DOI: https://doi.org/10.36910/6775-2524-0560-2023-51-03
Баранчук Сергій Анатолійович, студент
Бортник Катерина Яківна, к.т.н., доцент
https://orcid.org/0000-0001-5282-099x
Луцький національний технічний університет, м. Луцьк, Україна

АВТОМАТИЗОВАНА СИСТЕМА ТОРГІВЛІ НА КРИПТОВАЛЮТНИХ БІРЖАХ НА ОСНОВІ NODE.JS ТА БІБЛІОТЕКИ REACT

Баранчук С.А., Бортник К.Я. Автоматизована система торгівлі на криптовалютних біржах на основі Node.js та бібліотеки React. Розроблено повноцінний Fullstack веб-додаток (веб-систему) для взаємодії з криптовалютним ринком. Були розроблені необхідні скрипти для отримання, обробки та зберігання даних, зручний інтерфейс та систему авторизації користувачів. Дана система дозволяє в автоматизованому режимі знаходити можливості для заробітку під час торгівлі на криптовалютних біржах.
Ключові слова: криптовалюта, блокчейн, веб-технології, JavaScript, React, Node.js, стек MERN.

Baranchuk S., Bortnyk K. Automated trading system on cryptocurrency exchanges based on Node.js and React library. A full-stack web application (SaaS system) for interacting with the cryptocurrency market was developed. Necessary scripts for data retrieval, processing, and storage, convenient user interface, and user authorization system were developed. This system allows for automated identification of earning opportunities during cryptocurrency trading.
Keywords: cryptocurrency, blockchain, web technologies, JavaScript, React, Node.js, MERN stack.

Постановка наукової проблеми. Однією з найбільш динамічно зростаючих галузей останніх років є криптовалютна галузь, яка наразі налічує понад 9 тис. криптовалют та більше 400 крипто-бірж. Високе різноманіття та конкуренція на ринку вимагають розробки відповідних інструментів, що стануть у нагоді трейдерам, а конкретніше – автоматизованих систем, що будуть аналізувати криптовалютні рикни. Ці системи повинні враховувати різні типи даних, такі як дані про курси, статистичні дані про ринок, дані про замовлення на купівлю та продаж криптовалют, в залежності від обсягу торгів на конкретній біржі та вимог до системи торгівлі. Функціональний опис автоматизованої системи для торгівлі повинен включати аналіз даних для прийняття рішень про покупку чи продаж криптовалют та моніторинг ринку для виявлення нових можливостей для торгівлі. Важливо також визначити, як буде відбуватися взаємодія з біржами та як будуть оброблятися отримані з них дані. Створення автоматизованої системи торгівлі на криптовалютних біржах є доцільним з точки зору автоматизації процесу торгівлі та збільшення ефективності прийняття рішень. Використання високопродуктивних, сучасних веб-технологій та підходів, таких як Node.js та React, може допомогти забезпечити швидке та надійне функціонування систем.

Аналіз досліджень. Основна стратегія торгівлі, на яку спрямована розроблювана система - це арбітраж. Криптовалютним арбітражем називають перепродаж криптовалют з метою отримання прибутку за рахунок різниці в ціні на різних ринках (біржах). Арбітражист виявляє можливості купівлі активів на одній біржі за нижчою ціною, з метою подальшого продажу їх на іншій біржі за вищою ціною, отримуючи прибуток в результаті.

Криптобіржі бувають двох типів: централізовані (CEX) та децентралізовані (DEX). Централізована криптобіржа (CEX) - це підхід, коли кошти користувачів зберігаються безпосередньо на платформі, а процес обміну контролюється системою. Зазвичай такі біржі вимагають верифікацію особистості, щоб користувачі могли отримати доступ до усіх функцій та можливостей. Децентралізована криптобіржа (DEX), навпаки, не зберігає кошти користувачів і не контролює процес обміну. Все відбувається через напряму через блокчейн і без посередників, а користувачі самостійно керують власними активами.

Цікаво відзначити, що часто централізовані біржі автоматично виявляють можливості арбітражу на своїх власних ринках і самі займаються внутрішнім криптовалютним арбітражем. Тому, при розробці торгової системи, основним пріоритетом буде орієнтація на міжбіржевий арбітраж.

У відміну від централізованих бірж, децентралізовані біржі майже не займаються самостійним арбітражем, тому варто окремо розглянути цей ринок. Як вже зазначено вище, процес торгів на DEX-біржах відбувається за допомогою звичайних блокчейн-транзакцій.

Блокчейн (рис. 1) є фундаментом криптовалютної індустрії, технологія якого полягає у розподіленій базі даних, що зберігається на багатьох комп'ютерах та містить записи про транзакції. Кожен блок у ланцюжку містить хеш попереднього блока, що забезпечує безпеку та надійність

системи. Коли нові транзакції виникають, вони збираються в блок, який містить хеш-функцію попереднього блока, що створює зв'язок між блоками у формі ланцюжка. Хеш-функція - це математична функція, яка перетворює вхідні дані в однозначний вихідний код фіксованої довжини, який унікально ідентифікує блок. Це забезпечує інтегритет даних та дозволяє виявляти будь-які зміни в блоках, так як будь-яка зміна в транзакції або блоку призведе до зміни його хеш-коду.

Після того як блок доданий до ланцюжка, він стає незмінним, що робить блокчейн надійним та стійким до фальсифікації даних. Кожен блок містить інформацію про транзакції, такі як дата, час, сума та відправник/одержувач, а також підтвердження криптографічного підпису, що забезпечує автентифікацію та безпеку транзакцій.

Одна з важливих особливостей блокчейн - розподіленість. База даних розподілена на багато комп'ютерів (вузлів), які співпрацюють між собою і підтверджують достовірність транзакцій. Це робить блокчейн відкритим, прозорим та має потенціал виключити посередників, такі як банки, забезпечуючи безпосередні транзакції між сторонами. Кожен вузол має копію повної бази даних блокчейну, і зміни в базі даних можуть бути внесені тільки після згоди більшості вузлів, що забезпечує консенсус в мережі.

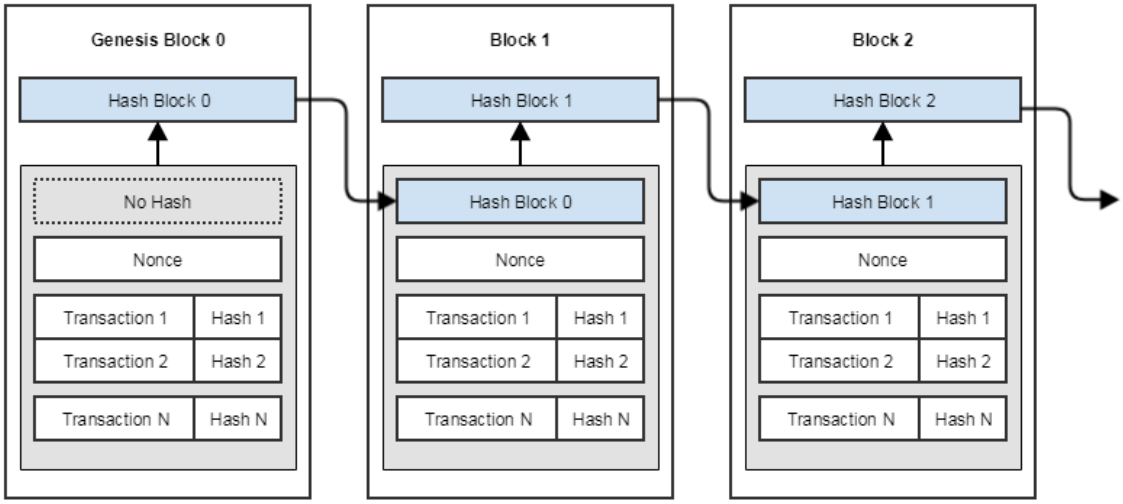

Рис. 1 – Загальна схема архітектури блокчейну

Блокчейн також може мати різні типи консенсусу, такі як "доказ роботи" (proof-of-work) або "доказ власності" (proof-of-stake), які використовуються для підтвердження транзакцій та додавання нових блоків до ланцюжка. Деякі блокчейн мережі також підтримують "розумні контракти" (smart contracts), які є самовиконуваними програмами, що автоматизують виконання умов та угод між сторонами без необхідності посередників.

Використання функціоналу "batching", який дозволяє комбінувати кілька транзакцій в один блок для подальшої обробки, може забезпечити ефективність операцій. Наприклад, під час криптовалютного арбітражу користувач може об'єднати свої операції до одного блоку та відправити його в блокчейн, зменшуючи загальну комісію, оскільки комісія сплачується лише за сам блок, а не за кожну окрему операцію.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження. Для розробки програмного забезпечення була вибрана мова програмування JavaScript з такої причини, що вона є однією з найпопулярніших мов для веб-розробки, має велику кількість бібліотек та фреймворків, що значно полегшує розробку програмного забезпечення. На основі JavaScript побудований стек технологій MERN (MongoDB, Express.js, React.js, Node.js), який дозволяє розробляти сучасні Fullstack веб-додатки та системи:

- MongoDB - це документо-орієнтована система управління базами даних, яка забезпечує зручний доступ до даних та можливість простого масштабування.

- Express.js - це веб-фреймворк на основі Node.js для розробки серверних додатків на мові JavaScript. Він дозволяє швидко та просто створювати веб-додатки з різноманітними функціями, такими як маршрутизація, обробка запитів та відповідей, робота з параметрами, обробка помилок і багато іншого.

- React.js - це бібліотека JavaScript для розробки користувацьких інтерфейсів (UI) веб-додатків. Вона дозволяє розробникам створювати ефективні та інтерактивні інтерфейси з використанням компонентної архітектури.

- Node.js - це середовище виконання JavaScript на серверному боці, яке базується на двигуні V8 JavaScript від Google. Воно дозволяє виконувати код JavaScript на серверному боці та забезпечує подієвий та неблокуючий ввід/вивід, що робить його ефективним для розробки масштабованих та високопродуктивних серверних додатків. Node.js також має багатий набір модулів, які дозволяють розширювати його функціональність для різних застосувань, таких як розробка веб-серверів, API, мережевих додатків, роботи з базами даних та багато іншого.

Для взаємодії з криптовалютними біржами, система буде використовувати REST API - архітектурний стиль для побудови веб-сервісів, що базується на протоколах HTTP і URI. REST API дозволяє взаємодіяти між клієнтом та сервером за допомогою HTTP-запитів, таких як GET, POST, PUT, DELETE та інші. Кожен ресурс у REST API представлений у вигляді URI-адреси, за якою можна звернутися для отримання або зміни його стану. Такий підхід дозволяє користувачам отримувати необхідні дані безпосередньо через URL-адресу веб-сайту. Приклад такого запиту зображений на рисунку 2. В коді системи кожна криптобіржа представлена у вигляді окремого класу, а отримані дані обробляються функціями-конструкторами та приводяться до єдиного вигляду, для подальшої обробки.

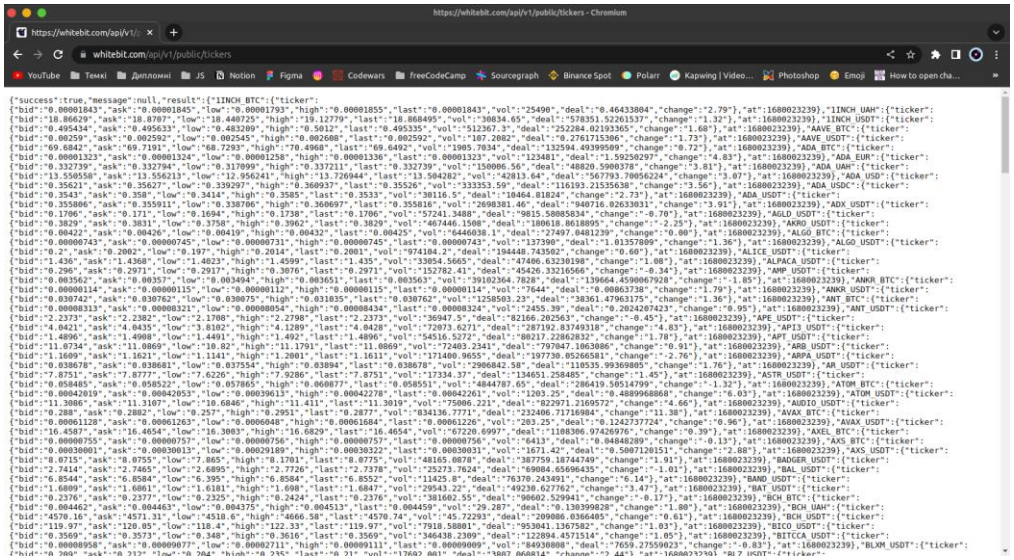

Рис. 2 – HTTP запит до криптобіржі на отримання цін

База даних MongoDB, яка використовується системою, містить таблицю "users", де зберігаються дані про користувачів системи. У цій таблиці є наступні поля:

- id (тип: ObjectId): унікальний ідентифікатор користувача, який автоматично генерується системою.
- fullName (тип: String): повне ім'я користувача.
- login (тип: String): логін користувача, який використовується для входу в систему.
- passwordHash (тип: String): хеш пароля користувача.
- userPhoto (тип: String): файл base64 з зображенням користувача.
- role (тип: String): роль користувача в системі.

Mongoose (JavaScript-фреймворк для взаємодії з MongoDB) автоматично додає поле "id" до всіх моделей даних. Наразі, база даних використовується лише для авторизації користувачів, але завдяки можливостям MongoDB, модель користувача може бути легко розширена, наприклад, для зберігання даних про транзакції, доходи користувача та інші. Основна архітектура створеної системи містить 3 шари:

- шар представлення даних на клієнтській стороні;
- сервер додатку;
- шар, що керує ресурсами у базі даних.

Ця архітектура може бути розширена до багатошарової, додавши додаткові сервери або інші об'єкти взаємодії. На рисунку 3 зображена трьохшарова архітектура веб-додатку з додатковим шаром, в який входять зовнішні джерела, з якими він взаємодіє.

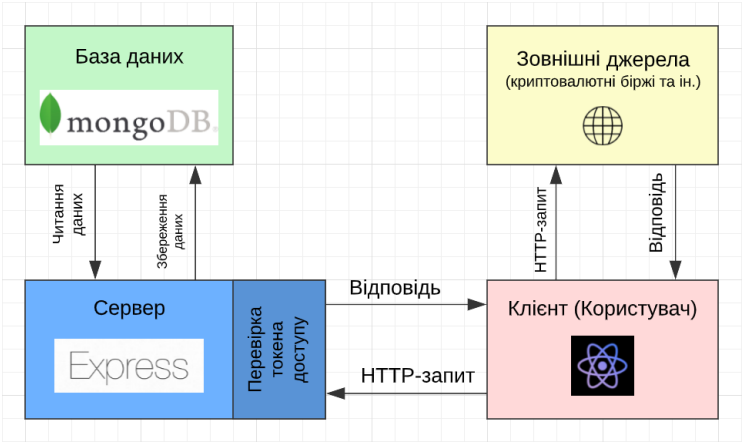

Рис. 3 – Архітектура розробленої системи

Для створення компонентів у React використовується JSX - розширення синтаксису JavaScript, яке дозволяє використовувати HTML-подібні теги в JavaScript коді для створення користувацького інтерфейсу. Це дає розробникам можливість використовувати знайомий синтаксис HTML для створення елементів інтерфейсу, які можуть містити вбудований JavaScript код для динамічних функцій та обробки подій. JSX є частиною бібліотеки React. Код JSX компілюється в стандартний JavaScript, який виконується в браузері або на сервері.

Усі компоненти розробленої системи мають дуже схожу структуру. Для прикладу, код головного компонента App, що відповідає за відображення розробленого React-проекту, показаний на рисунку 4.

```
function App() {
    const dispatch = useDispatch()
    const isAuth = useSelector(selectIsAuth)

    React.useEffect(() => {
        dispatch(fetchAuthMe())
    }, [])

    if (!isAuth) {
        return (
            <div className="app">
                <Routes>
                    <Route path="/" element={<HomePage/>}/>
                    <Route path="/login" element={<Login/>}/>
                </Routes>
            </div>
        )
    }

    return (
        <div className="app">
            <Routes>
                <Route path="/" element={<HomePage/>}/>
                <Route path="/login" element={<Login/>}/>
                <Route path="/profile" element={<Profile/>}/>
                <Route path="/adminpanel" element={<AdminPanel/>}/>
                <Route path="/settings" element={<Settings/>}/>
                <Route path="/cexArbitrage" element={<CEXArbitrage/>}/>
                <Route path="/crossExArbitrage" element={<CrossExArbitragePage/>}/>
            </Routes>
        </div>
    )
}

export default App;
```

Рис. 4 – Код базового компоненту системи

Спочатку, в компоненті App встановлюється зв'язок між компонентом і Redux store за допомогою хука useSelector. Змінна isAuth містить значення true, якщо користувач автентифікувався, і false в іншому випадку. В компоненті App також використовується хук useEffect з порожнім масивом залежностей, оскільки ми хочемо відправити запит на сервер лише один раз, при завантаженні компонента. Це досягається за допомогою функції dispatch, яка відправляє дію

fetchAuthMe до Redux store. Ця дія запитує інформацію про автентифікацію користувача на сервері. Якщо значення isAuth дорівнює false, то компонент повертає JSX код з двома маршрутами: / і /login. Якщо значення isAuth дорівнює true, то компонент повертає JSX код з декількома маршрутами: /, /login, /profile, /adminpanel, /settings, /cexArbitrage та /crossExArbitrage.

У цьому коді також використовується компонент Routes з бібліотеки react-router-dom, який дозволяє відображати різний вміст на основі шляху URL. Компонент Route визначає маршрути, пов'язані з певним вмістом, який відображається на сторінці.

Розроблена система має сторінку авторизації користувача (рис. 5), особистий кабінет (рис. 6), та безпосередньо, меню з пошуком можливостей для торгівлі (рис. 7, рис. 8).

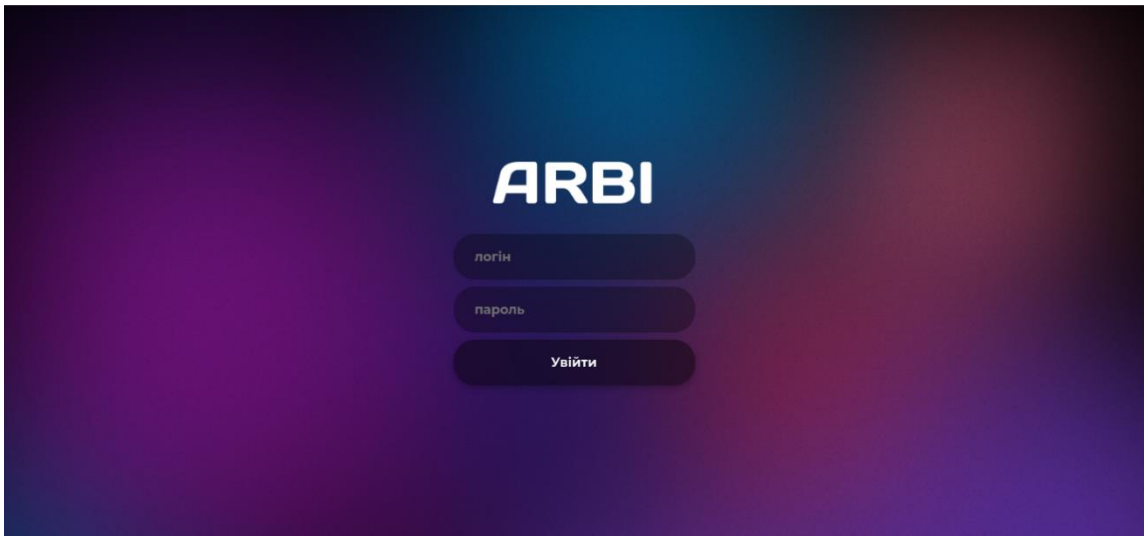

Рис. 5 – Сторінка авторизації користувача

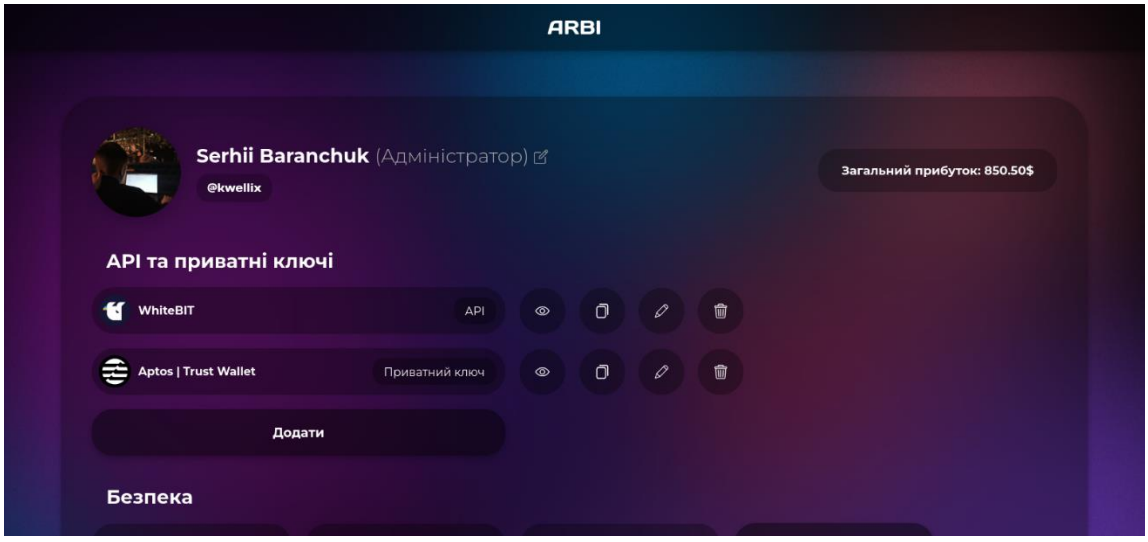

Рис. 6 – Особистий кабінет користувача

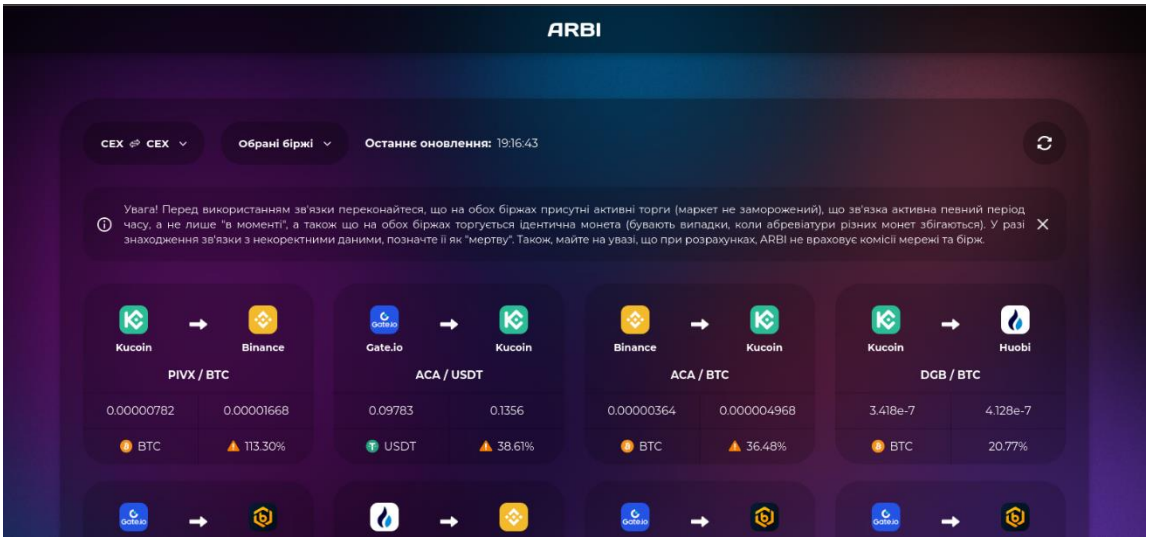

Рис. 7 – Таблиця знайдених для торгівлі можливостей

На рисунку 1.8 видно, щоб під час роботи з системою, була знайдена арбітражна зв'язка для валютної пари MDX/BTC. В даному випадку, трейдер купляє за наявні в нього біткоїни валюту MDX на одній біржі по ціні 0.00000252 BTC, і продає її на іншій біржі по ціні 0.00000299 BTC, тобто з різницею в +18.65%. Для детальнішого аналізу об'ємів потенційної покупки, необхідно дослідити книги замовлень на кожній з двох бірж. Переказ коштів між біржами відбувається за допомогою блокчейну, шляхом виведення коштів з першої біржі, на адресу гаманця користувача на другій біржі.

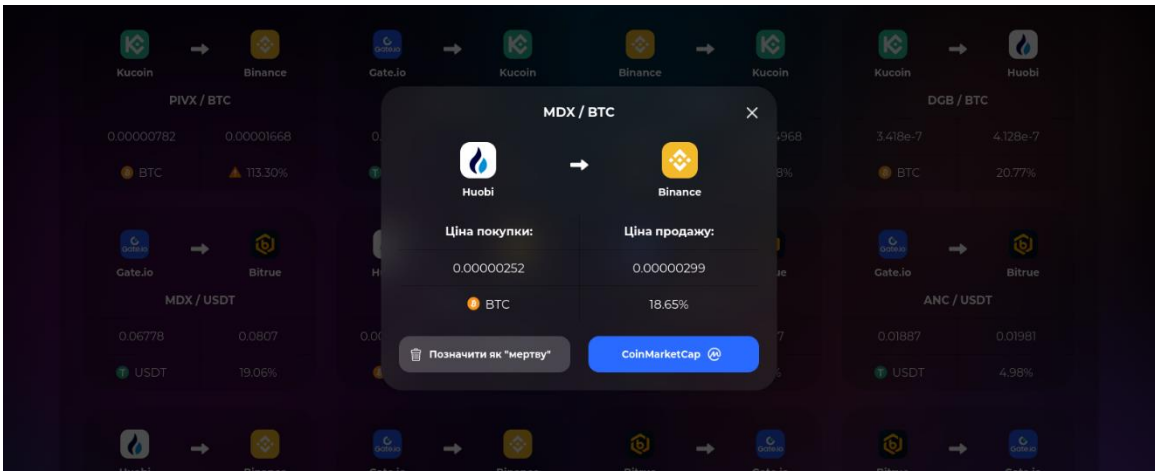

Рис. 8 – Детальний опис знайденої торгової можливості

Висновки та перспективи подальшого дослідження. У даній роботі було спроектовано та розроблено систему для автоматизації процесу торгівлі на криптовалютних біржах. Вона дозволяє в автоматичному режимі знаходити можливості для заробітку на криптовалютному ринку. Система розроблялася з використанням останніх сучасних веб-технологій, а тому має дуже хороші можливості для розширення функціоналу, масштабування та модернізації. Був забезпечений зв'язок з базою даних, процес авторизації користувачів, налагоджено процес отримання даних з зовнішніх джерел, для їх подальшої обробки та відображення результатів. Дана система може використовуватися трейдерами різного рівня з метою заробітку, а також легко бути перетворена в SaaS (програмне забезпечення як послуга) сервіс, та надаватися в користування на платній основі.

Список бібліографічного опису

1. Поченчук Г.М. Фінансові технології: розвиток і регулювання. Економіка і суспільство. 2017. №13. С. 1193—1200.
2. Балазюк О. Ю., Пилявець В. М. Технологія блокчейн: дослідження суті та аналіз сфер використання. Економіка та суспільство. 2022. № 43. С. 8.

3. Жеребцов О.А, Іжиков А.Ю Використання функціонального програмування у JavaScript та фреймворках. Інноваційні рішення в інженерії програмного забезпечення. 2022. №9. С. 162—169.
4. Крашівський А.І. Розробка веб-системи з використанням Node.js та MongoDB наприкладі системи автоматизації HR-процесів. Тернопільський національний технічний університет імені Івана Пулюя. 2022 р. с. 3
5. Міковскі Майкл, Пауелл Джош. Розробка односторінкових веб-додатків. ДМК Пресс, 2014. 512 с.
6. Lambert M. Surhone, Mariam T. Tennoe, and Susan F. Henssonow. 2010. Node.js. Betascript Publishing, Beau Bassin, MUS.

References

1. Pochenchuk H.M. Financial technologies: development and regulation. Economy and society. 2017. No. 13. P. 1193—1200.
2. Balazyuk O. Yu., Pyliavets V. M. Blockchain technology: a study of the essence and analysis of areas of use. Economy and society. 2022. No. 43. P. 8.
3. O. A. Zherebtsov, A. Izhikov. Use of functional programming in JavaScript and frameworks. Innovative solutions in software engineering. 2022. No. 9. P. 162—169.
4. Krashivskyi A.I. Development of a web system using Node.js and MongoDB, for example, an HR process automation system. Ternopil National Technical University named after Ivan Pulyu. 2022. p. 3
5. Mikowski Michael, Powell Josh. Development of single-page web applications. DMK Press, 2014. 512 p.
6. Lambert M. Surhone, Mariam T. Tennoe, and Susan F. Henssonow. 2010. Node.js. Betascript Publishing, Beau Bassin, MUS.