

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Розробка веб-орієнтованої системи аналізу інвестиційних портфелів»

Виконав студент групи ІПЗм-23-1
ЖУРБЕЙ Артем Валентинович

Керівник роботи:
ЛОБАНЧИКОВА Надія Миколаївна

Рецензент:
ЄФРЕМОВ Юрій Миколайович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«01» жовтня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу

Здобувач вищої освіти: **ЖУРБЕЙ Артем Валентинович**

Керівник роботи: **ЛОБАНЧИКОВА Надія Миколаївна**

Тема роботи: **«Розробка веб-орієнтованої системи аналізу інвестиційних портфелів»**, затверджена Наказом закладу вищої освіти від **«01» жовтня 2024 р. № 520/с.**

Вихідні дані для роботи: Об'єктом дослідження є технології розробки веб-додатків для автоматизованого аналізу інвестиційного портфеля, включаючи методи обробки фінансових даних, оцінки ризиків і прибутковості.

Предметом дослідження є моделі, методи та алгоритми розробки веб-додатку для проведення глибокого аналізу інвестиційного портфеля, його оптимізації та надання рекомендацій користувачам щодо покращення фінансової ефективності.

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Лобанчикова Н.М	02.10.2024р.	02.10.2024р.
2	Лобанчикова Н.М.	02.10.2024р.	02.10.2024р.
3	Лобанчикова Н.М.	02.10.2024р.	02.10.2024р.

РЕФЕРАТ

Метою кваліфікаційної роботи є розробка веб-орієнтованої системи аналізу інвестиційних портфелів.

Об'єктом дослідження є технології розробки веб-додатків для автоматизованого аналізу інвестиційних портфелів з методами обробки фінансових даних, оцінки ризиків і прибутковості.

Предметом дослідження є моделі, методи та алгоритми розробки веб-орієнтованої системи для проведення глибокого аналізу інвестиційного портфеля, його оптимізації та надання рекомендацій користувачам щодо покращення фінансової ефективності.

Теоретична значущість роботи полягає у вивченні теоретичних основ аналізу інвестиційного портфеля, розробці моделей оцінки ризику, диверсифікації активів, прогнозування фінансових показників та оптимізації складу портфеля з урахуванням індивідуальних потреб користувача.

Практична цінність роботи полягає в дослідженні шляхів впровадження сучасних інформаційних технологій у сферу управління фінансовими активами. Запропонований програмний додаток дозволяє підвищити ефективність інвестиційної діяльності користувачів за рахунок аналізу інвестиційного портфелю, та вказує на дії необхідні щоб виправити недоліки, це сприяє покращенню фінансово-економічної стабільності інвесторів.

Магістерська атестаційна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел, із 24 найменувань (3 сторінки), 2-х додатків(15 сторінок), 4 таблиць, 41 ілюстрація по тексту. Повний обсяг роботи становить 110 сторінок, з них 84 сторінки основного тексту.

КЛЮЧОВІ СЛОВА: АНАЛІЗ, ДОХІДНІСТЬ, ЕФЕКТИВНІСТЬ, ОПТИМІЗАЦІЯ ПОРТФЕЛЮ, РЕЗУЛЬТАТ, РИЗИК.

ABSTRACT

The purpose of the qualification work is to implement an investment portfolio analysis system.

The object of research is the technology of developing web applications for automated analysis of the investment portfolio, including methods of processing financial data, risk and profitability assessment.

The subject of the study is the models, methods and algorithms for developing a web application for conducting an in-depth analysis of an investment portfolio, optimizing it and providing recommendations to users to improve financial efficiency.

The theoretical significance of the work is to study the theoretical foundations of investment portfolio analysis, develop models for risk assessment, asset diversification, financial performance forecasting, and portfolio optimization, taking into account the individual needs of the user.

The practical value of the work lies in the study of ways to introduce modern information technologies in the field of financial asset management. The proposed software application allows to increase the efficiency of investment activity of users by analyzing the investment portfolio, and indicates the actions necessary to correct the shortcomings, which contributes to the improvement of financial and economic stability of investors.

The master's thesis consists of an introduction, 3 chapters, conclusions, a list of references, 24 titles (3 pages), 2 appendices (15 pages), 4 tables, 41 illustrations in the text. The total volume of the work is 110 pages, including 84 pages of the main text.

Keywords: analysis, efficiency, portfolio optimization, results, return, risk.

ЗМІСТ

ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ АНАЛІЗУ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ	8
1.1 Аналіз існуючих методів	8
1.2 Аналіз предметної області.....	11
1.3 Аналіз аналогів програмного продукту	13
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ АНАЛІЗУ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ	27
2.1 Визначення варіантів використання.....	27
2.2 Розробка бази даних системи.....	62
2.3 Проєктування та реалізація алгоритмів роботи системи	64
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБДОДАТКОМ.....	68
3.1 Створення програмного додатку	68
3.2 Структура інтерфейсу вебдодатку.....	76
3.3 Тестування роботи програмного комплексу	87
ВИСНОВКИ.....	90
ДОДАТКИ.....	94

ВСТУП

У сучасному цифровому світі постають все нові задачі автоматизації, підтримки прийняття рішень, особливо актуальною є фінансова сфера та сфера надання фінансових послуг, тому тема "Розробка веб-орієнтованої системи аналізу інвестиційних портфелів" є актуальною. Завдяки швидким технологічним змінам та постійному розвитку фінансових ринків, інвесторам стає все складніше приймати обґрунтовані рішення щодо своїх портфелів. Тому знайдення нових інформаційних технологій для аналізу інвестиційних портфелів дозволить інвесторам ефективно використовувати різноманітні фінансові інструменти та стратегії для максимізації дохідності та мінімізації ризику.

Доволі актуальною є розробка веб-орієнтованої системи аналізу інвестиційних портфелів, що буде допомагати інвесторам зрозуміти диверсифікацію, аналізувати ринкові тенденції, оцінювати ризики та дохідність різних активів, а також визначати оптимальний склад інвестиційного портфеля. Таким чином система буде допомагати інвесторам зробити осмислені та обґрунтовані рішення на фінансових ринках, що в свою чергу сприятиме їх фінансовій стабільності.

Метою роботи є розробка веб-орієнтованої системи аналізу інвестиційних портфелів, який дозволить автоматизувати оцінку його ефективності, оцінювати ризики та надавати рекомендації для оптимізації. Встановлена мета обумовлює такі **завдання**:

- провести аналіз існуючих методів і моделей аналізу інвестиційних портфелів;
- дослідити підходи до прогнозування ринкових показників;
- розробити систему для аналізу інвестиційного портфеля з функцією генерації рекомендацій;
- оцінити ефективність реалізованого додатку через тестування на прикладних даних.

Об’єктом дослідження є технології розробки веб-додатків для автоматизованого аналізу та оптимізації інвестиційних портфель.

Предметом дослідження є моделі, методи та алгоритми розробки веб-орієнтованих систем аналізу та оптимізації інвестиційних портфель.

У кваліфікаційній роботі використано такі методи: класифікації даних – для аналізу структури активів та визначення ризикових категорій; математичне моделювання – для оцінки прибутковості та ризику; візуальне програмування – для реалізації інтерфейсу веб-додатку.

Наукова новизна роботи полягає у застосуванні інструментів машинного навчання для реалізації функції автоматизації аналізу інвестиційного портфеля. Було розроблено алгоритми для оптимізації структури портфеля, а також інтегровано ці інструменти у веб-орієнтовану систему для зручності використання.

Практична цінність роботи полягає у створенні системи, яка сприяє підвищенню ефективності управління інвестиціями. Програмний додаток дозволяє інвесторам швидко оцінювати стан своїх портфель, визначати оптимальні стратегії для розподілу активів і приймати обґрунтовані фінансові рішення, що сприяє зростанню економічної стабільності та розвитку інвестиційної діяльності.

Особистим внеском здобувача є розробка алгоритму для оптимізації структури портфелю, а також тестування та оптимізація системи на реальних даних. Усі результати роботи отримані автором самостійно.

Апробація результатів роботи. Матеріали кваліфікаційної роботи доповідалися на Міжнародній науково-практичній конференції студентів, аспірантів та молодих вчених “Молодіжна наука: інновації та глобальні виклики”, 06 листопада 2024 року, Полтава та XVII Міжнародній науково-практичній конференції “Інформаційні технології і автоматизація – 2024”, 31 жовтня - 1 листопада 2024 року, Одеса.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ АНАЛІЗУ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ

1.1 Аналіз існуючих методів

У сфері управління інвестиціями існує широкий набір методів для аналізу інвестиційного портфеля. Вони дозволяють оцінювати його ефективність, ризики, дохідність і відповідність інвестиційним цілям. Основні методи аналізу поділяються на кількісні, якісні та комбіновані. Нижче детально розглянемо кожен із цих методів та їх особливості.

1. Кількісні методи аналізу – ці методи базуються на математичних та статистичних обчисленнях, що дозволяють отримати об’єктивні оцінки параметрів портфеля. Найбільш поширеними з них є:

1.1. Метод середньої дохідності, [1] – цей метод оцінює середню дохідність портфеля за визначений період. Використовується для оцінки ефективності управління. Нище наведено формулу методу, (1.1):

$$R_{avg} = \frac{\sum_{i=1}^n R_i}{n}, \quad (1.1)$$

де R_i – дохідність портфеля в окремий період;

n – кількість періодів;

R_{avg} – середня дохідність портфеля.

1.2. Коефіцієнт Шарпа, [2] – показує, наскільки дохідність портфеля перевищує безризикову ставку з урахуванням ризику (волатильності), (1.2):

$$S = \frac{R_p - R_f}{\sigma_p}, \quad (1.2)$$

де R_p - середня дохідність портфеля;

R_f – безризикова ставка;

σ_p – стандартне відхилення.

1.3. Коефіцієнт Трейнора, [3] – враховує систематичний ризик портфеля (бета-коефіцієнт) відповідно до формули (1.3).

$$T = \frac{R_p - R_f}{\beta_p}, \quad (1.3)$$

де R_p - середня дохідність портфеля;

R_f – безризикова ставка;

β_p – бета-коефіцієнт портфеля.

2. Якісні методи аналізу – ці методи ґрунтуються на експертній оцінці, аналізі макроекономічних факторів та вивченні стратегічного середовища, зокрема:

2.1. Фундаментальний аналіз, [4] – базується на оцінці внутрішньої вартості активів за допомогою фінансових показників, ринкових трендів та макроекономічних даних. Має велику кількість формул, нище наведено формула, яка використовується найчастіше (модель Гордона [5], формула 1.4).

$$P = \frac{D}{r-g}, \quad (1.4)$$

де P – внутрішня вартість акції;

D – прогнозований дивіденд (або дохід на акцію);

r – ставка дисконту (враховує ризик інвестиції);

g – очікуваний темп зростання дивідендів.

2.2. SWOT-аналіз, [6] – визначає сильні та слабкі сторони, можливості та загрози, пов'язані з конкретними активами портфеля.

2.3. Сценарний аналіз, [7] – прогнозує розвиток подій на ринку та оцінює, як вони вплинуть на дохідність портфеля, формула 1.5.

$$R = \sum_{i=1}^n P_i * R_i, \quad (1.5)$$

де R – очікувана дохідність портфеля;

P_i – ймовірність сценарію;

R_i – дохідність портфеля у сценарії.

n – кількість сценаріїв.

3. Комбіновані методи – ці методи поєднують кількісні обчислення з якісним аналізом, зокрема:

3.1. Метод Value at Risk, [8] – оцінює потенційні збитки портфеля при певному рівні довіри протягом фіксованого періоду, що визначається за формулою 1.6:

$$\text{VaR} = u - z * a, \quad (1.6)$$

де VaR – максимальні потенційні збитки при заданому рівні довіри;

u – середня дохідність портфеля;

z – коефіцієнт, що відповідає обраному рівню довіри (наприклад, 1.645 для 95% довіри);

a – стандартне відхилення дохідності портфеля (волатильність).

3.2. Монте-Карло моделювання, [9] – використовує числові моделі для прогнозування можливих результатів інвестицій у портфель, відповідно до формули 1.7:

$$R_{\text{scen}} = R_{\text{avg}} + a * z, \quad (1.7)$$

де R_{scen} – прогнозована дохідність портфеля в конкретному сценарії;

R_{avg} – середня дохідність портфеля за історичними даними;

a – волатильність;

z – випадкове значення, згенероване із стандартного нормального розподілу.

3.3. Аналіз ефективного портфеля (теорія Марковіца), [10] – дозволяє формувати оптимальний портфель із максимальним рівнем дохідності за мінімального рівня ризику, формула 1.8:

$$a_p = \sqrt{\sum_{i=1}^n w_i^2 * a_i^2 + 2 \sum_{i=1}^n \sum_{j=i+1}^n w_i * w_j * p_{ij} * a_i * a_j} \quad (1.8)$$

де a_p – волатильність портфеля;

w_i – вага i -го активу в портфелі;

a_i —стандартне відхилення дохідності i -го активу;

ρ_{ij} — кореляція між дохідностями активів i та j ;

n — кількість активів у портфелі.

Проведений аналіз дозволив визначити методи оцінки інвестиційних проєктів для впровадження в веб-орієнтовану систему.

1.2 Аналіз предметної області

Система аналізу інвестиційного портфеля є важливим інструментом у фінансовій сфері, що дозволяє інвесторам ефективно управляти своїми активами, оптимізувати стратегії та приймати обґрунтовані інвестиційні рішення. Вона спрямована на детальний аналіз складу портфеля, оцінку ризиків, моніторинг ринкових змін і вдосконалення структури активів для досягнення максимальної дохідності за умов мінімальних ризиків. Управління інвестиційним портфелем є багатокomпонентним процесом, що включає оцінку поточного стану ринку, прогнозування трендів і адаптацію стратегії до змін зовнішнього середовища.

Інвестиційний портфель являє собою сукупність фінансових інструментів, які можуть включати акції, облігації, деривативи, нерухомість, криптовалюти, а також альтернативні активи, такі як дорогоцінні метали чи предмети мистецтва. Основною метою формування такого портфеля є досягнення фінансових цілей інвестора, які можуть варіюватися від забезпечення стабільного пасивного доходу до отримання високої дохідності в короткостроковій перспективі. Успіх інвестиційного процесу значною мірою залежить від правильного вибору активів, їхнього розподілу між категоріями та постійного моніторингу їхньої ефективності.

Завдяки розвитку сучасних технологій інвестиційна діяльність стає все більш доступною як для великих корпорацій, так і для індивідуальних інвесторів. Виникнення цифрових платформ, таких як брокерські сервіси й аналітичні інструменти, дозволяє значно спростити процес управління портфелем. Однак

інвестування залишається складним завданням, що вимагає глибоких знань ринкових механізмів, обґрунтованого аналізу ризиків та вміння прогнозувати зміни в економіці. Навіть добре диверсифікований портфель може зазнати втрат, якщо не враховуються макроекономічні чинники, політичні ризики чи нестабільність окремих секторів.

У рамках аналізу портфеля використовується широкий спектр інструментів і показників. Основними з них є ROI [11] (рентабельність інвестицій), що дозволяє оцінити прибутковість активу щодо вкладених коштів, волатильність [12], яка характеризує рівень ризику через коливання ціни, β -коефіцієнт [13], що визначає взаємозв'язок активу з ринком у цілому, та VaR (вартість під ризиком), який оцінює можливі втрати при заданому рівні довіри. Аналіз цих показників є ключовим для оцінки ефективності інвестиційного портфеля та виявлення його слабких місць.

Одним із важливих завдань системи аналізу є забезпечення належного рівня диверсифікації. Диверсифікація [14] дозволяє мінімізувати ризики шляхом розподілу активів між різними секторами, галузями чи географічними регіонами. Наприклад, у разі негативних змін в одному секторі, інвестор може уникнути значних втрат завдяки наявності активів у стабільніших галузях. Для оцінки диверсифікації використовуються математичні моделі, що дозволяють визначити рівень концентрації активів та ефективність їхнього розподілу.

Методи аналізу та прогнозування є невід'ємною складовою системи. Серед класичних підходів виділяється модель Марковіца, яка дозволяє побудувати ефективний портфель, враховуючи співвідношення між ризиком і доходністю. Цей метод став основою для багатьох сучасних фінансових теорій та інструментів. Крім того, сучасні технології, такі як Монте-Карло симуляції, дають змогу проводити моделювання й оцінювати поведінку портфеля за різних ринкових умов. Ці підходи дозволяють інвесторам краще розуміти можливі сценарії розвитку подій та адаптувати свою стратегію відповідно до прогнозів.

Цільова аудиторія таких систем є досить широкою. Це як індивідуальні інвестори, які прагнуть досягти фінансової незалежності чи створити пасивний

дохід, так і професійні гравці ринку, такі як інвестиційні компанії та фінансові консультанти. Для перших важливі зрозумілі й доступні інструменти, що допомагають управляти капіталом без глибоких знань фінансів. Для других – можливість проводити детальний аналіз великих обсягів даних, формувати звіти й оптимізувати портфелі клієнтів.

Таким чином, система аналізу інвестиційного портфеля є не лише інструментом для моніторингу й оптимізації фінансових активів, але й важливим елементом у прийнятті стратегічних рішень. Вона дозволяє інвесторам і професіоналам сфери фінансів ефективно управляти ризиками, прогнозувати можливі результати та досягати поставлених фінансових цілей, забезпечуючи стабільність і зростання капіталу в умовах сучасного динамічного ринку.

1.3 Аналіз аналогів програмного продукту

У процесі проектування застосунків важливо вивчати наявні аналоги, щоб краще зрозуміти вимоги користувачів та сформувану унікальну ціннісну пропозицію для нашого продукту, завдяки якій наше рішення зможе створити конкуренцію на ринку. Вивчення досвіду інших проєктів дозволяє уникнути помилок, з якими вже стикалися розробники, а також віднайти ефективні підходи, що зарекомендували себе на практиці. Такий аналіз допомагає не лише вдосконалити функціональність продукту, а й врахувати сучасні тренди та технології, які визначають розвиток ринку. Унікальна цінність створюється через поєднання глибокого розуміння потреб користувачів та інноваційного підходу до їх задоволення. Для порівняння були обрані наступні застосунки:

1. Investopedia.
2. Valueinvesting.

Investopedia – це популярний освітній вебсайт, який надає інформацію про фінанси, інвестиції та різні економічні концепції. Він також пропонує ряд інструментів для інвесторів, зокрема функцію аналізу портфелю. Ось плюси і мінуси цього сайту, орієнтуючись на можливості аналізу портфелю.

На рисунку 1.1 зображено головну сторінку програмного продукту.

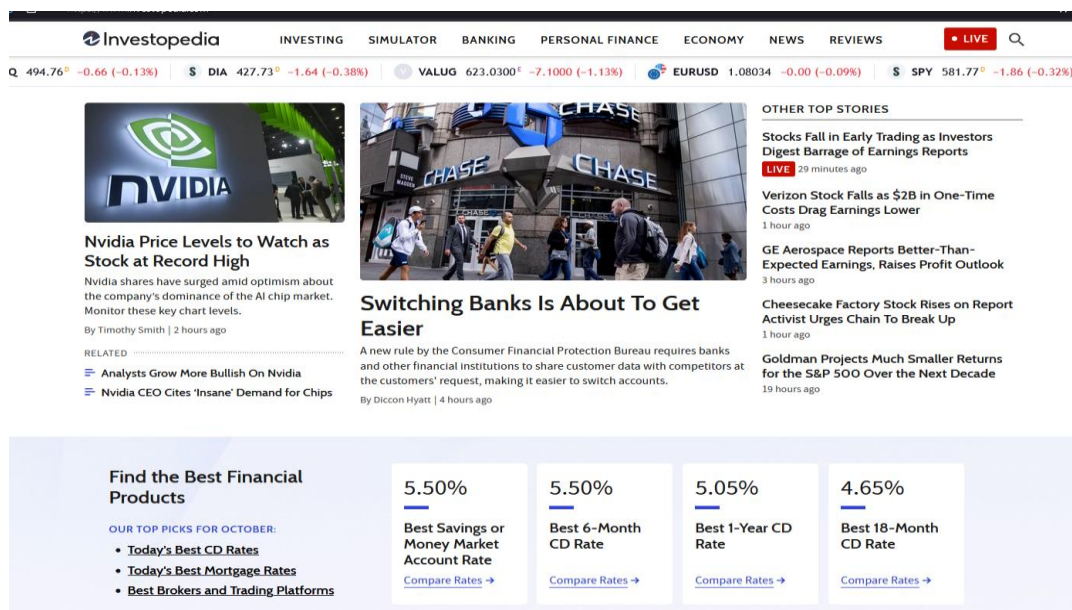


Рис. 1.1. Investopedia (головна сторінка)

Переваги Investopedia включають доступність та зрозумілість, адже інформація на сайті викладена простою мовою, що робить її зрозумілою як для початківців, так і для досвідчених інвесторів, допомагаючи швидко розібратися в базових принципах аналізу портфеля. Сайт також пропонує калькулятори та інструменти для аналізу, зокрема аналізатор портфеля, який дозволяє вводити інвестиції і отримувати базові аналітичні дані, такі як ризики, доходність та диверсифікація. Крім того, професійне консультування доступне через інструменти для симуляцій та порівняння інвестиційних стратегій, що дозволяє оцінити ефективність активів у різних сценаріях.

Водночас Investopedia має й недоліки: функції аналізу досить базові у порівнянні з професійними програмами, що може не задовольнити професійних інвесторів, які шукають детальніші та точніші дані. Інструменти аналізу можуть не враховувати останніх ринкових подій чи змін у реальному часі, що робить аналіз менш точним для поточної ринкової ситуації. Також відсутні індивідуалізовані рекомендації, які є доступними на платних фінансових

платформах, зокрема автоматизовані поради щодо оптимізації портфеля на основі конкретних ризикових профілів.

Valueinvesting – цей сайт фокусується на принципах і стратегіях інвестування за підходом "інвестування в цінності". Це популярний підхід серед довгострокових інвесторів, які шукають активи, недооцінені ринком, (рис 1.2).

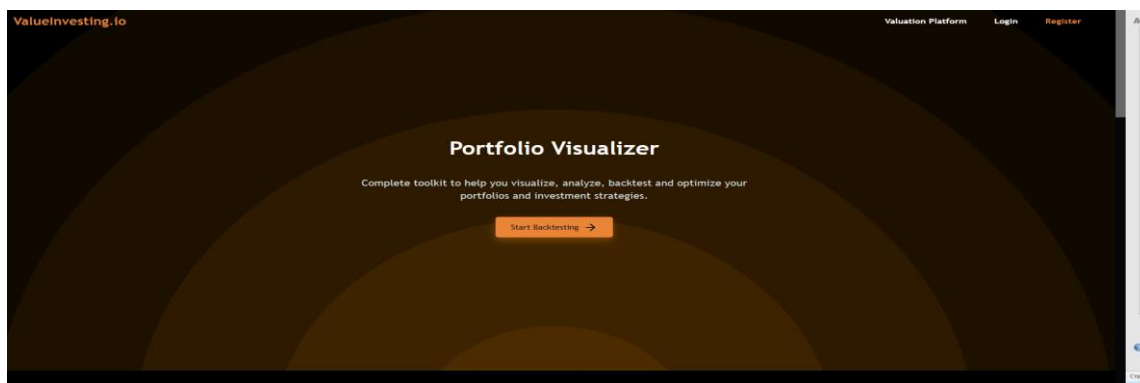


Рис. 1.2. Valueinvesting (головна сторінка)

Переваги Valueinvesting включають глибокий фокус на інвестуванні в цінності, оскільки сайт надає спеціалізовані матеріали та інструменти, орієнтовані на принципи Бенджаміна Грехема та Уоррена Баффета, що особливо корисно для прихильників цього підходу. Освітні ресурси на сайті пропонують можливість вивчення фундаментального аналізу, рентабельності акцій та розрахунку внутрішньої вартості компаній. Також Valueinvesting зосереджений на довгостроковій оцінці компаній через фундаментальний аналіз, що допомагає визначити, чи активи недооцінені або переоцінені, і сприяє створенню надійного портфеля.

Водночас є й недоліки: новачкам може бути складно зрозуміти інтерфейс та використовувати його ефективно. Обмежений діапазон інструментів для різних стратегій робить сайт менш універсальним, оскільки він зосереджений лише на підході інвестування в цінності. Крім того, через фокус на фундаментальному аналізі та специфічних фінансових показниках, новачкам без додаткового навчання важко розібратися в усіх аспектах запропонованої стратегії.

Організуємо представлення отриманих даних за допомогою табличної структури (табл. 1.1).

Таблиця 1.1

Порівняння платформ

Характеристика	Investopedia	ValueInvesting	Власний вебдодаток
Дизайн	Зрозумілий	Не зрозумілий	Зрозумілий
Вибір фінансових інструментів	Широкий спектр доступних інструментів	Вузьконаправлений спектр доступних інструментів	Вузьконаправлений спектр доступних інструментів
Освітні ресурси	Присутні	Присутні	Відсутні
Професійні інструменти аналізу	Так	Можливо, але обмеженіше	Відсутні
Можливість відслідковувати відразу декілька портфелів	Відсутня	Присутня	Присутня

Після порівняння характеристик трьох платформ для аналізу інвестиційного портфеля – Valueinvesting, Investopedia і власного вебдодатка – можна зробити такі висновки: по-перше, дизайн. Valueinvesting і Investopedia мають доволі непривітний і перевантажений дизайн під час проведення аналізу, тоді як власний вебдодаток пропонує мінімалістичний і сучасний інтерфейс, що робить процес аналізу більш комфортним. По-друге, мультифункціональність. Valueinvesting і Investopedia є багатофункціональними платформами, що може ускладнювати їх використання через надмірність інструментів і інформації. У той час, власний вебдодаток зосереджений виключно на аналізі портфеля, забезпечуючи простоту та ефективність роботи. По-третє, локалізація. Valueinvesting і Investopedia доступні лише англійською мовою, що може стати бар'єром для українських користувачів. Власний вебдодаток розроблений із урахуванням локальних потреб, що дозволяє забезпечити зручність і зрозумілість для користувачів з України.

Тому для початківців найкращим вибором буде запропонований вебдодаток. Це обумовлено тим, що розроблений вебдодаток має інтуїтивно зрозумілий дизайн, що дозволяє користувачам легко орієнтуватися в інтерфейсі

без необхідності навчання або додаткових інструкцій. Крім того, сайт має вузьку спеціалізацію, що значно полегшує його використання.

Вибір архітектури системи аналізу інвестиційного портфелю. Вебдодаток складається з компонентів клієнтської та серверної частин. Серверна частина, яка являє собою Web API [15], була спроектована з використанням принципів SOLID [16] та розділенням відповідальностей.

Основні компоненти архітектури включають: сервер, який відповідає за обробку запитів і надання відповідей. Бази даних яка використовується для зберігання даних, включаючи міграції для керування схемою бази даних і її налаштування. Джерело даних містить сутності, які відображають структуру таблиць бази даних і взаємозв'язки між ними. Модулі забезпечують логічний поділ функціональності програми, наприклад, модуль "portfolio", який включає контролери для обробки HTTP-запитів [17], інтерфейси для опису даних, налаштування залежностей, сервіси для бізнес-логіки, а також DTO [18] для передачі даних між компонентами. Сервіси є перевикористовуваними компонентами, доступними в різних модулях програми. Головний модуль і головний сервіс об'єднують всі модулі та сервіси, забезпечуючи цілісну роботу програми.

В цілому, архітектура застосунку використовує незалежні модулі, використання сервісів для бізнес-логіки та інші принципи, що допомагають створити розширювану та легко зрозумілу програму.

На рисунку 1.3 зображено спрощену архітектуру проєкту.

На сьогоднішній день існує розмаїття архітектурних підходів у вебдодатках, кожен з яких має свої переваги та недоліки. Вибір конкретної архітектури залежить від поставлених цілей та вимог. Для розробки фронтенд частини можна розглянути такі архітектурні рішення:

1. монолітна [19];
2. мікросервісна [20];
3. безсерверна [21].

Monolithic Architecture (Монолітна архітектура). Монолітна архітектура має кілька важливих переваг. Вона відзначається простотою в розробці та випуску, що робить її привабливим вибором для багатьох проєктів. Її легко масштабувати на невеликому рівні, що дозволяє ефективно використовувати ресурси, а також вона проста у тестуванні та супроводі, оскільки всі компоненти знаходяться в єдиній системі.

Серед недоліків монолітної архітектури варто виділити її схильність до зростання розмірів проєкту, що може створювати труднощі з управлінням та розширенням системи. Також існує потенційна залежність між компонентами, яка може негативно впливати на стійкість та надійність усієї системи. Крім того, оновлення окремих компонентів є складним завданням, оскільки зміни можуть впливати на роботу всієї архітектури.

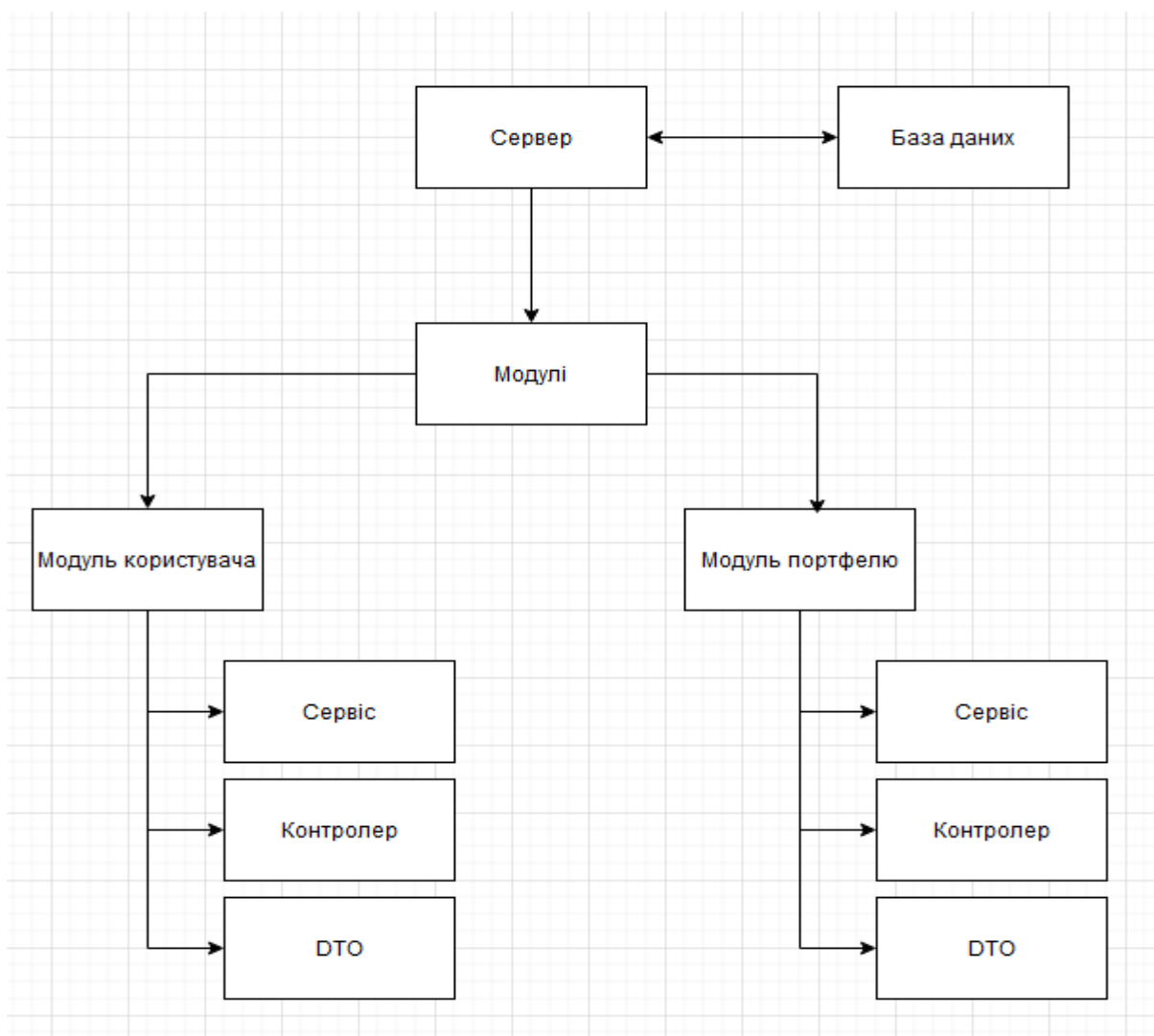


Рис. 1.3. Спрощена архітектура проєкту

Microservices Architecture (Мікросервісна архітектура). Мікросервісна архітектура має низку переваг. Однією з ключових є можливість незалежного розгортання та масштабування окремих мікросервісів, що дозволяє оптимізувати ресурси й адаптуватися до змін. Розробка, оновлення та тестування окремих компонентів також є значно простішими, адже кожен мікросервіс функціонує автономно. До того ж, ця архітектура дозволяє використовувати різні технології та мови програмування для кожного мікросервісу, надаючи гнучкість у виборі інструментів і підходів.

Проте є і недоліки. Управління та координація між мікросервісами є складним завданням, що вимагає розвиненої організації. Велика кількість мережових викликів може спричинити збільшення затримок у роботі системи, а також ускладнити її тестування. Крім того, мікросервісна архітектура потребує розробки та підтримки складної інфраструктури для управління взаємодією мікросервісів, що може стати додатковим викликом для команди розробників.

Serverless Architecture (Безсерверна архітектура). Безсерверна архітектура пропонує значні переваги. По-перше, вона забезпечує автоматичне масштабування та керування ресурсами залежно від навантаження, що дозволяє ефективно обробляти пікові навантаження без необхідності вручну керувати інфраструктурою. Витрати на інфраструктуру також знижені, адже оплата здійснюється лише за фактичне виконання функцій, а не за постійно активні сервери. Такий підхід сприяє швидкому випуску нових функцій і дозволяє розробникам зосередитися на розробці функціоналу, а не на управлінні серверами.

Однак у цієї архітектури є й недоліки. Наприклад, обмежені можливості контролю та налаштування інфраструктури, оскільки керування серверною частиною відбувається на стороні хмарного провайдера. Розробка та планування функцій потребують ретельного підходу, щоб уникнути збільшення витрат через помилки або некоректне виконання. Крім того, тестування та локальна розробка можуть бути складними, адже функції виконуються безпосередньо в хмарному

середовищі, що створює труднощі у відтворенні та налагодженні роботи на локальному рівні.

Після проведення порівнянь згаданих варіантів та аналізу вимог додатку було обрано монолітну архітектуру, (Рис. 1.4).

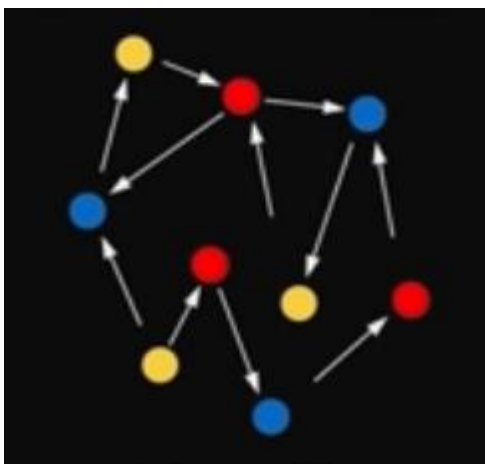


Рис. 1.4. Результат використання монолітної архітектури

Для побудови структури серверної частини було обрано підхід DDD [22], який добре поєднується з монолітною архітектурою. Основною перевагою DDD є чітке розділення відповідальності між доменними областями, що дозволяє організувати код у вигляді модулів, кожен з яких відповідає за конкретну бізнес-логіку. У контексті монолітної архітектури це забезпечує кращу структурування коду, зменшує залежності між модулями та полегшує підтримку системи. DDD покращує взаємодію між розробниками та бізнес-експертами, дозволяючи точно враховувати вимоги бізнесу. Завдяки розділенню на шари (доменний, застосунку, інфраструктурний) монолітна система стає гнучкою для змін, а модульна структура дозволяє тестувати частини системи окремо, не впливаючи на інші.

DDD добре інтегрується з NestJS, який надає зручні інструменти для побудови модульних систем. Використання декораторів, інверсії залежностей і вбудованої підтримки модульності в NestJS дозволяє легко впроваджувати основні принципи DDD. У монолітній архітектурі така інтеграція забезпечує

збереження цілісності системи, водночас дозволяючи виділяти логіку за доменами, що спрощує масштабування та розвиток.

Всі описані компоненти: сервер, клієнтська частина та база даних сполучаються двонаправленим зв'язком між собою (Рис. 1.5).

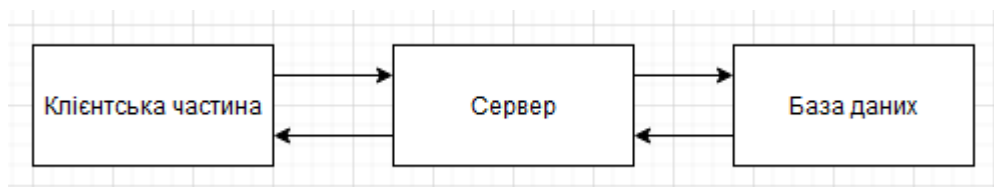


Рис. 1.5. Взаємодія компонентів системи

Розробка системи аналізу інвестиційного портфелю вимагає ретельного підходу та уваги до деталей. Нижче буде приведено перелік обраних для розробки технологій. Розробка клієнтського та серверного додатку відбуватиметься в інтегрованому середовищі розробки – Visual Studio Code.

Visual Studio Code є одним із найбільш популярних середовищ розробки, що ідеально підходить для веб-розробки завдяки своїй легкості, гнучкості та потужному функціоналу. Це середовище підтримує велику кількість мов програмування, таких як JavaScript, TypeScript, HTML, CSS та інші, що дозволяє використовувати одну платформу для роботи з різними технологіями. Окрім того, VS Code має велику кількість розширень і плагінів, які дають можливість адаптувати редактор під конкретні потреби розробника, додаючи підтримку нових фреймворків, бібліотек або інструментів для автоматичного форматування коду, перевірки на помилки і т.д. Вбудований термінал у VS Code дозволяє зручно працювати з командами та скриптами без необхідності переходити між редактором і терміналом, що значно підвищує ефективність роботи. Інтеграція з Git також є однією з переваг VS Code, оскільки дозволяє здійснювати всі основні операції з репозиторіями прямо з редактора, спрощуючи роботу в команді. Інтуїтивно зрозумілий інтерфейс і можливість налаштування робочого простору під індивідуальні потреби робить VS Code доступним для новачків, але водночас достатньо потужним для досвідчених розробників. Вбудована підтримка

“відладки” дозволяє швидко знаходити та виправляти помилки в коді, що є критично важливим при роботі з сучасними веб-технологіями. Крім того, VS Code доступний для всіх основних операційних систем – Windows, macOS та Linux, що робить його універсальним і зручним інструментом для розробників на будь-яких платформах.

Для розробки клієнтської частини були обрані наступні засоби:

React – було обрано через те що, це популярний JavaScript-фреймворк, який активно використовується для розробки користувацького інтерфейсу (UI) вебдодатків. Він дозволяє створювати інтерактивні та динамічні веб-сторінки, що швидко реагують на зміни в даних, завдяки концепції компонентно-орієнтованого підходу. Однією з основних переваг React є його здатність до повторного використання компонентів, що дозволяє значно зменшити кількість коду та спростити підтримку і оновлення програми. Фреймворк використовує концепцію "віртуального DOM", що дозволяє значно знизити кількість оновлень реального DOM, покращуючи швидкодію вебдодатків. React також має велику екосистему, включаючи бібліотеки для управління станом, маршрутизації та роботи з формами, що дозволяє швидко додавати функціональність до додатку без необхідності створювати все з нуля. React дозволяє розробникам зосередитися на побудові інтерфейсу без необхідності турбуватися про багато складних аспектів, таких як маніпулювання DOM чи реалізація низькорівневих оптимізацій, що робить його ідеальним вибором для розробки сучасних інтерактивних веб-додатків.

Next.js – було обрано через те що, це фреймворк для розробки вебдодатків на базі React, який дозволяє створювати як серверні, так і клієнтські рендеринг-сторінки, забезпечуючи при цьому високу продуктивність і зручність розробки. Він підтримує серверний рендеринг, статичну генерацію сторінок та гібридні підходи, що дозволяє ефективно оптимізувати вебдодатки для різних сценаріїв використання, забезпечуючи швидке завантаження і покращену взаємодію з користувачем. Однією з основних переваг Next.js є автоматичне управління маршрутизацією, що значно спрощує процес створення і підтримки складних

вебдодатків. Він підтримує гаряче перепризначення, що дозволяє швидко бачити зміни на сторінці без необхідності перезавантажувати вебдодаток, і забезпечує зручний механізм для роботи з API, що робить інтеграцію з бекендом набагато простішою. Крім того, Next.js має вбудовану підтримку для оптимізації зображень та автоматичне виконання бандлінгу, що дозволяє значно покращити продуктивність додатку, зменшуючи час завантаження сторінок.

Zustand [23] – було обрано через те що, це бібліотека для управління станом в JavaScript-додатках, яка дозволяє ефективно і зручно зберігати та змінювати стан додатка. Вона розроблена з урахуванням простоти, гнучкості і високої продуктивності, що робить її ідеальним вибором для розробників, які шукають швидкий і ефективний спосіб керувати станом додатка без складності, властивій більш важким бібліотекам, як Redux. Основною перевагою Zustand є її легкість у використанні та відсутність необхідності у великих додаткових налаштуваннях або шаблонах. Вона надає дуже простий API для створення глобального стану, що може бути доступним у будь-якому компоненті додатку, без необхідності в складних концепціях, як actions або reducers. Zustand дозволяє безперешкодно використовувати прості функції для оновлення стану, що робить код чистішим і зрозумілішим.

Material UI – було обрано через те що, це популярна бібліотека компонентів для розробки користувацького інтерфейсу (UI) вебдодатків, яка побудована на основі React. Вона надає набір готових компонентів, що відповідають принципам Material Design, що дозволяє швидко створювати сучасні, естетичні та функціональні інтерфейси. Використання Material UI спрощує розробку, оскільки зменшується потреба в ручному створенні UI елементів, таких як кнопки, текстові поля, таблиці, діалогові вікна та інші стандартні компоненти.

TypeScript – було обрано через те що, це розширення для JavaScript, яке додає статичну типізацію, об'єктно-орієнтований підхід та інші інструменти, що значно покращують процес розробки вебдодатків. Типізація в TypeScript дозволяє вказувати типи даних для змінних, функцій, об'єктів та інших структур,

що допомагає виявляти помилки на етапі компіляції, ще до виконання коду. Це дозволяє уникнути багатьох типових помилок, що можуть виникати під час розробки на чистому JavaScript, таких як помилки типів або невірне використання функцій.

ESLint – було обрано через те що, це інструмент статичного аналізу коду, який використовується для виявлення потенційних проблем, помилок і недоліків форматування в коді, зокрема в JavaScript і TypeScript проєктах. Він дозволяє розробникам перевіряти свій код на відповідність певним стандартам якості, забезпечуючи чистоту та узгодженість коду.

Завдяки обраним інструментам та технологіям, реалізація клієнтського додатку була легкою та зрозумілою.

Розробка серверної частини сервісу виконувалася за допомогою наступних інструментів:

Node.js – було обрано через те що, це відкрите середовище виконання JavaScript, побудоване на рушії V8, яке дозволяє виконувати JavaScript-код на серверній стороні. Його основною перевагою є можливість розробляти масштабовані та ефективні серверні додатки, використовуючи одну мову програмування як на клієнтській, так і на серверній стороні. Node.js оптимізовано для асинхронної обробки запитів, що дозволяє обробляти велику кількість одночасних з'єднань з високою продуктивністю.

NestJS – було обрано через те що, це фреймворк для розробки серверних додатків на мові TypeScript або JavaScript, побудований на основі Node.js. Він надає потужну архітектуру для створення масштабованих і підтримуваних серверних рішень. NestJS використовує підхід, що поєднує найкращі практики об'єктно-орієнтованого програмування, функціонального програмування та реактивного програмування, що дозволяє розробникам створювати високоякісні додатки з чіткою структурою.

TypeORM – було обрано через те що, це бібліотека для роботи з базами даних в середовищі Node.js, яка надає ORM для зручного та ефективного взаємодії з реляційними базами даних. Вона дозволяє розробникам

використовувати об'єктно-орієнтовані концепції для взаємодії з базами даних, таким чином спрощуючи роботу з SQL-запитами та забезпечуючи більш зрозумілу та зручну роботу з даними.

pgAdmin 4 – було обрано через те що, це інтегроване середовище розробки для керування базами даних SQL, зокрема для роботи з PostgreSQL. Це потужний інструмент, який дозволяє розробникам і адміністраторам баз даних легко здійснювати управління та обслуговування баз даних через графічний інтерфейс.

Вказані інструментальні засоби забезпечують чудову продуктивність й швидкодію системи при великій кількості запитів та об'єму даних.

Для належної роботи вебсервісу, серверне обладнання має відповідати наступним вимогам. Оперативна пам'ять повинна бути не менше 32 GB, що дозволяє ефективно працювати з великими обсягами даних та забезпечує швидку обробку запитів. Для зберігання даних, потрібен жорсткий диск об'ємом 10 TB, що забезпечить необхідний обсяг для збереження інформації, файлів та резервних копій. Операційна система, яка використовується на сервері – Windows 10 22H2, що є сучасною та підтримує всі необхідні функції для роботи вебсервісу. Процесор має частоту не менше 3.5 GHz, що забезпечує високопродуктивну обробку запитів та швидке виконання завдань. Що стосується робочої станції користувача, то гаджет повинен бути оснащений браузером, що підтримує Google Chrome версії 80 і вище, Safari 10 і вище або iOS Safari 10 і вище, Firefox ESR або новіші версії, а також Opera 50 і вище, що гарантує коректну роботу вебсервісу на основних браузерах.

Час реакції програми на дії користувача (маніпуляції з пристроями введення даних) не повинен перевищувати 0,5 с.

Час виконання команд меню не більше 1 с.

Відображення масивів даних за запитами не більше 2 секунд.

Доступність БД – 90% цілодобово.

Операції з'єднання з БД не більше 2 секунд.

Програма повинна нормально функціонувати при безперебійній роботі ПК.

Висновки до першого розділу

Аналіз предметної області дозволив визначити основні вимоги та функціональні можливості вебдодатку, який спрощує проведення аналізу інвестиційного портфелю. У процесі дослідження аналогів програмного продукту було проаналізовано різні системи та платформи, що надають подібні послуги. Проведена порівняльна характеристика за конкуруючих продуктів за визначеними параметрами.

Аналіз архітектурних підходів допоміг нам визначити подальшу архітектуру проєкту, та дав зрозуміти які можуть виникнути проблеми в майбутньому при покращенні додатка.

На основі поставлених вимог, визначених критеріїв та обраних архітектурних рішень був сформований набір технологій та засобів для розробки програмного продукту. Крім того, були визначені апаратні вимоги для кожного елемента системи з метою ефективного використання ресурсів та відповідності програмного забезпечення усім визначеним критеріям та потребам.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ АНАЛІЗУ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ

2.1 Визначення варіантів використання

За ціллю візуалізації взаємозв'язків між компонентами системи, була створена діаграма варіантів використання (Рис. 2.1).



Рис. 2.1. Діаграма варіантів використання

Створення портфелю – користувач має змогу створити свій інвестиційний портфель для подальшого моніторингу за ним. Взаємодія з акціями портфелю – користувач має змогу додати до портфелю акції або видалити їх. Отримання аналізу портфелю – користувач має змогу замовити аналіз портфелю та отримати інформацію після успішного аналізу. Взаємодія з авторизацією та реєстрацією – користувач має змогу завести акаунт на сайті або увійти у свій акаунт.

Головна мета створення системи аналізу інвестиційного портфелю полягає в полегшенні проведення та отримання аналізу інвестиційних портфелів для користувачів. Проект буде реалізовано у вигляді вебсайту, який дозволить користувачам реєструватися, авторизуватися та створювати інвестиційні портфелі. Користувачі зможуть додавати опис до кожного портфелю та отримувати детальний аналіз своїх інвестицій. Для задоволення потреб користувачів визначено кілька ключових вимог: користувачі хочуть мати доступ до актуальної інформації про свої інвестиції та їхню ринкову вартість, можливість візуалізувати дані про дохідність, диверсифікацію портфелю, ризики через графіки та таблиці, а також просто і зручно додавати інвестиції. Інтерфейс сайту має бути простим, зрозумілим і зручним для користування, з можливістю авторизації та створення акаунтів. Нефункціональні вимоги включають забезпечення інтуїтивно зрозумілого інтерфейсу, мінімум помилок у роботі сайту (не більше 1 на 1000 запитів), а також високу продуктивність, що дозволяє безперебійно працювати з понад 50 користувачами одночасно.

За основу об'єктно-орієнтованої структури серверу було обрано архітектуру на основі фреймворка NestJS, яка поєднує різні аспекти FRP, OOP і FP та розділення відповідальностей. Основними структурними елементами серверу є: моделі-сутності, API-класи, контролери та сервіси.

Проаналізувавши вимоги користувачів, була створена база даних (Рис. 2.2).

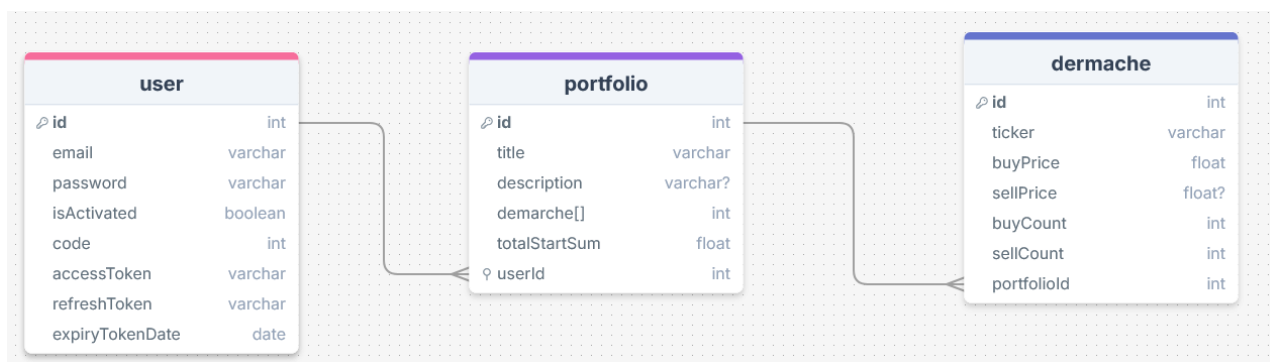


Рис. 2.2. Діаграма бази даних

User – модель-сутність, яка являє собою користувача. Містить наступні поля:

id (унікальне значення, тип int) – унікальний ідентифікатор користувача;
 email (унікальне значення, тип varchar) – електронна адреса;
 password (тип varchar) – пароль користувача;
 isActivated (тип boolean) – активований аккаунт;
 code (тип int) – код активації акаунту;
 accessToken (тип varchar) – токен доступу;
 refreshToken (тип varchar) – токен оновлення токена доступу;
 expiryTokenDate (тип date) – дата до якої діє токен оновлення;

Portfolio – модель-сутність, яка являє собою портфель. Містить наступні поля:

id (унікальне значення, тип int) – унікальний ідентифікатор портфелю;
 title (тип varchar) – назва портфелю;
 description (тип varchar) – опис портфелю;
 demarche (тип int) – масив id акцій;
 totalStartSum (тип float) – сума яку витратив користувач на покупку акцій;
 userId (тип int) – ідентифікатор користувача якому належить портфель.

Dermache – модель-сутність, яка являє собою куплені акції в портфелі. Містить наступні поля:

id (унікальне значення, тип int) – унікальний ідентифікатор акції;
 ticker (тип varchar) – тікер акції;
 buyPrice (тип float) – середня ціна покупки акції;
 sellPrice (тип float) – середня ціна продажу акції;
 buyCount (тип int) – кількість куплених акцій;
 sellCount (тип int) – кількість проданих акцій;
 portfolioId (тип int) – ідентифікатор портфелю якому належить акція;

Деталізовану структуру класів, що визначають основну логіку сервісу у системі, представлено у вигляді діаграми класів на рисунку 2.3.

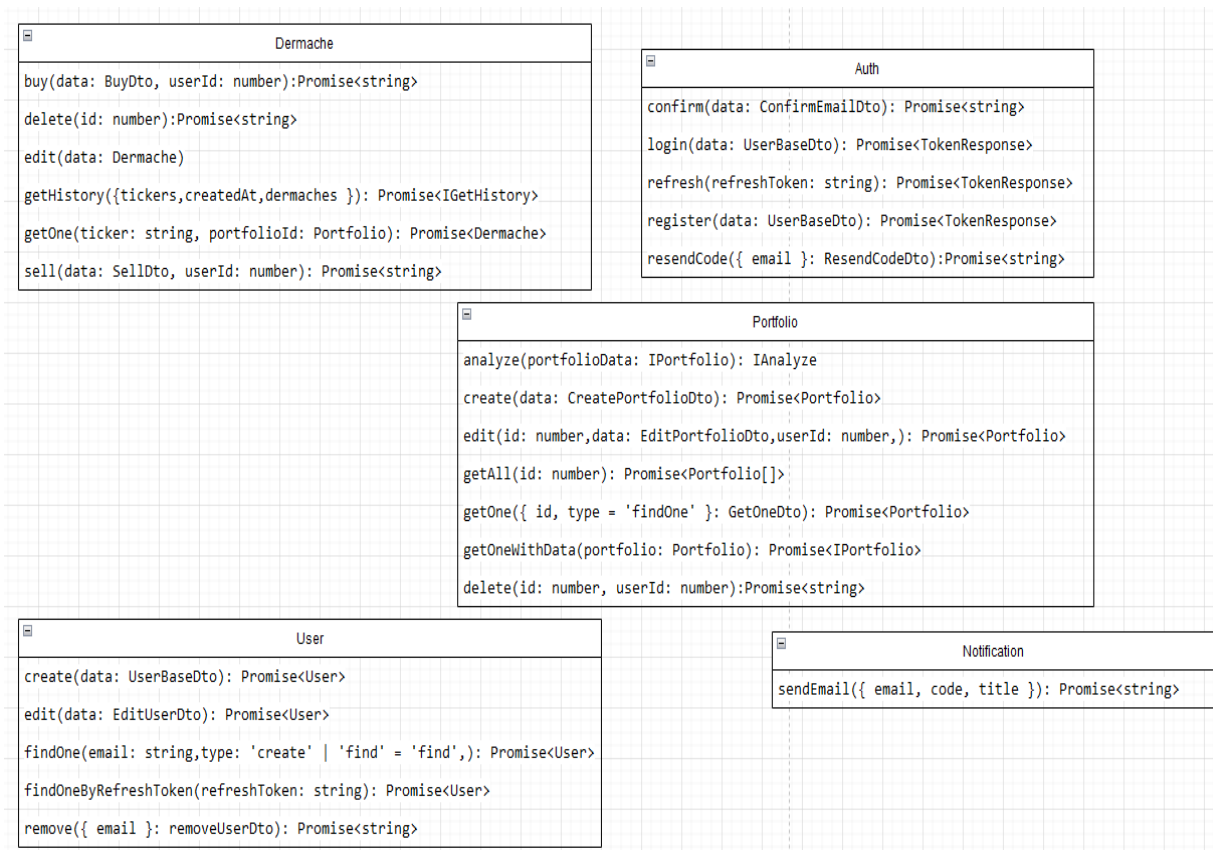


Рис. 2.3. Діаграма класів сервісів

User service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю User, зокрема передбачає створення користувача, оновлення, пошук, пошук за токеном оновлення та видаленням користувача з бази даних. Кожен з цих сервісів очікує на вхід данні, які типізовані та мають бути коректно введені. Для реалізації даного класу використані наступні методи:

`create(data: UserBaseDto): Promise<User>` – створює користувача. На вхід функція очікує об'єкт типу `UserBaseDto`. Цей об'єкт є обов'язковим і містить основні дані для створення користувача, зокрема він очікує `email`, `password` та перевірі їхню валідність, приклад об'єкту наведено нижче:

`UserBaseDto {`

`@IsEmail({}, { message: "Введіть коректний email" })`

`@IsNotEmpty({ message: "Поле email обов'язкове" })`

`email: string;`

`@MinLength(8, {`

```

    message: "Мінімальна довжина паролю повинна бути не менш ніж 8
символів",
  })
  @IsNotEmpty({ message: "Поле пароль обов'язкове" })
  password: string;
}

```

Нижче наведено приклад даних для створення користувача:

```

{
  "email": "zzombi200125@gmail.com",
  "password": "12345678"
}

```

На вихід повертає користувача.

`edit(data: EditUserDto): Promise<User>` – оновлює інформацію про користувача. На вхід функція очікує об'єкт типу `EditUserDto`. Цей об'єкт є обов'язковим і містить основні дані для створення користувача, зокрема він очікує `email`, `password`, `isActivated` та перевіряє їхню валідність, приклад об'єкту наведено нижче:

```

EditUserDto {
  @IsEmail({}, { message: "Введіть коректний email" })
  @IsNotEmpty({ message: "Поле email обов'язкове" })
  email: string;
  @MinLength(8, {
    message: «Мінімальна довжина паролю повинна бути не менш ніж 8
символів»,
  })
  @IsOptional()
  password?: string;
  @IsBoolean()
  @IsOptional()
  isActivated?: boolean;
}

```

```

    @IsNumber()
    @IsOptional()
    code?: number;
    @IsString()
    @IsOptional()
    accessToken?: string;
    @IsString()
    @IsOptional()
    refreshToken?: string;
    @IsDate()
    @IsOptional()
    expiryTokenDate?: Date;
}

```

Нижче наведено приклад даних для редагування користувача:

```

{
  "email": "zzombi200125@gmail.com",
  "password": "123454321"
}

```

`findOne(email: string, type: 'create' | 'find' = 'find'):` `Promise<User>` – повертає користувача за поштою, окрім цього, якщо передати тип пошуку можна використовувати в інших сервісах без помилки про те, що користувач не знайшовсь. Приклад даних для пошуку користувача:

```

{
  "email": "zzombi200125@gmail.com",
  "type": "create"
}

```

`findOneByRefreshToken(refreshToken: string):` `Promise<User>` – повертає користувача за токеном оновлення. Приклад даних для пошуку користувача:

```

{
  "refreshToken": "b18bf834-6bd8-4450-9838-624af489dcbf",
}

```

```
}
```

`remove({email}: removeUserDto): Promise<string>` – видаляє користувача за його поштою. На вхід функція очікує об'єкт типу `removeUserDto`. Цей об'єкт є обов'язковим і містить основні дані для видалення користувача, зокрема він очікує `email` та перевіряє його валідність, приклад об'єкту наведено нижче:

```
RemoveUserDto {
  @IsEmail({}, { message: "Введіть коректний email" })
  @IsNotEmpty({ message: "Поле email обов'язкове" })
  email: string;
}
```

Приклад даних для видалення користувача:

```
{
  "email": "zzombi200125@gmail.com",
}
```

`Notification service` – клас-сервіс, який містить операції з поштою. Зокрема передбачає відправлення листа на пошту користувача. Кожен з цих сервісів очікує на вхід данні, які типізовані та мають бути коректно введені. Для реалізації даного класу використані наступні методи:

`sendEmail({email, code, title}: NotificationDto): Promise<string>` – відправляє листа на пошту. На вхід функція очікує об'єкт який містить `email`, `code`, `title`. Цей об'єкт є обов'язковим і містить основні дані для користувача, зокрема він очікує `email` та перевіряє його валідність, приклад об'єкту наведено нижче:

```
NotificationDto {
  @IsString({ message: "Введіть коректний заголовок" })
  @IsNotEmpty({ message: "Поле title обов'язкове" })
  title: string;

  @IsNumber({}, { message: "Введіть коректний код має бути число з 4х чисел" })
  @IsNotEmpty({ message: "Поле code обов'язкове" })
```

```
code: number;
```

```
@IsEmail({}, { message: "Введіть коректний email" })
```

```
@IsNotEmpty({ message: "Поле email обов'язкове" })
```

```
email: string;
```

```
}
```

Приклад даних для відправки повідомлення:

```
{
```

```
"email": "zzombi200125@gmail.com",
```

```
"code": 1234,
```

```
"title": "Код реєстрації в додатку",
```

```
}
```

Portfolio service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю Portfolio, зокрема передбачає створення інвестиційного портфелю, оновлення, пошук, пошук всіх портфелів, видалення, та аналізом портфелю з бази даних. Кожен з цих сервісів очікує на вхід данні, які типізовані та мають бути коректно введені. Для реалізації даного класу використані наступні методи:

analyze(portfolioData: IPortfolio): IAnalyze – проводить аналіз портфелю. На вхід функція очікує об'єкт типу IPortfolio. Цей об'єкт є обов'язковим і містить основні дані для проведення аналізу портфелю, зокрема він очікує score, issues, details та перевірі їхню валідність, приклад об'єкту наведено нижче:

```
IPortfolio{
```

```
description?: string;
```

```
currentPrice: number,
```

```
profit: number,
```

```
profitPercent: number,
```

```
startPrice: number,
```

```
dermaches: IDermacheWithCurrentPrice[],
```

```
dermachesHistory: {
```

```

    averages: number[],
    range: string[]
  }
}

```

Де:

```

IDermacheWithCurrentPrice extends Dermache{
  currentPrice: number;
  profitPercent: number;
  profit: number;
  weight?: number;
  volatility?: number;
}

```

Приклад даних для проведення аналізу портфелю представлено нижче:

```

{
  description: 'местовий портфель',
  currentPrice: 4825.76,
  profit: 3278.76,
  profitPercent: 211.9431157078216,
  startPrice: 1547,
  dermaches: [
    {
      id: 7,
      ticker: 'PLTR.US',
      buyPrice: 0.5,
      buyCount: 10,
      sellCount: null,
      sellPrice: null,
      currentPrice: 75.72,
      profit: 752.2,
      profitPercent: 15144,
    }
  ]
}

```

```

    weight: 15.69079274559862,
    volatility: 3.769756097560976
  },
  {
    id: 8,
    ticker: 'KBWY.US',
    buyPrice: 0.5,
    buyCount: 10,
    sellCount: null,
    sellPrice: null,
    currentPrice: 19.16,
    profit: 186.6,
    profitPercent: 3832,
    weight: 3.9703590729750333,
    volatility: 0.3036585365853655
  }
],
dermachesHistory: {
  averages: [
    4572.65, 4546.360000000001,      4546.08,
    4487.42, 4404.799999999999,      4367.02,
    4353.3,   4314.13, 4227.719999999999,
    4234.960000000001,   4235.97,   4275.82,
    4373.14, 4464.799999999999,      4461.65,
    4499.33,   4508.73,   4464.21,
    4475.5,   4443.07,   4449.43,
    4434.71,   4478.29, 4504.309999999995,
    4612.32,   4664.19,   4706.5,
    4760.73,   4749.09,   4761.32,
    4754.49,   4823.6,    4852.25,

```

```

        4885.35, 4958.5300000000001,      4831.68,
        4840.34,      4811.31,      4839.89,
        4859.08,      4824.04
    ],
    range: [
        '2024-10-20', '2024-10-21', '2024-10-22',
        '2024-10-23', '2024-10-24', '2024-10-27',
        '2024-10-28', '2024-10-29', '2024-10-30',
        '2024-10-31', '2024-11-03', '2024-11-04',
        '2024-11-05', '2024-11-06', '2024-11-07',
        '2024-11-10', '2024-11-11', '2024-11-12',
        '2024-11-13', '2024-11-14', '2024-11-17',
        '2024-11-18', '2024-11-19', '2024-11-20',
        '2024-11-21', '2024-11-24', '2024-11-25',
        '2024-11-26', '2024-11-27', '2024-11-28',
        '2024-12-01', '2024-12-02', '2024-12-03',
        '2024-12-04', '2024-12-05', '2024-12-08',
        '2024-12-09', '2024-12-10', '2024-12-11',
        '2024-12-12', '2024-12-15'
    ]}]

```

На вихід нам повернеться ось такий об'єкт:

```

export interface IAnalyze {
    score: number;
    details: {
        roiScore: number;
        volatilityScore: number;
        diversificationScore: number;
        balanceScore: number;
    }
    issues: string[];
}

```

```
}
```

Приклад результату виконання сервісу наведено нижче:

```
{"score":100,
  "details":{
    "roiScore":40,
    "volatilityScore":30,
    "diversificationScore":20,
    "balanceScore":10},
  "issues":[]}
```

`create(data: CreatePortfolioDto): Promise<Portfolio>` – створює інвестиційний портфель. На вхід функція очікує об'єкт типу `CreatePortfolioDto`. Цей об'єкт є обов'язковим і містить основні дані для створення портфелю, зокрема він очікує `title`, `description`, `user` та перевіряє їхню валідність, приклад об'єкту наведено нижче:

```
CreatePortfolioDto {
  @IsString()
  title: string;
  @IsString()
  @IsOptional()
  description?: string;
  @IsNumber()
  @IsOptional()
  user?: User;
}
```

Приклад даних для створення портфелю:

```
{
  "title":"test3",
  "description": "test3"
}
```

`edit(id: number, data: EditPortfolioDto, userId: number): Promise<Portfolio>` – оновлює дані портфелю. На вхід функція очікує `id`, `userId`, `data` яка є об'єктом типу `EditPortfolioDto`. Цей об'єкт є обов'язковим і містить основні дані для оновлення портфелю, зокрема він очікує `title`, `description`, `dermaches` та перевіряє їхню валідність, приклад об'єкту наведено нижче:

```
{
  id: number,
  data: EditPortfolioDto,
  userId: number
}
```

Де:

```
EditPortfolioDto {
  @IsString()
  @IsOptional()
  title?: string;
  @IsString()
  @IsOptional()
  description?: string;
  @IsArray()
  @IsOptional()
  dermaches?: Dermache[];
}
```

Приклад даних для оновлення портфелю:

```
{
  "title": "test12",
  "description": "zc"
}
```

`getAll (id: number): Promise<Portfolio[]>` – отримує список всіх портфелів користувача. Очікує ідентифікатор користувача наприклад: 2.

`getOne({ id, type = 'findOne' }: GetOneDto): Promise<Portfolio>` – повертає портфель за ідентифікатором. На вхід функція очікує об'єкт типу `GetOneDto`. Цей об'єкт є обов'язковим і містить основні дані для отримання інформації по портфелю, зокрема він очікує `id`, `type`, та перевіряє їхню валідність, приклад об'єкту наведено нижче:

```
GetOneDto {
  @IsNumber()
  id: number;
  @IsEnum(['delete', 'findOne'])
  @IsOptional()
  type?: 'delete' | 'findOne' | 'edit';
}
```

Поле `type` є не обов'язковим полем, якщо його не відправляти на серверну частину, в випадку пошуку портфелю нам повернеться помилка якщо портфель не буде знайдений, в інших випадках – просто повернеться `undefined`. Приклад даних для пошуку портфелю:

```
{
  "id": "2"
}
```

`getOneWithData(portfolio: Portfolio): Promise<IPortfolio>` – повертає портфель з прорахованими даними. На вхід функція очікує об'єкт типу `Portfolio` який є моделлю сутності `Portfolio`. Цей об'єкт є обов'язковим і містить основні дані для отримання повної інформації по портфелю. Приклад даних для пошуку портфелю з прорахованими даними:

```
Portfolio {
  id: 9,
  title: 'test9',
  description: 'тестовий портфель',
  totalStartSum: 0,
  createdAt: 2024-10-18T21:05:13.320Z,
```

```
dermaches: [  
  Dermache {  
    id: 7,  
    ticker: 'PLTR.US',  
    buyPrice: 0.5,  
    buyCount: 10,  
    sellCount: null,  
    sellPrice: null  
  },  
  Dermache {  
    id: 8,  
    ticker: 'KBWY.US',  
    buyPrice: 0.5,  
    buyCount: 10,  
    sellCount: null,  
    sellPrice: null  
  },  
  Dermache {  
    id: 5,  
    ticker: 'SWVL.US',  
    buyPrice: 50,  
    buyCount: 10,  
    sellCount: 10,  
    sellPrice: 60  
  },  
  Dermache {  
    id: 11,  
    ticker: 'PEP.US',  
    buyPrice: 2,  
    buyCount: 3,
```

```

        sellCount: null,
        sellPrice: null
    } ]
}

```

На повернення очікуєм ось такий об'єкт:

```

IPortfolio{
  description?: string;
  currentPrice: number,
  profit: number,
  profitPercent: number,
  startPrice: number,
  dermaches: IDermacheWithCurrentPrice[],
  dermachesHistory: {
    averages: number[],
    range: string[]
  }
}

```

`delete(id: number, userId: number): Promise<string>` – видаляє портфель. Очікує ідентифікатор портфелю та ідентифікатор користувача наприклад 1 та 2;

`Dermache service` – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю `Dermache`, зокрема передбачає купівлю акції, видалення, оновлення, отримання історії активів, отримання акції, та продаж акції з бази даних. Кожен з цих сервісів очікує на вхід данні, які типізовані та мають бути коректно введені. Для реалізації даного класу використані наступні методи:

`buy(data: BuyDto, userId: number): Promise<string>` – додає акцію до портфелю. На вхід функція очікує `userId`, `data` яка є об'єктом типу `BuyDto`. Цей об'єкт є обов'язковим і містить основні дані для оновлення портфелю, зокрема він очікує `userId`, `ticker`, `buyPrice`, `buyCount`, `portfilio` та перевіє їх валідність, приклад об'єкту наведено нижче:

```

BuyDto {
  @IsNumber()
  @IsOptional()
  userId?: number;
  @IsString()
  ticker: string;
  @IsNumber()
  buyPrice: number;
  @IsNumber()
  buyCount: number;
  @IsNumber()
  portfolio: Portfolio;
}

```

Приклад даних для додання акції:

```

{
  "ticker": "SPYD.US",
  "buyPrice": 50,
  "buyCount": 10,
  "portfolio": 9
}, 2

```

Повертає повідомлення про успішну операцію: "Ваша Акція успішно додана до портфелю" також в випадку помилки можуть бути й інші повідомлення, наприклад коли акцію яку ви купуєте не існує: "Нажаль такої акції не існує на нашому ринку", а також, коли ви намагаєтесь додати акцію до чужого портфелю – "Ви не можете купувати акції для чужих портфелів".

`delete(id: number): Promise<string>` – видаляє акцію за ідентифікатором наприклад 1 та повертає повідомлення про успішну операцію "Успішно";

`edit (data: Dermache)` – редагує акцію, очікує на вхід модель сутність `Dermache` та нічого не повертає;

`getHistory({tickers, createdAt, dermaches} : IGetHistoryDto): Promise<IGetHistory>` – отримує ціну акцій по тикерам та повертає її з датами. На вхід функція очікує об'єкт типу `IGetHistory`. Цей об'єкт є обов'язковим і містить основні дані для створення портфелю, зокрема він очікує `tickers`, `createdAt`, `dermaches` та перевіряє їхню валідність, приклад об'єкту наведено нижче:

```
IGetHistoryDto {
  tickers: string;
  createdAt: Date;
  dermaches: IDermacheWithCurrentPrice[]
}
```

Приклад даних для отримання ціни акцій:

```
{
  "tickers" : "PLTR.US,KBWY.US",
  "createdAt": "2024-10-18T21:05:13.320Z",
  "dermaches": [
    {
      id: 7,
      ticker: 'PLTR.US',
      buyPrice: 0.5,
      buyCount: 10,
      sellCount: null,
      sellPrice: null,
      currentPrice: 75.36,
      profit: 748.6,
      profitPercent: 15072,
      weight: 15.668402055854028
    },
    {
      id: 8,
      ticker: 'KBWY.US',
```

```

    buyPrice: 0.5,
    buyCount: 10,
    sellCount: null,
    sellPrice: null,
    currentPrice: 19.13,
    profit: 186.3,
    profitPercent: 3826,
    weight: 3.9773955855691017
  }}

```

`getOne (ticker: string, portfolioId: Portfolio): Promise<Dermache >` – повертає інформацію по акції за її тикером та ідентифікатором портфоліо в якому вона лежить. Приклад даних: PLTR.US, 9;

`sell (sell: SellDto, userId: number): Promise<string >` – додає інформацію про продаж акції в базу даних. На вхід функція очікує об'єкт типу `SellDto` та `userId`. Цей об'єкт є обов'язковим і містить основні дані для створення портфелю, зокрема він очікує `userId`, `ticker`, `sellPrice`, `sellCount`, `portfolio` та перевіє їхню валідність, приклад об'єкту наведено нижче:

```

SellDto {
  @IsNumber()
  @IsOptional()
  userId?: number;
  @IsString()
  ticker: string;
  @IsNumber()
  sellPrice: number;
  @IsNumber()
  sellCount: number;
  @IsNumber()
  portfolio: Portfolio;
}

```

Приклад даних:

```
{
  "ticker": "SPYD.US",
  "sellPrice": 10,
  "sellCount": 10,
  "portfolio": 9
}, 4
```

Auth service – клас-сервіс, який містить операції з базою даних для виконання операцій з сутністю User, зокрема передбачає авторизацію, підтвердження пошти, оновлення токenu авторизації, реєстрації та перевідправки коду на пошту з бази даних. Кожен з цих сервісів очікує на вхід данні, які типізовані та мають бути коректно введені. Для реалізації даного класу використані наступні методи:

login(dto: UserBaseDto): Promise<TokenResponse> – виконує вхід у систему під певним користувачем. На вхід функція очікує об'єкт типу UserBaseDto. Цей об'єкт є обов'язковим і містить основні дані для авторизації, зокрема він очікує email, password та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
UserBaseDto {
  @IsEmail({}, { message: 'Введіть коректний email' })
  @NotEmpty({ message: "Поле email обов'язкове" })
  email: string;
  @MinLength(8, {
    message: 'Мінімальна довжина паролю повинна бути не менш ніж 8 символів',
  })
  @NotEmpty({ message: "Поле пароль обов'язкове" })
  password: string;
}
```

Приклад даних:

```
{
```

```
"email": "zzombi200125@gmail.com",
"password": "123454321"
}
```

Повертає об'єкт наступного вигляду:

```
export interface TokenResponse {
  accessToken: string;
  refreshToken: string;
  expiryTokenDate: Date;
}
```

Результат виконання матиме вигляд:

```
{
  "accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWZpbCI6Inp6b21iaTIwMDEyNUBnbWZpbC5jb20iLCJpZCI6NSwiaXNBY3RpdmF0ZWQiOnRydWUsImVudCI6MTczNDM4NDMyNywiZXhwIjozMDNDcwNzI3fQ.lFeoGmsEWY98umMAWFyt1_pz_a_nrUqPMIrpo8Wo77A",
  "refreshToken": "fc668b90-96c7-4e7d-bf55-41b522521cbd",
  "expiryTokenDate": "2024-12-19T21:25:27.474Z"}
}
```

`confirm(dto: ConfirmEmailDto): Promise<string>` – переводить аккаунт в статус активованого в випадку коректних даних. На вхід функція очікує об'єкт типу `ConfirmEmailDto`. Цей об'єкт є обов'язковим і містить основні дані для підтвердження пошти, зокрема він очікує `email`, `code` та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
ConfirmEmailDto {
  @IsEmail({}, { message: 'Введіть коректний email' })
  @NotEmpty({ message: "Поле email обов'язкове" })
  email: string;
  @NotEmpty({ message: "Поле код обов'язкове" })
  @IsNumber()
  code: number;}
}
```

Приклад даних:

```
{"email": "zhurbeyartem@gmail.com",  
  "code": 9011}
```

Якщо код правильний, то користувач отримає відповідне повідомлення "Акаунт успішно активовано", якщо код введений не коректно, то він отримає ось таке повідомлення: "Невірний код".

refresh (refreshToken: string): Promise<TokenResponse> – *перегенерація refresh токена. Очікує на вхід токен оновлення, наприклад: `fc668b90-96c7-4e7d-bf55-41b522521cbd`*

Об'єкт результату:

```
interface TokenResponse {  
  accessToken: string;  
  refreshToken: string;  
  expiryTokenDate: Date;}
```

Результат матиме вигляд:

```
{"accessToken":  
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Inp6b21iaTlwMDEyNUBnbWFpbC5jb20iLCJpZCI6NSwiaXNBY3RpdmF0ZWQiOnRydWUsImVudCI6MTczNDM4NDMyNywiZXhwIjoxNzI1MjUyNzI1fQ.lFeoGmsEWY98umMAWFyt1_pz_a_nrUqPMIrpo8Wo77A",  
  "refreshToken": "fc668b90-96c7-4e7d-bf55-41b522521cbd",  
  "expiryTokenDate": "2024-12-19T21:25:27.474Z"}
```

register(data: UserBaseDto): Promise<TokenResponse> – *реєстрація користувача в системі. Об'єкт який очікується на вхід та вихід, а також дані такіж самі, як в сервісі login.*

resendCode({email}: ResendCodeDto): Promise<string> – *генерує новий код активації акаунта в базі даних та відправляє листа на пошту користувача. На вхід функція очікує об'єкт типу ResendCodeDto. Цей об'єкт є обов'язковим і містить основні дані для “перевідправки” коду на пошту, зокрема він очікує email та перевірі їхню валідність, приклад об'єкту наведено нижче:*

```
ResendCodeDto {
  @IsEmail({}, {
    message: "Введіть коректний email"
  })
  @IsNotEmpty({
    message: "Поле email обов'язкове"
  })
  email: string;}

```

Приклад даних:

```
{
  "email": zzombi200125@gmail.com
}
```

Результатом даного сервісу є повідомлення про успішне виконання операції: “Код успішно відправлений на пошту”

Діаграма класів, що ілюструє сервіси для обробки та управління даними, представлена на рисунку 2.4.

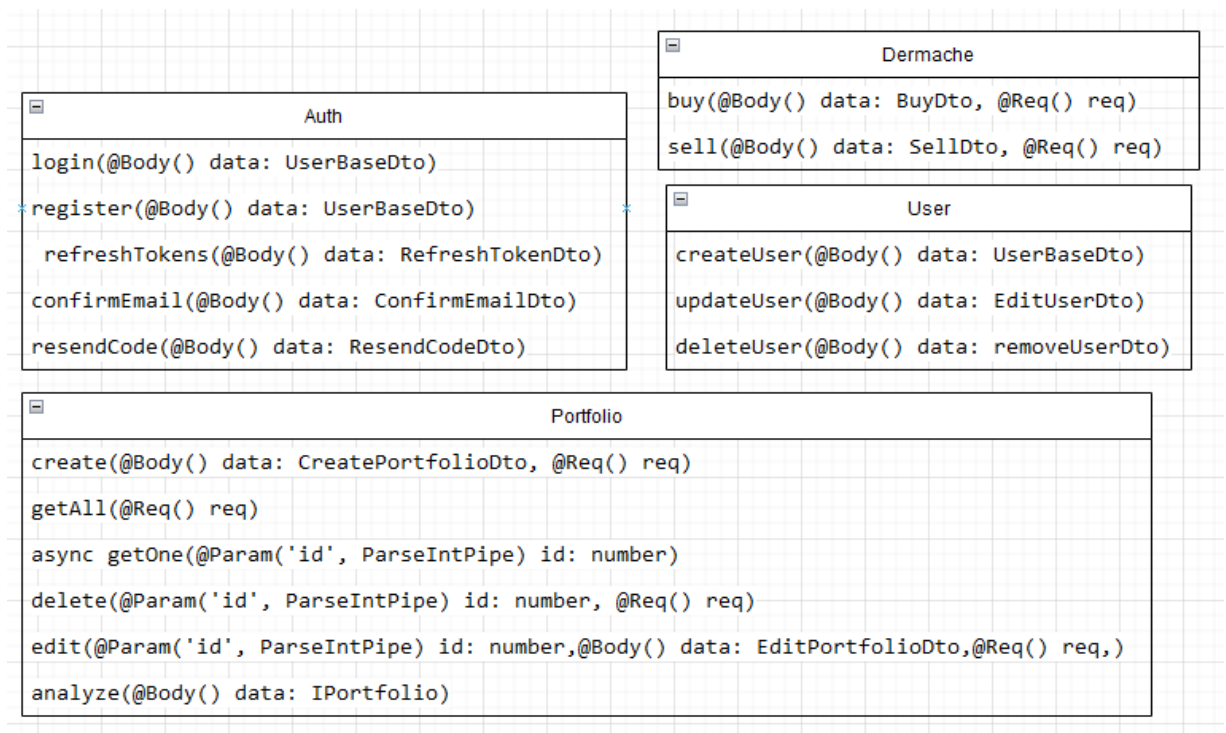


Рис. 2.4. Діаграма класів API-класів

Auth controller – клас, який містить методи операцій обробки даних та роботи з сутністю User, зокрема передбачає авторизацію, підтвердження пошти, оновлення токена авторизації, реєстрації та перевідправки коду на пошту даних. Кожен з цих сервісів очікує на вхід данні, які типізовані та мають бути коректно введені. Для реалізації даного класу використані наступні методи:

login(@Body() data: UserBaseDto) – авторизує користувача в системі. На вхід функція очікує об'єкт типу UserBaseDto. Цей об'єкт є обов'язковим і містить основні дані для авторизації, зокрема він очікує email, password та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
UserBaseDto {
    @IsEmail({}, { message: 'Введіть коректний email' })
    @NotEmpty({ message: "Поле email обов'язкове" })
    email: string;
    @MinLength(8, {
        message: 'Мінімальна довжина паролю повинна бути не менш ніж 8 символів',
    })
    @NotEmpty({ message: "Поле пароль обов'язкове" })
    password: string;
}
```

Приклад даних:

```
{
    "email": "zzombi200125@gmail.com",
    "password": "123454321"
}
```

Повертає об'єкт у наступному вигляді:

```
export interface TokenResponse {
    accessToken: string;
    refreshToken: string;
    expiryTokenDate: Date;
```

```
}
```

Результат виконання матиме вигляд:

```
{
```

```
  "accessToken":
```

```
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Inp6b21iaTIwMDEyNUBnbWFpbC5jb20iLCJpZCI6NSwiaXNBY3RpdmF0ZWQiOnRydWUsImVudCI6MTczNDM4NDMyNywiZXhwIjozM0NDcwNzI3fQ.lFeoGmsEWY98umMAWFyt1_pz_a_nrUqPMIrpo8Wo77A",
```

```
  "refreshToken": "fc668b90-96c7-4e7d-bf55-41b522521cbd",
```

```
  "expiryTokenDate": "2024-12-19T21:25:27.474Z"}
}
```

`register(@Body() data: UserBaseDto)` – реєструє користувача в базі даних. Об'єкт який очікується на вхід та вихід а також данні такіж самі як в котролері `login`.

`refreshTokens(@Body() data: RefreshTokenDto)` – регенерація refresh токєну. Очікує на вхід токен оновлення, наприклад: “fc668b90-96c7-4e7d-bf55-41b522521cbd”

Об'єкт результату матимє наступний вигляд:

```
interface TokenResponse {
```

```
  accessToken: string;
```

```
  refreshToken: string;
```

```
  expiryTokenDate: Date;}
}
```

Результат:

```
{"accessToken":
```

```
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Inp6b21iaTIwMDEyNUBnbWFpbC5jb20iLCJpZCI6NSwiaXNBY3RpdmF0ZWQiOnRydWUsImVudCI6MTczNDM4NDMyNywiZXhwIjozM0NDcwNzI3fQ.lFeoGmsEWY98umMAWFyt1_pz_a_nrUqPMIrpo8Wo77A",
```

```
  "refreshToken": "fc668b90-96c7-4e7d-bf55-41b522521cbd",
```

```
  "expiryTokenDate": "2024-12-19T21:25:27.474Z"}
}
```

`confirmEmail(@Body() data: ConfirmEmailDto)` – підтвердження пошти. На вхід функція очікує об'єкт типу `ConfirmEmailDto`. Цей об'єкт є обов'язковим і містить основні дані для підтвердження пошти, зокрема він очікує `email`, `code` та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
ConfirmEmailDto {
    @IsEmail({}, { message: 'Введіть коректний email' })
    @NotEmpty({ message: "Поле email обов'язкове" })
    email: string;
    @NotEmpty({ message: "Поле код обов'язкове" })
    @IsNumber()
    code: number;}

```

Приклад даних:

```
{"email": "zhurbeyartem@gmail.com",
  "code": 9011}
```

`resendCode(@Body() data: ResendCodeDto)` – повторне відправлення коду активації акаунту на пошту. На вхід функція очікує об'єкт типу `ResendCodeDto`. Цей об'єкт є обов'язковим і містить основні дані для перевідправки коду на пошту, зокрема він очікує `email` та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
ResendCodeDto {
    @IsEmail({}, { message: 'Введіть коректний email' })
    @NotEmpty({ message: "Поле email обов'язкове" })
    email: string;}

```

Приклад даних:

```
{"email": "zzombi200125@gmail.com"}
```

`Dermache controller` – клас, який містить методи операцій обробки даних та роботи з сутністю `Dermache`, зокрема передбачає купівлю акції, видалення, та продаж акції з бази даних. Кожен з цих сервісів очікує на вхід данні, які типізовані та мають бути коректно введені. Для реалізації даного класу використані наступні методи:

`buy(@Body() data: BuyDto, @Req() req)` – повертає список ставок за вхідними параметрами. На вхід функція очікує `data` яка є об'єктом типу `BuyDto`. Цей об'єкт є обов'язковим і містить основні дані для оновлення портфелю, зокрема він очікує `userId`, `ticker`, `buyPrice`, `buyCount`, `portfolio` та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
BuyDto {
  @IsNumber()
  @IsOptional()
  userId?: number;
  @IsString()
  ticker: string;
  @IsNumber()
  buyPrice: number;
  @IsNumber()
  buyCount: number;
  @IsNumber()
  portfolio: Portfolio;
}
```

Приклад даних для додавання акції матиме наступний вигляд:

```
{
  "ticker": "SPYD.US",
  "buyPrice": 50,
  "buyCount": 10,
  "portfolio": 9
}, 2
```

`sell(@Body() data: SellDto, @Req() req)` – повертає кількість ставок на роботу для конкретного користувача. На вхід функція очікує об'єкт типу `SellDto`. Цей об'єкт є обов'язковим і містить основні дані для створення портфелю, зокрема він очікує `userId`, `ticker`, `sellPrice`, `sellCount`, `portfolio` та перевіє їхню валідність, приклад об'єкту наведено нижче:

```

SellDto {
    @IsNumber()
    @IsOptional()
    userId?: number;
    @IsString()
    ticker: string;
    @IsNumber()
    sellPrice: number;
    @IsNumber()
    sellCount: number;
    @IsNumber()
    portfolio: Portfolio;
}

```

Приклад даних матиме наступний вигляд:

```

{
    "ticker": "SPYD.US",
    "sellPrice": 10,
    "sellCount": 10,
    "portfolio": 9
}, 4

```

`deleteDermache(@Param('id', ParseIntPipe) id: number)` – видаляє акцію з портфелю на вхід очікує ідентифікатор акції наприклад 45.

`User controller` – клас, який містить методи операцій обробки даних та роботи з сутністю `User`, зокрема передбачає створення користувача, оновлення та видалення користувача з бази даних. Кожен з цих сервісів очікує на вхід данні, які типізовані та мають бути коректно введені. Для реалізації даного класу використані наступні методи:

`createUser(@Body() data: UserBaseDto)` – створює користувача. На вхід очікує об'єкт типу `UserBaseDto`. Цей об'єкт є обов'язковим і містить основні дані

для створення користувача, зокрема він очікує email, password та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
UserBaseDto {
    @IsEmail({}, { message: 'Введіть коректний email' })
    @IsNotEmpty({ message: "Поле email обов'язкове" })
    email: string;
    @MinLength(8, {
        message: "Мінімальна довжина паролю повинна бути не менш ніж 8 символів",
    })
    @IsNotEmpty({ message: "Поле пароль обов'язкове" })
    password: string;
}
```

Приклад даних для створення користувача наведено нижче:

```
{
    "email": "zzombi200125@gmail.com",
    "password": "12345678"
}
```

updateUser(@Body() data: EditUserDto) – змінює данні користувачу. На вхід очікує об'єкт типу EditUserDto. Цей об'єкт є обов'язковим і містить основні дані для створення користувача, зокрема він очікує email, password, isActivated та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
EditUserDto {
    @IsEmail({}, { message: "Введіть коректний email" })
    @IsNotEmpty({ message: "Поле email обов'язкове" })
    email: string;
    @MinLength(8, {
        message: "Мінімальна довжина паролю повинна бути не менш ніж 8 символів",
    })
    password: string;
    isActivated: boolean;
```

```

    @IsOptional()
    password?: string;
    @IsBoolean()
    @IsOptional()
    isActivated?: boolean;
    @IsNumber()
    @IsOptional()
    code?: number;
    @IsString()
    @IsOptional()
    accessToken?: string;
    @IsString()
    @IsOptional()
    refreshToken?: string;
    @IsDate()
    @IsOptional()
    expiryTokenDate?: Date;
}

```

Приклад даних для редагування користувача:

```

{
  "email": "zzombi200125@gmail.com",
  "password": "123454321"
}

```

`deleteUser(@Body() data: RemoveUserDto)` – видаляєм користувача. На вхід очікує об'єкт типу `RemoveUserDto`. Цей об'єкт є обов'язковим і містить основні дані для видалення користувача, зокрема він очікує `email` та перевіряє його валідність, приклад об'єкту наведено нижче:

```

{
  "email": "zzombi200125@gmail.com",
}

```

Portfolio controller – клас, який містить методи операцій обробки даних та роботи з сутністю Portfolio, зокрема передбачає створення інвестиційного портфелю, оновлення, пошук за ідентифікатором, пошук всіх портфелів, видалення, та аналізом портфелю з бази даних. Кожен з цих сервісів очікує на вхід данні, які типізовані та мають бути коректно введені. Для реалізації даного класу використані наступні методи:

`create(@Body() data: CreatePortfolioDto, @Req() req)` – створення портфелю. На вхід очікує об'єкт типу `CreatePortfolioDto`. Цей об'єкт є обов'язковим і містить основні дані для створення портфелю, зокрема він очікує `title`, `description`, `user` та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
CreatePortfolioDto {
  @IsString()
  title: string;
  @IsString()
  @IsOptional()
  description?: string;
  @IsNumber()
  @IsOptional()
  user?: User;
}
```

Приклад даних для створення портфелю матиме вигляд:

```
{
  "title": "test3",
  "description": "test3"
}
```

`getAll(@Req() req)` – повертає портфелі які належать користувачу;

`getOne(@Param('id', ParseIntPipe) id: number)` – повертає портфель, за ідентифікатором, наприклад 3;

`delete@Param('id', ParseIntPipe) id: number, @Req() req)` – видалення портфелю за ідентифікатором, наприклад 2;

`edit(@Param('id', ParseIntPipe) id: number, @Body() data: EditPortfolioDto, @Req() req)` – редагування портфелю. На вхід функція очікує `id`, `userId`, `data`, яка є об'єктом типу `EditPortfolioDto`. Цей об'єкт є обов'язковим і містить основні дані для оновлення портфелю, зокрема він очікує `title`, `description`, `dermaches` та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
{
  id: number,
  data: EditPortfolioDto
}
```

Де:

```
EditPortfolioDto {
  @IsString()
  @IsOptional()
  title?: string;
  @IsString()
  @IsOptional()
  description?: string;
  @IsArray()
  @IsOptional()
  dermaches?: Dermache[];
}
```

`analyze(@Body() data: IPortfolio)` – аналіз портфелю. Цей об'єкт є обов'язковим і містить основні дані для проведення аналізу портфелю, зокрема він очікує `score`, `issues`, `details` та перевіє їхню валідність, приклад об'єкту наведено нижче:

```
IPortfolio{
  description?: string;
  currentPrice: number,
  profit: number,
  profitPercent: number,
```

```

    startPrice: number,
    dermaches: IDermacheWithCurrentPrice[],
    dermachesHistory: {
      averages: number[],
      range: string[]
    }
  }
}

```

Де:

```

IDermacheWithCurrentPrice extends Dermache{
  currentPrice: number;
  profitPercent: number;
  profit: number;
  weight?: number;
  volatility?: number;
}

```

Приклад даних для проведення аналізу портфелю матиме наступний вигляд:

```

{
  description: 'местовий портфель',
  currentPrice: 4825.76,
  profit: 3278.76,
  profitPercent: 211.9431157078216,
  startPrice: 1547,
  dermaches: [
    {
      id: 7,
      ticker: 'PLTR.US',
      buyPrice: 0.5,
      buyCount: 10,
      sellCount: null,
      sellPrice: null,
    }
  ]
}

```

```

currentPrice: 75.72,
profit: 752.2,
profitPercent: 15144,
weight: 15.69079274559862,
volatility: 3.769756097560976
},
{
id: 8,
ticker: 'KBWY.US',
buyPrice: 0.5,
buyCount: 10,
sellCount: null,
sellPrice: null,
currentPrice: 19.16,
profit: 186.6,
profitPercent: 3832,
weight: 3.9703590729750333,
volatility: 0.3036585365853655
}
],
dermachesHistory: {
averages: [
4572.65, 4546.360000000001, 4546.08,
4487.42, 4404.799999999999, 4367.02,
4353.3, 4314.13, 4227.719999999999,
4234.960000000001, 4235.97, 4275.82,
4373.14, 4464.799999999999, 4461.65,
4499.33, 4508.73, 4464.21,
4475.5, 4443.07, 4449.43,
4434.71, 4478.29, 4504.3099999999995,

```

```

4612.32, 4664.19, 4706.5,
4760.73, 4749.09, 4761.32,
4754.49, 4823.6, 4852.25,
4885.35, 4958.530000000001, 4831.68,
4840.34, 4811.31, 4839.89,
4859.08, 4824.04
],
range: [
'2024-10-20', '2024-10-21', '2024-10-22',
'2024-10-23', '2024-10-24', '2024-10-27',
'2024-10-28', '2024-10-29', '2024-10-30',
'2024-10-31', '2024-11-03', '2024-11-04',
'2024-11-05', '2024-11-06', '2024-11-07',
'2024-11-10', '2024-11-11', '2024-11-12',
'2024-11-13', '2024-11-14', '2024-11-17',
'2024-11-18', '2024-11-19', '2024-11-20',
'2024-11-21', '2024-11-24', '2024-11-25',
'2024-11-26', '2024-11-27', '2024-11-28',
'2024-12-01', '2024-12-02', '2024-12-03',
'2024-12-04', '2024-12-05', '2024-12-08',
'2024-12-09', '2024-12-10', '2024-12-11',
'2024-12-12', '2024-12-15'
]}}

```

Приклад результату виконання сервісу матиме вигляд:

```

{"score":100,
"details":{
"roiScore":40,
"volatilityScore":30,
"diversificationScore":20,
"balanceScore":10},

```

```
"issues":[]}
```

Слід зауважити на тому, що контролерів менше ніж сервісів та вони завжди очікують ті ж самі дані що й сервіси. На наступному етапі проведемо розробку бази даних.

2.2 Розробка бази даних системи

Для збереження сутностей та взаємодії з даними у нашій системі було розроблено базу даних, таблиці якої віддзеркалюють моделі-сутності за допомогою ORM. Ми розглянули п'ять популярних ORM: TypeORM, Sequelize, Prisma, Mongoose та Knex.js, щоб обрати найбільш відповідний інструмент для нашого вебсервісу. Після аналізу було обрано TypeORM, оскільки він найкраще відповідає нашим вимогам та архітектурі.

TypeORM є потужним інструментом для роботи з базами даних, який підтримує TypeScript "з коробки". Це дозволяє зручно створювати типізовані моделі, використовуючи декоратори, що спрощують опис сутностей, їх зв'язків і конфігурацій. Окрім цього, TypeORM підтримує широкий спектр баз даних, таких як PostgreSQL, MySQL, SQLite та MongoDB, що робить його гнучким вибором для різних проєктів. Важливою перевагою є система міграцій, яка дозволяє легко керувати змінами структури бази даних, а також вбудована система кешування, що допомагає оптимізувати складні запити. Однак TypeORM має й недоліки, серед яких складність налаштування для початківців і потенційні проблеми з продуктивністю при роботі з великими базами даних.

Серед інших розглянутих ORM варто згадати Sequelize, який пропонує зручний синтаксис для роботи з моделями, але не має повної підтримки TypeScript і вимагає більше ручної роботи для створення зв'язків і міграцій. Prisma, навпаки, має чудову інтеграцію з TypeScript і автоматично генерує моделі з бази даних, але обмежена у підтримці складних зв'язків і сумісності з деякими базами даних. Mongoose добре підходить для MongoDB завдяки своїй простоті й спеціалізації, але не підтримує реляційні бази даних, які нам потрібні. Knex.js є

query builder'ом, а не повноцінною ORM, що ускладнює роботу із сутностями та зв'язками.

TypeORM було обрано через його відповідність архітектурі нашого проєкту, інтеграцію з TypeScript, підтримку складних зв'язків між таблицями та зручність роботи з реляційними базами даних, такими як PostgreSQL. Це забезпечує ефективну розробку й масштабованість системи. Взаємодія з базою даних у нашій системі реалізується через контролери, що дозволяє чітко розділити логіку бізнес-процесів та роботу з даними. База даних сайту складається з 3 таблиць:

1. user – таблиця, яка зберігає дані про користувачів системи, та має зв'язок 1 до багатьох з таблицею portfolio;
2. portfolio – таблиця, яка зберігає дані про портфелі, та має зв'язок, 1 до багатьох з таблицею dermache, а також багато до 1 з таблицею user;
3. dermache – таблиця, яка зберігає дані про акції, та має зв'язок багато до 1 з таблицею portfolio.

Нижче наведений опис кожної з наведених таблиць бази даних (див. табл. 2.1 – 2.3)

Таблиця 2.1

Опис сутності User

Поле	Тип	Призначення
Id	Uniqueidentifier, PK	Унікальний ідентифікатор
email	VARCHAR(255)	Email адреса
password	VARCHAR(255)	Пароль
isActivated	BOOLEAN	Статус аккаунту
code	INT	Код активації
accessToken	VARCHAR(255)	Токен доступу
refreshToken	VARCHAR(255)	Токен оновлення
expiryTokenDate	DATE	Дата закінчення токена оновлення

Таблиця 2.2

Опис сутності Portfolio

Поле	Тип	Призначення
id	Uniqueidentifier, PK	Унікальний ідентифікатор
title	VARCHAR(255)	Заголовок
description	VARCHAR(255)	Опис
userId	INT, FK	Ідентифікатор власника портфелю

Таблиця 2.3

Опис сутності Dermache

Поле	Тип	Призначення
Id	Uniqueidentifier, PK	Унікальний ідентифікатор
ticker	VARCHAR(255)	Тікер
buyPrice	FLOAT	Середня ціна покупки
sellPrice	FLOAT	Середня ціна продажу
buyCount	INT	Кількість куплених
sellCount	INT	Кількість проданих
portfolioId	INT, FK	Ідентифікатор портфелю

На наступному етапі проведемо проєктування та реалізацію алгоритмів роботи системи.

2.3 Проєктування та реалізація алгоритмів роботи системи

Система аналізу інвестиційних портфелів побудована на компонентній архітектурі, що дозволяє досягти високої модульності та масштабованості. Кожен компонент відповідає за виконання окремої бізнес-логіки, а взаємодія між ними забезпечується через чітко визначені інтерфейси. Така архітектура дозволяє легко оновлювати, масштабувати та підтримувати систему, що є важливим аспектом для майбутнього розширення функціоналу.

Для того, щоб забезпечити зручність використання та ефективність роботи, кожен етап взаємодії користувача з вебсайтом передбачає чітке відображення результатів операцій. Діаграма активності дає змогу наочно простежити процеси, які відбуваються під час взаємодії користувача з системою, рис.2.5.

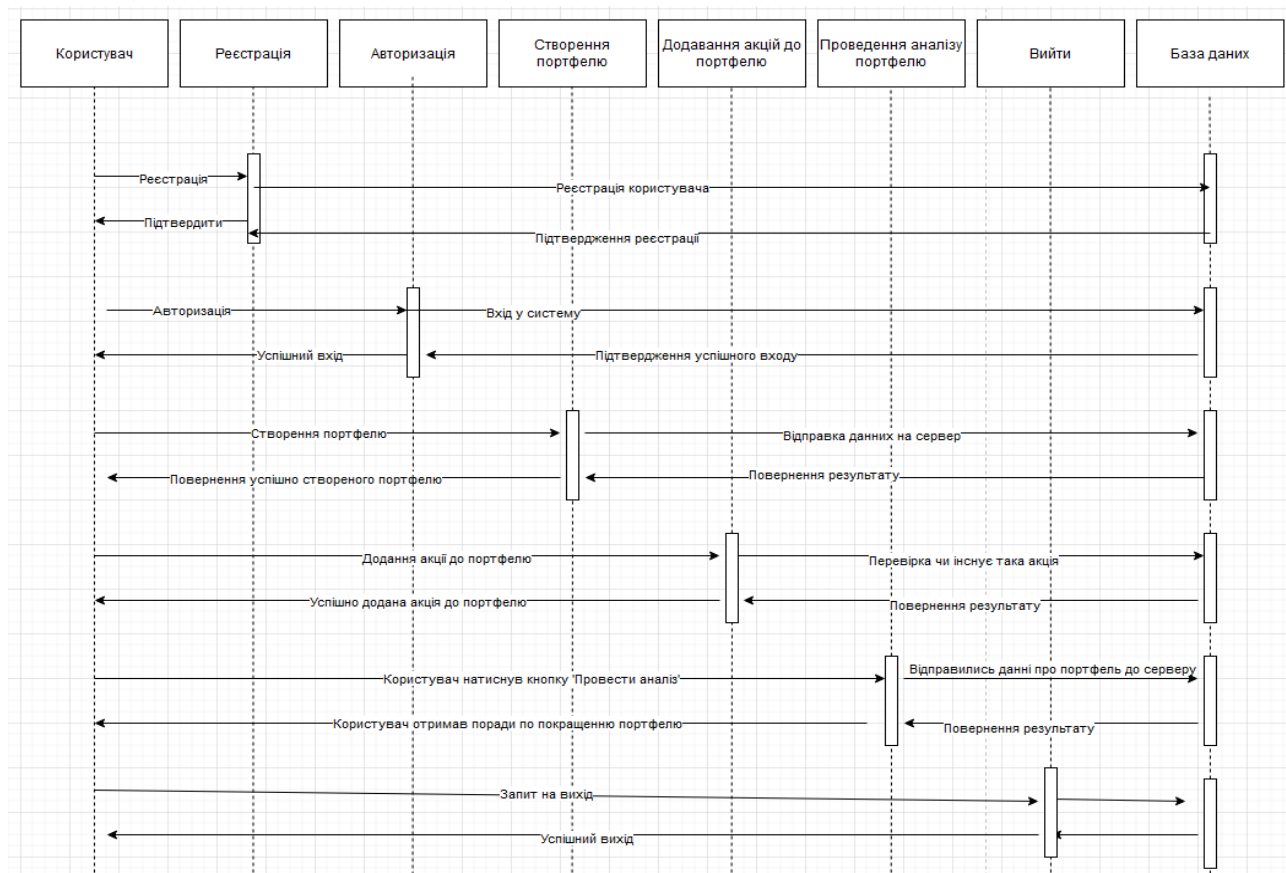


Рис. 2.5. Діаграма роботи сайту

Вона описує основні етапи, починаючи з того, як гість відвідує головну сторінку та завершується тим, що користувач отримує результат виконання своїх операцій на сайті.

1. Гість заходить на головну сторінку.
2. Після входу або реєстрації робиться відповідний запит на сервер.
3. Якщо є помилки у введених даних – відображається повідомлення про помилку.
4. Дані перевіряються на стороні серверу й повертається помилка у разі їх неправильності.

5. Після успішної авторизації або реєстрації, користувач може створювати портфелі.

6. Система виконує запити до БД та повертає дані.

7. Результат виконання операції відображається користувачу.

У результаті чітко визначеної роботи з користувачем система автоматично перевіряє введені дані та обробляє запити на сервері, що дозволяє забезпечити максимальну зручність і безпеку під час авторизації та реєстрації. Це важливо, оскільки гарантується, що лише правильні дані потрапляють до системи, що, в свою чергу, дозволяє підтримувати високу надійність та точність даних.

Після успішної авторизації або реєстрації користувач може приступити до створення інвестиційних портфелів. Важливою частиною цього процесу є взаємодія з базою даних, де зберігаються всі необхідні дані про портфелі та їх складові. База даних не тільки зберігає інформацію, але й дозволяє надавати користувачеві актуальні дані в режимі реального часу.

Подальша обробка даних, аналіз інвестиційного портфеля та візуалізація результатів виконуються за допомогою відповідних алгоритмів, які забезпечують точність і швидкість роботи системи. У кінцевому результаті користувач отримує чітке і зрозуміле представлення про стан свого портфеля, що дозволяє йому приймати обґрунтовані рішення щодо його коригування та подальших дій.

Загалом, система надає користувачам всі необхідні інструменти для створення, моніторингу та аналізу інвестиційних портфелів у зручному та інтуїтивно зрозумілому інтерфейсі, що є важливим для успішного впровадження та використання таких систем у реальних умовах.

Для візуалізації процесів взаємодії компонентів системи можна звернутися до діаграми активності, представленої на рисунку 2.5, яка відображає ключові етапи роботи вебсайту і взаємодії між користувачем, сервером і базою даних.

Висновки до другого розділу

У процесі проєктування системи аналізу інвестиційного портфелю, були аналізовані всі можливі варіанти використання програмного продукту відповідно до вимог, що стосуються програмного забезпечення. Отримані результати були використані для створення об'єктно-орієнтованої структури системи, включаючи моделювання взаємодії між складовими частинами відповідно до обраної архітектури.

Крім того, була розроблена база даних на основі сутностей та даних, які використовуються у системі. Зв'язки між таблицями бази даних та моделями-сутностями були налагоджені за допомогою ORM, що дозволило забезпечити зручний доступ та взаємодію з даними у системі.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБДОДАТКОМ

3.1 Створення програмного додатку

Створення програмного додатку починається з створення контролерів які відповідають за прийом HTTP-запитів і повернення відповідей. Це своєрідна точка входу до додатка для зовнішніх запитів. А також створення сервісів які відповідають за реалізацію бізнес-логіки та виконання операцій, таких як взаємодія з базою даних, виконання обчислень або запитів до зовнішніх API. Сервіси організовують логіку програми в окремі класи які можуть бути багаторазово використані.

Розпочнемо наше створення з GetOneService сервісу який відповідає за отримання коректної інформації для контролера портфелів та має 2 методи які відподні за отримку даних з бази даних та стороннього ресурсу. Розпочнемо з методу getOne який наведено нижче.

```

async getOne({ id, type = 'findOne' }: GetOneDto): Promise<Portfolio> {
  try {
    const portfolio = this.portfolioRepository
      .createQueryBuilder('portfolio')
      .leftJoinAndSelect('portfolio.dermaches', 'dermaches')
      .where('portfolio.id = :id', { id });

    if (type === 'delete' || type === 'edit') {
      portfolio
        .leftJoin('portfolio.user', 'user')
        .select(['portfolio', 'dermaches', 'user.id']);
    }

    const result = await portfolio.getOne();
    if (!result)
  
```

```

    throw new HttpException(
        'Портфоліо з таким id не існує',
        HttpStatus.NOT_FOUND,
    );
    return result
  } catch (error) {
    throw error;
  }
}

```

Даний метод відповідає за запит до бази даних який перевіряє чи є в наявності те чи інше портфоліо. Цей метод приймає об'єкт типу `GetOneDto` який очікує ідентифікатор портфелю (`id`) та за бажанням тип (`type`) він нам необхідний, щоб коли ми перевикористовуємо цей метод в інших сервісах наприклад `CreateService` ми не отримували помилку при перевірці чи існує вже такий портфель.

Наступним методом є `getOneWithData` він відповідає за отримання та обчислення всіх необхідних даних для відмальовки на стороні клієнту.

```

async getOneWithData(portfolio: Portfolio): Promise<IPortfolio> {
  try {
    const startPrice = countStartPrice(portfolio.dermaches)
    const      dermachesWithCurrentPrice      =      await
countCurrentPrice(portfolio.dermaches)
    const      dermaches      =      addInfoToDermaches(portfolio.dermaches,
dermachesWithCurrentPrice)
    const currentPrice = portfolioCurrentPrice(dermaches)
    dermaches.forEach(dermache => {
      dermache.weight = (((dermache.buyCount - dermache.sellCount || 0) *
dermache.currentPrice) / currentPrice) * 100
    })
  }
}

```

```

const profit = currentPrice - startPrice;
const profitPercent = ((currentPrice - startPrice) / startPrice) * 100;
const dermachesHistory = await this.dermacheGetHistory.getHistory({
  tickers: dermaches.map((dermache) => dermache.ticker).join(','),
  createdAt: portfolio.createdAt,
  dermaches: dermaches
})
return {
  description: portfolio.description,
  currentPrice,
  profit,
  profitPercent,
  startPrice,
  dermaches,
  dermachesHistory
};
}
catch (error) {
  throw error
}
}

```

Цей метод має кілька змінних які підраховують необхідні данні, групуються та повертаються в вигляді об'єкта. Детальний огляд змінних наведено нижще.

```
const startPrice = countStartPrice(portfolio.dermaches)
```

Ця зміна вираховує суму яка була витрачена на покупку всіх акцій в ній викликається функція countStartPrice яка проходиться по акціям за допомогою метода reduce, який використовується для обробки масивів і зведення їх до одного значення.

```
const dermachesWithCurrentPrice = await
countCurrentPrice(portfolio.dermaches)
```

Ця зміна отримує список акцій з актуальними цінами з брокера FreedomFinance за допомогою axios. Ця бібліотека є популярним інструментом для виконання HTTP-запитів у JavaScript-додатках. Axios забезпечує зручний синтаксис для роботи з API, підтримує проміси, що спрощує обробку асинхронних операцій, і має вбудовані можливості для перехоплення запитів та відповідей. Таким чином ми можемо створити красивий та легко читабельний синтаксис який буде зрозумілий для інших розробників. Запит для отримання інформації про акції наведено нижче.

```
export const countCurrentPrice = async (dermaches: Dermache[]):  
Promise<IDermacheFromFF[]> => {  
  const tickers = dermaches.map((dermache) => dermache.ticker).join('+')  
  const params = "ltp+c";  
  const { data } = await  
    axios.get(`https://freedom24.com/securities/export?tickers=${tickers}&params=${pa  
rams}`)  
  return data  
}
```

Наступна зміна повертає новий масив акцій з даними, а саме додається теперішня ціна, підраховується профіт акції в доларах та в відсотках.

```
const dermaches = addInfoToDermaches(portfolio.dermaches,  
dermachesWithCurrentPrice)
```

Наступна зміна нам знадобиться щоб підрахувати теперішню вартість портфелю з теперішньої ціни акцій яку ми мажм в dermaches.

```
const currentPrice = portfolioCurrentPrice(dermaches)
```

Підрахунок ваги акцій є ключовим моментом в нашому додатку так як вага акцій потрібна для проведення аналізу портфелю та коректного відображення на круговій діаграмі розподілу акцій. Саме для цього нам потрібний метод перебору масиву який наведено нижще.

```
dermaches.forEach(dermache => {
```

```

    dermache.weight = (((dermache.buyCount - dermache.sellCount || 0) *
dermache.currentPrice) / currentPrice) * 100
  })

```

Щоб інвестор міг легко визначити, скільки відсотків та доларів було зароблено, у системі використовуються спеціальні змінні, які наведені нижче.

```
const profit = currentPrice - startPrice;
```

```
const profitPercent = ((currentPrice - startPrice) / startPrice) * 100;
```

Наступна зміна нам потрібна щоб коректно відмальовувати графік ціни портфелю. В цій зміні робиться запит на стороній ресурс щоб отримати історію свічок всіх акцій які ми маєм в портфелі та визначити середнє значення на кождний день від дня створення портфелю. А також присвоїти кожній акції її волативність що є доволі важливим показником при аналізі портфелю.

```
const dermachesHistory = await this.dermacheGetHistory.getHistory({
```

```

    tickers: dermaches.map((dermache) => dermache.ticker).join(','),
    createdAt: portfolio.createdAt,
    dermaches: dermaches
  })

```

Наступним не менш важливим сервісом є AnalyzeService. Він відповідає за аналіз портфелю та надання рекомендацій для покращення стану портфелю. В цьому сервісі є один метод analyze який наведено нище.

```
analyze(portfolioData: IPortfolio): IAnalyze {
  let score = 0;
  const issues: string[] = [];
  const totalRoi = totalROI(portfolioData.dermaches)
  const roiScore = totalRoi > 20 ? 40 : totalRoi > 10 ? 20 : 0;
  if (roiScore === 0) issues.push("Низька прибутковість портфеля (ROI <
10%).\nДекілька порад які можуть покращити стан:\nОцініть поточний склад
портфеля: Перегляньте активи, які мають найменший вклад у
прибутковість.\nВизначте слабкі місця: Це можуть бути активи з низькою
віддачею або високою волатильністю, яка не компенсується
```

прибутковістю.\nПерегляньте розподіл ваги активів: Один актив може займати занадто велику частку, що впливає на загальний результат.");

```
score += roiScore;
```

```
const avgVolatility = averageVolatility(portfolioData.dermaches);
```

```
const volatilityScore = avgVolatility < 10 ? 30 : avgVolatility < 20 ? 15 : 0;
```

if (volatilityScore === 0) issues.push("Високий ризик через волатильність (> 20%).\nДекілька порад які можуть покращити стан:\nЗменшіть частку активів із високою волатильністю, якщо їхній внесок у загальний дохід невеликий\nДодайте інструменти хеджування, такі як опіони, ф'ючерси чи золото, які можуть захистити ваш портфель від значних коливань.\nЗбалансуйте активи, наприклад, 60% портфеля можуть складати надійні активи, а 40% — ризикові");

```
score += volatilityScore;
```

```
const isDiversified = portfolioData.dermaches.every(asset => asset.weight < 50);
```

```
const diversificationScore = isDiversified ? 20 : 0;
```

if (diversificationScore === 0) issues.push("Недостатня диверсифікація активів (один або більше активів $\geq 50\%$ ваги).\nЩо слід зробити:\n Проведіть ребалансування: перевірте співвідношення активів і відкорегуйте його.Максимальна частка одного активу у портфелі не повинна перевищувати 10–20%.\nДодайте активи з різних секторів (технології, медицина, енергетика, сільське господарство тощо).\nВикористовуйте біржові фонди(ETF), вони дозволяють інвестувати у широкий спектр активів з низькими витратами. Це знижує концентрацію в окремих позиціях.");

```
score += diversificationScore;
```

```
const overweightAsset = portfolioData.dermaches.some(asset => asset.weight > 60);
```

```
const balanceScore = overweightAsset ? -20 : 10;
```

if (balanceScore < 0) issues.push("Один із активів займає понад 60% ваги портфеля.\nПоради які допоможуть покращити стан:\nПроаналізуйте активи,

які мають найбільшу вагу у портфелі.\nЗменште частку активу(-ів), що перевищують 50%, за рахунок продажу частини цього активу.");

```

    score += balanceScore;
    score = Math.min(100, Math.max(0, score));
    return {
      score,
      details: {
        roiScore,
        volatilityScore,
        diversificationScore,
        balanceScore,
      },
      issues,
    };
  }

```

В даному методі аналіз проводиться по чотирьом ключовим показникам а саме ROI, волативність, дирефікація, збалансованість. Давайте розпочнемо з ROI. ROI – це показник, який використовується для оцінки ефективності інвестицій або порівняння прибутковості різних варіантів вкладень. Він відображає співвідношення отриманого прибутку до суми інвестицій і зазвичай виражається у відсотках. Формула розрахунку ROI виглядає так, (3.1):

$$ROI = \frac{\text{Чистий прибуток}}{\text{Сума інвестицій}} * 100 \quad (3.1)$$

Цей показник допомагає інвесторам і підприємцям зрозуміти, наскільки вигідними є їхні інвестиції. Високий ROI свідчить про те, що вкладення було ефективним і принесло значний прибуток у порівнянні з витратами. Фрагмент коду який відповідає за підрахунок ROI наведено нище.

```

export const totalROI = (dermaches: IDermacheWithCurrentPrice[]) => {
  const result = dermaches.reduce((sum, dermache) => {

```

```

    const roi = ((dermache.currentPrice - dermache.buyPrice) /
dermache.buyPrice) * 100;
    return sum + roi * (dermache.weight / 100);
  }, 0);
  return result
}

```

Наступним підраховується волативність. Волатильність — це показник, що характеризує ступінь коливань цін фінансових інструментів за певний період часу. Вона відображає, наскільки швидко і суттєво змінюється ціна активу, і є одним із ключових параметрів оцінки ризику на фінансових ринках. Висока волатильність вказує на значні зміни цін у короткі періоди часу, тоді як низька волатильність свідчить про стабільніші та передбачувані зміни. Волатильність використовується для аналізу ринків і прийняття рішень інвесторами та трейдерами. Вона допомагає оцінити ризики, пов'язані з інвестиціями у певний актив: чим вища волатильність, тим вищий ризик, але потенційно й більший прибуток. Наприклад, активи з високою волатильністю можуть бути привабливими для трейдерів, які прагнуть заробити на короткострокових коливаннях цін. Натомість інвестори, які віддають перевагу стабільним довгостроковим вкладенням, зазвичай обирають активи з низькою волатильністю. Фрагмент коду який відповідає за підрахунок волативності наведено нище.

```

export const averageVolatility = (dermaches: IDermacheWithCurrentPrice[])
=> {
  return dermaches.reduce((sum, asset) => {
    return sum + asset.volatility * (asset.weight / 100);
  }, 0);
}

```

Не менш важливим для інвесторів є і диверсифікація яка потрібна для зменшення впливу волатильності ринку та мінімізації ризиків, пов'язаних із вкладенням в один актив або сектор. Наприклад, інвестор може вкладати кошти

у акції різних галузей, облігації, нерухомість, золото чи інші активи. Це забезпечує більшу стабільність портфеля, оскільки падіння вартості одного активу не матиме критичного впливу на загальний результат. Фрагмент коду який відповідає за підрахунок волативності наведено нище.

```
const isDiversified = portfolioData.dermaches.every(asset => asset.weight < 50);
```

Останім етапом аналізу портфелю є перевірка його на збалансованість. Зазвичай люди плутають поняття збалансованості та диверсифікації, хоча вони мають різні цілі та підходи у фінансовому плануванні. Диверсифікація, як уже згадувалося, – це розподіл інвестицій між різними активами, секторами або регіонами, щоб знизити ризик втрат у разі падіння окремого активу чи сектора. Її головна мета – мінімізувати ризик за рахунок різноманітності портфеля. Збалансованість же стосується гармонійного розподілу активів у портфелі відповідно до цілей інвестора, його фінансових можливостей і схильності до ризику. Це означає створення портфеля, у якому ризик і дохідність відповідають очікуванням інвестора. Збалансований портфель зазвичай включає як ризиковані активи (наприклад, акції), так і консервативні (наприклад, облігації чи грошові фонди), щоб забезпечити оптимальне співвідношення між стабільністю та потенційним прибутком. Фрагмент коду який відповідає за підрахунок волативності наведено нище.

```
const overweightAsset = portfolioData.dermaches.some(asset => asset.weight > 60).
```

3.2 Структура інтерфейсу вебдодатку

Взаємодія користувачів з системою відбувається за допомогою браузера. При переході на сторінку сайту, користувач бачить перед собою початкову сторінку (Рис. 3.1).

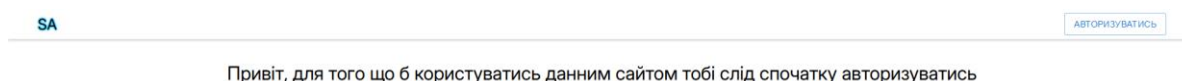


Рис. 3.1. Початкова сторінка

Для отримання додатково повноцінного функціоналу, користувач повинен авторизуватись. Якщо відвідувач не має облікового запису, необхідно пройти етап реєстрації на вебресурсі (рисунки. 3.2 - 3.6). Форма реєстрації та авторизації передбачає введення електронної пошти, паролю. Встановлені наступні вимоги до паролів: мінімальна кількість символів 8, та перевірка електронної пошти за допомогою регулярного вираження `'/^[\w-\.]++@([\w-]+\.)+[\w-]{2,4}$/'`. Для форми реєстрації також зрівнюються паролі.

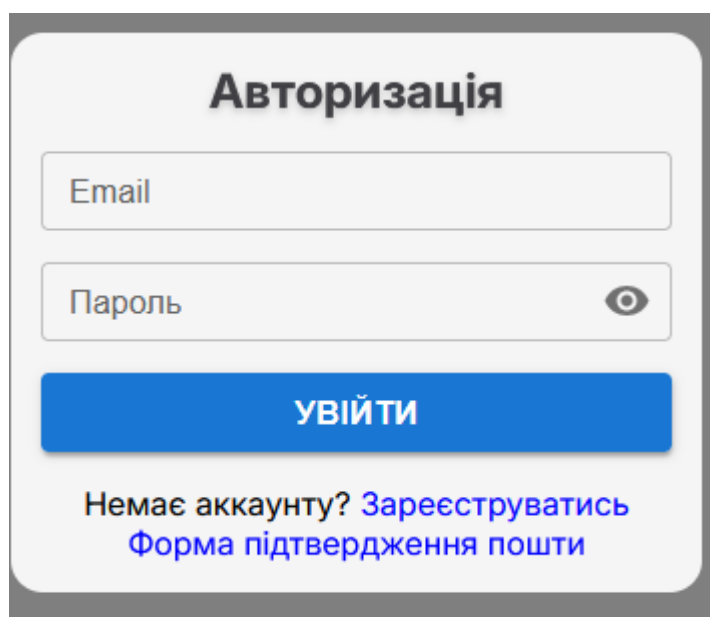


Рис. 3.2. Форма авторизації

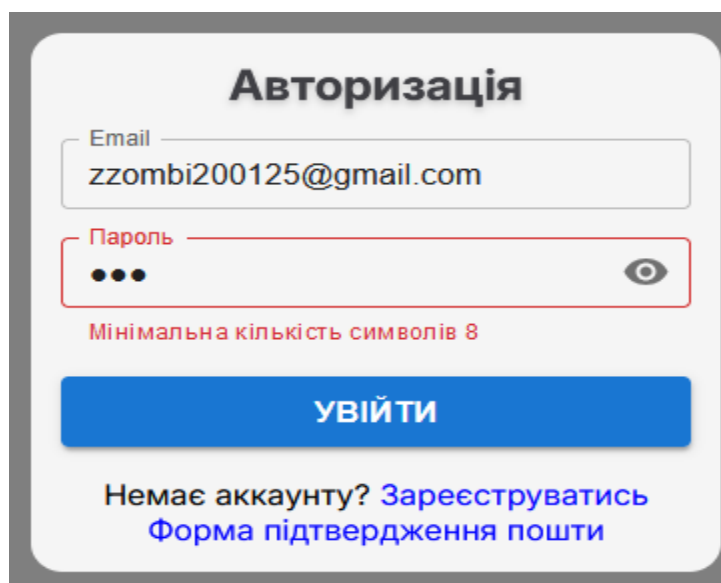
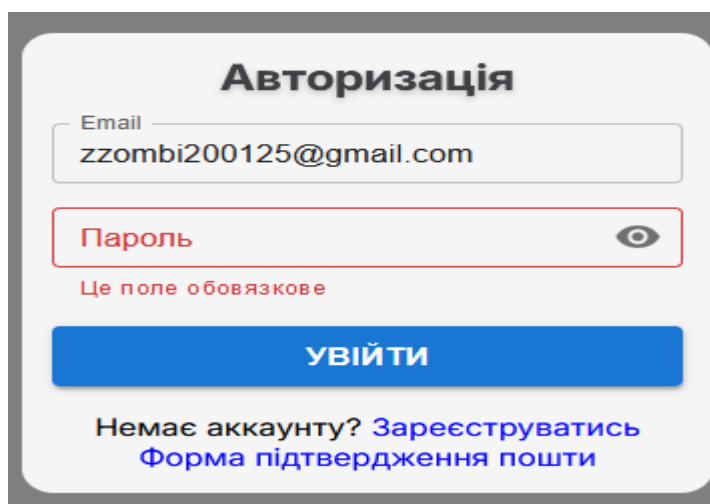


Рис. 3.3. Форма авторизації з занадто коротким паролем



Авторизація

Email
zzombi200125@gmail.com

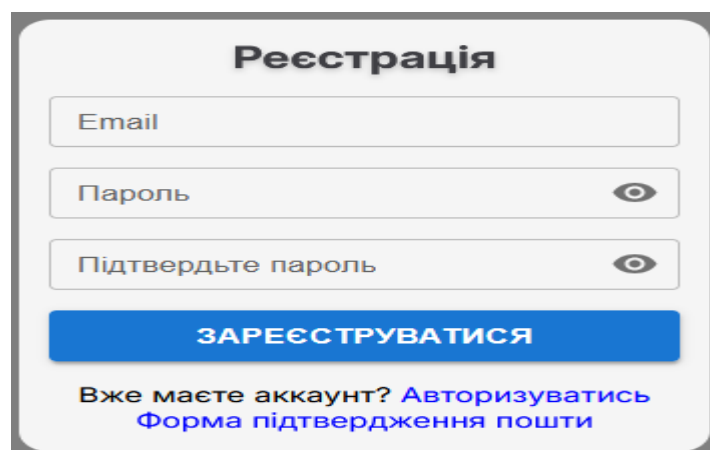
Пароль 

Це поле обов'язкове

УВІЙТИ

Немає аккаунту? [Зареєструватись](#)
[Форма підтвердження пошти](#)

Рис. 3.4. Форма авторизації з пустим полем пароллю



Реєстрація

Email

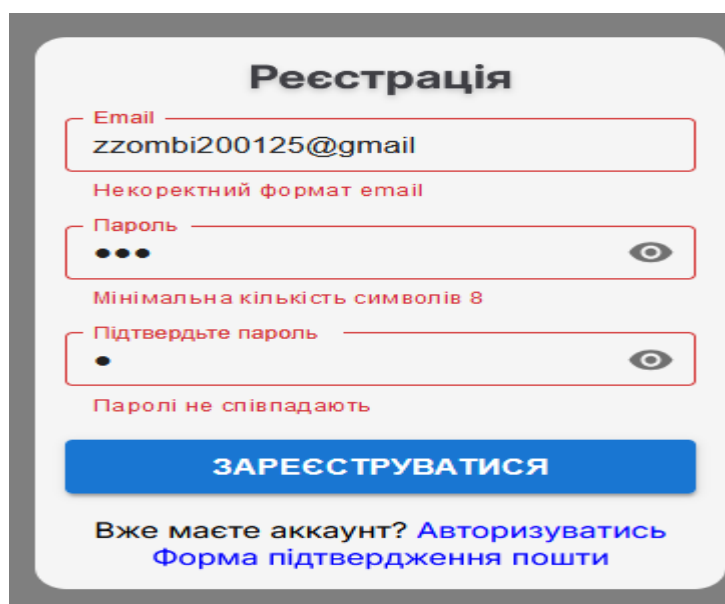
Пароль 

Підтвердьте пароль 

ЗАРЕЄСТРУВАТИСЯ

Вже маєте аккаунт? [Авторизуватись](#)
[Форма підтвердження пошти](#)

Рис. 3.5. Форма реєстрації



Реєстрація

Email
zzombi200125@gmail
Некоректний формат email

Пароль 
...
Мінімальна кількість символів 8

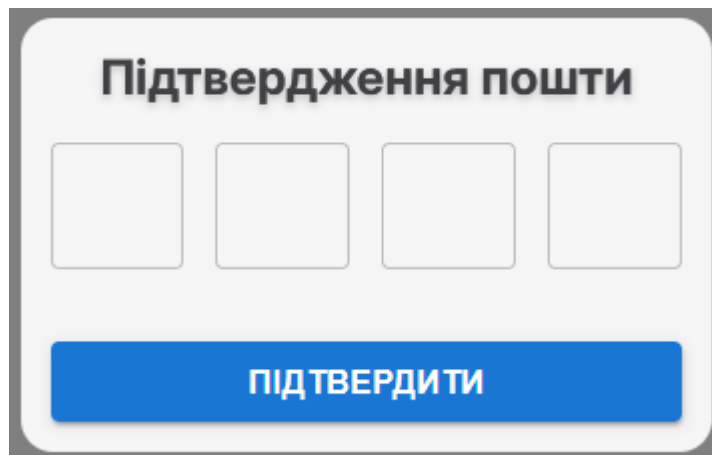
Підтвердьте пароль 
.
Паролі не співпадають

ЗАРЕЄСТРУВАТИСЯ

Вже маєте аккаунт? [Авторизуватись](#)
[Форма підтвердження пошти](#)

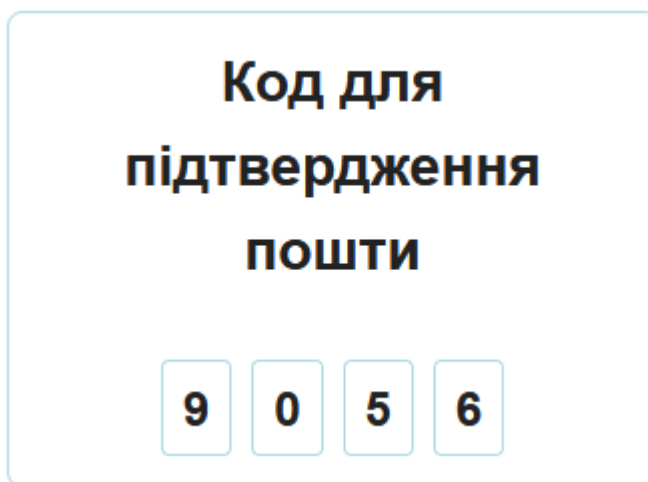
Рис. 3.6. Форма з неправильно заповненими полями

Наступним етапом після реєстрації є підтвердження аккаунту через форму вводу коду (рисунки 3.7 – 3.8). Форма підтвердження електронної пошти, передбачає введення чотирьох значного числа.



The image shows a web form titled "Підтвердження пошти" (Email Verification). It contains four empty square input boxes arranged horizontally. Below these boxes is a blue rectangular button with the text "ПІДТВЕРДИТИ" (CONFIRM) in white capital letters.

Рис. 3.7. Форма підтвердження пошти



The image shows a web form titled "Код для підтвердження пошти" (Code for email verification). Below the title, four square boxes display the digits "9", "0", "5", and "6" in sequence.

Рис. 3.8. Код який приходить на пошту

Якщо ж користувач випадково закрити форму підтвердження пошти він може відчинити її заново натиснувши “Форма підтвердження пошти” та ввести свою пошту куди знову прийде лист (рисунки 3.9). Форма відправки коду на електронну пошту, передбачає введення електронної пошти яка перевіряється за допомогою регулярного вираження `'/^[\w-\.]++@([\w-]+\.)+[\w-]{2,4}$/'`.

Підтвердження пошти

Email

ВІДПРАВИТИ КОД НА ПОШТУ

Рис. 3.9. Форма відправки листа на пошту

Також на сайті є сповіщення про успішне виконання операції (рис. 3.10 – 3.11).

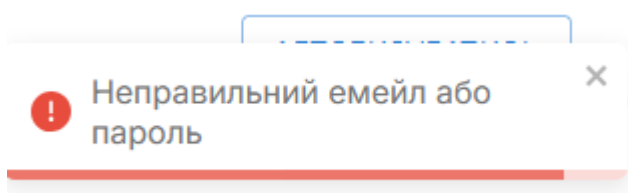


Рис. 3.10. Запит виконався з помилкою

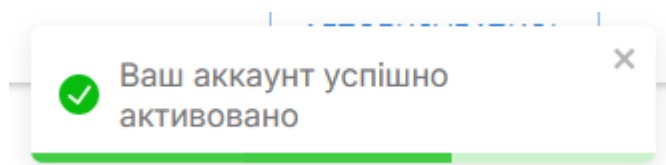


Рис. 3.11. Запит виконався успішно

Після успішної авторизації користувач побачить перед собою головну сторінку (Рис. 3.12).

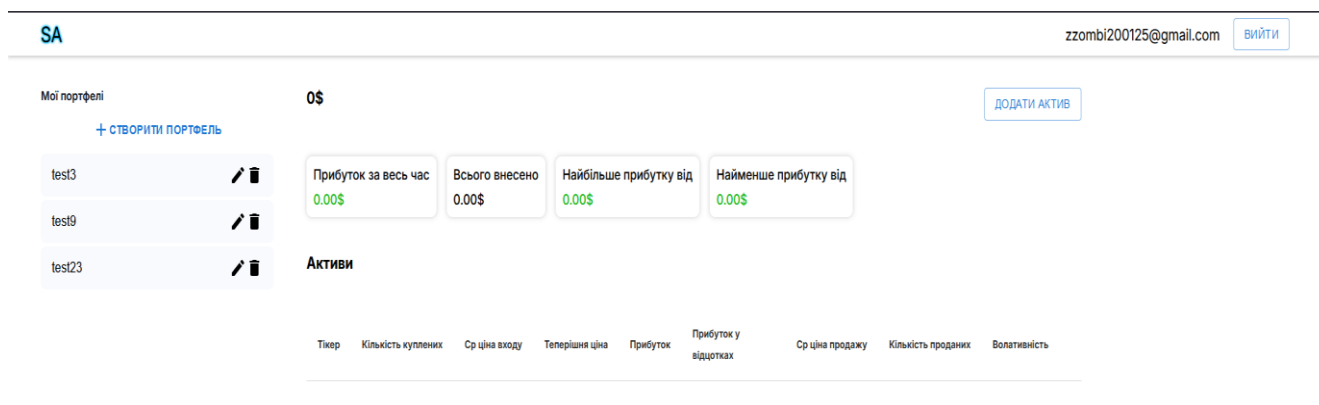


Рис. 3.12. Головна сторінка авторизованого користувача

На головній сторінці користувач може створити собі інвестиційний портфель натиснувши на кнопку “Створити портфель” (рисунки 3.13 – 3.14). Та заповнивши відповідну форму з двома полями які мають тип текст.

Мої портфелі

+ СТВОРИТИ ПОРТФЕЛЬ

Рис. 3.13. Кнопка “створити портфель”

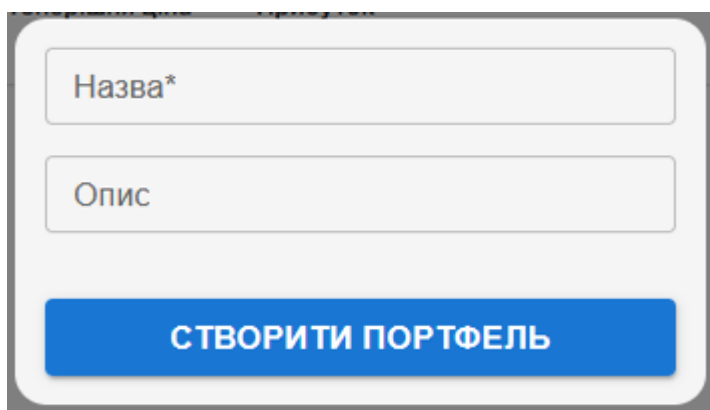
A modal form for creating a portfolio. It contains two text input fields: the first is labeled "Назва*" (Name*) and the second is labeled "Опис" (Description). Below the fields is a blue button with the text "СТВОРИТИ ПОРТФЕЛЬ" (CREATE PORTFOLIO).

Рис. 3.14. Модальне вікно для створення портфелю

Після успішного створення портфелю він з’явиться у нас в списку з можливістю редагування або видалення (рисунки 3.15-3.17).

Мої портфелі

+ СТВОРИТИ ПОРТФЕЛЬ

test1



test2



Рис. 3.15. Список портфелів

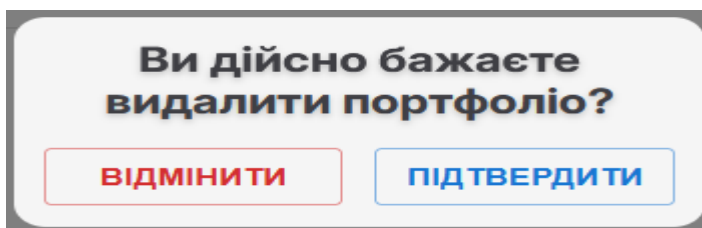
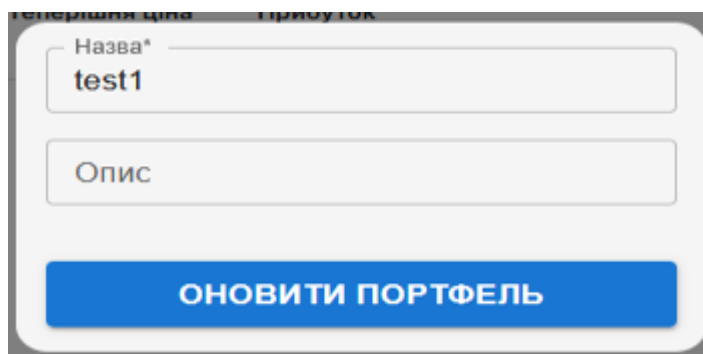
A confirmation modal with the text "Ви дійсно бажаєте видалити портфоліо?" (Do you really want to delete the portfolio?). Below the text are two buttons: "ВІДМІНИТИ" (Cancel) in red and "ПІДТВЕРДИТИ" (Confirm) in blue.

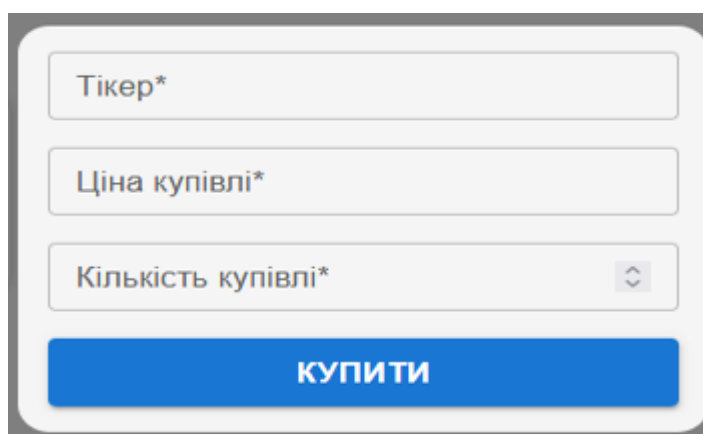
Рис. 3.16. Модальне вікно підтвердження видалення портфелю



Modal window for editing a portfolio. It contains two input fields: 'Назва*' (Name) with the value 'test1' and 'Опис' (Description). Below the fields is a blue button labeled 'Оновити портфель' (Update portfolio).

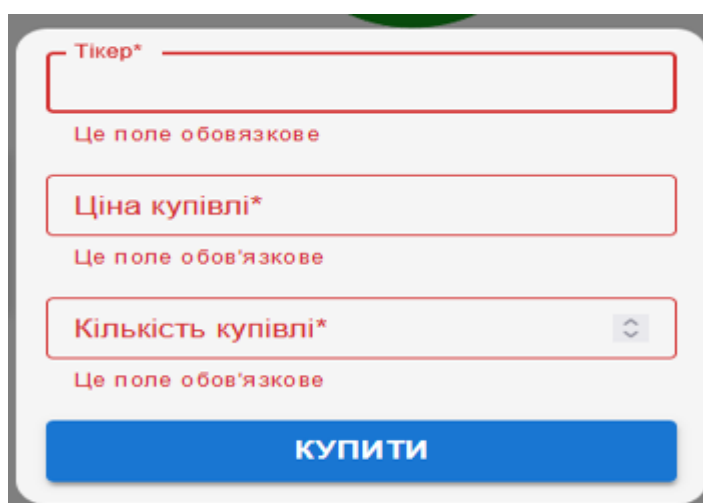
Рис. 3.17. Модальне вікно редагування портфелю

Після створення портфелю можна перейти до заповнення його акціями (рисунки 3.18 – 3.19). Форма купівлі акцій передбачає введення назви акції (тікер) тип текст, ціни купівлі в \$ та кількості куплених акцій типу числа.



Modal window for buying shares. It contains three input fields: 'Тікер*' (Ticker), 'Ціна купівлі*' (Purchase price), and 'Кількість купівлі*' (Purchase quantity). Below the fields is a blue button labeled 'КУПИТИ' (BUY).

Рис. 3.18. Модальне вікно купівлі акції



Modal window for buying shares with error messages. The input fields 'Тікер*', 'Ціна купівлі*', and 'Кількість купівлі*' are highlighted with red borders. Below each field is a red error message: 'Це поле обов'язкове' (This field is required). The blue 'КУПИТИ' button is still visible at the bottom.

Рис. 3.19. Модальне вікно купівлі акції з помилками

Після успішного заповнення портфелю акціями на головній сторінці додаються декілька нових елементів (рисунки 3.20-3.22).



Рис. 3.20. Графік середньої ціни портфелю

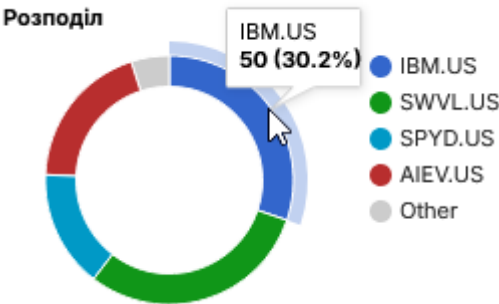


Рис. 3.21. Кругова діаграма розподілу активів

Активи									
Тікер	Кількість куплених	Ср ціна входу	Теперішня ціна	Прибуток	Прибуток у відсотках	Ср ціна продажу	Кількість проданих	Волативність	
IBM.US	10	50.00\$	227\$	1770\$	454%			4.96	<button>ПРОДАТИ</button>
PLTR.US	10	0.50\$	66.13\$	656.3\$	13226%			3.19	<button>ПРОДАТИ</button>
KBWY.US	10	0.50\$	20.08\$	195.8\$	4016%			0.32	<button>ПРОДАТИ</button>
SWVL.US	10	50.00\$	6\$	100\$	20%	60\$	10	0.62	<button>ПРОДАТИ</button>
PEP.US	3	2.00\$	162.39\$	481.17\$	8119.5%			2.73	<button>ПРОДАТИ</button>
SPYD.US	20	25.25\$	47.32\$	-405\$	-60.4%	10\$	10	0.48	<button>ПРОДАТИ</button>
CTRN.US	6	4.33\$	20.02\$	94.12\$	462%			1.43	<button>ПРОДАТИ</button>

Рис. 3.22. Оновлений вигляд таблиці активів

Якщо користувач випадково купив не ту акції він може її видалити натиснувши на значок корзини (Рис. 3.23).

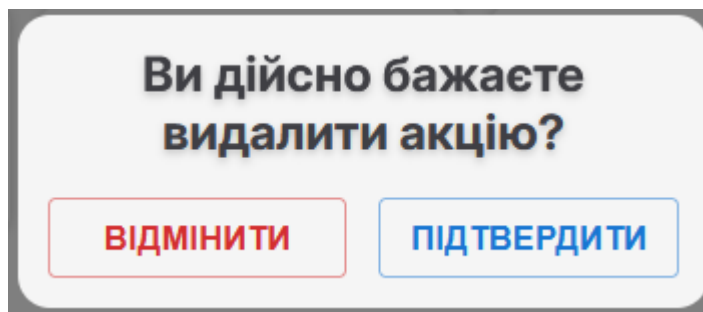


Рис. 3.23. Модальне вікно видалення акції з портфелю

Щоб записати продаж акції та зупинити підрахунок прибутку потрібно натиснути на кнопку "Продати" після чого з'явиться відповідне модальне вікно (рисунки 3.24-3.26) з двома полями в які можна ввести лише числа.

A modal window for selling a stock. It features two input fields with red borders and red text. The first field is labeled "Ціна продажу*" and the second "Кількість продажу*", both with a red asterisk indicating they are required. Below each field is a red error message: "Це поле обов'язкове". At the bottom is a large blue button with the text "ПРОДАТИ".

Рис. 3.24. Модальне вікно з пропущеними заповненими полями

A modal window for selling a stock, similar to the previous one, but with invalid input values. The "Ціна продажу*" field contains "-1" and the "Кількість продажу*" field also contains "-1". Below each field is a red error message: "Мінімальна ціна продажу повина бути > 0" and "Мінімальна кількість повина бути > 0". At the bottom is a large blue button with the text "ПРОДАТИ".

Рис. 3.25. Модальне вікно з неправильно заповненими полями

Рис. 3.26. Правильно заповнене модальне вікно

Коли користувач заповнить своє портфоліо та натисне “Аналіз портфелю” йому відкриється відповідне модальне вікно з детальною інформацією (рисунки 3.27-3.31).

Рис. 3.27. Вигляд модального вікна без помилок

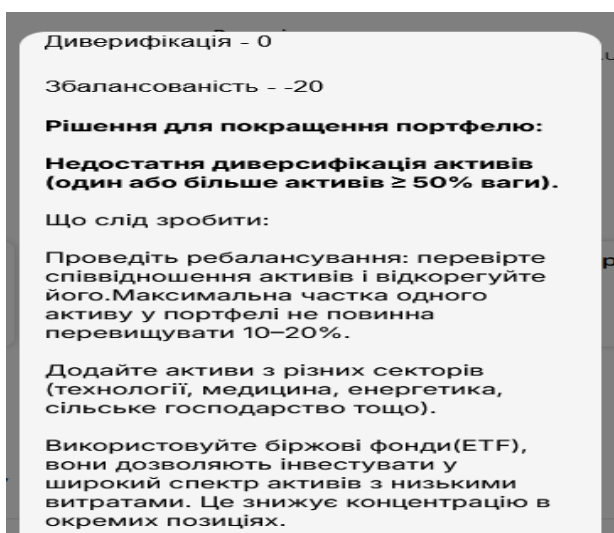


Рис. 3.28. Вигляд модального вікна з помилкою про диверсифікацію активів

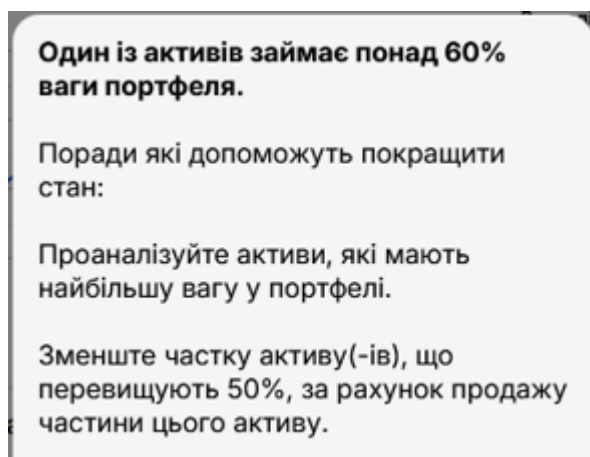


Рис. 3.29. Вигляд модального вікна з помилкою про недостатню збалансованість активів



Рис. 3.30. Вигляд модального вікна з помилкою про низьку прибутковість



Рис. 3.31. Вигляд модального вікна з помилкою про високу волатильність

На наступному етапі проведемо тестування програмного комплексу.

3.3 Тестування роботи програмного комплексу

Важливим етапом розробки програмного забезпечення є процес тестування та виправлення дефектів. Для перевірки коректності роботи сайту було проведено тестування основних функцій, таких як створення портфеля, заповнення його акціями та запит аналізу. Ось кроки тестування:

Першим кроком є вхід в обліковий запис. Передумовою є те, що користувач знаходиться на формі авторизації. Кроки включають введення облікових даних та натискання кнопки "Увійти". Очікуваний результат: користувач буде перенаправлений на головну сторінку. Фактичний результат підтвердив, що користувач справді був перенаправлений на головну сторінку.

Другим етапом є створення інвестиційного портфелю. Передумови: користувач аутентифікований у системі. Кроки включають натискання кнопки “Створити портфель” та заповнення форми. Очікуваний результат: користувач повинен побачити портфель у списку портфелів. Фактичний результат підтвердив, що портфель дійсно з’явився в списку портфелів.

Третій крок включає додавання акції до портфелю. Передумови: користувач знаходиться на сторінці необхідного портфелю. Кроки: коректне заповнення форми й відправка її на сервер. Очікуваний результат: користувач має побачити сповіщення про успішне додавання акції до портфелю, а також саму акцію в портфелі. Фактичний результат підтвердив, що сповіщення було відображено, а акція з’явилася в портфелі.

Четвертий етап – видалення акції з портфелю. Передумови: користувач знаходиться на сторінці необхідного портфелю. Кроки: користувач знаходить необхідну акцію, натискає на іконку корзини та підтверджує видалення акції. Очікуваний результат: акція має зникнути зі списку, а також висвітитись сповіщення про успішне видалення. Фактичний результат підтвердив, що акція дійсно зникла, а сповіщення було відображене.

П’ятий крок – продаж акції в портфелі. Передумови: користувач знаходиться на сторінці необхідного портфелю. Кроки: користувач знаходить акцію, натискає на кнопку “Продати” та коректно заповнює форму, після чого знову натискає на кнопку “Продати”. Очікуваний результат: з’являється нотифікація про успішну транзакцію. Фактичний результат підтвердив, що нотифікація з’явилася.

Шостим етапом є редагування інформації про портфель. Передумови: користувач знаходиться на головній сторінці або на сторінці портфеля. Кроки: користувач натискає на іконку олівця на необхідному портфелі та змінює його назву або опис, після чого натискає кнопку “Оновити портфель”. Очікуваний результат: повинна з’явитися нотифікація про успіх. Фактичний результат підтвердив, що нотифікація з’явилася.

Останній етап – аналіз портфелю. Передумови: користувач знаходиться на сторінці портфеля, який потребує аналізу, та має мінімум одну акцію в ньому. Кроки: користувач натискає кнопку “Аналіз портфелю”. Очікуваний результат: відкривається модальне вікно з оцінкою портфелю та порадами щодо його покращення, якщо в портфелі є проблеми. Фактичний результат підтвердив, що модальне вікно з оцінкою та порадами відкривається.

Ці тести дозволили переконатися в тому, що всі основні функціональні можливості системи працюють правильно, і користувач може успішно взаємодіяти з платформою, створюючи та редагуючи свої інвестиційні портфелі, додаючи акції та проводячи аналіз.

Висновки до третього розділу

У даному розділі наведено опис структури для системи аналізу портфелю. Описано кожен елемент системи та його взаємодію з іншими компонентами.

Також був представлений опис інтерфейсу розробленого програмного продукту. У цьому описі були візуально продемонстровані всі функціональні можливості додатку.

Для перевірки коректності роботи додатку було проведене тестування на тестовому прикладі. Тестування було проведено поетапно з метою порівняння очікуваних результатів з фактичними результатами кожного етапу.

ВИСНОВКИ

В результаті проведених досліджень були отримані наступні висновки: проведено аналіз предметної області, що дозволило визначити основні вимоги до системи. На основі цього аналізу було складено набір функціональних можливостей для веб-орієнтованої системи. З метою розробки конкурентоспроможного програмного забезпечення було проведено порівняльну характеристику платформ, які найбільш наближені до завдань дослідження. Це дало змогу сформувати набір найважливіших характеристик для веб-орієнтованої системи. Відповідно до обраної архітектури та поставлених вимог були вибрані необхідні інструментальні засоби для реалізації проєкту. Були визначені засоби використання платформи користувачами, також була спроектована об'єктно-орієнтована схема моделі, детально описано кожен її структурний елемент.

На основі визначених сутностей системи була спроектована база даних, яка використовує ORM для збереження даних вебдодатку та виконання операцій над ними. Описано алгоритм взаємодії компонентів системи та логіку виконання операцій з даними.

Проведено опис інтерфейсу додатку, його структурних елементів та функціональних можливостей, інструментів роботи з даними. Також було проведено тестування додатку з метою перевірки коректності його роботи.

Реалізований програмний комплекс готовий до використання. Сайт може спростити процес проведення аналізу інвестиційного портфелю та використовуватись для відслідковування інвестицій. Цей програмний продукт не є кінцевою версією і може бути поліпшений шляхом впровадження нового функціоналу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Average Gain. [Електронний ресурс] – Режим доступу: https://admainnew.morningstar.com/directhelp/Glossary/Custom_Statistics/Average_Gain.htm
2. Класичні показники ризику для інвестиційних портфелів. [Електронний ресурс] – Режим доступу: https://stud.com.ua/96925/investuvannya/klasichni_pokazniki_riziku_investitsiynih_portfeliv#google_vignette
3. Співвідношення Трейнора - що це таке, визначення та поняття. [Електронний ресурс] – Режим доступу: https://uk.economy-pedia.com/11032493-treynor-ratio#goog_rewarded
4. Основа фундаментального аналізу. [Електронний ресурс] – Режим доступу: https://moodle.znu.edu.ua/pluginfile.php/55953/mod_resource/content/1/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F%201.pdf
5. Математична та економічна сутність моделі Гордона. [Електронний ресурс] – Режим доступу: <http://www.afo.com.ua/uk/news/2/1413>
6. How to Perform a SWOT Analysis. [Електронний ресурс] – Режим доступу: <https://www.investopedia.com/terms/s/swot.asp>
7. Scenario Analysis. [Електронний ресурс] – Режим доступу: <https://corporatefinanceinstitute.com/resources/financial-modeling/scenario-analysis/>
8. Understanding Value at Risk (VaR) and How It's Computed? [Електронний ресурс] – Режим доступу: <https://www.investopedia.com/terms/v/var.asp>.
9. What is The Monte Carlo Simulation? [Електронний ресурс] – Режим доступу: <https://aws.amazon.com/what-is/monte-carlo-simulation/>

10. Модель Марковіца як основа побудови алгоритму формування ефективного інвестиційного портфеля [Електронний ресурс] – Режим доступу: <https://acadrev.duan.edu.ua/images/PDF/2010/2/8.pdf>
11. What is ROI? How to calculate return on investment. [Електронний ресурс] – Режим доступу: <https://www.techtarget.com/searchcio/definition/ROI>
12. What is Volatility? [Електронний ресурс] – Режим доступу: <https://www.businessinsider.com/personal-finance/investing/what-is-volatility>
13. Beta Coefficient. [Електронний ресурс] – Режим доступу: <https://corporatefinanceinstitute.com/resources/data-science/beta-coefficient/>
14. Why diversification matters. [Електронний ресурс] – Режим доступу: <https://www.fidelity.com/learning-center/investment-products/mutual-funds/diversification>
15. Introduction to web APIs. [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction
16. JavaScript SOLID Principles: How to Write Maintainable Code. [Електронний ресурс] – Режим доступу: <https://www.syncfusion.com/blogs/post/solid-principles-in-javascript>
17. What is Hypertext Transfer Protocol (HTTP)? [Електронний ресурс] – Режим доступу: <https://www.f5.com/glossary/hypertext-transfer-protocol-http>
18. DTO explained in NestJS. [Електронний ресурс] – Режим доступу: <https://medium.com/@yelinliu/dto-explained-in-nestjs-3a296498d77b>
19. What is monolithic architecture in software? [Електронний ресурс] – Режим доступу: <https://www.techtarget.com/whatis/definition/monolithic-architecture>
20. What is Microservices Architecture? [Електронний ресурс] – Режим доступу: <https://cloud.google.com/learn/what-is-microservices-architecture>

21. An Introduction to Serverless Architecture. [Электронный ресурс] – Режим доступа: <https://www.digitalocean.com/resources/articles/serverless-architecture>
22. NestJS and Project Structure - What to Do? [Электронный ресурс] – Режим доступа: <https://dev.to/smolinari/nestjs-and-project-structure-what-to-do-1223>
23. Zustand, When, how and why. [Электронный ресурс] – Режим доступа: <https://dev.to/ricardogesteves/zustand-when-how-and-why-1kpi>
24. NestJS: The Perfect Javascript Backend Framework for Structure, Clean Code, and Modularity. [Электронный ресурс] – Режим доступа: <https://medium.com/@ajonesb/nestjs-the-perfect-javascript-backend-framework-for-structure-clean-code-and-modularity-ae1b3a6e1418>

ДОДАТКИ

Контролер portfolio:

```
import {
  Body,
  Controller,
  Delete,
  Get,
  Param,
  ParseIntPipe,
  Post,
  Put,
  Req,
  UseGuards,
} from '@nestjs/common';
import { CreatePortfolioDto } from './create/create.dto';
import { CreateService } from './create/create.service';
import { AuthGuard } from '../auth/auth.guard';
import { GetAllService } from './get-all/get-all.service';
import { GetOneService } from './get-one/get-one.service';
import { RemoveService } from './remove/remove.service';
import { EditService } from './edit/edit.service';
import { EditPortfolioDto } from './edit/edit.dto';
import { IPortfolio } from './get-one/get-one.interface';
import { AnalyzeService } from './analyze/analyze.service';

@UseGuards(AuthGuard)
@Controller('portfolio')
export class PortfolioController {
  constructor(
    private createService: CreateService,
    private getAllService: GetAllService,
    private getOneService: GetOneService,
    private deleteService: RemoveService,
    private editService: EditService,
    private analyzeService: AnalyzeService
  ) { }

  @Post('create')
  create(@Body() data: CreatePortfolioDto, @Req() req) {
    const userId = req.user.id;
    return this.createService.create({ ...data, user: userId });
  }
}
```

```

@Get("")
getAll(@Req() req) {
  const userId = req.user.id;
  return this.getAllService.getAll(userId);
}

```

```

@Get('/:id')
async getOne(@Param('id', ParseIntPipe) id: number) {
  const portfolio = await this.getOneService.getOne({ id: id })
  if (portfolio.dermaches.length < 1) return portfolio
  return this.getOneService.getOneWithData(portfolio);
}

```

```

@Delete('/:id')
delete(@Param('id', ParseIntPipe) id: number, @Req() req) {
  const userId = req.user.id;
  return this.deleteService.delete(id, userId);
}

```

```

@Put('/:id')
edit(
  @Param('id', ParseIntPipe) id: number,
  @Body() data: EditPortfolioDto,
  @Req() req,
) {
  const userId = req.user.id;
  return this.editService.edit(id, data, userId);
}

```

```

@Post('/analyze')
analyze(
  @Body() data: IPortfolio
) {
  return this.analyzeService.analyze(data)
}
}

```

Cepbic register:

```

import { HttpException, HttpStatus, Injectable } from '@nestjs/common';
import { JwtService } from '@nestjs/jwt';
import * as bcrypt from 'bcryptjs';

```

```

import { CreateService } from 'src/modules/user/create/create.service';
import { GetOneService } from 'src/modules/user/get-one/get-one.service';
import { UserBaseDto } from 'src/modules/user/user.dto';

```

```

import { generateToken } from '../auth.utils';
import { TokenResponse } from '../interfaces/token.interface';
import { EditService } from 'src/modules/user/edit/edit.service';

@Injectable()
export class RegisterService {
  constructor(
    private createUserService: CreateService,
    private editUserService: EditService,
    private userGetOneService: GetOneService,
    private jwtService: JwtService,
  ) {}

  async register(data: UserBaseDto): Promise<TokenResponse> {
    try {
      const candidate = await this.userGetOneService.findOne(
        data.email,
        'create',
      );
      if (candidate)
        throw new HttpException(
          'Користувач з таким емейлом вже існує',
          HttpStatus.BAD_REQUEST,
        );
      const hashPassword = await bcrypt.hash(data.password, 5);
      const user = await this.createUserService.create({
        ...data,
        password: hashPassword,
      });
      const tokens = generateToken(user, this.jwtService);

      await this.editUserService.edit({ ...user, ...tokens });

      delete tokens.expiryTokenDate;

      return tokens;
    } catch (error) {
      throw error;
    }
  }
}

```

Сервіс відправки повідомлень на пошту:

```

import { Injectable } from '@nestjs/common';
import * as nodemailer from 'nodemailer';

@Injectable()
export class NotificationService {
  async sendEmail({ email, code, title }): Promise<string> {
    try {
      const transporter = nodemailer.createTransport({
        host: process.env.NOTIFICATION_HOST,
        port: process.env.NOTIFICATION_PORT,
        auth: {
          user: process.env.NOTIFICATION_EMAIL,
          pass: process.env.NOTIFICATION_PASSWORD,
        },
      });

      await transporter.sendMail({
        from: process.env.NOTIFICATION_EMAIL,
        to: email,
        subject: title,
        html: `<table role="presentation" style="padding: 10px; border: 1px solid
lightblue; font-size: 24px; margin: 0 auto; width: 326px; border-radius: 8px; font-
weight: 700; text-align: center;">
  <tr>
    <td>
      <h3 style="margin-top: 0;">${title}</h3>
    </td>
  </tr>
  <tr>
    <td>
      <table role="presentation" style="display: inline-table;">
        <tr style="display: flex;">
          ${String(code)
            .split("")
            .map(
              (el, index, array) =>
                `<td style="padding: 5px 10px; border: 1px solid lightblue; border-
radius: 4px; ${index !== array.length - 1 ? 'margin: 0 10px 0 0;' : 'margin:
0'};">${el}</td>`,
            )
            .join("")}
        </tr>
      </table>
    </td>
  </tr>
</table>`
      });
    }
  }
}

```

```
        </tr>
    </table>`,
    });

    return 'Успішно';
  } catch (error) {
    throw error;
  }
}
}
7;
```

Публікації

Proceedings of the XVII International scientific and practical conference «Information technologies and automation – 2024»

**Міністерство освіти і науки України
Одеський національний технологічний університет
Інститут комп'ютерної інженерії, автоматизації,
робототехніки та програмування ім.П.Н.Платонова**

**«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ І
АВТОМАТИЗАЦІЯ – 2024»**

***МАТЕРІАЛИ
XVII МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ***



31 ЖОВТНЯ - 1 ЛИСТОПАДА 2024 р.

м.Одеса

**ПРЕЗИДІЯ ТА ОРГКОМІТЕТ КОНФЕРЕНЦІЇ
PRESIDIUM AND ORGANIZING COMMITTEE OF THE CONFERENCE**

**ГОЛОВА ПРЕЗИДІЇ
CHAIRMAN OF THE PRESIDIUM**

Богдан Єгоров, Президент ОНТУ, академік НААН України, д.т.н., професор

**ЧЛЕНИ ПРЕЗИДІЇ
MEMBERS OF THE PRESIDIUM**

Надія Дец, к.т.н., доцент, в.о.ректора Одеського національного технологічного університету

Ольга Ольшевська, к.т.н., доцент, проректор з наукової роботи і міжнародних зв'язків Одеського національного технологічного університету.

**ГОЛОВА ОРГКОМІТЕТУ
CHAIRMAN OF THE ORGANIZING COMMITTEE**

Сергій Котлик, к.т.н., доц. каф. ІТтаКБ, ОНТУ

**ЗАСТУПНИК ГОЛОВИ ОРГКОМІТЕТУ
DEPUTY CHAIRMAN OF THE ORGANIZING COMMITTEE**

Виктор Хобін – д.т.н., професор кафедри АТПтаРС ОНТУ

**ЧЛЕНИ ОРГКОМІТЕТУ
MEMBERS OF THE ORGANIZING COMMITTEE**

Panagiotis Tzionas, prof. (Thessaloniki, Greece)

Qiang Huang, prof. (Los Angeles C.A., USA)

Yangmin Li, prof (Macao, China)

Артеменко С.В., проф., (Одеса, Україна)

Романюк О.Н., проф. (Вінниця, Україна)

Габко В.В., проф. (Вінниця, Україна)

Жученко А.І., проф. (Київ, Україна)

Ладанюк А.П., проф. (Київ, Україна)

Лисенко В.Ф., проф. (Київ, Україна)

Любчик Л.М., проф. (Харків, Україна)

Палов І., проф. (Русе, Болгарія)

Стовкова В.Д., доц. (Тракия, Болгарія)

Суслів В., доц. (Кошалін, Польща)

Артем'єв П., проф. (Ольштин, Польща)

Судацевські В., доц. (Кишинів, Молдова)

Аманжолова С., доц. (Алмати, Казахстан)

Інформаційні технології і автоматизація – 2024 / Матеріали XVII міжнародної науково-практичної конференції. Одеса, 31 жовтня - 1 листопада 2024 р. - Одеса, Видавництво ОНТУ, 2024 р. – 847 с.

Збірник включає матеріали доповідей учасників конференції, які об'єднані за тематичними напрямками конференції.

Збірник буде корисним як для фахівців і працівників фірм, зайнятих в області ІТ та автоматизації, так і для викладачів, магістрів і студентів вищих навчальних закладів, які навчаються за напрямками і спеціальностями програмного забезпечення, обчислювальної техніки і автоматизованих систем, прикладної математики та обробки інформації, буде корисним професіоналам з комп'ютерного моделювання та розробки комп'ютерних ігор.

Результати досліджень у збірнику представляють собою своєрідний зріз сучасного стану справ в перерахованих галузях знань, який може допомогти як фахівцям, так і студентам університетів скласти загальну картину розвитку інформаційних технологій та пов'язаних з ними питань.

Наукові праці згруповані за напрямками роботи конференції та наведені в алфавітному порядку прізвищ авторів.

Матеріали (тези доповідей) друкуються в авторській редакції. Відповідальність за якість та зміст публікацій несе автор.

Матеріали подано українською та англійською мовами.

Головний редактор збірника Сергій Котлик

ОПТИМІЗАЦІЯ НАВАНТАЖЕННЯ МУЛЬТИПРОЦЕСОРНИХ СИСТЕМ. Косолап А.І. (Дніпровський національний університет ім. О. Гончара, Україна)	72
ЕФЕКТИВНЕ ЗАСТОСУВАННЯ DATA SCIENCE ДЛЯ СТАЛОГО/СТІЙКОГО ФІНАНСОВОГО МЕНЕДЖМЕНТУ. Краснюк М.Т., Краснюк І.М. (Київський національний економічний університет імені Вадима Гетьмана, Україна, Київський національний університет технологій та дизайну, Україна)	74
ОБЧИСЛЕННЯ ПОКАЗНИКІВ ЛЯПУНОВА ДЛЯ ДИНАМІЧНОЇ СИСТЕМИ ЗАСОБАМИ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ. Кривченко Ю.В., Кривченко А.А. (ВСП "Одеський технічний фаховий коледж ОНУ", Україна)	77
ДОСЛІДЖЕННЯ МЕТОДОМ КЛАСТЕРИЗАЦІЇ СТАНУ ЦИФРОВОГО РОЗВИТКУ УКРАЇНИ СЕРЕД ДЕРЖАВ ЄВРОПЕЙСЬКОГО СОЮЗУ. Барченко Н.Л., Мартинова Н.С. (Сумський державний університет, Україна)	80
ОПТИМІЗАЦІЯ ПРЯМИХ КРИТЕРІЇВ ЯКОСТІ ІНФОРМАЦІЙНИХ УПРАВЛЮЮЧИХ СИСТЕМ. МIRONENKO A.O., SEVERIN V.P. (Національний технічний університет "Харківський політехнічний інститут" Україна)	83
РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ РОЗРАХУНКУ ЯКІСНИХ ПОКАЗНИКІВ МОЛОКА З ЛАКТУЛОЗОЮ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЇЇ РЕАЛІЗАЦІЇ. Найдюнов О.Ю., Котлик С.В. (Одеський національний технологічний університет, Україна)	84
МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ КІНЕТИКИ СТУПІНЧАСТОГО ПРОЦЕСУ НЕПРЯМОГО ВІДНОВЛЕННЯ ВІСІТИТУ З ГЕМАТИТУ ГАЗОМ СО. Пантелейков С. П. (Дніпровський державний технічний університет, Україна)	86
АНАЛІЗ ТИПІВ МОДЕЛЕЙ НЕЙРОННИХ МЕРЕЖ ДЛЯ ЗАДАЧІ ПРОГНОЗУВАННЯ ФІНАНСОВИХ ПОКАЗНИКІВ. Пасічник В. А., Коваленко О. О. (Вінницький Національний Технічний Університет, Україна)	89
ПОРІВНЯЛЬНИЙ АНАЛІЗ МОДЕЛЕЙ КРИПТІНГУ ЗА НІДХЕМОМ І ФЛАБЕЛЕМ. Пелих В. П. (Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ», Україна)	90
МОДЕЛЮВАННЯ АЕРОДИНАМІЧНИХ ПРОЦЕСІВ. Петров В.М., Гончаренко Д.Л. (Одеський національний технологічний університет, Україна), Познар С.С. (НВО Агро-Сімо-Машбуд, Україна), Жданов О.О. (Одеська державна академія будівництва та архітектури, Україна)	93
АНАЛІЗ ВПЛИВУ ЗОВНІШНІХ ФАКТОРІВ НА ІНДЕКС ЯКОСТІ ПОВІТРЯ. ПІДХІД НА ОСНОВІ МАШИННОГО НАВЧАННЯ. Петунін Г.Е.; Бойко Н.І., (Національний університет «Львівська політехніка», Україна)	96
ВИКОРИСТАННЯ МОВИ ПРОГРАМУВАННЯ PYTHON ДЛЯ РЕАЛІЗАЦІЇ МЕТОДУ ЯКОБІ ДЛЯ РОЗВ'ЯЗКУ СИСТЕМ ЛІНІЙНИХ РІВНЯНЬ. Плутенко О. Ю., Базиль О. О., Шовкопляс О. А. (Сумський державний університет, Україна)	99
ІМОВІРНІСНИЙ ГРАФ ЕКСПЛУАТАЦІЙНОЇ ФУНКЦІЙНОЇ ПОВЕДІНКИ ВУЗЛА КОМУТАЦІЇ З АДАПТИВНИМ МЕТОДОМ ВИВЕДЕННЯ ПАКЕТІВ З ДВОХ БУФЕРНИХ ЗАПАМ'ЯТОВУВАЛЬНИХ ПРИСТРОЇВ. Приймак Назар, Жук Юрій (Національний університет «Львівська політехніка», Україна)	100
МОДЕЛЮВАННЯ ДИНАМІКИ ОСАДЖЕННЯ ВОДИ В НАФП ЗА ДОПОМОГОЮ ЛІСТ-МЕРЕЖІ. Путятін Р.О., Цапар В.С. (Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Україна)	102
АНАЛІЗ ЕФЕКТИВНОСТІ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ ЗА ДОПОМОГОЮ МЕТОДУ VAR. Журбей А.В., Савіцький Р.С. (Державний університет «Житомирська політехніка», Україна)	105
РЕАЛІЗАЦІЯ ОБЧИСЛЮВАЛЬНО-ГЕОМЕТРИЧНИХ МЕТОДІВ ДЛЯ АНАЛІЗУ СКЛАДНИХ ПРОСТОРОВИХ ДАНИХ. Сенчило Т.С. (Житомирський державний університет імені І. Я. Франка, Україна)	106
ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВДОСКОНАЛЕННЯ КОРПОРАТИВНИХ РІШЕНЬ, КЕРОВАНИХ ВИГОДАМИ. Сініцина І.П., Слабоспицька О.О., Ігнатенко П.П. (Інститут програмних систем НАНУ, Україна)	109

АНАЛІЗ ЕФЕКТИВНОСТІ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ ЗА ДОПОМОГОЮ МЕТОДУ VaR

Журбей А.В., Савіцький Р.С. (ipzm231_zhav@student.ztu.edu.ua, kklk_srs@ztu.edu.ua)

Державний університет «Житомирська політехніка» (Україна)

У тезах ми розглядаємо метод VaR (Value at Risk) як один з ключових інструментів для оцінки ризиків інвестиційного портфеля. VaR дозволяє кількісно оцінити потенційні втрати за певний період часу з заданою ймовірністю. В роботі описані основні методи розрахунку VaR, включаючи історичне моделювання, аналітичний метод та метод Монте-Карло. Проведено аналіз переваг та недоліків кожного з методів, а також визначено їхню застосовність для різних типів портфелів. Особливо розглянуто обмеження методу VaR, зокрема припущення про нормальний розподіл доходностей та залежність від історичних даних, а також можливості їх подолання за допомогою інших підходів, таких як Conditional VaR. Результати дослідження підкреслюють важливість використання VaR у комплексі з іншими методами для ефективного управління інвестиційними ризиками.

Постановка проблеми. Метод VaR (Value at Risk) є одним із основних інструментів для кількісної оцінки ризиків інвестиційних портфелів, проте його застосування має низку обмежень і недоліків, таких як припущення про нормальний розподіл доходностей і залежність від історичних даних. В умовах нестабільного ринку та високих ризиків, викликаних зовнішніми факторами, існує необхідність розробки комплексного підходу до управління ризиками, що поєднує VaR з іншими методами, такими як Conditional VaR (CVaR) та стрес-тестування.

При дослідженні проблематики поставлено задачі:

- провести детальний аналіз різних методів розрахунку VaR (історичне моделювання, аналітичний метод, метод Монте-Карло) та визначити їхні переваги і недоліки;
- визначити доцільність використання кожного з методів VaR для різних типів інвестиційних портфелів;
- оцінити обмеження методу VaR, пов'язані з нормальним розподілом доходностей і залежністю від історичних даних;
- проаналізувати можливості подолання обмежень VaR шляхом застосування додаткових підходів, таких як CVaR та стрес-тестування.

Метод VaR дозволяє визначити максимальну можливу втрату за певний період часу з заданою ймовірністю [1]. Основна ідея полягає в оцінці рівня втрат, які можуть бути перевищеними лише у визначений відсоток випадків. Наприклад, для 95%-ого рівня довіри VaR показує таку суму втрат, яку інвестор не перевищить у 95% випадків за певний період. Існують кілька основних методів розрахунку VaR: метод історичного моделювання, метод варіаційно-коваріаційного аналізу [2] (аналітичний метод) та метод Монте-Карло [3]. Метод історичного моделювання [4] базується на припущенні, що майбутні ринкові рухи можуть бути змодельовані на основі історичних даних. Аналітичний метод передбачає, що доходності активів мають нормальний розподіл, і використовується для великих портфелів, дозволяючи швидко розраховувати VaR. Метод Монте-Карло моделює можливі сценарії розвитку ринку з урахуванням випадкових факторів, що робить його найточнішим, але водночас найбільш ресурсоземним.

Метод VaR є універсальним інструментом, який дозволяє порівнювати ризики між різними типами активів або портфелями. Однією з основних переваг є уніфікація оцінки ризиків, що робить його застосовним незалежно від структури портфеля. Результати розрахунків легко інтерпретувати та використовувати для прийняття управлінських рішень, що дозволяє інвесторам оцінити безпеку своїх інвестицій. Крім того, VaR може бути адаптований до різних часових горизонтів і рівнів довіри, що робить його гнучким для різних сценаріїв. Його також можна використовувати для оптимізації структури портфеля, мінімізуючи ризики шляхом зменшення експозиції до активів з вищим VaR або збільшення частки менш ризикованих активів.

Попри численні переваги, метод VaR має і свої обмеження. Основною проблемою є припущення про нормальний розподіл доходностей активів, що не завжди є коректним, особливо під час кризових періодів. Наприклад, метод не враховує "хвості розподілу", тобто події з

низькою ймовірністю, але великими втратами, такі як фінансові кризи або несподівані ринкові шоки. Ще одним недоліком є залежність від історичних даних. Оскільки VaR базується на припущенні, що майбутнє схоже на минуле, він може бути недостатньо точним у випадках раптових змін ринку. Також VaR показує лише рівень втрат до певного порогу, але не дає уявлення про можливі втрати за межами цього порогу. Для таких випадків використовуються додаткові методи, такі як Conditional VaR (CVaR) [5], які оцінюють середні втрати в умовах екстремальних подій.

Метод VaR широко застосовується в практиці фінансових установ, зокрема банками та інвестиційними фондами, для управління ризиками на рівні окремих активів, портфелів, а також для моніторингу загального ризику організації. Наприклад, банки використовують VaR для визначення рівня капіталу, необхідного для покриття можливих втрат на ринках, а інвестиційні фонди — для оцінки ризиків своїх стратегій та прийняття рішень щодо диверсифікації активів. Одним із практичних прикладів може бути використання VaR для оцінки ризиків у портфелях, що складаються з різних класів активів, таких як акції, облігації та деривативи. Завдяки VaR інвестори можуть визначити, які активи мають найвищий ризик і чи варто зменшувати їхню частку в портфелі для зниження загальних ризиків.

Таким чином, метод VaR є потужним інструментом для оцінки ризиків та ефективності інвестиційного портфеля. Він дозволяє інвесторам і фінансовим установам кількісно оцінювати рівень потенційних втрат та приймати обґрунтовані рішення щодо структури та управління портфелем. Однак для більш точного аналізу ризиків необхідно враховувати обмеження цього методу і застосовувати його в комплексі з іншими підходами, такими як CVaR або стрес-тестування, для забезпечення максимальної точності та надійності аналізу.

Список використаної літератури

- [1] Understanding Value at Risk (VaR) and How It's Computed. [Online]. Available: <https://www.investopedia.com/terms/v/var.asp>.
- [2] What is the variance-covariance matrix? [Online]. Available: <https://support.minitab.com/en-us/minitab/help-and-how-to/statistical-modeling/anova/supporting-topics/anova-statistics/what-is-the-variance-covariance-matrix/>.
- [3] Quickly compute Value at Risk with Monte Carlo. [Online]. Available: <https://www.pyquantnews.com/the-pyquant-newsletter/quickly-compute-value-at-risk-with-monte-carlo>.
- [4] The historical method for VaR calculation. [Online]. Available: https://www.simtrade.fr/blog_simtrade/historical-method-var-calculation/.
- [5] Conditional Value at Risk (CVaR): Definition, Uses, Formula. [Online]. Available: https://www.investopedia.com/terms/c/conditional_value_at_risk.asp.

РЕАЛІЗАЦІЯ ОБЧИСЛЮВАЛЬНО-ГЕОМЕТРИЧНИХ МЕТОДІВ ДЛЯ АНАЛІЗУ СКЛАДНИХ ПРОСТОРОВИХ ДАНИХ

Сенчило Т.С. (senchilo578512@gmail.com)

Житомирський державний університет імені І. Я. Франка (Україна)

Дана робота присвячена вивченню і практичному застосуванню сучасних обчислювально-геометричних методів аналізу складних просторових даних. Продемонстровано, основні алгоритми та їх застосування, зосереджуючись на трьох основних проблемах побудови откритої оболонки, триангуляції Делоне та пошуку найближчих сусідів. Дане дослідження починається з огляду важливості обчислювальної геометрії в різних наукових галузях, вкочуючи комп'ютерну графіку, географічні інформаційні системи та робототехніку.

У сучасному світі стрімко зростають обсяги та складність просторових даних, що використовуються в різних галузях, таких як комп'ютерна графіка, географічні інформаційні системи (ГІС) та робототехніка. Це зростання створило нагальну потребу в ефективних способах обробки та аналізу таких даних.

Національний університет «Полтавська політехніка імені Юрія Кондратюка»
 Департамент економічного розвитку, торгівлі та залучення інвестицій
 Полтавської обласної військової адміністрації
 Полтавська торгово-промислова палата
 Університет Флорида (США)
 "1 DECEMBRIE 1918" University of Alba Iulia (Румунія)
 Білостоцький технологічний університет (Польща)
 Вільнюський університет прикладних наук (VIKO) (Литва)
 London Metropolitan University (Велика Британія)
 Словацький технологічний університет (Словаччина)
 Рада молодих вчених Національної академії наук України
 Рада молодих вчених Національного університету «Запорізька політехніка»
 Рада молодих вчених Національного технічного університету «Дніпровська політехніка»
 Рада молодих вчених Національного університету «Чернігівська політехніка»
 Рада молодих вчених Національного університету «Одеська політехніка»
 Рада молодих вчених Одеського національного університету імені І.І. Мечникова
 Рада молодих вчених Ізмайльського державного гуманітарного університету
 Рада молодих вчених Глухівського національного педагогічного університету
 імені Олександра Довженка
 Рада молодих вчених Сумського національного аграрного університету
 Рада молодих вчених Національного технічного університету України
 «Київський політехнічний інститут імені Ігоря Сікорського»
 Рада молодих вчених Харківського національного педагогічного університету імені Г.С. Сковороди
 Рада молодих вчених Чернівецького національного університету імені Юрія Федьковича
 Рада молодих вчених Хмельницького національного університету
 Рада молодих вчених Київського національного університету будівництва та архітектури
 Рада молодих вчених Херсонського державного аграрно-економічного університету

МОЛОДІЖНА НАУКА: ІННОВАЦІЇ ТА ГЛОБАЛЬНІ ВИКЛИКИ

ЗБІРНИК ТЕЗ

**Міжнародної науково-практичної конференції студентів,
аспірантів та молодих вчених**



Полтава, 06 листопада 2024 року

Молодіжна наука: інновації та глобальні виклики // Збірник тез за матеріалами Міжнародної науково-практичної конференції студентів, аспірантів та молодих вчених – Полтава: НУПІ імені Юрія Кондратюка, 2024. – 727 с.

У збірнику тез опубліковані результати наукових досліджень студентів, аспірантів та молодих учених, які були включені в програму Міжнародної науково-практичної конференції студентів, аспірантів та молодих вчених «Молодіжна наука: інновації та глобальні виклики» (06 листопада 2024 року, Національний університет «Полтавська політехніка імені Юрія Кондратюка»). Тези подано в авторському оригіналі українською чи англійською мовами.

Редакційна колегія

МАСЛІЙ Олександра – голова ради молодих вчених Національного університету «Полтавська політехніка імені Юрія Кондратюка», к.е.н., доцент;

ЧАЙКІНА Аліна – голова ради навчально-наукового інституту фінансів, економіки, управління та права, к.е.н., доцент;

ПАЛІЙ Катерина – голова ради молодих вчених факультету філології, психології та педагогіки, к.ф.н., доцент кафедри;

КРАВЧЕНКО Михайло – голова ради факультету фізичної культури та спорту, асистент кафедри;

ГАСЕНКО Антон – голова ради молодих вчених навчально-наукового інституту архітектури, будівництва та землеустрою, д.т.н., професор кафедри;

БОРЯК Богдан – голова ради молодих вчених навчально-наукового інституту інформаційних технологій та робототехніки, к.т.н., доцент кафедри;

ЛЕВЧЕНКО Ірина – доцент кафедри міжнародних економічних відносин та туризму, доктор філософії;

МАКСЮТА Наталія – начальниця відділу аналізу інноваційної діяльності та інтелектуальної власності, доктор філософії;

ЛЕВЧЕНКО Валерій – Президент Студентського парламенту університету;

БУРЯК Альона – доцент кафедри міжнародних економічних відносин та туризму, к.е.н., доцент;

УСЕНКО Дмитро – доцент кафедри хімії та фізики, магістр фізики, доктор філософії;

ЙОПА Тетяна – старший викладач кафедри фізичної культури та спорту, кандидат в майстри спорту України з дзю-до;

РІЗНИК Юлія – старший викладач кафедри загального мовознавства та іноземних мов;

ЛАКТИОНОВ Олександр – доцент кафедри автоматики електроніки та телекомунікацій, к.т.н., доцент кафедри.

© Національний університет «Полтавська політехніка імені Юрія Кондратюка», 2024 рік

<i>Жмуренко Віталій Геннадійович</i>	103
ГЛОБАЛЬНІ ТRENДИ РОЗВИТКУ МІЖНАРОДНОЇ ТОРГІВЛІ	103
<i>Журбей Артем Валентинович</i>	105
<i>Савіцький Роман Святославович</i>	105
МОДЕЛЮВАННЯ РИЗИКІВ ТА ДОХІДНОСТІ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ НА ОСНОВІ ТЕОРІЇ САРМ	105
<i>Зарва Аліна Олександрівна</i>	107
<i>Науковий керівник: Штанова Альона Леонідівна</i>	107
ТРЕНДИ КОМУНІКАЦІЙ В СИСТЕМІ МАРКЕТИНГУ МАРКЕТИНГУ ПІД ЧАС ВІЙНИ	107
<i>Зута Катерина</i>	109
DEVELOPMENT OF THE JURY SYSTEM IN UKRAINE	109
<i>Івахненко Ірина Сергіївна</i>	112
<i>Іванко Вікторія Олександрівна</i>	112
ЗАРУБІЖНИЙ ДОСВІД ВПРОВАДЖЕННЯ УПРАВЛІНСЬКИХ ІННОВАЦІЙ НА ПІДПРИЄМСТВАХ БУДІВЕЛЬНОЇ ГАЛУЗІ	112
<i>Кандзюба Інга Володимирівна</i>	114
<i>Гитик Анна Олексіївна</i>	114
СОЦІАЛЬНІ НАСЛІДКИ ІННОВАЦІЙ	114
<i>Кандзюба Інга Володимирівна</i>	116
<i>Гитик Яна Олексіївна</i>	116
ТЕХНОЛОГІЧНІ ТRENДИ ТА ЇХНІЙ ВПЛИВ НА ЕКОНОМІКУ	116
<i>Кандзюба Інга Володимирівна</i>	118
<i>Науковий керівник: Чичкало-Кондрацька Ірина Борисівна</i>	118
МІЖНАРОДНЕ СПІВРІБІТНИЦТВО В ГАЛУЗІ ІННОВАЦІЙ	118
<i>Карпінєць Анатолій Олександрович</i>	121
ФУНКЦІЇ ПОДАТКІВ	121
<i>Коваленко Поліна Миколаївна</i>	123
ШТУЧНИЙ ІНТЕЛЕКТ ЯК ДРАЙВЕР РОЗВИТКУ НАЦІОНАЛЬНОЇ ЕКОНОМІКИ	123
<i>Козлова Вікторія Сергіївна</i>	125
<i>Науковий керівник: Чубур Наталія Володимирівна</i>	125
ВПЛИВ ЗАСОБІВ МАСОВОЇ ІНФОРМАЦІЇ НА РОЗВИТОК ДЕМОКРАТІЇ	125
<i>Комаровський Іван Вікторович</i>	127
ЗАСТОСУВАННЯ ПОЛОЖЕНЬ ТЕОРІЇ «СЛАБКИХ СИГНАЛІВ» В СИСТЕМІ ПЛАНУВАННЯ	127

УДК 004.4

Журбей Артем Валентинович
студент
Савіцький Роман Святославович
аспірант

Державний університет «Житомирська політехніка»

МОДЕЛЮВАННЯ РИЗИКІВ ТА ДОХІДНОСТІ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ НА ОСНОВІ ТЕОРІЇ CAPM

Сучасна теорія портфелю є фундаментальною концепцією в управлінні інвестиціями, що допомагає інвесторам оптимізувати ризик та доходність своїх портфелів. Одним з ключових концептів теорії є модель оцінки капітальних активів (Capital Asset Pricing Model — CAPM) [1], яка описує взаємозв'язок між очікуваною доходністю активу та ризиком, вираженим через бета-коефіцієнт. CAPM є потужним інструментом для моделювання та аналізу ризиків і доходності інвестиційних портфелів, дозволяючи інвесторам приймати зважені рішення на основі обґрунтованих розрахунків. Модель CAPM відіграє важливу роль у теорії інвестицій і широко застосовується для оцінки ефективності інвестиційних стратегій та прийняття фінансових рішень.

Основна ідея CAPM полягає в тому, що очікувана доходність активу залежить від безризикової ставки доходності, ринкової премії за ризик та бета-коефіцієнта активу. Бета-коефіцієнт вимірює чутливість доходності активу до змін загального ринку. Якщо бета-коефіцієнт більше ніж 1, актив більш чутливий до змін на ринку та несе вищий ризик, ніж ринок загалом. Якщо бета менше за 1, актив є менш ризиковим. Модель CAPM передбачає, що інвестори отримують компенсацію за прийняття систематичного ризику, який неможливо усунути за допомогою диверсифікації.

Основне рівняння моделі CAPM виглядає так:

$$E(R_i) = R_f + \beta_i(E(R_m) - R_f)$$

$E(R_i)$ — очікувана доходність активу i ;

R_f — безризикова ставка доходності;

β_i — бета-коефіцієнт активу i ;

$E(R_m)$ — очікувана доходність ринкового портфеля;

$E(R_m) - R_f$ — ринкова премія за ризик.

Модель CAPM дозволяє інвесторам визначити мінімально прийнятну доходність для активу або портфелю активів із певним рівнем ризику. В рамках сучасної теорії портфелю, інвестори можуть комбінувати різні активи для досягнення оптимальної комбінації ризику та доходності, використовуючи капіталізаційно-зважену ринкову лінію (Capital Market Line — CML) [2] та лінію ринку цінних паперів (Security Market Line —

SML) [3]. Інструменти допомагають візуалізувати співвідношення між ризиком і доходністю, що є критично важливим для прийняття інвестиційних рішень. За допомогою бета-коефіцієнтів інвестори передбачають поведінку активів у відповідь на зміни ринкових умов. Інвестор, який очікує зростання ринку, може обрати активи з високим значенням бета для збільшення потенційної доходності, водночас розуміючи, що активи будуть більш ризикованими.

Однією з головних критичних обмежень CAPM є припущення про ідеальний ринок, де всі інвестори мають однакову інформацію, не існує транзакційних витрат, а всі активи можуть бути куплені або продані без обмежень. У реальному світі умови рідко виконуються, що призводить до певних викривлень у результатах моделі. Крім того, CAPM базується на історичних даних, і хоча вона може бути точною при моделюванні минулих ринкових умов, це не завжди є надійно для передбачення майбутніх подій, особливо під час кризових періодів або нестабільності ринків.

Важливим аспектом є те, що CAPM розглядає лише систематичний ризик, не враховуючи інші фактори, такі як ліквідність активів, поведінкові аспекти інвесторів або макроекономічні зміни. Для більш точного моделювання ризиків та доходностей багато інвесторів використовують додаткові моделі, такі як трьохфакторна модель Фама-Френча [4], яка враховує розмір компанії та вартісну премію. Таким чином, модель CAPM є потужним інструментом для оцінки співвідношення ризику та доходності в інвестиційних портфелях, що дозволяє інвесторам приймати обґрунтовані рішення щодо управління активами.

Список використаних джерел

1. Capital Asset Pricing Model (CAPM): Definition, Formula, and Assumptions [Електронний ресурс]. – Режим доступу: <https://www.investopedia.com/terms/c/capm.asp>
2. Capital Market Line (CML): Definition & Formula [Електронний ресурс]. – Режим доступу: <https://www.freshbooks.com/glossary/financial/capital-market-line>
3. Security Market Line (SML) [Електронний ресурс]. – Режим доступу: <https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/security-market-line-sml/>
4. How Does the Fama French 3 Factor Model Work? [Електронний ресурс]. – Режим доступу: <https://smartasset.com/investing/fama-french-3-factor-model>