якої та компонується, і движок бази даних стає частиною програми). SQL-база даних цілком зберігається в одному файлі (всі таблиці, індекси і т.д.). Формат файлу бази даних є кросплатформним, база може бути безперешкодно скопійована або перенесена, наприклад, з 32-бітної системи на 64-бітну. Ця особливість робить SQLite широко поширеним вибором в якості формату файлу програми (файл, який використовується для збереження стану

програми на диску або для обміну інформацією між додатками).

SQLite добре показує себе в таких випадках використання:

– вбудовуване обладнання та Інтернет речей;
– формат файлу програми;
– веб-сайти;
– аналіз даних;
– кеш для корпоративних даних;
– база даних на сервері;
– файлові архіви;
– заміна ad hoc файлів на диску;
– внутрішні або тимчасові бази даних;
– заміна корпоративної бази даних в демо-версії і при тестуванні;
– освіту і навчання;
– експериментальні розширення мови SQL.

Читати дані з бази можуть кілька процесів або потоків одночасно, але запис можна здійснити лише за умови, що ніяких інших запитів в даний момент не обслуговується. Інакше спроба запису буде невдалою, в програму буде повернуто код помилки. Як альтернативний варіант, можна повторювати спроби записи протягом заданого проміжку часу.

Бібліотека SQLite дуже компактна – її розмір з повним набором функцій, в залежності від цільової платформи і налаштувань оптимізації компілятора, може бути менше 500 Кб. Якщо опціональні функції опущені, розмір бібліотеки SQLite може бути зменшений до менш ніж 300 Кб. SQLite може бути адаптована для роботи в мінімальному просторі стека (4 Кб) і в дуже маленькій «купі» (heap, 100 Кб), що робить її популярним движком БД для пристроїв з обмеженими можливостями пам'яті, такими як телефони і MP3-плеєри. Існує компроміс між використанням пам'яті і швидкодією. SQLite зазвичай працює тим швидше, чим більше пам'яті їй виділяється. Проте, продуктивність, як правило, досить хороша навіть в середовищах з низькою пам'яттю.

Сама бібліотека написана на мові C, але існує безліч прив'язок до інших

мов програмування (C++, Java, Python, Perl і багатьох інших, повний список розміщений на інтернет-сторінці проекту).

SQLite, за твердженням розробників, дуже ретельно тестується перед кожним новим випуском і має репутацію надійного рішення. Автоматизований набір тестів реалізує величезну кількість тестових сценаріїв, що включають в себе мільйони індивідуальних SQL-запитів, і досягає 100% покриття гілок тестами. SQLite грамотно реагує на помилки виділення пам'яті і помилки введення-виведення. Транзакції зберігають ACID-властивості, навіть якщо вони були перервані через збої системи або харчування, що підтверджено тестуванням з імітацією збоїв системи. Якщо при експлуатації все ж виявляються помилки, вони публікуються в списках помилок.

Код SQLite підтримується інтернаціональною командою розробників, які продовжують розширювати можливості і підвищувати надійність і продуктивність даного продукту. Хоча вихідний код постачається вільно, професійна підтримка також доступна. [12]

4.3 Висновки до четвертого розділу

У четвертому розділі дипломної роботи наведено основні можливості програмного забезпечення, яке використовується для створення сервісу, зокрема мови програмування Python та системи керування базами даних SQLite.

5 ОБҐРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Метою дипломної роботи є створення сервісу, який призначений для управління механізмом авторських прав на мультимедійні файли. Головною метою розділу є обґрунтування економічної ефективності впровадженої даної розробки.

Щоб виконати оцінку економічної ефективності необхідно розрахувати трудомісткість реалізації проекту, витрати на оплату праці найманим працівникам, витрати апаратного і програмного забезпечення, амортизаційні відрахування, витрати енергоресурсів та інші витрати які є основними пунктами виконання обчислень, а також показники економічної ефективності розробки проекту.

5.1 Розрахунок норм часу на виконання науково-дослідної роботи

Реалізація проекту створення сервісу, який призначений для управління механізмом авторських прав на мультимедійні файли складається з низки послідовних та взаємопов’язаних етапів.

Кожен із етапів реалізації проекту характеризується метою та змістом, оцінкою часу виконання, кількістю та спеціалізацією виконавців, а також приблизною оцінкою вартості.

Реалізація проекту створення сервісу складається із підготовчого етапу, етапу технічної пропозиції, створення технічного завдання, проектування системи, практичної реалізації, тестування, верифікації та заключного етапу.

Норми часу на виконання науково-дослідницької роботи розраховуватимуться на основі середнього часу виконання стадії в годинах, що наведені в таблиці 5.1 разом із інформацією про виконавців і сумарною кількості затраченого часу.

Таблиця 5.1 – Операції технологічного процесу та їх час виконання

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	Підготовча стадія	Проектний менеджер	10
		Інженер-програміст	
2	Технічна пропозиція	Проектний менеджер	10
		Інженер-програміст	
3	Створення технічного завдання	Проектний менеджер	20
		Інженер-програміст	
4	Проектування системи	Інженер-програміст	20
5	Практична реалізація	Інженер-програміст	135
6	Тестування системи	Тестувальник	20
7	Верифікація системи	Тестувальник	20
		Інженер-програміст	
		Проектний менеджер	
8	Створення документації	Інженер-програміст	20
9	Заключна стадія	Проектний менеджер	10
Разом			265

В підсумку на реалізацію проекту сервісу, який призначений для управління механізмом авторських прав на мультимедійні файли необхідно 265 людино-годин, залучення трьох спеціалістів та виконання дев'яти різноманітних стадій реалізації проекту.

5.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

Визначення витрат на оплату праці та відрахувань на соціальні заходи прямо залежить від кількості витраченого працівниками часу на роботу, ставки в

годину чи місяць, кількість відрахувань на соціальні заходи встановлених в законному порядку на час розрахунку.

В результаті розрахунку потрібно визначити основну та додаткову заробітну плату, витрати на соціальні заходи та на основі цих даних визначити сумарні витрати на оплату праці. Основна заробітна плата нараховується за виконану роботу за тарифними ставками, відрядними розцінками чи посадовими окладами. Додаткова заробітна плата – це складова заробітної плати працівників, до якої включають витрати на оплату праці, не пов'язані з виплатами за фактично відпрацьований час.

При розрахунку заробітної плати кількість робочих днів у місяці слід в середньому приймати – 24,5 дні/міс., або ж 196 год./міс. (тривалість робочого дня – 8 год.).

Наймані працівники для розробки сервісу, який призначений для управління механізмом авторських прав на мультимедійні файли працюють згідно контракту, який в якому вказано їхню погодинну ставку. Тобто розрахунок заробітної плати працівників відбуватиметься на базі тарифної ставки та кількості відпрацьованих годин.

У штаті найманих працівників для розробки сервісу залучено проектного менеджера, інженера-програміста і тестувальника.

Тарифні ставки учасників процесу розробки сервісу, який призначений для управління механізмом авторських прав на мультимедійні файли:

– Проектний менеджер – 150 грн./год.

– Інженер-програміст – 130 грн./год.

– Тестувальник – 100 грн./год.

Основна заробітна плата розраховується за формулою 5.1:

$$З_{осн.} = Т_с \cdot К_г , \qquad (5.1)$$

де $Т_с$ – тарифна ставка, грн.; $К_г$ – кількість відпрацьованих годин.

Оскільки всі види робіт в виконує три спеціаліста, то основна заробітна плата буде розраховуватись за даною формулою 5.1;

$$З_{осн.} = 150 \cdot 35 + 130 \cdot 200 + 100 \cdot 30 = 34250 \text{ грн.}$$

Додаткова заробітна плата становить 10–15 % від суми основної заробітної плати й визначається за формулою 5.2.

Коефіцієнт додаткових виплат працівникам становить 0,1.

$$Здод. = Зосн. \cdot Кдопл., \quad (5.2)$$

де $К_{допл}$ – коефіцієнт додаткових виплат працівникам

$$З_{дод} = 34250 \cdot 0,1 = 3425 \text{ грн.}$$

Звідси загальні витрати на оплату праці (фонд заробітної плати) визначаються за формулою 5.3:

$$Во.п. = Зосн. + Здод. \quad (5.3)$$

$$В_{о.п.} = 34250 + 3425 = 37675 \text{ грн.}$$

З цієї суми утримуються обов'язкові відрахування на заробітну плату:

– Єдиний соціальний внесок (ЄСВ), що становить 22%;

– Військовий збір (ВЗ), що становить 1,5%;

Сума відрахувань становить 23,5% від фонду оплати праці та визначається за формулою 5.4:

$$В_{с.з.} = Ф_{оп} \cdot 0,235 \quad (5.4)$$

де $Ф_{оп}$ – фонд оплати праці, грн.

$$B_{с.з.} = 37675 \cdot 0,235 = 8853,62$$

Усі витрати обчислюються детально наведені в таблиці 5.2 та обчислюються за формулою 5.5:

$$Взп = ФЗП + ФОП \quad (5.5)$$

$$В_{зп} = 34250 + 8853,62 = 43103,62 \text{ грн.}$$

Таблиця 5.2 – Розрахунки витрат на оплату праці

з/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	Нарахув. на ФОП, грн.	Всього витрати на плату праці, грн. (6=3+4+5)
		Тарифна ставка, грн.	Кількість відпрацьова них год.	Фактично нарах. з/пл., грн.			
А	Б	1	2	3	4	5	6
1.	Проектний менеджер	150	35	5250	525	-	-
2.	Інженер-програміст	130	200	26000	2600	-	-
3.	Тестувальник	100	30	3000	300	-	-
Разом		380	265	34250	3425	8853,62	43103,62

Опираючись на розрахунки витрат на оплату та таблицю результатів 5.2 видно, що всього витрати на плату праці становлять 43103,62 грн.

5.3 Розрахунок матеріальних витрат

Матеріальні витрати є невід'ємною частиною розробки сервісу, який призначений для управління механізмом авторських прав на мультимедійні файли та визначаються як добуток кількості витрачених матеріалів та їх ціни за формулою 5.6:

$$M_{вi} = q_i \cdot p_i \,, \tag{5.6}$$

де: q_i – кількість витраченого матеріалу і–го виду; p_i – ціна матеріалу і–го виду.

Звідси, загальні матеріальні витрати можна визначити за формулою 5.7:

$$З_{м.в.} = \sum M_{вi} \,. \tag{5.7}$$

Результати проведених розрахунків наведено у таблиці 5.3.

Таблиця 5.3 – Результати розрахунків матеріальних витрат.

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Фактично витрачено матеріалів	Ціна одиниці, грн.	Загальна сума витрат, грн.
1	CD диски	шт.	2	7,45	14,90
2	Папір для друку	листів	500	0,15	75,00
3	Чорнила для принтера	шт.	1	80,00	80,00
Всього					169,90

Згідно проведених розрахунків, матеріальні витрати становлять 169,90 грн.

5.4 Розрахунок витрат на електроенергію

Однією із статей витрат є витрати на електроенергію під час проходження усіх етапів реалізації кінцевого продукту.

Затрати на електроенергію одиниці обладнання визначаються за формулою 5.8:

$$З_в = W \cdot T \cdot S, \tag{5.8}$$

де W – необхідна потужність, кВт; T – кількість годин на реалізацію розробки; S – вартість кіловат-години електроенергії.

Вартість кіловат-години електроенергії слід приймати згідно існуючих на даний час тарифів. Отже, 1 кВт з ПДВ коштує 2,42 грн.

Потужність комп'ютерів для реалізації кінцевого продукту – 400 Вт, кількість годин роботи обладнання згідно таблиці 5.1 – 265 годин.

Визначимо витрати на електроенергію згідно формули 5.11:

$$З_в = 0{,}4 \cdot 265 \cdot 2{,}42 = 256{,}52 \text{ грн.}$$

Згідно формули затрати на електроенергію становлять 256,52 грн.

5.5 Розрахунок суми амортизаційних відрахувань

Для будь якої діяльності характерною є властивість зношування на зниження якості властивостей інструментарію та фондів за допомогою яких ведеться діяльність. Для вирішення проблеми із відновленням даних фондів використовується амортизація, що являє собою процес трансформації вартості основних фондів на вартість продукції, яка щойно була створена, задля повного відновлення основних фондів.

Для визначення амортизаційних відрахувань використовується формула 5.9:

$$A = \frac{Б_В \cdot Н_А}{100\%} \qquad (5.9)$$

де A – амортизаційні відрахування за звітний період, грн.; $Б_В$ – балансова вартість групи основних фондів на початок звітного періоду, грн.; $Н_А$ – норма амортизації.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Річний робочий фонд становитиме 2352 годин, так як робочий день становить 8 годин, а кількість робочих днів в місяці становить 24,5 годин.

Для даної розробки засобом розробки є комп'ютер. Його сума становить 18500 грн. Отже, амортизаційні відрахування будуть рівні:

$$A = 18500 \cdot 5\% / 100\% = 925 \text{ грн.}$$

Згідно проведених обчислень амортизаційні відрахування становлять 925 грн.

5.6 Обчислення накладних витрат

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління спілкою та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$H_в = B_{о.п.} \cdot 0{,}2 \ldots 0{,}6 \ , \qquad (5.10)$$

де $H_в$ – накладні витрати.

Отже, накладні витрати становлять згідно формули 5.10:

$$H_в = 37675 \cdot 0{,}2 = 7535 \text{ грн.}$$

Накладні витрати згідно розрахунку формули, становить 7535 грн.

5.7 Складання кошторису витрат та визначення собівартості науково-дослідницької роботи

Результати проведених вище розрахунків наведено у таблиці 5.4.

Таблиця 5.4 – Кошторис витрат на НДР

Зміст витрат	Сума, грн.	В % до загальної суми
Витрати на оплату праці	43103,6	70,8
Відрахування на соціальні заходи	8853,6	14,6
Матеріальні витрати	169,9	0,3
Витрати на електроенергію	256,52	0,4
Амортизаційні відрахування	925	1,5
Накладні витрати	7535	12,4
Собівартість	60843,66	100

Собівартість ($С_в$) програмного продукту розрахуємо за формулою:

$$С_в = В_{о.п.} + В_{с.з.} + З_{м.в.} + З_в + А + Н_в . \quad (5.11)$$

Отже, собівартість програмного продукту дорівнює:

Св = 43103,6+ 8853,6+ 169,90 + 256,52+ 925+ 7535= 60843,66 грн.

Загальний кошторис витрат та визначення собівартості науково-дослідницької роботи становить 60843,66 грн.

5.8 Розрахунок ціни програмного продукту

Ціну науково-дослідної роботи можна визначити можна визначити за формулою:

$$Ц = \frac{С_В \cdot (1 + Р_{рен}) + К \cdot В_{н.і.}}{К} \cdot (1 + ПДВ) , \quad (5.12)$$

де $Р_{рен.}$ – рівень рентабельності (30 %); $К$ – кількість замовлень, од. (встановлюється лише при розробці програмного продукту та мікропроцесорних систем); $В_{н.і.}$ – вартість носія інформації, грн. (встановлюється лише при розробці програмного продукту); *ПДВ* – ставка податку на додану вартість (20%).

Оскільки розробка є прикладною, і використовуватиметься тільки для одного підприємства, то для розрахунку ціни не потрібно вказувати коефіцієнти К та $В_{н.і.}$, оскільки їх в даному випадку не потрібно.

Тоді, формула для обчислення ціни розробки буде мати вигляд:

$$Ц = C_B \cdot (1 + Р_{рен}) \cdot (1 + ПДВ) \tag{5.13}$$

Звідси ціна на роботу складе:

$$Ц = 60843{,}66 \cdot (1 + 0{,}3) \cdot (1 + 0{,}2) = 94916{,}11 \text{ грн.}$$

Загальний розрахунок ціни програмного продукту становить 94916,11 грн.

5.9 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність ($Е_р$) полягає у відношенні результату виробництва до затрачених ресурсів:

$$Е_р = \frac{П}{C_B}, \tag{5.14}$$

де $П$ – прибуток; $С_В$ – собівартість.

Плановий прибуток ($П_{пл}$) знаходимо за формулою:

$$П_{пл} = Ц - С_в . \quad (5.15)$$

Розраховуємо плановий прибуток:

$$П_{пл} = 94916{,}11 - 60843{,}66 = 34072{,}45 \text{ грн.}$$

Отже, формула для визначення економічної ефективності набуде вигляду:

$$E_р = \frac{П}{С_В}. \quad (5.16)$$

Тоді,

$$E_р = 34072{,}45 / 60843{,}66 = 0{,}56.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($Т_р$):

$$Т_р = \frac{1}{E_р} , \quad (5.17)$$

Термін окупності дорівнює:

$$Т_р = 1 / 0{,}5 = 1{,}78 \text{ р.}$$

Згідно формул плановий прибуток від розробки становить 34072,45 грн., економічна ефективність дорівнює 0,56, а термін окупності становить 1,78 роки що вважається доцільним та економічно вигідним.

5.10 Висновки до п’ятого розділу

В організаційно-економічній частині дипломної роботи освітнього рівня «магістр» було розраховано основні техніко-економічні показники створення сервісу, який призначений для управління механізмом авторських прав на мультимедійні файли (див. таблиця 5.5).

Орієнтоване значення економічної ефективності становить 0,56, що є достатньо високим значенням.

Період окупності повинен варіюватися від 1 до 3 років, тоді розвиток вважається доцільним та економічно вигідним. Термін окупності даної роботи становить 1,78 років.

Таблиця 5.5 – Техніко-економічні показники науково-дослідної роботи

№ п/п	Показник	Значення
1	Собівартість, грн.	60843,66
2	Плановий прибуток, грн.	34072,45
3	Ціна, грн.	94916,11
4	Економічна ефективність	0,56
5	Термін окупності, рік	1,78

На основі проведених обрахунків можна зробити висновок, що створення сервісу, який призначений для управління механізмом авторських прав на мультимедійні файли є доцільним у зв’язку з невеликим терміном окупності та великим обсягом планового прибутку.

6 ЕКОЛОГІЯ

6.1 Екологізація виробництв

Сучасний стан розвитку суспільних відносин з постійно зростаючим антропогенним впливом на навколишнє природне середовище вимагає зміни відношення людства до процесів виробництва та споживання товарів. Якщо раніше в основу будь-яких дій людини ставилися економічні пріоритети, то сьогодні на перший план виходять екологічні цілі. Іншими словами, на нинішньому етапі розвитку виробництва кожне рішення щодо освоєння нових або модернізації старих виробництв необхідно оцінювати з позиції зменшення негативного впливу на довкілля. Такий підхід називають екологізацією виробництва. Таким чином, екологізація – це зменшення інтегрального екодеструктивного впливу процесів виробництва та споживання товарів у розрахунку на одиницю сукупного суспільного продукту. Екологізація є більш широким поняттям ніж природоохоронна діяльність, оскільки її ціллю є не охорона природного середовища, а запобігання його пошкодженню через вилучення з виробництва і споживання природо-небезпечних товарів. [13]

Під екологізацією виробництва потрібно розуміти генерування ідей, формування інформаційних матеріалів, створення технічних засобів і технологічних рішень, що сприяють розвитку екологічно обумовлених виробничих систем. Для екологізації виробництва потрібні соціальні, економічні і технологічні передумови.

В процесі розвитку екологічної свідомості людство пройшло чотири етапи екологізації виробництва: розвиток екологічного обладнання; екологічно обумовлене вдосконалення технологій; підвищення ефективності складових життєвого циклу виробів і послуг; виробництво товарів, що обслуговують принципово новий (екологічно зберігаючий) спосіб життя.

Можна виділити наступні основні напрямки екологізації суспільного виробництва: збереження і відновлення екологічних систем; впровадження прогресивних технологій видобутку природної сировини; раціональне

використання матеріальних ресурсів; створення та впровадження маловідходних і безвідходних виробництв; екологічно прийнятне розміщення і територіальна організація виробництва; скорочення та ліквідація забруднення навколишнього природного середовища.

Україна за рівнем екологізації виробничих процесів значно відстає від розвинутих країн світу (у ФРН на частку екологічних товарів і послуг припадає близько 60 - 70 % експортних продажів). Тому для посилення екологічної спрямованості національного виробництва та споживання товарів Україні першочергово потрібно вирішити такі завдання:

– забезпечити моніторинг навколишнього природного середовища та посилити екологічні вимоги;

– забезпечити збільшення випуску та підвищення якості очисного обладнання, на яке збільшиться попит після підвищення екологічних вимог;

– забезпечити збільшення виробництва та встановлення лічильників води і газу з метою економії енергоносіїв і ресурсів;

– запровадити ресурсозберігаючі (енерго- і матеріалозберігаючі) технології, обладнання і матеріали;

– збільшити випуск засобів індивідуального екологічного моніторингу, захисту та контролю (екологічних індикаторів, фільтрів води та повітря);

– запровадити технології підвищення екологічності виробництва сільськогосподарської продукції (контроль на екологічність насіння, засобів механізації, добрив, засобів захисту рослин та переробки продукції);

– запровадити технології переробки, повторного використання та знешкодження відходів виробництва;

– збільшити питому вагу еколого-інформаційних послуг, екологічно-чистого туризму, розведення декоративних рослин, унікального тваринного світу;

– збільшити питому вагу власних фармацевтичних засобів на рослинній основі;

– покращити якість та збільшити привабливість курортних зон і оздоровчих закладів.

У формуванні інфраструктури екологічного ринку, як свідчить досвід,

можна виділити такі рівні: підприємство, корпорація; регіон; держава і міжнародний рівень.

Реалізація державної екологічної політики здійснюється на трьох рівнях управління: національному, регіональному, місцевому.

Державна екологічна політика – це комплексна програма цілей і дій держави, які направлені на зменшення екодеструктивного впливу суспільства на навколишнє природне середовище. На базі концепції державної екологічної політики формується стратегія екологізації виробничого комплексу країни та визначаються механізмії реалізації.

Завданням екологізації виробництва може бути: - реструктуризація економіки галузей і регіонів; - перепрофілювання підприємств; - усунення (зменшення) потреби в екологічно несприятливих видах продукції чи послуг; - заміна екологічно несприятливих техпроцесів; - зниження ресурсомісткості продукції тощо. Об'єкт екологізації визначається, виходячи із рівнів управління. Так, для державного управління об'єктом екологізації може бути окрема галузь: вугільна, металургійна, аграрна, лісова, автомобілебудівна, енергетична, нафтохімічна та ін.

На підприємствах в якості об'єктів екологізації виступають конкретні виробництва та технології, в процесі використання яких завдається найбільше шкоди навколишньому природному середовищу, економічним інтересам підприємств та здоров'ю населення. До таких об'єктів можуть належати процеси видобування вугілля, виплавки сталі та чавуну, виробництва хімічних добрив, електроенергії та теплоносіїв, нанесення лакофарбового та гальванічного покриття, вирощування продуктів рослинництва, переробки сміття, захоронення шкідливих відходів та ін.

6.2 Гости і стандарти на монітори і ПЕОМ

Виділяють дві основні групи стандартів і рекомендацій - з безпеки (UL, CSA, DHHS, CE, скандинавські SEMRO, DEMKO, NEMKO і FIMKO, а також FCC Class B) та ергономіки.

Ергономічність монітора і відповідність стандартам безпеки є дуже важливими для користувача. На користувача діють: рентгенівське випромінювання, електростатичні, електричні і магнітні поля. Робота за комп'ютером може погіршити зір. Певною гарантією можуть служити стандарти, що пред'являються до моніторів:

– ISO 9241-3 – стандарт на ергономічні вимоги;

– MPR II, MPR 1990:10 – Шведські стандарти безпеки по випромінюванню, електричному і магнітним полям (стандарти ЄС);

– TCO-1992, TCO-1995, TCO-1999 – стандарти Шведського союзу професійних службовців по візуальних ергономічних параметрах, змінних електричних і магнітних полях (допустимий рівень напруженості електромагнітного поля залишається незмінним у всіх трьох редакціях: в діапазоні частот від 5 Гц до 2 кГц - 10 В/м; в діапазоні частот від 2 до 400 кГц - 1 В/м);

– Blue Angel – відмова від використання токсичних матеріалів при виробництві;

– NUTEK – понижене споживання енергії;

– EPA –Energy Star – американський стандарт на енергозбереження.

В Україні також існують нормативні документи, що визначають шкідливість роботи з комп'ютером в цілому і монітором зокрема:

– ГОСТ 50948-96. «Засоби відображення інформації індивідуального користувача. Загальні ергономічні вимоги і вимоги безпеки»;

– ГОСТ 50923-96. «Дисплеї. Робоче місце оператора. Загальні ергономічні вимоги і вимоги до виробничого середовища. Методи вимірювання»;

– Санітарні правила і норми. САНПІН 2.2.2.542-96. «Гігієнічні вимоги до відео дисплейних терміналів, персональним електронно-обчислювальним машинам і організація роботи».

Монітори персональних комп'ютерів і робочих станцій при обов'язковій сертифікації піддаються сертифікаційним випробувань за такими параметрами:

– параметри безпеки - електрична, механічна, пожежна безпека (ГОСТ 50377 - 92);

– санітарно-гігієнічні вимоги - рівень звукових шумів (ГОСТ 26329 - 84 або ГОСТ 2718 - 88), ультрафіолетове, рентгенівське випромінювання і показники якості зображення (ГОСТ 27954-88);

– електромагнітна сумісність - випромінюються радіоперешкоди (ГОСТ 29216 - 91);

Сертифікат видається тільки на весь комплекс перерахованих вище ГОСТів.

Стандарт ГОСТ 27954 - 88 на відеомонітори персональних ЕОМ. Вимоги цього стандарту є обов'язковими для будь-якого монітора, що продається в Україні. Основні вимоги наведені в таблиці 6.1.

Таблиця 6.1 – Вимоги на характеристики на монітор за ГОСТ 27954 - 88

Характеристика монітора	Вимога
Частота кадрів при роботі з позитивним контрастом	Не менш 60 Гц
Частота кадрів в режимі обробки тексту	Не менш 72 Гц
Тремтіння елементів зображення	Не більше 0,1 мм
Покриття антивідблиску	обов'язково
Допустимий рівень шуму	Не більше 50 дБА
Потужність дози рентгенівського випромінювання на відстані 5 см від екрану при 41-годинному робочому тижні	Не більше 0,03 мкР/с

Крім того, цим стандартом не допускається застосування вибухонебезпечних електронно-променевих трубок, регламентується ступінь деталізації технічної документації на монітори, а також встановлюються вимоги стандартизації і уніфікації, технологічності, ергономіки та технічної естетики, безпеки, технічного ремонту та обслуговування, а також надійності.

6.3 Висновки до розділу

В цьому розділі описано екологізацію виробництв та гости і стандарти на монітори і ПЕОМ.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Міжнародне співробітництво в галузі охорони праці

Важливими нормативними актами з питань охорони праці є міжнародні договори та угоди, до яких приєдналась Україна. Закон “Про охорону праці" передбачає, якщо міжнародним договором, згода на обов’язковість якого надана Верховною Радою України, встановлено інші норми, ніж ті, що передбачені законодавством України про охорону праці, застосовуються норми міжнародного договору.

На міжнародному рівні проблемами охорони праці займається спеціалізована установа Організації Об’єднаних націй – Міжнародна організація праці (МОП), крім неї, свій внесок у справу охорони праці роблять також Міжнародне агентство з атомної енергії (МАГАТЕ), Всесвітня організація охорони здоров’я (ВООЗ), Міжнародна організація зі стандартизації (ІСО), Міжнародна організація авіації (ІКАО) та ряд інших. [14]

Переважна більшість міжнародних договорів та угод, в яких бере участь Україна і які більшою або меншою мірою стосуються охорони праці, – це наступні чотири групи документів: Конвенції та рекомендації Міжнародної Організації Праці; Директиви Європейського Союзу; договори та угоди, підписані в рамках Співдружності Незалежних Держав; двосторонні договори та угоди.

Значне місце серед міжнародних договорів, якими регулюються трудові відносини, займають конвенції Міжнародної Організації Праці у галузі поліпшення умов праці та рекомендації щодо їх застосування. Структурно МОП складається з Міжнародної Конференції праці, Адміністративної Ради та Міжнародного Бюро праці. Міжнародна Конференція праці – вищий орган МОП і тому вона зветься також Всесвітнім Парламентом праці – проводиться щороку за участі представників всіх країн членів. Міжнародне Бюро праці – це постійний секретаріат організації, який розробляє кодекси практичних заходів, здійснює моніторинг фінансових справ, розробляє порядок денний наступних

Міжнародних Конференцій праці. Адміністративна Рада здійснює контроль за діяльністю Міжнародного Бюро праці та зв'язок між ним і Міжнародною конференцією праці.

Всі механізми прийняття рішень в МОП пов'язані з її структурою, яка базується на принципі трипартизму, тобто рівного представництва трьох сторін – уряду, роботодавців і робітників. До основних напрямів діяльності МОП належать: участь у міжнародно-правовому регулюванні праці шляхом розроблення та ухвалення нормативних актів (конвенцій і рекомендацій) з питань умов праці та життя; розроблення та здійснення міжнародних цільових програм, спрямованих на вирішення важливих соціально-трудових проблем (зайнятість, умови праці та ін.); надання допомоги державам–членам МОП в удосконаленні національного трудового законодавства, професійно-технічної підготовки працівників, поліпшенні умов праці шляхом здійснення міжнародних програм технічного співробітництва, проведення дослідницьких робіт та видавничої діяльності.

З часу свого існування МОП ухвалила понад 180 Конвенцій і понад 190 рекомендацій з різних соціально-трудових проблем.

Україна є членом МОП із 1954 року, але ефективна робота нашої країни в рамках цієї організації почалася фактично після 1991 року. На цей час Україна ратифікувала 58 конвенцій МОП, серед яких – найважливіші нормативні акти, що стосуються основоположних прав людини та охорони праці.

Особливе місце серед Конвенцій МОП займає Конвенція № 155 "Про безпеку і гігієну праці та виробничу санітарію", яка закладає міжнародно-правову основу національної політики щодо створення всебічної і послідовної системи профілактики нещасних випадків на виробництві і професійних захворювань. [14]

У МОП діє система контролю за застосуванням в країнах-членах Організації конвенцій і рекомендацій. Кожна держава зобов'язана подавати доповіді про застосування на своїй території ратифікованих нею конвенцій, а також інформації про стан законодавства і практики з питань, що порушуються в окремих, не ратифікованих нею конвенціях.

Участь України у міжнародних договорах і угодах, що стосуються охорони праці не обмежується конвенціями та рекомендаціями МОП. Значне місце серед міжнародних договорів, якими регулюються трудові відносини в нашій країні займають також Директиви Європейського Союзу, договори та угоди, підписані в рамках Співдружності незалежних держав, двосторонні договори та угоди.

Директиви, що приймаються в рамках ЄС і є законами для всіх його країн, відповідають конвенціям МОП. З іншого боку, при розробці нових конвенцій, рекомендацій та інших документів МОП враховує передовий досвід країн-членів ЄС. Все зростаюча важливість директив ЄС обумовлена багатьма причинами, серед яких найсуттєвішими є наступні: спільні стандарти здоров'я і безпеки сприяють економічній інтеграції, оскільки продукти не можуть вільно циркулювати всередині Союзу; скорочення людських, соціальних та економічних витрат, пов'язаних з нещасними випадками та професійними захворюваннями; запровадження найбільш ефективних методів роботи повинно принести з собою ріст продуктивності, зменшення експлуатаційних (поточних) витрат і покращення трудових стосунків; регулювання певних ризиків повинно узгоджуватись на наднаціональному рівні в зв'язку із масштабом ресурсних затрат і з тим, що будь-яка невідповідність в суті і використанні таких положень призводить до "викривлень" у конкуренції і впливає на ціни товарів.

Україна не є членом ЄС, але неодноразово на найвищих рівнях заявляла про своє прагнення до вступу до цієї організації. Однією з умов прийняття нових країн до ЄС є відповідність їхнього законодавства законодавству ЄС, тому в нашій країні ведеться актив на робота по узгодженню вимог законів та інших нормативно-правових актів директивам ЄС.

7.2 Гігієнічні вимоги до виробничих приміщень з ЕОМ

На робочих місцях працівників, які відповідальні за експлуатацію сервісу управління механізмом авторських прав на мультимедійні файли, необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від

14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Приміщення для роботи з ЕОМ мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією. У приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості та рухливості повітря відповідно до норм та правил, а також ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування», затверджених наказом Мінрегіону від 25.01.2013 р. № 24. Відповідно до санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 в офісних приміщеннях, обладнаних ЕОМ, температура повітря повинна становити 22-25°С, відносна вологість повітря – 40-60 %, швидкість руху повітря – не більше 0,1 м/с. [15]

Приміщення, призначені для роботи з ЕОМ, повинні мати природне освітлення. У виробничих приміщеннях, обладнаних ЕОМ, необхідно створити належне освітлення. При експлуатації сервісу управління механізмом авторських прав на мультимедійні файли, важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18 [16]. Природне світло повинно бути бічним, зорієнтованим, як правило, на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості не нижче 1,5%. При виробничій потребі дозволяється експлуатувати ЕОМ у приміщеннях без природного освітлення за узгодженням з органами Держпромгірнагляду та органами й установами санітарно-епідеміологічної служби. Вікна приміщень повинні мати регулювальні пристрої для відчинення, а також жалюзі, штори тощо. Штучне освітлення приміщення з робочими місцями, обладнаними відеотерміналами ЕОМ загального та персонального користування, має бути всеосяжним і рівномірним. У випадку, коли переважають роботи з документами, допускається комбіноване освітлення (додатково до загального освітлення встановлюється світильники місцевого освітлення). Світильники розміщуються збоку від робочих місць (переважно ліворуч), або локально над робочим місцем (при розташуванні відеотерміналів ЕОМ за периметром приміщення). Як джерело світла при

штучному освітленні застосовуються, як правило, люмінесцентні лампи. У світильниках місцевого освітлення допускається застосування ламп розжарювання. Рівень освітленості на робочому місці має становити 300-500 лк. При використанні комбінованого освітлення не допускається відблисків на поверхні екрана та збільшення освітлення екрана вище 300 лк.

Орієнтація вікон повинна бути на північ або північний схід, вікна повинні мати жалюзі, які можна регулювати, або штори; не дозволяється розміщувати кабінети обчислювальної техніки у підвальних приміщеннях будинків; кабінети, обладнані комп'ютерною технікою, в навчальних закладах повинні розміщуватись в окремих приміщеннях з природним освітленням та організованим обміном повітря; стіни, стеля і підлога та обладнання кабінетів комп'ютерної техніки повинні мати покриття із матеріалів з матовою фактурою з коефіцієнтом відбиття: стін — 40- 50 %, стелі — 70 - 80 %, підлоги — 20-30 %, предметів обладнання — 40-60 % (робочого столу — 40-50 %, корпуса дисплею та клавіатури — 30-50 %, стелажів — 40-60 %); поверхня підлоги повинна мати антистатичне покриття та бути зручною для вологого прибирання; забороняється використовувати для оздоблення інтер'єру приміщень комп'ютерних кабінетів полімерні матеріали (дерев'яно-стружкові плити, шпалери, що придатні для миття, плівкові та рулонні синтетичні матеріали, шаровий паперовий пластик та ін.), що виділяють у повітря шкідливі хімічні речовини, які перевищують гранично допустимі концентрації; вміст шкідливих хімічних речовин в повітрі дошкільних та учбових приміщень з комп'ютерною технікою не повинен перевищувати середньодобові концентрації. [15]

Організація робочого місця фахівця із експлуатації сервісу повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги».

Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Рівень шуму не повинний перевищувати: на місцях, де працюють програмісти та оператори ЕОМ, 55 дБА, у лабораторіях, де складаються

алгоритми та ведеться робота з документацією – 60 дБА, у машинному залі – 65 дБА, у приміщеннях, де розміщені гучні агрегати обчислювальних машин, – 75 дБА.

7.3 Планування заходів цивільного захисту на об'єкті у випадку надзвичайної ситуації

Найбільш повне та організоване виконання заходів цивільного захисту (ЦЗ) на об'єкті досягається завчасною розробкою плану заходів, які необхідно проводити при загрозі або виникненні надзвичайної ситуації (НС).

План дій органів управління і сил ЦЗ (міністерств, відомств, областей, районів, міст, підприємств, установ і організацій) із запобігання і ліквідації НС розробляється на підставі законодавчих, директивних і нормативних документів і призначений для координації і діяльності центральних і місцевих органів виконавчої влади, керівництва, а також оперативності їх реагування на загрозу і виникнення НС, відвернення або зниження можливої загибелі людей, мінімізація матеріальних збитків і втрат та організацію задоволення першочергових потреб населення, яке постраждало [15].

План визначає порядок дій і відповідальність керівництва відповідних органів управління підприємств, установ і організацій, а також основні заходи щодо організації і проведення робіт із запобігання і ліквідації НС техногенного і природного характеру, узгодження термінів їх виконання, фінансові, матеріальні та інші ресурси, які необхідні для цих заходів і робіт. У план дій включаються заходи щодо захисту робітників і службовців, підтримування виробничої діяльності та інші з урахуванням обстановки після виникнення НС, передбачаються необхідна кількість сил і засобів для ліквідації наслідків НС. При плануванні використовуються необхідні вихідні дані та довідкові матеріали з урахуванням специфіки роботи та особливостей щодо відомчої та регіональної діяльності підприємства, організації чи установи. Основними вихідними даними при розробці плану дій на об'єкті є рішення та вказівки вищого штабу ЦЗ, розпоряджень начальника ЦЗ об'єкта, документів, що характеризують об'єкт (комунально-

енергетичні мережі, стан будівель і споруд, вододжерела та ін.). План дій розробляється на підставі наказу начальника ЦЗ об'єкта. До розробки документів плану залучається керівний склад і спеціалісти об'єкта. Начальник штабу ЦЗ складає графік розробки окремих документів (розділів) і контролює його виконання.

План дій розробляється у двох (при необхідності і більше) примірниках. Підписується план дій начальником штабу ЦЗ об'єкта, погоджується з територіальними управліннями (відділами) з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи і затверджується начальником ЦЗ об'єкта. Після затвердження зміст плану дій доводиться до виконавців. План дій органів управління і сил ЦЗ із запобігання та ліквідації НС – це програма здійснення запобіжних та захисних заходів. Він дозволяє цілеспрямовано та організовано вирішувати завдання ЦЗ в умовах НС мирного та воєнного часу. Основу плану складають заходи щодо захисту робітників, службовців і членів їх сімей. При визначенні цих заходів враховується важливість та особливості виробничої діяльності об'єкта, основні завдання органів управління та сил ЦЗ щодо запобігання і ліквідації НС. План дій органів управління та сил ЦЗ на мирний час складається із п'яти розділів текстової частини і додатків до них. Текстова частина плану включає такі розділи: 1. Висновки із оцінки обстановки на території об'єкта. 2. Приведення в готовність та організація роботи органів управління в НС. 3. Сили ЦО об'єкта, що залучаються до виконання аварійно-рятувальних, пошукових та відновлювальних робіт. 4. Організація забезпечення заходів та дій ЦЗ. 5. Організація управління, оповіщення і зв'язку.

Окремо розробляється "План дій органів управління та сил ЦЗ об'єкта при переведенні з мирного на воєнний стан" за ступенями готовності воєнного часу та раптовому нападі супротивника. Крім цього, на об'єкті господарської діяльності розробляються плани служб ЦЗ, щодо забезпечення заходів і дій органів управління і сил ЦЗ при загрозі і виникненні НС та при переведенні органів управління і сил з мирного на воєнний стан.

7.4 Врахування шкідливих і небезпечних умов праці персоналу в ході провадження виробничої діяльності суб'єктами господарювання

Умови праці на виробництві диференціюються залежно від фактично визначених рівнів та факторів виробничого середовища порівняно із санітарними нормами, правилами, гігієнічними нормативами, а також з урахуванням їх можливого шкідливого впливу на стан здоров'я працюючих. Шкідливі умови — характеризуються такими рівнями шкідливих виробничих факторів, які перевищують гігієнічні нормативи і здатні чинити несприятливий вплив на організм працюючого та (або) його майбутніх нащадків. Шкідливі умови за показниками перевищення гігієнічних нормативів та вираженості можливих змін в організмі працюючих поділяються на чотири ступеня, де четвертий (найтяжчий) – це небезпечні (екстремальні) умови.

В Україні створена належна база нормативно-правових актів щодо організації і проведення атестації робочих місць за умовами праці, а також з надання працівникам підприємств пільг і компенсацій за особливо важкі та особливо шкідливі також за важкі і шкідливі умови праці. Зокрема Постанова Кабінету Міністрів України від 12.07.2005 р. № 576 «Про затвердження переліку робіт із важкими, шкідливими та особливо шкідливими умовами праці у будівництві, на яких встановлюється підвищена оплата праці», Державні санітарні норми та правила «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» (затверджені наказом Міністерства охорони здоров'я України від 08.04.2014 № 248) та інші документи.

Для врахування шкідливих і небезпечних умов праці персоналу в ході провадження виробничої діяльності суб'єктами господарювання (підприємствами, установами і організаціями незалежно від форм власності і господарювання, де технологічний процес, використовуване обладнання, сировина та матеріали є потенційними джерелами шкідливих і небезпечних виробничих факторів), проводиться атестація робочих місць.

Відповідно до ст. 153 Кодексу законів «Про працю в Україні», на всіх

підприємствах, в установах, організаціях створюються безпечні і нешкідливі умови праці. Забезпечення безпечних і нешкідливих умов праці покладається на власника або уповноважений ним орган. Власник або уповноважений ним орган зобов'язаний провести (забезпечити) лабораторні дослідження робочих місць, на яких існують шкідливі і важкі умови праці, встановити пільги і гарантії для працівників, що працюють у цих умовах, та розробити заходи, що забезпечують усунення причин виникнення нещасних випадків і професійних захворювань. Для цього проводиться атестація робочих місць за умовами праці. Керівник підприємства затверджує перелік робочих місць, виробництв, робіт, професій і посад працівників, яким підтверджено право на пільги і компенсації, а також на пільгове пенсійне забезпечення. Шкідливі умови праці суттєво впливають не тільки на виникнення професійних захворювань, а й на виникнення і тривалість загальних захворювань. Відповідно, особи, які будуть працювати на робочих місцях із шкідливими та небезпечними умовами праці, підвищеними фізичними та емоційними навантаженнями насамперед підлягають професійному відбору, попередньому та періодичному медичних оглядах.

Підприємство повинно враховувати витрати на компенсацію за роботу в несприятливих умовах, що не відповідають санітарним нормам (надавати пільги за важкі і шкідливі умови праці), зокрема додаткові відпустки; скорочений робочий день; лікувально-профілактичне харчування; безкоштовна видача молока чи інших рівноцінних продуктів; підвищені тарифні ставки; доплати за умови та інтенсивність праці; пільгові пенсії. Також працівники зі шкідливими умовами праці повинні забезпечуватися спеціальними захисними засоби, такими як: спецодяг та спецвзуття; засоби захисту рук (рукавиці), захисно-профілактичні засоби (пасти, мазі) та очисники шкіри (мило, синтетичні мийні засоби); засоби індивідуального захисту органів дихання; засоби захисту голови (каски, шоломи); засоби захисту очей і обличчя (захисні окуляри та маски); засоби захисту органу слуху (шоломи, антифони).

7.5 Висновки до сьомого розділу

В цьому розділі розглянуто важливі питання охорони праці та безпеки в надзвичайних ситуаціях, зокрема міжнародне співробітництво в галузі охорони праці та гігієнічні вимоги до виробничих приміщень з ЕОМ. Також описано процес планування заходів цивільного захисту на об'єкті у випадку надзвичайної ситуації та розглянуто проблему шкідливих і небезпечних умов праці персоналу в ході роботи з програмних засобом і запропоновано варіанти компенсації за роботу у таких умовах та спеціальні засоби захисту.

14. Міжнародне співробітництво України у галузі охорони праці [Електронний ресурс] - Режим доступу: http://www.lnu.edu.ua/life-safety/wp-content/uploads/2018/10/OP-2018_Part-9.pdf (дата звернення: 07.11.2019)

15. Козлов С.С. Методичні вказівки до виконання розділу "Охорона праці та безпека в надзвичайних ситуаціях" в дипломних проектах для підготовки студентів факультету електроніки за освітньо-кваліфікаційним рівнем "Спеціаліст" та "Магістр". "Вимоги безпеки під час експлуатації обчислювальної техніки" / К.:НТУУ "КПІ", 2015, - 30 с.

16. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. [Електронний ресурс] - Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 07.11.2019)

ВИСНОВКИ

В рамках виконання дипломної роботи були розроблені системні вимоги, побудовані архітектури і програмно реалізовано функції для двох масштабних проектів.

У разі повноцінної реалізації проекту для сценарію «Музика і метадані» очевидно поліпшення рівня захисту авторських прав і забезпечення більшої прозорості і доступності правової інформації (при повсюдному використанні сервісу).

При реалізації проекту для сценарію «Аудіо-фрагменти і метадані» може бути отримано програмний засіб, здатний зібрати воєдино і перевірити на наявність маніпуляцій безліч записів для подальшого використання, що запобігає завантаження дублікатів.

Відповідно до поставленої мети та завдань було досягнуто наступних результатів:

– проведено дослідження ситуації в сфері дотримання авторських прав на музичні записи;

– здійснено аналіз основних можливостей технології Blockchain щодо її використання у медіа сфері, зокрема для боротьби з піратством, полегшення контролю за контентом та управління правами користувачів;

– досліджено основні можливості смарт-контракту як способу обміну цифровими цінностями (даними) в технології Blockchain;

– проведено аналіз різних видів архітектури взаємодії «клієнт-сервер»;

– розглянуті сценарії впровадження технології Blockchain для реєстрації та аутентифікації аудіоматеріалу для подальшого розповсюдження;

– запропоновано варіанти використання (use cases), які є засобом опису функціональних вимог до сервісу;

– побудовані діаграми компонентів, які в різних аспектах ілюструють склад і поведінку розроблюваних програмних елементів сервісу.

У плани подальшої роботи входять:

– забезпечення можливості використання криптовалют для проведення грошових транзакцій, оскільки у багатьох країнах криптовалюта не визнається;

– проведення грошових операцій через існуючу банківську систему, не відмовляючись від зберігання інформації про транзакції в Blockchain;

– розширення сервісу за допомогою нейронної мережі, призначеної для вирішення конфліктів на грунті сумнівів щодо оригінальності аудіо-фрагмента (відсутності маніпуляцій з аудіо-фрагментом).

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Don Tapscott, Alex Tapscott Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World / Don Tapscott, Alex Tapscott Blockchain – K.: Information Systems, 2016 – pp. 100 – 150.
2. Pavan Duggal Blockchain Contracts and Cyberlaw / Pavan Duggal – K. : Information Systems, 2015 – pp. 15 – 39.
3. Jacob William Blockchain: The Simple Guide To Everything You Need To Know / Jacob William – K. : Information technologies, 2016 – pp. 40 – 50.
4. How Do Ethereum Smart Contracts Work? // CoinDesk, Inc. [Електронний ресурс] – Режим доступу: https://www.coindesk.com/information/ethereum-smart-contracts-work (дата звернення: 07.10.2019).
5. What is Gas? // MyEtherWallet – open-source interface for generating Ethereum wallets & more. [Електронний ресурс] – Режим доступу: https://kb.myetherwallet.com/gas/what-is-gasethereum.html (дата звернення: 02.11.2019).
6. http://www.multichain.com/ [Електронний ресурс] – Режим доступу: https://blockchain.info/api/ – (дата звернення: 30.10.2019).
7. Blockchain Wallet API: Bitcoin Wallet API – Blockchain [Електронний ресурс] – Режим доступу: https://blockchain.info/api/ (дата звернення 15.11.2019).
8. Blockchain Applications in Music. [Електронний ресурс] – Режим доступу: https://www.blockchaintechnologies.com/applications/music/ – (дата звернення: 15.11.2019).
9. Копанецький О. Використання технології Blockchain для управління механізмом авторських прав на аудіофайли / О. Копанецький – Матеріали VII науково-технічній конференції «Інформаційні моделі, системи та технології» – Тернопіль, ТНТУ, 11-12 грудня 2019 р. – с. 53.
10. Майк МакГрайт. Программирование на Python для начинающих: [перевод с англ. М.А. Райтмана] - М .: Эскмо, 2015. - 192 с.
11. Марк Саммерфилд. Python на практике. / Пер. с англ. Слинкин А.А. -

М .: ДМК Пресс, 2016. - 338 с .: илл.

12. Неполное Руководство по SQLite для пользователей Windows. [Електронний ресурс] – Режим доступу: http://er.nau.edu.ua/bitstream/NAU/10100/6/SQLite.Allow.0.90.pdf - (дата звернення 26.09.2019).

13. Джигирей В.С. Екологія та охорона навколишнього природного середовища. Навчальний посібник. – К.: Знання, 2006. – 219 с.

14. Міжнародне співробітництво України у галузі охорони праці [Електронний ресурс] - Режим доступу: http://www.lnu.edu.ua/life-safety/wp-content/uploads/ 2018/10/OP-2018_Part-9.pdf (дата звернення: 07.10.2019)

15. Козлов С.С. Методичні вказівки до виконання розділу “Охорона праці та безпека в надзвичайних ситуаціях” в дипломних проектах для підготовки студентів факультету електроніки за освітньо-кваліфікаційним рівнем “Спеціаліст” та ”Магістр”. "Вимоги безпеки під час експлуатації обчислювальної техніки" / К.:НТУУ ”КПІ”, 2015. - 30 с.

16. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. [Електронний ресурс] - Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 07.11.2019)

ДОДАТКИ

ДОДАТОК А
Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

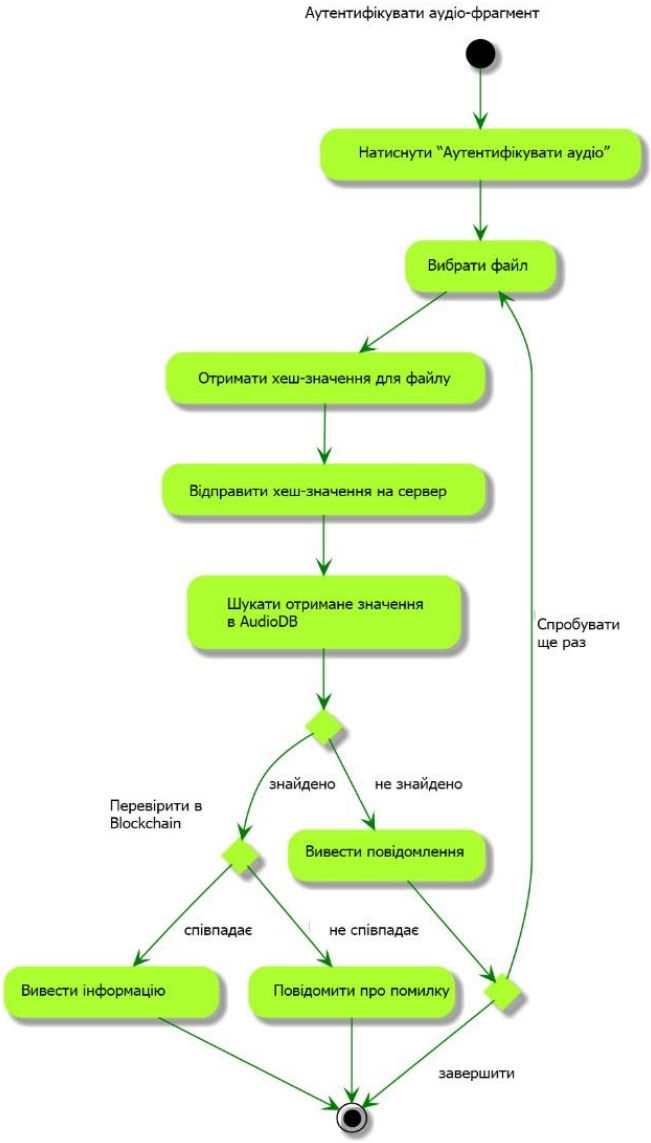
«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ »

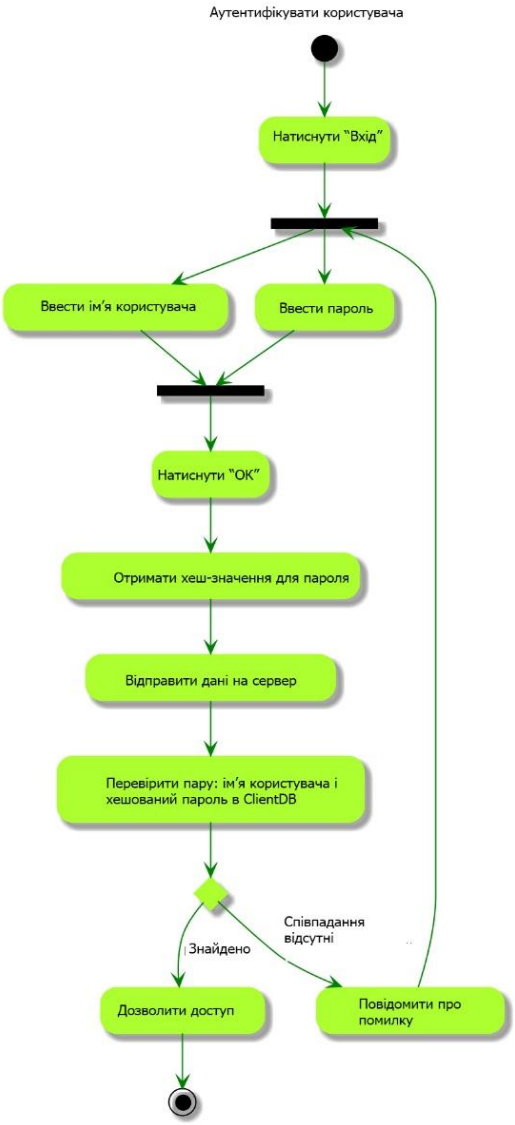
11–12 грудня 2019 року

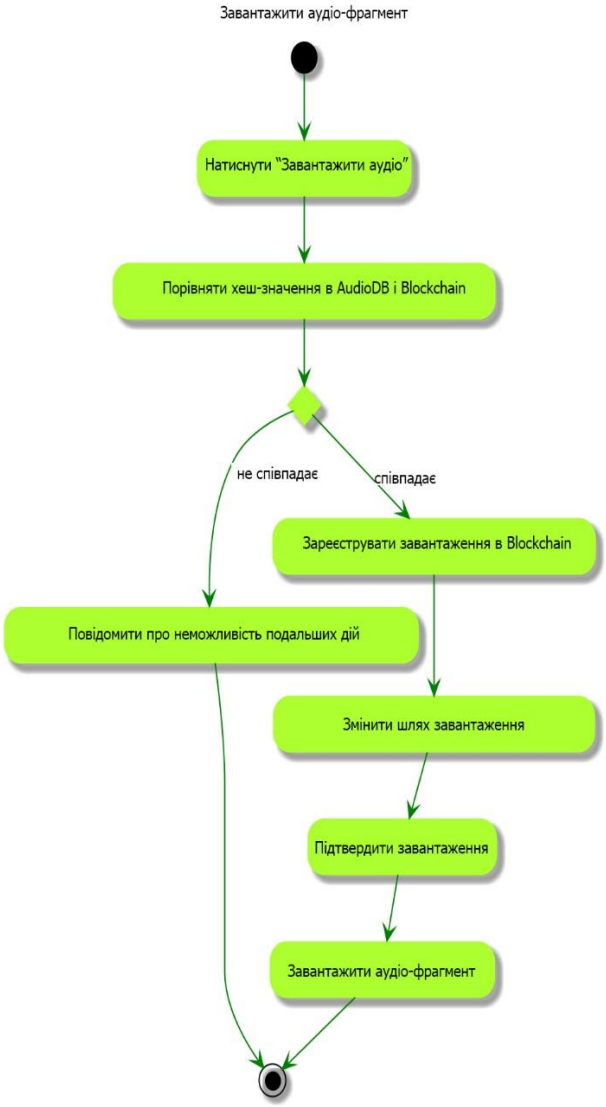
ТЕРНОПІЛЬ
2019

ДОДАТОК Б

Діаграми діяльності для сценарію «Аудіо-фрагменти і метадані»

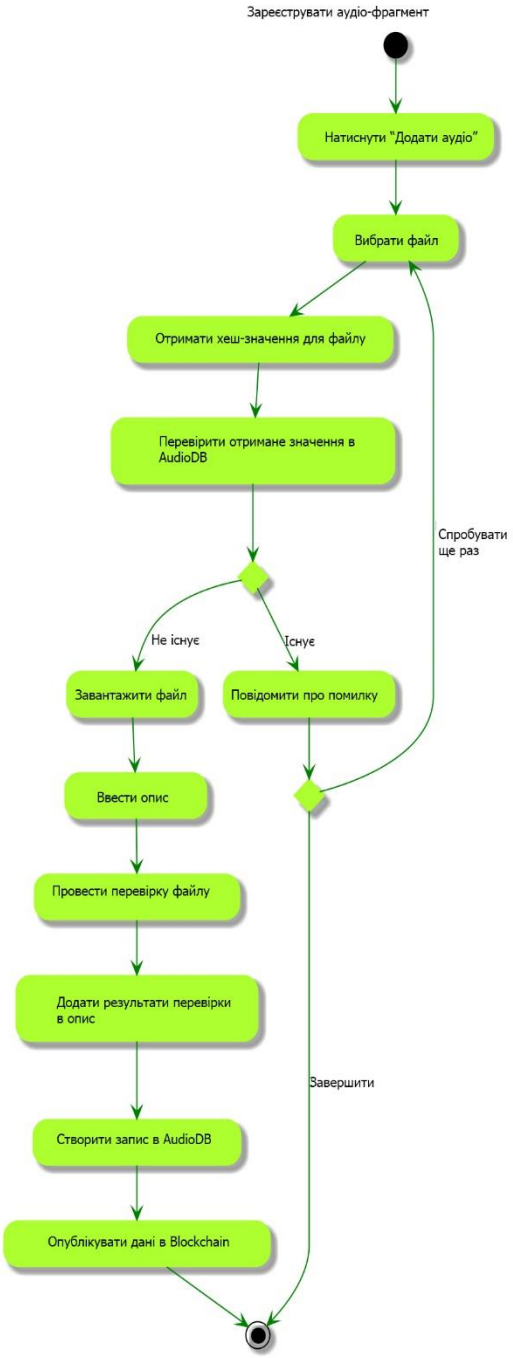


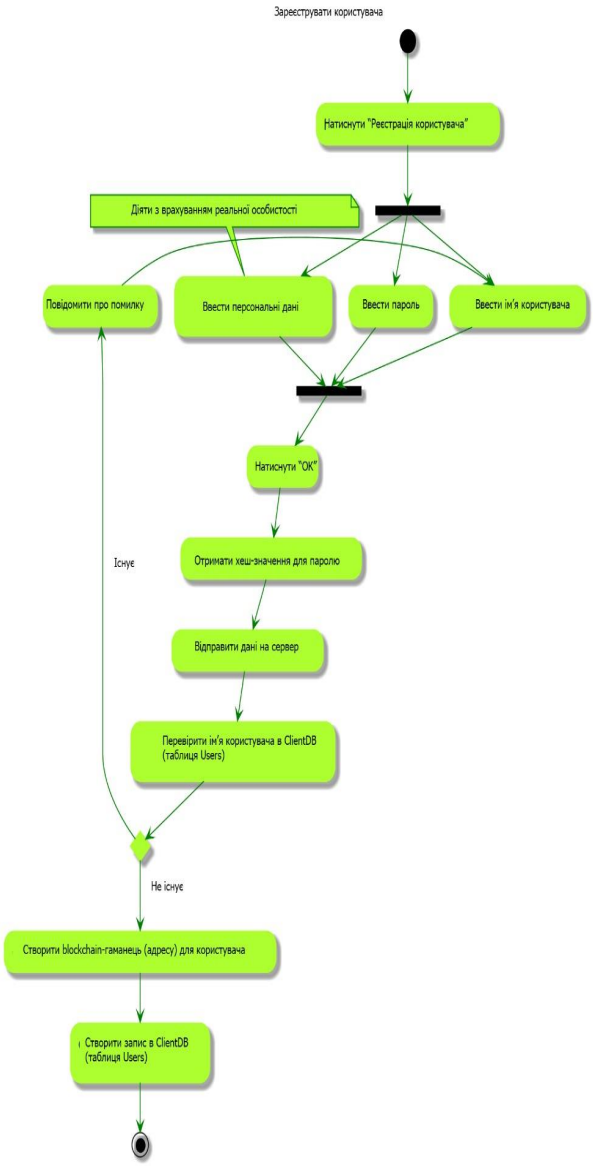

Аутентифікувати користувача
Натиснути "Вхід"
Ввести ім'я користувача
Ввести пароль
Натиснути "ОК"
Отримати хеш-значення для пароля
Відправити дані на сервер
Перевірити пару: ім'я користувача і хешований пароль в ClientDB
Знайдено
Співпадання відсутні
Дозволити доступ
Повідомити про помилку


Завантажити аудіо-фрагмент
Натиснути "Завантажити аудіо"
Порівняти хеш-значення в AudioDB і Blockchain
не співпадає
співпадає
Зареєструвати завантаження в Blockchain
Повідомити про неможливість подальших дій
Змінити шлях завантаження
Підтвердити завантаження
Завантажити аудіо-фрагмент


Прочитати метадані
Знайти аудіо-фрагмент
Відкрити сторінку метаданих

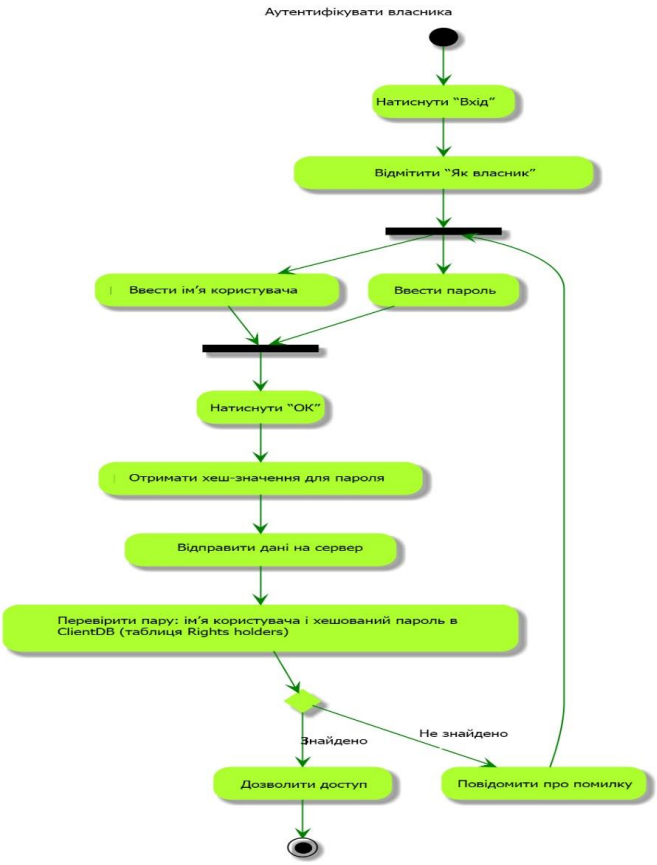

Знайти аудіо-фрагмент
Ввести параметр пошуку
Шукати в AudioDB
Вивести результати

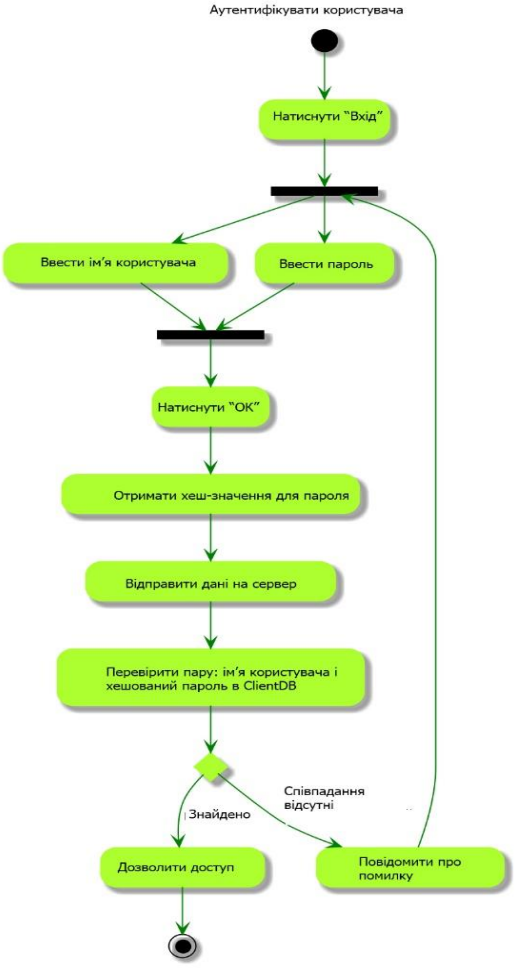

Зареєструвати аудіо-фрагмент
Натиснути "Додати аудіо"
Вибрати файл
Отримати хеш-значення для файлу
Перевірити отримане значення в AudioDB
Не існує
Існує
Завантажити файл
Повідомити про помилку
Спробувати ще раз
Ввести опис
Провести перевірку файлу
Додати результати перевірки в опис
Завершити
Створити запис в AudioDB
Опублікувати дані в Blockchain

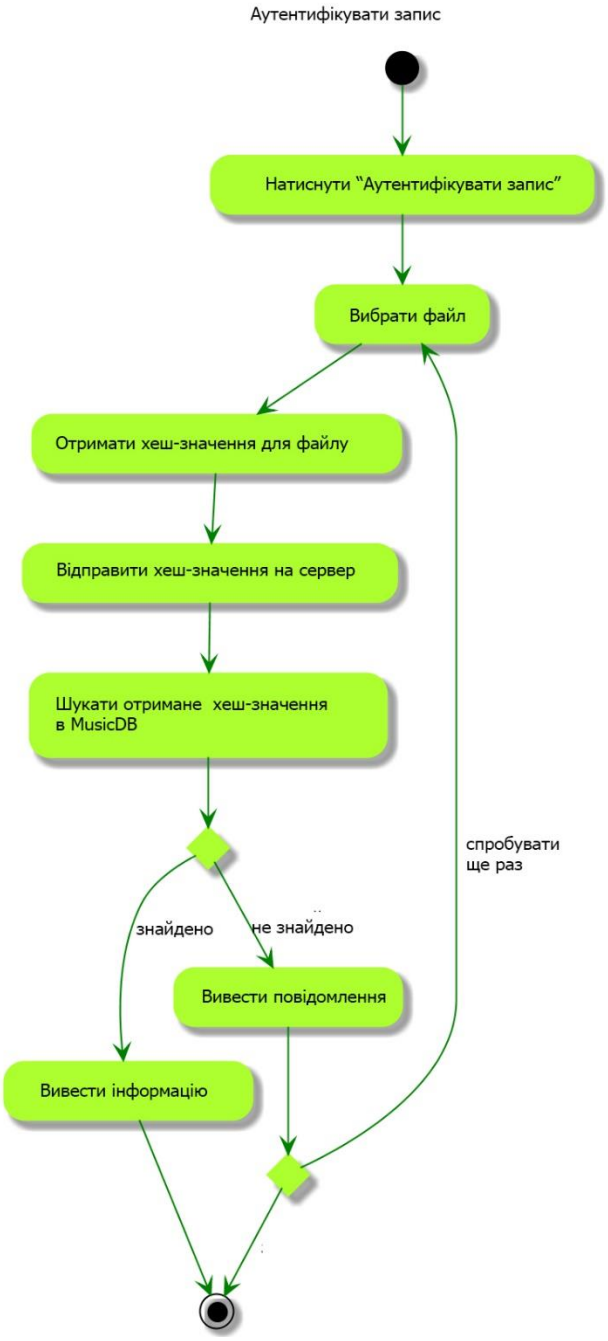

Зареєструвати користувача
Натиснути "Реєстрація користувача"
Діяти з врахуванням реальної особистості
Повідомити про помилку
Ввести персональні дані
Ввести пароль
Ввести ім'я користувача
Натиснути "ОК"
Існує
Отримати хеш-значення для паролю
Відправити дані на сервер
Перевірити ім'я користувача в ClientDB (таблиця Users)
Не існує
Створити blockchain-гаманець (адресу) для користувача
Створити запис в ClientDB (таблиця Users)

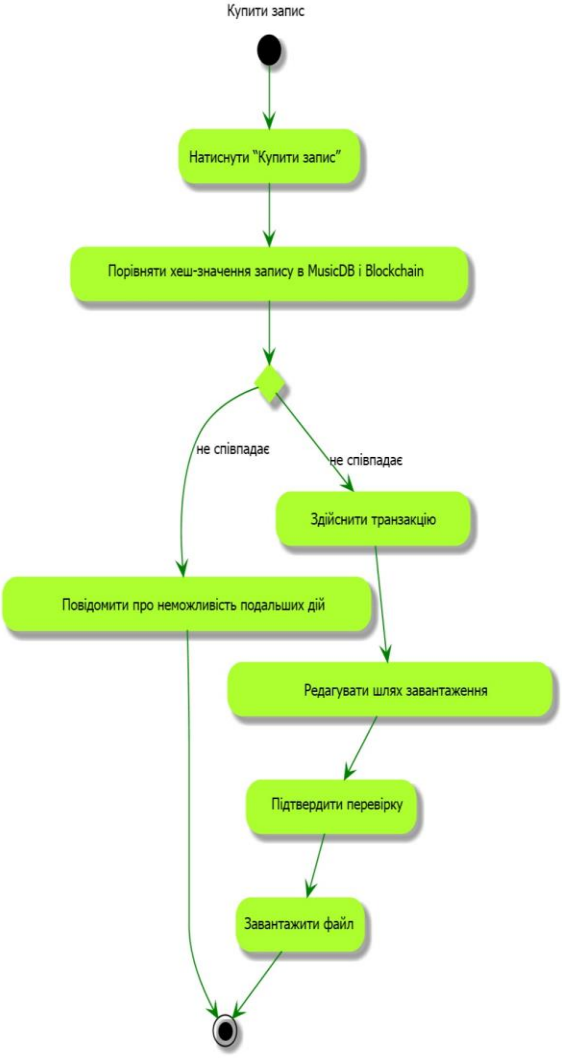
ДОДАТОК В

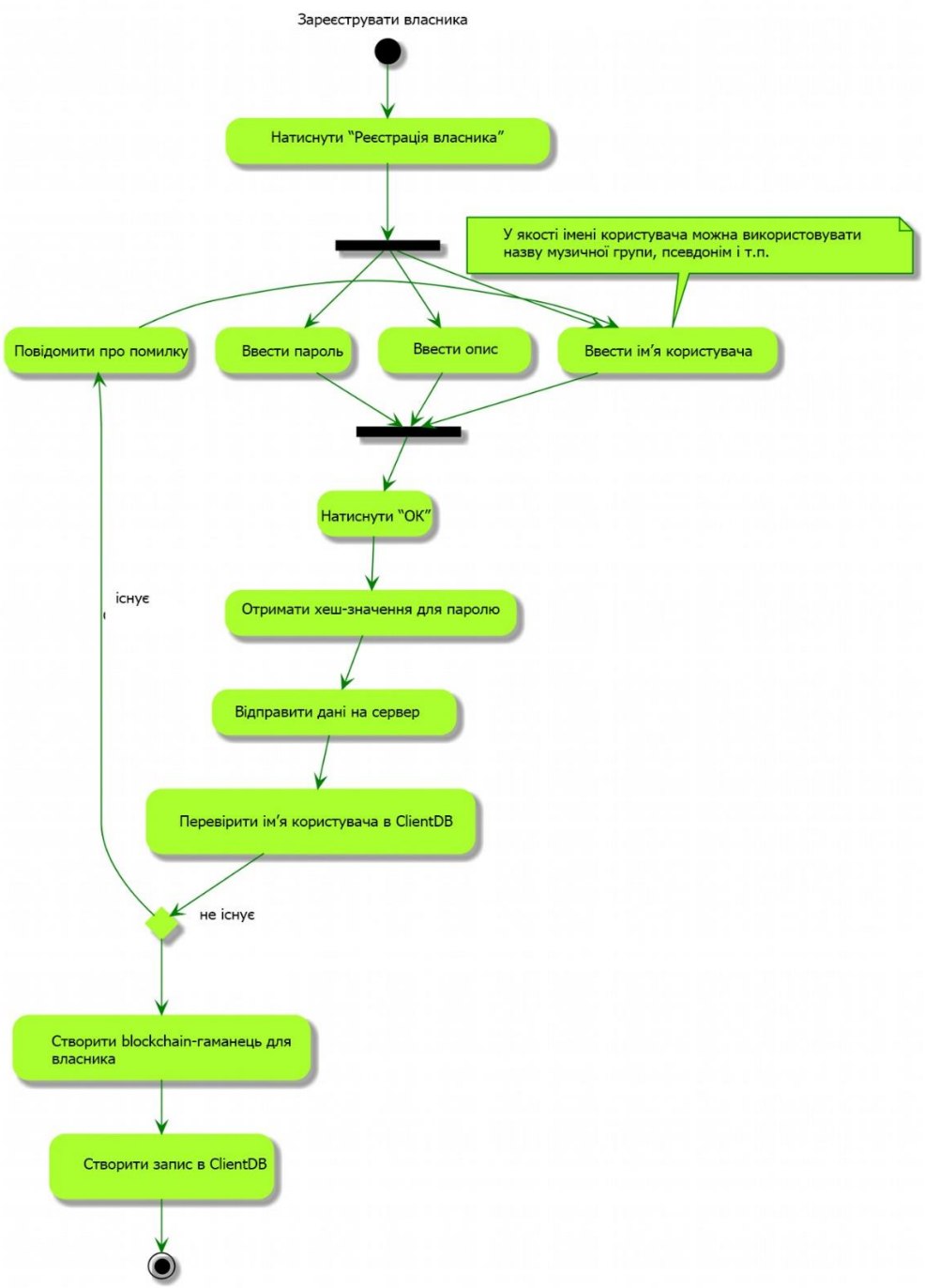
Діаграми діяльності для сценарію «Музика і метадані»

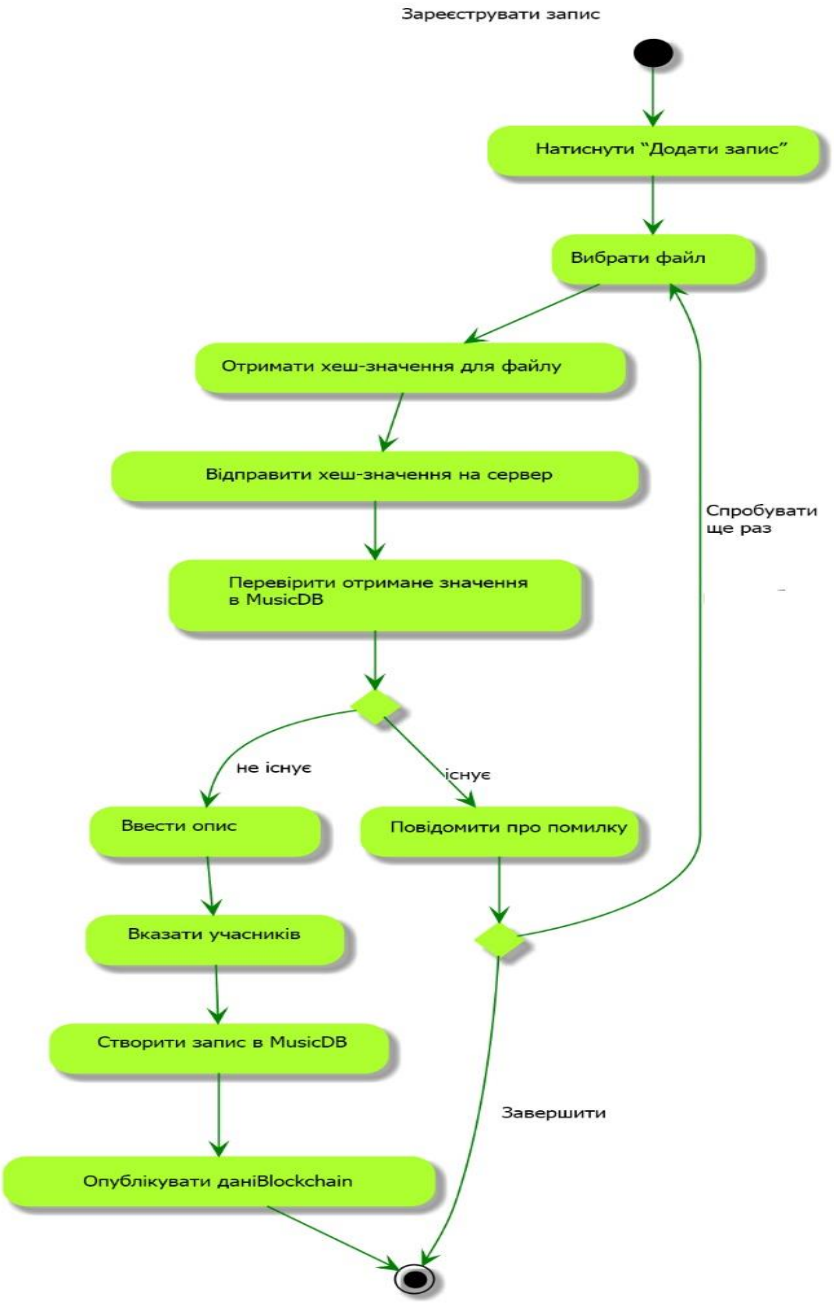


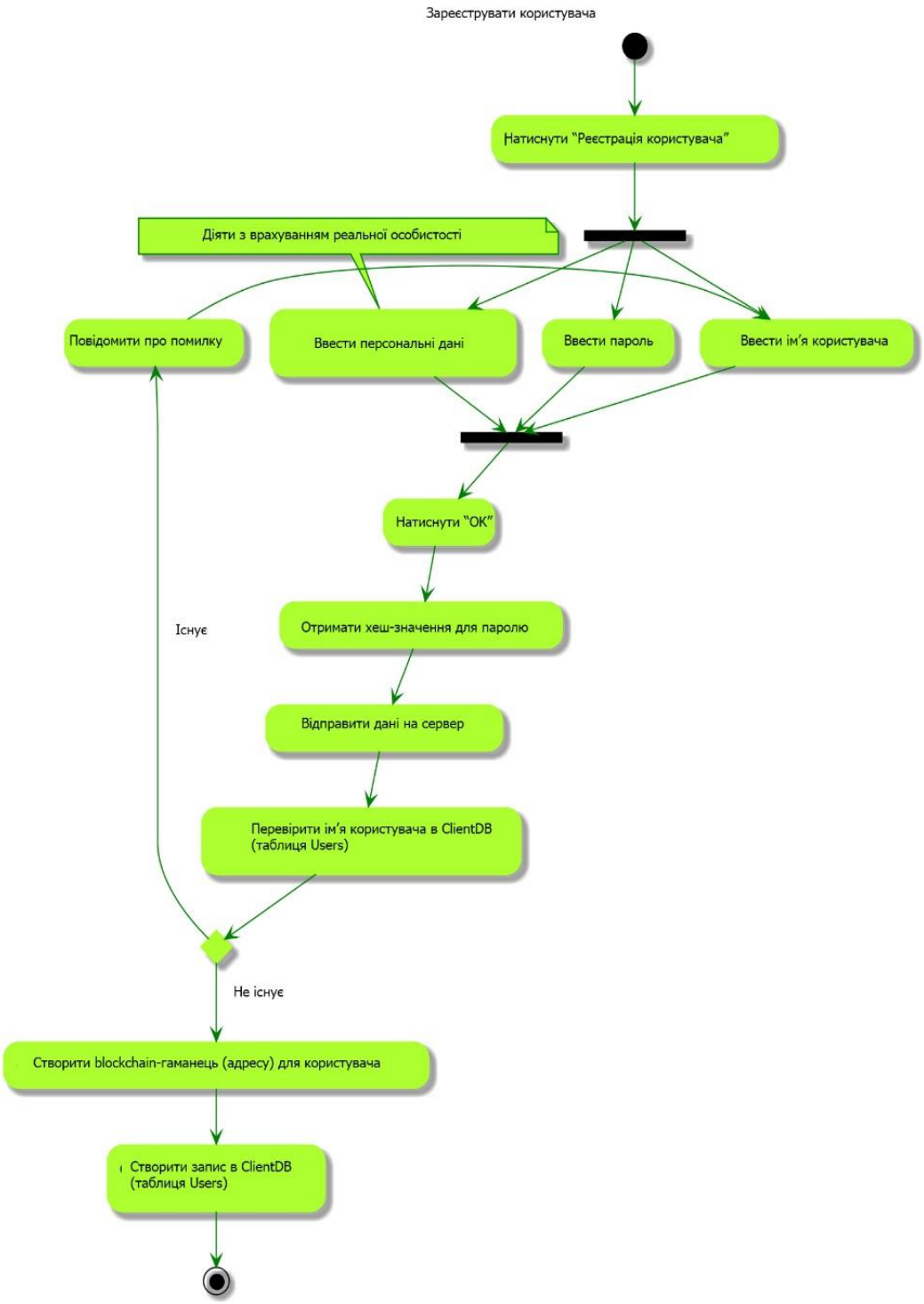



Аутентифікувати запис
Натиснути "Аутентифікувати запис"
Вибрати файл
Отримати хеш-значення для файлу
Відправити хеш-значення на сервер
Шукати отримане хеш-значення в MusicDB
знайдено
не знайдено
Вивести повідомлення
Вивести інформацію
спробувати ще раз


Купити запис
Натиснути "Купити запис"
Порівняти хеш-значення запису в MusicDB і Blockchain
не співпадає
не співпадає
Здійснити транзакцію
Повідомити про неможливість подальших дій
Редагувати шлях завантаження
Підтвердити перевірку
Завантажити файл

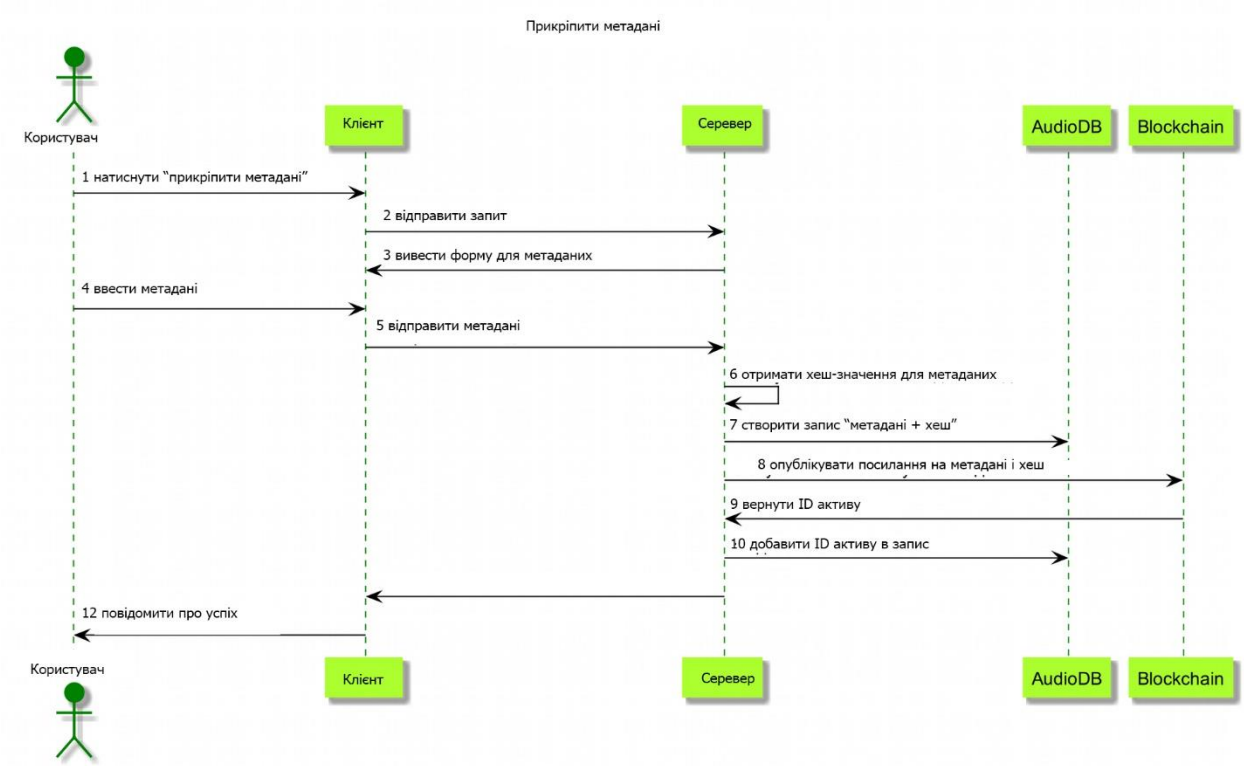

Зареєструвати власника
Натиснути "Реєстрація власника"
У якості імені користувача можна використовувати назву музичної групи, псевдонім і т.п.
Повідомити про помилку
Ввести пароль
Ввести опис
Ввести ім'я користувача
Натиснути "ОК"
існує
Отримати хеш-значення для паролю
Відправити дані на сервер
Перевірити ім'я користувача в ClientDB
не існує
Створити blockchain-гаманець для власника
Створити запис в ClientDB

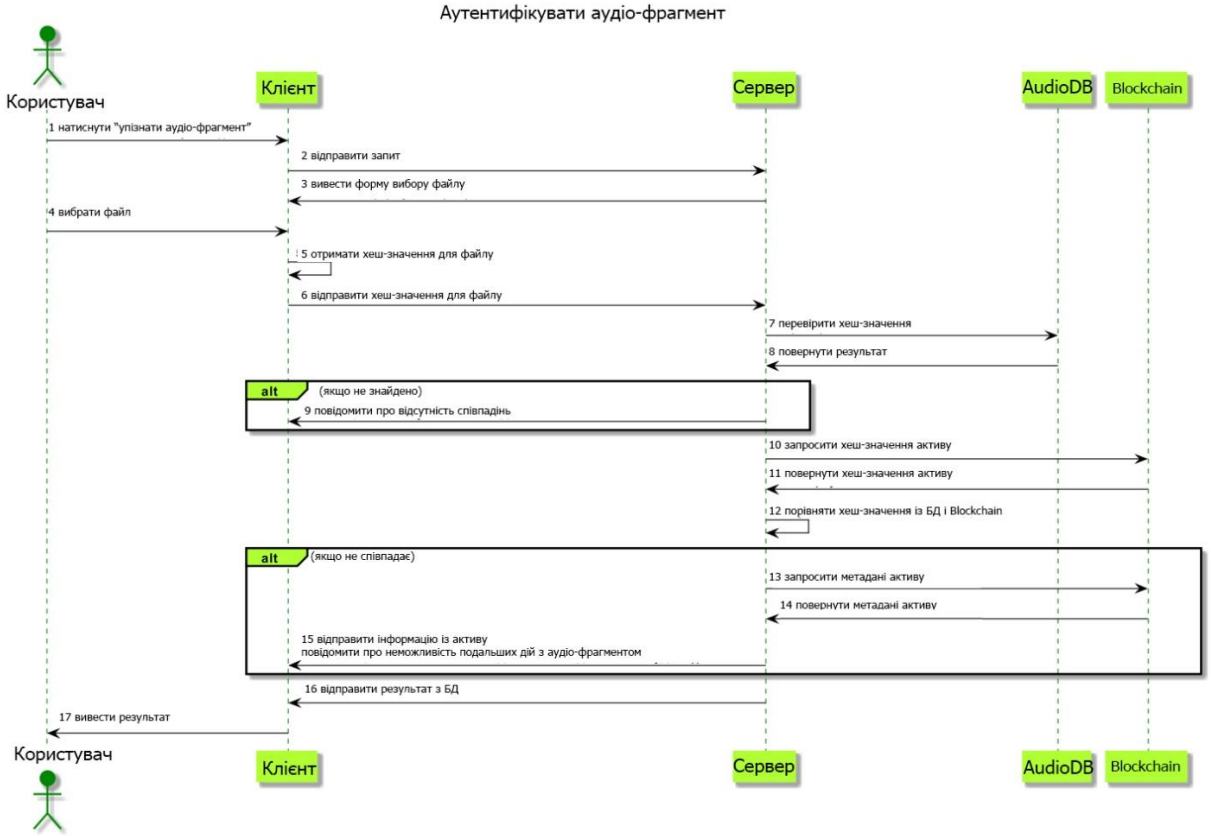

Зареєструвати запис
Натиснути "Додати запис"
Вибрати файл
Отримати хеш-значення для файлу
Відправити хеш-значення на сервер
Перевірити отримане значення в MusicDB
Спробувати ще раз
не існує
існує
Ввести опис
Повідомити про помилку
Вказати учасників
Створити запис в MusicDB
Завершити
Опублікувати данiBlockchain

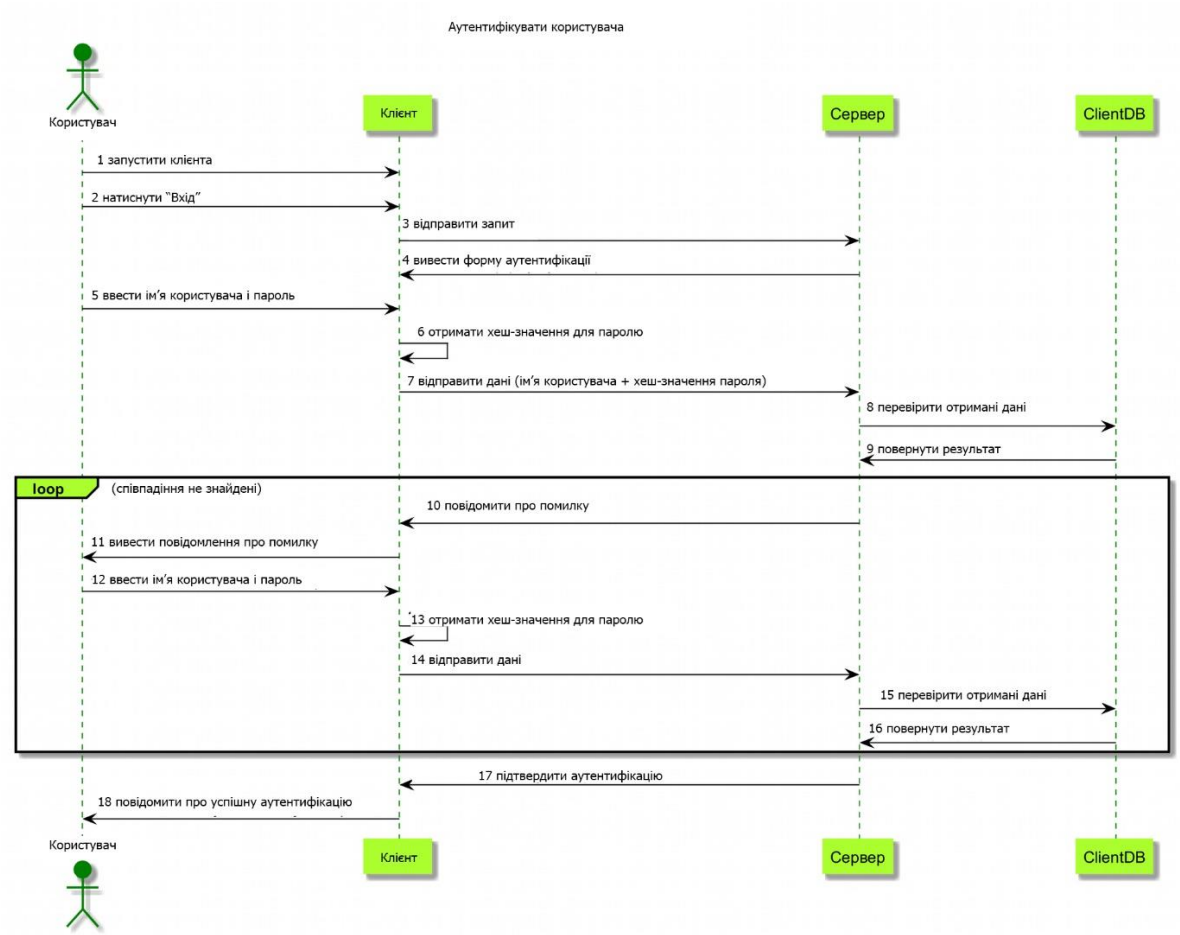

Зареєструвати користувача
Натиснути "Реєстрація користувача"
Діяти з врахуванням реальної особистості
Повідомити про помилку
Ввести персональні дані
Ввести пароль
Ввести ім'я користувача
Натиснути "ОК"
Існує
Отримати хеш-значення для паролю
Відправити дані на сервер
Перевірити ім'я користувача в ClientDB (таблиця Users)
Не існує
Створити blockchain-гаманець (адресу) для користувача
Створити запис в ClientDB (таблиця Users)

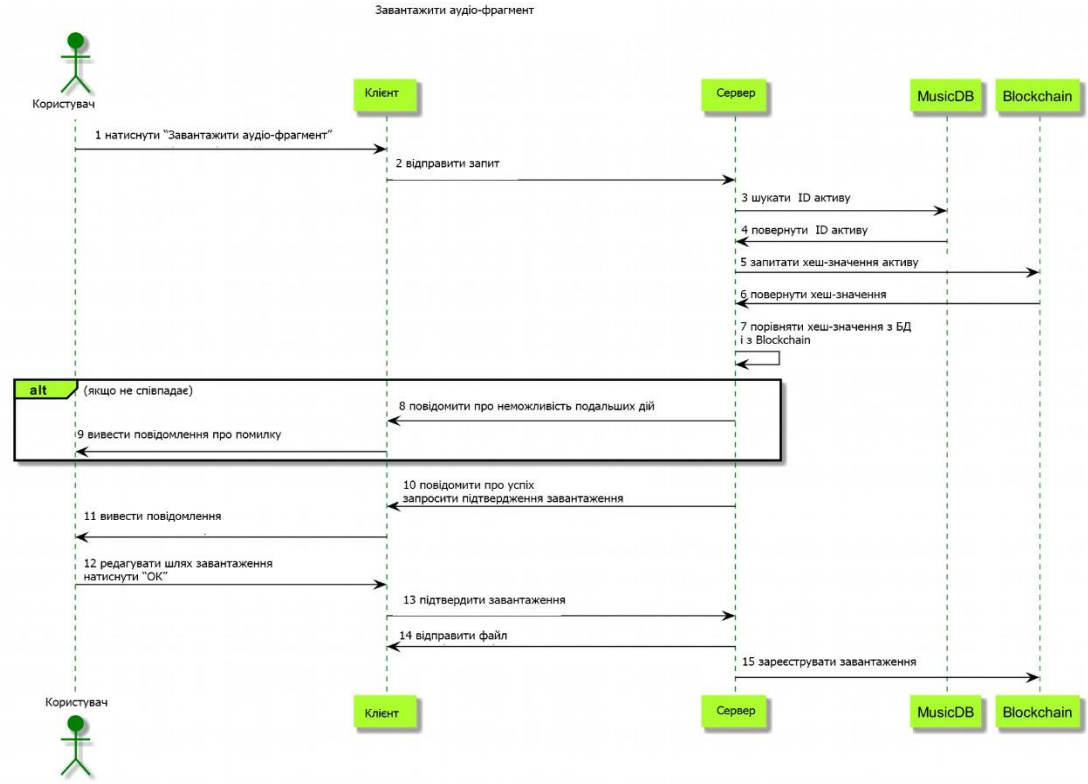
ДОДАТОК Г

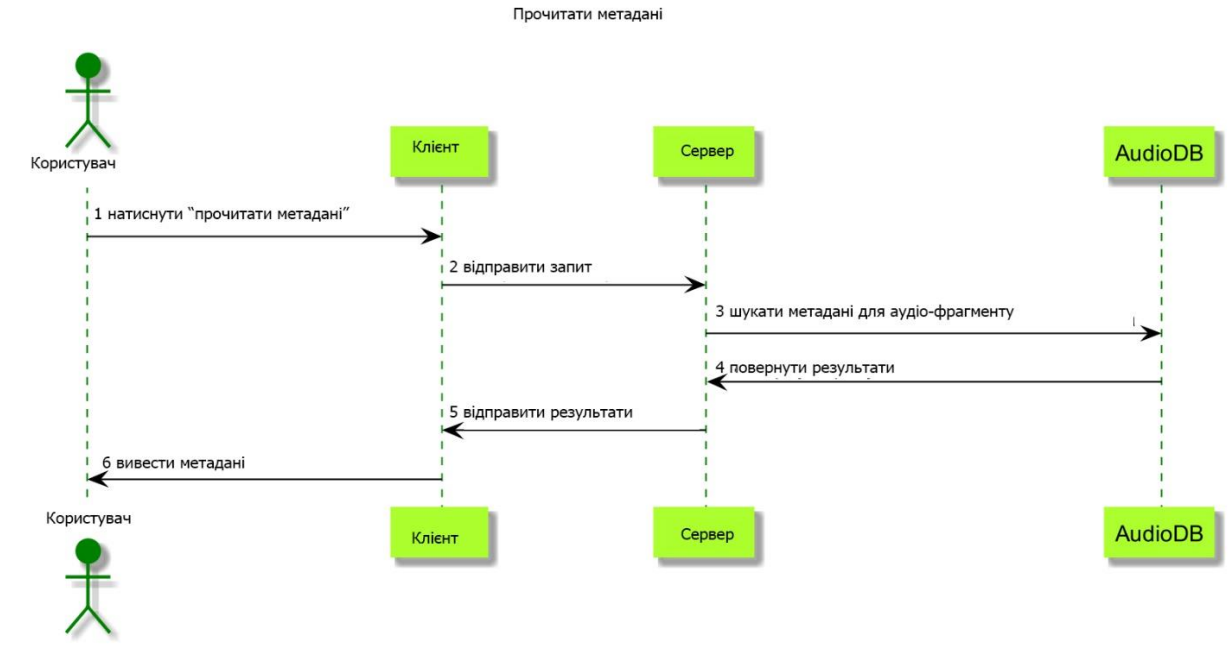
Діаграми послідовності для сценарію «Аудіо-фрагменти і метадані»

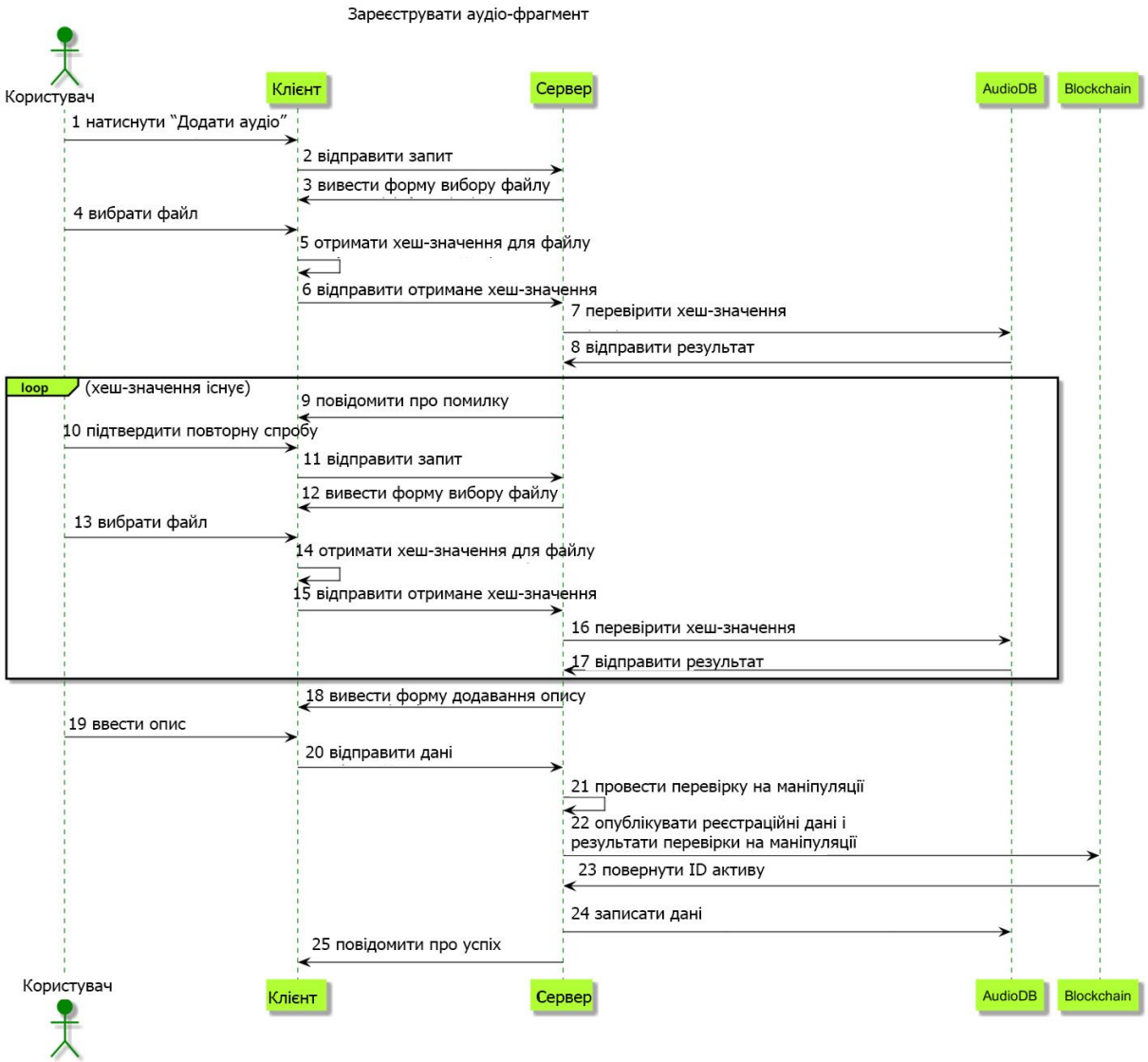


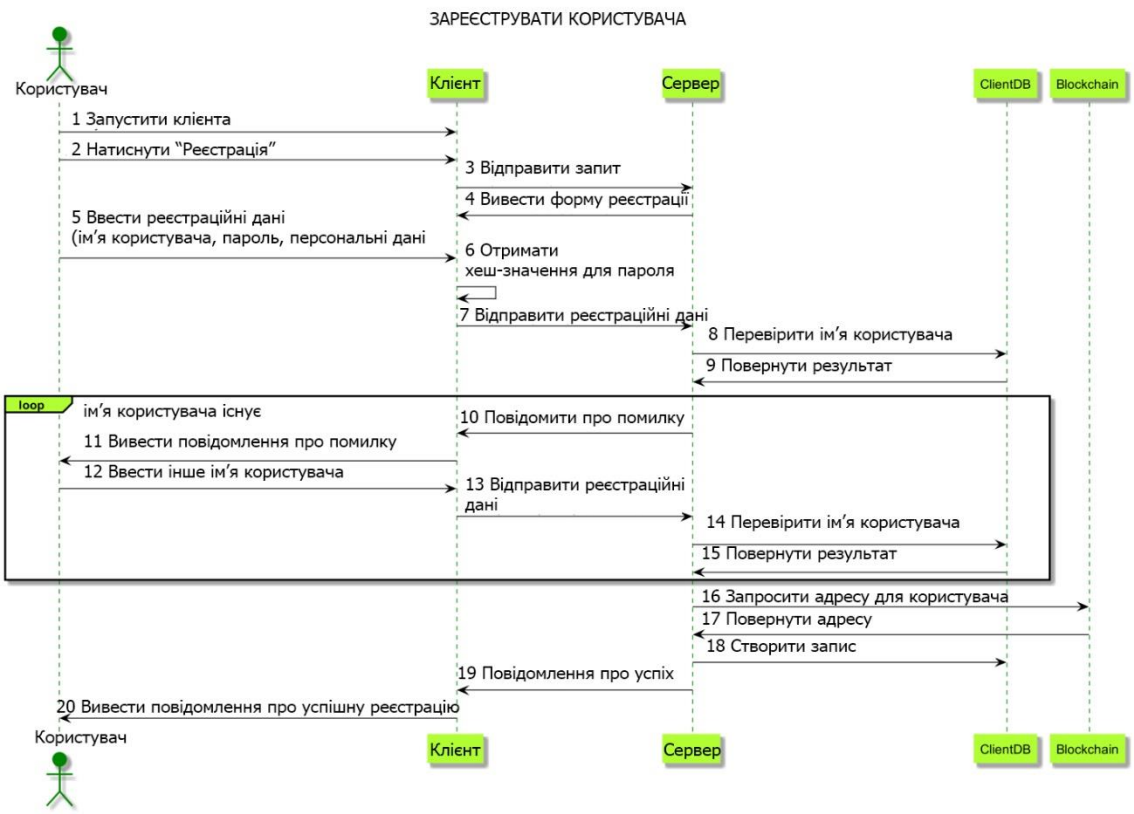


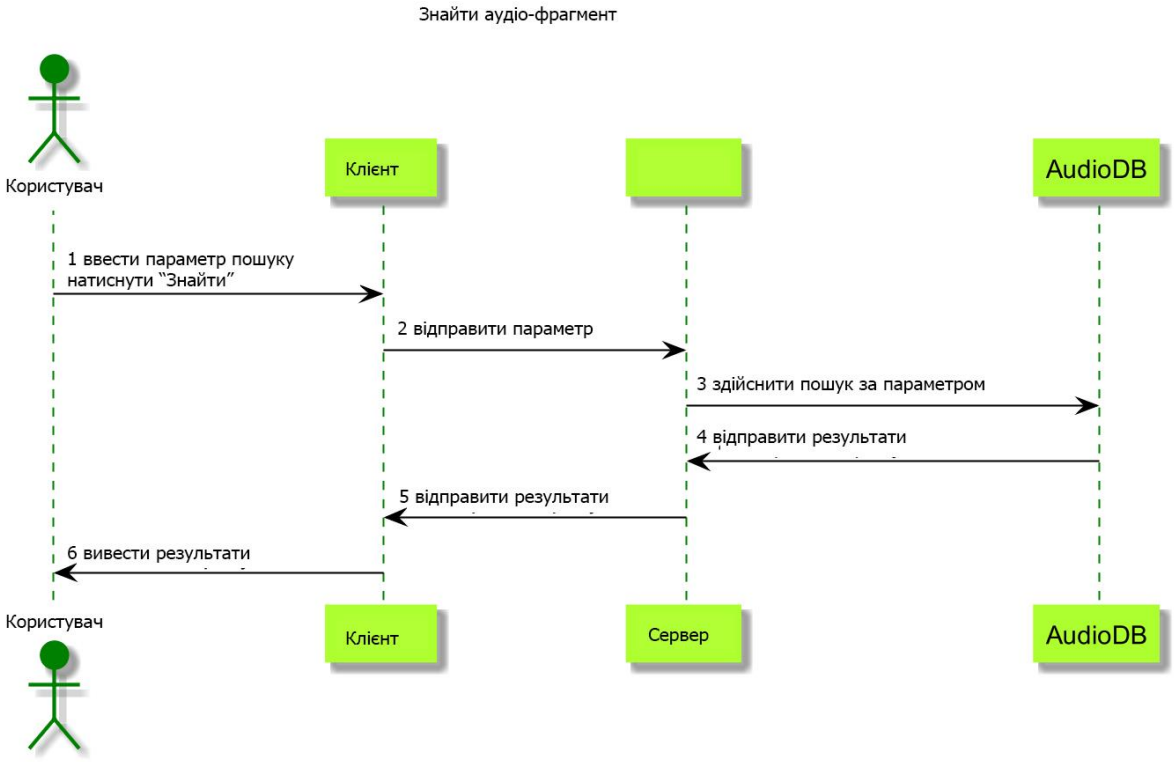

Аутентифікувати користувача
Користувач
Клієнт
Сервер
ClientDB
1 запустити клієнта
2 натиснути "Вхід"
3 відправити запит
4 вивести форму аутентифікації
5 ввести ім'я користувача і пароль
6 отримати хеш-значення для паролю
7 відправити дані (ім'я користувача + хеш-значення пароля)
8 перевірити отримані дані
9 повернути результат
loop
(співпадіння не знайдені)
10 повідомити про помилку
11 вивести повідомлення про помилку
12 ввести ім'я користувача і пароль
13 отримати хеш-значення для паролю
14 відправити дані
15 перевірити отримані дані
16 повернути результат
17 підтвердити аутентифікацію
18 повідомити про успішну аутентифікацію
Користувач
Клієнт
Сервер
ClientDB


Завантажити аудіо-фрагмент
Користувач
Клієнт
Сервер
MusicDB
Blockchain
1 натиснути "Завантажити аудіо-фрагмент"
2 відправити запит
3 шукати ID активу
4 повернути ID активу
5 запитати хеш-значення активу
6 повернути хеш-значення
7 порівняти хеш-значення з БД
і з Blockchain
alt
(якщо не співпадає)
8 повідомити про неможливість подальших дій
9 вивести повідомлення про помилку
10 повідомити про успіх
запросити підтвердження завантаження
11 вивести повідомлення
12 редагувати шлях завантаження
натиснути "ОК"
13 підтвердити завантаження
14 відправити файл
15 зареєструвати завантаження
Користувач
Клієнт
Сервер
MusicDB
Blockchain


Прочитати метадані
Користувач
Клієнт
Сервер
AudioDB
1 натиснути "прочитати метадані"
2 відправити запит
3 шукати метадані для аудіо-фрагменту
4 повернути результати
5 відправити результати
6 вивести метадані
Користувач
Клієнт
Сервер
AudioDB

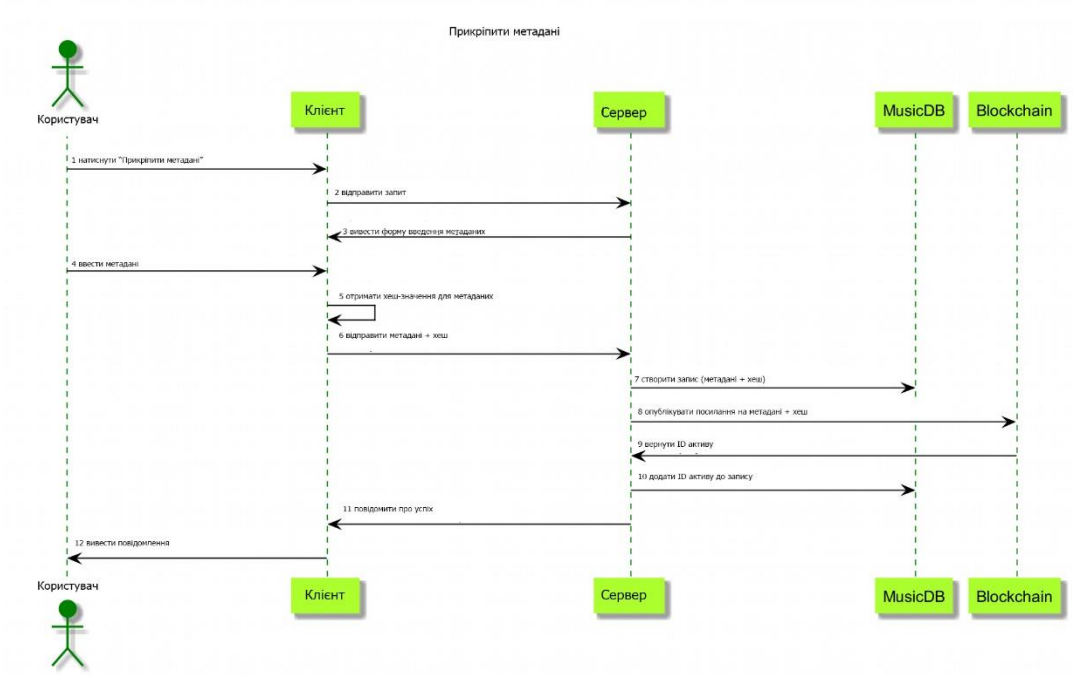

Зареєструвати аудіо-фрагмент
Користувач
Клієнт
Сервер
AudioDB
Blockchain
1 натиснути "Додати аудіо"
2 відправити запит
3 вивести форму вибору файлу
4 вибрати файл
5 отримати хеш-значення для файлу
6 відправити отримане хеш-значення
7 перевірити хеш-значення
8 відправити результат
loop
(хеш-значення існує)
9 повідомити про помилку
10 підтвердити повторну спробу
11 відправити запит
12 вивести форму вибору файлу
13 вибрати файл
14 отримати хеш-значення для файлу
15 відправити отримане хеш-значення
16 перевірити хеш-значення
17 відправити результат
18 вивести форму додавання опису
19 ввести опис
20 відправити дані
21 провести перевірку на маніпуляції
22 опублікувати реєстраційні дані і
результати перевірки на маніпуляції
23 повернути ID активу
24 записати дані
25 повідомити про успіх
Користувач
Клієнт
Сервер
AudioDB
Blockchain

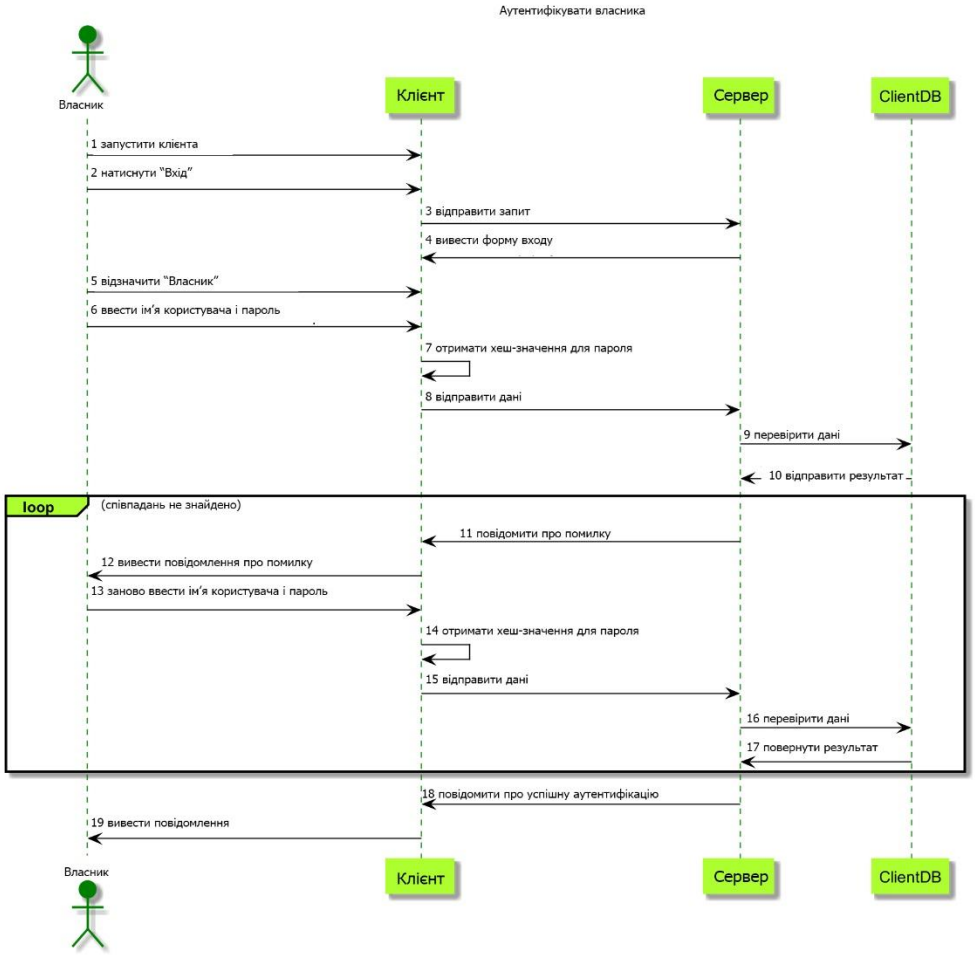

ЗАРЕЄСТРУВАТИ КОРИСТУВАЧА
Користувач
Клієнт
Сервер
ClientDB
Blockchain
1 Запустити клієнта
2 Натиснути "Реєстрація"
3 Відправити запит
4 Вивести форму реєстрації
5 Ввести реєстраційні дані
(ім'я користувача, пароль, персональні дані
6 Отримати
хеш-значення для пароля
7 Відправити реєстраційні дані
8 Перевірити ім'я користувача
9 Повернути результат
loop
ім'я користувача існує
10 Повідомити про помилку
11 Вивести повідомлення про помилку
12 Ввести інше ім'я користувача
13 Відправити реєстраційні
дані
14 Перевірити ім'я користувача
15 Повернути результат
16 Запросити адресу для користувача
17 Повернути адресу
18 Створити запис
19 Повідомлення про успіх
20 Вивести повідомлення про успішну реєстрацію
Користувач
Клієнт
Сервер
ClientDB
Blockchain

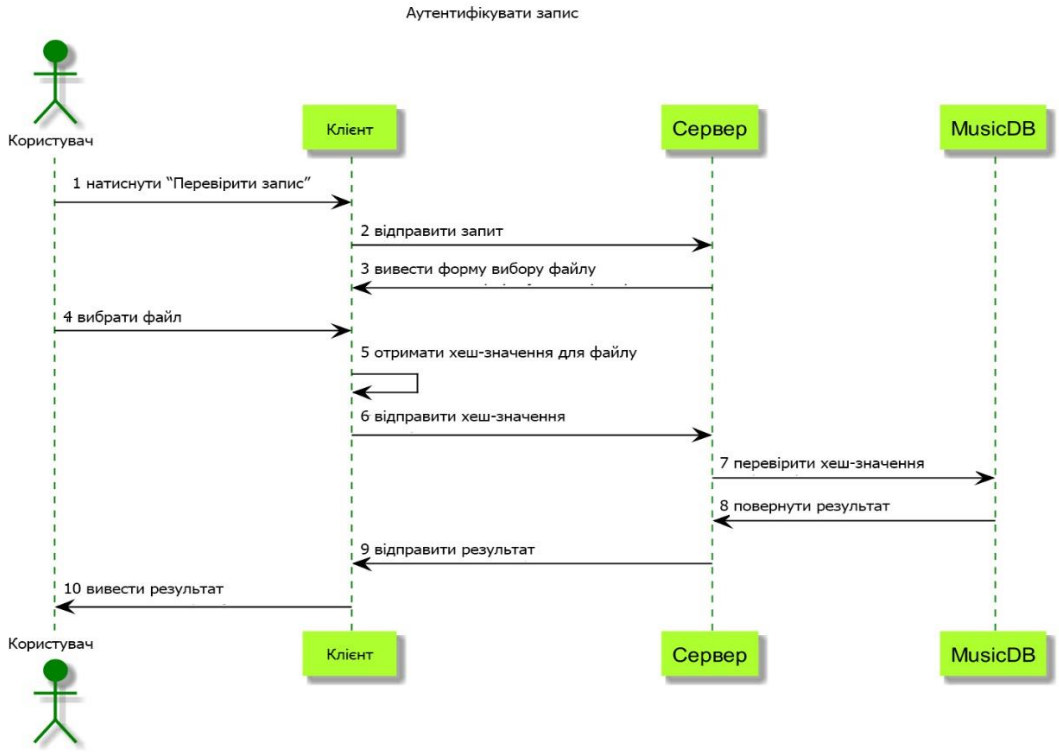

Знайти аудіо-фрагмент
Користувач
Клієнт
AudioDB
1 ввести параметр пошуку
натиснути "Знайти"
2 відправити параметр
3 здійснити пошук за параметром
4 відправити результати
5 відправити результати
6 вивести результати
Користувач
Клієнт
Сервер
AudioDB

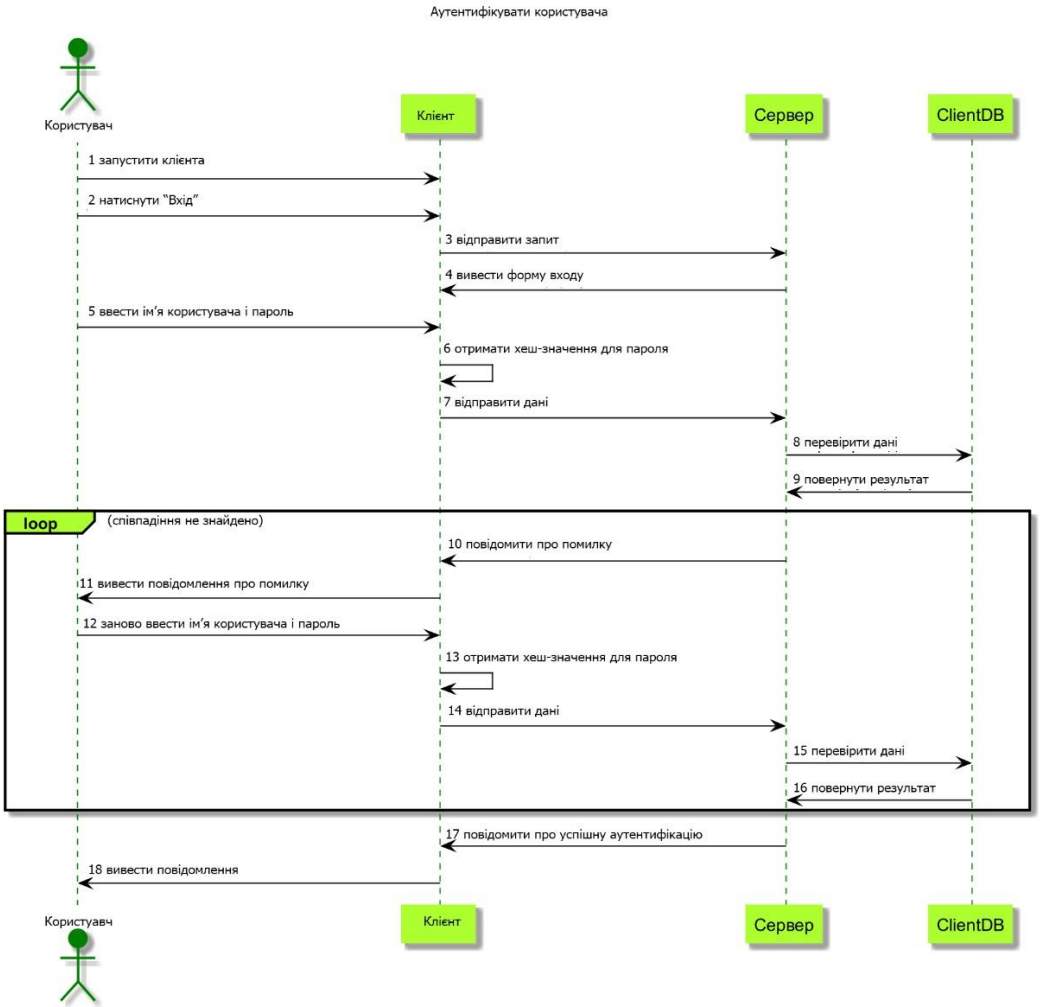
ДОДАТОК Д

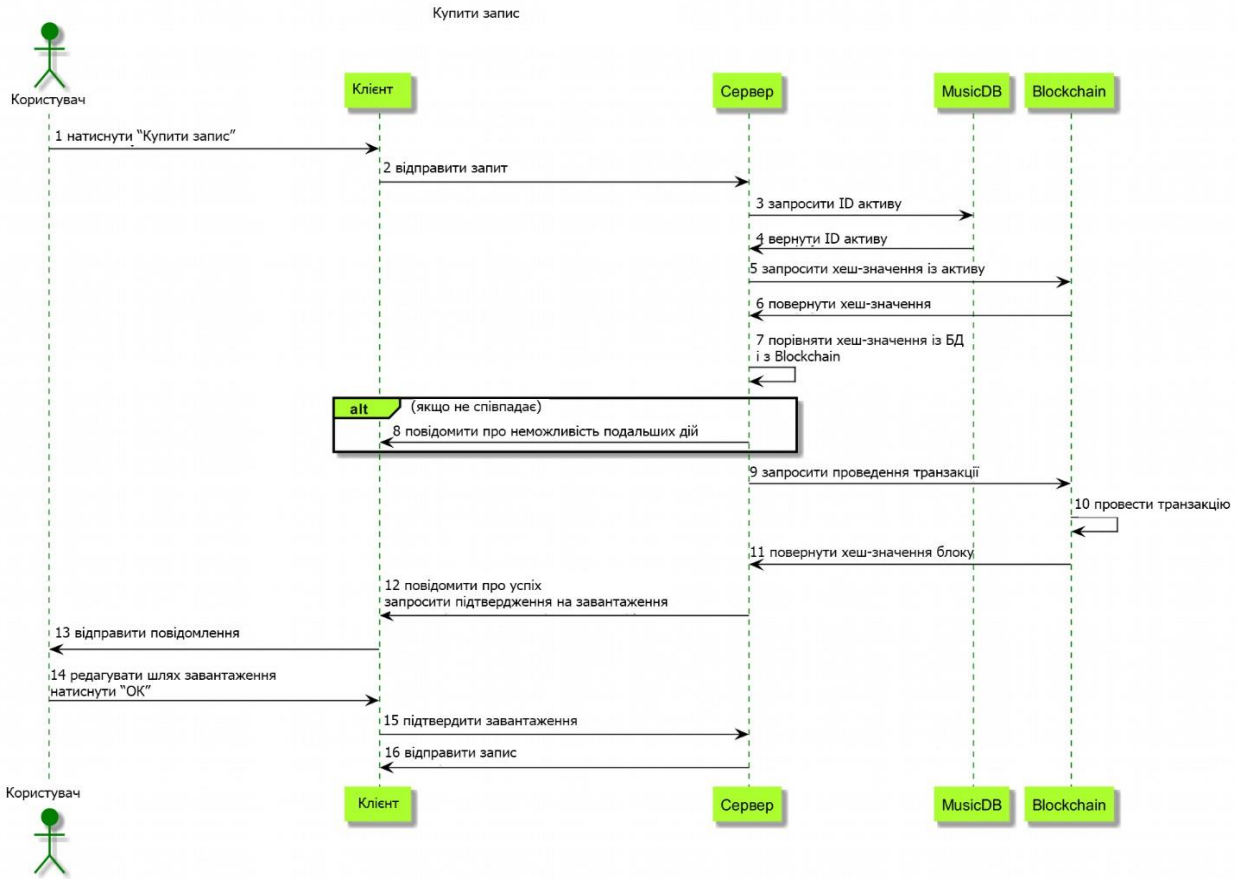
Діаграми послідовності для сценарію «Музика і метадані»

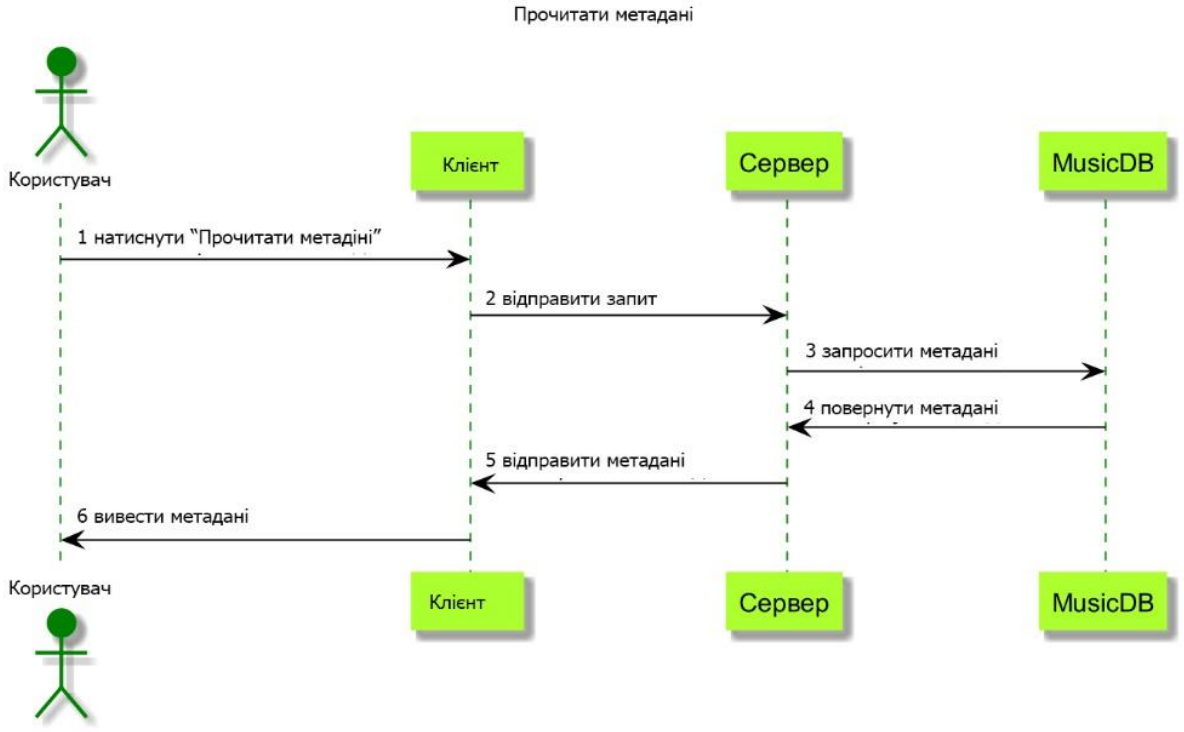


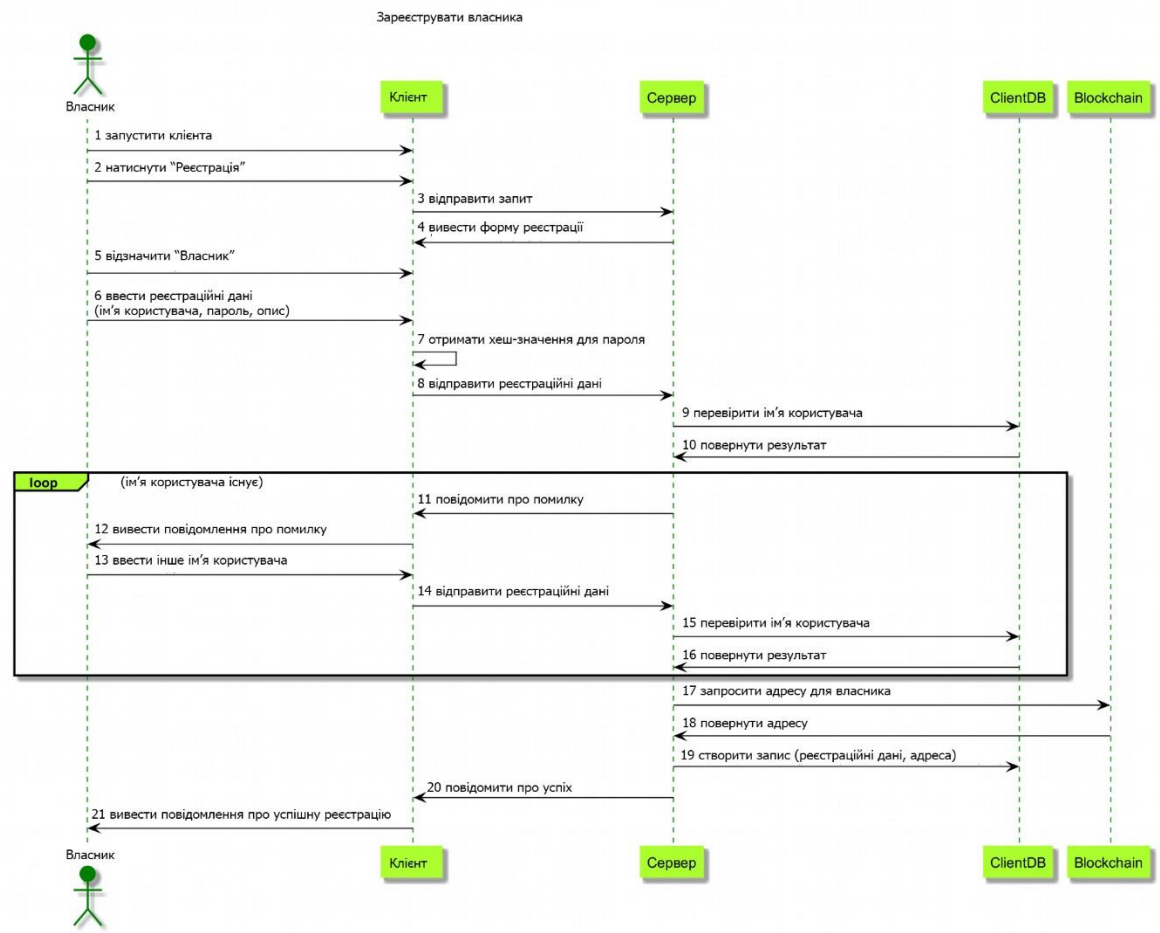


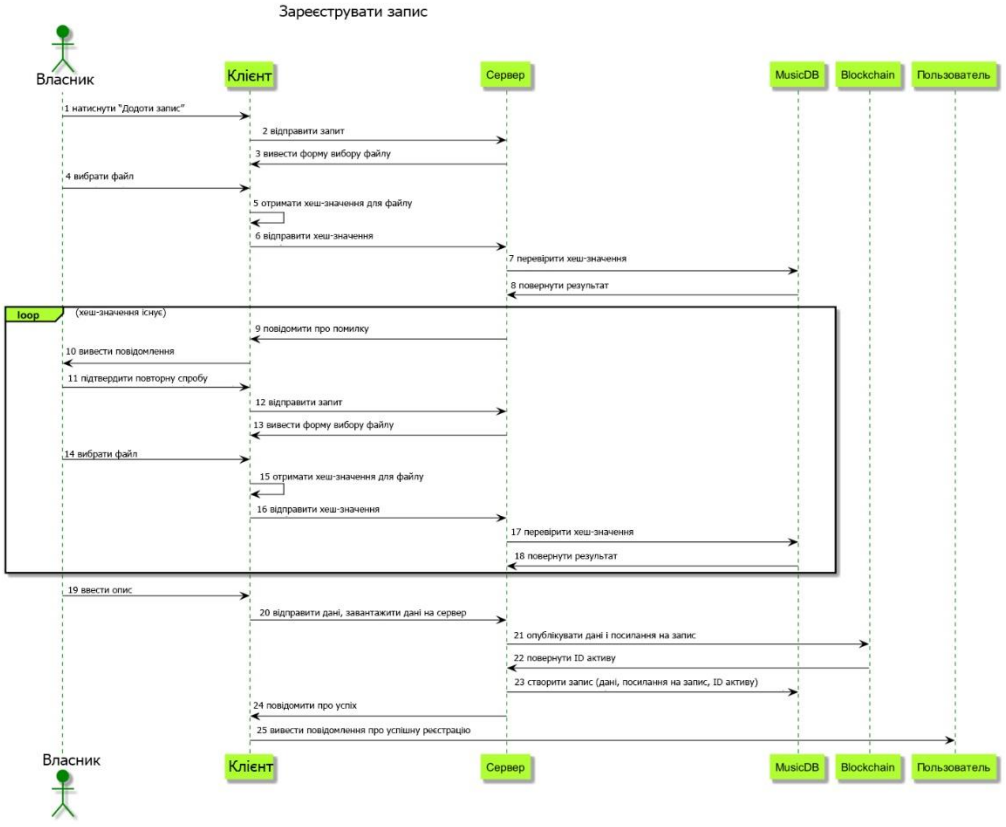

Аутентифікувати запис
Користувач
Клієнт
Сервер
MusicDB
1 натиснути "Перевірити запис"
2 відправити запит
3 вивести форму вибору файлу
4 вибрати файл
5 отримати хеш-значення для файлу
6 відправити хеш-значення
7 перевірити хеш-значення
8 повернути результат
9 відправити результат
10 вивести результат
Користувач
Клієнт
Сервер
MusicDB

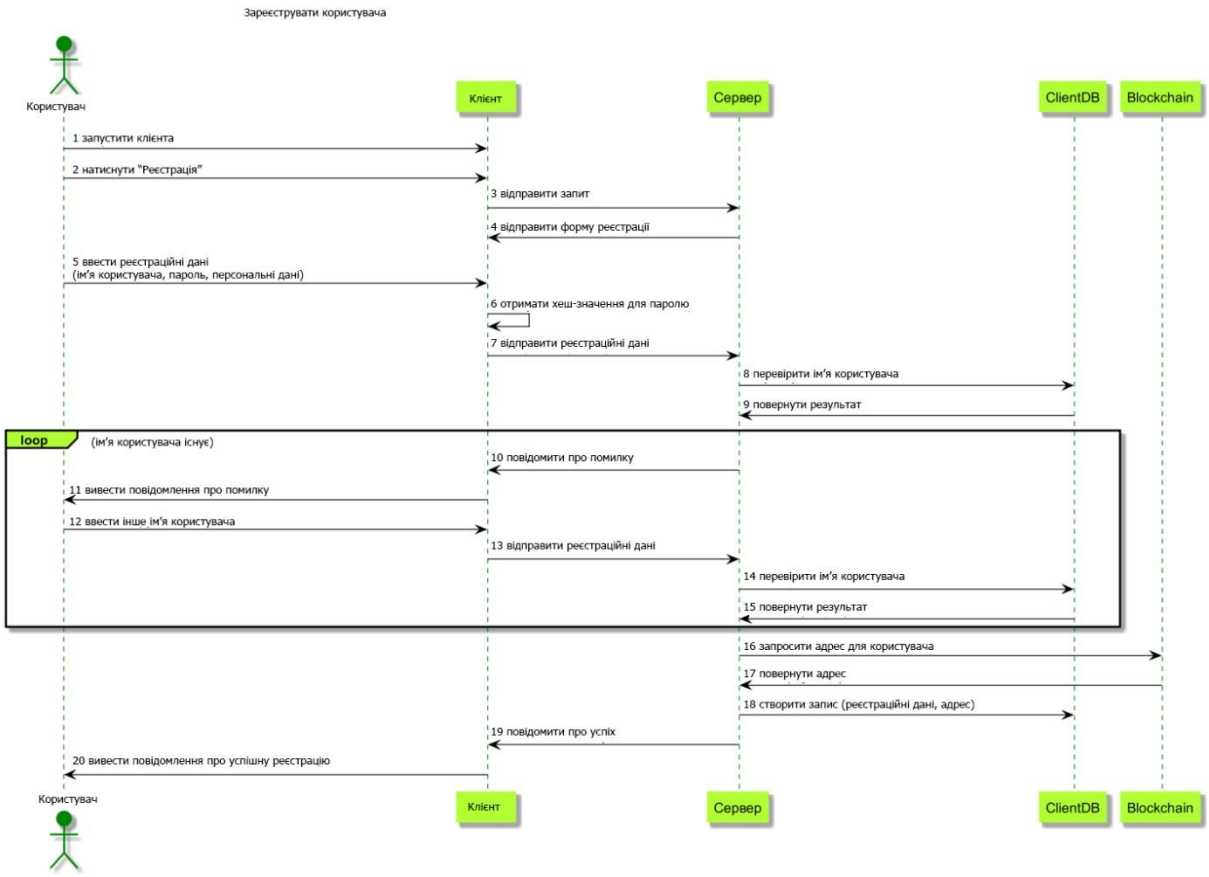

Аутентифікувати користувача
Користувач
Клієнт
Сервер
ClientDB
1 запустити клієнта
2 натиснути "Вхід"
3 відправити запит
4 вивести форму входу
5 ввести ім'я користувача і пароль
6 отримати хеш-значення для пароля
7 відправити дані
8 перевірити дані
9 повернути результат
loop
(співпадіння не знайдено)
10 повідомити про помилку
11 вивести повідомлення про помилку
12 заново ввести ім'я користувача і пароль
13 отримати хеш-значення для пароля
14 відправити дані
15 перевірити дані
16 повернути результат
17 повідомити про успішну аутентифікацію
18 вивести повідомлення
Користуавч
Клієнт
Сервер
ClientDB


Купити запис
Користувач
Клієнт
Сервер
MusicDB
Blockchain
1 натиснути "Купити запис"
2 відправити запит
3 запросити ID активу
4 вернути ID активу
5 запросити хеш-значення із активу
6 повернути хеш-значення
7 порівняти хеш-значення із БД і з Blockchain
alt
(якщо не співпадає)
8 повідомити про неможливість подальших дій
9 запросити проведення транзакції
10 провести транзакцію
11 повернути хеш-значення блоку
12 повідомити про успіх
запросити підтвердження на завантаження
13 відправити повідомлення
14 редагувати шлях завантаження
натиснути "ОК"
15 підтвердити завантаження
16 відправити запис
Користувач
Клієнт
Сервер
MusicDB
Blockchain


Прочитати метадані
Користувач
Клієнт
Сервер
MusicDB
1 натиснути "Прочитати метадіні"
2 відправити запит
3 запросити метадані
4 повернути метадані
5 відправити метадані
6 вивести метадані
Користувач
Клієнт
Сервер
MusicDB


Зареєструвати власника
Власник
Клієнт
Сервер
ClientDB
Blockchain
1 запустити клієнта
2 натиснути "Реєстрація"
3 відправити запит
4 вивести форму реєстрації
5 відзначити "Власник"
6 ввести реєстраційні дані
(ім'я користувача, пароль, опис)
7 отримати хеш-значення для пароля
8 відправити реєстраційні дані
9 перевірити ім'я користувача
10 повернути результат
loop
(ім'я користувача існує)
11 повідомити про помилку
12 вивести повідомлення про помилку
13 ввести інше ім'я користувача
14 відправити реєстраційні дані
15 перевірити ім'я користувача
16 повернути результат
17 запросити адресу для власника
18 повернути адресу
19 створити запис (реєстраційні дані, адреса)
20 повідомити про успіх
21 вивести повідомлення про успішну реєстрацію
Власник
Клієнт
Сервер
ClientDB
Blockchain


Зареєструвати запис
Власник
Клієнт
Сервер
MusicDB
Blockchain
Пользователь
1 натиснути "Додати запис"
2 відправити запит
3 вивести форму вибору файлу
4 вибрати файл
5 отримати хеш-значення для файлу
6 відправити хеш-значення
7 перевірити хеш-значення
8 повернути результат
loop
(хеш-значення існує)
9 повідомити про помилку
10 вивести повідомлення
11 підтвердити повторну спробу
12 відправити запит
13 вивести форму вибору файлу
14 вибрати файл
15 отримати хеш-значення для файлу
16 відправити хеш-значення
17 перевірити хеш-значення
18 повернути результат
19 ввести опис
20 відправити дані, завантажити дані на сервер
21 опублікувати дані і посилання на запис
22 повернути ID активу
23 створити запис (дані, посилання на запис, ID активу)
24 повідомити про успіх
25 вивести повідомлення про успішну реєстрацію
Власник
Клієнт
Сервер
MusicDB
Blockchain
Пользователь


Зареєструвати користувача
Користувач
Клієнт
Сервер
ClientDB
Blockchain
1 запустити клієнта
2 натиснути "Реєстрація"
3 відправити запит
4 відправити форму реєстрації
5 ввести реєстраційні дані
(ім'я користувача, пароль, персональні дані)
6 отримати хеш-значення для паролю
7 відправити реєстраційні дані
8 перевірити ім'я користувача
9 повернути результат
loop
(ім'я користувача існує)
10 повідомити про помилку
11 вивести повідомлення про помилку
12 ввести інше ім'я користувача
13 відправити реєстраційні дані
14 перевірити ім'я користувача
15 повернути результат
16 запросити адрес для користувача
17 повернути адрес
18 створити запис (реєстраційні дані, адрес)
19 повідомити про успіх
20 вивести повідомлення про успішну реєстрацію
Користувач
Клієнт
Сервер
ClientDB
Blockchain


Знайти запис
Користувач
Клієнт
Сервер
MusicDB
1 ввести параметр пошуку
натиснути "Знайти"
2 відправити параметр
3 здійснити пошук за параметром
4 вернути результати
5 відправити результати
6 вивести результати
Користувач
Клієнт
Сервер
MusicDB

ДОДАТОК Е

Приклади коду функцій клієнта

```
#
# Account Management
#
def reg_user (client_socket):
username = str(input('Enter username: '))
str_json = json.dumps({'method':'check_username', 'type':'u',
'username':username}) # make a string from dictionary
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # send data to server (to check if
username is available)
data = client_socket.recv(1024) # get response from server
data == data.decode() # bytes to str
if data == 'Fail': # if username already exists
answer = '0'
while True:
if answer == '0':
answer = str(input('Username already exists!\n 1.
enter another username\n 2. cancel\n'))
if answer == '1':
username = str(input('\n Enter username: ')) #
try again with another username
str_json = json.dumps({'method':'check_username',
'type':'u', 'username':username}) # make a string from
dictionary
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # check new username
data = sock.recv(1024) # get response from server
data == data.decode() # bytes to str
if data == 'Fail': # username already exists
answer = '0'
else:
break

elif answer == '2': # cancel registration, end
function early
print ('\n Registration canceled!')
return
else:
answer = input('Incorrect answer! Try again.\n')
password = string(input('Enter password: '))
pass_hash = sha256.update(password) # generate hash value for
password
while True:
answer = input('Do you want to enter your personal data?
[y/n] \n')
if answer == 'y': # to act with real identity
```

```
first_name = string(input('Enter your first name:\n'))
last_name = string(input('Enter your last name:\n'))
birthday = input('Enter your birthday
(DD.MM.YYYY):\n')
break
elif answer == 'n': # to act pseudonym
first_name = None
last_name = None
birthday = None
break
else: # for other answers
answer = input('Incorrect answer! Try again.\n')
str_json = json.dumps({'method':'reg_user', 'username':username,
'pass_hash':pass_hash, 'first_name':first_name,
'last_name':last_name,
'birthday':birthday}) # make a string from dictionary
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # send registration data to server
data = client_socket.recv(1024) # get response from server
data == data.decode() # bytes to str
if data == 'Fail': # if registration failed
return False
else: # if registration complete
print(data) # print blockchain address
return True
```

```
def reg_rights_holder (client_socket):
username = str(input('Enter username: '))
str_json = json.dumps({'method':'check_username', 'type':'rh',
'username':username}) # make a string from dictionary
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # send data to server (to check if
username is available)
data = client_socket.recv(1024) # get response from server
data == data.decode() # bytes to str
if data == 'Fail' # if username already exists
answer = '0'
while True:
if answer == '0':
answer = str(input('Username already exists!\n 1.
enter another username\n 2. cancel\n'))
if answer == '1':
username = str(input('\n Enter username: ')) #
try again with another username
str_json = json.dumps({'method':'check_username',
'type':'rh', 'username':username}) # make a string from
dictionary
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # check new username
data = sock.recv(1024) # get response from server
```

```
data == data.decode() # bytes to str
if data == 'Fail': # if username already exists
answer = '0'
else:
break
elif answer == '2': # cancel registration, end
function early
print ('\n Registration canceled')
return
else:
answer = input('Incorrect answer! Try again.\n')
password = str(input('Enter password: '))
pass_hash = sha256.update(password) # generate hash value for
password

description = str(input('Enter description'))
str_json = json.dumps({'method':'reg_rights_holder',
'username':username, 'pass_hash':pass_hash,
'description':description}) # make a string from dictionary
data = str_json.encode("utf-8") # str to bytes
client_socket.send(data) # sendregistration data to server
data = client_socket.recv(1024) # get response from server
data == data.decode() # bytes to str
if data == 'Fail': # registration failed
return False
else: # registration complete
print(data) # print blockchain address
return True
def auth_user (client_socket):
username = str(input('Enter username'))
password = str(input('Enter password'))
pass_hash = sha256.update(password) # generate hash value for
password
str_json = json.dumps({'method':'auth_all', 'type':'u',
'username':username, 'pass_hash':pass_hash}) # make a string
from
dictionary
data = str_json.encode("utf-8") # str to bytes
client_socket.send(data) # send authentication data to server
if data == 'Fail': # if authentication failed
answer = '0'
while True:
if answer == '0':
answer = str(input('Authentication failed!\n 1.
try again\n 2. cancel\n'))
if answer == '1': # try again
username = str(input('\n Enter username: '))
password = str(input('Enter password'))
pass_hash = sha256.update(password) # generate
hash value for password
```

```
str_json = json.dumps({'method':'auth_all',
'type':'u', 'username':username, 'pass_hash':pass_hash}) # make
a
string from dictionary

data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # send data to server
data = client_socket.recv(1024) # get response
from server
data == data.decode() # bytes to str
if data == 'Fail': # authentication failed
answer = '0'
else:
break
elif answer == '2': # cancel authentication, end
function early
print ('\n Authentication canceled!')
return
else:
answer = input('Incorrect answer! Try again.\n')
print ('Hello, ', username)
def auth_rights_holder (client_socket):
username = str(input('Enter username'))
password = str(input('Enter password'))
pass_hash = sha256.update(password) # generate hash value for
password
str_json = json.dumps({'method':'auth_all', 'type':'rh',
'username':username, 'pass_hash':pass_hash}) # make a string
from
dictionary
data = str_json.encode("utf-8") # str to bytes
client_socket.send(data) # send authentication data to server
if data == 'Fail': # if authentication failed
answer = '0'
while True:
if answer == '0':
answer = str(input('Wrong username or password!\n
1. try again\n 2. cancel\n'))
if answer == '1': # try again
username = str(input('\n Enter username: '))
password = str(input('Enter password'))

pass_hash = sha256.update(password) # generate
hash value for password
str_json = json.dumps({'method':'auth_all',
'type':'rh', 'username':username, 'pass_hash':pass_hash}) # make
a
string from dictionary
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # send authentication
```

```
data to server
data = client_socket.recv(1024) # get response
from server
data == data.decode() # bytes to str
if data == 'Fail': # if authentication failed
answer = '0'
else:
break
elif answer == '2': # cancel authentication, end
function early
print ('\n Authentication canceled!')
return False
else:
answer = input('Incorrect answer! Try again.\n')
print ('Hello, ', username)
return True
def log_out(client_socket):
client_socket.close()
#
# Music Management
#
def reg_song (client_socket):
print ('Enter full path to file!\n')
with open(sys.argv[1], 'rb') as f:
while True:
data = f.read(BUF_SIZE)
if not data:
break

sha256.update(data)
file_hash = sha256.hexdigest()
str_json = json.dumps({'method':'auth_song', 'hash':file_hash})
#
make a string from dictionary
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # send file hash to server for checking
data = client_socket.recv(1024)
data == data.decode()
if data == '':
print('This file is already registered!')
return
else:
print('Enter song description!\n')
song_title = str(input('Song title:\n'))
performer = str(input('Performers name:\n'))
album = str(input('Album name:\n'))
year = str(input('Album year:\n'))
str_json = json.dumps({'method':'reg_song',
'hash':file_hash, 'title':song_title, 'performer':performer,
'album':album, 'year':year})
```

```
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # send registration data to server
def auth_song (client_socket):
print ('Enter full path to file!\n')
with open(sys.argv[1], 'rb') as f:
while True:
data = f.read(BUF_SIZE)
if not data:
break
sha256.update(data)
file_hash = sha256.hexdigest()
str_json = json.dumps({'method':'auth_song', 'hash':file_hash})
#
make a string from dictionary
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data) # send file hash to server
data = client_socket.recv(1024)

data == data.decode()
print (data)
def search(client_socket):
parameter = string(input('Enter search parameter!\n'))
str_json = json.dumps({'method':'search',
'parameter':parameter})
# make a string from dictionary
data = str_json.encode("utf-8") # string to bytes
client_socket.send(data)
data = sock.recv(1024) # get data from server
data = data.decode() # bytes to string
print (data)
```

Додаток Ж

Приклади коду функцій сервера

```
#
# Account Management
#
def check_username(username, type):
conn_client = sqlite3.connect('client.db') # connect to data
base
with user's and rights holder's data
cur_client = conn_client.cursor() # create cursor for connection
to data base with user's and rights holder's data
if type == 'u':
cur_client.execute("SELECT * FROM users WHERE username=?",
username) # search for username in data base
elif type == 'rh':
cur_client.execute("SELECT * FROM rights_holders WHERE
username=?", username) # search for username in data base
check = cur_client.fetchone() # get result row
conn_client.close()
if check != None: # username already exists
return False
else:
return True
def reg_user (username, pass_hash, first_name, last_name,
birthday)
address = api.getnewaddress() # get multichain address for user
insert = (username, pass_hash, address, first_name, last_name,
birthday)
try:
conn_client = sqlite3.connect('client.db') # connect to data
base with user's and rights holder's data
cur_client = conn_client.cursor() # create cursor for
connection to data base with user's and rights holder's data
cur_client.execute("INSERT INTO users
VALUES(?, ?, ?, ?, ?, ?)", insert) # create record in data base
except lite.Error, e:
if conn_client:
conn_client.rollback()
print "Error %s:" % e.args[0]
return False
finally:
if conn_client:
conn_client.commit() # commit changes, make them
visible for other connections
conn_client.close()
return address
def reg_rights_holder (username, pass_hash, description)
```

```
address = api.getnewaddress() # get multichain address for
rights
holder
insert = (username, pass_hash, address, description)
try:
conn_client = sqlite3.connect('client.db') # connect to data
base with user's and rights holder's data
cur_client = conn_client.cursor() # create cursor for
connection to data base with user's and rights holder's data
cur_client.execute("INSERT INTO users VALUES(?, ?, ?, ?,)",
insert) # create record in data base
except lite.Error, e:
if conn_client:
conn_client.rollback()
print "Error %s:" % e.args[0]
return False
finally:
if conn_client:
conn_client.commit() # commit changes, make them
visible for other connections
conn_client.close()
return address
def auth_all(username, pass_hash, type):
conn_client = sqlite3.connect('client.db') # connect to data
base
with user's and rights holder's data
cur_client = conn_client.cursor() # create cursor for connection
to data base with user's and rights holder's data
if type == 'u': # authenticate user
cur_client.execute("SELECT * FROM users WHERE username=? AND
pass_hash=?", (username, pass_hash))
if type == 'rh': # authenticate rights holder
cur_client.execute("SELECT * FROM rights_holders WHERE
username=? AND pass_hash=?", (username, pass_hash))
check = cur_client.fetchone()
conn_client.close()
if check == None: # if no matches found
return False
else:
return True
#
# Music Management
#
def reg_song(song_title, performer, album, year, file_hash):
asset_ID = api.issue({"asset_name":"song", "title":song_title,
"performer":performer, "album":album, "year":year,
"hash":file_hash})#
not sure, how to use this command
insert = (song_title, performer, album, year, file_hash,
```

```
asset_ID)conn_music = sqlite3.connect('music.db') # connect to
music
data base
try:
cur_music = conn_music.cursor() #create cursor for
connection to music data base
cur_music.execute("INSERT INTO songs
VALUES(?, ?, ?, ?, ?, ?)", insert)
except lite.Error, e:
if conn_client:
conn_client.rollback()
print "Error %s:" % e.args[0]
return False
finally:
if conn_client
conn_music.commit()
conn_music.close()
return True
def auth_song(file_hash):
conn_music = sqlite3.connect('music.db') # connect to music data
base
cur_music = conn_music.cursor() #create cursor for connection to
music data base
cur_music.execute("SELECT * FROM songs WHERE hash=?",
(file_hash))
result = cur_music.fetchone()
conn_music.close()
return result
def search(parameter):
conn_music = sqlite3.connect('music.db') # connect to music data
base
cur_music = conn_music.cursor() #create cursor for connection to
music data base
# search in title
cur_music.execute("SELECT title, performer, album, hash FROM
songs WHERE title=?", (parameters)) # search in data base
results_title = 'In song title:\n'
for row in cur_music: # go thru results
results_title = results_title + str('Song title: ' +
str(row[0]) + ' Performer: ' + row[1] + ' Album: ' + row[2] + '
Hash:
' + row[3] + '\n') # save results
results_title = results_title + '\n\n'
# search in performer
cur_music.execute("SELECT title, performer, album, hash FROM
songs WHERE performer=?", (parameters)) # search in data base
results_performer = 'In performer:\n'
for row in cur_music: # go thru results
results_performer = results_performer + str('Song title: ' +
```

```
str(row[0]) + ' Performer: ' + row[1] + ' Album: ' + row[2] + '
Hash:
' + row[3] + '\n') # save results
results_performer = results_performer + '\n\n'
# search in album
cur_music.execute("SELECT title, performer, album, hash FROM
songs WHERE album=?", (parameters)) # search in data base
results_album = 'In album:\n'
for row in cur_music: # go thru results
results_performer = results_performer + str('Song title: ' +
str(row[0]) + ' Performer: ' + row[1] + ' Album: ' + row[2] + '
Hash:
' + row[3] + '\n') # save results
results_performer = results_performer + '\n\n'
# collect results
results = '*** Search results ***\n' + results_title + '\n' +
results_performer + '\n' + results_album
conn_music.close()
return results
```

УДК 004.043
О. Копанецький
Тернопільський національний технічний університет імені Івана Пулюя

ВИКОРИСТАННЯ ТЕХНОЛОГІЇ BLOCKCHAIN ДЛЯ УПРАВЛІННЯ МЕХАНІЗМОМ АВТОРСЬКИХ ПРАВ НА АУДІОФАЙЛИ

UDC 004.043
O. Kopanetskii
(Ternopil Ivan Puluj National Technical University, Ukraine)

USE OF BLOCKCHAIN TECHNOLOGY TO CONTROL THE COPYRIGHT MECHANISM OF AUDIO FILES

До головних проблем музичної індустрії відносяться прозорість, ясність і функціонування механізму поширення ліцензій. Технологія Blockchain може сприяти раціоналізації механізму авторських прав і забезпечити прозорість і чесність оплати праці музикантів і інших правовласників [1]. З моменту початку розповсюдження музики в мережі Інтернет, музична індустрія веде пошуки шляхів монетизації цифрових музичних записів. Існуючі застарілі бази даних авторських прав і система збору ліцензійного мита в цілому ускладнює отримання музики з легітимних джерел. Завдяки використанню технології Blockchain і смарт-контрактів для створення загальної і по-справжньому децентралізованої бази даних авторських прав на музичні записи існує можливість забезпечити миттєве і повністю прозоре перерахування ліцензійного мита, включаючи розподіл коштів співавторам, продюсерам, технічним партнерам, видавництвам і лейблам. Смарт-контракти можуть забезпечити при кожній оплаті музичного запису автоматичний розподіл коштів відповідно до зазначених умов, а на рахунку кожного з учасників миттєво відображатиметься надходження коштів.

З використанням технології Blockchain спроектовано і реалізовано прототип програмного сервісу, який призначений для управління механізмом авторських прав на аудіо файли. Наведено основні характеристики технології Blockchain, зокрема її використання у медіа сфері, для боротьби з піратством, полегшення контролю за контентом та управління правами користувачів. Описано можливості смарт-контракту як способу обміну цифровими цінностями в Blockchain. Розглянуто особливості та моделі архітектури клієнт-сервер, яка використана в розробці сервісу. В роботі застосована технологія Multichain, яка дозволяє користувачу контролювати приватність і публічність ланцюга, цільовий час блоків, способи взаємодії сторін, вибір цих сторін, максимальний розмір блоку і метадані, які можна включати в транзакції. Для даної технології існує оболонка для виклику API-команд в програмному коді для мови Python. Multichain підтримує велику кількість різних API-команд, необхідних для реалізації сервісу. Описано основні системні вимоги до сервісу, шаблони, розроблені для опису вимог, а також самі вимоги за сценаріями «Музика і метадані» та «Аудіо-фрагменти і метадані». Розглянуті сценарії впровадження технології Blockchain для реєстрації та аутентифікації аудіоматеріалу з метою розповсюдження. Запропоновано варіанти використання (use cases), які є засобом опису функціональних вимог до розробки. Побудовано та описано діаграми варіантів використання, компонентів, діяльності та послідовності. Дані діаграми в різних аспектах ілюструють склад і поведінку розроблюваних програмних елементів сервісу. Програмне забезпечення, яке використовується для створення сервісу - мова Python та СКБД SQLite.

У разі повноцінної реалізації сервісу очевидно поліпшення рівня захисту авторських прав та забезпечення більшої прозорості і доступності правової інформації, а також можливість зібрати воєдино і перевірити на наявність маніпуляцій безліч записів для подальшого використання, що запобігає завантаженню дублікатів файлів.

Література

1. Blockchain Applications in Music. [Електронний ресурс] – Режим доступу: https://www.blockchaintechnologies.com/applications/music/ – (дата звернення: 05.11.2019)