НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»
на тему:

«Реалізація програмного рішення інформаційної системи обліку енергоресурсів з використанням технології блокчейн»

Виконав: студент 4 курсу, групи ТМ-61

Шаповал Віталій Олегович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к.е.н., доцент Сегеда І.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент д.т.н., професор завідувач кафедри ТПТ Варламов Г.Б. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту
Шаповалу Віталію Олеговичу
(прізвище, ім’я, по батькові)

1. Тема роботи Реалізація програмного рішення інформаційної системи обліку енергоресурсів з використанням технології блокчейн

керівник роботи к.е.н., доцент Сегеда Ірина Василівна
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи: ”10” червня 2019р.
3. Вихідні дані до роботи: мова програмування — Javascript, середовище розробки — MS Visual Studio Code
4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити): провести аналіз предметної області, обрати технології для розробки, спроектувати моделі даних, що будуть використовуватися в системі, запропонувати архітектурне рішення системи, спроектувати серверну реалізацію, спроектувати графічний інтерфейс користувача та міжсерверну комунікацію
5. Перелік ілюстративного матеріалу: 1. Архітектура розподіленої системи. 2. Технології та засоби розробки. 3. Складові системи. 4. Формат транзакції. 5. Формат блоку. 6. Валідація та майнинг блоку. 7. Завантажуючий вузол. 8. Створення новогокористувача. 9. Авторизація користувача та створення транзакції. 10. Перегляд транзакцій користувача.
6. Дата видачі завдання «11» жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	12.02.2020 – 18.03.2020	
2	Розробка архітектури та загальної структури системи	27.03.2020 – 03.04.2020	
3.	Розробка структур окремих підсистем	09.04.2020 – 12.04.2020	
4.	Програмна реалізація системи	13.04.2020 – 17.05.2020	
5.	Оформлення пояснювальної записки	19.05.2020 – 04.06.2020	
6.	Захист програмного продукту	18.05.2020	
7.	Передзахист	1.06.2020 – 05.06.2020	
8.	Захист	15.06.2020 – 19.06.2020	

Студент _____ Шаповал В.О.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Сегеда І.В.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

В даній роботі приведено програмне рішення автоматизованої системи з обліку енергоресурсів на основі технології блокчейн. Описано задачу та можливий спосіб її вирішення у вигляді спроектованої архітектури та виконаного програмного рішення.

Записка містить 42 сторінки, 24 рисунки, 1 таблицю, 5 додатків і 44 бібліографічних найменування за «Переліком посилань».

Результатом апробації даної роботи є стаття в науковому журналі.

Ключові слова: блокчейн, блок, транзакція, майнинг, однорангова децентралізована система, Javascript, Node.js, React.js, MongoDB, JSON, REST.

ABSTRACT

This work represents a software solution to the power engineering resources accounting and selling based on blockchain technology. The problem and a possible solution as an project’s architecture and implemented software are depicted.

Aa a result of this work, a research article has already been published.

The note contains 42 pages, 24 figures 1 table, 5 attachments and 44 links.

Keywords: blockchain, block, transaction, mining, peer-to-peer decentralized system, Javascript, Node.js, React.js, MongoDB, JSON, REST

ЗМІСТ

Перелік умовних позначень, скорочень і термінів 7

Вступ 8

1. Задача розробки автоматизованої системи з обілку та продажу енергоресурсів на основі блокчейну 9

2. Розгляд та аналіз алгоритму роботи блокчейну .....**Ошибка! Закладка не определена.**

3. Засоби розробки .....**Ошибка! Закладка не определена.**

3.1 Обґрунтування вибору технологій та їх опис .....**Ошибка! Закладка не определена.**

3.2 Графічний інтерфейс користувача .....**Ошибка! Закладка не определена.**

3.3 Серверна частина 20

3.3 Документоорієнтована система управління базами даних з відкритим вихідном кодом MongoDB 24

4. Опис програмної реалізації автоматизованої системи 27

4.1 Опис архітектурного рішення 27

4.2 Механізм авторизації та аутентифікації користувачів за допомогою JSON Web tokens.....**Ошибка! Закладка не определена.**

4.3 Алгоритм валідації нового блоку .....**Ошибка! Закладка не определена.**

4.4 Опис завантажуючого (bootstrapper) сервера .....**Ошибка! Закладка не определена.**

4.4.1 Процес запуску завантажуючого серверу .....**Ошибка! Закладка не определена.**

4.4.2 Функціонал завантажуючого серверу ....**Ошибка! Закладка не определена.**

4.5 Опис сервера майнера 37

4.5.1 Процес запуску сервера майнера 37

4.5.2 Функціонал сервера майнера 37

4.6 Опис сервера користувача 38

4.6.1 Процес запуску сервера користувача 38
4.6.2 Функціонал серверу користувача 39
5. Робота користувача з програмною системою 41
5.1 Системні вимоги для додаткове програмне забезпечення 41
5.2 Результати виконання програми 41
Висновки 44
Список використаних джерел 45
Додаток 1 49
Додаток 2 51
Додаток 3 526
Додаток 4 75
Додаток 5 80

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Blockchain (блокчейн) — архітектурний принцип збереження даних у формі блоків, пов’язаних один із одним.

Block (блок) — одиниця блокчейну, яка містить в собі транзакції.

Transaction (транзакція) — одиниця блоку, яка зберігає корисні дані.

Smart contract (розумний контракт) — інформаційна одиниця, яка зберігає в собі дані про домовленості між сторонами.

Mining (майнинг) — процес обчислення хеш-значення методом перебору

React.js — фреймоврк, базований на мові Javascript для створення графічних інтерфейсів користувача.

Node.js — фреймоврк, базований на мові Javascript для створення веб-серверів.

REST (Representational state transfer) — архітектурна концепція передачі даних.

MongoDB — нереляційна база даних, зі слабко вираженими зв’язками між даними.

JSON (JavaScript Object Notation) — спосіб представлення даних.

ВСТУП

Розвиток IT-технологій щодня пропонує світові нові інструменти для оптимізації бізнес-процесів. Одним з останніх таких інструментів є технологія ─ блокчейн (blockchain technology).

Блокчейн ─ технологія здатна докорінно змінити енергетичну систему, спочатку шляхом трансформації окремих секторів і, нарешті, шляхом трансформації всього ринку електроенергії.

Основна перевага даної технології — однорангова взаємодія між користувачами. Іншими словами, користувачі мають можливість здійснювати передачу цінної інформації або грошові перекази один між одним без посередників, що потенціально може зменшити ціну на товари, якими торгують за допомогою блокчейну. Даний підхід вимагає спеціальних алгоритмів та особливої архітектури.

Мета даної роботи полягає в проектуванні і створенні повноцінної автоматизованої системи з обліку та продажу енергоресурсів, яка змогла би запропонувати альтернативу обліку і продажу енергоресурсів на ринку.

При проектуванні системи застосовано технології React.js, Node.js, MongoDB.

У першому розділі описано поточний стан на ринку енергоресурсів та можливі способи удосконалення торгівлі енергоресурсами. У другому розділі стисло описано принцип роботи блокчейну. В третьому представлено технології, які були використані при проектуванні та створенні автоматизованої системи. Четвертий розділ демонструє програмне рішення з детальним описом технічних деталей. П’ятий розділ показує взаємодію користувача із створеною системою.

1 ЗАДАЧА РОЗРОБКИ АВТОМАТИЗОВАНОЇ СИСТЕМИ І ОБЛІКУ ЕНЕРГОРЕСУРСІВ НА ОСНОВІ БЛОКЧЕЙНУ

Думки експертів щодо ідеї впровадження криптовалют розділилися: одні вважають це справді новим етапом, але не зовсім зрозуміло, чи буде дане рішення революційним. Другі — це інновації, які потребують значної адаптації. Тому питання подолання багатозначності і практичного використання є актуальними та потребують більшого дослідження.

В сучасних умовах прийнята технологія обліку і контролю енергоресурсів застаріла через організаційну та технічну недосконаліть структур. Ці проблеми стають причиною постійних збитків, що показує про необхідність створення нової автоматизованої системи.

Метою дослідження є аналіз перспектив та варіантів використання блокчейн технологій в енергетиці та розгляд системи обліку споживання енергоресурсів.

Міжнародні енергетичні компанії розробляють проекти, які надалі з'єднають усіх споживачів в одну мережу — децентралізовану систему. Існуюча багаторівнева система складається з виробників електроенергії, операторів розподільної мережі, операторів-постачальників, постачальників платіжних послуг банківських послуг, споживачів та трейдерів. Усі транзакції щодо отримання та оплати за енергію здійснюватимуться в мережі, об'єднуючи учасників — виробників та споживачів енергії. Це зробить енергію дешевшою.

Всі транзакції будуть відкритими. Люди не зможуть прострочити платіж за споживання енергії — буде контролюватися виконання всіх операцій. Система сама заплатить за себе, тобто спише стільки криптовалюти, скільки Вам знадобиться для транзакції по передачі енергії.

Завдяки блокчейну всі дії будуть захищені від сторонніх користувачів. Це

дозволить сертифікувати електроенергію, перевірити квоти на допустимі викиди. Ця технологія функціонує як база даних транзакцій, тому за допомогою блокчейну можна створити архів для збереження всіх даних за виставленими рахунками за електроенергію. Споживачі отримають можливість контролю за своїми договорами на постачання електроенергії, а також дані про споживання електроенергії. Усі записи зберігатимуться у відкритому доступі в блокчейні, який буде коригувати всі питання права власності та поточний стан активів.

Технологія блокчейну, крім того, що використовується для проведення операцій з постачання енергії, може бути основою для процесів вимірювання кількості споживаної електроенергії, формування рахунків за спожиту енергію та подальшу їх оплату. Інші можливі додатки включають право власності на активи, управління активами, систему сертифікатів квоти на викиди вуглекислого газу та сертифікати, що підтверджують виробництво електроенергії на основі використання відновлюваних джерел енергії (ВДЕ). Можливості використання технологій в енергетиці представлені в таблиці 1:

Таблиця 1 —Варіанти використання блокчейну в енергетиці

<table>
<tr><th>Транзакції і «розумні контракти»</th><th>Права власності на активи і управління ними</th><th>Децентралізовані інформаційні системи</th></tr>
<tr><td>Децентралізована торгівля</td><td>Реєстрація власності та ведення реєстру активів</td><td>Облік електроспоживання та виставлення рахунків за електроенергію</td></tr>
<tr><td>Можливості для просьюмерів</td><td>«Зелені» сертифікати</td><td>Облік споживання тепла і виставлення рахунків за нього</td></tr>
<tr><td>Впровадження криптовалют</td><td rowspan="3">Квоти на викиди вуглекислого газу і сертифікація виробництва електроенергії на основі відновлюваних джерел енергії</td><td rowspan="3">Оплата зарядки електромобілів</td></tr>
<tr><td>Зарядка електромобілів</td></tr>
<tr><td>IOT</td></tr>
</table>

Існуючі блокчейн додатки можна розділити на три великі категорії залежно від рівня розробки: додатки версій 1.0, 2.0 та 3.0. Blockchain 3.0 — це етап розвитку технологій, на якому здійснюється подальший розвиток концепції "смарт контракту" з метою створення децентралізованих, автономних організаційних підсистем, які керуються власними законами та працюють майже незалежно. Децентралізована система енергетичних транзакцій та постачання енергії представлена на рисунку 1.1.

Рисунок 1.1 — Децентралізована система енергетичних транзакцій і енергопостачання

Звернемо увагу на проблему обліку споживання та сбір платежів за енергоресурси. Технологія обліку і контролю енергоресурсів, яка сьогодні застосовується, завдає шкоди суб'єктам господарювання. Головна причина це

організаційна і технічна деградація структур. Ці проблеми стають причиною постійних збитків, що свідчить про необхідність створення нової автоматизованої системи.

Нижче наводяться вимоги, які необхідно виконати для створення повноцінної автоматизованої системи.

- Система має проводити облік та торгівлю енергоресурсів за принципами блокчейну.
- Система повинна бути децентралізована;
- Користувач повинен керувати лише власними даними;
- Користувач має можливість переглядати будь-які дані блокчейну;
- Користувач може вносити цінну інформацію і проводити оплату за енергоресурси за допомогою криптовалюти.

Крім того, система повинна мати зручний та сучасний графічний інтерфейс користувача з можливістю працювати у веб-браузері та бути імплементована відповідно до шаблонів задля гнучкого внесення змін та додавання нового функціоналу.

2 РОЗГЛЯД ТА АНАЛІЗ АЛГОРИТМУ РОБОТИ БЛОКЧЕЙНУ

Головний принцип блокчейн технології – перманентне обчислення зашифрованої хеш-функції. Хешування – перетворення n-бітового рядку (n – задається) довільного входу за допомогою хеш-функції , наприклад, у 256-бітний хеш-рядок. При зміні вхідного повідомлення (хоч на знак!), результат, що отримується застосуванням хеш-функції – повністю змінюється (забезпечується стійкість криптокоду).

Приклад хеш-функції – діленням входу на поліном по модулю 2:

$$h(n, m) = nmod(m)$$

де n – вхід («ключ»), m – кількість входів («хешів»), mod – цілочисельне ділення

Можна хеш-функцію представити набором коефіцієнтів полінома. Для цього ділимо дані (вхід) по модулю 2, m – ступінь 2, бінарні ключі

$$K = K_{n-1}K_{n-2} \dots K_0$$

представляють поліномами, а хеш-код – значеннями коефіцієнтів

$$a_{m-1}, a_{m-2}, \dots, a_0$$

Полінома, отриманого як залишок ділення полінома K(x) на інший, що задається поліном ступеня m:

$$K(x)modP(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \dots + a_1x^1 + a_0$$

$$h(x) = a_{m-1}, a_{m-2}, \dots, a_1 a_0$$

Майнинг блоків базується на криптоалгоритмі SHA-256 (обчислення криптокоду по хеш-функції зі значеннями 256-бітових рядків).

Як висновок, можна зазначити, що дані можливо вносити без участі посередників. Ввесь ланцюг розподілений між комп'ютерами по всьому світу. Центральний сервер, який було б можливо зламати не існує, щоб нанести шкоду системі. Нова технологія дозволяє продавцю і покупцю електроенергії підключившись до мережі, напряму взаємодіяти один із одним через Інтернет, проводячи грошові розрахунки. Традиційні посередники, такі як банки, платіжні системи в даній моделі не потрібні, оскільки всі абоненти мережі виступають свідками транзакцій і можуть підтвердити її деталі.

3 ЗАСОБИ РОЗРОБКИ

Базовою концепцією при проектуванні автоматизованої системи було забезпечення масштабованості, гнучкості та надійності програмного продукту з можливістю подальшого удосконалення або додавання функціоналу. Саме тому було обрано мову програмування Javascript та базу даних MongoDB.

3.1 Обґрунтування вибору технологій та їх опис

Прототип системи для дослідження можливостей концепції з мінімальними часовими затратами було виконано мовою Java. Реалізація системи була здійснена за допомогою мови програмування Javascript, де серверна частина написана за допомогою фреймворку Node.js, графічний інтерфейс користувача – React.js. Вище зазначені фреймворки дозволяють писати шаблонізований код, який простіше супроводжувати і змінювати у порівнянні із ситуацією без їх використання.

Вказані вище технології базуються на мові програмування Javascript, в них одна екосистема, широкий вибір додаткових бібліотек з простотою їх підключення. Мова Javascript швидка у вивченні і зручна для використання.

Автоматизована система представлена у вигляді веб додатку, взаємодія з яким забезпечена за допомогою веб-браузеру. Припускається, що система буде працювати в мережі Інтернет. Вся взаємодія між користувачем та системою в мережі Інтернет відбувається за допомогою протоколу HTTPS. Передача даних здійснюється на основі архітектурного шаблону REST, де всі операції маніпулювання з даними в рамках Інтернету виконуються за допомогою 4 типів HTTP запитів:

- GET – на отримання інформації;
- POST – на збереження інформації;
- PUT – на оновлення інформації;

- DELETE - на видалення інформації.

Корисна інформація в рамках HTTP запитів передається у форматі JSON.

Реалізована система використовує перші два типи запитів.

Сховищем даних було вирішено обрати NoSQL (нереляційну) базу даних MongoDB. Як відомо, нереляційні бази не зберігають логічної цілісності даних, тому для роботи з ними необхідно забезпечувати додаткові застережні методи валідації та контролю збереженої інформації. Цей факт також унеможливлює наявність транзакцій в рамках терміну SQL, але нереляційні БД не мають механізмів забезпечення логічної цілісності і тому їх швидкодія більше, вони більш гнучкі в проектах з важко формалізованими вимогами, прості у масштабуванні (як у горизонтальному, так і у вертикальному), що важливо для децентралізованої блокчейн-системи.

3.2 Графічний інтерфейс користувача

Наразі найбільшого використання отримали односторінкові веб-застосунки (SPA). Веб-застосунок - тип сайту, який вміщується на одній сторінці з метою забезпечити користувачу досвід близький до користування настільною програмою.

Для спрощення їх побудови використовуються бібліотеки та фреймворки. Одним з найпопулярніших інструментів, що використовується для побудови веб-застосунків даного типу є React.js.

React.js - це бібліотека для створення користувацьких інтерфейсів. Її головне завдання - спростити і автоматизувати написання веб UI інтерфейсів.

React значно полегшує створення інтерфейсів завдяки ефективному підходу розбиття кожної сторінки на невеликі фрагменти. Фрагменти називаються компонентами.

Кожен виділений фрагмент сторінки, показаної на малюнку, є компонентом (рисунок 3.1).

Рисунок 3.1 — Приклад розбивки сторінки на компоненти

Компонент React - це ділянка коду, який представляє частину веб-сторінки. Кожен компонент - це JavaScript функція, яка повертає частину коду, що представляє фрагмент сторінки [7].

Для формування сторінки ці функції викликаються в певному порядку. Після цього зібраний воєдино з компонентів результат відображається користувачеві

React розроблений навколо концепції багаторазових компонентів. Будуються невеликі за розміром та логікою невеликі компоненти, які відповідають та описують одну функцію. Далі вони поєднуються, щоб сформувати більші компоненти. Всі компоненти, маленькі чи великі, можуть використовуватися повторно, навіть у різних проектах.

Компоненти бувають 2 типів:

- Презентаційні (dummy) – відповідають за відображення контенту;
- Контейнери (smart) – відповідають за надання логіки і даних для презентації компонентів.

```
const App = ({ location }) => (
    <div className="ui-container">
        <Route path="/" exact component={LoginPage} />
        <Route location={location} exact path="/login" component={LoginPage} />
        <Route location={location} exact path="/dashboard" component={Dashboard} />
    </div>
)
```

Рисунок 3.2 — Приклад компонента

React використовує мову розмітки, так званий JSX, який схожий на мову розмітки HTML, але працює всередині мови JavaScript, що відрізняє його від HTML.

Проте в браузері використовується скомпільовану версію. JSX - це технологія, яка дозволяє нам розробляти компоненти React використовуючи HTML-подібний синтаксис (рисунок 3.3).

```
render() {
    return (
        <div className="login-page">
            <Grid centered>
                <Grid.Row>
                    <Grid.Column mobile={14} tablet={10} widescreen={6} largeScreen={6}>
                        <h1 className="login-h1">Login Page</h1>
                        <LoginForm submit={this.submit} />
                    </Grid.Column>
                </Grid.Row>
            </Grid>
        </div>
    );
}
```

Рисунок 3.3 — Приклад розмітки

Компоненти React можна поміщати в інші компоненти. Саме так сторінки збирають з фрагментів, написаних на React - вкладаючи компоненти один в одного. В компонент LoginPage поміщений компонент LoginForm (рисунок 3.4).

Рисунок 3.4 — LoginPage компонент

Стан - це інструмент, що дозволяє оновлювати призначений для користувача інтерфейс, ґрунтуючись на події [7]. У кожного компоненту є власний стан. У стані зберігається внутрішні дані компонента, при зміні яких відбувається перемалювання компонента (і вкладених в нього компонентів, якщо вони є) (рисунок 3.5).

```
state = {
    table: [],
    user: {
        id: localStorage.getItem("userToken")
    },
    loading: false,
    isLeftMenuActive: false,
    activeTable: ""
}
```

Рисунок 3.5 — Приклад стану компонента

Коли користувач клацає на кнопку login, відбувається оновлення стану компонента і з цієї причини запит на сервер буде відправлений.

Компоненти можуть «спілкуватися» один з одним. Можлива передача даних від одного компонента до іншого через властивості (props).

Властивості - це інформація, колективно використовувана батьківським компонентом і компонентами-нащадками (рисунок 3.6).

```
const App = ({ location }) => (
    <div className="ui-container">
        <Route path="/" exact component={LoginPage} />
        <Route location={location} exact path="/login" component={LoginPage} />
        <Route location={location} exact path="/dashboard" component={Dashboard} />
    </div>
)
```

Рисунок 3.6 — path, exact, component - приклад властивостей компонентів

Таким чином можна реалізувати інкапсуляцію даних. Вкладений компонент отримує тільки необхідні йому дані для роботи через властивості. Це дозволяє створювати слабозв’язні компоненти і перевикористовувати їх кілька разів у проекті за необхідністю.

3.3 Серверна частина

Node.js призначений для створення масштабованих мережевих додатків як асинхронна машина виконання JavaScript. Після кожного з'єднання зворотний виклик буде знято, але якщо роботи не буде виконано то Node.js перейде у режим сну.

Це контрастує на відміну від більш поширеної моделі сучасності, в якій використовуються потоки ОС. Мережа на основі потоків порівняно неефективна і дуже складна у використанні. Крім того, користувачі Node.js звільняються від турботи про “dead locking” процесу, оскільки блокувань взагалі немає. Майже жодна функція в Node.js безпосередньо не виконує введення-виведення, тому процес ніколи не блокується. Оскільки нічого не блокує, масштабовані системи дуже резонно розробляти з використанням Node.js.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Рисунок 3.7 — Приклад створення серверу

Node.js схожий за дизайном та під впливом таких систем, як Ruby's Event Machine і Python's Twisted. Node.js розширює поняття моделі подій. Він представляє цикл подій як конструкцію виконання, а не як бібліотеку. В інших системах завжди є виклик блокування для запуску циклу подій. Зазвичай поведінка визначається через зворотні виклики на початку сценарію, а в кінці сервер запускається через блокуючий виклик. У Node.js немає такого виклику циклу запуску події. Node.js просто входить у цикл подій після виконання сценарію введення і виходить з циклу подій, коли більше немає зворотних викликів для виконання. Така поведінка схожа на JavaScript браузера — цикл подій прихований від користувача.

HTTP — це модуль у Node.js, розроблений з урахуванням потокової та низької затримки. Це робить Node.js добре підходящим для створення веб-бібліотеки чи фреймворку.

Node.js, розроблений без потоків, це означає, що не можливо скористатися кількома ядрами в оточенні. Дочірні процеси можна породжувати за допомогою API

child_process.fork і вони розроблені так, щоб було легко комунікувати. На цьому ж інтерфейсі побудований модуль кластера, який дозволяє обмінюватися сокетами між процесами, щоб забезпечити балансування навантаження над вашими ядрами.

Блокування — це коли виконання додаткового JavaScript коду у процесі Node.js має зачекати, поки не завершиться операція, що не стосується JavaScript. Це трапляється тому, що цикл подій не в змозі продовжувати працювати під час операції блокування.

У Node.js операції, які демонструють низьку продуктивність через інтенсивність процесора, а не очікування операції, наприклад, вводу / виводу, зазвичай не відноситься до блокуючих. Синхронні методи в стандартній бібліотеці Node.js, які використовують libuv, є найбільш часто використовуваними операціями блокування.

Усі методи вводу-виводу в стандартній бібліотеці Node.js забезпечують асинхронні версії, які не блокують, і приймають функції зворотного виклику (рисунки 3.8 – 3.9)

```
const fs = require('fs');
const data = fs.readFileSync('/file.md'); // blocks here until file is read
```

Рисунок 3.8 — Приклад читання файлу

```
const fs = require('fs');
fs.readFile('/file.md', (err, data) => {
  if (err) throw err;
});
```

Рисунок 3.9 — Приклад читання файлу із зворотним викликом

Будь-який веб додаток рано чи пізно повинен створити об’єкт веб-сервера. Це можливо зробити, використовуючи метод http.createServer(). Функція, яка

передається в createServer викликається для кожного HTTP запиту який надходить до серверу, тому її називають обробником запиту. Насправді, об’єкт Server, що повертає createServer є емітером події.

Аналогічно NodeJS має зручні засоби для обробки заголовків запитів, робота з помилками. Код для серверу, який приймає HTTP запит, оброблює тіло запиту та обробляє помилки, наведено нижче:

```
const http = require('http');

http.createServer((request, response) => {
  const { headers, method, url } = request;
  let body = [];
  request.on('error', (err) => {
    console.error(err);
  }).on('data', (chunk) => {
    body.push(chunk);
  }).on('end', () => {
    body = Buffer.concat(body).toString();
    // BEGINNING OF NEW STUFF

    response.on('error', (err) => {
      console.error(err);
    });

    response.statusCode = 200;
    response.setHeader('Content-Type', 'application/json');
    // Note: the 2 lines above could be replaced with this next one:
    // response.writeHead(200, {'Content-Type': 'application/json'})

    const responseBody = { headers, method, url, body };

    response.write(JSON.stringify(responseBody));
    response.end();
    // Note: the 2 lines above could be replaced with this next one:
    // response.end(JSON.stringify(responseBody))

    // END OF NEW STUFF
  });
}).listen(8080);
```

Рисунок 3.10 — Приклад створення серверу для оброки HTTP запиту

Отже внаслідок того, що технологія Node.js як відносно простий інструмент для розробки системи до якої вимагається наявність потенціалу з масштабування, готовність до внесення архітектурних змін із найменшими втратами часу найкраще підходить до поставленої задачі.

3.4 Документоорієнтована система управління базами даних з відкритим вихідним кодом MongoDB

MongoDB - це база даних загального призначення, заснована на документах, створена для сучасного застосування.

MongoDB реалізує новий підхід до побудови базових даних, де немає таблиць, схем, запитів SQL, наявних ключів і багатьох інших речей, які наявні в об'єктах даних, що мають абсолютно реальне значення.

Відмінність від реляційних даних бази MongoDB пропонує документоорієнтовану модель даних, завдяки чому MongoDB в багатьох випадках працює швидше, маючи кращу спроможність до масштабування.

Але навіть зважаючи на всі недоліки реляційних баз даних і переваги MongoDB, важливо врахувати, що задачі бувають різні, як і методи їх рішення. У якомусь випадку ситуація MongoDB дійсно спрощує розробку, наприклад, якщо потрібно зберігати складні дані за структурою. У іншій ситуації найкраще використовувати традиційні реляційні бази даних. Крім того, можна використовувати смішний підхід: зберігати один тип даних у MongoDB, а інший тип даних - у традиційних БД.

Вся система MongoDB може представляти не тільки одну базу даних, що знаходиться на одному фізичному сервері. Функціональність MongoDB дозволяє розташувати кілька баз даних на декількох фізичних серверах, і ці бази даних зможуть легко обмінюватися даними і зберігати цілісність.

Одним з популярних стандартів обміну даними та їх зберігання є JSON (JavaScript Object Notation). JSON ефективно описує складні за структурою дані. Спосіб зберігання даних в MongoDB в цьому плані схожий на JSON, хоча формально

JSON не використовується. Для зберігання в MongoDB застосовується формат, який називається BSON (Бісон) або скорочення від binary JSON.

MongoDB написана на C++, тому її легко перенести на найрізноманітніші платформи. MongoDB може бути розгорнута на платформах Windows, Linux, MacOS, Solaris. Можна також завантажити вихідний код і самому скомпілювати MongoDB.

Однак при всіх відмінностях є одна особливість, яка зближує MongoDB і реляційні бази даних. У реляційних СУБД зустрічається таке поняття як первинний ключ. Це поняття описує якийсь стовпець, який має унікальні значення. У MongoDB для кожного документа є унікальний ідентифікатор, який називається _id. І якщо явно не вказати його значення, то MongoDB автоматично згенерує для нього значення.

Якщо в традиційному світі SQL є таблиці, то в світі MongoDB є колекції. І якщо в реляційних БД таблиці зберігають однотипні жорстко структуровані об'єкти, то в колекції можуть містити найрізноманітніші об'єкти, що мають різну структуру і різний набір властивостей.

Система зберігання даних в MongoDB представляє набір реплік. У цьому наборі є основний вузол, а також може бути набір вторинних вузлів. Всі вторинні вузли зберігають цілісність і автоматично оновлюються разом з оновленням головного вузла. І якщо основний вузол з якихось причин виходить з ладу, то один з вторинних вузлів стає головним.

На відміну від реляційних СУБД MongoDB дозволяє зберігати різні документи з різним набором даних, однак при цьому розмір документа обмежується 16 мб. Але MongoDB пропонує рішення — спеціальну технологію GridFS, яка дозволяє зберігати дані за розміром більше, ніж 16 мб.

Документ можна уявити як об'єкт, який зберігає деяку інформацію. У певному сенсі він подібний до рядкам в реляційних СУБД, де рядки зберігають інформацію про окремий елемент. Наприклад, типовий документ:

Документ являє набір пар ключ-значення. Ключі представляють рядки. Значення ж можуть відрізнятися за типом даних. В даному випадку у нас майже всі значення також представляють строковий тип, і лише один ключ (company) посилається на окремий об'єкт. Всього є наступні типи значень:

- string: строковий тип даних, як в наведеному вище прикладі (для рядків використовується кодування UTF-8);
- array (масив): тип даних для зберігання масивів елементів;
- binary data (двійкові дані): тип для зберігання даних в бінарному форматі;
- boolean: булевий тип даних, який зберігає логічні значення TRUE або FALSE;
- date: зберігає дату в форматі часу Unix;
- double: числовий тип даних для зберігання чисел з плаваючою точкою;
- integer: використовується для зберігання цілочисельних значень;
- javaScript: тип даних для зберігання коду javascript;
- min key / max key: використовуються для порівняння значень з найменшим / найбільшим елементів BSON;
- null: тип даних для зберігання значення Null;
- object: строковий тип даних, як в наведеному вище прикладі;
- objectID: тип даних для зберігання id документа;
- regular expression: застосовується для зберігання регулярних виразів;
- symbol: тип даних, ідентичний строковому. Використовується переважно для тих мов, в яких є спеціальні символи;
- timestamp: застосовується для зберігання часу.

Для кожного документа в MongoDB визначено унікальний ідентифікатор, який називається _id. При додаванні документа в колекцію, даний ідентифікатор створюється автоматично. Однак, розробник може сам явно задати ідентифікатор, а не покладатися на автоматично генеруються, вказавши відповідний ключ і його значення в документі.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ

Ґрунтуючись на описаних в розділі 3 технологіях, в даному описано архітектурне рішення, структура проекту, організація бази даних та алгоритми роботи серверів.

4.1 Опис архітектурного рішення

Система представлена у вигляді децентралізованої мережі серверів для кожного користувача в залежності від відведеної ролі та одного завантажуючого сервера (bootstrapper server) який виконує синхронізацію користувачів між собою. Завантажуючий сервер містить колекцію користувачів.

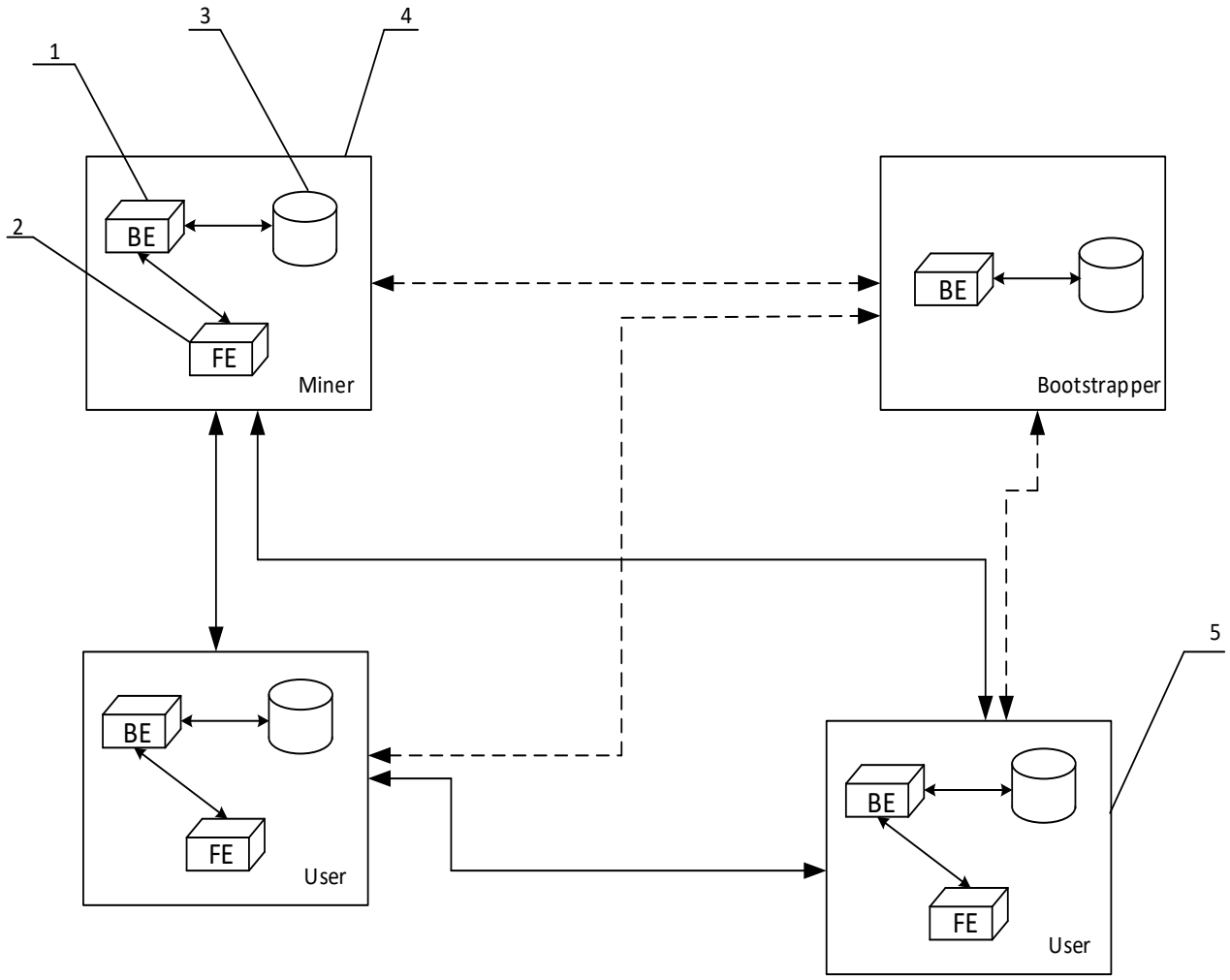
Для того, щоб почати роботу із системою, користувачеві необхідно запустити у себе на локальній машині сервер, який приймає запити і виконує обробку; та сервер, який надає графічний інтерфейс користувача.

У мережі наявно 2 ролі користувачів: користувач, що виконує майнинг блоків; користувач, що виконує створення транзакцій для оплати рахунків.

Для отримання даних про інших користувачів мережі, між користувачем та завантажуючим сервером здійснюється взаємодія. Аналогічна робота між майнером та завантажуючим сервером.

Робота між майнером та користувачем полягає у тому, що майнер отримує транзакції від користувача і після здійснення майнингу блоку здійснює поширення всім вузлам нового блоку; якщо блок пройшов валідацію і не був раніше доданий до блокчейну — він додається, майнер отримує грошову винагороду за майнинг.

Робота між користувачем та майнером полягає у тому, що користувач створює транзакцію і поширює її всім майнерам.



Умовні позначення:
1 – оброблюючий сервер (back-end server)
2 – сервер графічного інтерфейсу користувача
3 – локальна база даних користувача
4, 5 – вузол блокчейн мережі

⟷ - двонаправлений зв’язок для обміну даними в межах вузла

⟷ - двонаправлений зв’язок для обміну даними між вузлами в мережі

⇠⇢ - двонаправлений зв’язок для обміну даними між вузлом мережі та запускаючим сервером

Рисунок 4.1 — Приклад мережі з двома користувачами та одним майнером

Рисунок 4.2 — Діаграма прецедентів системи, що проектується

Рисунок 4.3 показує структуру проекту серверу користувача, а рисунки 4.4 — 4.6 показують колекції в БД MongoDB:

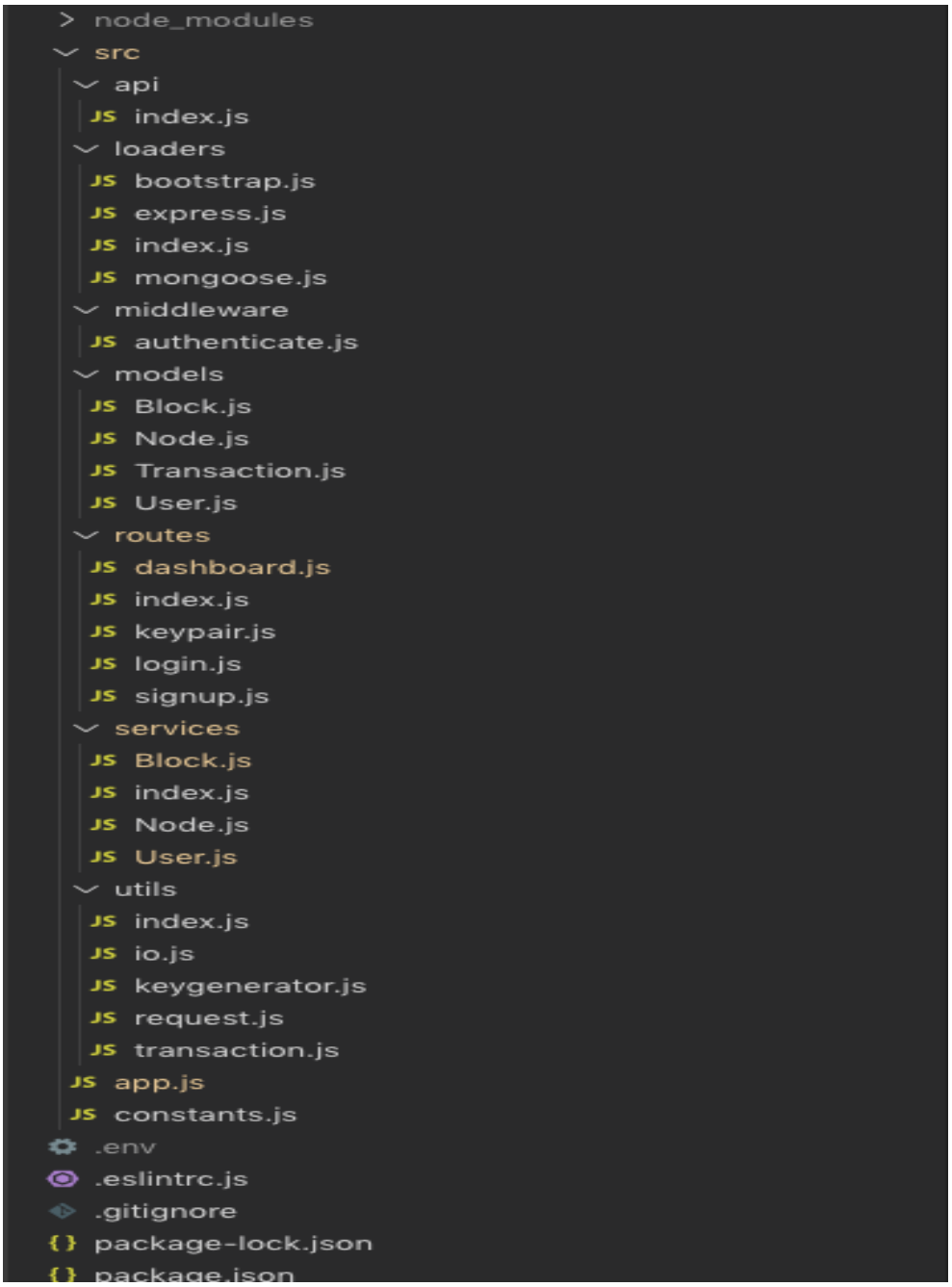


Рисунок 4.3 — Структура серверу користувача

Так як була обрана база даних MongoDB, система має 3 основних колекції: db.blockchain — колекція, що зберігає блокчейн; db.nodes— колекція, що зберігає інформацію про сервери, які працюють в мережі; db.users— колекція, що зберігає особисту інформацію про користувачів.

Структура db.blockchain:

- _id: унікальне-хеш значення яке ідентифікує запис;
- hash: текстове хеш-значення блоку;
- date: дата створення блоку;
- nonce: значення лічильника при якому майнинг блоку був завершений;
- transactions: список транзакцій у вигляді масиву;
- minerReward: числове значення, вказує яку нагороду отримає майнер після успішного завершення майнингу блоку.

Структура transaction:

- _id: унікальне-хеш значення яке ідентифікує запис;
- from: публічний ключ особи, що відправляє транзакцію;
- to: публічний ключ особи, що отримує транзакцію;
- amount: числове значення суми, що передається (у криптовалюті);
- description: текстовий опис транзакції, заповнюється користувачем.

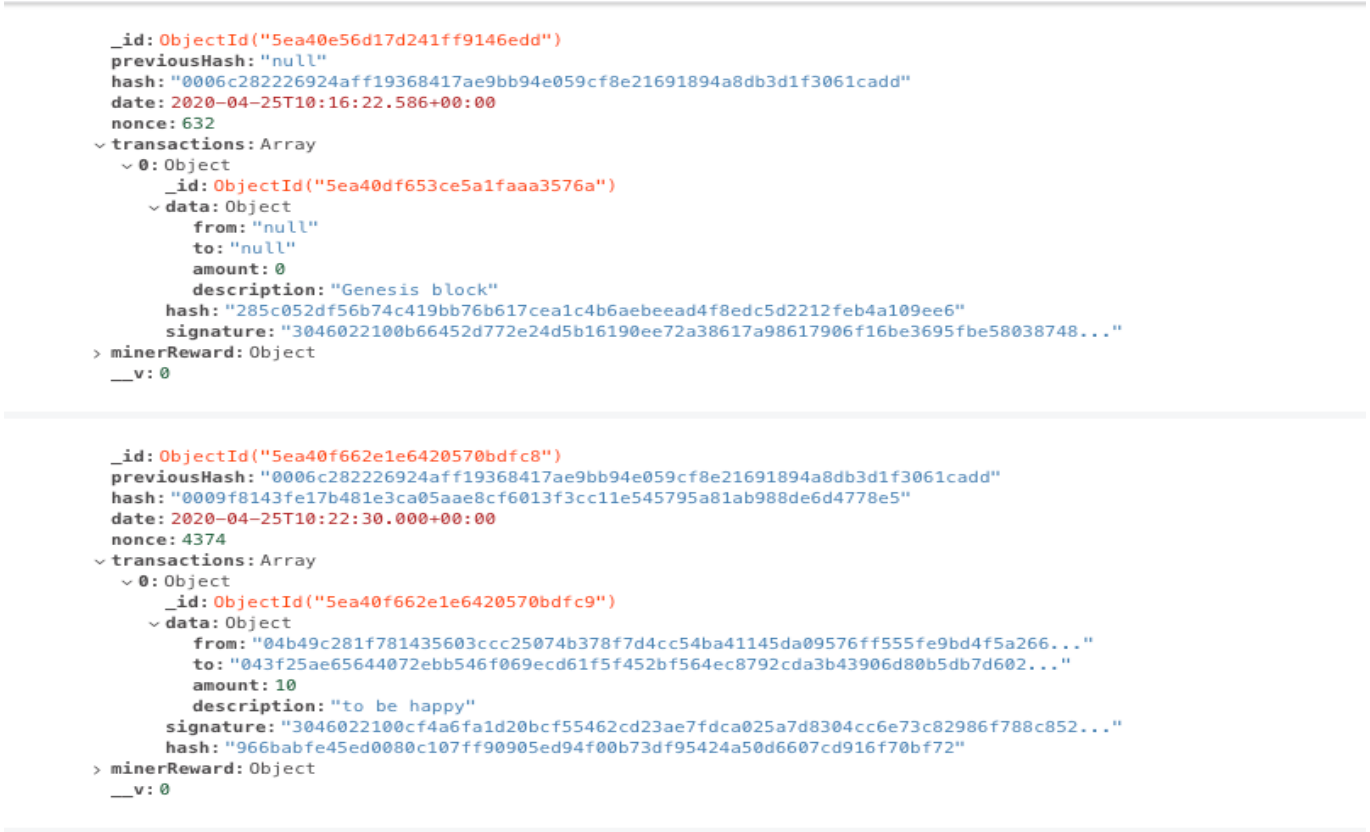


Рисунок 4.4 — Колекція db.blockchain

Структура db.nodes:

- _id: унікальне-хеш значення яке ідентифікує запис;
- email: текстове значення, адреса електронної пошти користувача;
- firstName: текстове значення, ім’я користувача;
- lastName: текстове значення, прізвище користувача;
- phoneNumber: текстове значення, номер телефону користувача;
- registrationDate: текстове значення, дата реєстрації користувача;
- passwordHash: текстове значення, пароль користувача у вигляді хеш-значення;
- publicKey: текстове значення, публічний ключ користувача, іншими словами — адреса гаманця користувача
- privateKey: текстове значення, приватний ключ користувача за допомогою якого здійснюється підпис транзакцій.

```
_id: ObjectId("5e0384063ab96e20368fac46")
email: "user1@gmail.com"
firstName: "Vitalii"
lastName: "Shapoval"
phoneNumber: "(380) 999-57-50-21"
registrationDate: 2019-12-25T15:45:10.215+00:00
passwordHash: "$2a$10$xa3duPWd7keIhu7rF50g2uXvd.JzCTWOS1bA6Hp7727sdHNkDhccu"
publicKey: "04b49c281f781435603ccc25074b378f7d4cc54ba41145da09576ff555fe9bd4f5a266..."
privateKey: "1816a48f2401909291b1a4e18d2b61362e4b1e8889197feb395922f6c3921cb9"
__v: 0
```

```
_id: ObjectId("5e0384833ab96e20368fac47")
email: "bootstrapper@gmail.com"
firstName: "Vitalii"
lastName: "Shapoval"
phoneNumber: "(380) 999-57-50-21"
registrationDate: 2019-12-25T15:47:15.524+00:00
passwordHash: "$2a$10$xa3duPWd7keIhu7rF50g2uXvd.JzCTWOS1bA6Hp7727sdHNkDhccu"
publicKey: "0465a5501942190f25b827b23c1db54bcbaf6e41d17c5f420e261eff8cd7a386a822e4..."
privateKey: "75f66bb935ac083d845a53ec0a82d503f8593e0b38706fb31e750df9d2e2d546"
__v: 0
```

```
_id: ObjectId("5e0384ca3ab96e20368fac4a")
email: "miner@gmail.com"
firstName: "Vitalii"
lastName: "Shapoval"
phoneNumber: "(380) 999-57-50-21"
registrationDate: 2019-12-25T15:48:26.757+00:00
passwordHash: "$2a$10$xa3duPWd7keIhu7rF50g2uXvd.JzCTWOS1bA6Hp7727sdHNkDhccu"
publicKey: "04b2bf466c3d05a8f5869e61fd88c9af5824e7aa267f2c3dd101ff32888bcbb3e9d3fc..."
privateKey: "aac697d9d8a4eb6a9ed26ae1e537621d93820ad50e7c891923d7568829dbfe59"
__v: 0
```

```
_id: ObjectId("5e0897d36b6143b60225a5d7")
email: "user2@gmail.com"
firstName: "Eugene"
lastName: "Lokotairiev"
phoneNumber: "(380) 964-98-17-56"
registrationDate: 2019-12-29T12:10:59.472+00:00
passwordHash: "$2a$10$xa3duPWd7keIhu7rF50g2uXvd.JzCTWOS1bA6Hp7727sdHNkDhccu"
publicKey: "043f25ae65644072ebb546f069ecd61f5f452bf564ec8792cda3b43906d80b5db7d602..."
privateKey: "db2901b9d44691fefaf1acc69b2e96e3ce2ae948baf4adcfe0078d92864c2ec5"
__v: 0
```


Рисунок 4.5 — Колекція db.nodes

Структура db.nodes:

- _id: унікальне-хеш значення яке ідентифікує запис;
- walletAdress: текстове значення, адреса електронної пошти користувача;
- __v: числове значення, поточний грошовий баланс користувача;
- ip: текстове значення, IP-адреса, за якою працює сервер в мережі;
- port: текстове значення, номер порту, на якому працює сервер;
- role: текстове значення, описує які задачі доступні для користувача;

Рисунок 4.6 — Колекція db.users

4.2 Механізм авторизації та аутентифікації користувачів за допомогою JSON Web tokens

Генерація токена:

Відправлення POST запитів /signup, POST /login. При подальших запитах якщо сервер не знайде токену, то видасть у відповіді помилку 401 з описом проблеми.

Процес перевірка токена:

Здійснюється перевірка заголовку Autorization на наявність тексту. Якщо він відсутній – відправляється відповідне повідомлення. В іншому випадку проводиться валідація токена і пошук користувача; якщо процес успішний – HTTP запит виконується, інакше забороняється.

4.3 Алгоритм валідації нового блоку

Для визнання блоку валідним, необхідно виконання наступних умов:

- Попереднє хеш-значення в новому блоці повинно бути таким же самим як і хеш-значення попереднього блоку;
- Хеш-значення що вказане в новому блоці повинно бути так же саме, якщо його перерахувати хеш-значенням враховуючи поля:

```
- previous hash (хеш-значення попереднього блоку)
- transactions (массив транзакцій блоку)
- nonce (число, яке було отримано під час майнингу блоку)
- date (дата створення блоку)
```

При умові виконання цих двох умов новий блок вважається валідним, його ніхто не підробив і його можна зберігати в блокчейн.

4.4 Опис завантажуючого (bootstrapper) сервера

4.4.1 Процес запуску завантажуючого сервера

Спочатку сервер встановлює зв’язок з БД:

```
const connection = await mongoose.connect(process.env.MONGODB_URL,
 {
   useNewUrlParser: true,
   useFindAndModify: false,
   useUnifiedTopology: true,
 });
```

Якщо підключення не буде успішним – подальша робота зупиняється.

Далі йде процес налаштування express server. Здійснюється процес налаштування роботи з HTTP, CORS.

Після цього здійснюється запис в пул користувачів інформації про себе:

```
- ip,
- port
- role (user | boostrapper | miner)
- email
- wallet address (публічний ключ)
```

На останньому етапі налаштовується завантажувач блокчейну (blockchain-loader). Здійснюється перевірка первинного блоку (genesis block) в блокчейні. Якщо його немає (це означає, що блокчейн пустий) – створюється первинний блок.

4.4.2 Функціонал завантажуючого серверу

Після завантаження, сервер може приймати наступні HTTP запити:

GET /users – отримання всіх вузлів. Не приймає вхідних даних. Результатом запиту буде список всіх користувачів з полями:

```
- ip,
- port
- role
- email
- wallet address (публічний ключ)
```

POST /user – додавання нового вузла. Даний запит очікує на вході наступні поля:

```
- ip,
- port,
- wallet address,
- role,
- email
```

Унікальність користувачів визначається за адресою електронної пошти. Якщо користувач з такою поштою існує, система видає відповіддю відповідне повідомлення.

GET /blockchain. Не приймає вхідних даних. Результатом запиту буде отримання всього блокчейну. Дана кінцева точка (endpoint) необхідний у випадку, коли новий користувач долучається до мережі і необхідно провести перше завантаження блокчейну.

4.5 Опис сервера майнера

4.5.1 Процес запуску сервера майнера

Сервер встановлює зв’язок з БД.

Якщо підключення не буде успішним – подальша робота зупиняється.

Налаштування express server. Здійснюється процес налаштування роботи з HTTP, CORS.

Здійснюється запис в пул користувачів інформації про себе:

Відправлення запиту на отримання всіх вузлів мережі до завантажуючого сервера

Після отримання списку вузлів, сервер зберігає в себе локальну копію в колекцію nodes. При подальшому оновленні інформації про користувачів, сервер при новому запуску здійснить новий аналогічний запит.

Оновлення блокчейну. Здійснюється запит до завантажуючого серверу на отримання блокчейну, валідації і подальшого збереження в локальній базі даних у випадку якщо валідація успішна.

4.5.2 Функціонал сервера майнера

POST /transaction На вході приймає інформацію про нову транзакцію. Додає транзакцію в чергу транзакцій, які очікують майнингу. Після завершення майнингу черга очищається.

POST /new-block. Приймає на вході інформацію відносно нового блоку. Якщо блок валідний, то сервер додає його в колекцію blockchain.

Окремо слід зазначити, що на даному сервері наявний механізм здійснення майнингу у вигляді утилітного методу:

```
minePendingTransactions(
  DIFFICULTY - складність майнингу
  transactions - черга транзакцій
  lastBlockHash - хеш попереднього блоку
  date - дата майнингу блоку
)
```

В основі майнингу закладена концепція доведення виконаної роботи (proof of work).

4.6 Опис сервера користувача

4.6.1 Процес запуску сервера користувача

Сервер встановлює зв’язок з БД

Якщо підключення не буде успішним – подальша робота зупиняється.

Налаштування express server. Здійснюється процес налаштування роботи з HTTP, CORS.

Здійснюється запис в пул користувачів інформації про себе:

Відправлення запиту на отримання всіх вузлів мережі до завантажуючого сервера

Після отримання списку вузлів, сервер зберігає в себе локальну копію в колекцію nodes. При подальшому оновленні інформації про користувачів, сервер при новому запуску здійснить новий аналогічний запит.

Оновлення блокчейну. Здійснюється запит до завантажуючого серверу на отримання блокчейну, валідації і подальшого збереження в локальній базі даних у випадку якщо валідація успішна.

4.6.2 Функціонал серверу користувача

Після завантаження, сервер може приймати наступні запити:

POST /signup. На вході приймаються поля як email, password, first name, last name, phone number для реєстрації нового користувача. У випадку успішної реєстрації користувача генерується JSON Web Token.

POST /login. На вході приймаються як email, password здійснення процесу аутентифікації та авторизації. У випадку успішної реєстрації створюється персональний токен який перенаправляє користувача до власного кабінету та надає можливість виконувати HTTP запити.

GET /transactions. На вході – інформація, за якою можна ідентифікувати користувача.

З початку виконується запит до колекції blockchain, знаходяться всі транзакції, у яких поля from, to співпадають з публічним ключем користувача;

По знайденим транзакціям виконується запит для отримання інформації по користувачам за електронною адресою та публічним ключем;

Знайдені транзакції відправляються користувачу у наступній формі:

```
{
  data: {
    from - публічний ключ відправника
    to - публічний ключ отримувача
    amount - кількість монет
    description - описання транзакції (опціонально)
  }
  from: {
    email: email відправника
  },
  to: {
    email: email отримувача
  }
}
```

POST /new-block. На вході – інформація про новий блок. Після валідації блока, він вноситься в локальну БД користувача.

POST /transaction/create – сервер отримує дані від користувача на створення транзакції:

```
{
  to – кому
  amount – кількість відправленої криптовалюти
  description – опис транзакції
}
```

Додатково із локальної БД здійснюється процес отримання публічного та приватного ключа.

Далі розраховується хеш транзакції. На основі хешу транзакції та приватного ключа здійснюється цифровий підпис транзакції і транзакція поширюється всім майнерам у вигляді:

```
{
  data: {
    from,
    to,
    amount,
    description,
  },
  signature,
  hash,
}
```

5 Робота користувача з програмною системою

5.1 Системні вимоги для додаткове програмне забезпечення

Для роботи необхідно апаратне забезпечення, що задовольняє вимогам: процессор Intel Core i3, 8 Гб оперативної пам'яті, 256 ГБ вільного місця на жорсткому диску.

Додатково необхідно встановити програмне забезпечення: MongoDB, NPM, Node.js збережння блокчейну на локальній машині та для запуску серверів.

5.2 Результати виконання програми

Після того як користувач має все необхідне програмне забезпечення, йому необхідно перейти на голову сторінку и одну з дій: вхід в систему, або реєстрація нового користувача. Сторінка входу систему показано на рисунку 5.1, де необхідно ввести електронну пошту и пароль.

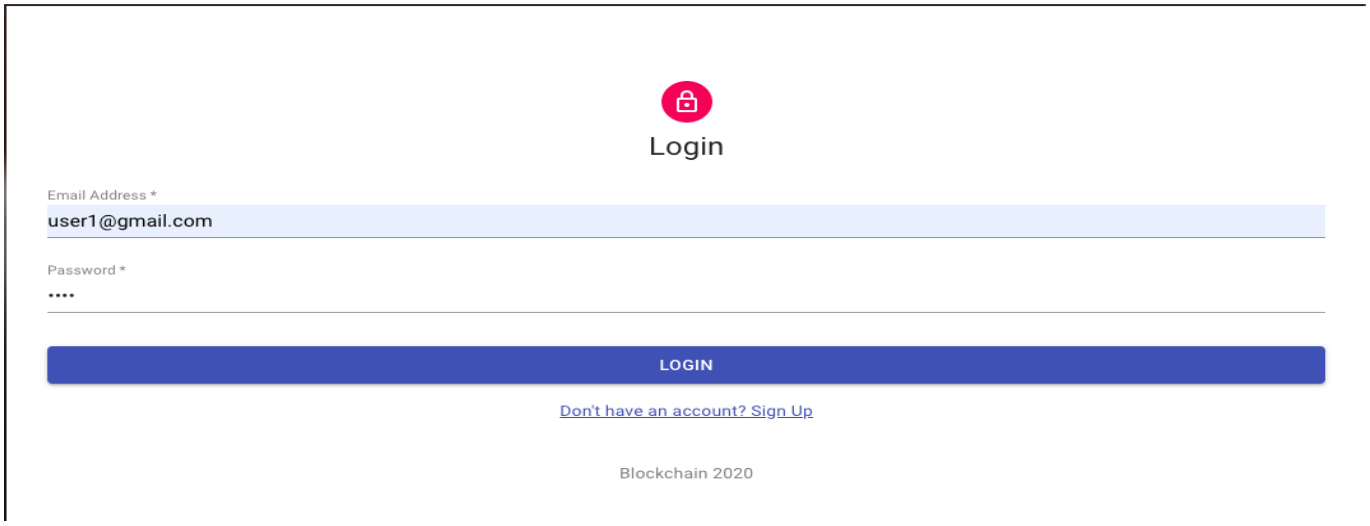


Рисунок 5.1 — Сторінка аутентифікації та авторизації

Для реєстрації нового користувача необхідно перейти на відповідну сторінку, ввести електронну адресу, прізвища, ім’я, номер телефону та пароль (рисунок 5.2). Після чого натиснути кнопку «Sign in».

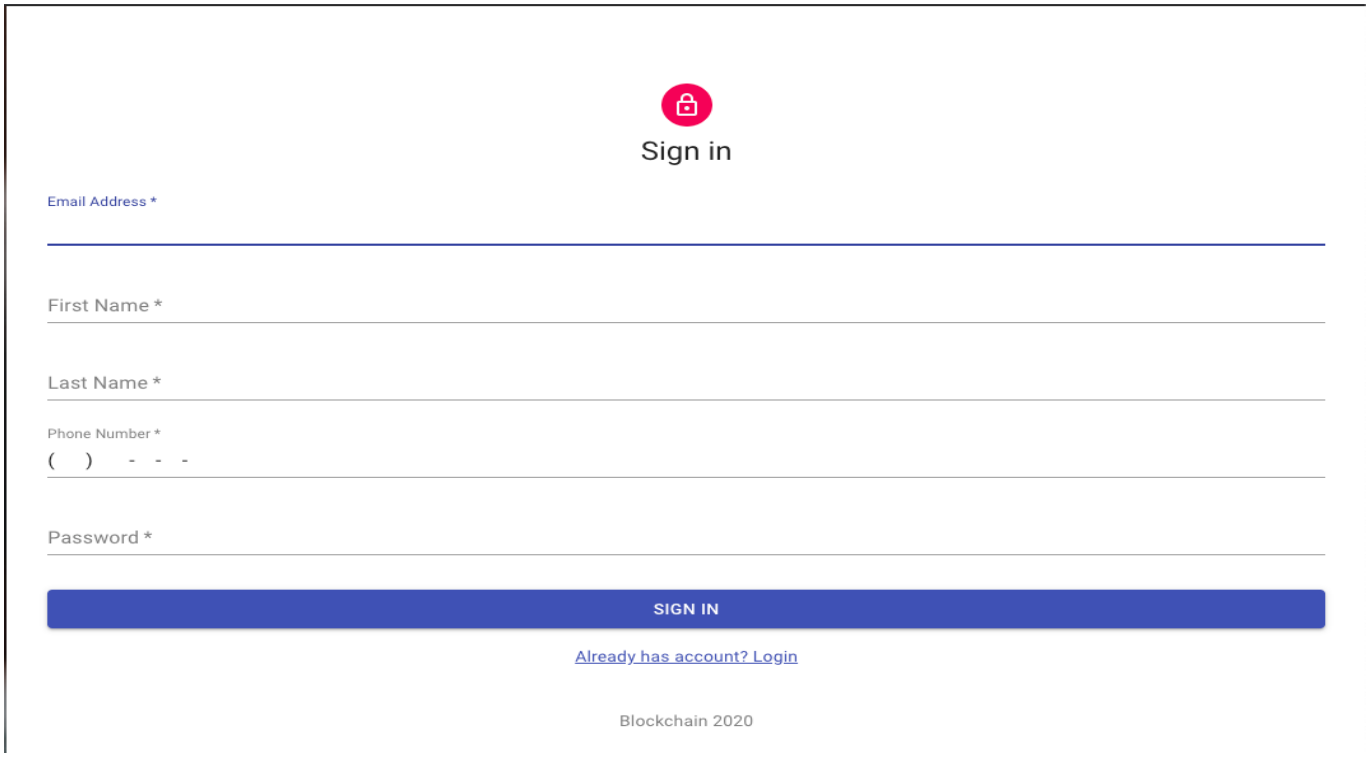


Рисунок 5.2 — Сторінка реєстрації

Після того, як користувач увійшов в систему, він може створити транзакцію на переведення криптовалюти на спеціальній для цього сторінці (рисунок 5.3), де користувачу необхідно ввести адресу електронного гаманця отримувача, кількість валюти, яку необхідно передати та опис транзакції, в якому можна вказати для чого передаються гроші.

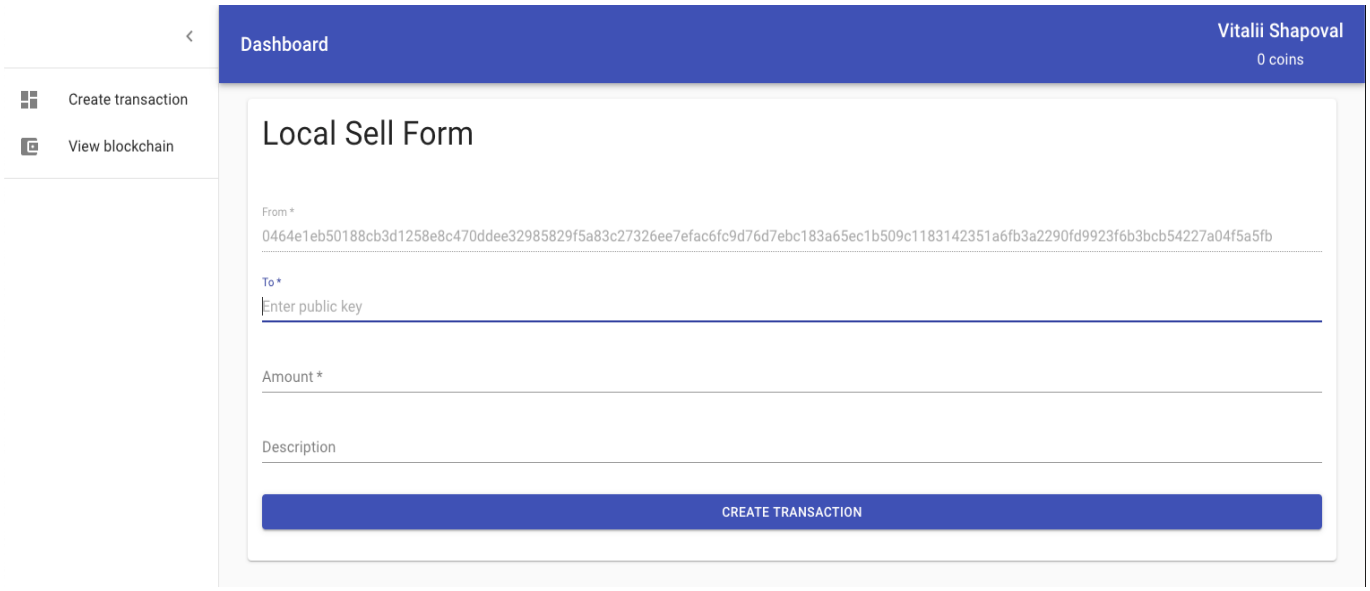


Рисунок 5.3 — Сторінка створення транзакції

Після того, як користувач створив транзакцію, вона була внесена в блок і був успішно проведений майнинг, криптовалюта буде переведена, а користувач зможе побачити виконану транзакцію (рисунок 5.4).

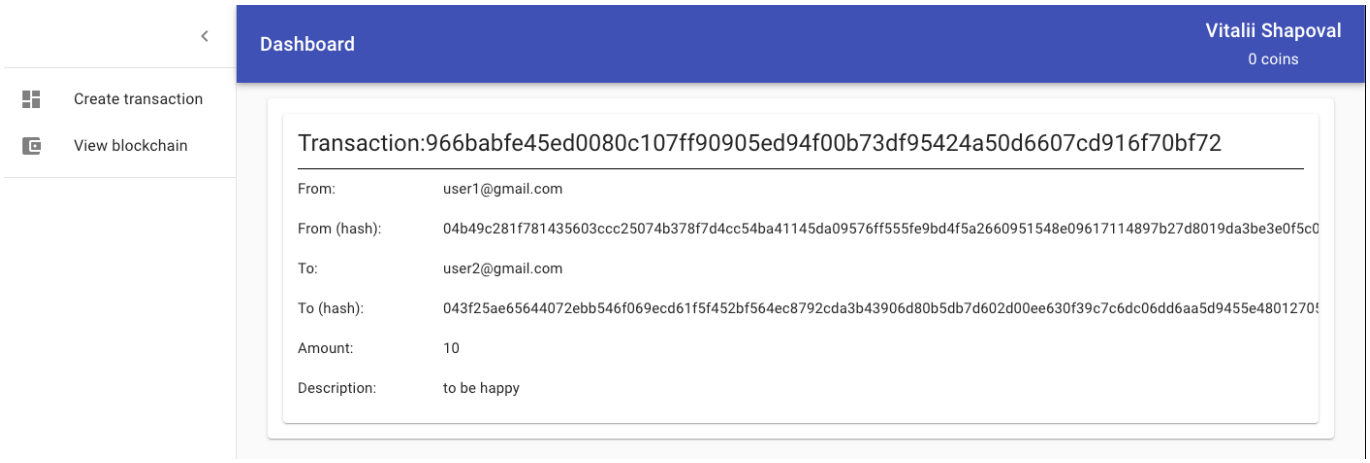


Рисунок 5.4 — Сторінка перегляду транзакцій

ВИСНОВКИ

У ході виконання роботи був проведений аналіз предметної області, аналіз вимог та проектування архітектури системи і як наслідок було створено програмний продукт, який дозволяє організувати однорангову взаємодію між користувачами за допомогою технології блокчейн.

Основна задача системи — проведення фінансових операцій над енергоресурсами за новою концепцією, де кожен із користувачів в мережі має безпосередній зв’язок один із одним і здатний комунікувати без посередників, що потенціально може змінити ринок енергоресурсів та в загалом зробить ресурси дешевшими.

Даний підхід вимагає спеціальних алгоритмів, таких як хешування та побудова даних у вигляді ланцюгів, де забезпечується висока стійкість до підробки.

Проектування системи було виконано таким чином, щоб було можливо забезпечити масштабування та простоту у внесенні змін або додаванні нового функціоналу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сегеда І.В., Локотарев Є.О., Шаповал В.О. Реалізація використання блокчейн-технологій у енергетичному секторі. / Вчені записки Таврійського національного університету імені В.І. Вернадського Серія:Економіка і управління Том 30 (69). № 4, 2019, С. 160-165 (DOI: https://doi.org/10.32838/2523-4803/69-4-51) .
2. Segeda I. Blockchain as a digital economy promotion tool in energy industry. Modern Aspects of Software Development: Proceedings of VI International Scientific and Practical Virtual Conference of Software Development Specialists, June, 24 2019 p. – Kyiv: Igor Sikorsky KPI, 2019. – p. 139-146 .
3. Nakamoto S. A Peer-to-Peer Electronic Cash System [Електронний ресурс] // Bitcoin. – Режим доступ до ресурсу: https://bitcoin.org/bitcoin.pdf .
4. How Blockchain Technology Works. Guide for Beginners [Електронний ресурс] / – Режим доступу до ресурсу: https://cointelegraph.com/bitcoin-for-beginners/how-blockchain-technology-works-guide-for-beginners#distributed-database
5. Блокчейн: как он работает, и почему эта технология изменит мир [Електронний ресурс] /– Режим доступу до ресурсу: https://habr.com/ru/company/iticapital/blog/340992/
6. Технічна документація мови JavaScript [Електронний ресурс] / – Режим доступу до ресурсу: https://learn.javascript.ru/
7. Технічна документація React.js [Електронний ресурс] / – Режим доступу до ресурсу: https://devdocs.io/react/
8. Технічна документація npm [Електронний ресурс] / – Режим доступу до ресурсу: https://docs.npmjs.com/
9. Технічна документація Node.js [Електронний ресурс] / – Режим доступу до ресурсу: https://nodejs.org/dist/latest-v12.x/docs/api/
10. Технічна документація MongoDB [Електронний ресурс] / – Режим доступу до ресурсу: https://docs.mongodb.com/manual/

11. Gackenheimer, C. (2015). What is react?. In Introduction to React. Apress, Berkeley, CA. - p. 1-20
12. Fedosejev A. React. js essentials. Packt Publishing Ltd; 2015 Aug 27.
13. Vainikka, J., 2018. Full-stack web development using Django REST framework and React.
14. Stefanov S. React: up & running: building web applications. " O'Reilly Media, Inc."; 2016 Jul 14.
15. Domes S. Progressive Web Apps with React: Create lightning fast web apps with native power using React and Firebase. Packt Publishing Ltd; 2017 Oct 24.
16. Satheesh M, D'mello BJ, Krol J. Web development with MongoDB and NodeJs. Packt Publishing Ltd; 2015 Oct 30.
17. Laksono D. Testing spatial data deliverance in SQL and NoSQL database using NodeJS fullstack web app. In2018 4th International Conference on Science and Technology (ICST) 2018 Aug 7. p. 11-28. IEEE.
18. Hahn E. Express in Action: Writing, building, and testing Node. js applications. Manning Publications,; 2016.
19. Mardan A. Pro express. js. Apress; 2014. - p. 40-56.Цикл событий Node.js, таймеры и process.nextTick() [Електронний ресурс] / - Режим доступу до ресурсу: https://medium.com/devschacht/event-loop-timers-and-nexttick-18579cd122e0
20. Введение в JSON [Електронний ресурс] / - Режим доступу до ресурсу: https://www.json.org/json-ru.html
21. What Is MongoDB’s _id Field and How to Use It [Електронний ресурс] / - Режим доступу до ресурсу: https://orangematter.solarwinds.com/2019/12/22/what-is-mongodbs-id-field-and-how-to-use-it/
22. This is How you Bootstrap a Blockchain Relationship Ecosystem (Part 1) [Електронний ресурс] / - Режим доступу до ресурсу: https://hackernoon.com/this-is-how-you-bootstrap-a-blockchain-relationship-ecosystem-part-1-b7b7c8d29eae

23. Blockchain using NodeJS and socket.io [Електронний ресурс] / - Режим доступу до ресурсу: https://dev.to/sadarshannaiynar/blockchain-using-nodejs-and-socketio-5gbe

24. How To Build A Simple Cryptocurrency Blockchain In Node.js [Електронний ресурс] / - Режим доступу до ресурсу: https://www.smashingmagazine.com/2020/02/cryptocurrency-blockchain-node-js/

25. JSON web token [Електронний ресурс] / - Режим доступу до ресурсу: https://jwt.io/

26. Chodorow C. Introduction to mongodb. InFree and Open Source Software Developers European Meeting (FOSDEM) 2010. - p. 110-117.

27. Li, Chaokui, and Wu Yang. "The distributed storage strategy research of remote sensing image based on Mongo DB." 2014 Third International Workshop on Earth Observation and Remote Sensing Applications (EORSA). IEEE, 2014.

28. Kvalheim C. The Little Mongo DB Schema Design Book. The Blue Print SeriesMembrey, P., Hows. D., Plugge, E.(2014). MongoDB Basics. Apress. 2015.

29. Boicea A, Radulescu F, Agapin LI. MongoDB vs Oracle--database comparison. In2012 third international conference on emerging intelligent data and web technologies 2012 Sep 19. - p. 330-335.

30. Werner MJ, Vogt C. Implementation and Evaluation of a DHT-based content distribution system using WebRTC.

31. Goldfeder S, Kalodner H, Reisman D, Narayanan A. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. Proceedings on Privacy Enhancing Technologies. 2018 Oct 1. - p 179-199.

32. Bogner A, Chanson M, Meeuw A. A decentralised sharing app running a smart contract on the ethereum blockchain. InProceedings of the 6th International Conference on the Internet of Things 2016 Nov 7. - p. 160-178.

33. Crosby M, Pattanayak P, Verma S, Kalyanaraman V. Blockchain technology: Beyond bitcoin. Applied Innovation. 2016 Jun.

34. ones M, Campbell B, Mortimore C. JSON Web Token (JWT) profile for OAuth 2.0 client authentication and authorization Grants. 2015 May.
35. Brier E, Coron JS, Icart T, Madore D, Randriam H, Tibouchi M. Efficient indifferentiable hashing into ordinary elliptic curves. InAnnual Cryptology Conference 2010 Aug 15. - p. 237-254
36. Masse M. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. " O'Reilly Media, Inc."; 2011 Oct 18.
37. Paterson KG, Srinivasan S. Building key-private public-key encryption schemes. InAustralasian Conference on Information Security and Privacy 2009 Jul 1. - p. 276-292
38. Hash values [Електронний ресурс] / - Режим доступу до ресурсу: https://www.trendmicro.com/vinfo/us/security/definition/hash-values
39. Nonce [Електронний ресурс] / - Режим доступу до ресурсу: https://en.bitcoin.it/wiki/Nonce
40. Что такое CORS [Електронний ресурс] / - Режим доступу до ресурсу: https://webdevblog.ru/chto-takoe-cors/
41. A Deep Dive on End-to-End Encryption: How Do Public Key Encryption Systems Work? [Електронний ресурс] / - Режим доступу до ресурсу: https://ssd.eff.org/en/module/deep-dive-end-end-encryption-how-do-public-key-encryption-systems-work
42. What are Hash Functions and How to choose a good Hash Function? [Електронний ресурс] / - Режим доступу до ресурсу: https://www.geeksforgeeks.org/what-are-hash-functions-and-how-to-choose-a-good-hash-function/
43. SHA-256 – живая классика среди алгоритмов хеширования [Електронний ресурс] / - Режим доступу до ресурсу: https://tutdenegki.com/crypta/sha-256.html
44. Distributed Systems [Електронний ресурс] / - Режим доступу до ресурсу: https://www.tutorialspoint.com/Distributed-Systems

ДОДАТОК 1

Реалізація програмного рішення інформаційної системи обліку енергоресурсів з використанням технології блокчейн

СПЕЦИФІКАЦІЯ

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б

Аркушів 2

Київ 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б	Записка.docx	Текстова частина дипломної роботи
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б	Діаграми.vsdx	UML діграми
Компоненти		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б	user	Сервер користувача
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б	ui	Сервер графічного інтерфейсу користувача
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б	bootstrapper	Сервер-бутстрапер
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б	miner	Сервер майнера

ДОДАТОК 2

Реалізація програмного рішення інформаційної системи обліку енергоресурсів з використанням технології блокчейн

ТЕКСТ ПРОГРАМНОГО МОДУЛЮ

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б

Аркушів 15

Київ 2020

Тест програми

src/app.js

```
const express = require('express');
const dotenv = require('dotenv');

const loaders = require('./loaders');
const mountRoutes = require('./routes');

const user1 = {
  publicKey:
'04b49c281f781435603ccc25074b378f7d4cc54ba41145da09576ff555fe9bd4f5a2660951548e09617114897b27d8019da3be3e0
f5c0d229bf5d9dbd71fd80882',
  email: 'user1@gmail.com',
  ip: '0.0.0.0',
  port: 5052,
  role: 'user',
};

const user2 = {
  publicKey:
'043f25ae65644072ebb546f069ecd61f5f452bf564ec8792cda3b43906d80b5db7d602d00ee630f39c7c6dc06dd6aa5d9455e4801
2705f5e8d3013e8afdbc39100',
  email: 'user2@gmail.com',
  ip: '0.0.0.0',
  port: 5052,
  role: 'user',
};

async function startServer() {
  const app = express();

  dotenv.config();
  await loaders({
    app,
    ...user2,
  });
  mountRoutes(app);

  app.listen(5052, () => {
    console.log(`Running on ${5052}`);
  });
}

startServer();
```

src/constants.js

```
const modes = {
  USER: 'user',
  BOOTSTRAPPER: 'bootstrapper',
  MINER: 'miner',
};

module.exports = {
  modes,
};
```

src/package.json

```
{
  "name": "p2p",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon src/app.js",
    "lint": "eslint --debug src/**/*.{js,jsx}",
    "lint:write": "eslint --debug src/**/*.{js,jsx} --fix --quiet",
    "prettier": "prettier --write src/**/*.{js,jsx}",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "axios": "^0.19.0",
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "crypto-js": "^3.1.9-1",
    "dotenv": "^8.1.0",
    "elliptic": "^6.5.0",
    "express": "^4.17.1",
    "express-promise-router": "^3.0.3",
    "get-port": "^5.0.0",
    "husky": "^3.0.1",
    "ip": "^1.1.5",
    "jsonwebtoken": "^8.5.1",
    "lint-staged": "^9.2.1",
    "lodash": "^4.17.15",
    "mongodb": "^3.3.2",
    "mongoose": "^5.6.13",
    "request-promise": "^4.2.4"
  },
  "devDependencies": {
    "eslint": "^5.16.0",
    "eslint-config-airbnb-base": "^13.2.0",
    "eslint-plugin-import": "^2.18.2",
    "nodemon": "^1.19.1",
    "prettier": "^1.18.2",
    "prettier-eslint": "^9.0.0"
  },
  "husky": {
    "hooks": {
      "pre-commit": "lint-staged",
      "pre-push": "npm test"
    }
  },
  "lint-staged": {
    "*.{js,jsx}": [
      "eslint --fix --quiet",
      "git add"
    ]
  }
}
```

src/api/index.js

```
const axios = require('axios');
```

```
const fetchUsers = async () => {
  try {
    const response = await axios.get('http://0.0.0.0:5050/users');
    // console.log('users', response.data);

    return response.data.users;
  } catch (error) {
    console.log('\n fetch-users \n');
    console.log(error.response.data.error);
  }
};

const registerUser = async ({
  ip,
  port,
  walletAddress,
  role,
  email,
}) => {
  try {
    await axios.post(
      'http://0.0.0.0:5050/user',
      {
        walletAddress,
        ip,
        port,
        role,
        email,
      },
    );
    console.log('user has been added to bootstrapper');
  } catch (error) {
    console.log('\n register-user \n');
    console.log(error.response.data.error);
  }
};

const fetchBlockchain = async () => {
  try {
    const response = await axios.get('http://0.0.0.0:5050/blockchain');
    // console.log('blockchain', response.data.blockchain);
    return response.data.blockchain;
  } catch (error) {
    console.log('\n fetch-blockchain \n');
    console.log(error.response.data.error);
  }
};

module.exports = {
  fetchUsers,
  registerUser,
  fetchBlockchain,
};
```

src/loaders/bootstrap.js

```
const {
  fetchUsers,
  fetchBlockchain,
  registerUser,
} = require('../api');
```

```
const {
  Node,
  Block,
} = require('../services');

module.exports = async ({
  publicKey,
  ip,
  port,
  role,
  email,
}) => {
  await registerUser({
    ip,
    port,
    walletAddress: publicKey,
    role,
    email,
  });

  const nodes = await fetchUsers();
  nodes.forEach(async (node) => {
    if (node.walletAddress !== publicKey) {
      await Node.addOrUpdateNode(node);
    }
  });

  const blocks = await fetchBlockchain();
  await Block.updateChain(blocks);
};
```

src/loaders/express.js

```
const bodyParser = require('body-parser');
const cors = require('cors');

module.exports = async ({ app }) => {
  app.use(
    bodyParser.urlencoded({
      extended: true,
    }),
  );
  app.use(bodyParser.json());
  app.use(cors());

  return app;
};
```

src/loaders/index.js

```
const expressLoader = require('./express');
const mongooseLoader = require('./mongoose');
const bootstrapLoader = require('./bootstrap');

module.exports = async ({
  app,
  publicKey,
  ip,
  port,
  role,
  email,
}) => {
```

```
  await mongooseLoader();
  console.log('MongoDB Initialized');

  await expressLoader({ app });
  console.log('Express Initialized');

  await bootstrapLoader({
    publicKey,
    ip,
    port,
    role,
    email,
  });
  console.log('Bootstrap Initialized');
};
```

src/loaders/mongoose.js

```
const mongoose = require('mongoose');

module.exports = async () => {
  const connection = await mongoose.connect(process.env.MONGODB_URL,
    {
      useNewUrlParser: true,
      useFindAndModify: false,
      useUnifiedTopology: true,
    });
  return connection.connection.db;
};
```

src/middleware/autenticate.js

```
const jwt = require('jsonwebtoken');
const { User } = require('../services/index');

const authenticate = (req, res, next) => {
  const authorization = req.header('Authorization');

  if (!authorization) {
    return res.status(401).json({ error: 'No token provided' });
  }

  jwt.verify(authorization, process.env.JWT_SECRET, async (err, { email }) => {
    if (err) {
      return res.status(401).json({ error: 'Invalid token' });
    }

    try {
      const user = await User.find({ email });

      if (user.error) {
        return res.status(401).json({ error: 'No user with provided credentials' });
      }

      req.currentUser = user;
      next();
    } catch (error) {
      res.status(401).json({ error: 'Something when wrong' });
    }
  });
};
```

```
module.exports = {
  authenticate,
};
```

src/models/Block.js

```
const mongoose = require('mongoose');

const { Schema } = mongoose;
const { TransactionSchema } = require('./Transaction');

const BlockSchema = new Schema({
  date: {
    type: Date,
    required: true,
  },
  previousHash: {
    type: String,
    required: true,
  },
  hash: {
    type: String,
    required: true,
  },
  nonce: {
    type: Number,
    required: true,
  },
  transactions: [TransactionSchema],
  minerReward: {
    wallet: {
      type: String,
      required: true,
    },
    amount: {
      type: Number,
      required: true,
    },
  },
});

module.exports = mongoose.model('Blockchain', BlockSchema);
```

src/models/Node.js

```
const mongoose = require('mongoose');

const { Schema } = mongoose;

const NodeSchema = new Schema({
  ip: {
    type: String,
    required: true,
    lowercase: true,
    trim: true,
  },
  port: {
    type: Number,
    required: true,
  },
  walletAddress: {
    type: String,
```

```
    required: true,
    lowercase: true,
    unique: true,
  },
  role: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
  },
});

module.exports = mongoose.model('Node', NodeSchema);
```

src/models/Transaction.js

```
const mongoose = require('mongoose');

const { Schema } = mongoose;

const TransactionSchema = new Schema({
  hash: {
    type: String,
    required: true,
  },
  signature: {
    type: String,
    required: true,
  },
  data: {
    from: {
      type: String,
      required: true,
    },
    to: {
      type: String,
      required: true,
    },
    amount: {
      type: Number,
      required: true,
    },
    description: {
      type: String,
    },
  },
});

module.exports = {
  TransactionSchema,
};
```

src/models/User.js

```
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const { generateKeyPair } = require('../utils/keygenerator');
```

```
const { Schema } = mongoose;

const UserSchema = new Schema({
  email: {
    type: String,
    required: true,
    lowercase: true,
    trim: true,
    unique: true,
  },
  passwordHash: {
    type: String,
    require: true,
  },
  phoneNumber: {
    type: String,
    required: true,
  },
  firstName: {
    type: String,
    required: true,
  },
  lastName: {
    type: String,
    required: true,
  },
  registrationDate: {
    type: Date,
    required: true,
    default: Date.now,
  },
  publicKey: {
    type: String,
    required: true,
  },
  // TODO: salt privateKey?
  privateKey: {
    type: String,
    required: true,
  },
});

UserSchema.methods.isValidPassword = function isValidPassword(password) {
  return bcrypt.compareSync(password, this.passwordHash);
};

UserSchema.methods.setPassword = function setPassword(password) {
  this.passwordHash = bcrypt.hashSync(password, process.env.JWT_SECRET);
};

UserSchema.methods.generateJWT = function generateJWT() {
  return jwt.sign(
    {
      registrationDate: this.registrationDate,
      email: this.email,
    },
    process.env.JWT_SECRET,
    {
      expiresIn: '30d',
    },
  );
};
```

```
UserSchema.methods.generateKeyPair = function genereteKeyPair() {
  const { publicKey, privateKey } = generateKeyPair();

  this.publicKey = publicKey;
  this.privateKey = privateKey;
};

module.exports = mongoose.model('User', UserSchema);
```

src/routes/dashboard.js

```
const Router = require('express-promise-router');
const axios = require('axios');
const _ = require('lodash/fp');

const {
  authenticate,
} = require('../middleware/authenticate');
const {
  signTransaction,
  calculateHash,
} = require('../utils/transaction');
const {
  Block,
  User,
} = require('../services');

const router = new Router();

router.post('/transaction/create', authenticate, async (req, res) => {
  const {
    publicKey,
    privateKey,
  } = req.currentUser;
  const {
    to,
    amount,
    description,
  } = req.body;

  const hash = calculateHash(publicKey, to, amount, description);
  const signature = signTransaction(hash, privateKey);

  try {
    await axios.post(
      'http://0.0.0.0:5051/transaction/',
      {
        data: {
          from: publicKey,
          to,
          amount,
          description,
        },
        signature,
        hash,
      },
    );
  } catch (error) {
    console.log('error', error);
  }
```

```
  res.json({});
});

router.post('/new-block', async (req, res) => {
  await Block.updateChain([req.body]);
  res.json({ });
});

router.get('/transactions', authenticate, async (req, res) => {
  const getUserAdresses = _.pipe(
    _.map(tx => [tx.data.from, tx.data.to]),
    _.flatten,
    _.uniq,
  );

  const txs = await Block.findUserTransactions(req.currentUser.publicKey);
  const userAdresses = getUserAdresses(txs);

  const users = await User.findByPublicKey(userAdresses);

  const mappedTxsWithUsers = _.pipe(
    _.map(tx => ({
      data: tx.data,
      hash: tx.hash,
      from: users[tx.data.from],
      to: users[tx.data.to],
    })),
    _.reverse,
  );

  res.json({ txs: mappedTxsWithUsers(txs) });
});

module.exports = router;
```

src/routes/index.js

```
const keypair = require('./keypair');
const signup = require('./signup');
const login = require('./login');
const dashboard = require('./dashboard');

module.exports = (app) => {
  app.use('/keypair', keypair);
  app.use('/signup', signup);
  app.use('/login', login);
  app.use('/dashboard', dashboard);
};
```

src/routes/keypair.js

```
const Router = require('express-promise-router');

const { keyPairFormatter, streamToString } = require('../utils/io');

const router = new Router();

router.get('/', async (req, res) => {
  const keyPair = await streamToString('./keypair.txt', keyPairFormatter);

  if (!keyPair.publicKey || !keyPair.privateKey) {
    res.status(400).json({ error: 'No keypair' });
```

```
    return;
  }

  res.json({
    publicKey: keyPair.publicKey,
  });
});

module.exports = router;
```

src/routes/login.js

```
const Router = require('express-promise-router');

const { User } = require('../services');

const router = new Router();

router.post('/', async (req, res) => {
  // TODO: add validation here

  const {
    email,
    password,
  } = req.body;

  try {
    const user = await User.find({ email });

    if (user && !user.error && user.isValidPassword(password)) {
      res.json({
        token: user.generateJWT(),
      });
    } else {
      res.status(400).json({
        error: 'Invalid credentials provided',
      });
    }
  } catch (error) {
    console.log(error);
    res.status(400).json({ error: error.message });
  }
});

module.exports = router;
```

src/routes/signup.js

```
const Router = require('express-promise-router');

const { User } = require('../services');

const router = new Router();

router.post('/', async (req, res) => {
  // TODO: add validation here

  const {
    email,
    firstName,
    lastName,
    phoneNumber,
```

```
    password,
  } = req.body;

  try {
    const user = await User.create({
      email,
      firstName,
      lastName,
      phoneNumber,
      password,
    });

    res.json({
      reregistered: true,
      token: user.generateJWT(),
    });
  } catch (error) {
    console.log(error);
    res.status(400).json({ error: error.message });
  }
});

module.exports = router;
```

src/utils/io.js

```
const fs = require('fs');

const keyPairFormatter = (data) => {
  const [publicKey, privateKey] = data.split(':');

  return {
    publicKey,
    privateKey,
  };
};

const streamToString = (filePath, formatter = data => data) => {
  const stream = fs.createReadStream(filePath);
  const chunks = [];

  return new Promise((resolve, reject) => {
    stream.on('data', (chunk) => {
      chunks.push(chunk);
    });
    stream.on('error', reject);
    stream.on('end', () => {
      const data = formatter(Buffer.concat(chunks).toString());
      resolve(data);
    });
  });
};

const stringToStream = (filePath, data) => {
  const stream = fs.createWriteStream(filePath);

  return new Promise((resolve, reject) => {
    stream.write(data);
    stream.end();

    stream.on('error', reject);
    stream.on('finish', resolve);
  });
```

```
};

module.exports = {
  stringToStream,
  streamToString,
  keyPairFormatter,
};
```

src/utils/keygenerator.js

```
const EC = require('elliptic').ec;

const ec = new EC('secp256k1');

const generateKeyPair = () => {
  const key = ec.genKeyPair();
  const publicKey = key.getPublic('hex');
  const privateKey = key.getPrivate('hex');

  return {
    key,
    publicKey,
    privateKey,
  };
};

module.exports = {
  generateKeyPair,
};
```

src/utils/request.js

```
const request = require('request-promise');

// FIXME
// For each request need to change IP
const registerUser = (walletAddress, ip, port) => {
  const options = {
    method: 'POST',
    uri: 'http://0.0.0.0:5050/user/',
    body: {
      walletAddress,
      ip,
      port,
    },
    json: true,
  };

  return request.post(options);
};

const fetchUsers = () => {
  const options = {
    method: 'GET',
    uri: 'http://0.0.0.0:5050/users/',
  };

  return request.get(options).then(res => JSON.parse(res));
};

module.exports = {
  registerUser,
  fetchUsers,
```

```
};
```

src/utils/transaction.js

```
const SHA256 = require('crypto-js/sha256');
const EC = require('elliptic').ec;

const ec = new EC('secp256k1');

// TODO: make calculating suitable to all cases
const calculateHash = (...params) => {
  const concatenatedString = params.reduce((acc, current) => acc + current, '');

  return SHA256(concatenatedString).toString();
};

const signTransaction = (hash, privateKey) => {
  const myKey = ec.keyFromPrivate(privateKey);

  const sig = myKey.sign(hash, 'base64');
  const signature = sig.toDER('hex');

  return signature;
};

const verifyTransaction = (hash, publicKey, signature) => {
  const pKey = ec.keyFromPublic(publicKey, 'hex');

  return pKey.verify(hash, signature);
};

module.exports = {
  calculateHash,
  signTransaction,
  verifyTransaction,
};
```

ДОДАТОК З

Реалізація програмного рішення інформаційної системи обліку енергоресурсів з використанням технології блокчейн

ОПИС ПРОГРАМНОГО МОДУЛЮ

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б

Аркушів 9

Київ 2020

АНОТАЦІЯ

Додаток містить опис автоматизованої системи обліку та торгівлі енергоресурсами.

В додатку виконуються такі функції:

— введення вхідних даних для авторизації, даних про споживання енергоресурсів;

— обробка транзакцій та формування блоку;

— переведення криптовалюти**;**

— запис інформації в блокчейн**.**

Вхідні та вихідні дані отримуються засобами реалізованими React.js.

Обчислювальний сервер виконано з використанням технології Node.js. Сервер окремо підключається до бази MongoDB.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ 69
ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ 70
ОПИС ЛОГІЧНОЇ СТРУКТУРИ 71
ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ 72
ВИКЛИК І ЗАВАНТАЖЕННЯ 73
ВХІДНІ І ВИХІДНІ ДАНІ 74

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку описано систему з обліку та продажу енергоресурсів на основі технології блокчейну.

Модулі системи описані в ДОДАТКУ 2.

Додаток працює в операційних системах, таких як Windows7, Windows8, Windows10, Unix.

Компоненти необхідні для установки: NPM, Node.js, React.js.

Використана мова програмування— Javascript.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблена система призначена для обліку та продажу енергоресурсів за допомогою криптовалюти засобами технології блокчейн.

Дана система може бути використана особами, які бажають здійснювати облік, продаж або купівлю енергоресурсів.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Система являє собою децентралізовану однорангову структуру, де кожен користувач являється вузлом мережі. Сам вузол розроблений за принципом клієнт-серверного підходу. Доступ до бази даних здійснюється засобами Node.js.

Після запуску серверів необхідно в браузері перейти на адресу локального хосту із вказанням порту та зареєструватися як новий користувач, або увійти як вже існуючий. Після успішної авторизації, користувач має можливість здійснювати облік та продаж енергоресурсів.

ВИКОРИСТОВУВАНІ ТЕХІЧНІ ЗАСОБИ

Компоненти необхідні для установки системи: NPM, Node.js, React.js.

Використана мова програмування системи — Javascript.

Розроблена автоматизована система працює в операційних системах Windows7, Windows8, Windows10 та потребує встановлення компонентів: NPM, Node.js, MongoDB.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для запуску необхідно викликати shell команди і дочекатися запуску серверів. Після цього необхідно в браузері перейти за адресою локального хосту із вказанням порту на головну сторінку з подальшою реєстрацією нового або авторизацією вже існуючого користувача. Після успішної авторизації користувач може працювати із автоматизованою системою.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідні дані можуть бути текстового або цифрового значення в залежності від призначення форми. Вхідні дані вводяться та зчитуються засобами React.js, логіка обробки даних здійснюється за допомогою мови Javascript.

Аналогічно, вихідні дані представлені у вигляді текстового або цифрового значення у форматі JSON та збережені в базах даних користувачів.

Вихідні дані отримуються з бази даних і представляються користувачам у текстовому або цифровому вигляді.

ДОДАТОК 4

Реалізація використання блокчейн-технологій у енергетичному секторі

ТЕКСТ ТЕЗИ КОНФЕРЕНЦІЇ

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б

Аркушів 5

Київ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVIII Міжнародної науково-практичної конференції молодих вчених і студентів 2020 року

ТОМ 2

Київ- 2020

УДК 620.9(062)+621.311(062)
С91

Сучасні проблеми наукового забезпечення енергетики: Матеріали XVIII Міжнародної науково-практичної конференції молодих вчених і студентів 2020 року. У 2 т. – К. : 7 КПІ ім. Ігоря Сікорського, 2020. – Т. 2. – 160 с.

ISBN 978-966-622-997-0
ISBN 978-966-622-999-4 (Т.2)

Подано тези доповідей XVIII Міжнародної науково-практичної конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики» за напрямами: автоматизація теплоенергетичних процесів, геометричне моделювання та проблеми візуалізації, програмне забезпечення інформаційних систем та мережних комплексів, моделювання та аналіз теплоенергетичних процесів, сучасні проблеми сталого розвитку енергетики.

Друкується в авторській редакції за рішенням Вченої ради теплоенергетичного факультету Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (протокол № 9 від 29 квітня 2020 р.)

ISBN 978-966-622-997-0
ISBN 978-966-622-999-4 (Т.2)

Розрахунок еквідистант 3D моделей. 78
БОЙКО І.В., магістрант гр. ТМ-91мп
Керівник - доц., к.т.н. Демчишин А.А.
Система обліку енергоресурсів на основі блокчейну. 79
ШАПОВАЛ В.О., студент гр. ТМ-61
Керівник - доц., к.т.н. Сегеда І.В.
Автоматизація документообігу в страховій компанії. 80
СІКОЛЕНКО Е.В., студент гр. ТМ-61
Керівник - доц., к.е.н. Сегеда І.В.
Мікросервіс розрахунку мінімального габаритного циліндру 3D моделі. 81
СВЄТЛА Л.В., студент гр. ТР-61
Керівник - доц., к.т.н. Демчишин А.А.
Система збору та аналізу даних дорожньо-транспортного руху. 82
ПАЩЕНКО Д.О., студент гр. ТМ-61
Керівник - доц., к.е.н. Гусєва І.І.
Аналіз відеопотоку: класифікація кримінальних сцен. 83
ПАВЛЕНКО М.Р., студент гр. ТМ-61
Керівник - проф., д.е.н. Сігайов А.О.
Створення графічного запису трикотажу основов'язаних переплетень. 84
НАЗАРЧУК Д.К., студент гр. ТР-62
Керівник - проф., д.т.н. Аушева Н.М.
Блокчейн - регулятор просування цифрової економіки в енергетиці. 85
ЛОКОТАРЬОВ Є.О., студент гр. ТМ-62
Керівник - доц., к.е.н. Сегеда І.В.
Система обліку відвідування на основі технології біконів. 86
ЛЕБЕДИК Т.О., студент гр. ТМ-62
Керівник - доц., к.е.н. Гусєва І.І.
Проектування та розроблення web додатків на платформі контролювання доступу "intteks aks". 87
КОЧКАРЬОВ С.В., студент гр. ТМ-61
Керівник - проф., д.е.н. Сігайов А.О.
Інтерполяційна функція Гауса як засіб мобільного аналізу даних. 88
ГОРОДЕЦЬКИЙ М.В., студент гр. ТР-62
Керівник - доц., к.т.н. Сидоренко Ю.В.
Аналіз відеопотоку: ідентифікація людей за статтю та віковою групою. 89
ГЕРАСИМОВА М.В., студент гр. ТВ-61
Керівник - проф., д.е.н. Сігайов А.О.
It-solutions in ukraine's energy sector 90
КРИВДА Д.О., студент гр. ТВ-91
Керівник - доц., к.е.н. Кривда О.В.

СЕКЦІЯ №9 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ТА МЕРЕЖНИХ КОМПЛЕКСІВ 91

Система генерації сигналу з використанням геометричної моделі розповсюдження звуку в водному середовиші. 92
ЄВТУШЕНКО А.М., аспірант
Керівник - доц., к.т.н. Гагарін О.О.
Інтелектуалізація CAM-систем. 93
БАРАНІЧЕНКО О.М., аспірант
Керівник - доц., к.т.н. Шаповалова С.І.
Генерація нових образів на основі нейронних мереж. 94

УДК 342.951

Студент 4 курсу, гр. ТМ-61 Шаповал В.О.
Доц., к.т.н. Сегеда І.В.

СИСТЕМА ОБЛІКУ ЕНЕРГОРЕСУРСІВ НА ОСНОВІ БЛОКЧЕЙНУ

Поняття блокчейну є одною з найактуальніших тем за останні декілька років. Кожен день технологія успішно знаходить застосування в нових сферах та процесах, що дозволяє прибрати посередників та спростити життя людини [1].

На сьогоднішній день технології, які використовуються для обліку, купівлі та продажу енергоресурсів, вимагають істотних та кардинальних змін. Складність моніторингу та контролю ресурсів свідчать про необхідність переходу на сучасну автоматизовану систему. Блокчейн може бути використаний для позитивних змін в структурі енергетичного ринку, здешевлення вартості тарифів та відмову від посередників в мережі. Саме тому пропонується розробити систему обліку енергоресурсів на основі блокчейну [2].

Система по своїй суті представлена у вигляді децентралізованої мережі вузлів для кожного користувача з однаковим функціоналом в залежності від ролі та одного завантажуючого сервера який виконує інфраструктурну задачу та передає додаткову технічну інформацію між користувачами, відповідає за синхронізацію блокчейну між вузлами [3]. Завдяки своїй структурі і принципу роботи, в системі неможлива підробка даних.

Принцип роботи полягає в тому, що існує ланцюжок (chain), який складається з блоків, які формуються за фіксований проміжок часу. Ці блоки отримують і локально зберігають всі користувачі системи. Блок, в свою чергу складається з підписаних користувачами транзакцій.

Система складається з децентралізованих рівноправних вузлів (користувачів), де кожен користувач локально має ланцюжок блокчейну. Крім ролі звичайного користувача, система містить майнера (обрахунок хешу блока) та завантажуючого сервера (підтримує інфраструктуру шляхом передачі новим вузлам інформацію майнерів та користувачів)

Для того, щоб новому користувачу почати роботу, необхідно запустити на локальній машині сервер, який буде підписувати транзакції, отримувати, перевіряти та обновляти ланцюжок новими блоками; та сервер, який надає змогу використовувати графічний інтерфейс.

Основні задачі, які система вирішує:

1. Купівля енергоресурсів без посередників
2. Продаж енергоресурсів без посередників
3. Аналітичні відомості про транзакції користувача

Висновки: Використання технології блокчейн в енергетичному секторі дозволяє суттєво зменшити вартість енергоресурсів через відмову від посередників та прозорість системи, уникаючи шахрайства та монополії на ринку.

Перелік посилань:

1. Блокчейн: как он работает, и почему эта технология изменит мир [Електронний ресурс] /– Режим доступу до ресурсу: https://habr.com/ru/company/iticapital/blog/340992/
2. Nakamoto S. A Peer-to-Peer Electronic Cash System [Електронний ресурс] // Bitcoin. – Режим доступ до ресурсу: https://bitcoin.org/bitcoin.pdf
3. Сегеда І.В., Локотарев Є.О., Шаповал В.О. Реалізація використання блокчейн-технологій у енергетичному секторі. / Вчені записки Таврійського національного університету імені В.І. Вернадського Серія:Економіка і управління Том 30 (69). № 4, 2019, С. 160-165 (DOI: https://doi.org/10.32838/2523-4803/69-4-51)

ДОДАТОК 5

Реалізація використання блокчейн-технологій у енергетичному секторі

ТЕКСТ НАУКОВОЇ СТАТТІ

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61183_20Б

Аркушів 10

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТАВРІЙСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ В.І. ВЕРНАДСЬКОГО

Журнал заснований у 1918 році

ВЧЕНІ ЗАПИСКИ
ТАВРІЙСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ В.І. ВЕРНАДСЬКОГО

Серія: Економіка і управління

Том 30 (69). № 4, 2019
Частина 2

Київ
2019

Науковий журнал

ВЧЕНІ ЗАПИСКИ
ТАВРІЙСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ В.І. ВЕРНАДСЬКОГО

Серія: Економіка і управління

Том 30 (69). № 4, 2019
Частина 2

Коректура • *О. Скрипченко*

Комп'ютерна верстка • *В. Удовиченко*

Адреса редакції:
Таврійський національний університет імені В.І. Вернадського
м. Київ, вул. Івана Кудрі, 33
Телефон редакції: +38 (095) 430 01 12
Електронна пошта: editor@econ.vernadskyjournals.in.ua
Сторінка журналу: www.econ.vernadskyjournals.in.ua

Формат 60x84/8. Гарнітура Times New Roman.
Папір офсетний. Цифровий друк. Обл.-вид. арк. 19,14. Ум. друк. арк. 20,23.
Підписано до друку 27.09.2019. Замов. № 0919/198. Наклад 150 прим.

Видавництво і друкарня – Видавничий дім «Гельветика»
73034, м. Херсон, вул. Паровозна, 46-а, офіс 105
Телефон +38 (0552) 39 95 80
E-mail: mailbox@helvetica.com.ua
Свідоцтво суб'єкта видавничої справи
ДК № 6424 від 04.10.2018 р.

4. РОЗВИТОК ПРОДУКТИВНИХ СИЛ І РЕГІОНАЛЬНА ЕКОНОМІКА

Желізко Ю.М.
КЛЮЧОВІ ВІДМІННОСТІ СИСТЕМИ ЕКСПОРТНОГО КОНТРОЛЮ УКРАЇНИ ТА ПРОВІДНИХ ДЕРЖАВ СВІТУ **86**

Коротя М.І.
ЄВРОПЕЙСЬКИЙ ДОСВІД РЕГУЛЮВАННЯ РИНКУ ПРИРОДНОГО ГАЗУ: РОЗПОДІЛ, ТРАНСПОРТУВАННЯ **93**

Мороз С.Р., Феленчак Ю.Б.
СУЧАСНІ ТЕНДЕНЦІЇ РОЗВИТКУ САНАТОРНО-КУРОРТНОГО ГОСПОДАРСТВА У ТУРИСТИЧНОМУ КОМПЛЕКСІ УКРАЇНИ **99**

Pozdniakova Anna
ANALYSIS OF SMART CITY ARCHITECTURE MODELS **105**

5. ДЕМОГРАФІЯ, ЕКОНОМІКА ПРАЦІ, СОЦІАЛЬНА ЕКОНОМІКА І ПОЛІТИКА

Зуб М.Я.
СУЧАСНА ПАРАДИГМА ТРАНСФОРМАЦІЇ ІНСТИТУЦІЙ СОЦІАЛЬНО ОРІЄНТОВАНОГО РИНКУ ПРАЦІ **111**

Приятельчук О.А.
ФОРМУВАННЯ КОНЦЕПЦІЇ СОЦІАЛЬНОЇ ЛЮДИНИ В СУЧАСНИХ ЕКОНОМІЧНИХ МОДЕЛЯХ **117**

6. ГРОШІ, ФІНАНСИ І КРЕДИТ

Лантух К.О.
ПІДХОДИ ТА МЕТОДИ ФІНАНСУВАННЯ КУЛЬТУРИ ТА МИСТЕЦТВА: СВІТОВИЙ ДОСВІД І ПЕРСПЕКТИВИ ДЛЯ УКРАЇНИ **123**

Тищенко В.В.
КРАУДФАНДІНГ ЯК ФІНАНСОВИЙ ІНСТРУМЕНТ РЕАЛІЗАЦІЇ ІНВЕСТИЦІЙНИХ ПРОЕКТІВ **130**

7. БУХГАЛТЕРСЬКИЙ ОБЛІК, АНАЛІЗ ТА АУДИТ

Лега О.В.
ПРОФЕСІЯ «БУХГАЛТЕР»: ВІД МИНУЛОГО ДО ВИМОГ СУЧАСНОСТІ **139**

Скорнякова Ю.Б.
ЄДИНИЙ ПОДАТОК ЮРИДИЧНИХ ОСІБ: ПИТАННЯ ВІДОБРАЖЕННЯ В ОБЛІКУ ТА ФІНАНСОВІЙ ЗВІТНОСТІ **146**

Rozit Tatiana, Chorna Anna
THE PROBLEMS AND CHRONOLOGY OF INTEGRATION OF INTERNATIONAL FINANCIAL REPORTING STANDARDS INTO UKRAINIAN NATIONAL ACCOUNTING REGULATIONS (STANDARDS) **154**

8. МАТЕМАТИЧНІ МЕТОДИ, МОДЕЛІ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ЕКОНОМІЦІ

Сегеда І.В., Локотарев Є.О., Шаповал В.О.
РЕАЛІЗАЦІЯ ВИКОРИСТАННЯ БЛОКЧЕЙН-ТЕХНОЛОГІЙ У ЕНЕРГЕТИЧНОМУ СЕКТОРІ **160**

Вчені записки ТНУ імені В. І. Вернадського. Серія: Економіка і управління

8. МАТЕМАТИЧНІ МЕТОДИ, МОДЕЛІ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ЕКОНОМІЦІ

DOI: https://doi.org/10.32838/2523-4803/69-4-51

УДК 336. 621

Сегеда І.В.
кандидат економічних наук, доцент,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Локотарев Є.О.
бакалавр,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Шаповал В.О.
бакалавр,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Segeda Irina, Lokotariev Eugene, Shapoval Vitaliy
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

РЕАЛІЗАЦІЯ ВИКОРИСТАННЯ БЛОКЧЕЙН-ТЕХНОЛОГІЙ У ЕНЕРГЕТИЧНОМУ СЕКТОРІ

У статті проаналізовано процеси входження криптовалют в енергетичний сектор, питання подолання певної багатозначності їх віртуального та практичного використання. Автори концептуалізують цей процес як цифрову блокчейн-революцію. Зазначено, що блокчейн є механізмом верифікації транзакцій у мережі, підвищує довіру клієнтів і дає змогу позбутися посередників у фінансових операціях. Показано потенційні переваги «розумних контрактів», які здатні не тільки змінити технологічний уклад, але й повністю змінити взаємини суб'єктів суспільства, а також взаємини між суспільством і державою. Розглянуто децентралізовану модель енергетичних транзакцій та енергопостачання. Представлено головні тренди блокчейну в енергетиці. Запропоновано алгоритм системи обліку споживання та оплати платежів за енергоресурси як блокчейн-додаток, який буде реалізовувати одноранговy взаємодію між користувачами.

Ключові слова: *криптовалюта, блокчейн, блокчейн-технології, смарт-контракт, децентралізована модель, енергетичні транзакції.*

Постановка проблеми. Розвиток IT-технологій щодня пропонує світу нові інструменти для оптимізації бізнес-процесів. Одним з останніх таких інструментів є технологія блокчейну (blockchain technology).

Думки експертів про ідею впровадження криптовалют розділилися: одні вважають це справді революційним, але не зовсім розуміють, чи вдасться здійснити цю революцію, а інші називають це інноваціями, які потребують значної адаптації та не є революційними. Отже, питання подолання певної багатозначності їх віртуального й практичного використання є актуальними та потребують подальшого дослідження.

В сучасних умовах прийнята технологія обліку й контролю енергоресурсів застаріла через організаційну та технічну недосконалість структур, що здійснюють облік. Ці проблеми стають причиною постійних збитків, що явно свідчить про необхідність створення сучасної автоматизованої системи.

Аналіз останніх досліджень і публікацій. Найвідоміша робота з технології блокчейну датується 2008 роком. Її автором є Сатоші Накамото [1], що описує технічні рішення, які використовуються в проектах Bitcoin і Blockchain. Одночасно в документі з'явилися терміни "bitcoin" та "blockchain", які досі часто плута-

ють. Вперше технологія блокчейну була впроваджена у 2009 році як інструмент проведення операцій з біткоїнами в електронній валюті, тобто вона була спільним реєстром транзакцій. Нині блокчейн-технологія розвивається як автономна технологія, яку можна використовувати поза системами криптовалют. Ден Тапскотт і Алекс Тапскотт, що є авторами публікацій з блокчейну, проаналізували програми, послуги, бізнес-моделі, ринки, організації та навіть уряди, які керують блокчейном. Визначено закономірності, сформульовано сім принципів, на які покладаються послідовники технологій [2].

Формулювання цілей статті. Метою статті є систематизація інформації про технологію блокчейну, аналіз її перспектив та варіантів використання в енергетиці, вивчення основних блокчейн-тенденцій в енергетиці, розгляд алгоритму системи обліку споживання та платежів за енергоресурси.

Виклад основного матеріалу. Енергетичний сектор перебуває на стадії переходу від централізованого ринку до децентралізованого, оцифрованого сектору, в якому люди та компанії можуть досягти повної автономії шляхом виробництва та зберігання енергії.

Парадигма на поточне десятиліття, яка спирається на розрахунки, засновується на криптографії блокчейну. На наш погляд, це є процесом цифрової блокчейн-революції. Революційний потенціал блокчейн-технологій буде розвиватись та впроваджуватись досить швидко завдяки наявності Інтернету та мобільного зв'язку. Блокчейн стає невіддільним економічним шаром Інтернету. Реалізований функціонал в процесі цифрової блокчейн-революції – це повноцінний економічний шар, якого в Інтернеті ще не було. На інтегрованому фізичному рівні перебувають розрахунки з багатьма пристроями, на вершині яких є шар для оброблення таких платежів, як мікроплатежі, децентралізована біржа, заробіток та витрати токенів, отримання та передача цифрових активів, створення та виконання смарт-контрактів. Простіше кажучи, якщо ця революційна технологія буде впроваджена у повсякденне життя, тоді контроль над банками, державними установами, аудиторами, регуляторами, страховими компаніями чи реєстраторами просто не знадобиться. Сьогодні блокчейн дає змогу трансформувати цілі галузі економіки й економити великі кошти.

Нині існують три концепції, цілі та перспективи є різними, але вони мають одну технологічну базу та учасників [3].

Криптовалюта передбачає мінімізацію потреби в довірі, адже її концепція полягає у здійсненні безпечних операцій без централізованого контролю. Блокчейн-трекінг передбачає, що учасники мережі можуть приймати спільне рішення щодо інформації, яка перебуває в межах довіри. Криптоактиви відповідають за торгівлю, адже ця концепція передбачає, що віртуальні валюти можуть бути фінансовими інструментами та використовуватися як продані активи.

Слід зазначити, що вони не є взаємовиключними.

Автори зосереджуються на використанні технології блокчейну. У звіті Всесвітнього економічного форуму (ВЕФ) наведено таке визначення технології "blockchain", або технології розподіленого реєстру (distributed ledger technology, DLT): технологічний протокол, що дає змогу обмінюватися даними безпосередньо між різними сторонами в мережі, не потребуючи посередників [4]. Учасники мережі зв'язуються із зашифрованими ідентифікаторами (анонімно), а потім кожна транзакція додається до незмінного ланцюга транзакцій і розподіляється по всіх мережевих вузлах.

Через активний розвиток технології "blockchain" «розумні контракти» замінили звичайні контракти. «Розумні контракти» (смарт-контракти) – це одне із застосувань блокчейну, яке викликає найбільший інтерес. «Розумний контракт» – це договір між двома сторонами, що зберігається в блокчейні. Такі договори можуть укладатися між двома людьми, (P2P), людиною та організацією (P2O), людиною та машиною (P2M). «Розумні контракти» дають змогу автоматизувати виплати та перекази валют чи інших активів відповідно до встановлених умов. Як тільки умова, зазначена у смарт-договорі, виконується, договір укладається автоматично, відбувається обмін активами (готівкою, цифровою валютою, правом власності) між сторонами, які домовляються. Потім транзакція реплікується та перевіряється у ланцюзі блоків. «Розумні контракти» дають змогу обмінятися активами, якщо треті сторони не знають про передачу. Це відкриває можливість створення нової форми віртуального контракту. Однак через фрагменти коду, які автоматично виконують дії за певних умов, «розумні договори» ще не можуть конкурувати зі звичайними контрактами. Проте вони можуть бути використані як доказ розв'язання конкретної задачі [5].

Назвемо можливості використання блокчейну в енергетичній галузі. Блокчейн-технологія здатна докорінно змінити енергетичну систему спочатку шляхом трансформації окремих секторів, а потім шляхом трансформації всього ринку електроенергії.

Міжнародні енергетичні компанії розробляють проекти, які надалі з'єднають усіх споживачів в одну мережу, тобто децентралізовану систему. За допомогою «розумних контрактів» буде спрощена наявна багаторівнева система, що складається з виробників електроенергії, операторів розподільної мережі, операторів-постачальників, постачальників платіжних послуг банківських послуг, споживачів та трейдерів. Усі транзакції щодо отримання енергії та оплати за неї здійснюватимуться безпосередньо в мережі, об'єднуючи рівних учасників, тобто споживачів та виробників енергії. Завдяки цьому електроенергія буде дешевою.

Крім того, всі транзакції будуть відкритими. Люди не зможуть прострочити платіж за споживання енергії, адже «розумний контракт» контролюватиме виконання всіх операцій. Система сама заплатить за себе, тобто спише стільки криптовалюти, скільки вам знадобиться для транзакції з передачі енергії.

Моделі транзакцій на блокчейні базуються на тому, що вся електроенергія, що подається в електромережу, може бути чітко віднесена до обліку конкретних спо-

живачів за короткий проміжок часу. Це означає, що розрахунок за всю вироблену та спожиту електроенергію може бути дуже точно проведений за змінними цінами. Електрика буде продовжувати надходити до кінцевого споживача безпосередньо від найближчого виробника електроенергії. База даних, що зазнала значного поліпшення, дасть змогу точно «налаштувати» операції в мережі як на рівні передачі електроенергії, так і на рівні розподілу. Спрощений процес взаєморозрахунків приведе до зниження обсягу балансу енергії, рахунки на яку виставляються учасникам ринку.

Завдяки блокчейну всі потоки електроенергії захищені від сторонніх маніпуляцій. Це дасть змогу сертифікувати електроенергію, перевірити квоти на допустимі викиди, кількість яких регулюється законодавством. Децентралізована технологія функціонує як база даних транзакцій, побудована за принципом розподіленого реєстру, тому за допомогою блокчейну можна створити універсальний архів для зберігання всіх даних за виставленими рахунками за електроенергію. Споживачі отримають можливість розширеного контролю за своїми договорами на постачання електроенергії, а також дані про споживання електроенергії. Всі записи зберігатимуться у відкритому доступі в блокчейн-реєстрі, який буде коригувати всі питання права власності та поточний стан активів, тобто розумних інтернет-речей (Інтернет речей, IoT).

Технологія блокування не тільки використовується для проведення операцій з постачання енергії, але й може слугувати основою для процесів вимірювання кількості споживаної електроенергії, виставлення рахунків за споживання кількості та проведення розрахунків. Інші можливі додатки включають право власності на активи, управління активами, систему сертифікатів гарантованого походження, квоти на викиди вуглекислого газу та сертифікати, що підтверджують виробництво електроенергії на основі використання відновлюваних джерел енергії (ВДЕ). Можливості використання технологій в енергетиці представлені в табл. 1.

Під час об'єднання окремих блокчейн-додатків у майбутньому може з'явитися децентралізована система енергетичних транзакцій та постачання енергії. Постачання електроенергії, виробленої на об'єктах малої енергетики, кінцевим споживачам здійснюватиметься через мікромережі. Кількість виробленої та спожитої електроенергії вимірюватиметься за допомогою розумних лічильників, а операції з торгівлі енергією та сплата криптовалютою контролюватимуться за допомогою смарт-контрактів та здійснюватимуться з використанням блокчейну.

Слід зазначити, що наявні блокчейн-додатки можна розділити на такі три великих категорії залежно від рівня розроблення, як блокчейн-додатки версій 1.0, 2.0 та 3.0. Технологія блокчейну нового покоління, тобто блокчейн 3.0, ще розробляється. Blockchain 3.0 – це етап розвитку технологій, на якому здійснюється подальший розвиток концепції «розумного контракту» задля створення децентралізованих, автономних організаційних підрозділів, які керуються власними законами та працюють майже незалежно. Децентралізована система енергетичних транзакцій та постачання енергії представлена на рис. 1.

Прозоре та децентралізоване врегулювання угод на вітчизняному енергетичному ринку збільшить частку електроенергії, отриманої від відновлюваних джерел енергії.

Блокчейн чітко фіксує джерело походження кожної кіловат-години в загальній мережі й дає покупцю гарантію, що він отримає енергію вітру, а не енергію, згенеровану газовою станцією.

Отже, доцільно поєднувати нову технологічну систему з інноваційною ідеєю та блокчейн-технологією в тих установках, які генерують «зелену» енергію, тобто екологічно чисту й невичерпну за людськими мірками енергію сонця, вітру, місяця, води, гейзерів тощо. Для адаптації відновлюваних джерел енергії до повсякденного життя необхідно автоматизувати систему за допомогою спеціального обладнання нового типу [6].

Назвемо основні тренди технологій блокчейну в енергетиці.

1)Вихід на ринок блокчейн-технологій компаній гігантів. Партнери-засновники "Rocky Mountain Institute" та "Grid Singularity" мають намір створити нову платформу для торгівлі енергоресурсами на основі блокчейну (Енергетична веб-платформа), яка забезпечує функціонал, необхідний для реалізації різних варіантів використання в енергетичному секторі.

2)Знищення оптового ринку електроенергії. Компанія "Ponton" має на меті створення першого розподіленого оптового ринку для позабіржової торгівлі

Таблиця 1

Різні варіанти використання блокчейну в енергетиці

Транзакції та «розумні контракти»	Права власності на активи та управління ними	Децентралізовані інформаційні системи
Децентралізована торгівля електроенергією	Реєстрація власності та ведення реєстру активів	Облік електроспоживання та виставлення рахунків за електроенергію
Особливі можливості для просьюмерів	«Зелені» сертифікати	Облік споживання тепла та виставлення рахунків за нього
Впровадження криптовалюти	Квоти на викиди вуглекислого газу, сертифікація виробництва електроенергії на основі відновлюваних джерел енергії	Оплата зарядки електромобілів
Зарядка електромобілів		
Управління розумними пристроями в Інтернеті речей		

Рис. 1. Децентралізована система енергетичних транзакцій та енергопостачання

оптовою енергетичної продукцією за допомогою свого проекту "Enerchain".

3)Підвищення енергоефективності. У Німеччині проект працює з компанією "Sonnen", щоби використовувати склад як буфер для поглинання перевантажень електроенергії від вітроелектростанцій. Ця система виявилась корисною для зменшення витрат на скорочення вітрогенераторів.

4)Торгівля енергією за допомогою токенів. Проекти спрямовані на використання блокчейну для забезпечення торгівлі електроенергією. Проекти створюють ринок локально генерованої електроенергії, "Grid+" зосереджується більше на зниженні роздрібних цін на енергоносії, тоді як "Power Ledger" зосереджується на створенні ринку надлишкової енергії [7].

5)Розвиваючі стартапи. Згідно з даними "Reuters" енергетичні блокчейн-стартапи зібрали близько 200 мільйонів доларів завдяки "Initial coin offering" у 2018 році. Згідно з даними провайдера даних ринку "PitchBook" венчурні інвестиції капіталу в криптовалюти та блокчейн-стартапи встановлять новий рекорд у 2019 році.

Необхідно звернути увагу на проблему обліку споживання та збирання платежів за енергоресурси. Технологія обліку й контролю енергоресурсів, яка сьогодні застосовується, завдає шкоди суб'єктам господарювання, які її використовують. Головною причиною є організаційна й технічна недосконалість структур, які здійснюють облік. Ці проблеми стають причиною постійних збитків, що свідчить про необхідність створення сучасної автоматизованої системи.

Авторами розроблено алгоритм системи обліку споживання та платежів за енергоресурси, на основі якого буде розроблена автоматизована система.

Планується створити додаток, який буде реалізовувати однорангову взаємодію між користувачами (один з них буде оплачувати рахунки за енергоресурси, другий буде, власне, надавати ці енергоресурси). Той, хто оплачує використання енергоресурсів, водночас зможе надавати їх (наприклад, людина платить за газ, але у неї стоїть вітряна електростанція, тому надлишок виробленої електроенергії вона продає в мережу).

Додаток буде мати два типи серверів.

1)Сервер, який буде працювати тільки з блокчейном. Блокчейн з усіма транзакціями буде зберігатись в нереляційній базі даних (БД). Сервер буде проводити валідацію транзакцій і всього ланцюжка, здійснювати переказ криптовалюти (планується вводити свою криптовалюту, якою сплачуватимуться рахунки). Якщо брати як приклад систему «Біткоїн», то кожному користувачу встановлюється на ПК програма, яка зберігає ланцюжок і надає функціонал для роботи з нею. В нашому випадку кожному користувачу планується розгортати вузол (екземпляр сервера) в кластері з окремою БД. Іншими словами, інфраструктура буде представлена децентралізованою системою серверів.

2)Сервер, який буде надавати функціонал користувача. Всі дані будуть зберігатися в реляційній БД. Планується розгорнути один сервер, який буде давати можливість працювати зі своїми рахунками й контрактами. З іншого боку, можна сказати, що цей сервер буде проміжною ланкою між користувачами, а його основним завданням буде комунікація між користувачами.

– Користувачі зможуть вручну вносити показання лічильників (можна ще зробити заглушку, яка буде імітувати роботу лічильника, який автоматично вносить на рахунок показники за використані енергоресурси).

– Користувачі зможуть здійснювати оплату рахунків за енергоресурси.

– Користувачі зможуть бачити різноманітну статистику за надані (якщо користувач надає енергоресурси) і, відповідно, спожиті енергоресурси (якщо він їх споживає).

– Можна імітувати інтеграцію з банківськими сервісами (перетворення паперових грошей на криптовалюту, яка буде використовуватися в додатку). В реальних умовах лічильники мають можливість спілкуватись із сервером і передавати йому свідчення за використані енергоресурси.

Наведемо алгоритм роботи блокчейну.

Ланцюжок складається з набору блоків. Кожен блок зберігає в собі транзакції. Транзакція містить публічний ключ (адресу гаманця) відправника й одержувача. Також у транзакції зберігаються грошова сума, яку відправник передає одержувачу, та інформація про транзакції, яку заповнює користувач або сама система.

Користувач має також приватний ключ, яким підписує транзакції, говорячи «Це мій переказ грошей». Приватний ключ доступний тільки користувачу, тому ніхто, крім нього, не має до нього доступу. За допомогою асиметричного алгоритму шифрування є можливість перевірити, що користувач із приватним ключем – це користувач із публічним ключем. Це й буде гарантією того, що дані не були підроблені.

Користувач підписує транзакції за допомогою свого приватного ключа. Отже, є можливість перевірки «публічний ключ = підпис».

У разі невалідності транзакції (коли підпис і публічний ключ відправника не будуть збігатися) вона не буде розглядатися як об'єкт подальшого майнінгу.

Наприклад, у системі «Біткоїн» накопичуються транзакції. Далі ці транзакції валідуються майнерами. За успішної валідації всіх транзакцій формується новий блок, у якому зберігаються всі валідні транзакції, переводяться гроші з рахунку на рахунок, а майнер отримує нагороду.

Кожен блок у ланцюжку має обчислений хеш, а також хеш попереднього блоку (виняток стосується тільки першого блоку, в якому немає хешу попереднього блоку). За зміни будь-якого хешу відбудеться перерахунок хешу всіх наступних блоків, що буде говорити про те, що ланцюжок став не валідним. Саме ця особливість дає змогу будувати однорангові з'єднання між користувачами, які не можуть підробити інформацію про транзакції.

Висновки. Проведені дослідження показали, що технологія блокчейну активно розвивається й буде отримувати суттєві інвестиції. Частина галузей будуть революційним чином перебудовані; децентралізація змінить бізнес-логіку багатьох компаній та сервісів. За ступенем того, як ринок лібералізується, а ВДЕ зростає, технологія блокчейну пропонує спосіб, що дає змогу краще справлятися з усе більш складними та децентралізованими транзакціями між користувачами, великими й дрібними виробниками, промисловими підприємствами та навіть роздрібними торговцями й комунальними компаніями.

Фінансові додатки на основі технології блокчейну вже досягли високого рівня. Однак тільки час покаже, чи зможе ця технологія зробити революцію в енергетичному секторі. Перші пілотні проекти дають загальне уявлення про колосальні вигоди, які можуть забезпечити блокчейн-додатки щодо економії на витратах, а також швидкості та гнучкості. Україні необхідно активно включатися в цю сферу. Використання всіх видів технологій блокчейну, включаючи криптовалюту, має стати основою державної стратегії.

У сучасних умовах прийнята технологія обліку й контролю енергоресурсів не виправдовує себе. Причиною є організаційна й технічна недосконаліть структур, що здійснюють облік. Це стає причиною постійних збитків, тому автори статті розробили алгоритм системи обліку споживання та оплати платежів за енергоресурси. Надалі на підставі цього алгоритму буде створена сучасна автоматизована система.

Список літератури:

1. Nakamoto S. A Peer-to-Peer Electronic Cash System Bitcoin, 1–8. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 04.05.2019).

2. Tapscott D., Tapscott A. Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World. New York : Penguin Random House, 2016. 432 p.

3. Скворцов В. Блокчейн – не революция, это две инновации и одна потенциально успешная идея. URL: https://vc.ru/crypto/41715-blokcheyn-ne-revolyuciya-eto-dve-innovacii-i-odna-potencialno-uspeshnaya-ideya (дата звернення: 12.06.2019).

4. Deep Shift – Technology Tipping Points and Societal Impact (2015) / World Economic Forum Survey Report. URL: http://www3.weforum.org/docs/WEF_GAC15_Technological_Tipping_Points_re-port_2015.pdf#page=24 (дата звернення: 10.05.2019).

5. Tar A. Smart Contracts, Explained. Cointelegraph. 2017. URL: https://cointelegraph.com/explained/smart-contracts-explained (дата звернення: 10.05.2019).

6. Segeda I. Blockchain as a digital economy promotion tool in energy Industry. *Modern Aspects of Software Development* : Proceedings of VI International Scientific and Practical Virtual Conference of Software Development Specialists, June, 24 2019. Kyiv : Igor Sikorsky KPI, 2019. P. 139–146.

7. Халезов А. Блокчейн и энергетика: 4 главных тренда. URL: https://medium.com/@khalezov/%D0%B1%D0%BB%D0%BE%D0%BA%D1%8 (дата звернення: 12.06.2019).

References:

1. Nakamoto S. (2009). Peer-to-Peer Electronic Cash System. *Bitcoin*, 1–8. Available at: https://bitcoin.org/en/bitcoin-paper (accessed: 04.05.2019).
2. Dan Tapscott & Alex Tapscott (2016). *Blockchain Revolution How the Technology Behind Bitcoin is Changing Money, Business, and the World*. New York : Penguin Random House. (in English)
3. Vadim Skvortsov (2018). Blokcheyn – ne revolyutsiya, eto dve innovatsii i odna potentsial'no uspeshnaya ideya [Blockchain is not a revolution, it is two innovations and one potentially successful idea]. Available at: https://vc.ru/crypto/41715-blokcheyn-ne-revolyuciya-eto-dve-innovacii-i-odna-potencialno-uspeshnaya-ideya) (accessed: 12.06.2019).
4. Deep Shift-Technology Tipping Points and Societal Impact (2015). *World Economic Forum Survey Report*. Available at: http://www3.weforum.org/docs/WEF_GAC15_Technological_Tipping_Points_re-port_2015.pdf#page=24 (accessed: 10.05.2019).
5. Tar A. (2017). Smart Contracts, Explained. Cointelegraph. Available at: https://cointelegraph.com/explained/smart-contracts-explained (accessed: 10.05.2019).
6. Segeda I. (2019). Blockchain as a digital economy promotion tool in energy Industry. Modern Aspects of Software Development: Proceedings of *VI International Scientific and Practical Virtual Conference of Software Development Specialists (Ukrainian, Kyiv, June, 24, 2019 r.)*. Kyiv : Igor Sikorsky KPI, pp. 139–146.
7. Aleksey Khalezov (2018). Blokcheyn i energetika: 4 glavnykh trenda [Blockchain and energy: 4 main trends]. Available at: https://medium.com/@khalezov/%D0%B1%D0%BB%D0%BE%D0%BA%D1%8 (accessed: 04.08.2019).

РЕАЛИЗАЦИЯ ИСПОЛЬЗОВАНИЯ БЛОКЧЕЙН-ТЕХНОЛОГИЙ В ЭНЕРГЕТИЧЕСКОМ СЕКТОРЕ

В статье проанализированы процессы вхождения криптовалют в энергетический сектор, вопросы преодоления определенной многозначности их виртуального и практического использования. Авторы концептуализируют этот процесс как цифровую блокчейн-революцию. Указано, что блокчейн является механизмом верификации транзакций в сети, повышает доверие клиентов и позволяет избавиться от посредников в финансовых операциях. Показаны потенциальные преимущества «умных контрактов», которые способны не только изменить технологический уклад, но и полностью поменять взаимоотношения субъектов общества, а также взаимоотношения между обществом и государством Рассмотрена децентрализованная модель энергетических транзакций и энергоснабжения. Представлены главные тренды блокчейна в энергетике. Предложен алгоритм системы учета потребления и оплаты платежей за энергоресурсы как блокчейн-приложение, которое будет реализовывать одноранговое взаимодействие между пользователями.

Ключевые слова: *криптовалюта, блокчейн, блокчейн-технологии, смарт-контракт, децентрализованная модель, энергетические транзакции.*

ACTUALIZATION OF BLOCKCHAIN-TECHNOLOGIES USE IN THE ENERGY SECTOR

Currently, blockchain technology is being developed as a stand-alone technology that can be applied outside of cryptocurrency systems. Blockchain and cryptocurrencies cannot be equated, to understand the essence of the matter, the authors consider features that are most clearly manifested when using blockchain technologies. The purpose of this study is to analyze the potential impact of blockchain technologies on the energy sector and explore the opportunities it may open to buyers and consumers of electricity. The paper analyzes the process of digital blockchain revolution and the impact of blockchain on the development of the economy's infrastructure. The functionality implemented in the process of the digital blockchain revolution may be a fully-fledged economic sphere that has not existed prior to the revolution. The modern concepts for the notions of cryptocurrency; blockchain; crypto assets given. In this paper options for using blockchain in the energy sector are presented. In particular, smart contracts allow you to interact with both: well-known and little-known partners, that is, they ensure trustful interaction between the parties automatically. The issue of assessing the complexity of transactions, a decentralized system of energy transactions and energy supply were examined in detail. Blockchain technology can provide the basis for creating a decentralized energy supply system. If conditions are created under which producers and consumers can interact directly, carrying out transactions directly in the network, electricity will become cheap. In this paper the main modern trends in the development of blockchain in the energy sector are considered. In today's world, the adopted technology of accounting and control of energy resources does not justify itself, and sometimes can be detrimental to the business entities using it. The paper presents an algorithm for a system of accounting for consumption and carrying out payments for energy resources on the basis of which an automated system will be developed. Blockchain technology is capable of fundamentally changing the energy system we are used to, first by transforming individual sectors, and ultimately by transforming the entire electricity market.

Key words: *cryptocurrency, blockchain, blockchain technology, smart contract, decentralized model, energy transactions.*