

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Вебзастосунок для адміністрування децентралізованих систем
ідентифікації DID»**

Виконав студент групи ЗІПЗ-22-1:
ЯРОШ Володимир Володимирович.

Керівник роботи:
СУГОНЯК Інна Іванівна

Рецензент:
ТАЛАВЕР Олег Володимирович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення Тетяна ВАКАЛЮК

«28» лютого 2026р.

ЗАВДАННЯ

на кваліфікаційну роботу

Здобувач вищої освіти: **ЯРОШ Володимир Володимирович**

Керівник роботи: **СУГОНЯК Інна Іванівна**

Тема роботи: **«Вебзастосунок для адміністрування децентралізованих систем ідентифікації DID»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026р., №93/с**

Вихідні дані для роботи: технології децентралізованої ідентифікації DID, блокчейн Ethereum, смарт-контракти Solidity, стандарти W3C DID та Verifiable Credentials, засоби криптографічної перевірки цифрових документів, вебтехнології React та Node.js для створення вебзастосунку адміністрування цифрових документів.

Консультанти з кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	СУГОНЯК І.І	10.03.2026	11.03.2026
2	СУГОНЯК І.І	24.04.2026	26.05.2026
3	СУГОНЯК І.І	15.05.2026	15.05.2026

РЕФЕРАТ

Кваліфікаційна робота складається з вебзастосунку для адміністрування децентралізованих систем ідентифікації та пояснювальної записки. Пояснювальна записка містить 87 сторінок, 39 рисунків, 20 таблиць, 13 додатків та 30 використаних джерел.

Метою роботи є розробка вебзастосунку для створення, видачі, отримання, перевірки та анулювання цифрових документів із використанням технологій децентралізованої ідентифікації та блокчейн-технологій.

У роботі виконано аналіз сучасних підходів до цифрової ідентифікації, досліджено принципи функціонування децентралізованих ідентифікаторів, верифікованих цифрових облікових даних та верифікованих цифрових представлень, проаналізовано існуючі програмні рішення у сфері цифрової ідентифікації та розроблено програмний засіб для адміністрування цифрових документів.

Розроблений вебзастосунок забезпечує створення цифрових документів різних типів, їх видачу користувачам, підтвердження права володіння документом за допомогою криптографічного підпису, перевірку справжності документів за QR-кодом та анулювання документів із фіксацією відповідного статусу.

Під час розробки використано мову програмування JavaScript, бібліотеку React, платформу Node.js, фреймворк Express, систему керування базами даних SQLite, блокчейн Ethereum та стандарти децентралізованої цифрової ідентифікації.

КЛЮЧОВІ СЛОВА: ДЕЦЕНТРАЛІЗОВАНИЙ ІДЕНТИФІКАТОР, ВЕРИФІКОВАНИЙ ЦИФРОВИЙ ДОКУМЕНТ, ВЕРИФІКОВАНЕ ЦИФРОВЕ ПРЕДСТАВЛЕННЯ, БЛОКЧЕЙН, ЦИФРОВА ІДЕНТИФІКАЦІЯ, ЦИФРОВИЙ ДОКУМЕНТ, ВЕБЗАСТОСУНОК.

					ІПЗ.КР.Б-121-26-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.	Ярош В.В.				Вебзастосунок для адміністрування децентралізованих систем ідентифікації DID Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник	Сугоняк І.І.						3	87
Рецензент	Талавер О.В.					Житомирська політехніка, ЗІПЗ-22-1		
н. контр.								
Зав. каф.	Вакалюк Т.А.							

ABSTRACT

The qualification work consists of a web application for the administration of decentralized identity systems and an explanatory note. The explanatory note contains 87 pages, 39 figures, 20 tables, 13 appendices and 30 references.

The purpose of the work is to develop a web application for creating, issuing, claiming, verifying and revoking digital documents using decentralized identity technologies and blockchain technologies.

The work analyzes modern approaches to digital identity, examines the principles of decentralized identifiers, verifiable digital credentials and verifiable digital presentations, analyzes existing software solutions in the field of digital identity, and develops a software tool for the administration of digital documents.

The developed web application provides the creation of different types of digital documents, their issuance to users, confirmation of document ownership by means of a cryptographic signature, verification of document authenticity by QR code, and revocation of documents with the corresponding status recorded.

The development used the JavaScript programming language, the React library, the Node.js platform, the Express framework, the SQLite database management system, the Ethereum blockchain and decentralized digital identity standards.

KEYWORDS: DECENTRALIZED IDENTIFIER, VERIFIABLE DIGITAL DOCUMENT, VERIFIABLE DIGITAL PRESENTATION, BLOCKCHAIN, DIGITAL IDENTITY.

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ЗМІСТ	5
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ІДЕНТИФІКАЦІЇ.....	11
1.1 Постановка задачі.....	11
1.2 Аналіз аналогів програмного продукту.....	14
1.3 Вибір архітектури вебзастосунку для адміністрування децентралізованих систем ідентифікації.....	18
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	25
Висновки до першого розділу	27
РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ДЛЯ АДМІНІСТРУВАННЯ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ІДЕНТИФІКАЦІЇ.....	29
2.1 Вимоги до системи	29
2.2 Проєктування бази даних	41
2.3 Проєктування та реалізація алгоритмів роботи системи.....	47
2.4 Реалізація функціоналу вебзастосунку	55
Висновки до другого розділу	63
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБЗАСТОСУНКОМ ДЛЯ АДМІНІСТРУВАННЯ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ІДЕНТИФІКАЦІЇ.....	64

3.1 Розгортання та налаштування вебзастосунку	64
3.2 Користувацький інтерфейс системи	67
3.3 Інтерфейс адміністратора системи	71
3.4 Особливості видачі, отримання та перевірки документів	77
3.5 Тестування програмного комплексу	81
Висновки до третього розділу.	84
ВИСНОВКИ	85
ЛІТЕРАТУРА	88
ДОДАТКИ	91

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API — Application Programming Interface, інтерфейс програмування застосунків.

CRUD — Create, Read, Update, Delete, базові операції роботи з даними.

DID — Decentralized Identifier, децентралізований ідентифікатор.

Express — програмний фреймворк для Node.js.

Ethereum — блокчейн-платформа для роботи зі смарт-контрактами.

JSON — JavaScript Object Notation, формат обміну даними.

MetaMask — криптогаманець для взаємодії з блокчейн-мережею Ethereum.

Node.js — програмна платформа для виконання JavaScript на сервері.

PDF — Portable Document Format, формат електронних документів.

ПЗ — програмне забезпечення.

QR — Quick Response, двовимірний матричний код.

React — JavaScript-бібліотека для створення користувацьких інтерфейсів.

REST API — програмний інтерфейс взаємодії між клієнтською та серверною частинами системи.

Sepolia — тестова мережа блокчейну Ethereum.

SQLite — система керування базами даних.

БД — база даних.

UI — User Interface, користувацький інтерфейс.

URL — Uniform Resource Locator, адреса ресурсу в мережі Інтернет.

UX — User Experience, взаємодія користувача із системою.

VC — Verifiable Credential, верифікований цифровий документ.

VP — Verifiable Presentation, верифіковане цифрове представлення.

W3C — World Wide Web Consortium, міжнародний консорціум, який розробляє та публікує рекомендації для розвитку вебтехнологій, зокрема DID Core та Verifiable Credentials Data Model.

Web3 — концепція децентралізованих вебтехнологій.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У сучасних умовах цифрової трансформації суспільства інформаційні технології відіграють важливу роль у забезпеченні ефективної взаємодії між користувачами, організаціями та державними установами. Одним із ключових напрямів розвитку цифрових сервісів є впровадження електронних документів та систем цифрової ідентифікації, які дозволяють спростити процеси обміну інформацією, підтвердження особи та перевірки достовірності документів [1, 2].

Традиційні системи ідентифікації переважно базуються на централізованих підходах до зберігання та обробки даних. У таких системах інформація про користувачів та документи знаходиться під контролем окремих організацій або постачальників послуг. Незважаючи на широке поширення подібних рішень, вони мають низку недоліків, серед яких залежність від центрального вузла керування, ризики компрометації даних, можливість несанкціонованого доступу до інформації та складність незалежної перевірки справжності цифрових документів.

Розвиток технологій розподілених реєстрів та блокчейн-платформ сприяв появі нових підходів до побудови систем цифрової ідентифікації. Одним із найбільш перспективних напрямів є використання децентралізованих ідентифікаторів (Decentralized Identifier, DID), які дозволяють користувачам самостійно керувати власною цифровою ідентичністю без необхідності використання централізованих реєстрів. Концепція децентралізованої ідентифікації є складовою підходу Self-Sovereign Identity, відповідно до якого власник цифрової ідентичності отримує повний контроль над своїми персональними даними [2, 4, 7].

Важливим елементом сучасних децентралізованих систем є використання верифікованих цифрових документів та механізмів їх криптографічного підтвердження. Завдяки застосуванню рекомендацій World Wide Web Consortium, зокрема DID Core та Verifiable Credentials Data Model,

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

забезпечується можливість створення цифрових документів, достовірність яких може бути перевірена незалежно від організації, що їх видала [2, 3].

Актуальність теми кваліфікаційної роботи зумовлена необхідністю створення сучасних програмних засобів для адміністрування цифрових документів із використанням технологій децентралізованої ідентифікації. Впровадження таких систем дозволяє підвищити рівень безпеки, прозорості та надійності процесів видачі, отримання та перевірки документів, а також забезпечити можливість незалежного підтвердження їх справжності.

Метою кваліфікаційної роботи є розробка вебзастосунку для адміністрування децентралізованих систем ідентифікації DID, який забезпечує створення, видачу, отримання, перевірку та анулювання цифрових документів із використанням технологій децентралізованої ідентифікації та блокчейн-технологій.

Для досягнення поставленої мети необхідно вирішити такі завдання: виконати аналіз сучасних підходів до цифрової ідентифікації та децентралізованих систем управління цифровими документами, дослідити принципи функціонування децентралізованих ідентифікаторів та цифрових облікових даних, проаналізувати існуючі програмні рішення у сфері децентралізованої ідентифікації, спроектувати архітектуру вебзастосунку, розробити структуру бази даних для зберігання цифрових документів, реалізувати функціонал створення, видачі, отримання, перевірки та анулювання цифрових документів, забезпечити інтеграцію із блокчейн-мережею для підтвердження достовірності документів, провести тестування розробленого програмного забезпечення.

Об'єктом дослідження є процес адміністрування цифрових документів у децентралізованих системах ідентифікації.

Предметом дослідження є методи, моделі та програмні засоби реалізації вебзастосунків для роботи з цифровими документами на основі технологій децентралізованої ідентифікації.

Під час виконання роботи використано методи системного аналізу, об'єктно-орієнтованого проєктування, моделювання інформаційних систем, вебпрограмування та технології розподілених реєстрів.

Практичне значення отриманих результатів полягає у створенні програмного засобу, який може використовуватися для адміністрування цифрових документів різного призначення, зокрема сертифікатів, перепусток, посвідчень та інших електронних документів, що потребують підтвердження справжності та захисту від підробки.

Структура кваліфікаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі виконано аналіз предметної області та сучасних підходів до реалізації децентралізованих систем ідентифікації. У другому розділі розглянуто питання проєктування та програмної реалізації вебзастосунку. У третьому розділі наведено опис інтерфейсу користувача, порядку роботи із системою та результатів тестування програмного забезпечення.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ІДЕНТИФІКАЦІЇ

1.1 Постановка задачі

У сучасному цифровому середовищі питання ідентифікації користувачів, підтвердження достовірності інформації та перевірки електронних документів набувають все більшої актуальності. Стрімкий розвиток інформаційних технологій сприяв переходу значної кількості процесів із паперової форми у цифрову. Сьогодні електронні документи активно використовуються у сфері освіти, державного управління, охорони здоров'я, фінансових послуг та корпоративного сектору. Разом із розвитком електронного документообігу виникає необхідність забезпечення достовірності, цілісності та безпечності цифрових документів.

Більшість сучасних інформаційних систем побудовані за централізованим принципом. У таких системах зберігання інформації та управління доступом здійснюється через центральний сервер або організацію, яка виступає єдиним джерелом довіри. Централізовані підходи мають ряд переваг, серед яких простота адміністрування та централізований контроль над інформацією. Однак одночасно вони характеризуються рядом недоліків, що особливо проявляються при роботі з цифровими документами та системами ідентифікації [8, 9].

Однією з основних проблем централізованих систем є залежність від центрального вузла управління. У випадку відмови сервера, втрати доступу до бази даних або компрометації інформаційної системи можуть виникати серйозні ризики для безпеки даних. Крім того, користувачі змушені повністю довіряти організації, яка зберігає їхні персональні дані та здійснює перевірку документів. У разі несанкціонованого доступу або внутрішніх порушень можливе внесення змін до інформації без відома власників документів.

Додатковою проблемою є поширення підроблених електронних документів. Незважаючи на широке використання електронного

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

документообігу, перевірка справжності багатьох документів досі здійснюється шляхом звернення до організації-видавця або через внутрішні бази даних.

Такий підхід ускладнює процес перевірки та створює залежність від доступності відповідних інформаційних ресурсів. У багатьох випадках третя сторона не має можливості самостійно перевірити справжність документа без залучення організації, яка його видала.

Вирішення зазначених проблем стало можливим завдяки розвитку технологій розподілених реєстрів та децентралізованої цифрової ідентифікації. Одним із найбільш перспективних напрямів розвитку сучасних систем цифрової ідентифікації є концепція Self-Sovereign Identity, відповідно до якої користувач самостійно контролює власні цифрові дані та визначає порядок їх використання. Основою такого підходу є використання децентралізованих ідентифікаторів, які дозволяють створювати незалежні цифрові ідентичності без необхідності використання централізованих реєстрів [3, 4].

Децентралізований ідентифікатор являє собою унікальний ідентифікатор, що належить конкретному суб'єкту та може бути перевірений незалежно будь-яким учасником системи. На відміну від традиційних ідентифікаторів, децентралізовані ідентифікатори не прив'язані до конкретної організації або постачальника послуг. Це забезпечує більшу незалежність користувачів та підвищує рівень захисту персональних даних.

Важливим елементом сучасних децентралізованих систем є використання верифікованих цифрових документів. Такі документи містять набір атрибутів, що підтверджують певні факти про власника документа, а також криптографічний доказ їх достовірності. Завдяки використанню цифрових підписів будь-яка зацікавлена сторона може переконатися у справжності документа та цілісності його даних без необхідності звернення до організації-видавця [5, 6].

Для забезпечення додаткового рівня довіри можуть використовуватися блокчейн-технології. Розміщення контрольних даних документа у блокчейні

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволяє фіксувати факт його видачі та унеможливорює непомітне внесення змін до інформації після створення документа. Використання блокчейн-мереж також забезпечує прозорість процесів перевірки та дозволяє незалежно підтверджувати достовірність документів [7, 10].

Застосування технологій децентралізованої ідентифікації відкриває можливості для створення універсальних систем цифрових документів, які можуть використовуватися у різних сферах діяльності. До таких документів належать сертифікати, посвідчення, перепустки, довідки, дипломи та інші електронні документи, що потребують перевірки справжності та захисту від підробки.

У рамках даної кваліфікаційної роботи розглядається задача створення вебзастосунку для адміністрування децентралізованих систем ідентифікації, який забезпечуватиме створення, видачу, отримання, перевірку та анулювання цифрових документів. Особливістю системи є використання технологій децентралізованої цифрової ідентифікації, криптографічних механізмів підтвердження достовірності та блокчейн-технологій для забезпечення цілісності інформації.

Для досягнення поставленої мети необхідно забезпечити реалізацію таких функціональних можливостей: створення цифрових документів різних типів, ведення обліку виданих документів, формування та зберігання цифрових облікових даних, видачу документів користувачам, підтвердження права володіння документом за допомогою криптографічного підпису, перевірку справжності документів через вебінтерфейс, перевірку документів за допомогою QR-кодів, анулювання документів із фіксацією відповідного статусу, інтеграцію з блокчейн-мережею для підтвердження достовірності документів, забезпечення адміністрування та контролю цифрових документів.

Результатом виконання кваліфікаційної роботи має стати вебзастосунок, який дозволить реалізувати повний життєвий цикл цифрового документа від моменту його створення до перевірки справжності та анулювання, забезпечуючи при цьому високий рівень безпеки, прозорості та надійності.

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2 Аналіз аналогів програмного продукту

Перед розробкою власного вебзастосунку було проведено аналіз існуючих систем, які використовуються для цифрової ідентифікації користувачів та роботи з цифровими документами. Аналіз аналогів дозволяє зрозуміти, які функції вже реалізовані в сучасних системах, а також визначити переваги та недоліки існуючих рішень.

Сьогодні існує багато платформ, які використовують технології децентралізованої ідентифікації, цифрових документів та блокчейн-технології. Основною задачею таких систем є створення, видача та перевірка цифрових документів без використання паперових носіїв.

Одним із найбільш відомих рішень у сфері цифрової ідентифікації є Microsoft Entra Verified ID. Дана система розроблена компанією Microsoft та використовується для створення і перевірки цифрових документів та цифрових облікових даних користувачів [11].

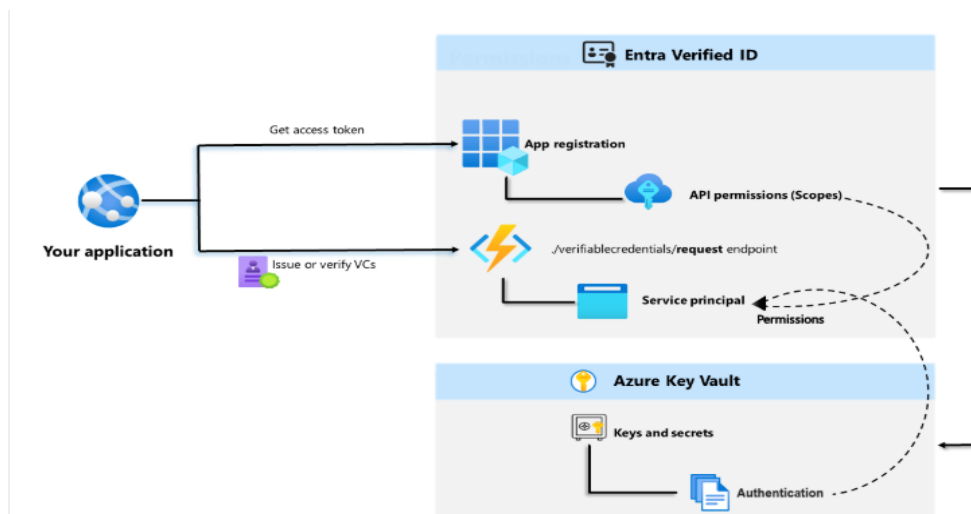


Рис.1.1 Інтерфейс створення верифікованого документа в Microsoft Entra Verified ID

Платформа дозволяє організаціям видавати користувачам цифрові документи та виконувати їх перевірку за допомогою криптографічних механізмів. Система підтримує стандарти DID та VC, що дозволяє використовувати децентралізовану цифрову ідентифікацію.

Перевагою Microsoft Entra Verified ID є високий рівень безпеки та інтеграція з іншими сервісами Microsoft.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

Недоліком системи є залежність від інфраструктури Microsoft та орієнтація переважно на великі організації. Для невеликих проєктів або навчальних систем використання такої платформи може бути занадто складним.

Ще одним прикладом системи цифрової ідентифікації є Dock. Дана платформа використовується для роботи з децентралізованими ідентифікаторами та цифровими документами. Dock дозволяє створювати цифрові документи, видавати їх користувачам та перевіряти достовірність інформації. Для забезпечення цілісності даних у системі використовуються блокчейн-технології [12].

До переваг Dock можна віднести підтримку стандартів DID та VC, можливість інтеграції з іншими сервісами та підтримку блокчейн-технологій. Недоліком є складність налаштування системи та необхідність додаткових знань для роботи з блокчейн-мережею.

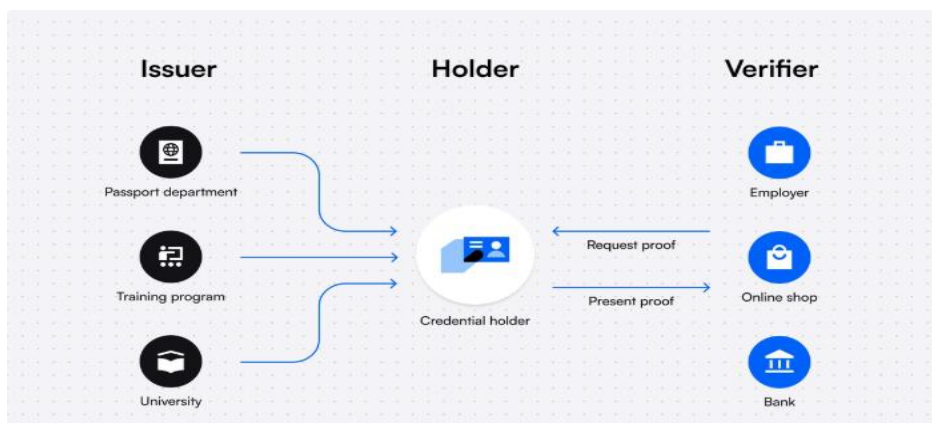


Рис. 1.2 Інтерфейс роботи з верифікованими документами в Truvera

Також було розглянуто European Blockchain Services Infrastructure. Даний проєкт створюється в межах Європейського Союзу та використовується для розвитку цифрової ідентифікації й обміну цифровими документами між різними організаціями [13].

Система використовує блокчейн-технології, децентралізовані ідентифікатори та цифрові документи для перевірки достовірності інформації. Основною перевагою EBSI є використання міжнародних стандартів та високий рівень надійності.



Рис. 1.3 Схема роботи верифікованих цифрових документів в EBSI

Недоліком системи є складна структура та значні вимоги до інфраструктури. Крім того, система орієнтована переважно на державні та великі корпоративні проєкти.

Виходячи з наявних переваг та недоліків, складемо порівняльну таблицю вищезгаданих програмних аналогів. (табл. 1.1).

Таблиця 1.1

Таблиця порівняння

Критерій	Microsoft Entra Verified ID	Truvera / Dock	EBSI	Розроблювана система
Основне призначення	Корпоративна цифрова ідентифікація та керування	Видача, зберігання та перевірка verifiable credentials	Європейська інфраструктура довіри для цифрових ідентифікаторів	Адміністрування цифрових документів у межах одного вебзастосунку
Основні ролі	Видавець, власник, перевіряюча сторона	Видавець, власник, перевіряюча сторона	Видавець, власник, перевіряюча сторона, довірені реєстри	Адміністратор, користувач, перевіряюча сторона
Підтримка DID / VC	Так	Так	Так	Так

Продовження таблиці 1.1

Використання блокчейну	Не є основним видимим елементом для користувача	Використовується у межах платформи	Є частиною інфраструктури довіри	Використовується мережа Ethereum Sepolia для збереження контрольного хешу документа
Типи документів	Корпоративні облікові дані та посвідчення	Різні типи verifiable credentials відповідно до заданих схем	Освітні, державні, професійні та інші цифрові документи	Сертифікати, перепустки, посвідчення та інші цифрові документи
Анулювання документів	Підтримується	Підтримується	Підтримується через механізми статусів	Реалізовано через зміну статусу документа
Складність впровадження	Висока, оскільки потребує використання інфраструктури Microsoft	Середня або висока, оскільки потрібне налаштування схем, API та процесів видачі	Висока, оскільки система є частиною масштабної європейської інфраструктури	Низька, оскільки основні функції зібрані в одному вебзастосунку

Проведений аналіз показав, що сучасні аналоги мають розвинений функціонал для роботи з цифровими документами, однак більшість із них орієнтована на корпоративні або державні сценарії використання. Microsoft Entra Verified ID потребує інфраструктури Microsoft, Truvera передбачає роботу зі схемами, шаблонами, API та механізмами відкликання, EBSI є масштабною європейською інфраструктурою довіри [11, 12, 13].

Розроблювана система не є прямим конкурентом зазначених платформ за масштабом, однак має інше призначення. Вона демонструє повний життєвий цикл цифрового документа в межах одного вебзастосунку: створення документа адміністратором, формування QR-коду, підтвердження володіння через MetaMask, збереження контрольного хешу в блокчейні Ethereum, перевірку документа та його анулювання. Саме тому розробка

власного вебзастосунок є доцільною для практичного дослідження технологій DID, VC, криптографічного підпису та блокчейн-перевірки.

1.3 Вибір архітектури вебзастосунок для адміністрування децентралізованих систем ідентифікації

Під час розробки вебзастосунок для адміністрування децентралізованих систем ідентифікації необхідно було визначити архітектуру, яка забезпечить не лише зручне створення цифрових документів, а й можливість перевірки їхньої достовірності після видачі. Для даної теми важливою є не сама наявність електронного запису або PDF-документа, а можливість довести, що документ справді був створений уповноваженим видавцем, не був змінений після видачі та може бути перевірений третьою стороною.

За результатами аналізу аналогів було визначено, що складність сучасних систем цифрової ідентифікації полягає у поєднанні двох протилежних вимог. З одного боку, організації потрібен зручний інструмент для адміністрування документів: створення записів, перегляд списку, формування посилань, генерація QR-кодів, облік статусів, анулювання та перевірка. З іншого боку, цифровий документ не повинен повністю залежати лише від внутрішньої бази даних одного сервісу, оскільки у випадку недоступності сервера перевіряюча сторона втрачає можливість підтвердити його справжність.

У звичайних централізованих системах перевірка документа здебільшого виконується через сервер організації-видавця. Такий підхід є зручним для адміністрування, але має суттєвий недолік: довіра до документа фактично прив'язується до доступності одного сервісу. Якщо сервер не працює, база даних пошкоджена або доступ до сервісу обмежений, перевіряюча сторона не може самостійно встановити, чи був документ виданий, чи не змінювалися його дані та чи не було його анульовано. Саме цю проблему частково вирішують децентралізовані підходи, у яких частина доказів достовірності виноситься за межі локальної бази даних.

Розглянуті аналоги також показують, що готові рішення у сфері DID та VC часто є складними для впровадження. Корпоративні платформи потребують налаштування хмарної інфраструктури, доменів, політик доступу та інтеграції з іншими сервісами. Спеціалізовані VC-платформи вимагають роботи зі схемами документів, шаблонами перевірки, API, цифровими гаманцями та механізмами відкликання. Інфраструктурні рішення на зразок EBSI орієнтовані на великі міжорганізаційні або державні сценарії. Тому в межах даної кваліфікаційної роботи поставлено задачу створити простіший вебзастосунок, який не конкурує з такими платформами за масштабом, але демонструє повний життєвий цикл цифрового документа та використовує базові механізми децентралізованої перевірки.

Повністю децентралізована архітектура для даної системи не є доцільною. Якщо всі операції з документами перенести лише у блокчейн або в повністю децентралізоване середовище, це ускладнить роботу адміністратора, збільшить кількість технічних операцій, підвищить вартість взаємодії з мережею та створить ризики щодо збереження персональних даних. Цифрові документи можуть містити персональну інформацію, тому зберігати повний зміст таких документів у публічному блокчейні недоцільно. Через це блокчейн у розроблюваній системі використовується не як основне сховище документів, а як засіб фіксації контрольної інформації.

Для адміністративної частини системи було обрано клієнт-серверну архітектуру. Вона дозволяє реалізувати звичний вебінтерфейс, централізовано обробляти запити, зберігати службові дані у базі даних і забезпечити просту роботу адміністратора. Клієнтська частина відповідає за відображення інтерфейсу та взаємодію користувача із системою. Серверна частина обробляє запити, формує цифрові документи, створює QR-коди, генерує контрольний хеш, працює з базою даних і взаємодіє з блокчейн-мережею Ethereum. База даних використовується для збереження службових відомостей про документи, їхні статуси, власників, адреси гаманців, хеші та інформацію про анулювання [14].

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

Водночас обрана архітектура не є звичайною централізованою базою документів. Її особливість полягає в тому, що сервер забезпечує адміністрування, але механізм довіри до документа підсилюється за рахунок криптографічних і децентралізованих елементів. До таких елементів належать DID видавця, структура Verifiable Credential, підпис власника через MetaMask і контрольний хеш документа, який може бути зіставлений із записом у блокчейн-мережі Ethereum. Це дозволяє відокремити службову частину системи від доказів достовірності документа.

Децентралізована мережа добре підходить для задачі перевірки цифрових документів, оскільки вона не залежить від одного сервера застосунку. Блокчейн складається з багатьох вузлів, які підтримують спільний стан реєстру та підтверджують транзакції відповідно до правил мережі. Після фіксації контрольної інформації у блокчейні її складно непомітно змінити, оскільки зміна записів порушила б цілісність ланцюга блоків. Для цифрових документів це корисно тим, що у блокчейні можна зберігати не сам документ, а його хеш або інші контрольні дані. Якщо документ змінити після видачі, новий хеш відрізнятиметься від зафіксованого, і це дозволить виявити підміну [7, 10].

У розроблюваній системі Ethereum використовується як незалежний розподілений реєстр для фіксації контрольного хешу документа. Повний зміст документа та персональні дані користувача не записуються у блокчейн. Такий підхід зменшує ризики розкриття персональної інформації та водночас дозволяє підтвердити незмінність документа. Блокчейн у цьому випадку виконує роль зовнішнього джерела підтвердження, з яким можна зіставити контрольні дані документа.

DID використовується для ідентифікації видавця документа. Децентралізований ідентифікатор пов'язує видавця з криптографічними ключами та дозволяє перевірити, чи міг саме цей видавець сформувати відповідний цифровий документ. У межах системи DID не замінює всі механізми авторизації, але використовується як частина моделі довіри до

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

документа. Завдяки цьому перевірка документа не обмежується лише записом у базі даних, а може включати перевірку походження документа від конкретного видавця [3, 4].

У сучасних системах децентралізованої ідентифікації використовуються різні DID-методи, які відрізняються способом формування ідентифікатора, підтримкою blockchain та інтеграцією з Web3-інфраструктурою. Для вибору DID-методу в межах розроблюваної системи було проведено порівняння основних підходів.

Таблиця 1.2

Порівняння DID-методів

Критерій	did:key	did:web	did:ethr
Спосіб формування	Локальний ключ	Вебдомен	Ethereum-адреса
Використання blockchain	Ні	Ні	Так
Підтримка MetaMask	Ні	Ні	Так
Підтримка Web3 DID Document	Часткова	Ні	Так
Незалежна перевірка	Так	Так	Так
Інтеграція з Ethereum	Ні	Ні	Так
Доцільність для Web3-систем	Низька	Середня	Висока

Verifiable Credential використовується як структурована форма цифрового документа. На відміну від звичайного електронного запису, VC містить не тільки дані документа, а й службову інформацію, необхідну для перевірки: ідентифікатор, видавця, дату видачі, дані власника, тип документа та криптографічний доказ. У такій моделі є три основні ролі: видавець створює і підписує документ, власник отримує його та може пред'явити для перевірки, а перевіряюча сторона аналізує достовірність отриманих даних [5, 6].

MetaMask використовується для підтвердження зв'язку між документом і власником криптогаманця. Користувач підписує повідомлення власним гаманцем, після чого система зберігає адресу гаманця та результат

підтвердження. При цьому приватний ключ користувача не передається на сервер і не зберігається в базі даних. Це важливо, оскільки право володіння підтверджується криптографічною дією з боку користувача, а не простим натисканням кнопки у вебінтерфейсі [15].

Основний сценарій роботи системи побудований так. Спочатку адміністратор створює цифровий документ через вебінтерфейс. Сервер формує структуру документа, присвоює йому унікальний номер, створює VC, генерує QR-код і обчислює контрольний хеш. Службові дані зберігаються у базі даних, а контрольна інформація фіксується у блокчейн-мережі Ethereum. Після цього користувач отримує посилання на документ і підтверджує право володіння через MetaMask. Перевіряюча сторона може відкрити сторінку перевірки за QR-кодом або номером документа й отримати результат перевірки статусу, хешу, підпису та даних документа.

У межах реалізованого прототипу основний сценарій перевірки виконується через вебзастосунок. Це зроблено для того, щоб користувач і перевіряюча сторона могли отримати результат перевірки через звичайний браузер без ручної роботи з VC-файлами, ключами або блокчейн-інструментами. Сервер автоматично отримує потрібні дані, порівнює їх, перевіряє статус документа та показує зрозумілий результат перевірки.

У разі недоступності серверної частини функціональність системи буде обмежена. Адміністратор не зможе створювати нові документи, переглядати список документів, виконувати анулювання або користуватися адміністративною панеллю. Також не працюватиме стандартна перевірка за QR-кодом, якщо QR-код веде на сторінку вебзастосунку. Однак це не означає, що всі докази достовірності документа втрачаються разом із сервером. Якщо перевіряюча сторона має сам VC-документ, збережений DID-документ або публічний ключ видавця, підписане повідомлення власника та доступ до Ethereum, вона може виконати окрему перевірку криптографічного підпису й контрольного хешу через сторонній verifier-модуль або інший програмний засіб.

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Отже, у роботі реалізовано комбіновану архітектуру. Клієнт-серверна частина забезпечує зручне створення, зберігання, адміністрування та перевірку документів через вебінтерфейс. Децентралізована частина використовується для підвищення довіри до документа за рахунок DID, VC, підпису MetaMask і контрольного хешу в Ethereum. Система не усуває залежність від сервера повністю, але зменшує її порівняно зі звичайною централізованою базою документів, оскільки частина доказів достовірності може бути перевірена незалежно від основної бази даних.

Повністю незалежний verifier-модуль у межах прототипу розглядається як напрям подальшого розвитку. Його призначенням може бути перевірка VC-документа, DID-документа видавця, підпису власника та blockchain-hash без звернення до основного сервера системи. Такий модуль дозволив би розширити можливості незалежної перевірки та ще більше зменшити залежність від доступності вебзастосунку.

Залежність основних функцій системи від серверної та децентралізованої частин наведено в таблиці 1.3.

Таблиця 1.3

Залежність функцій системи від серверної та децентралізованої частин

Функціональність	Залежність від сервера застосунку	Можливість роботи або перевірки без сервера
Створення нового документа	Потрібні адміністративна панель, серверна логіка та база даних	Ні, оскільки документ ще не сформовано
Отримання документа за посиланням	Потрібен сервер застосунку	Ні, якщо посилання веде на сторінку вебзастосунку
Перевірка за QR-кодом через браузер	Потрібен сервер або доступна сторінка перевірки	Ні, якщо QR-код містить лише посилання на вебзастосунок
Перевірка VC-підпису	Може виконуватися сервером або окремим verifier-модулем	Так, за наявності VC, DID-документа та відкритого ключа

Перевірка підпису власника MetaMask	Може виконуватися через сервер або сторонній засіб перевірки підписів	Так, якщо є підписане повідомлення та адреса гаманця
Перевірка контрольного хешу в Ethereum	Не потребує саме сервера застосунку	Так, через доступ до Ethereum, блокчейн-експлорер або API
Перевірка актуального статусу анулювання	Потребує бази даних або дублювання статусу в блокчейні	Частково, залежно від того, чи зафіксовано анулювання у blockchain-записі

Загальна архітектура вебзастосунку включає клієнтський вебінтерфейс, серверну частину, базу даних, модуль формування VC, модуль роботи з DID, модуль генерації QR-кодів, модуль взаємодії з MetaMask, модуль роботи з Ethereum та модуль перевірки документів. Кожен із цих компонентів виконує окрему функцію в загальному процесі створення, видачі та перевірки цифрового документа. Такий поділ дозволяє відокремити інтерфейс користувача від серверної логіки, службового зберігання даних і децентралізованих механізмів підтвердження достовірності.

Загальна архітектура вебзастосунку для адміністрування DID-документів

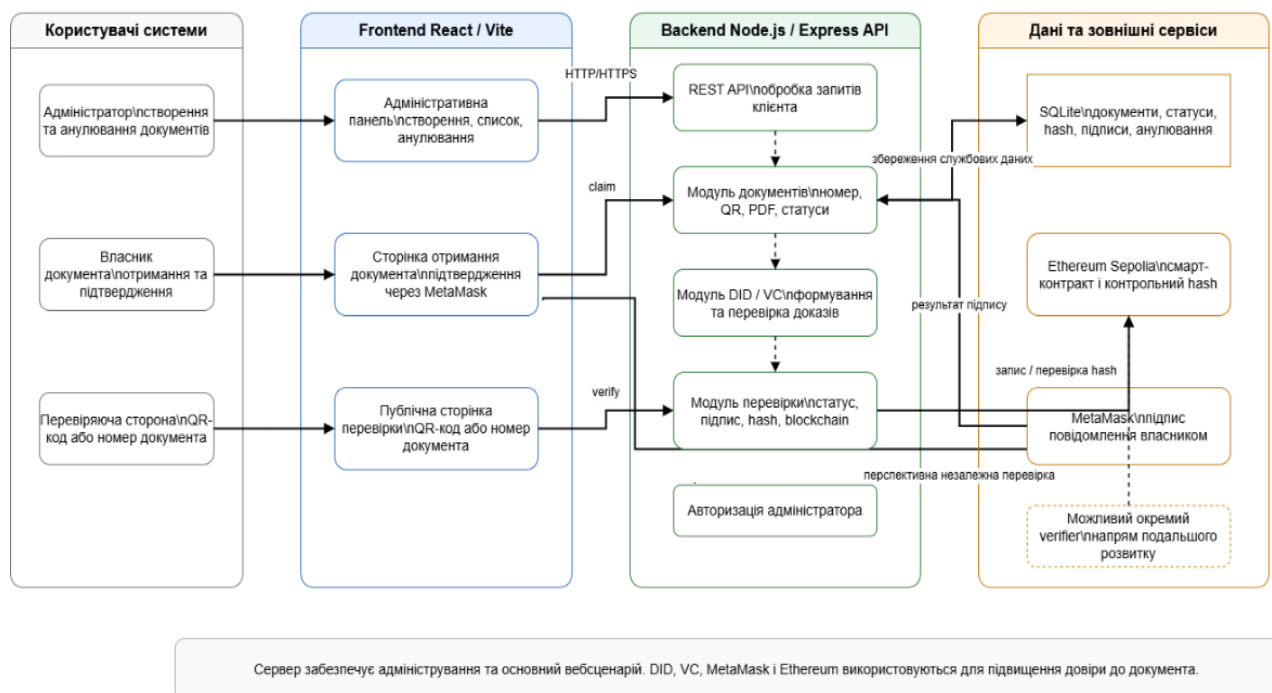


Рис. 1.4 Загальна архітектура вебзастосунку

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Під час розробки вебзастосунку для адміністрування децентралізованих систем ідентифікації важливим етапом є вибір програмних засобів та технологій, які будуть використовуватись для реалізації системи.

Від правильного вибору інструментів залежить зручність розробки, стабільність роботи застосунку та можливість подальшого розширення функціоналу. Для створення клієнтської частини вебзастосунку було використано бібліотеку React. Дана бібліотека дозволяє створювати сучасні вебінтерфейси та забезпечує зручну роботу з компонентами сторінки. Використання React дало можливість реалізувати динамічний інтерфейс користувача, сторінки створення та перевірки цифрових документів, адміністративну панель та форми введення даних.

Для серверної частини застосунку було використано платформу Node.js. Основною причиною вибору Node.js стала можливість використовувати JavaScript як для клієнтської, так і для серверної частини системи. Це спрощує розробку та дозволяє працювати з однією мовою програмування в межах усього проєкту.

Для створення серверної логіки використовувався фреймворк Express. Він використовувався для обробки HTTP-запитів, створення API та взаємодії між клієнтською частиною і базою даних. Express має просту структуру та добре підходить для створення вебзастосунків невеликого та середнього рівня складності.

Для зберігання інформації про цифрові документи використовувалась система керування базами даних SQLite. Дана СУБД не потребує складного налаштування та є зручною для локальних і невеликих вебпроєктів. У базі даних зберігається інформація про створені документи, статуси документів, хеші, адреси криптогаманців та інші службові дані.

Водночас під час розгортання серверної частини у хмарному середовищі необхідно враховувати особливості зберігання SQLite-бази даних. Оскільки

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

SQLite зберігає дані у файловій базі, для стабільної роботи у продакшн-середовищі потрібне постійне сховище або окремий persistent-диск. У межах прототипу SQLite використовується як просте та зручне рішення для демонстрації роботи системи, однак під час промислового впровадження доцільно використовувати persistent-диск Render або перейти на окрему серверну СУБД, наприклад PostgreSQL [17, 26].

Однією з основних особливостей розроблюваної системи є використання блокчейн-технологій. Для підтвердження достовірності цифрових документів було використано блокчейн Ethereum. Дана мережа підтримує роботу зі смарт-контрактами та дозволяє зберігати контрольні дані документів у розподіленому реєстрі.

Для тестування роботи системи використовувалась тестова мережа Sepolia. Використання тестової мережі дозволило перевіряти роботу блокчейн-функцій без використання реальних коштів.

Для взаємодії користувача з блокчейн-мережею використовувався криптогаманець MetaMask. За допомогою MetaMask користувач може підтверджувати право володіння цифровим документом шляхом створення криптографічного підпису.

Під час розробки також використовувались стандарти DID та VC. DID використовуються для створення децентралізованих ідентифікаторів, а VC — для формування цифрових документів із криптографічним підтвердженням достовірності. Для створення QR-кодів використовувалися JavaScript-бібліотеки генерації QR-кодів. QR-коди використовуються для швидкого переходу на сторінку перевірки цифрового документа.

Під час розробки використовувалась система контролю версій Git. Вона дозволяла зберігати історію змін проєкту та контролювати процес розробки.

Для розміщення вебзастосунку використовувалися Render та Vercel. Render використовувався для серверної частини застосунку, а Vercel — для клієнтської частини. Це дозволило отримати доступ до системи через браузер та перевіряти роботу застосунку через мережу Інтернет [16].

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

Для організації клієнтської частини також було використано інструмент Vite, який забезпечує швидке збирання проєкту, запуск локального середовища розробки та зручну роботу з модулями JavaScript. У поєднанні з React це дало змогу швидко перевіряти зміни інтерфейсу, тестувати сторінки адміністратора, сторінки отримання документа та сторінки перевірки без складного налаштування збірки [18, 25].

Використання JavaScript у клієнтській і серверній частинах було доцільним, оскільки система містить значну кількість взаємодій між інтерфейсом, REST API, базою даних та блокчейн-компонентом. REST API дозволяє розділити інтерфейс користувача і серверну логіку: клієнт надсилає запити на створення, отримання, перевірку або анулювання документа, а сервер виконує необхідні дії з базою даних, формує відповідь і повертає результат у форматі JSON [12, 13, 15, 20].

Для роботи з блокчейн-компонентом у системі використовується взаємодія зі смарт-контрактом Ethereum. Смарт-контракт застосовується для фіксації контрольного хешу цифрового документа, збереження ознак видачі, отримання та анулювання. Повний зміст документа не розміщується у блокчейні, що є важливим з погляду захисту персональних даних. У блокчейні зберігається лише контрольна інформація, яка використовується для перевірки незмінності документа [4, 5, 21].

До апаратного забезпечення для роботи із системою не висуваються підвищені вимоги. Для користувача достатньо персонального комп'ютера або ноутбука з доступом до мережі Інтернет, сучасним веббраузером і розширенням MetaMask для сценаріїв отримання документа. Для локального розгортання серверної частини доцільно використовувати комп'ютер із процесором не нижче двоядерного рівня, оперативною пам'яттю від 4 ГБ та вільним місцем на диску для файлів проєкту, встановлених програмних бібліотек та файлу бази даних SQLite.

Для розміщення системи у хмарному середовищі основними вимогами є підтримка запуску Node.js-застосунку, можливість збереження змінних

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

середовища, доступ до зовнішнього RPC-вузла Ethereum Sepolia та наявність постійного сховища для бази даних або підключення до окремої СУБД.

Отже, обраний набір інструментальних засобів є достатнім для реалізації поставлених у роботі завдань: React і Vite забезпечують інтерфейс, Node.js та Express — серверну логіку й REST API, SQLite — зберігання службових даних, а Ethereum, MetaMask, DID та VC — криптографічні механізми підтвердження достовірності цифрових документів [16, 26, 27].

Висновки до першого розділу

У першому розділі було проведено аналіз предметної області та сучасних підходів до реалізації децентралізованих систем ідентифікації. Було розглянуто існуючі програмні рішення у сфері цифрової ідентифікації та цифрових документів, визначено їх основні переваги та недоліки.

У процесі аналізу були сформовані основні вимоги до вебзастосунку для адміністрування цифрових документів, а також обрано клієнт-серверну архітектуру та основні технології реалізації системи.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ДЛЯ АДМІНІСТРУВАННЯ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ІДЕНТИФІКАЦІЇ

2.1 Вимоги до системи

Під час проєктування вебзастосунку для адміністрування децентралізованих систем ідентифікації DID було визначено основні вимоги до функціональності системи, ролей користувачів, порядку роботи з цифровими документами та способів перевірки їхньої достовірності. Розроблювана система призначена для створення, видачі, отримання, перевірки та анулювання цифрових документів із використанням стандартів DID, Verifiable Credential, криптографічного підпису власника та блокчейн-мережі Ethereum [3, 5, 6, 7].

Особливістю розроблюваної системи є поєднання клієнт-серверної архітектури з децентралізованими механізмами перевірки. Серверна частина використовується для адміністрування документів, збереження службових даних, формування QR-кодів, створення структури Verifiable Credential, обробки запитів користувачів та виконання перевірки. Блокчейн-мережа Ethereum використовується не як основне сховище цифрових документів, а як незалежний реєстр контрольної інформації. У блокчейні фіксуються контрольні дані, які дозволяють підтвердити факт створення документа, зіставити його хеш із поточними даними та перевірити ознаку анулювання [7, 10].

У межах системи цифровий документ розглядається не лише як запис у базі даних або PDF-представлення, а як структурований цифровий об'єкт, що має службові дані, криптографічний доказ видавця, хеш документа, QR-код, статус та дані про підтвердження володіння. Такий підхід дозволяє перевірити не тільки наявність документа в системі, а й його цілісність, походження від видавця та актуальний стан.

Основними учасниками системи є адміністратор, власник документа та перевіряюча сторона. Адміністратор виконує функції видавця цифрового документа. Він створює документ, вводить дані власника, формує запис у

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

системі, ініціює фіксацію контрольних даних у блокчейні та може анулювати документ. Власник документа отримує документ за спеціальним посиланням і підтверджує зв'язок із документом за допомогою криптогаманця MetaMask. Перевіряюча сторона не має доступу до адміністративної панелі, але може перевірити документ за QR-кодом або номером документа через публічну сторінку перевірки.

Таблиця 2.1

Ролі користувачів системи

Роль користувача	Призначення ролі	Основні дії в системі	Результат виконання дій
Адміністратор	Виконує функції видавця цифрових документів	Авторизація, створення документа, перегляд списку документів, видача документа, анулювання документа	У системі створюється або змінюється запис цифрового документа
Власник документа	Отримує документ і підтверджує зв'язок із ним	Перехід за claim-посиланням, підключення MetaMask, підпис повідомлення, перегляд документа	До документа прив'язується адреса гаманця власника та криптографічний підпис
Перевіряюча сторона	Перевіряє справжність і актуальність документа	Сканування QR-коду або введення номера документа	Отримує результат перевірки: документ дійсний, анульований, змінений або потребує додаткової перевірки

Загальна логіка роботи системи передбачає проходження документа через декілька етапів. Спочатку адміністратор створює документ у вебінтерфейсі. Після цього система формує службові дані, структуру Verifiable Credential, хеш документа та QR-код. Контрольний хеш документа фіксується у блокчейн-мережі Ethereum. Далі власник документа переходить за

посиланням отримання, підключає MetaMask і підписує повідомлення для підтвердження володіння. Після цього перевіряюча сторона може перевірити документ за QR-кодом або номером.

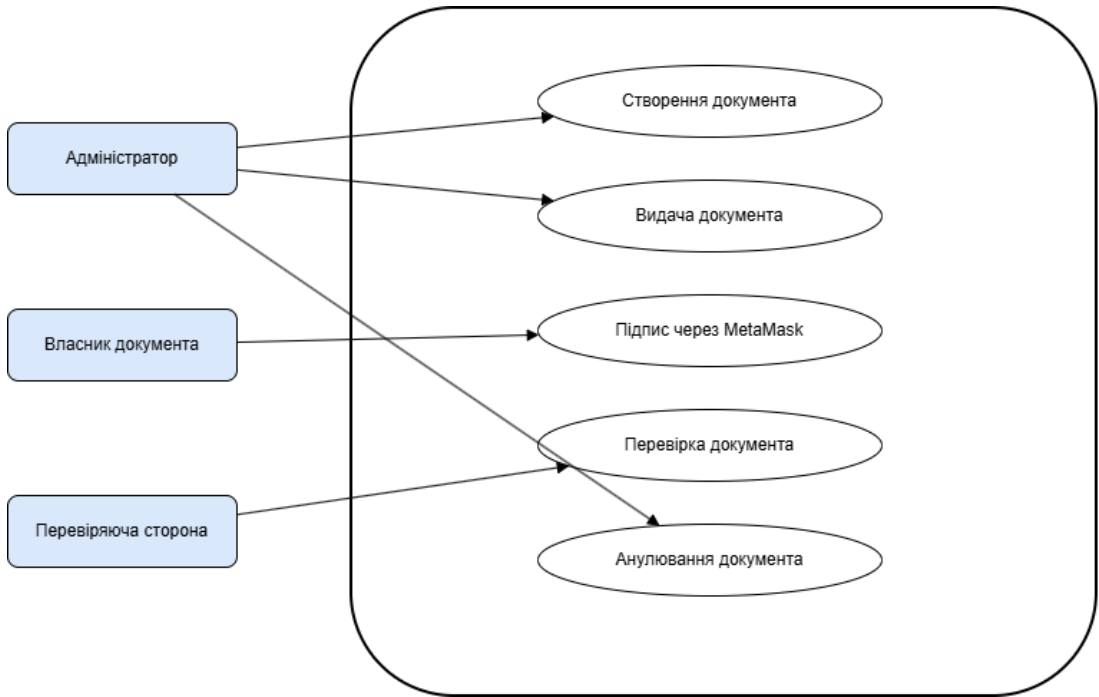


Рис. 2.1 Діаграма варіантів використання

Діаграма варіантів використання відображає поділ системи на три групи функцій. Перша група пов’язана з адмініструванням цифрових документів і доступна лише адміністратору. Друга група стосується отримання документа власником та підтвердження володіння за допомогою MetaMask. Третя група призначена для перевіряючої сторони, яка може перевірити документ без авторизації в адміністративній панелі.

Таблиця 2.2.

№	Варіант використання	Учасник	Вхідні дані	Результат
1	Авторизація адміністратора	Адміністратор	Логін і пароль	Надання доступу до адміністративної панелі
2	Створення цифрового документа	Адміністратор	Тип документа, ПІБ, email, назва, дата видачі, строк дії, опис	Створення запису документа в базі даних

Продовження таблиці 2.2

3	Формування VC-документа	Система	Дані створеного документа	Створення структури Verifiable Credential
4	Формування контрольного хешу	Система	Стабільне JSON-представлення документа	Створення SHA-256 хешу документа
5	Фіксація документа у блокчейні	Система / адміністратор	Номер документа та його контрольний хеш	Збереження контрольних даних у смарт-контракті Ethereum
6	Формування QR-коду	Система	Номер документа та посилання на сторінку перевірки	Створення QR-коду для швидкої перевірки
7	Видача документа	Адміністратор	Дані документа та claim-посилання	Передача власнику посилання для отримання документа
8	Отримання документа	Власник документа	Claim-посилання та токен отримання	Відкриття сторінки отримання документа
9	Підтвердження володіння	Власник документа	Адреса MetaMask і підписане повідомлення	Прив'язка документа до адреси гаманця власника
10	Перевірка документа	Перевіряюча сторона	Номер документа або QR-код	Виведення результату перевірки документа
11	Анулювання документа	Адміністратор	Номер документа, причина анулювання, hash транзакції анулювання	Зміна статусу документа на анульований

Продовження таблиці 2.2

12	Перевірка статусу документа	Система / перевіряюча сторона	Номер документа	Отримання актуального статусу документа
----	-----------------------------	-------------------------------	-----------------	---

Функціональні вимоги визначають перелік можливостей, які повинна забезпечувати система. Вони охоплюють роботу з адміністративною панеллю, цифровими документами, blockchain-підтвердженням, MetaMask-підписом і публічною перевіркою.

Таблиця 2.3

№	Вимога	Опис вимоги	Пов'язаний модуль системи
1	Авторизація адміністратора	Система повинна надавати доступ до адміністративної панелі лише авторизованому адміністратору	Модуль авторизації
2	Створення документа	Система повинна дозволяти створювати документи різних типів: сертифікат, перепустка, посвідчення	Модуль створення документів
3	Збереження даних документа	Система повинна зберігати основні та додаткові поля документа в базі даних	База даних
4	Автоматичне формування номера	Система повинна автоматично формувати унікальний номер документа	Серверний модуль документів
5	Формування VC	Система повинна створювати Verifiable Credential на основі даних документа	Модуль VC
6	Формування підпису видавця	Система повинна створювати службовий proof для підтвердження того, що документ сформовано видавцем	Модуль криптографічного підпису
7	Формування хешу документа	Система повинна створювати SHA-256 хеш стабільного JSON-представлення документа	Модуль хешування
8	Фіксація хешу в Ethereum	Система повинна забезпечувати збереження номера документа та його хешу у смарт-контракті Ethereum	Blockchain-модуль

Продовження таблиці 2.3

9	Формування claim-посилання	Система повинна створювати спеціальне посилання для отримання документа власником	Модуль видачі документа
10	Генерація QR-коду	Система повинна формувати QR-код для переходу на сторінку перевірки документа	Модуль QR
11	Підтвердження через MetaMask	Система повинна дозволяти власнику підписати повідомлення через MetaMask	Модуль отримання документа
12	Збереження адреси гаманця	Система повинна зберігати walletAddress і holderDid після підтвердження документа	База даних
13	Перевірка підпису власника	Система повинна перевіряти, що MetaMask-підпис відповідає адресі гаманця власника	Модуль перевірки
14	Перевірка документа за номером	Система повинна дозволяти знайти та перевірити документ за його номером	Модуль перевірки
15	Перевірка документа за QR-кодом	Система повинна дозволяти перевірити документ через QR-код без авторизації	Публічна сторінка перевірки
16	Перевірка хешу	Система повинна повторно обчислювати хеш поточних даних і порівнювати його зі збереженим хешем	Модуль перевірки
17	Перевірка blockchain-запису	Система повинна перевіряти наявність документа в Ethereum та відповідність його хешу	Blockchain-модуль
18	Анулювання документа	Система повинна дозволяти адміністратору анулювати документ із зазначенням причини	Модуль анулювання
19	Відображення результату перевірки	Система повинна показувати стан документа: дійсний, анульований, змінений або такий, що потребує додаткової перевірки	Сторінка перевірки
20	Облік перевірок	Система повинна зберігати відомості про кількість перевірок і дату останньої перевірки документа	Модуль статистики

У системі реалізовано роботу з кількома типами цифрових документів. Базова логіка для них є спільною: кожен документ має номер, власника, email,

назву, дату видачі, статус, хеш, VC-структуру та дані про blockchain-транзакції. Водночас кожен тип документа має власний набір додаткових полів, що відображають його призначення.

Таблиця 2.4

Тип документа	Префікс номера	Призначення	Додаткові поля
Сертифікат	CERT	Підтвердження проходження курсу, навчання або іншого результату	Назва курсу, оцінка, кількість годин
Перепустка	PASS	Підтвердження права доступу до певних зон або ресурсів	Тип перепустки, роль, підрозділ, рівень доступу, зони доступу
Посвідчення	ID	Підтвердження належності особи до організації або посади	Ідентифікатор, посада, підрозділ, організація

Для забезпечення достовірності документа система повинна перевіряти не один окремий параметр, а сукупність ознак. Перевірка документа включає пошук запису в базі даних, перевірку статусу, перевірку VC-структури, перевірку DID видавця, повторне обчислення хешу, зіставлення хешу із blockchain-записом, перевірку MetaMask-підпису власника та перевірку ознаки анулювання.

Таблиця 2.5

Елемент перевірки	Що перевіряється	Джерело даних	Результат перевірки
Номер документа	Наявність документа в системі	База даних documents	Документ знайдено або не знайдено
Статус документа	Значення status і credentialStatus	База даних documents	Документ активний, виданий, отриманий або анульований
VC-документ	Наявність VC-структури та коректність credentialSubject	Поле з VC-даними документа	Підтвердження структури цифрового документа

Продовження таблиці 2.5

DID видавця	Відповідність issuer.id системному DID	VC JSON і налаштування сервера	Підтвердження походження документа від видавця
Proof видавця	Наявність і коректність службового proof	База даних і відкритий ключ видавця	Підтвердження формування документа видавцем
Хеш документа	Повторно обчислений SHA-256 хеш	Поточні дані документа	Виявлення зміни даних після видачі
Blockchain -запис	Наявність номера документа та відповідність хешу	Смарт- контракт Ethereum	Підтвердження фіксації документа в блокчейні
Підпис власника	Відповідність MetaMask- підпису адреси walletAddress	holderSignature і walletAddress	Підтвердження володіння документом
VP- структура	Наявність verifiable presentation після отримання документа	VP-дані документа	Підтвердження представлення документа власником
Анулюван ня	Наявність revokedAt, revokeReason або ознаки revoked у blockchain	База даних і смарт-контракт	Визначення, чи документ втратив чинність

Окремою вимогою до системи є розмежування функцій між базою даних, серверною частиною та blockchain-мережею. База даних застосунку зберігає повну службову інформацію, необхідну для роботи вебінтерфейсу: дані документа, власника, статуси, VC-структуру, підписи, QR-посилання та результати перевірок. Блокчейн-мережа Ethereum не використовується для збереження персональних даних. У ній фіксуються лише контрольні дані, які потрібні для підтвердження факту видачі документа, його хешу та ознаки анулювання.

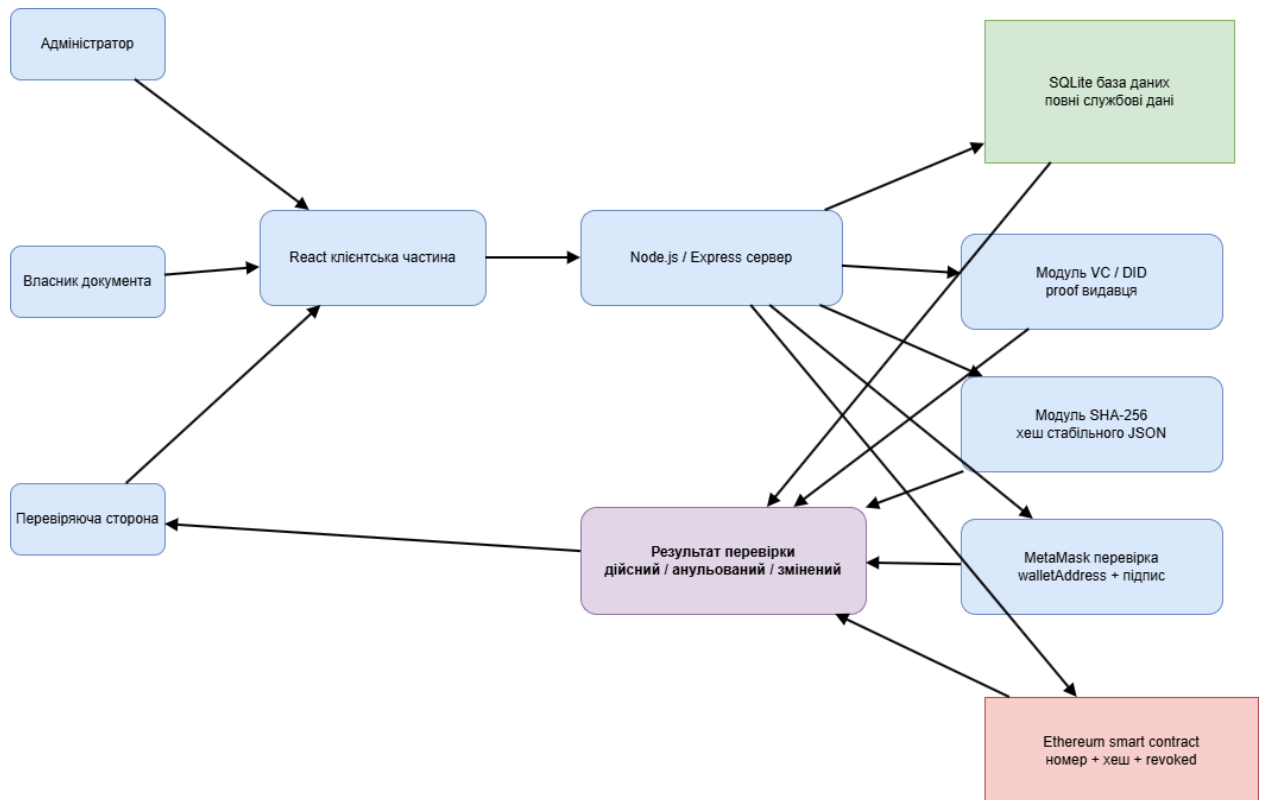


Рис. 2.2 Схема взаємодії компонентів

Схема взаємодії компонентів показує, що користувачі працюють із клієнтською частиною вебзастосунку, яка передає запити до серверної частини. Серверна частина виконує основну бізнес-логіку: створює цифрові документи, формує VC-структуру, обчислює хеш, зберігає дані в базі даних, взаємодіє з Ethereum та виконує перевірку. База даних використовується для збереження повних службових даних документа, а смарт-контракт Ethereum — для збереження контрольних доказів.

У межах системи також необхідно врахувати залежність окремих функцій від доступності сервера. Створення, перегляд, видача та анулювання документів потребують роботи серверної частини, оскільки ці дії пов'язані з базою даних і адміністративною панеллю. Водночас окремі докази достовірності можуть бути перевірені незалежно від основного вебзастосунку, якщо перевіряюча сторона має VC-документ, DID-документ або відкритий ключ видавця, підписане повідомлення власника та доступ до Ethereum.

Таблиця 2.6

Залежність функцій системи від сервера та децентралізованих механізмів

Функціональність	Залежність від сервера застосунку	Можливість роботи або перевірки без сервера
Створення нового документа	Потрібні адміністративна панель, серверна логіка та база даних	Ні, оскільки документ ще не сформовано
Перегляд списку документів	Потрібні сервер і база даних	Ні, це адміністративна функція системи
Видача документа	Потрібні сервер, база даних і blockchain-транзакція	Частково, blockchain-запис можна переглянути окремо, але видати документ без сервера неможливо
Отримання документа за посиланням	Потрібен сервер застосунку	Ні, якщо claim-посилання веде на сторінку вебзастосунку
Перевірка за QR-кодом через браузер	Потрібен сервер або доступна сторінка перевірки	Ні, якщо QR-код містить тільки посилання на вебзастосунок
Перевірка VC	Може виконуватися сервером або окремим verifier-модулем	Так, якщо є VC, DID-документ або відкритий ключ видавця
Перевірка MetaMask-підпису	Може виконуватися сервером або стороннім інструментом перевірки підписів	Так, якщо є підписане повідомлення та адреса гаманця
Перевірка хешу в Ethereum	Не потребує саме сервера застосунку	Так, через Ethereum, blockchain-експлорер або API
Перевірка факту анулювання	Потребує бази даних або blockchain-запису про анулювання	Частково, якщо анулювання зафіксовано у смарт-контракті
Повне адміністрування системи	Потребує сервера, бази даних і адміністративної панелі	Ні

Наявність часткової незалежної перевірки є важливою вимогою до системи. У звичайному сценарії перевірка виконується через вебсторінку, оскільки такий підхід є зручним для користувача. Проте система не повинна зводитися лише до централізованої бази даних. Якщо перевіряюча сторона має

сам VC-документ, DID-документ або відкритий ключ видавця, підписане повідомлення власника та доступ до блокчейн-мережі Ethereum, вона може окремо перевірити криптографічні докази та blockchain-запис. Це зменшує залежність від одного сервера та підвищує рівень довіри до цифрового документа.

Нефункціональні вимоги визначають якісні характеристики системи. Для розроблюваного вебзастосунку важливими є безпека, цілісність даних, захист персональної інформації, зручність перевірки та можливість подальшого розширення.

Таблиця 2.7

Нефункціональні вимоги до системи

№	Вимога	Опис вимоги
1	Безпека доступу	Адміністративна панель повинна бути доступна лише авторизованому адміністратору
2	Захист персональних даних	ПІБ, email та інші персональні дані не повинні записуватися у публічний блокчейн
3	Цілісність документа	Система повинна виявляти зміну даних документа після видачі за допомогою повторного обчислення хешу
4	Незмінність контрольного доказу	Хеш документа повинен фіксуватися у blockchain-записі, який не може бути непомітно змінений
5	Зручність перевірки	Перевіряюча сторона повинна мати можливість перевірити документ через браузер без авторизації
6	Прозорість результату перевірки	Система повинна пояснювати, чому документ підтверджено, анульовано або позначено як такий, що потребує додаткової перевірки
7	Можливість розширення	Архітектура повинна дозволяти додавати нові типи документів без повної зміни логіки системи
8	Сумісність із Web3	Система повинна підтримувати роботу з MetaMask та адресами Ethereum
9	Мінімізація залежності від сервера	Критичні докази достовірності повинні мати форму, яку можна перевірити окремими засобами
10	Надійність анулювання	Анулювання документа повинно відображатися як у базі даних, так і, за наявності транзакції, у blockchain-частині системи

Послідовність роботи системи показує, що створення документа починається з дій адміністратора, але після видачі документ проходить додаткові етапи підтвердження та перевірки. Під час створення документа система формує VC, обчислює хеш і зберігає контрольний запис у Ethereum. Під час отримання документа власник підтверджує володіння через MetaMask. Під час перевірки система порівнює дані з бази даних із хешем, перевіряє blockchain-запис і актуальний статус документа.

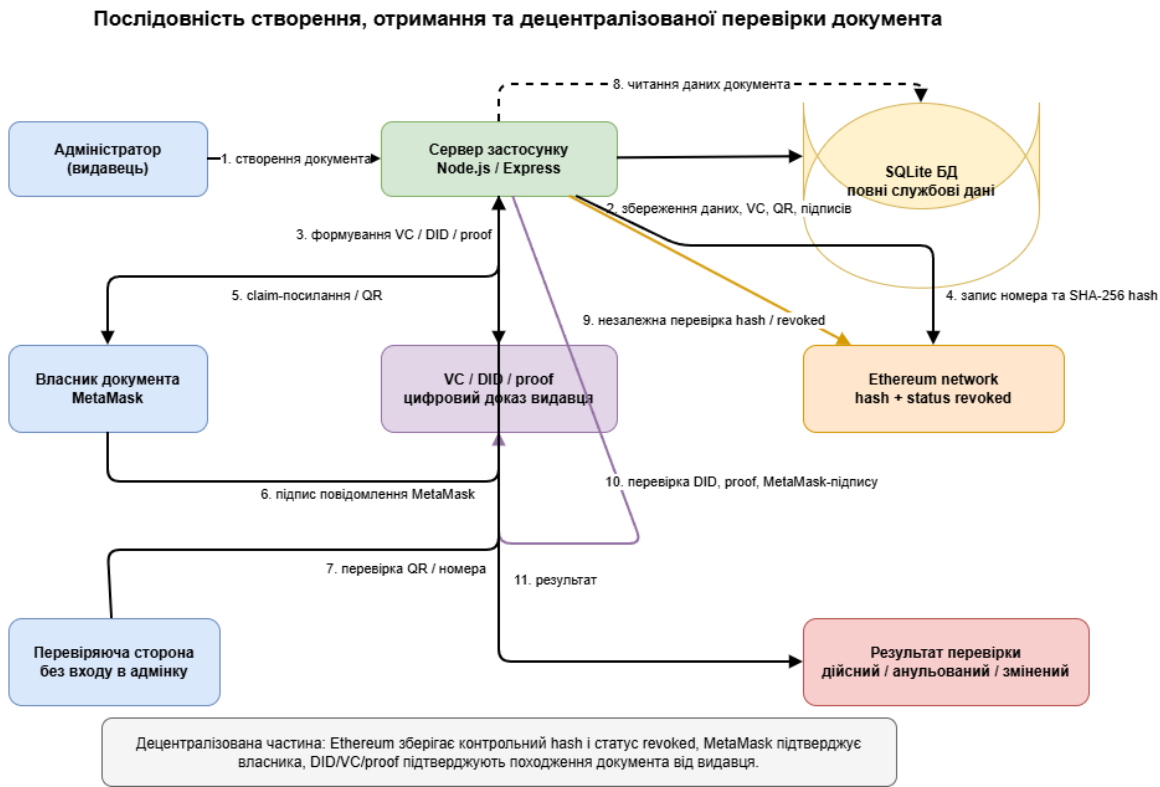


Рис. 2.3 Послідовність створення, отримання та перевірки

У результаті сформовані вимоги визначають логіку подальшого проєктування системи. База даних повинна забезпечувати збереження повної службової інформації про цифрові документи, їх власників, статуси, VC-структури, підписи та результати перевірок. Блокчейн повинен використовуватися для збереження контрольних доказів, зокрема номера документа, хешу, адреси видавця, адреси отримувача та ознаки анулювання. Алгоритми роботи системи повинні забезпечувати створення документа, формування VC, підпис видавця, отримання документа власником, перевірку MetaMask-підпису, перевірку хешу та блокчейн запису.

Таким чином, розроблювана система повинна забезпечувати не лише зручне адміністрування цифрових документів, а й можливість перевірки їхньої достовірності за допомогою криптографічних і децентралізованих механізмів. Це дозволяє відрізнити систему від звичайного вебреєстру документів і обґрунтовує використання DID, VC, MetaMask та Ethereum у межах кваліфікаційної роботи.

2.2 Проєктування бази даних

База даних розроблюваного вебзастосунку призначена для збереження службових і прикладних даних цифрових документів. Вона забезпечує роботу адміністративної панелі, створення документів, збереження VC-структури, QR-посилань, статусів, підписів, даних про отримання документа власником, результатів перевірок та інформації про анулювання.

У системі використовується SQLite, а взаємодія з базою даних реалізована на серверній частині за допомогою бібліотеки better-sqlite3. Основною таблицею є таблиця documents, у якій зберігаються цифрові документи різних типів: сертифікати, перепустки та посвідчення. Окремо використовується таблиця document_verifications, яка призначена для обліку кількості перевірок документа та дати останньої перевірки. Реалізація роботи з документами виконується через універсальну таблицю documents [19].

ER-діаграма зберігання даних цифрових документів

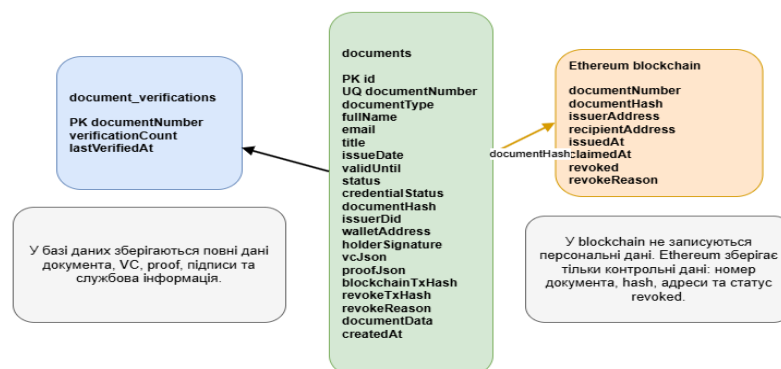


Рис. 2.4 ER-діаграма зберігання даних цифрових документів та взаємодії з blockchain-мережею Ethereum

ER-діаграма показує, що центральною сутністю системи є цифровий документ. Саме з ним пов'язані дані власника, VC-структура, службовий

proof, хеш документа, статус, claim-посилання, адреса гаманця, MetaMask-підпис і відомості про blockchain-транзакції. Таблиця document_verifications пов'язана з документом через номер документа та використовується для фіксації факту перевірки. Окремо на діаграмі показано смарт-контракт Ethereum, оскільки він не є таблицею бази даних, але зберігає контрольні дані, які використовуються під час перевірки документа [3, 5, 6, 7].

Основною таблицею бази даних є documents. Вона є універсальною, оскільки дозволяє зберігати різні типи цифрових документів в одній структурі. Поле documentType визначає тип документа, а поле documentData використовується для збереження додаткових атрибутів, які відрізняються для сертифіката, перепустки або посвідчення.

Таблиця 2.8

Опис полів таблиці documents

Поле	Тип даних	Призначення	Використання в системі
id	INTEGER	Унікальний ідентифікатор запису	Внутрішній первинний ключ таблиці
document Type	TEXT	Тип цифрового документа	Визначає, чи є документ сертифікатом, перепусткою або посвідченням
document Number	TEXT	Унікальний номер документа	Використовується для пошуку, QR-перевірки та зв'язку з blockchain-записом
fullName	TEXT	ПІБ власника документа	Використовується для відображення документа та формування VC
email	TEXT	Електронна пошта власника	Використовується для видачі документа та ідентифікації отримувача
title	TEXT	Назва документа	Відображається у документі та входить до даних VC
issueDate	TEXT	Дата видачі документа	Використовується для формування документа і перевірки строків

Продовження таблиці 2.8

validUntil	TEXT	Дата завершення чинності	Використовується для перевірки актуальності документа
description	TEXT	Опис документа	Зберігає додаткову текстову інформацію
issuer	TEXT	Назва видавця документа	Використовується у VC та інтерфейсі документа
status	TEXT	Внутрішній статус документа	Визначає стан документа в системі: draft, issued, claimed, revoked
credential Status	TEXT	Статус VC-документа	Використовується під час перевірки чинності цифрового документа
document Hash	TEXT	SHA-256 хеш документа	Порівнюється з поточними даними та blockchain-записом
claimToken	TEXT	Токен отримання документа	Забезпечує доступ власника до сторінки отримання
claimUrl	TEXT	Посилання для отримання документа	Передається власнику документа
walletAddress	TEXT	Адреса криптогаманця власника	Зберігається після підтвердження документа через MetaMask
holderDid	TEXT	DID власника документа	Формується на основі адреси гаманця
holderSignature	TEXT	MetaMask-підпис власника	Підтверджує зв'язок документа з адресою гаманця
blockchainTxHash	TEXT	Hash транзакції запису документа	Підтверджує фіксацію хешу документа в Ethereum
blockchainStatus	TEXT	Стан blockchain-публікації	Показує, чи документ був опублікований у blockchain
verificationCount	INTEGER	Кількість перевірок документа	Використовується для статистики перевірок
lastVerifiedAt	TEXT	Дата останньої перевірки	Фіксує останній момент перевірки документа

revokedAt	TEXT	Дата анулювання документа	Показує, коли документ було анульовано
revokeReason	TEXT	Причина анулювання	Відображається під час перевірки анульованого документа
revokeTxHash	TEXT	Hash транзакції анулювання	Підтверджує фіксацію анулювання в Ethereum
issuerDid	TEXT	DID видавця документа	Використовується для перевірки походження документа
vcJson	TEXT	JSON-структура Verifiable Credential	Зберігає цифровий документ у форматі VC
proofJson	TEXT	Службовий proof видавця	Підтверджує формування документа видавцем
vpJson	TEXT	JSON-структура Verifiable Presentation	Зберігає представлення документа після отримання власником
vpProofJson	TEXT	Proof представлення документа	Використовується для перевірки представлення документа власником
documentData	TEXT	Додаткові дані документа у JSON	Зберігає поля, специфічні для сертифіката, перепустки або посвідчення
createdAt	TEXT	Дата створення запису	Фіксує момент створення документа в системі
updatedAt	TEXT	Дата останнього оновлення	Фіксує останню зміну запису документа

Таким чином, таблиця documents поєднує основні дані цифрового документа, службову інформацію для його видачі та технічні поля, необхідні для перевірки достовірності через DID, VC, MetaMask і блокчейн-запис.

Таблиця document_verifications використовується для обліку перевірок документа. Її дані не визначають зміст документа, але дозволяють фіксувати активність перевірки та дату останнього звернення до документа.

Опис полів таблиці document_verifications

Поле	Тип даних	Призначення	Використання в системі
documentNumber	TEXT	Номер документа	Первинний ключ таблиці та зв'язок із документом
verificationCount	INTEGER	Кількість перевірок	Використовується для обліку звернень до сторінки перевірки
lastVerifiedAt	TEXT	Дата останньої перевірки	Показує, коли документ перевірявся востаннє

Окрім основних полів документа, система повинна забезпечувати підтримку різних типів цифрових документів. Для цього використовується універсальна структура зберігання, у якій базові службові поля є спільними для всіх документів, а додаткові атрибути залежать від типу документа.

Наприклад, для сертифікатів додатково можуть використовуватися назва курсу, оцінка та кількість годин, для перепусток — рівень доступу, зони доступу та підрозділ, а для посвідчень — посада, організація та ідентифікатор особи. Такий підхід дозволяє використовувати єдину структуру роботи з цифровими документами та спрощує подальше розширення системи.

Таблиця 2.10

Опис додаткових даних цифрових документів

Тип документа	Додаткові поля	Призначення
Сертифікат	Назва курсу, оцінка, кількість годин	Підтвердження проходження навчання або отриманого результату
Перепустка	Роль, підрозділ, рівень доступу, зони доступу	Контроль доступу до визначених зон або ресурсів
Посвідчення	Посада, організація, ідентифікатор	Підтвердження належності особи до організації або посади

Окремо необхідно розмежувати дані, які зберігаються у базі даних застосунку, та дані, які фіксуються у блокчейн-мережі Ethereum. Це важливо, оскільки цифрові документи містять персональні дані власника. Такі дані не повинні записуватися у публічний blockchain.

У blockchain передаються контрольні дані, необхідні для підтвердження факту створення документа, перевірки цілісності документа та фіксації анулювання.

Таблиця 2.11

Розмежування даних між базою даних і blockchain-мережею Ethereum

Дані	Місце зберігання	Призначення
ПІБ власника	База даних	Відображення документа та формування VC
Email власника	База даних	Видача документа та ідентифікація власника
Тип документа	База даних	Визначення структури цифрового документа
VC JSON	База даних	Збереження структури Verifiable Credential
Proof видавця	База даних	Підтвердження походження документа
MetaMask-підпис	База даних	Підтвердження володіння документом
SHA-256 hash	База даних + Ethereum	Перевірка цілісності документа
Номер документа	База даних + Ethereum	Пошук і blockchain-підтвердження
Адреса видавця	Ethereum	Підтвердження blockchain-запису
Адреса отримувача	Ethereum	Фіксація отримання документа
Ознака revoked	Ethereum	Підтвердження анулювання документа

Таким чином, база даних використовується для повного збереження цифрового документа та роботи вебзастосунку, а blockchain-мережа Ethereum — для незалежного збереження контрольних доказів. Під час перевірки система повторно обчислює SHA-256 хеш документа, порівнює його зі значенням у базі даних і blockchain-записом, а також перевіряє статус документа, DID видавця та MetaMask-підпис власника.

Під час створення документа серверна частина формує структуру Verifiable Credential, створює SHA-256 хеш стабільного JSON-представлення документа та зберігає його в базі даних. Після цього контрольний хеш і номер

документа передаються до smart contract Ethereum. У blockchain не передаються персональні дані власника, що дозволяє уникнути їх публічного розповсюдження [5, 6, 23].

Після отримання документа власник підтверджує володіння ним через MetaMask. У результаті система зберігає адресу криптогаманця, DID власника та MetaMask-підпис. Ці дані використовуються під час подальшої перевірки документа та підтвердження зв'язку документа з конкретним користувачем.

Під час анулювання документа система змінює статус документа у базі даних, зберігає причину анулювання та формує blockchain-транзакцію для оновлення ознаки revoked у smart contract Ethereum. Завдяки цьому факт анулювання може бути перевірений не лише через вебзастосунок, а й через blockchain-запис.

Такий підхід дозволяє поєднати централізовану роботу вебзастосунку з елементами децентралізованої перевірки цифрових документів. База даних забезпечує роботу адміністративної панелі та збереження службової інформації, тоді як blockchain використовується як незалежний контрольний реєстр для перевірки цілісності документа та факту його анулювання.

2.3 Проєктування та реалізація алгоритмів роботи системи

Під час розробки вебзастосунку для адміністрування децентралізованих систем ідентифікації було реалізовано основні алгоритми роботи системи, які забезпечують створення, видачу, перевірку та анулювання цифрових документів [3, 5, 6, 7]. Основною задачею алгоритмів є забезпечення правильної взаємодії між користувачем, серверною частиною застосунку, базою даних та блокчейн-мережею.

Окремо під час проєктування алгоритмів було визначено, які саме дані використовуються на кожному етапі роботи системи, які з них зберігаються у базі даних SQLite, які передаються до blockchain-мережі Ethereum, що саме підписується користувачем через MetaMask та на основі яких даних формується SHA-256 hash документа.

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

Одним із головних алгоритмів системи є алгоритм створення цифрового документа. Під час створення документа адміністратор вводить необхідні дані користувача через вебінтерфейс. Після цього система формує цифровий документ, створює унікальний номер документа та генерує QR-код для подальшої перевірки.

Після формування документа система створює хеш документа та виконує запис контрольних даних у блокчейн-мережу Ethereum. SHA-256 hash формується не лише на основі візуального вмісту документа, а на основі стабільного JSON-представлення цифрового документа та його метаданих. До структури, на основі якої формується hash, входять: номер документа, тип документа, дані власника, credentialSubject, issuer DID, статус документа, VC JSON [23].

Такий підхід дозволяє виявляти будь-які зміни цифрового документа після його видачі, далі інформація про документ зберігається у базі даних, а користувач отримує можливість перегляду або отримання цифрового документа.

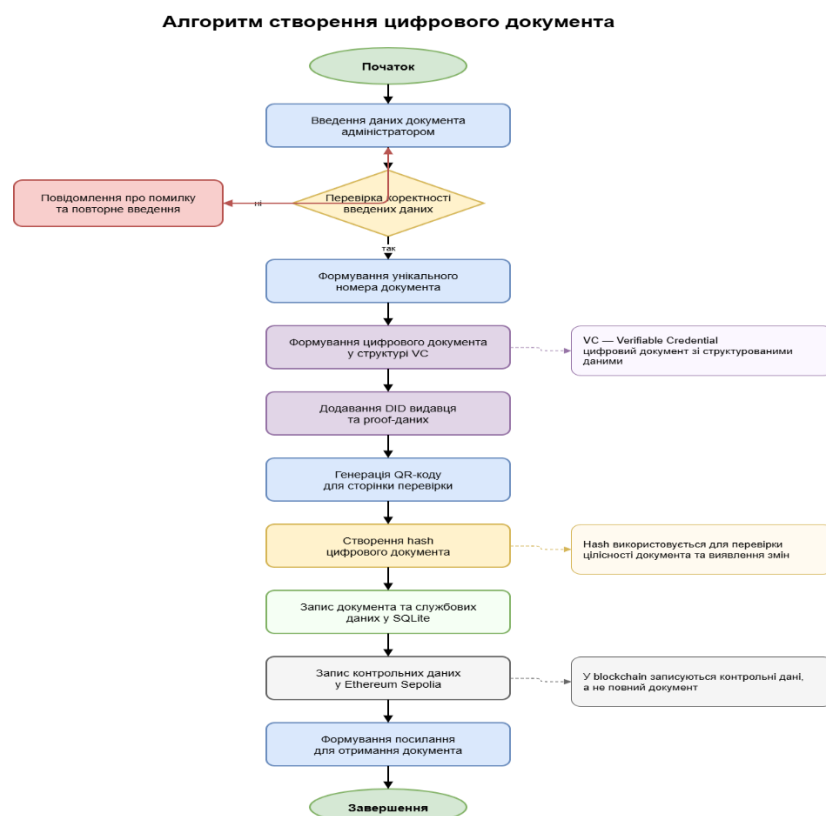


Рис. 2.5 Алгоритм створення цифрового документа

Дані, які використовуються під час створення цифрового документа

Етап	Дані	Призначення	Місце зберігання
Формування документа	fullName, email, documentType, title, issueDate	Створення цифрового документа	SQLite
Формування VC	issuer DID, credentialSubject, credentialStatus	Створення структури Verifiable Credential	SQLite
Формування hash	documentNumber, VC JSON, metadata	Створення SHA-256 hash документа	Використовується для перевірки
Blockchain-запис	documentNumber, SHA-256 hash, issuerAddress	Незалежна blockchain-перевірка	Ethereum
Формування QR	verify URL, documentNumber	Швидка перевірка документа	SQLite / QR

Під час створення документа у blockchain не передається повний зміст цифрового документа. У Ethereum записуються лише контрольні дані: номер документа, SHA-256 hash та адреса видавця. Повні дані документа, VC JSON, службовий proof, дані власника та QR-посилання зберігаються у базі даних SQLite.

Алгоритм видачі документа складається з декількох етапів. Після створення документа система формує спеціальне посилання для отримання документа користувачем. Користувач переходить за посиланням та підтверджує право володіння документом за допомогою криптографічного підпису через MetaMask.

У межах системи підписується не PDF-файл документа, а службове повідомлення, яке містить: номер документа, адресу криптогаманця, timestamp, claimToken [15].

MetaMask використовує приватний ключ користувача для створення криптографічного підпису, який підтверджує право володіння цифровим документом. Приватний ключ не передається серверній частині та не зберігається у системі.

Після успішного підтвердження система зберігає адресу криптогаманця користувача та оновлює статус документа. Це дозволяє підтвердити зв'язок між цифровим документом та власником документа.

Алгоритм отримання та підтвердження цифрового документа

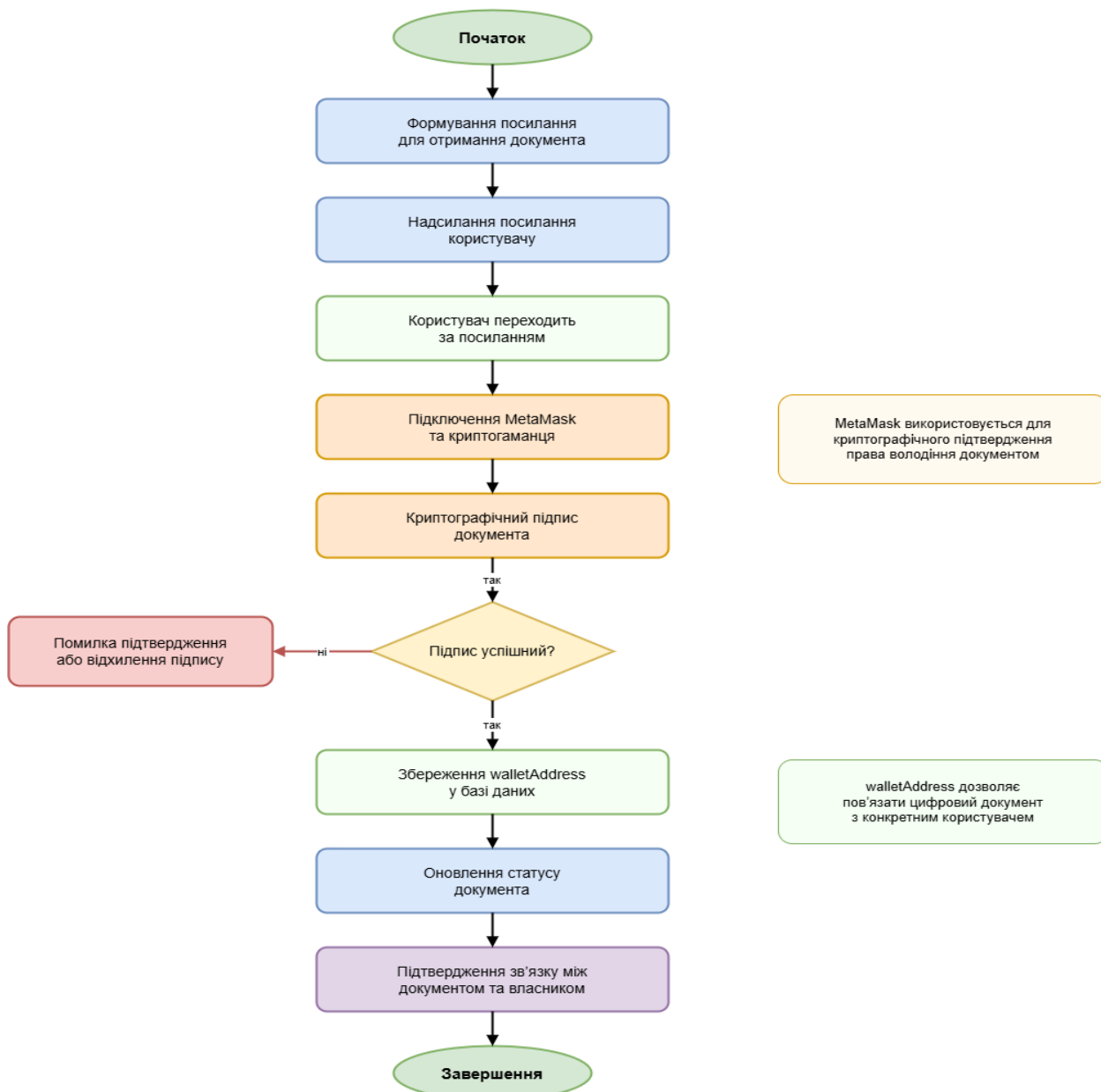


Рис. 2.6 Алгоритм отримання та підтвердження цифрового документа

Одним із важливих алгоритмів системи є алгоритм перевірки цифрового документа. Перевірка може виконуватися за номером документа або через QR-код. Після отримання запиту система знаходить документ у базі даних та виконує перевірку його статусу.

Дані, які використовуються під час підтвердження документа

Етап	Дані	Призначення	Місце зберігання
Claim-посилання	claimToken, documentNumber	Ідентифікація документа під час переходу за посиланням	SQLite
Підключення MetaMask	walletAddress	Отримання адреси криптогаманця власника	SQLite
Підпис повідомлення	documentNumber, walletAddress, timestamp, claimToken	Підтвердження володіння документом	SQLite
Формування holder DID	Ethereum address	Створення DID власника документа	SQLite
Оновлення статусу	status = claimed	Фіксація отримання документа власником	SQLite

Під час отримання документа підписується не файл документа, а службове повідомлення. До нього входять номер документа, адреса криптогаманця, timestamp та claimToken. Такий підпис підтверджує, що власник Ethereum-адреси справді виконав дію отримання документа.

Під час перевірки система також виконує перевірку хешу документа та інформації, яка була записана у блокчейн-мережі. Якщо інформація збігається та документ не був анульований, система повідомляє про успішне підтвердження достовірності документа.



Рис. 2.5 Алгоритм перевірки цифрового документа

Рис. 2.7 Алгоритм перевірки цифрового документа

Під час перевірки система повторно формує стабільне JSON-представлення документа та повторно обчислює SHA-256 hash. Отримане значення hash порівнюється:– із hash, що зберігається у базі даних, із hash, який був записаний у blockchain Ethereum. Якщо значення hash не збігаються, система визначає, що документ був змінений після видачі.

Таблиця 2.14

Дані, які використовуються під час перевірки документа

Етап	Дані	Призначення	Джерело
Пошук документа	documentNumber	Отримання документа	SQLite
Перевірка hash	VC JSON, metadata	Повторне обчислення SHA-256	SQLite
Blockchain-перевірка	documentHash	Порівняння hash	Ethereum
Перевірка DID	issuer DID	Перевірка видавця	VC JSON
Перевірка MetaMask	holderSignature, walletAddress	Підтвердження власника	SQLite
Перевірка статусу	status, credentialStatus, revoked	Перевірка чинності	SQLite + Ethereum

У випадку анулювання документа система повинна відображати відповідний статус документа та причину анулювання. Для цього у вебзастосунку реалізовано алгоритм анулювання документа.

Під час анулювання адміністратор обирає документ у системі та вказує причину анулювання. Після цього статус документа змінюється у базі даних, а інформація про анулювання стає доступною під час перевірки документа.

Під час анулювання документа система оновлює дані як у базі даних SQLite, так і у blockchain-компоненті системи [7, 10, 19].

У базі даних змінюються: status, credentialStatus, revokedAt, revokeReason, revokeTxHash, у smart contract Ethereum оновлюється ознака revoked та службові дані анулювання документа. Під час розробки системи також було реалізовано алгоритм роботи з децентралізованими ідентифікаторами та верифікованими цифровими документами. Для кожного

документа система формує структуру VC та DID, що дозволяє використовувати сучасні підходи до цифрової ідентифікації.

Для кожного цифрового документа система формує структуру Verifiable Credential відповідно до моделі W3C VC Data Model. VC містить issuer DID, credentialSubject, дату видачі документа, тип документа та службовий proof.

Таблиця 2.15

Дані, які оновлюються під час анулювання документа

Етап	Дані	Призначення	Місце оновлення
Оновлення status	status = revoked	Позначення анульованого документа	SQLite
Оновлення credentialStatus	credentialStatus = revoked	Оновлення VC-статусу	SQLite
Збереження revokeReason	Причина анулювання	Відображення під час перевірки	SQLite
Blockchain revoke	revoked = true	Blockchain-підтвердження анулювання	Ethereum
Збереження revokeTxHash	Hash транзакції анулювання	Пошук blockchain-транзакції	SQLite

Під час анулювання документа змінюється не повний VC JSON, а службові поля, які відповідають за актуальний статус документа. У базі даних оновлюються status, credentialStatus, revokedAt, revokeReason та revokeTxHash.

У smart contract Ethereum оновлюється ознака revoked. Завдяки цьому документ може залишатися криптографічно цілісним, але бути недійсним через втрату чинності.

У межах системи використовується DID-метод did:ethr, який дозволяє використовувати Ethereum-адресу як децентралізований ідентифікатор та забезпечує інтеграцію з MetaMask і Web3-інфраструктурою. Ethereum-адреса власника після підтвердження документа використовується для формування holder DID, який зв'язує цифровий документ із конкретним криптогаманцем користувача. [3, 4, 10].

Алгоритм анулювання цифрового документа

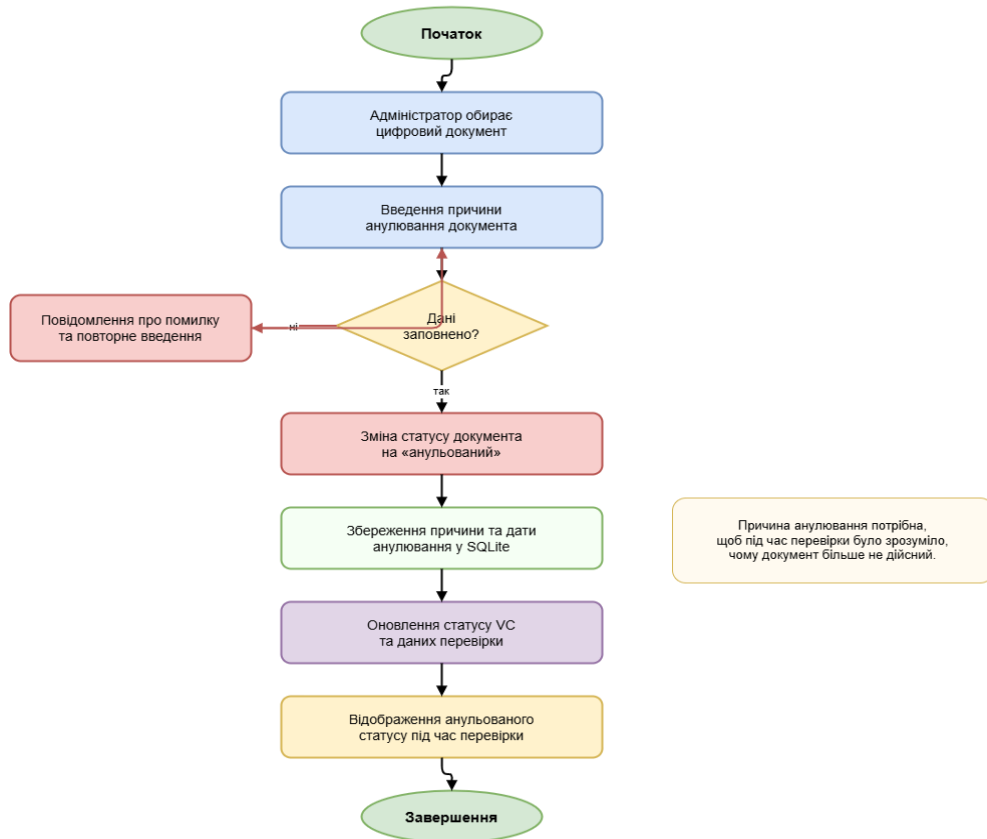


Рис. 2.8 Алгоритм анулювання цифрового документа

VC JSON не перезаписується повністю після анулювання документа. Система оновлює лише `credentialStatus` та службові параметри перевірки документа. Це дозволяє зберегти початкову структуру цифрового документа та одночасно відобразити втрату його чинності. Під час подальшої перевірки система також перевіряє ознаку `revoked` у smart contract Ethereum.

Окрему увагу було приділено алгоритму взаємодії з блокчейн-мережею. Для запису контрольних даних у блокчейн використовуються смарт-контракти Ethereum. Система виконує запис хешу документа та зберігає інформацію про блокчейн-транзакцію.

Для спрощення роботи користувача у системі реалізовано алгоритм автоматичної генерації QR-коду. QR-код містить посилання на сторінку перевірки документа та дозволяє швидко перейти до перевірки через браузер або мобільний пристрій.

Під час проєктування алгоритмів також враховувалась можливість подальшого розширення функціоналу системи. Реалізована структура

дозволяє додавати нові типи цифрових документів та додаткові механізми перевірки без суттєвої зміни основної логіки роботи вебзастосунку.

Таким чином, реалізовані алгоритми забезпечують повний життєвий цикл цифрового документа: створення, формування VC, запис контрольного hash у Ethereum, підтвердження власником через MetaMask, перевірку цілісності та анулювання. Використання DID, Verifiable Credential, MetaMask та blockchain Ethereum дозволяє підвищити рівень довіри до цифрового документа та забезпечити можливість незалежної перевірки його контрольних даних.

2.4 Реалізація функціоналу вебзастосунку

Після проєктування структури системи та основних алгоритмів роботи було виконано реалізацію функціоналу вебзастосунку для адміністрування децентралізованих систем ідентифікації. Основний функціонал системи реалізовано у вигляді клієнтської та серверної частин, які взаємодіють між собою через REST API [15, 20, 30].

Користувачі взаємодіють із системою через браузер за допомогою вебінтерфейсу. Клієнтська частина реалізована за допомогою React, а серверна частина — на Node.js та Express. Для зберігання інформації використовується SQLite, а для реалізації blockchain-взаємодії використовується Ethereum Sepolia бібліотека ethers.js.

Одним із основних елементів системи є адміністративна панель. Через адміністративну панель адміністратор може створювати цифрові документи, переглядати список документів, виконувати перевірку документів та змінювати їх статус. У системі реалізовано підтримку різних типів цифрових документів, що дозволяє використовувати єдину логіку роботи для сертифікатів, посвідчень та перепусток.

Приклад коду наведено нижче:

```
if (documentType === "certificate") {  
  return <CertificateForm />;  
}  
if (documentType === "pass") {  
  return <PassForm />;  
}
```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

```
if (documentType === "id_card") {
  return <IdCardForm />;
}
```

Під час створення документа адміністратор вводить дані користувача через спеціальну форму. Після заповнення форми система автоматично створює цифровий документ, генерує унікальний номер документа та QR-код для подальшої перевірки.

Приклад коду наведено нижче:

```
const year = new Date().getFullYear();
const documentNumber =
  `${config.prefix}-${year}-${crypto.randomBytes(4)
    .toString("hex").toUpperCase()}`;
```

У системі реалізовано підтримку різних типів цифрових документів. Для кожного типу документа визначається власний префікс, тип Verifiable Credential та шаблон відображення.

Приклад коду наведено нижче:

```
const DOCUMENT_TYPES = {
  certificate: {
    prefix: "CERT",
    label: "Сертифікат",
    vcType: "DigitalCertificate"
  },
  pass: {
    prefix: "PASS",
    label: "Перепустка",
    vcType: "AccessPass"
  },
  id_card: {
    prefix: "ID",
    label: "Посвідчення",
    vcType: "IdentityDocument"
  }
};
```

Після створення документа система формує криптографічний hash документа та виконує запис контрольних даних у blockchain-мережу Ethereum. Blockchain використовується як незалежний механізм перевірки цілісності документа.

Приклад коду наведено нижче:

```
return crypto.createHash("sha256").update(stableStringify({
  documentType: document.documentType,
  documentNumber: document.documentNumber,
  fullName: document.fullName,
  email: document.email,
  issuerDid: document.issuerDid || ISSUER_DID,
  documentData: data
})).digest("hex");
```

У наведеному фрагменті коду формується SHA-256 hash документа. Надалі сформований hash використовується для blockchain-перевірки та порівняння із записом у blockchain.

Процес створення та blockchain-фіксації документа

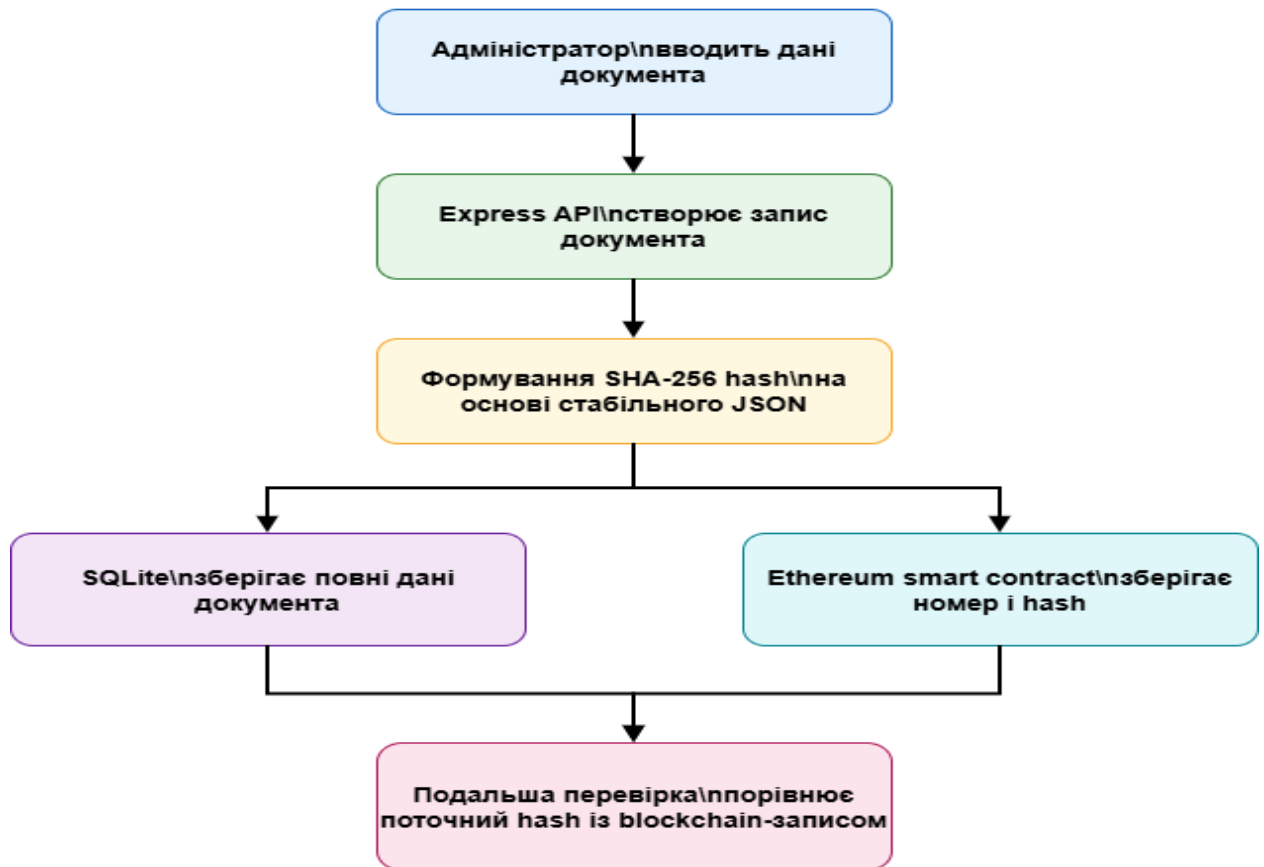


Рис. 2.9 Схема створення та blockchain фіксація документа

Для підтвердження права володіння цифровим документом у системі реалізовано взаємодію з криптогаманцем MetaMask. Користувач підтверджує право володіння документом шляхом криптографічного підпису повідомлення. Після підтвердження система зберігає адресу криптогаманця користувача та виконує формування децентралізованого ідентифікатора [15].

Приклад коду наведено нижче:

```

const accounts = await window.ethereum.request({
  method: "eth_requestAccounts"
});
const walletAddress = accounts[0];
const provider = new ethers.BrowserProvider(window.ethereum);
const signer = await provider.getSigner();
const acceptMessage = `Accept credential ${documentNumber}`;
const signature = await signer.signMessage(acceptMessage);
  
```

У межах вебзастосунку реалізовано підтримку децентралізованих ідентифікаторів. Даний DID-метод базується на blockchain-мережі Ethereum та дозволяє використовувати Ethereum-адресу користувача як основу цифрової ідентичності.

Особливістю DID є відсутність централізованого сервера керування ідентифікаторами. Ідентифікатор формується на основі Ethereum-адреси користувача та може бути незалежно перевірений будь-яким учасником системи. Формування DID виконується після отримання Ethereum-адреси користувача через MetaMask.

Приклад коду наведено нижче:

```
did:ethr:${walletAddress};
```

Приклад сформованого децентралізованого ідентифікатора:

```
did:ethr:0x742d35Cc6634C0532925a3b844Bc454e4438f44e
```

Сформований DID використовується під час створення Verifiable Credential та підтвердження права володіння цифровим документом. Для забезпечення незалежної перевірки DID у системі реалізовано підтримку DID Document. DID Document містить інформацію про відкритий ключ видавця, метод перевірки та параметри криптографічного підтвердження цифрових документів.

Приклад коду наведено нижче:

```
app.get("/.well-known/did.json", (req, res) => {
  res.json({
    "@context": ["https://www.w3.org/ns/did/v1"],

    id: ISSUER_DID,

    verificationMethod: [{
      id: DID_VERIFICATION_METHOD,
      type: "JsonWebKey2020",
      controller: ISSUER_DID,
      publicKeyPem: ISSUER_PUBLIC_KEY
    }],
    assertionMethod: [
      DID_VERIFICATION_METHOD
    ]
  });
});
```

У наведеному фрагменті коду формується DID Document відповідно до стандарту W3C DID Core. Поле verificationMethod містить інформацію про метод перевірки та відкритий ключ видавця цифрових документів.

Під час реалізації системи використовувались технології W3C Verifiable Credentials. Для кожного документа формується структура Verifiable Credential, що містить інформацію про власника документа, видавця документа та криптографічний proof достовірності.

Приклад формування Verifiable Credential наведено нижче:

```
const vc = {
  "@context": [
    "https://www.w3.org/2018/credentials/v1"
  ],

  id:
    urn:uuid:${document.documentNumber},

  type: [
    "VerifiableCredential",
    config?.vcType || "DigitalDocument"
  ],

  issuer: {
    id: ISSUER_DID,
    name: document.issuer
  },

  issuanceDate: document.issueDate,

  credentialSubject: {
    id: document.holderDid || "",
    documentType: document.documentType,
    documentNumber: document.documentNumber,
    fullName: document.fullName,
    email: document.email,
    title: document.title
  }
};
```

Поле issuer містить DID видавця документа, а credentialSubject містить інформацію про власника документа та його децентралізований ідентифікатор. Для криптографічного підтвердження достовірності документа у системі використовується цифровий proof.

Приклад коду наведено нижче:

```
const vcJson = buildVerifiableDocumentCredential(document);
const proofJson = buildIssuerProof(vcJson);

db.prepare(`
  UPDATE documents SET
    vcJson = ?,
    proofJson = ?,
```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        updatedAt = ?
        WHERE documentNumber = ?
    `).run(
        JSON.stringify(vcJson),
        JSON.stringify(proofJson),
        new Date().toISOString(),
        document.documentNumber
    );

```

У наведеному фрагменті коду створюється криптографічний proof документа. Поле verificationMethod містить посилання на DID видавця документа, а поле jws містить криптографічний підпис документа.

Використання криптографічного proof дозволяє забезпечити перевірку цілісності цифрового документа та підтвердити справжність його походження. У випадку зміни структури документа або його вмісту результат криптографічної перевірки буде недійсним, що дозволяє виявити спробу модифікації цифрового документа.

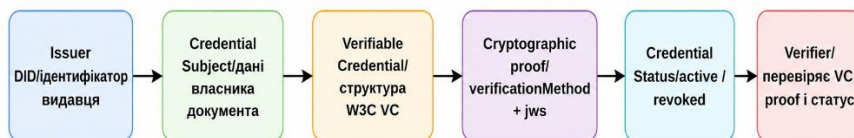


Рис. 2.10 Схема формування Verifiable Credential та криптографічного підтвердження

Однією з важливих функцій вебзастосунку є перевірка цифрових документів. Перевірка може виконуватися за номером документа або через QR-код. Під час перевірки система аналізує статус документа, перевіряє DID видавця, криптографічний proof, blockchain-дані та credentialStatus документа.

Приклад коду наведено нижче:

```

if (data.revoked) {
    setVerifyState("error");
    setMessage("Документ анульовано");
    return;
}
if (data.valid) {
    setVerifyState("success");
    setMessage("Документ підтверджено");
    return;
}

```

Процес перевірки цифрового документа DID/VC

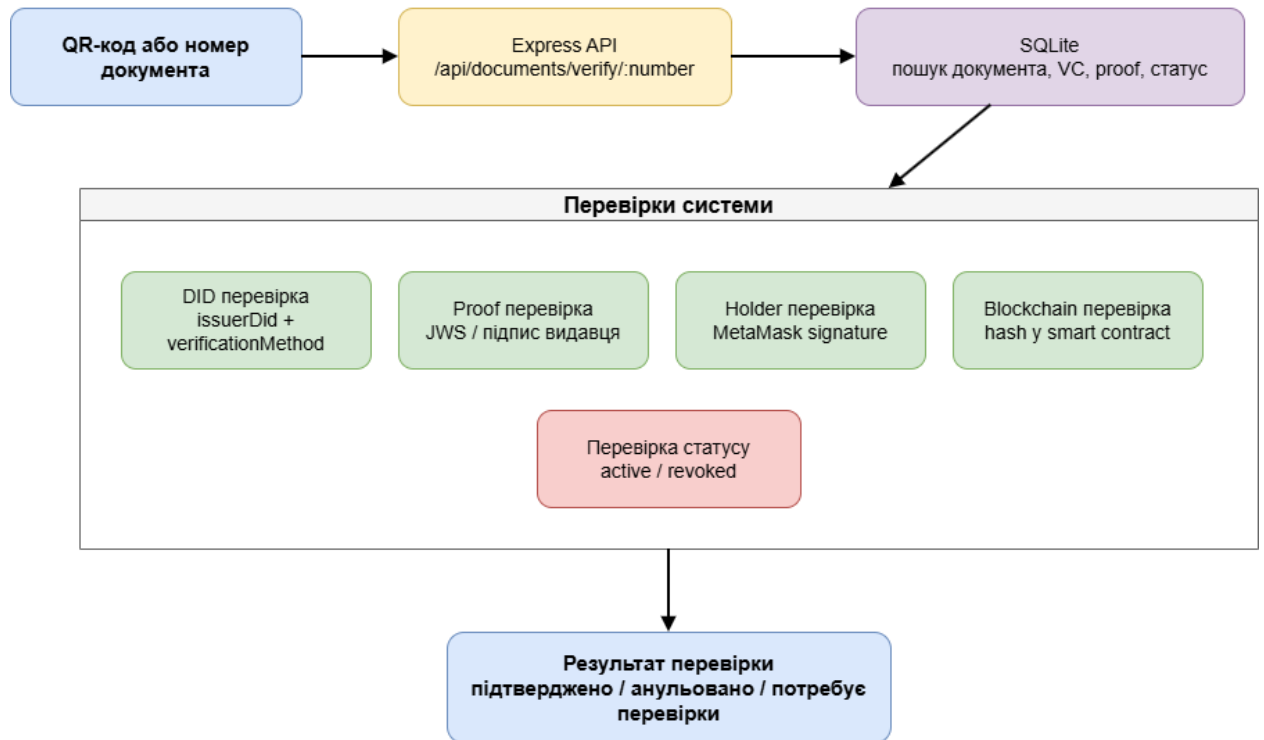


Рис. 2.11 Схема процесу перевірки цифрового документа

У вебзастосунку також реалізовано функцію анулювання цифрових документів. Адміністратор може змінити статус документа та вказати причину анулювання. Після цього інформація про анулювання відображається під час перевірки документа.

Приклад коду наведено нижче:

```

db.prepare( UPDATE documents SET status = ?, credentialStatus = ?, revokedAt =
?, revokeReason = ?, revokeTxHash = ?, updatedAt = ? WHERE documentNumber =
?).run(
"revoked",
"revoked",
new Date().toISOString(),
reason.trim(),
revokeTxHash,
new Date().toISOString(),
documentNumber
);
  
```

Для генерації QR-кодів використовувалися JavaScript-бібліотеки, які дозволяють автоматично створювати QR-коди для кожного документа.

Приклад коду наведено нижче:

```

const claimUrl =
${FRONTEND_URL}/receive/${document.documentNumber}?token=${claimToken};
  
```

Серверна частина вебзастосунку реалізована на Node.js та Express. Сервер обробляє запити користувачів, виконує роботу з базою даних, забезпечує взаємодію з blockchain-мережею та реалізує логіку перевірки документів.

Приклад коду наведено нижче:

```
app.get("/api/documents/",
  (req, res) => {

  const document =
  getDocumentByNumber(
    req.params.documentNumber
  );

  if (!document) {
    return res.status(404).json({
      success: false,
      message: "Документ не знайдено"
    });
  }
  return res.json({
    success: true,
    document:
      normalizeDocument(document)
  });
});
```

Для зберігання інформації про документи використовується SQLite. У базі даних зберігаються дані документів, статуси, адреси криптогеманців, hash документів, VC-структури та криптографічні proof.

Приклад структури таблиці наведено нижче:

```
db.exec(`
  CREATE TABLE IF NOT EXISTS documents (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    documentType TEXT NOT NULL,
    documentNumber TEXT UNIQUE NOT NULL,
    fullName TEXT NOT NULL,    email TEXT NOT NULL,
    status TEXT NOT NULL DEFAULT 'draft',
    walletAddress TEXT,
    documentHash TEXT,
    vcJson TEXT,
    proofJson TEXT
  )
`);
```

Для розміщення вебзастосунку використовувалися Render та Vercel. Це дозволило забезпечити доступ до системи через мережу Інтернет та перевірити роботу застосунку в реальних умовах.

Під час тестування вебзастосунку було перевірено роботу створення документів, підтвердження права володіння, перевірки документів та

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

анулювання документів. Результати тестування показали коректну роботу основного функціоналу системи.

Таким чином, реалізований вебзастосунок забезпечує створення, видачу, перевірку та анулювання цифрових документів із використанням технологій децентралізованої ідентифікації DID, W3C Verifiable Credentials та blockchain-технологій.

Висновки до другого розділу

У другому розділі було виконано проєктування та реалізацію вебзастосунку для адміністрування децентралізованих систем ідентифікації DID. Було спроектовано структуру бази даних, реалізовано серверну та клієнтську частини системи, а також алгоритми створення, видачі, перевірки та анулювання цифрових документів.

Крім цього, було реалізовано інтеграцію з блокчейн-мережею Ethereum, підтримку QR-кодів та механізми підтвердження справжності цифрових документів.

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБЗАСТОСУНКОМ ДЛЯ АДМІНІСТРУВАННЯ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ІДЕНТИФІКАЦІЇ

3.1 Розгортання та налаштування вебзастосунку

Після завершення розробки вебзастосунку було виконано його розгортання у хмарному середовищі. Для забезпечення незалежної роботи клієнтської та серверної частин застосунку використано окремі платформи розгортання. Клієнтська частина розгортається на платформі Vercel, серверна частина — на платформі Render, а взаємодія зі смарт-контрактом здійснюється через blockchain-мережу Ethereum Sepolia. Такий підхід дозволяє незалежно оновлювати окремі компоненти системи, забезпечує їх масштабованість та спрощує подальшу підтримку програмного забезпечення [16, 17, 20, 21].

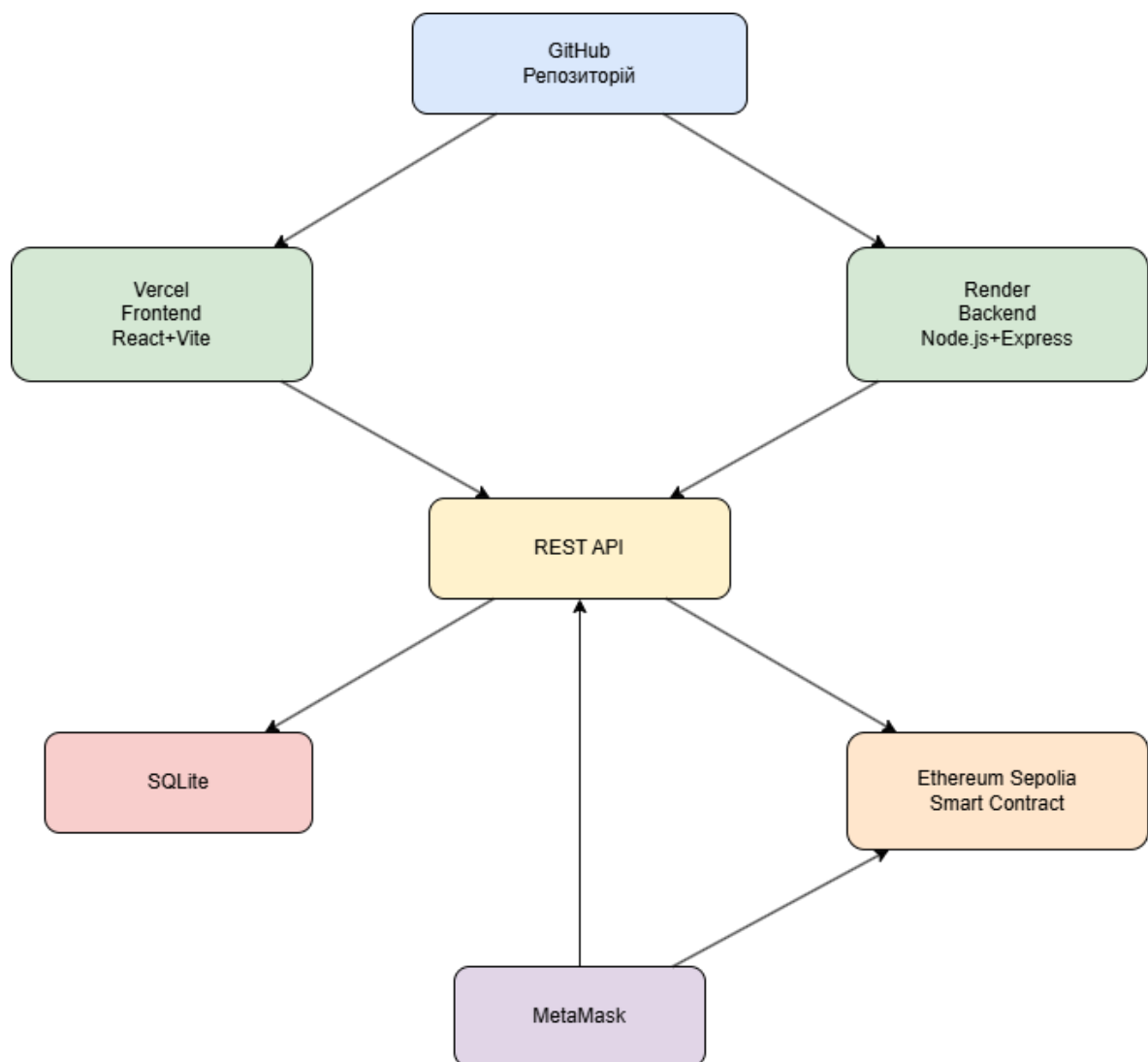


Рис. 3.1 Схема розгортання вебзастосунку

Вихідний код програмного забезпечення зберігається у репозиторії GitHub. Після внесення змін до репозиторію автоматично запускається процес збірки та оновлення окремих компонентів системи. Клієнтська частина автоматично розгортається платформою Vercel, а серверна частина — платформою Render. Це дозволяє підтримувати актуальну версію вебзастосунку без ручного копіювання файлів на сервер.

Клієнтська частина вебзастосунку реалізована із використанням React та Vite. Вона забезпечує відображення інтерфейсу адміністратора, сторінок отримання документа, перевірки документа та взаємодію користувача із серверною частиною через REST API. Після виконання автоматичної збірки Vercel публікує нову версію клієнтської частини та забезпечує її доступність через HTTPS.

Серверна частина побудована на платформі Node.js із використанням Express. Backend забезпечує роботу REST API, взаємодію із базою даних SQLite, формування структур Verifiable Credential, створення SHA-256 hash документа, взаємодію зі смарт-контрактом Ethereum та перевірку цифрових документів. Після автоматичного розгортання Render запускає серверний застосунок та забезпечує його постійну доступність [17, 18].

Для зберігання службових даних використовується база даних SQLite. У ній зберігаються цифрові документи, VC JSON, службові proof, адреси криптогаманців власників, результати перевірок, інформація про анулювання документів та інші службові дані. База даних використовується лише серверною частиною вебзастосунку і недоступна безпосередньо користувачам.

У межах розробленого прототипу SQLite використовується для спрощення розгортання та тестування системи. Однак під час розміщення backend-частини на Render потрібно враховувати, що локальна файлова система без persistent-диска не гарантує довготривале збереження змін після перезапуску або повторного розгортання сервісу. Тому для стабільної

експлуатації системи необхідно використовувати persistent-диск або винести зберігання даних в окрему керовану базу даних [26].

Blockchain-компонент системи працює у мережі Ethereum Sepolia. Взаємодія із мережею виконується через RPC-провайдера Infura із використанням бібліотеки ethers.js. Смарт-контракт забезпечує збереження контрольного SHA-256 hash документа, фіксацію адреси видавця, інформації про отримання документа та ознаки його анулювання. Персональні дані власників документів до blockchain не записуються [21, 28].

Для підтвердження права володіння цифровим документом використовується криптогаманець MetaMask. Під час отримання документа користувач підписує службове повідомлення власним приватним ключем. Сервер отримує лише результат криптографічного підпису, тоді як приватний ключ користувача не передається та не зберігається у системі [22].

Під час розгортання застосунку використовуються змінні середовища, які дозволяють відокремити службові параметри конфігурації від програмного коду.

Таблиця 3.1

Основні параметри конфігурації вебзастосунку

Параметр	Призначення
VITE_API_URL	Адреса REST API серверної частини
FRONTEND_URL	Адреса клієнтської частини
VITE_CONTRACT_ADDRESS	Адреса смарт-контракту для frontend
CONTRACT_ADDRESS	Адреса смарт-контракту Ethereum для backend
VITE_SEPOLIA_RPC_URL	RPC-підключення frontend до мережі Ethereum Sepolia
ISSUER_PRIVATE_KEY	Приватний ключ видавця для підпису blockchain-транзакцій
RESEND_API_KEY	API-ключ для надсилення електронних листів
INFURA_API_KEY	API-ключ доступу до Ethereum RPC через Infura

Оновлення вебзастосунку автоматизовано за допомогою GitHub, Vercel та Render. Після внесення змін до основної гілки репозиторію автоматично

запускається процес повторної збірки клієнтської та серверної частин. Це дозволяє швидко розгортати нові версії вебзастосунку без зупинки його роботи та мінімізує ризик помилок під час оновлення програмного забезпечення [26, 27].

3.2 Користувацький інтерфейс системи

Під час розробки вебзастосунку значна увага приділялась створенню зручного та зрозумілого користувацького інтерфейсу. Інтерфейс системи повинен забезпечувати просту взаємодію користувачів із цифровими документами та основними функціями вебзастосунку [25].

Основною задачею користувацького інтерфейсу є забезпечення швидкого доступу до функцій створення, перевірки та отримання цифрових документів. Інтерфейс вебзастосунку реалізований у вигляді односторінкового вебзастосунку, що дозволяє швидко переходити між сторінками без повного перезавантаження браузера.

Головна сторінка системи містить основну інформацію про вебзастосунок та навігаційне меню. Через головну сторінку користувач може перейти до сторінки перевірки документа, сторінки отримання документа або адміністративної панелі.

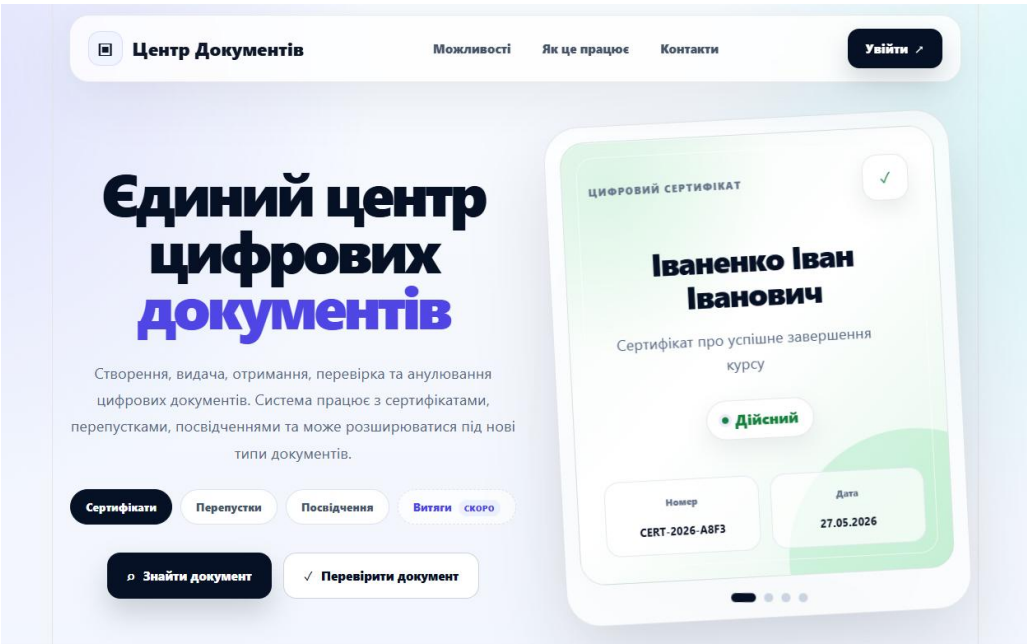


Рис. 3.2 Головна сторінка вебзастосунку

Для перевірки цифрових документів у системі реалізовано окрему сторінку перевірки. Користувач може ввести номер документа або перейти на сторінку перевірки через QR-код.

Після виконання перевірки система відображає інформацію про документ та його статус.

Центр Документів

На головну

Перевірка документа

Перевірити документ

Введіть номер документа. Система перевірить W3C VC, DID емітента, Holder DID, підпис власника, статус відкликання та blockchain hash.

Номер документа

CERT-2026-XXXX або PASS-2026-XXXX

Перевірити документ

Рис. 3.3 Сторінка перевірки документа

У випадку успішної перевірки користувач отримує повідомлення про достовірність документа.

Центр Документів

Нова перевірка

На головну

✓

Публічна перевірка

Документ підтверджено

Номер документа: CERT-2026-36AF9DE1

i

Звірте документ з офіційними даними

Порівняйте дані документа з інформацією, що відображається нижче. Якщо ПІБ, номер документа, статус або інші реквізити не збігаються, не довіряйте документу та вважайте його недійсним до додаткової перевірки. Офіційними є лише дані, отримані через сторінку перевірки.

Рис. 3.4 Статус підтвердженого документа

Якщо документ був анульований або не існує у системі, відображається відповідний статус перевірки.

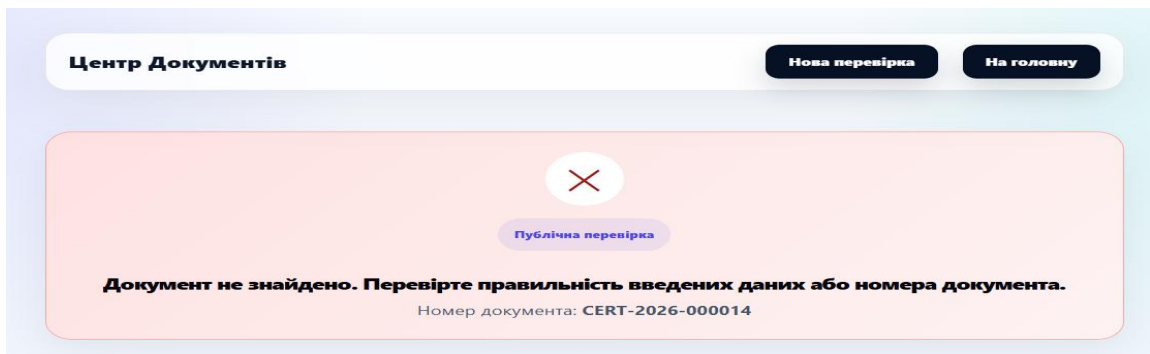


Рис. 3.5 Статус незнайденого документа

Якщо документ потребує додаткової перевірки, відображається відповідний статус перевірки.

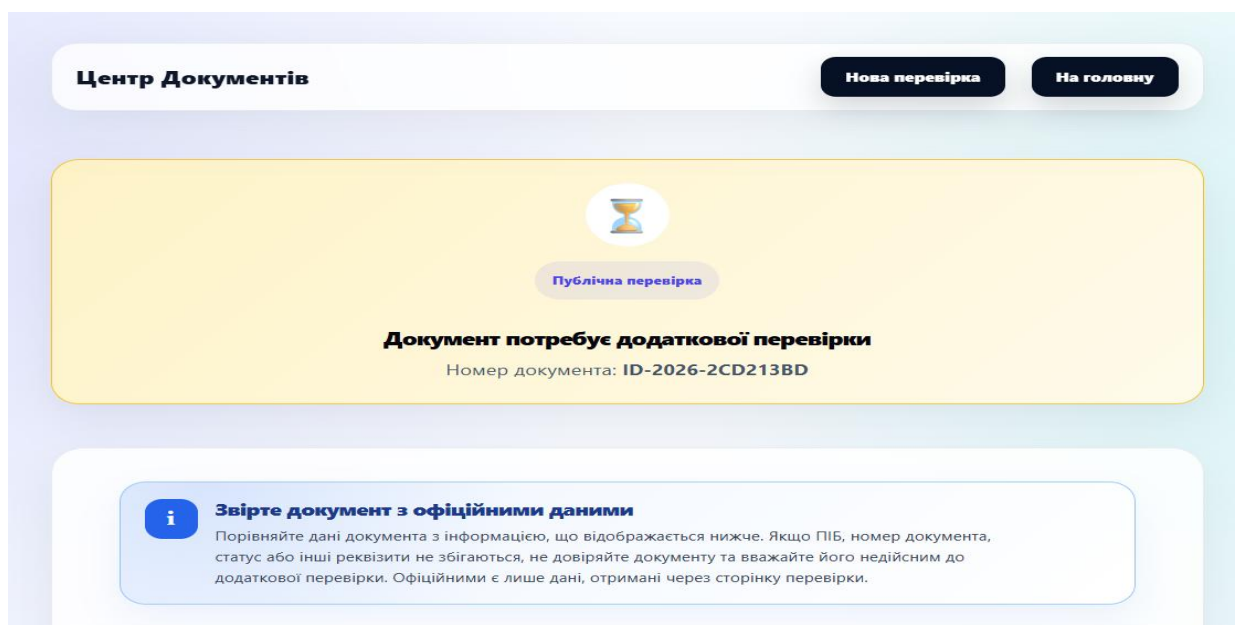


Рис. 3.6 Статус документа, який потребує додаткової перевірки

Для спрощення роботи користувачів у системі реалізовано використання QR-кодів. Кожен цифровий документ містить QR-код, який дозволяє швидко перейти до сторінки перевірки документа через браузер або мобільний пристрій.

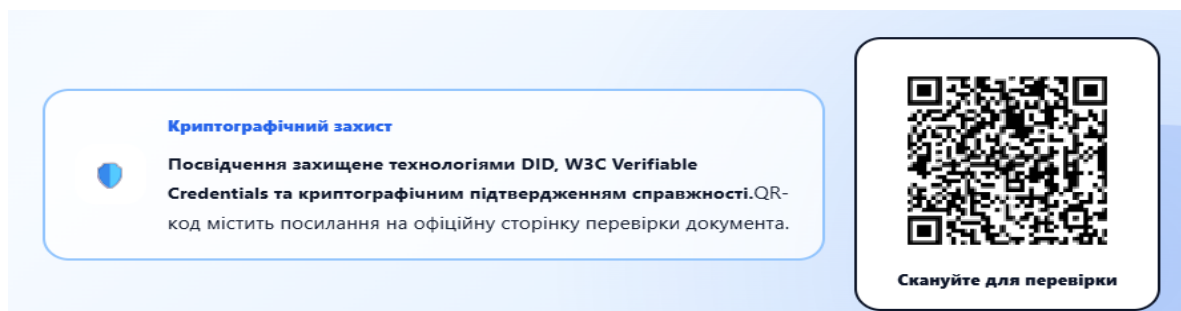


Рис. 3.7 Нижня частина візуального відображення документа (QR-код)

У вебзастосунку також реалізовано сторінку отримання документа. Після переходу за спеціальним посиланням користувач може переглянути цифровий документ та підтвердити право володіння документом за допомогою криптографічного підпису через MetaMask. Під час підтвердження виконується підпис службового повідомлення, яке містить номер документа, адресу криптогаманця, часову позначку та унікальний токен отримання документа (claimToken). Після успішної перевірки підпису система пов'язує Ethereum-адресу користувача з цифровим документом та оновлює його статус, підтверджуючи факт отримання документа законним власником.

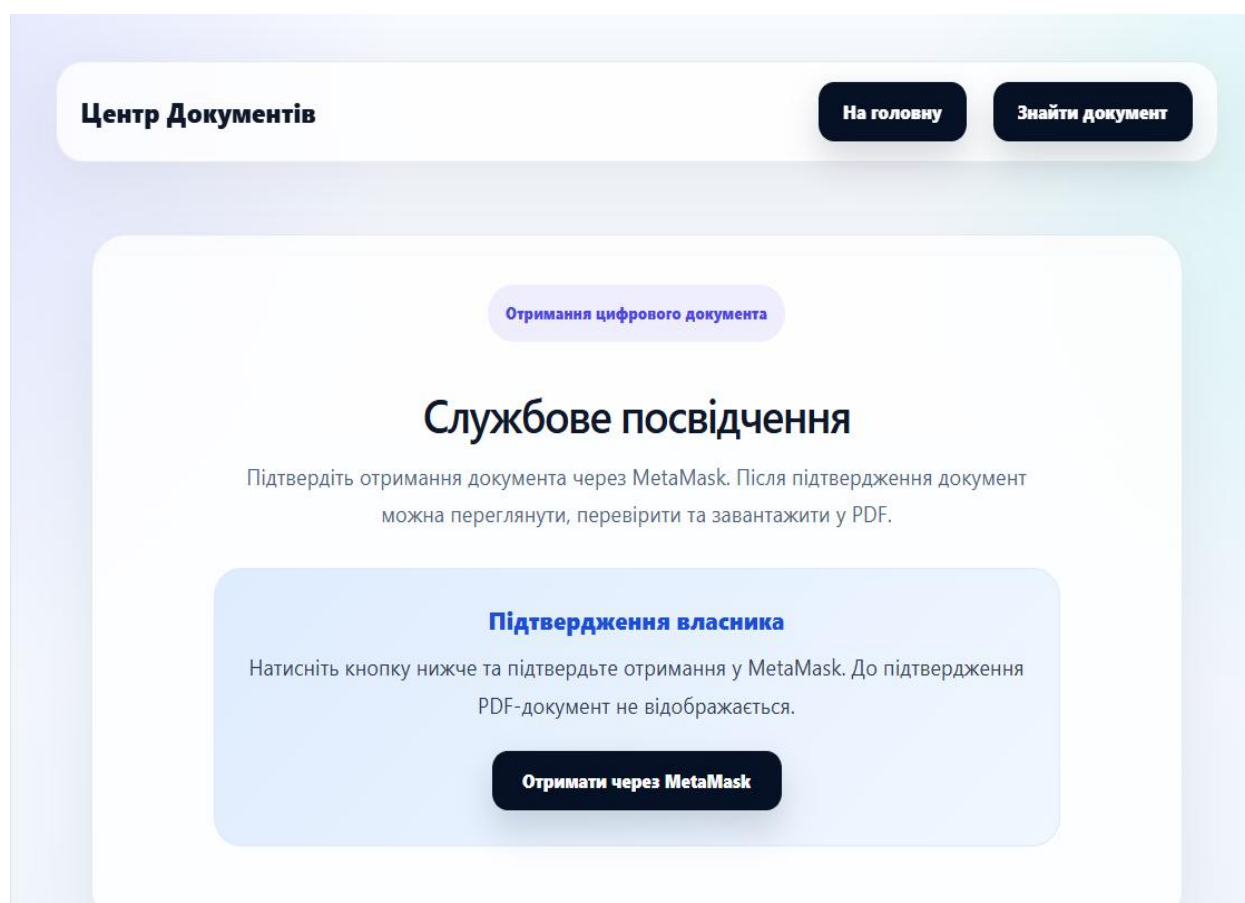


Рис. 3.8 Сторінка отримання документа через MetaMask

Після створення документа система виконує формування blockchain-запису та збереження контрольних даних документа. Даний підхід дозволяє забезпечити додатковий рівень захисту та підтвердити цілісність цифрового документа.

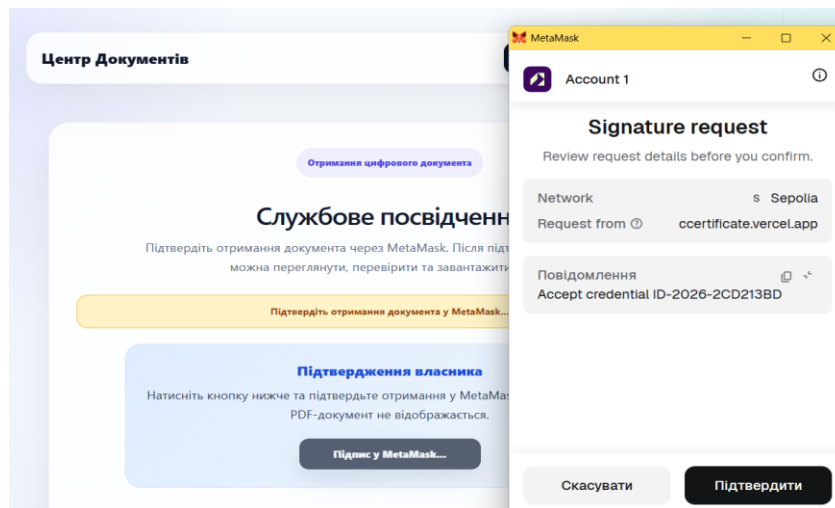


Рис. 3.9 Підтвердження документа через MetaMask

Під час розробки інтерфейсу особлива увага приділялась зручності використання системи. Для цього використовувались зрозумілі елементи навігації, форми введення даних та адаптивне розташування елементів сторінки.

Інтерфейс вебзастосунку підтримує роботу через сучасні веббраузери та дозволяє використовувати систему як на персональних комп'ютерах, так і на мобільних пристроях. Для оформлення сторінок використовувались CSS-стилі, які забезпечують єдиний стиль оформлення інтерфейсу та зручне відображення інформації.

Таким чином, реалізований користувацький інтерфейс забезпечує зручну взаємодію користувачів із системою та дозволяє виконувати основні операції з цифровими документами через веббраузер.

3.3 Інтерфейс адміністратора системи

Однією з основних частин вебзастосунку є адміністративна панель. Саме через адміністративний інтерфейс виконується створення, перегляд та анулювання цифрових документів.

Для доступу до адміністративної панелі використовується окрема сторінка авторизації адміністратора. Після успішного входу адміністратор отримує доступ до функцій керування цифровими документами.

Адміністративний доступ

Вхід до системи

Авторизуйтеся для створення, видачі та керування цифровими документами.

Логін

Пароль

Назад
Увійти

Рис. 3.10 Сторінка авторизації адміністратора

Основною сторінкою адміністративної панелі є сторінка створення документа. Через дану сторінку адміністратор може обрати тип документа та ввести необхідні дані користувача. Після заповнення форми система автоматично формує цифровий документ, генерує унікальний номер документа, QR-код для перевірки та створює криптографічний доказ справжності. Також адміністратор має можливість переглянути попередній вигляд документа перед його видачею користувачу. Усі створені документи зберігаються у базі даних системи та можуть бути перевірені через публічний verify-інтерфейс або QR-код.

Керування цифровими документами

Адміністратор працює не з окремими сертифікатами чи перепустками, а з єдиною системою документів. Тип документа обирається під час створення, а DID/VC, QR, видача, отримання, перевірка та анулювання працюють за спільною логікою.

+

Створити документ

Оберіть шаблон: сертифікат, перепустка або посвідчення. Після вибору відкриється форма саме цього типу документа.

Створити документ

□

Список документів

Перегляд усіх цифрових документів із фільтром за типом: усі, сертифікати, перепустки або посвідчення.

Відкрити список

🔍

Знайти документ

Пошук документа за номером або за даними власника. Працює для всіх типів документів.

Знайти документ

✓

Перевірити документ

Перевірка W3C VC, DID емітента, Holder DID, підпису, blockchain hash та статусу відкликання.

Перевірити документ

Рис. 3.11 Сторінка створення документа

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		72

Під час створення документа адміністратор заповнює форму, у якій вказуються основні дані документа, дані власника документа та додаткова інформація. Після підтвердження введених даних система автоматично створює цифровий документ.

Після створення документа система формує унікальний номер документа, генерує QR-код та створює запис у базі даних. Також система формує хеш документа та виконує запис контрольних даних у блокчейн Ethereum.

В адміністративній панелі реалізовано сторінку перегляду документів. На даній сторінці адміністратор може переглядати список створених документів, їх статуси та інформацію про власників документів.

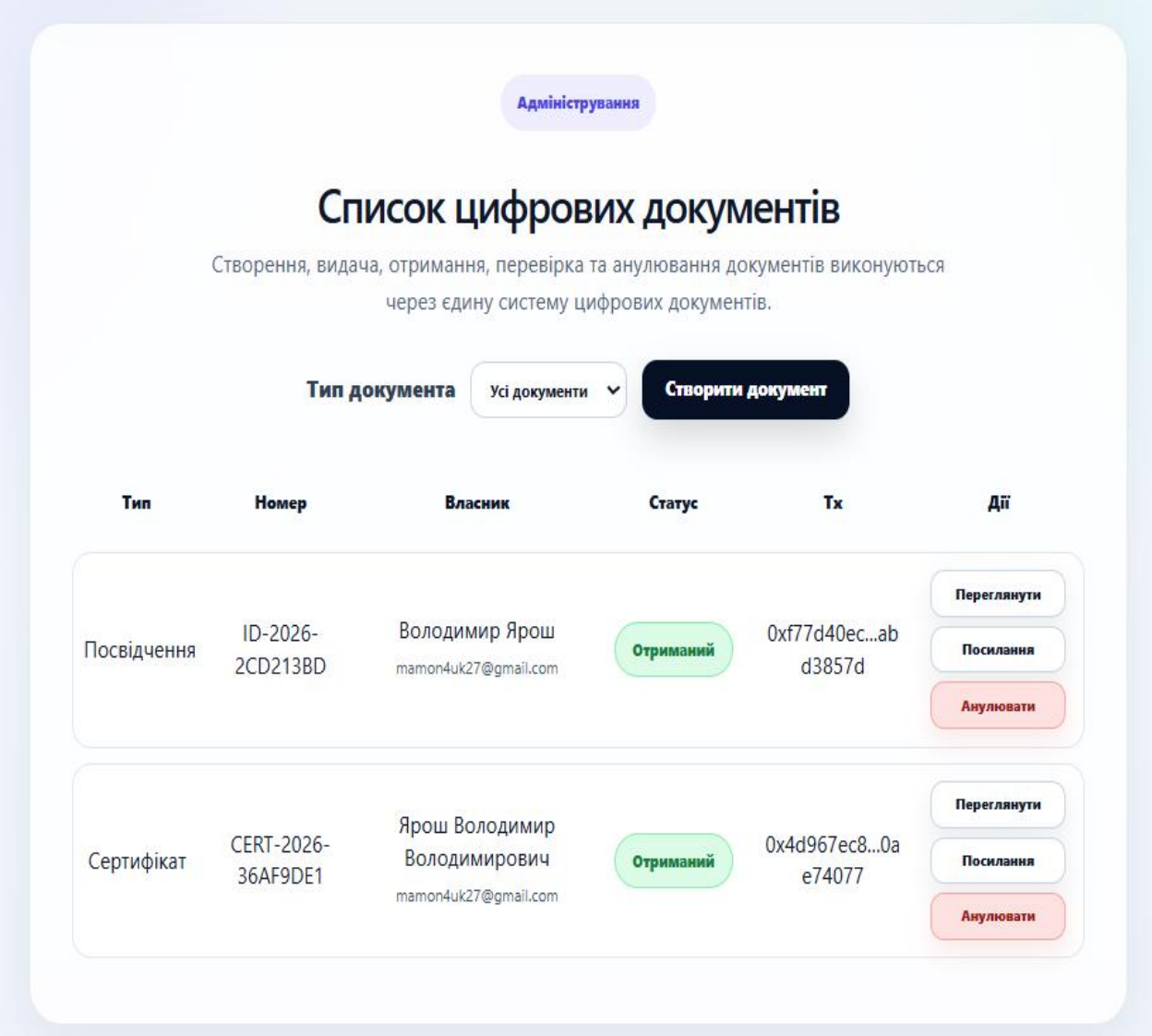


Рис. 3.12 Сторінка перегляду списку документів

Для спрощення роботи з документами у системі реалізовано пошук документів за номером документа або даними користувача. Це дозволяє швидко знаходити необхідний документ у системі.

Однією з важливих функцій адміністративної панелі є можливість анулювання цифрових документів. У випадку анулювання адміністратор може змінити статус документа та вказати причину анулювання.

АНУЛЮВАННЯ ДОКУМЕНТА

Підтвердіть відкликання цифрового документа

Після анулювання документ отримає статус "Анульований", стане недійсним під час перевірки та збереже причину відкликання в системі.

ТИП ДОКУМЕНТА

Посвідчення

НОМЕР ДОКУМЕНТА

ID-2026-2CD213BD

ВЛАСНИК

Володимир Ярош

Причина анулювання *

Наприклад: документ містить помилкові дані або втратив актуальність

Важливо

Анулювання може виконати лише гаманець видавця документа. Переконайтесь, що у MetaMask активний саме той обліковий запис, який має право відкликати документ. Після успішної транзакції QR-перевірка показуватиме документ як анульований.

Скасувати

Підтвердити анулювання

Рис. 3.13 Анулювання цифрового документу

Після анулювання інформація про зміну статусу документа зберігається у базі даних та відображається під час перевірки документа через вебінтерфейс. Одночасно у blockchain-мережі оновлюється ознака анулювання документа, що дозволяє підтвердити втрату його чинності незалежно від роботи серверної частини вебзастосунку. Завдяки цьому під час перевірки користувач отримує актуальну інформацію про статус документа, дату анулювання та причину втрати його чинності [10].

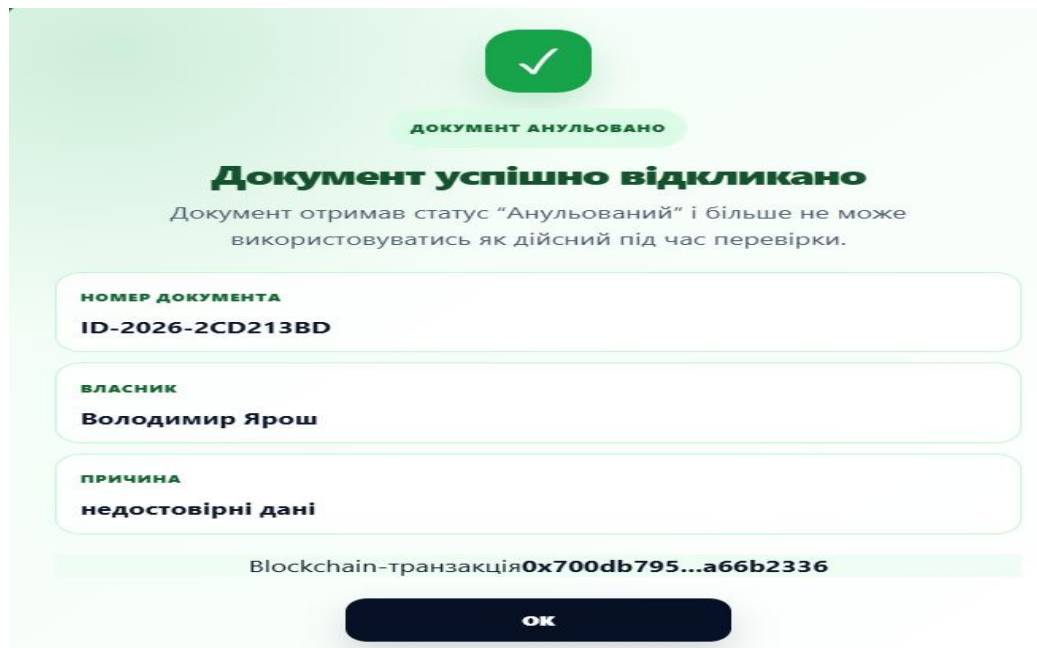


Рис. 3.14 Підтвердження анулювання документу

У системі також реалізовано сторінку перегляду детальної інформації про документ. Адміністратор може переглядати номер документа, статус, адресу криптогаманця користувача, інформацію про блокчейн-транзакцію та QR-код документа.

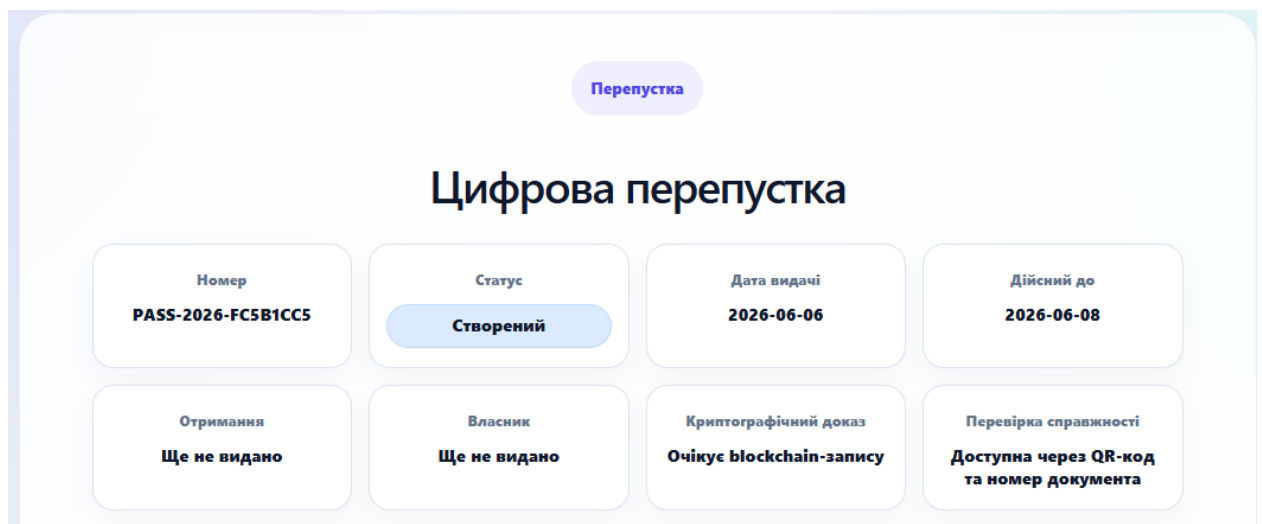


Рис. 3.15 Детальна інформація про документ

Сторінка verify-перевірки дозволяє користувачу отримати інформацію про статус цифрового документа та результат криптографічної перевірки. Перевірка може виконуватись як за номером документа, так і через QR-код. У результаті користувач бачить, чи є документ чинним, чи відповідає його хеш збереженим контрольним даним, а також чи не був документ анульований.

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

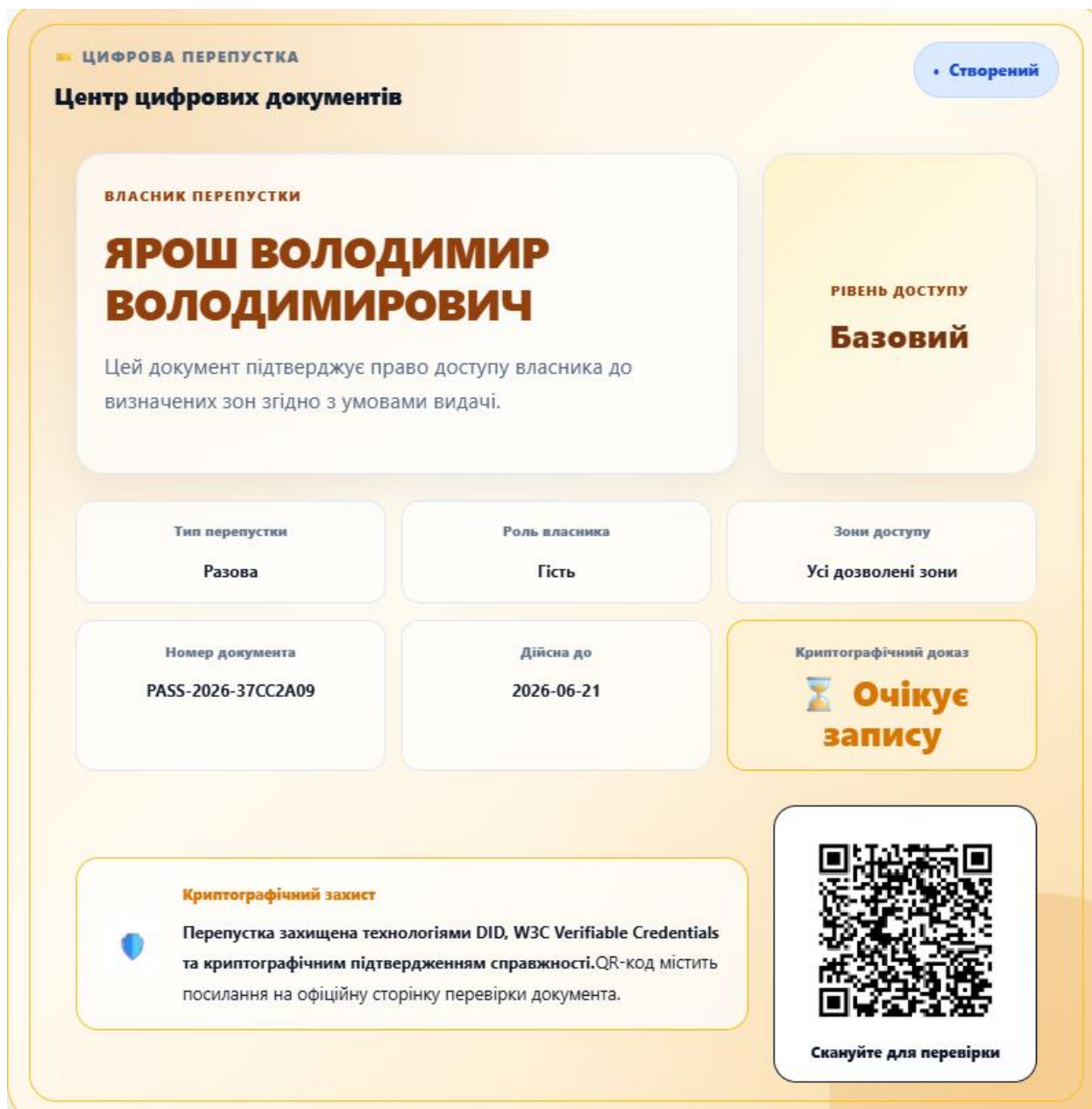


Рис. 3.16 Візуальне відображення документа

Для роботи з децентралізованими цифровими документами система використовує стандарти DID та VC. Під час створення документа система автоматично формує структуру цифрового документа та додає необхідні службові дані.

Під час розробки адміністративного інтерфейсу особлива увага приділялась зручності роботи адміністратора. Для цього використовувались зрозумілі форми введення даних, таблиці перегляду документів та елементи навігації між сторінками.

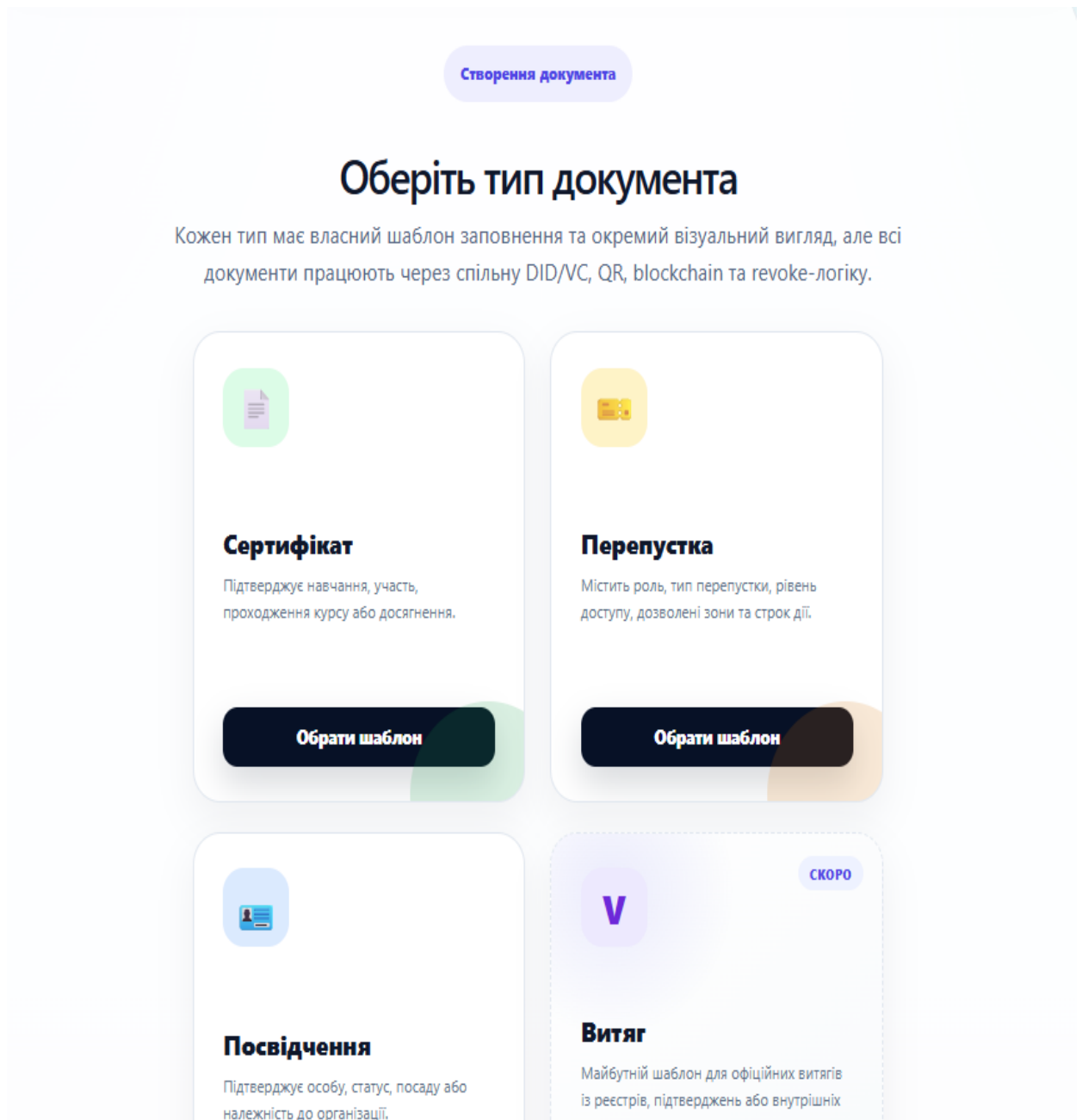


Рис. 3.17 Сторінка вибору типу документа

Таким чином, реалізований адміністративний інтерфейс забезпечує зручне керування цифровими документами та дозволяє виконувати основні операції адміністрування через веббраузер.

3.4 Особливості видачі, отримання та перевірки документів

Однією з основних функцій розроблюваного вебзастосунку є створення, видача, отримання та перевірка цифрових документів. Для реалізації даного функціоналу у системі було реалізовано механізми роботи з цифровими документами, QR-кодами, блокчейн-мережею та криптографічним підтвердженням права володіння документом.

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

Після створення цифрового документа адміністратор може видати документ користувачу. Під час видачі система формує спеціальне посилання для отримання документа. Дане посилання використовується користувачем для переходу на сторінку отримання документа.

Тип	Номер	Власник	Статус	Тх	Дії
Сертифікат	CERT-2026-7F01CC27	Ярош Володимир Володимирович mamon4uk27@gmail.com	Створений	—	<button>Переглянути</button> <button>Видати</button>

Рис. 3.18 Вигляд в адміністративній панелі створеного документа

Для підтвердження права володіння цифровим документом використовується криптографічний підпис через MetaMask. Після підтвердження система зберігає адресу криптогаманця користувача та оновлює статус документа.

Перепустка	PASS-2026-FC5B1CC5	Ярош Володимир Володимирович mamon4uk27@gmail.com	В очікуванні	0хаааае7са...fc980с65	<button>Переглянути</button> <button>Посилання</button> <button>Анулювати</button>
------------	--------------------	---	--------------	-----------------------	--

Рис. 3.19 Вигляд створеного та виданого документа (очікування підтвердження власником)

Після переходу за посиланням користувач може переглянути інформацію про цифровий документ та виконати підтвердження права володіння документом. Для цього використовується криптогаманець MetaMask.

ПОСИЛАННЯ ДЛЯ ОТРИМАННЯ	<button>Скопіювати</button>
https://ccertificate.vercel.app/receive/PASS-2026-FC5B1CC5?token=9618ef2a5bfdd0eed225b93fde693756	

Рис. 3.20 Вигляд посилання для отримання

Підтвердження права володіння документом виконується шляхом криптографічного підпису повідомлення через MetaMask. Після успішного підтвердження система зберігає адресу криптогаманця користувача та оновлює інформацію про документ.

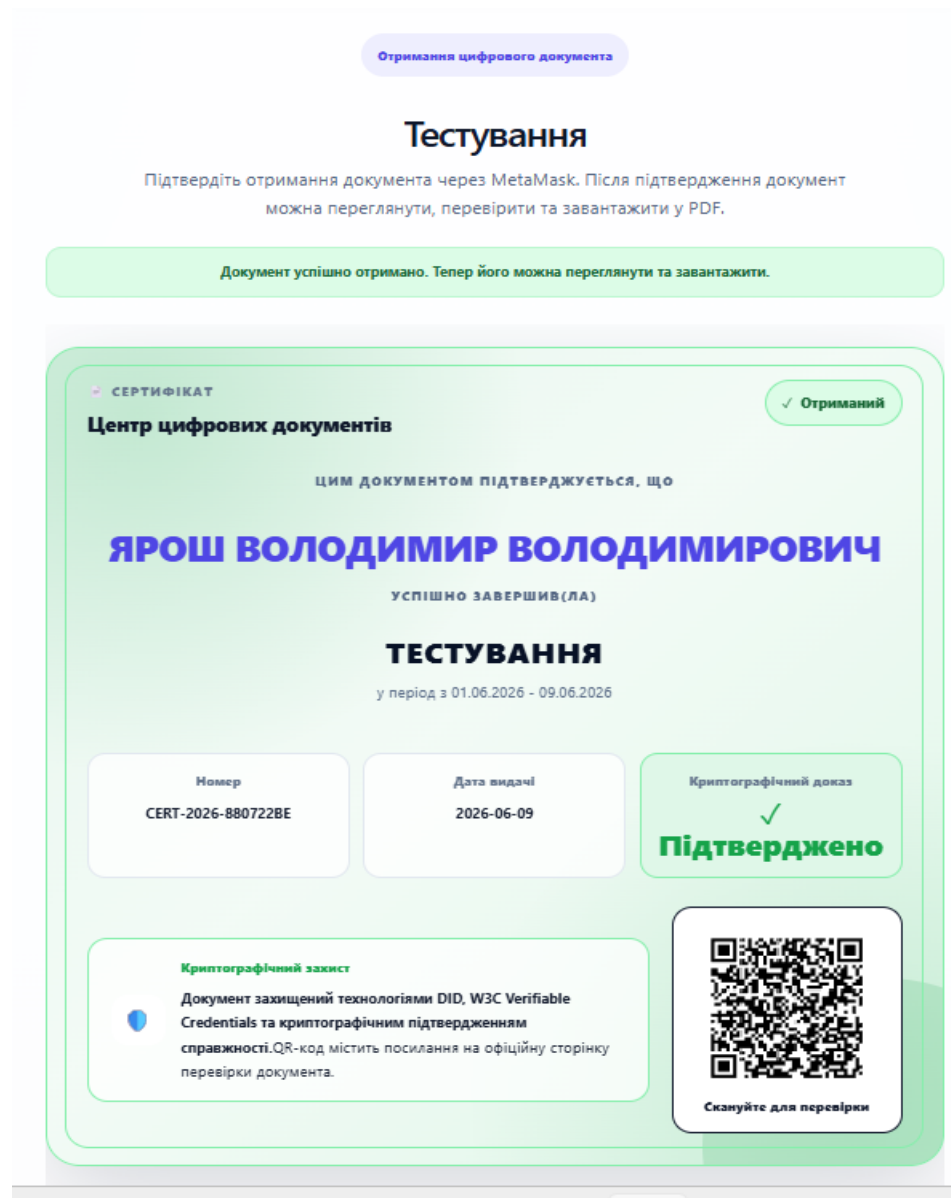


Рис. 3.21 Сторінка вигляду цифрового документа (після підтвердження право власності)

Однією з важливих функцій системи є перевірка достовірності цифрових документів. Для цього у вебзастосунку реалізовано окрему сторінку перевірки документа. Перевірка документа може виконуватись двома способами: за номером документа, за допомогою QR-коду.

Під час перевірки система виконує пошук документа у базі даних та перевіряє його статус. Додатково перевіряється інформація про документ, яка була збережена у блокчейн-мережі Ethereum. У випадку успішної перевірки система відображає інформацію про документ, його статус та підтвердження

					ІПЗ.КР.Б-121-26-ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		79

достовірності. Якщо документ був анульований або не існує у системі, користувач отримує відповідне повідомлення.

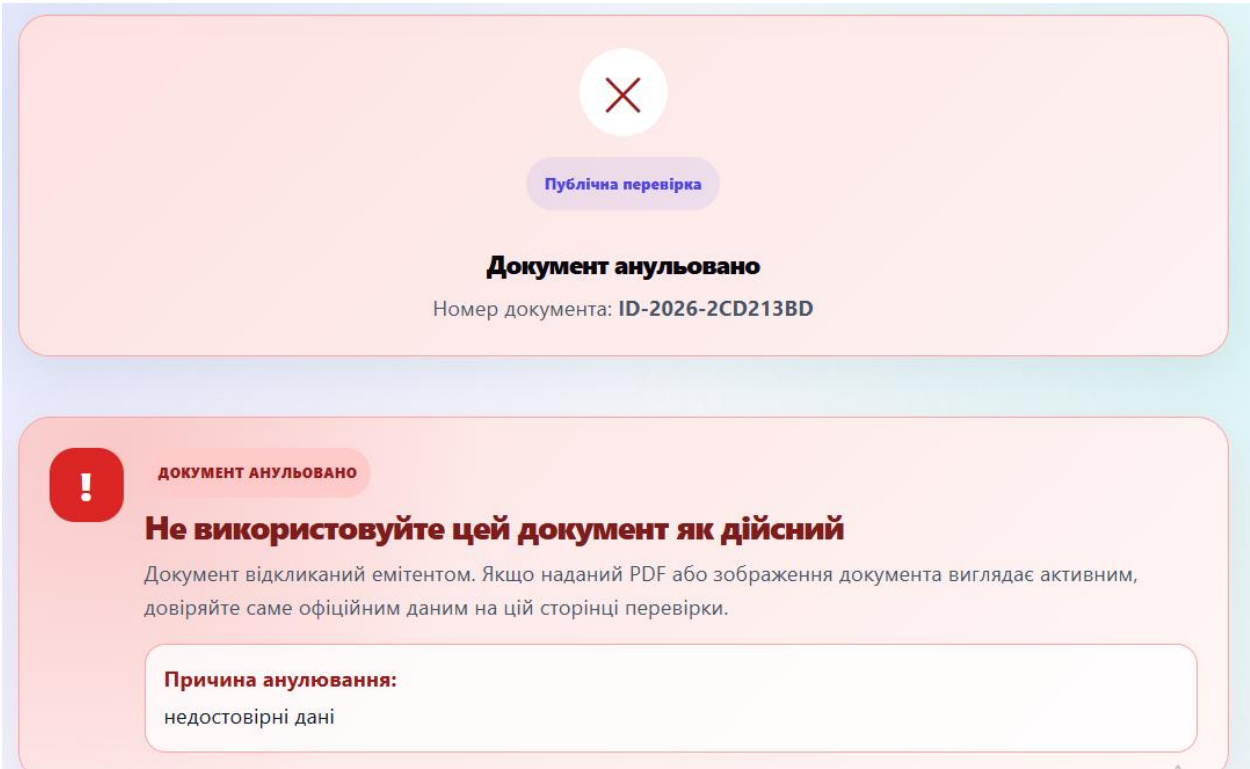


Рис. 3.22 Сторінка перевірки документа(анульованого) за номером

Для спрощення процесу перевірки кожен цифровий документ містить QR-код. Після сканування QR-коду користувач автоматично переходить на сторінку перевірки документа через браузер або мобільний пристрій.

У системі також реалізовано механізм анулювання документів. У випадку анулювання адміністратор може змінити статус документа та вказати причину анулювання. Після цього інформація про анулювання документа відображається під час перевірки.

Для забезпечення достовірності цифрових документів у системі використовуються технології DID та VC. Для кожного документа формується структура верифікованого цифрового документа, що містить інформацію про власника документа, видавця та криптографічне підтвердження достовірності.

Таким чином, реалізовані механізми видачі, отримання та перевірки документів забезпечують зручну роботу користувачів із системою та дозволяють підтверджувати достовірність цифрових документів через вебінтерфейс.

3.5 Тестування програмного комплексу

Після реалізації основного функціоналу вебзастосунку було проведено тестування програмного комплексу. Основною метою тестування була перевірка коректності роботи системи, стабільності роботи вебзастосунку та перевірка реалізації основних функцій.

Таблиця 3.2

Таблиця результатів функціоналу системи

№	Функція системи	Результат
1	Створення цифрового документа	Успішно
2	Генерація QR-коду	Успішно
3	Формування SHA-256 hash	Успішно
4	Запис даних у blockchain Ethereum	Успішно
5	Підпис документа через MetaMask	Успішно
6	Перевірка цифрового документа	Успішно
7	Перевірка QR-посилання	Успішно
8	Анулювання документа	Успішно
9	Перевірка revoked-документа	Успішно
10	Тестування REST API	Успішно
11	Виявлення зміни даних документа після видач	Успішно

Під час тестування перевірялась робота створення цифрових документів, видачі документів користувачам, підтвердження права володіння документом, перевірки достовірності документів та анулювання документів. Одним із перших етапів тестування була перевірка створення цифрових документів через адміністративну панель. Під час тестування адміністратор створював цифрові документи різних типів та перевіряв правильність збереження інформації у базі даних. У результаті тестування було встановлено, що система коректно формує номер документа, створює QR-код та зберігає службову інформацію про документ.

Наступним етапом тестування була перевірка механізму видачі документів. Для цього перевірялась робота посилань отримання документа та взаємодія з криптогаманцем MetaMask.

Для автоматизації перевірки функціоналу системи було створено окремі тестові сценарії Node.js. Автоматизовані тести перевіряли роботу REST API, формування цифрових документів, механізм підтвердження через MetaMask, перевірку статусів документів та взаємодію з blockchain-мережею Ethereum.

Під час тестування користувач переходив за посиланням отримання документа та виконував криптографічний підпис повідомлення через MetaMask. Після підтвердження система успішно зберігала адресу криптогаманця користувача та оновлювала статус документа.

Також було проведено тестування перевірки цифрових документів. Перевірка виконувалась за номером документа та через QR-код. У результаті тестування було встановлено, що система коректно відображає інформацію про документ, його статус та результат перевірки. У випадку відсутності документа або анулювання документа система відображала відповідне повідомлення.

Окрему увагу було приділено тестуванню механізму анулювання документів. Під час тестування адміністратор змінював статус документа та вказував причину анулювання. Після анулювання система коректно змінювала статус документа у базі даних та відображала інформацію про анулювання під час перевірки документа.

Під час тестування також перевірялась взаємодія системи з блокчейн-мережею Ethereum. Для цього виконувалась перевірка запису контрольних даних документа у блокчейн та перевірка інформації про блокчейн-транзакції. Окремо перевірялось, чи відповідає хеш, сформований на основі даних цифрового документа, контрольному запису, зафіксованому в мережі Ethereum, а також чи коректно відображається інформація про транзакцію в інтерфейсі системи.

Результати тестування показали коректну роботу механізму взаємодії з блокчейн-мережею та правильне збереження контрольних даних документів.

```
PS C:\Users\user\desktop\навчання\didocumentssystem\backend> npm test

> backend@1.0.0 test
> node --test tests/*.test.js

✓Full API flow: create, issue, accept, verify (8565.6182ms)
✓Backend exposes blockchain configuration status (164.7115ms)
✓Issue document saves blockchain transaction hash (94.0035ms)
✓Verification response contains blockchain check object (8785.426ms)
✓Accept document with valid wallet signature (251.523ms)
✓Reject document accept with wrong wallet signature (62.7304ms)
✓Revoke document changes status to revoked (322.4092ms)
✓Verify revoked document returns red state (8718.3648ms)
✓Verify not existing document returns 404 (167.8383ms)
✓Health endpoint returns backend status (90.2566ms)
i tests 10
i suites 0
i pass 10
i fail 0
i cancelled 0
i skipped 0
i todo 0
i duration_ms 9249.9505
```

Рис. 3.23 Результату запуску автоматизованих тестів

Після завершення процесу видачі було виконано ручну зміну даних документа безпосередньо у базі даних SQLite. Зокрема було змінено прізвище власника документа та адресу електронної пошти. Під час повторної перевірки документа система повторно сформувала контрольний hash на основі поточних даних документа та виконала порівняння з hash, який був записаний у блокчейн під час створення документа.

У результаті перевірки система успішно виявила невідповідність між поточними даними документа та інформацією, що була зафіксована у блокчейн-мережі. Крім загального повідомлення про невідповідність даних, система також відобразила конкретні поля документа, які були змінені після видачі документа.

Результат тестування підтвердив коректну роботу механізму контролю цілісності цифрових документів та можливість виявлення несанкціонованої зміни даних після створення документа. Це дозволило перевірити, чи здатна система виявляти зміну важливих реквізитів документа після його видачі та порівнювати поточні дані з контрольними даними, зафіксованими у блокчейні.



Публічна перевірка

Документ потребує додаткової перевірки

Номер документа: CERT-2026-83606282



ВИЯВЛЕНО НЕВІДПОВІДНІСТЬ ДАНИХ

ПІБ власника

БУЛО

Володимир Ярош

ЗАРАЗ

Мамкін хакер

Email власника

БУЛО

mamon4uk27@gmail.com

ЗАРАЗ

test@example.com

Рис. 3.24 Сторінка верифікації документа

Крім того, було проведено тестування роботи системи через мережу Інтернет після розгортання вебзастосунку на платформах Render та Vercel. У результаті тестування було підтверджено коректну роботу клієнтської та серверної частин вебзастосунку, а також можливість використання системи через браузер.

Таким чином, проведене тестування підтвердило працездатність програмного комплексу та коректну роботу основних функцій вебзастосунку для адміністрування децентралізованих систем ідентифікації.

Висновки до третього розділу.

У третьому розділі було розглянуто особливості інтерфейсу користувача та адміністратора системи, а також порядок роботи із вебзастосунком для адміністрування цифрових документів.

Було проведено тестування основних функцій програмного комплексу та перевірено коректність роботи механізмів створення, видачі, перевірки та анулювання цифрових документів. Результати тестування підтвердили працездатність та можливість практичного використання розробленої системи.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено вебзастосунок для адміністрування децентралізованих систем ідентифікації DID, призначений для створення, видачі, отримання, перевірки та анулювання цифрових документів. Розроблена система поєднує клієнт-серверну архітектуру з елементами децентралізованої перевірки, що дозволяє забезпечити зручне адміністрування документів та підвищити рівень довіри до результатів їх перевірки.

У процесі виконання роботи було проведено аналіз сучасних підходів до цифрової ідентифікації, досліджено принципи роботи децентралізованих ідентифікаторів, верифікованих цифрових документів та blockchain-технологій. Також було проаналізовано існуючі програмні рішення у сфері цифрової ідентифікації, зокрема системи, що підтримують DID, VC та перевірку цифрових облікових даних. На основі проведеного аналізу було визначено основні вимоги до розроблюваного вебзастосунку та обґрунтовано доцільність використання Ethereum, MetaMask, DID і Verifiable Credential.

У межах проєктування було визначено основні ролі користувачів системи: адміністратора, власника документа та перевіряючої сторони. Для кожної ролі було визначено функціональні можливості та сценарії роботи. Адміністратор виконує створення, видачу та анулювання документів, власник документа підтверджує право володіння документом за допомогою MetaMask, а перевіряюча сторона може перевірити справжність документа за QR-кодом або номером без доступу до адміністративної панелі.

Під час виконання роботи було спроектовано архітектуру вебзастосунку, структуру бази даних та алгоритми роботи системи. У базі даних SQLite зберігаються повні службові дані цифрових документів, зокрема дані власника, статус документа, VC JSON, proof, MetaMask-підпис, QR-посилання та інформація про анулювання. У blockchain-мережу Ethereum передаються лише контрольні дані, зокрема номер документа, SHA-256 hash, адреса видавця, адреса отримувача та ознака анулювання. Такий розподіл дозволяє

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

не розміщувати персональні дані у публічному blockchain та водночас використовувати його для незалежної перевірки контрольних доказів.

У практичній частині роботи було реалізовано основний функціонал вебзастосунку. Система забезпечує створення цифрових документів різних типів, формування структури Verifiable Credential, генерацію QR-коду, створення SHA-256 hash документа, запис контрольних даних у blockchain-мережу Ethereum, підтвердження володіння документом через MetaMask, перевірку достовірності документа та його анулювання. Під час отримання документа користувач підписує службове повідомлення, а не сам файл документа, що дозволяє підтвердити зв'язок між документом та Ethereum-адресою власника без передачі приватного ключа на сервер.

Реалізований механізм перевірки документа дозволяє перевіряти не лише наявність запису у базі даних, а й цілісність документа, його статус, blockchain-запис, DID видавця, proof та MetaMask-підпис власника. Під час перевірки система повторно обчислює SHA-256 hash документа та порівнює його зі значенням, яке було збережено у базі даних і записано у blockchain. Якщо документ був змінений після видачі, система виявляє невідповідність і повідомляє про необхідність додаткової перевірки.

Окремо було реалізовано механізм анулювання цифрового документа. Під час анулювання змінюється статус документа у базі даних, зберігається причина анулювання та оновлюється ознака revoked у smart contract Ethereum. Завдяки цьому під час перевірки документа користувач отримує актуальну інформацію про його чинність, а факт анулювання може бути підтверджений не лише через базу даних вебзастосунку, а й через blockchain-запис.

У третьому розділі було описано порядок розгортання та налаштування вебзастосунку. Клієнтська частина розгортається на платформі Vercel, серверна частина — на платформі Render, а вихідний код зберігається у репозиторії GitHub. Для роботи із blockchain використовується Ethereum Sepolia, RPC-провайдер Infura та криптогаманець MetaMask. Оновлення

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

вебзастосунку автоматизовано через GitHub, Vercel та Render, що спрощує процес публікації нових версій програмного забезпечення.

Під час тестування було перевірено роботу основних компонентів системи: створення цифрового документа, генерацію QR-коду, формування SHA-256 hash, запис контрольних даних у blockchain, підтвердження документа через MetaMask, перевірку документа, перевірку QR-посилання, анулювання документа та виявлення зміни даних після видачі. Результати тестування підтвердили коректну роботу програмного комплексу та можливість його практичного використання.

Практичне значення роботи полягає у створенні програмного засобу, який може використовуватися як основа для систем адміністрування цифрових документів у навчальних закладах, організаціях, підприємствах або інших установах, де необхідно забезпечити створення, видачу та перевірку електронних документів. Розроблена система демонструє можливість поєднання традиційного вебзастосунку з децентралізованими механізмами перевірки, що дозволяє підвищити рівень довіри до цифрових документів.

Розроблений вебзастосунок може бути розширений у майбутньому шляхом додавання нових типів цифрових документів, підтримки інших blockchain-мереж, інтеграції з апаратними криптографічними носіями, реалізації окремого verifier-модуля для незалежної перевірки документів без використання основного вебзастосунку, а також удосконалення механізмів керування DID та Verifiable Credential.

Отже, поставлену мету кваліфікаційної роботи досягнуто. Було розроблено вебзастосунок, який забезпечує повний життєвий цикл цифрового документа та використовує сучасні технології децентралізованої ідентифікації, blockchain-перевірки та криптографічного підтвердження володіння документом.

ЛІТЕРАТУРА

1. Berners-Lee T., Fielding R., Masinter L. Uniform Resource Identifier (URI): Generic Syntax : RFC 3986. IETF, 2005. URL: <https://www.rfc-editor.org/rfc/rfc3986> (дата звернення: 12.02.2026).
2. Sporny M., Longley D., Chadwick D. Decentralized Identifiers (DIDs) v1.0. W3C Recommendation. 2022. URL: <https://www.w3.org/TR/did-core/> (дата звернення: 18.02.2026).
3. Sporny M., Longley D., Sabadello M. Verifiable Credentials Data Model v1.1. W3C Recommendation. 2022. URL: <https://www.w3.org/TR/vc-data-model/> (дата звернення: 21.02.2026).
4. Buterin V. Ethereum Whitepaper. 2014. URL: <https://ethereum.org/en/whitepaper/> (дата звернення: 04.03.2026).
5. Wood G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Yellow Paper. 2014. URL: <https://ethereum.github.io/yellowpaper/paper.pdf> (дата звернення: 06.03.2026).
6. Antonopoulos A., Wood G. Mastering Ethereum: Building Smart Contracts and DApps. Sebastopol : O'Reilly Media, 2018. 414 p.
7. Bashir I. Mastering Blockchain. 3rd ed. Birmingham : Packt Publishing, 2020. 816 p.
8. Narayanan A., Bonneau J., Felten E., Miller A., Goldfeder S. Bitcoin and Cryptocurrency Technologies. Princeton : Princeton University Press, 2016. 336 p.
9. Tapscott D., Tapscott A. Blockchain Revolution: How the Technology Behind Bitcoin and Other Cryptocurrencies Is Changing the World. New York : Portfolio, 2018. 320 p.
10. Freeman A. Pro React 18. New York : Apress, 2023. 950 p.
11. Banks A., Porcello E. Learning React. 2nd ed. Sebastopol : O'Reilly Media, 2020. 310 p.
12. Haverbeke M. Eloquent JavaScript. 3rd ed. San Francisco : No Starch Press, 2018. 472 p.

13. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs // IEEE Internet Computing. 2010. Vol. 14, No. 6. P. 80–83.
14. Holmes S. Getting MEAN with Mongo, Express, Angular and Node. 2nd ed. Shelter Island : Manning Publications, 2019. 440 p.
15. Richardson L., Ruby S. RESTful Web Services. Sebastopol : O'Reilly Media, 2007. 448 p.
16. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.
17. SQLite Documentation. URL: <https://www.sqlite.org> (дата звернення: 14.03.2026).
18. React Documentation. URL: <https://react.dev> (дата звернення: 19.03.2026).
19. Node.js Documentation. URL: <https://nodejs.org> (дата звернення: 23.03.2026).
20. Express.js Documentation. URL: <https://expressjs.com> (дата звернення: 28.03.2026).
21. Ethereum Documentation. URL: <https://ethereum.org> (дата звернення: 03.04.2026).
22. MetaMask Documentation. URL: <https://docs.metamask.io> (дата звернення: 09.04.2026).
23. W3C DID Core Specification. URL: <https://www.w3.org/TR/did-core/> (дата звернення: 15.04.2026).
24. W3C Verifiable Credentials Data Model. URL: <https://www.w3.org/TR/vc-data-model/> (дата звернення: 21.04.2026).
25. Mozilla Developer Network (MDN) Web Docs. URL: <https://developer.mozilla.org> (дата звернення: 26.04.2026).
26. Render Documentation. URL: <https://render.com/docs> (дата звернення: 02.05.2026).

27. Vercel Documentation. URL: <https://vercel.com/docs> (дата звернення: 06.05.2026).

28. Sepolia Testnet Documentation. URL: <https://sepolia.dev> (дата звернення: 11.05.2026).

29. Resend Documentation. URL: <https://resend.com/docs> (дата звернення: 16.05.2026).

30. OpenAPI Specification. URL: <https://swagger.io/specification/> (дата звернення: 20.05.2026).

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг смарт-контракту для фіксації та перевірки цифрових документів

```

pragma solidity ^0.8.20;

contract CertificateContract {
    address public owner;

    struct CertificateRecord {
        string certificateNumber;
        string certificateHash;
        address issuer;
        address recipient;
        uint256 issuedAt;
        uint256 claimedAt;
        uint256 revokedAt;
        bool exists;
        bool claimed;
        bool revoked;
        string revokeReason;
    }

    mapping(string => CertificateRecord) private certificates;

    event CertificateStored(
        string certificateNumber,
        string certificateHash,
        address issuer,
        uint256 issuedAt
    );

    event CertificateClaimed(
        string certificateNumber,
        address recipient,
        uint256 claimedAt
    );

    event CertificateRevoked(
        string certificateNumber,
        address issuer,
        uint256 revokedAt,
        string reason
    );

    constructor() {
        owner = msg.sender;
    }

    function storeCertificate(
        string memory certificateNumber,
        string memory certificateHash
    ) public {
        require(!certificates[certificateNumber].exists, "Certificate already exists");

        certificates[certificateNumber] = CertificateRecord({
            certificateNumber: certificateNumber,
            certificateHash: certificateHash,
            issuer: msg.sender,
            recipient: address(0),
            issuedAt: block.timestamp,
            claimedAt: 0,
            revokedAt: 0,
        });
    }
}

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        exists: true,
        claimed: false,
        revoked: false,
        revokeReason: ""
    });

    emit CertificateStored(
        certificateNumber,
        certificateHash,
        msg.sender,
        block.timestamp
    );
}

function claimCertificate(string memory certificateNumber) public {
    require(certificates[certificateNumber].exists, "Certificate does not exist");
    require(!certificates[certificateNumber].revoked, "Certificate is revoked");
    require(!certificates[certificateNumber].claimed, "Certificate already claimed");

    certificates[certificateNumber].recipient = msg.sender;
    certificates[certificateNumber].claimedAt = block.timestamp;
    certificates[certificateNumber].claimed = true;

    emit CertificateClaimed(certificateNumber, msg.sender, block.timestamp);
}

function revokeCertificate(
    string memory certificateNumber,
    string memory reason
) public {
    require(certificates[certificateNumber].exists, "Certificate does not exist");
    require(!certificates[certificateNumber].revoked, "Certificate already revoked");
    require(
        msg.sender == certificates[certificateNumber].issuer || msg.sender == owner,
        "Only issuer can revoke"
    );

    certificates[certificateNumber].revoked = true;
    certificates[certificateNumber].revokedAt = block.timestamp;
    certificates[certificateNumber].revokeReason = reason;

    emit CertificateRevoked(
        certificateNumber,
        msg.sender,
        block.timestamp,
        reason
    );
}

function verifyCertificate(
    string memory certificateNumber,
    string memory certificateHash
) public view returns (bool) {
    if (!certificates[certificateNumber].exists) {
        return false;
    }
}

```

```

        if (certificates[certificateNumber].revoked) {
            return false;
        }

        return (
            keccak256(abi.encodePacked(certificates[certificateNumber].certificateHash))
            ==
            keccak256(abi.encodePacked(certificateHash))
        );
    }
}

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						94
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг структури бази даних цифрових документів

```

const Database = require("better-sqlite3");

const db = new Database("./database.db");

db.exec(`
  CREATE TABLE IF NOT EXISTS documents (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    documentType TEXT NOT NULL,
    documentNumber TEXT UNIQUE NOT NULL,
    fullName TEXT NOT NULL,
    email TEXT NOT NULL,
    title TEXT NOT NULL,
    issueDate TEXT NOT NULL,
    validUntil TEXT,
    description TEXT,
    issuer TEXT NOT NULL,
    status TEXT NOT NULL DEFAULT 'draft',
    credentialStatus TEXT DEFAULT 'draft',
    documentHash TEXT,
    claimToken TEXT,
    claimUrl TEXT,
    walletAddress TEXT,
    holderDid TEXT,
    holderSignature TEXT,
    blockchainTxHash TEXT,
    blockchainStatus TEXT DEFAULT 'not_published',
    verificationCount INTEGER DEFAULT 0,
    lastVerifiedAt TEXT,
    revokedAt TEXT,
    revokeReason TEXT,
    revokeTxHash TEXT,
    issuerDid TEXT,
    vcJson TEXT,
    proofJson TEXT,
    vpJson TEXT,
    vpProofJson TEXT,
    documentData TEXT,
    createdAt TEXT NOT NULL,
    updatedAt TEXT
  )
`);

db.exec(`
  CREATE TABLE IF NOT EXISTS document_verifications (
    documentNumber TEXT PRIMARY KEY,
    verificationCount INTEGER DEFAULT 0,
    lastVerifiedAt TEXT
  )
`);

module.exports = db;

```

Лістинг формування W3C Verifiable Credential

```

function buildVerifiableDocumentCredential(document) {
  const data = typeof document.documentData === "string"
    ? parseJsonField(document.documentData) || {}
    : document.documentData || {};

  const config = getDocumentTypeConfig(document.documentType);

  const vc = {
    "@context": ["https://www.w3.org/2018/credentials/v1"],
    id: `urn:uuid:${document.documentNumber}`,
    type: ["VerifiableCredential", config?.vcType || "DigitalDocument"],
    issuer: {
      id: ISSUER_DID,
      name: document.issuer
    },
    issuanceDate: document.issueDate,
    credentialSubject: {
      id: document.holderDid || "",
      documentType: document.documentType,
      documentNumber: document.documentNumber,
      fullName: document.fullName,
      email: document.email,
      title: document.title,
      description: document.description || "",
      ...data
    },
    credentialStatus: {
      id: `${BACKEND_URL}/api/documents/status/${document.documentNumber}`,
      type: "CredentialStatusList2021"
    }
  };

  if (document.validUntil) {
    vc.expirationDate = document.validUntil;
  }

  return vc;
}

```

					ІІЗ.КР.Б-121-26-ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		96

Лістинг генерації криптографічного доказу документа

```

function buildIssuerProof(payload) {
  if (!ISSUER_PRIVATE_KEY) {
    return {
      type: "DevelopmentProof",
      created: new Date().toISOString(),
      verificationMethod: DID_VERIFICATION_METHOD,
      proofPurpose: "assertionMethod",
      jws: crypto
        .createHash("sha256")
        .update(stableStringify(payload))
        .digest("hex")
    };
  }

  const signer = crypto.createSign("SHA256");
  signer.update(stableStringify(payload));
  signer.end();

  return {
    type: "EcdsaSecp256r1Signature2019",
    created: new Date().toISOString(),
    verificationMethod: DID_VERIFICATION_METHOD,
    proofPurpose: "assertionMethod",
    jws: signer.sign(ISSUER_PRIVATE_KEY, "base64")
  };
}

function verifyIssuerProof(payload, proof) {
  if (!payload || !proof?.jws) return false;
  if (!ISSUER_PUBLIC_KEY || proof.type !== "DevelopmentProof") return true;

  try {
    const verifier = crypto.createVerify("SHA256");
    verifier.update(stableStringify(payload));
    verifier.end();

    return verifier.verify(ISSUER_PUBLIC_KEY, proof.jws, "base64");
  } catch {
    return false;
  }
}

```

Лістинг перевірки документа у блокчейн-мережі

```

async function verifyOnBlockchain(documentNumber, documentHash) {
  if (!isBlockchainConfigured() || !documentHash) {
    return {
      checked: false,
      exists: false,
      hashValid: false,
      revoked: false,
      message: "Blockchain is not configured or document hash is missing"
    };
  }

  try {
    const contract = getContract();
    const chainDocument = await contract.getCertificate(documentNumber);

    const exists = Boolean(chainDocument[6]);

    if (!exists) {
      return {
        checked: true,
        exists: false,
        hashValid: false,
        revoked: false,
        message: "Document was not found in blockchain"
      };
    }

    const hashValid = await contract.verifyCertificate(
      documentNumber,
      documentHash
    );

    return {
      checked: true,
      exists,
      hashValid: Boolean(hashValid),
      revoked: Boolean(chainDocument[8]),
      storedHash: chainDocument[1],
      issuer: chainDocument[2],
      recipient: chainDocument[3],
      issuedAt: Number(chainDocument[4]),
      claimedAt: Number(chainDocument[5]),
      revokedAt: Number(chainDocument[9]),
      revokeReason: chainDocument[10],
      message: Boolean(hashValid)
        ? "Blockchain hash is valid"
        : "Blockchain hash is invalid"
    };
  } catch (error) {
    return {
      checked: false,
      exists: false,
      hashValid: false,
      revoked: false,
      message: error.message
    };
  }
}

```

Лістинг отримання документа власником через MetaMask

```

const acceptDocument = async () => {
  try {
    if (!window.ethereum) {
      setTypedMessage("MetaMask не знайдено", "error");
      return;
    }

    if (isRevoked) {
      setTypedMessage("Анульований документ неможливо отримати", "error");
      return;
    }

    setIsLoading(true);
    setTypedMessage("Підтвердіть отримання документа у MetaMask...", "info");

    const accounts = await window.ethereum.request({
      method: "eth_requestAccounts"
    });

    const walletAddress = accounts[0];

    const provider = new ethers.BrowserProvider(window.ethereum);
    const signer = await provider.getSigner();

    const acceptMessage = `Accept credential ${documentNumber}`;
    const signature = await signer.signMessage(acceptMessage);

    const response = await fetch(
      `${API_URL}/api/documents/${documentNumber}/accept`,
      {
        method: "PATCH",
        headers: {
          "Content-Type": "application/json"
        },
        body: JSON.stringify({
          token,
          walletAddress,
          signature
        })
      }
    );

    const data = await response.json();

    if (!response.ok || !data.success) {
      setTypedMessage(data.message || "Не вдалося отримати документ", "error");
      return;
    }

    setDocument({
      ...document,
      ...(data.document || {}),
      status: data.document?.status || "claimed",
      walletAddress: data.document?.walletAddress || walletAddress,
      holderSignature: data.document?.holderSignature || signature
    });

    setOwnerVerified(true);
    setAcceptedLocally(true);
  }
}

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						99
Змн.	Арк.	№ докум.	Підпис	Дата		

```
        setTypedMessage(  
            "Документ успішно отримано. Тепер його можна переглянути та  
завантажити.",  
            "success"  
        );  
    } catch (error) {  
        setTypedMessage(error?.message || "Помилка отримання документа", "error");  
    } finally {  
        setIsLoading(false);  
    }  
};
```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						100
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг серверної обробки підтвердження документа власником

```

app.patch("/api/documents/:documentNumber/accept", (req, res) => {
  try {
    const { documentNumber } = req.params;
    const { token, walletAddress, signature } = req.body;

    if (!walletAddress || !signature) {
      return res.status(400).json({
        success: false,
        message: "Wallet address and signature are required"
      });
    }

    const document = getDocumentByNumber(documentNumber);

    if (!document) {
      return res.status(404).json({
        success: false,
        message: "Документ не знайдено"
      });
    }

    if (document.claimToken !== token) {
      return res.status(403).json({
        success: false,
        message: "Недійсне посилання або токен отримання"
      });
    }

    if (document.status === "revoked" || document.revokedAt) {
      return res.status(400).json({
        success: false,
        message: "Анульований документ неможливо отримати"
      });
    }

    const acceptMessage = `Accept credential ${documentNumber}`;
    const recovered = ethers.verifyMessage(acceptMessage, signature);

    if (recovered.toLowerCase() !== walletAddress.toLowerCase()) {
      return res.status(403).json({
        success: false,
        message: "Недійсний підпис гаманця"
      });
    }

    const normalizedWallet = ethers.getAddress(walletAddress);
    const holderDid = `did:pkh:eip155:11155111:${normalizedWallet}`;

    const finalDocument = {
      ...document,
      walletAddress: normalizedWallet,
      holderDid,
      holderSignature: signature,
      status: "accepted",
      credentialStatus: "active"
    };

    const vcJson = buildVerifiableDocumentCredential(finalDocument);
    const proofJson = buildIssuerProof(vcJson);
  }
});

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const vpJson = {
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  type: ["VerifiablePresentation"],
  holder: holderDid,
  verifiableCredential: [{ ...vcJson, proof: proofJson }]
};

const vpProofJson = {
  type: "EthereumPersonalSignature2021",
  created: new Date().toISOString(),
  proofPurpose: "authentication",
  verificationMethod: `${holderDid}#ethereum-address`,
  challenge: acceptMessage,
  jws: signature
};

db.prepare(`
  UPDATE documents SET
    status = ?, credentialStatus = ?, walletAddress = ?, holderDid = ?,
    holderSignature = ?, vcJson = ?, proofJson = ?, vpJson = ?,
    vpProofJson = ?, updatedAt = ?
  WHERE documentNumber = ?
`).run(
  "accepted",
  "active",
  normalizedWallet,
  holderDid,
  signature,
  JSON.stringify(vcJson),
  JSON.stringify(proofJson),
  JSON.stringify(vpJson),
  JSON.stringify(vpProofJson),
  new Date().toISOString(),
  documentNumber
);

return res.json({
  success: true,
  message: "Document accepted with Holder DID",
  document: normalizeDocument(getDocumentByNumber(documentNumber))
});
} catch (error) {
  return res.status(500).json({
    success: false,
    message: "Document accept error",
    error: error.message
  });
}
});

```

Лістинг перевірки цифрового документа

```

async function verifyDocumentHandler(req, res) {
  try {
    const { documentNumber } = req.params;
    const document = getDocumentByNumber(documentNumber);

    if (!document) {
      return res.status(404).json({
        success: false,
        valid: false,
        message: "Документ не знайдено."
      });
    }

    const normalized = normalizeDocument(document);
    const vcJson = normalized.vcJson;
    const proofJson = normalized.proofJson;

    const currentDocumentHash = buildDocumentHash(document);
    const storedDocumentHash = document.documentHash;
    const localHashValid = currentDocumentHash === storedDocumentHash;

    const blockchain = await verifyOnBlockchain(
      document.documentNumber,
      currentDocumentHash
    );

    const proofValid = verifyIssuerProof(vcJson, proofJson);

    const issuerDidValid =
      vcJson?.issuer?.id === ISSUER_DID &&
      proofJson?.verificationMethod === DID_VERIFICATION_METHOD;

    const credentialSubjectValid =
      vcJson?.credentialSubject?.documentNumber === document.documentNumber;

    const holderDidExists = Boolean(document.holderDid);
    const holderSignatureExists = Boolean(document.holderSignature);

    let holderSignatureValid = false;

    if (document.walletAddress && document.holderSignature) {
      try {
        const recovered = ethers.verifyMessage(
          `Accept credential ${document.documentNumber}`,
          document.holderSignature
        );

        holderSignatureValid =
          recovered.toLowerCase() === document.walletAddress.toLowerCase();
      } catch {
        holderSignatureValid = false;
      }
    }

    const revoked =
      document.status === "revoked" ||
      Boolean(document.revokedAt) ||
      Boolean(blockchain.revoked);

    const issued =

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						103
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    ["issued", "accepted", "claimed"].includes(document.status) &&
    Boolean(document.blockchainTxHash);

const valid =
    !revoked &&
    issued &&
    localHashValid &&
    proofValid &&
    issuerDidValid &&
    credentialSubjectValid &&
    blockchain.hashValid;

return res.json({
    success: true,
    valid,
    revoked,
    state: revoked ? "red" : valid ? "green" : "yellow",
    message: revoked
        ? "Документ анульовано"
        : valid
            ? "Документ підтверджено"
            : "Документ потребує додаткової перевірки",
    document: normalized,
    blockchain,
    checks: {
        vcExists: Boolean(vcJson),
        proofExists: Boolean(proofJson),
        localHashValid,
        proofValid,
        issuerDidValid,
        credentialSubjectValid,
        holderDidExists,
        holderDidValid: holderDidExists
            ? document.holderDid.startsWith("did:pkh:eip155:11155111:")
            : false,
        holderSignatureExists,
        holderSignatureValid,
        blockchainChecked: blockchain.checked,
        blockchainExists: blockchain.exists,
        blockchainHashValid: blockchain.hashValid,
        notRevoked: !revoked
    }
});
} catch (error) {
    return res.status(500).json({
        success: false,
        valid: false,
        message: "Document verification error",
        error: error.message
    });
}
}
}

```

Лістинг frontend-перевірки документа

```

useEffect(() => {
  countedRef.current = false;
  setWalletConfirmed(false);
  setWalletMessage("");
  setWalletMessageType("warning");

  const countHit = async () => {
    if (countedRef.current) return;
    countedRef.current = true;

    try {
      await fetch(`${API_URL}/api/documents/${certificateNumber}/verify-hit`,
{
      method: "POST"
    });
    } catch (error) {
      console.log(error);
    }
  };

  const verify = async () => {
    try {
      setVerifyState("loading");
      setMessage("Перевірка документа...");
      const response = await fetch(
        `${API_URL}/api/documents/verify/${certificateNumber}`,
        { cache: "no-store" }
      );
      const data = await response.json();
      if (!response.ok || !data.success) {
        setVerifyState("error");
        setMessage(data.message || "Документ не знайдено");
        return;
      }
      setVerification(data);
      await countHit();
      if (data.revoked) {
        setVerifyState("error");
        setMessage("Документ анульовано");
        return;
      }
      if (data.valid) {
        setVerifyState("success");
        setMessage("Документ підтверджено");
        return;
      }
      setVerifyState("warning");
      setMessage("Документ потребує додаткової перевірки");
    } catch (error) {
      setVerifyState("error");
      setMessage("Помилка перевірки документа");
    }
  };

  verify();
}, [certificateNumber]);

```

Лістинг blockchain-видачі документа адміністратором

```

const issueDocument = async (document) => {
  try {
    if (!window.ethereum) {
      return alert("MetaMask не знайдено");
    }

    if (!canIssueDocument(document)) {
      return alert("Цей документ уже видано, отримано або анульовано");
    }

    if (!document.documentHash) {
      return alert("У документа немає hash.");
    }

    setLoadingId(document.id);

    await ensureSepolia();

    await window.ethereum.request({
      method: "eth_requestAccounts"
    });

    const provider = new ethers.BrowserProvider(window.ethereum);
    const signer = await provider.getSigner();

    const contract = new ethers.Contract(
      CONTRACT_ADDRESS,
      CONTRACT_ABI,
      signer
    );

    const tx = await contract.storeCertificate(
      document.documentNumber,
      document.documentHash
    );
    const receipt = await tx.wait();
    const txHash = receipt.hash || tx.hash;
    const response = await fetch(
      `${API_URL}/api/documents/id/${document.id}/issue`,
      {
        method: "PATCH",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ txHash })
      }
    );
    const data = await response.json();
    if (!response.ok || !data.success) {
      return alert(data.message || "Не вдалося оновити статус документа");
    }
    alert(`Документ видано та записано у blockchain.\nTx: ${txHash}`);
    await loadDocuments();
  } catch (error) {
    alert(error?.reason || error?.message || "Помилка видачі документа");
  } finally {
    setLoadingId(null);
  }
};

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						106
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг анулювання цифрового документа

```
const revokeOnBlockchain = async (document, reason) => {
  await ensureSepolia();

  await window.ethereum.request({
    method: "eth_requestAccounts"
  });

  const provider = new ethers.BrowserProvider(window.ethereum);
  const signer = await provider.getSigner();

  const contract = new ethers.Contract(
    CONTRACT_ADDRESS,
    CONTRACT_ABI,
    signer
  );

  if (typeof contract.revokeCertificate !== "function") {
    throw new Error("У CONTRACT_ABI немає revokeCertificate.");
  }

  const tx = await contract.revokeCertificate(
    document.documentNumber,
    reason
  );

  const receipt = await tx.wait();

  return receipt.hash || tx.hash;
};
```

					ПЗ.КР.Б-121-26-ПЗ	Арк.
						107
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг універсальної форми створення документів

```

function DocumentFormLayout({
  documentType,
  badge,
  icon,
  title,
  description,
  formClass,
  defaultTitle,
  titleOptions = [],
  titleMode = "select",
  titleLabel = "Назва документа",
  fields,
  requiresValidUntil = false,
  validUntilLabel = "Дійсний до",
  descriptionLabel = "Опис / примітка"
}) {
  const navigate = useNavigate();

  const [formData, setFormData] = useState({
    fullName: "",
    email: "",
    title: defaultTitle,
    issueDate: new Date().toISOString().slice(0, 10),
    validUntil: "",
    description: ""
  });

  const [documentData, setDocumentData] = useState({});
  const handleSubmit = async (event) => {
    event.preventDefault();

    const payload = {
      documentType,
      ...formData,
      validUntil: requiresValidUntil ? formData.validUntil : null,
      documentData,
      ...documentData
    };

    const response = await fetch(`${API_URL}/api/documents`, {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify(payload)
    });

    const data = await response.json();

    if (!response.ok || !data.success) {
      setMessageType("error");
      setMessage(data.message || "Не вдалося створити документ");
      return;
    }

    setMessageType("success");
    setMessage("Документ створено.");

    setTimeout(() => {
      navigate("/admin/documents");
    }, 1200);
  };
}

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						108
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг шаблонів різних типів документів

```

function CertificateForm() {
  return (
    <DocumentFormLayout
      documentType="certificate"
      badge="Шаблон сертифіката"
      icon="📄"
      title="Створити документ: Сертифікат"
      description="Мінімальний шаблон сертифіката."
      formClass="certificate-form-template"
      titleMode="input"
      titleLabel="Назва сертифіката"
      fields={}
      descriptionLabel="Опис / період"
    />
  );
}

function CertificateForm() {
  return (
    <DocumentFormLayout
      documentType="certificate"
      badge="Шаблон сертифіката"
      icon="📄"
      title="Створити документ: Сертифікат"
      description="Мінімальний шаблон сертифіката."
      formClass="certificate-form-template"
      titleMode="input"
      titleLabel="Назва сертифіката"
      fields={}
      descriptionLabel="Опис / період"
    />
  );
}

function IdCardForm() {
  return (
    <DocumentFormLayout
      documentType="id_card"
      badge="Шаблон посвідчення"
      icon="🇺🇦"
      title="Створити документ: Посвідчення"
      description="Шаблон посвідчення містить установу, номер і термін дії."
      formClass="id-form-template"
      defaultTitle="Цифрове посвідчення"
      titleMode="input"
      requiresValidUntil
      validUntilLabel="Дата закінчення дії"
      fields={[
        {
          name: "organization",
          label: "Назва установи",
          type: "text",
          required: true
        },
        {
          name: "identifier",
          label: "Унікальний номер",
          type: "text",
          required: true
        }
      ]}
    />
  );
}

```

```
}  
    }  
    />  
);  
}
```

					ПЗ.КР.Б-121-26-ПЗ	Арк.
						110
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг конфігурації взаємодії зі смарт-контрактом

```

export const CONTRACT_ADDRESS = import.meta.env.VITE_CONTRACT_ADDRESS;

export const SEPOLIA_RPC_URL = import.meta.env.VITE_SEPOLIA_RPC_URL;

export const CONTRACT_ABI = [
  {
    inputs: [
      {
        internalType: "string",
        name: "certificateNumber",
        type: "string"
      },
      {
        internalType: "string",
        name: "certificateHash",
        type: "string"
      }
    ],
    name: "storeCertificate",
    outputs: [],
    stateMutability: "nonpayable",
    type: "function"
  },
  {
    inputs: [
      {
        internalType: "string",
        name: "certificateNumber",
        type: "string"
      },
      {
        internalType: "string",
        name: "reason",
        type: "string"
      }
    ],
    name: "revokeCertificate",
    outputs: [],
    stateMutability: "nonpayable",
    type: "function"
  },
  {
    inputs: [
      {
        internalType: "string",
        name: "certificateNumber",
        type: "string"
      },
      {
        internalType: "string",
        name: "certificateHash",
        type: "string"
      }
    ],
    name: "verifyCertificate",
    outputs: [
      {
        internalType: "bool",
        name: "",
        type: "bool"
      }
    ]
  }
]

```

					ІІЗ.КР.Б-121-26-ІІЗ	Арк.
						111
Змн.	Арк.	№ докум.	Підпис	Дата		

```
    }  
  ],  
  stateMutability: "view",  
  type: "function"  
}  
];
```

					ПЗ.КР.Б-121-26-ПЗ	Арк.
						112
Змн.	Арк.	№ докум.	Підпис	Дата		