Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для голосувань на основі блокчейну. Бек-енд та блокчейн
(тема)

Виконав:
здобувач 4 року навчання
групи ПЗПІ-21-9

Данііл ЛОЗОВИЙ
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Ірина КИРИЧЕНКО
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Лозовому Даніілу Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система для проведення голосувань на основі блокчейну. Бек-енд та блокчейн
Затверджена наказом по університету від 19.05. 2025р. № 397 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 11.06.2022
3. Вихідні дані до роботи методи розробки програмних продуктів, методи розробки мікросервісних додатків, методи розробки блокчейн контрактів, мови програмування.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз проблемної галузі і постановка задачі, опис вимог до програмної системи, опис використовуваних методів та алгоритмів, опис розробленої програмної системи, аналіз можливих застосувань, додатки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	10.05-11.05.2025.	*виконано*
2	Створення специфікації ПЗ	11.05-13.05.2025	*виконано*
3	Проектування ПЗ	13.05-16.05.2025	*виконано*
4	Розробка ПЗ	16.05-01.06.2025	*виконано*
5	Тестування ПЗ	01.06-03.06.2025	*виконано*
6	Оформлення пояснювальної записки	03.06-05.06.2025	*виконано*
7	Підготовка презентації та доповіді	05.06-06.06.2025	*виконано*
8	Попередній захист	06.06-08.06.2025	*виконано*
9	Нормоконтроль, рецензування	08.06-10.06.2025	*виконано*
10	Здача роботи у електронний архів	11.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	11.06.2025	*виконано*

Дата видачі завдання «04» « квітня » 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. кафедри ПІ Ірина КИРИЧЕНКО
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 73 стор., 41 рис., 17 джерел.

БЛОКЧЕЙН, ВЕБ-САЙТ, ГОЛОСУВАННЯ, ПРОГРАМА, ПРОГРАМНА СИСТЕМА, ETHERIUM, GOLANG, SOLIDITY, SQL

Об’єкт розробки – програмна система для голосувань на основі блокчейну, Backend & Blockchain частини.

Мета розробки – створити універсальну та захищену систему, що забезпечує прозорість електронних голосувань, використовуючи переваги як стандартних серверних рішень так і блокчейну.

Метод рішення – середовище розробки GoLand/VS Code, платформа Go для бекенду, мови програмування Golang, Solidity та JavaScript, веб-фреймворк Gin, блокчейн-платформа Ethereum з Layer 2 рішенням (Polygon/Arbitrum), СУБД PostgreSQL для зберігання даних.

У результаті розробки створено Backend & Blockchain частини програмної системи, які дозволяють проводити онлайн-голосування різних типів.

ABSTRACT

BLOCKCHAIN, ETHEREUM, GOLANG, PROGRAM, SOFTWARE SOLIDITY, SQL, VOTING SYSTEM, WEBSITE

The object of development is a blockchain-based voting software system, specifically the Backend & Blockchain components.

The purpose of the development is to create a universal and secure system that ensures transparency of electronic voting, utilizing the advantages of both standard server solutions and blockchain technology.

Solution method – GoLand/VS Code development environment, Go platform for backend, programming languages Golang, Solidity and JavaScript, Gin web framework, Ethereum blockchain platform with Layer 2 solution (Polygon/Arbitrum), PostgreSQL database for data storage.

As a result of the development, the Backend & Blockchain parts of the software system have been created, which allow conducting various types of online voting.

ЗМІСТ

ВСТУП

Проведення голосувань є одним із фундаментальних механізмів прийняття колективних рішень у демократичному суспільстві. Однак традиційні методи голосування часто страждають від недоліків, пов’язаних із прозорістю, анонімністю, безпекою та достовірністю результатів. З розвитком цифрових технологій з'явилася можливість переосмислити сам процес голосування, зробивши його більш надійним, зручним та доступним.

Особливо гострими проблемами сучасних систем голосування залишаються забезпечення конфіденційності виборців при одночасній можливості перевірки факту участі, запобігання маніпуляціям з підрахунком голосів, а також створення інфраструктури, яка б дозволила проводити голосування різного масштабу. Традиційні електронні системи для голосування часто викликають питання щодо захищеності від втручання, неможливості незалежної перевірки результатів, аудитів та вразливості до технічних збоїв чи кібератак, через свою централізованість. Технологія блокчейн, завдяки своїм фундаментальним властивостям незмінності, розподіленої архітектури та криптографічного захисту, пропонує принципово новий підхід до вирішення цих проблем [1]. Використання блокчейну дозволяє створити систему, де кожен голос є одночасно анонімним та перевіреним, результати неможливо сфальсифікувати після їх запису, а весь процес є прозорим для всіх учасників без розкриття конфіденційної інформації.

Розробка електронних систем голосування набула значної актуальності в сучасному світі, особливо після COVID-19. Пандемія змусила країни переосмислити традиційні методи проведення виборів, оскільки фізичне скупчення людей на виборчих дільницях створювало значні ризики для населення. Електронні системи голосування можуть забезпечити не лише безпеку виборчого процесу в умовах санітарних обмежень, але й підвищити доступність голосування для людей з обмеженими можливостями, тих, хто перебуває за кордоном, або у віддалених регіонах. Також такі системи здатні значно прискорити підрахунок голосів, зменшити витрати на організацію виборів та підвищити прозорість електорального

процесу, що є особливо важливим для зміцнення довіри громадян до демократичних інститутів.

Метою даної роботи є розробка універсальної блокчейн-платформи для проведення голосувань різного типу, яка забезпечить високий рівень прозорості процесу при збереженні анонімності учасників, а також надасть гнучкі можливості для організації голосувань з різними вимогами та правилами. Платформа розроблена на основі сучасних блокчейн-технологій, що гарантує незмінність внесених даних та децентралізацію процесу підрахунку результатів.

Робота включає розробку архітектури системи, яка підтримуватиме різні типи голосувань з різними механізмами верифікації учасників, проектування смарт-контрактів[2] для забезпечення логіки голосувань у блокчейні, проектування бекенд-сервісів для індексації даних з блокчейну та обробки бізнес-логіки, аналіз можливостей використання криптографічних механізмів для забезпечення анонімності в закритих голосуваннях при збереженні можливості перевірки права голосу.

Галузь застосування результатів розробки є досить широкою. Платформа може використовуватися для проведення корпоративних голосувань, голосувань у громадських організаціях, органах місцевого самоврядування, під час виборів студентського самоврядування в університетах, а також як технологічна база для проведення референдумів та опитувань громадської думки. У перспективі, з відповідною сертифікацією, система може бути адаптована для використання при проведенні виборів різного рівня. Основною перевагою розроблюваної системи є поєднання прозорості, характерної для відкритих голосувань, та анонімності закритих голосувань, з надійним механізмом верифікації, що унеможливлює фальсифікації. Така комбінація властивостей, реалізована на основі перевірених блокчейн-технологій, дозволить створити універсальний інструмент для демократичного прийняття рішень, що відповідає найвищим стандартам безпеки та надійності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Публічні голосування як прояв волевиявлення народу є одним з основних стовпів будь-якого сучасного демократичного суспільства. Механізми голосування забезпечують легітимність влади, дозволяють вирішувати важливі суспільні питання та приймати колективні рішення. Історично вони змінювалися та вдосконалювалися, відображаючи технологічний розвиток суспільства. Із розвитком інформаційних технологій та переходом більшості процесів у цифровий формат, електронні системи голосування набувають все більшого поширення. Ця тенденція посилилася особливо після пандемії COVID-19, коли потреба у них стала критичною. Електронні голосування дозволяють значно скоротити витрати на організацію процесу, забезпечити швидший підрахунок результатів та зробити участь доступнішою для широкого кола громадян, включаючи осіб з обмеженими можливостями або тих, хто перебуває за кордоном.

Однак традиційні централізовані електронні системи голосування викликають значні сумніви щодо прозорості та надійності. Виборець фактично не має технічної можливості переконатися, що його голос був правильно врахований, а процес підрахунку проводився без маніпуляцій та втручання. Системи, де всі дані зберігаються на централізованих серверах, створюють єдину точку відмови та вразливі до різноманітних кібератак [3].

Блокчейн-технологія пропонує принципово новий підхід до проведення електронних голосувань. Блокчейн представляє собою розподілену базу даних, організовану у вигляді послідовності блоків, кожен з яких містить набір транзакцій та посилання на попередній блок через криптографічний хеш. Така структура забезпечує незмінність вже записаних даних, оскільки будь-яка модифікація попереднього блоку змінить його хеш та порушить цілісність всього ланцюжка. Децентралізація та незмінність виключають проблему єдиної точки відмови традиційних систем, а прозорість в купі з криптографічним захистом гарантують чесність та неможливість маніпуляцій голосами. Кожен голос записується як транзакція, що включається в блок і стає частиною незмінного ланцюжка,

створюючи можливість для виборця перевірити, що його голос був включений у блокчейн без змін. Велика кількість досліджень свідчить про дійсність цих переваг над звичайними голосуваннями [4, 5].

Реальним прикладом впровадження блокчейн-технології для голосування можна назвати систему Voatz, яка була використана під час проміжних виборів у США в 2018 році. Зокрема, у Західній Вірджинії ця платформа дозволила військовослужбовцям, які перебували за кордоном, проголосувати через смартфони. Голоси були зашифровані та записані в приватний блокчейн, використовуючи біометричну автентифікацію для підтвердження особи виборця. Хоча масштаб цього експерименту був вкрай обмежений (лише близько 144 виборців взяли участь), він продемонстрував практичну можливість використання блокчейну для офіційних виборів [6].

Аналізуючи існуючі блокчейн-рішення для голосування, можна виділити кілька основних платформ. Follow My Vote є однією з перших блокчейн-платформ для голосування, із фокусом на прозорості всього процесу [7]. Horizon State пропонує рішення для корпоративного управління та громадських консультацій, впроваджуючи систему власних токенів для забезпечення права голосу [8]. Vocdoni, розроблена на базі мережі Ethereum, надає інструменти для проведення децентралізованих голосувань з відкритим вихідним кодом, орієнтуючись на громадські організації та кооперативи [9]. Ці платформи, хоча й демонструють потенціал блокчейн-технології для проведення голосувань, мають певні обмеження. Більшість з них спеціалізується на конкретних сценаріях використання – корпоративне голосування, або виборчий процес, не пропонуючи універсального рішення. Крім того, вони часто потребують значного рівня технічної грамотності від користувачів, що створює бар'єр для масового впровадження. Соціальний аспект є вкрай важливим для інтеграції подібної системи, оскільки без довіри та обізнаності населення попит на неї буде майже нульовим.

Слід зазначити дослідження проти впровадження блокчейну у голосування. Американський фонд «U.S. VOTE» у своїй статті назвав безпеку такого підходу

«міфом», назвавши проблеми колізій декількох ланцюжків блоків від різних організацій [10]. Авторитетний «Journal of Cybersecurity» на Oxford Academic, що виборці «мають усі причини бути збентеженими безпекою онлайн голосувань», а усі домисли стосовно збільшення безпеки завдяки онлайн та блокчейн голосуваннями називають оманливими [11].

Важливим аспектом є також інтеграція блокчейн-систем голосування з традиційними процесами ідентифікації та автентифікації. Це може включати використання цифрових ID, біометричних даних або традиційних документів, що посвідчують особу. Наприклад можливе використання Дії для громадян України або BankID, який, в тому чи іншому вигляді, існує в більшості країн світу.

1.2 Виявлення та вирішення проблем

Одна з основних проблем полягає у забезпеченні балансу – гарантувати, що голосувати можуть лише авторизовані особи, які відповідають бажаним критеріям, при цьому зберігаючи повну анонімність їх вибору. У традиційних паперових системах це досягається фізичним розділенням процесів ідентифікації та голосування. Виборець ідентифікується перед отриманням бюлетеня, але голосує в окремій кабінці. У цифрових системах цей принцип реалізувати значно складніше, оскільки у традиційних системах встановити який саме бюлетень належить конкретному виборцю дуже складно, якщо взагалі можливо, в той час як в електронних голосуваннях кожен користувач так чи інакше матиме ідентифікатор, який, теоретично, можна буде пов’язати з особою громадянина який пройшов верифікацію. Для вирішення цієї проблеми найбільш перспективним є підхід з використанням сліпих підписів та доказів з нульовим розголошенням (Zero-Knowledge Proofs, ZKP) [12]. Механізм сліпих підписів, запропонований ще у минулому сторіччі, дозволяє отримати підпис на документі, не розкриваючи його вмісту. У контексті голосування це працює наступним чином: користувач генерує унікальний ідентифікатор для участі у голосуванні, "засліплює" його та відправляє на підпис довіреній особі або організатору голосування. Довірена особа, перевіривши право користувача на участь у голосуванні, підписує "засліплений"

ідентифікатор. Користувач "розсліплює" підпис і отримує підпис свого ідентифікатора, не розкриваючи зв'язку між своєю особистістю та цим ідентифікатором. Додатковим рівнем захисту є використання ZKP-протоколів, які дозволяють довести володіння певними даними без їх розкриття. Таким чином технічно буде відсутня можливість розкриття особистості, при цьому зберігаючи можливість встановлення обмежень та критеріїв для виборців.

Інша критична проблема полягає у запобіганні подвійному голосуванню. У блокчейні кожна транзакція незмінно фіксується, але це не запобігає можливості створення кількох транзакцій одним користувачем. Рішенням є створення реєстру використаних ідентифікаторів, який зберігається у смарт-контракті голосування. Якщо ідентифікатор вже використаний, голос відхиляється. При цьому, завдяки впровадженню сліпих підписів та ZKP, реєстр містить лише анонімні ідентифікатори, не пов'язані з реальними особами, що зберігає анонімність голосування.

Також важливою проблемою є масштабованість блокчейн-систем. Основні публічні блокчейни, як-от Ethereum, мають обмежену пропускну здатність, що може стати критичним обмеженням при проведенні масштабних голосувань, а високі комісії за транзакції роблять подібні голосування економічно невигідним. Для вирішення цієї проблеми необхідно використовувати рішення другого рівня (Layer 2) на базі Ethereum, такі як Polygon, Arbitrum або Optimism. Вони забезпечують значно вищу пропускну здатність (тисячі транзакцій на секунду) та нижчу вартість транзакцій, зберігаючи при цьому безпеку основного блокчейну [13]. Таким чином, можна уникнути використання приватної блокчейн мережі і зберегти повну прозорість.

Значною проблемою є і доступність блокчейн-систем для широкого кола користувачів. Більшість існуючих рішень вимагають певного рівня технічної грамотності, розуміння принципів роботи з криптовалютними гаманцями, приватними ключами, інтерфейсами блокчейн-додатків. Це створює бар’єр для недосвідчених або людей похилого віку. Вирішення цієї проблеми полягає у

створенні інтуїтивно зрозумілого інтерфейсу, який приховує технічну складність від кінцевого користувача.

Серйозною проблемою можна назвати забезпечення стійкості системи до різноманітних атак. Блокчейн-системи, попри свою захищеність, все ще мають загрози, такі як атаки Sybil (створення багатьох фіктивних цифрових особистостей), атаки на мережеву інфраструктуру, соціальна інженерія тощо [14]. Комплексне вирішення цієї проблеми включає декілька підходів. По-перше, використання надійної системи ідентифікації користувачів, що запобігає легкому та швидкому створенню фіктивних аккаунтів. По-друге, забезпечення можливості аудиту системи незалежними експертами та пошук акаунтів з підозрілою активністю. Ця проблема є глобальною і має часткові рішення в усіх сферах де використовується блокчейн.

Дослідження демонструють, що Smart-контракти додають ризики, яких немає в більшості звичайних програм. Це може бути злом контракту, програмні помилки в коді або протоколі. Якщо врахувати відносну безпеку blockchain, ці концепції тісно взаємопов'язані [15].

1.3 Постановка задачі

Для вирішення вищезазначених проблем, потрібно розробити універсальну блокчейн-платформу для проведення електронних голосувань. Платформа повинна забезпечувати можливість проведення як відкритих (прозорих) голосувань, так і закритих (анонімних) з різними механізмами верифікації учасників та методами підрахунку результатів. Система має базуватися на сучасних блокчейн-технологіях для швидкої та дешевої обробки транзакцій, при цьому забезпечуючи зручний доступ через веб-інтерфейс та мобільний додаток.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

- розробити архітектуру гібридної системи розподіливши функціонал між блокчейном та бекендом;
- розробити модель даних із чітким розмежуванням прав доступу для різних ролей користувачів;

- забезпечити анонімність в закритих голосуваннях з використанням сліпих підписів та ZKP;
- розробити механізми верифікації голосів зберігаючи анонімність виборців;
- забезпечити масштабованість системи для підтримки великої кількості одночасних голосувань;
- оптимізувати блокчейн-взаємодію для мінімізації вартості транзакцій.

1.3.1 Цільова аудиторія

Основною цільовою аудиторією розроблюваної платформи будуть:

1) Організатори голосувань:
   - представники державних органів, відповідальні за проведення виборів;
   - керівництво компаній, що проводить голосування серед акціонерів або співробітників;
   - адміністрація навчальних закладів для організації виборів студентського самоврядування;
   - керівники громадських організацій та ініціатив;
   - органи місцевого самоврядування для проведення референдумів та опитувань;
   - звичайні люди, які бажають провести голосування всередині інтернет-спільнот.
2) Учасники голосувань:
   - будь-хто, залежно від встановлених обмежень на голосування. Можуть бути як тільки окремі громадяни або акціонери компаній, так і будь-яка людина яка зареєстрована на платформі.

1.3.2 Типи користувачів

Система передбачає такі типи користувачів з різними рівнями доступу та функціональними можливостями:

1) Адміністратор платформи – відповідає за загальне функціонування системи, підтримку її інфраструктури, управління глобальними налаштуваннями та доступом супер-користувачів. Не можуть впливати на процес голосування окрім його примусового завершення, за порушення правил платформи;
2) Стандартний користувач – базовий тип користувача з можливістю участі в різних голосуваннях та створення відкритих голосувань;
3) Супер-користувач – користувачі з розширеними можливостями, які можуть створювати закриті голосування з розширеними параметрами безпеки та анонімності, управляти списком довірених осіб. Цей статус може надаватися адміністрацією або за додаткову оплату;
4) Довірена особа – спеціальний тип користувача, який призначається для конкретних закритих голосувань і відповідає за верифікацію учасників. Довірені особи перевіряють право користувачів на участь у голосуванні та видають їм сліпі підписи, як підтвердження права на голосування.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1. Загальний опис системи

Архітектурно система складається з чотирьох основних компонентів – блокчейн-частини, бекенд-сервісів, фронтенд-частини та мобільного додатку. Далі розглядаємо проектування лише блокчейн- та бекенд-частин.

Блокчейн-частина системи базується на технології Ethereum з використанням рішень другого рівня (Layer 2) для забезпечення масштабованості та зниження вартості транзакцій. В основі лежать смарт-контракти, розроблені на мові Solidity, що реалізують усю ллогіку різних типів голосувань, яка потребує відкритого збереження даних у блокчейні.

Бекенд-частина має мікросервісну архітектуру, складаючись з декількох сервісів, що взаємодіють між собою та з блокчейном через API. Ці сервіси забезпечують аутентифікацію користувачів, управління ролями та правами доступу, обробку запитів від клієнтських додатків та генерацію статистики.

2.2. Функціональні вимоги

Вимоги до блокчейн-частини:

- зберігання криптографічних хешів бюлетенів та результатів голосувань;
- перевірка ZKP-доказів для підтвердження права голосу без розкриття особистості;
- ведення реєстру використаних анонімних ідентифікаторів проти повторного голосування;
- фіксація фінальних результатів голосувань у незмінному вигляді;
- підтвердження коректності підрахунку голосів;
- зберігання публічних ключів Довірених осіб, авторизованих для верифікації учасників.

Вимоги до бекенд-частини:

- авторизація та автентифікація користувачів;

- управління користувачами та їх ролями;
- генерація та безпечне зберігання криптографічних ключів користувачів;
- реалізація off-chain протоколу сліпих підписів для верифікації учасників;
- генерація та перевірка ZKP-доказів для підтвердження права голосу;
- створення, налаштування та управління всіма параметрами голосувань;
- підрахунок голосів;
- відправка транзакцій у блокчейн від імені користувачів;
- управління списком Довірених осіб, включаючи додавання та відкликання;
- формування статистики та аналітичних звітів по голосуванням.

2.3 Розподіл відповідальності між компонентами системи

Блокчейн-частина відповідає виключно за:

- незмінне зберігання криптографічних підтверджень голосів;
- запобігання подвійному голосуванню через реєстр використаних ідентифікаторів;
- перевірку коректності ZKP-доказів права на голосування;
- підтвердження фінальних результатів голосування.

Бекенд-частина відповідає за:

- всю бізнес-логіку створення та управління голосуваннями;
- криптографічні операції генерації сліпих підписів та ZKP-доказів;
- автентифікацію та авторизацію користувачів;
- управління життєвим циклом голосування;
- підрахунок та аналіз результатів;
- забезпечення користувацького інтерфейсу через API.

2.4 Нефункціональні вимоги

Вимоги до безпеки:

- шифрування всіх чутливих даних при зберіганні та передачі;
- неможливість встановлення зв'язку між ідентифікатором користувача та його голосом у закритих голосуваннях;
- захист від типових веб-атак (SQL-ін'єкції, атаки на JWT-токени);
- впровадження механізму контролю доступу на основі ролей.

Надійність:

- обробка помилок з інформативними повідомленнями без розкриття чутливої інформації;
- логування всіх критичних операцій системи для подальшого аналізу.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 UML проєктування ПЗ

UML-діаграма класів (Class Diagram) є основною діаграмою. Вона включатиме ключові елементи системи (див. рис. 3.1). Ключовими класами в системі є User (Користувач), Poll (Голосування), Vote (Голос).

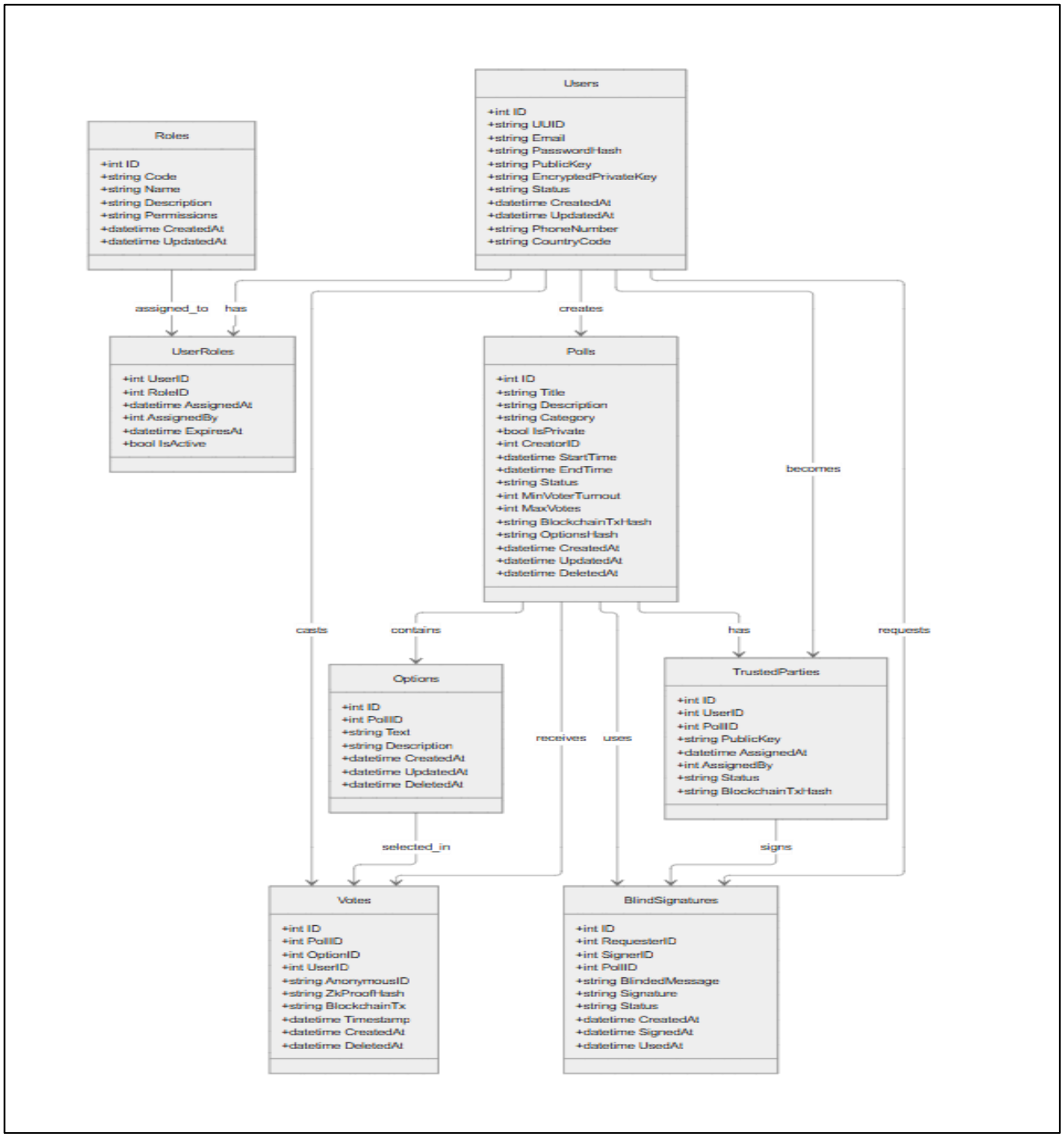

Рисунок 3.1 – UML-діаграма класів (рисунок виконаний самостійно)

Діаграма компонентів – включає бекенд та блокчейн-частини (див. рис. 3.2). На верхньому рівні знаходяться клієнтські додатки (Web Application, Mobile Application, Admin Dashboard), які взаємодіють із системою через API Gateway. Основний функціонал забезпечується чотирма мікросервісами: Identity Service (управління користувачами), Poll Management Service (керування голосуваннями), Verification Service (перевірка прав голосу) та Blockchain Integration Service (взаємодія з блокчейном). Система використовує три бази даних для зберігання різних типів інформації та інфраструктурні компоненти (Logger Service, Config Service). Блокчейн-частина представлена L2 Blockchain Network з Voting Contract System. Діаграма чітко демонструє модульну мікросервісну архітектуру з розподілом відповідальності між компонентами та їх взаємозв'язками.

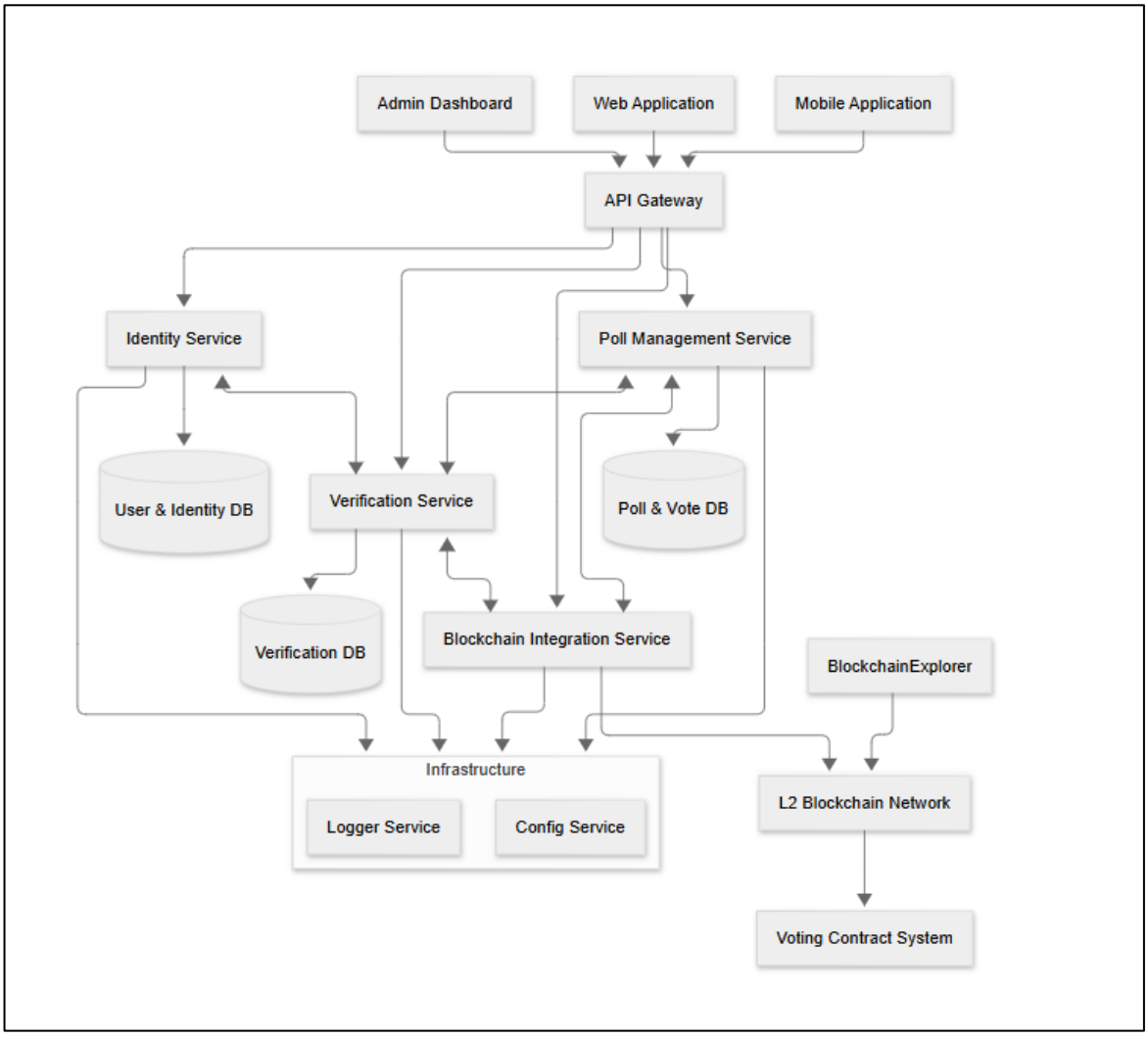

Рисунок 3.2 – UML-діаграма компонентів (рисунок виконаний самостійно)

Для більш детального розуміння ключових процесів у системі, створено діаграми послідовності для процесу створення голосування, відправки голосу, закінчення голосування та підрахунку голосів.

Для створення голосування послідовність починається із заповнення форми голосування у веб-додатку, який відправляє запит до API Gateway. Далі відбувається валідація JWT-токена через Identity Service, після чого Poll Management Service створює голосування, перевіряє параметри та генерує ID. Залежно від типу (відкрите чи закрите), голосування зберігається у Poll & Vote DB, потім ID повертається через API Gateway до веб-додатку, який відображає створене голосування адміністратору (див. рис. 3.3).

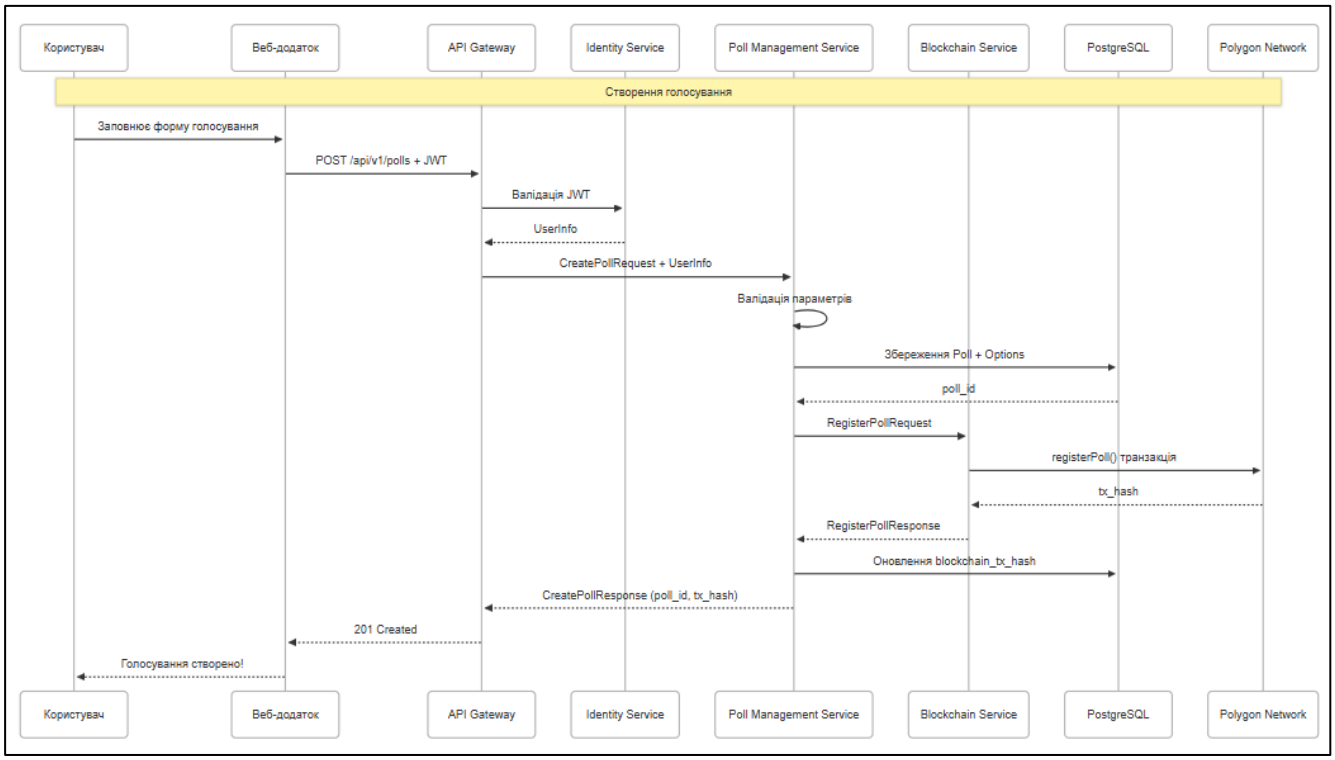

Рисунок 3.3 – UML-діаграма послідовності для створення голосування (рисунок виконаний самостійно)

Для голосування користувач звертається у Poll Management Service. Але перед початком сервіс надсилає токен авторизації користувача до Identity Service, там йде перевірка що користувач не заблокований і має право голосувати. Далі йде перевірка на те, що голосування та обраний варіант існують, варіант належить до

голосування, воно ще не закрилось та час не вичерпався. І тільки після цього, за отриманим публічним ключем користувача з Identity Service генерується його унікальна адреса яка прикріплюється до його голосу. Голос передається до Blockchain Service, який, в свою чергу, зв’язується зі смарт контрактами та зберігає голос у блокчейні. Далі підтвердження повертається по ланцюжку користувачеві. (див. рис. 3.4).

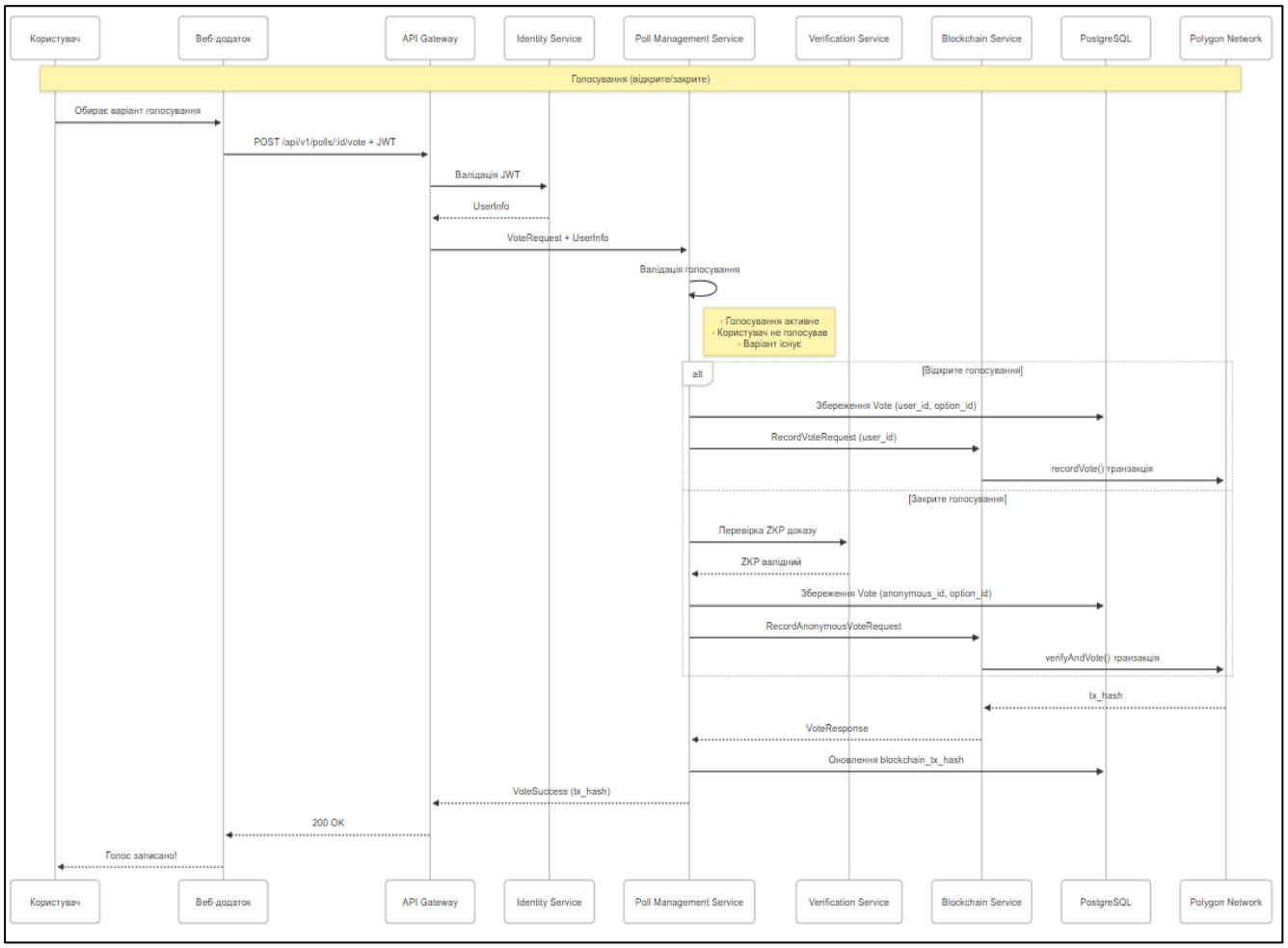

Рисунок 3.4 – UML-діаграма послідовності для збереження голосу (рисунок виконаний самостійно)

Для закриття голосувань та підрахунку голосів використовуються сервіси, які запущені у окремих процесах, так звані Cro Jobs. Вони кожний деякий період часу перевіряють блокчейн за приводу появи необроблених транзакцій або подій.(див. рис. 3.5). Вони індексують цю інформацію у БД. Крім того, один з таких процесів і займається перевірками обмежень часу для голосувань та їх закриттям.

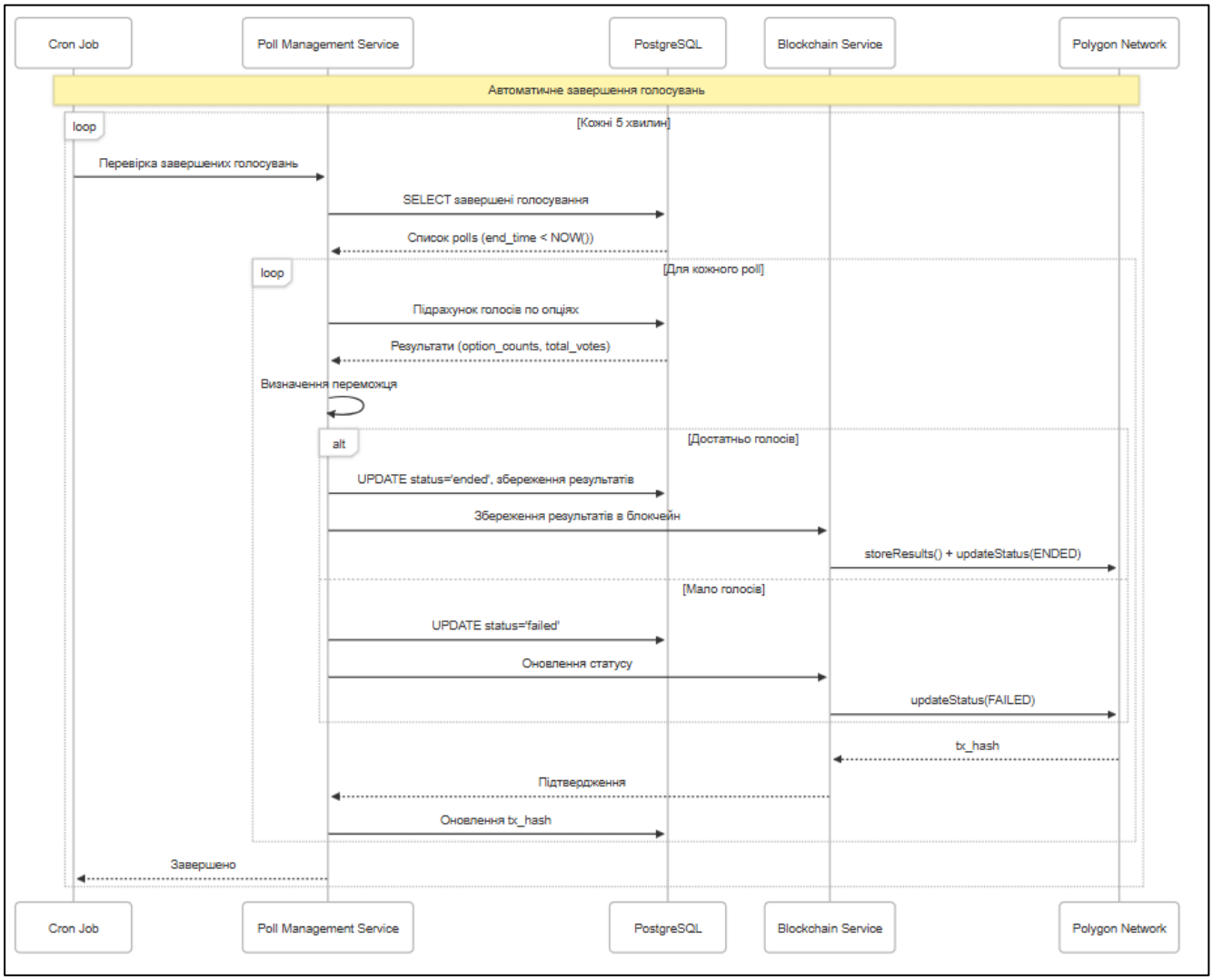

Рисунок 3.5 – UML-діаграма послідовності для голосування з ZKP (рисунок виконаний самостійно)

3.2 Проєктування архітектури ПЗ

Архітектура платформи для голосувань побудована на гібридній моделі, що поєднує переваги блокчейн-технологій з ефективністю традиційних серверних рішень. Це дозволяє забезпечити необхідний баланс між безпекою, продуктивністю та масштабованістю системи. Блокчейн-частина відповідає за зберігання критичних даних, які вимагають гарантії незмінності та прозорості: хеші голосів, реєстр використаних анонімних ідентифікаторів, публічні ключі довірених осіб та фінальні результати голосувань. Бекенд-частина в свою чергу забезпечує бізнес-логіку, аутентифікацію користувачів, генерацію криптографічних доказів,

індексацію даних та інтерфейси взаємодії. Використання саме гібридної моделі дозволяє вирішити одну з ключових проблем традиційних блокчейн-систем, а саме низьку продуктивність та високу вартість транзакцій, зберігаючи при цьому необхідний рівень децентралізації та незмінності даних.

Для ефективної організації бекенд-частини системи було обрано мікросервісну архітектуру, яка забезпечує гнучкість розробки, незалежне масштабування сервісів та дозволяє позбутись єдиної точки відмови. Система розділена на п'ять ключових сервісів:

- Identity Service відповідає за управління користувачами та їх профілями, аутентифікацію та авторизацію, генерацію та управління криптографічними ключами користувачів;
- Poll Management Service забезпечує створення та налаштування голосувань, обробку голосів користувачів, управління усіма аспектами голосувань та реалізацію алгоритмів підрахунку результатів;
- Verification Service займається генерацією сліпих підписів для верифікації права на участь, створенням та перевіркою доказів з нульовим розголошенням (ZKP);
- Blockchain Integration Service забезпечує взаємодію з блокчейн-мережею, створення та підписання транзакцій, індексацію блокчейн-даних, а також моніторинг статусу транзакцій та обробку подій з блокчейну.

Комунікація між сервісами здійснюється за допомогою двох основних механізмів. Синхронна взаємодія через REST API використовується для операцій, що вимагають негайної відповіді. Взаємодія також відображена на діаграмі розгортання (див. рис. 3.6)

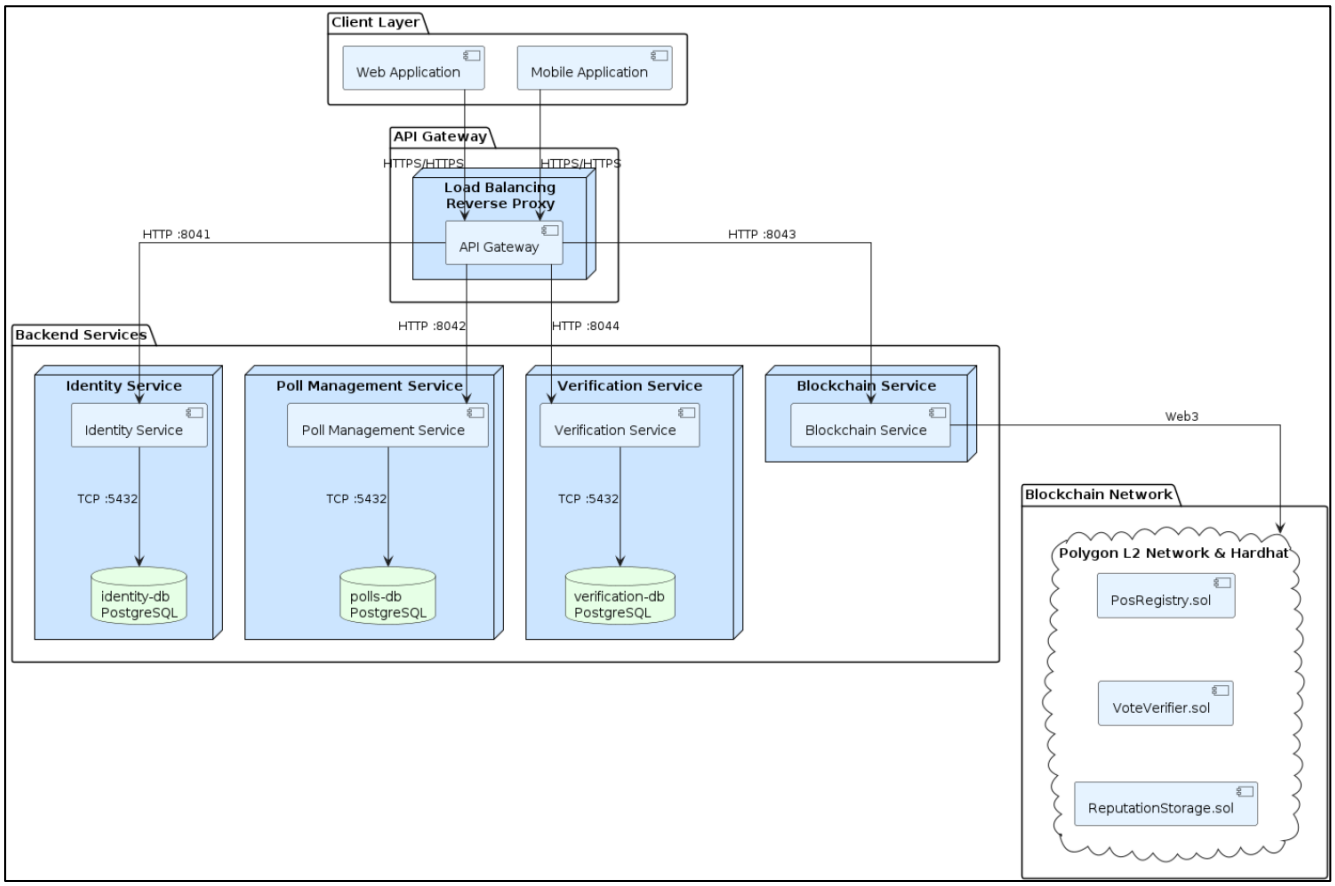

Рисунок 3.6 – Діаграма розгортання (рисунок виконаний самостійно)

Для реалізації блокчейн-частини системи обрано L2-рішення на базі Ethereum (Polygon або Arbitrum), переваги котрих було розглянуто у розділі 2. Смарт-контракти розроблені з використанням мови Solidity та оптимізовані для мінімізації використання газу. Основні смарт-контракти системи включають PollRegistry – реєстр голосувань з основними метаданими, VoteVerifier – контракт для перевірки легітимності голосів та ZKP-доказів, TrustedPartyRegistry – реєстр довірених осіб для закритих голосувань, та ResultStorage – зберігання фінальних результатів голосувань.

Для забезпечення надійності та продуктивності система включає додаткові інфраструктурні компоненти. PostgreSQL слугує реляційною базою даних для зберігання структурованих даних. API Gateway виступає єдиною точкою входу для зовнішніх запитів з функціями маршрутизації та обмеження навантаження.

3.3 Проєктування структури зберігання даних

Після аналізу вимог системи та сутностей, які використовуються в ній, створено низку взаємозв'язків між ними. В основі системи лежать користувачі, які можуть мати різні ролі з відповідними правами. Вони створюють голосування, які містять варіанти відповідей. Для закритих голосувань призначаються довірені особи, які підтверджують право користувачів на участь через систему сліпих підписів. Користувачі беруть участь у голосуваннях, подаючи свої голоси. Для забезпечення анонімності використовуються криптографічні докази, а для забезпечення цілісності результатів використаємо записи в блокчейні.

Основними сутностями системи є користувачі, ролі, голосування, варіанти відповідей, голоси, довірені особи, сліпі підписи, криптографічні докази та блокчейн-транзакції. Кожна з цих сутностей представлена відповідною таблицею в реляційній базі даних, а взаємозв'язки між ними забезпечуються через систему первинних та зовнішніх ключів. Зв’язки та взаємодії сутностей відображено на ER-діаграмі (див. рис. 3.7).

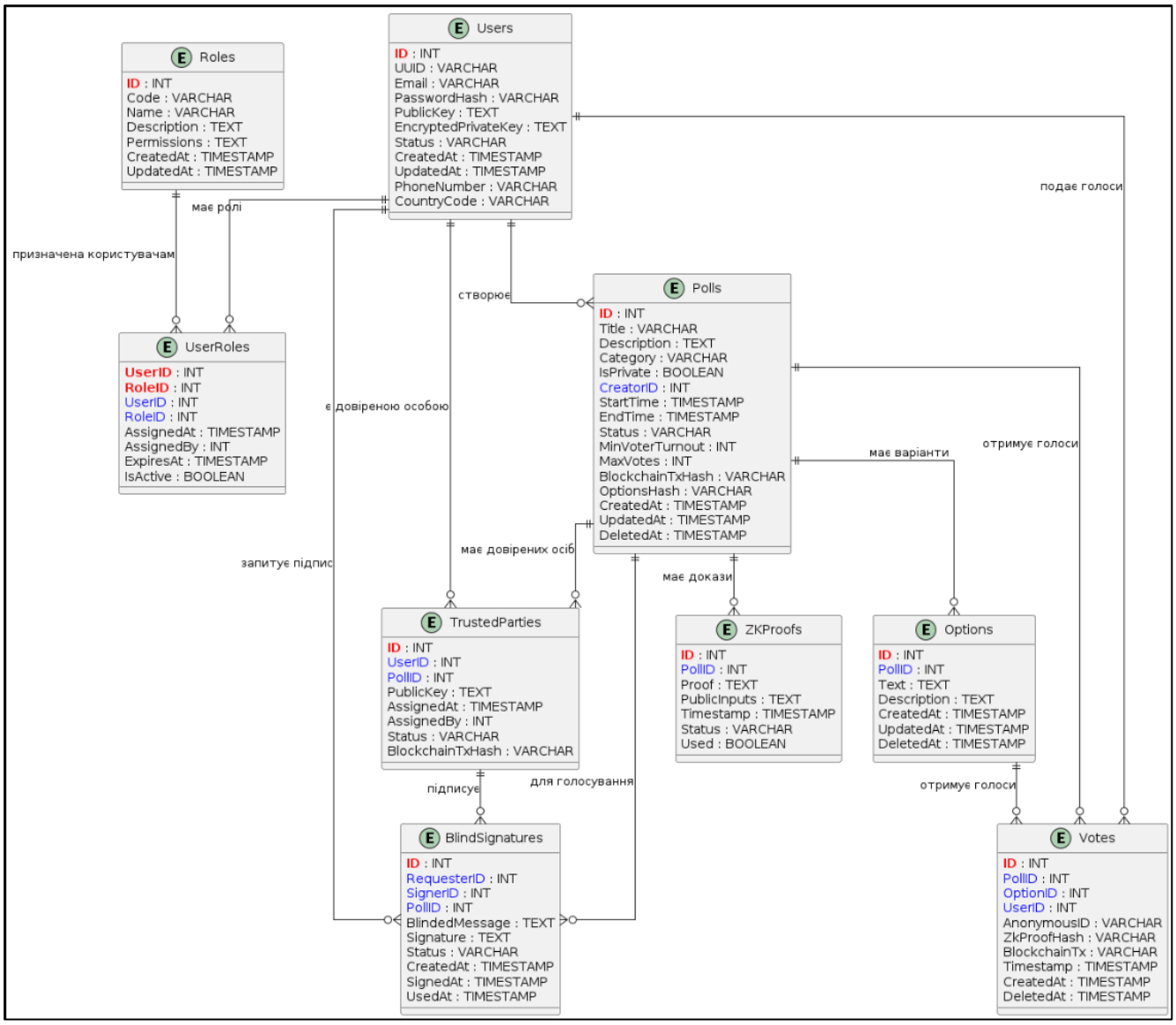

Рисунок 3.7 – ER-діаграма (рисунок виконаний самостійно)

Отже, в базі даних наявні наступні таблиці:

– таблиця Users (Користувачі) має у собі: первинний ключ ID (користувач_id), UUID (унікальний ідентифікатор), Email (електронна пошта), PasswordHash (хеш паролю), PublicKey (публічний ключ), EncryptedPrivateKey (зашифрований приватний ключ), Status (статус користувача), CreatedAt (дата створення), UpdatedAt (дата оновлення), PhoneNumber (номер телефону), CountryCode (код країни). Кожен атрибут залежить тільки від первинного ключа та відсутні транзитивні зв'язки, а отже таблиця знаходиться в 3НФ;

- таблиця Roles (Ролі) має у собі: первинний ключ ID (роль_id), Code (код ролі), Name (назва ролі), Description (опис ролі), Permissions (перелік дозволів), CreatedAt (дата створення), UpdatedAt (дата оновлення). Кожен атрибут залежить тільки від первинного ключа та відсутні транзитивні зв'язки, а отже таблиця знаходиться в 3НФ;
- таблиця UserRoles (РоліКористувачів) має у собі: комбінований первинний ключ (UserID, RoleID), де UserID є зовнішнім ключем до таблиці Users, а RoleID є зовнішнім ключем до таблиці Roles, AssignedAt (дата призначення), AssignedBy (хто призначив), ExpiresAt (дата закінчення дії), IsActive (активність ролі). Кожен атрибут залежить тільки від первинного ключа та відсутні транзитивні зв'язки, а отже таблиця знаходиться в 3НФ;
- таблиця Polls (Голосування) має у собі: первинний ключ ID (голосування_id), Title (назва), Description (опис), Category (категорія), IsPrivate (приватність), CreatorID (автор, зовнішній ключ до таблиці Users), StartTime (час початку), EndTime (час закінчення), Status (статус голосування), MinVoterTurnout (мінімальна явка), MaxVotes (максимальна кількість голосів), BlockchainTxHash (хеш блокчейн транзакції), OptionsHash (хеш варіантів), CreatedAt (дата створення), UpdatedAt (дата оновлення), DeletedAt (дата видалення). Основні метадані голосування зберігаються в блокчейні для забезпечення незмінності. Кожен атрибут залежить тільки від первинного ключа та відсутні транзитивні зв'язки, а отже таблиця знаходиться в 3НФ;
- таблиця Options (ВаріантиВідповідей) має у собі: первинний ключ ID (варіант_id), PollID (зовнішній ключ до таблиці Polls), Text (текст варіанту), Description (опис варіанту), CreatedAt (дата створення), UpdatedAt (дата оновлення), DeletedAt (дата видалення). Хеші варіантів відповідей зберігаються в блокчейні через поле OptionsHash у таблиці Polls. Кожен атрибут залежить тільки від первинного ключа та відсутні транзитивні зв'язки, а отже таблиця знаходиться в 3НФ;

- таблиця Votes (Голоси) має у собі: первинний ключ ID (голос_id), PollID (зовнішній ключ до таблиці Polls), OptionID (зовнішній ключ до таблиці Options), UserID (користувач, зовнішній ключ), AnonymousID (анонімний ідентифікатор), ZkProofHash (хеш ZK доказу), BlockchainTx (блокчейн транзакція), Timestamp (час подачі голосу), CreatedAt (дата створення), DeletedAt (дата видалення). Голоси записуються в блокчейн через анонімний ідентифікатор без можливості встановлення зв'язку з конкретним користувачем. Кожен атрибут залежить тільки від первинного ключа та відсутні транзитивні зв'язки, а отже таблиця знаходиться в 3НФ;
- таблиця TrustedParties (ДовіреніОсоби) матиме у собі: первинний ключ ID, UserID (зовнішній ключ до Users), PollID (зовнішній ключ до Polls), PublicKey (публічний ключ), AssignedAt (дата призначення), AssignedBy (хто призначив), Status (статус), BlockchainTxHash (хеш транзакції). Публічні ключі довірених осіб зберігатимуться в блокчейні для перевірки сліпих підписів;
- таблиця BlindSignatures (СліпіПідписи) матиме у собі: первинний ключ ID, RequesterID (зовнішній ключ до Users), SignerID (зовнішній ключ до TrustedParties), PollID (зовнішній ключ до Polls), BlindedMessage (засліплене повідомлення), Signature (підпис), Status (статус), CreatedAt, SignedAt, UsedAt (часові мітки). Реєстр використаних підписів зберігатиметься в блокчейні;
- таблиця ZKProofs (ДоказиЗНульовимРозголошенням) має у собі: первинний ключ ID (доказ_id), PollID (зовнішній ключ до таблиці Polls), Proof (доказ), PublicInputs (публічні входи), Timestamp (час створення), Status (статус доказу), Used (використано). Перевірка доказів відбувається в смарт-контракті блокчейну. Кожен атрибут залежить тільки від первинного ключа та відсутні транзитивні зв'язки, а отже таблиця знаходиться в 3НФ.

Всі таблиці знаходяться в 3НФ, а значить уся база даних знаходиться в 3НФ.

3.4 Приклади найцікавіших алгоритмів та методів

Одним із найскладніших криптографічних механізмів, використаних у системі, є докази з нульовим розголошенням (Zero-Knowledge Proofs, ZKP). Ця технологія дозволяє користувачу довести факт володіння певною інформацією без розкриття самої інформації, що є критичним для забезпечення анонімності голосувань.

3.4.1 Теоретичне підґрунтя

Доказ з нульовим розголошенням – це криптографічний протокол, у якому одна сторона (доказник) може переконати іншу сторону (верифікатор) у правдивості певного твердження, не розкриваючи жодної додаткової інформації, крім факту істинності самого твердження. У контексті нашої системи ZKP використовується для вирішення фундаментальної проблеми електронних голосувань: доведення права участі в голосуванні без розкриття особистості виборця. Конкретно, користувач має довести що:

1) Володіє дійсним сліпим підписом від довіреної особи для цього конкретного голосування.
2) Даний сліпий підпис ще не було використано.

Для реалізації ZKP в системі використовується протокол zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge), який дозволяє створювати компактні та ефективні докази, що можуть бути верифіковані без взаємодії з доказником.

3.4.2 Приклад реалізації

Для більш детальної демонстрації принципу роботи протоколу, використаємо псевдокод для наглядної демонстрації (див. додаток В). Алгоритм також можна відобразити у вигляді діаграми послідовності (див. рис. 3.8).

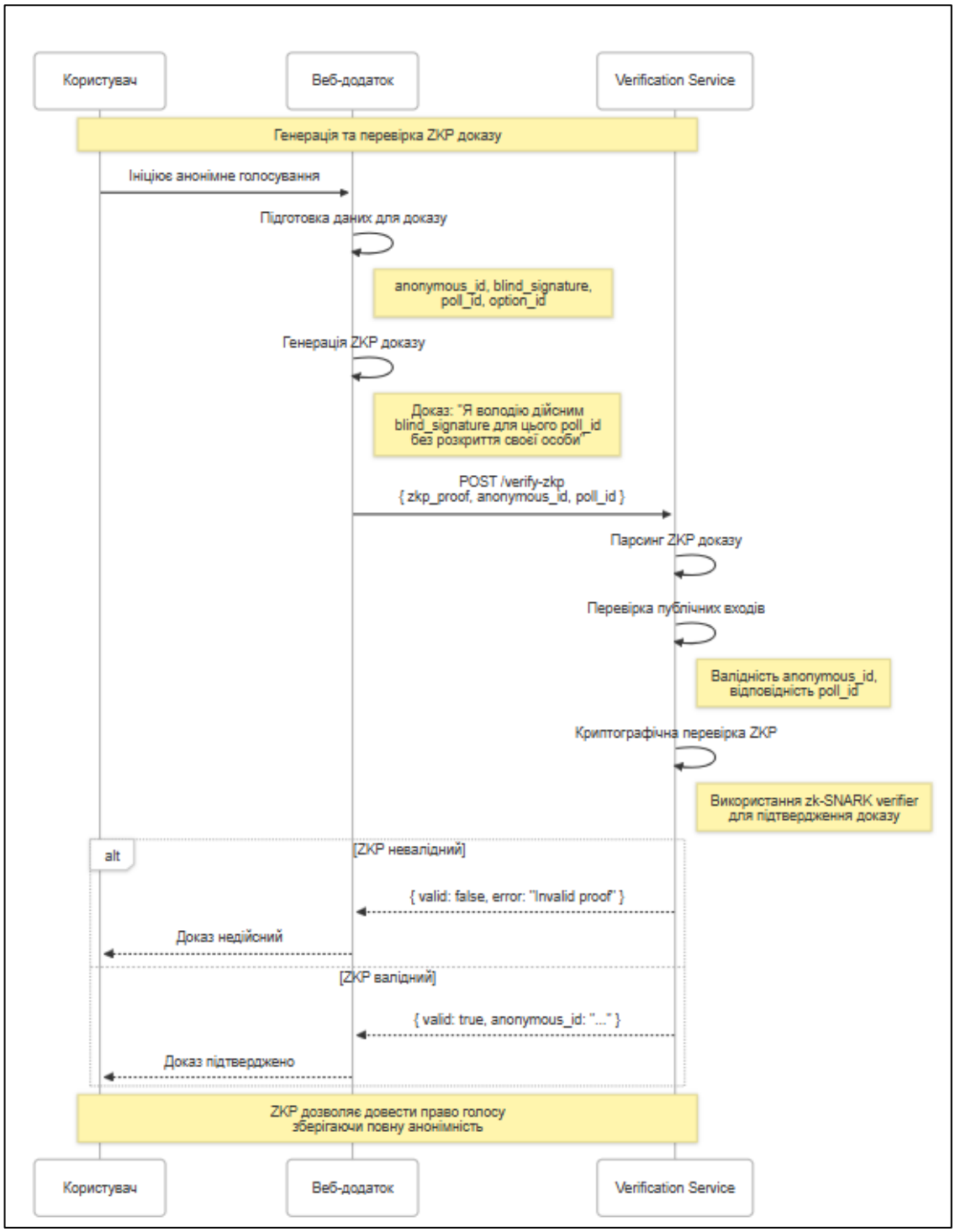

Рисунок 3.8 – UML-діаграма послідовності протоколу ZKP (рисунок виконаний самостійно)

Процес складається з трьох основних учасників: Користувач, Бекенд-система та Смарт-контракт у блокчейні. Спочатку користувач надсилає запит на отримання сліпого підпису, який повертається бекенд-системою. Потім користувач генерує ZKP-доказ володіння дійсним підписом без розкриття своєї особистості і

відправляє цей доказ разом з анонімним голосом до бекенд-системи. Бекенд передає доказ та хеш варіантів голосу до блокчейн-смарт-контракту, який верифікує ZKP-доказ, перевіряє його унікальність та активність голосування. Після запису анонімного голосу в блокчейн, смарт-контракт повертає статус транзакції до бекенд-системи, яка повідомляє користувача про успішне голосування.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Використані технології

Під час розробки серверної частини було обрано гібридну архітектуру, яка використовуватиме переваги децентралізованих технологій блокчейну та традиційних рішень на стороні сервера. Цей підхід дозволяє збалансувати безпеку з продуктивністю системи та масштабованістю, оскільки критично важливі дані потрібно зберігати в блокчейні, щоб гарантувати незмінність, а бізнес-логіка може оброблятися більш ефективними серверними компонентами. Серверна система базувалася на мові програмування Go через її високу швидкість, вбудовану підтримку паралелізму та чудові можливості розробки мікросервісів. Go пропонує швидку компіляцію, ефективне керування пам’яттю та чудову підтримку мережевого програмування, що дуже важливо для цієї системи, оскільки вона має активно взаємодіяти з мережею блокчейн. Крім того, в екосистемі Go доступні потужні бібліотеки для роботи з Ethereum, що значно спрощує інтеграцію з Blockchain. Для веб-фреймворку було обрано Gin, який відзначається високою швидкістю обробки HTTP-запитів та надає зручний API для створення REST endpoints та middleware.

Система бази даних побудована на PostgreSQL. Вона забезпечує повну ACID-сумісність, що критично важливо для безпечного зберігання даних. Потужні можливості індексування PostgreSQL, включаючи B-tree, GIN та GiST індекси, дозволяють ефективно виконувати складні запити для фільтрації голосувань за різними критеріями та швидкого підрахунку результатів.

Для взаємодії з базою даних використовується ORM GORM, яка забезпечує зручний об'єктно-реляційний маппінг з підтримкою автоматичних міграцій, зв'язків між таблицями та безпечних запитів з захистом від SQL ін'єкцій. GORM також підтримує connection pooling, що дозволяє ефективно управляти з'єднаннями з базою даних в умовах високого навантаження. Модель даних спроектована з урахуванням принципів нормалізації до третьої нормальної форми, що забезпечує цілісність даних та мінімізує дублювання інформації.

Блокчейн-частина системи реалізована на мережі Polygon, яка є рішенням другого рівня для Ethereum та використовує архітектуру sidechain з періодичним експортом в основну мережу Ethereum. Вибір Polygon обумовлений кількома ключовими факторами – вартість транзакцій в мережі Polygon складає лише частки цента порівняно з десятками доларів в основній мережі Ethereum, пропускна здатність досягає тисяч транзакцій на секунду порівняно з 15 TPS в Ethereum, час підтвердження транзакцій становить 2-3 секунди замість хвилин.

Критично важливою перевагою Polygon є повна сумісність з Ethereum Virtual Machine, що дозволяє використовувати всі існуючі інструменти розробки, бібліотеки та фреймворки екосистеми Ethereum без жодних модифікацій. Це включає Solidity як мову програмування, Remix та Hardhat як середовища розробки, Web3.js та Ethers.js як клієнтські бібліотеки. Безпека Polygon забезпечується через регулярний експорт в Ethereum, що означає, що навіть у випадку компрометації Polygon мережі, історичні дані залишаються захищеними в основній мережі. Усі тестування проводились у локальній блокчейн-мережі Hardhat.

4.2 Прийняті архітектурні рішення

Система побудована за принципами мікросервісної архітектури, що забезпечує можливість незалежного масштабування окремих компонентів та відсутність єдиної точки відмови. Ізоляція окремих компонентів також збільшує гнучкість розробки, через незалежність компонентів один від одного.

Poll Management Service є центральним компонентом системи, що відповідає за повний життєвий цикл голосувань від їх створення до підведення підсумків. Сервіс реалізує складну бізнес-логіку, включаючи перевірку параметрів голосування, управління часовими рамками, контроль доступу та автоматичне завершення голосувань за розкладом. Архітектура сервісу побудована за принципами чистої архітектури з чіткими шарами: моделі для бізнес-об'єктів, сервісний шар для бізнес-логіки, шар взаємодії з базою даних та HTTP сервер для прийому запитів та взаємодією з іншими сервісами системи. Такий підхід дозволяє легко змінювати систему зберігання даних або мережеву бібліотеку без впливу на

бізнес-логіку: сервісний шар приховує всю бізнес-логіку та координує взаємодію між різними компонентами системи. Це також забезпечує легкість тестування завдяки можливості заміни окремих компонентів або їх імітацією без впливу на решту системи. Наприклад, код структури, яка відображає шар бізнес-логіки виглядає наступним чином:

```
type Service struct {
Poll PollService
Vote VoteService
}
func NewService(deps Dependencies) *Service {
return &Service{
      Poll: NewPollService(deps),
      Vote: NewVoteService(deps),
}
}
```

У цьому коді PollService та VoteService – інтерфейси, які не залежать від реалізації та інших шарів системи, а NewPollService та NewVoteService – відповідно конструктори реалізацій цих інтерфейсів. Наприклад, інтерфейс VoteService виглядає наступним чином:

```
type VoteService interface {
// Голосування
CastVote(ctx context.Context, req model.VoteRequest, pollID uint,
userID *uint, jwtToken string) (*model.VoteResponse, error)

// Перевірки
HasUserVoted(ctx context.Context, pollID, userID uint) (bool, error)
CanUserVote(ctx context.Context, pollID uint, userID uint) (bool,
string, error)
}
```

Такий підхід надає велику гнучкість розробки. Функціонал можна легко розширити або видалити, достатньо змінити метод інтерфейсу та додати або прибрати його з реалізації, це не вплине на інші шари системи. Поза цими шарами також існують два сервіси, схожі на так звані Cron Jobs у Unix системах [16]. Це Blockchain Monitor та Poll Finalizer: вони працюють паралельно і незалежно від інших функцій та виконують свою роботу через певні проміжки часу. Blockchain

Monitor періодично перевіряє транзакції у блокчейні, які у базі даних мають статуси «Оброблюється» та «Невдача», отримує актуальні статуси з блокчейну та індексує їх у БД. Poll Finalizer перевіряє час закінчення усіх активних голосувань, порівнюючи їх з поточним часом. Якщо час вийшов, сервіс змінює статус на закінчений, підраховує результати та публікує їх у блокчейні.

Identity Service забезпечує централізоване управління користувачами, аутентифікацією та авторизацією через JWT-токени, що дозволяє створити архітектуру без необхідності зберігання сесій на сервері. JWT токени містять всю необхідну інформацію про користувача та його права доступу, що дозволяє сервісам незалежно перевіряти запити без звернення до централізованого сховища сесій. Сервіс також відповідає за генерацію та безпечне зберігання криптографічних ключів користувачів для взаємодії з блокчейном. Особливістю реалізації Identity Service є використання криптографічно стійких алгоритмів для хешування паролів та генерації ключів. Це стає можливим завдяки розвинутим стандартним пакетам Go, таким як “crypto” та “encoding”. Використання цих пакетів значно спрощує розробку сервісу, оскільки в них вже реалізовані усі необхідні алгоритми та дотримані найвищі стандарти безпеки, які, крім того, постійно оновлюються. Наприклад, так виглядає код створення приватного та публічного ключів, при реєстрації користувача:

```
func (s *EncryptionService) GenerateRSAKeyPair() (string, string,
error) {
// Generate a new RSA key pair
privateKey, err := rsa.GenerateKey(rand.Reader, 2048)
if err != nil {
      return "", "", err
}

// Convert private key to PEM format
privateKeyBytes := x509.MarshalPKCS1PrivateKey(privateKey)
privateKeyPEM := pem.EncodeToMemory(
      &pem.Block{
            Type:  "RSA PRIVATE KEY",
            Bytes: privateKeyBytes,
      },
)

// Extract public key and convert to PEM format
```

```
publicKeyBytes, err :=
x509.MarshalPKIXPublicKey(&privateKey.PublicKey)
if err != nil {
      return "", "", err
}
publicKeyPEM := pem.EncodeToMemory(
      &pem.Block{
            Type:  "RSA PUBLIC KEY",
            Bytes: publicKeyBytes,
      },
)

return string(publicKeyPEM), string(privateKeyPEM), nil
}
```

Тут rsa, x509, rand – частини пакету crypto, які реалізують відповідні алгоритми.

Система ролей та дозволів побудована на принципах RBAC, що дозволяє гнучко налаштовувати права доступу для різних типів користувачів без зміни коду. Для реалізації цих принципів використовується стороння бібліотека Casbin, яка надає зручні механізми для конфігурування дозволів та їх інтеграцію у middleware сервісу. Файл конфігурації для ролей виглядає наступним чином:

```
p, admin, users, read
p, admin, users, write
p, admin, users, delete
p, admin, polls, read
p, admin, polls, write
p, admin, polls, delete
p, admin, user_roles, read
p, admin, user_roles, write
p, admin, user_roles, delete
p, premium, users, read
p, premium, polls, read
p, premium, polls, write
p, premium, polls, create_private
p, premium, analytics, read
p, user, users, read
p, user, polls, read
p, user, polls, write
p, user, polls, participate
```

Verification Service реалізує найскладніші криптографічні операції системи, включаючи генерацію та верифікацію сліпих підписів для анонімної аутентифікації та створення доказів з нульовим розголошенням для анонімного голосування. Сервіс використовує перевірені криптографічні бібліотеки та протоколи для

забезпечення математичної коректності операцій та стійкості до атак. Реалізація сліпих підписів базується на RSA або elliptic curve криптографії, що дозволяє довіреним особам підписувати повідомлення без розкриття їх змісту. Це забезпечує неможливість встановлення зв'язку між особистістю користувача та його анонімним голосом навіть довіреними особами. Zero-knowledge proofs реалізовані через zk-SNARKs протокол, що дозволяє створювати компактні докази володіння дійсним сліпим підписом без розкриття самого підпису.

Blockchain Service виступає єдиною точкою взаємодії всіх інших сервісів з блокчейн-мережею, приховуючи всю складність Web3 операцій, управління приватними ключами, моніторингу транзакцій та обробки блокчейн подій. Сервіс автоматично керує газовими лімітами на основі поточного стану мережі, реалізує retry логіку для транзакцій, що застрягли, та забезпечує моніторинг статусу всіх відправлених транзакцій.

На базі створених публічних ключів створюються адреси, які додаються до транзакцій у блокчейні. Перетворення можна побачити у коді Blockchain Service:

```
func PublicKeyToAddress(publicKeyHex string) (common.Address, error)
{
publicKeyHex = strings.TrimPrefix(publicKeyHex, "0x")

if len(publicKeyHex) != 128 && len(publicKeyHex) != 130 {
      return common.Address{}, fmt.Errorf("invalid public key length:
expected 128 or 130 characters, got %d", len(publicKeyHex))
}

if len(publicKeyHex) == 130 && publicKeyHex[:2] == "04" {
      publicKeyHex = publicKeyHex[2:]
}

publicKeyBytes, err := hex.DecodeString(publicKeyHex)
if err != nil {
      return common.Address{}, fmt.Errorf("failed to decode public key
hex: %w", err)
}
if len(publicKeyBytes) != 64 {
      return common.Address{}, fmt.Errorf("invalid public key bytes
length: expected 64, got %d", len(publicKeyBytes))
}

pubKey := &ecdsa.PublicKey{
      Curve: crypto.S256(),
}
```

```
pubKey.X = new(big.Int).SetBytes(publicKeyBytes[:32])
pubKey.Y = new(big.Int).SetBytes(publicKeyBytes[32:])

if !pubKey.Curve.IsOnCurve(pubKey.X, pubKey.Y) {
      return common.Address{}, fmt.Errorf("public key point is not on
the secp256k1 curve")
}

address := crypto.PubkeyToAddress(*pubKey)

return address, nil
}
```

Інтеграція між сервісами здійснюється через REST API, кожен сервіс включає ендпоінти для моніторингу стану та graceful shutdown для безпечного завершення роботи.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування проводилось 2 методами: за допомогою програмного забезпечення Postman та мануальне із підключенням до Frontend частини. Для початку протестуємо реєстрацію користувачів. Для цього у Postman створюємо новий запит на зазначену адресу (див. рис. 5.1).

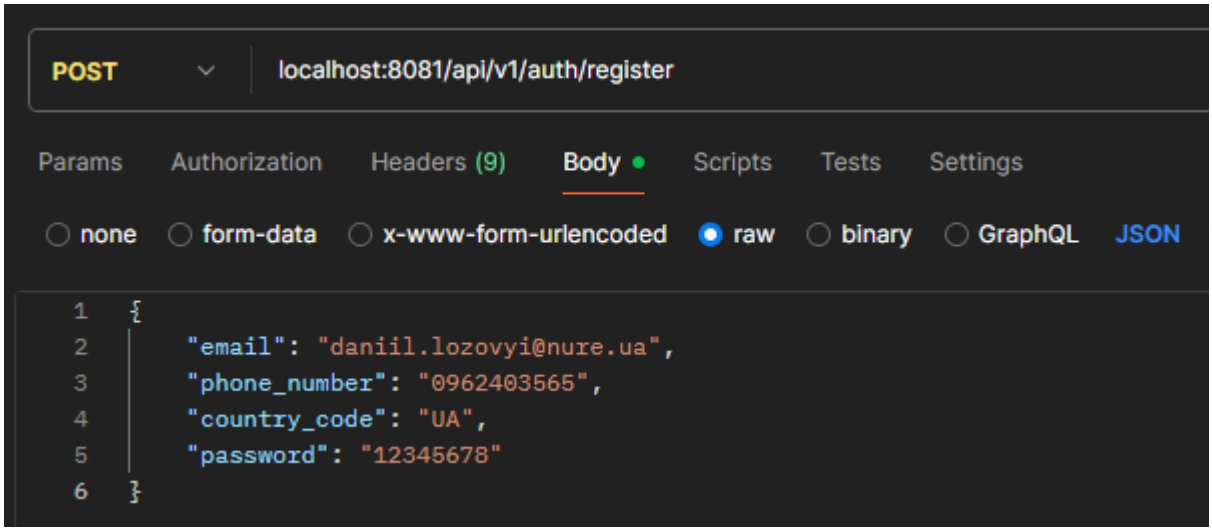

Рисунок 5.1 – Запит на реєстрацію (рисунок виконаний самостійно)

У випадку успіху отримаємо інформацію нового користувача у відповідь, включаючи унікальний ID та мітку часу (див. рис. 5.2) із кодом 201.

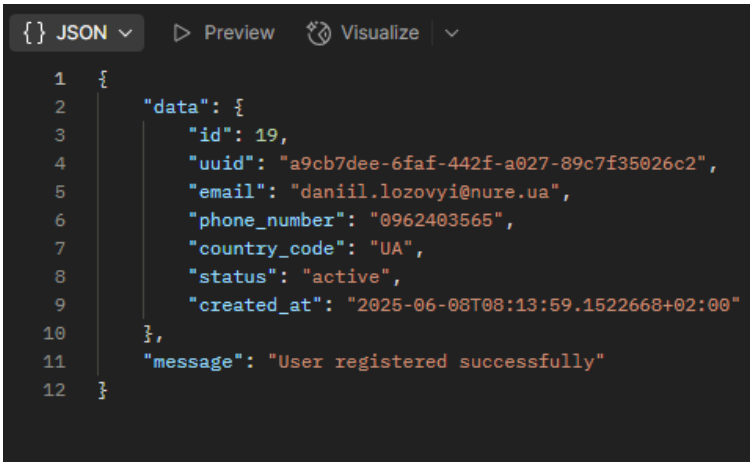

Рисунок 5.2 – Відповідь на успішну реєстрацію (рисунок виконаний самостійно)

У разі невдалої спроби отримаємо у відповідь подробиці помилки (див. рис. 5.3).

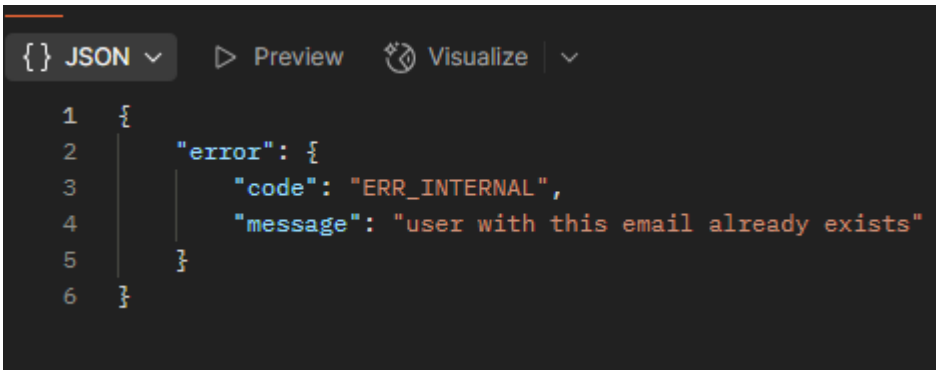

Рисунок 5.3 – Подробиці помилки реєстрації (рисунок виконаний самостійно)

Також, подивившись у таблицю користувачів у базі даних, можемо упевнитися що згенерувались публічний і приватний ключі (див. рис. 5.4), вони знадобляться у подальшому для транзакцій у блокчейні. Приватний ключ зашифрований паролем користувача.

	password_hash character varying (255)	public_key text
1	$2a$10$U82SxyeLIrwdN6n3l4yZgespH5zHPRDnPP5/xu0PoDKvJQ2HUE...	096d67875733f4f0308ffa0628a3a4328c8ea4327176
2	$2a$10$I3wZwiUMxUxcJ0dV8IrEFupfEcNjOg4TM7QgzBCLyDxZLnl3g6x...	1ba9ecb0be5c8fe1243a2b7d04e67bfba9bed041415
3	$2a$10$L9umrBduHakTs5zQUUHWAeo.ysRLg.87XcKgtj5z2184zLtnr3Rz.	b05092cf47ede633bb9fe12014dca43330732b954f22
4	$2a$10$RO9OFu9xeBIlKE34bYl0POBMOlfxevhzM8PrWz9b82DASuH5xYt...	751ee6d1a541045565238a4232a95e51c1b3dbc3d9a
5	$2a$10$8WHYPNkMF.T1B30WchZtU.Q88uLHgMWsnu0oO4QOe4wUoCZ...	8d8ff3a52f55d0e24899d3798b5c5551d0d47795538a
6	$2a$10$dNkmJKcknCO8tF0FtDEsZ.kaiDku/Mg5aiZnXC64q1zFtbD1.KcWK	9ab04154e23989588ddd0615a422315bdac5186373c

Рисунок 5.4 – Публічний ключ користувача (рисунок виконаний самостійно)

Тестування створення голосування – для цього надсилаємо усю необхідну інформацію на задану адресу (див. рис. 5.5). Час вказуємо у UTC +0000.

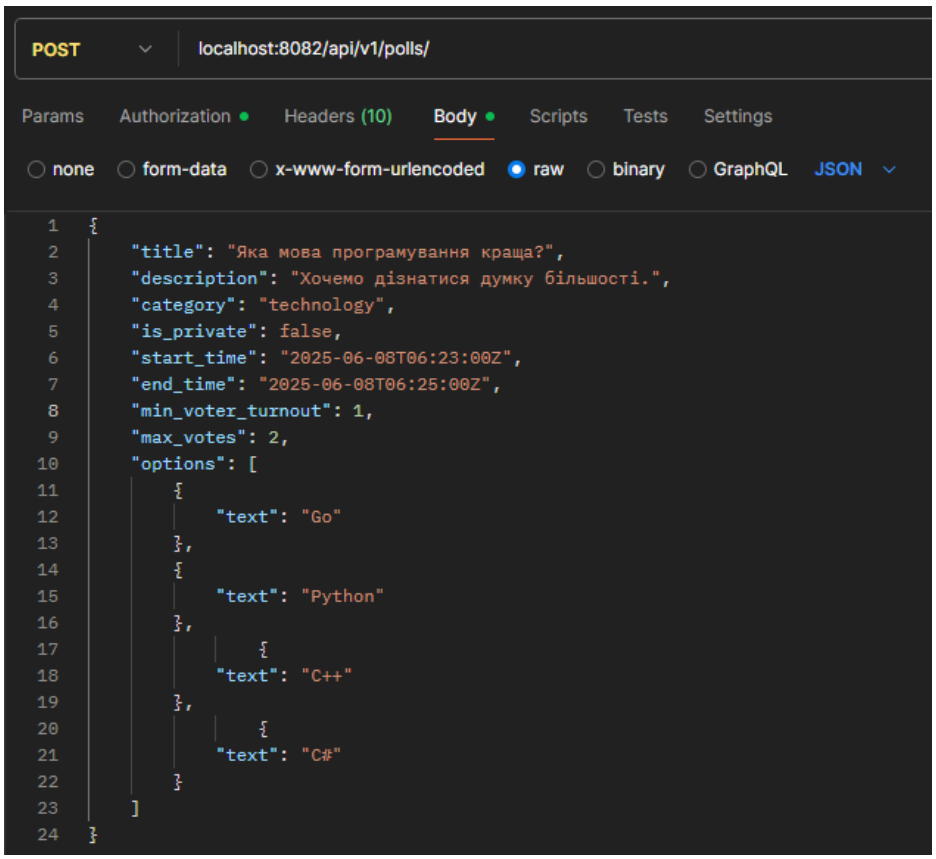

Рисунок 5.5 – Запит на створення голосування (рисунок виконаний самостійно)

Після проходження усіх перевірок, на дійсність часу та категорії, запит надсилається до блокчейну, де проходить повторні перевірки для гарантії вірності даних. Після цього створюється транзакція із згенерованою адресою користувача на основі її публічного ключа. У відповідь отримаємо хеш транзакції, який можемо використати для перевірки в оглядачі блоків.

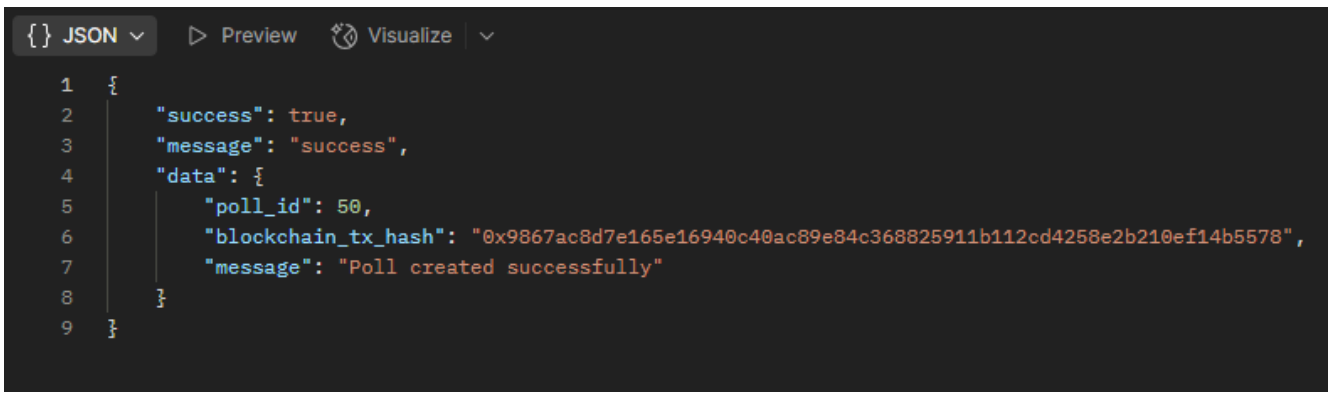

Рисунок 5.6 – Успішна відповідь на створення голосування (рисунок виконаний самостійно)

Тепер можна отримати інформацію про це голосування, як детально так і в числі списку усіх голосувань. Також можна проголосувати за окремий варіант (див. рис. 5.7). Така інформація також записується у блокчейн.

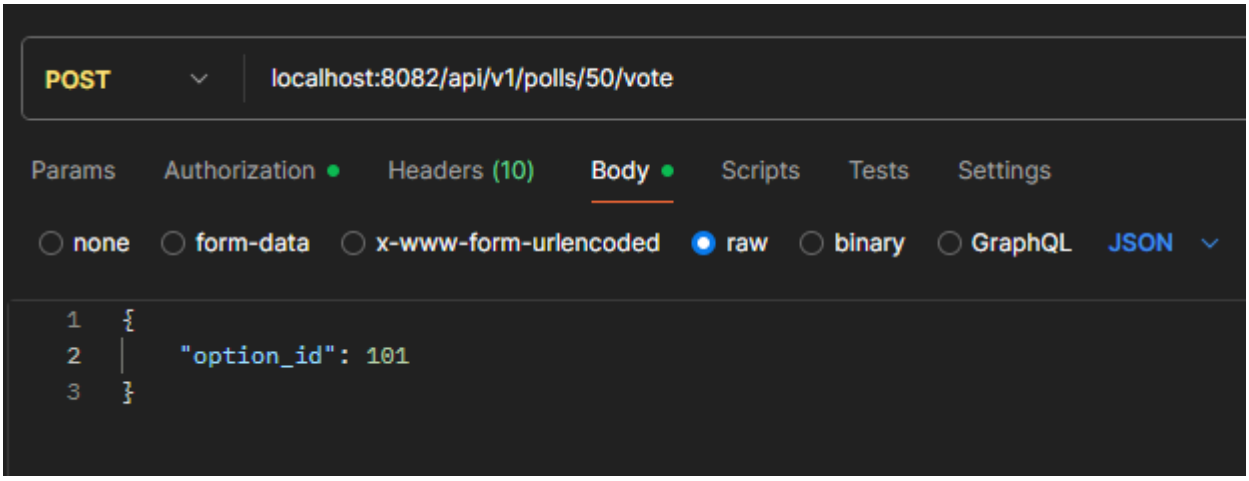

Рисунок 5.7 – Запит на голосування (рисунок виконаний самостійно)

Якщо голосування було відкрите, варіант існував та належав до обраного голосування, а також користувача не було заблоковано, голос приймається (див. рис. 5.8) та записується до блокчейну разом із хешем усіх варіантів цього голосування, щоб гарантувати цілісність даних.

Рисунок 5.8 – Успішна відповідь на голосування (рисунок виконаний самостійно)

Таким чином, можна голосувати допоки не спливе час голосування, або не буде досягнута вказана при створені позначка максимальної кількості голосів.

Після цього, голосування отримає статус «закінчено», результати будуть підраховані та опубліковані. Цим займається спеціальний сервіс, який працює у фоні і кожний заданий час перевіряє усі голосування із статусом «активно» (див. рис.5.9). Коли спливає час, голосування закривається, або закінчується достроково, коли досягається максимальна кількість голосів. У разі якщо голоси не досягли вказаної мінімальної позначки, голосування отримує статус «Провалено».

Рисунок 5.9 – Робота фонового сервісу моніторингу голосувань (рисунок виконаний самостійно)

В результаті у таблиці з результатами можна побачити переможця (див. рис. 5.10).

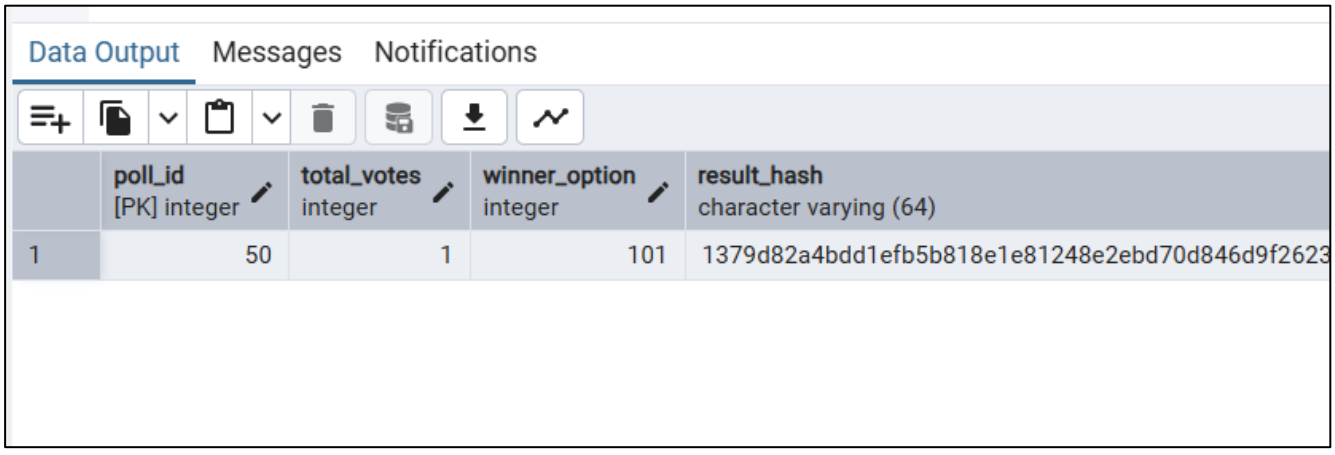

Рисунок 5.10 – Результати голосування (рисунок виконаний самостійно)

Більш детально можна переглянути у таблиці статистики по варіантах голосування (див. рис. 5.11).

	id [PK] integer	poll_id integer	option_id integer	vote_count integer	percentage numeric (5,2)	created_at timestamp without time zone
1	71	50	101	1	100.00	2025-06-08 08:25:00.079304
2	72	50	102	0	0.00	2025-06-08 08:25:00.079304
3	73	50	103	0	0.00	2025-06-08 08:25:00.079304
4	74	50	104	0	0.00	2025-06-08 08:25:00.079304

Рисунок 5.11 – Статистика по варіантах голосування (рисунок виконаний самостійно)

Усі ключові операції та їх метадані зберігаються у блокчейн. Під час тестування було використано мережу Hardhat, для підвищення швидкості та зменшення витрат. При відкритті консолі всередині запущеного вузлу, на якому розгорнуті контракти, можна побачити підтвердження як створення голосування та голосів (див. рис. 5.12), так і фонового збереження результатів (див. рис. 5.13).

```
eth_getTransactionCount
eth_gasPrice
eth_sendRawTransaction
  Contract call:       PollRegistry#registerPoll
  Transaction:         0x9867ac8d7e165e16940c40ac89e84c368825911b112cd4258e2b210ef14b5578
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH
  Gas used:            354580 of 3000000
  Block #24:           0x3e4d7dae4849bd47b7b3081dfab6c84d371e0108dc4939f8a06c7fe318213b4b

eth_call
  Contract call:       VoteVerifier#hasUserVoted
  From:                0x0000000000000000000000000000000000000000
  To:                  0xe7f1725e7734ce288f8367e1bb143e90bb3f0512

eth_getTransactionCount
eth_gasPrice
eth_sendRawTransaction
  Contract call:       VoteVerifier#recordVote
  Transaction:         0xa571f764e9963f9528235dc4a30fa485cb44fee71b5c4c6e3108a3a7862c4895
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
  Value:               0 ETH
  Gas used:            276031 of 3000000
  Block #25:           0x63b065f238d7d9d203f295871cc72ae7ee044efb5b33f53de064773e7560159b

eth_call
  Contract call:       PollRegistry#getPoll
  From:                0x0000000000000000000000000000000000000000
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
```

Рисунок 5.12 – Виклики до контрактів створення голосування та голосу (рисунок виконаний самостійно)

```
eth_getTransactionCount
eth_gasPrice
eth_sendRawTransaction
  Contract call:       PollRegistry#updatePollStatus
  Transaction:         0xa35bd5cbbe73f317e2b74e5c8f43ff463b1d03efb9db44633e792e7ac9352617
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH
  Gas used:            54648 of 3000000
  Block #26:           0x780a83fb6753436f71eab8a09c09c3ec9485332e0a6204918b86eab50b2ba5e4

eth_getTransactionCount
eth_gasPrice
eth_sendRawTransaction
  Contract call:       ResultStorage#storeResults
  Transaction:         0x44b4f59204ae1809dce42d0a2106d9fd5907ed60abebdeea016684d9b83a4025
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x9fe46736679d2d9a65f0992f2272de9f3c7fa6e0
  Value:               0 ETH
  Gas used:            325874 of 3000000
  Block #27:           0x943c548377df6581cd323a50d57b542d3fcd1e1202b8e07efade68b35f8fa201
```

Рисунок 5.13 – Автоматичний виклик оновлення статусу та збереження результату (рисунок виконаний самостійно)

Через переглядач блоків, за збереженим хешем можна продивитись детальну інформацію (див. рис. 5.14).

```
  blockHash: '0x63b065f238d7d9d203f295871cc72ae7ee044efb5b33f53de064773e7560159b',
  index: undefined,
  hash: '0xa571f764e9963f9528235dc4a30fa485cb44fee71b5c4c6e3108a3a7862c4895',
  type: 0,
  to: '0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512',
  from: '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266',
  nonce: 24,
  gasLimit: 3000000n,
  gasPrice: 20000000000n,
  maxPriorityFeePerGas: null,
  maxFeePerGas: null,
  maxFeePerBlobGas: null,
  data: '0xe79ceda20000000000000000000000000000000000000000000000000000000320000000000000000000000000000000000000000000000000
000000000000000000000653d16442b84e131889aa3fb9ed0990001fdfaea681002aac40d88d6810c5473c80000000000000000000000000009cbe8975aef45
be069198f765fee24a4383f8c02',
  value: 0n,
  chainId: 31337n,
  signature: Signature { r: "0xd3e3cff5d5a276429a0889a67114de4be0580b570df44e9231e88ec915fcc819", s: "0x3912488807842c98c569
24c8f8d2d04eb8a7ca02ea49f67b13ffa2d679119761", yParity: 0, networkV: 62709 },
  accessList: null,
  blobVersionedHashes: null
}
undefined
>
```

Рисунок 5.14 – Збережені дані транзакції (рисунок виконаний самостійно)

Всередині data зберігається передана інформація. Наприклад, на рисунку 5.13 можна побачити хеш "3d16442b84e131889aa3fb9ed0990001fdfaea681002aac40d88d6810c5473c8", що відповідає хешу опцій з бази даних.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено детальний аналіз предметної галузі розробки системи голосування, розглянуто можливі проблеми та варіанти їх вирішення за допомогою впровадження блокчейн системи, проаналізовано існуючі рішення на ринку та досвід інших країн. Засновуючись на результатах аналізу було сформовано функціональні на нефункціональні вимоги до майбутнього продукту. Відповідно до вимог розроблено повністю функціональну програмну систему для голосувань, яка складається з бекенд- та блокчейн-частин. Під час роботи встановлено критично важливі нюанси та аспекти, які були проаналізовані на етапі проектування – розроблені UML-діаграми класів, компонентів та послідовностей для ключових алгоритмів роботи. Відповідно до діаграми компонентів розроблено архітектуру, яка складається з 4 мікросервісів та смарт-контрактів у L2-рішеннях на базі Etherium. Засновуючись на розробленій діаграмі класів спроєктовано структуру бази даних, яка налічує 9 таблиць.

Запропоновані архітектурні рішення відповідають всім визначеним функціональним вимогам та створюють надійну основу для подальшої розробки та впровадження системи.

Таким чином, розроблена програмна система дозволяє організувати прозорий та безпечний процес електронного голосування, вирішуючи ключову проблему довіри до результатів. Особливістю запропонованого рішення є гібридна архітектура, що об’єднує переваги блокчейну та класичних бекенд-сервісів. Використання криптографічних механізмів, таких як сліпі підписи та докази з нульовим розголошенням, дозволяє гарантувати анонімність виборців при одночасній можливості верифікації легітимності голосів, що є фундаментальною вимогою для широкого впровадження електронного голосування.

Апробація роботи відбулася у вигляді представлення доповіді на I Міжнародній науково-технічній конференції «Сучасні інформаційні технології та системи штучного інтелекту» MIT@AIS-2025 (див. дод. Д), що засвідчує актуальність і практичну цінність результатів роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. The Fundamentals of Blockchain. Zerocap. URL: https://zerocap.com/insights/research-lab/fundamentals-of-blockchain/ (date of access: 08.06.2025).

2. Introduction to smart contracts. *ethereum.org*. URL: https://ethereum.org/en/developers/docs/smart-contracts/ (date of access: 08.06.2025).

3. Here's how hackers might mess with electronic voting on Election Day. PBS News. URL: https://www.pbs.org/newshour/science/heres-how-hackers-could-mess-with-electronic-voting (date of access: 08.06.2025).

4. The benefits of blockchain voting - Coin Rivet. Coin Rivet. URL: https://coinrivet.com/the-benefits-of-blockchain-voting/ (date of access: 08.06.2025).

5. Лозовий Д., Кириченко І. Використання Блокчейн-технологій для Забезпечення Прозорості Електронного Голосування. Сучасні інформаційні технології та системи штучного інтелекту : матеріали 1ї Міжнар. науково-практ. конф., м. Харків-Яремче, 19–22 трав. 2025 р. Харків, 2025. С. 50–53.

6. Wood A. West virginia secretary of state reports successful blockchain voting in 2018 midterm elections. Cointelegraph. URL: https://cointelegraph.com/news/west-virginia-secretary-of-state-reports-successful-blockchain-voting-in-2018-midterm-elections (дата звернення: 13.05.2025).

7. Secure decentralized application development - follow my vote. Follow My Vote. URL: https://followmyvote.com/ (дата звернення: 13.05.2025).

8. Horizonstate. Horizonstate. URL: https://horizonstate.com/ (дата звернення: 13.05.2025).

9. Vocdoni - digital voting. Vocdoni. URL: https://www.vocdoni.io/ (дата звернення: 13.05.2025).

10. The Myth of “Secure” Blockchain Voting | U.S. Vote Foundation. Register to Vote: Absentee Ballot Request & Voter Registration Services. URL: https://www.usvotefoundation.org/blockchain-voting-is-not-a-security-strategy (date of access: 08.06.2025).

11. Going from bad to worse: from Internet voting to blockchain voting / S. Park et al. Journal of Cybersecurity. 2021. Vol. 7, no. 1. URL: https://doi.org/10.1093/cybsec/tyaa025 (date of access: 08.06.2025).

12. Zero-knowledge proof (zkp) explained | chainlink. Chainlink: The backbone of blockchain. URL: https://chain.link/education/zero-knowledge-proof-zkp (дата звернення: 13.05.2025).

13. Media P. Layer 2 scaling solutions – an overview of polygon, arbitrum, and optimism. Binance. URL: https://www.binance.com/en/square/post/456009 (дата звернення: 13.05.2025).

14. GeeksforGeeks. Sybil attack - geeksforgeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/sybil-attack/ (дата звернення: 13.05.2025).

15. Шаронова Н. В., Терещенко Г. Ю. Проблеми і перспективи практичного застосування інформаційної технології blockchain в smart-контрактах. Інтелектуальні системи та інформаційні технологі (ISIT-2019). – Матеріали Міжн. Наук.-практ. Конф. – Одеса, 19 − 24 серпня 2019 р. – С. 214–219.

16. What is a CronJob? Definition and Explanation - Seobility Wiki. Seobility | Das SEO Tool für Onpage Optimierung. URL: https://www.seobility.net/en/wiki/CronJob (date of access: 08.06.2025).

17. GitHub. NureLozovyiDaniil/2025_B_PI_PZPI-21-9_Lozovyi_D_V. URL: https://github.com/NureLozovyiDaniil/2025_B_PI_PZPI-21-9_Lozovyi_D_V