ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ПРОГРАМНИЙ ІНТЕРФЕЙС НА ОСНОВІ БЛОКЧЕЙНУ ДЛЯ ЗАХИСТУ ТА ЗБЕРЕЖЕННЯ МЕДІА ДОКАЗІВ ВІЙСЬКОВИХ ЗЛОЧИНІВ»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ ____*Назар КОЖІН*__
(*підпис*) (*Ім'я, ПРІЗВИЩЕ здобувача*)

Виконав:
здобувач вищої освіти _ Назар КОЖІН___
група ІСДМ-61

Керівник: Віктор САГАЙДАК
науковий ступінь, доктор філософії
вчене звання

Рецензент:
науковий ступінь, _____
вчене звання Ім'я, ПРІЗВИЩЕ

Київ – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ
_____Кожіну Назару Олексійовичу_____
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: «Програмний інтерфейс на основі блокчейну для захисту та збереження медіа доказів військових злочинів»

керівник кваліфікаційної роботи ___Віктор САГАЙДАК_, доктор філософії
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «15» жовтня 2024 року № 320.

2. Строк подання кваліфікаційної роботи: 27 грудня 2024 року.

3. Вихідні дані до кваліфікаційної роботи:
алгоритми забезпечення незмінності даних ;
роль блокчейна у збереженні незмінних даних ;
задачі індексації даних ;
прототип програмного інтерфейсу для взаємодії з блокчейном ;
науково-технічна література з питань, пов'язаних з наукою про дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд існуючих технологій_____

2 Порівняльний аналіз існуючих технологій_____
3. Конструювання та аналіз розробленого рішення_____.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання: 15 жовтня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	15.10 – 23.10.24	
2	Дослідження наявних технологій	24.10 – 01.11.24	
3	Аналіз наявних алгоритмів	02.11 – 07.11.24	
4	Моделювання Програмного Забезпечення	08.11 – 16.11.24	
5	Розробка програмного забезпечення	17.11 – 21.11.24	
6	Тестування та аналіз розробки	22.11 – 11.12.24	
7	Оформлення роботи: вступ, висновки, реферат	12.12 – 20.12.24	
8	Розробка демонстраційних матеріалів	21.12 – 25.12.24	

Здобувач вищої освіти _____ ____Назар КОЖІН___
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник роботи
кваліфікаційної роботи _____ _Віктор САГАЙДАК_
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: ___76__ стор., ___24__ рис., __10__ табл., __40__ джерел.

Мета роботи – дослідження можливостей використання блокчейн-технологій для реєстрації та захисту доказів військових злочинів.

Об'єкт дослідження – програмний інтерфейс, що дозволяє зберігати та захищати медіа докази військових злочинів.

Предмет дослідження – процес реєстрації та підтвердження правдивості медіа доказів військових злочинів.

Короткий зміст роботи: У роботі проведено аналіз технології блокчейн, як технологічного рішення, та окремих компонентів на яких будується ця технологія, для визначення доцільності використання технології блокчейн задля збереження медіа доказів військових злочинів. У процесі дослідження було розроблено функціональний програмний інтерфейс, що спрощує роботу з технологією блокчейн та даними, що у ньому зареєстровані. Також проведено аналіз якості розробленого програмного рішення шляхом статичного моніторингу програмного коду.

КЛЮЧОВІ СЛОВА: BLOCKCHAIN, ВІЙСЬКОВІ ЗЛОЧИНИ, МЕДІА ДОКАЗИ, ІНДЕКСАЦІЯ ДАНИХ, ПРОГРАМНИЙ ІНТЕРФЕЙС, КРИПТОГРАФІЯ

ABSTRACT

Text part of the master's qualification work: ___76__ pages, ___24__ pictures, __10__ tables, __40__ sources.

The purpose of the work is to explore the possibilities of using blockchain technologies for registering and protecting evidence of war crimes.

Object of research – a software interface that allows storing and protecting media evidence of war crimes.

Subject of research – the process of registering and verifying the authenticity of media evidence of war crimes.

Summary of the work: This research analyzes blockchain technology as a technological solution, as well as individual components on which this technology is based, in order to determine the feasibility of using blockchain for the preservation of media evidence of war crimes. In the course of the study, a functional software interface was developed, simplifying work with blockchain technology and the data registered within it. Additionally, the quality of the developed software solution was evaluated through static code monitoring.

KEYWORDS: BLOCKCHAIN, WAR CRIMES, MEDIA EVIDENCE, DATA INDEXING, SOFTWARE INTERFACE, CRYPTOGRAPHY

ЗМІСТ

ВСТУП

Актуальність теми. В умовах сучасного світу, інформаційні технології відіграють ключову роль у всіх сферах життя, питання безпеки та збереження інформації не втрачає актуальність. Це стосується зокрема збереження медіа доказів військових злочинів, які є критично важливими для забезпечення правосуддя та притягнення винних до відповідальності. Війни, що відбуваються в різних куточках світу, зокрема військова агресія росії проти України, призводять до численних порушень міжнародного гуманітарного права та прав людини. Без структурування та збереження доказів злочинів такого типу складно уявити встановлення справедливості, настання наслідків правових порушень та панування миру.

Однак стандартні методи зберігання інформації, такі як централізовані бази даних, мають низку недоліків. Вони можуть бути піддані хакерським атакам, маніпуляціям або фізичному знищенню. Крім того, централізований характер таких систем робить їх вразливими до корупції та втручання з боку зацікавлених осіб. У цьому контексті виникає необхідність у пошуку нових технологічних рішень, які б забезпечували високу ступінь безпеки, цілісності та доступності даних.

Блокчейн-технологія, яка спочатку була розроблена для підтримки криптовалют, наразі знаходить застосування у різних галузях, включаючи фінанси, логістику, охорону здоров’я та навіть державне управління. Її основними перевагами є розподіленість, незмінність записів та можливість забезпечення високого рівня безпеки даних. Використання блокчейну в якості реєстратора медіа доказів військових злочинів має потенціал стати корисним рішенням, що покликане уникнути проблем, притаманних централізованим системам.

Актуальність дослідження полягає в тому, що на даний момент немає універсального рішення, яке б забезпечувало надійне зберігання та захист медіа доказів військових злочинів з використанням сучасних технологій. Розробка програмного інтерфейсу на основі блокчейну дозволить створити систему, яка

гарантуватиме незмінність та автентичність зібраних даних, що є надзвичайно важливим для правових процесів як на національному, так і на міжнародному рівнях.

Аналіз останніх досліджень і публікацій. Питання використання блокчейн-технологій для збереження та верифікації даних активно досліджується в науковому середовищі. У роботах авторства Сатоші Накамото було закладено основи блокчейну як розподіленого реєстру транзакцій [1]. Подальші дослідження спрямовані на застосування цієї технології в різних сферах. Наприклад, А. Азаріа та ін. [2] досліджували можливості блокчейну для зберігання медичних записів, використовуючи його переваги в захисті конфіденційності та цілісності даних.

У сфері збереження цифрових доказів блокчейн також знаходить застосування. Робота Абдулхаді та Альсаєді [3] присвячена використанню блокчейну для верифікації цифрових документів, де автори підкреслюють важливість незмінності та доступності даних. Проте питання застосування блокчейну для збереження медіа доказів військових злочинів досі залишається мало дослідженим.

У працях автора магістерської роботи було розглянуто можливості створення прототипу блокчейн-додатку для безпечного зберігання фотографічних доказів військових злочинів, а також розроблено алгоритм верифікації медіа доказів на основі блокчейну. Ці дослідження підтвердили перспективність використання блокчейну в даному контексті, але також виявили ряд проблем.

Мета і завдання дослідження. Метою даної кваліфікаційної роботи є розробка ефективного та безпечного програмного інтерфейсу на основі блокчейну для захисту та збереження медіа доказів військових злочинів, що забезпечить їх цілісність, автентичність та доступність для відповідних правових органів та міжнародних інстанцій.

Для досягнення поставленої мети необхідно вирішити такі *завдання*:

1. Провести детальний аналіз існуючих методів та технологій збирання, збереження та верифікації медіа доказів, визначити їх переваги та недоліки.

2. Дослідити сучасні блокчейн-платформи та визначити оптимальну для реалізації поставлених завдань.

3. Розробити концептуальну архітектуру програмного інтерфейсу, що включатиме механізми завантаження, збереження, верифікації, перевірки автентичності медіа даних та підтвердження авторства.

4. Розробити алгоритми забезпечення безпеки, конфіденційності та цілісності даних у межах запропонованої системи.

5. Реалізувати прототип програмного інтерфейсу та провести його тестування в умовах, наближених до реальних.

6. Оцінити ефективність розробленого рішення, порівняти його з існуючими методами та визначити перспективи подальшого розвитку.

Об'єкт дослідження. Об'єктом дослідження є процеси збирання, збереження та верифікації медіа доказів військових злочинів в умовах сучасних інформаційних технологій та вимог міжнародного права.

Предмет дослідження. Предметом дослідження є методи та засоби розробки програмного інтерфейсу на основі блокчейн-технологій для забезпечення захисту, цілісності та автентичності медіа доказів військових злочинів.

Методи дослідження. У процесі дослідження використано комплекс методів, спрямованих на досягнення поставленої мети. Теоретичні методи включають аналіз та синтез наукової літератури, нормативно-правових актів та існуючих технологічних рішень. Емпіричні методи передбачають моделювання та прототипування програмного забезпечення, проведення експериментів з використанням розробленого прототипу.

Для розробки програмного інтерфейсу застосовано методи об’єктно-орієнтованого програмування, використано сучасну мову програмування C# та фреймворк ASP.NET Core Web API. Також для зручності та наближення до реальних умов використаний інструмент для ізоляції процесів працюючого програмного коду під назвою Docker. Відповідальність за забезпечення контролю версій покладено на інструмент Git. Щоб забезпечити безпеку та цілісність даних використано криптографічні методи, включаючи хешування та цифрові підписи. Також застосовані статичні аналізатори коду, що дозволило підвищити загальну якість коду та виправити незначні архітектурні неточності в процесі імплементації. Методи тестування та відлагодження дозволили оцінити функціональність та надійність розробленого рішення.

Наукова новизна та практична значущість отриманих результатів. Наукова новизна роботи полягає в розробці нових підходів до збереження та верифікації медіа доказів військових злочинів з використанням блокчейн-технологій. Зокрема:

- Запропоновано концепцію програмного інтерфейсу, який дозволяє використати переваги та особливості блокчейну в процесі збирання медіа доказів, забезпечуючи їх незмінність та автентичність.

- Імплементовано алгоритми забезпечення конфіденційності та захисту даних в умовах відкритого блокчейну.

- Створено прототип системи, що підтвердив працездатність та ефективність запропонованих рішень.

Практична значущість роботи полягає у можливості застосування розробленого програмного інтерфейсу в діяльності правоохоронних органів, міжнародних судів та організацій, що займаються захистом прав людини. Це сприятиме підвищенню прозорості збирання та збереження доказів, забезпечить надійну доказову базу для притягнення винних до відповідальності.

Апробація результатів та публікації. Результати дослідження були апробовані на V науково-практичній конференції «Проблеми комп'ютерної інженерії», що відбулася у 2024 році в місті Київ, де були представлені тези на тему «Алгоритм верифікації медіа доказів військових злочинів на основі технології блокчейн».

Також результати дослідження опубліковані в журналі «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу» у статті «Створення прототипу блокчейн-додатку для безпечного зберігання фотографічних доказів військових злочинів».

Теоретична, методична та практична значущість отриманих результатів. Теоретична значущість роботи полягає у розвитку наукових уявлень про можливості застосування блокчейн-технологій у сфері збереження та верифікації критично важливих медіа-даних. Розроблені концепції та підходи можуть бути основою для подальших наукових досліджень у галузі цифрових технологій.

Методична значущість полягає у розробці методик та алгоритмів створення програмного забезпечення для збереження медіа даних на основі блокчейну. Ці методики є певною мірою абстрактними, що дозволяє використовувати їх при розробці різних систем на основі технології блокчейн, що пов'язані з медіа-даними. Наведені методики також можуть бути використані у навчальному процесі при підготовці фахівців з інформаційних технологій, а також розробниками, які працюють над подібними проєктами.

Практична значущість роботи полягає у можливості безпосереднього застосування розробленого рішення у діяльності правоохоронних органів та міжнародних організацій для забезпечення цілісності, автентичності та незмінності медіа доказів військових злочинів.

1 ОГЛЯД ТЕХНОЛОГІЙ

У сучасному світі інформаційні технології стрімко розвиваються, пропонуючи нові рішення для зберігання, обробки та захисту даних. Не можна не згадати про це у контексті питання збереження медіа доказів військових злочинів, де безпека та цілісність інформації відіграють критичну роль. У цьому розділі буде проведено детальний аналіз існуючих алгоритмічних та технічних рішень, а також буде розглянуто допоміжні програмні засоби та засоби розробки, які можуть бути використані для реалізації поставленої мети.

1.1 Блокчейн

За останні десятиліття інформаційні технології здійснили революційний вплив на суспільство, впливаючи на способи взаємодії, бізнес-моделі та підходи до зберігання та передачі інформації. Однією з найбільш визнаних інновацій у цій сфері стала технологія блокчейн [4]. Спочатку розроблена як основа для криптовалюти Bitcoin, вона швидко привернула увагу дослідників та підприємців у різних галузях завдяки своїм унікальним властивостям та потенціалу для трансформації традиційних систем.

Технологія блокчейн була вперше представлена у 2008 році анонімною особою або групою осіб під псевдонімом Сатоші Накамото в роботі “Bitcoin: A Peer-to-Peer Electronic Cash System”. Метою було створення децентралізованої цифрової валюти, яка не залежить від центральних банків чи фінансових установ. Блокчейн став основою цієї системи, забезпечуючи безпеку, прозорість та незмінність транзакцій.

1.1.1 Структура та принцип роботи блокчейну

Блокчейн складається з послідовності блоків, кожен з яких містить:

- *Заголовок блоку*: Містить метадані, такі як хеш попереднього блоку, часовий штамп, nonce та інші параметри.

- *Тіло блоку*: Містить набір транзакцій або записів даних.

Процес додавання нового блоку включає:

1. *Збір транзакцій*: Транзакції відправляються в мережу та збираються вузлами.

2. *Верифікація транзакцій*: Вузли перевіряють коректність та автентичність транзакцій.

3. *Створення блоку*: Вузли об’єднують транзакції у блок.

4. *Консенсусний механізм*: Визначає, який вузол додасть наступний блок у ланцюжок. Найпоширеніші механізми — Proof of Work (доказ виконаної роботи) та Proof of Stake (доказ частки володіння).

5. *Додавання блоку до ланцюжка*: Після успішного додавання блоку він поширюється по мережі, і всі вузли оновлюють свої копії блокчейну.

Описані етапи додавання блоку в блокчейн-системі продемонстровані на рис. 1.1.

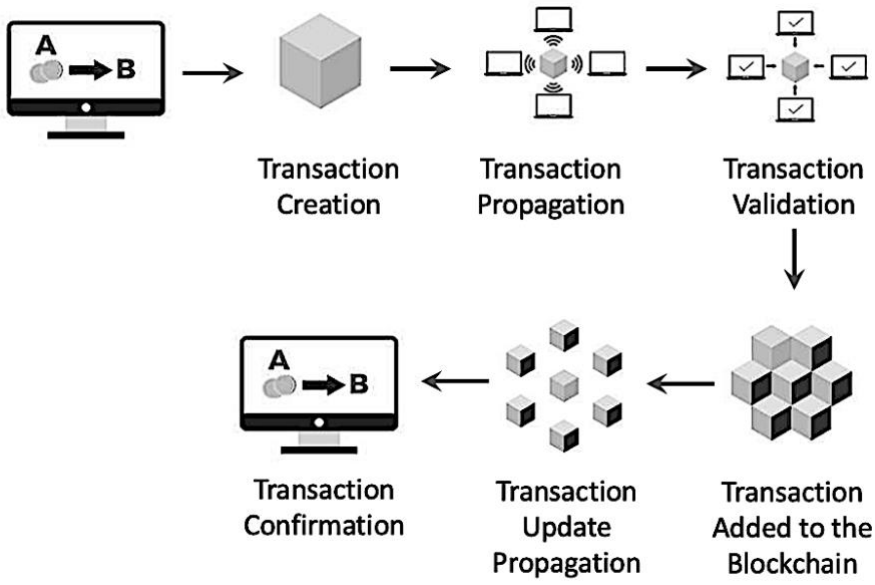


Рис. 1.1 Етапи реєстрації транзакцій у блокчейні

1.1.2 Криптографічні принципи в блокчейні

Блокчейн використовує різні криптографічні методи для забезпечення безпеки:

- *Хешування*: Кожен блок містить хеш попереднього блоку, що створює зв’язок між ними. Будь-яка зміна в попередньому блоці змінює його хеш, що робить неможливим непомітну модифікацію даних.

- *Цифрові підписи*: Використовуються для автентифікації транзакцій. Користувачі підписують транзакції своїми приватними ключами, що дозволяє іншим вузлам перевіряти їх автентичність за допомогою публічних ключів.

- *Криптографічні ключі*: Кожен учасник має пару ключів (публічний та приватний), що забезпечує безпеку та конфіденційність

Кожен з перелічених принципів присутній на рис. 1.2. На рисунку також можна побачити взаємодію окремих компонентів транзакції.

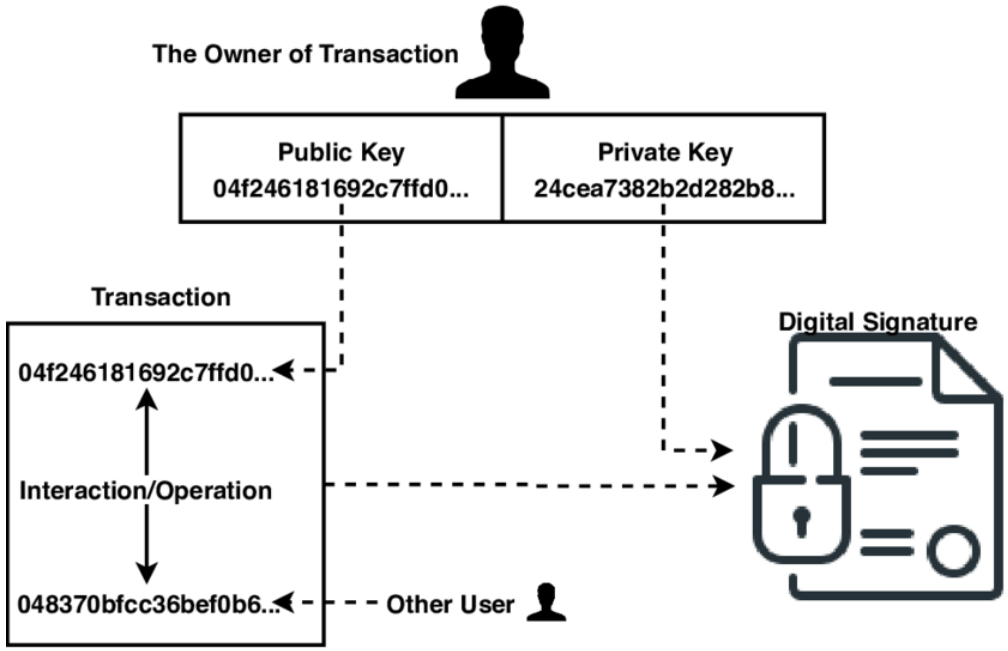


Рис. 1.2 Роль хешу, ключів та цифрового підпису в блокчейні

1.1.3 Консенсусні механізми

Консенсусні механізми — це алгоритми, які дозволяють розподіленій мережі досягати згоди щодо стану блокчейну [5]. Основні механізми:

- *Proof of Work (PoW)*: Вимагає від вузлів вирішення складних математичних задач для додавання нового блоку. Це забезпечує безпеку, але споживає багато енергії.

- *Proof of Stake (PoS)*: Вузли обираються для додавання блоку на основі їх частки володіння криптовалютою. Менш енерговитратний, але має інші ризики.

- *Інші механізми*: Delegated Proof of Stake, Proof of Authority, Proof of Elapsed Time та інші, які намагаються вирішити проблеми масштабованості та енергоефективності.

Наявність цих та інших консенсусних механізмів можна бачити на рис. 1.3. На рисунку також висвітлена необхідність у використанні різних механізмів в залежності від типу та особливостей використаного блокчейна.

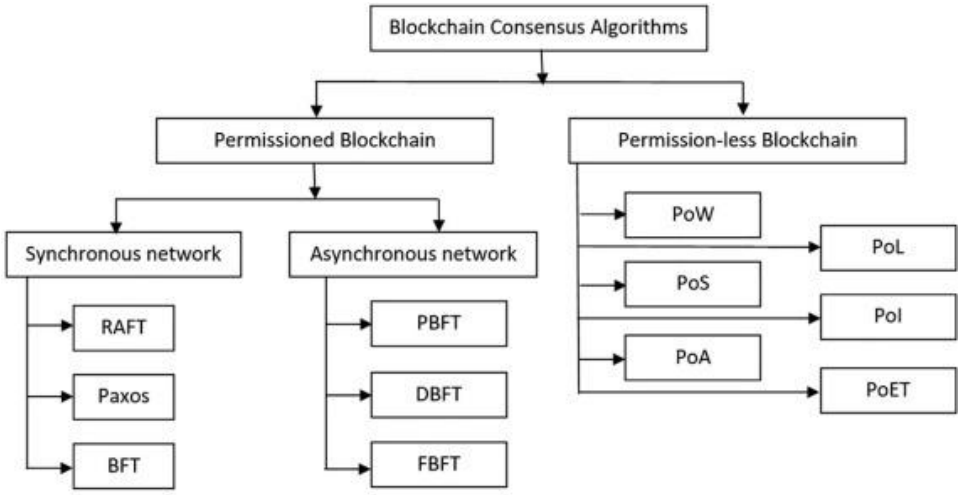


Рис. 1.3 Схема відношення класу блокчейну до підтримуваних консенсусних механізмів

Недостатнє розуміння технології серед широкого загалу та спеціалістів може стати проблемою. Необхідність адаптації та сумісності з поточними технологічними рішеннями завадить впровадженню розглянутої технології. Також варто вказати, що стереотипи та недовіра до нових технологій можуть уповільнювати впровадження.

1.2 Перелік відомих алгоритмічних та технічних рішень

1.2.1 Відомі алгоритми хешування

Алгоритми хешування є фундаментальними у сфері криптографії та інформаційної безпеки. Вони перетворюють вхідні дані будь-якого розміру в фіксоване хеш-значення, що використовується для перевірки цілісності даних, автентифікації та цифрових підписів. Розглянемо основні з них.

MD5 (Message Digest Algorithm 5). Розроблений Рональдом Рівестом у 1991 році, MD5 генерує 128-бітове хеш-значення. Він був широко використовуваний для перевірки цілісності файлів та зберігання паролів [6]. Однак у 1996 році були виявлені вразливості, що дозволяють генерувати колізії – різні вхідні дані, які дають однакове хеш-значення. Це робить MD5 непридатним для використання в сучасних безпечних системах.

SHA-1 (Secure Hash Algorithm 1). Розроблений NIST у 1995 році, SHA-1 генерує 160-бітове хеш-значення. Він був стандартом для багатьох криптографічних застосувань [7], але у 2005 році були виявлені теоретичні вразливості, а у 2017 році дослідники з Google та CWI Amsterdam продемонстрували практичну атаку на SHA-1, що робить його небезпечним для використання в нових проєктах.

SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512). Сімейство алгоритмів SHA-2 було прийняте як стандарт NIST у 2001 році. SHA-256, який генерує 256-бітове хеш-значення, є найбільш популярним у цьому сімействі [8]. Він використовується в багатьох сучасних системах, включаючи блокчейн-технології (наприклад, у мережі

Bitcoin). На сьогоднішній день не існує відомих практичних атак на SHA-256, що робить його надійним вибором для забезпечення цілісності даних.

SHA-3. У 2015 році NIST прийняв SHA-3 як новий стандарт хешування. Він базується на алгоритмі Keccak, який переміг у конкурсі NIST на новий алгоритм хешування [9]. SHA-3 має іншу криптографічну конструкцію порівняно з SHA-2, що надає додаткову безпеку в разі виявлення вразливостей у SHA-2.

BLAKE2 та BLAKE3. BLAKE2 є вдосконаленням оригінального алгоритму BLAKE, який був фіналістом конкурсу SHA-3. Він пропонує високу швидкість та безпеку, а також є простішим у реалізації. BLAKE3, випущений у 2020 році, є ще швидшим та більш ефективним, зберігаючи високий рівень безпеки [10].

1.2.2 Відомі алгоритми асиметричного шифрування

Асиметричне шифрування використовує пару ключів – відкритий та закритий – для шифрування та розшифрування даних. Це забезпечує конфіденційність та автентичність інформації.

RSA (Rivest-Shamir-Adleman). RSA є одним із найстаріших та найвідоміших алгоритмів асиметричного шифрування, розробленим у 1977 році. Його безпека базується на складності факторизації великих простих чисел [11]. RSA широко використовується для цифрових підписів та обміну ключами. Основні недоліки RSA – потреба у великих розмірах ключів (2048 біт та більше) для забезпечення високого рівня безпеки, що впливає на продуктивність.

ECC (Elliptic Curve Cryptography). Криптографія на еліптичних кривих базується на математичних властивостях еліптичних кривих над кінцевими полями. Вона забезпечує той самий рівень безпеки, що й RSA, але з меншими розмірами ключів [12]. Наприклад, 256-бітний ключ ECC еквівалентний за безпекою 3072-бітному ключу RSA. Це робить ECC більш ефективним з точки зору продуктивності та ресурсів.

ElGamal та DSA (Digital Signature Algorithm). Алгоритм ElGamal базується на складності розв'язання дискретного логарифму. DSA, стандартизований NIST, також базується на цій проблемі та використовується для цифрових підписів [13]. Варіант DSA на основі еліптичних кривих – ECDSA – поєднує переваги ECC та DSA.

1.2.3 Огляд JSON та XML

У контексті розробки програмного інтерфейсу на основі блокчейну для захисту та збереження медіа доказів військових злочинів, важливо обрати оптимальний формат передачі даних, який забезпечить ефективність, зручність та надійність. У цьому підрозділі проведемо детальний аналіз та порівняння JSON та XML, розглянемо їхні переваги та недоліки, а також визначимо, який формат є більш придатним для реалізації поставлених завдань.

JSON — це текстовий формат обміну даними, заснований на підмножині мови JavaScript. Він був розроблений Дугласом Крокфордом у 2001 році [14]. JSON є легким, простим для читання та запису форматом, який використовується для передачі структурованих даних між клієнтом та сервером у веб-додатках.

Основні характеристики JSON:

- Простий синтаксис, заснований на парі "ключ-значення".
- Підтримує основні типи даних: рядки, числа, об'єкти, масиви, булеві значення та null.
- Легкий для парсингу у багатьох мовах програмування завдяки наявності стандартних бібліотек.
- Менший розмір даних порівняно з XML, оскільки не використовує відкриваючі та закриваючі теги.

XML — це розширювана мова розмітки, розроблена W3C у 1998 році. Вона призначена для представлення структурованих даних у вигляді текстових документів [15]. XML є самодокументованою мовою, яка дозволяє визначати власні теги та структуру документа.

Основні характеристики XML:

- Гнучкість та розширюваність: можливість визначення власних тегів та структур.
- Самодокументованість: структура даних та їх значення можуть бути зрозумілі без додаткової інформації.
- Підтримка схем та валідації: можливість визначення схем (DTD, XSD) для валідації структури та типів даних.
- Широка підтримка: XML використовується у багатьох стандартах та протоколах.

1.3 Перелік розглянутих архітектурних рішень

Архітектура додатку відіграє ключову роль у визначенні його масштабованості, продуктивності, гнучкості та зручності використання. Правильний вибір архітектурного підходу та технологій може значно вплинути на успішність проєкту, особливо коли мова йде про системи, що працюють з критично важливими даними, такими як медіа докази військових злочинів. У цьому підрозділі проведемо детальний аналіз основних архітектурних рішень, таких як REST, GraphQL та gRPC, а також розглянемо архітектурні патерни MVC, клієнт-сервер та мікросервісну архітектуру.

REST є архітектурним стилем, що використовує протокол HTTP для взаємодії між клієнтом та сервером. Він базується на принципах клієнт-серверної архітектури, безстанності (stateless), кешування, уніфікованого інтерфейсу,

багатошаровості системи [16]. Схему взаємодії RESTful API та клієнта що можна побачити на рис. 1.4.

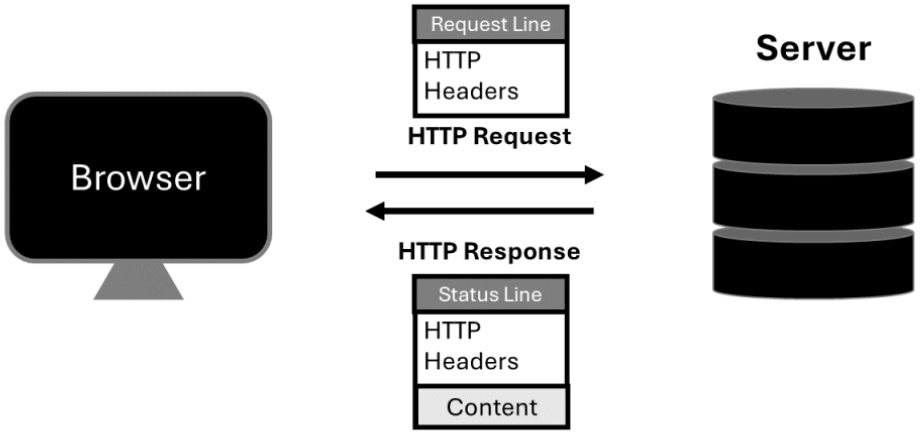


Рис. 1.4 Схема роботи REST API

Переваги:

- Простота та легкість розуміння: REST використовує стандартні методи HTTP (GET, POST, PUT, DELETE), що робить його інтуїтивно зрозумілим для розробників.

- Широка підтримка та сумісність: REST підтримується практично всіма мовами програмування та фреймворками.

- Масштабованість: Завдяки безстанності сервера, REST-додатки легко масштабуються горизонтально.

- Кешування: Можливість кешування відповідей підвищує продуктивність та знижує навантаження на сервер.

Недоліки:

• Надлишковість даних: REST може передавати більше даних, ніж потрібно клієнту, особливо коли API повертає великі об’єкти.

• Обмеженість функціональності: REST не підтримує двонаправлений стрімінг даних та може бути неефективним для деяких типів запитів.

GraphQL — це мова запитів для API, розроблена Facebook. Вона дозволяє клієнтам запитувати саме ті дані, які їм потрібні, що зменшує обсяг переданих даних та кількість запитів [17]. Схема роботи GraphQL висвітлена на таблиці 1.5.

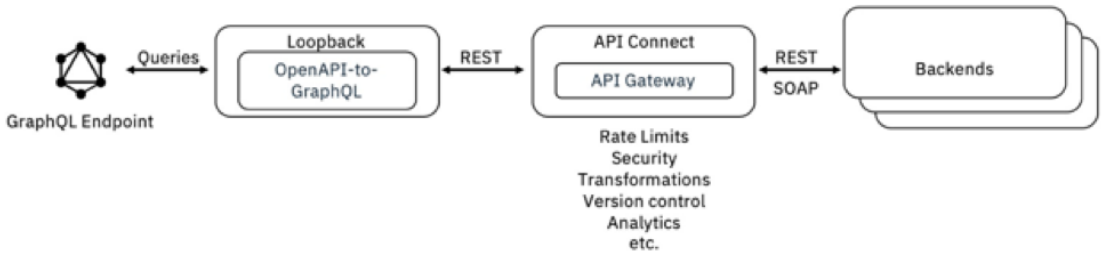


Рис. 1.5 Схема роботи GraphQL

Переваги:

• Гнучкість запитів: Клієнт може точно визначити структуру та обсяг даних, які він хоче отримати.

• Зменшення кількості запитів: Можливість об’єднувати кілька запитів в один, що знижує латентність.

• Схемна система: GraphQL використовує схему, що дозволяє краще документувати та перевіряти API.

Недоліки:

• Складність у налаштуванні та підтримці: Потребує додаткового налаштування сервера та вивчення специфіки GraphQL.

• Відсутність кешування на рівні HTTP: Стандартні механізми кешування HTTP не застосовні, що може вимагати реалізації власних механізмів кешування.

• Проблеми з безпекою: Потрібно додатково обробляти складні та потенційно шкідливі запити.

gRPC — це високопродуктивний фреймворк для віддаленого виклику процедур, розроблений Google. Він використовує протокол HTTP/2 та формат серіалізації Protobuf [18]. Компоненти високого рівня фреймворку gRPC позначені на рис. 1.6.

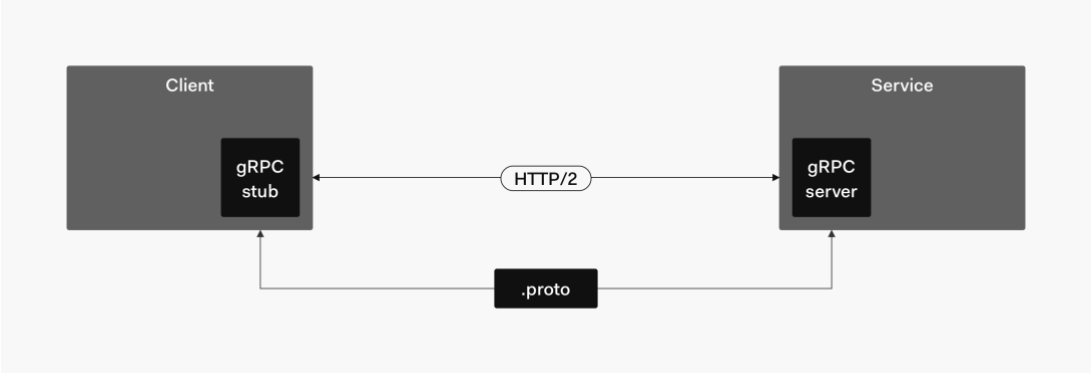


Рис. 1.6 Схема роботи gRPC

Переваги:

• Висока продуктивність та ефективність: Завдяки використанню HTTP/2 та Protobuf, gRPC забезпечує швидку передачу даних з низькою латентністю.

• Двонаправлений стрімінг: Підтримує як клієнтський, так і серверний стрімінг даних.

• Сувора типізація: Визначення сервісів та повідомлень у Protobuf забезпечує сувору типізацію та спрощує генерацію коду клієнтів та серверів.

Недоліки:

- Складність у налагодженні: Менш розповсюджений порівняно з REST, що може ускладнити розробку та налагодження.

- Обмежена підтримка браузерами: gRPC не підтримується безпосередньо браузерами, що потребує використання спеціальних шлюзів або альтернативних рішень для веб-клієнтів.

- Вищий поріг входу: Потрібно освоїти Protobuf та специфіку gRPC.

1.4 Перелік розглянутих архітектурних патернів

Окрім вибору протоколу та формату передачі даних, важливим є визначення загальної архітектури додатку. Розглянемо основні патерни та їх застосування.

Патерн MVC (Model-View-Controller). MVC розділяє додаток на три основні компоненти [19]:

- *Модель (Model):* Відповідає за дані та бізнес-логіку.

- *Представлення (View):* Відповідає за відображення даних користувачу.

- *Контроллер (Controller):* Обробляє вхідні запити, взаємодіє з моделлю та передає дані представленню.

Взаємодія описаних компонентів висвітлена на рис. 1.7.

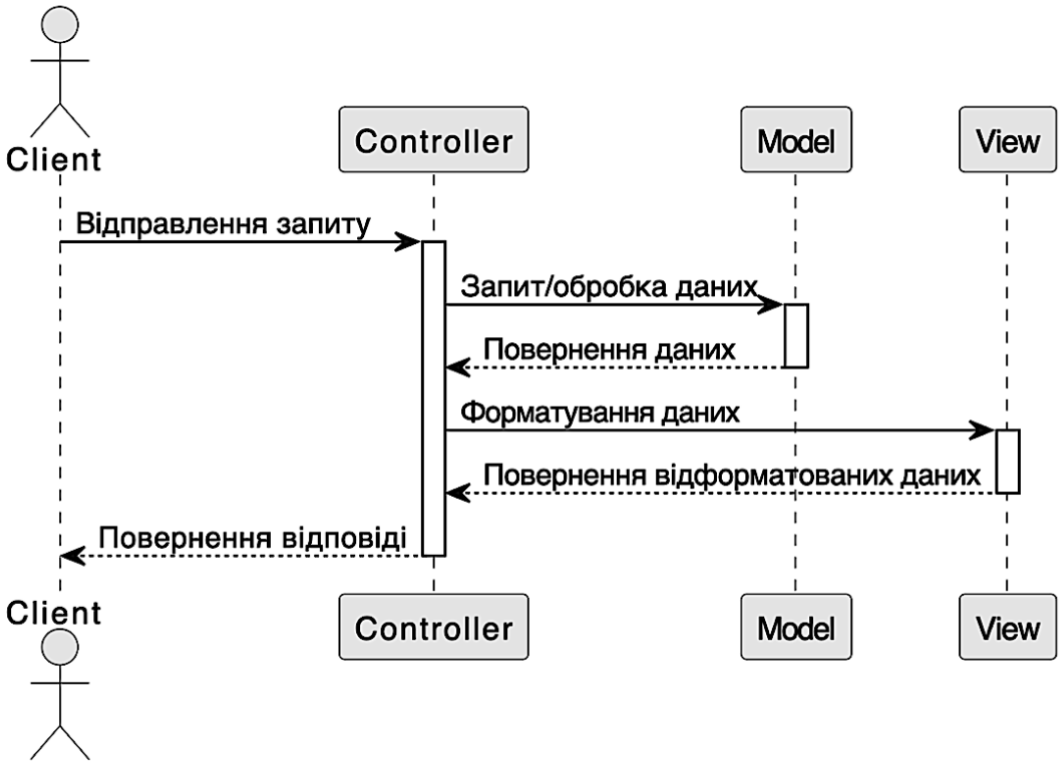


Рис. 1.7 Схема взаємодії компонентів MVC

Переваги:

- *Розділення відповідальностей:* Полегшує розробку та підтримку коду.
- *Масштабованість:* Кожен компонент можна розвивати незалежно.
- *Полегшене тестування:* Можливість окремого тестування кожного компонента.

Недоліки:

- *Складність для малих проєктів:* Може бути надлишковим для простих додатків.
- *Потенційна складність зв'язків між компонентами:* Потребує чіткого визначення інтерфейсів та взаємодії.

Клієнт-серверна архітектура. Традиційна модель, де клієнт відправляє запити до сервера, який обробляє їх та повертає відповіді [20]. Зображена на рис. 1.8.

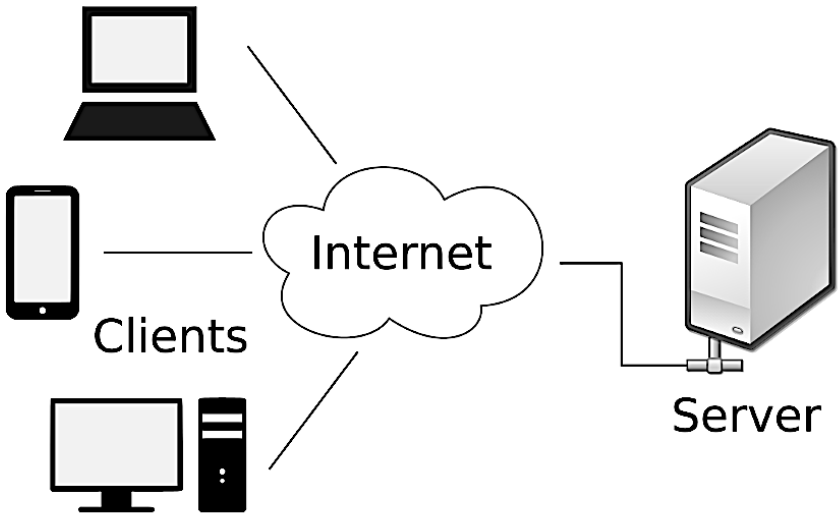


Рис. 1.8 Схема клієнт-серверної архітектури

Переваги:

- Простота розуміння та реалізації.
- Чітке розмежування ролей клієнта та сервера.

Недоліки:

- Обмежена масштабованість: Сервер може стати “вузьким місцем”.
- Менш гнучка для сучасних вимог до продуктивності та доступності.

Мікросервісна архітектура. Додаток розбивається на набір незалежних сервісів, кожен з яких виконує певну функцію та може розгортатися окремо [21]. Приклад використання та компоненти, що беруть участь у обробці запиту, зображені на рис. 1.9.

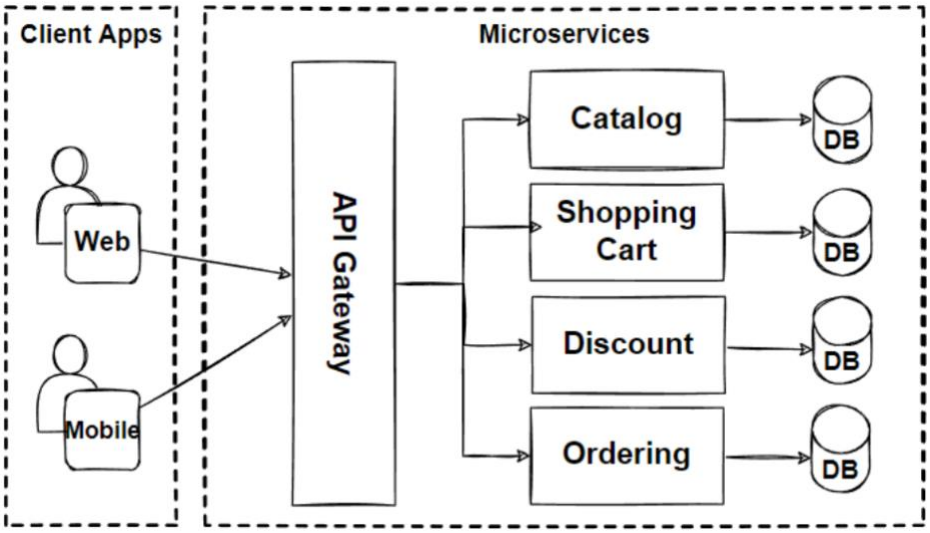


Рис. 1.9 Мікросервісна архітектура

Переваги:

- *Масштабованість*: Можливість масштабувати окремі сервіси за потреби.

- *Гнучкість розробки*: Різні команди можуть працювати над різними сервісами.

- *Незалежність технологій*: Кожен сервіс може бути написаний на різних мовах програмування або використовувати різні бази даних.

Недоліки:

- *Складність управління*: Потребує складних інструментів для оркестрації та моніторингу.

- *Підвищені вимоги до інфраструктури*: Потрібна надійна система управління контейнерами (наприклад, Kubernetes).

- *Складність у налагодженні та тестуванні*: Важче відслідковувати помилки, що виникають в системі.

1.5 Перелік розглянутих варіантів індексації

У сучасних інформаційних системах ефективний доступ до даних є одним із ключових факторів їх продуктивності та зручності використання. У контексті розробки програмного інтерфейсу на основі блокчейну для захисту та збереження медіа доказів військових злочинів виникає потреба в швидкому та ефективному пошуку блоків за певними параметрами. Це особливо актуально, коли обсяг даних постійно зростає, і без належної індексації пошук може стати непрактично повільним.

Для вирішення цієї проблеми було вирішено реалізувати додатковий модуль індексації, який дозволяє індексувати блоки за певними параметрами. Цей модуль підтримує як одиночні ключі, так і складні ключі, що складаються з декількох полів. У цьому підрозділі буде проведено детальний аналіз різних алгоритмів індексації, які можуть бути реалізовані з нуля, а також обґрунтовано вибір хеш-таблиць як основного механізму індексації в даному проєкті.

Існує кілька основних алгоритмів та структур даних, які можуть бути використані для індексації:

- Лінійний пошук
- Бінарний пошук
- Хеш-таблиці
- B-дерева
- Trie (префіксні дерева)
- Інвертовані індекси
- Skip List (списки зі стрибками)

Розглянемо кожен із цих варіантів детальніше, оцінюючи їхні переваги та недоліки в контексті нашого проєкту.

1.5.1 Лінійний пошук

Принцип роботи: Послідовний перегляд кожного елемента колекції до знаходження потрібного або до кінця списку. Приклад роботи алгоритму висвітлений на рис. 1.10.

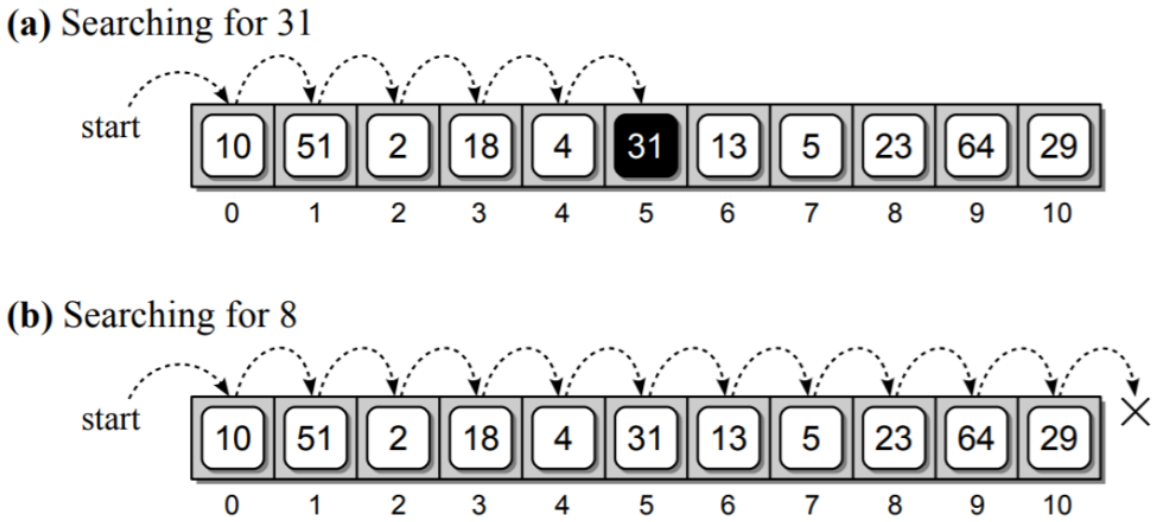


Рис. 1.10 Лінійний пошук

Переваги:

- Простота реалізації.
- Не потребує додаткової пам’яті.

Недоліки:

- Часова складність O(n), де n — кількість елементів.
- Непридатний для великих обсягів даних.

Застосування в проєкті: Непридатний, оскільки обсяг даних може бути значним, а швидкість пошуку низькою.

1.5.2 Бінарний пошук

Принцип роботи: Працює на відсортованих даних. На кожному кроці пошуку список ділиться навпіл, і визначається, в якій половині знаходиться шуканий елемент. Один з кроків бінарного пошуку зображений на рис. 1.11.

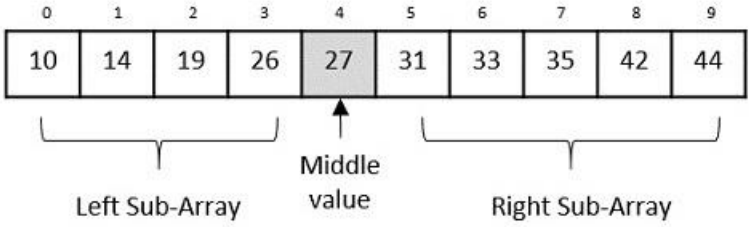


Рис. 1.11 Операція в бінарному пошуку

Переваги:

- Часова складність O(log n).
- Відносно проста реалізація.

Недоліки:

- Потребує підтримки відсортованих даних.
- Вставка та видалення елементів складніші та можуть бути ресурсомісткими.

Застосування в проєкті: Не оптимальний, оскільки дані в блокчейні постійно додаються, і підтримка сортування може бути складною.

1.5.3 Хеш-таблиці

Принцип роботи: Використовує хеш-функцію для перетворення ключа в індекс масиву, де зберігається значення. Забезпечує швидкий доступ до даних.

Приклад відношення списку даних до хеш-таблиці зображений на рис. 1.12.

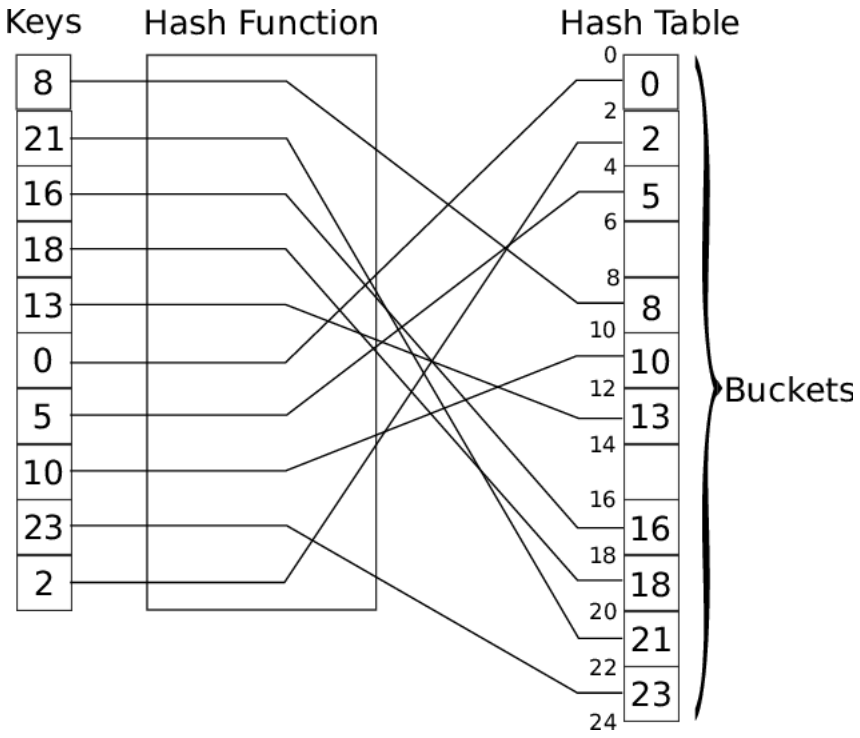


Рис. 1.12 Схема додавання елементів у хеш-таблицю та читання

Переваги:

- Часова складність доступу O(1) в середньому випадку.
- Ефективність при великому обсязі даних.
- Підтримка динамічного додавання та видалення елементів.
- Можливість використання складних ключів.

Недоліки:

- Можливість колізій.
- Потреба в добрій хеш-функції.

Застосування в проєкті: Відповідає вимогам швидкості та ефективності, підтримує складні ключі.

1.5.4 B-дерева

Принцип роботи: Збалансоване дерево, в якому кожен вузол може містити посилання на більш ніж один ключ. Широко використовується в базах даних для зберігання великих обсягів даних на диску. Уявлення цього типу даних зображено на рис. 1.13.

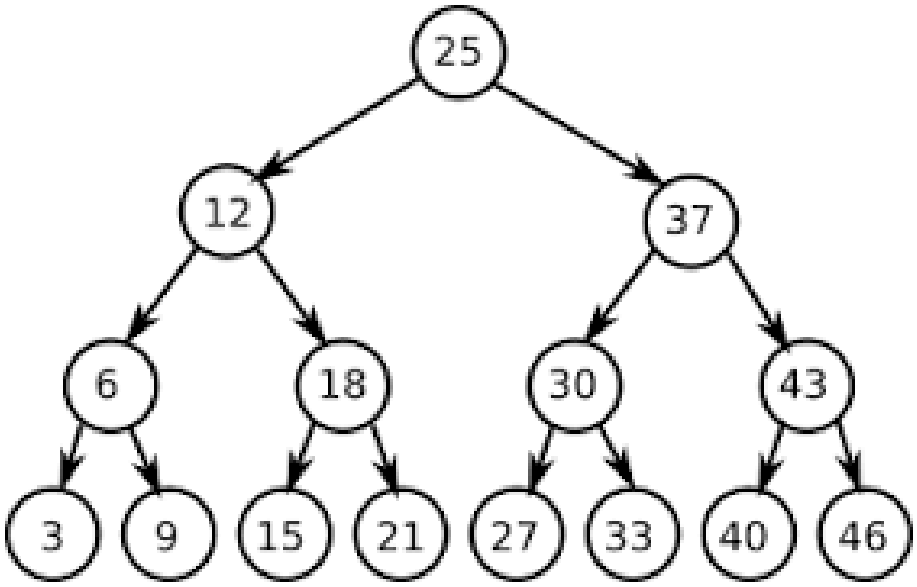


Рис. 1.13 Бінарне дерево

Переваги:

- Часова складність O(log n).
- Ефективне використання дискового простору.
- Добре підходить для зберігання на диску.

Недоліки:

- Складна реалізація.
- Менша швидкість порівняно з хеш-таблицями в оперативній пам’яті.

Застосування в проєкті: Не оптимальний, оскільки наша система орієнтована на швидкий доступ в пам’яті.

1.5.5 Trie (префіксні дерева)

Принцип роботи: Дерево, в якому кожен вузол представляє символ або частину ключа. Використовується для зберігання та пошуку рядкових ключів. Приклад префіксного дерева для пошуку латинських префіксів наведено на рис. 1.14.

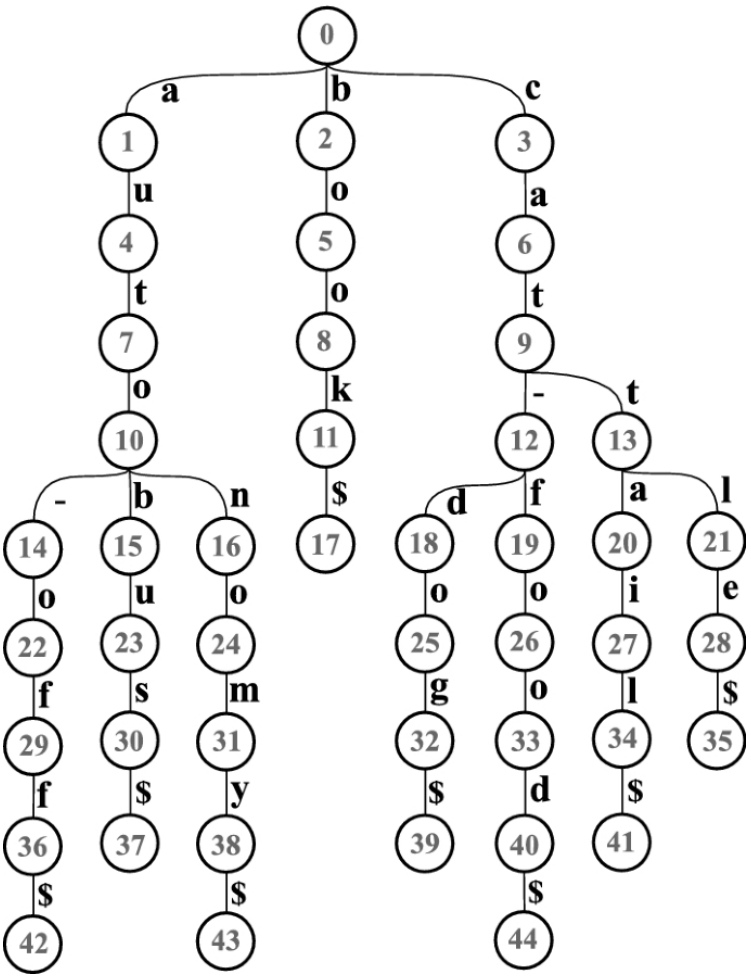


Рис. 1.14 Префіксне дерево

Переваги:

- Швидкий пошук рядкових ключів.

- Ефективний для автодоповнення та пошуку за префіксом.

Недоліки:

- Великі витрати пам’яті.
- Неефективний для нестрокових ключів.

Застосування в проєкті: Не підходить, оскільки не забезпечує необхідної ефективності для нашого типу даних.

1.5.6 Інвертовані індекси

Принцип роботи: Структура даних, яка відображає значення в список документів або місць, де це значення зустрічається, або де значення і є цим документом. Широко використовується в пошукових системах. Відносне порівняння розташування тих самих даних в прямому індексі та інвертованому індексі зображено на рис. 1.15.

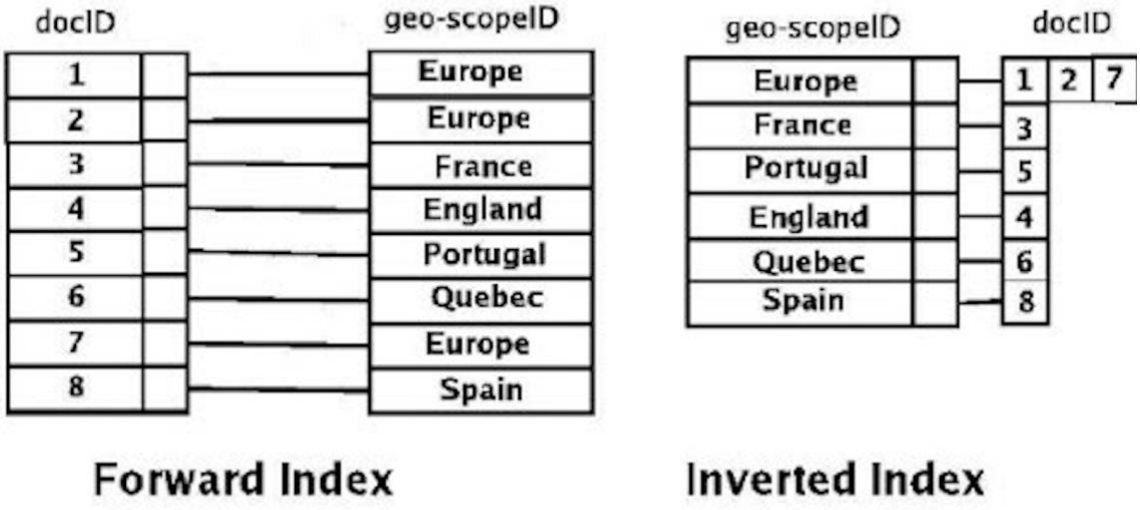


Рис. 1.15 Порівняння традиційного і інвертованого індексів

Переваги:

- Ефективний для повнотекстового пошуку.
- Підтримка пошуку за різними параметрами.

Недоліки:

- Великі витрати пам’яті.
- Складність оновлення індексу.

Застосування в проєкті: Не оптимальний, оскільки наші дані структуровані і не стоїть вимога повнотекстового пошуку.

1.6 Тестування програмного забезпечення

Існують кілька основних підходів до розробки програмного забезпечення. Приклад циклу розробки можна бачити на рис. 1.16. Кожен з підходів включає у себе етап тестування створеного продукту. Тестування програмного забезпечення є критично важливим етапом у процесі розробки, який забезпечує якість, надійність та відповідність програмного продукту вимогам користувачів. Воно дозволяє виявити та виправити помилки, покращити функціональність та забезпечити безпеку системи. У цьому підрозділі розглянемо основні види тестування, їх цілі, методи та сфери застосування.

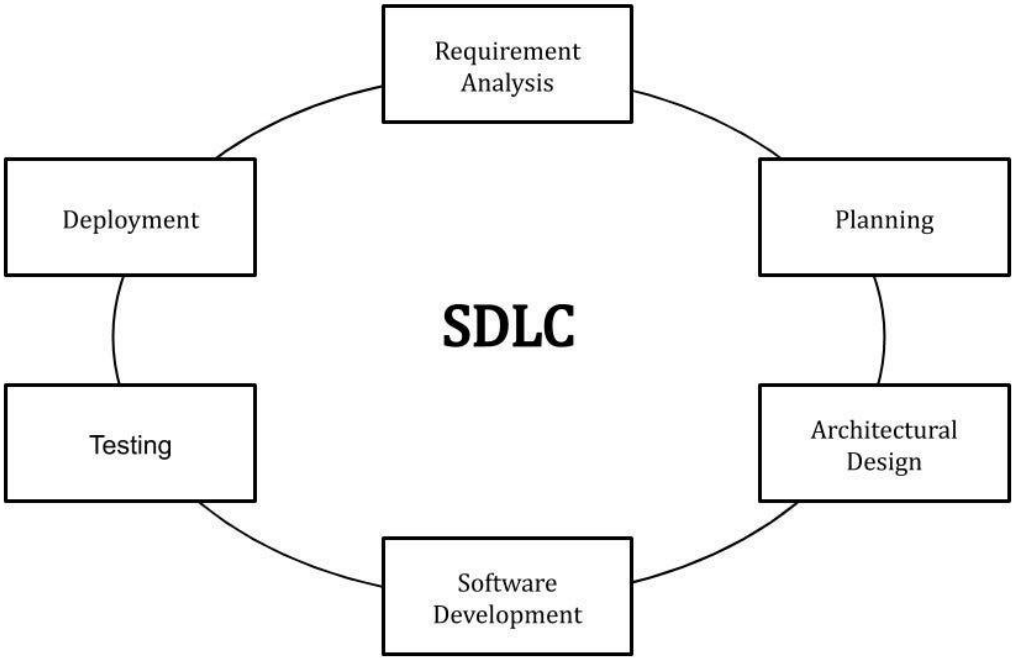


Рис. 1.16 Приклад циклу розробки програмного забезпечення (Software Development Lifecycle)

Модульне тестування. Модульне тестування перевіряє окремі компоненти або модулі програми ізольовано від інших частин системи. Метою є виявлення помилок на ранніх етапах розробки та забезпечення коректності роботи кожного модуля. Основні характеристики такого рівня тестування є те що воно виконуються розробниками, чи тестувальниками, потребує знання внутрішньої структури коду. Такі тести часто автоматизуються шляхом використання спеціалізованих утиліт та інструментів.

Інтеграційне тестування. Інтеграційне тестування перевіряє взаємодію між різними модулями після їх об’єднання. Метою є виявлення дефектів в інтерфейсах та протоколах взаємодії між компонентами.

Підходи до інтеграційного тестування:

- *Верхньо-низовий підхід***:** Починається з верхніх модулів і поступово додає нижчі.

- *Низько-верхній підхід***:** Починається з нижніх модулів і поступово додає вищі.

- *Сандвіч-підхід***:** Поєднання двох попередніх методів.

Системне тестування. Системне тестування перевіряє повністю інтегровану систему на відповідність встановленим вимогам. Воно охоплює як функціональні, так і нефункціональні аспекти програми. Вирізняє такий рівень тестування те, що воно виконується, зазвичай, незалежною командою тестувальників та включає різні види тестування: продуктивність, безпека, сумісність тощо.

Автоматизоване тестування. Використовує інструменти та скрипти для автоматизації процесу тестування. Підходить для повторюваних та об’ємних тестів. Якщо тести автоматизовані, то це пришвидшує швидкість виконання тестів, їхню точність. Також підвищує можливу частоту виконання тестів. Але недоліком слугує те що розробка таких тестів займає більше часу та на їх підтримку потрібні ресурси розробників.

1.6.1 Важливість тестування для якості програмного забезпечення

Тестування допомагає виявити та виправити помилки до випуску продукту, забезпечити відповідність функціональних та нефункціональних вимог. Також тестування повинне підвищити задоволеність користувачів та довіру до продукту. Важливим аспектом тестування є зниження ризиків та витрат, пов’язаних з можливими дефектами в майбутньому.

Розуміння різних видів та методів тестування є ключем для стабільної розробки, покращення та впровадження програмного забезпечення. Вибір конкретних стратегій тестування дозволяє забезпечити високу якість продукту, відповідність вимогам та очікуванням користувачів, а також, мінімізувати ризики,

пов’язані з помилками та недоліками в системі, іншими словами, мінімізувати людський фактор.

Тестування повинно бути інтегрованим процесом, культурою розробки, яка супроводжує процес створення програмного продукту на всіх її етапах, включаючи постійну перевірку та вдосконалення програмного забезпечення.

1.7 Інтеграція бізнес-правил у системі

Розглянемо іншу проблему, з якою можна зіткнутися в процесі проєктування та імплементації системи.

Окрім абстрактної логіки індексації даних для швидкого доступу до них, подібним чином постає питання архітектурного плану інтеграції бізнес-правил у систему. За умови абстрагування, ми зможемо вирішити низку проблем у системі, які, виникли б на якомусь з етапів. Цю проблему потрібно врахувати і розглянути варіанти вирішення на етапі проєктування програмного забезпечення.

1.8 Висновки до розділу

У першому розділі було проведено теоретико-методичний аналіз ключових аспектів, пов’язаних з розробкою програмного інтерфейсу на основі блокчейну для захисту та збереження медіа доказів військових злочинів. Було продемонстровано науково-дослідницькі компетенції через детальне вивчення сучасного стану наукової думки щодо даної проблематики.

Розгляд алгоритмів хешування включав аналіз відомих криптографічних функцій, таких як MD5, SHA-1, SHA-256, SHA-3, BLAKE2 та BLAKE3. Було досліджено їхні характеристики, особливості роботи та сфери застосування, що повинно дозволити зрозуміти наскільки вони підходять для забезпеченні цілісності даних у блокчейн-технологіях.

Вивчення алгоритмів асиметричного шифрування охопило такі методи, як RSA та ECC (Elliptic Curve Cryptography). Було розглянуто їхні математичні основи, криптографічні властивості та особливості застосування в системах захисту інформації.

Огляд архітектурних рішень включав дослідження різних протоколів та підходів для побудови API, зокрема REST, GraphQL та gRPC. Було проаналізовано їхні основні принципи, переваги та обмеження, а також можливості застосування в різних контекстах.

Вивчення форматів передачі даних зосереджувалося на JSON та XML. Було розглянуто їхні синтаксичні особливості, структурні відмінності та області застосування, що є важливим при виборі інтерфейсу для передачі інформації.

Аналіз методів індексації даних охоплював різні алгоритми та структури даних, включаючи хеш-таблиці, B-дерева, інвертовані індекси та інші. Було досліджено принципи їх роботи, ефективність та придатність для застосування в системах з великим обсягом даних.

Розгляд абстрактних правил та їх реалізації підкреслив важливість забезпечення цілісності та коректності даних у інформаційних системах. Було висвітлено критичність абстрагування правил бізнес-логіки для їх подальшого впровадження в програмні продукти.

Завдяки загальному огляду технології блокчейн, було досягнуто глибоке розуміння її фундаментальних принципів, включаючи децентралізацію, незмінність записів, прозорість та безпеку. Було досліджено різні типи блокчейнів, механізми консенсусу та сфери їх застосування поза межами криптовалют.

Аналіз видів тестування програмного забезпечення включав вивчення модульного, інтеграційного, системного, функціонального та нефункціонального тестування. Було розглянуто цілі, методи та значення кожного виду тестування для забезпечення якості програмного продукту.

У ході критичного огляду наукових джерел було виявлено сучасні тенденції та підходи до розробки безпечних та ефективних інформаційних систем. Було висловлено позицію щодо важливості поєднання перевірених технологій з новітніми розробками для досягнення оптимальних результатів.

Сформульовано проблемні питання, які стосуються вибору оптимальних алгоритмів та технологій для забезпечення захисту медіа доказів. Визначено напрямки подальшого дослідження, зосереджені, в першу чергу, на практичному застосуванні вивчених методик.

Методики та методичні підходи, викладені в цьому розділі, створюють основу для подальшої роботи. Вони будуть використані в наступних розділах для проєктування та розробки програмного інтерфейсу, а також для оцінки його ефективності та відповідності поставленим вимогам.

Підсумовуючи, даний розділ забезпечив глибоке теоретичне підґрунтя для розуміння складових частин та принципів роботи систем на основі блокчейну. Проведений аналіз сучасних теорій, концепцій та технологій дозволив окреслити рамки дослідження та підготуватися до практичної реалізації проєкту. Була проведена критична оцінка наукових джерел, сформульовані власні погляди та визначені методологічні підходи, необхідні для досягнення поставленої мети дослідження.

2 ПОРІВНЯЛЬНИЙ АНАЛІЗ ІСНУЮЧИХ ТЕХНОЛОГІЙ

У попередньому розділі було здійснено огляд існуючих технологій та алгоритмічних рішень, що можуть бути застосовані для розробки програмного інтерфейсу на основі блокчейну з метою захисту та збереження медіа доказів військових злочинів. Проведено аналіз різних алгоритмів хешування та шифрування, технологій індексації даних, архітектурних підходів та засобів розробки. Це створило теоретичну основу для подальшої роботи.

Метою другого розділу є практичне дослідження та обґрунтування вибору конкретних рішень для реалізації поставлених завдань. У цьому розділі буде представлено:

- *Аналіз практичних аспектів* використання технологій, розглянутих у першому розділі, на основі зібраної інформації та досвіду, набутого під час практики.

- *Порівняння та оцінка* альтернативних рішень з акцентом на їх ефективності, надійності та відповідності вимогам проєкту.

- *Фактологічне підтвердження* існування проблеми збереження та захисту медіа доказів, визначеної у предметі дослідження, з урахуванням наукових надбань останніх років.

- *Апробація методик*, розглянутих у першому розділі, шляхом їх практичного застосування та оцінки результатів.

У розділі будуть наведені фактичні дані та порівняльні таблиці. Представлені матеріали будуть супроводжуватись тлумаченням та висновками, що дозволить визначити сутність досліджуваних процесів, їх характеристики та тенденції розвитку.

Аналіз проблеми здійснюватиметься з урахуванням як позитивних, так і негативних чинників, що впливають на функціонування системи. Це дозволить сформувати об’єктивну картину та обґрунтувати вибір найбільш доцільних рішень.

Під час підготовки розділу використовуватимуться методи аналізу та обробки даних, що забезпечить точність та надійність отриманих результатів. Особлива увага приділятиметься чіткому узагальненню фактичного матеріалу, його групуванню та аналізу, на основі чого будуть сформульовані пропозиції щодо подальшого розвитку та вдосконалення системи.

Таким чином, другий розділ стане логічним продовженням теоретичного аналізу, проведеного у першому розділі, та покликаний забезпечити практичне підтвердження обраних методик і технологій. Розділ сприятиме глибшому розумінню досліджуваної проблеми та надасть основу для формування конкретних рекомендацій та висновків у подальших розділах роботи.

2.1 Аналіз загальних характеристик блокчейну

Блокчейн є революційною технологією, яка має потенціал трансформувати багато галузей, надаючи нові можливості для безпечного, прозорого та ефективного управління даними та транзакціями. У контексті захисту та збереження медіа доказів військових злочинів блокчейн пропонує унікальні переваги, такі як незмінність даних, прозорість та децентралізація, що є критично важливими для забезпечення справедливості та правосуддя.

Розуміння основних принципів блокчейну та його можливостей є необхідним кроком для успішної реалізації проекту, спрямованого на збереження та захист важливих доказів. Подальші розділи роботи будуть присвячені конкретним аспектам використання блокчейну в нашому проекті, деталізації технічних рішень та аналізу їх ефективності.

2.2 Аналіз та порівняння відомих алгоритмів хешування

Для даного проєкту критично важливо обрати алгоритм хешування, який забезпечує високу стійкість до колізій та швидкість обчислень. Враховуючи сучасні стандарти та вимоги безпеки, оптимальним вибором є SHA-256. Він широко використовується, має перевірену безпеку та підтримується більшістю платформ та мов програмування. Альтернативно, для підвищення продуктивності можна розглянути використання BLAKE3, який пропонує високу швидкість без компромісів у безпеці. Результати порівняння розглянутих хеш-функцій можна бачити на табл. 2.1.

Планується дозволити легку заміну імплементації алгоритму хешування у прототипі завдяки використанню стандарту об’єктно-орієнтованого програмування [22] – впровадження залежностей Dependency Inversion та властивості фреймворку Dependency Injection (DI Container) шляхом абстрагування від деталей імплементації конкретного алгоритму. Це дозволить обирати алгоритм в залежності від поставлених завдань, об’єму транзакцій, платформи тощо.

Таблиця 2.1

Порівняльний аналіз алгоритмів хешування [23]

Алгоритм	Розмір хешу	Стійкість до колізій	Швидкість	Стан безпеки
MD5	128 біт	Низька	Висока	Небезпечний
SHA-1	160 біт	Низька	Середня	Небезпечний
SHA-256	256 біт	Висока	Середня	Безпечний
SHA-3	256 біт	Висока	Низька	Безпечний
BLAKE2	256 біт	Висока	Висока	Безпечний
BLAKE3	256 біт	Висока	Дуже висока	Безпечний

2.3 Аналіз та порівняння відомих алгоритмів асиметричного шифрування

Зважаючи на вимоги до безпеки та продуктивності, ECC є оптимальним вибором. Він забезпечує високу безпеку при менших розмірах ключів та кращу продуктивність порівняно з RSA. ECDSA може бути використаний для цифрових підписів, що є важливим для автентифікації та верифікації даних у нашому проєкті. Порівняння узагальнених даних про досліджені алгоритми асиметричного шифрування наведені у табл. 2.2.

Так само, як і з алгоритмом хешування, планується дозволити на рівні архітектури підмінювати імплементацію алгоритму асиметричного шифрування, що може бути корисно для гнучкості та ширшої підтримки програмного забезпечення.

Таблиця 2.2

Порівняльний аналіз алгоритмів асиметричного шифрування [24]

Алгоритм	Розмір хешу	Стійкість до колізій	Швидкість	Стан безпеки
MD5	128 біт	Низька	Висока	Небезпечний
SHA-1	160 біт	Низька	Середня	Небезпечний
SHA-256	256 біт	Висока	Середня	Безпечний
SHA-3	256 біт	Висока	Нижча	Безпечний
BLAKE2	256 біт	Висока	Висока	Безпечний
BLAKE3	256 біт	Висока	Дуже висока	Безпечний

2.4 Порівняльний аналіз JSON та XML

Поглянемо на формати передачі даних у контексті поточного дослідження. Врахуємо особливості доменної області та вимоги, поставлені до програмного продукту.

JSON є основним форматом передачі даних в RESTful API. Його легкий синтаксис та ефективність роблять його ідеальним для веб-сервісів, особливо у взаємодії з клієнтськими додатками на JavaScript.

XML часто використовується в SOAP-веб-сервісах та в середовищах, де потрібна складна валідація та самодокументованість.

У контексті розробки програмного інтерфейсу на основі блокчейну для захисту та збереження медіа доказів військових злочинів, важливо забезпечити ефективність, безпеку та зручність роботи з даними. Враховуючи проведений аналіз, можна зробити наступні висновки:

Продуктивність та ефективність передачі даних є критичними, оскільки система повинна бути готова до обробки великого обсягу інформації. JSON забезпечує менший розмір даних та швидший парсинг, що покращує продуктивність [25].

Простота інтеграції з веб-додатками та мобільними клієнтами. Оскільки клієнтські додатки можуть бути написані на JavaScript або інших мовах, JSON буде більш зручним форматом для взаємодії.

Безпека даних є ключовим фактором. JSON, при правильній обробці, може бути більш безпечним у контексті деяких вразливостей, властивих XML.

Потреба у валідації даних. Якщо система потребує строгого контролю структури та типів даних, можна використовувати JSON Schema для валідації JSON-документів.

Масштабованість та підтримуваність. JSON є більш популярним та сучасним форматом, що полегшує пошук інструментів та бібліотек, а також підтримку та розвиток системи.

Зважаючи на переваги та недоліки обох форматів, для розробки програмного інтерфейсу на основі блокчейну доцільно обрати JSON як основний формат передачі даних. Це обумовлено наступними причинами:

- *Ефективність*: JSON забезпечує менший розмір даних та швидкий парсинг, що підвищує продуктивність системи.
- *Сумісність*: Широка підтримка JSON у веб-технологіях та мовах програмування полегшує інтеграцію з різними клієнтами та сервісами.
- *Простота розробки*: Інтуїтивно зрозумілий синтаксис JSON спрощує розробку та підтримку коду.
- *Безпека*: Менша вразливість до деяких атак, властивих XML, підвищує безпеку системи.

Однак, якщо виникне потреба у використанні розширених можливостей валідації або роботи зі складними типами даних, можна розглянути використання JSON Schema для визначення та валідації структури даних.

Порівняння JSON та XML показує, що кожен з форматів має свої сильні та слабкі сторони. У контексті розробки сучасних веб-додатків та API, особливо тих, що потребують високої продуктивності та ефективності, JSON є більш прийнятним вибором. Він відповідає вимогам сучасних технологій, забезпечує легку інтеграцію та підтримку, а також сприяє швидкій та безпечній передачі даних.

Обравши JSON, ми забезпечуємо відповідність системи сучасним стандартам розробки програмного забезпечення та створюємо основу для ефективної та надійної роботи програмного інтерфейсу для захисту та збереження медіа доказів військових злочинів.

2.5 Використання GraphQL або gRPC у контексті проєкту

Хоча GraphQL та gRPC мають свої переваги, їх використання може бути недоцільним у контексті даного проєкту. GraphQL підходить для додатків з багатими та взаємозалежними даними, де клієнтам потрібно гнучко запитувати різні поля [26]. У нашому випадку структура даних є достатньо стабільною та передбачуваною, і переваги GraphQL можуть не виправдати складності його впровадження. А gRPC забезпечує високу продуктивність, але обмежена підтримка браузерами та складність у розробці можуть стати перешкодою. Крім того, вимоги до швидкості та двонаправленого стрімінгу не є критичними для даного програмного рішення.

2.5.1 Перспективи використання мікросервісної архітектури

Мікросервісна архітектура може бути розглянута в майбутньому, коли проєкт досягне значних масштабів. На початкових етапах розвитку проєкту монолітна архітектура з використанням MVC буде більш ефективною [27]. Це проглядається у простоті розгортання та управління: Менше компонентів для розгортання та моніторингу. Швидкості розробки**:** Менше витрат на налаштування інфраструктури. Зниження складності: Полегшує налагодження та тестування системи, інтеграцію з іншими рішеннями.

2.5.2 Заключні висновки щодо вибору архітектури

Зважаючи на всі розглянуті фактори, для розробки програмного інтерфейсу на основі блокчейну для захисту та збереження медіа доказів військових злочинів рекомендується наступне:

- *Архітектура:* Використання патерну MVC для чіткого розділення логіки додатку.

- *Протокол та формат даних:* RESTful API з використанням HTTP/HTTPS та формату JSON.

- *Можливість масштабування:* Монолітна архітектура з потенційною можливістю розбиття на мікросервіси в майбутньому.

- *Забезпечення безпеки:* Впровадження стандартних механізмів захисту на рівні архітектури та коду.

Таке поєднання дозволить створити ефективний, безпечний та масштабований додаток, який відповідає вимогам проєкту та сучасним стандартам розробки програмного забезпечення. Порівняльний аналіз досліджених комбінацій архітектурних рішень, протоколів та форматів передачі даних висвітлено в табл. 2.3.

Таблиця 2.3

Порівняння архітектурних рішень

Критерій	REST	GraphQL	gRPC
Протокол	HTTP/1.1	HTTP/1.1	HTTP/2
Формат даних	JSON, XML	JSON	Protobuf
Гнучкість запитів	Низька	Висока	Середня
Продуктивність	Середня	Середня	Висока
Кешування	Підтримується на рівні HTTP	Не підтримується	Обмежене
Підтримка стрімінгу	Ні	Ні	Так
Сумісність з браузерами	Висока	Висока	Низька
Складність розробки	Низька	Середня	Висока

В загальному, можемо бачити переваги використання обраного підходу в контексті даної роботи у таблиці 2.3. Вибір REST зумовлений особливостями та вимогами до продукту, порівняння виконано у контексті згаданих вимог.

2.6 Аналіз розглянутих варіантів індексації

Як наведено в попередньому розділі, використання коректного механізму індексації – важливий крок на шляху до створення системи, дружньої до користувача. Це дозволить проводити базові операції швидше, та розширить можливість для глибшого аналізу даних з часом, коли даних у блокчейні буде більше [28]. Ми повинні врахувати певні параметри та особливості алгоритмів індексації при виборі того, який використовувати. Це дозволить утилізувати особливості конкретної імплементації на користь продукту, і зменшить шанс проявлення її недоліків. У таблиці 2.4 проведений порівняльний аналіз розглянутих алгоритмів у контексті розроблюваної системи.

Таблиця 2.4

Порівняльний аналіз алгоритмів індексації [29]

Алгоритм	Час пошуку	Витрати пам’яті	Складність реалізації	Підтримка складних ключів	Придатність до проєкту
Лінійний пошук	O(n)	Низькі	Низька	Так	Низька
Бінарний пошук	O(log n)	Низькі	Низька	Обмежена	Середня
Хеш-таблиці	O(1)	Помірні	Середня	Так	Висока
B-дерева	O(log n)	Помірні	Висока	Так	Середня
Trie	O(k), де k - довжина ключа	Високі	Середня	Обмежена	Низька

Інвертовані індекси	O(1)	Високі	Висока	Так	Низька
Skip List	O(log n)	Помірні	Середня	Так	Середня

2.6.1 Обґрунтування вибору хеш-таблиць

Враховуючи потреби розроблюваного проєкту, хеш-таблиці є оптимальним вибором з нижченаведених причин:

- Висока продуктивність: Забезпечує доступ до елементів за час O(1), що критично для швидкого пошуку в великому обсязі даних [30].

- Підтримка складних ключів: Легко реалізується шляхом об’єднання кількох полів у один ключ, що відповідає нашим вимогам до індексації за декількома параметрами.

- Динамічність: Хеш-таблиці добре працюють з динамічними даними, дозволяючи ефективно додавати та видаляти елементи.

- Відносна простота реалізації: Хоча вимагають уважного підходу до вибору хеш-функції та обробки колізій, загалом хеш-таблиці не надто складні в реалізації.

2.6.2 Вибір та налаштування хеш-таблиць

Вибір хеш-функції

Щоб у результаті отримати ефективне рішення, що дозволить вирішити поставлені проблеми, поставимо вимоги до окремих елементів механізму індексації за допомогою хеш-таблиць [31]. Таким чином ми зможемо впевнитись у достатній ефективності в контексті даної роботи.

Якісна хеш-функція повинна рівномірно розподіляти ключі по таблиці, мінімізуючи колізії, бути швидкою в обчисленні, щоб не знижувати загальну

продуктивність. А також підтримувати складні ключі, об’єднуючи кілька полів в один хеш.

Прикладом може бути використання криптографічної хеш-функції, такої як SHA-256, хоча, якщо ставити на перше місце ефективність використання ресурсів, можна використовувати більш прості, некриптографічні хеш-функції [32].

Підтримка одинарних та складних ключів

За умови підтримки різних видів індексації, у нашому випадку, базової підтримки індексації по одному конкретному полю, а також, підтримки індексації по списку полів, ми надаємо кінцевому користувачеві ширше поле для дій. Також не обмежуємо себе, як проектанта, у видах операцій з пошуку даних у нашій системі. Тим самим, підвищуємо гнучкість більшості операцій.

Одинарні ключі. Для одиночних ключів все просто: ключ перетворюється хеш-функцією в індекс таблиці.

Складні ключі. Для складних ключів, які складаються з кількох полів, можна використовувати різні способи об’єднання даних.Можна об’єднати значення полів в один рядок і застосувати хеш-функцію. Або обчислити хеш кожного поля окремо та об’єднати їх за допомогою операцій XOR, додавання чи іншої функції. Головне – щоб функції при додаванні ключа в хеш-таблицю, та пошуку по ній, були однаковими.

Проведений аналіз показав, що для реалізації ефективного та швидкого пошуку в даному проєкті найбільш підходящим є використання хеш-таблиць для індексації. Реалізація власного алгоритму індексації на основі хеш-таблиць дозволяє оптимізувати його під специфічні вимоги проєкту, забезпечуючи високу продуктивність та надійність системи.

Використання хеш-таблиць для індексації в проєкті є оптимальним рішенням, яке відповідає вимогам до швидкості, ефективності та гнучкості системи.

Реалізація власного алгоритму індексації дозволяє забезпечити необхідний рівень продуктивності та адаптуватися до специфічних потреб зберігання та пошуку медіа доказів військових злочинів.

Також можливе абстрагування від конкретної імплементації індексування. За підходящої архітектури можна без великих зусиль використовувати різні імплементації індексів в залежності від функціональних вимог до програмного забезпечення. Використовувати різні імплементації можна навіть одночасно, за наявності, наприклад, вимоги швидкого сортування по певному полю. З цією задачею бінарне дерево впораються краще, ніж хеш-таблиця.

2.7 Аналіз та порівняння розглянутих видів тестування програмного забезпечення

У процесі розробки програмного інтерфейсу на основі блокчейну для захисту та збереження медіа доказів військових злочинів було прийнято рішення використовувати комбінацію методів тестування для забезпечення максимальної якості та надійності програмного забезпечення. Основний акцент був зроблений на юніт-тестуванні, доповненому ручним тестуванням на різних етапах розробки. Такий підхід покликаний дозволити виявити та виправити помилки на ранніх стадіях, забезпечити відповідність поставленим функціональним вимогам та покращити загальну якість системи.

2.7.1 Обґрунтування вибору юніт-тестування

Юніт-тестування дозволяє перевіряти окремі модулі або компоненти програми в ізоляції, що сприяє високій якості коду. Це особливо важливо в контексті блокчейн-додатків, де помилки в одному модулі можуть призвести до серйозних наслідків, таких як втрата даних або компрометація безпеки.

Використання юніт-тестів забезпечує можливість виявлення помилок на ранніх етапах розробки. Це знижує час, витрачений на їх виправлення, та запобігає накопиченню дефектів у системі.

При внесенні змін у код або додаванні нових функціональностей, юніт-тести допомагають впевнитися, що існуюча функціональність не була порушена. Це спрощує процес рефакторингу та підтримки коду.

Юніт-тести можуть слугувати додатковою документацією, показуючи, як певні модулі повинні працювати. Це полегшує розуміння коду для усіх членів команди, а також, для нових розробників.

Для імплементації юніт-тестування розглядається фреймворк NUnit, який є популярним інструментом для тестування в середовищі .NET [33].

Використання юніт-тестування за допомогою NUnit у поєднанні з ручним тестуванням повинне забезпечити високий рівень якості та надійності програмного забезпечення. Такий підхід сприяє виявленню та виправленню помилок на ранніх етапах розробки, підвищує ефективність процесу розробки та забезпечує відповідність системи вимогам та очікуванням користувачів.

2.8 Реалізація абстрактних правил для забезпечення цілісності даних

В попередньому розділі розглянута важливість зручного додавання та зміни бізнес-правил у подібній системі. Далі проведемо аналіз проблеми та знайдемо рішення, яке дозволить вирішити задачу та надмірно не ускладнить програмне забезпечення.

Отже, в контексті розробки програмного інтерфейсу на основі блокчейну для захисту та збереження медіа доказів військових злочинів, важливо не лише забезпечити безпечне зберігання даних, але й гарантувати цілісність та коректність операцій, що виконуються в системі. Для досягнення цієї мети був реалізований модуль абстрактних правил, які визначають логіку перевірки та валідації даних перед їх додаванням до блокчейну.

Реалізація абстрактних правил спрямована на забезпечення:

- *Цілісності даних:* Запобігання несанкціонованим або некоректним змінам у даних, що зберігаються в блокчейні.

- *Автентичності транзакцій:* Гарантування того, що лише авторизовані особи можуть виконувати певні дії, такі як реєстрація чи перевірка певних об’єктів.

- *Відповідності бізнес-логіці:* Впровадження специфічних правил та умов, що відповідають вимогам системи та правовим нормам.

Система абстрактних правил побудована на основі інтерфейсів та узагальнених класів, що дозволяє легко розширювати та модифікувати правила без впливу на основну структуру додатку [34].

2.8.1 Механізм виконання правил

Послідовне застосування правил: Перед додаванням нової транзакції до блокчейну, модуль послідовно застосовує всі відповідні правила до цієї транзакції.

Обробка винятків: У випадку невідповідності правилу, генерується виключення з детальним описом помилки, що дозволяє користувачу розуміти причину відмови виконувати запит.

Розширюваність: Архітектура дозволяє додавати нові правила або модифікувати існуючі без суттєвих змін в коді основної системи.

2.8.2 Приклад застосування правил

Сценарій реєстрації нового медіа-об’єкта:

- Користувач бажає зареєструвати новий об’єкт (наприклад, цифрове фото).

- Система перевіряє, чи не зареєстрований цей об’єкт раніше.

- Якщо об’єкт унікальний, транзакція успішно додається до блокчейну.

- Якщо об’єкт вже зареєстрований, система відмовляє в реєстрації та повідомляє про дублювання.

Реалізація абстрактних правил є важливим кроком у забезпеченні надійності та безпеки системи зберігання та управління медіа доказами військових злочинів. Впровадження таких правил дозволить підвищити довіру користувачів до системи за рахунок прозорості та чіткості дій. Окрім того, дозволить забезпечити відповідність правовим нормам та вимогам щодо обробки та зберігання доказів. А також створити основу для подальшого розвитку системи, додаючи нові функціональності та можливості без суттєвих змін в архітектурі.

Цей підхід демонструє, як продумана архітектура та використання абстракцій можуть сприяти створенню гнучких, масштабованих та безпечних систем, що відповідають сучасним вимогам до програмного забезпечення у критично важливих сферах.

2.9 Висновки до розділу

У даному розділі було проведено комплексний порівняльний аналіз існуючих технологій, алгоритмічних та технічних рішень, а також допоміжних програмних засобів, які можуть бути використані для розробки програмного інтерфейсу на основі технології блокчейн, для захисту та реєстрації медіа доказів військових злочинів.

Алгоритми хешування були розглянуті з метою знаходження найбільш безпечного, ефективного та перевіреного для забезпечення цілісності даних. У результаті аналізу було обрано SHA-256 як найбільш оптимальний та збалансований алгоритм, який забезпечує високий рівень безпеки.

При синтезі алгоритмів асиметричного шифрування було визначено, що ECC (Elliptic Curve Cryptography) та його варіант ECDSA надають переваги у вигляді відносно меншого розміру ключів при збереженні високого рівня безпеки. Це

робить їх оптимальним вибором для даної системи. Ефективність та швидкодія на першому місці.

Порівняльний аналіз архітектурних рішень дав зрозуміти, що використання архітектурного патерну MVC у поєднанні з RESTful API та форматом передачі даних JSON є виправданим. Такий підхід забезпечує простоту та зрозумілість розробки, легку інтеграцію з різними клієнтами та системами, а також відповідає сучасним стандартам веб-розробки.

У підрозділі, присвяченому порівнянню JSON та XML, було обґрунтовано вибір JSON як основного формату передачі даних. JSON має менший об’єм, простіший синтаксис, швидший парсинг та широку підтримку в різних мовах програмування, що підвищує продуктивність та зручність використання системи.

Аналіз варіантів індексації даних, в даному випадку блоків, привів до вибору хеш-таблиць як основного алгоритму індексації. Хеш-таблиці забезпечують швидкий доступ до даних з часовою складністю O(1), підтримують як одиночні, так і складні ключі, в залежності від імплементації, а також дозволяють ефективно працювати з великими обсягами інформації. Це критично важливо для забезпечення швидкого пошуку та обробки медіа доказів у системі.

При розгляді допоміжних програмних засобів та засобів розробки було обрано C# як основну мову програмування та ASP.NET Core як фреймворк, на який буде покладено відповідальність контролю життєвих циклів об’єктів програми. Враховуючи що основна частина дослідження буде проведена на комп’ютері під управлінням ОС MacOS, JetBrains Rider був обраний як інтегроване середовище розробки завдяки своїй продуктивності, широкій підтримці зовнішніх плагінів та підтримці на платформі MacOS [35]. Для контейнеризації додатку було вирішено використовувати Docker, що забезпечує ізоляцію середовища виконання та спрощує процес розгортання.

У підсумку, проведений аналіз дозволив досягти таких цілей:

- Визначити підходящі алгоритми криптографії для забезпечення безпеки даних.

- Обрати архітектурні підходи та формати даних, що забезпечують продуктивність та гнучкість системи.

- Визначити, які будуть використані алгоритми індексації та типи даних, щоб відповідати вимогам до швидкого доступу та пошуку даних.

- Обрати сучасні та ефективні інструменти розробки та контейнеризації, що сприяють швидкій та якісній реалізації та тестуванню проєкту.

Також вирішено проблему інтеграції бізнес-правил у систему. Шляхом абстрагування залежностей, вдалося спроєктувати модуль, що відповідає виключно за забезпечення виконання бізнес-правил у блокчейні.

Підсумовуючи, було сформовано міцну теоретичну та технологічну базу для проєктування, подальшої імплементації програмного інтерфейсу, а також його розвитку. Обрані рішення покликані забезпечити високу безпеку, ефективність та масштабованість системи, без чого досягти достатнього рівня захисту та ефективного збереження медіа доказів військових злочинів було б складно. Наступний розділ роботи буде присвячений практичній реалізації цих рішень та оцінці їх ефективності в умовах, наближених до реальних.

3 КОНСТРУЮВАННЯ ТА АНАЛІЗ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У попередніх розділах було проведено теоретичний аналіз та дослідження існуючих технологій і методів, що можуть бути використані для розробки програмного інтерфейсу на основі блокчейну з метою захисту та збереження медіа доказів військових злочинів. Було визначено основні вимоги до системи, розглянуто різні алгоритми та архітектурні підходи, а також проведено аналіз проблематики в контексті сучасних тенденцій.

Метою даного розділу є представлення конструктивних пропозицій та практичних результатів, спрямованих на досягнення поставленої мети дослідження. У цьому розділі застосовані творчо-комбінаторні навички, шляхом використання теоретичних знань на практиці та розробляючи конкретні програмні рішення для досягнення визначених цілей.

У розділі буде висвітлено наступні аспекти:

1. *Приклади взаємодії з програмним інтерфейсом:* будуть представлені сценарії використання системи, описані за допомогою діаграм BPMN та варіантів використання (Use Case). Це дозволить детальніше проілюструвати послідовність дій користувача та взаємодію з системою, що покликане сприяти кращому розумінню функціональності та потенційних шляхів удосконалення.

2. *Результати створення програмного інтерфейсу:* буде наведено опис ключових класів та їх взаємодії, а також представлені діаграми класів різних рівнів системи. Ціль такої документації – надати повне уявлення про архітектуру розробленого програмного забезпечення та підходи, використані при його реалізації.

3. *Покриття юніт-тестами:* буде продемонстровано підхід до тестування розробленого програмного забезпечення за допомогою юніт-тестів,

наведено приклади кількох тестових сценаріїв. Це, у свою чергу, підтвердить надійність та якість коду, а також відповідність його функціональності вимогам.

4. *Використані зовнішні утиліти та бібліотеки:* буде проаналізовано використані зовнішні утиліти та бібліотеки, обґрунтовано їх вибір та описано, як вони інтегровані в систему. Це дозволить оцінити ефективність використання сторонніх рішень та їх вплив на розробку.

5. *Результат аналізу якості коду:* буде проведено аналіз якості коду за допомогою статичного аналізатора, включаючи оцінку цикломатичної складності, виявлені небажані залежності та можливі антипатерни. Це надасть змогу оцінити оптимальність та підтримуваність коду, а також визначити напрямки для його вдосконалення.

Розділ базується на результатах, отриманих у теоретико-методичному та аналітико-дослідницькому розділах, і спрямований на практичне застосування розглянутих методик та технологій. Використання сучасних підходів до розробки програмного забезпечення та інструментів аналізу дозволить забезпечити високий рівень якості та ефективності розробленої системи.

Таким чином, третій розділ є кульмінацією роботи, де теоретичні знання та аналітичні висновки втілюються у конкретні практичні рішення. Він має демонструвати здатність розробляти інноваційні підходи та впроваджувати їх у життя, сприяючи досягненню поставленої мети дослідження та внеску у розвиток обраної галузі.

3.1 Приклади взаємодії з програмним інтерфейсом

Для опису бізнес процесу, що означає дію перевірки наявності певного медіа-об’єкта в числі зареєстрованих, а також авторства (власності) об’єкта, використовується BPMN модель, що зображена на рисунку 3.1.

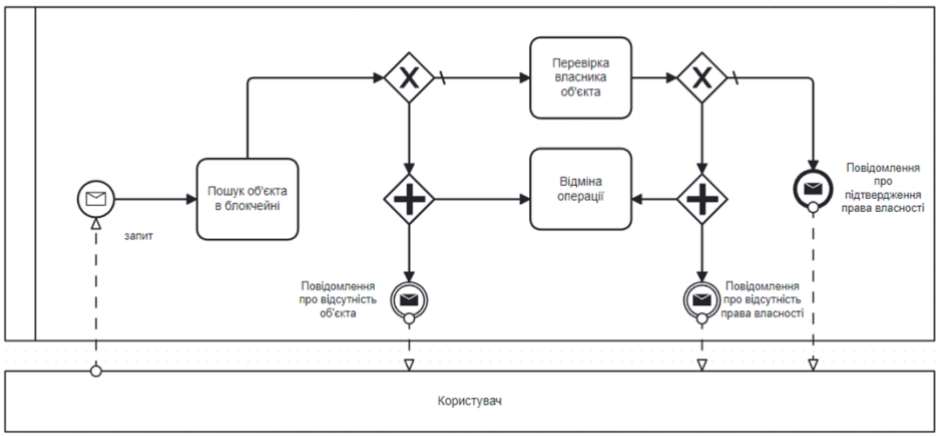


Рис. 3.1 – BPMN-діаграма процесу перевірки наявності та авторства певного об’єкта

Опис послідовності підтвердження права власності на віртуальний об’єкт:

- користувач надсилає запит на підтвердження наявності об’єкта та авторства на медіа-об'єкт;
- відбувається перевірка валідності запиту;
- відбувається пошук відповідного об'єкта в блокчейні;
- якщо об'єкт знайдено, відбувається перевірка, чи є певний користувач автором цього об'єкта;
- якщо користувач має авторство об'єкта, відбувається підтвердження наявності об’єкта з запитаним авторством, і відправляється відповідний результат;
- якщо відправлений ідентифікатор користувача не є ідентифікатором автора об'єкта, результатом операції є повідомлення з відповідною інформацією;
- якщо об'єкт не вдалося знайти в числі зареєстрованих, користувач отримує повідомлення про те, що запитаний об'єкт не зареєстрований.

Подібним чином описано процес реєстрації обєкта в системі. Визначені основні етапи проводження такого запиту. Схематичне зображення процесу висвітлене на рис. 3.2 у нотації BPMN.

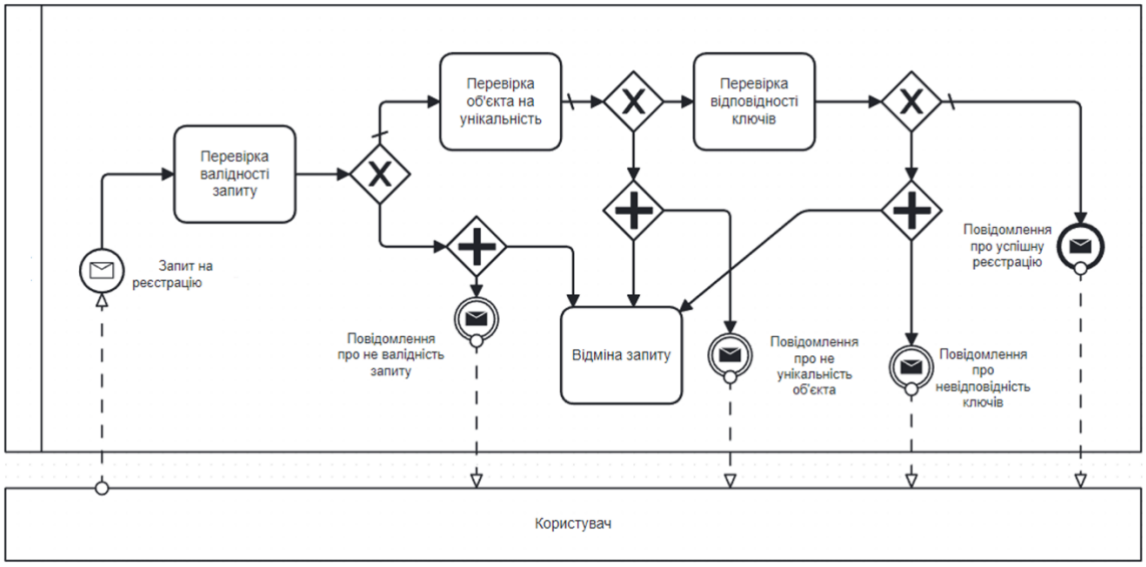


Рис. 3.2 – BPMN-діаграма процесу реєстрації медіа-об’єкта

Опис послідовності реєстрації віртуальний об’єкта у власність:

- надсилається запит на реєстрацію медіа-доказу та авторства на цей об'єкт;
- проводиться перевірка правильності запиту;
- задіюється логіка для визначення унікальності запитуваного медіа-об'єкта в блокчейні;
- активується логіка для визначення відповідності приватного і публічного ключів актора;
- якщо об'єкт є унікальним, новий блок додається в кінець блокчейна та присвоюється авторство користувача;
- актор отримує підтвердження реєстрації нового об’єкта;
- якщо об'єкт вже зареєстрований в блокчейні, користувач отримує повідомлення про те, що такий об’єкт вже існує і не може бути зареєстрований;
- у випадку невідповідності приватного та публічного ключів, результат запиту – помилка з даними перевірки.

3.2 Результати створення програмного інтерфейсу

Розроблену систему можна поділити на три основних рівні:

Нижній рівень (Untyped layer). У фундаментальному шарі системи розташована логіка, яка відповідає за обробку даних без визначеного типу. Оскільки система оперує різноманітними типами даних (об’єктами різних класів), цей шар забезпечує уніфікацію процесу їх збереження. Він містить інтерфейси та реалізації для читання, запису, обробки та зберігання інформації. Ці компоненти відіграють ключову роль у підтримці стійкості та безпеки даних.

Середній рівень (Business Logic Layer). Цей шар надає абстракції для побудови бізнес-логіки програмного забезпечення. Він включає інтерфейси для створення об’єктів, які допомагають виконувати різні бізнес-завдання: аутентифікацію користувачів, управління транзакціями, розширену валідацію даних (додатково до базової, що існує на нижньому рівні), правила додавання блоків та інше. Також тут знаходяться абстракції для створення об’єктів, які забезпечують денормалізацію схеми збереження даних для підвищення швидкості доступу.

3.2.1 Рівень програмного інтерфейсу (API Layer)

Цей рівень відповідає за обробку запитів, що надходять через протокол HTTP. Він містить контролери та сервіси для опрацювання вхідних HTTP-запитів і формування відповідей.

У наступних підрозділах детально розглянуто кожен із цих шарів. На рис. 3.3 представлено діаграму класів основних компонентів двох нижніх шарів.

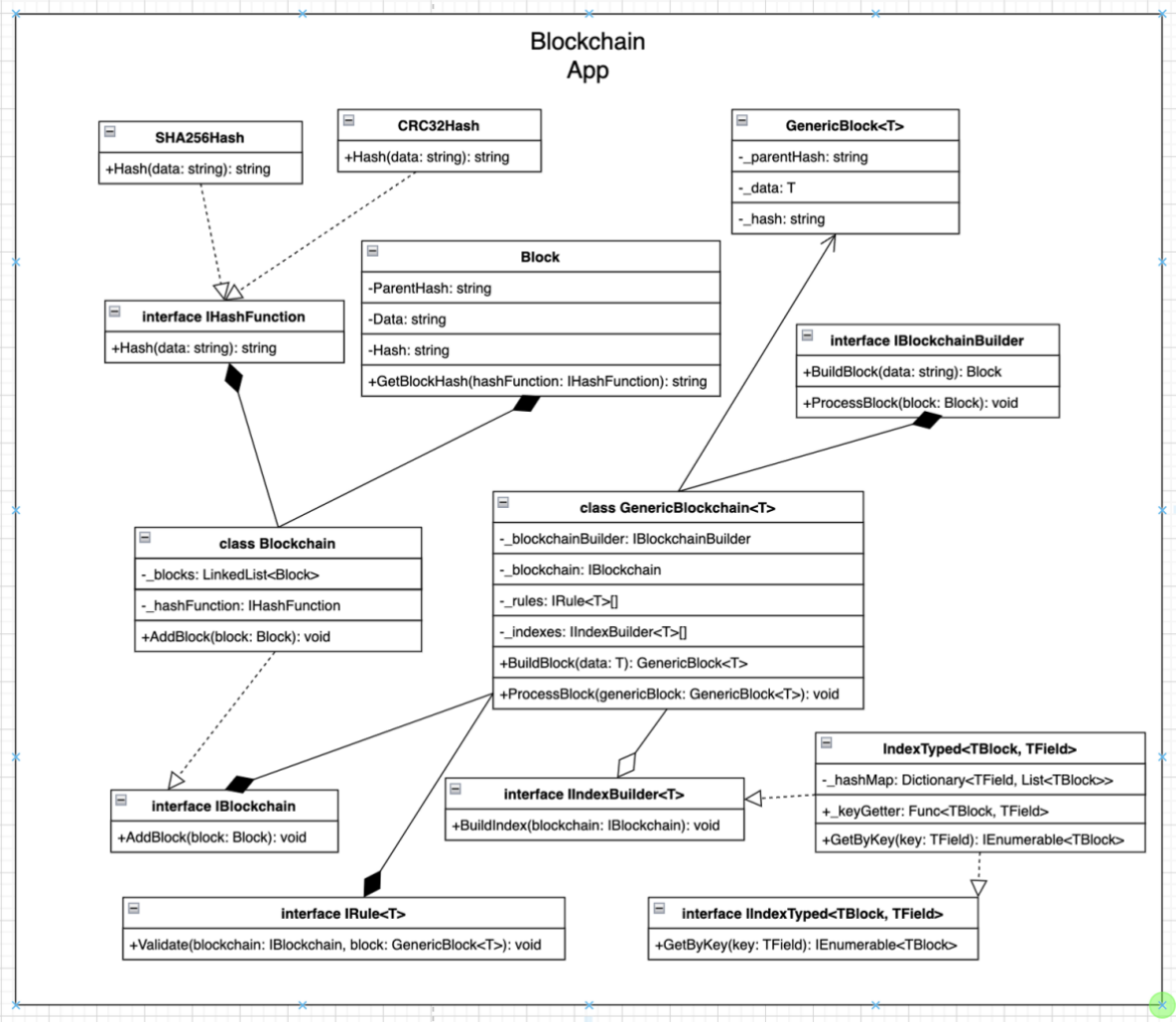


Рис. 3.3 – Діаграма класів ключових елементів двох перших рівнів

Подібна архітектура дозволяє досягти необхідної гнучкості для реалізації специфічних бізнес-завдань, забезпечуючи високий рівень стабільності та безпеки програмного забезпечення.

3.2.2 Архітектура верхніх шарів програмного забезпечення

На верхньому рівні системи знаходяться два основні підпроєкти: HighLevelBlockschain та BlockschainApplication.

HighLevelBlockschain слугує абстракцією для реалізації різноманітних бізнес-завдань. Він містить усі необхідні функціональні можливості для роботи з блокчейном: створення нових блоків, додавання транзакцій до блоків, валідацію та

інтеграцію блоків у ланцюжок. HighLevelBlockschain використовує компоненти нижнього рівня для забезпечення коректної роботи з даними блокчейну.

BlockschainApplication — це клас, що реалізує інтерфейс для обробки HTTP-запитів. Він містить декілька обробників, відповідальних за різні аспекти взаємодії користувача з блокчейном. Використовуючи функціонал, наданий HighLevelBlockschain, BlockschainApplication реалізує логіку в межах контрактів, визначених нижчими шарами. Це включає правила, що регулюють додавання нових транзакцій, наприклад, передачу прав власності іншому користувачу або реєстрацію нового віртуального об’єкта у власність. Загалом, верхній рівень використовує можливості нижчих шарів для виконання відповідних операцій над блокчейном у контексті бізнес-правил.

BlockschainApplication співпрацює з HighLevelBlockschain для забезпечення повної функціональності системи. При отриманні HTTP-запиту від користувача, BlockschainApplication звертається до методів HighLevelBlockschain для його обробки. Наприклад, якщо користувач бажає додати нову транзакцію до блокчейну, BlockschainApplication приймає цей запит і викликає відповідні методи HighLevelBlockschain для створення та додавання транзакції. На рисунку 3.4 показано детальну діаграму взаємодії компонентів цих двох підпроєктів.

Такий підхід забезпечує високий рівень абстракції та інкапсуляції, що дозволяє легко модифікувати та розширювати функціональність системи без впливу на інші її частини. Це також полегшує процес тестування програмного забезпечення, сприяючи ефективному проведенню юніт-тестування, інтеграційного тестування та інших видів перевірок.

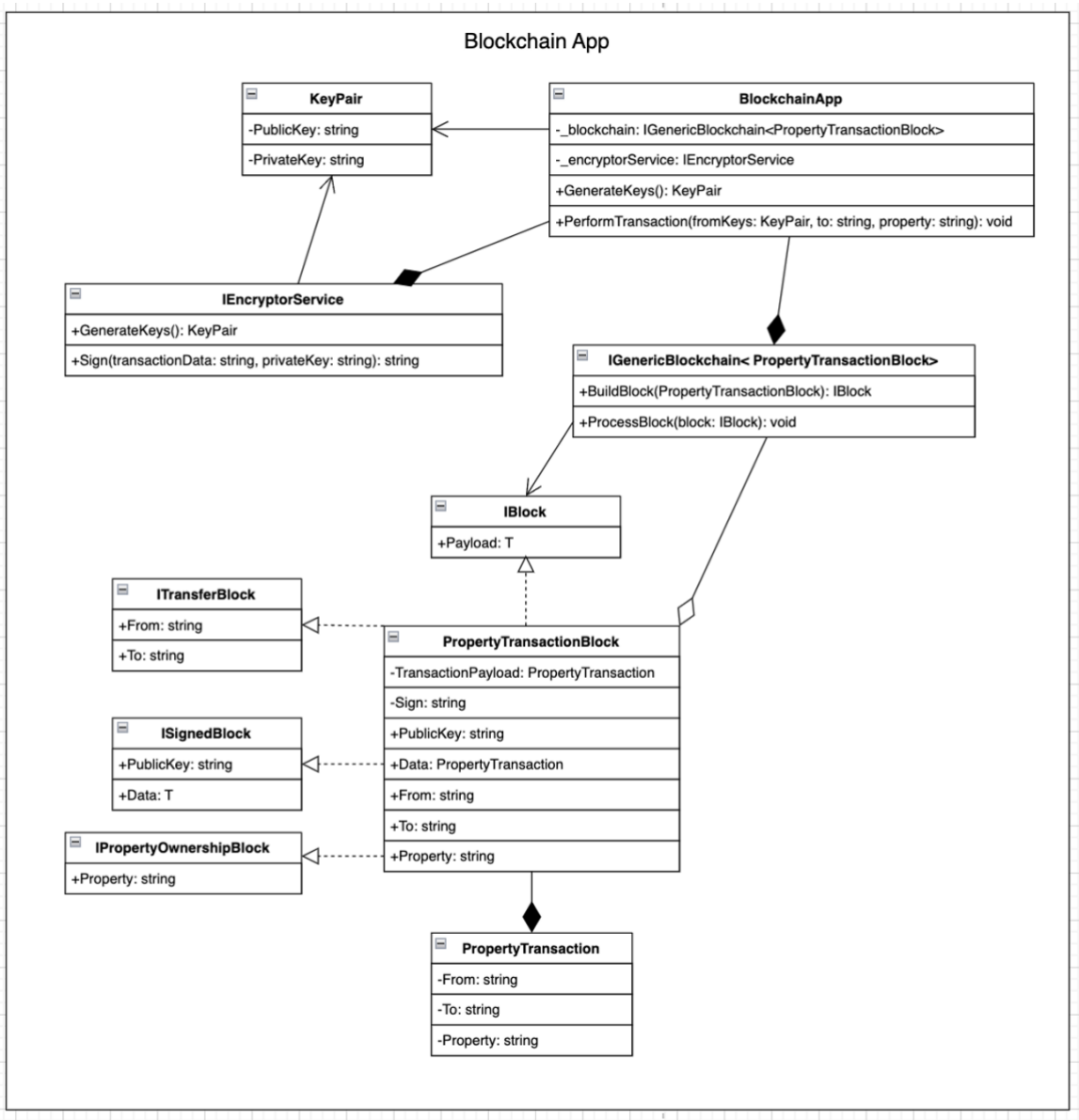


Рис. 3.4 – Діаграма класів елементів верхніх рівнів системи

3.2.3 Алгоритм індексування блоків

Однією з основних функцій нашого програмного забезпечення є індексування блоків у блокчейні, що дозволяє швидко знаходити блоки за заданими ключами. Для цього використовується спеціалізований алгоритм індексування, який працює за таким принципом:

- *Унікальний ключ блоку:* кожен блок у блокчейні асоціюється з унікальним ключем, визначеним на основі його вмісту або певних атрибутів.

- *Обчислення та збереження ключа:* при додаванні нового блоку до ланцюжка, його ключ обчислюється та зберігається разом із самим блоком, що дозволяє його ідентифікувати в майбутньому.

- *Додавання до хеш-таблиці:* ключ блоку та посилання на блок додаються до хеш-таблиці. Якщо в таблиці вже існує запис з таким ключем, новий блок додається до списку значень, пов'язаних із цим ключем, що дозволяє підтримувати множинні блоки для одного ключа.

- *Пошук за ключем:* при необхідності знайти блок за ключем, алгоритм звертається до хеш-таблиці. Якщо ключ присутній, повертається список усіх блоків, асоційованих із цим ключем, що забезпечує швидкий доступ до необхідної інформації.

Цей підхід до індексування значно підвищує ефективність роботи блокчейну, оскільки дозволяє уникнути повного перебору блоків при пошуку та забезпечує оперативний доступ до даних.

3.2.4 Структури даних

Блок та блокчейн

У контексті нашого програмного забезпечення, **блок** є фундаментальною структурною одиницею блокчейну. Він складається з кількох ключових компонентів:

- *Транзакції*: кожен блок містить набір транзакцій, які були здійснені в мережі. Транзакції є записами про передачу прав власності на віртуальні об'єкти між користувачами. Кожна транзакція включає детальну інформацію про відправника, отримувача, об'єкт транзакції та цифровий підпис відправника, який підтверджує його згоду на передачу прав.

- *Хеш попереднього блоку*: для забезпечення цілісності та незмінності ланцюга блоків, кожен блок містить хеш попереднього блоку. Це гарантує, що будь-яка зміна в попередньому блоці призведе до зміни хешу, порушуючи цілісність ланцюга.

- *Комбінований хеш*: для додаткового захисту від підробок, кожен блок містить хеш, обчислений на основі хешу попереднього блоку та власних даних блоку. Це створює тісно пов’язану ланцюгову структуру, де кожен блок залежить від попереднього, забезпечуючи незмінність та можливість відстеження всіх транзакцій.

Важливо зазначити, що блок не має окремого унікального ключа, оскільки його унікальність визначається його вмістом та хешем попереднього блоку. Хеш блоку, обчислений на основі його даних, фактично слугує його унікальним ідентифікатором у системі.

Для ефективного індексування блоків за ключами використовується хеш-таблиця. Вона забезпечує швидкий доступ до блоків за певними критеріями, що особливо важливо при великому обсязі даних у блокчейні. Кожен ключ у хеш-таблиці відповідає списку блоків, які задовольняють певне правило або критерій, визначене при створенні індексу, подібно до індексів у базах даних.

GenericBlock<T> служить основою для створення всіх інших блоків у системі. Ця структура включає такі ключові поля:

- *Hash*: хеш-сума блоку, яка обчислюється за допомогою хеш-функції, заданої в конфігурації проєкту. Цей хеш слугує для ідентифікації блоку та забезпечення його унікальності.

- *ParentHash*: хеш-сума попереднього блоку в ланцюжку, що забезпечує зв’язок між блоками та підтримує цілісність блокчейну.

- *Raw*: сирі, необроблені дані блоку, які представляють початкову інформацію перед її обробкою.

- *Data*: об’єкт даних типу T, який може бути будь-якого типу залежно від бізнес-завдань, що вирішуються сервісом. Це дозволяє блоку зберігати різноманітні дані, необхідні для функціонування системи. Тип цього об’єкта детально описаний у таблиці 3.1.

Таблиця 3.1

Опис типу GenericBlock<T>

Тип	Назва поля	Тип даних	Опис
GenericBlock<T>	Hash	string	хеш-сума блоку
	ParentHash	string	хеш-сума батьківського блоку
	Raw	string	сирий, сереалізований блок даних
	Data	<T>	типізовані дані для обробки блоку на рівні бізнес-сервісів

Розробник, який підтримує або розширює програмне забезпечення, не буде створювати екземпляри GenericBlock<T> безпосередньо. Ця структура використовується внутрішніми сервісами середнього рівня для побудови блокчейну. Вона є одним із ключових компонентів, що забезпечують безпеку та

цілісність даних у блокчейні, а її створення та обробка відбувається автоматично під час виконання транзакцій.

Структура Block є базовим елементом, який зберігається в блокчейні. Кожен блок містить такі поля:

- ParentHash: хеш попереднього блоку, що забезпечує неперервність та послідовність ланцюжка блоків.
- Data: поле для збереження даних транзакції або іншої відповідної інформації.
- Hash: унікальний хеш-код самого блоку, обчислений на основі його вмісту. Цей хеш використовується для перевірки цілісності блоку та його ідентифікації.

Детальний опис структури Block представлений у таблиці 3.2.

Таблиця 3.2

Опис типу Block

Тип	Назва поля	Тип даних	Опис
Block	ParentHash	string	хеш-сума попереднього блоку
	Data	string	дані транзакції
	Hash	string	хеш-сума блоку

Поле Data може містити дані будь-якого типу. У рамках цього програмного забезпечення це зазвичай об’єкт типу PropertyTransaction. Проте завдяки універсальному дизайну блоку та використанню абстрактного типу для

збереження даних транзакцій, блок може бути адаптований для зберігання будь-яких необхідних типів даних, залежно від конкретних потреб використання.

Структура Block виконує роль фундаментальної будівельної одиниці блокчейну і знаходиться на найнижчому рівні архітектури. Вона не лише зберігає дані, але й забезпечує цілісність та безпеку інформації всередині блокчейну. Це досягається завдяки використанню криптографічних хеш-функцій та зв’язуванню блоків через поле ParentHash, що робить систему стійкою до змін та зовнішніх впливів.

3.3 Тестування програмного забезпечення

Важливість тестування програмного забезпечення було доведено в попередніх розділах дослідження. Фокус цього підрозділу – опис проведених тестів, наведення прикладів та результати тестування.

У таблицях 3.3-3.5 наведені згруповані юніт-тести. Вони згруповані по тому, який саме шар системи вони повинні тестувати.

Таблиця 3.3

Група тестів 1

Назва тесту	AuthenticityBlockTests
Модуль	Перевірка коректності реалізації об’єкта
Початковий стан системи	Програма повинна працювати коректно без привязки до стану.
Вхідні дані	Тестові дані, які включають значення OwnerKey, Media, Time та Sign.
Опис тесту	Цей набір тестів перевіряє основні функції TransactionBlock, який імплементує кілька інтерфейсів. Ми використовуємо тестові значення і перевіряємо, чи

	вони правильно оброблюються через певні поля інтерфейсів. Кожен тест-кейс повинний згенерувати новий блок з відомими значеннями і потім порівняти отримані поля з очікуваними значеннями.

Таблиця 3.4

Група тестів 2

Назва групи	BlockschainAppTests
Модуль	Перевірка коректної роботи вижчого рівня системи
Початковий стан системи	Програма повинна працювати коректно без привязки до стану.
Вхідні дані	Тестові дані, які включають ключі OwnerKey, Media, Signature.
Опис тесту	Цей набір тестів перевіряє основні функції BlockschainApp, вижчого рівня системи, включаючи генерацію ключів і реєстрацію транзакцій. Ми використовуємо тестові об’єкти для IGenericBlockschain та IEncryptorService для імітації взаємодії з цими сервісами. В тесті генерації ключів ми перевіряємо, чи повертаються очікувані ключі від IEncryptorService. У тесті виконання транзакцій ми впевнюємося, що викликаються методи побудови і обробки блоків, а у кінці тестуємо наявність блоків у списку.

Таблиця 3.5

Група тестів 3

Назва тесту	BlockschainTests
Модуль	Перевірка коректності роботи середнього рівня системи
Початковий стан системи	Тести повинні виконуватися успішно в будь-якому стані системи.
Вхідні дані	Тестові дані включають різні рядкові значення (_data та _moreData).
Опис тесту	Цей набір тестів перевіряє основні функції Blockschain, включаючи додавання блоків. За допомогою тестового об’єкта IHashFunction, що повертає вхідний рядок у вигляді хешу, для імітації реального хешування. В тестах додавання блоків ми створюємо нові блоки з тестовими значеннями _data та _moreData і додаємо їх до блокчейну, а потім тестуємо, чи вони були додані без помилок.

Як результат проведених тестів, можна звернутись до значення відсотка покриття юніт-тестами [36]. Відсоток тестового покриття та кількості тестів в кожному з модулів (груп) можна побачити на рисунках 3.5 та 3.6 відповідно.

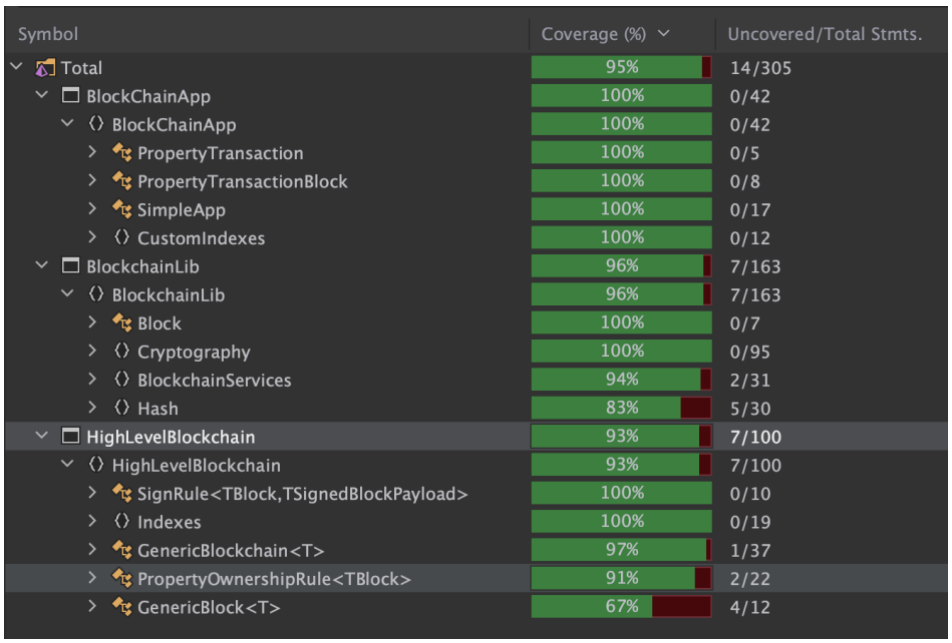


Рис. 3.5 – відсоток покриття юніт-тестами

Рисунок 3.6 – Усі наявні групи юніт-тестів

3.4 Використання зовнішніх бібліотек та утиліт

У ході розробки програмного забезпечення були залучені різноманітні бібліотеки та утиліти, які суттєво спростили процес створення та підвищили якість кінцевого продукту.

У таблиці 3.6 представлено детальний опис утиліт та додаткового програмного забезпечення, що використовувалися протягом процесу розробки.

Таблиця 3.6

Опис використаних утиліт

№ п/п	Назва утиліти	Опис застосування
1	JetBrains Rider	Головне IDE (середовище розробки), відладки (debug) та роботи з системою контролю версій програмного продукту.
2	Postman	Програмне забезпечення використане для мануального тестування запитів по протоколу HTTP. Утиліта була використана для тестування API інтерфейсу, та проведення клієнтських запитів.
3	Git	Система контролю версій, що була використана в роботі.
4	Docker	Програмне забезпечення для контейнеризації.

3.5 Аналіз якості програмного забезпечення

Для оцінки якості програмного коду проекту було використано інструмент NDepend [37]. Цей аналіз базувався на ряді метрик, серед яких цикломатична складність, залежності між різними частинами коду та виявлення антипатернів [38].

Цикломатична складність є метрикою, що допомагає визначити складність програмного коду [39]. Вона вимірюється кількістю незалежних лінійних шляхів

через вихідний код програми та є важливим індикатором для визначення необхідної кількості тестових випадків для повного покриття коду.

Згідно зі звітом, загальна цикломатична складність для неабстрактних методів становить 1,61, при цьому медіанне значення дорівнює 1,49. Максимальне значення цикломатичної складності, яке було зафіксовано, становить 12 і стосується методу Execute(), що належить типу HighLevelBlockschain.MediaAuthenticityRule<TBlock>.

Ці показники свідчать про те, що в цілому код добре структурований і не має надмірної складності. Однак у деяких випадках, як-от метод Execute у згаданому класі, підвищена складність може вказувати на потребу в додатковому рефакторингу та оптимізації цього фрагмента коду.

3.5.1 Загальні результати статичного аналізу коду

На рис. 3.7 представлено узагальнений результат аналізу за основними метриками.

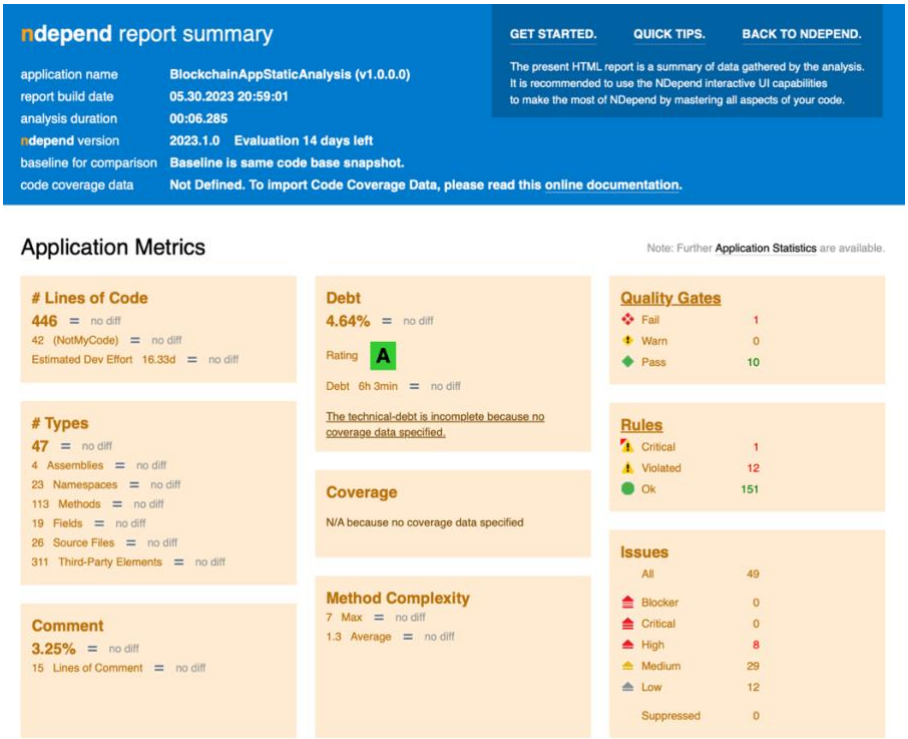


Рисунок 3.7 – Основна сторінка статичного аналізу NDepend

З цього рисунка видно кількість правил, відповідність яким було перевірено в коді, а також кількість правил, які виконуються та які порушуються.

На рис. 3.8 показані результати аналізу високого рівня щодо дотримання загальних принципів об’єктно-орієнтованого програмування (ООП), дизайну та архітектури.

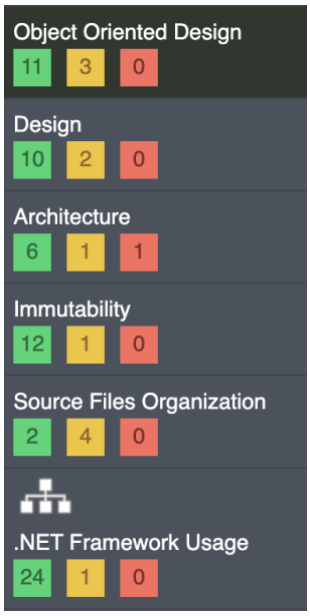


Рисунок 3.8 – Згорнуте представлення статичного аналізу NDepend

Рис. 3.9 демонструє розгорнуті результати перевірки програмного забезпечення на відповідність контрольним точкам якості коду (Quality Gates).

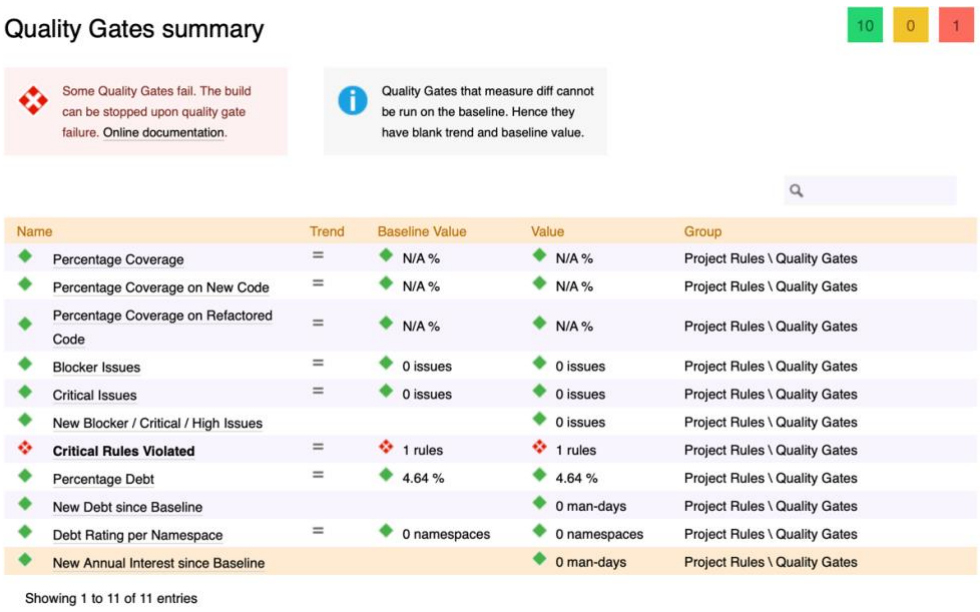


Рис. 3.9 – Розгорнутий результат підтримки коду контрольних точок якості статичного аналізу NDepend

Статичний аналіз дозволив ідентифікувати слабкі місця в програмному забезпеченні, що спростить їх локалізацію та виправлення в майбутньому.

3.6 Висновки до розділу

У третьому розділі було представлено конструктивні пропозиції та практичні результати, спрямовані на досягнення мети дослідження — розробки програмного інтерфейсу на основі блокчейну для захисту та збереження медіа доказів військових злочинів. Спираючись на теоретико-методичні та аналітичні висновки попередніх розділів, були реалізовані конкретні рішення, для імплементації яких були застосовані творчо-комбінаторні здібності.

Моделювання взаємодії з програмним інтерфейсом. Було розроблено сценарії використання системи, представлені за допомогою діаграм BPMN та Use Case. Ці моделі детально описують послідовність дій користувача та взаємодію з системою, що дозволяє чітко розуміти процеси, які відбуваються під час роботи з програмним забезпеченням. Такий підхід сприяє виявленню потенційних оптимізацій та вдосконалень у користувацькому досвіді.

Реалізація програмного інтерфейсу. На основі розроблених моделей було створено програмний інтерфейс, який відповідає вимогам проєкту. Детальний опис ключових класів та їх взаємодії представлено у вигляді діаграм класів на різних рівнях системи. Архітектура програмного забезпечення передбачає розподіл на шари:

- *Нижній рівень (Untyped Layer):* забезпечує роботу з нетипізованими даними, уніфікуючи процес збереження різних типів інформації.

- *Середній рівень (Business Logic Layer):* містить бізнес-логіку, включаючи аутентифікацію, управління транзакціями та валідацію даних.

- *Рівень програмного інтерфейсу (API Layer)*: відповідає за обробку HTTP-запитів, надаючи зручний інтерфейс для взаємодії з системою.

Такий розподіл забезпечує гнучкість, масштабованість та полегшує подальший розвиток програмного забезпечення.

Тестування та забезпечення якості. Для перевірки функціональності та надійності розробленого програмного забезпечення було застосовано юніт-тестування з використанням фреймворку NUnit. Написані тести охоплюють основні компоненти системи, а точніше, 95% програмного коду, забезпечуючи виявлення та виправлення помилок на ранніх етапах розробки. Це підвищує стабільність та якість коду, що є критично важливим для систем, які оперують з чутливими даними.

Використання зовнішніх бібліотек та інструментів. У процесі розробки були інтегровані зовнішні бібліотеки та утиліти, що спростили реалізацію складних функцій:

- *Криптографічні бібліотеки*: для генерації ключів та верифікації цифрових підписів, щоб забезпечити безпеку та автентичність даних.

- *Бібліотеки для роботи з HTTP*: спростили обробку запитів та відповідей, полегшуючи комунікацію між клієнтом та сервером.

Залучення цих інструментів дозволило зосередитися на вирішенні основних бізнес-завдань та прискорило процес розробки.

Аналіз якості коду. За допомогою статичного аналізатора NDepend було проведено глибокий аналіз коду. Оцінено цикломатичну складність, виявлено залежності між компонентами та ідентифіковано потенційні антипатерни:

- *Цикломатична складність* показала, що в цілому код добре структурований, хоча деякі методи потребують рефакторингу.

- *Аналіз залежностей* виявив місця з високим рівнем зв’язаності, що може впливати на підтримуваність та розширюваність коду.

- *Виявлені антипатерни* надали можливість покращити код, усунувши потенційні проблеми та підвищивши його якість.

Проведений аналіз сприяв покращенню структури коду та підвищенню його відповідності кращим практикам створення програмних продуктів.

Розділ продемонстрував практичне втілення теоретичних знань та методик, розглянутих у попередніх частинах роботи. Було застосовано творчі та дослідницькі навички для розробки функціонального програмного інтерфейсу, що відповідає вимогам безпеки та ефективності.

Запропоновані рішення базуються на сучасних технологіях та інструментах, що дозволяє забезпечити високу якість та надійність системи. Використання економіко-математичних методів, моделювання та інструментів аналізу коду підвищило ефективність розробки та надало можливість прогнозувати та запобігати потенційним проблемам.

Таким чином, досягнуто поставлену мету дослідження: розроблено програмний інтерфейс на основі блокчейну для захисту та збереження медіа доказів військових злочинів. Отримані результати можуть бути використані для подальшого розвитку системи та її впровадження у реальних умовах, що сприятиме підвищенню безпеки та прозорості в зберіганні критично важливої інформації.

ВИСНОВКИ

У процесі проведення даного дослідження було досягнуто поставленої мети — розроблено ефективний та безпечний програмний інтерфейс на основі блокчейн-технології для захисту та збереження медіа доказів військових злочинів. Розроблене рішення забезпечує цілісність, автентичність та доступність даних для відповідних правових органів та міжнародних інстанцій.

Аналіз існуючих методів та технологій. Проведено детальний аналіз традиційних методів колекціонування, збереження та верифікації медіа доказів. Виявлено недоліки централізованих систем, зокрема їх вразливість до хакерських атак, маніпуляцій над даними та фізичного знищення. Оцінено можливості блокчейн-технологій як альтернативи для забезпечення високого рівня безпеки та незмінності даних.

Розробка архітектури програмного інтерфейсу. Створено концептуальну архітектуру системи, яка включає механізми завантаження, збереження, верифікації та підтвердження автентичності медіа даних. Запропоновано багаторівневу архітектуру, що складається з нижнього рівня (робота з нетипізованими даними), середнього рівня (бізнес-логіка) та рівня програмного інтерфейсу (API).

Використання алгоритмів забезпечення безпеки. Інтегровано криптографічні алгоритми хешування та цифрових підписів для забезпечення цілісності та автентичності даних. Реалізовано механізми автентифікації користувачів та верифікації транзакцій.

Реалізація та тестування прототипу. Розроблено прототип програмного інтерфейсу з використанням мови програмування C# та фреймворку ASP.NET Core. Проведено юніт-тестування за допомогою NUnit, що підтвердило коректність роботи основних компонентів системи. Використано статичний аналіз коду за допомогою NDepend для покращення якості коду програмного забезпечення.

Оцінка ефективності та перспективи розвитку. Проведено порівняння розробленого рішення з існуючими методами, що показало переваги в безпеці та надійності. Також визначено можливості масштабування системи та інтеграції з іншими технологіями.

Практичні рекомендації та доцільність впровадження. Розроблений підхід може бути використаний для реалізації інструментів що використовуються правоохоронними органами та міжнародними судами для забезпечення надійного зберігання та верифікації медіа доказів. Це сприятиме підвищенню ефективності розслідувань та забезпеченню справедливості в процесі притягнення винних до відповідальності. А також вирішить проблему прозорості медіа матеріалів розслідувань, шляхом надання доступу до авторизованих даних зі сторони третіх осіб, таких як громадських організацій, журналістів, активістів тощо.

Подальше вдосконалення системи. Рекомендується розширити функціональність системи, додавши підтримку інших типів даних та інтеграцію зі смарт-контрактами для автоматизації процесів. Також потрібно розглянути можливість впровадження механізмів машинного навчання для автоматичної класифікації та аналізу медіа доказів.

Міжнародна співпраця. Враховуючи глобальний характер військових конфліктів, здається доцільним налагодити співпрацю з міжнародними організаціями та правозахисними інституціями для спільного використання та розвитку системи.

Науковий внесок. Результати дослідження можуть стати основою для подальших наукових робіт у галузі застосування блокчейн-технологій для забезпечення безпеки даних та, особливо, медіа доказів військових злочинів.

Загальний підсумок. Дослідження підтвердило перспективність використання блокчейн-технологій для збереження та захисту медіа доказів військових злочинів. Розроблений програмний інтерфейс забезпечує високий рівень безпеки, цілісності

та доступності даних, що є критично важливим для правосуддя та міжнародного права. Запропоновані рішення та рекомендації мають практичну цінність та можуть бути впроваджені в реальних умовах, сприяючи встановленню справедливості та захисту прав людини.

Таким чином, мета дослідження досягнута, завдання виконані, а отримані результати демонструють новий підхід до збереження доказів військових злочинів завдяки розробленому прототипу, що підтверджує теорію на практиці. Це, у свою чергу, підтверджує актуальність і важливість проведеної роботи.

ПЕРЕЛІК ПОСИЛАНЬ

1) Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. *Independent*. 2008.

2) MedRec: Using Blockchain for Medical Data Access and Permission Management / A. Azaria та ін. *2016 2nd International Conference on Open and Big Data (OBD)*, м. Vienna, Austria, 22–24 серп. 2016 р. 2016. URL: https://doi.org/10.1109/obd.2016.11(дата звернення: 03.09.2024).

3) Sadeq Abdulhadi A. J., Alsaedi N. Trusted surveillance system based on blockchain-internet of spatial things for smart cities. *International Journal of Advanced Technology and Engineering Exploration*. 2023. Т. 10, № 106. URL: https://doi.org/10.19101/ijatee.2023.1010184(дата звернення: 03.09.2024).

4) Allen D. W. E. Blockchain Innovation Commons. *SSRN Electronic Journal*. 2017. URL: https://doi.org/10.2139/ssrn.2919170(дата звернення: 04.09.2024).

5) Nakamoto S. The Genesis Block: The proof of work. Benjamin Capital Management Group Inc, 2018. 503 с.

6) Xijin W., Linxiu F. The Application Research of MD5 Encryption Algorithm in DCT Digital Watermarking. *Physics procedia*. 2012. Т. 25. С. 1264–1269. URL: https://doi.org/10.1016/j.phpro.2012.03.231 (дата звернення: 11.09.2024).

7) Wang X., Yu H., Yin Y. L. Finding Collisions in the Full SHA-1. *International Association for Cryptologic Research*. URL: https://www.iacr.org/archive/crypto2005/36210017/36210017.pdf (дата звернення: 06.09.2024).

8) Martino R., Cilardo A. Designing a SHA-256 processor for blockchain-based IoT applications. *Internet of things. 2020*. Т. 11. С. 100254. URL: https://doi.org/10.1016/j.iot.2020.100254 (дата звернення: 23.09.2024).

9) Chen Z., Ye G. An asymmetric image encryption scheme based on hash SHA-3, RSA and compressive sensing. *Optik. 2022*. Т. 267. С. 169676. URL: https://doi.org/10.1016/j.ijleo.2022.169676 (дата звернення: 08.09.2024).

10) Adepoju O. J. A., Olabiyisi, S. O., Ismaila W. O. An Enhanced Blockchain-based Voting System Using Blake3 Cryptographic Hash Function. *Advances in*

Multidisciplinary & Scientific Research Journal Publications. 2024. Т. 15, № 3. С. 1–14. URL: https://doi.org/10.22624/aims/cisdi/v15n3p1(дата звернення: 09.09.2024).

11) Durga R., Sudhakar P. Implementing RSA algorithm for network security using dual prime secure protocol in crypt analysis. *International Journal of Advanced Intelligence Paradigms*. 2023. Vol. 24, no. 3/4. P. 355. URL: https://doi.org/10.1504/ijaip.2023.129183(дата звернення: 28.09.2024).

12) Stevens Z. R. J. Elliptic Curve Cryptography. *Oxford : Oxford Brookes University, 2003*.

13) Koç Ç. K., Özdemir F., Ödemiş Özger Z. ElGamal algorithm. Partially homomorphic encryption. *Cham, 2021*. С. 51–62. URL: https://doi.org/10.1007/978-3-030-87629-6_5 (дата звернення: 21.09.2024).

14) Smith B. Serving JSON. *Beginning JSON*. Berkeley, CA, 2015. С. 159–189. URL: https://doi.org/10.1007/978-1-4842-0202-9_10(дата звернення: 18.09.2024).

15) Guild T. X. Advanced XML Applications from the Experts at The XML Guild. Course Technology PTR, 2006. 384 p.

16) Shekhar S., Xiong H. Representational state transfer services. *Encyclopedia of GIS*. Boston, MA, 2008. С. 957. URL: https://doi.org/10.1007/978-0-387-35973-1_1119 (дата звернення: 13.10.2024).

17) Doolittle J. APIs With GraphQL. *IEEE software*. 2023. Т. 40, № 2. С. 118–120. URL: https://doi.org/10.1109/ms.2022.3227254 (дата звернення: 10.10.2024).

18) Sazanavets F. Overview of gRPC on ASP.NET core. *Beginning gRPC on ASP.NET core*. Berkeley, CA, 2020. URL: https://doi.org/10.1007/978-1-4842-6211-5_1 (дата звернення: 13.10.2024).

19) Sharan K. Model-View-Controller pattern. *Learn JavaFX 8*. Berkeley, CA, 2015. С. 419–434. URL: https://doi.org/10.1007/978-1-4842-1142-7_11 (дата звернення: 15.10.2024).

20) Oluwatosin H. S. Client-Server Model. *IOSR Journal of Computer Engineering*. 2014. Т. 16, № 1. С. 57–71. URL: https://doi.org/10.9790/0661-16195771(дата звернення: 05.10.2024).

21) Esparza-Peidro J., Muñoz-Escoí F. D., Bernabéu-Aubán J. M. Modeling microservice architectures. *Journal of Systems and Software*. 2024. С. 112041. URL: https://doi.org/10.1016/j.jss.2024.112041(дата звернення: 01.11.2024).

22) Noback M. The Dependency Inversion Principle. *Principles of Package Design*. Berkeley, CA, 2018. С. 67–104. URL: https://doi.org/10.1007/978-1-4842-4119-6_5(дата звернення: 18.10.2024).

23) Knebl H. Hashing. *Algorithms and Data Structures*. Cham, 2020. С. 105–128. URL: https://doi.org/10.1007/978-3-030-59758-0_3(дата звернення: 25.10.2024).

24) Stohrer C., Lugrin T. Asymmetric Encryption. *Trends in Data Protection and Encryption Technologies*. Cham, 2023. С. 11–14. URL: https://doi.org/10.1007/978-3-031-33386-6_3(дата звернення: 27.10.2024).

25) Helland P. XML and JSON are like cardboard. *Communications of the ACM*. 2017. Т. 60, № 12. С. 46–47. URL: https://doi.org/10.1145/3132269(дата звернення: 02.11.2024).

26) Margański P., Pańczyk B. REST and GraphQL comparative analysis. *Journal of Computer Sciences Institute*. 2021. Т. 19. С. 89–94. URL: https://doi.org/10.35784/jcsi.2473(дата звернення: 02.11.2024).

27) Model-View-Controller. *Flexible, Reliable Software*. 2010. С. 367–372. URL: https://doi.org/10.1201/9781439882726-37(дата звернення: 02.11.2024).

28) Christen P. Indexing. *Data Matching*. Berlin, Heidelberg, 2012. С. 69–100. URL: https://doi.org/10.1007/978-3-642-31164-2_4(дата звернення: 03.11.2024).

29) Mohamed M. A., Abdel-Fattah M. A., Khedr A. E. Challenges and Recommendations in Big Data Indexing Strategies. *International Journal of e-Collaboration*. 2021. Т. 17, № 2. С. 22–39. URL: https://doi.org/10.4018/ijec.2021040102(дата звернення: 03.11.2024).

30) Maurer W. D., Lewis T. G. Hash Table Methods. *ACM Computing Surveys*. 1975. Т. 7, № 1. С. 5–19. URL: https://doi.org/10.1145/356643.356645(дата звернення: 03.11.2024).

31) Moro M. M., Zhang D., Tsotras V. J. Hash-Based Indexing. *Encyclopedia of Database Systems*. New York, NY, 2017. С. 1–2. URL: https://doi.org/10.1007/978-1-4899-7993-3_756-2(дата звернення: 03.11.2024).

32) Akoto-Adjepong V., Asante M., Okyere-Gyamfi S. An Enhanced Non-Cryptographic Hash Function. *International Journal of Computer Applications*. 2020. Т. 176, № 15. С. 10–17. URL: https://doi.org/10.5120/ijca2020920014(дата звернення: 07.11.2024).

33) Pragmatic Unit Testing in C# with Nunit. S.l : Pragmatic Bookshelf, 2006.

34) Wilson W. G. Use of Logic Programming for Complex Business Rules. *Logic Programming*. Berlin, Heidelberg, 2005. С. 14–20. URL: https://doi.org/10.1007/11562931_4(дата звернення: 07.11.2024).

35) Smith D. macOS Security. *Apple macOS and iOS System Administration*. Berkeley, CA, 2020. С. 83–108. URL: https://doi.org/10.1007/978-1-4842-5820-0_3(дата звернення: 09.11.2024).

36) Mythical Unit Test Coverage / V. Antinyan та ін. *IEEE Software*. 2018. Т. 35, № 3. С. 73–79. URL: https://doi.org/10.1109/ms.2017.3281318(дата звернення: 12.11.2024).

37) Improve .NET code quality with NDepend. *NDepend*. URL: https://www.ndepend.com (дата звернення: 13.11.2024).

38) Mukherjee S. Code Quality Metrics. *Source Code Analytics With Roslyn and JavaScript Data Visualization*. Berkeley, CA, 2016. С. 15–44. URL: https://doi.org/10.1007/978-1-4842-1925-6_2 (дата звернення: 13.11.2024).

39) Cyclomatic Complexity / C. Ebert та ін. *IEEE Software*. 2016. Т. 33, № 6. С. 27–29. URL: https://doi.org/10.1109/ms.2016.147 (дата звернення: 14.11.2024).

Програмний інтерфейс на основі блокчейну для захисту та збереження медіа доказів військових злочинів

Виконав: Кожін Назар Олексійович

Керівник: Сагайдак Віктор Анатолійович

Актуальність теми

- Проблема збереження медіа доказів військових злочинів.
- Вразливість традиційних систем до атак та маніпуляцій.
- Потенціал блокчейну для підвищення безпеки та незмінності записів.

Мета, об’єкт та предмет дослідження

Мета: Розробити безпечний і ефективний програмний інтерфейс на базі блокчейну для збереження медіа доказів.

Об’єкт: Процеси збирання, збереження та верифікації медіа доказів військових злочинів.

Предмет: Методи та засоби розробки програмного інтерфейсу з використанням блокчейн-технологій.

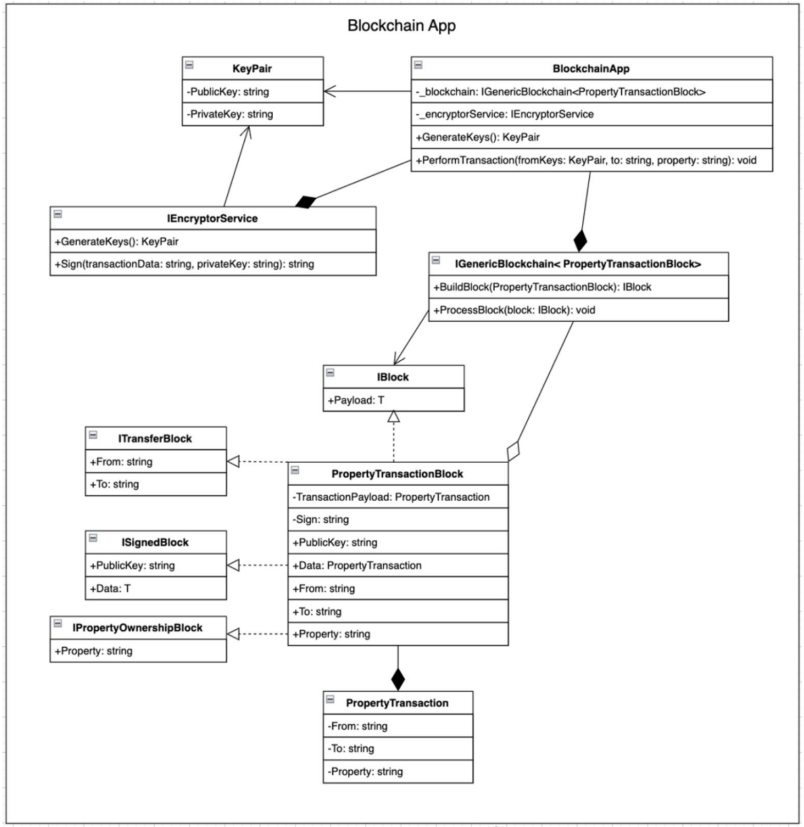
Методи та інструменти дослідження

- Моделювання (class diagram, bpmn, use-case)
- Прототипування (C#, ASP.NET Core Web API).
- Контейнеризація (Docker).
- Контроль версій (Git).
- Криптографія (хеші, цифрові підписи).
- Статичний аналіз коду (NDepend).

Архітектура рішення

- **Untyped Layer**: робота з нетипізованими даними.
- **Business Logic Layer**: валідація, управління транзакціями, правила додавання блоків.
- **API Layer**: обробка HTTP-запитів, інтерфейс взаємодії.

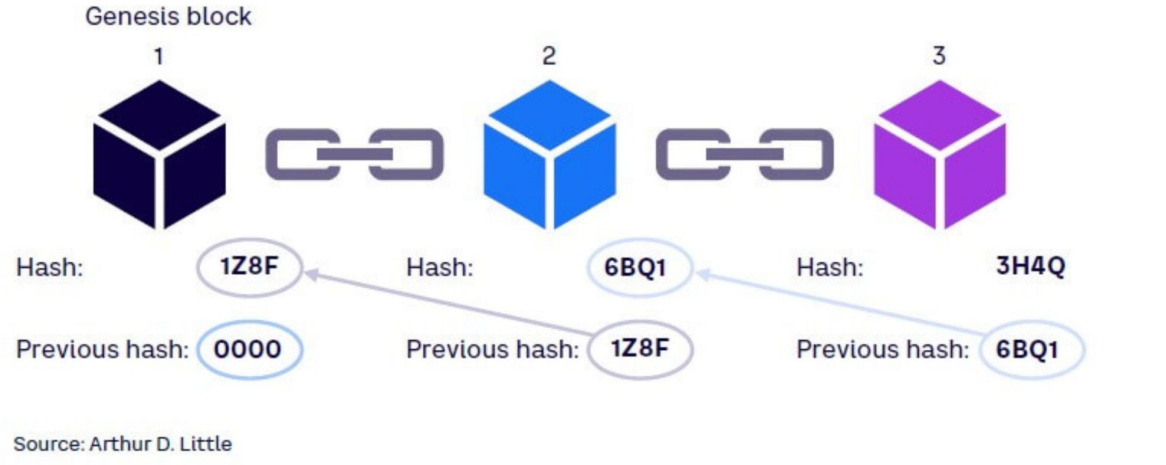
Архітектура рішення (діаграма класів)



Блокчейн у проєкті

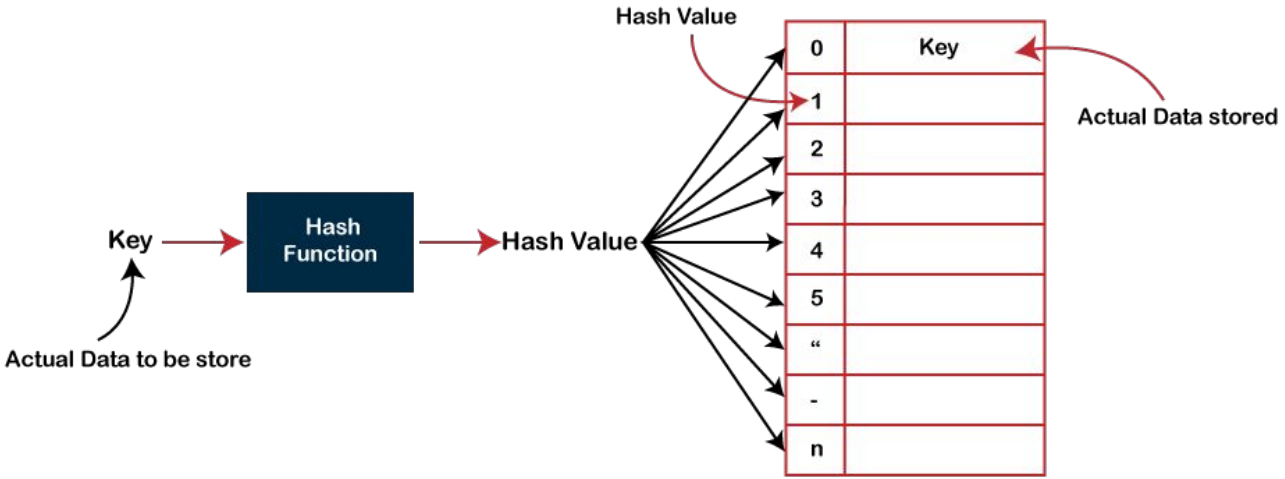
- Роль блокчейну: незмінність, високий рівень безпеки, прозорість.
- Механізми збереження медіа даних: хешування, верифікація, реєстр транзакцій.

Використання хеш-таблиць для індексації блоків.



Реалізація індексації та пошуку

- Використання хеш-таблиць для швидкого доступу до блоків.
- Зв’язок ключа з блоком: запис у таблиці, пошук за ключем (чи композитним ключем).
- Переваги: швидкість, гнучкість і простота розширення.



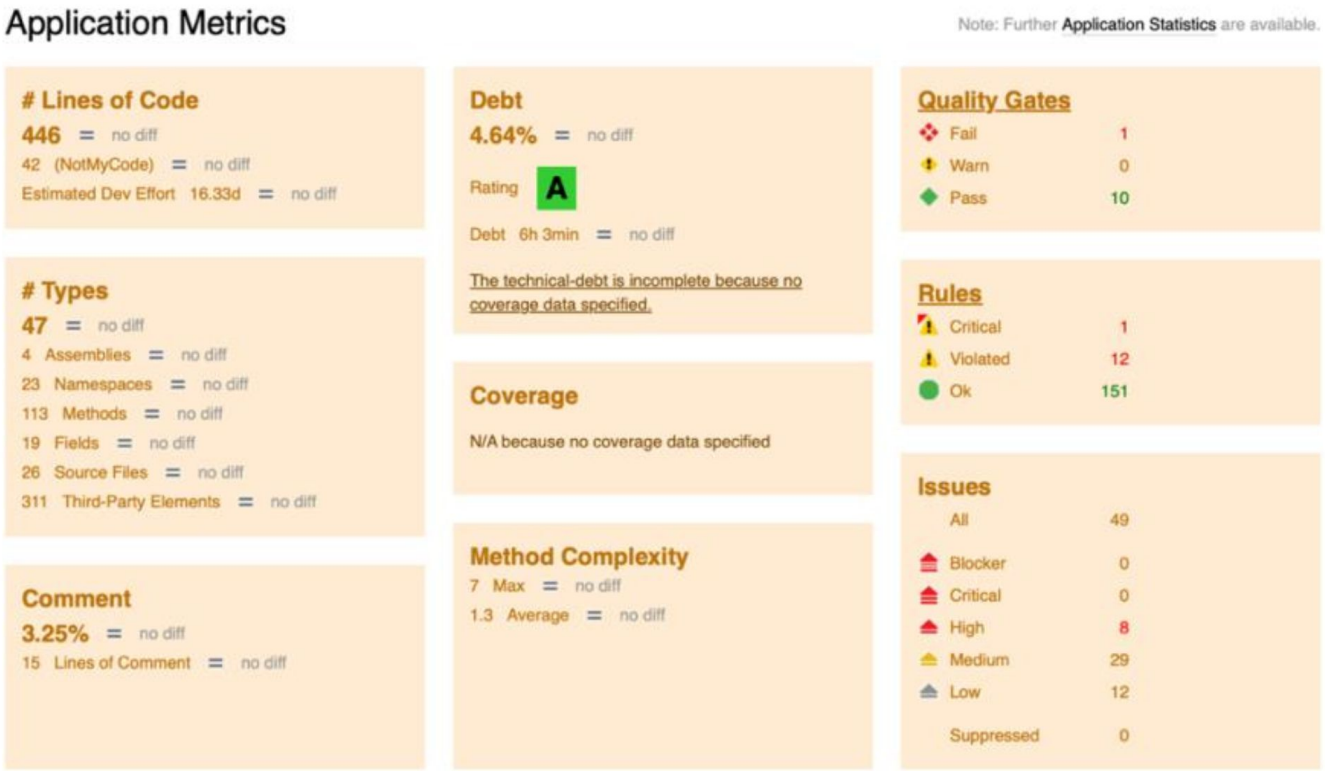
Тестування програмного інтерфейсу

- Юніт-тести (NUnit): протестовано основний функціонал (додавання блоку, верифікація транзакцій, модулі хешу та цифрового підпису).
- Статичний аналіз (NDepend): показники цикломатичної складності, виявлення антипатернів, Quality Gates.

Основні висновки: загалом код структурований, деякі методи потребують рефакторингу.

Тестування програмного інтерфейсу (візуалізація)

Метрики коду (NDepend)



Практичне застосування

- Потенціал впровадження у правоохоронних органах, міжнародних судових організаціях.
- Підвищення прозорості розслідувань, незмінність доказів.
- Можливість інтеграції з іншими системами зберігання даних або сервісами.

Перспективи розвитку

- Розширення підтримки різних типів контенту (відео, аудіо).
- Інтеграція зі смарт-контрактами для автоматизованих правових процедур.
- Подальше підвищення продуктивності й масштабованості.
- Використання методів машинного навчання для аналізу/категоризації медіа доказів.

Висновки

- Створено архітектуру програмного інтерфейсу на основі блокчейну, що дозволяє надійно зберігати та верифікувати медіа докази військових злочинів.
- Використано криптографічні алгоритми, які мінімізують ризики підроблення чи знищення даних.
- Запропонований підхід допомагає подолати недоліки традиційних систем, шляхом забезпечення доступності та незмінності медіа-об’єктів.
- Нижній рівень додатку та абстракції дозволяють адаптуватися до інших сфер зберігання важливих даних (медична документація, державні реєстри тощо).
- Інтеграція зі смарт-контрактами, розширення на інші типи медіаданих, оптимізація продуктивності та масштабування в розподілених середовищах.

Апробація

II Всеукраїнська науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу»

Представлено основні результати розробки архітектури та тестування прототипу. А також висвітлено особливості використання блокчейн-технологій для зберігання медіа доказів військових злочинів.

«Проблеми комп’ютерної інженерії»

На конференції представлено доповідь про розроблений алгоритм верифікації медіа доказів, що використаний у даній роботі.

Дякую за увагу!