ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-сайту для проведення електронних голосувань з використанням блокчейн технологій»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Михайло ІЩУК
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Михайло ІЩУК

Керівник: Світлана ШЕВЧЕНКО
к.п.н., доцент

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Іщуку Михайлу Олександровичу

1. Тема кваліфікаційної роботи: «Розробка web-сайту для проведення електронних голосувань з використанням блокчейн технологій»
керівник кваліфікаційної роботи к.п.н., доцент Світлана ШЕВЧЕНКО,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про проблему електронного голосування, програмні рішення для електронного голосування, програмні засоби для розробки web-сайту для проведення електронних голосувань з використанням блокчейн технологій.

4. Зміст розрахунково-пояснювальної записки:
1. Провести аналіз літературних джерел з проблематики електронного голосування, визначити ризики та виклики традиційних і електронних систем голосування, описати можливості застосування блокчейн технологій для забезпечення довіри, безпеки та прозорості.

2. Здійснити огляд та аналіз існуючих платформ електронного голосування визначити їх архітектурні особливості, переваги та недоліки, розглянути приклади успішного застосування в реальних умовах.
3. Сформувати вимоги до розробки web-сайту для проведення електронних голосувань з використанням блокчейн технологій.
4. Визначити програмні засоби для реалізації web-сайту.
5. Розробити web-сайт для проведення електронних голосувань з використанням блокчейн технологій.
6. Провести тестування розробленого web-сайту для проведення голосувань з використанням блокчейн технологій.

5. Перелік графічного матеріалу: *презентація*
   1. Аналіз аналогів
   2. Вимоги до додатку
   3. Програмні засоби реалізації
   4. Діаграма варіантів використання
   5. Діаграми послідовності авторизації
   6. Структура бази даних
   7. Архітектура додатку
   8. Діаграма класів основної логіки
   9. Діаграма класів авторизації
   10. Мапи веб-застосунку
   11. Екранні форми
   12. Демонстрація роботи програми
   13. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження програм аналогів	05.03-13.03.2025	
3	Аналіз засобів реалізації	14.03-20.03.2024	
4	Проєктування архітектури системи	21.03-27.03.2024	
5	Розробка програмного забезпечення	28.03-20.04.2024	
6	Тестування системи	21.04-28.04.2024	

7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Михайло ІЩУК
(підпис)

Керівник
кваліфікаційної роботи _____ Світлана ШЕВЧЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 57 стор., 2 табл., 39 рис., 29 джерел.

Мета роботи – удосконалення забезпечення прозорості, безпеки та достовірності електронного голосування шляхом використання технології блокчейн.

Об'єкт дослідження – процес електронного голосування.

Предмет дослідження – web-сайт для проведення електронних голосувань з використанням блокчейн технологій.

*Короткий зміст роботи***:** В роботі проведено огляд та аналіз існуючих платформ електронного голосування, визначено їх архітектурні особливості, переваги та недоліки. Розглянуто загрози, що виникають при застосуванні централізованих систем електронного голосування, та запропоновано підходи до їх вирішення шляхом децентралізації та використання блокчейну.

Описано архітектуру програмного застосунку та структуру інтерфейсу користувача. Основними технологіями є мова програмування Typescript, бекенд фреймворк Nest.js у поєднанні з об'єктно-реляційною системою для керування базами даних PostgreSQL, бібліотека Web3.js для розробки бекенду додатку. Для розробки фронтенду використовувалася мова програмування TypeScript та фреймворк React у поєднанні з фреймворком Tailwind CSS. Результатом роботи є web-сайт для проведення електронних голосувань з використанням блокчейн технологій.

Цільовою аудиторією додатку є громадяни, які хочуть брати участь у виборах зручно та безпечно.

КЛЮЧОВІ СЛОВА**:** ЕЛЕКТРОННЕ ГОЛОСУВАННЯ, БЛОКЧЕЙН, СМАРТ-КОНТРАКТ, ДЕЦЕНТРАЛІЗАЦІЯ, БЕЗПЕКА, АНОНІМНІСТЬ, WEB-ЗАСТОСУНОК, ПРОЗОРІСТЬ, ETHEREUM, WEB3

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

REST – Representational State Transfer

API – Application Programming Interface

JSON – Java Script Object Notation

JWT – JSON Web Token

VS Code – Visual Studio Code

EVM – Ethereum Virtual Machine

ВСТУП

Право брати участь у виборах є однією з основних засад демократичного суспільства — це невід'ємна свобода кожної людини, гарантована на конституційному рівні. З розвитком сучасних цифрових технологій виникла можливість реалізовувати це право за допомогою електронних засобів, що дало поштовх до появи такого явища, як електронне голосування або е-голосування.

Ця форма участі у виборах базується на використанні технічних інструментів для організації виборчого процесу, проведення голосування та підрахунку результатів, що дає змогу значно підвищити ефективність усієї виборчої системи.

Автоматизація процесів дозволяє мінімізувати витрати, пов'язані з організацією виборів, скорочує потребу в людському ресурсі та знижує ймовірність помилок, які можуть виникнути через людський фактор. Особливо актуальним є впровадження механізмів дистанційного голосування — воно відкриває нові можливості для громадян, які з тих чи інших причин не можуть особисто з'явитися на виборчу дільницю. Завдяки цьому підходу, кожен виборець отримує шанс реалізувати своє право голосу незалежно від географічного положення, перебуваючи у будь-якому зручному для нього місці та обираючи відповідний час.

Інтеграція блокчейн-технологій у виборчий процес відкриває нову епоху прозорості та безпеки. Такий підхід дозволяє створювати цифрові платформи, які захищені від несанкціонованого втручання та забезпечують незмінність даних. Системи е-голосування, що базуються на блокчейні, здатні суттєво зменшити ризики махінацій, фальсифікацій та порушень під час голосування. При цьому гарантованим залишається принцип анонімності виборця. Таким чином, дана тема є актуальною у контексті виборчих процесів демократичного суспільства.

Метою роботи є удосконалення забезпечення прозорості, безпеки та достовірності електронного голосування шляхом використання технології блокчейн.

Для досягнення мети в роботі необхідно вирішити такі завдання:

1. Провести аналіз літературних джерел з проблематики електронного голосування, визначити ризики та виклики традиційних і електронних систем голосування, описати можливості застосування блокчейн технологій для забезпечення довіри, безпеки та прозорості.

2. Здійснити огляд та аналіз існуючих платформ електронного голосування визначити їх архітектурні особливості, переваги та недоліки, розглянути приклади успішного застосування в реальних умовах.

3. Сформувати вимоги до розробки web-сайту для проведення електронних голосувань з використанням блокчейн технологій.

4. Визначити програмні засоби для реалізації web-сайту.

5. Розробити web-сайт для проведення електронних голосувань з використанням блокчейн технологій.

6. Провести тестування розробленого web-сайту для проведення голосувань з використанням блокчейн технологій.

Об'єкт дослідження є процес електронного голосування.

Предметом дослідження є web-сайт для проведення електронних голосувань з використанням блокчейн технологій.

Методи дослідження. Для досягнення поставлених завдань у роботі використовуються наступні методи: системно-структурний аналіз, порівняльний метод, а також методи передачі, зберігання і обробки даних. Для розробки застосунку застосовані сучасні технології: мова програмування Typescript, бекенд фреймворк Nest.js у поєднанні з об'єктно-реляційною системою для керування базами даних PostgreSQL, бібліотека Web3.js для розробки бекенду додатку, для розробки фронтенду використовувалася мова програмування TypeScript та фреймворк React у поєднанні з фреймворком Tailwind CSS.

Наукова новизна полягає у створенні web-сайту для електронного голосування з використанням блокчейну, який дозволить виборцям підвищити довіру до виборного процесу.

Перший розділ дослідження надає загальний огляд платформ для проведення електронних голосувань. У цьому розділі подано аналіз досліджень, присвячених електронним системам голосування та їхньому впливу на довіру виборців і демократичні процеси. Виокремлено ключові технологічні підходи, що забезпечують безпеку, прозорість і доступність голосування, а також окреслено стратегії мінімізації пов’язаних із ними ризиків.

Другий розділ присвячено проєктуванню архітектури майбутнього додатку. Також зроблено моделювання вимог та огляд засобів реалізації.

Третій розділ описує реалізацію програмного застосунку. В даному розділі надається діаграма основних класів, мапи сайту та зображення готового додатку.

Четвертий розділ описує тестування функціоналу застосунку згідно з розроблених функціональних вимог.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

Сучасний світ характеризується стрімким розвитком у галузі інформаційних технологій, що супроводжується швидкою еволюцією електронних виборчих систем. Протягом останніх двох десятиліть цифрове голосування стало дедалі важливішим компонентом демократичних процесів у всьому світі. Воно відіграє ключову роль у забезпеченні доступності голосування, формуванні цифрових виборчих спільнот, гарантуванні безпеки подачі голосів та відстеженні результатів виборів у режимі реального часу, а також дозволяє громадянам брати участь у демократичних процесах з будь-якої точки світу.

1.1 Актуальність

Електронні виборчі системи є одними з найважливіших технологічних нововведень сучасної демократії. Впровадження електронного голосування продовжує стабільно зростати. Станом на 2024 рік приблизно 20% країн світу впровадили певну форму електронного голосування у свої виборчі процеси [1].

Варто зазначити, що особливе місце у світі електронного голосування займають системи інтернет-голосування. Цей метод є одним із найзручніших та найдоступніших, забезпечуючи легкий та швидкий спосіб участі у виборах.

1.2 Загальний аналіз та характеристика електронних систем голосування

Системи електронного голосування дозволяють громадянам брати участь у демократичних процесах, безпечно голосувати та долучатися до виборчих процедур. Варто зазначити, що системи електронного голосування можуть мати різні функції та спеціалізації, наприклад, інтернет-голосування для віддаленої участі, електронні машини для голосування на виборчих дільницях або системи на

основі блокчейну для підвищення прозорості. Зазвичай система електронного голосування - це поєднання апаратного та програмного забезпечення.

Основною цільовою аудиторією систем електронного голосування є громадяни, які хочуть брати участь у виборах зручно та безпечно. Ця аудиторія може бути різною за віком, технічними навичками, місцезнаходженням та потребами в доступності. Вони можуть використовувати систему е-голосування для реалізації своїх демократичних прав, перевірки своїх голосів та отримання підтвердження своєї участі. Для них важливо мати простий і безпечний інтерфейс для автентифікації та голосування, а також можливість перевірити свій голос і зрозуміти процес. Цільовою аудиторією таких систем також можуть бути специфічні групи, такі як виборці за кордоном, люди з інвалідністю або мешканці віддалених районів, які стикаються з проблемами при використанні традиційних методів голосування. Вони можуть використовувати систему для подолання географічних та фізичних бар'єрів на шляху до участі в голосуванні, гарантуючи, що їхні голоси будуть підраховані, зберігаючи при цьому таємницю та цілісність голосування [2].

1.3 Проблема недовіри виборців

У зв'язку з дедалі ширшим впровадженням електронних систем голосування існує потреба в постійному вдосконаленні засобів для безпечного та доступного процесу голосування. Однак існуючі системи електронного голосування стикаються зі значними проблемами у забезпеченні доброчесності виборів та суспільної довіри. Одним із таких критичних факторів є «парадокс безпеки-прозорості». Ця проблема виникає через внутрішню суперечність між збереженням таємниці голосування та забезпеченням прозорості системи, оскільки система повинна одночасно захищати індивідуальні голоси та дозволяти громадську перевірку. Це часто призводить до зростання занепокоєння щодо цілісності системи та можливості маніпуляцій, створюючи те, що експерти називають «розривом довіри».

Це знижує довіру громадськості до систем електронного голосування і все частіше змушує виборчі органи переглядати або відмовлятися від ініціатив щодо електронного голосування через ці побоювання. Наприклад, згідно з дослідженням Європейського парламенту «Digital technology in elections» (2018) [3], хоча системи електронного голосування покращили доступність для певних груп виборців, вони також викликали значне занепокоєння щодо вразливостей безпеки та прозорості системи. У дослідженні підкреслюється, що «впровадження електронного голосування вимагає ретельного врахування факторів безпеки, доступності та суспільної довіри».

1.4 Стратегії забезпечення безпеки та верифікації в електронному голосуванні

Одне з головних завдань системи електронного голосування - забезпечити безпечну та перевірювану обробку голосів. Для цього використовуються різні стратегії обробки голосів та перевірки безпеки. До найпоширеніших стратегій належать:

- індивідуальна верифікація голосу, коли системи використовують криптографічні методи, щоб дозволити виборцям перевірити свій голос, зберігаючи при цьому таємницю голосування;
- групові заходи безпеки, коли голоси обробляються партіями з криптографічними доказами, щоб забезпечити цілісність і водночас захистити індивідуальну конфіденційність;
- моніторинг результатів у режимі реального часу, де системи відстежують моделі голосування та виявляють аномалії в усіх голосуваннях;
- наскрізна верифікація, коли весь процес голосування від голосування до підрахунку голосів перевіряється незалежними спостерігачами

Ці стратегії часто поєднуються для створення комплексного та безпечного досвіду голосування. Однак слід пам'ятати, що однією з головних проблем сучасних систем електронного голосування є баланс між безпекою, доступністю та суспільною довірою.

1.5 Аналіз існуючих рішень, що використовуються для електронного голосування

При аналізі програмного забезпечення для проведення електронних голосувань необхідно врахувати деякі важливі аспекти, а саме:

- безпека і анонімність системи;
- прозорість і довіра;
- незмінність даних
- надійність і відмовостійкість;

Аналіз цих аспектів допоможе побудувати чітку картину для розуміння роботи наявних програмних рішень, їх недоліків та переваг.

1.5.1 i-Voting

Коли мова заходить про електронні системи голосування, неможливо не згадати про естонську систему i-Voting, яка стала піонером інтернет-голосування. Згідно зі статистикою, з моменту запуску у 2005 році система i-Voting в Естонії опрацювала понад 1,3 мільйона голосів, досягнувши піку використання — 44% від загальної кількості голосів на парламентських виборах 2019 року [4].

i-Voting [5] – система інтернет-голосування в Естонії, запущена у 2005 році, є однією з найповніших та найдовше діючих у світі. Вона застосовувалася у численних національних виборах — парламентських, місцевих та виборах до Європарламенту.

Користувачі можуть пройти аутентифікацію (див.рисунок 1.5.1.), використовувати систему з будь-якого пристрою під’єднаного до мережі, проголосувати протягом періоду попереднього голосування (зазвичай 7 днів), змінювати свій голос кілька разів протягом періоду голосування, перевірити, що їхній голос був правильно врахований. скасувати свій голос, якщо вони передумали.

Рис. 1.5.1 Екранна форма аутентифікації i-Voting

Щоб забезпечити надійність та захист системи голосування, було впроваджено низку заходів безпеки, кожен із яких відіграє важливу роль у збереженні цілісності виборчого процесу:

Коли виборець надсилає свій голос, система додає до нього цифровий підпис. Це схоже на унікальну печатку, яка гарантує, що голос дійсно надійшов від того, хто мав право голосувати, і не був змінений.

Кожен голос автоматично отримує позначку часу, коли він був поданий. Це допомагає системі вести точний облік поданих голосів і уникати плутанини з черговістю їх надходження.

Кожен виборець може проголосувати лише один раз. Щоб цього дотримуватись, система перевіряє особу виборця перед тим, як дозволити подати голос.

Якщо хтось спробує проголосувати знову — система це розпізнає і не допустить подвійного голосування.

Після аутентифікації перед користувачем постає список відкритих голосувань (див. рисунок 1.5.2.) при натисканні на голосування голосування кандидат бачить список опцій і обравши опцію може за неї проголосувати.

Рис. 1.5.2 Екранна форма голосування платформи i-Voting

Основними функціями i-Voting є [6]:

- аутентифікація за допомогою національного посвідчення особи;
- використання мобільний ідентифікатор як альтернативний метод аутентифікації;
- доступ до системи з будь-якого пристрою, підключеного до Інтернету;
- голосування протягом періоду попереднього голосування (зазвичай 7 днів)
- можливість зміни голосу кілька разів протягом періоду голосування
- можливість переконатися, що голос користувача був правильно зафіксований

До переваг даної платфоми можна віднести:

- пришвидшення процесу підрахунку голосів;
- мінімізація людських помилок при підрахуванні голосів
- дозволяє багаторазову зміну голосу протягом періоду голосування;
- миттєве підтвердження голосу;
- забезпечує криптографічну перевірку;
- забезпечує наскрізне шифрування;
- пропонує кілька методів аутентифікації;
- дозволяє самостійно перевіряти голоси;

До недоліків даної платфоми можна віднести:

- можлива відсутність довіри громадськості до безпеки голосування та відсутності зовнішнього впливу на результат голосування;
- закритий код частини компонентів.

1.5.2 SwissPost eVoting

SwissPost eVoting [7] - Система електронного голосування SwissPost eVoting розроблена у співпраці з урядом Швейцарії з акцентом на безпеку та перевірюваність. Використовується на кантональних та федеральних голосуваннях з 2019 року.

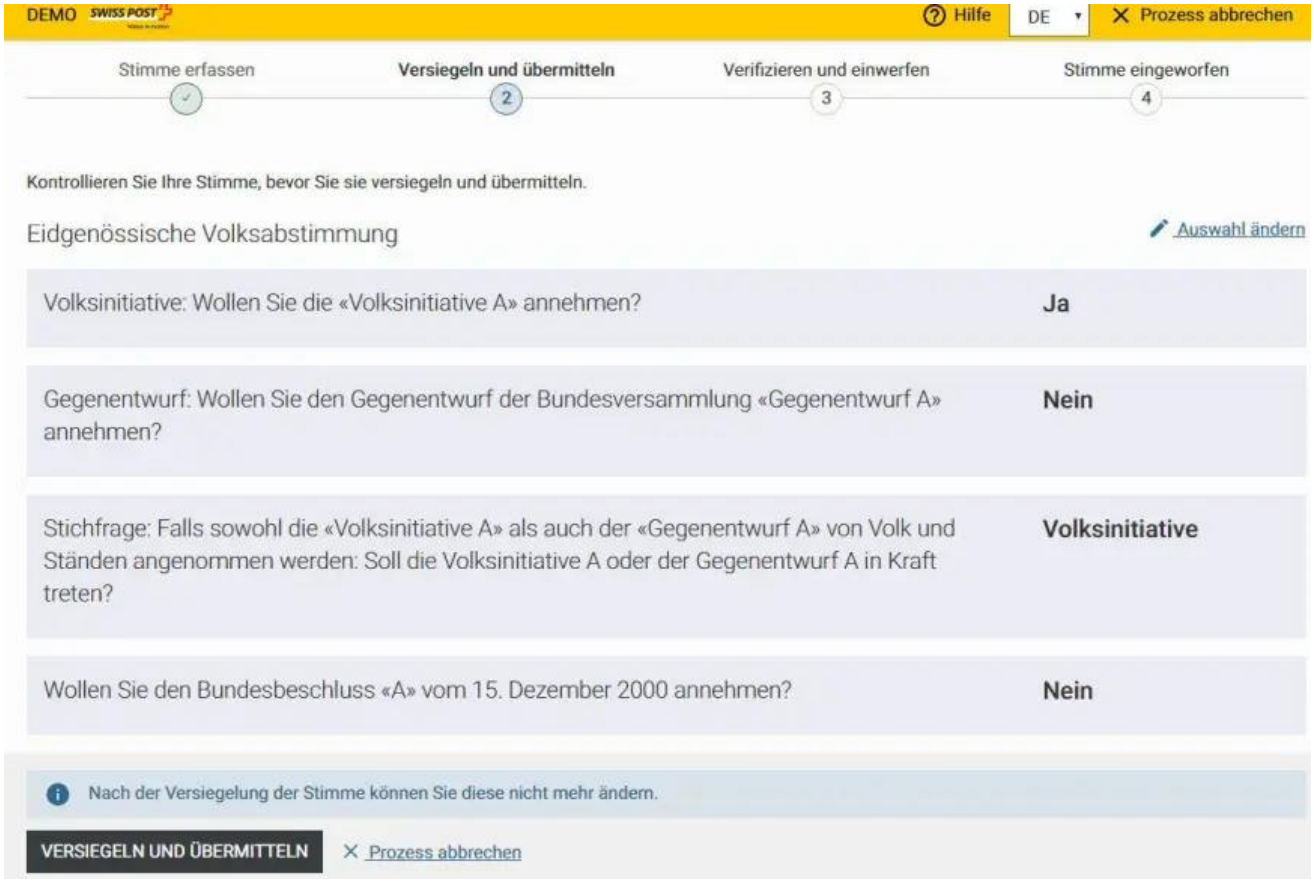


Рис. 1.5.3 Демо-сайт SwissPost eVoting

SwissPost eVote дозволяє громадянам брати участь у різних типах виборів за допомогою захищеної онлайн-платформи. Система надає кілька каналів голосування, де громадяни можуть віддати свої голоси на різних типах виборів, від місцевих муніципальних виборів до федеральних референдумів. Крім того, функція верифікації дозволяє виборцям підтвердити, що їхні голоси були враховані

правильно, зберігаючи при цьому повну анонімність. SwissPost eVote надає громадянам можливість брати участь у виборах дистанційно, незалежно від того, перебувають вони у Швейцарії чи за кордоном. Ці функції допомагають створити більш доступне і зручне середовище для голосування, ніж традиційні методи голосування на паперових носіях. Дані в системі SwissPost eVote обробляються прозоро і піддаються перевірці, що усуває необхідність ручного підрахунку голосів. Це дозволяє виборчим органам систематично перевіряти голоси, що забезпечує прозорість і передбачуваність виборчого процесу. Користувачі можуть скористатися функцією верифікації, щоб підтвердити, що їхній голос був зареєстрований правильно, за допомогою унікального верифікаційного коду. SwissPost eVote відрізняється від більшості систем електронного голосування тим, що використовує наскрізну верифікацію. Процес голосування організований таким чином, що забезпечує повну таємницю, але дозволяє індивідуальну перевірку, показуючи виборцям, що їхні голоси були підраховані належним чином.

SwissPost eVote може працювати з різними типами виборів на основі категорій, таких як муніципальні, кантонні та федеральні вибори, але вона не є універсальною системою голосування в традиційному розумінні, оскільки вона спеціально розроблена для виборчих процесів у Швейцарії.

Переваги SwissPost eVote:

- щоб виявити потенційні уразливості до того, як ними скористається зловмисник, система періодично проходить тестування на проникнення. До процесу залучаються як внутрішні фахівці, так і зовнішні експерти, які діють за сценарієм “етичних хакерів”. Це дозволяє проактивно зміцнювати систему.
- дані, пов'язані з голосами, зашифровуються на різних етапах шляху. Починаючи з моменту, коли виборець робить свій вибір, і закінчуючи їх зберіганням на серверах. Кожен рівень використовує сучасні криптографічні алгоритми, щоб захистити інформацію навіть у випадку часткового компрометування інфраструктури.

- для уникнення концентрації повноважень в одних руках система використовує модель розподіленої довіри (distributed trust model). Це означає, що ключові операції — як-от розшифрування голосів або доступ до серверів — виконуються за участі кількох незалежних сторін. Таким чином, навіть у разі компрометації одного учасника система залишається стійкою.
- частина коду системи доступна публічно, що дозволяє незалежним дослідникам аналізувати її безпеку, повідомляти про вади, а також підвищувати довіру громадськості. Хоча не весь код відкритий (через вимоги безпеки або захисту комерційних інтересів), прозорість у ключових модулях відіграє важливу роль у забезпеченні надійності.

Недоліки:

- попри те, що деякі частини системи відкриті для громадськості, значна частина програмного забезпечення залишається закритою. Це унеможливлює повноцінний аудит усієї системи незалежними експертами. Виборці, фахівці та науковці не мають змоги переконатися, що в коді немає прихованих вразливостей чи бекдорів, що знижує рівень довіри;
- станом на останні роки система була доступна лише в окремих кантонах і для обмеженої кількості виборців. Це частково пов’язано з консервативною політикою швейцарського уряду щодо цифрового голосування, а також з технічними й політичними складнощами розширення;
- у 2019 році під час "public intrusion test" (публічного тесту на злам) дослідники виявили серйозну криптографічну вразливість, яка теоретично дозволяла змінити голос, не порушуючи верифікаційний процес. Хоча цю ваду оперативно виправили, скандал значно вдарив по репутації системи та призвів до тимчасового припинення її використання в деяких кантонах
- через складну криптографічну архітектуру (енд-ту-енд шифрування, подвійне шифрування, протоколи верифікації) впровадження системи є технічно та фінансово витратним. Її інтеграція у виборчу інфраструктуру вимагає підготовленого персоналу, модернізованого обладнання та постійного аудиту, що може бути недоступним для менших регіонів.

Попри високий рівень безпеки, частина громадян і політиків залишається скептичною щодо цифрового голосування загалом. Наявність навіть одного скандалу чи витоку може призвести до хвилі недовіри, що несе політичні ризики і впливає на легітимність виборів. У Швейцарії це призвело до гальмування процесу на кілька років.

1.5.3 Voatz

Voatz [8] - це мобільна платформа для голосування, яка поєднує блокчейн з біометричною аутентифікацією. Використовувалася в пілотних проектах і невеликих виборах у США.

Система Voatz позиціонується як одна з найпрогресивніших у сфері мобільного голосування, і її архітектура розроблена з особливою увагою до кібербезпеки.

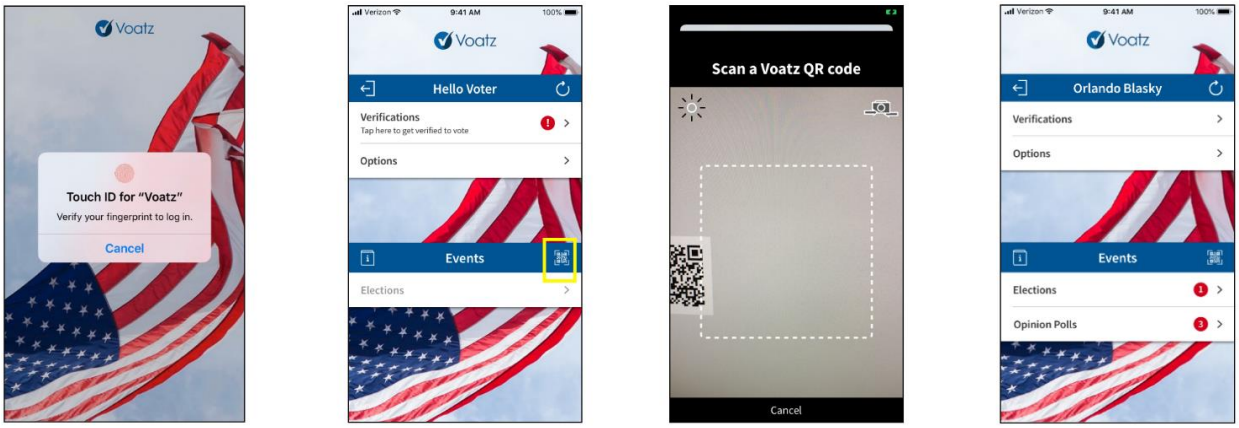


Рис. 1.5.4 Приклади екранних форм додатку Voatz

Voatz використовує одразу кілька рівнів аутентифікації, щоб впевнитися в особі користувача. Це включає:

- біометричну аутентифікацію (відбиток пальця, розпізнавання обличчя, голосова перевірка);
- ідентифікація через документи;
- SMS/електронна пошта.

Однією з центральних особливостей Voatz є зберігання голосів у блокчейні. Це означає, що після запису голосу жоден учасник системи не може змінити або видалити голос без порушення ланцюжка хешів.

Блокчейн дозволяє зберігати незмінний публічний або напівпублічний журнал транзакцій, де зашифровані голоси записуються у захищеному вигляді.

Переваги Voatz:

- Voatz систематично проходить аудити та тести на проникнення (penetration testing), які виконуються незалежними сторонніми компаніями. До того ж проводяться відкриті програми Bug Bounty;
- всі дані зашифровані з використанням сучасних алгоритмів криптографії;
- лише кінцеві точки мають доступ до дешифрування, а сам голос зберігається у зашифрованому вигляді в блокчейні;
- використовується TLS/SSL між клієнтом і сервером для захисту від MITM-атак.

Voatz інтегрує сучасні інструменти кібербезпеки, включаючи блокчейн, багатофакторну аутентифікацію та енд-ту-енд шифрування. Однак, як і будь-яка система, вона не є ідеальною — були критичні зауваження від спільноти безпекових дослідників щодо прозорості та архітектурних рішень. Але з точки зору формальних заходів безпеки, платформа дотримується високих стандартів..

Недоліки Voatz:

- у 2020 році дослідники з Массачусетського технологічного інституту (MIT) опублікували звіт [9], в якому продемонстрували серйозні вразливості в клієнтській частині Voatz.Voatz публічно заперечила висновки, але відмова від відкритого аудиту лише підсилила критику;
- на відміну від систем типу Estonian i-Voting, Voatz не забезпечує прозорої математичної перевірки результатів голосування;
- лише кінцеві точки мають доступ до дешифрування, а сам голос зберігається у зашифрованому вигляді в блокчейні;
- на відміну від відкритих систем Voatz має повністю закриту архітектуру;

– Voatz використовує біометричну автентифікацію (розпізнавання обличчя, відбитки пальців) та особисті документи для реєстрації.Що порушує принцип таємного голосування, оскільки потенційно можливо співвіднести особу з конкретним голосом.

1.5.4 Helios

Helios [10] — це система голосування з відкритим кодом та повною перевірюваністю, розроблена Беном Адідою. Застосовується в академічних та організаційних виборах по всьому світу.

Основною перевагою Helios є не апаратна безпека чи закритість, а максимальна математична прозорість і орієнтація на криптографічну перевірюваність.

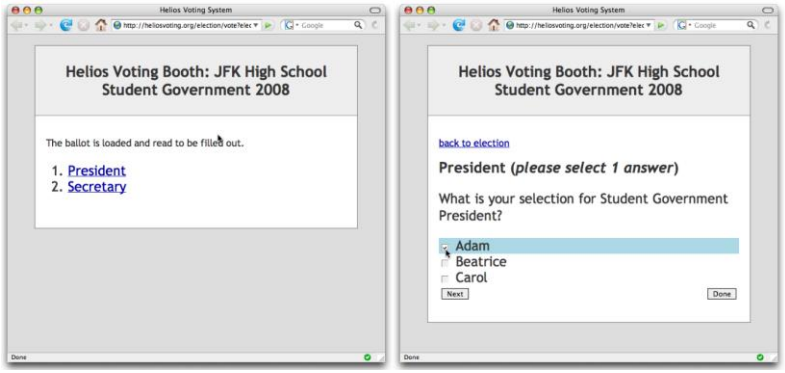


Рис. 1.5.5 Екранна форма голосування Helios

Однією з головних цінностей Helios є end-to-end перевірюваність голосування.Helios використовує сучасні криптографічні методи для забезпечення шифрування голосів на стороні клієнта.Кожен голос супроводжується математичним доказом того, що він є коректним, не розкриваючи його змісту. Гомоморфне шифрування: дозволяє обчислювати результати виборів без розшифровки окремих голосів, зберігаючи конфіденційність.

Helios створена так, щоб будь-який охочий (виборець, дослідник, сторонній спостерігач) міг завантажити всі зашифровані бюлетені, разом з відповідними доказами, перевірити коректність криптографічного підрахунку та запустити власну перевірку результатів без участі розробників системи.

На відміну від багатьох рішень, які потребують апаратної безпеки (смарт-карт, апаратних токенів, спеціалізованих сканерів), Helios працює повністю у веб-браузері.

Переваги:

- повна прозорість системи;
- публічна перевірка коду;
- відкритий вихідний код
- покращення, ініційовані спільнотою;
- незалежний аудит безпеки;
- налаштовується під різні потреби.

Недоліки:

- чудово підходить для студентських асоціацій, внутрішніх опитувань чи голосувань у невеликих організаціях. Проте коли кількість виборців вимірюється сотнями тисяч або мільйонами, система стикається із суттєвими технічними та організаційними викликами.;
- фокусується на перевірці цілісності та підрахунку голосів, але не забезпечує самостійну систему надійної ідентифікації користувачів;
- гарантує анонімність виборців у ході підрахунку, але сама реєстрація та отримання доступу до голосування часто відбувається через централізовані сервіси.

1.5.5 Зведені результати аналізу

Зведені результати аналізу характеристик розглянутих платформ наведено у таблиці 1.5.6.

Таблиця 1.5.6

Зведені результати аналізу характеристик СЕГ

Показник	**Estonia i-Voting**	**SwissPost eVoting**	**Voatz**	**Helios**	**Майбтуній додаток**
Аутентифікація виборця	ID-картка	Набір кодів + криптографічний підпис	Біометрія + номер мобільного телефону	Email/логін	Ethereum адреса + Криптографічний підпис
Цілісність голосу	Так	Так	Так	Так	Так
Можливість верифікації виборцем	Так	Так	Обмежена	Обмежена	Так
Анонімність голосування	Так	Так	Частково	Частково	Так
Використання блокчейну	Ні	Ні	Так	Ні	Так

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

2.1 Моделювання вимог

Проаналізувавши особливості роботи існуючих СЕГ, було сформульовано функціональні вимоги до майбутнього програмного продукту:

- користувачі можуть увійти в систему;
- користувач може проголосувати лише один раз у межах одного голосування;
- результат голосування записується в блокчейн для забезпечення незмінності;
- перегляд результатів після завершення голосування;
- забезпечити проведення голосування в онлайн режимі.

Для спрощення реалізації було обрано англійську мову локалізації, так як це є основною мовою спілкування в мережі Інтернет.

2.2 Огляд засобів реалізації web-сайту для проведення електронних голосувань з використанням блокчейн технологій

2.2.1 Технологія блокчейн

Блокчейн — це інноваційна технологія зберігання даних, яка базується на принципах децентралізації, прозорості та захисту від підробок. У найпростішому вигляді блокчейн є послідовним ланцюгом блоків, у яких містяться записи транзакцій або інших даних. Ці блоки організовані у хронологічному порядку, і кожен новий блок додається до ланцюга лише після проходження механізму консенсусу, що забезпечує узгодженість даних між усіма учасниками мережі [11].

Кожна транзакція в мережі перевіряється децентралізованою групою учасників, відомих як вузли або майнери (залежно від механізму консенсусу, наприклад, Proof of Work або Proof of Stake). Лише після підтвердження більшістю таких учасників транзакція потрапляє до блоку та стає частиною блокчейну.

Однією з ключових характеристик блокчейну є його розподілена структура. На відміну від традиційних централізованих баз даних, у блокчейні копії всього

ланцюга зберігаються одночасно на багатьох вузлах по всій мережі. Це забезпечує високий рівень надійності та стійкості до атак чи збоїв, оскільки жоден окремий учасник не має повного контролю над системою [12].

Кожен блок у блокчейні має унікальний криптографічний хеш, який створюється на основі вмісту блоку. Крім цього, блок також містить хеш попереднього блоку, утворюючи таким чином нерозривний ланцюг. Завдяки цьому механізму будь-яка спроба змінити дані в одному з попередніх блоків призведе до невідповідності хешів у наступних блоках, і система автоматично відкине змінений блок як недійсний [13]. Вузли мережі зможуть виявити фальсифікацію, оскільки збережений хеш не відповідатиме фактичному вмісту блоку.

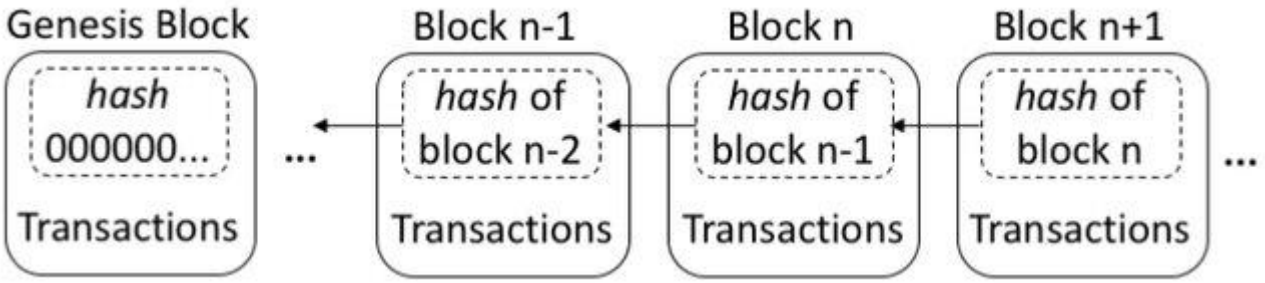


Рис. 2.1.1 Ланцюг блокчейн [14]

Як вже було зазначено раніше, для забезпечення цілісності та достовірності даних у блокчейн-мережі використовується механізм консенсусу — алгоритм, який дозволяє незалежним учасникам мережі (вузлам) досягати згоди щодо дійсності транзакцій і порядку їхнього додавання до ланцюга. Завдяки цьому механізму система може функціонувати без централізованого управління, залишаючись безпечною та узгодженою [15].

Тип механізму консенсусу може відрізнятися залежно від конкретної блокчейн-платформи. Найбільш відомими є:

- Proof of Work (PoW) — механізм, при якому майнери змагаються у розв’язанні складних криптографічних задач. Перший, хто знайде правильне рішення, отримує право додати новий блок до блокчейну та винагороду у вигляді криптовалюти. Цей підхід використовується, зокрема, у Bitcoin і

забезпечує високу безпеку, але потребує значних обчислювальних ресурсів [11].

- Proof of Stake (PoS) — більш енергоефективна альтернатива PoW. У цій моделі учасники блокчейну (валідатори) вносять заставу у вигляді криптовалюти, і ймовірність бути обраним для створення наступного блоку пропорційна розміру їхньої ставки. Цей метод зменшує витрати енергії та підвищує масштабованість. Прикладом реалізації є Ethereum 2.0 [16].

Після успішного проходження консенсусу блок додається до блокчейну та стає незмінною частиною загального журналу транзакцій. Усі дані у блокчейні є публічно доступними (в публічних мережах), що дозволяє будь-кому перевірити правильність транзакцій. Такий рівень прозорості та верифікації зміцнює довіру до системи й забезпечує високу підзвітність.

Однією з перспективних сфер застосування блокчейну є електронне голосування, де особливо важливими є безпека, прозорість і неможливість фальсифікацій. Кожен голос у такій системі розглядається як транзакція, яка після подання:

- криптографічно захищається (хешується);
- додається до блоку;
- стає частиною незмінного ланцюга транзакцій;

Оскільки дані зберігаються одночасно на багатьох вузлах мережі, жодна окрема організація чи суб’єкт не може змінити результати голосування без згоди більшості учасників. Це практично унеможливлює маніпуляції результатами, забезпечуючи високу надійність та довіру до виборчого процесу [17].

Завдяки своїм властивостям блокчейн відкриває нові горизонти для реалізації доброчесних та перевірених систем голосування, особливо в країнах із низьким рівнем довіри до традиційних виборчих структур.

2.2.2 Можливості мережі блокчейн Ethereum

Блокчейн Ethereum пропонує потужну та гнучку інфраструктуру, яка підтримує децентралізовані застосунки та виконання складних смарт-контрактів.

Незважаючи на наявні виклики, властиві блокчейн-технологіям — зокрема проблеми масштабованості, швидкості транзакцій та енергоефективності — Ethereum зробив значний прогрес завдяки останнім оновленням та інноваціям.

Історично Ethereum стикався з обмеженнями, подібними до інших блокчейнів першого покоління: високі комісії за транзакції (газ) та низька пропускна здатність у періоди високого навантаження. Ці проблеми були вирішені завдяки запуску Ethereum 2.0 — масштабному оновленню, спрямованому на покращення масштабованості, безпеки та сталого розвитку. Основні інновації Ethereum 2.0 — це перехід до механізму консенсусу Proof of Stake (PoS) та модульна масштабованість через шардинг.

На відміну від Proof of Work (PoW), який потребує великих обчислювальних ресурсів і споживає багато енергії, Proof of Stake (PoS) значно зменшує навантаження на мережу. Валідатори обираються для створення та підтвердження блоків на основі кількості заблокованих (stake) монет ETH, а не обчислювальної потужності. Це не лише зменшує споживання енергії, але й сприяє децентралізації та безпеці, оскільки знижує бар’єр для участі [16].

Щоб подолати обмеження масштабованості, Ethereum планує впровадити шардинг — технологію поділу блокчейну на менші паралельні ланцюги (шарди), кожен з яких обробляє власну частину даних та транзакцій. Завдяки цьому Ethereum зможе обробляти тисячі транзакцій на секунду (TPS), порівняно з поточними 15–30 TPS на основному ланцюзі без шардингу.

Однак до повного впровадження шардингу Ethereum вже підтримує Layer 2-рішення, зокрема Optimistic Rollups та ZK-Rollups, які об’єднують декілька транзакцій поза мережею і передають їх як одну стиснуту операцію. Це дозволяє значно зменшити вартість і час виконання транзакцій, роблячи Ethereum придатним для використання в навантажених системах, таких як голосування або фінансові сервіси [18].

Ethereum забезпечує конфіденційність, автентичність і цілісність даних завдяки використанню потужних криптографічних алгоритмів. Подібно до мережі NEAR, Ethereum використовує еліптичну криптографію (ECC) — зокрема криву

secp256k1 — для створення ключів, цифрових підписів і адрес гаманців. Кожна транзакція підписується приватним ключем і перевіряється за допомогою відповідного публічного ключа. Це гарантує, що лише авторизовані сторони можуть ініціювати транзакції чи взаємодіяти з контрактами [12].

Крім того, Ethereum підтримує розширені механізми шифрування через віртуальну машину Ethereum (EVM), що дозволяє розробникам реалізовувати додаткові рівні криптографічного захисту в межах смарт-контрактів. Це робить Ethereum придатним для застосунків, що вимагають захисту особистих даних, цифрової ідентифікації та конфіденційного зберігання голосів.

Ethereum використовує механізм фінальності Casper FFG, який забезпечує остаточне підтвердження блоків без можливості їх зміни чи скасування, якщо тільки більшість валідаторів не діятиме зловмисно в координації. Це критично важливо для таких сфер, як електронне голосування, де потрібно забезпечити цілісність результатів.

Таким чином, Ethereum є зрілою, перевіреною на практиці платформою, яка забезпечує масштабованість, сильні криптографічні основи та активну спільноту розробників. Перехід до Proof of Stake, Layer 2-рішення та майбутній шардинг роблять його потужним варіантом для побудови безпечних, прозорих та масштабованих застосунків на основі блокчейну — зокрема в таких чутливих сферах, як електронне голосування та цифрове врядування.

2.2.3 Технології розробки бекенду

Для реалізації серверної частини додатку для голосування було обрано середовище виконання Node.js. Однією з ключових переваг Node.js є його орієнтована на події, неблокуюча модель введення/виведення, яка забезпечує високу ефективність при обробці одночасних з'єднань та реального часу. Це робить Node.js особливо придатним для застосунків, які потребують високої масштабованості та обробки даних у реальному часі. Node.js підтримує асинхронне програмування, що дозволяє ефективно обробляти декілька операцій без

блокування головного потоку. Node.js широко використовується у таких сферах, як вебзастосунки, системи реального часу та мікросервісна архітектура.

Node.js є одним із найпопулярніших середовищ виконання для серверної розробки. Згідно з опитуванням розробників Stack Overflow 2024 року, Node.js залишається найчастіше використовуваною веб-технологією — 40.8% розробників повідомили про її використання [19]. Також, за даними звіту GitHub State of the Octoverse, Typescript продовжує бути в топ 3 найпопулярніших мов програмування [20]. Однією з переваг вибору Node.js є велика кількість документації, широка екосистема пакетів через npm, а також широке поширення в сучасних серверних рішеннях.

Щоб спростити процес розробки серверної частини, було обрано NestJS — один із найнадійніших та найуніверсальніших фреймворків для Node.js. Він високо цінується у розробці корпоративних застосунків завдяки надійності, модульності, впровадженню залежностей (DI) та простоті інтеграції з іншими технологіями. NestJS, як прогресивний та модульний фреймворк, сприяє модульності застосунку та надає широкий набір функцій для веб- та мобільних застосунків. NestJS використовує принцип впровадження залежностей (DI), що дозволяє винести конфігураційні налаштування та управління залежностями з логіки додатку. Це сприяє модульності та забезпечує повторне використання коду.

NestJS [21] забезпечує вбудовану підтримку TypeScript, що дозволяє використовувати статичну типізацію та краще організовувати код. NestJS також надає вбудовану підтримку таких функцій, як валідація, логування та обробка помилок.

Для зберігання та керування офчейн даних було використано PostgreSQL та ORM Prisma, а також Web3.js для взаємодії з блокчейном.

PostgreSQL — це сучасна система управління базами даних з відкритим кодом, яка поєднує надійність, масштабованість і потужний функціонал. Вона підтримує як традиційні реляційні моделі, так і об'єктні розширення, дозволяючи ефективно працювати зі складними типами даних [22]. PostgreSQL легко

масштабується для роботи з великими обсягами даних та забезпечує високу надійність і цілісність даних.

Як ORM для PostgreSQL було обрано Prisma. Prisma надає типобезпечний клієнт до бази даних, що дозволяє уникати помилок під час виконання та підвищує продуктивність розробника. Серед функцій — автоматичні міграції, керування схемою та потужний API для запитів [23]. Типобезпека Prisma та інтуїтивний API спрощують роботу з базою даних і знижують ймовірність помилок.

Для написання смарт-контракту було використано мову Solidity[24]. Solidity — це об’єктно-орієнтована, високорівнева мова програмування, спеціально створена для написання смарт-контрактів, які виконуються на віртуальній машині Ethereum (EVM).

Для взаємодії з блокчейном Ethereum було обрано бібліотеку Web3.js. Вона надає повний набір інструментів для створення децентралізованих застосунків, включаючи взаємодію зі смартконтрактами, керування акаунтами та обробку транзакцій [24]. Web3.js забезпечує надійне та ефективне з'єднання з блокчейном, дозволяючи застосунку використовувати переваги децентралізованих технологій.

Для реалізації безпеки було використано JWT (JSON Web Tokens) для аутентифікації та авторизації. JWT забезпечує захищений та безстанний спосіб керування сесіями користувача та контролю доступу до API [25]. У системі також реалізовано додаткові заходи безпеки, такі як обмеження частоти запитів (rate limiting), валідація введених даних та захист CORS, щоб забезпечити безпечну роботу застосунку.

Інфраструктура серверної частини була контейнеризована за допомогою Docker, що забезпечує сталі середовища розгортання та спрощує процеси розробки та розгортання [26]. Docker-контейнери гарантують, що застосунок буде однаково працювати у різних середовищах, а також спрощують масштабування та супровід системи.

2.2.4 Технології розробки фронтенд

Мову програмування TypeScript було обрано для реалізації клієнтської частини застосунку для голосування. Однією з ключових переваг TypeScript є статична типізація, яка дозволяє виявляти помилки ще на етапі компіляції та покращує якість коду. Це робить TypeScript добре придатною для розробки масштабованих застосунків, які потребують високої підтримуваності та надійності. TypeScript підтримує об'єктно-орієнтований підхід до програмування, що дозволяє організовувати код у вигляді класів та інтерфейсів. TypeScript використовується в різних сферах, зокрема у розробці вебзастосунків, мобільних і десктопних додатків.

Для спрощення процесу розробки клієнтської частини було обрано React — одну з найнадійніших та найуніверсальніших JavaScript-бібліотек для створення користувацьких інтерфейсів. React високо цінується в розробці сучасних вебзастосунків завдяки своїй надійності, компонентній архітектурі та легкості інтеграції з іншими технологіями. React як декларативна та ефективна бібліотека сприяє модульності застосунків та надає потужний набір функцій для створення інтерактивних інтерфейсів. React базується на компонентному підході до розробки, що дає змогу створювати універсальні та зручні в підтримці елементи інтерфейсу. Такий підхід забезпечує хорошу структурованість проєкту та дозволяє повторно використовувати код у різних частинах застосунку.

Однією з ключових сильних сторін React є його здатність легко масштабуватися, що стає можливим завдяки використанню компонентної архітектури та вбудованим можливостям, що робить його однією з найпотужніших бібліотек для фронтенд-розробки [27]. Наприклад, React надає віртуальний DOM, що дозволяє ефективно оновлювати та рендерити інтерфейс користувача. Компонентна архітектура бібліотеки забезпечує просту інтеграцію додаткових функцій, а система керування станом спрощує управління даними застосунку. React також підтримує такі функції, як hooks, context та обробка помилок.

Однією з важливих переваг React є його модульна структура, яка дає змогу розробникам підключати тільки ті компоненти, що потрібні для реалізації конкретного функціоналу. Крім того, React полегшує роботу з залежностями

завдяки інтегрованому менеджеру пакетів, що автоматично обробляє створення та інтеграцію необхідних об’єктів у застосунок.

Для стилізації та дизайну було використано Tailwind CSS — CSS-фреймворк, орієнтований на утиліти. Tailwind CSS надає набір попередньо визначених утилітних класів, які дозволяють швидко створювати й кастомізувати інтерфейси. Він підтримує адаптивний дизайн, темну тему, а також має плагіни для розширення функціоналу [28]. Утилітарний підхід Tailwind CSS спрощує роботу зі стилями та зменшує ймовірність помилок.

Як інструмент збирання для фронтенд-застосунку було обрано Vite. Vite забезпечує швидке та ефективне середовище розробки, що дозволяє оперативно створювати прототипи та проводити тестування. Серед його можливостей — гаряча заміна модулів, яка дозволяє миттєво оновлювати інтерфейс під час розробки [29]. Швидкий і зручний процес збирання Vite полегшує роботу з застосунком і знижує ймовірність виникнення помилок.

Клієнтська інфраструктура контейнеризована за допомогою Docker.

2.2.5 Середовище розробки

У процесі створення клієнтської частини системи було обрано Visual Studio Code (VS Code) як основне середовище для розробки. Це сучасний інструмент, що поєднує в собі легкість, високу продуктивність та багатий набір функцій, необхідних для ефективної роботи над веб-застосунками. Його простий, але функціональний інтерфейс дозволяє зосередитися на розробці без зайвих налаштувань, а підтримка розширень відкриває широкі можливості для адаптації під будь-який стек технологій.

Однією з ключових переваг VS Code є його глибока інтеграція з Docker, що дозволяє створювати ізольовані середовища розробки безпосередньо з редактора. Такий підхід спрощує процес тестування, знижує ризик конфліктів залежностей та забезпечує стабільну роботу застосунку у будь-якому середовищі.

Docker виступає основою інфраструктури цього проєкту, дозволяючи упаковувати програмні компоненти разом із усіма необхідними залежностями в

компактні, ізольовані контейнери. Такий підхід гарантує, що система працює однаково в різних середовищах — від локальної розробки до продакшену. Контейнери забезпечують високу гнучкість у масштабуванні, швидкий процес розгортання, а також спрощують підтримку та оновлення застосунку.

Завдяки Docker, клієнтська частина, бекенд і база даних працюють в окремих середовищах, що дозволяє уникнути перехресного впливу та полегшує виявлення помилок. Контейнеризована інфраструктура також підвищує стабільність релізів, мінімізує людський фактор при розгортанні та робить систему більш стійкою до змін.

2.3 Проектування структури бази даних

В проекті буде використовуватись об'єктно-реляційна система керування базами даних PostgreSQL для збереження офчейн даних.

На рисунку 2.3.1 представлено основні таблиці бази даних, які будуть використовуватися у додатку.

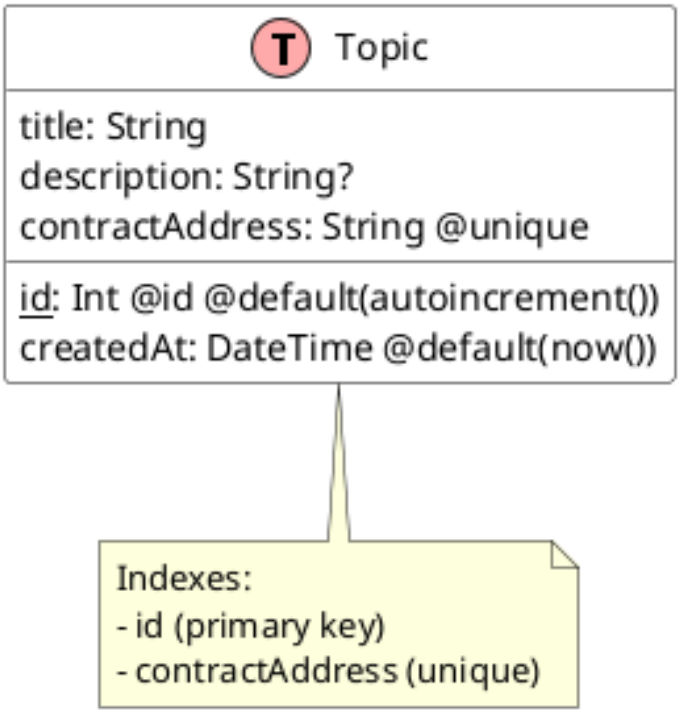


Рис. 2.3.1 Структура бази даних

Таблиця Topic зберігає основні дані про тему для голосування, назва теми, опис теми, унікальна адреса смарт-контракту, порядковий id теми, дату створення. Це було використано для того, аби frontend мав високу швидкість відклику і швидко підвантажував теми для голосувань. Всі інші дані будуть зберігатися у блокчейні.

2.4 Проектування архітектури

Ключовим елементом в системі є браузер користувача, де працює клієнтський застосунок, створений з використанням бібліотеки Web3.js. Web3.js виконує функцію посередника між застосунком і блокчейн-мережею, надаючи зручний API для формування та надсилання запитів.

Ці запити використовують протокол JSON-RPC — компактний та ефективний механізм віддаленого виклику функцій, що забезпечує стандартизовану взаємодію з вузлом Ethereum (Ethereum node) (див.рисунок 2.4.1).

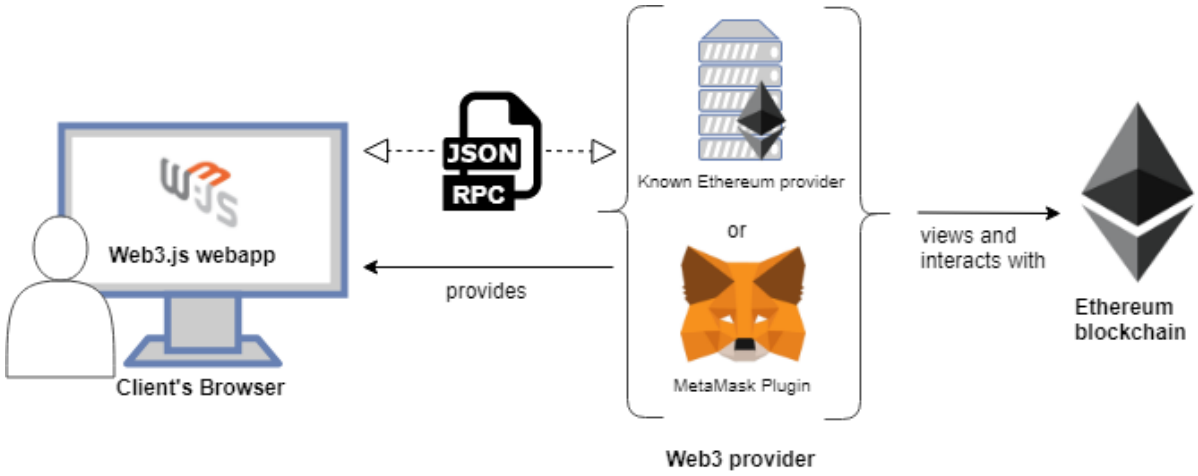


Рис. 2.4.1 Приклад архітектури блокчейн додатку

Використання Web3.js разом з TypeScript дає змогу створювати децентралізовані додатки (DApp) з підвищеною надійністю завдяки статичній типізації. TypeScript дозволяє визначати типи змінних та параметрів, що значно спрощує розробку, знижує ризик помилок під час виконання та полегшує підтримку коду. Така інтеграція забезпечує зручну та безпечну роботу з Ethereum API, що особливо важливо у сфері блокчейн-розробки.

Web3-провайдер є невід'ємною ланкою, що забезпечує взаємодію між браузерним додатком користувача та блокчейном Ethereum. У проєктах на кшталт електронного голосування, які базуються на блокчейн-технологіях, коректна інтеграція такого провайдера є визначальною для безпеки, ефективності й зручності користування системою. Завдяки Web3-провайдеру додаток може надсилати транзакції, зчитувати інформацію з блокчейну, підписувати дані та працювати зі смарт-контрактами, що дозволяє реалізувати повноцінну децентралізовану логіку.

Додаток використовуватиме трирівневу архітектуру, яка зображена на рисунку 2.4.2. Клієнт-серверна трирівнева архітектура є різновидом клієнт-серверної моделі, яка, окрім основних двух описаних вище рівнів, виділяє ще один додатковий рівень - рівень даних.

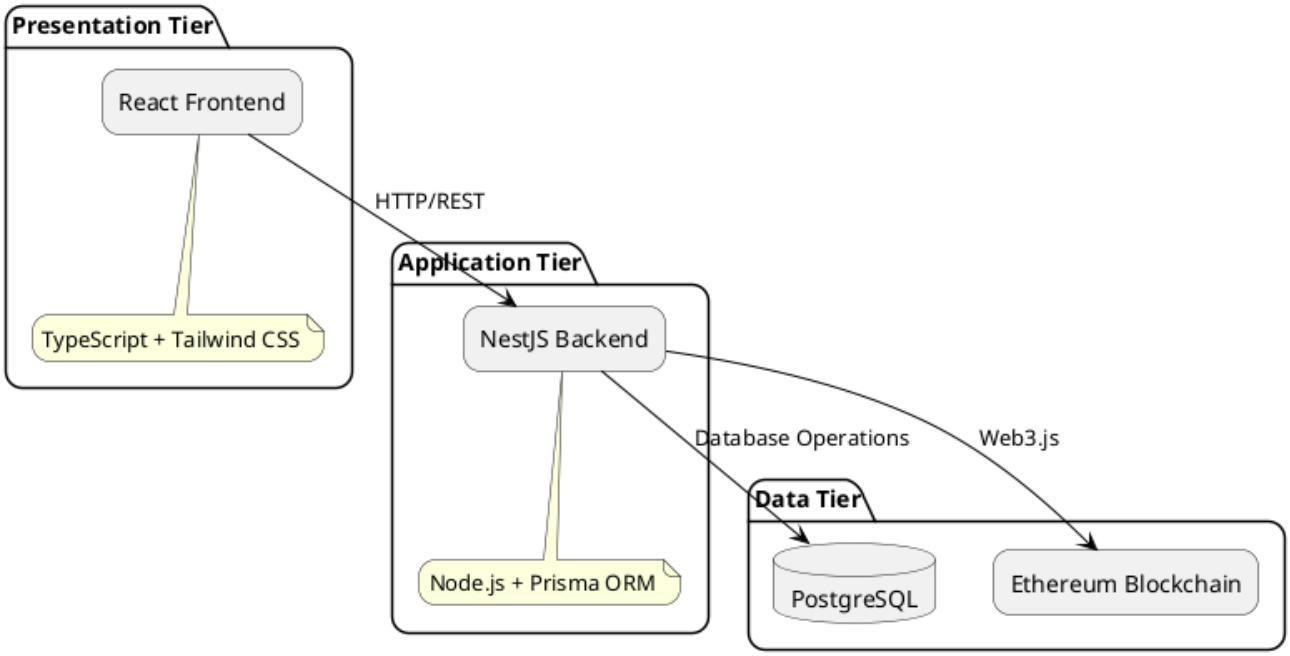


Рис. 2.4.2 Трирівнева архітектура

Загалом, додаток для голосування реалізує трирівневу архітектуру, яка чітко розділяє завдання системи на окремі рівні.

Рівень презентації, реалізований як односторінковий застосунок на основі React, слугує основним інтерфейсом для виборців та адміністраторів. Цей рівень побудований з використанням TypeScript для забезпечення типобезпеки та Tailwind

CSS для стилізації, що забезпечує сучасний і адаптивний користувацький досвід. Фронтенд включає два основні інтерфейси: панель виборця для участі в голосуваннях і адмін-панель для керування темами голосування. Рівень презентації обробляє підключення гаманця, відображає теми голосування з поточним статусом, а також надає зворотний зв'язок у режимі реального часу при голосуванні. Комунікація з бекендом здійснюється через REST API, а автентифікація користувачів забезпечується через інтеграцію з Web3-гаманцем.

Рівень бізнес-логіки, реалізований за допомогою сервісів NestJS, формує ядро серверної логіки застосунку. Цей рівень відповідає за обробку всіх операцій голосування, керування життєвим циклом тем, а також координує взаємодію між фронтендом і блокчейном. Сервіси обробляють критичні операції, такі як створення тем, валідація голосів і керування правами голосування. До рівня бізнес-логіки також входить спеціальний сервіс BlockchainService, який керує всіма взаємодіями з мережею Ethereum через Web3.js, включаючи деплой смарт-контрактів, відправлення голосів та отримання результатів. Цей рівень гарантує дотримання бізнес-правил та забезпечує цілісність і безпеку системи

Рівень даних складається з двох систем зберігання: реляційної бази даних PostgreSQL для збереження позаблокчейнових даних, таких як інформація про теми та користувачів, і блокчейну Ethereum — для запису незмінних транзакцій голосування та керування смарт-контрактами.

Таке розділення відповідальностей забезпечує модульність, підтримуваність і масштабованість додатку, а інтеграція технології блокчейн надає необхідний рівень безпеки та прозорості у процесі голосування.

Рівень доступу до даних, реалізований за допомогою Prisma ORM, керує всіма операціями з базою даних і забезпечує типобезпечний інтерфейс для збереження даних. Цей рівень включає модель Topic, що представляє теми голосування у базі даних, і виконує всі CRUD-операції для управління цими темами. Рівень доступу до даних спроектовано для безперебійної роботи з PostgreSQL, що забезпечує ефективне зберігання та отримання даних при збереженні їх цілісності. Крім того, цей рівень взаємодіє з блокчейном через рівень

бізнес-логіки, забезпечуючи узгодженість даних на ланцюгу та поза ним. Комбінація Prisma ORM і PostgreSQL забезпечує надійну основу для керування даними, а інтеграція з блокчейном гарантує незмінність та прозорість записів голосування.

Загальну архітектуру додатку було спроектованого у вигляді діаграми, як показано на рисунку 2.4.3.

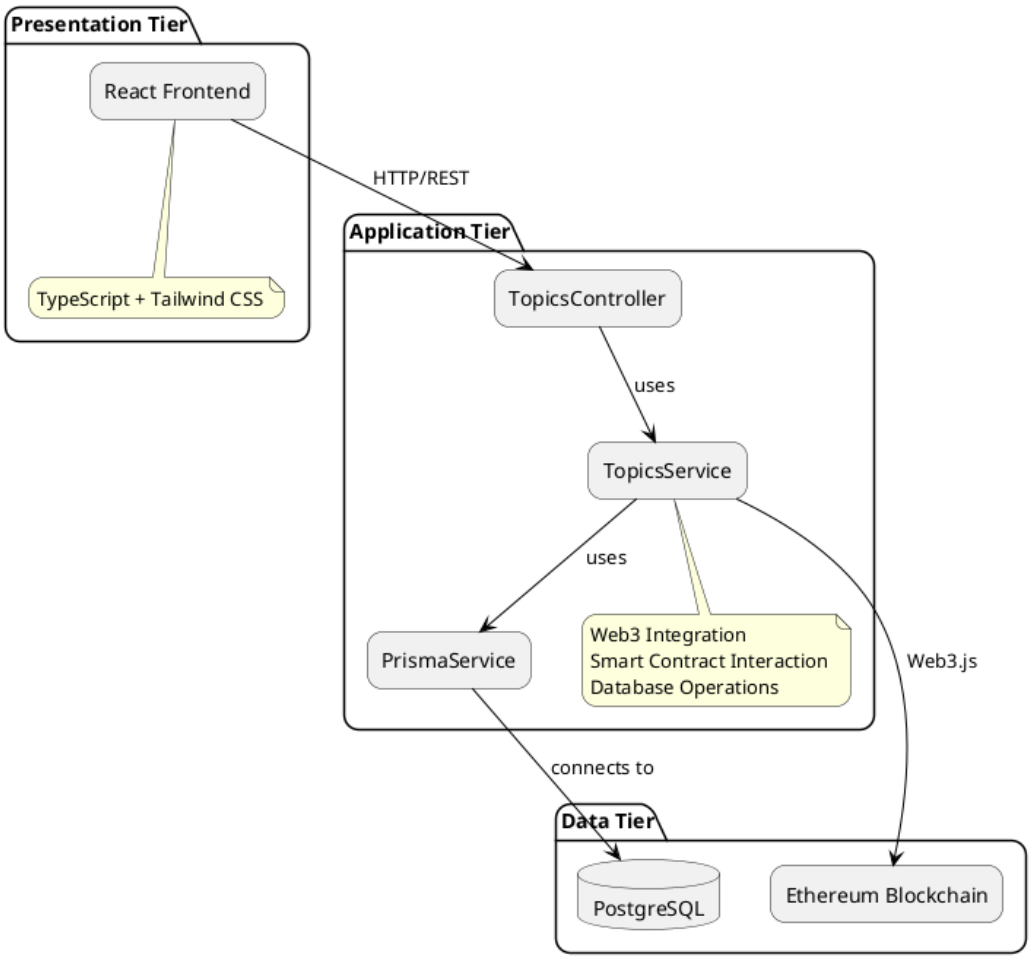


Рис. 2.4.3 Загальна структура архітектури

2.5 Інтерфейс

2.5.1 Інтерфейс користувача

Інтерфейс користувача буде складатись з наступних сторінок:

- сторінка авторизації;

- сторінка перегляду існуючих голосувань;
- сторінка детального перегляду теми для голосуваннння;
- сторінка профілю;
- сторінка з повідомленням що ресурсу не знайдено;

Для виборців автентифікація здійснюється через інтеграцію з гаманцем MetaMask. Неавторизовані користувачі мають доступ лише до головної сторінки входу (/), де вони можуть підключити свій Ethereum-гаманець для участі в голосуванні. Після автентифікації виборці отримують доступ до панелі голосування (/dashboard), де можуть переглядати активні теми голосування та брати в них участь, а також переглядати деталі тем (/topic/:contractAddress) для голосування.

2.5.2 Інтерфейс адміністратора

Інтерфейс адміністратора буде складатись з наступних сторінок:

- сторінка авторизації;
- сторінка перегляду існуючих голосувань;
- форма створення нової теми;
- форма додавання виборців до голосування;
- сторінка з повідомленням що ресурсу не знайдено;

Адміністратори мають окремий процес аутентифікації з використанням API-ключів. Сторінка входу адміністратора (/admin) є публічною, але панель адміністратора (/admin/dashboard) доступна лише для автентифікованих адміністраторів. Адміністратори можуть керувати темами голосування, створювати нові опитування та керувати правами виборців.

3 ОПИС РЕАЛІЗАЦІЇ СИСТЕМИ

Клієнт частина, Серверна частина та база даних PostgreSQL розроблялись в ізольованих контейнерах Docker як повноцінні функціональні одиниці і взаємодіяли один з одним через порти. Деталі реалізації представлені в коді.

3.1 Реалізація блокчейн частини

Основна логіка смарт-контракту Ballot (див. рисунок 3.1.1) організована навколо ключових структурних компонентів: структур Voter та Proposal та набору функцій для управління процесом голосування. Контракт Ballot призначений для реалізації безпечного механізму електронного голосування, придатного для навантажених сценаріїв без делегування голосів.

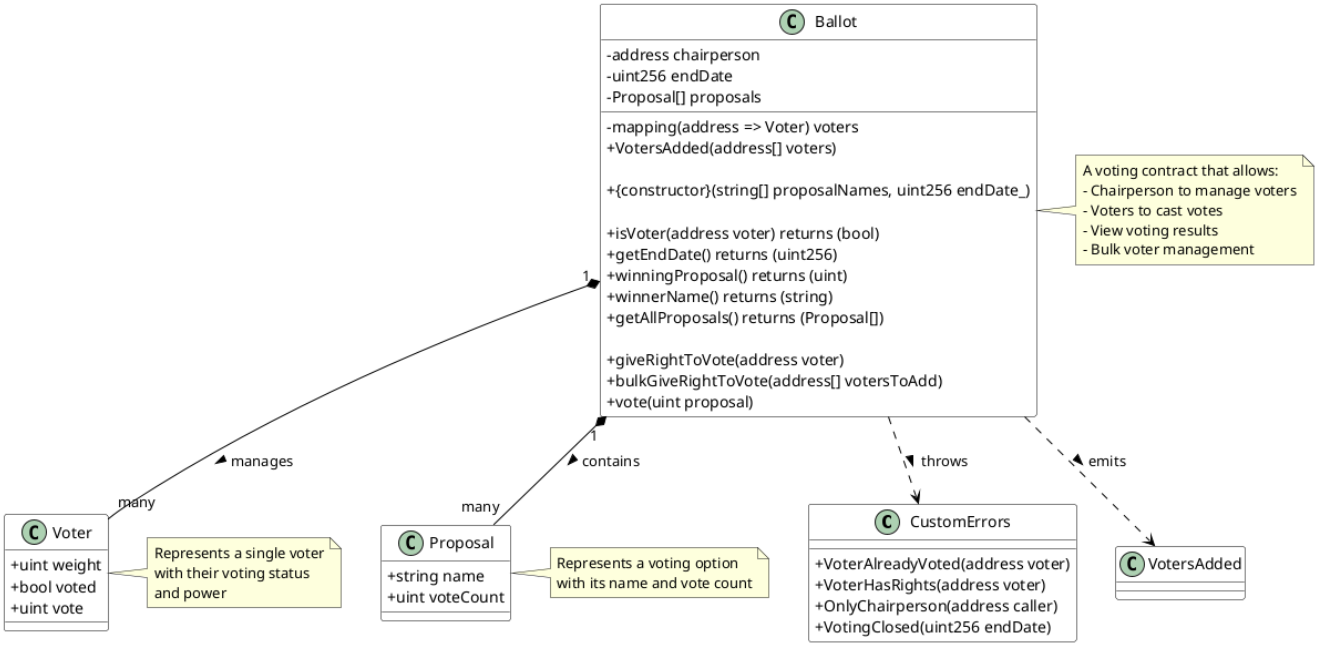


Рис. 3.1.1 Функції та структури смартконтракту

Структура Voter зберігає інформацію про виборця, включаючи вагу голосу, ознаку участі у голосуванні та індекс обраної пропозиції. Структура Proposal представляє окремі варіанти для голосування та підраховує кількість отриманих голосів.

Змінна chairperson визначає адміністратора голосування. Ця роль має виключний доступ до адміністративних дій, таких як надання права голосу. Для реалізації контролю доступу контракт визначає користувацькі помилки: OnlyChairperson, VoterAlreadyVoted, VoterHasRights та VotingClosed.

Відображення voters зв'язує кожну адресу з відповідним записом Voter, тоді як масив proposals динамічно зберігає всі створені пропозиції, ініціалізовані у конструкторі. Конструктор Ballot також встановлює кінцеву дату голосування (endDate) і надає початкову вагу голосу голові.

Права голосу призначаються за допомогою функцій giveRightToVote та bulkGiveRightToVote. Ці функції гарантують, що лише голова може надавати право голосу, і що жоден виборець не отримає права повторно або після вже здійсненого голосування. Функція vote реалізує основну логіку голосування. Вона перевіряє право виборця на участь, дотримання строків голосування та реєструє голос. Голоси підсумовуються по кожній пропозиції шляхом збільшення лічильника голосів з урахуванням ваги виборця.

Допоміжні функції, такі як winningProposal, winnerName та getAllProposals, забезпечують доступ до результатів голосування. Функція winningProposal виконує лінійний пошук для виявлення пропозиції з найбільшою кількістю голосів. Функція winnerName повертає назву переможної пропозиції, тоді як getAllProposals надає повну інформацію про всі варіанти голосування та кількість голосів за кожен.

3.2 Реалізація сервісної логіки

Реалізацію бізнес-логіки представлено на діаграмі класів основної логіки (див. рисунок 3.2.1).

Освновним сервісом є **TopicsService** (Лістинг 3.2.1) він інкапсулює шар взаємодії з блокчейном, керуючи підключеннями Web3 та операціями зі смарт-контрактами. Він зберігає три ключові приватні властивості: екземпляр Web3 для зв'язку з блокчейном, об'єкт облікового запису для підпису транзакцій та екземпляр контракту Multicall для пакетних операцій. Сервіс ініціалізується параметрами конфігурації з змінних середовища, включаючи кінцеву точку Infura, що являється Web3 провайдером та приватний ключ адміністратора.

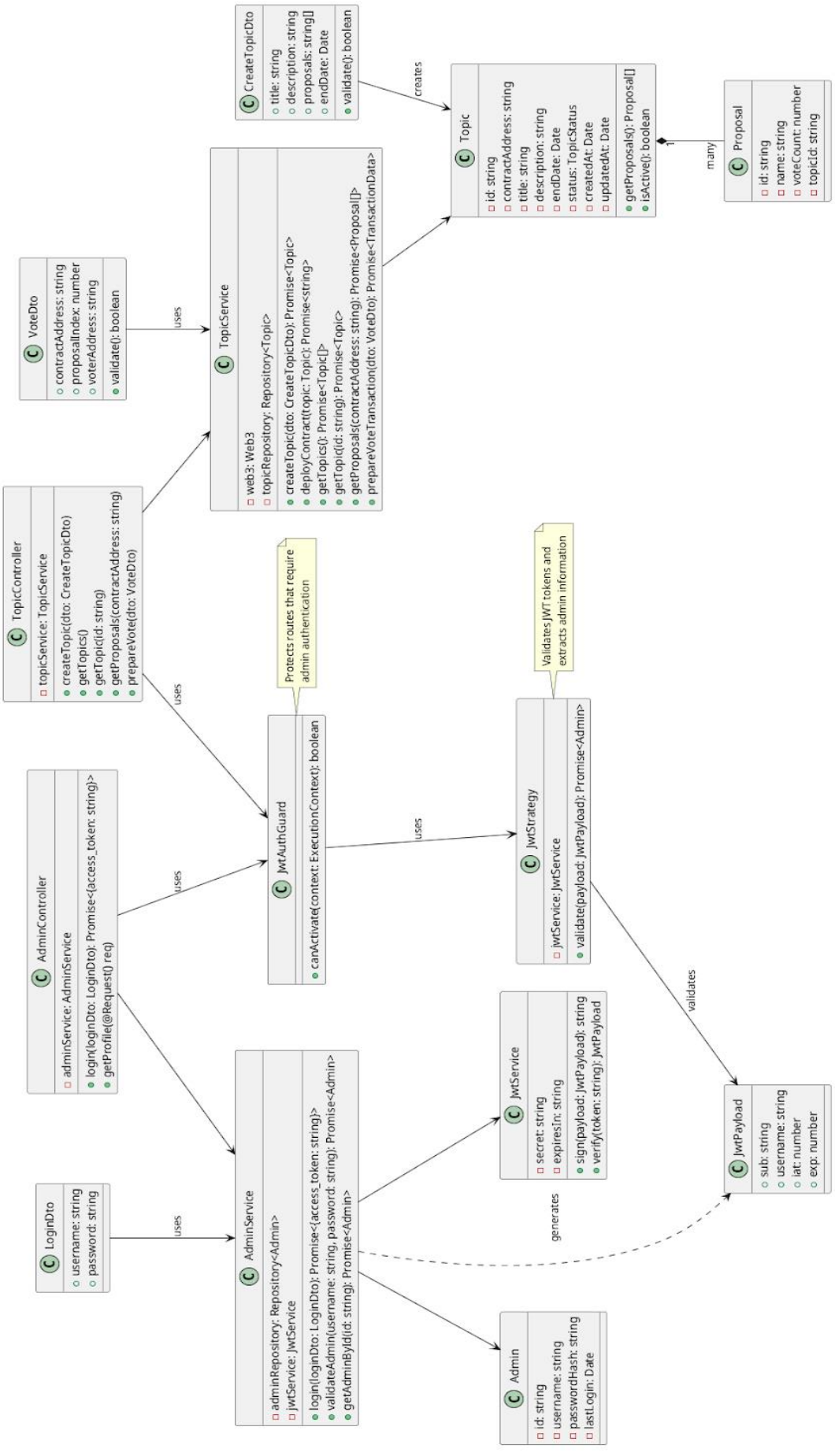


Рис. 3.2.1 Діаграма класів основної логіки

Метод createTopic організовує розгортання нових контрактів Ballot. Він перевіряє адміністративний доступ через верифікацію API-ключа, розгортає контракт із заданими параметрами та зберігає адресу контракту в базі даних. Метод використовує оцінку витрат газу та підпис транзакцій для безпечного розгортання.

Методи giveVotingRight і bulkGiveVotingRights використовуються для надання можливості голосування виборцям. Вони реалізують адміністративні функції та обробляють як індивідуальну, так і масову видачу дозволу на голосування для виборців. Методи включають оцінку витрат газу та підписування транзакцій для безпечного виконання.

Методи getVoteData, getProposals та getTopics відповідають за реалізацію основної функціональності голосування та отримання даних.

Важливим також є онтроллер TopicsController. Він реалізує шар REST API, надаючи доступ до наведених вище операцій через HTTP-ендпоінти.

На діаграмі 3.2.2 наведена логіка авторизації користувачів, та адміністраторів.

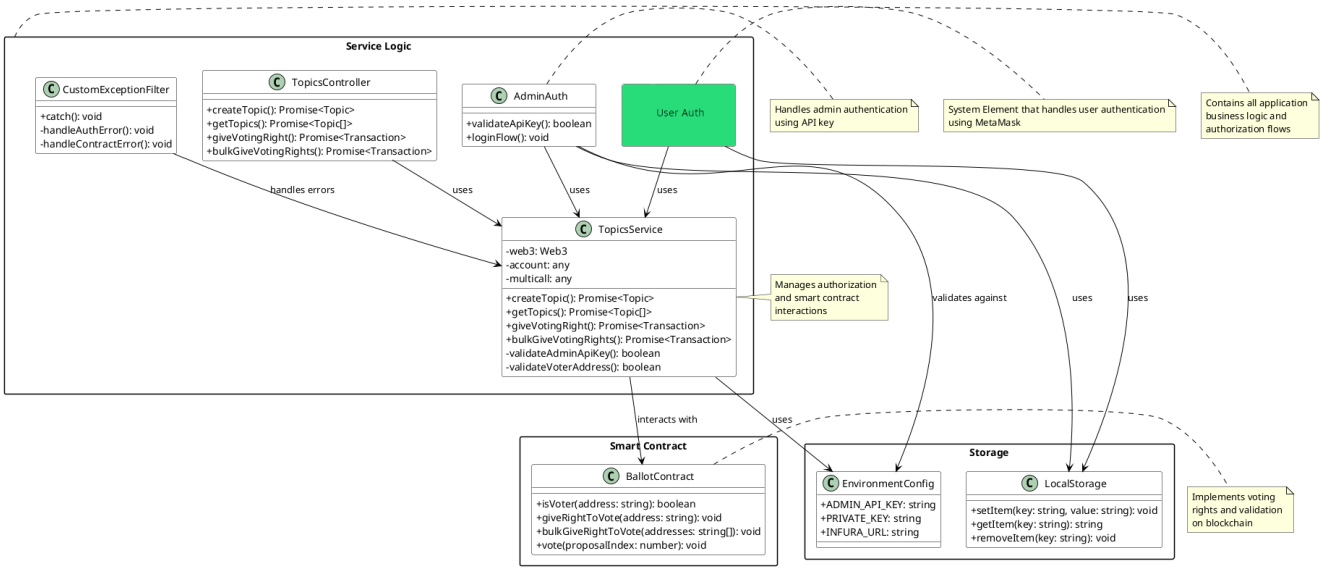


Рис. 3.2.2 Логіка авторизації

Класс AdminAuth здійснює автентифікацію для адміністративних дій за допомогою заздалегідь визначеного API-ключа, збереженого у конфігурації. До його задач входить перевірка наданого адміністративного API-ключа відповідно до

значень. Взаємодія з TopicsService для авторизації привілейованих дій, таких як керування правами виборців.

Компонент UserAuth керує автентифікацією кінцевих користувачів через MetaMask або сумісний Ethereum-гаманець. Цей компонент ініціює процес підключення гаманця через MetaMask, Отримує публічну Ethereum-адресу користувача, валідує формат або наявність адреси за допомогою TopicsService.

3.3 Реалізація рівня презентації

Рівень презентації представляє собою повноцінний Web-додаток, та складається з клієнт частини та проміжного програмного забезпечення(middleware) Ethereum гаманця Metamask. Структуру клієнтського рівня, що взаємодіє безпосередньо з користувачем за допомогою web-сторінок представлено мапою сайту для користувача наведено на рисунку 3.3.1.

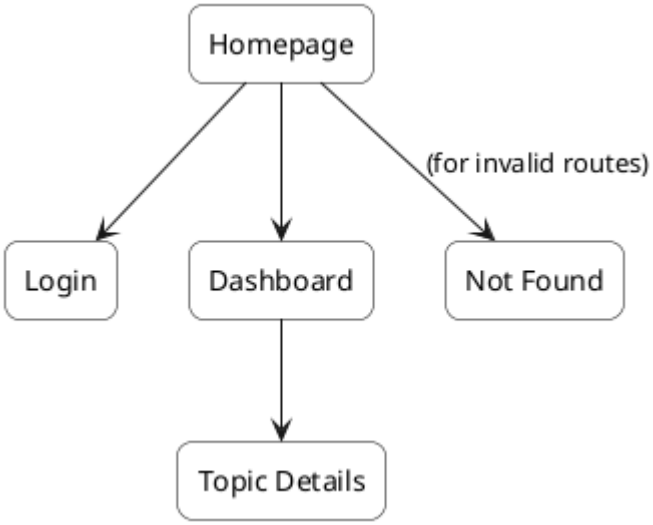


Рис. 3.3.1 Мапа сайту для користувача

Усі сторінки представлені на мапі сайту для користувача є приватними (окрім Homepage).

Структуру клієнтського рівня, що взаємодіє з адміністратором за допомогою web-сторінок представлено мапою сайту для адміністратора.

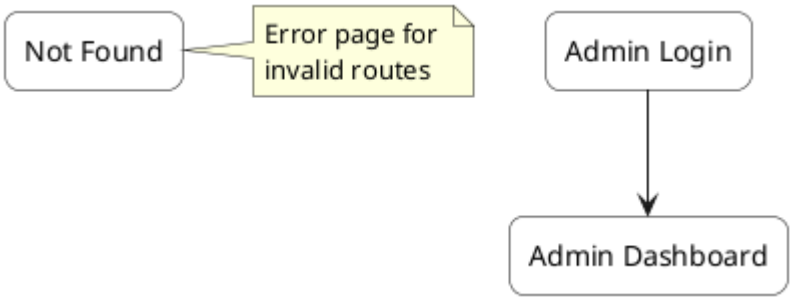


Рис. 3.3.2 Мапа сайту для адміністратора

Клієнт частина отримує усі запити, відправлені на домен, і, за необхідності перенаправляє користувача (або адміністратора) на сторінку з написом, що даний ресурс відсутній.

Рівень сервісної логіки у застосунку для голосування виступає як основний проміжний шар між інтерфейсом та смарт-контрактами блокчейну. Коли користувач або адміністратор взаємодіє з веб застосунки — наприклад, входить у систему, переглядає теми або голосує — сервісна логіка приймає ці запити, обробляє автентифікацію та авторизацію, а потім за потреби взаємодіє з блокчейном або базою даних.

Наприклад:

- Коли користувач входить у систему через MetaMask, логіка сервісу перевіряє адресу його гаманця та перевіряє його право голосу через смарт-контракт;
- Коли адміністратор входить за допомогою API-ключа, логіка сервісу валідує цей ключ і надає доступ до адміністративних функцій;
- Коли здійснюється голосування або створюється нова тема, логіка сервісу обробляє запит, виконує необхідні перевірки та взаємодіє зі смарт-контрактом для запису цієї дії в блокчейн.

3.4 Зовнішній вигляд рівня презентації

Після відкриття застосунку користувач перенаправляється на інтерфейс аутентифікації, де йому пропонується здійснити вхід для подальшого доступу до функціоналу системи (див. рисунок 3.4.1).

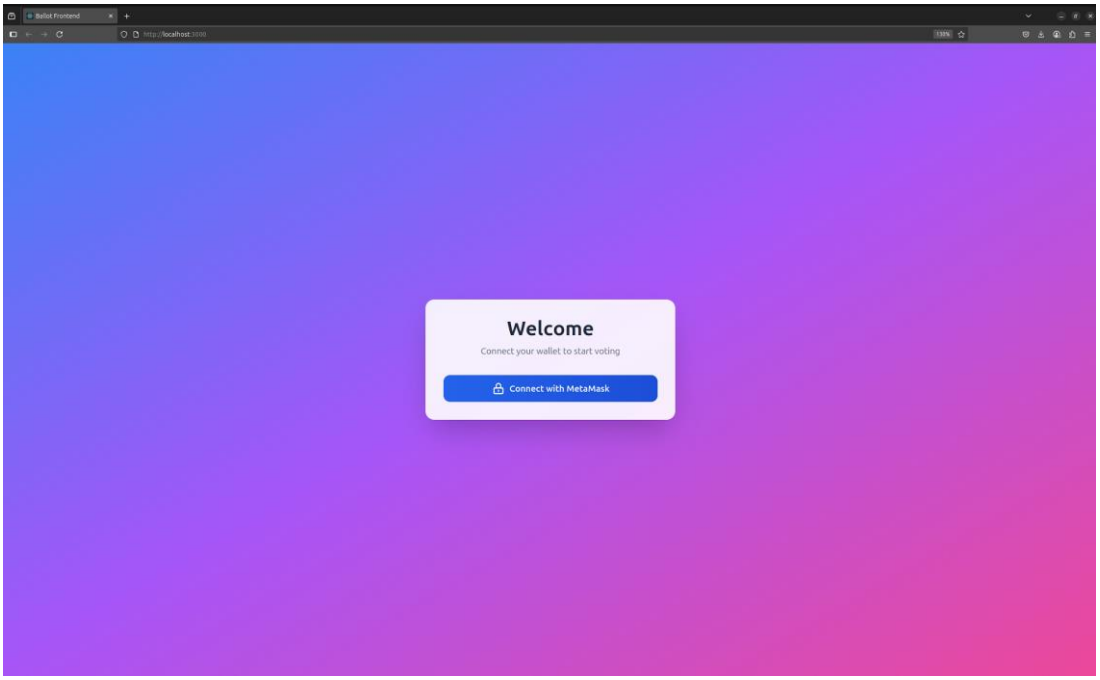


Рис. 3.4.1 Сторінка авторизації

Коли користувач натисне на кнопку “Connect with Metamask”. Розширення гаманця Metamask запросить дозвіл користувача на вхід до платформи застосунку (див. рисунок 3.4.2).

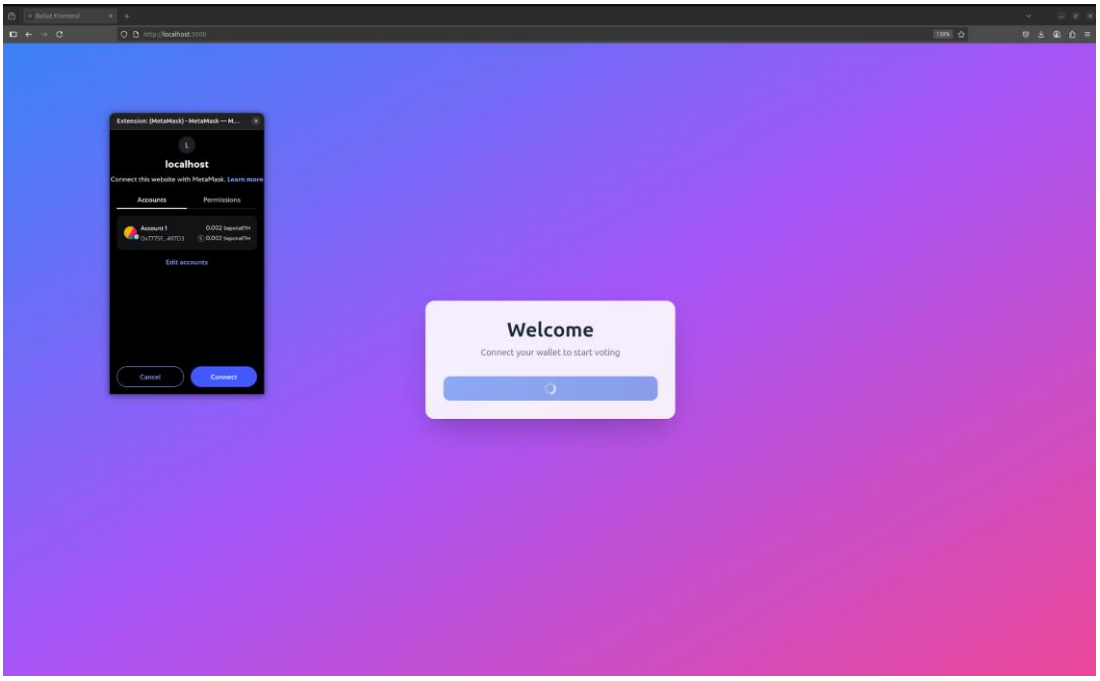


Рис. 3.4.2 Запит дозволу на підключення від гаманця Metamask

У вікні Metamask користувач може обрати аккаунт яким він хоче увійти в систему, та переглянути дозволи які потрібно надати для доступу до системи (див. рисунок 3.4.3).

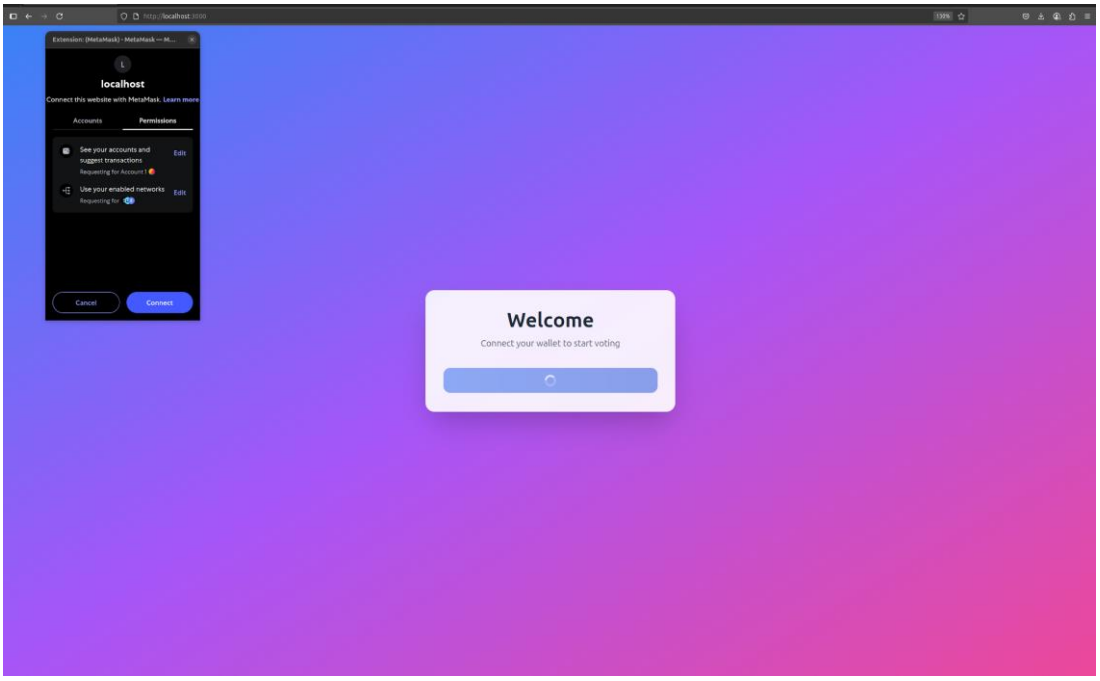


Рис. 3.4.3 Дозволи на які потрібно надати для доступу до системи

Після натискання на кнопку “Connect” у вікні Metamask користувач потрапляє на головну сторінку (Voting Dashboard), зображено на рисунку 3.4.4.

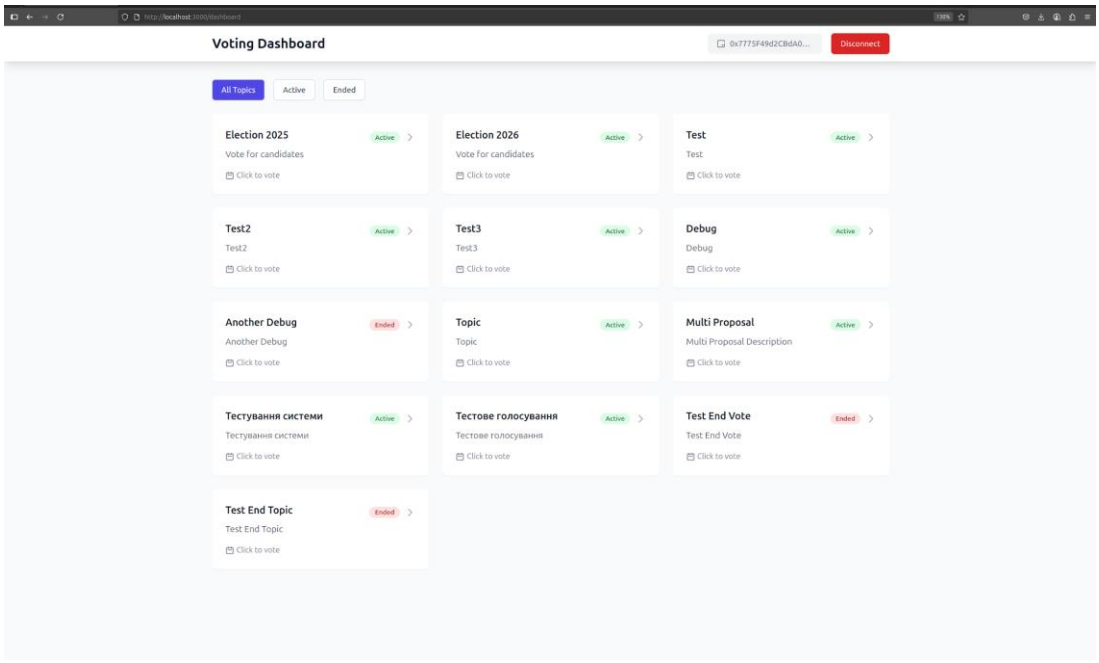


Рис. 3.4.4 Головна сторінка платформи

На цій сторінці користувач може побачити адресу власного гаманця, доступні йому голосування, а також фільтр для вибору голосувань для перегляду (активні та завершені).

Якщо користувач натисне на будь-яке з голосувань – відкривається сторінка з деталями цього голосування (див. рисунок 3.4.5).

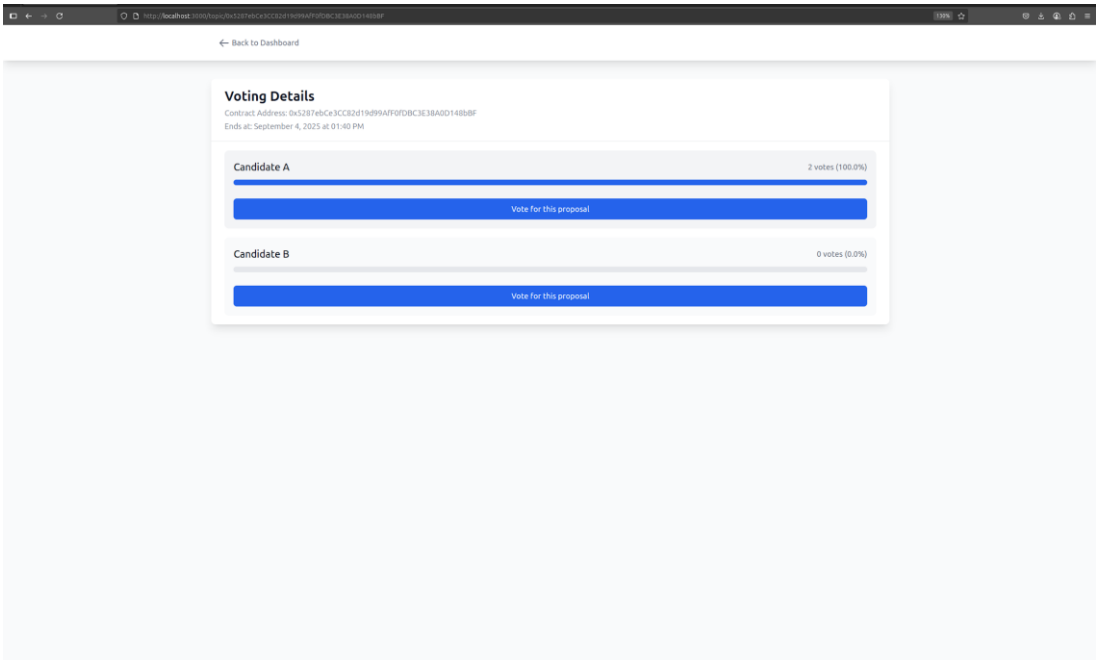


Рис. 3.4.5 Сторінка голосування

На цій сторінці можна ознайомитися з деталями голосування, переглянути адресу смарт контракту для цього голосування, дату закінчення прийому голосів, кількість опцій до вибору, кількість відданих голосів за кожну опцію і власне віддати голос.

Для взаємодії адміністраторів з системою є окремі елементи платформи. Для переходу на них потрібно перейти за маршрутом /admin (див. рисунок 3.4.6).

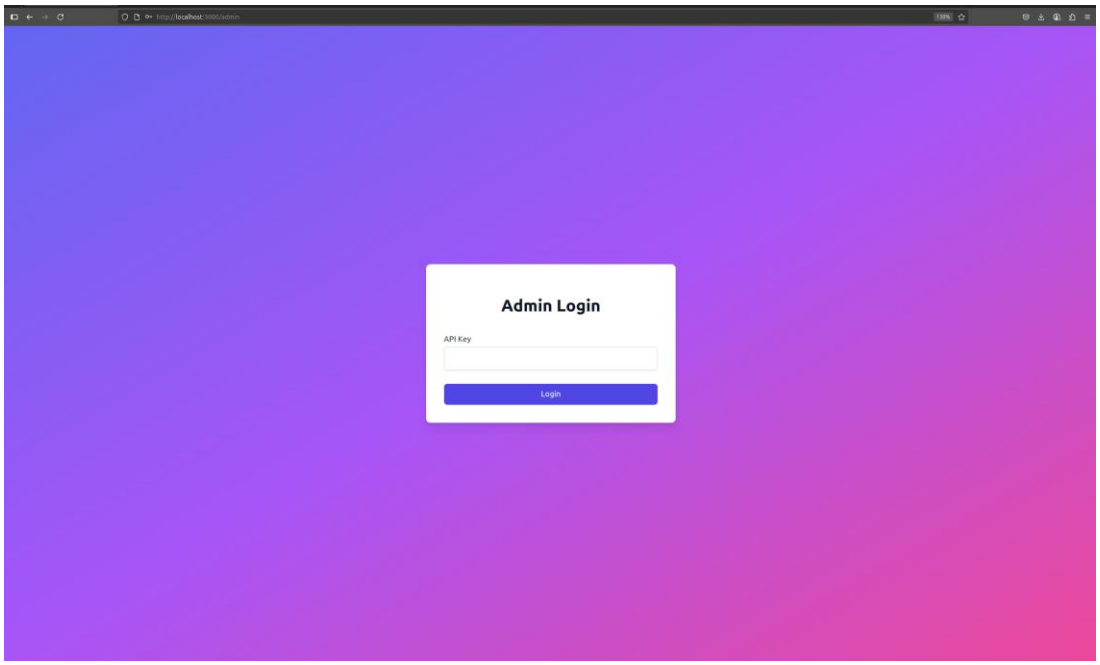


Рис. 3.4.6 Сторінка авторизації адміністратора

Після успішної авторизації відкривається головна сторінка для адміністрування, яка наведена нижче.

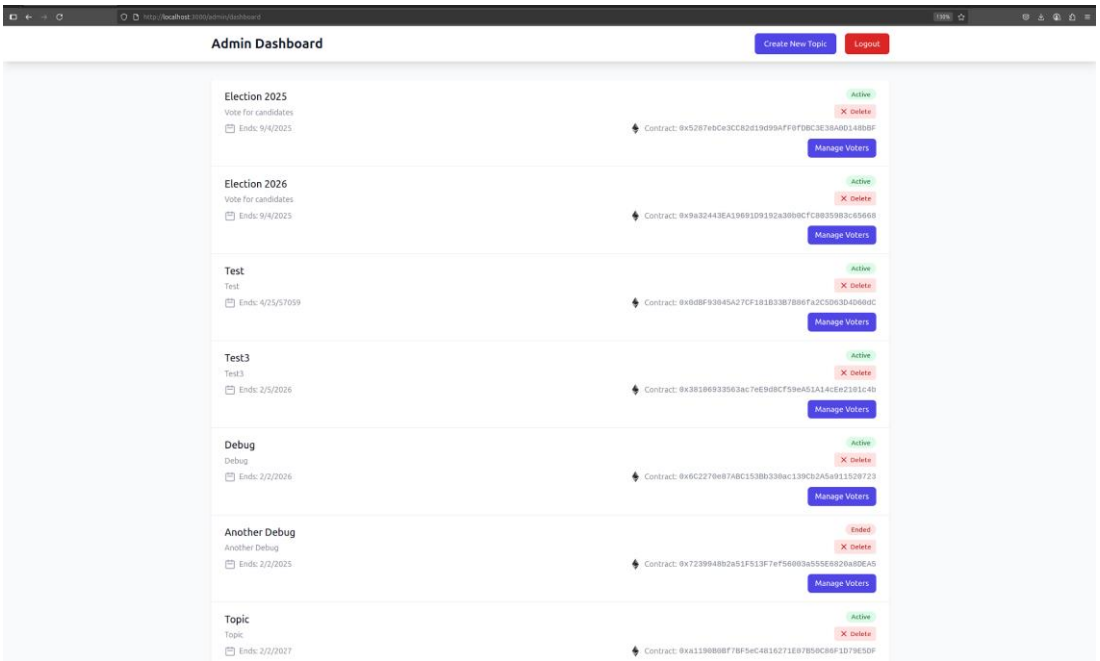


Рис. 3.4.7 Головна сторінка адміністратрування

На даній сторінці адміністратори можуть створювати голосування, видаляти існуючі голосування, переглядати їх статус, та додавати виборців (див. рисунки 3.4.8-3.4.10).

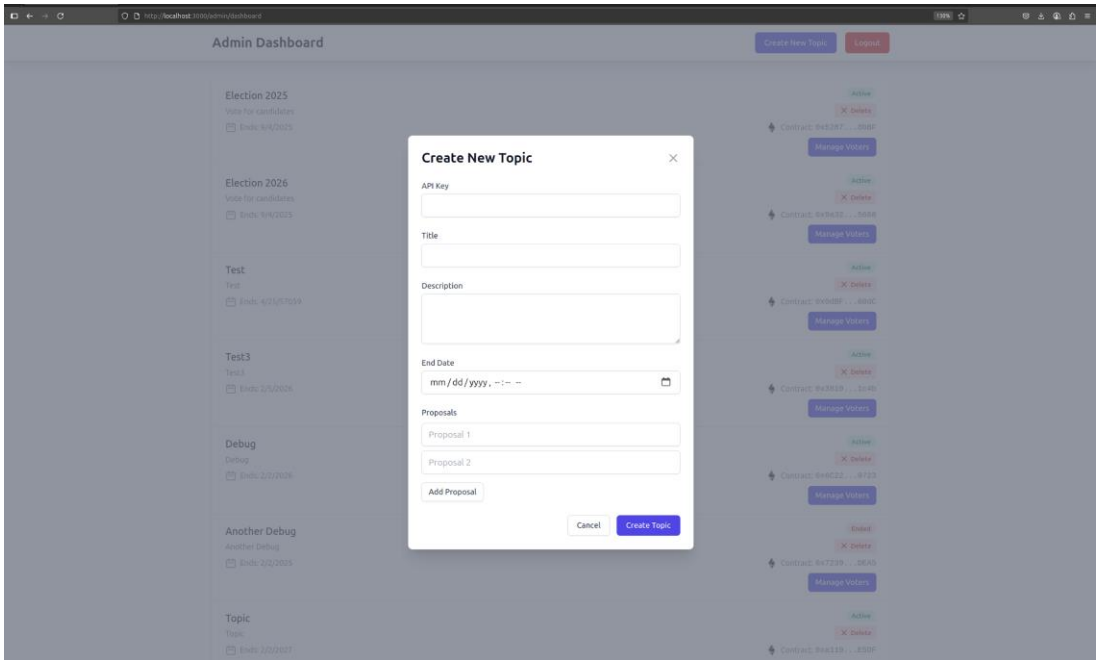


Рис. 3.4.8 Екранна форма створення нового голосування

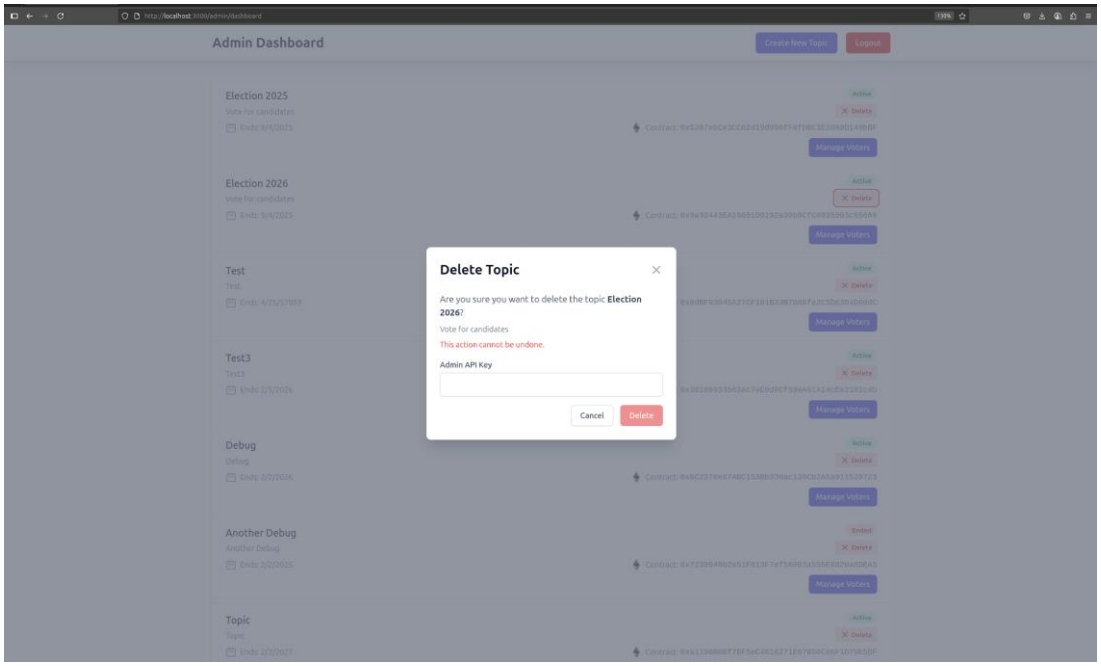


Рис. 3.4.9 Екранна форма для видалення голосування

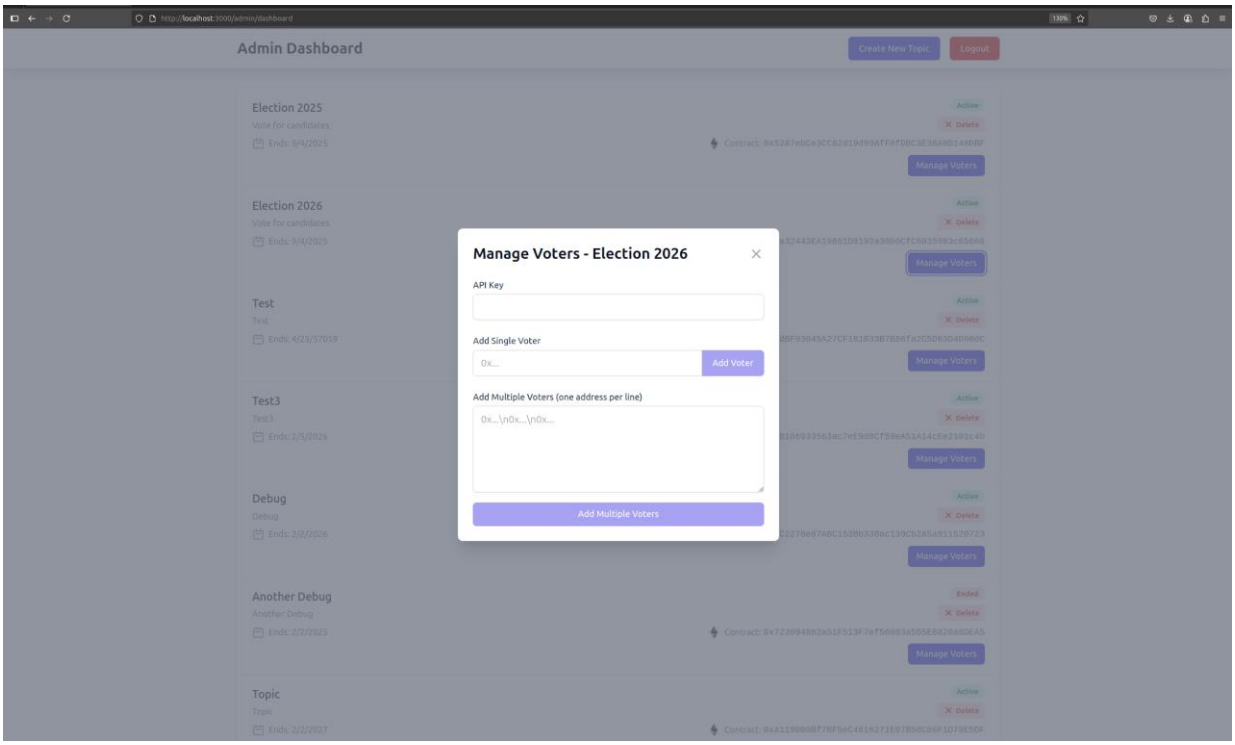


Рис. 3.4.10 Екранна форма для додавання користувачів до голосування

4 ТЕСТУВАННЯ ПРОГРАМИ

Тестування — ключовий етап розробки, що гарантує якість, надійність і безпеку ПЗ. Застосування різноманітних методів тестування дозволяє виявляти дефекти на кожному етапі життєвого циклу та забезпечити успішне впровадження продукту. Тому вибір тестових стратегій має здійснюватися з особливою увагою та відповідальністю.

Функціональне тестування — це процес перевірки функціональності програмного забезпечення, щоб переконатися, що воно працює відповідно до специфікацій і вимог.

Функціональні вимоги, що будуть протестовані:

- користувачі можуть увійти в систему;
- користувач може проголосувати лише один раз у межах одного голосування;
- результат голосування записується в блокчейн для забезпечення незмінності;
- перегляд результатів після завершення голосування;

Для тестування даних вимог було створено 3 Ethereum гаманця в Sepolia Testnet.

Таблиця 4.1

Дані гаманців, що були використані в тестуванні

Адреса	Приватний ключ
0x7775F49d2CBdA055bb6364392026c438dE5497D3	3313a0d2e199d9d4ebbc80e0affd151543dc8c37902518bf7779fd330855a83d
0xC2Dd312233FBFb791FF355800c1a5754d6E8827e	f200d0c59c8b01f2816b2779865602016e9877bdca6e8ab79954b66a0fbe67f8
0x374c411DC71CB75c3aFcbFa17FAc0fF80aA6b061	296cc0196c1c21722e842daa725236a261b02298486d26a81ba70d492561897f

Для виконання процесу тестування, контракт був завантажений в тестову мережу Ethereum і є доступним через інтернет.

У зв’язку з тим, що авторизацію було продемонстровано у минулому розділі, перейдемо одразу до інших функцій.

На рисунку 4.2 зображена сторінка деталей голосування.

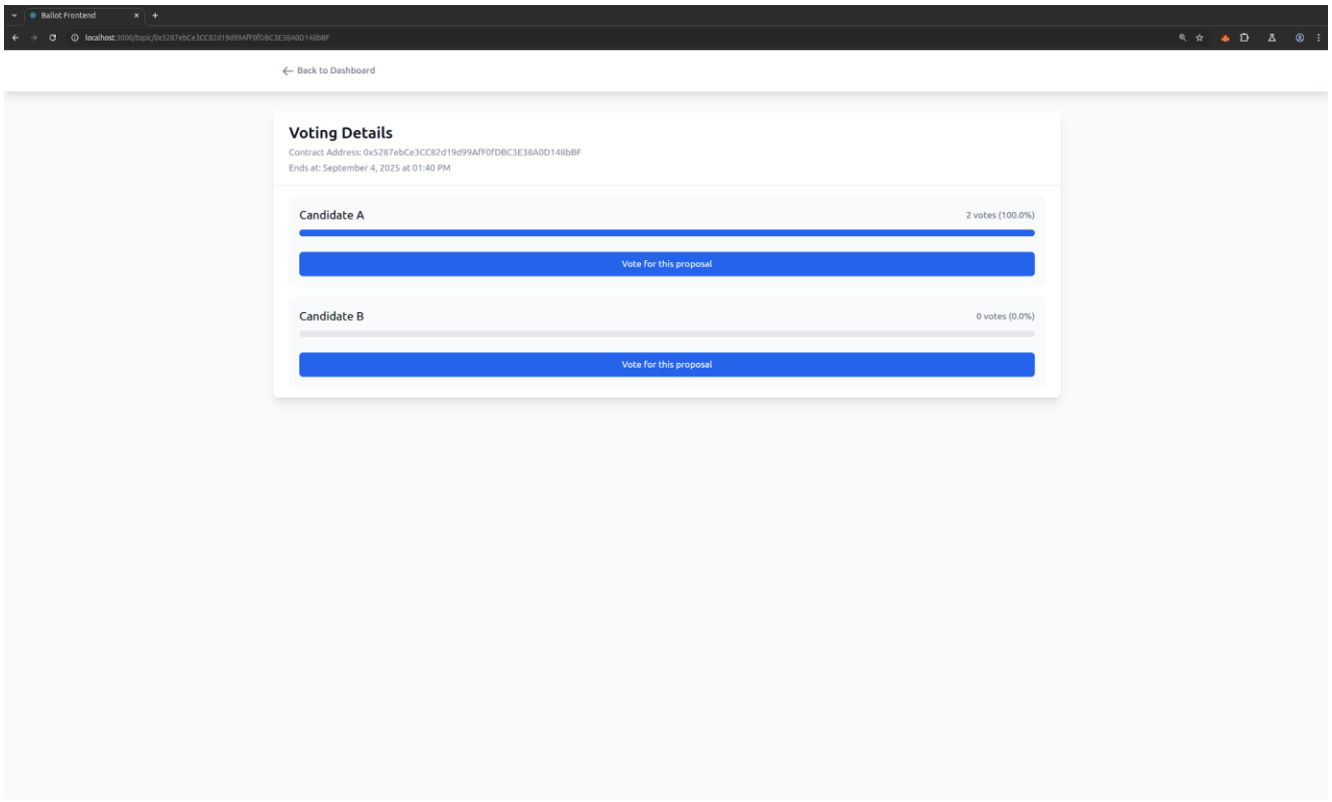


Рис. 4.2 Сторінка з деталями голосування

На рисунку 4.3 показано процес вибору однієї з опції. В правому верхньому кутку з’явилося вікно гаманця Metamask з запитом на підтвердження транзакції, Під час цього кнопки вибору під опціями стають неактивними.

Після підтвердження транзакції з’являється напис Confirming зі статусом виконання (див. рисунок 4.4), який сигналізує користувачу, що його голос взято в обробку і його вибір зараз записується в блокчейн. Система чекає поки пройде транзакція і дані запишуться в блокчейн.

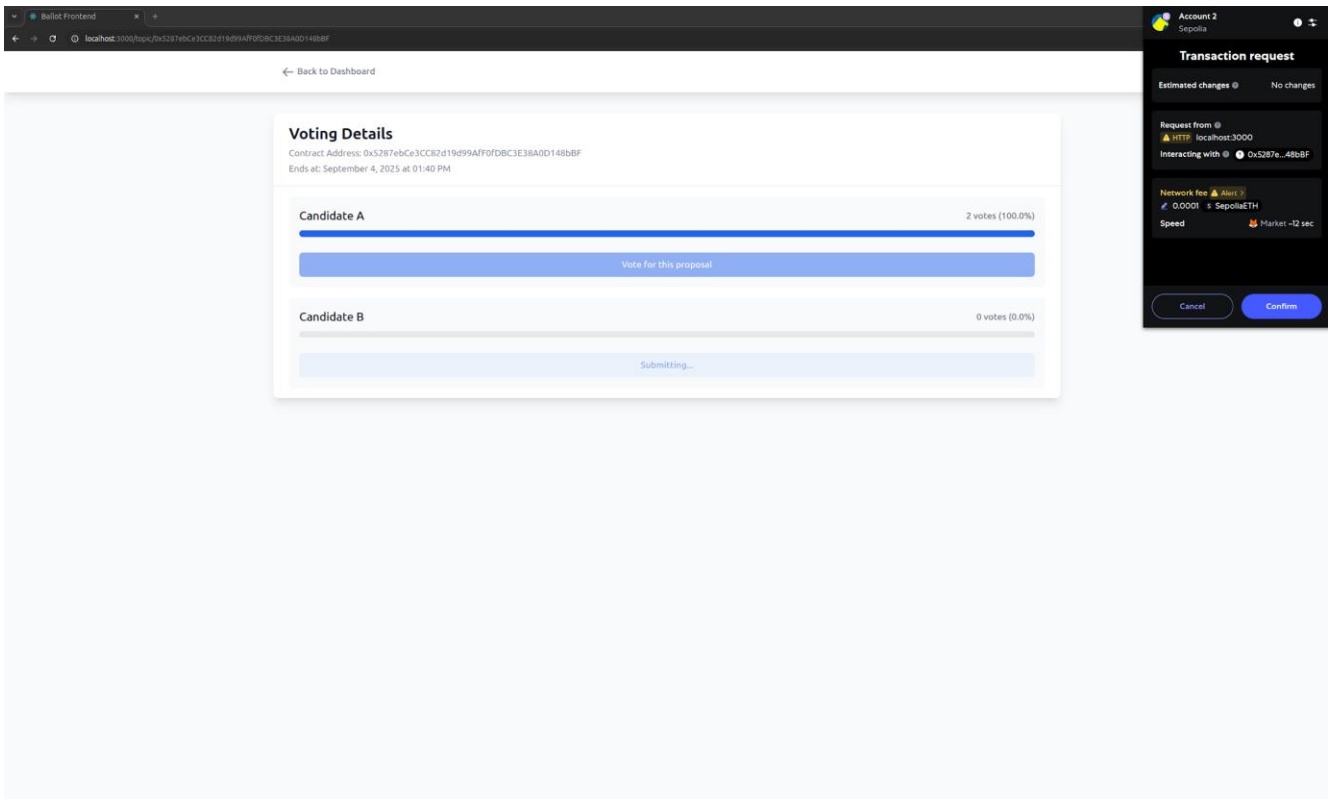


Рис. 4.3 Підтвердження транзакції користувачем

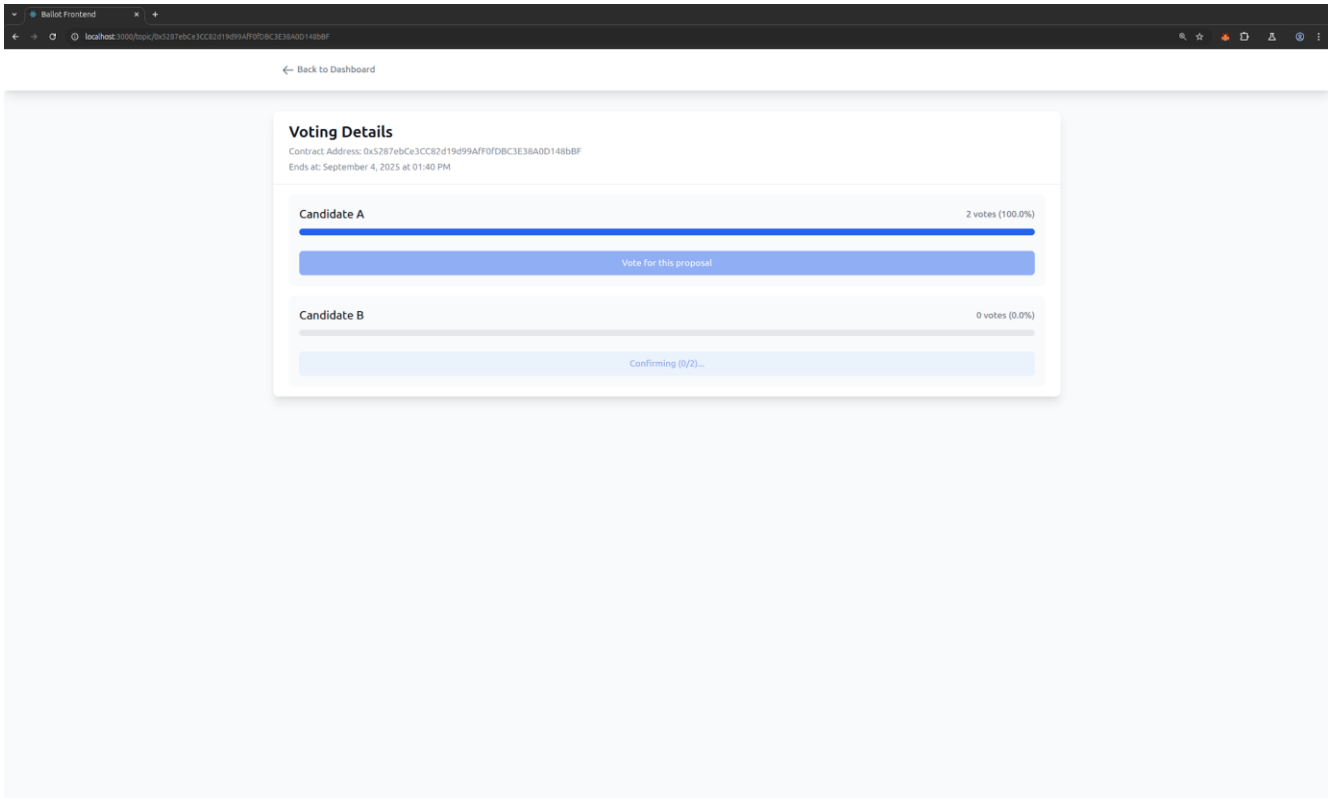


Рис. 4.4 Процес обробки відповіді

На рисунку 4.5 можна побачити, що голос було враховано. Статистика оновлена. Результат голосування записано в блокчейн

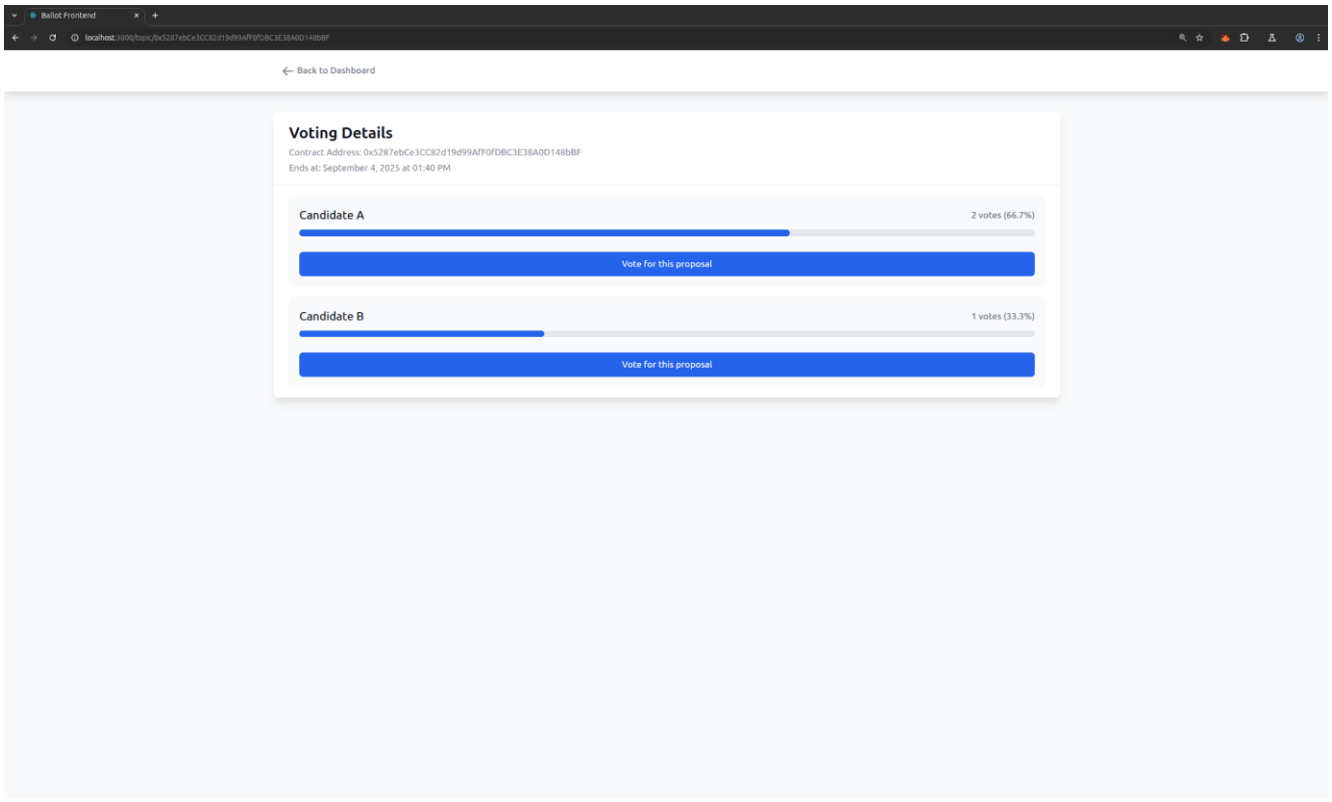


Рис. 4.5 Оновлена статистика

При спробі проголосувати знову, система видасть помилку, статистика не змінилася, як зазначено на рисунку 4.6.

Перевірити незмінність вибору можна перегнувши дані транзакції. Для цього заходимо в будь-який оглядач блокчейну. Для прикладу було обрано Etherscan і вводимо адресу гаманця або ж ідентифікатор транзакції (див. рисунок 4.7).

Адреса гаманця 0x374c411DC71CB75c3aFcbFa17FAc0fF80aA6b061, адреса смарт контракту 0x5287ebCe3CC82d19d99AfF0fDBC3E38A0D148bBF, ідентифікатор транзакції 0x7ed4f8b476b75107c19d7a37838dab2924ac59b3e59cb1f7cbb0b15c829298ee.

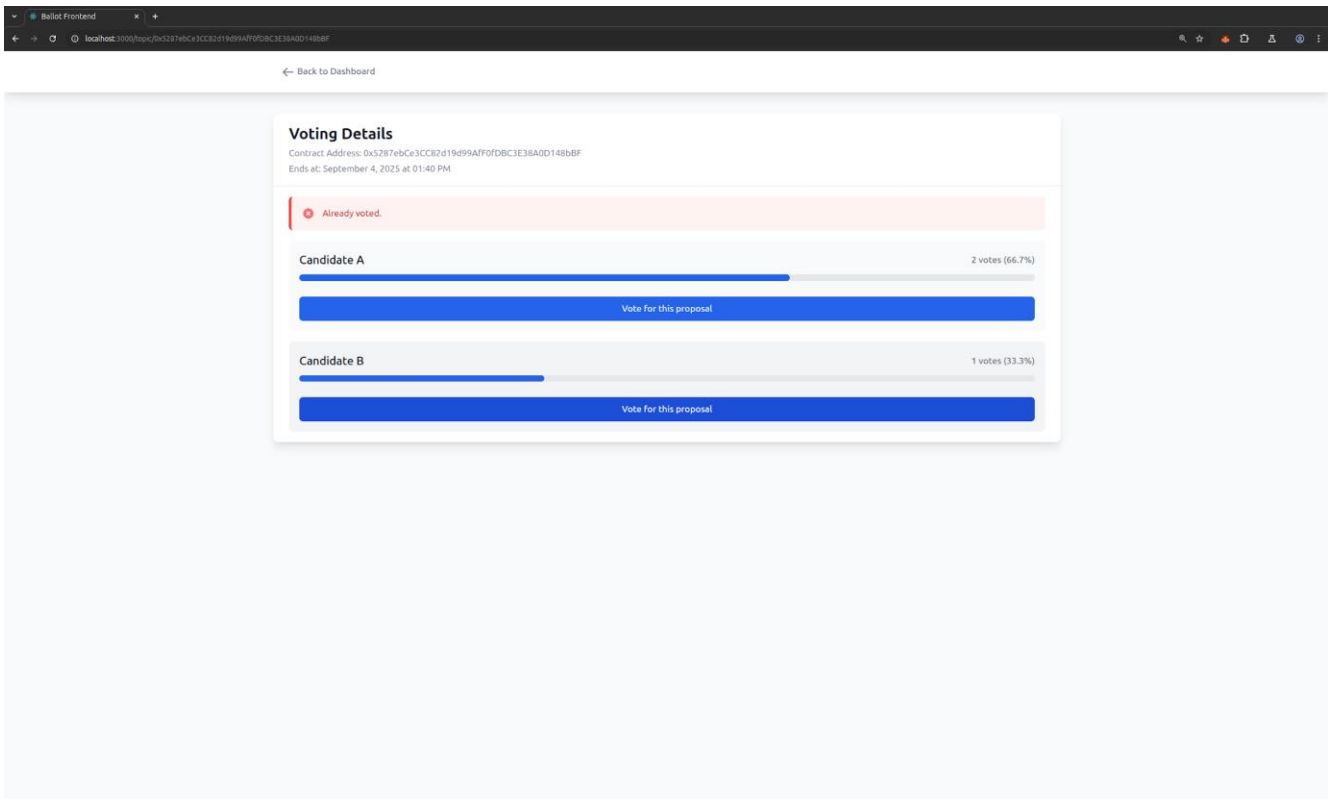


Рис. 4.6 Система забезпечує неможливість проголосувати двічі

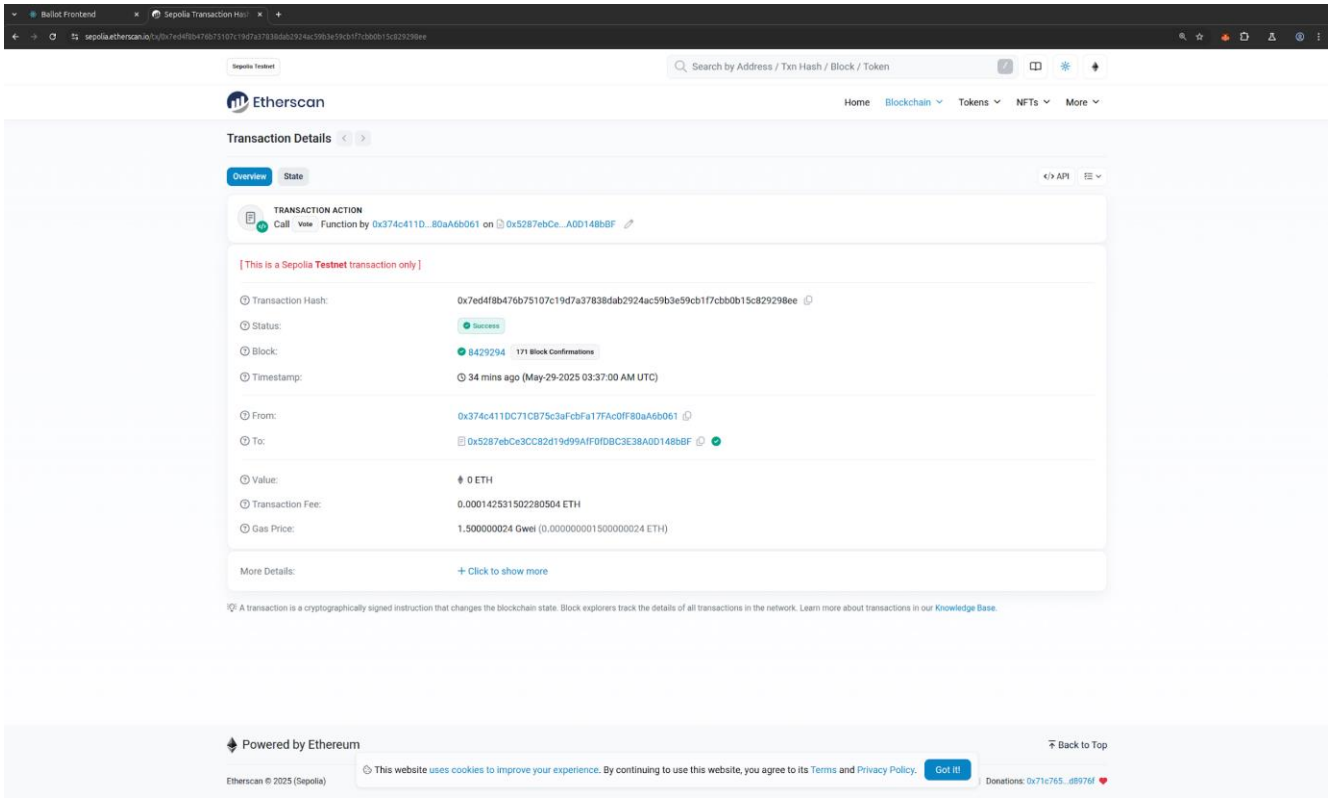


Рис. 4.7 Перегляд транзакції в блокчейні

Натискаємо на кнопку “Click to show more”.Після чого обираємо “Decode Input Data” для розшифровки вмісту транзакції. Після розшифровки вмісту транзакції бачимо що, користувач проголосував за опцію 1 (нумерація елементів відповіді починається з 0 елементу, тож за фактом це є опція номер 2).

Дані не змінено (див. рисунок 4.8).

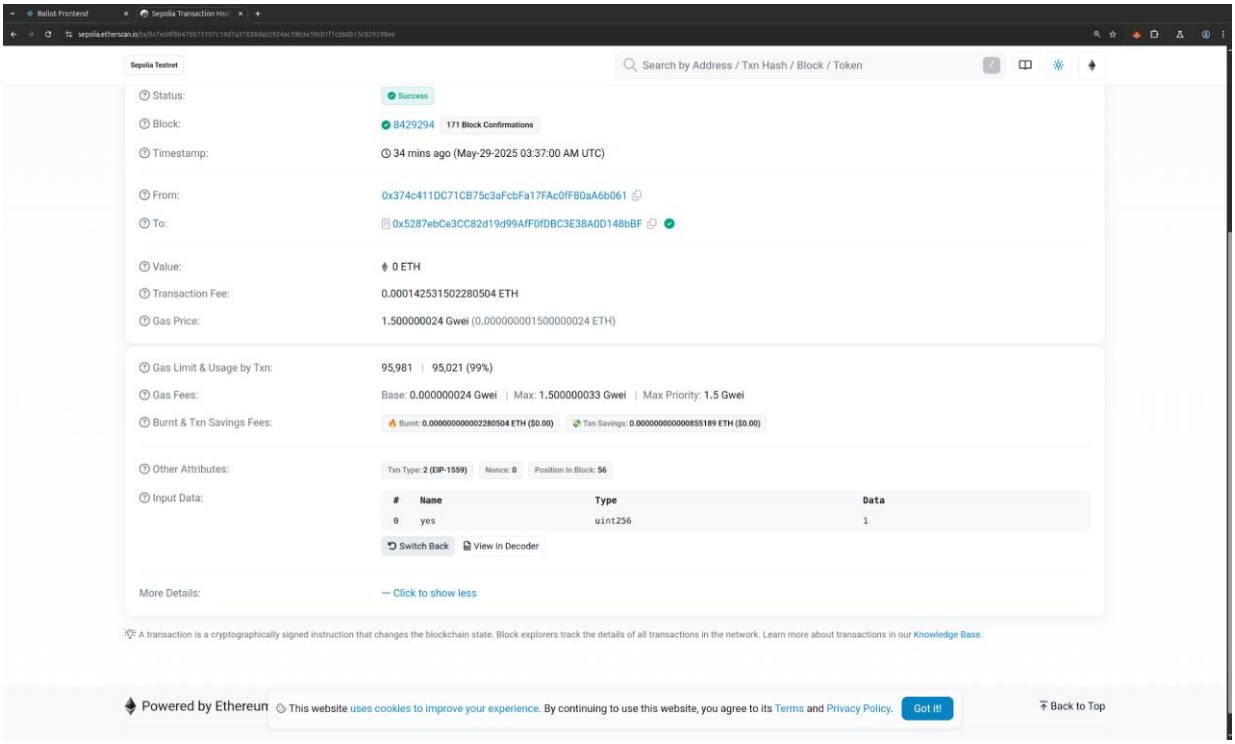


Рис. 4.8 Перегляд вмісту транзакції

З інших гаманців аналогічно було проведено голосування.

На рисунку 4.9 зображено список всіх транзакцій, що були проведені при виконанні опитування.

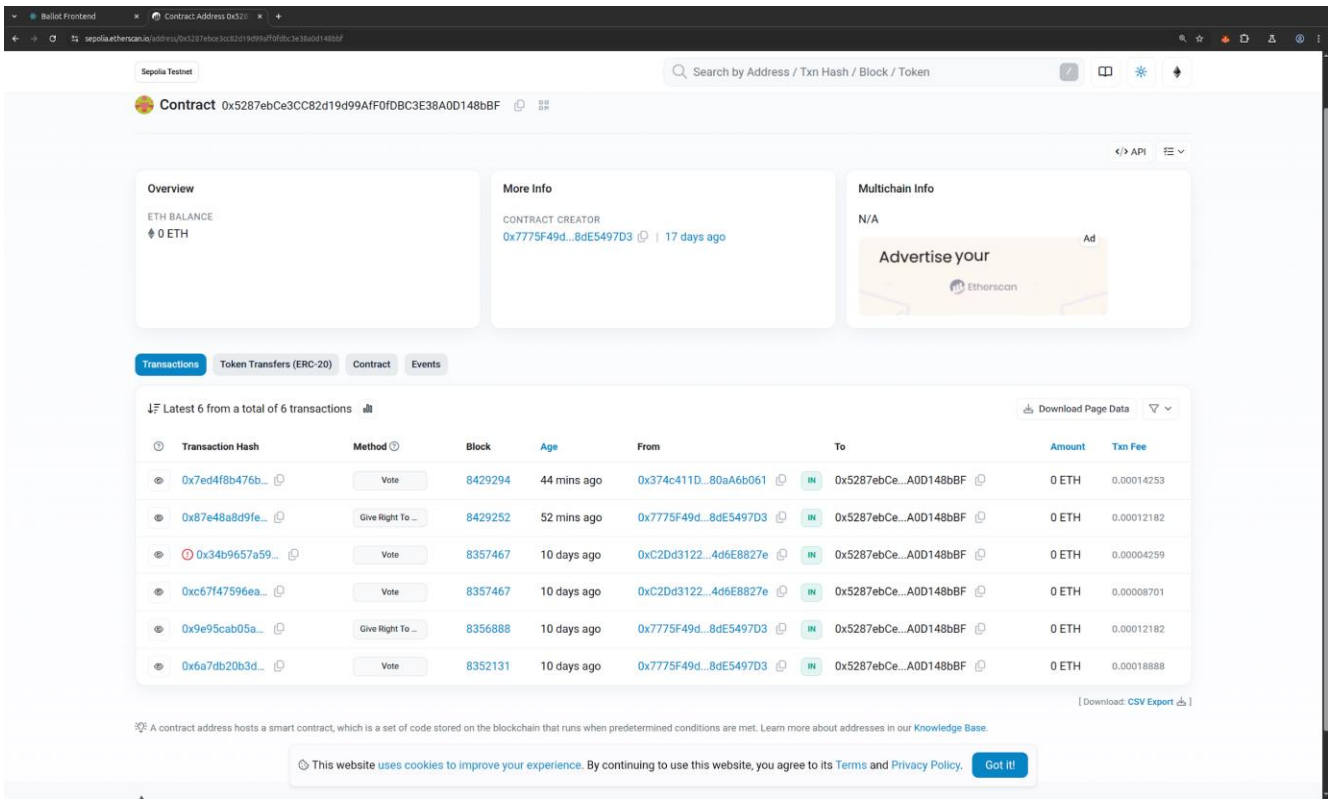


Рис. 4.9 Перелік транзакцій

Так само перевіримо адміністраторський функціонал. На рисунку 4.10 зображено процес створення нового голосування, на рисунку 4.11 успішне створення.

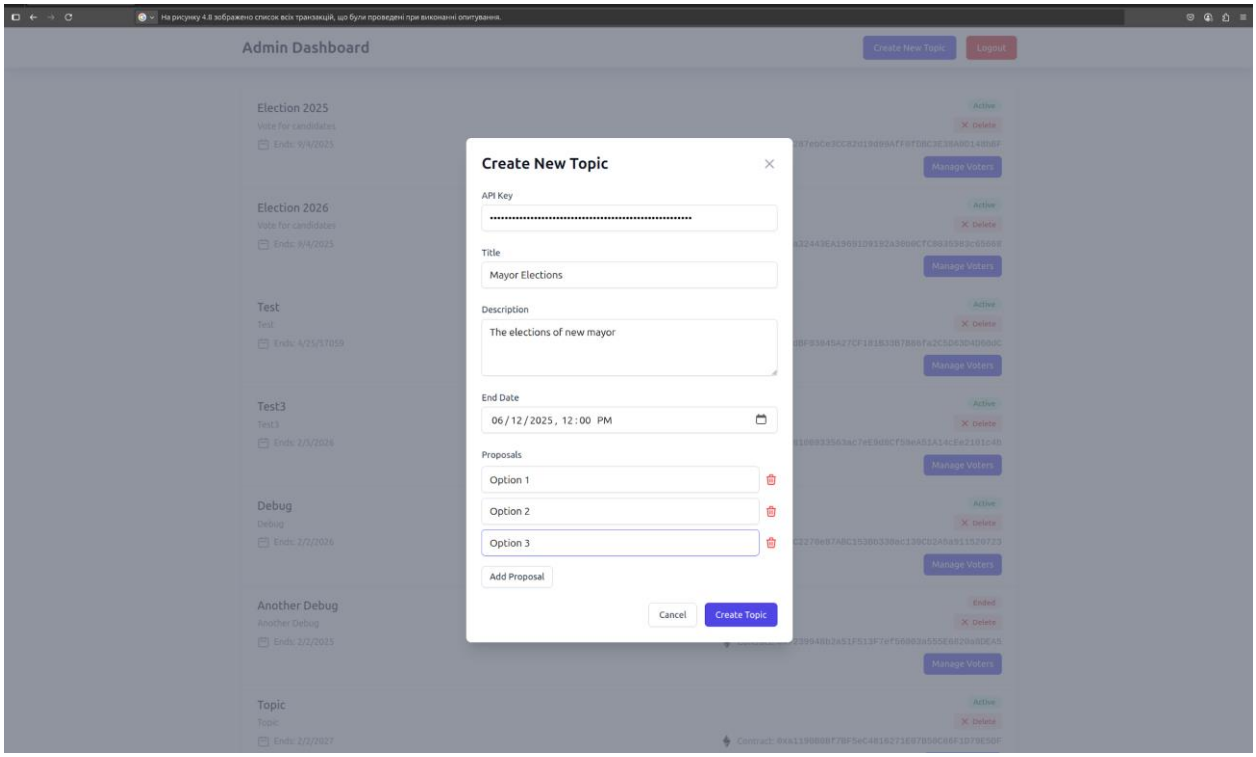


Рис. 4.10 Створення голосування

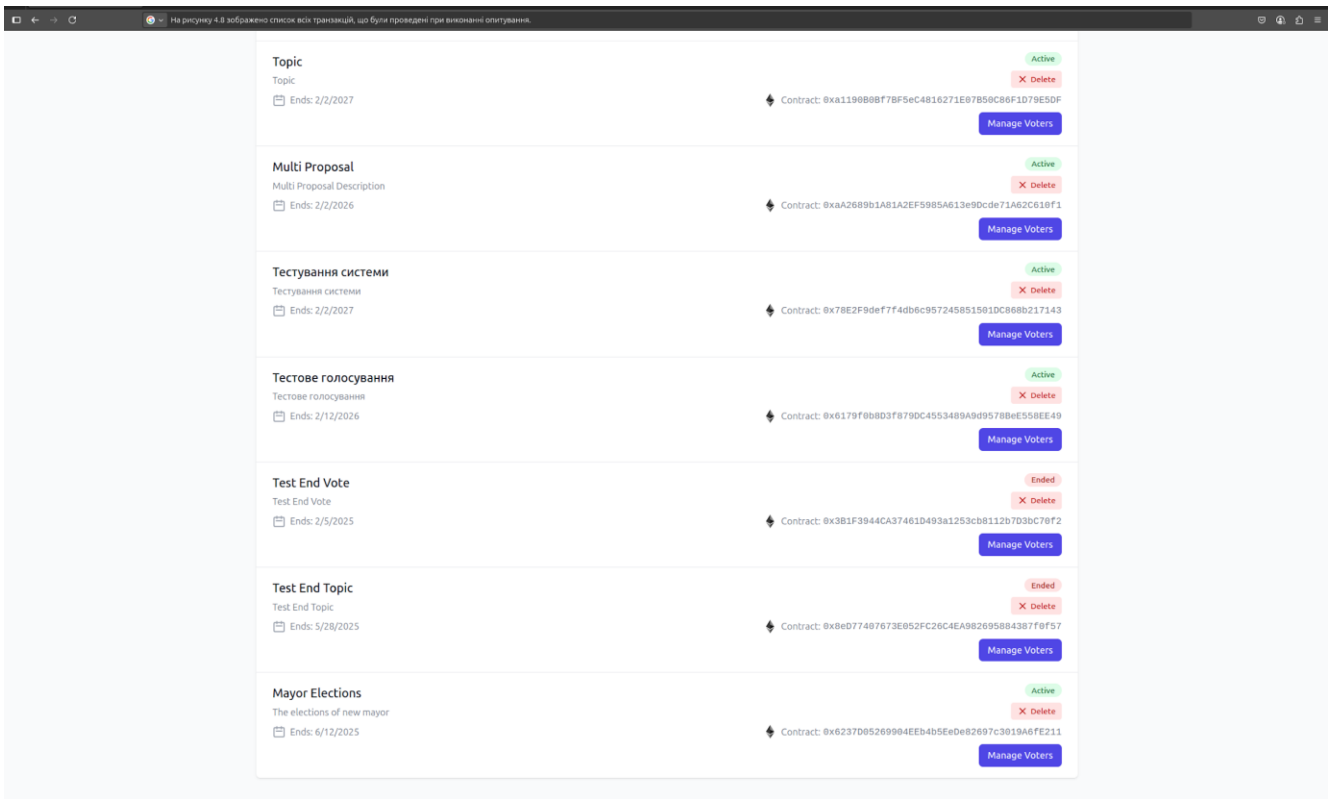


Рис. 4.11 Успішне створення голосування

На рисунку 4.12 показано процес додавання користувача до голосування.

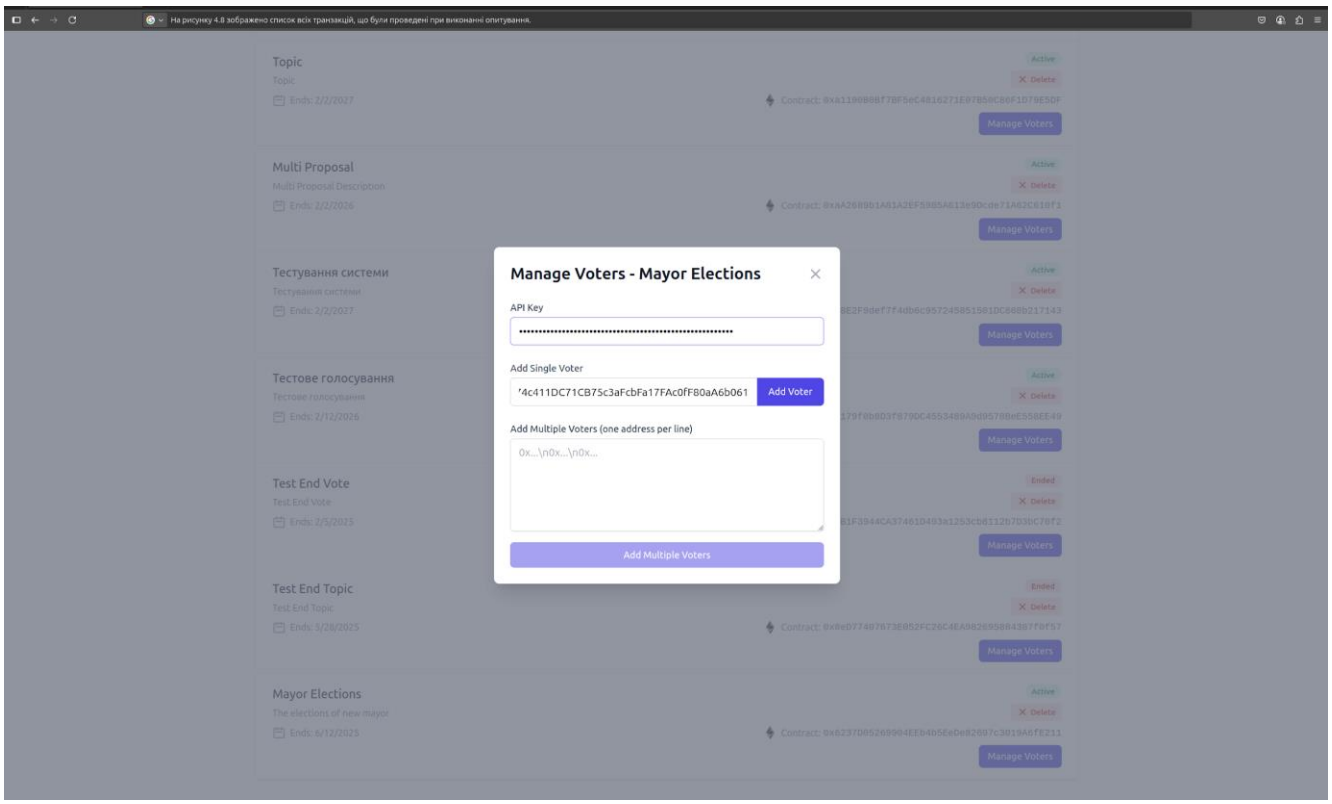


Рис. 4.12 Додавання користувача до голосування

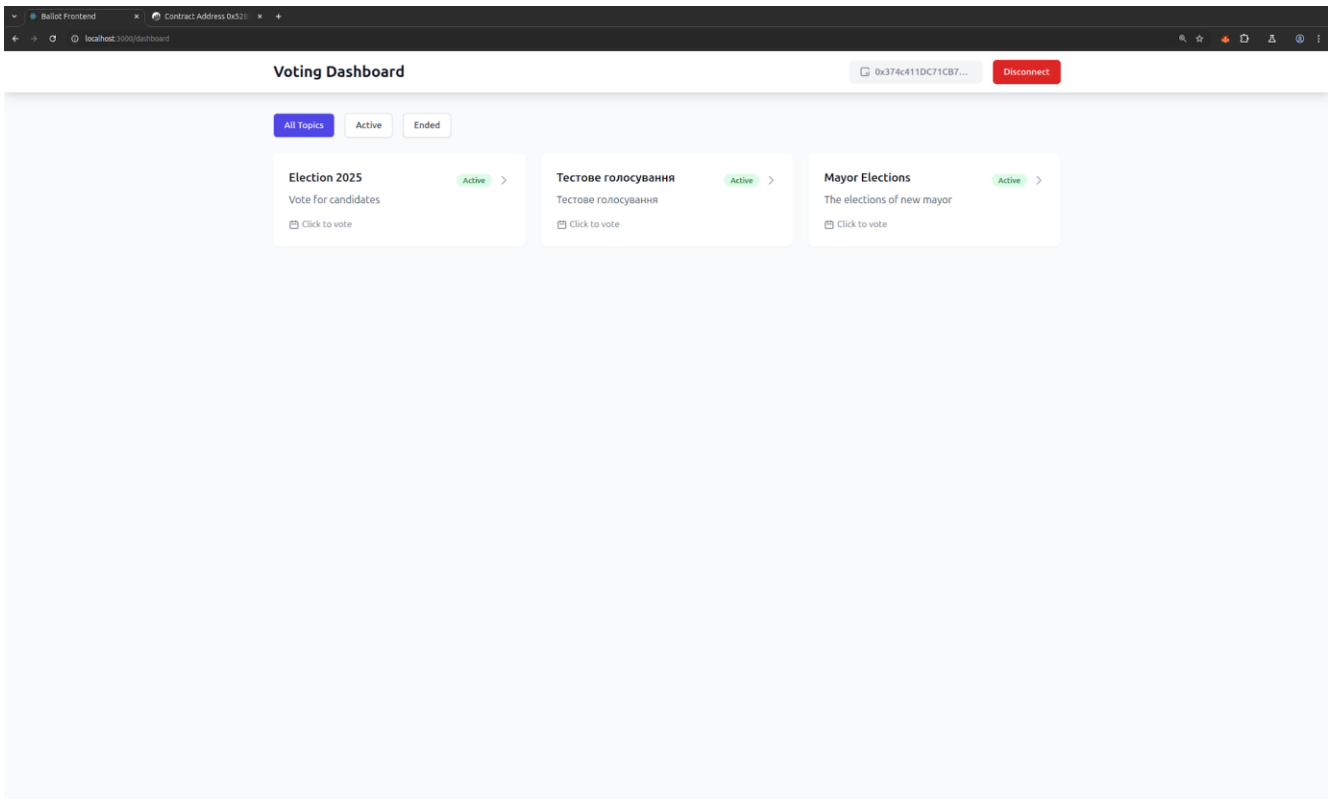


Рис. 4.13 Голосування з’явилося на головній у користувача якому дали до нього доступ

На рисунку 4.14 показано процес видалення теми для голосування

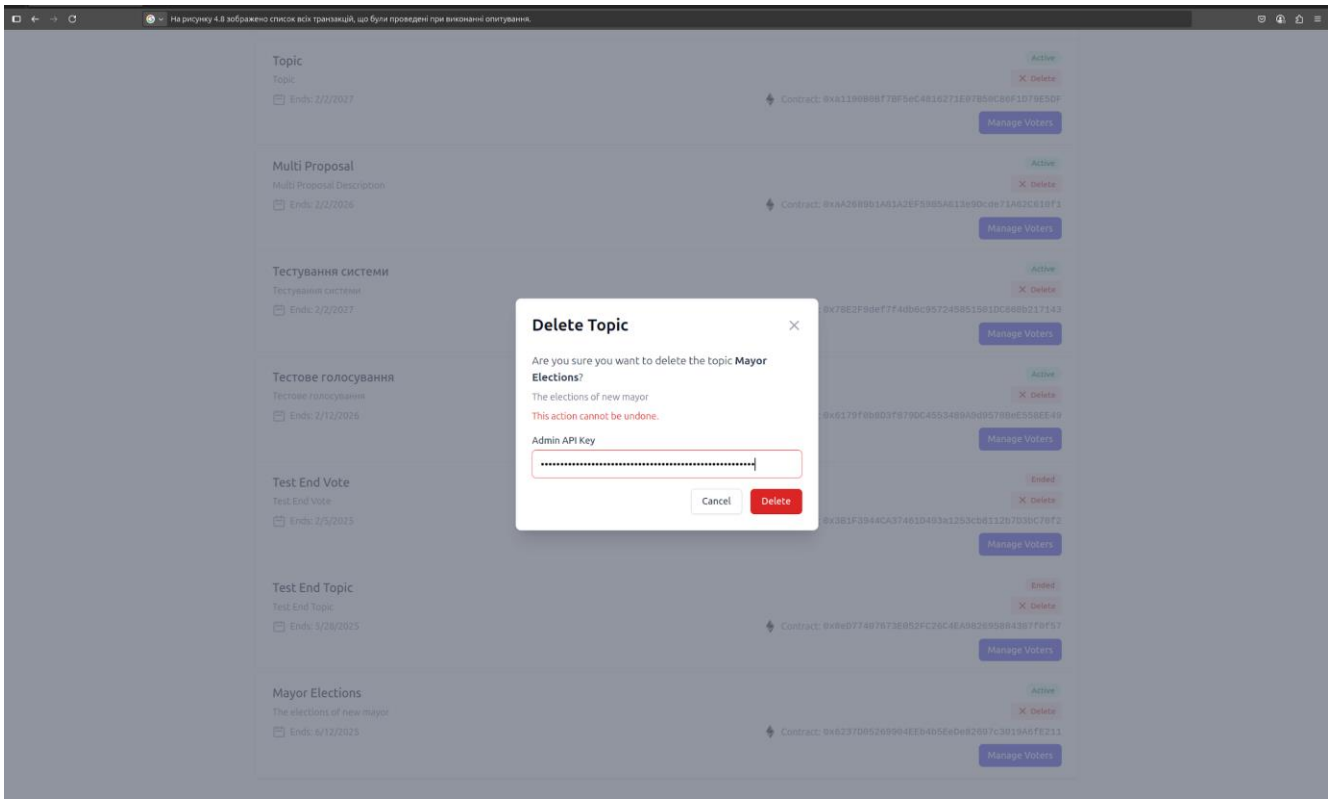


Рис. 4.14 Видалення теми

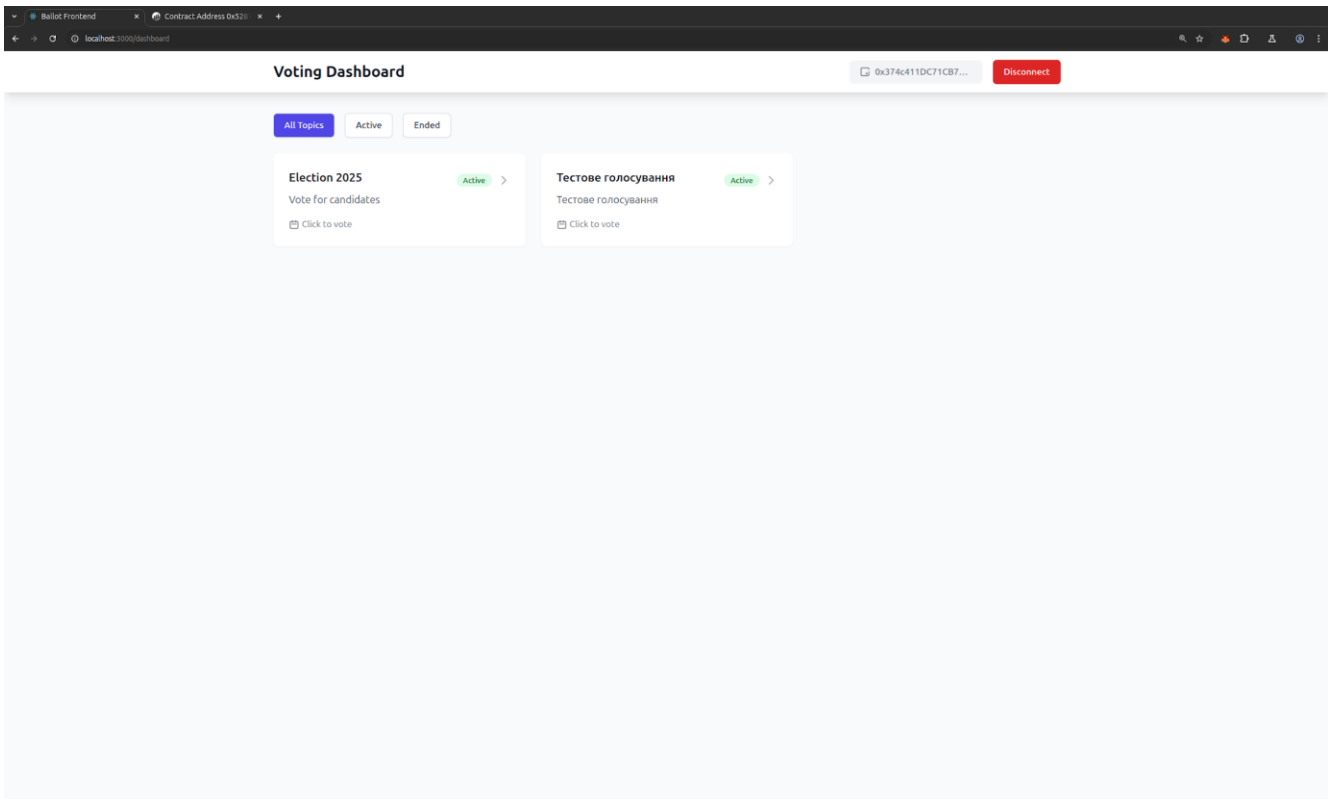


Рис. 4.15 Тема зникла у користувача, у якого був доступ до неї

ВИСНОВКИ

Під час роботи над дипломним проєктом було виконано усі поставлені задачі:

1. Проведено аналіз сучасних вимог до web-сайтів, а саме: розглянуто вимоги до безпеки, продуктивності, масштабованості, а також принципи UI/UX

2. Досліджено фундаментальні принципи електронного голосування, зокрема вимоги до безпеки, анонімності, цілісності та прозорості; детально розглянута технологія блокчейн, її принципи децентралізації, незмінності даних та механізм консенсусу.

3.Проведено огляд прикладів впровадження електронного голосування в окремих країнах, зокрема Естонії та Швейцарії. Визначено характерні риси, сильні сторони та потенційні обмеження різних підходів, включаючи використання смарт-карт, електронної ідентифікації, криптографічних методів і блокчейн-технологій.

4. Сформовано функціональні та нефункціональні вимоги до web-сайту для електронного голосування. Серед основних вимог: користувач може проголосувати лише один раз у межах одного голосування та результат голосування записується в блокчейн для забезпечення незмінності.

5. Досліджено інструменти та технології для розробки web-сайту для електронного голосування, а саме: мова програмування Typescript, бекенд фреймворк Nest.js у поєднанні з об'єктно-реляційною системою для керування базами даних PostgreSQL, бібліотека Web3.js для розробки бекенду додатку; для розробки фронтенду використовувалася мова програмування TypeScript та фреймворк React у поєднанні з фреймворком Tailwind CSS.

6. Розроблено web-сайт для проведення електронного голосування з використанням блокчейн технологій.

7. Проведено функціональне та нефункціональне тестування системи, яке показало, що продукт відповідає зазначеним вимогам.

Розроблений демо-застосунок успішно реалізує дві важливі вимоги, з якими стикаються традиційні електронні системи: незмінність голосів після їх подання та

можливість перевірки правильності їх збереження. Отже, завдання дипломного проєкту виконано в повному обсязі.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Іщук М.О., Шевченко С.М. Можливості застосування технології блокчейн для проведення електронних голосувань // Матеріали Всеукраїнської науково-практичної Інтернет-конференції «Стратегії кіберстійкості: управління ризиками та безперервність бізнесу» 27.02.2025, ДУІКТ, Київ, С.200-204.
2. Іщук М.О., Шевченко С.М. Розробка web-сайту для проведення електронних голосувань з використанням блокчейн технологій // Апаратне і програмне забезпечення інформаційних технологій. Збірник матеріалів Всеукраїнської конференції молодих учених "Інформаційні технології" (м. Київ, 15 травня 2025 року) / Факультет інформаційних технологій та математики Київського столичного університету імені Бориса Грінченка. Київ, 2025. С.131-133.
3. Іщук М.О., Шевченко С.М. Порівняльний аналіз ефективності різних блокчейн-платформ для електронного голосування // «Кібербезпека: освіта, наука, техніка», 2025, №2 (подано до друку).

ПЕРЕЛІК ПОСИЛАНЬ

1. *The International Institute for Democracy and Electoral Assistance (International IDEA*), Feb 06, 2023. Use of E-Voting Around the World. URL: https://www.idea.int/news-media/multimedia-reports/use-e-voting-around-world

2. Meredith Applegate, Thomas Chanussot, Vladlen Basysty, April, 2020, Considerations on Internet Voting: An Overview for Electoral Decision-Makers. URL: https://ifesukraine.org/wp-content/uploads/2020/04/IFES-White-Paper-Applegate-Chanussot-Basysty-%E2%80%98Considerations-on-Internet-Voting%E2%80%99-Mar-2020-Eng-1.pdf

3. Martin Russell, Ionel Zamfir Dixon, Sep, 2018. Digital technology in elections. URL: https://www.europarl.europa.eu/RegData/etudes/BRIE/2018/625178/EPRS_BRI(2018)625178_EN.pdf

4. Piret Ehin, Mihkel Solvak, Jan Willemson, Priit Vinkel. Apr 30, 2021. Internet voting in Estonia 2005–2019: Evidence from eleven elections. URL: https://www.sciencedirect.com/science/article/pii/S0740624X2200051X?via%3Dihub

5. Stages of i-voting in voter application. URL: https://www.valimised.ee/en/internet-voting/guidelines/stages-i-voting-voter-application

6. Internet voting in Estoia. URL: https://www.valimised.ee/en/internet-voting-estonia

7. E-voting in Switzerland. URL: https://www.evoting-info.ch/gut-zu-wissen/e-voting-in-der-schweiz

8. How does Voatz work? URL: https://voatz.com/how-it-works/

9. Michael A. Specter, James Koppel, Daniel Weitzner. The Ballot is Busted Before the Blockchain. A Security Analysis of Voatz, the First Internet Voting Application Used in U.S. Federal Elections. 5 August 2019. URL: https://internetpolicy.mit.edu/wp-content/uploads/2020/02/SecurityAnalysisOfVoatz_Public.pdf

10. Helios voting system. URL: https://vote.heliosvoting.org/

11. Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*.. URL: https://bitcoin.org/bitcoin.pdf

12. Wood, G. (2014). *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. URL: https://ethereum.org/en/whitepaper/

13. Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V., Jun 2016. *Blockchain technology: Beyond bitcoin*. Applied Innovation Review, Issue 2. URL: https://scet.berkeley.edu/wp-content/uploads/AIR-2016-Blockchain.pdf

14. Marcela T. Oliveira, Gabriel R. Carrara, Natalia C. Fernandes, C´elio V. N. Albuquerque,Ricardo C. Carrano, Dianne S. V. Medeiros and Diogo M. F. Mattos, Feb 2019. *Towards a Performance Evaluation of Private Blockchain Frameworks using a Realistic Workload*. URL: https://www.researchgate.net/publication/332379108_Towards_a_Performance_Evaluation_of_Private_Blockchain_Frameworks_using_a_Realistic_Workload

15. Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, George Danezis, October 21-23, 2019. *SOK: Consensus in the Age of Blockchains*. Communications of the ACM. URL: https://dl.acm.org/action/showFmPdf?doi=10.1145%2F3318041

16. Buterin, V. (2020). *Ethereum 2.0: Proof of Stake*. URL: https://ethereum.org/en/roadmap/beacon-chain

17. Zyskind, G., Nathan, O., & Pentland, A. (2015). Decentralizing privacy: Using blockchain to protect personal data. IEEE Security and Privacy Workshops. URL: https://ieeexplore.ieee.org/document/7163223

18. Buterin, V. (2021). *A rollup-centric Ethereum roadmap*. Ethereum Blog. URL: https://ethereum.org/en/developers/docs/scaling/

19. StackOverflow Developer Survey, 2024. URL: https://survey.stackoverflow.co/2024/technology#most-popular-technologies-webframe

20. Octoverse: AI leads Python to top language as the number of global developers surges. URL: https://github.blog/news-insights/octoverse/octoverse-2024/

21. NestJS Documentation. URL: https://docs.nestjs.com/

22. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/

23. Prisma Documentation. URL: https://www.prisma.io/docs/
24. Solidity Documnetation. URL: https://docs.soliditylang.org/en/v0.8.30/
25. Web3.js Documentation. URL: https://web3js.readthedocs.io/
26. JWT Documentation. URL: https://jwt.io/introduction/
27. Docker Documentation. URL: https://docs.docker.com/
28. React Documentation. URL: https://react.dev/learn
29. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs
30. Vite Documentation. URL: https://vitejs.dev/guide/

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА WEB-САЙТУ ДЛЯ ПРОВЕДЕННЯ ЕЛЕКТРОННИХ ГОЛОСУВАНЬ З ВИКОРИСТАННЯМ БЛОКЧЕЙН ТЕХНОЛОГІЙ

Виконав студент 4 курсу
групи ПД-43
Іщук Михайло Олександрович
Керівник роботи
к. п. н., доц, доцент кафедри ІПЗ Шевченко Світлана Миколаївна

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - удосконалення забезпечення прозорості, безпеки та достовірності електронного голосування шляхом використання технології блокчейн.
- **Об’єкт дослідження -** процес електронного голосування.
- **Предмет дослідження –** web-сайт для проведення електронних голосувань з використанням блокчейн технологій.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз джерел щодо вимог до розробки web-сайту.
2. Здійснити огляд та аналіз існуючих рішень, що використовуються для електронного голосування.
3. Сформувати вимоги до розробки web-сайту для проведення електронних голосувань з використанням блокчейн технологій.
4. Визначити програмні засоби для реалізації web-сайту.
5. Розробити web-сайт для проведення електронних голосувань з використанням блокчейн технологій.
6. Провести тестування розробленого web-сайту.

3

АНАЛІЗ АНАЛОГІВ

	Estonia i-Voting	**SwissPost eVoting**	**Voatz**	**Helios**	**Ballot**
Аутентифікація виборця	ID-картка	Набір кодів + криптографічний підпис	Біометрія + номер мобільного телефону	Email/логін	Ethereum адреса + Криптографічний підпис
Цілісність голосу	Так	Так	Так	Так	Так
Можливість верифікації виборцем	Так	Так	Обмежена	Так	Так
Анонімність голосування	Так	Так	Частково	Обмежена	Так
Використання блокчейну	Ні	Ні	Так	Ні	Так

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги до програмного забезпечення

1. Користувачі можуть увійти в систему.
2. Користувач може проголосувати лише один раз у межах одного голосування.
3. Результат голосування записується в блокчейн для забезпечення незмінності.
4. Перегляд результатів після завершення голосування

Нефункціональні вимоги до програмного забезпечення

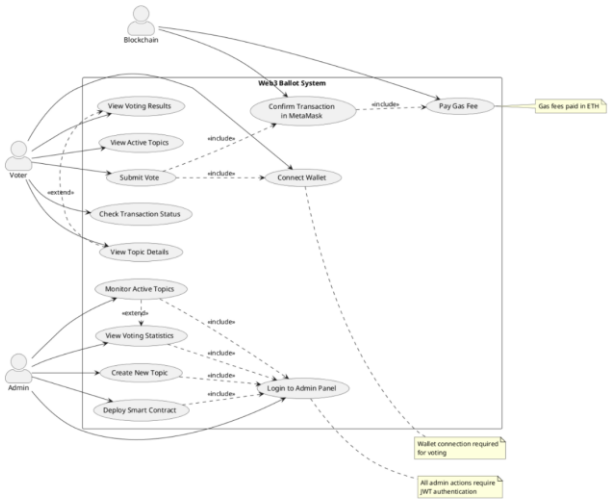
1. Система повинна гарантувати захист голосів від несанкціонованої модифікації
2. Цілісність даних навіть при збоях
3. Забезпечити проведення голосування в онлайн режимі.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

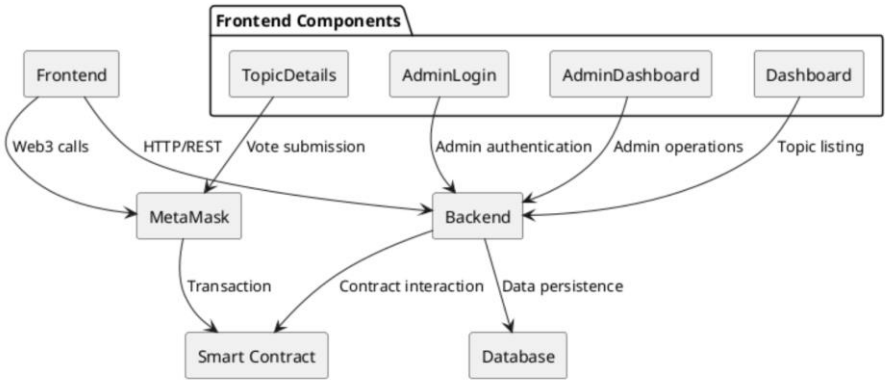
6

МОДЕЛЬ ПРЕЦЕДЕНТІВ



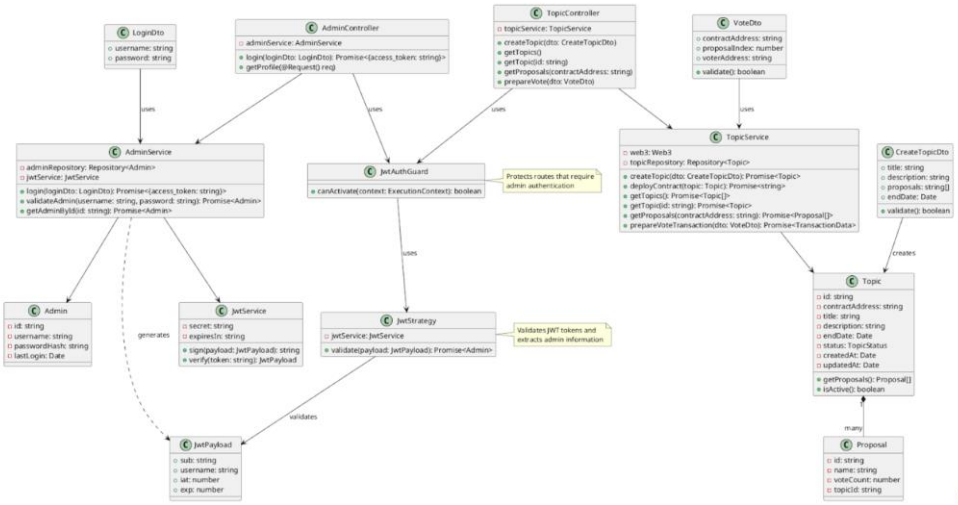
7

АРХІТЕКТУРА СИСТЕМИ



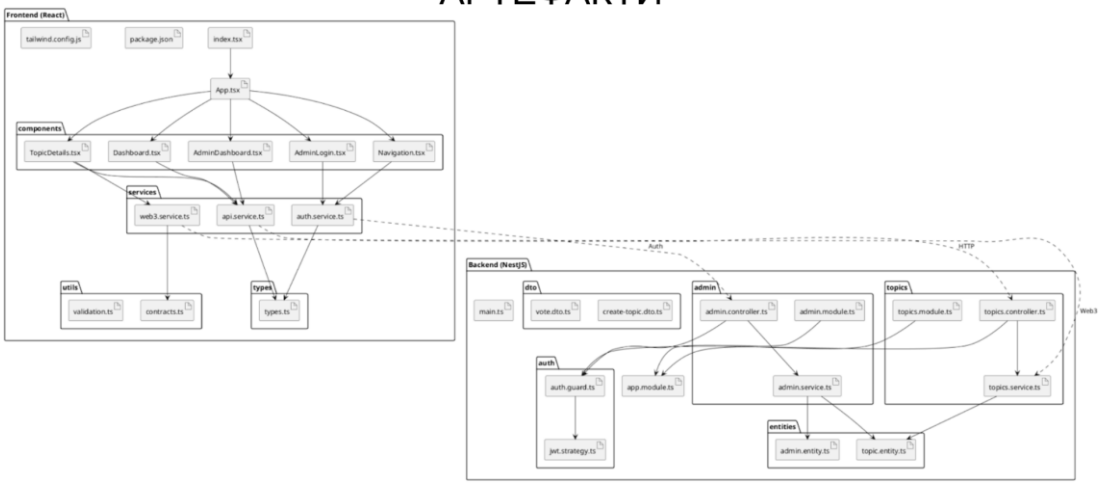
8

КЛАСИ



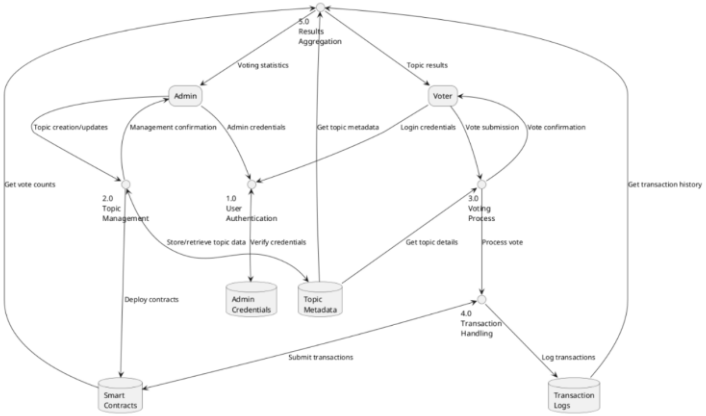
9

АРТЕФАКТИ



10

ПОТОКИ



11

ЕКРАННІ ФОРМИ

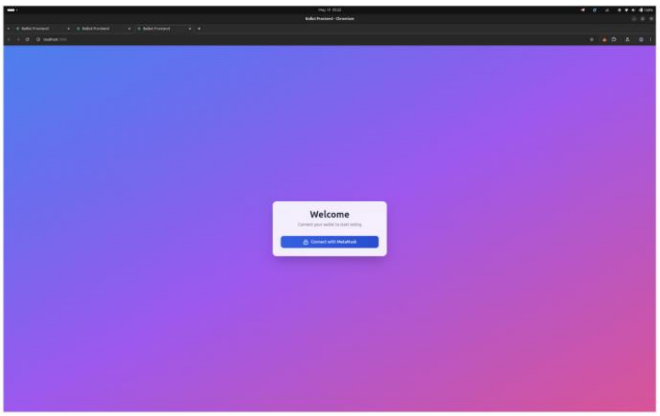


рис. 1 Екранна форма авторизації

12

ЕКРАННІ ФОРМИ

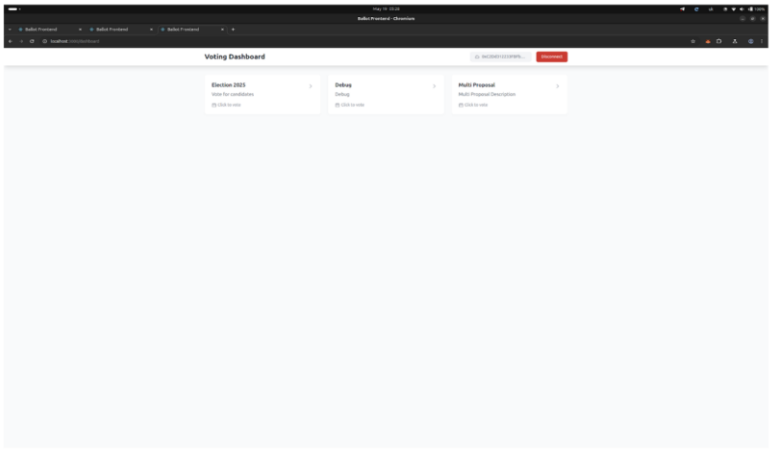


рис. 2 Екранна форма що дозволяє переглянути доступні голосування

ЕКРАННІ ФОРМИ

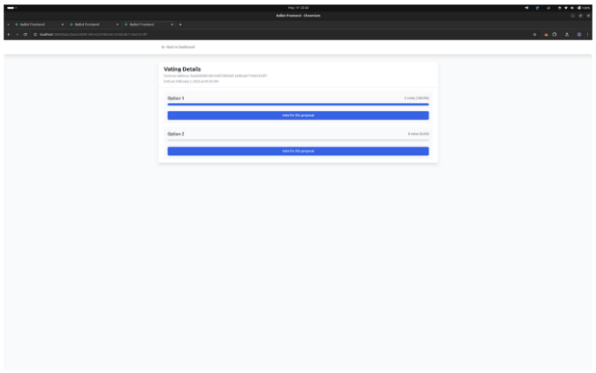


рис. 3 Екранна форма що дозволяє переглянути деталі голосування

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Іщук М.О., Шевченко С.М. Можливості застосування технології блокчейн для проведення електронних голосувань // Матеріали Всеукраїнської науково-практичної Інтернет-конференції «Стратегії кіберстійкості: управління ризиками та безперервність бізнесу» 27.02.2025, ДУІКТ, Київ, С. 200-204
2. Іщук М.О., Шевченко С.М. Розробка web-сайту для проведення електронних голосувань з використанням блокчейн технологій // Апаратне і програмне забезпечення інформаційних технологій. Збірник матеріалів Всеукраїнської конференції молодих учених "Інформаційні технології" (м. Київ, 15 травня 2025 року) / Факультет інформаційних технологій та математики Київського столичного університету імені Бориса Грінченка. Київ, 2025. – С. 131-133.
3. Іщук М.О., Шевченко С.М. Порівняльний аналіз ефективності різних блокчейн-платформ для електронного голосування // «Кібербезпека: освіта, наука, техніка», 2025, №2 (подано до друку).

ВИСНОВКИ

1. Виконано аналіз актуальних вимог до веб-сайтів, зокрема розглянуто аспекти безпеки, продуктивності, масштабованості, а також основні принципи UI/UX-дизайну.
2. Досліджено фундаментальні принципи електронного голосування, зокрема вимоги до безпеки, анонімності, цілісності та прозорості; детально розглянута технологія блокчейн, її принципи децентралізації, незмінності даних та механізм консенсусу.
3.Проведено огляд прикладів впровадження електронного голосування в окремих країнах, зокрема Естонії та Швейцарії. Визначено характерні риси, сильні сторони та потенційні обмеження різних підходів, включаючи використання смарт-карт, електронної ідентифікації, криптографічних методів і блокчейн-технологій.
4. Сформовано функціональні та нефункціональні вимоги до web-сайту для електронного голосування. Серед основних вимог: результат голосування записується в блокчейн для забезпечення незмінності, система повинна гарантувати захист голосів від несанкціонованої модифікації
5. Досліджено інструменти та технології для розробки web-сайту для електронного голосування, а саме: React для фронтенду, Node.js для бекенду, PostgreSQL для зберігання даних, Web3.js як фреймворк для взаємодії з Web3.
6. Розроблено web-сайт для проведення електронного голосування з використанням блокчейн технологій.
7. Проведено функціональне та нефункціональне тестування системи, яке показало, що продукт відповідає зазначеним вимогам.

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

```
import { Injectable, UnauthorizedException } from
'@nestjs/common';
import { PrismaService } from '../prisma/prisma.service';
import Web3 from 'web3';
import BallotABI from '../contracts/Ballot.json';
import MulticallABI from '../contracts/Multicall3.json';
import { ConfigService } from '@nestjs/config';

@Injectable()
export class TopicsService {
  private web3: Web3;
  private account: any;
  private multicall: any;

  constructor(
    private config: ConfigService,
    private prisma: PrismaService,
  ) {
    this.web3 = new Web3(this.config.get('INFURA_URL'));
    const privateKey = this.config.get('PRIVATE_KEY');
    if (!privateKey) {
      throw new Error('PRIVATE_KEY is not defined in .env');
    }
    this.account =
this.web3.eth.accounts.privateKeyToAccount(privateKey);
    this.web3.eth.accounts.wallet.add(this.account);
    this.multicall = new this.web3.eth.Contract(
      MulticallABI,
      this.config.get('MULTICALL_ADDRESS'),
    );
  }

  async createTopic(
    title: string,
    description: string,
    proposalNames: string[],
    endDate: number,
    apiKey: string
  ) {
    if (apiKey !== this.config.get('ADMIN_API_KEY')) {
      throw new UnauthorizedException('Invalid API key');
    }
    const contract = new this.web3.eth.Contract(BallotABI.abi);
    const deployTx = contract.deploy({
      data: BallotABI.bytecode,
      arguments: [proposalNames, endDate],
    });
    const gas = await deployTx.estimateGas({ from:
this.account.address });
    const deployed = await deployTx.send({
      from: this.account.address,
      gas: gas.toString(),
    });
    const contractAddress = deployed.options.address;
    if (!contractAddress) {
      throw new Error('Contract deployment failed: no address
returned');
    }
    const topic = await this.prisma.topic.create({
      data: {
        title,
        description,
        contractAddress,
      },
    });
    return topic;
  }

  async getTopics(voterAddress: string) {
    const topics = await this.prisma.topic.findMany();
    if (topics.length === 0) return [];

    const authCalls = topics.map((topic) => ({
      target: topic.contractAddress,
      callData: this.web3.eth.abi.encodeFunctionCall(
        {
          name: 'isVoter',
          type: 'function',
          inputs: [{ type: 'address', name: 'voter' }],
        },
        [voterAddress]
      ),
    }));

    const endDateCalls = topics.map((topic) => ({
      target: topic.contractAddress,
      callData: this.web3.eth.abi.encodeFunctionCall(
        {
          name: 'getEndDate',
          type: 'function',
          inputs: [],
        },
        []
      ),
    }));

    const { returnData } = await this.multicall.methods
      .aggregate([...authCalls, ...endDateCalls])
      .call();

    const authResults = returnData.slice(0, topics.length);
    const endDateResults = returnData.slice(topics.length);

    const authorizedTopics = topics.filter((_, i) =>
      this.web3.eth.abi.decodeParameter('bool', authResults[i])
    ).map((topic, i) => ({
      ...topic,
      endDate:
Number(this.web3.eth.abi.decodeParameter('uint256',
endDateResults[i]))
    }));

    return authorizedTopics;
  }

  async getProposals(contractAddress: string) {
  console.log('getProposals called with contractAddress:',
contractAddress);
  if (!contractAddress || !/^(0x)?[0-9a-fA-
F]{40}$/.test(contractAddress)) {
    throw new Error('Failed to fetch proposals: Invalid or
missing contract address');
  }
  const contract = new this.web3.eth.Contract(BallotABI.abi,
contractAddress);
  try {
    const [proposals, endDate] = await Promise.all([
      contract.methods.getAllProposals().call(),
```

```
      contract.methods.getEndDate().call(),
    ]);
    if (!Array.isArray(proposals)) {
      throw new Error('getAllProposals did not return an array');
    }
    return {
      proposals: proposals.map((p: { name: string; voteCount:
string }) => ({
        name: p.name,
        voteCount: Number(p.voteCount),
      })),
      endDate: Number(endDate) * 1000,
    };
  } catch (error: any) {
    throw new Error(`Failed to fetch proposals:
${error.message}`);
  }
}

async vote(contractAddress: string, proposalIndex: number,
voterAddress: string) {
  const contract = new this.web3.eth.Contract(BallotABI.abi,
contractAddress);
  try {
    const gas = await
contract.methods.vote(proposalIndex).estimateGas({ from:
voterAddress });
    const tx = await
contract.methods.vote(proposalIndex).send({
      from: voterAddress,
      gas: gas.toString(),
    });
    return { transactionHash: tx.transactionHash };
  } catch (error: any) {
    console.log('Vote error:', JSON.stringify(error, null, 2));
    const errorData = error.data || error.innerError?.data ||
error.cause?.data;
    if (errorData) {
      const selector = errorData.slice(0, 10);
      console.log('Error selector:', selector);
      if (selector === '0x27ead06f') {
        throw new Error('Voter has already voted');
      }
      const errors = BallotABI.abi.filter((item: any) => item.type
=== 'error');
      for (const err of errors) {
        const errorSignature =
`${err.name}(${err.inputs.map((input: any) =>
input.type).join(',')})`;
        const expectedSelector =
this.web3.utils.keccak256(errorSignature).slice(0, 10);
        console.log(`Checking error: ${err.name}, expected
selector: ${expectedSelector}`);
        if (expectedSelector && selector === expectedSelector) {
          try {
            const params: { [key: string]: unknown } =
err.inputs.length
              ? this.web3.eth.abi.decodeParameters(err.inputs,
errorData.slice(10))
              : {};
            console.log('Decoded params:', params);
            if (err.name === 'VotingClosed') {
              const endDate = typeof params[0] === 'string' ?
parseInt(params[0]) : Number(params[0]);
              throw new Error(`Voting closed on ${new
Date(endDate * 1000).toISOString()}`);
            }
            if (err.name === 'VoterAlreadyVoted') {
              throw new Error('Voter has already voted');
            }
            if (err.name === 'VoterHasRights') {
              throw new Error('Voter already has voting rights');
            }
            if (err.name === 'OnlyChairperson') {
              throw new Error('Only the chairperson can perform
this action');
            }
            throw new Error(`${err.name}:
${JSON.stringify(params)}`);
          } catch (decodeError) {
            console.log(`Failed to decode params for ${err.name}:`,
decodeError);
            throw new Error(`Failed to decode ${err.name} error`);
          }
        }
      }
      if (selector === '0x08c379a0') {
        try {
          const data = errorData.slice(10);
          const offset = parseInt(data.slice(0, 64), 16) * 2;
          const length = parseInt(data.slice(offset, offset + 64), 16)
* 2;
          const encodedString = data.slice(offset + 64, offset + 64
+ length);
          const decoded = Buffer.from(encodedString,
'hex').toString('utf8');
          if (decoded === 'Already voted.') {
            throw new Error('Voter has already voted');          }
          if (decoded === 'Has no right to vote') {
            throw new Error('Voter has no right to vote');
          }
          throw new Error(decoded);
        } catch (decodeError) {
          console.log('Failed to decode string revert:',
decodeError);
          throw new Error('Failed to decode contract error');
        }
      }
      console.log('Unhandled selector:', selector);
    }
    throw new Error('Failed to vote');
  }
}

async giveVotingRight(contractAddress: string, voterAddress:
string, apiKey: string) {
  if (apiKey !== this.config.get('ADMIN_API_KEY')) {
    throw new UnauthorizedException('Invalid API key');
  }

  const contract = new this.web3.eth.Contract(BallotABI.abi,
contractAddress);
  try {
    const gas = await
contract.methods.giveRightToVote(voterAddress).estimateGas
({ from: this.account.address });
    const tx = await
contract.methods.giveRightToVote(voterAddress).send({
      from: this.account.address,
      gas: gas.toString(),
    });
    return { transactionHash: tx.transactionHash };
  } catch (error: any) {
    throw new Error(`Failed to give voting right:
${error.message}`);
  }
}

async bulkGiveVotingRights(contractAddress: string,
voterAddresses: string[], apiKey: string) {
```

```
  if (apiKey !== this.config.get('ADMIN_API_KEY')) {
    throw new UnauthorizedException('Invalid API key');
  }

  const contract = new this.web3.eth.Contract(BallotABI.abi,
contractAddress);
  try {
    const gas = await
contract.methods.bulkGiveRightToVote(voterAddresses).estim
ateGas({ from: this.account.address });
    const tx = await
contract.methods.bulkGiveRightToVote(voterAddresses).send(
{
      from: this.account.address,
      gas: gas.toString(),
    });
    return { transactionHash: tx.transactionHash };
  } catch (error: any) {
    throw new Error(`Failed to give voting rights:
${error.message}`);
  }
}

async getVoteData(contractAddress: string, proposalIndex:
number, voterAddress: string) {
  const contract = new this.web3.eth.Contract(BallotABI.abi,
contractAddress);
  try {
    const voterData = await
contract.methods.voters(voterAddress).call() as { voted:
boolean };
    if (voterData.voted) {
      const error = new Error('Already voted.');
      error['code'] = 'ALREADY_VOTED';
      throw error;
    }

    const hasRights = await
contract.methods.isVoter(voterAddress).call();
    if (!hasRights) {
      const error = new Error('Has no right to vote');
      error['code'] = 'NO_VOTING_RIGHTS';
      throw error;
    }

    const endDate = await contract.methods.getEndDate().call();
    if (Date.now() / 1000 > Number(endDate)) {
      const error = new Error(`Voting closed on ${new
Date(Number(endDate) * 1000).toISOString()}`);
      error['code'] = 'VOTING_CLOSED';
      throw error;
    }

    const data =
contract.methods.vote(proposalIndex).encodeABI();
    const gas = await
contract.methods.vote(proposalIndex).estimateGas({ from:
voterAddress });

    return {
      data,
      gas: gas.toString()
    };
  } catch (error: any) {
    console.error('Smart contract error:', error);

    if (error.code) {
      throw error;
    }

    if (error.message.includes('revert')) {
      if (error.message.includes('Already voted')) {
        const customError = new Error('Already voted.');
        customError['code'] = 'ALREADY_VOTED';
        throw customError;
      } else if (error.message.includes('Has no right to vote')) {
        const customError = new Error('Has no right to vote');
        customError['code'] = 'NO_VOTING_RIGHTS';
        throw customError;
      } else if (error.message.includes('Voting closed')) {
        const customError = new Error('Voting period has ended');
        customError['code'] = 'VOTING_CLOSED';
        throw customError;
      }
    }

    const wrappedError = new Error(`Smart contract
error: ${error.message}`);
    wrappedError['code'] = 'CONTRACT_ERROR';
    throw wrappedError;
  }
}

async deleteTopic(contractAddress: string, apiKey:
string) {
  if (apiKey !==
this.config.get('ADMIN_API_KEY')) {
    throw new UnauthorizedException('Invalid API
key');
  }
  return this.prisma.topic.delete({ where: {
contractAddress } });
}

}
```