

ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально - науковий інститут математики та інформаційних технологій

Кафедра математики та інформатики

Сич Олександр Ігорович

ВЕБ-ДОДАТОК ДЛЯ КРИПТОВАЛЮТНОЇ ТОРГІВЛІ

кваліфікаційна робота
здобувача вищої освіти першого (бакалаврського) рівня
освітньої програми «Комп’ютерні науки та інформаційні технології»
за спеціальністю 122 „Комп’ютерні науки ”

Особистий підпис

Олександр СИЧ

Науковий керівник

Владислав КОЗУБ, д-р філософії

В.о.завідувача кафедри

Ліна БОНДАРЕНКО, к.пед.н.

Полтава – 2024

Міністерство освіти і науки України

ДЗ „Луганський національний університет імені Тараса Шевченка”
Навчально-науковий інститут математики та інформаційних технологій

Кафедра математики та інформатики

Освітній рівень бакалавр

Спеціальність 122 - Комп'ютерні науки

Галузь знань 12 Інформаційні технології

ЗАТВЕРДЖУЮ

В.о.зав. кафедри_

Ліна БОНДАРЕНКО

(підпис)

(ініціали, прізвище)

“ ” 202 р.

**ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ**

Сичу Олександрю Ігоровичу

1. Тема роботи

**Веб-додаток для криптовалютної
торгівлі**

Керівник кваліфікаційної роботи

Козуб В.Ю., д-р флософії

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету

**2. Строк подання здобувачем вищої освіти
проєкту**

03.06.2024

3. Вихідні дані до проєкту

Провести аналіз методів створення веб-додатків

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)

**4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити) Постановка задачі та аналіз предметної області**

Аналіз існуючих принципів побудови та засоби розробки веб-додатків

Розробка веб-додатку

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада Консультанта | Підпис, дата | |
|--------|--|-------------------|---------------------|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання « 15 » лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проекту (роботи) | Строк виконання етапів проекту (роботи) | Примітка |
|-------|---|--|----------|
| 1. | Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника. | до 1 лютого | |
| 2. | Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень. | другий тиждень лютого | |
| 3. | Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником. | до 1 квітня | |
| 4. | Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання. | до 15 квітня | |
| 5. | Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи. | перший тиждень квітня | |
| 6. | Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації. | до 20 травня | |
| 7. | Попередній захист роботи на кафедрі | Травень | |
| 8. | Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії | За 10 днів до державної атестації | |
| 9. | Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом. | За 5 днів до державної атестації | |

Студент

підпис

О.І. Сич

(ініціали, прізвище)

Керівник роботи

підпис

В.Ю.Козуб

(ініціали, прізвище)

АНОТАЦІЯ

Сич І. В. Веб-додаток для криптовалютної торгівлі. Кваліфікаційна робота бакалавра. 2024. 41с.

Бакалаврська робота складається зі вступу, 2-х розділів, списку використаних джерел, містить 14 рисунків.

У роботі представлено розроблений веб-додаток, описано його функціональне призначення, що полягає в наданні можливості відвідувачеві здійснювати: проводити торгівельні операції з криптовалютою.

В роботі представлено моделювання баз даних, розробка веб-додатків, дизайну інтерфейсу користувача, управління проектами через версії вихідного коду тощо.

Ця програма дозволяє користувачам купувати та продавати сотні криптовалют, отриманих із зовнішнього API.

Ключові слова: інтернет, база даних, мова програмування, веб-сторінка.

ABSTRACT

Sych O. I. Web application for cryptocurrency trading. Bachelor's qualifying work. 2024. 41 p.

The bachelor thesis consists of an introduction, 2 chapters, a list of used sources, contains 14 figures.

The work presents the developed web application, describes its functional purpose, which consists in providing the opportunity for the visitor to: carry out trading operations with cryptocurrency.

The work presents database modeling, web application development, user interface design, project management through source code versions, etc.

This application allows users to buy and sell hundreds of cryptocurrencies obtained from an external API.

Keywords: Internet, database, programming language, web page.

Зміст

| | |
|--|----|
| Вступ..... | 7 |
| Розділ 1. Використовувані технології..... | 9 |
| Розділ 2. Архітектура..... | 10 |
| 2.1. База даних | 10 |
| 2.2. Сервер..... | 13 |
| 2.2.1. Пакет моделей..... | 14 |
| 2.2.2. Пакет репозиторій..... | 15 |
| 2.2.3. Пакет послуг..... | 17 |
| 2.2.4. Пакет контролерів..... | 21 |
| 2.2.5. Пакет аналізу даних..... | 25 |
| 2.2.6. Пакет конфігурації..... | 26 |
| 2.2.7. Пакет виключень | 28 |
| 2.3. Інтерфейс..... | 29 |
| 2.3.1 Перегляд для реєстрації нових користувачів - /register або "/"..... | 29 |
| 2.3.2. Перегляд входу вже існуючих користувачів - /login | 31 |
| 2.3.3. Перегляд домашньої сторінки – "/home" | 32 |
| 2.3.4. Перегляд додавання та виведення коштів із програми – "/balance"..... | 32 |
| 2.3.5. Перегляд для відображення криптовалют, які належать користувачу, який увійшов у систему - "/myCrypto"..... | 33 |
| 2.3.6. Перегляд для відображення всіх операцій, здійснених користувачем, який увійшов у систему – "/myTransactions" | 35 |
| 2.3.7. Перегляд для відображення всіх транзакцій, здійснених усіма користувачами в додаток - "/allTransactions"..... | 36 |
| Висновки..... | 40 |
| Список використаних джерел..... | 41 |

ВСТУП

Гроші використовувалися щодня протягом тисячоліть назад, і оскільки ми живемо у світі з глобальною економікою, це відбувається потреба в глобальних цифрових валютах.

Криптовалюта — це віртуальна або цифрова валюта, яка працює на блокчейні.

Нинішня фінансова система централізована і керується великими хлопцями банки та уряди по всьому світу. З розвитком і прогресом світу ми приходимо необхідність децентралізованої системи, яка буде відрізнятися від існуючих сучасні способи оплати. Чудова аналогія децентралізованої системи наступний приклад. Якщо у нас є банкнота в 1000 гривень, то ми можемо роздати її безпосередньо будь-кому, незалежно одному або іншому об'єкту, наприклад, банку. За допомогою цифрових валют і блокчейн ми приходимо до такого сценарію.

Системи криптовалюти дозволяють здійснювати транзакції швидко, безпечно, забезпечують доступ до цих систем двадцять чотири години на добу, сім днів щотижня. Також немає додаткових комісій при здійсненні міжнародних платежів, і будь-хто може використовувати їх, просто створивши обліковий запис користувача.

Значна частина населення світу, хоча й має смартфон, проте не має доступу до фінансової системи. Сила криптовалют полягає в тому, що вони просто за допомогою смартфона та підключення до Інтернету, дають змогу бути частиною глобальної економіки.

Криптовалюти вже кілька років є гарячою темою розмов люди, інвестування в них, чи варто це того, як це працює, хто це контролює, чи є якісь схеми і тактики, які можна використовувати для отримання прибутку і так далі.

Оскільки блокчейн-системи непрості, контролюються окремою особою, групою людей, урядом або центром влади, то це є однією з найбільших негативних сторін, незаконної діяльності, що здійснюються через них.

Після всіх раніше зазначених причин, можна зазначити, що сама тема досить цікава, і тому актуальною задачею є побудова такої системи, яка забезпечить інтерфейс користувача, за допомогою якого продавці та покупці зможуть здійснювати покупки та продажі криптовалюти лише кількома простими кліками.

РОЗДІЛ 1. ВИКОРИСТОВУВАНІ ТЕХНОЛОГІЇ

Для розробки цього додатка було використано кілька технологій, які є в даний час дуже актуальними і використовуються щодня в багатьох ІТ-компаніях.

Для основних функцій програми, тобто для серверної частини, використано Spring Boot в поєднанні з мовою програмування Java. Всі інтерфейси користувача побудовані в React JS, бібліотеці JavaScript.

Перш ніж почати реалізацію інтерфейсної частини, необхідно протестувати усі API, доступні з контролерів, за допомогою Postman. Перш за все, перевірено виклики API [1] завдяки тому, що в Postman слід токенізувати під час надсилання запитів. Йдеться про базу даних PostgreSQL, і про саме моделювання. В роботі використано безкоштовний інструмент із відкритим кодом TerraER [2].

Для стилізації коду та акуратного перегляду на екранах різних розмірів використано Bootstrap і CSS3.

Залежності, які надходять із зовнішніх бібліотек, визначені в pom.xml, який постачається з Maven. Генерація документу PDF4, який є підтвердженням успішного платежу, означає, що використовувалася бібліотека lowagie. З цією метою значення криптовалюти беруться із зовнішнього API

РОЗДІЛ 2. АРХІТЕКТУРА

2.1. База даних

Реляційна діаграма бази даних складається з двох частин: перша частина складається з таблиць, які пов'язані одна з одною та мають сильну залежність між ними, а інша частина являє собою окрему таблицю, яка має вказівку для зберігання ціни криптовалюти. Перша частина бази даних складається з чотирьох таблиць. Перша таблиця така «Available_app_crypto» містить три стовпці, унікальний ідентифікатор, який є основним ключем для цієї таблиці, назва криптовалюти та відповідне значення холдингу однакові. Усі доступні в додатку криптовалюти зберігаються в цій таблиці. Для покупок користувачів, тобто якщо користувач хоче купити деяку валюту, в цьому таблиці виробляються прокери, що дуже з цієї площі він володіє цією валютою і може її продати. Ця таблиця відноситься до 1-1. Друга таблиця «Transaction» складається з рядків, які необхідні для збереження всіх успішно здійснених транзакцій у вкладеннях. Містить унікальний ідентифікатор, який вважається первинним ключем цієї таблиці, параметр для визначення чи продає, чи купує користувач валюту, назву валюти, час виконання транзакцій, вартість валюти в доларах США та вартість у криптовалютах за поточною ціною, за кожною валютою. За тією логікою, що один користувач може мати більше транзакцій на відміну від іншого, а з іншого боку, транзакція є унікальною і може бути здійснена лише однією особливим обліковим записом користувача, ми створюємо до зв'язок 1-N між таблицями «Transaction» і «User». Таблиця користувача містить найбільшу кількість рядків. І знову у нас унікальний ідентифікатор виконує роль первинного ключа, імена та прізвища користувача, ім'я користувача, хешований пароль, наявні доступні ресурси в доларах США, за які можна купити криптовалюту з застосунком, кредитна картка для того, щоб за бажанням користувача зняти гроші, вони

надсилаються на відповідний рахунок, а також чотири параметри, важливі з точки зору безпеки, тобто перевірки, чи обліковий запис користувача дійсний.

Також підходимо до останньої таблиці першої частини «Crypto_in_wallet», призначенням якої є збереження усіх криптовалют, що належать користувачам додатку із зазначенням того, чи мають вони адміністративні чи звичайні привілеї.

З точки зору рядків таблиці маємо наступне: ідентифікатор, який є первинним ключем, ім'я валюта, якою володіє користувач і в якій кількості вона у нього є. Крім того, як ми бачимо із зображення нижче, таблиця «User» має рв'язок 1-N із таблицею "Crypto_in_wallet", що означає, що користувачі мають можливість торгувати та володіти кількома криптовалютами одночасно.

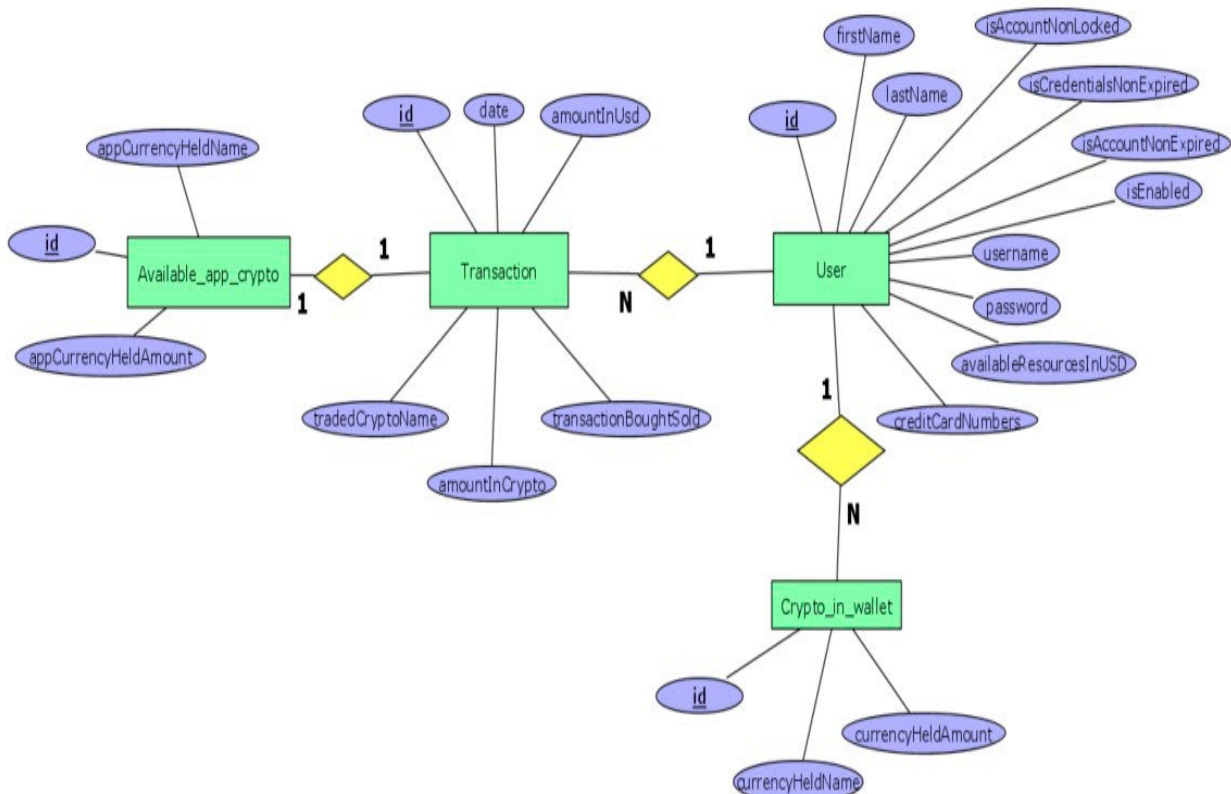


Рис. 1. База даних – перша частина

У другій частині бази даних використовується одна таблиця "CryptoHistoryGraphData" яка містить дані про сто криптовалют, що надходять із зовнішнього джерела. Ось ця таблиця призначена для збору даних, необхідних

для відображення графік історії криптовалюти, що відображається в інтерфейсі користувача.

Таблиця складається з шести рядків, ідентифікатор, який повністю є первинним ключем назва валют, час додавання стовпців до бази даних, ціна за якою мати на той момент символи, за якими стоять криптовалюти та ринок вартість валют.

Реляційну діаграму цієї таблиці можна побачити на зображенні нижче.

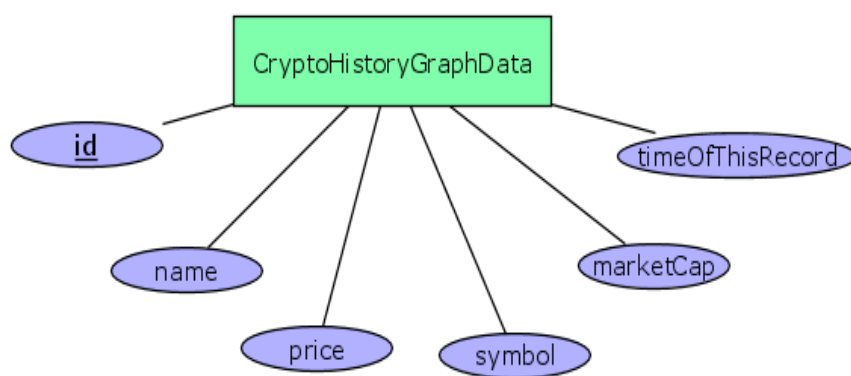


Рис. 2. База даних – друга частина

2.2. Сервер

Це частина програми, яку користувач безпосередньо не бачить, але відіграє ключову роль для його функціональності. Додаток побудовано за багаторівневою архітектурою, що складається з семи пакетів, в яких організовано весь код серверної частини.

Композиція архітектури базується на чотирьох пакетах: модель, репозиторій, логіку служби та контролери або веб-рівень, а також три інші пакети, які несуть що з самі окремі логіки. Такі поділки зроблені для кращої організації коду.

2.2.1. Пакет моделей

Як випливає з назви пакету, ось усі моделі, які були описано в розділі бази даних, і вся логіка, безпосередньо пов'язана з цим їх.

У кожен модель включено такі анотації: Spring Boot, @Getter, @Setter, @AllArgsConstructor – конструктор, який містить усі атрибути; @NoArgsConstructor – конструктор, який не містить жодних атрибутів; @Entity which вказує класи, що представляють таблиці в базі даних, а також анотацію @Table, яка явно вказує назву класів, оскільки вони будуть відображені в базі даних даних. Значення можна додати до всіх цих полів за допомогою Set функцій, які вмикаються через анотацію @Setter, а також отримати доступ за допомогою функцій Get, які ввімкнено через анотацію @Getter.

Спільним елементом для всіх класів є вище згаданий ідентифікатор, який є первинним ключем для кожного з них. Ідентифікатор має дві анотації @Id, яка чітко вказує, що цей атрибут є первинним ключем у кожній з таблиць відповідно, та @GeneratedValue, яка призначає унікальне значення ідентифікатора кожного разу, коли потрібно створити нову чергу.

У середині пакета Model розроблений пакет "перерахування", який складається з одного переліку «Role», який реалізує інтерфейс «GrantedAuthority» і має два поля, які допомагають виділити адміністративні та звичайні привілеї користувача.

У той же час пакет для передачі даних між бекендом і інтерфейсом під назвою dto. Цей пакет містить п'ять класів, кожен з яких містить один або кілька атрибутів, які отримують контролери для успішного встановлення пересилання даних.

1. Клас User

Цей клас відображається в базі даних під назвою **"users"**. Крім уже згаданих і описаних речей, ця таблиця має ще кілька своїх унікальних властивостей.

Для всіх користувачів, які будуть додані до бази даних, стовпець «username» повинен бути обов'язковим має бути унікальним, інакше буде відображено повідомлення про те, що вже існує користувач з таким іменем.

Клас «User» має один атрибут із переліку ролей. Крім того, у цьому класі завдається список транзакцій, які зробили користувачі. Це стало можливим завдяки збереженню списку об'єктів, звідки походять клас Transaction і зіставлення з анотацією @OneToMany.

Також у цьому класі створюється інше зіставлення @OneToMany для доступу до всіх криптовалют, якими володіють користувачі. Це реалізовано шляхом збереження списку об'єктів класу CryptoInWallet.

2. Доступний клас AppCrypto

Як випливає з назви, цей клас складається з трьох атрибутів, які містити всі криптовалюти, доступні в додатку для продажу. Ось цей клас відображається в базі даних під назвою "available_app_crypto".

3. Клас Transaction

У класі транзакцій, на додаток до вже описаних черг з бази даних даних, також реалізовано два додаткові об'єкти для забезпечення бажаного порядку функціонування додатку.

Один з об'єктів належить до класу User, що зберігає користувача, який здійснив цю операцію. При цьому цей об'єкт зіставляється з анотацією @ManyToOne, а також має анотацію @JsonIgnore.

Інший об'єкт, який є частиною цього класу, є екземпляром класу AvailableAppCrypto який відповідним чином відображається за допомогою анотації @OneToOne і містить дані для криптовалюти, якою торгують у даній транзакції.

4. Клас CryptoInWallet

Клас CryptoInWallet містить усі атрибути бази даних плюс додатково відображений об'єкт @ManyToOne класу User, який позначає баланс криптовалют, якими володіють користувачі. Він також має лише один визначений конструктор із двома параметрами, назвою криптовалюти та вартістю цієї валюти у даного користувача.

5. Клас CryptoHistoryGraphData

На відміну від інших класів Java, це незалежний клас відносно решти в цьому архітектурному шарі. Що стосується атрибутів, то він уже містить описані на реляційній діаграмі атрибути відповідно з уже згаданими анотаціями загальними для всіх класів і анотації первинного ключа.

2.2.2. Пакет репозиторій

Пакет є частиною рівня архітектури, яка відповідає за підключення Spring Boot додатку бази даних. Він складається з п'яти інтерфейсів, кожен із яких призначається одному об'єкту в базі даних. При цьому кожен з них створює розширення інтерфейсу JpaRepository, яке вже надає деякі основні функції пошуку в базі даних.

Іменування всіх інтерфейсів цього шару є іменем класу попереднього шару з розширенням «Repository».

1. UserRepository

На додаток до вже визначених функцій, які походять від JPA додано іншу функцію, яка шукає користувача за його іменем який передається як параметр функції.

2. AvailableAppCryptoRepository

Також це сховище має лише одну визначену функцію *party*, яка повертає об'єкт класу *AvailableAppCrypto*. Ідея цієї функції полягає у поверненні об'єкта, який відшукується за назвою криптовалюти, яка є передається як параметр.

3. TransactionRepository

На відміну від попереднього, тут ми маємо дві функції, які повертають список транзакції, об'єкти класу *Transaction*.

Одна з функцій має завдання повернути всі виконані в додатку транзакції в порядку спадання, або, іншими словами, це транзакція, яка виконана останньою, буде показана першою.

Інша функція повертає всі транзакції, здійснені для даного користувача-одержувача як параметр функції та додатково сортує їх за датою занесення в базу в порядку спадання.

4. CryptoInWalletRepository

Цей репозиторій не містить жодних додаткових функцій реалізованих, ті, що вже визначені JPA, достатні для того, що потрібно на цю програму.

5. CryptoHistoryGraphDataRepository

Репозиторій, що складається з двох визначених функцій.

Одна функція повертає відсортований в порядку спадання список об'єктів класу *CryptoHistoryGraphData*. Ця функція служить для повернення JSON5 об'єкт до інтерфейсу з даними, необхідними для візуалізації.

Друга функція повертає символи всіх криптовалют, з якими що ця програма працює. Це робиться за допомогою *nativeQuery* з *@Query* анотацією, що містить запитання PostgreSQL.

2.2.3. Пакет послуг

У пакеті реалізовано бізнес-логіку вище зазначених процесів.

Додаток містить чотири інтерфейси з уже визначеними іменами функції зі своїми параметрами, які успадковують від них відповідні класи. При цьому для кращої організації коду визначена реалізація пакета, з якою ці класи об'єднані з інтерфейсами.

1. UserServiceImplementation

Клас UserServiceImplementation відіграє ключову роль і містить найбільше бізнес-логіки у додатку.

У верхній частині класу приватні та кінцеві змінні визначаються з репозиторію попереднього архітектурного шару. Ця концепція включення попереднього шару в наступний рівень або рівень того самого рівня називається ін'єкція залежностей.

Застосування цієї концепції до цього класу дозволяє створити конструктор для ініціалізація всіх об'єктів з попередніх шарів. У цьому випадку ці об'єкти відносяться до класів:

UserRepository,
 PasswordEncoder,
 CryptoInWalletRepository,
 AvailableAppCryptoRepository, TransactionRepository.

Цей клас складається з восьми функцій.

Register

Ця функція створює та реєструє нового вхідного користувача в програмі. Функція дозволяє отримати параметри, які є частиною класу User, відповідно перевіряє чи поля імені користувача та пароля заповнені правильно, перевіряє пароль та повторний пароль, хешують і зберігають пароль користувача, якщо всі

зазначені раніше перевірки пройшли успішно. В іншому випадку створюються відповідні виключення.

LoadUserByUsername

Ця функція досить проста, вона викликає функцію пошуку користувача за іменем користувача зі схо вища `userRepository`.

AddUSDMoneyInWallet

Це функція отримує параметр, тобто депозит, щой дозволяє користувачеві додавати нові доступні активи, з якими можна продовжувати торгувати. Функція отримує поточного користувача через клас `SecurityContextHolder`, приймає вже наявні ресурси користувача і просто додає депозит.

buyCrypto

Одна з найскладніших функцій.

Вона приймає два параметри: назву валюти, яку користувач хоче купити, і її кількість. Вартість валюти, яку бажає придбати користувач, вказана в доларах США.

Спочатку проводиться перевірка, чи доступний додаток до продажу криптовалюта в заданому значенні, інакше створюється виключення `NotEnoughAppResourcesException` із повідомленням «На жаль, у нас для цього недостатньо валюти!».

Наступна перевірка стосується кількості доступних торгових коштів користувача при поточному вході та перевіряється його статус. При наявності коштів менше, ніж запитані кошти на закупівлю, створюється виключення `NotEnoughUserResourcesException` із повідомленням «Вибачте, вам для цього недостатньо валюти!».

Щоб зробити точний розрахунок відповідно до поточної вартості валюти, яку користувач хоче придбати, здійснюється виклик функції `getCurrencyRealTimePrice`.

Наступна частина логіки цієї функції — перевірити, чи є у користувача у володінні ця валюта, чи вона доступна. Придбані активи додаються до вже

наявних на акаунті користувача. Але якщо він купує цю валюту вперше, то додається новий стовпець до бази даних.

Наступне, що відбувається, це зменшення рахунку вільних коштів на користувача і кількість доступної для продажу валюти зменшується зі сторони додатку.

Створюється нова транзакція, яка фіксує факт здійснення покупки валюта і, нарешті, записи заносяться в бази даних у всіх суб'єктах, які мають що прямий вплив на розрахунки вартості.

SellCrypto

Як і попередня функція, ця функція також отримує два параметри, назву валюти, яку користувач хоче продати, і кількість, яку він хоче продати.

Перевіряється наявність у користувача цієї валюти у сумі, що він хоче продати. Якщо ні, створюється виключення `NotEnoughUserResourcesException` із повідомленням «У вас недостатньо криптовалюти на продаж!»

Наступна дія, яка виконується, це додавання доступних коштів на стороні користувача в доларах США, а також додавання проданої криптовалюти на стороні додатку.

Крім того, тут створюється нова транзакція про те, що було здійснено продаж криптовалюти користувачем, який увійшов у систему, і все це зберігається в базі даних зміни цінностей.

withdrawAmount

Ця функція отримує параметри, що вказують на вартість грошей, що користувач хоче отримати від програми, а також від самого користувача.

Користувача перевіряють, чи відповідає значення, яке він хоче отримати, вартості наявних активів. Якщо вартість бажаної кількості перевищує вартість наявних активів створюється виключення `NotEnoughUserResourcesException` із повідомленням «На жаль, у вас немає доступних ресурсів», де сума `ToWithdraw` динамічно повертає значення, яке користувач намагається зняти.

Якщо є вільні кошти, вони віднімаються з вже наявних і все фіксувати зміни в базі даних.

GetLoggedUserAvailableResource

За допомогою класу SecurityContextHolder фіксується вхід користувача у програму, а також дані користувача витягуються з бази даних, зокрема доступні кошти, якими володіє користувач володіє, та повертається назад до функції, з якої клас викликаний.

getCurrentRealTimePrice

Ця функція приймає параметр назви криптовалюти та повертає поточний параметр вартості шуканої валюти. Логіка цієї функції така більш детально описана в розділі CronJobServiceImpl.

2. TransactionServiceImplementation

У цьому класі виконується повторна реалізація концепції ін'єкції залежностей від попереднього рівня архітектури шляхом ініціалізації компонентів з інтерфейсів TransactionRepository та UserRepository через конструктор.

Реалізуються дві функції, які повертають список об'єктів класу Transaction. Одна функція повертає всі транзакції, зроблені в додатку за умови, що поточний користувач є адміністратором, інакше повертається порожній список. Друга функція відповідає за повернення всіх транзакцій зроблених користувачем, який увійшов у систему, відсортованих в порядку спадання.

3. CryptoHistoryGraphDataServiceImplementation

Цей клас відповідає за збереження історії криптовалюти. Здійснює виклик кількох функцій із відповідного репозиторію та є одна функція, яка реалізована додатково. Додаткова функція викликає параметри з класу CronJobServiceImpl і відповідає за створення записів у об'єкт crypto_history_graph_data.

4. PDFGeneratorServiceImpl

Метою цього класу є створення PDF-файлу, який буде позначати підтвердження того, що це платіж, здійснений програмою користувачеві, який

бажає зняти гроші з рахунку. При цьому він перевіряє, чи достатньо грошей у користувача для виконання операції. Перевірка виконується викликом функції `removeAmount` для реалізації інтерфейсу `UserService`.

5. CronJobServiceImpl

Клас призначений для включення виклику зовнішнього сервера для отримання даних про криптовалюту через `RestTemplate` [3]. Ім'я розміщується в заголовній частині запиту згідно з документацією `CoinMarketCap` [4] відповідно зі значенням згенерованого ключа. Завдання `Cron` визначається за допомогою анотації `@Scheduled`. Функція виконується кожні п'ять хвилин і здійснює дзвінок на сервер.

Для виконання функцій також повинен бути встановлений певний час. Анотація `@EnableScheduling` [5] у класі `CryptotradingApplication`.

Визначається URL, до якого здійснюються виклики, "`https://pro-api.coinmarketcap.com/v1/cryptocurrency/listings/latest`" і виконується запит на сервер. Коли дані повертаються за допомогою `ObjectMapper`, однозначно виконуються зіставлення значень із класом `APIResponseCryptocurrencies`, який є пояснюється трохи нижче.

Крім того, ці значення зберігаються в суб'єкті, який відповідає за історію `crypto_history_graph_data`.

2.2.4. Пакет контролерів

Пакет, у якому знаходяться контролери, відповідає за обробку вхідні запити REST API, внесення змін до серверної частини або прийняття дані з нього та повернення їх через JSON.

Загалом, перш ніж усі класи контролерів будуть визначені додаток містить деякі необхідні анотації для його роботи у бажаний спосіб.

Перша анотація — `@RestController` [6], яка повідомляє Spring Boot додатку, що цей клас є контролером.

Більшість контролерів мають заздалегідь визначений маршрут, яким ми повинні слідувати. Для організації доступу до ресурсів контролерів у цій програмі використовується анотація `@RequestMapping("/")`.

Для доступу до програми із зовнішніх маршрутів, які не є прямими частиною Spring Boot, використовується анотація `@CrossOrigin(value="**")`, де зірочка означає, що до цього ресурсу можна отримати доступ з усіх зовнішніх маршрутів.

З анотаціями `@CrossOrigin` і `@RequestMapping` є лише невелика різниця у контролерах для входу та реєстрації нових користувачів, і ці відмінності є описані у відповідних розділах нижче в поясненнях.

1. CryptoApiController

Знову ось поява та реалізація концепції ін'єкції залежностей з попереднього архітектурного рівня класу `CryptoHistoryGraphDataService`.

У цьому контролері реалізовано дві функції.

Перша функція бере ресурси з сервера шляхом призначення анотації `@GetMapping` на маршруті `"/crypto"`, який повертає список об'єктів класу `CryptoHistoryGraphData`. Цей список криптовалют, який зараз обробляється, використовується для відображення даних на діаграмі інтерфейсу користувача. В той самий час отримує один параметр через анотацію `@RequestParam`, який є символом, для якого користувач вибрав отримання діаграми цієї криптовалюти. Здійснюється виклик функції класу `CryptoHistoryGraphDataService` у логіці служби, яка повертає список об'єктів, які передаються через Java Stream, який виконує фільтрацію відповідно до символу криптовалюти, яку вибрав користувач, і вводиться обмеження на тринадцять об'єктів, які надсилаються назад. Обмеження у тринадцять об'єктів обумовлено відображенням діаграми за одну годину, тобто в один новий об'єкт кожні п'ять хвилин.

Друге зіставлення також є `@GetMapping` на маршруті `"/cryptoSymbolName"`, здійснює виклик класу `CryptoHistoryGraphDataService` і повертає всі унікальні символи ста криптовалют. Список користувацького

інтерфейсу дозволяє користувачам вибрати символ, яким виконується перенаправлення до маршруту `"/crypto"`, описаного вище.

2. LoginRestController

Це єдиний контролер, який має обмеження доступу до ресурсів від зовнішніх об'єктів. Всередині анотації `@CrossOrigin` визначено маршрути доступу лише для викликів, що надходять з `"http://localhost:3000"` і `"http://localhost:3001"`.

Об'єкт, який походить із класу, ініціалізується через конструктор `JWTAuthenticationFilter`.

Анотація `@RequestMapping` визначена на базовому маршруті `"/api/login"` і таким чином, єдина функція в цьому класі, яка має анотацію `@PostMapping`, не є визначеним маршрутом, але призначена для входу користувача, якщо він відповідає умовам функції, до якої здійснюються виклики, визначені в класі `JWTAuthenticationFilter`.

3. PDFController

Після визначення класу можна побачити, що створюється об'єкт класу `PDFGeneratorServiceImpl`. За допомогою доступу до маршруту `"/generatePDF"`, який є зіставлено за допомогою `@GetMapping`. `headerKey` і `headerValue` встановлюються відповідно параметри, необхідні для створення PDF-файлу, що виконує функція експорту з класу `PDFGeneratorServiceImpl`. Крім того, функція маршруту `"/generatePDF"` отримує три параметри `HttpServletResponse`, сума яких дозволяє вийти за бажанням з додатку. [7]

4. RegisterController

Основним маршрутом для доступу до ресурсів з цього контролера є `"/register"`. Тут виконується ін'єкція залежностей з інтерфейсу `UserService` і має одну функцію `@PostMapping`, для якої немає визначеного маршруту. Тобто доступ до нього здійснюється за допомогою POST виклику на базовий маршрут.

Отримує об'єкт `@RequestBody` класу `RegisterUserDto`, який містить дані необхідні для реєстрації нового користувача. Виклик функції реєстрації здійснюється через об'єкт із `UserService`, де зареєстрований існуючий користувач та реєструється новий користувач в базі даних. Логіка реєстрації пояснюється в розділі конфігураційного пакета де знаходиться все, що пов'язане з безпекою програм.

5. *TransactionController*

Цей контролер має дві функції, які мають досить схожі завдання.

Обидві функції мають анотацію `@GetMapping`. Перша функція, яку було зіставлено на маршруті `"/transactions"` повертає всі транзакції, які були зроблені в додаток. Для того повернення цих транзакцій користувач повинен мати адміністративні привілеї.

Друга функція полягає в тому, щоб приймати всі транзакції, здійснені за особистим входом користувача. Маршрут цієї функції – `"/loggedUserTransactions"`.

Обидві функції викликають клас `TransactionService`, з якого ми маємо ініціалізований об'єкт у контролері за допомогою ін'єкції залежностей.

6. *UserController a*

Як ми могли бачити, у нас найбільше бізнес-логіки в розділі Клас `UserServiceImplementation`. Тому тут у нас найбільше функцій до яких увімкнено доступ.

На початку класу ми маємо ін'єкцію залежностей з двох архітектурних рівнів `UserService` та `UserRepository`.

Маршрути, зіставлені за допомогою `@PostMapping` у цьому контролері, є `"/addMoney"`, `"/buyCrypto"` та `"/sellCrypto"`. Усі три параметри отримують об'єкти передачі даних через анотацію `@RequestBody`. Всередині цих об'єктів ми маємо значення, які використовуються для відповідних змін в базі даних.

DepositCashDto — це клас, успадкований від функції маршруту /addMoney і відповідно додає доступні кошти на рахунок користувача.

Ця функція здійснює виклик функції addUSDMoneyInWallet з інтерфейсу UserService.

BuyCryptoDto - це клас, який відповідає за прийняття значень, для яких криптовалют і в якій кількості користувач хоче купити, і при цьому здійснює виклик функції buyCrypto з інтерфейсу UserService. Шлях для купівлі криптовалюти зіставлено з "/buyCrypto".

Остання функція зіставлення @PostMapping у цьому класі отримує об'єкт від класу SellCryptoDto, який приймає значення для якої валюти та в якій кількості користувач хоче продати, і при цьому здійснює виклик функції sellCrypto з Інтерфейсу UserService.

Далі йдуть функції цього класу, які мають відображення @GetMapping. Ці функції не приймають жодних параметрів.

Функція з маршрутом "/getLoggedUserCryptocurrencies" повертає всі криптовалюти, які якими володіє користувач, який увійшов у систему. Цей результат виходить шляхом створення виклику функції getCryptoInWallet з інтерфейсу UserRepository.

Маршрут "/loggedUser" також здійснює виклик функції з інтерфейсу UserRepository. За допомогою виклику findByUsername отримуємо поточний вхід користувача звернення до інтерфейсу користувача.

Останнім маршрутом цього класу є "/availableResources", який повертає значення доступних ресурсів від зареєстрованого користувача до інтерфейсу користувача через виклик функції з інтерфейсу UserService.

2.2.5. Пакет аналізу даних

Пакет призначений для відповідного аналізу та отримання даних з об'єктів JSON, що надходять під час виклику маршруту "https://proapi.coinmarketcap.com/v1/cryptocurrency/listings/latest», який

використовується для отримання інформації про останні ціни на сто криптовалют в доларах США.

Організація процесів у цьому пакеті здійснюється відповідно до інформації, що надходить з об'єкта JSON, який завантажується зі згаданого API і приймається лише інформація, необхідна для розробки цієї програми.

Під час використання ObjectMapper імена JSON мають збігатися з іменами змінних у класах Java, і тому кілька разів є з'являється анотація `@JsonProperty("")`, яка вказує, що запис в лапках відповідає значенню змінної, визначеної в об'єкті JSON, і дозволяє мати різні назви в класах Java для кращої організації коду.

За класом `APIResponseCryptocurrencies` ховається вся структура даних, які ми маємо в об'єкті JSON. Всередині визначається список об'єктів класу `Cryptocurrency`, який містить ідентифікатор, назву криптовалюти, символ і об'єкт класу `Quote`. Клас `Quote` має лише один об'єкт класу `USD`, який містить інформацію про ціну та ринкову вартість криптовалюти.

2.2.6. Пакет конфігурації

Це пакет, який відповідає за безпеку програми. Вся логіка служби, пов'язана з входом і реєстрацією як нових, так і раніше наявних користувачів, є частиною класів, які можна знайти в цьому пакеті.

Крім того, всередині пакета конфігурації є інший пакет, де все є попередньо визначені фільтри для автентифікації та авторизації JWT.

1. *WebSecurityConfig*

Цей клас містить оголошення двох змінних класу `PasswordEncoder` і класу `CustumUsernamePasswordAuthenticationProvider`, а також їх ініціалізацію через конструктор.

Є два маркери, які мають однакові назви під назвою `configure`.

Перший метод отримує об'єкт класу `HttpSecurity` і в ньому визначається які маршрути вважаються основними та доступними для всіх користувачів програми, для яких від вони вимагаються наявність користувачів адміністративних привілеїв користувача, а також видалення файлів `cookie`, коли користувач хоче вийти з програми.

Інша функція отримує об'єкт класу `AuthenticationManagetBuilder` і викликає його функція аутентифікації `Provider`, яка приймає об'єкт класу `CustomUsernamePasswordAuthenticationProvider`.

2. JWTWebSecurityConfig

Логіка реалізації подібна до логіки класу `WebSecurityConfig`, але тут використовуються визначені фільтри автентифікації та авторизації користувачів. Ще одна відмінність полягає в тому, що була введена нова логіка, яка не зберігає стан сесії. Коли є зв'язок від зовнішніх служб із `Spring Boot` програма завжди повинна пересилати токен `JWT` для автентифікації користувач на стороні сервера, тим самим повідомляючи серверу, що це дійсний запит `HTTP`.

Токен `JWT` завжди має бути закодований під час пересилання, оскільки додаток цього не робить.

3. JWTAuthConstants

Для кращої організації коду створено клас, який містить `final` визначені змінні, які не змінюватимуться та необхідні для `JWT` автентифікації.

4. CustomUsernamePasswordAuthenticationProvider

Клас, що складається з двох функцій.

Перша функція підтримує, яка лише перевіряє, чи параметр отримано від класу `UsernamePasswordAuthenticationToken`.

Клас автентифікації містить більшу частину інформації, пов'язаною із логікою цього класу. Цей метод викликається, коли надсилається запит `POST` до маршруту `"/login"`.

Після того, як користувач введе ім'я користувача та пароль у полі форми інтерфейсу користувача, дані надходять як параметр, який що отримує цю функцію. Перевіряється правильність заповнення цих полів, чи однакові хешовані значення паролів користувача, який пароль походить від форми з інтерфейсу користувача, що знаходиться в базі даних. Якщо всі перевірки пройдуть успішно, користувач отримає сформований токен JWT, який буде використовуватися деякий час протягом виконання дій у програмі, інакше буде створено виключення із повідомленням «Недійсні облікові дані користувача».

Програма має використовувати токен JWT для зв'язку в кожному запиті. Через форму з інтерфейсу користувача надсилається POST запит на сервер. Якщо сервер виявляє, що користувач дійсний, тобто його облікові дані правильні, тоді створить для нього токен JWT за допомогою секретного коду.

Згенерований токен надсилається назад у браузер, який записує його інформацію у відповідному файлі cookie. Крім того, має бути додана інформація з кожним новим запитом до цього сервера, що служить для автентифікації користувача.

Формат заголовка авторизації такий: Authorization: Bearer <token>.

Сервер зчитує інформацію із заголовка авторизації, переглядає підпис JWT і якщо вони дійсні, він продовжує обробку запиту від користувача.

2.2.7. Пакет виключень

Пакет Exceptions package створений для більш точного відображення помилок. Всього створено шість різних типів виключень, які знадобилися під час розробки програми. Усі класи успадковують з класу Exception і надіслати повідомлення іншого типу в конструкторі.

Типи виключень:

- i. InvalidCryptocurrencySearchException
- ii. InvalidUserCredentialsException
- iii. InvalidUserPasswordsException

- iv. `NotEnoughAppResourcesException`
- v. `NotEnoughUserResourcesException`
- vi. Виключення `UserAlreadyExists`

2.3. Інтерфейс

Для створення користувацьких інтерфейсів я використовував React, JavaScript бібліотека.

Код, набраний для потреб програми, знаходиться в пакеті `"/src"`, який містить масив пакетів для кращої організації коду.

Базовий інтерфейсний маршрут програми – `«http://localhost:3000/»`.

Додаток складається з кількох переглядів:

2.3.1. Перегляд реєстрації нового користувача - `/register` або `"/`

Під час доступу до самої програми цей перегляд є основним, тобто першим яку побачить кожен користувач.

Після перефарбування цього перегляду можна побачити форму, на якій розміщені нові користувачі, які хочуть використовувати цю програму, можуть створити облікові записи користувачів. При цьому, на цю форму необхідно внести дані про їх ім'я користувача, пароль, повторний пароль, їх ім'я та прізвище, дані кредитної картки та, нарешті, чи матиме користувач регулярні або адміністративні привілеї. Якщо будь яке з цих полів не заповнено правильно або не заповнено взагалі, з'явиться повідомлення про помилку спроби створити обліковий запис.

Крім того, внизу, після форми, є посилання на вікно входу для користувачів, які вже мають свій обліковий запис. Можна натиснути на це посилання, яке буде переспрямовувати на маршрут `"/login"`.



Register Form

Username

Password

Repeat Password

Firstname

Lastname

Credit Card

Role

Have account? [Sign in here](#)

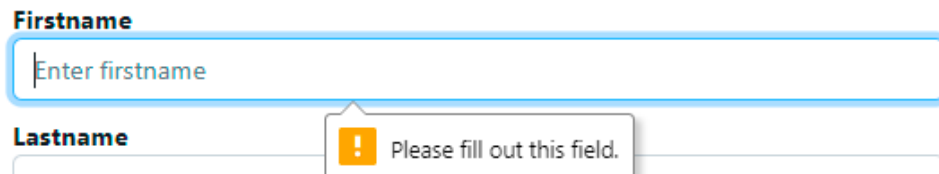
Рис. 3. Заповнена форма реєстрації нового користувача

Role

Bad Request - Check the credentials, otherwise the username is already taken!

Have account? [Sign in here](#)

Рис. 4. Повідомлення при неправильному заповненні полів реєстраційної форми



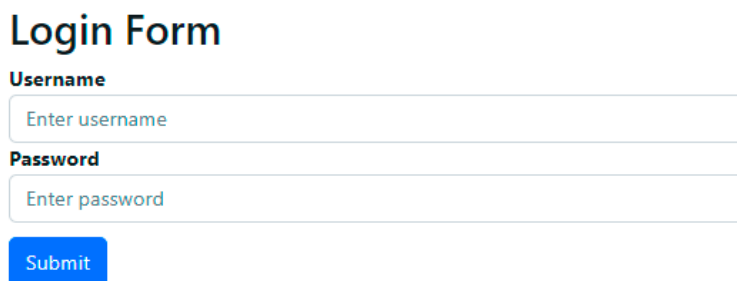
The image shows a registration form with two input fields: 'Firstname' and 'Lastname'. The 'Firstname' field is highlighted with a blue border and contains the placeholder text 'Enter firstname'. The 'Lastname' field is empty and has a validation error message 'Please fill out this field.' displayed next to it, indicated by an orange exclamation mark icon.

Рис.5. Перевірка полів реєстрації нових користувачів

2.3.2. Перегляд входу вже існуючих користувачів - /login

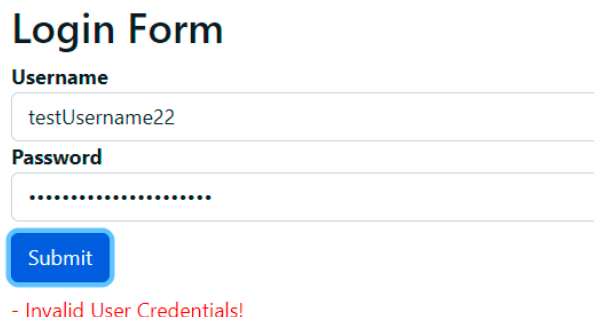
Якщо користувачі вже мають облікові дані для входу в програму вони можуть використовувати їх у такому вигляді. Все, що вони повинні зробити користувачі - заповнити форму, яка складається з двох полів, ім'я користувача і пароль.

Якщо будь-яке з полів заповнено неправильно або не заповнено взагалі на екрані користувача формується повідомлення про помилку. Якщо все в порядку повинно бути переспрямування на маршрут "/home".



The image shows a login form titled 'Login Form'. It contains two input fields: 'Username' with the placeholder 'Enter username' and 'Password' with the placeholder 'Enter password'. Below the fields is a blue 'Submit' button.

Рис. 6. Форма входу користувача програми



The image shows the same login form as in Figure 6, but with the 'Username' field containing the text 'testUsername22' and the 'Password' field filled with dots. Below the 'Submit' button, there is a red error message: '- Invalid User Credentials!'.

Рис. 7. Повідомлення при неправильному заповненні полів форми входу

2.3.3. Перегляд домашньої сторінки – "/home"

У лівій частині екрана розміщується діаграма з історичними цінами базової криптовалюти Bitcoin до USD за останню годину з інтервалом кожні п'ять хвилин.

Трохи вище графіка є список із ста криптовалют, якими користуються користувачі вони можуть торгувати. У будь-який момент вони можуть вибрати нову криптовалюту з для відповідної валюти буде відображено список і графік історії.

Значення осі Y відображаються динамічно для більш чіткого перегляду графік.

Під графіком ви розміщені дані для вибраної криптовалюти, його символ і ринкову вартість. При цьому з кожним новим виділенням нова криптовалюта на графіку, ці дані оновлюються негайно.

Праворуч від графіка розміщено розділ, де ви можете щось робити операції. Цей розділ складається з двох форм, які служать для покупки та продаж криптовалюти. Обидві форми приховані від імені криптовалюти с які користувачі хочуть здійснити покупку чи продаж, а також вартість які вони хочуть, щоб торгівля здійснювалася в доларах США. Якщо користувачі не мають недостатньо коштів або інша перевірка серверної частини не дозволила завершити транзакції, в інтерфейсі користувача буде відображено відповідне повідомлення про помилку.

2.3.4. Перегляд для додавання та виведення коштів із програми - "/баланс"

За допомогою цього перегляду користувачі можуть додавати доступні коштів на свої рахунки, а також зняття коштів із програми. Це може щоб зробити це за допомогою двох різних форм, які мають лише одне поле сума в доларів США, за які вони хочуть виконати відповідну дію.



Рис. 8. Домашня сторінка

Якщо користувачі не задовольняють перевірці серверної частини зняття бажаної суми або додавання нових доступних коштів для подальшої торгівлі буде виведена відповідна помилка користувача інтерфейс.

При цьому, якщо користувачі знімуть гроші з програми, вони будуть їхніми сформований рахунок-фактура, який є підтвердженням того, що гроші були сплачені. Ось цей дія виконується через перегляд документа PDF.

2.3.5. Перегляд, щоб відобразити власні криптовалюти зареєстрованого користувача - "/myCrypto"

Запис результатів успішних покупок криптовалюти, а також Відрахування з уже наявних можна відслідковувати через попереднє замовлення цей погляд.

У верхній частині екрана ви відображено який користувач увійшов і скільки доступних коштів, якими він володіє. Трохи нижче ви наведено таблицю з двома стовпцями, назва криптовалюта та сума, яку користувач має відповідно в криптовалюти.



Рис. 9. Вибір іншої криптовалюти для відображення діаграми історії

The screenshot shows the BLOCK TRADER interface. At the top, there's a navigation bar with links: My Transactions, All Transactions, and a Log out button. Below the navigation bar, there's a 'Make Transaction:' section. The first form is for buying crypto, with fields for 'Cryptocurrency Name' (set to 'Bitcoin') and 'Amount(USD)' (set to '500'), and a 'Buy Crypto' button. The second form is for selling crypto, with fields for 'Cryptocurrency Name' and 'Amount(USD)', and a 'Sell Crypto' button. To the right of the forms, there's a red error message: 'Something went wrong, the transaction is not completed: Internal Server Error'.

Make Transaction:

Cryptocurrency Name:

Amount(USD):

Buy Crypto

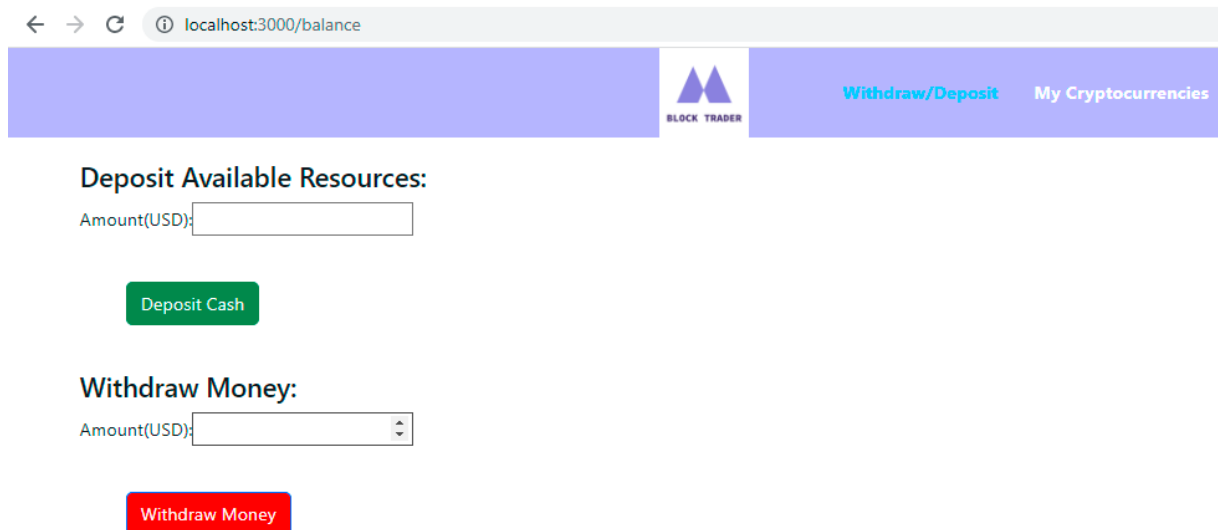
Cryptocurrency Name:

Amount(USD):

Sell Crypto

Something went wrong, the transaction is not completed:
Internal Server Error

Рис. 10. Повідомлення про невдачу транзакції під час купівлі криптовалюти



← → ↻ ⓘ localhost:3000/balance

Deposit Available Resources:

Amount(USD):

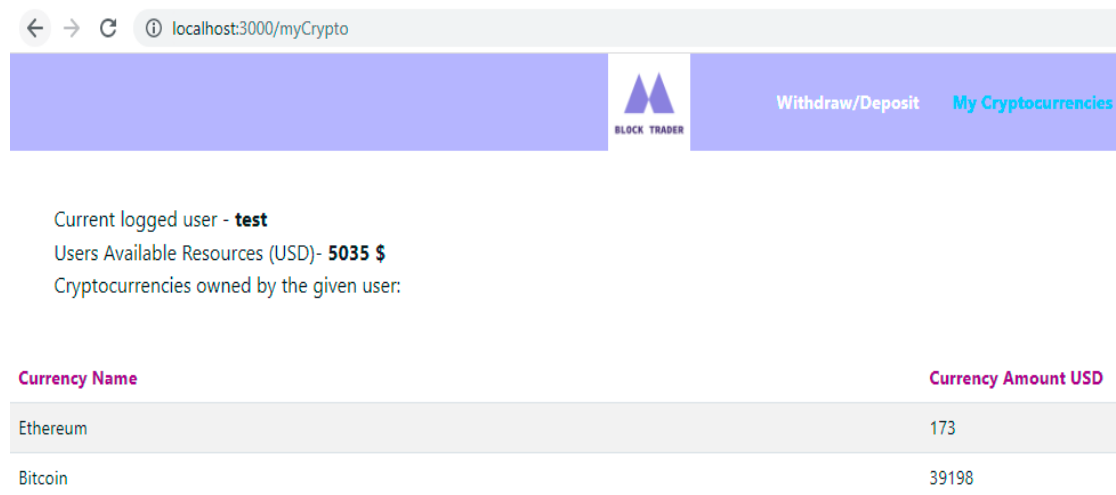
Deposit Cash

Withdraw Money:

Amount(USD):

Withdraw Money

Рис. 11. Перегляд для додавання та вилучення доступних активів



← → ↻ ⓘ localhost:3000/myCrypto

Current logged user - test

Users Available Resources (USD)- 5035 \$

Cryptocurrencies owned by the given user:

| Currency Name | Currency Amount USD |
|---------------|---------------------|
| Ethereum | 173 |
| Bitcoin | 39198 |

Рис. 12. Перегляд для відображення всіх криптовалют, якими володіє користувач, який наразі увійшов у систему

2.3.6. Перегляд для відображення всіх транзакцій, здійснених користувачем, який увійшов у систему – "/myTransactions"

Усі покупки та продажі криптовалют користувачем, який увійшов у систему, можна побачити в цьому поданні, яке надає дані про них у вигляді таблиці.

Таблиця складається з п'яти стовпців, назва криптовалюти, до складу якої входить торгівля, сума в криптовалюті, сума валюти в доларах США, тип транзакції чи валюта була куплена або продана та час завершення цієї операції.

Якщо авторизований користувач ще не здійснив жодної транзакції, на екрані з'явиться повідомлення «Вибачте, транзакцій для показу немає!».



| Traded Crypto | Crypto Amount | Crypto USD | Transaction Type | Date |
|---------------|------------------------|------------|------------------|----------------------------|
| Bitcoin | 0.3642488072729707 | 7000 | Bought | 2024-05-08T14:42:49.817934 |
| Bitcoin | 0.0861209417634373 | 2000 | Bought | 2024-04-06T15:24:01.493496 |
| Bitcoin | 0.008022725702336244 | 190 | Bought | 2024-04-31T10:37:04.322806 |
| Bitcoin | 0.008022725702336244 | 190 | Sold | 2024-04-31T10:36:26.487332 |
| Bitcoin | 0.0008453464698310848 | 20 | Bought | 2024-04-31T10:19:20.111614 |
| Bitcoin | 0.0005045753416305411 | 12 | Sold | 2024-04-29T19:22:07.154584 |
| Bitcoin | 0.0008409589027175685 | 20 | Bought | 2024-04-29T19:22:01.276768 |
| Bitcoin | 0.41935293570073157 | 10000 | Bought | 2024-04-28T22:19:09.750384 |
| Bitcoin | 0.00008396693400006089 | 2 | Sold | 2024-04-28T20:21:22.538572 |

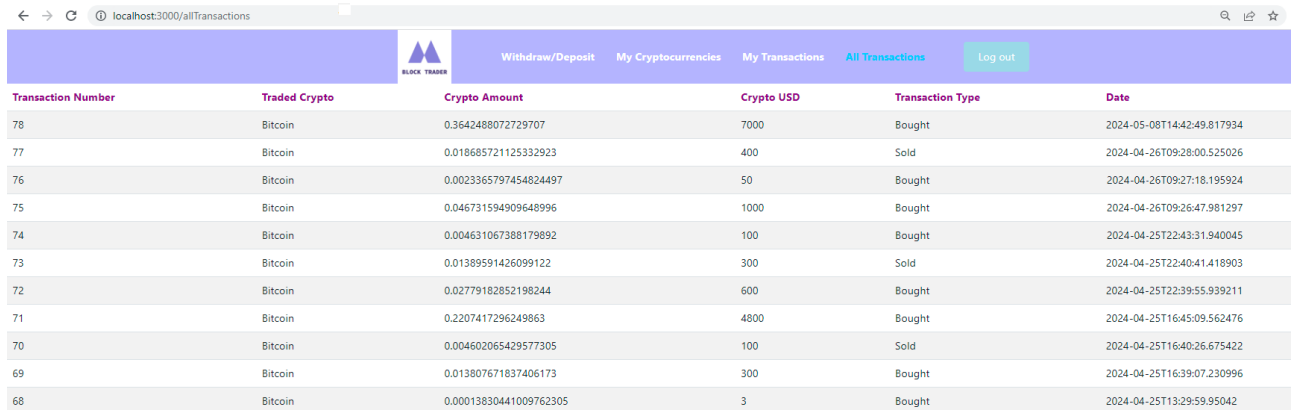
Рис. 13. Перегляд для відображення всіх транзакцій, здійснених користувачем, який наразі увійшов у систему

2.3.7. Перегляд, щоб показати всі транзакції, зроблені всіма користувачами в програмі - `"/allTransactions"`

Доступ до цього перегляду дозволено лише користувачам із правами адміністратора в додатку. Він показує всі транзакції, які були здійснені у додатку всіма користувачами. Також тут дані відображаються через табличне подання, яке містить ті самі стовпці, що й подання `"/myTransactions"` із додатковим стовпцем на початку таблиці, який вказує номер операції.

Якщо в додатку не було здійснено транзакцій або якщо користувач немає прав адміністратора, на екрані з'явиться повідомлення «На жаль, зараз у вас немає дозволу на цю дію або немає транзакцій!».

Усі встановлені можливі маршрути, не згадані раніше переспрямовувати на `"/PageNotFound"`, який показує, що цей маршрут не існує користувач може повернутися до `"/register"` за посиланням



| Transaction Number | Traded Crypto | Crypto Amount | Crypto USD | Transaction Type | Date |
|--------------------|---------------|------------------------|------------|------------------|----------------------------|
| 78 | Bitcoin | 0.3642488072729707 | 7000 | Bought | 2024-05-08T14:42:49.817934 |
| 77 | Bitcoin | 0.018685721125332923 | 400 | Sold | 2024-04-26T09:28:00.525026 |
| 76 | Bitcoin | 0.0023365797454824497 | 50 | Bought | 2024-04-26T09:27:18.195924 |
| 75 | Bitcoin | 0.046731594909648996 | 1000 | Bought | 2024-04-26T09:26:47.981297 |
| 74 | Bitcoin | 0.004631067388179892 | 100 | Bought | 2024-04-25T22:43:31.940045 |
| 73 | Bitcoin | 0.01389591426099122 | 300 | Sold | 2024-04-25T22:40:41.418903 |
| 72 | Bitcoin | 0.02779182852198244 | 600 | Bought | 2024-04-25T22:39:55.939211 |
| 71 | Bitcoin | 0.2207417296249863 | 4800 | Bought | 2024-04-25T16:45:09.562476 |
| 70 | Bitcoin | 0.004602065429577305 | 100 | Sold | 2024-04-25T16:40:26.675422 |
| 69 | Bitcoin | 0.013807671837406173 | 300 | Bought | 2024-04-25T16:39:07.230996 |
| 68 | Bitcoin | 0.00013830441009762305 | 3 | Bought | 2024-04-25T13:29:59.95042 |

Рис. 14. Перегляд для відображення всіх транзакцій, здійснених усіма користувачами

У верхньому правому куті панелі навігації створено кнопку «exit», яка виконує дії переспрямування на сторінку `"/login"`, де той самий або інший користувач може увійти на сторони. У той же час, відтворюючи перегляд `"/login"`, JWT також видаляється токен користувача, який увійшов у систему з браузер.

Усі згадані раніше маршрути визначені в компоненті `App.js` які доступні в додатку шляхом реалізації функцій з бібліотека `react-router-dom`. Водночас тут вказано, які компоненти будуть відображається під час доступу до будь-якого з маршрутів.

Крім того, для реалізації цих представлень код організовано в кілька пакетів:

Пакет компонентів

Більшість інтерфейсної логіки міститься в цьому пакеті додатку. Визначено дванадцять компонентів, які впорядковані відобразити лише за потреби під час доступу до відповідного перегляду інтерфейс користувача.

Визначено такі компоненти `AddMoneyInWallet`,

`AllTransactions`, `Balance`, `BuyCrypto`, `CashChanges`, `Graph`, `HomePage`, `MyCrypto`, `MyTransactions`, `NavBarm NavBarElements`, `PageNotFound`, `SellCrypto`

та WithdrawMoney. Крім того, усі вони мають розширення .js відповідно після назви.

Bootstrap [8] використовується для покращення зовнішнього вигляду екрана користувача дозволяє програмі добре виглядати на екранах різних розмірів.

Пакет custom-axios

Створення нового екземпляра для доступу до маршрутів до серверної частини бібліотека "axios". У цьому випадку базова URL-адреса визначається шляхом додавання необхідні функції запитів, що досягають серверної частини як маркер JWT, контроль доступу та тип очікуваної відповіді.

Пакет зображень

Містить зображення, яке є логотипом програми.

Пакет входу

Цей пакет містить один компонент, який є новою формою входу користувача.

Коли здійснюється доступ до компонента, токен JWT видаляється, якщо він уже існує, розміщується сховище браузера. Він використовує відповідні стани для імені користувача та пароля, а також коли натиснуто кнопку увійти ті самі під потоки пересилаються на серверну частину через пакет сховища.

Реєстраційний пакет

Реєстраційна форма має свій пакет, в якому визначаються штати з полів форми. Коли користувач правильно заповнює поля у формі відповідний запит HTTP надсилається до серверної частини.

Крім того, тут визначається маршрут переходу до форми "/login", якщо користувач уже має обліковий запис користувача або запит для реєстрація нового користувача.

Пакет репозиторію

Цей пакет має один компонент, який використовує вже попередньо визначений маршрут axios з пакету "custom-axios". Крім того, всі параметри

визначені тут, тип запитів і маршрутів, з яких можна отримати доступ під час надсилання даних фронтенд до серверного і навпаки.

Тут ви можете побачити всі параметри, яким перешкоджають форми з інтерфейс та імена, під якими вони визначені для прийняття серверна частина.

Пакет стилів

Щоб додати стилі до програми, окрім використання Bootstrap, також додається файл `style.css`.

ВИСНОВКИ

В рамках цієї дипломної роботи було успішно розроблено веб-додаток для торгівлі криптовалютою. Ця програма дозволяє користувачам купувати та продавати сотні криптовалют, отриманих із зовнішнього API.

Цей тип програми має великий потенціал для оновлення і успішного використання у майбутньому оскільки цифрові валюти стають все більш прийнятними в багатьох країнах світу.

В роботі представлено моделювання баз даних, розробка веб-додатків, дизайну інтерфейсу користувача, управління проектами через версії вихідного коду тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. JWT - <https://www.javainuse.com/spring/boot-jwt>
2. GitHub Project Repository - <https://github.com/gabrieldim/Crypto-Trading-Graduate-Thesis>
3. RestTemplate - <https://www.baeldung.com/rest-template>
4. CoinMarketCap - <https://coinmarketcap.com/api/documentation/v1/>
5. Spring Boot Scheduling - <https://www.baeldung.com/spring-scheduled-tasks>
6. @RestController - <https://www.baeldung.com/spring-controller-vs-restcontroller>
7. PDF Generator Tutorial - <https://www.youtube.com/watch?v=S7udzd3xjGQ&t=33s>