Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Калюги Олексій Романовича*
(ПІБ)

академічної групи *122-18-3*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка сервісу для моніторингу та проведення операції на ринку цінних паперів та криптовалют на основі Java/Spring*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц Реута О.В.*			
розділів:				
спеціальний	*доц Реута О.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-3* *Калюга О.Р.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка сервісу для моніторингу та проведення операції на ринку цінних паперів та криптовалют на основі Java/Spring*

затверджена наказом ректора НТУ «ДП» від *18.05.2022* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Реута О.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Калюга О.Р.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*
Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 90 с., 34 рис., 3 дод., 22 джерел.

Об'єкт розробки: сервіс для моніторингу та проведення операції на ринку цінних паперів та криптовалют

Мета кваліфікаційної роботи: створення сервісу з використанням мови програмування Java та її фреймворку Spring, який забезпечує належну роботу усіх наведених функції.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточняється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленого додатку, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у проектувані та створені сервісу з використанням сучасних практик розробки, що дозволить отримувати необхідну інформацію про цінні папери та криптовалюти а також проводити операції над ними.

Актуальність розробленого сервісу визначається великим інтересом суспільства до сфери інвестиції, цінних паперів та криптовалют, збереження, примноження або диверсифікації свого капіталу з використанням аналогічних до розробленого веб ресурсів, які доступні через глобальну мережу інтернет.

Список ключових слів: РОЗРОБКА, СЕРВІС, JAVA, SPRING, АКЦІЙНІ ПАПЕРИ, КРИПТОВАЛЮТИ, ОПЕРАЦІЇ.

ABSTRACT

Explanatory note: 90 p., 34 figs., 3 appx., 22 sources.

Object of development: service for monitoring and conducting operations in the securities and cryptocurrency market.

Purpose of the qualification work is creation of a service using the Java programming language and its Spring framework, which ensures the proper operation of all these functions.

In the introduction it is considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the second section it is analyzes the existing solutions, selected platforms for development, designed and developed the program, describes the program, algorithm and structure of its operation, as well as calling and loading the program, determines the input and output data, describes the parameters of hardware.

In the first section the subject branch is analyzed, the urgency of the task and purpose of development are defined, the statement of the task is formulated, requirements to software realization, technologies and software are specified.

In the economic section it is determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical value is to design and create service using modern development practices, which will provide the necessary information about securities and cryptocurrencies and allow to conduct operations on them.

The relevance of the developed service is determined by the great interest of society in the investment, securities and cryptocurrencies, saving, increase or diversification of its capital using similar to developed web resources, which are available through the Internet.

List of keywords: DEVELOPING, SERVICE, JAVA, SPRING, SECURITIES, CRYPTOCURRENCIES, OPERATIONS.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ФР – фондовий ринок;

ФА- фінансові активи;

ЦП- цінні папери;

ФО- фінансові операції;

AWS- Amazon Web Services;

MA- Microsoft Azure;

GCP- Google Cloud Platform;

БД - база даних;

JRE- Java Runtime environment;

JVM- Java Virtual Machine;

IOC – Inversion of Controle;

DI – Dependency Injection;

HTTP – HyperText Transfer Protocol;

MVC- Model View Controller;

AOP – aspect oriented programming;

JPA- Java persistent API;

ORM- Object relation mapping;

JSON- Jayson;

JWT- JSON web token

HTML-HyperText Markup Language;

CSS- Cascade Style Sheets;

СУБД-система управління базою данних;

SQL- Structure query language.

ACID- Atomacity Consistency Isolation Durability

ЗМІСТ

ВСТУП

Темою даної кваліфікаційної роботи є розробка веб сервісу з використанням сучасного стеку технології мови програмування Java та фреймворку Spring, для отримування поточної інформації щодо вартості цінних паперів(ЦП), аналізу історії цих змін та можливістю проведення фінансових операції(ФО) на біржі.

Метою даної кваліфікаційної роботи є вивчення інструментів та засобів розробки на мові програмування Java та фреймворку Spring, дослідження і використання кращих сучасних практик щодо розробки та проектування програмного забезпечення, використання отриманих знань у розробці веб сервісу для роботи у сфері ЦП та криптовалют.

Створений продукт може використовуватися фізичними або юридичними особами з дійсною брокерскою ліцензією для надання послуг у сфері інвестиції через глобальну мережу інтернет.

Усі фінансові активи(ФА) до яких належать криптовалюти та ЦП, належать сфері діяльності фондового ринку(ФР). ФР займається пошуком інвестиції. Інвестиції надходять завдяки продажу ЦП та криптовалют. Покупець ЦП та криптовалют може примножити свій капітал за рахунок росту ціни або отримання доходу від ЦП та криптовалют

ФР з’явився на межі 16 та 17 століття, приблизно у 1500[1]. Перші операції з ЦП відбувалися на оптових ринках і товарних біржах[1]. Батьківщиною біржі вважається бельгійський порт Антверпен, який відіграв значну роль у світовій торгівлі[1]. ФР забезпечував переливання грошових активів у галузі, які були потрібні суспільству[1].

На кінці 20 століття популярність ФР, кількість його користувачів, цінність ЦП та інтенсивність торгів різко збільшились. Це сталося завдяки багатьом факторам – розвитку інформаційних технологій, самого ринку,

приватизації та появи акціонерних товариств, налагодження позабіржового обігу ЦП[1].

Інформаційні технології автоматизували роботу ФР, зробили можливим проводення ФО та отримуваня поточної і іншої необхідної інформацію в реальному часі у будь-якій точці світу. Завдяки чому почали з'являтися різні сервіси, що дозволяються працювати на ФР, сидячі за комп'ютером вдома через інтернет. Кожен рік збільшується кількість таких сервісів або покращуються вже існуючі, впроваджуються та проектуються нові ідеї та функіонал, створюються нові технології для покращення роботи старих або впровадження нових можливих функції.

Ще більший інтерес до сфери інвестиції світ отримав у 2018 та 2021 році. Пов'язано це з першим та другим криптовалютними бумом. При першому - світ дізнався та зацікавився новим типом ФА який став новим новим напрямком для інвестиції – криптовалюти. При другому – криптовалюти ще більше закріпилися на ФР та у суспільній думці.

Сучасний ФР продовжує розвиватися:відбувається подальша автоматизація фондових операції з поєднанням систем ФР у всесвітню мережу, триває робота над створенням нових видів і модифікацій цінних паперів (передовсім безпаперових), збільшується інтернаціоналізація фондової діяльності, відповідно до загального процесу економічної інтеграції країн[1].

Побудований сервіс для роботи з ФР, створений за допомогою мови програмування Java та його технології, допоможе отримувати поточну інформацію щодо поточної ціни ЦП та криптовалют та інформацію про її зміни та проводити ФО з активами. Ця програма буде корисна усім кожна фізична особа зацікавлена в збагаченні свого капіталу, кожній людині необхідно сберігати свої кошти від інфляцію або у деверсифікованому вигляді. Тому цінність цього сервісу буде помітна не тількі професіоналам у сфері інвестиції а й звичайним користувачам.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Основним завданням для ФР є збагачення покупця та знаходження інвестицій для продавця. Відбувається це завдяки продажу ФА, які можуть приносити якийсь прибуток або просто зростати у ціні знаходячись у володіні користувача, який може їх потім продати за новою цінною.

Для початку, слід розібратися з необхідною термінологією у сфері інвестиції. ФР – це сукупність економічних та торгових відносин[1]. Ринок на якому здійснються купівля та продаж ФА. Суб'єктами в цих відносинах виступають брокери(продавці) та покупці. Головними товарами на сучасному ФР виступають ФА до яких належать ЦП та з недавнього часу криптовалюти. В свою чергу ЦП поділяються на акції та облігації.

Акції – це один з видів ЦП, який посвідчує майнові права його власника(акціонера), що стосуються акціонерного товариства, включаючи право на отримання частини прибутку акціонерного товариства у вигляді дивідендів та право на отримання частини майна акціонерного товариства у разі його ліквідації, право на управління акціонерним товариством[1].

Облігації – це один з видів ЦП, що має певну вартість, встановлену емітентом (організацією-продавцем), та гарантує виплату покупцю цієї вартості у вигляді грошей чи майна у встановлений продавцем строк[1].

Криптовалюти – це цифрова валюта існуюча у глобальній мережі інтернет. З недавнього часу є одним з головних ФА що торгуються на біржі. Тоді як ЦП оперуються в ціллих одиницях, критовалюти можуть оперуватися в не цілих числах. Це дуже вигідно для звичайних користувачів, оскільки дозволяє роботи інвестиції в дорогі криптовалюти при малому капіталі.

Операції – операції на ФР називають покупку та продаж ФА. Такі операції змінюють стан портфоліо користувача. В аналогічних сервісах виділяються автоматичні та звичайні операції. До автоматичних належать покупка або продаж ФА при спрацюванні підписки.

Підписка – відстежування і реагування на зміни ФА, та проведення автоматичних дії(продаж, покупка або інформування) при виконанні умови спрацювання. Завдяки підписці можливо автоматично оброблювати зміни які відбуваються з ФА. Наприклад користувач може зробити так що якщо ціна активу впаде нижче якогось значення - то увесь або частина цього активу з портфоліо буде автоматично продано.

Портфоліо – це усі наявні активи користувача та їх кількість. Усі куплені ФА потрапляють у портфоліо, при продажі активи пропадають з портфоліо користувача і більше йому не належать.

ФР виконує роль посередника завдяки якому дві сторони фінансових відносин(брокер та покупець) можуть досягти необхідних їм ціллей. Для покупця однією з головних ціллей є збереження свого капіталу від інфляції завдяки покупці ФА. Активи через виплату фіксованої суми, майна або через звичайне зростання власної ціни дозволяють зберігати капітал від знецінення. Додатково для збереження капіталу використовують диверсифікацію. Диверсифікація – це розподіл капіталу між різними ФА, для зниження ризиків їх знецінення. Інвестування завжди пов’язане з ризиками, але якщо гроші рівномірно вкладати в різні активи, то падіння прибутків в одному сегменті компенсується зростанням доходів в іншому. У середньому диверсифікований портфель дасть стабільний плюс незалежно від ситуації у світовій економіці.

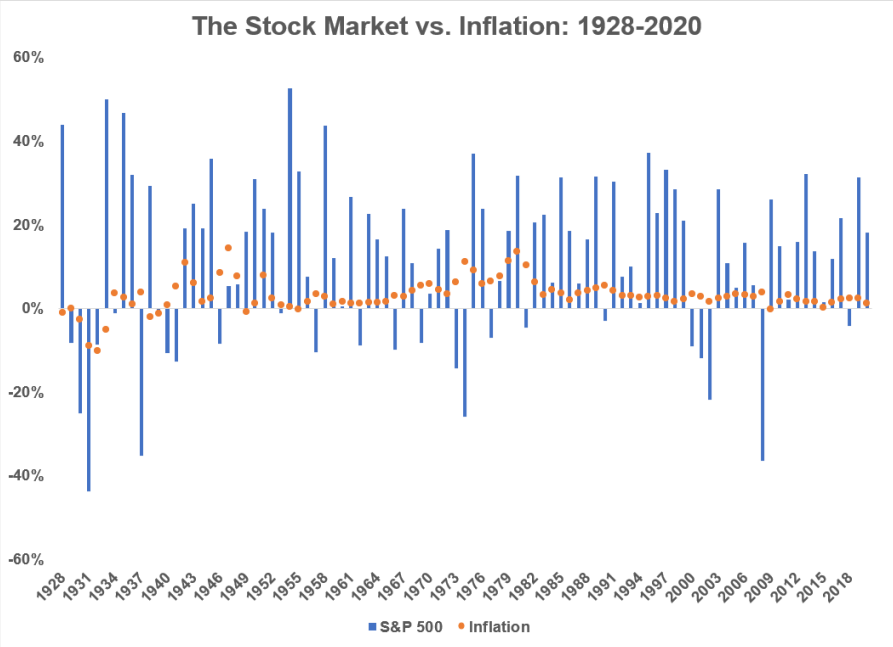


Рис 1.1 Відсоток росту ціни індексу S&P 500 до якого входять акції 500 найбільших компанії світу проти інфляції

Також ціллю покупця є збагачення свого капіталу завдяки інвестуванню у різні ФА. Існують два види інвестиції – довгострокові та короткострокові. Короткострокові інвестиції більш ризикові, вимагають більшого стартового капіталу, зусередженості, навичок, але в свою чергу такий тип інвестиції може принести найшвидше збагачення. Довгострокові інвестиції – натомість меньш ризикові, вимагають меньше стартового капіталу, зусередженості та навичок, але в свою чергу приносять меньше доходу. До тогож у більшості країн короткострокові інвестиції оподатковуються більшим відсотком ніж довгострокові.

Біржа як і все у світі підпорядковується якимось принципам та має свої закономірності. Зрозуміло що ФР завжди реагує на тренди та події у світі. Досліджуючи та аналізуючи реакцію біржі можливо визначити закономірності та принципи через які відбуваються ті чи інші зміни. Такий аналіз дозволяє отримувати необхідні знання для планування своїх майбутніх операції з ФО, корегувати свое портфоліо а також попередньо розуміти реакцію ринку на аналогічні події. З часом з’являються нові

тренди, закономірності та принципи. Через це покупці ФА повинні завжди аналізувати та адаптуватися до нових змін, оновлювати і підтримувати свої знання та навички.

Компанії що торгують своїми ЦП на біржі також адаптуються до сучасних реалії. Компанії впорядковують чи використовують світові тренди(наприклад нові технології). Завдяки трендам компанії приваблюють зацікавлену та небайдужу до цих трендів аудиторію, що приводить до росту популярності цієї компанії та закономірного росту вартості їх ЦП. Через тренди відбувається ріст вартості ЦП[2] і інтересу інвесторів. З цього можна зробити висновок що тренди це важливий рушій для інвестування.

Майже усі компанії які працюють з ФР приділяють дуже багато уваги медійності своєї компанії та рекламі. Наприклад одна з найвідоміщих і найбільших американських компанії що торгує ЦП на ФР та спеціалізується на розробці технічних засобів(телефонів, ноутбуків, комп'ютерів і т.д.), аксесуарів та програмного забезпечення до них – компанія Apple.

Компанія Apple має великий вплив на сучасний світ інформаційних технологій, тому кожна презентація її нових продуктів має бути беззуперечена як і самі продукти. Apple приділяє багато уваги реакції спільноти до своїх продуктів, наприклад є не офіційна теорія, що представники компанії після створення готових повноцінних концептів своїх продуктів, створюють так званні зливи - коли в глобальній мережі інтернет з'являються ці самі концепти на огляд суспільства.

Аналізуючи реакцію спільноти і ФР компанія робить висновки, вводить поправки або взагалі повністю відмовляються від цих концептів. Додатково багато уваги приділяється вже офіційній презентації своїх продуктів. Сама презентація виглядає як свято, все робиться в friendly-family стилі(для повного охоплення усієї можливої аудиторії), використовуються привабливі кольори, а саму презентацію проводять найголовніші особи в компанії, що ніби підкреслює дружні відносини компанії зі своєю аудиторією. Компанія

виставляє свою новинку як найсучаснішу і не маючу аналогів у світі, завдяки проведеню наглядного порівняння з аналогічними продуктами з налогошенням сильних сторін свого продукту перед аналогами. Додатково робиться акцент на новому функціоналі і технологіях, що піднімає увагу до компанії та зацікавлює потенційних покупців товару. Це сприяє підвищенню вартості ЦП компанії і зацікавленості інвесторів.

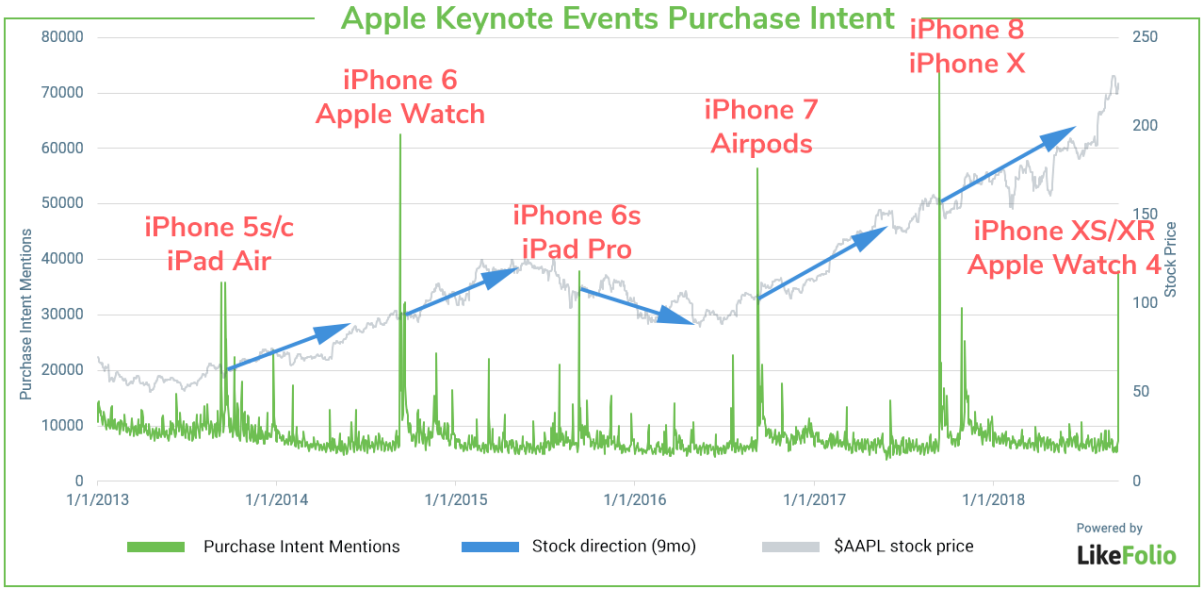


Рис 1.2 Ціна і інтенсивність покупок акції компанії Apple з прив'язкою до презентації нових продуктів.

Новим подихом у сфері інвестицій стали криптовалюти. Для розвитку нового ФА було багато причин: стався великий приток нових інвесторів з інших сфер інвестицій, популярність та трендовість використання та інвестування у криптовалюти а також розчарування суспільства у фіатних грошах через інфляцію.

Криптовалюти також підпорядковуються різним принципам та мають свої закономірності. Криптовалюти - це цифровий актив тому він відповідає сучасному тренду про діджиталізацію світу а також чутливо реагує на усі події у світі. Робота з криптовалютами сьогодні є трендом. Багато відомих людей та компанії цим користуються. Наприклад Ілон Маск для своєї

компанії Tesla закупив велику кількість криптовалюти Bitcoin та дозволив платити за товари своєї компанії криптовалютою. Це призвело як і до росту цінності Bitcoin так і до росту ціни ЦП компанії. Виходячи з цього можливо зрозуміти що криптовалюти чутливо реагують на згадування себе у медійній сфері.

Криптовалюти також підпорядковуються різним принципам та мають свої закономірності. Криптовалюти - це цифровий актив тому він відповідає сучасному тренду про діджиталізацію світу а також чутливо реагує на усі події у світі. Робота з криптовалютами сьогодні є трендом. Багато відомих людей та компанії цим користуються. Наприклад Ілон Маск для своєї компанії Tesla закупив велику кількість криптовалюти Bitcoin та дозволив платити за товари своєї компанії криптовалютою. Це призвело як і до росту цінності Bitcoin так і до росту ціни ЦП компанії. Виходячи з цього можливо зрозуміти що криптовалюти чутливо реагують на згадування себе у медійній сфері.

Криптовалютні активи дуже популярний ФА для інвестиції з різних причин. Однією з основних є інтенсивність зміни їх ціни. Криптовалюти частіше змінюються у ціні через що вони більш популярні для короткострокових інвестиції. Також короткострокові інвестиції у криптовалюті не оподатковуються більшим відсотком як ЦП, що робить їх більш вигідними для короткострокових інвестиції.

Криптовалюти зайняли своє місце у світі а також отримали таку популярність завдяки своєму швидкому становленю у світі. Багатьом сучасним компаніям знадобилось багато років щоб отримати капіталізацію в розмірі 1 трільйон долларів США, тоді як Bitcoin – одна з найпопулярніших криптовалют дісталась цієї відмітки найшвидше усіх в історії(рис. 1.3). Bitcoin як і інші криптовалюти продовжують свій розвиток, та роблять це бистріше ніж інші ФА.

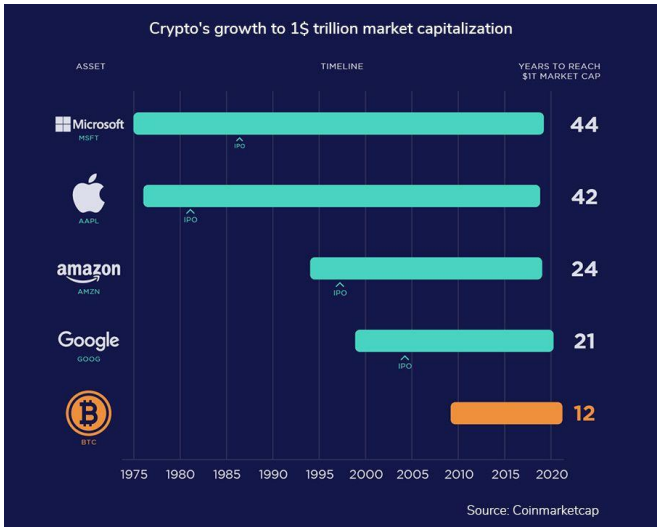


Рис 1.3 Кількість років для отримання капіталізації в 1 трільйон долларів США

Сучасна біржа чітко розуміє важливість, цінність та потенціал криптовалют. Криптовалюти – найпопулярніший тип інвестиції у сучасному діджиталізованому світі. Зараз майже немає людей які б не чули про криптовалюти. Через це кожного року з’являються все нові й нові криптовалюти(рис. 1.4). Інвестори цікавляться кожною новую криптовалютою через її потенціал, і роблять інвестиції після чого відстежують та аналізують свої інвестиції та самі криптовалюти.

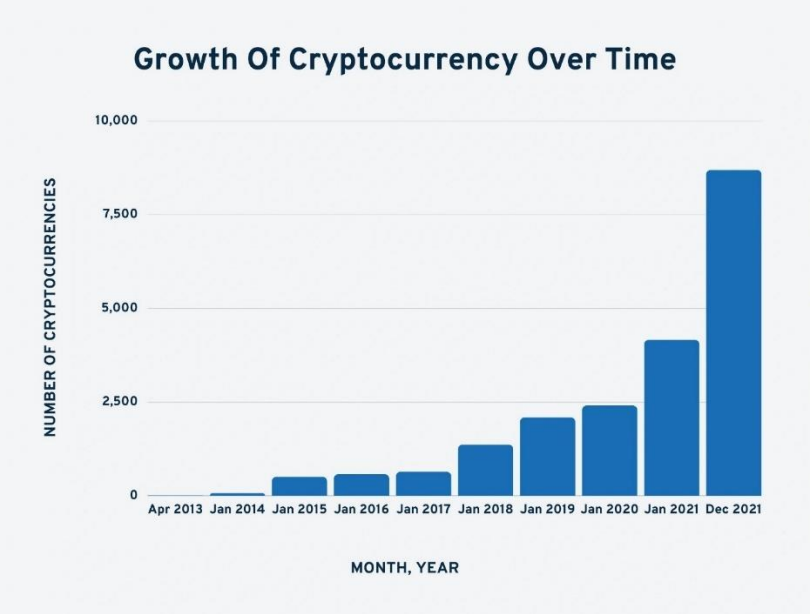


Рис 1.4 Зростання кількості видів криптовалют

ФР не зупиняється та продовжує свій розвиток та зростання, завдяки чому на ринок приходять нові компанії або криптовалюти, з’являються нові способи зберігти та примножити свій капітал, знайти інвестиції для своєї компанії. Через це ринок насичується новими інвесторами.

Інвестування дуже тісно інтегрується в наше сучасне життя. Наприклад в США користується популярністю така пенсійна система(401k) при якій людина довгостроково інвестує свої пенсійні збереження у ФА. Це дозволяє примножити свій пенсійний капітал для його подальшого використання при виході на пенсію або у пенсійному віці. Звичайно є й інші приклади використання інвестування, але лише з такого малого прикладу можливо зробити висновок - що інвестування дуже необхідно сучасній людині у житті. Через це людям необхідні сервіси для роботи у сфері інвестування, необхідні експерти та навички для розробки та покращення цих сервісів, створення нових та поліпшення роботи старих технології та функції для цих сервісів

1.2. Призначення розробки та галузь застосування

Сервіс, що розроблений для кваліфікаційної роботи, має назву «Розробка сервісу для моніторінгу та проведення операції на ринку ЦП та криптовалют на основі Java/Spring».

Ключові слова:

Java – одна з найпопулярніших високорівневих мов програмування. Має велику кількість додаткових фреймворків та бібліотек для вирішення різних проблем та полегшення розробки додатків.

Spring – найпопулярніший фреймворк для мови програмування Java. Має велику кількість готових рішень, які можна використовувати для реалізації необхідних функції додатку.

ЦП – один з видів активів які використовують для торгівлі на біржі. До ЦП відносять облігації та акційні папери.

Криптовалюти – вид цифрової валюти яка існує в рамках глобальной мережі інтернет.

Моніторинг – отримування інформації щодо ФА з ресурсів які мають доступ до інформації ФР. До інформації відноситься поточна ціна ФА або історія її зміни.

Операції – ФО з ЦП та криптовалютами такі як їх покупка або продаж з портфоліо користувача.

Розроблений продукт може використовуватися на підприємствах або фізичними особами які мають брокерську ліцензію та право виступати продавцями на біржі, а також мають доступ до інформаційних систем та сервісів ФР, завдяки яким можна проводити ФО та отримувати необхідну інформацію про ФА.

Призначення розробки - надати можливість користувачам додатку доступ до данних з біржі а також можливість проводити операції з ФА щодо покупки та продажі на ФР.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

- освітня програма спеціальності 122 “Комп’ютерні науки”;
- навчальний план та графік навчального процесу;

- наказ ректора Національного технічного університету “Дніпровська політехніка” № 268-с від 18.05.2022р;

- завдання на кваліфікаційну роботу на тему “ Розробка сервісу для моніторінгу та проведення операції на ринку цінних паперів та криптовалют на основі Java/Spring”.

1.4. Постановка завдання

Метою проекту є розробити сервіс для моніторингу поточної інформації з біржі та можливістю проведення операції з ФА.

Даний продукт дозволить отримувати поточну інформацію щодо поточних цін на ФР та їх змін у зручному для користувача виглядії проводити ФО з покупки-продажі ЦП та криптовалют на біржі а також створювати автоматичні реакції на зміни ФА.

Основними характеристиками розробки повинні бути:

- інтуїтивно зрозумілий інтерфейс;
- постійне оновлення та обробка данних;
- використання зовнішніх ресурсів для отримання необхідної інформації;
- налагодженість логіки у алгоритмах роботи;
- належно побудована структура бази данних(БД)

Поставлена задача може бути досягнута при виконанні наступних вимог:

- вивчення предметної області завдання;
- проведення порівняльної характеристики можливостей аналогічних програм;
- належне проектування структури та функціоналу системи;
- налагодження алгоритмів роботи системи;

– вибір технології для розробки;
– написання програмного коду згідно стандартів розробки.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт повинен дотримуватися наступних функціональних вимог:

– підтримка авторизації з використанням зовнішніх сервісів;
– надання користувачу інструментів для автоматичного реагування на зміни;
– інформування користувача щодо спрацювання підписок;
– можливість роботи з рахунком користувача;
– забезпечення проведення ФО з ЦП та криптовалютами;
– постійне оновлення інформації та обробка данних;
– збереження проведених операції з ФА та рахунком;
– перегляд усіх операції з рахунком та активами;
– отримання інформації щодо поточної та попередньої інформації про ФА;
– доступ до зрозумілого портфоліо з усіма наявними активами та інформацією про них.

1.5.2. Вимоги до інформаційної безпеки

Організація безпеки в веб додатку має бути організована на програмних засобах самого додатку та його технологіях. Дана вимога включає в себе загальні завдання забезпечення безпеки, такі як захист окремих функціональних частин додатку, перевірка авторизації

користувачів а також застосування ефективної політики збереження, шифрування та перевірки паролів користувачів.

Безпека БД має бути налаштована за допомогою механізмів MySQL. Сюди можна віднести завдання побудови безпечних інтерфейсів і механізмів доступу та роботи з даними.

Безпечна організація і робота з даними. Питання організації даних і управління ними є ключовим. У цю область входять завдання організації даних з контролем цілісності яка має бути спроєктована та підтримуватися на рівні додатків та БД.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для належної роботи сервіс буде поділено на два додатка, кожен з яких буде знаходитись на окремому сервері. Додатково необхідни окремий сервер для бази данних.

Для серверу з БД можна використовувати спеціалізовані готові сервера у хмарних сервісах таких як Amazon Web Service(AWS) – AWS Relation Database Service, Microsoft Azure(MA) – Microsoft Azure SQL Database або Google Cloud Platform(GCP) – Cloud SQL. Також можна використовувати звичайний сервер з наступною конфігурацією:

- операційна система: Linux-типу: Ubuntu 20.04.4 LTS, Read Hat Enterprise Linux 8, Debian 11.3, Windows Server 2018;
- оперативної пам'яті 4 гігабайт;
- процесор x64 та мінімум 4 ядрами з тактовою частотою 1,4 ГГц;
- 10 гігабайт вільного місця на диску.

Для серверу з додатком який буде проводити постійну обробку та оновлення данних можна використовувати звичайні готові сервера у хмарних сервісах таких як AWS – EC2, MA – Windows Server або GCP –

Cloud Computing Services. Також можна використовувати звичайний сервер з наступною конфігурацією:

- операційна система: Linux-типу: Ubuntu 20.04.4 LTS, Read Hat Enterprise Linux 8, Debian 11.3, Windows Server 2018;
- оперативної пам'яті 16 гігабайт;
- процесор x64 та мінімум 8 ядер з тактовою частотою 2,4 ГГц;
- 6 гігабайт вільного місця на диску.

Для серверу з веб додатком можна використовувати аналогічні до попереднього сервера з аналогічною конфігурацією.

1.5.4. **Вимоги до інформаційної та програмної сумісності**

На серверах на яких будуть розгорнуті додатки необхідно розмістити програмне забезпечення Java Runtime Environment(JRE), що дозволить запускати скомпільовані Java додатки.

На сервері з БД має бути розмішенно програмне забезпечення MySQL, що буде виконувати роль самої БД. Також БД має бути налаштована належним чином щоб програмні додатки мали доступ до усіх її функції та данних.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом даної кваліфікаційної роботи має бути набір з додатків, перший з яких виконує роль веб сервера з графічним інтерфейсом завдяки якому користувач може отримувати інформацію а також виконувати різні дії. Другий додаток буде постійно оброблювати та оновлювати інформацію з базі данних завдяки отримуванню інформації з зовнішнього ресурсу.

Основне призначення сервісу:

- отримування поточної інформації щодо ЦП та криптовалют з зовнішнього ресурсу;
- отримування інформації щодо попередніх оновлень ціни ФА;
- проведення операцій з ЦП та криптовалютами;
- робота з рахунком користувача;
- зберерігання інформації щодо проведених операцій;
- створення підписок для автоматичного реагування на зміни ФА;
- інформування користувача щодо спрацювання підписок;
- перегляд та робота з портфоліо.

Для досягнення поставленої задачі додаток повинен вміти постійно отримувати та оновлювати інформацію з використанням зовнішнього ресурсу, оброблювати великі об’єми данних з використанням чітко налагоджених механізмів та алгоритмів обробки, а також видавати потрібну інформацію для користувача через графічний інтерфейс у зрозумілому вигляді для користувача вигляді.

2.2. Опис застосованих математичних методів

Під час проектування та розробки інформаційной системи математичні методи не використовувалися, були проведені лише базові математичні дії такі як додавання, віднімання та множення.

2.3. Опис використаних технологій та мов програмування

Дана комп’ютерна система розроблена за допомогою наступних інформаційних технологій:

- мова програмування Java;
- фреймворк Spring;
- додаткові бібліотеки фреймворку Spring;
- додаткові зовнішні бібліотеки;
- HyperText Markup Language(HTML);
- Cascade Style Sheets(CSS);
- Javascript;
- Bootstrap;
- Thymeleaf;
- Maven;
- Liquibase;
- Hibernate;
- MySQL.

Java – високо рівнева мова програмування, загального призначення з статистичною строгою типізацією і автоматичним управлінням пам’ятю, високоефективна в плані швидкості, підтримує мультипоточність і має величезну кількість додаткових бібліотек та інструментів для вирішення проблем розробки додатків.

Основною парадигмою мови програмування є об’єктно орієнтована. Тобто елементами програми на Java є об’єкти. Внутрішня структура

об’єктів(методи та поля) описана у классах. Поля класу можуть бути як об’єктами так і множиною об’єктів або звичайними данними(текст, символ, цифри і т.д.) або їх множиною. Класи можуть містити статичні елементи та методи які можуть існувати та виконуватись без створених об’єктів цього класу. Java підтримує наслідування один від одного(тобто клас може наслідуватися лише від одного классу). При наслідуванні клас стає підвидом класу від якого наслідується та отримує його властності, методи та поля. Усі об’єкти в Java наслідуються від классу Object , тому мають звичайний набір методів такі як: toString – метод для переводу об’єкта у формат тексту, equals – метод для порівнювання об’єктів, hashcode – метод для отримання хеш-коду об’єкту(цифрове значення для роботи в хеш структурах) і т.д. Методи можуть бути перевизначені і отримувати іншу реалізацію. Також мова підтримує перегрузку – коли клас може мати методи з одним і тим же ім’ям але різними типами і кількістю вхідних параметрів та вихідним типом.

Синтаксис мови відноситься до классу С-подібних мов програмування. Головною особливістю вирізняють принцип “скомпілюй один раз, та запускай всюди” (*англ.* “compile once, run everywhere”), тобто скомпільований Java додаток у одній системі можливо запустити усюди. Досягається це завдяки складовій JRE – Java Virtual Machine(JVM) а також особливістю компіляції Java-додатків. При компіляції Java-додатків, на виході ми отримаємо файл з джава байт-кодом, в свою чергу завдяки JVM можливо запустити цей додаток, JVM інтерпретує скомпільований джава байт-код у системні команди операційной системи для їх подальшого виконання.

Spring – Java фреймворк для полегшення розробки, включає в себе багато різних модулів(рис. 2.1), головним і основним з яких є Spring Core Container. Головним елементом фреймворку є Inversion of control(IOC) та Dependency Injection(DI) контейнер(або Application Context). IOC та його

реалізація DI це основні патерни на яких побудовано фреймворк Spring та Spring container. Контейнер контролює життєві процесси елементів додатку(створення, знищення і т.д.). Елементом додатку називають бін(*англ.* Bean). Можливо створювати додаткові необхідні біни при наданні додатковій конфігурації і необхідної інформації для їх роботи . Також фреймворк підтримує спеціальну мову для написання таких конфігурації.

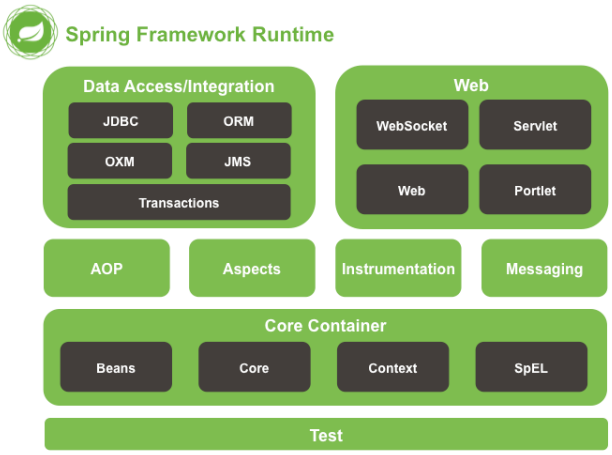


Рис. 2.1 Модулі фреймворку Spring

Додатково було використано велику кількість додаткових бібліотек фреймворку Spring для роботи необхідно функціоналу. Нижче наведено перелік усіх використаних бібліотек Spring.

Spring Web – додаткова бібліотека Spring фреймворку для розробки веб-додатків з використанням протоколу HyperText Transfer Protocol(HTTP) на основі технології сервлетів. Сервлет – це розширення функціоналу сервера, інтерфейс завдяки якому додаток може отримувати, обролювати HTPP запити та відправляти відповіді. Працює Spring Web на основі патернів Model View Controller(MVC) та Front Controller(рис. 2.2).

MVC - паттерн Front Controller – паттерн при якому усі запити на сервер, проходять через один сервлет(контролер) який переселає їх далі на обробку, в Spring Web таким сервлетом(контролером) виступає Dispatcher Servlet.

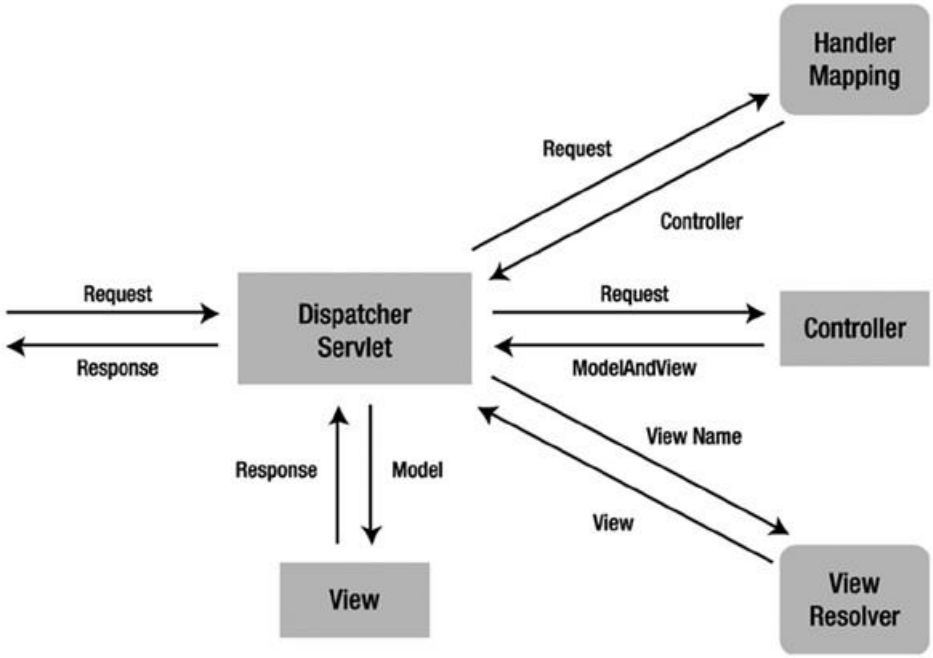


Рис. 2.2 MVC та Front Controller паттерн в Spring Web

Модуль дозволяє розробляти додатки як з графічним(на основі HTML) так і без графічного інтерфейсу. Також включає в себе додаткові інструменти для роботи з зовнішніми ресурсами.

Додаткам на основі Spring Web потрібно додаткова інфраструктура у вигляді контейнеру сервлетів для розгортки та запуску додатка. Найчастіше для розгортання таких веб додатків використовують Apache Tomcat. Apache Tomcat – найпопулярніший контейнер сервлетів який використовується для розгортки веб-додатків на основі сервлетів та Java. Контейнер сервлетів повністю контролює життєвий цикл веб елементів додатку(тобто сервлетів, фільтрів і т.д.).

Spring WebSocket – додаткова бібліотека Spring для забезпечення роботи з технологією WebSocket. WebSocket – технологія прямого неприривного підключення, є розширенням базового протоколу TCP. Завдяки підключенню Websocket можливо швидко та безперервно отримувати або відсилати данні.

Websocket підключення працює по наступному алгоритму, спочатку відсилається звичайний HTTP запит з додатковою опцією для модифікації поточного підключення до підключення на основі WebSocket. Після цього встановлюється Websocket підключення з притаманим номером сессії(для роботи з багатьма підключеннями) і стає можливим двостороній обмін данними.

Spring Security – додаткова бібліотека для конфігурації безпеки веб-додатку на основі Spring. Бібліотека надає можливість створювати різні рівні доступу до функціоналу системи, а також проводити властну обробку запитів перед їх отриманням.

Spring Java Persistence API(JPA) – набір інтерфейсів для роботи з БД через Spring додаток. В модуль не включена реалізація, найчастіше для реалізації інтерфейсів модулю використовують додатковий зовнішній фреймворк Hibernate.

Spring Aspect-Orienting Programming(AOP) – додаткова бібліотека та модуль Spring, який забезпечує обробку та формування наскрізного функціоналу додатку. Наскрізний функціонал - це коли окрім видимого функціоналу спрацьовує додатковий(невидимий) який доповнює видимий функціонал чи виконує власний незалежний функціонал. Для роботи наскрізного функціоналу необхідний об’єкт при початку роботи додатку обгортається у проксі – об’єкт який буде використовуватися замість цього об’єкту і мати в собі увесь функціонал та данні самого об’єкта та реалізацію додаткового наскрізного функціоналу.

Завдяки AOP можна отримувати інформацію про те який метод і з якими параметрами почав роботу, заміняти або доповнювати вхідні параметри та результат виконання. Головними елементами AOP є aspect, join-point, pointcut, advice. Aspect це набір з join-point, pointcut та advice, який і є формуванням наскрізного функціоналу. Join-point це момент при якому буде спрацьовувати нашо наскрізний функціонал. Pointcut – набір

join-point. Advice – реалізація наскрізного функціоналу яка буде виконуватися при його спрацюванні.

Spring OAuth2 – додаткова бібліотека для Spring Security, завдяки якому можливо налаштовувати авторизацію з використанням зовнішніх сервісів. Пряцює на основі OAuth2 принципу. Коли після авторизації на зовнішньому ресурсі користувач отримує токен який містить необхідну інформацію щодо юзера і підтверджує його авторизацію через зовнішній ресурс.

Spring Boot – бібліотека Spring, яка використовується для прискорення та облегшення розробки. Головною особливістю є наявність великой кількості підмодулей(стартерів) які забезпечують створення необхідних бінів та інфраструктури додатку для роботи з різними технологіями. Створення та налаштування інфраструктури та бінів відбувається завдяки використаню конфігураційних файлів що мають містити необхідні данні для роботи того чи іншого функціоналу. Нижче наведено список стартерів Spring Boot що були використані.

Spring Boot Web – підмодуль Spring Boot який прискорює розробку веб додатків. Додатково має при собі вбудовану систему розгортки веб додатків на основі сервлетів Apache Tomcat, завдякі якій на сервері на якому будуть розгортати додаток не буде потрібно додатково встановлювати контейнер сервлетів.

Spring Boot Security – підмодуль Spring Boot який прискорює налаштування безпеки веб-додатку.

Spring Boot Oauth2 – підмодуль Spring Boot який прискорює налаштування авторизації у додатку за допомогою зовнішніх ресурсів на основі OAuth2.

Spring Boot Data Jpa – підмодуль Spring Boot який прискорює налаштування для роботи додатку з БД.

Spring Boot Thymeleaf – підмодуль для роботи технології Thymeleaf у графічному інтерфейсі Spring Web додатку на основі HTML.

Також невід'ємною частиною для поліпшення та прискорення розробки на мові програмування Java є використання вже готових рішень на основі бібліотек. В цьому проекті були використані додаткові бібліотеки, що перелічені нижче.

Lombok – одна з найпопулярніших бібліотек для розробки на мові програмування Java. Прискорює розробку завдяки генерації типових методів в классах (сеттери, геттери, методи toString, equals, hashcode, а також генерацію конструкторів), це допомагає запобігати написаню типового коду та дозволяє фокусуватися на інших задачах.

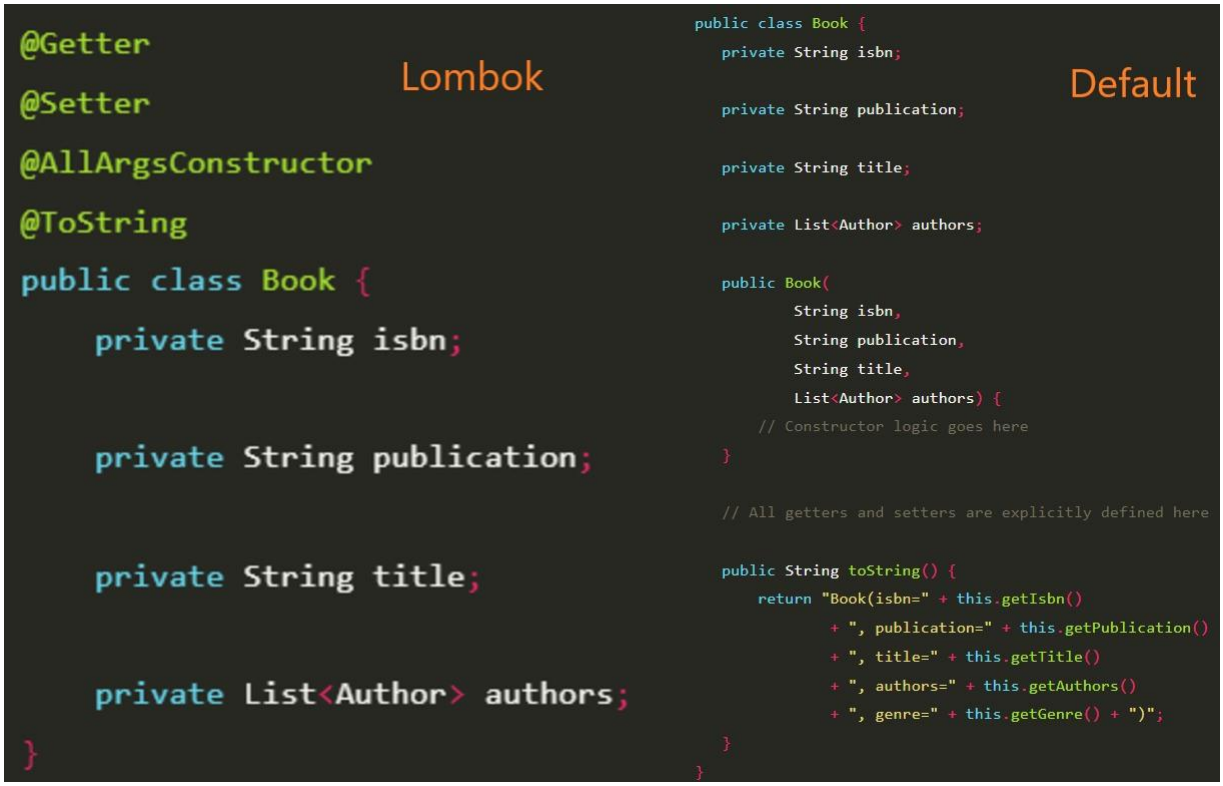


Рис. 2.3 Вигляд коду з використанням Lombok та без

Java Jason Web Token – бібліотека для роботи з Jason Web Token(JWT) для додатків на основі Java. Бібліотека пропонує вже готові методи для генерації, перевірки та обробки JWT токенів. JWT – один з

найпопулярніших способів захисту данних і роботи з захищеними ресурсами та сервісами. Також цей спосіб популярний завдяки тому що JWT окрім перевірки доступу до ресурсу може використовуватися для передачі інформації яка має бути захищена(наприклад логін користувача).

MySQL Connector – набір інтерфейсів, що дозволяють додатку працювати з MySQL. Містить необхідні методи для підключення, забезпечення безпеки, цілостності данних та створення запитів до БД.

Jackson – бібліотека для ефективної та шкидкої роботи з форматом Jayson(JSON) для мови програмування Java. Дозволяє перетворювати JSON на об’єкт Java або навпаки. Також забезпечує звичайну обробку JSON задля отримання данних або створення динамічного JSON.

HTML – мові розмітки для документів які використовуються для відображення інформації та графічного інтерфейсу у веб додатках. Також HTML може використовувати додаткові технології такі як JavaScript для створення скриптів та динамічних події у документі, та CSS для зміни візуальних атрибутів елементів HTML.

Javascript – динамічно типізована скриптована мова програмування. Найчастіше використовується для створення динамічних елементів на HTML сторінці.

CSS – мова для налаштування графічного стилю HTML сторінок.

Thymeleaf – сучасний Java серверний двигун для HTML веб сторінок. Забезпечує інтеграцію HTML сторінки з Java додатком. Дозволяють спрошювати створення та налаштування елементів HTML завдяки використанню Java та її об’єктів..

Bootstrap – фреймворк для налаштування стилю веб сторінок, на основі вже підготовленого дизайну для швидкого налаштування вигляду веб сторінки в належному візуальному стилі.

Maven – інструмент для автоматичної побудови та налаштування зовнішніх залежностей у Java додатках. Також має автоматичного

розгортання додатку на локальній машині або віддаленому сервері та запускати необхіді скриптів, налаштування процеси та етапи побудови та розгортки додатків.

Liquibase – інструмент для роботи з реляційними БД, який забезпечує відстеження, адміністрування, та впровадження змін до схеми або записів в БД завдяки скриптам. Також звичайні Liquibase скрипти незалежні до синтаксису окремої БД, тому кожен скрипт який працює на одній БД, може бути використаний і для іншої з іншим синтаксисом. Також є і підтримка скриптів у звичайному залежному діалекті БД. Головним форматом скриптів на Liquibase є XML, але також є підтримка YAML та JSON.

Hibernate – Object Relation Mapping(ORM) для роботи з БД. ORM – це технологія яка здатна об'єднувати таблиці та данні з БД з об'єктами у обьєктно-орієнтованій мові програмування, в нашому випатку з об'єктами Java. Об'єкти Hibernate також мають властний життєвий цикл. Hibernate може використовувати увесь функціонал реляційних БД(ізоляції транзакції, цільність данних і т.д.). Працює Hibernate на основі Java Database Connectivity (JDBC) – який є самим базовим інструментом для роботи з БД який є в мові програмування Java. Головним елементом Hibernate є SessionFactory – елемент який існує на всьому циклі життя додатку, і відповідає за створення сессії та підключення до БД. SessionFactory створює Session – короткоживучий елемент, через який проводяться операції з БД.

Операції з БД виконуються завдяки об'єктам. Об'єкти Hibernate це відображення майбутніх та існуючих записів та сутностей в БД. Об'єкти Hibernate мають властний життєвий цикл.

Hibernate також має додатковий власний функціонал, наприклад кешування. Кешування дозволяє знизити час який потрібен на те, щоб отримати інформацію, оскільки замість прямого запиту в БД, інформація достається з локального кешу. В Hibernate існує два рівні кешу: кеш першого рівню та кеш другого рівню. Кеш першого рівню існує у кожній

сессії(в об’єкті Session) і працює завжди без додаткової конфігурації. Запит до кешу першого рівня спрацьовує коли в однії сессії роботи з БД, виконується запит однієї і тієїж інформації(якщо вона не була змінена). Кеш другого рівня існує у всьому додатку(в об’єкті SessionFactory) без прив’язки до сессії, але для його роботи необхідна додаткова конфігурація.

Об’єкти мають властну структуру яка дозволяє мати вкладені об’єкти або їх списки. Hibernate може повністю повторювати цю структуру, завдяки чому при запиті лише однієї сутності можливо дістати й інші пов’язані з нею завдяки додатковим запитам до БД без їх явного виклику. Все це буде надано у вигляду одного об’єкту з вкладеними у нього сутностями.

Також Hibernate має функціонал відложеного запиту для вкладених об’єктів – FetchType.Lazy. При відложеному запиті спершу з БД достаються лише первині ключі необхідних записів які заносяться у проксі об’єкти що заміняють звичайні об’єкти. Коли нам потрібно отримати іншу інформацію окрім первиних ключів об’єктів, спрацьовує AOP та створює запит до БД і заміняє проксі на звичайний об’єкт з усіма необхідними данними.

Hibernate має власний діалект запитів завдяки якому можливо створювати незалежні до синтаксису БД запити. Але при необхідності можливо використовувати звичайні запити до БД.

Додатково слід виділити функціонал роботи Hibernate з транзакціями БД в Hibernate. Робота з транзакціями працює на основі AOP. Існує декілька способів роботи з транзакціями завдяки яким можливо створювати, закінчувати, модифікувати або взагалі ніколи не виконувати транзакції.

Останьою використаною технологією є система управління базою данних(СУБД) MySQL. MySQL використовує Structured Query Language(SQL) в якості головної мови для розробки та формування запитів до БД. Завдяки SQL можливо зчитувати, записувати, змінювати, модифікувати данні. Данні у БД зберігаються у кортежах, а кортежі у таблиці.

MySQL – реляційна БД, тому таблиці мають чітку структуру полів завдяки яким підтримується цілісність данних у базі данних, згідно з чого кортеж який записується у таблицю має мати структуру як і в таблиці куди записується. Створювати, видаляти, модифікувати структуру таблиць можна також за допомогою SQL.

MySQL підтримує комплексні запити тому у вихідному кортежі данних який можна отримати при запиті у базу данних може містити дані не тільки з однієї таблиці.

Таблиці можуть мати зв'язки між собою і БД буде підтримувати цілісніть цих зв'язків. Зв'язок між таблицями працює на основі ключів. Виділяють два типи ключів: зовнішній(foreign key) та первинний ключ(primary key). Первинний ключ - унікальний, тому що його використовують для того щоб ссилатися на цей кортеж. Також первиний ключ може існувати лише один у таблиці. Зовнішній ключ може містити значення лише існуючого первиного ключа на який він зсилається. Кількість зовнішніх ключів необмежена, тобто таблиця може зсилатися на одну або більше таблиць. Головним обмеженням є те, що таблиці з данними на які зовнішній ключ потрібен зсилатися мають мати первиний ключ. Також ключі завжди мають мати значення, тобто не бути пустими.

MySQL база данних відповідає усім вимогам ACID – тому її можна назвати транзакційною системою. Транзакційна система – це система в якій відбуваються транзакції з данними. Транзакція – це набір закінчених дії з данними(зчитування, записування, видалення, змінення). ACID розшифровується як Atomicity, Consistency, Isolation, Durability. Atomicity або атомарність – означає те, що кожна транзакція може тільки пройти або не пройти(тобто транзакція не може бути частично завершеною). Consistency або консістентність – означає те, що структура та данні кортежу які використовуються у транзакції повині бути лише валідними та відповідними. Isolation – або ізоляція – означає те, що транзакції не можут

впливати на одне одного. Durability або стійкість – означає те, що данні які були записані не будуть втрачені.

Також слід виділити підтримку ізоляцій транзакції. Всього існує чотири рівня ізоляції які вирішують властну проблему і проблему попереднього рівня: Read Uncommited, Read Commited, Repeatable Read, Serializable.

Read Uncommited – найнижчий серед рівнів ізоляцій транзакції, дозволяє зчитувати усі данні без блокування завдяки чому є найшвидшим у роботі рівнем ізоляції транзакції.

Read Committed – другий рівень ізоляції транзакції, при якому можливо зчитувати лише закомічені дані. Вирішує проблему Dirty Read – коли транзакція зчитує не закомічені данні.

Repeatable Read – третій рівень ізоляції транзакції, при якому данні над якими проводяться операції блокуються до кінця транзакції(тобто над ними неможливо провести ніякі операції), вирішує проблему Non-Repeatable Read – коли під час транзакції проводяться два однакових зчитування і отримаються різні данні в отриманому кортежі.

Serializable – останній четвертий рівень ізоляції транзакції, при якому вся таблиця в якій проводяться операції блокується до кінця транзакції. Вирішує проблему Phantom Read – коли проводиться дві однакових операції зчитування та отримується різні кортежі або їх кількість.

2.4. Опис структури системи та алгоритмів її функціонування

Для логічного розмежування функціоналу сервіс поділений на 2 окремих додатка. Перший додаток – головний додаток, який виконує роль веб ресурсу з графічним інтерфейсом. Доступ до додатку можливо отримати з любого інтернет браузеру завдяки HTTP запиту. В системі відмічено два типа користувачів – неавторизований та авторизований.

Авторизований користувач має більш розширений функціонал такий як проведення операцій з ФА, створення або видалення підписок, робота з особовим рахунком, перегляд портфоліо та повідомлень. Тоді як неавторизований користувач має лише можливість перегляду інформації щодо ФА.

Другий додаток – самостійний додаток який постійно оновлює данні завдяки отриманю інформації з зовнішнього ресурсу, та оброблює активні підписки у системі. В якості зовнішнього ресурсу використовується Twelve Data. Twelve Data – відкритий інтернет ресурс, який надає користувачам можливість отримати доступ до різної необхідної інформації щодо активів (наприклад до поточної ціни). Додаток підключається до віддаленого ресурсу завдяки протоколу Websocket через який система отримує інформацію щодо оновлення ФА(акції та криптовалют). Після отримання інформації проводиться обробка данних з оновленням інформації у базі даних та обробці підписок користувачів умова яких виконується. При обробці підписки створюється повідомлення для користувача, та при необхідності може проводитись закуп або продажа активів користувача.

Для збереження данних використовується СУБД MySQL. Структура схеми БД системи(рис.) має наступні сутності та їх атрибути:

1. Таблиця користувачів (user). Атрибути:
   - ID користувача;
   - електронна адреса;
   - електронна адреса в справжньому вигляді;
   - пароль;
   - баланс;
   - ім’я;
   - прізвище.
2. Таблиця активів (valuable). Атрибути:

- ID активу;
- Символ активу;
- ім’я активу;
- тип активу;
- ціна;
- дата та час останьої змінни.

3. Таблиця замовлень (order). Атрибути:
   - ID замовлення;
   - ID активу;
   - ID користувача;
   - ціна активу;
   - повна ціна замовлення;
   - кількість активу;
   - дата та час замовлення;
   - тип замовлення;
   - статус замовлення.
4. Таблиця активів користувача (user_valuables). Атрибути:
   - ID;
   - ID активу;
   - ID користувача;
   - кількість активів;
   - повна ціна замовлення;
   - кількість активу;
   - кількість активу для продажу.
5. Таблиця банківських карт користувача (card). Атрибути:
   - ID картки;
   - ID користувача;
   - CVC номер картки;

- номер картки;
- місяць кінця дійсності картки;
- рік кінця дійсності картки.

6. Таблиця історії змін активу (valuable_history). Атрибути:
   - ID зміни;
   - ID активу;
   - нова ціна;
   - стара ціна;
   - дата та час зміни;
   - зміна.
7. Таблиця історії операції з балансом (balance_history_update). Атрибути:
   - ID зміни;
   - ID користувача;
   - тип операції;
   - зміна;
   - дата та час зміни;
   - картка якою проводилася операція.
8. Таблиця підписок (subscription). Атрибути:
   - ID підписки;
   - ID користувача;
   - ID активу;
   - тип підписки;
   - кількість активів для підписки;
   - оператор для умови підписки;
   - тип умови для підписки;
   - нижня ціна для підписки;
   - верхня ціна для підписки;

- ціна для підписки;
- чи продовжувати підписку;
- чи резервувати активи;
- чи зберігати підписку при помилці.

9. Таблиця повідомлень(alerts). Атрибути:
   - ID повідомлення;
   - ID користувача;
   - повідомлення
   - дата та час повідомлення.

Далі наведено ключі, за рахунок яких пов’язані сутності БД.

1. Користувачі (ID користувача).
   - ID користувача - первинний ключ.
2. Активи (ID активу).
   - ID активу – первинний ключ.
3. Замовлення (ID замовлення, ID користувача, ID активу).
   - ID замовлення - первинний ключ;
   - ID користувача - зовнішній ключ;
   - ID активу – зовнішній ключ.
4. Активи користувача (ID списку активів, ID користувача, ID активу).
   - ID списку активів - первинний ключ;
   - ID користувача - зовнішній ключ;
   - ID активу - зовнішній ключ.
5. Банківські карти користувача (ID картки, ID користувача).
   - ID картки - первинний ключ;
   - ID користувача - зовнішній ключ.
6. Історія зміни активу (ID історії, ID активу).
   - ID історії - первинний ключ;

- ID активу - зовнішній ключ.

7. Історія операції з балансом (ID онлайн-уроку, ID студента, ID вчителя).
    - ID історії - первинний ключ;
    - ID користувача - зовнішній ключ.
8. Підписка (ID підписки, ID користувача, ID активу).
    - ID підписки - первинний ключ;
    - ID користувача - зовнішній ключ;
    - ID активу - зовнішній ключ.
9. Повідомлення користувача (ID повідомлення, ID користувача).
    - ID повідомлення - первинний ключ;
    - ID користувача - зовнішній ключ.

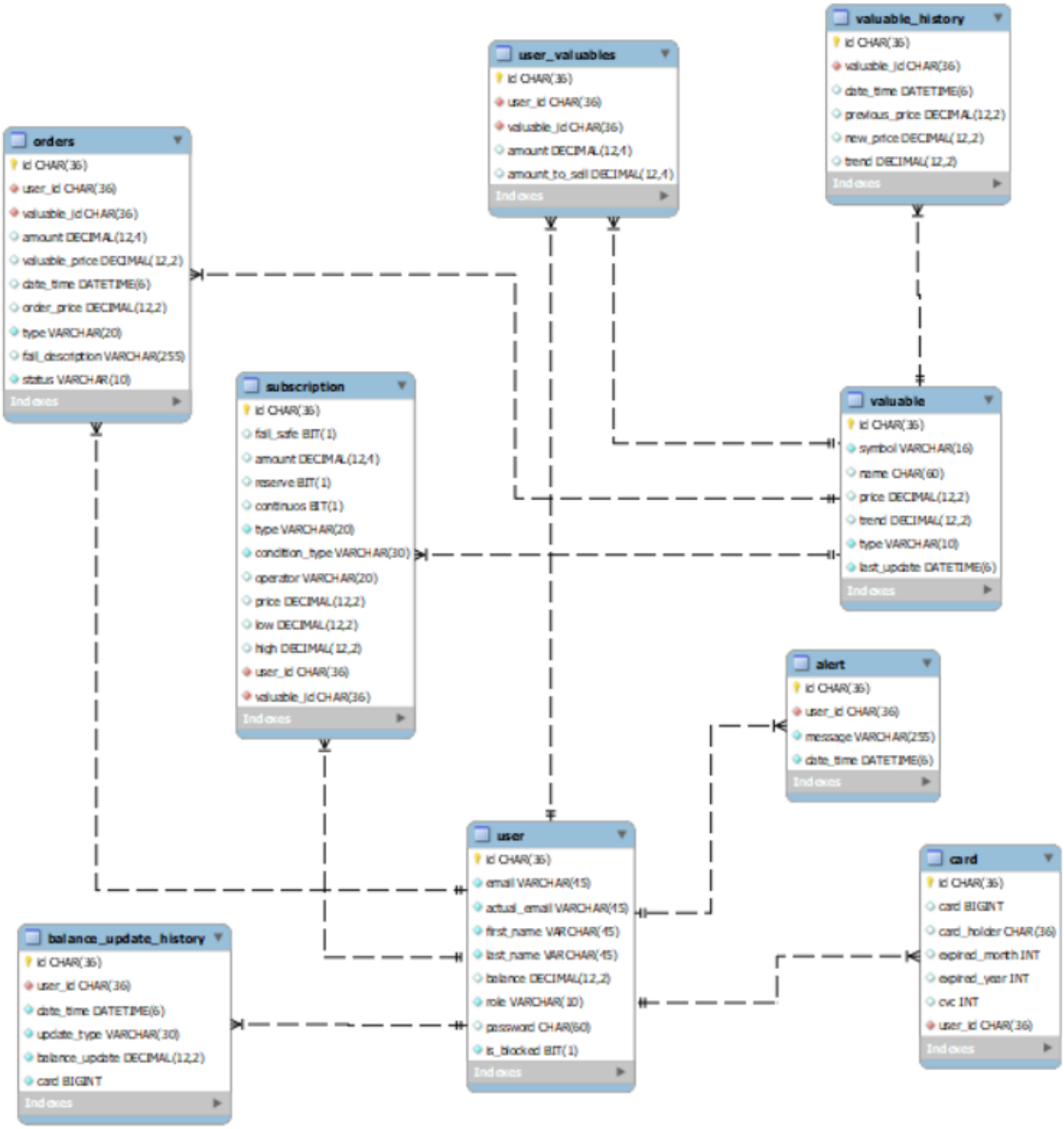


Рис. 2.4 Схема розробленої БД

Сервіс використовує складні алгоритми для забезпечення належного функціонування системи та обробки усіх можливих події при свої роботі.

Наприклад для здійснення покупки або продажі , замовлення перевіряє тип активу який купується і якщо актив не належить до криптовалют то перевіряється те що кількість у замовленні рівна цілому числу. Також для покупки перевіряється чи достатньо балансу на рахунку користувача, а при продажі чи є достатня незарезервована кількість активу у портфоліо користувача.

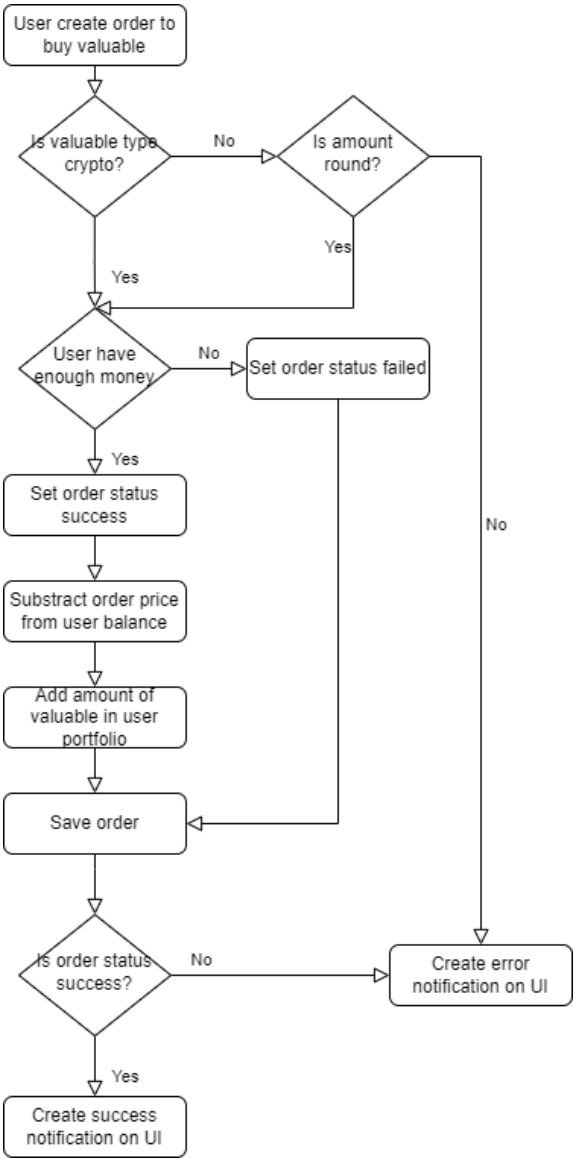


Рис. 2.5 Алгоритм покупки

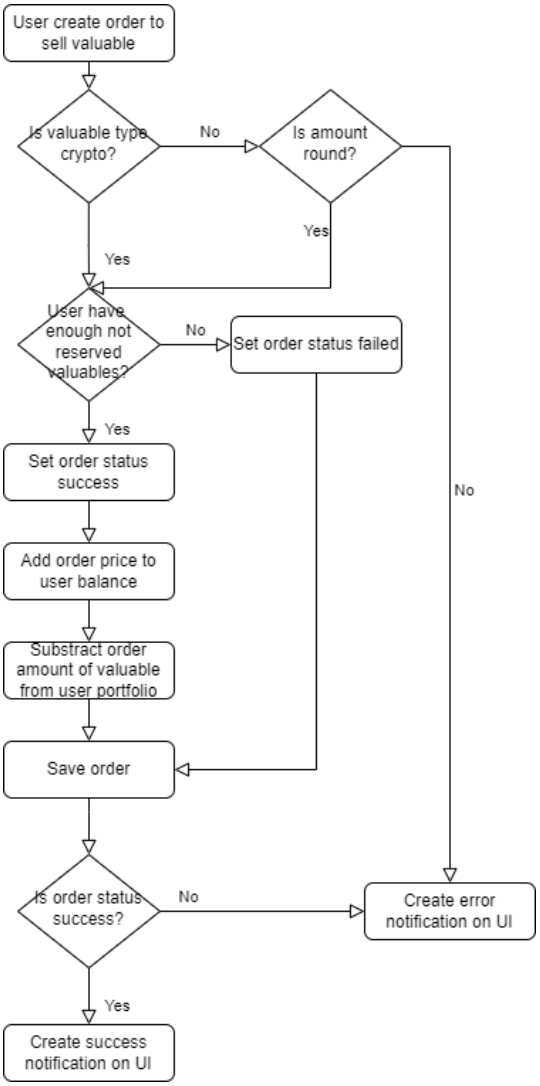


Рис. 2.6 Алгоритм продажу

При створені підписки відбувається перевірка типу підписки і типу активу на яку вона створена. Якщо тип активу не криптовалюта і тип підписки не INFORM то перевіряється чи кількість у підписці є рівним числом, якщо ні підписка не створюється а користувач отримує повідомлення про помилку при створенні підписки. Якщо підписка типу SELL і встановлена опція reserve, то в портфоліо користувача резервується необхідна кількість цього ФА яка потім не може бути продана і буде використана для цієї підписки. При успішному створеню підписки користувач отримує повідомлення про успішне створення підписки.

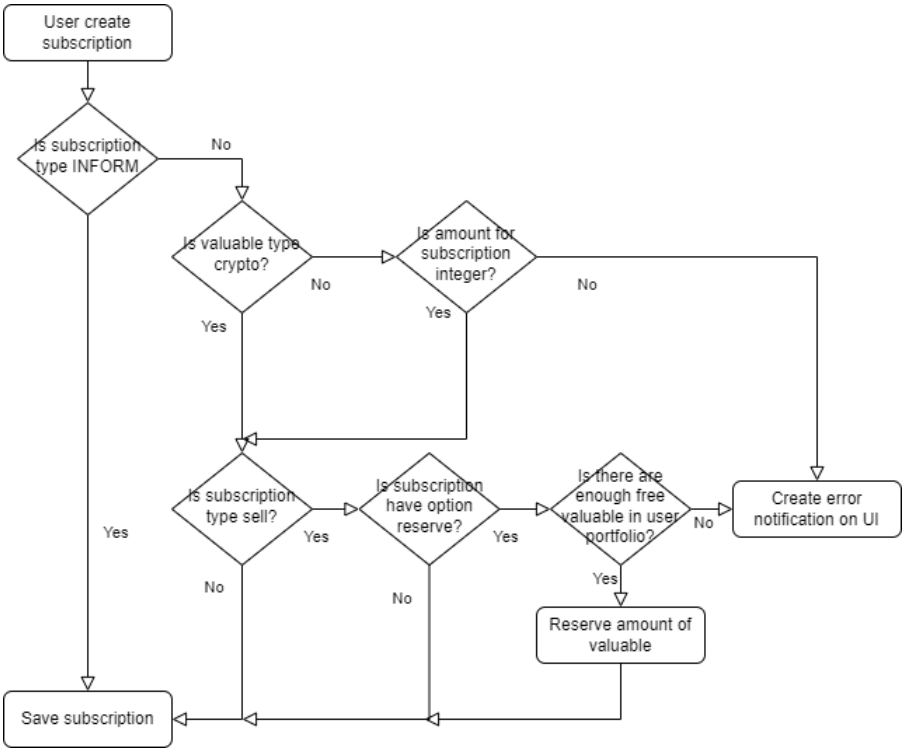


Рис. 2.7 Алгоритм створення підписки

При необхідності підписку можливо видалити і якщо підписка яка видаляється відноситься до типу SELL і має опцію резервації то резервація відміняється.

Система постійно оновлює поточні данні ФА щодо поточної ціни, тренду та останьої дати і часу оновлення завдяки отриманню інформації з зовнішнього ресурсу, після чого запускає обробку підписок які до нього відносяться. Після цього проводиться фільтрація для того щоб обробити лише ті підписки умови яких виконуються. Додатково при оновленні в системі зберігається історія щодо зміни ціни ФА.

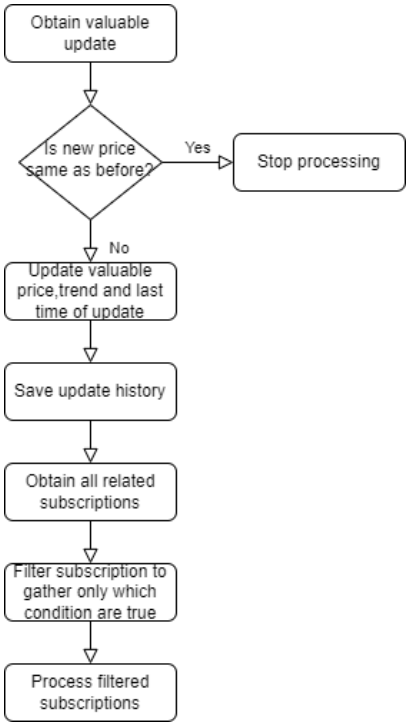


Рис. 2.8 Алгоритм оновлення інформації та запуску обробки підписок

У системі є три типи підписок кожна з яких оброблюється властним алгоритмом. Найлегша в плані обробки підписка належить до типу INFORM. Вона призначена для інформування завдяки повідомленю користувача про те що ФА відповідає необхіднии йому умовам. Також підписка типу INFORM може працювати постійно а не лише один раз якщо при її створенні була встановлена опція continuos.

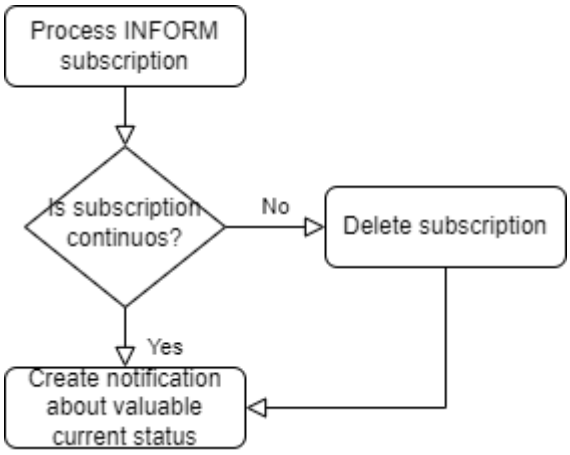


Рис. 2.9 Алгоритм обробки підписки типу INFORM

Підписка типу BUY починає автоматичну закупку ФА як при звичайній покупці. Якщо при покупці сталася помилка(наприклад у користувача недостатньо балансу) то створюється повідомлення про помилку при спрацювані підписки для користувача та підписка видаляється або не видаляється якщо при створенні була встановлена опція failSafe. При успішному виконанні користувач отримує повідомлення про успішне спрацювання підписки.

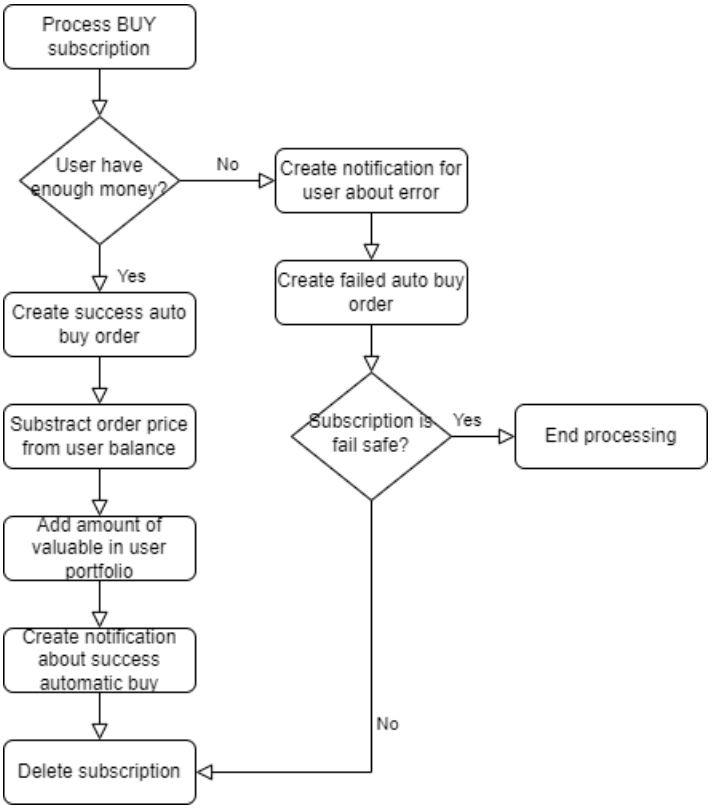


Рис. 2.10 Алгоритм обробки підписки типу BUY

Підписка типу SELL починає автоматичний продаж необхідної кількості ФА з портфоліо користувача. Обробка автоматичної покупки аналогічна до звичайної покупки. При помилці в обробці користувач отримує повідомлення про помилку та підписка видаляється або не видаляється при встановлені опції failSafe. Якщо підписка успішно відпрацювала користувач отримує повідомлення про успішне відпрацювання підписки.

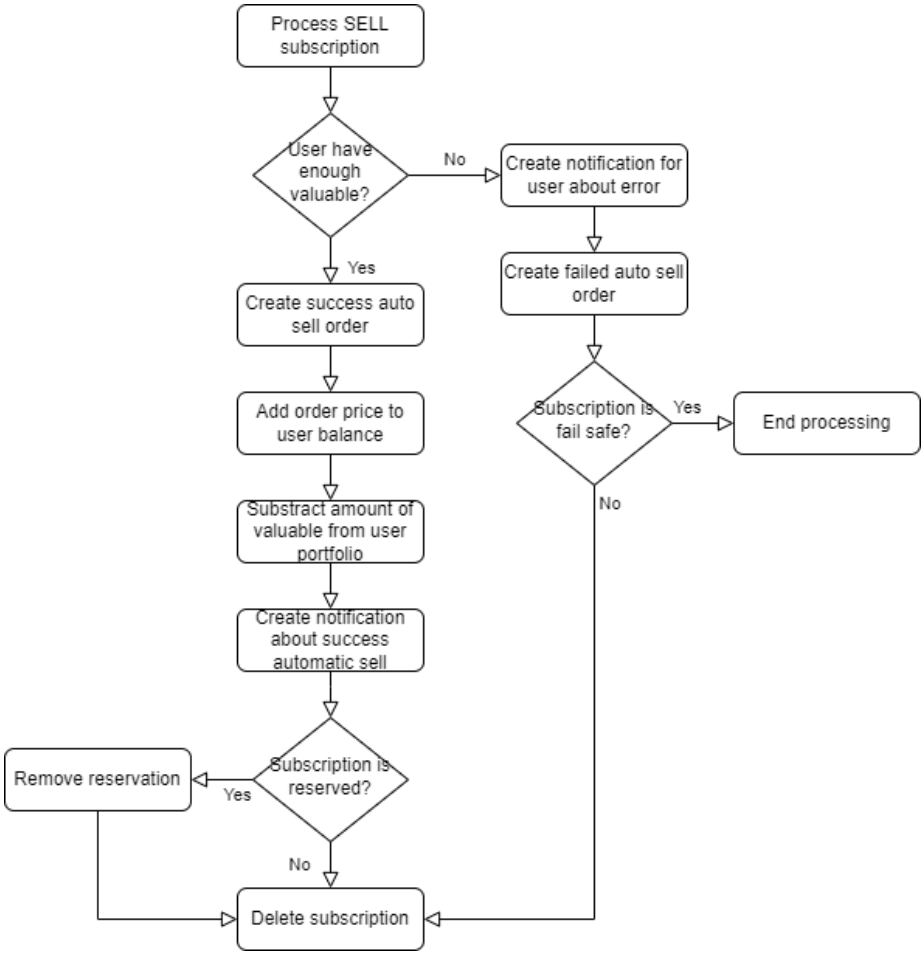


Рис. 2.11 Алгоритм обробки підписки типу SELL

2.5. Обгрунтування та організація вхідних та вихідних даних програми

В даній кваліфікаційній роботі головними вхідними данними що постійно оброблюються є данні з зовнішнього ресурса. Такі данні приходять у форматі JSON який містить велику кількість різної інформації серед якої використовується інформація щодо символу активу та його поточна ціну. При обробці таких данних в базі данних шукається актив с таким же символом і оновлюється поточна ціна, різниця з попередньою цінною(тренд) та дата останього оновлення. На виході ми отримаємо оновленний актив з новою цінною, датою обробкою та трендом.

Також додаток оброблює підписки які відносять до оновленого ФА, після чого на виході ми отримаємо повідомлення, нові замовлення, та оновлені баланси користувачів та їх портфоліо.

Щодо вхідних данних які система отримає безпосередньо від користувача завдяки графічному інтерфейсу, до таких належать наступні данні: для регестрації(електрона адресса, пароль, ім’я та прізвище) та авторизації(електрона адреса та пароль), символ для пошуку активу, кількість активу для покупки/продажі, данні для створення підписки(кількість активу, умова спрацювання і її ціни, дії при спрацювані) та данні банківської картки для проведення операції з рахунком користувача.

До вихідних данних відносяться поточні данні ФА(ціна, тип, символ, ім’я, дата та час останьої обробки, тренд) та історія зміни їх ціни(дата та час змінни, стара ціна, нова ціна, різниця), данні користувача(баланс, ім’я , прізвище, електрона почта), замволення користувача(сума замовлення, тип замовлення, статус замовлення, актив замовлення, ціна одиниці на момент замовлення), його портфоліо(усі наявні ФА, їх кількість, поточна ціна, зарезервована кількість, символ, тренд), проведені операції з рахунком(сума операції, дата проведення, тип операції, номер картки) та данні збережених банківських карток а також повідомлення користувача.

2.6. Опис розробленого програмного продукту

2.6.1. Використані технічні засоби

При розробці та тестувані роботи програми була використана персональний ноутбук від компанії Hewlett-Packard(HP), серії ELITEBOOK та моделі 850 G7 на основі операційної системи Windows 10 pro з наступними характеристиками:

– Процесор Intel Core i7-1185G7;

– Відеопроцесор Intel Iris Xe Graphics G7 96EUs (400-1350 mhz);

– Оперативна пам'ять 32 ГБ DDR4(3200 mhz).

2.6.2. Використані програмні засоби

Додаток було написано в середовищі розробки компанії JetBrains - Intellij Idea Ultimate 2022 року Studio на основі ліцензії для студентів, що надає інтегровані інструменти для розробки та налагодження додатків на основі Java. У процесі тестування веб додатку був також використаний веб браузер від компанії Google – Google Chrome. Для створення властних данних для оновлення інформації активів та запуску обробки підписок було використано програмне забезпечення Postman. Postman – додаток для створення властних повністю налаштовуємих HTTP запитів. Також для роботи локальної бази данних, її налагодження та перегляду було використано програмне забезпечення MySQL WorkBench – додаток для розробки та адміністрування СУБД MySQL.

2.6.3. Виклик та завантаження програми

Для запуску програми необхідно мати програмне забезпечення для запуску додатків Java – JRE. Запуск програми виконується завдяки консолі та команді “java” в якості параметра передаемо ім’я файлу скомпільованого додатку. Для повноціної роботи системи потрібно запустити два додатки.

2.6.4. Опис інтерфейсу користувача

Інтерфейсом користувача є веб додаток який доступний завдяки сторінці інтернет браузера.

Після переходу на веб адресу сервісу з’являється головна сторінка (рис. 2.12). Також зверху знаходиться навігаційний бар який відображається

на кожній сторінці додатку. На головній сторінці відображаються усі доступні ФА та інформація про них(ім’я, символ, дата оновлення, тип активу, ціна). Додатково на сторінці можливо шукати необхідний актив за символом.

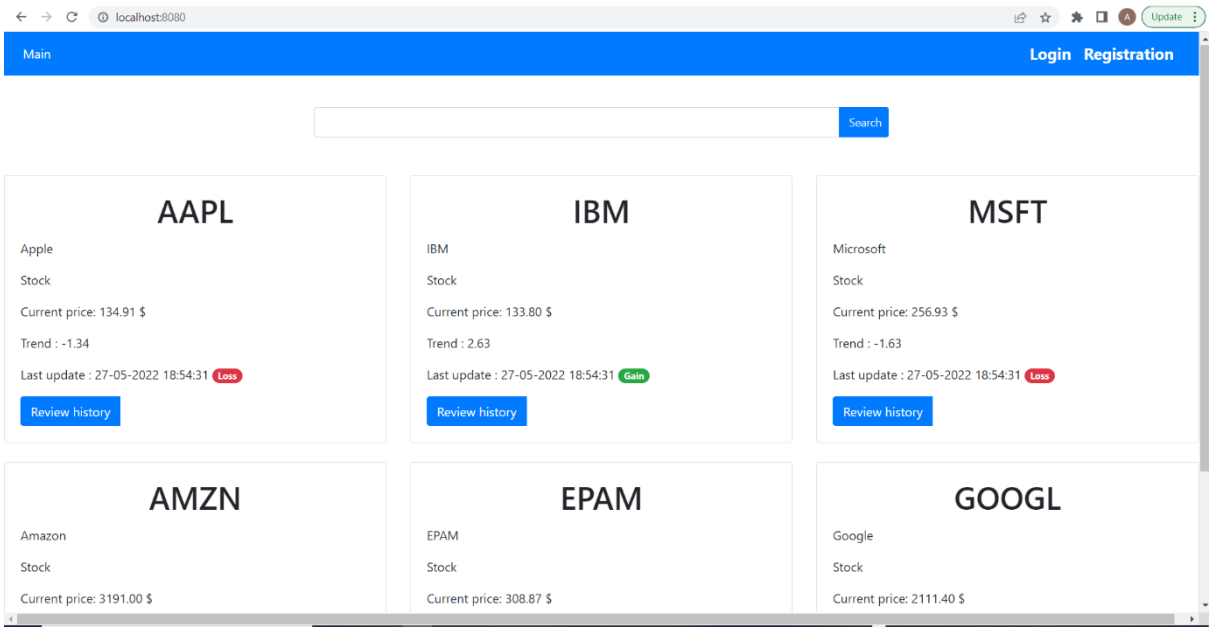
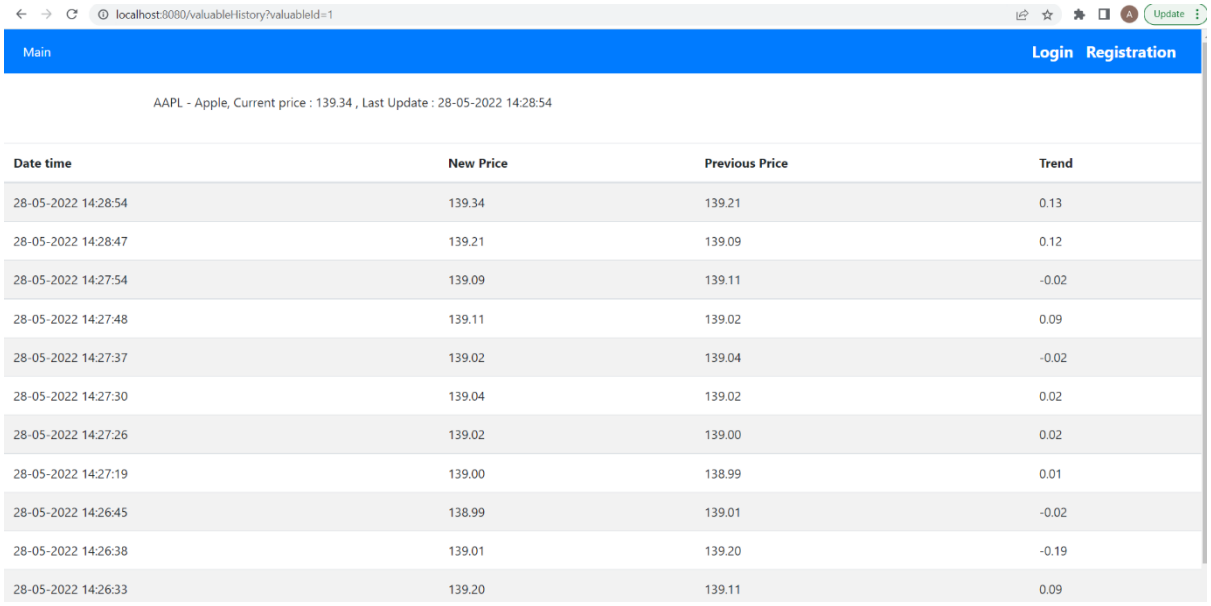


Рис. 2.12 Головна сторінка(вигляд для не авторизованого користувача)

Під кожним ФА можна знайти кнопку для переходу на сторінку історії зміни ціни (рис. 2.13) - Review history



Date time	New Price	Previous Price	Trend
28-05-2022 14:28:54	139.34	139.21	0.13
28-05-2022 14:28:47	139.21	139.09	0.12
28-05-2022 14:27:54	139.09	139.11	-0.02
28-05-2022 14:27:48	139.11	139.02	0.09
28-05-2022 14:27:37	139.02	139.04	-0.02
28-05-2022 14:27:30	139.04	139.02	0.02
28-05-2022 14:27:26	139.02	139.00	0.02
28-05-2022 14:27:19	139.00	138.99	0.01
28-05-2022 14:26:45	138.99	139.01	-0.02
28-05-2022 14:26:38	139.01	139.20	-0.19
28-05-2022 14:26:33	139.20	139.11	0.09

Рис. 2.13 Приклад сторінки історії активу

Також неавторізований користувач має можливість перейти на сторінку реєстрації та сторінку авторизації завдяки кнопкам у правому верхньому куті навігаційного бару Registration та Login, або повернутися назад на головну сторінку завдяки кнопці Main у лівому верхньому куті.

На сторінці реєстрації (рис. 2.14) можливо зареєструвати нового користувача надавши електрону почту , логін, пароль, ім’я та прізвище користувача.

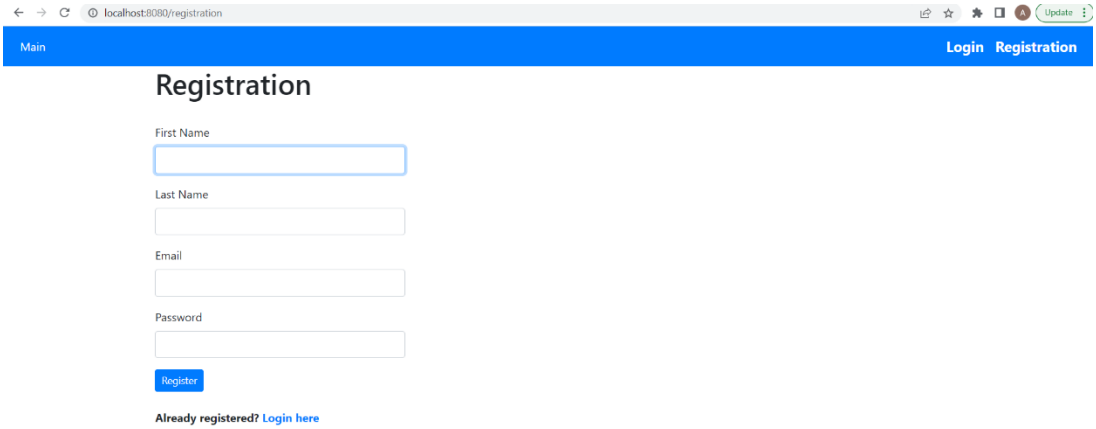


Рис. 2.14 Сторінка регестрації

Через сторінку авторизації (рис. 2.15) можливо скористатися авторизацією завдяки сервісу гугл або локальною через електрону почту та пароль. Також є пряме посилання на реєстрацію, якщо користувач не має аккаунта.

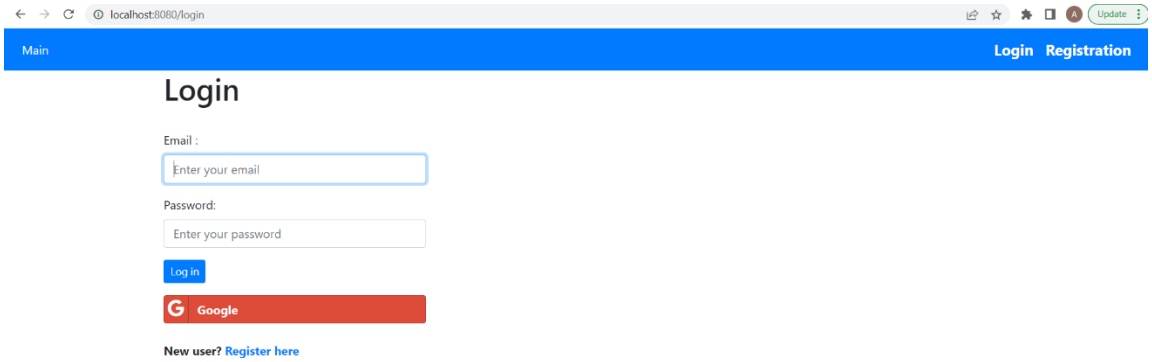


Рис. 2.15 Сторінка авторизації

Для авторизованого користувача головна сторінка та навігаційний бар мають інший вигляд(рис. 2.16).

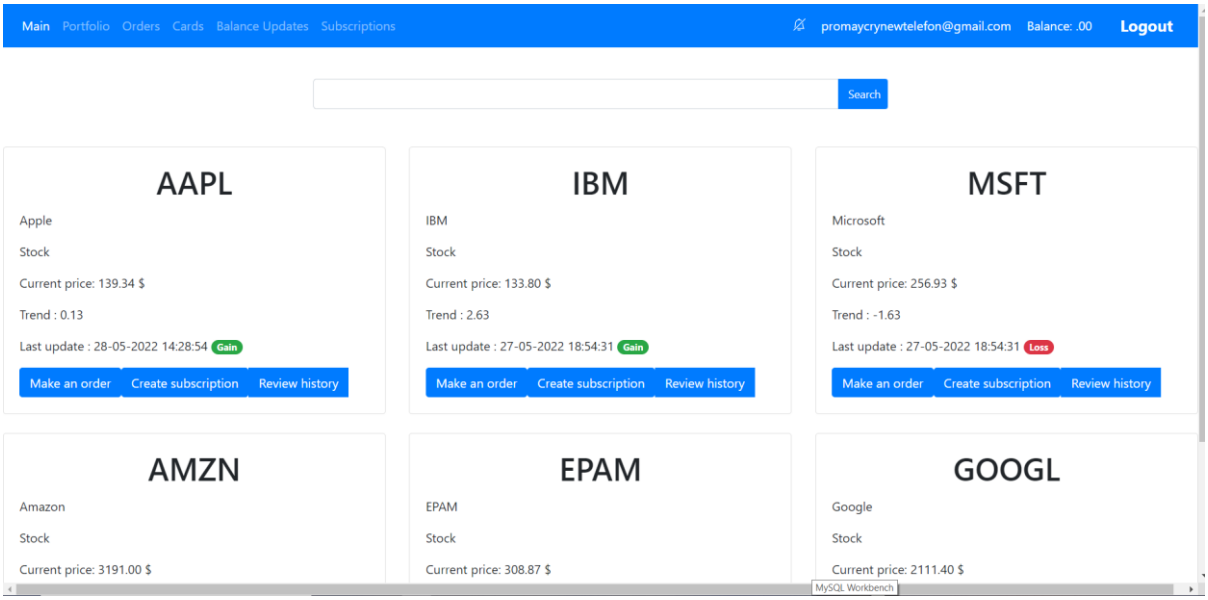


Рис. 2.16 Головна сторінка та навігаційний бар(вид для авторизованого користувача)

Під кожним активом додатково з'являються кнопки створення замовлення –Make an order та створення підписки –Create subscription.

На сторінці створення замовлення(рис. 2.17) можливо побачити поточний ФА а також вибрати тип замовлення та кількість для покупки/продажі і створити замовлення.

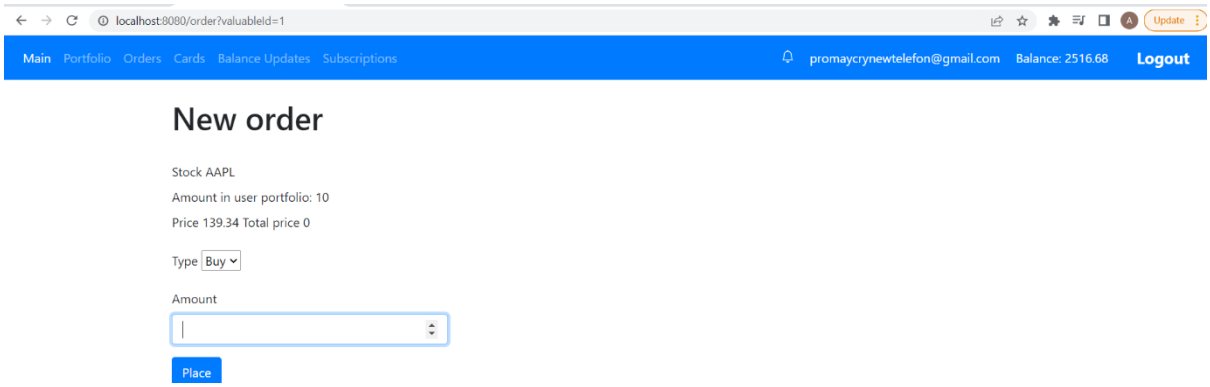


Рис. 2.17 Сторінка створення нового замовлення

На сторінці створення підписки(рис.- 2.18) можливо вибрати тип підписки та умови її спрацювання.

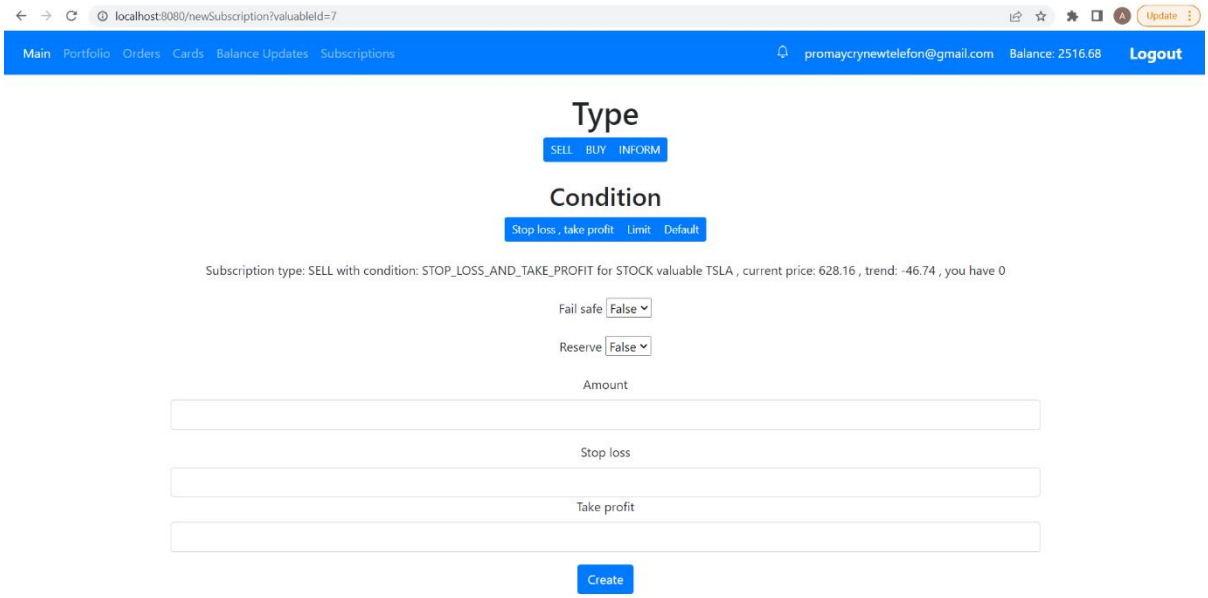


Рис. 2.18 Сторінка створення нової підписки

Завдяки розширеному навігаційному бару можливо перейти на додаткові сторінки, які доступні тільки для авторизованих користувачів.На сторінці за кнопкою навігаційного бару - Portfolio можливо знайти усі ЦП та криптовалюти поточного користувача які є в його наявності(рис. 2.19) та перейти на сторінки створення підписки та замовлення для окремого активу.

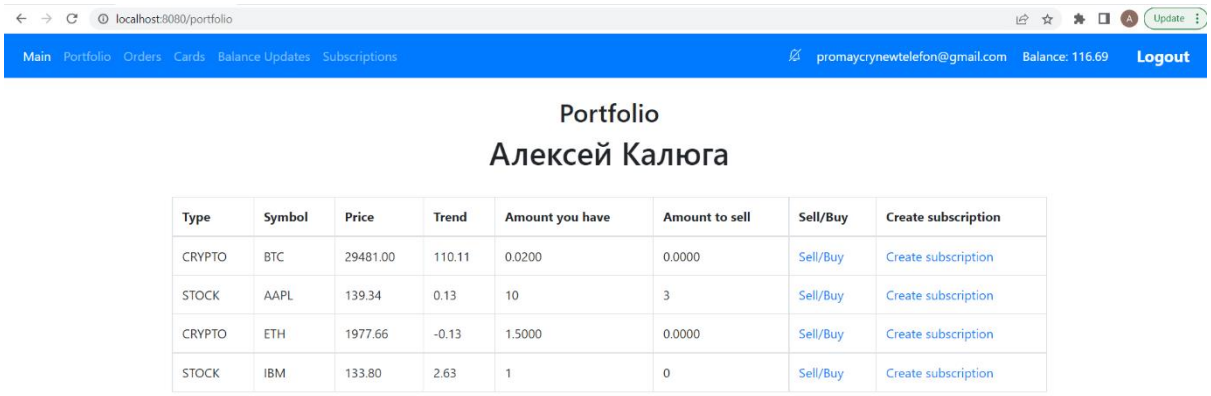


Рис. 2.19 Портфіліо користувача

Сторінка Orders містить усі автоматичні та звичайні замовлення користувача з детальною інформацію про них(рис. 2.20).

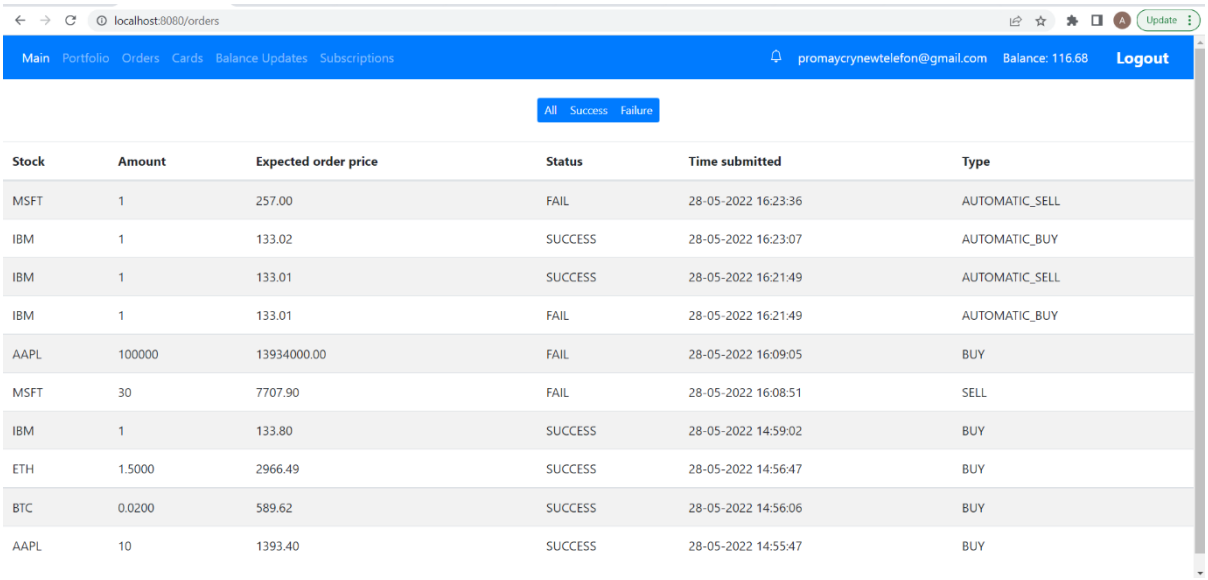


Рис. 2.20 Сторінка усіх замовлень користувача

Сторінка Cards відображає усі збережені банківські картки користувача(рис. 2.21), а також дає можливість використати їх для операції з рахунком, видалити, змінити, додати або скористатися новою картк оз подальшим її збереженням або без.

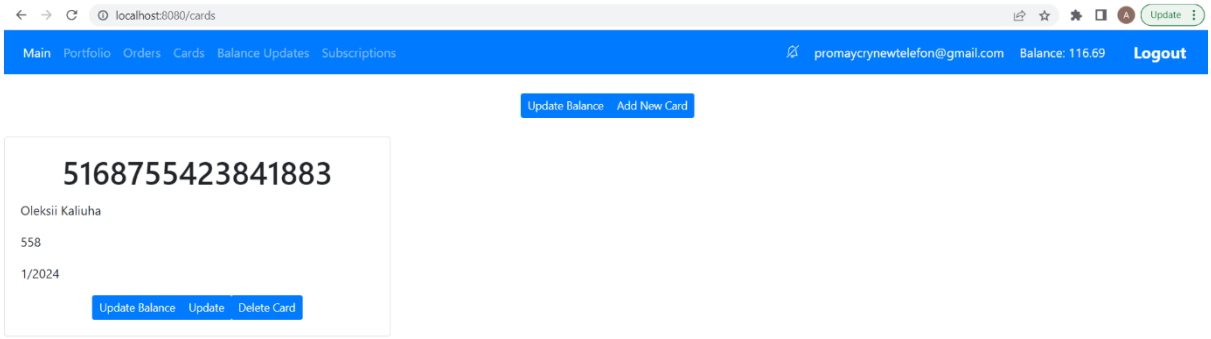


Рис. 2.21 Сторінка банківських карток користувача

Для сбереження нової картки необхідно натиснути Add New Card, що переведе нас на сторінку для додавання картки(рис. 2.22)

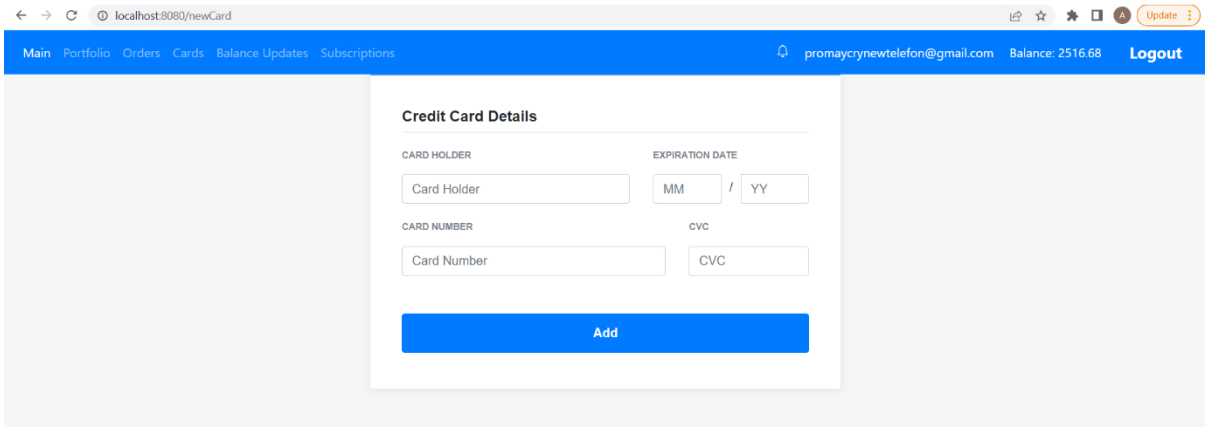


Рис. 2.22 Сторінка збереження нової картки

При натискані Update на картці можливо оновити данні картки на окремій сторінці(рис. 2.23).

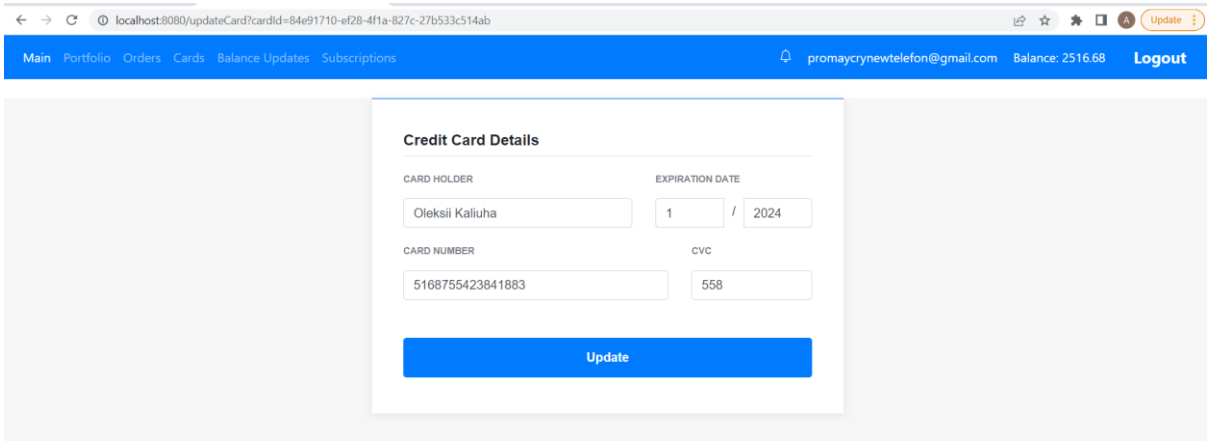


Рис. 2.23 Сторінка оновлення картки

При натискані на Update Balance на окремій картці нас переведе на сторінку роботи з рахунком користувача(рис. 2.24) з вже введеними даними поточної картки які можна використати для операції з рахунком.

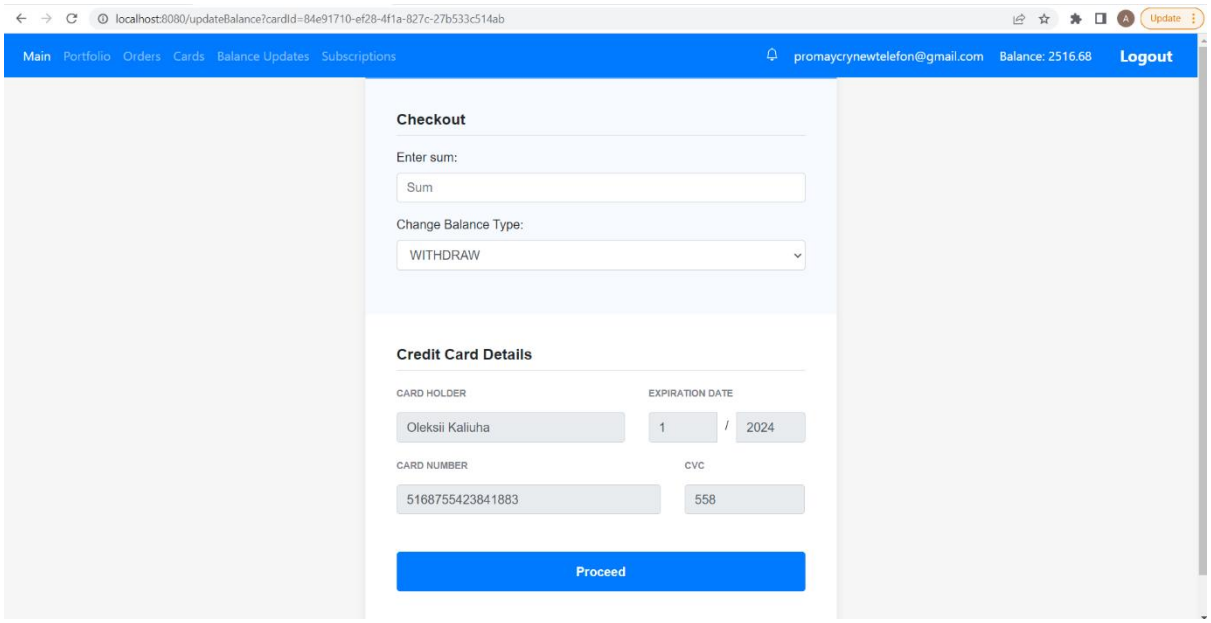


Рис. 2.24 Сторінка проведення операції з балансом рахунку з використанням збереженої картки

При натискані Update Balance не на картці нас переведе на сторінку роботи з рахунком користувача(рис. 2.25) без данних збереженої картки.

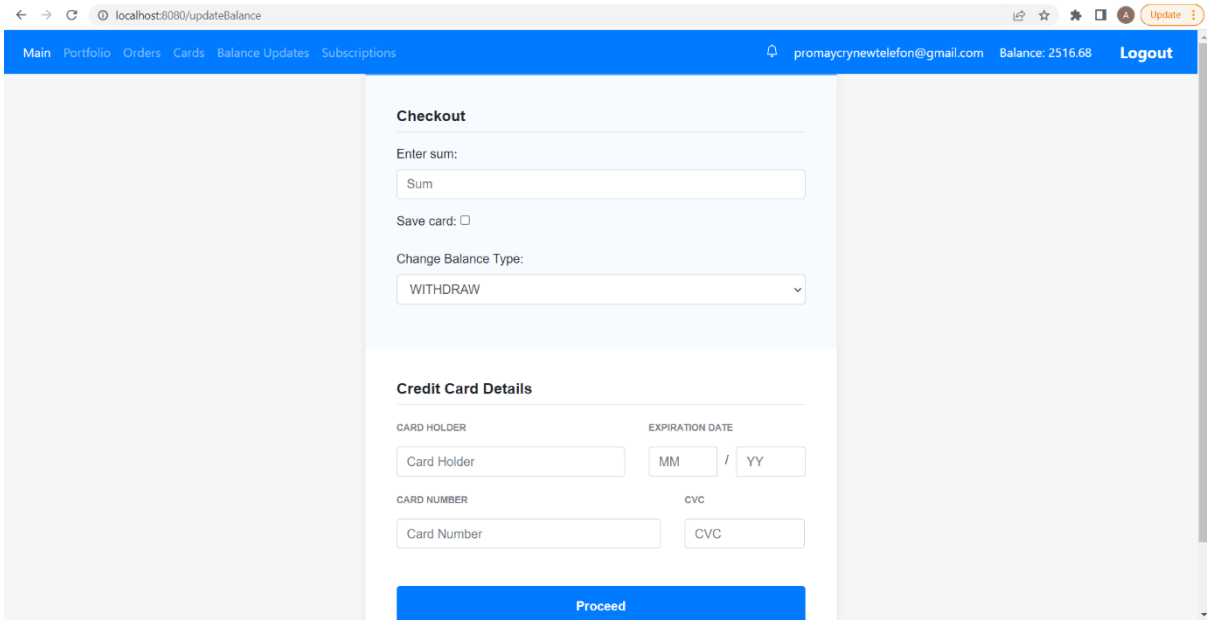
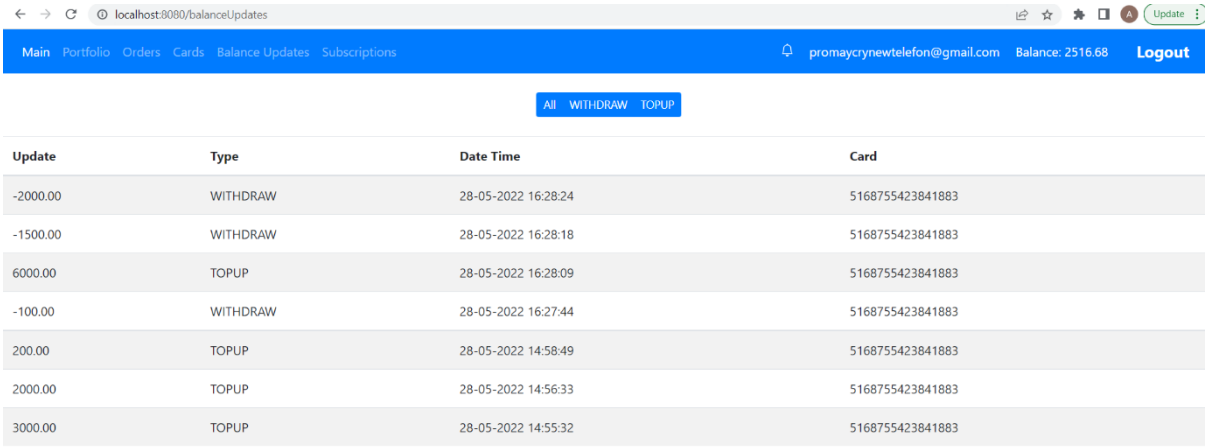


Рис. 2.25 Сторінка проведення операції без збереженої картки

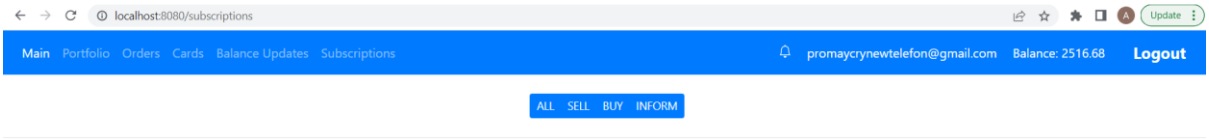
Наступною на навігаційному барі є сторінка Balance Updates знаходяться усі проведені операції з поповнення та зняття коштів з рахунку(рис. 2.26), а також можливість окремо відображати тільки поповнення або зняття коштів з рахунку.



Update	Type	Date Time	Card
-2000.00	WITHDRAW	28-05-2022 16:28:24	5168755423841883
-1500.00	WITHDRAW	28-05-2022 16:28:18	5168755423841883
6000.00	TOPUP	28-05-2022 16:28:09	5168755423841883
-100.00	WITHDRAW	28-05-2022 16:27:44	5168755423841883
200.00	TOPUP	28-05-2022 14:58:49	5168755423841883
2000.00	TOPUP	28-05-2022 14:56:33	5168755423841883
3000.00	TOPUP	28-05-2022 14:55:32	5168755423841883

Рис. 2.26 Сторінка усіх проведених операції з рахунком користувача

Сторінка Subscriptions надає перелік усіх підписок поточного користувача та інформацію про них(рис. 2.27), отримати лише окремий тип підписок а також можливість видалити необхідну підписку завдяки кнопці Delete.



Valuable	Valuable Type	Type	Amount	Condition	Fail safe	Continuos	Reserve	Delete
EPAM	STOCK	INFORM	0	When price in limit from 300.00 to 350.11	false	true	false	Delete
IBM	STOCK	BUY	30	When price are LOWER than 111.00	true	false	false	Delete
BTC	CRYPTO	BUY	0.4000	When price are GREATER_AND_EQUALS than 30000.00	true	false	false	Delete
AAPL	STOCK	BUY	3	When price are GREATER than 140.00	false	false	false	Delete
IBM	STOCK	BUY	30	When price are LOWER_AND_EQUALS than 120.00	false	false	false	Delete
AAPL	STOCK	SELL	5	When price in limit from 120.11 to 129.47	false	false	true	Delete
MSFT	STOCK	SELL	4	When price lower-equals to 240.00 and bigger-equals to 280.00	false	false	false	Delete
AAPL	STOCK	INFORM	0	When price are EQUALS than 138.30	false	false	false	Delete

Рис. 2.27 Сторінка підписок користувача

Останьою є сторінка повідомлень користувача(рис. 2.28) яка доступна за значком колокольчика у правій стороні навігаційного бару. Якщо колокольчик перекреслений то повідомлень немає, в іншому випадку у користувача є повідомлення для перегляду та підтвердження.

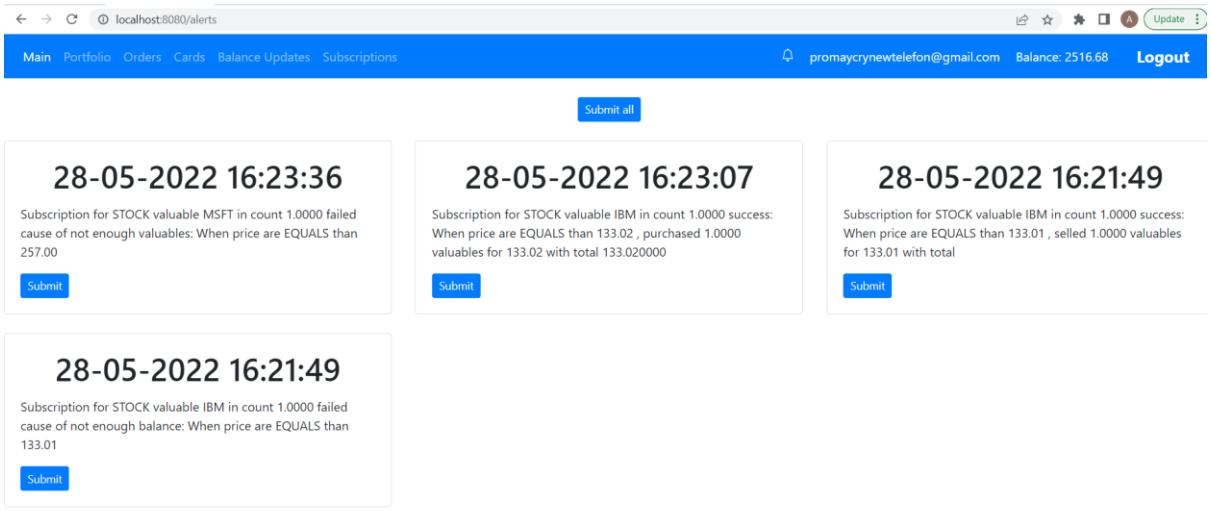


Рис. 2.28 Сторінка повідомлень користувача

На сторінці повідомлень можливо переглянути усі повідомлення а також підвердити перегляд усіх повідомлень за допомогою кнопки Submit All або лише окремого за допомогою кнопк Submit .

Також при помилці у роботі додатку користувач отримує червоне повідомлення у верхній частині додатку(рис. 2.29). Аналогічно при створенні підписки, успішній роботі з рахунком або успішному проведені операції з активами користувач отримує повідомлення зеленого кольору(рис. 2.30)

Рис. 2.29 Приклад повідомлення про помилку (під час покупки активу)

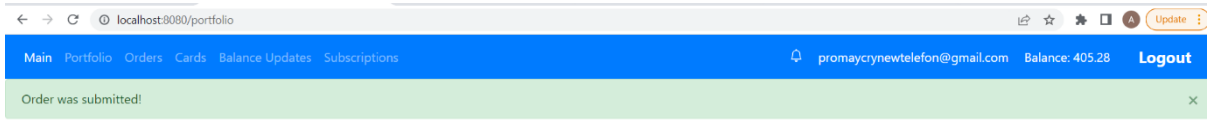


Рис. 2.30 Приклад повідомлення про успішну обробку(успішне замовлення)

РОЗДІЛ 3 ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми - 800;
2. коефіцієнт складності програми – 1,5;
3. коефіцієнт корекції програми в ході її розробки – 0,08;
4. годинна заробітна розробника Java – 167 грн/год;
5. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,8;
6. вартість машино-години ЕОМ – 14 грн/год

Середня година зарплати розробника Java була вирахувана виходячи з даних «Української спільноти програмістів (DOU)». Середньоукраїнська заробітна плата розробника, який володіє навичками розробки на мові програмування Java з досвідом роботи близько року дорівнює 1000 американських доларів у місяць. При курсі валют НБУ на початок червня 2022 року один американський долар дорівнює 29,34 грн, тому середня зарплата в гривнях дорівнює 29340 грн. При восьмигодинному робочому дні (176 годин у місяць в середньому) середня зарплата за годину буде становити 166.7 грн.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$– витрати праці на дослідження алгоритму рішення задачі;

$t_а$– витрати праці на розробку блок-схеми алгоритму;

$t_п$– витрати праці на програмування по готовій блок-схемі;

$t_{отл}$– витрати праці на налагодження програми на ЕОМ;

$t_д$– витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q=q\cdot C\cdot(1+p), \text{ де} \qquad (3.2)$$

q– передбачуване число операторів;

C– коефіцієнт складності програми;

p– коефіцієнт кореляції програми в ході її розробки.

$$Q= 800\cdot 1,5\cdot(1+0,08)=1296;$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_и = \frac{Q \bullet B}{(75\ldots85) \bullet K}, \text{ людино-годин,} \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$$t_и = \frac{1296 \cdot 1,5}{80 \cdot 0,8} = 30.37 \text{ людино} - \text{годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_п = \frac{Q}{(20\ldots25)\cdot k}, \text{ людино-годин.} \qquad (3.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), людино-годин:

$t_a = \frac{1296}{20 \cdot 0{,}8} = 81$, людино-годин.

Витрати на складання програми по готовій блок-схемі:

$t_п = \frac{Q}{(20 \ldots 25) \cdot k}$, людино-годин. (3.5)

$t_a = \frac{1296}{25 \cdot 0{,}8} = 64{,}8$ людино-годин.

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$t_{отл} = \frac{Q}{(4..5) \cdot k}$, людино-годин. (3.6)

$t_n = \frac{1296}{4 \cdot 0{,}8} = 405$, людино-годин,

- за умови комплексного налагодження завдання:

$t^{к}_{отл} = 1{,}2 \cdot t_{отл}$; (3.7)

$t^{к}_{отл} = 1{,}2 \cdot 360 = 486$ людино-годин

Витрати праці на підготовку документації:

$t_д = t_{дp} + t_{до}$; (3.8)

де $t_{дp}$ – трудомісткість підготовки матеріалів і рукопису

$t_{дp} = \frac{Q}{(15 \ldots 20) \cdot K}$; (3.9)

$t_{дp} = \frac{1296}{15 \cdot 0{,}8} = 108$ людино-годин.

$t_{до}$– трудомісткість редагування, печатки й оформлення документації

$t_{до} = 0{,}75 \cdot t_{дp}$; (3.10)

$t_{до} = 0{,}75 \cdot 96 = 81$, людино-годин.

$t_д = 108 + 81 = 189$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t =50+30,37+81+64,8+486+189=901,17, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 901,17 людино-годин для розробки даного сервісу.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$– середня годинна заробітна плата програміста, грн/година

$З_{ЗП} = 901{,}17 \cdot 167 = 150495{,}95$, грн.

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{Мч}$– вартість машино-години ЕОМ, грн/год.

$З_{МВ} = 901{,}17 \cdot 14 = 12616{,}38$, грн.

$К_{ПО} = 150495{,}95 + 12616{,}38 = 163122{,}33$, грн.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \qquad (3.14)$$

де B_k- число виконавців;

F_p– місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$T = \frac{901,17}{1 \cdot 176} = 5,12$ міс.

На розробку даного сервісу піде 901,17 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 5,12 місяців при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення сервісу складатимуть 150495,95 грн.

ВИСНОВКИ

Метою кваліфікаційної роботи було розробити сервіс для моніторингу ЦП та криптовалют а також можливістю проведення операції над ними. Розробкою буде зацікевлені особи які цікавлені сферою інвестиції, ФР, ЦП і криптовалютами а також той хто бажає спробувати себе в досліджені, аналізу та роботі у цій сфері з використанням розробленого ресурсу.

Дослідження сфери інвестиції дуже складне та цікаве. Інвестиції дуже важливі в сучасному світу, оскільки дозволяють отримувати додатковий прибуток або зберігати свій капітал від знецінення. Сервіс розроблений у кваліфікаційні роботі дозволить робити це у зрозумілому та приємному для користувача вигляді. Додатково забезпечить користувача усім необхідним функціоналом для проведення операцій з покупки та продажу ФА, а також автоматичного реагування на необхідні користувачу умови.

Розроблена система призначена для моніторингу та відстежування змін на біржі з використанням зовнішнього ресурсу. Система надає доступ до поточних данних щодо ціни фінансових активів, а також історії її змін.

Також система надає можливість авторизованим користувачам проводити ФО для продажу та покупки ЦП або криптовалют, створювати автоматичні реакції на необхідні умови користувачу умови, такі як автоматичний продаж чи покупка ФА, або звичайне повідомлення користувача при спрацюванні.

Основною метою для створення цього сервісу було надання можливостей для роботи та моніторингу біржі звичайним користувачам глобальної мережі інтернет а також створення конкуренції для аналогічних сервісів у сфері інвестиції.

Створена система дозволить своїм користувач зберігати свії капітал від знецінення, диверсифікувати та примножувати його. Завдяки даним з сервісу можливо проводити аналіз та робити висновки виходячи з

отриманої інформації. Отримані знання можуть бути використані для планування чи проведення операції з ФА, їх дослідження та збереження.

Система існує у вигляді двох додатків, де перший – веб-додаток виступає у ролі графічного інтерфейсу та оброблює усі дії користувача, а другий – проводить постійне оновлення та обробку данних у системі.

Сервіс був розроблений завдяки мові програмування Java, фреймворку Spring, та іншими додатковим бібліотекам та фреймворкам. Для збереження данних була використана база данних MySQL автоматично створена та налаштована засобами Liquibase. Безпека у веб-додатку була налаштована програмними засобами фреймворку Spring.

Під час виконання даного кваліфікаційної роботи були виконані наступні задачі:

- проаналізовано предметну область задачі, що розв’язується;
- проведено проектування системи;
- обрано раціональну структуру сервісу;
- створена належна структура бази данних;
- створено зрозумілий графічний інтерфейс;
- налагоджені алгоритми роботи та функціонування системи;
- розроблено необхідний функціонал з використанням стандартів.

Також у кваліфікаційній роботі було визначено трудомісткість розробленого програмного продукту (901.17 люд-год), проведений підрахунок вартості роботи по створенню програми (150495,95) грн. та розраховано час на його створення (5,12 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мозговий О.М. Фондовий ринок . Навч. посібник. — К.: КНЕУ. 1999.— 316 с.
2. Investopedia “Forces that move stock prices”. URL: https://www.investopedia.com/articles/basics/04/100804.asp
3. TheConversetion. “Bitcoin: why the price has exploded – and where it goes from here”. URL: https://theconversation.com/bitcoin-why-the-price-has-exploded-and-where-it-goes-from-here-152765
4. Leander Kahney. Jony Ive: The Genius Behind Apple's Greatest Products Book. 2013 – 349 с.
5. Investopedia. “A History of apple stock increases”. URL: https://www.investopedia.com/articles/stocks/12/history-apple-stock-increases.asp
6. Dividend.com. “How does apple stock react o product releases”. URL: https://www.dividend.com/how-to-invest/how-does-apple-stock-react-to-product-releases/
7. IGeeksBlog. “Apple marketing strategy”. URL: https://www.igeeksblog.com/apple-marketing-strategy/
8. Bruce Eckel. Thinking in Java. 4th edition. 2006 – 1150 с
9. Craig Wall and Ryan Breidenbach. Spring in Action. Fifth Edition. 2018 – 520 с.
10. JavatPoint. “Spring MVC tutorial”. URL: https://www.javatpoint.com/spring-mvc-tutorial
11. Geeks for Geeks. “What is web socket and how it is different from HTTP” URL: https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/
12. Javarush. “Дружим обычный вход через email и OAuth2 в Spring Security на примере сервиса заметок”. URL:

https://javarush.ru/groups/posts/2269-druzhim-obihchnihy-vkhod-cherez-email-i-oauth2-v-spring-security-na-primere-servisa-zametok

13. Spring docs. “Aspect Orienting Programming with Spring”. URL: https://docs.spring.io/spring-framework/docs/2.5.x/reference/aop.html

14. Antonio Sansa, Justin Richer. OAuth2 In Action. 2017 – 461 с

15. Reflectoring.IO. “When to use Lombok” URL: https://reflectoring.io/when-to-use-lombok/

16. Jennifer Niederst Robbins. Learning Web Design. 2012 – 801 с

17. Baeldung. “Introduction to Using Thymeleaf in Spring”. URL: https://www.baeldung.com/thymeleaf-in-spring-mvc

18. Raghuram Bharathan. Apache Maven Cookbook. 2015 – 272 с

19. Habr. “Версіонування структури БД за допомогою Liquibase” URL: https://habr.com/ru/post/548882/

20. Christian Bauer, Gary Gregory, Gaving King. Java persistence with Hibernate. 2006 – 876 с

21. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. - СПб.: КОРОНА принт, 2008. - 416с

22. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” Викладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

ДОДАТОК А

КОД ПРОГРАМИ
Liquibase скрипт для бази данних:

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
      xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
                  http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd">

   <changeSet id="1" author="Oleksii Kaliuha">
      <createTable tableName="user">
         <column name="id" type="char(36)">
            <constraints primaryKey="true"/>
         </column>
         <column name="email" type="varchar(45)">
            <constraints nullable="false" unique="true"/>
         </column>
         <column name="actual_email" type="varchar(45)">
            <constraints nullable="false" unique="true"/>
         </column>
         <column name="first_name" type="nvarchar(45)">
            <constraints nullable="false"/>
         </column>
         <column name="last_name" type="nvarchar(45)">
            <constraints nullable="false"/>
         </column>
         <column name="balance" type="DECIMAL(12, 2)"/>
         <column name="role" type="varchar(10)" defaultValue="USER">
            <constraints nullable="false"/>
         </column>
         <column name="password" type="char(60)"/>
         <column name="is_blocked" type="boolean" defaultValue="false">
            <constraints nullable="false"/>
         </column>
      </createTable>
   </changeSet>

   <changeSet id="2" author="Oleksii Kaliuha">
      <createTable tableName="valuable">
         <column name="id" type="char(36)">
            <constraints primaryKey="true"/>
         </column>
         <column name="symbol" type="varchar(16)">
            <constraints nullable="false" unique="true"/>
         </column>
         <column name="name" type="char(60)"/>
         <column name="price" type="DECIMAL(12, 2)"/>
         <column name="trend" type="DECIMAL(12, 2)"/>
         <column name="type" type="varchar(10)">
            <constraints nullable="false"/>
         </column>
         <column name="last_update" type="DATETIME(6)">
            <constraints nullable="false"/>
         </column>
      </createTable>
   </changeSet>
```

```
<changeSet id="3" author="Oleksii Kaliuha">
   <createTable tableName="orders">
      <column name="id" type="char(36)">
         <constraints primaryKey="true"/>
      </column>
      <column name="user_id" type="char(36)">
         <constraints nullable="false"/>
      </column>
      <column name="valuable_id" type="char(36)">
         <constraints nullable="false"/>
      </column>
      <column name="amount" type="DECIMAL(12, 4)"/>
      <column name="valuable_price" type="DECIMAL(12, 2)"/>
      <column name="date_time" type="DATETIME(6)"/>
      <column name="order_price" type="DECIMAL(12, 2)"/>
      <column name="type" type="varchar(20)">
         <constraints nullable="false"/>
      </column>
      <column name="fail_description" type="varchar(255)"/>
      <column name="status" type="varchar(10)">
         <constraints nullable="false"/>
      </column>
   </createTable>
</changeSet>


<changeSet id="4" author="Oleksii Kaliuha">
   <createTable tableName="user_valuables">
      <column name="id" type="char(36)">
         <constraints primaryKey="true"/>
      </column>
      <column name="user_id" type="char(36)">
         <constraints nullable="false"/>
      </column>
      <column name="valuable_id" type="char(36)">
         <constraints nullable="false"/>
      </column>
      <column name="amount" type="DECIMAL(12, 4)"/>
      <column name="amount_to_sell" type="DECIMAL(12, 4)"/>
   </createTable>
</changeSet>

<changeSet id="5" author="Oleksii Kaliuha">
   <createTable tableName="card">
      <column name="id" type="char(36)">
         <constraints primaryKey="true"/>
      </column>
      <column name="card" type="bigint"/>
      <column name="card_holder" type="char(36)"/>
      <column name="expired_month" type="int" />
      <column name="expired_year" type="int" />
      <column name="cvc" type="int" />
      <column name="user_id" type="char(36)">
         <constraints nullable="false"/>
      </column>
   </createTable>
</changeSet>

<changeSet id="6" author="Oleksii Kaliuha">
```

```
    <createTable tableName="valuable_history">
        <column name="id" type="char(36)">
            <constraints primaryKey="true"/>
        </column>
        <column name="valuable_id" type="char(36)">
            <constraints nullable="false"/>
        </column>
        <column name="date_time" type="DATETIME(6)"/>
        <column name="previous_price" type="DECIMAL(12, 2)"/>
        <column name="new_price" type="DECIMAL(12, 2)"/>
        <column name="trend" type="DECIMAL(12, 2)"/>
    </createTable>
</changeSet>

<changeSet id="7" author="Oleksii Kaliuha">
    <createTable tableName="balance_update_history">
        <column name="id" type="char(36)">
            <constraints primaryKey="true"/>
        </column>
        <column name="user_id" type="char(36)">
            <constraints nullable="false"/>
        </column>
        <column name="date_time" type="DATETIME(6)">
            <constraints nullable="false"/>
        </column>
        <column name="update_type" type="varchar(30)">
            <constraints nullable="false"/>
        </column>
        <column name="balance_update" type="DECIMAL(12, 2)">
            <constraints nullable="false"/>
        </column>
        <column name="card" type="bigint">
            <constraints nullable="false"/>
        </column>
    </createTable>
</changeSet>

<changeSet id="8" author="Oleksii Kaliuha">
    <createTable tableName="subscription">
        <column name="id" type="char(36)">
            <constraints primaryKey="true"/>
        </column>
        <column name="fail_safe" type="bit(1)"/>
        <column name="amount" type="DECIMAL(12,4)"/>
        <column name="reserve" type="bit(1)"/>
        <column name="continuos" type="bit(1)"/>
        <column name="type" type="varchar2(20)">
            <constraints nullable="false"/>
        </column>
        <column name="condition_type" type="varchar(30)">
            <constraints nullable="false"/>
        </column>
        <column name="operator" type="varchar2(20)"/>
        <column name="price" type="DECIMAL(12,2)"/>
        <column name="low" type="DECIMAL(12,2)"/>
        <column name="high" type="DECIMAL(12,2)"/>
        <column name="user_id" type="char(36)">
            <constraints nullable="false"/>
        </column>
        <column name="valuable_id" type="char(36)">
```

```
            <constraints nullable="false"/>
        </column>
    </createTable>
</changeSet>

<changeSet id="9" author="Oleksii Kaliuha">
    <createTable tableName="alert">
        <column name="id" type="char(36)">
            <constraints primaryKey="true"/>
        </column>
        <column name="user_id" type="char(36)">
            <constraints nullable="false"/>
        </column>
        <column name="message" type="varchar2(255)">
            <constraints nullable="false"/>
        </column>
        <column name="date_time" type="DATETIME(6)">
            <constraints nullable="false"/>
        </column>
    </createTable>
</changeSet>


<changeSet id="10" author="Oleksii Kaliuha">
    <addForeignKeyConstraint baseColumnNames="user_id"
                             baseTableName="user_valuables"
                             constraintName="fk_user_id"
                             referencedColumnNames="id"
                             referencedTableName="user"/>
    <addForeignKeyConstraint baseColumnNames="valuable_id"
                             baseTableName="user_valuables"
                             constraintName="fk_valuable_id"
                             referencedColumnNames="id"
                             referencedTableName="valuable"/>
    <addForeignKeyConstraint baseColumnNames="user_id"
                             baseTableName="orders"
                             constraintName="fk_orders_user_id"
                             referencedColumnNames="id"
                             referencedTableName="user"/>
    <addForeignKeyConstraint baseColumnNames="valuable_id"
                             baseTableName="orders"
                             constraintName="fk_orders_valuable_id"
                             referencedColumnNames="id"
                             referencedTableName="valuable"/>
    <addForeignKeyConstraint baseColumnNames="user_id"
                             baseTableName="card"
                             constraintName="fk_card_user_id"
                             referencedColumnNames="id"
                             referencedTableName="user"/>
    <addForeignKeyConstraint baseColumnNames="user_id"
                             baseTableName="balance_update_history"
                             constraintName="fk_balance_update_history_user_id"
                             referencedColumnNames="id"
                             referencedTableName="user"/>
    <addForeignKeyConstraint baseColumnNames="valuable_id"
                             baseTableName="valuable_history"
                             constraintName="fk_valuable_history_valuable_id"
                             referencedColumnNames="id"
                             referencedTableName="valuable"/>
    <addForeignKeyConstraint baseColumnNames="user_id"
```

```
                baseTableName="subscription"
                constraintName="fk_subscription_user_id"
                referencedColumnNames="id"
                referencedTableName="user"/>
    <addForeignKeyConstraint baseColumnNames="valuable_id"
                baseTableName="subscription"
                constraintName="fk_subscription_valuable_id"
                referencedColumnNames="id"
                referencedTableName="valuable"/>
    <addForeignKeyConstraint baseColumnNames="user_id"
                baseTableName="alert"
                constraintName="fk_alert_user_id"
                referencedColumnNames="id"
                referencedTableName="user"/>
</changeSet>

<changeSet id="11" author="Oleksii Kaliuha">
    <insert tableName="valuable">
        <column name="id">1</column>
        <column name="name">Apple</column>
        <column name="symbol">AAPL</column>
        <column name="trend">-1.34</column>
        <column name="price">134.91</column>
        <column name="type">STOCK</column>
        <column name="last_update" valueDate="now()"/>
    </insert>
    <insert tableName="valuable">
        <column name="id">2</column>
        <column name="name">IBM</column>
        <column name="symbol">IBM</column>
        <column name="trend">2.63</column>
        <column name="price">133.80</column>
        <column name="type">STOCK</column>
        <column name="last_update" valueDate="now()"/>
    </insert>
    <insert tableName="valuable">
        <column name="id">3</column>
        <column name="name">Microsoft</column>
        <column name="symbol">MSFT</column>
        <column name="trend">-1.63</column>
        <column name="price">256.93</column>
        <column name="type">STOCK</column>
        <column name="last_update" valueDate="now()"/>
    </insert>
    <insert tableName="valuable">
        <column name="id">4</column>
        <column name="name">Amazon</column>
        <column name="symbol">AMZN</column>
        <column name="trend">-139.11</column>
        <column name="price">3191</column>
        <column name="type">STOCK</column>
        <column name="last_update" valueDate="now()"/>
    </insert>
    <insert tableName="valuable">
        <column name="id">5</column>
        <column name="name">EPAM</column>
        <column name="symbol">EPAM</column>
        <column name="trend">21.46</column>
        <column name="price">308.87</column>
        <column name="type">STOCK</column>
```

```
            <column name="last_update" valueDate="now()"/>
        </insert>
        <insert tableName="valuable">
            <column name="id">6</column>
            <column name="name">Google</column>
            <column name="symbol">GOOGL</column>
            <column name="trend">-110.36</column>
            <column name="price">2111.40</column>
            <column name="type">STOCK</column>
            <column name="last_update" valueDate="now()"/>
        </insert>
        <insert tableName="valuable">
            <column name="id">7</column>
            <column name="name">Tesla</column>
            <column name="symbol">TSLA</column>
            <column name="trend">-46.74</column>
            <column name="price">628.16</column>
            <column name="type">STOCK</column>
            <column name="last_update" valueDate="now()"/>
        </insert>
        <insert tableName="valuable">
            <column name="id">8</column>
            <column name="name">Bitcoin</column>
            <column name="symbol">BTC</column>
            <column name="trend">110.11</column>
            <column name="price">29481</column>
            <column name="type">CRYPTO</column>
            <column name="last_update" valueDate="now()"/>
        </insert>
        <insert tableName="valuable">
            <column name="id">9</column>
            <column name="name">Etherium</column>
            <column name="symbol">ETH</column>
            <column name="trend">-0.13</column>
            <column name="price">1977.66</column>
            <column name="type">CRYPTO</column>
            <column name="last_update" valueDate="now()"/>
        </insert>
    </changeSet>

</databaseChangeLog>
```

Конфігурація веб додатку:

```
# Database config
spring.datasource.url=jdbc:mysql://localhost:3306/valuables_exchange?useUnicode=true&characterEncoding=u
tf-8&createDatabaseIfNotExist=true&serverTimezone=UTC
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=root

#ORM configuration
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

#JWT
jwt.token.expired=3600000
jwt.secret=ZXBhbXN0b2NrZXhjaGFuZ2U=
```

```
#Security
security.anonymousEndPoints=/login,/registration,/loginError
security.freeEndPoints=/,/oauth2/**, /processOrders, /valuableHistory
security.adminEndpoints=/admin/block/**,/admin/users/**,/admin/changeRole/**,
admin/changeUserBalance/**
security.userEndPoints=/portfolio/**,/wallet/**,/updateBalance/**,/orders/**,/updateUserStockInfo/**,/order/*
*,/cancelOrder/**

#Google OAuth2
spring.security.oauth2.client.registration.google.client-id=264771707593-
bhin3ktrf711bpbudnak7ock4384gggd.apps.googleusercontent.com
spring.security.oauth2.client.registration.google.client-secret=Yw478OhzPFjFa_n7y0eAr1Go

#OAuth2
security.oauth2.endpoint=/oauth2/authorization/

#Registration bonus
registration.bonus = 50.0

#pagination
pagination.amount=6
table.pagination.amount=20
subscription.pagination.size=10

#Liquibase
spring.liquibase.enabled=true
spring.liquibase.change-log=classpath:liquibase/changelog.xml
```

Конфігурація додатку відповідального за постійне оновлення та обробку інформації:

```
#Tomcat config
server.port=8181

# Database config
spring.datasource.url=jdbc:mysql://localhost:3306/valuables_exchange?useUnicode=true&characterEncoding=u
tf-8&createDatabaseIfNotExist=true&serverTimezone=UTC
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

#developer.mode
developer.mode=${DEVELOPER.MODE}

#Valuable symbols
valuable.api.stock.symbols=AAPL,IBM,MSFT,AMZN,EPAM,GOOGL,TSLA,FB
valuable.api.crypto.symbols=BTC/USD,ETH/USD

# Api config
valuable.api.twelvedata.baseurl=wss://ws.twelvedata.com/v1/quotes/price?apikey=
valuable.api.twelvedata.key.stock=bff956116a53494ca75b6985d80be522
valuable.api.twelvedata.key.crypto=01e83dd76ec640208944f1cfd7f43c6f
```

Реалізація створення та обробки замовлень:

```
package com.epam.rd.stock.exchange.facade.impl;

import com.epam.rd.stock.exchange.dto.OrderCreateDto;
import com.epam.rd.stock.exchange.dto.OrderViewDto;
import com.epam.rd.stock.exchange.exception.ProcessOrderException;
import com.epam.rd.stock.exchange.facade.OrderFacade;
import com.epam.rd.stock.exchange.facade.ValuableFacade;
import com.epam.rd.stock.exchange.mapper.OrderMapper;
import com.epam.rd.stock.exchange.mapper.UserValuableInfoMapper;
import com.epam.rd.stock.exchange.model.Order;
import com.epam.rd.stock.exchange.model.User;
import com.epam.rd.stock.exchange.model.Valuable;
import com.epam.rd.stock.exchange.model.enums.OrderStatus;
import com.epam.rd.stock.exchange.model.enums.ValuableType;
import com.epam.rd.stock.exchange.service.OrderService;
import com.epam.rd.stock.exchange.service.ValuableService;
import com.epam.rd.stock.exchange.service.UserService;
import com.epam.rd.stock.exchange.service.UserValuableInfoService;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Isolation;
import org.springframework.transaction.annotation.Transactional;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.time.LocalDateTime;

@Service
@RequiredArgsConstructor
public class OrderFacadeImpl implements OrderFacade {

    private final OrderService orderService;
    private final ValuableService valuableService;
    private final ValuableFacade valuableFacade;
    private final UserService userService;
    private final OrderMapper orderMapper;

    @Override
    @Transactional(isolation = Isolation.READ_COMMITTED)
    public OrderViewDto submit(OrderCreateDto orderCreateDto) {
        LocalDateTime timeSubmitted = LocalDateTime.now();

        Valuable valuable = valuableService.findById(orderCreateDto.getValuableId());
        User user = userService.findByEmail(orderCreateDto.getUserEmail());

        if(!valuable.getType().equals(ValuableType.CRYPTO) &&
!isIntegerValue(orderCreateDto.getAmount())){
            throw new ProcessOrderException("This valuable are not available to operate in not round amounts!");
        }

        Order order = orderMapper.toOrder(valuable, user, orderCreateDto);

        order.setValuablePrice(valuable.getPrice());
        order.setDateTime(timeSubmitted);

        order.setOrderPrice(calculateOrderPrice(valuable.getPrice(), order.getAmount()));
        try {
```

```
            switch (order.getType()) {
                case BUY:
                    valuableFacade.buy(order);
                    break;
                case SELL:
                    valuableFacade.sell(order);
                    break;
            }
            order.setStatus(OrderStatus.SUCCESS);
        } catch (ProcessOrderException e) {
            order.setStatus(OrderStatus.FAIL);
            order.setFailDescription(e.getMessage());
        }
        LocalDateTime timeProcessed = LocalDateTime.now();
        order.setDateTime(timeProcessed);

        Order newOrder = orderService.save(order);
        return orderMapper.toOrderDto(newOrder);
    }

    @Override
    public Page<OrderViewDto> findByUserIdAndStatus(String userId, OrderStatus status, Integer page, int
size) {
        Pageable pageable = PageRequest.of(page - 1, size);
        if (status == null) {
            return orderMapper.toPageOrderDto(orderService.findByUserId(userId, pageable));
        }
        return orderMapper.toPageOrderDto(orderService.findByUserIdAndStatus(userId, status, pageable));
    }


    private BigDecimal calculateOrderPrice(BigDecimal valuablePrice, BigDecimal amount) {
        return valuablePrice.multiply(amount).setScale(2, RoundingMode.HALF_UP);
    }

    private boolean isIntegerValue(BigDecimal bd) {
        boolean ret;
        try {
            bd.toBigIntegerExact();
            ret = true;
        } catch (ArithmeticException ex) {
            ret = false;
        }
        return ret;
    }
}
```

Реалізація функціоналу для роботи з підписками:

```
package com.epam.rd.stock.exchange.facade.impl;

import com.epam.rd.stock.exchange.dto.CreateSubscriptionDto;
import com.epam.rd.stock.exchange.dto.SubscriptionViewDto;
import com.epam.rd.stock.exchange.exception.CreateSubscriptionException;
import com.epam.rd.stock.exchange.exception.UserDontHaveEnoughValuablesForSubscriptionException;
import com.epam.rd.stock.exchange.facade.SubscriptionFacade;
import com.epam.rd.stock.exchange.mapper.SubscriptionMapper;
import com.epam.rd.stock.exchange.model.Subscription;
import com.epam.rd.stock.exchange.model.User;
import com.epam.rd.stock.exchange.model.UserValuableInfo;
import com.epam.rd.stock.exchange.model.Valuable;
```

```
import com.epam.rd.stock.exchange.model.enums.SubscriptionType;
import com.epam.rd.stock.exchange.model.enums.ValuableType;
import com.epam.rd.stock.exchange.service.SubscriptionService;
import com.epam.rd.stock.exchange.service.UserService;
import com.epam.rd.stock.exchange.service.ValuableService;
import com.epam.rd.stock.exchange.service.impl.UserValuableInfoServiceImpl;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Isolation;
import org.springframework.transaction.annotation.Transactional;

import java.math.BigDecimal;

@Service
@RequiredArgsConstructor
@Slf4j
public class SubscriptionFacadeImpl implements SubscriptionFacade {

    private final SubscriptionService subscriptionService;

    private final UserService userService;

    private final ValuableService valuableService;

    private final UserValuableInfoServiceImpl userValuableInfoService;

    private final SubscriptionMapper subscriptionMapper;

    @Override
    @Transactional(isolation = Isolation.REPEATABLE_READ)
    public SubscriptionViewDto createSubscription(String email, CreateSubscriptionDto createSubscription) {
        User user = userService.findByEmail(email);
        Valuable valuable = valuableService.findById(createSubscription.getValuableId());
        if(!valuable.getType().equals(ValuableType.CRYPTO) &&
!createSubscription.getSubscriptionType().equals(SubscriptionType.INFORM) &&
!isIntegerValue(createSubscription.getAmount())){
            throw new CreateSubscriptionException("This valuable are not available to operate in not round
amounts!");
        }

        Subscription subscription = subscriptionMapper.toSubscription(createSubscription, user, valuable);
        UserValuableInfo userValuableInfo = userValuableInfoService.findUserStockInfo(user.getId(),
valuable.getId());
        if(userValuableInfo == null && subscription.isReserve() &&
subscription.getType().equals(SubscriptionType.SELL) ){
            throw new UserDontHaveEnoughValuablesForSubscriptionException("User don't have enough
valuables to reserve");
        }
        else if( userValuableInfo != null && subscription.isReserve() &&
subscription.getType().equals(SubscriptionType.SELL)&&

userValuableInfo.getSellAmount().add(subscription.getAmount()).compareTo(userValuableInfo.getAmount())>
0){
            throw new UserDontHaveEnoughValuablesForSubscriptionException("User don't have enough
valuables to reserve");
        }
```

```
        else if( userValuableInfo != null && subscription.isReserve() &&
subscription.getType().equals(SubscriptionType.SELL)&&

userValuableInfo.getSellAmount().add(subscription.getAmount()).compareTo(userValuableInfo.getAmount())<
=0){
            userValuableInfo.setSellAmount(userValuableInfo.getSellAmount().add(subscription.getAmount()));
            userValuableInfoService.save(userValuableInfo);
        }
        return subscriptionMapper.toUserView(subscriptionService.createSubscription(subscription));
    }

    @Override
    @Transactional(isolation = Isolation.REPEATABLE_READ)
    public void deleteSubscription(String email, String subscriptionId) {
        User user = userService.findByEmail(email);
        Subscription  subscription = subscriptionService.get(user,subscriptionId);
        UserValuableInfo userValuableInfo = userValuableInfoService.findUserStockInfo(user.getId(),
subscription.getValuableId());
        if(subscription.getType().equals(SubscriptionType.SELL) && subscription.isReserve()){

userValuableInfo.setSellAmount(userValuableInfo.getSellAmount().subtract(subscription.getAmount()));
            userValuableInfoService.save(userValuableInfo);
        }
        subscriptionService.deleteSubscription(user, subscriptionId);
    }


    @Override
    public Page<SubscriptionViewDto> get(String email, int page, int pageSize) {
        User user = userService.findByEmail(email);
        Pageable pageable = PageRequest.of(page - 1, pageSize);
        return subscriptionService.get(user,pageable).map(subscriptionMapper::toUserView);
    }

    @Override
    public Page<SubscriptionViewDto> get(String email, SubscriptionType subscriptionType, int page, int
pageSize) {
        User user = userService.findByEmail(email);
        Pageable pageable = PageRequest.of(page - 1, pageSize);
        return subscriptionService.get(user, subscriptionType, pageable).map(subscriptionMapper::toUserView);
    }

    private boolean isIntegerValue(BigDecimal price){
        boolean ret;
        try {
            price.toBigIntegerExact();
            ret = true;
        } catch (ArithmeticException ex) {
            ret = false;
        }
        return ret;
    }
}
```

Реалізація функціоналу для роботи з користувачами:

```
package com.epam.rd.stock.exchange.facade.impl;

import com.epam.rd.stock.exchange.dto.*;
import com.epam.rd.stock.exchange.dto.enums.ChangeBalanceType;
```

```
import com.epam.rd.stock.exchange.exception.AuthenticationException;
import com.epam.rd.stock.exchange.exception.NotEnoughBalanceException;
import com.epam.rd.stock.exchange.exception.UserNotFoundException;
import com.epam.rd.stock.exchange.facade.UserFacade;
import com.epam.rd.stock.exchange.mapper.UserMapper;
import com.epam.rd.stock.exchange.mapper.UserValuableInfoMapper;
import com.epam.rd.stock.exchange.model.BalanceUpdateHistory;
import com.epam.rd.stock.exchange.model.User;
import com.epam.rd.stock.exchange.model.enums.BalanceUpdateType;
import com.epam.rd.stock.exchange.service.BalanceUpdateHistoryService;
import com.epam.rd.stock.exchange.service.UserService;
import com.epam.rd.stock.exchange.util.CardValidationUtil;
import lombok.RequiredArgsConstructor;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Isolation;
import org.springframework.transaction.annotation.Transactional;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.List;
import java.util.stream.Collectors;

@Service
@RequiredArgsConstructor
public class UserFacadeImpl implements UserFacade {

    private final UserService userService;

    private final UserValuableInfoMapper userValuableInfoMapper;

    private final UserMapper userMapper;

    private final PasswordEncoder passwordEncoder;

    private final BalanceUpdateHistoryService balanceUpdateHistoryService;

    @Override
    public UserViewDto findById(String id) {
        User user = userService.findById(id);
        return userMapper.toUserViewDto(user);
    }

    @Override
    @Transactional(isolation = Isolation.REPEATABLE_READ)
    public BigDecimal updateBalance(String email, ChangeBalanceDto changeBalanceDto) {
        CardValidationUtil.validateUserCard(changeBalanceDto);
        User user = userService.findByEmail(email);
        BigDecimal balanceUpdate = changeBalanceDto.getSum();
        BalanceUpdateType type = BalanceUpdateType.TOPUP;
        if(changeBalanceDto.getChangeBalanceType().equals(ChangeBalanceType.WITHDRAW)){
            verifyUserBalance(user.getBalance(), balanceUpdate);
            balanceUpdate = balanceUpdate.multiply(BigDecimal.valueOf(-1));
            type = BalanceUpdateType.WITHDRAW;
        }
        userService.updateBalance(user.getId(), balanceUpdate);
        BalanceUpdateHistory newBalanceUpdateHistory = BalanceUpdateHistory.builder()
                .update(balanceUpdate)
                .type(type)
                .userId(user.getId())
```

```
                .card(changeBalanceDto.getCard())
                .dateTime(LocalDateTime.now())
                .build();

        balanceUpdateHistoryService.save(newBalanceUpdateHistory);
        return balanceUpdate;
    }

    @Override
    public UserViewDto signIn(UserSignInDto userSignInDto) {
        User user = userService.findByEmail(userSignInDto.getEmail());
        if (!passwordEncoder.matches(userSignInDto.getPassword(), user.getPassword())) {
            throw new AuthenticationException("Invalid login or password");
        }
        return userMapper.toUserViewDto(user);
    }

    @Override
    public UserViewDto signInWithSocialNetwork(UserCreateDto userCreateDto) {
        User user;
        try {
            user = userService.findByEmail(userCreateDto.getEmail());
        } catch (UserNotFoundException e) {
            user = userService.save(userMapper.toUserSocial(userCreateDto));
        }
        return userMapper.toUserViewDto(user);
    }

    @Override
    public UserViewDto findByEmail(String email) {
        User user = userService.findByEmail(email);
        UserViewDto userViewDto = userMapper.toUserViewDto(user);
        List<UserValuableInfoViewDto> userValuableInfoViewDtoList = user.getValuables()
                .stream().map(userValuableInfoMapper::toUserValuableInfoViewDto).collect(Collectors.toList());
        userViewDto.setValuables(userValuableInfoViewDtoList);
        return userViewDto;
    }

    @Override
    public UserViewDto registration(UserCreateDto userCreateDto) {
        User user = userService.save(userMapper.toUser(userCreateDto));
        return userMapper.toUserViewDto(user);
    }

    private void verifyUserBalance(BigDecimal userBalance, BigDecimal balanceUpdate){
        if(userBalance.compareTo(balanceUpdate) < 0){
            throw new NotEnoughBalanceException("User don't have enough balance to withdraw");
        }
    }
}
```

Реалізація роботи з ЦП, криптовалютами та портфоліо користувачів

```
package com.epam.rd.stock.exchange.facade.impl;

import com.epam.rd.stock.exchange.dto.ValuableHistoryViewDto;
import com.epam.rd.stock.exchange.dto.ValuableViewDto;
import com.epam.rd.stock.exchange.exception.ProcessOrderException;
import com.epam.rd.stock.exchange.facade.ValuableFacade;
import com.epam.rd.stock.exchange.mapper.ValuableMapper;
```

```
import com.epam.rd.stock.exchange.model.*;
import com.epam.rd.stock.exchange.service.UserService;
import com.epam.rd.stock.exchange.service.ValuableHistoryService;
import com.epam.rd.stock.exchange.service.ValuableService;
import com.epam.rd.stock.exchange.service.UserValuableInfoService;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.math.BigDecimal;

@Service
@RequiredArgsConstructor
public class ValuableFacadeImpl implements ValuableFacade {

    private final ValuableService valuableService;
    private final UserValuableInfoService userValuableInfoService;
    private final UserService userService;
    private final ValuableHistoryService valuableHistoryService;
    private final ValuableMapper valuableMapper;

    @Override
    public Page<ValuableViewDto> findStocksBySymbol(String symbol, int page, int size) {
        Pageable pageable = PageRequest.of(page - 1, size);
        return valuableMapper.toPageValuableDto(valuableService.findStocksBySymbol(symbol, pageable));
    }

    @Override
    public UserValuableInfo buy(Order order) {
        User user = order.getUser();
        Valuable valuable = order.getValuable();
        UserValuableInfo userValuableInfo = userValuableInfoService.findUserStockInfo
                (user.getId(), valuable.getId());

        boolean enoughMoney = user.getBalance().doubleValue() >= order.getOrderPrice().doubleValue();
        if (enoughMoney) {
            if (userValuableInfo != null) {
                BigDecimal newAmount = userValuableInfo.getAmount().add(order.getAmount());
                userValuableInfo.setAmount(newAmount);
            } else {
                userValuableInfo = UserValuableInfo.builder()
                        .valuable(valuable)
                        .user(user)
                        .sellAmount(BigDecimal.ZERO)
                        .amount(order.getAmount())
                        .build();
            }
        } else {
            throw new ProcessOrderException("User doesn't have enough money for this order.");
        }
        userValuableInfoService.save(userValuableInfo);
        userService.updateBalance(user.getId(), order.getOrderPrice().multiply(BigDecimal.valueOf(-1)));
        return userValuableInfo;
    }

    @Override
    public UserValuableInfo sell(Order order) {
```

```
        User user = order.getUser();
        UserValuableInfo userValuablesInfo = userValuableInfoService.findUserStockInfo
                (user.getId(), order.getValuable().getId());
        if (userHasEnoughStocks(userValuablesInfo, order)) {
            BigDecimal newAmount = userValuablesInfo.getAmount().subtract(order.getAmount());
            if (newAmount.compareTo(BigDecimal.ZERO) == 0) {
                userValuableInfoService.delete(userValuablesInfo);
            } else {
                userValuablesInfo.setAmount(newAmount);
                userValuableInfoService.save(userValuablesInfo);
            }
        } else {
            throw new ProcessOrderException("User doesn't have enough valuables for this order.");
        }
        userService.updateBalance(user.getId(), order.getOrderPrice());
        return userValuablesInfo;
    }

    @Override
    public ValuableViewDto findById(String stockId) {
        return valuableMapper.toValuableDto(valuableService.findById(stockId));
    }

    @Override
    public Page<ValuableHistoryViewDto> findHistoryById(String valuableId, int page, int size) {
        Pageable pageable = PageRequest.of(page - 1, size);
        Page<ValuableHistory> valuableHistories =
valuableHistoryService.findValuableHistoryByValuableId(valuableId, pageable);
        return valuableHistories.map(this::toUserView);
    }

    private boolean userHasEnoughStocks(UserValuableInfo userValuableInfo, Order order) {
        return userValuableInfo != null &&

(userValuableInfo.getAmount().subtract(userValuableInfo.getSellAmount()).compareTo(order.getAmount()) >=
0);
    }

    private ValuableHistoryViewDto toUserView(ValuableHistory valuableHistory){
        return ValuableHistoryViewDto.builder()
                .dateTime(valuableHistory.getDateTime())
                .newPrice(valuableHistory.getNewPrice())
                .previousPrice(valuableHistory.getPreviousPrice())
                .trend(valuableHistory.getTrend())
                .build();
    }
}
```

Програмне налаштування безпеки веб додатку:

```
package com.epam.rd.stock.exchange.config;

import com.epam.rd.stock.exchange.handler.OAuth2AuthenticationSuccessHandler;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.oauth2.client.endpoint.DefaultAuthorizationCodeTokenResponseClient;
```

```
import org.springframework.security.oauth2.client.endpoint.OAuth2AccessTokenResponseClient;
import org.springframework.security.oauth2.client.endpoint.OAuth2AuthorizationCodeGrantRequest;
import org.springframework.security.oauth2.client.web.AuthorizationRequestRepository;
import org.springframework.security.oauth2.client.web.HttpSessionOAuth2AuthorizationRequestRepository;
import org.springframework.security.oauth2.core.endpoint.OAuth2AuthorizationRequest;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

@Configuration
@RequiredArgsConstructor
public class SecurityConfig extends WebSecurityConfigurerAdapter {

   @Value("${security.anonymousEndPoints}")
   private final String[] anonymousEndPoints;

   private final OAuth2AuthenticationSuccessHandler successHandler;

   @Value("${security.freeEndPoints}")
   private final String[] freeEndPoints;

   @Value("${security.oauth2.endpoint}")
   private String authorizationRequestBaseUri;

   @Value("${security.userEndPoints}")
   private final String[] userEndPoints;

   @Override
   protected void configure(HttpSecurity http) throws Exception {
      http.authorizeRequests()
            .antMatchers(freeEndPoints).permitAll()
            .antMatchers(anonymousEndPoints).anonymous()
            .antMatchers(userEndPoints).hasAuthority("USER")
            .anyRequest().authenticated()

            .and()
            .oauth2Login()
            .loginPage("/login")
            .authorizationEndpoint()
            .baseUri(authorizationRequestBaseUri)
            .authorizationRequestRepository(authorizationRequestRepository())
            .and()
            .successHandler(successHandler)
            .tokenEndpoint()
            .accessTokenResponseClient(accessTokenResponseClient())

            .and()
            .failureUrl("/loginError")

            .and()
            .logout()
            .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
            .logoutSuccessUrl("/")
            .invalidateHttpSession(true)
            .deleteCookies("JSESSIONID")
            .permitAll();
   }


   @Bean
   public AuthorizationRequestRepository<OAuth2AuthorizationRequest> authorizationRequestRepository() {
      return new HttpSessionOAuth2AuthorizationRequestRepository();
```

```
    }

    @Bean
    public OAuth2AccessTokenResponseClient<OAuth2AuthorizationCodeGrantRequest>
accessTokenResponseClient() {
        return new DefaultAuthorizationCodeTokenResponseClient();
    }

}
```

Фільтр для перевірки JWT токену:

```
package com.epam.rd.stock.exchange.filter;

import com.epam.rd.stock.exchange.dto.UserViewDto;
import com.epam.rd.stock.exchange.exception.InvalidTokenException;
import com.epam.rd.stock.exchange.exception.UserBlockedException;
import com.epam.rd.stock.exchange.facade.UserFacade;
import com.epam.rd.stock.exchange.mapper.UserMapper;
import com.epam.rd.stock.exchange.service.AlertService;
import com.epam.rd.stock.exchange.service.AuthenticationService;
import com.epam.rd.stock.exchange.util.JwtTokenUtil;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.annotation.Order;
import org.springframework.http.HttpMethod;
import org.springframework.security.web.csrf.CsrfToken;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.math.BigDecimal;
import java.util.Arrays;

@Component
@RequiredArgsConstructor
@Slf4j
@Order(1)
public class TokenFilter extends OncePerRequestFilter {

    private static final String LOGIN_ENDPOINT = "/login";

    private static final String CSRF_TOKEN_ATTRIBUTE = "_csrf";

    @Value("${security.anonymousEndPoints}")
    private final String[] anonymousEndPoints;

    @Value("${security.freeEndPoints}")
    private final String[] freeEndPoints;

    private final UserFacade userFacade;

    private final UserMapper userMapper;
```

```
    private final AuthenticationService authenticationService;

    private final JwtTokenUtil jwtTokenUtil;

    private final AlertService alertService;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain
filterChain) throws ServletException, IOException {
        String path = request.getServletPath();
        String method = request.getMethod();
        if (HttpMethod.POST.matches(method) && path.equals(LOGIN_ENDPOINT)) {
            doLoginPost(request);
        } else if (!Arrays.asList(freeEndPoints).contains(path) &&
!Arrays.asList(anonymousEndPoints).contains(path)) {
            doOnNotFreeEndpoints(request);
        }
        addSessionParamsForWallet(request);
        filterChain.doFilter(request, response);
    }

    private void doLoginPost(HttpServletRequest request) {
        String email = request.getParameter("email");
        String password = request.getParameter("password");
        userFacade.signIn(userMapper.toUserSignInDto(email, password));
        String userId = userFacade.findByEmail(email).getId();
        UserViewDto user = userFacade.findById(userId);
        checkForBlocking(user);

        jwtTokenUtil.createAndAddTokenIntoSession(userId, request);
        authenticationService.authenticateUser(userMapper.toUserSignInDto(email, password));
    }

    private void doOnNotFreeEndpoints(HttpServletRequest request) {
        CsrfToken csrfToken = (CsrfToken) request.getAttribute(CSRF_TOKEN_ATTRIBUTE);
        String userId = jwtTokenUtil.getUserIdFromToken(csrfToken.getToken(), request);
        if (userId != null) {
            UserViewDto user = userFacade.findById(userId);
            checkForBlocking(user);
            authenticationService.authenticateUser(userMapper.
                    toUserSignInDto(userFacade.findById(userId).getEmail(), ""));
        }
    }

    private void addSessionParamsForWallet(HttpServletRequest request) {
        CsrfToken csrfToken = (CsrfToken) request.getAttribute(CSRF_TOKEN_ATTRIBUTE);
        try {
            String userId = jwtTokenUtil.getUserIdFromToken(csrfToken.getToken(), request);
            UserViewDto user = userFacade.findById(userId);
            boolean notificate = alertService.checkAlerts(userId);
            String email = user.getEmail();
            BigDecimal balance = user.getBalance();
            HttpSession session = request.getSession();
            session.setAttribute("notificate", notificate);
            session.setAttribute("email", email);
            session.setAttribute("balance", balance);
        } catch (InvalidTokenException e) {
            log.info("User is not authorized");
        }
```

```
    }

    private void checkForBlocking(UserViewDto user) {
        if (user.isBlocked()) {
            authenticationService.removeUserFromSecurityContext();
            throw new UserBlockedException("Your account was blocked");
        }
    }
}
```

Реалізація роботи з JWT токеном:

```
package com.epam.rd.stock.exchange.util;

import com.epam.rd.stock.exchange.exception.InvalidTokenException;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.ExpiredJwtException;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.MalformedJwtException;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.SignatureException;
import io.jsonwebtoken.UnsupportedJwtException;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.web.csrf.CsrfToken;
import org.springframework.security.web.csrf.DefaultCsrfToken;
import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.util.Date;

@Slf4j
@Component
public class JwtTokenUtil {

    private static final String CSRF_TOKEN_ATTRIBUTE = "_csrf";
    private static final String CSRF_HEADER = "X-CSRF-TOKEN";
    private static final String CSRF_ATTRIBUTE_NAME =
"org.springframework.security.web.csrf.HttpSessionCsrfTokenRepository.CSRF_TOKEN";
    private static final String CLAIM_USER_ID = "user_id";
    private static final String EXCEPTION_MESSAGE = "Token is invalid or expired ";

    @Value("${jwt.token.expired}")
    private long validityInMilliseconds;

    @Value("${jwt.secret}")
    private String secret;

    public String createAndAddTokenIntoSession(String id, HttpServletRequest request) {
        String token = createToken(id);
        DefaultCsrfToken csrfToken = new DefaultCsrfToken(CSRF_HEADER, CSRF_TOKEN_ATTRIBUTE,
token);
        addTokenInSession(csrfToken, request);
        return token;
    }

    private String createToken(String userId) {
        Date now = new Date();
        Date validity = new Date(now.getTime() + validityInMilliseconds);
```

```
        return Jwts.builder()
                .claim(CLAIM_USER_ID, userId)
                .setIssuedAt(now)
                .setExpiration(validity)
                .signWith(SignatureAlgorithm.HS512, secret)
                .compact();
    }

    private void addTokenInSession(CsrfToken token, HttpServletRequest request) {
        HttpSession session = request.getSession();
        session.setAttribute(CSRF_ATTRIBUTE_NAME, token);
    }

    public String getUserIdFromToken(String token, HttpServletRequest request) {
        Claims claims;
        try {
            claims = Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody();
        } catch (ExpiredJwtException e) {
            String userId = e.getClaims().get(CLAIM_USER_ID, String.class);
            log.info("Refresh token");
            String newToken = createAndAddTokenIntoSession(userId, request);
            claims = Jwts.parser().setSigningKey(secret).parseClaimsJws(newToken).getBody();
        } catch (UnsupportedJwtException | MalformedJwtException | SignatureException |
IllegalArgumentException e) {
            throw new InvalidTokenException(EXCEPTION_MESSAGE + e.getMessage());
        }
        return claims.get(CLAIM_USER_ID, String.class);
    }
}
```

Реалізація оновлення активу та обробки підписок:

```
package com.epam.rd.stock.exchange.facade.impl;

import com.epam.rd.stock.exchange.dto.ValuableUpdateDto;
import com.epam.rd.stock.exchange.facade.UpdateFacade;
import com.epam.rd.stock.exchange.model.*;
import com.epam.rd.stock.exchange.model.enums.OrderStatus;
import com.epam.rd.stock.exchange.model.enums.OrderType;
import com.epam.rd.stock.exchange.model.enums.SubscriptionType;
import com.epam.rd.stock.exchange.model.util.ConditionVerifyUtil;
import com.epam.rd.stock.exchange.service.*;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Isolation;
import org.springframework.transaction.annotation.Transactional;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.List;

@Service
@RequiredArgsConstructor
@Slf4j
public class UpdateFacadeImpl implements UpdateFacade {

    private final ValuableService valuableService;

    private final ValuableHistoryService valuableHistoryService;
```

```
    private final SubscriptionService subscriptionService;

    private final UserValuableInfoService userValuableInfoService;

    private final OrderService orderService;

    private final UserService userService;

    private final AlertService alertService;

    @Override
    @Transactional(isolation = Isolation.REPEATABLE_READ)
    public void update(ValuableUpdateDto valuableUpdateDto) {
       Valuable valuable = valuableService.get(valuableUpdateDto.getSymbol());
       if(valuable.getPrice().compareTo(valuableUpdateDto.getPrice()) == 0){
          return;
       }
        LocalDateTime now = LocalDateTime.now();
        BigDecimal trend = valuableUpdateDto.getPrice().subtract(valuable.getPrice());
       Valuable newValuable = Valuable.builder()
             .id(valuable.getId())
             .name(valuable.getName())
             .symbol(valuableUpdateDto.getSymbol())
             .price(valuableUpdateDto.getPrice())
             .trend(trend)
             .lastUpdate(now)
             .type(valuable.getType())
             .build();

        //Create history for update
        ValuableHistory valuableHistory = ValuableHistory.builder()
              .valuableId(valuable.getId())
              .trend(trend)
              .dateTime(now)
              .previousPrice(valuable.getPrice())
              .newPrice(newValuable.getPrice())
              .build();

        valuableHistoryService.addHistory(valuableHistory);
        valuableService.updateValuable(newValuable);

        List<Subscription> subscriptionList = subscriptionService.findByValuableId(newValuable.getId());
        subscriptionList.parallelStream().filter(subscription -> this.checkSubscription(subscription,
newValuable.getPrice()) )
              .forEach(subscription -> this.processSubscription(subscription, valuable));
    }

    private void processSubscription(Subscription subscription, Valuable valuable){
       if(subscription.getType().equals(SubscriptionType.INFORM)){
          performInform(subscription, valuable);
       }
       else if(subscription.getType().equals(SubscriptionType.SELL)){
          performSell(subscription, valuable);
       }
       else if(subscription.getType().equals(SubscriptionType.BUY)){
          performBuy(subscription, valuable);
       }
    }
```

```
    private void performInform(Subscription subscription, Valuable valuable){
        Alert alert = Alert.builder()
                .dateTime(LocalDateTime.now())
                .message(String.format("Alert for %s valuable %s condition : %s , current price is %s",
valuable.getType(),
                        valuable.getSymbol(), subscription.conditionToString(), valuable.getPrice()))
                .userId(subscription.getUserId())
                .build();
        alertService.createAlert(alert);
        if(!subscription.isContinuos()){
            subscriptionService.remove(subscription);
        }
    }

    private void performBuy(Subscription subscription, Valuable valuable){
        User user = userService.get(subscription.getUserId());
        Alert alert = Alert.builder()
                .dateTime(LocalDateTime.now())
                .userId(user.getId()).build();
        BigDecimal price = subscription.getAmount().multiply(valuable.getPrice());
        if(user.getBalance().subtract(price).compareTo(BigDecimal.ZERO) < 0){

            if(subscription.isFailSafe()){
                alert.setMessage(String.format("Subscription for %s valuable %s in count %s failed cause of not
enough balance(without disable): %s",
                        valuable.getType(), valuable.getSymbol(), subscription.getAmount(),
subscription.conditionToString()));
            }
            else{
                alert.setMessage(String.format("Subscription for %s valuable %s in count %s failed cause of not
enough balance: %s",
                        valuable.getType(), valuable.getSymbol(), subscription.getAmount(),
subscription.conditionToString()));
                subscriptionService.remove(subscription);
             }
            Order order = Order.builder()
                    .dateTime(LocalDateTime.now())
                    .orderPrice(price)
                    .valuablePrice(valuable.getPrice())
                    .amount(subscription.getAmount())
                    .status(OrderStatus.FAIL)
                    .type(OrderType.AUTOMATIC_BUY)
                    .valuable(valuable)
                    .user(user)
                    .failDescription("User don't have enough money for this order")
                    .build();
            orderService.createOrder(order);
        } else{
            alert.setMessage(String.format("Subscription for %s valuable %s in count %s success: %s , purchased
%s valuables for %s with total %s",
                    valuable.getType(), valuable.getSymbol(), subscription.getAmount(),
subscription.conditionToString(),
                    subscription.getAmount(), valuable.getPrice(), price));

            user.setBalance(user.getBalance().subtract(price));
            UserValuableInfo userValuableInfo = userValuableInfoService.getByUserAndValuable(user, valuable);
            if(userValuableInfo == null){
                userValuableInfo = UserValuableInfo.builder()
                        .valuable(valuable)
                        .user(user)
```

```
                .amount(subscription.getAmount())
                .sellAmount(BigDecimal.ZERO)
                .build();
            userValuableInfoService.create(userValuableInfo);
        } else {
            userValuableInfo.setAmount(userValuableInfo.getAmount().add(subscription.getAmount()));
            userValuableInfoService.update(userValuableInfo);
        }
        Order order = Order.builder()
                .dateTime(LocalDateTime.now())
                .orderPrice(price)
                .valuablePrice(valuable.getPrice())
                .amount(subscription.getAmount())
                .status(OrderStatus.SUCCESS)
                .type(OrderType.AUTOMATIC_BUY)
                .valuable(valuable)
                .user(user)
                .build();
        orderService.createOrder(order);
        subscriptionService.remove(subscription);
        userService.update(user);
    }
    alertService.createAlert(alert);
}

private void performSell(Subscription subscription, Valuable valuable){
    User user = userService.get(subscription.getUserId());
    Alert alert = Alert.builder()
            .dateTime(LocalDateTime.now())
            .userId(user.getId()).build();
    UserValuableInfo userValuableInfo = userValuableInfoService.getByUserAndValuable(user, valuable);
    BigDecimal price = subscription.getAmount().multiply(valuable.getPrice());
    if(userValuableInfo == null) {
        failSafeCheck(subscription, valuable, alert, price, user);
    }
    else{

if(userValuableInfo.getAmount().subtract(subscription.getAmount()).compareTo(BigDecimal.ZERO)>=0){
            alert.setMessage(String.format("Subscription for %s valuable %s in count %s success: %s , selled %s
valuables for %s with total ",
                    valuable.getType(), valuable.getSymbol(), subscription.getAmount(),
                    subscription.conditionToString(), subscription.getAmount(), valuable.getPrice(), price));
            user.setBalance(user.getBalance().add(price));

            Order order = Order.builder()
                    .dateTime(LocalDateTime.now())
                    .orderPrice(price)
                    .valuablePrice(valuable.getPrice())
                    .amount(subscription.getAmount())
                    .status(OrderStatus.SUCCESS)
                    .type(OrderType.AUTOMATIC_SELL)
                    .valuable(valuable)
                    .user(user)
                    .build();
            userValuableInfo.setAmount(userValuableInfo.getAmount().subtract(subscription.getAmount()));
            if(subscription.isReserve()){

userValuableInfo.setSellAmount(userValuableInfo.getSellAmount().subtract(subscription.getAmount()));
            }
            userValuableInfoService.update(userValuableInfo);
```

```
            userService.update(user);
            orderService.createOrder(order);
            subscriptionService.remove(subscription);
        }
        else{
            failSafeCheck(subscription, valuable, alert, price, user);
        }
    }
    alertService.createAlert(alert);

}

private void failSafeCheck(Subscription subscription, Valuable valuable, Alert alert, BigDecimal price, User
user) {
    if(subscription.isFailSafe()) {
        alert.setMessage(String.format("Subscription for %s valuable %s in count %s failed(without delete)
cause of not enough valuables: %s",
                valuable.getType(), valuable.getSymbol(), subscription.getAmount(),
subscription.conditionToString()));
    } else{
        alert.setMessage(String.format("Subscription for %s valuable %s in count %s failed cause of not enough
valuables: %s",
                valuable.getType(), valuable.getSymbol(), subscription.getAmount(),
subscription.conditionToString()));
        subscriptionService.remove(subscription);
    }
    Order order = Order.builder()
            .dateTime(LocalDateTime.now())
            .orderPrice(price)
            .valuablePrice(valuable.getPrice())
            .amount(subscription.getAmount())
            .status(OrderStatus.FAIL)
            .type(OrderType.AUTOMATIC_SELL)
            .valuable(valuable)
            .failDescription("User don't have enough valuables for this order")
            .user(user)
            .build();
    orderService.createOrder(order);
}


private boolean checkSubscription(Subscription subscription, BigDecimal price){
    return ConditionVerifyUtil.verify(subscription, price);
}

}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

Перелік файлів на диску

Перелік документів на магнітному носії

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Калюга О.Р 122-18-3.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Калюга О.Р 122-18-3.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Калюга О.Р 122-18-3.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Калюга О.Р 122-18-3.ppt	Презентація кваліфікаційної роботи