ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Дослідження та автоматизація процесів ведення податкової звітності з інвестиційної діяльності на основі технологій Java»

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Назарій БАБІЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Назарій БАБІЙ
група КНДМ-61

Керівник: Віктор ВИШНІВСЬКИЙ
науковий ступінь, вчене звання д.т.н., професор

Рецензент: _____
науковий ступінь, вчене звання

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Бабію Назарію Юрійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Дослідження та автоматизація процесів ведення податкової звітності з інвестиційної діяльності на основі технологій Java

керівник кваліфікаційної роботи Віктор Вишнівський, д.т.н, професор,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» жовтня 2023 р. № 145

2. Строк подання кваліфікаційної роботи «20» грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: Науково-технічна література з питань, що пов'язані з побудови клієнт-серверної архітектури

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження процесів податкової звітності в інвестиційній діяльності
2. Дослідження застосування мови програмування Java в фінансовій сфері

3. Розробка прототипу системи автоматичного розрахунку інвестиційного прибутку і підготовки податкової звітності

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	01.09 – 05.09	
2	Дослідження поставленої задачі	06.09 – 14.09	
3	Аналіз методів розв’язання задачі	14.09 – 25.09	
4	Застосування клієнт-серверної архітектури на практиці	26.09 – 06.10	
5	Аналіз отриманих результатів	07.10 – 14.10	
6	Розробка прототипу системи	15.10 – 10.12	
7	Вступ, висновки, реферат	15.10 – 25.10	
8	Основні розділи	25.10 – 10.12	

Здобувач вищої освіти _____ Назарій БАБІЙ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Віктор ВИШНІВСЬКИЙ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 84 стор., 22 рис., 4 табл., 26 джерел.

Мета роботи – побудова прототипу серверного рішення для веб-платформи, яка надає можливість розрахувати інвестиційний прибуток з торгівлі акціями та сформувати додаток Ф1 для податкового звіту.

Об'єкт дослідження – системи фінансового обліку.

Предмет дослідження – розробка серверної частина проєкту за допомогою мови програмування Java та Spring Framework.

Короткий зміст роботи: У дослідженні податкової звітності в інвестиційній діяльності українських роздрібних інвесторів виявлено, що більшість з них використовують міжнародні брокерські компанії, такі як Freedom Broker і Interactive Brokers, для торгівлі на зарубіжних фондових біржах. Однак ці компанії не є податковими резидентами в Україні, що вимагає від інвесторів самостійно готувати та подавати податкові декларації згідно з українським законодавством.

Аналізуючи податкову базу України, виявлено правила обчислення та податкові ставки для оподаткування інвестиційної діяльності. Україна також приєдналася до асоціації для автоматичного обміну даними про фінансові рахунки, що може призвести до штрафів для інвесторів, якщо вони не декларуватимуть свої доходи.

Автор, який працює інженером-програмістом в українському банку, розглядає можливість автоматизації підготовки податкового звіту з інвестиційної діяльності за допомогою Java та Spring Framework. Висвітлені п'ять завдань, вирішення яких допоможе побудувати комплексний прототип для формування податкового звіту.

Розроблено прототип програмного рішення, яке являє собою серверний застосунок на Spring Framework, що запакований у docker контейнер і розгорнуто на хостингу Digital Ocean. Серверний застосунок використовує надійну і швидкісну базу даних PostgreSQL, як сервіс хостингу. Клієнтський застосунок отримує доступ завдяки REST API та HTTP протоколу. Кінцевий користувач, а саме український інвестор може розрахувати і сформувати податковий звіт за 5-10 хвилин.

КЛЮЧОВІ СЛОВА: ІНВЕСТИЦІЙНА ДІЯЛЬНІСТЬ, ПОДАТКОВА ЗВІТНІСТЬ, JAVA, SPRING FRAMEWORK, FIFO, БРОКЕРСЬКІ ЗВІТИ, ФОНДОВА БІРЖА, КЛІЄНТ-СЕРВЕПНА АРХІТЕКТУРА.

ABSTRACT

Text part of the master's qualification work:
84 pages, 26 pictures, 4 tables, 26 sources.

The purpose of the work is developing a prototype of a server solution for a web platform that provides an opportunity to calculate investment income from share trading and generate an application F1 for a tax report.

Object of research – financial accounting systems.

Subject of research – development of the server part of the project using the Java programming language and the Spring Framework.

Summary of the work: In the study of tax reporting in the investment activities of Ukrainian retail investors, it was found that most of them use international brokerage companies, such as Freedom Broker and Interactive Brokers, for trading on foreign stock exchanges. However, these companies are not tax residents in Ukraine, which requires investors to independently prepare and submit tax returns in accordance with Ukrainian law.

Analyzing the tax base of Ukraine, calculation rules and tax rates for taxation of investment activities were revealed. Ukraine has also joined an association for the automatic exchange of data on financial accounts, which can lead to fines for investors if they do not declare their income.

The author, who works as a software engineer in a Ukrainian bank, is considering the possibility of automating the preparation of a tax report on investment activities using Java and the Spring Framework. Five tasks are highlighted, the solution of which will help to build a complex prototype for the formation of a tax report.

A prototype software solution was developed, which is a server application based on the Spring Framework, packaged in a docker container and deployed on Digital Ocean hosting. The server application uses a reliable and high-speed PostgreSQL database as a hosting service. The client application is accessed through REST API and HTTP protocol. The end user, namely the Ukrainian investor, can calculate and generate a tax report in 5-10 minutes.

KEYWORDS: INVESTMENT ACTIVITIES, TAX REPORTING, JAVA, SPRING FRAMEWORK, FIFO, BROKERAGE REPORTS, STOCK EXCHANGE, CUSTOMER SERVICE ARCHITECTURE.

ЗМІСТ

ВСТУП 10

1 ПОДАТКОВА ЗВІТНІСТЬ В ІНВЕСТИЦІЙНІЙ ДІЯЛЬНОСТІ 12

1.1 Інвестиційне середовище в Україні 12
1.1.1 Хто такий український роздрібний інвестор? 12
1.1.2 Внутрішній ринок 14
1.1.3 Зарубіжні ринки 15
1.1.4 Хто дає доступ до фондових бірж? 17
1.1.5 Якими інвестиційними інструментами користуються інвестори? 18
1.2 Законодавча податкова база 19
1.2.1 Аналіз правил обчислення інвестиційного прибутку 20
1.2.2 Автоматичний обмін даними 24
1.3 Процес ведення податкової звітності 26
1.3.1 Документація, податкові ставки та штрафні санкції 26
1.3.2 FIFO принцип 28
1.3.3 Як це інвестори роблять зараз? 30
1.4 Висновки до розділу 32

2 ОГЛЯД ТЕХНОЛОГІЙ JAVA ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ 33

2.1 Мова програмування Java 33
2.2 Фреймворки для розробки підприємницьких додатків 39
2.3 Детальніше про Spring Framework 44
2.4 Використання технологій Java у фінансовій сфері 50

3 ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО РІШЕННЯ 57

3.1 Функціональні та технічні вимоги до програмного забезпечення 57
3.2 Проектування архітектури системи 60
3.3 Модель даних для збереження операцій торгової активності інвестора зі звітів брокерів 63
3.4 Збагачення даними операцій з брокерських звітів актуальними обмінними курсами валют НБУ 67
3.5 Алгоритм розрахунку інвестиційного прибутку 70
3.6 Механізм генерації вихідних файлів для зручного імпортування в податковий кабінет 75
3.7 Шифрування звітів брокерів та вихідних файлів для забезпечення безпеки передачі даних між серверним і клієнтським застосунками 78
3.8 API 82

4 ВПРОВАДЖЕННЯ ТА ТЕСТУВАННЯ 86

4.1 Процес впровадження програмного продукту 86
4.2 Тестування та оцінка ефективності 89

ВИСНОВКИ 91

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 94

ДОДАТОК А. ПРОГРАМНИЙ КОД АЛГОРИТМУ РОЗПОДІЛУ ОПЕРАЦІЙ ЗА ПРИНЦИПОМ FIFO 97

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ) 104

ВСТУП

Тема автоматизації процесів податкової звітності в інвестиційні діяльності є актуальною з багатьох причин. Звісно головна причина сплати податків – це закон України про Податковий Кодекс, який зобов'язує оподатковувати інвестиційну діяльність. Та сам процес розрахунку і формування податкової звітності є рутинним через велику кількість розрахунків, а як показує практика, інвестор не завжди володіє інформацією, як саме цей розрахунок виконати.

Метою дослідження є побудова прототипу серверного рішення для веб-платформи, яка надає можливість розрахувати інвестиційний прибуток з торгівлі акціями та сформувати додаток Ф1 для податкового звіту. Для її досягнення необхідно розв'язати наступні задачі, а саме дослідити та реалізувати:

1. Модель даних для збереження операцій торгової активності інвестора зі звітів брокерів;
2. Збагачення даними операцій з брокерських звітів актуальними обмінними курсами валют НБУ;
3. Алгоритм розрахунку інвестиційного прибутку;
4. Механізм генерації вихідних файлів в форматі податкового кабінету для зручного імпортування в податковий кабінет;
5. Шифрування звітів брокерів та вихідних файлів для забезпечення безпеки передачі даних між серверним і клієнтським застосунками.

Об'єктом дослідження являються системи фінансового обліку.

Предмет дослідження – розробка серверної частина проєкту за допомогою мови програмування Java та Spring Framework.

Методи дослідження – виявлення вимог через проведення опитувань, створення ПЗ для взаємодії з третіми сервісами, побудова та розробка REST API для клієнтських застосунків і сервісного модуля за допомогою мови програмування Java та Spring Framework.

Сфера застосування – сфера бухгалтерських послуг

Джерела дослідження:

- «Java in Action» авторства Raoul-Gabriel Urma, Mario Fusco, та Alan Mycroft – для вивчення реальних прикладів використання Java в бізнес-застосунках;
- "Spring in Action" авторства Craig Walls - для поглиблення розуміння використання Spring Framework у розробці програмного забезпечення.
- Офіційна документація Spring Framework та Java – для отримання технічних деталей та прикладів використання технологій.
- Спільноти програмістів на форумах, таких як Stack Overflow та Reddit, для обговорення питань, пов'язаних із застосуванням Java в інвестиційній діяльності.
- Участь в вебінарах та конференціях, присвячених Java-розробці та фінансовим технологіям, для отримання інсайтів від експертів галузі.
- Доступ до офіційних сайтів податкових органів для отримання інформації про вимоги до податкової звітності та стандарти її автоматизації.
- Документація та інформація від фінансових організацій, які використовують Java-технології для автоматизації інвестиційної звітності.

Наукова новизна одержаних результатів: розроблено алгоритм розподілу торгових операцій за принципом FIFO, що є основою для розрахунку інвестиційного прибутку і похідних показників, таких як податки.

Практичне значення одержаних результатів: вирішення поставлених задач дозволило реалізувати комплексне рішення, що автоматизує процес формування податкової звітності з інвестиційної діяльності українського інвестора за увесь період його торгів.

1 ПОДАТКОВА ЗВІТНІСТЬ В ІНВЕСТИЦІЙНІЙ ДІЯЛЬНОСТІ

1.1 Інвестиційне середовище в Україні

Пропоную спершу розглянути інвестиційне середовище в Україні, бо тема оподаткування є похідною від нього.

1.1.1 Хто такий український роздрібний інвестор?

Український роздрібний інвестор на фондовій біржі – це фізична особа, яка вкладає свої гроші в різні фінансові інструменти, такі як акції, облігації, фонди та інші цінні папери, з метою отримання прибутку для забезпечення свого майбутнього, зростання капіталу або отримання додаткового доходу.

Україна, як і багато інших країн, має свою фондову біржу, де інвестори можуть купувати та продавати цінні папери. Українські роздрібні інвестори можуть вибирати серед різноманітних інструментів та стратегій відповідно до своїх фінансових цілей та ризикового профілю.

Цілі українських роздрібних інвесторів на фондовій біржі можуть бути різноманітними, і вони часто залежать від індивідуальних фінансових цілей, терміну інвестування та рівня ризику. Ось деякі загальні цілі, які можуть переслідувати роздрібні інвестори:

- **Забезпечення пенсійного забезпечення**: Деякі інвестори можуть вкладати гроші в цінні папери з метою накопичення достатніх коштів для забезпечення комфортного пенсійного життя.
- **Зростання капіталу:** Інші можуть прагнути збільшити свій початковий капітал, вкладаючи в акції та інші інструменти з потенційно високим прибутковим потенціалом.

- **Отримання стабільного доходу:** Деякі інвестори можуть обирати облігації або інші інструменти, що приносять проценти, з метою отримання стабільного доходу.
- **Диверсифікація портфеля:** Інші можуть прагнути розширити свій портфель, розподіляючи ризики між різними видами активів.
- **Спекуляції та короткостроковий прибуток:** Деякі інвестори можуть бути готові приймати більше ризику з метою отримання швидкого короткострокового прибутку, наприклад, торгівля акціями на короткі терміни або участь у високоризикових інвестиціях.

Найважливішим є те, що кожен інвестор повинен чітко визначити свої фінансові цілі, термін інвестування та рівень комфорту з ризиком перед тим, як вибирати конкретні інвестиційні стратегії та інструменти.

На жаль відкритої статистики про портрет українського інвестора немає, але за даними аналітики Альфа-Банку основні характеристики клієнтів, які торгують акціями у цифровому банку (в періоді з 28 грудня 2021 року до 10 січня 2022 року):

- 87% чоловіки;
- 60% у віці 30 — 45 років;
- середній вік — 37 років;
- 44% проживають у Києві та Київській області;
- ТОП-5 регіонів розподілилися так: 1 — Київ та Київська область; 2 — Харківська область; 3 — Дніпропетровська область, 4 — Одеська область; 5 — Львівська область;
- середній чек за угодами — близько 10 тис грн.

Середньостатистичний портфель інвестора в Sense SuperApp:

- портфель складається з акцій 3−4 компаній;
- сума портфеля понад 55 тис грн.

Загалом обсяг торгів (купівель та продажів акцій) у період з 28 грудня 2021 року до 21 січня 2022 року становив понад 43 млн грн. І якщо під час пілотного проєкту клієнти вивчали інструмент та купували 1-2 акції, то зараз(на момент публікації аналітичних даних, 24 січня 2021) кількість акцій у кожного клієнта збільшується та в середньому становить 8 шт. різних компаній.

1.1.2 Внутрішній ринок

Внутрішній інвестиційний ринок України функціонує у рамках ряду регуляторів та інших інстанцій. Основним регулятором є Національна комісія з цінних паперів та фондового ринку (НКЦПФР). Ось кілька ключових аспектів:

- **Національна комісія з цінних паперів та фондового ринку (НКЦПФР):** Це головний регулятор фондового ринку в Україні. Вона відповідає за розробку та впровадження правил і стандартів для функціонування фондового ринку, реєстрацію емітентів, ліцензування учасників ринку, контроль за дотриманням ними встановлених норм і багато іншого.
- **Національний банк України (НБУ):** НБУ відіграє важливу роль у фінансовій системі України та може впливати на інвестиційний клімат через монетарну політику та інші заходи.
- **Державна комісія з регулювання ринків фінансових послуг України (ДКРРФП):** Ця інстанція також взаємодіє з ринком цінних паперів та має на меті забезпечення стабільності фінансової системи та захист прав споживачів фінансових послуг.
- **Міністерство фінансів України:** Міністерство фінансів грає роль у розробці фінансової політики країни, а також може вносити пропозиції щодо регулювання ринку цінних паперів.

- **Українська біржа (зокрема, Українська фондова біржа та Міжнародна валютна біржа):** Ці біржі відіграють ключову роль у функціонуванні фондового ринку, надаючи платформу для торгівлі акціями, облігаціями та іншими цінними паперами.
- **Державна податкова служба України (ДПСУ):** відіграє важливу роль у зборі податків та визначенні податкових ставок для різних видів інвестицій та фінансових операцій.

На жаль українська фондова біржа «мертва». Ринок існує з часу зародження незалежності, але в реальності він не виконує своїх базових функцій. Відсутність повноцінних фондових майданчиків позбавляє нашу економіку мільярдних інвестицій, які йдуть в інші країни. Тому українські роздрібні інвестори вибирають зарубіжні ринки.

1.1.3 Зарубіжні ринки

Роздрібнім інвесторам з України є доступ до численних зарубіжних фондових ринків, і цей доступ зазвичай здійснюється через міжнародні брокерські компанії або фінансові інститути. Ось деякі з найпопулярніших зарубіжних фондових ринків, до яких можуть мати доступ українські інвестори:

- **NYSE (New York Stock Exchange) та NASDAQ в США:** Два з найбільших фондових ринків у світі, розташовані в Сполучених Штатах.
- **LSE (London Stock Exchange) в Великобританії:** Один із найбільших фондових ринків у Європі.
- **TSE (Tokyo Stock Exchange) в Японії:** Найбільший фондовий ринок в Азії.
- **Euronext в Європі:** Операції в амстердамському, бельгійському, французькому та іспанському фінансових ринках.
- **HKEX (Hong Kong Stock Exchange) в Гонконзі:** Один з найважливіших фондових ринків у Південній Азії.

Для доступу до цих ринків українські роздрібні інвестори можуть відкрити торговий рахунок у міжнародних брокерських компаніях або банках, які надають послуги міжнародного інвестування. Ці брокери часто мають онлайн-платформи, які дозволяють інвесторам здійснювати торгівлю акціями, облігаціями та іншими фінансовими інструментами на світових ринках.

Щоб відкрити міжнародний торговий рахунок, інвестор повинен буде надати необхідні документи та проходити процес верифікації. Також важливо враховувати валютний ризик, оскільки торгівля на іноземних ринках виконується часто в інших валютах.

Кожна країна має свої власні фінансові регулятори, які надають нагляд та регулюють фінансові ринки. Нижче перераховані деякі з основних фінансових регуляторів у країнах зазначених фондових ринків:

- **Сполучені Штати:** *SEC (U.S. Securities and Exchange Commission):* Основний федеральний регулятор цінних паперів та фондових ринків у Сполучених Штатах.
- **Велика Британія:** *FCA (Financial Conduct Authority):* Орган нагляду за фінансовою діяльністю та поведінковими стандартами в Великобританії.
- **Японія:** *FSA (Financial Services Agency):* Орган урядового нагляду за фінансовими інститутами та фондовим ринком в Японії.
- **КНР (Гонконг):** *SFC (Securities and Futures Commission):* Орган, що регулює ринки цінних паперів та ф’ючерсів у Гонконзі.

Це лише кілька прикладів, і регуляторів може бути більше, оскільки кожна країна має свою систему фінансового регулювання. Інші європейські країни мають власні органи регулювання, такі як BaFin в Німеччині, AMF у Франції, або CONSOB в Італії.

При інвестуванні на зарубіжних фондових ринках, важливо бути свідомим та вивчити правила та регуляції, які стосуються конкретного ринку та інвестиційних продуктів.

1.1.4 Хто дає доступ до фондових бірж?

Українські інвестори можуть отримати доступ до світових фондових бірж через міжнародні брокерські компанії. Ці компанії спеціалізуються на наданні послуг з міжнародного інвестування та торгівлі цінними паперами на різних фондових ринках. Лідери ринку з міжнародних брокерських компаній, які надають доступ для українських інвесторів, включають Interactive Brokers і Freedom Broker.

Interactive Brokers (IBKR) — американська транснаціональна біржова брокерська фірма. Управляє найбільшою електронною торговою платформою в США за кількістю угод із середньодобовим доходом. Ключові характеристики:

- **Глобальний доступ:** володіє широким глобальним покриттям і надає доступ до багатьох світових фондових бірж та ринків;
- **Широкий вибір інструментів:** Платформа пропонує інвестиції в різноманітні фінансові інструменти, включаючи акції, облігації, опціони, фьючерси та інші;
- **Професійні функції:** IBKR славиться своєю потужною та професійною торговельною платформою, яка надає широкі аналітичні та технічні інструменти;
- **Мінімальні комісії:** Interactive Brokers може бути особливо привабливим через свої низькі комісії для торгівлі.

Freedom Broker є однією з міжнародних брокерських компаній, яка працює в Україні та надає доступ для українських інвесторів до світових фондових ринків. Freedom Broker пропонує можливість торгівлі акціями, облігаціями, фондами та іншими інвестиційними інструментами на різних біржах.

Основні аспекти, які надає Freedom Broker:

- **Доступ до світових ринків:** Забезпечення можливості торгівлі на різних фондових біржах у США, Європі, Азії та інших регіонах.

- **Онлайн-платформа:** Надання інвесторам можливості користуватися онлайн-платформою для виконання торгівельних операцій та ведення портфеля.
- **Доступність інвестиційних продуктів:** Пропозиція різних інвестиційних продуктів, включаючи акції, облігації, етф та інші.
- **Зручність та обслуговування клієнтів:** Забезпечення зручності та обслуговування клієнтів через підтримку, освіту та інші послуги.

1.1.5 Якими інвестиційними інструментами користуються інвестори?

Українські роздрібні інвестори користуються різноманітними інвестиційними інструментами в залежності від своїх фінансових цілей, ризикового профілю та інвестиційного досвіду. Ось деякі з популярних інвестиційних інструментів, якими можуть користуватися українські роздрібні інвестори:

- **Акції (акції):** Інвестори можуть купувати частки компаній на фондових ринках, отримуючи прибуток від зростання вартості акцій та дивідендів.
- **Облігації:** Інвестування в облігації надає можливість отримувати фіксований дохід від відсоткових виплат, які робляться емітентом.
- **Інвестиційні фонди:** Це колективні інвестиції, де кошти різних інвесторів об'єднуються для інвестування в портфель цінних паперів.
- **ETF (біржові фонди):** Торгові фонди, які відтворюють рух ринку та можуть бути куплені та продані на фондовій біржі.
- **Криптовалюта:** Деякі інвестори обирають інвестувати в криптовалюту, таку як Bitcoin, Ethereum та інші, як альтернативну форму активів.
- **Фізичне золото:** Інвестори можуть володіти золотом у фізичній формі, купуючи монети чи слитки.
- **Валютний ринок:** Деякі інвестори можуть займатися торгівлею на валютному ринку, спрогнозовуючи зміни в обмінних курсах.

- **Нерухомість:** Інвестування в нерухомість може включати купівлю житла, комерційних об'єктів або участі в фондах нерухомості.

Ці інвестиційні інструменти можуть бути використані окремо або в комбінації для створення диверсифікованого портфеля з метою забезпечення стабільності та доходності. Однак перед тим, як здійснювати інвестиції, важливо оцінювати свої фінансові цілі, ризикотерпіння та ретельно досліджувати кожен конкретний інвестиційний інструмент.

В підсумок до розділу хочу виділити, що тема інвестицій в Україні популярна, але реалізується на зарубіжних ринках та платформах, а саме InteractiveBrokers та Freedom Broker. Інвестори вміють підбирати активи відповідно до своїх цілей та стратегій, але питання, яке не є широко-висвітленим – це **оподаткування доходу** з інвестиційної діяльності, а цей процес **є ключовим з точки зору захисту та легалізації свого прибутку**.

1.2 Законодавча податкова база

Як згадувалося раніше, основним органом який визначає основні правила оподаткування – **Державна податкова служба України**. Відіграє важливу роль у зборі податків та визначенні податкових ставок для різних видів інвестицій та фінансових операцій.

Податковий кодекс України — основний документ, кодифікований закон України, який регулює відносини, що виникають у сфері справляння податків і зборів.

В Податковому кодексі вперше в Україні здійснено об'єднання в одному нормативно-правовому акті норм права, якими врегульовано відносини в сфері оподаткування.

Податковий кодекс України зокрема визначає:

- вичерпний перелік податків та зборів, що справляються в Україні, та порядок їх адміністрування;

- платників податків та зборів, їх права та обов'язки;
- компетенцію контролюючих органів, повноваження і обов'язки їх посадових осіб під час здійснення податкового контролю;
- відповідальність за порушення податкового законодавства.

Після прийняття Податкового кодексу в Україні все ще діють 135 платежів (податків, зборів та ін.), що є найгіршим показником у світі. І що не дає змогу розвиватися економіці країни.

1.2.1 Аналіз правил обчислення інвестиційного прибутку

Розглянемо Податковий Кодекс України (далі – ПКУ) щодо визначення правил обчислення інвестиційного прибутку. З цим питанням мені допоміг інвестиційний консультант Твердохліб Сергій, з яким я співпрацюю в створенні веб-платформи з можливістю формування податкової звітності з інвестиційної діяльності в Україні.

Відповідно до пп. 14.1.811 статті 14 ПКУ, інвестиційний прибуток для розділу IV ПКУ (цей розділ регулює порядок та правила оподаткування доходів фізичних осіб) – це дохід у вигляді позитивної різниці між доходом, отриманим платником податку від проведення операцій з цінними паперами з урахуванням курсової різниці, деривативами та корпоративними правами, випущеними в інших, ніж цінні папери, формах, та витратами на придбання таких інвестиційних активів.

Аналогічне визначення інвестиційного прибутку міститься у розділу IV ПКУ, відповідно до п.170.2.2 статті 172 якого, інвестиційний прибуток розраховується як позитивна різниця між доходом, отриманим платником податку від продажу окремого інвестиційного активу з урахуванням курсової різниці (за наявності), та його вартістю, що визначається із суми документально підтверджених витрат на придбання такого активу або вартістю інвестиційного активу, що була задекларована особою як об’єкт декларування у порядку одноразового (спеціального) добровільного декларування

відповідно до підрозділу 94 розділу XX ПКУ з урахуванням норм підпунктів 170.2.4-170.2.6 цього пункту (крім операцій з деривативами).

При цьому, відповідно до п.170.2.3 ПКУ, якщо в результаті розрахунку інвестиційного прибутку за правилами, встановленими цією статтею, виникає від'ємне значення, воно вважається інвестиційним збитком.

Відповідно до пп. 170.2.1 ПКУ, облік загального фінансового результату операцій з інвестиційними активами ведеться платником податку самостійно, окремо від інших доходів і витрат. Для цілей оподаткування інвестиційного прибутку звітним періодом вважається календарний рік, за результатами якого платник податку зобов'язаний подати річну податкову декларацію, в якій має відобразити загальний фінансовий результат (інвестиційний прибуток або інвестиційний збиток), отриманий протягом такого звітного року.

Згідно положень п. 170.2.6 ПКУ, до складу загального річного оподатковуваного доходу платника податку включається позитивне значення загального фінансового результату операцій з інвестиційними активами за наслідками такого звітного (податкового) року. Загальний фінансовий результат операцій з інвестиційними активами визначається як сума інвестиційних прибутків, отриманих платником податку протягом звітного (податкового) року, зменшена на суму інвестиційних збитків, понесених платником податку протягом такого року.

Якщо загальний фінансовий результат операцій з інвестиційними активами має від'ємне значення, його сума переноситься у зменшення загального фінансового результату операцій з інвестиційними активами наступних років до його повного погашення.

Важливо, що платник податку – фізична особа визначає фінансовий результат за операціями з цінними паперами чи деривативами, що перебувають в обігу на фондовій біржі, окремо від фінансового результату за операціями з цінними паперами чи деривативами, що не перебувають в обігу на фондовій біржі (абзац 4 п.170.2.6 ПКУ).

Важливим необхідно відмітити правила щодо перерахування у гривню доходів та витрат, отриманих/понесених платником податку – фізичною особою в іноземній валюті.

Відповідно до положень п. 164.4 статті 164 розділу IV ПКУ, під час нарахування (отримання) доходів, отриманих у вигляді валютних цінностей або інших активів (вартість яких виражена в іноземній валюті або міжнародних розрахункових одиницях), такі доходи перераховуються у гривні за валютним курсом Національного банку України, що діє на момент нарахування (отримання) таких доходів.

Розділ IV ПКУ не містить прямої норми щодо порядку перерахування у гривню понесених витрат у іноземній валюті. На нашу думку, при діючій редакції ПКУ щодо порядку та правил обчислення інвестиційного прибутку фізичної особи, перерахунок понесених витрат у гривню повинен відбуватися за цими самими правилами, як і перерахунок доходів платника податку – фізичної особи. Такий однаковий підхід, на наш погляд, є справедливим та єдиним можливим для платника податку – фізичної особи, який в момент понесення витрат у іноземній валюті фіксує витрати і у гривні.

За словами Сергія, аналогічна позиція була висловлена ДПС України у Індивідуальній податковій консультації (ІПК) від. Зокрема, податковий орган у цій ІПК висловив думку, що якщо операції з інвестиційними активами здійснюються в іноземній валюті, то з метою визначення доходу у вигляді інвестиційного прибутку сума витрат визначається за офіційним курсом Національного банку України на дату понесення таких витрат, а сума доходу, отримана від продажу інвестиційного активу визначається за офіційним курсом Національного банку України на дату продажу такого активу.

Щодо врахування комісійної винагороди торговця цінним паперами при розрахунку інвестиційного прибутку/збитку.

У ПКУ відсутня пряма норма щодо врахування у витратах винагороди (комісії) торговця цінними паперами при розрахунку інвестиційного прибутку платника податку – фізичної особи. Проте, з досвіду Сергія, опосередковано про можливість

врахування таких витрат зазначено в самому визначенні інвестиційного прибутку, як доходу у вигляді позитивної різниці між доходом та витратами на придбання таких інвестиційних активів.

Безумовно, до витрат фізичної особи належать винагороди/комісії торговця, адже на фондовій біржі не є можливим придбання цінних паперів без участі відповідної професійної особи.

Крім того, можливість врахування таких витрат при розрахунку інвестиційного прибутку фізичної особи прямо передбачена Методикою визначення інвестиційного прибутку професійним торговцем цінними паперами при виконанні функцій податкового агенту, затвердженою Наказом Мінфіну від 22.11.2011р. № 1484 (далі – Методика). Ця Методика використовується банками та іншими установами при визначенні інвестиційного прибутку/збитку фізичних осіб, для яких такі установи є податковими агентами. Так відповідно до п. 3.10 розділу II Методики:

3.10. Доходи за операціями з цінними паперами зменшуються на суму додаткових фактично сплачених витрат, що відповідно до законодавства здійснюються у зв’язку з отриманням такого доходу, а саме:

- винагорода (комісія) торговцю цінними паперами;
- біржовий збір відповідно до тарифів, затверджених фондовою біржою.

Щодо документального підтвердження понесених витрат платника податку – фізичної особи, ПКУ прямо передбачено, що таким підтвердженням (первинним документом) доходів та витрат за операціями з інвестиційними активами, укладеними в електронній формі на фондовій біржі для клієнтів - учасників фондової біржі, визнається звіт торговця цінними паперами (брокера), який формується на базі біржового звіту та договору на брокерське обслуговування (п.170.2.2. ПКУ).

Щодо звільнення від оподаткування інвестиційного прибутку платника податку – фізичної особи. Відповідно до пп. «а» пп. 170.2.8 ПКУ: 170.2.8. Не підлягає оподаткуванню та не включається до загального річного оподатковуваного доходу:

а) дохід, отриманий платником податку протягом звітного податкового року від продажу інвестиційних активів, якщо сума такого доходу не перевищує суму, визначену в абзаці першому підпункту 169.4.1 пункту 169.4 статті 169 цього Кодексу.

Тобто в разі отримання фізичною особою інвестиційного прибутку у встановленому розмірі (наприклад, за 2022 рік у розмірі, що не перевищує 3470 грн.), такий прибуток не підлягає оподаткуванню ПДФО та ВЗ.

1.2.2 Автоматичний обмін даними

У серпні 2022 року Україна приєдналася до Багатосторонньої угоди компетентних органів про автоматичний обмін інформацією про фінансові рахунки (далі - Багатостороння угода CRS). Обов'язковою умовою якої є впровадження у національне законодавство загального стандарту звітності та належної перевірки інформації про фінансові рахунки, схваленого ОЕСР (далі - Стандарт CRS).

Прийнятий Закон про внесення змін до Податкового кодексу України щодо імплементації міжнародного стандарту автоматичного обміну інформацією про фінансові рахунки (проєкт №8131) розроблено для впровадження Стандарту CRS і приєднання України до міжнародної системи обміну інформацією. Це дозволить податковим органам України щорічно отримувати визначений масив даних в уніфікованому форматі щодо тих іноземних рахунків, які належать податковим резидентом України, або які належать організаціям, що перебувають під контролем резидентів України. У відповідь Україна надаватиме аналогічну інформацію податковим органам країнам-партнерів відносно рахунків їх податкових резидентів. В зв'язку із прийняттям закону, **з 1 липня 2023 року всі фінансові рахунки почнуть перевірятись для виявлення підзвітних рахунків, тобто тих, що підлягають автоматичному обміну**. Це буде стосуватись як вже відкритих станом на 1 липня 2023 року рахунків, так і тих, що будуть відкриті після цієї дати. **Нові рахунки та рахунки фізосіб з високою вартістю** (з залишком або вартістю

понад 1 млн доларів США станом на 30 червня 2023) будуть перевірені до 31.12.2023 року.

Рахунки фізосіб з низькою вартістю (з залишком або вартістю, що не перевищує 1 млн доларів США станом на 30 червня 2023) та рахунки організацій з залишком або вартістю, що перевищує 250 тис. доларів США станом на 30 червня 2023 – будуть перевірені до 31.12.2024 року.
Відомості про підзвітний рахунок фінансова установа зобов'язана включити до звіту та подати цей звіт до ДПС:

- до 1 липня 2024 року - по новим рахункам та рахунки фізосіб з високою вартістю;
- до 1 липня 2025 року - по рахункам фізосіб з низькою вартістю та рахункам організацій з залишком або вартістю, що перевищує 250 тис. доларів США станом на 30 червня 2023 року.

Автоматичний обмін інформацією про фінансові рахунки (обмін між країнами) здійснюється за резидентством клієнта. Тобто, фінансова установа перевіряє, як податковий статус клієнта – фізичної особи, так і податковий статус кінцевих бенефіціарних власників (контролюючих осіб) організації (юридичної особи, партнерства, траста чи іншого правового утворення).
Очікувана дата першого обміну з іншими країнами:

- 30 вересня 2024 року – по новим рахункам та рахунки фізосіб з високою вартістю;
- 30 вересня 2025 року –ß по рахункам фізосіб з низькою вартістю та рахункам організацій з залишком або вартістю, що перевищує 250 тис. доларів США станом на 30 червня 2023 року.

Тобто, вже **після 30 вересня 2024 року Україна почне отримувати інформацію про підзвітні рахунки своїх резидентів за кордоном, а відповідні країни інформацію про українські рахунки своїх резидентів**.

Такий обмін розкриє наявність та розмір доходів фізосіб для подальшого їх оподаткування в країні їх резиденції. Зокрема, резиденти України не зможуть приховати об'єкт оподаткування за кордоном.

1.3 Процес ведення податкової звітності

1.3.1 Документація, податкові ставки та штрафні санкції

Громадяни України, серед яких і числяться інвестори на фондовій біржі, зобов’язані декларувати свої доходи відповідно до Податкового Кодексу України.

Ці доходи декларуються через подання декларації під назвою «Податкової декларація про майновий стан і доходи» (код документа – F0100213). Для інвесторів, що торгують цінними паперами на фондовому ринку обов’язковим є додаток до декларації, а саме «РОЗРАХУНОК податкових зобов'язань з податку на доходи фізичних осіб та військового збору з доходів, отриманих від операцій з інвестиційними активами» (код документа – F0121213). Також цей додаток відомий, як «Форма Ф1» (далі так позначатиметься).

Декларація подається щороку незалежно від того чи інвестор здійснював ввід чи вивід коштів з рахунку. Термін подачі декларації починається з першого січня і закінчується тридцятого квітня щороку. В цей термін подається декларація за попередній календарний рік.

Важливо зазначити, що штрафна санкція за неподачу декларації за рік 340 грн. За другу прострочку 1020 грн і за кожну наступну також 1020 грн.

Також штрафна санкція нараховується +10% від доходу, якщо прострочка до 30 днів. Якщо більше, то 20%.

В цілому, існують наступні збори:

- Податок на доходи фізичних осіб (далі – ПДФО);
- Військовий збір.

Ці збори поширюються на наступні види інвестиційного доходів: з купівлі-продажу цінних паперів та дивідендних нарахувань. Вищезгадані збори мають наступні ставки:

- 18 % – ПДФО на дохід з купівлі-продажу цінних паперів;
- 9 % – ПДФО на дивіденди (іноземних компаній емітентів);
- 1,5 % – Військовий збір (для обох видів інвестиційного доходу).

Пропоную детальніше розібрати види інвестиційних доходів і ставками оподаткування. Під ставку 18 % + 1,5 % підпадають:

- Продаж акцій;
- Продаж ETF;
- Продаж/погашення облігацій (окрім державних українських);
- Опціони;
- Нарахування на депозитних рахунках;
- Нарахування за реферальними програмами;
- Списання комісій балами.

З дивідендами все простіше, застосовуємо вищезгадану ставку ПДФО 9% та військового збору 1,5 %.

Важливо зазначити, що збитки також декларуються. І якщо такі були наявні в попередньому періоді, то вони переносяться в поточний і впливають на фінансовий результат, а в результаті на податок.

Виходячи із законодавства згаданого вище, податковий розрахунок з операцій купівлі/продажу цінних паперів виконується за принципом від суми продажу активу(дохід) віднімаємо взяті разом суму купівлі, комісію купівлі та продажу(витрати) і отримуємо фінансовий результат.

Дивідендні нарахування сумують і до них уже застосується податкова ставка, все просто.

Проте для операцій над цінними паперами ще застосовується принцип FIFO (з англійської – «First In First Out», переклад – «перший увійшов, перший вийшов»). Цей принцип розглянемо детальніше у наступному підрозділі.

1.3.2 FIFO принцип

Першим прийшов, першим вийшов, широко відомий як FIFO, — це метод управління активами та оцінки, за якого активи, вироблені або придбані першими, продаються, використовуються або вибувають першими.

Для цілей оподаткування FIFO припускає, що активи з найдавнішими витратами включаються до собівартості проданих товарів (COGS) у звіті про прибутки та збитки. Решта інвентарних активів зіставляється з активами, які нещодавно були придбані або вироблені.

Ключові моменти:

- Першим надходить, першим виходить (FIFO) — це метод обліку, за яким активи, придбані або придбані першими, вибувають першими.
- FIFO припускає, що решта запасів складається з предметів, придбаних останніми.
- Альтернативою FIFO, LIFO є метод обліку, за яким активи, придбані або придбані останніми, вибувають у першу чергу.
- Часто на інфляційному ринку до собівартості товарів, проданих за методом FIFO, відносять нижчі, старі витрати, що призводить до вищого чистого доходу, ніж у випадку використання LIFO.

Метод FIFO використовується для припущення потоку витрат. У виробництві, коли товари просуваються до більш пізніх стадій розробки та продаються готові товарні запаси, пов’язані з цим продуктом витрати повинні визнаватися як витрати.

Відповідно до FIFO передбачається, що спочатку буде визнано вартість запасів, придбаних першими. Доларова вартість загальних запасів у цьому процесі

зменшується, оскільки запаси вилучено з власності компанії. Витрати, пов'язані з інвентаризацією, можна розрахувати декількома способами, одним з яких є метод FIFO.

Типові економічні ситуації включають інфляційні ринки та зростання цін. У цій ситуації, якщо FIFO призначає найдавніші витрати до собівартості проданих товарів, ці найдавніші витрати теоретично будуть оцінюватися нижче, ніж найновіші запаси, придбані за поточними завищеними цінами.

Ці менші витрати призводять до більшого чистого прибутку. Крім того, оскільки найновіший запас було придбано за загалом вищими цінами, кінцевий баланс запасів є завищеним.

Розглянемо принцип FIFO на прикладі. Запасам призначаються витрати, коли товари готуються до продажу. Це може відбутися через закупівлю запасів або витрати на виробництво, закупівлю матеріалів і використання праці. Ці призначені витрати базуються на порядку використання продукту, а для FIFO вони базуються на тому, що надійшло першим.

Уявіть собі, що компанія придбала 100 товарів по 10 доларів кожен, а потім придбала ще 100 товарів по 15 доларів кожен. Тоді компанія продала 60 найменувань. Згідно з методом FIFO, собівартість проданих товарів для кожного з 60 товарів становить 10 доларів США за одиницю, оскільки перші придбані товари є першими проданими товарами. Зі 140 предметів, що залишилися в інвентарі, вартість 40 предметів становить 10 доларів США за одиницю, а вартість 100 товарів – 15 доларів США за одиницю. Це пояснюється тим, що запасам призначається остання вартість за методом FIFO.

Маючи цей запас у 140 одиниць, скажімо, компанія продає ще 50 одиниць. Собівартість товарів, проданих для 40 із цих товарів, становить 10 доларів США, і все перше замовлення зі 100 одиниць було повністю продано. Інші 10 одиниць, які продаються, мають вартість 15 доларів США кожна, а решта 90 одиниць в інвентарі оцінюються в 15 доларів США кожна (остання сплачена ціна).

Метод FIFO дотримується логіки, згідно з якою, щоб уникнути морального старіння, компанія повинна спочатку продати найстаріші товарні запаси та зберегти найновіші товарні запаси. Хоча використаний метод оцінки фактичних запасів не обов'язково слідує фактичному потоку запасів через компанію, суб'єкт господарювання повинен мати можливість підтвердити, чому він вибрав використання певного методу оцінки запасів.

Такий принцип застосовується і при розрахунку інвестиційного прибутку для цілей оподаткування в Україні. Проте в більшості випадків, процес розподілу угод по FIFO, забирає доволі багато часу, бо інвестор не завжди купляє і продає рівні кількості цінних паперів, тому це накладає додаткової складності в розрахунку інвестиційного прибутку та податків.

1.3.3 Як це інвестори роблять зараз?

Загальна послідовність дій для подання податкової декларації наступна:

1. Завантажити брокерський звіт;
2. Перевести всі операції в національну валюту;
3. Розподілити угоди за принципом FIFO.
4. Виконати розрахунок інвестиційного доходу;
5. Створити ключ КЕП(кваліфікаційний електронний ключ);
6. Заповнити декларацію «Про майновий стан і доходи» та Форму Ф1;
7. Доповнити супровідним листом;
8. Підписати і відправити.

Після цих кроків інвестору залишається тільки оплатити податок і перевірити зарахування в податковому кабінету (cabinet.tax.gov). Проте не все так просто, насправді є дуже багато можливостей допустити помилку.

Процес переводу усіх операцій в національну валюту потребує отримати історичні курси валют НБУ по кожному дню, коли здійснювалися операції. За словами Сергія Твердохліба, у клієнтів, що до нього звертались за послугою розрахунку

інвестиційного прибутку та формування податкової звітності, – середнє число угод купівлі/продажу за рік 90. В угодах, наприклад, необхідно перевести вартість та комісії. Тобто це уже 180 перетворень. А ще є дивідендні нарахування, купонні виплати по облігаціям.

Необхідність розподіляти операції по принципу FIFO добавляє ще вирахування коефіцієнтів і знову ж таки перерахунок вартості та комісії, яка увійде той чи інший трейд, бо як ми згадували раніше, інвестор може придбати 5 акцій, а продати 3. Трейд – це операція, яка складається з двох угод купівлі і продажу. Тому у нього буде трейд купівлею і продажем 3-ох акцій і позитивний залишок на балансі 2 акції. Щоб порахувати прибуток по цьому трейду інвестору знадобиться розділити угоду купівлі 5 акцій на 3 і 2 штуки. При цьому потрібно взяти 0,6 від вартості та комісії для першої угоди і 0,4 для другої відповідно. Як бачимо процес доволі рутинний і вимагає дуже великої уважності.

Звісно інвестори не роблять це на папері вручну. Кожен інвестор, який професійно займається торгівлею, веде облік своїх активів на операцій над ними і для цього в більшості випадків застосовує електронні таблиці, наприклад, Microsoft Excel. Проте все одно інвестор сам повинен вносити вхідні дані про операції, курси і переносити результати розрахунків в податкову декларацію. Існує спосіб автоматизації отримання курсів валют НБУ, але ним володіє меншість.

Такий спосіб інвестору майже нічого не коштує з точки зору грошей, окрім ліцензії на ПО електронних таблиць.

Є можливість делегувати цей процес на професіоналів, а саме на:

- бухгалтера,
- аудитора,
- податкові компанії.

Відповідно затрат часу та енергії менше, але й вартість більша. В середньому, зі слів Сергія Твердохліба, бухгалтера просять за розрахунок інвестиційного прибутку від 11000 грн. Аудитори коштують від 2000 до 11000 грн за годинну консультацію.

Послуги податкових компаній стартують від 60000 грн. До цих компаній входять «Велика четвірка» (PricewaterhouseCoopers, Deloitte (Делойт), Ernst & Young (Ернст енд Янг) та KPMG) та Baker tilly.

Проте всюди присутній людський фактор і можливість допустити помилку, а як відомо, податкові інспектори повертали декларації де присутнє відхилення на одну копійку.

1.4 Висновки до розділу

Тема інвестицій у фондовий ринок актуальна та продовжує розвиватися, а за нею послуги обліку, аналітики. Процес оподаткування є невід'ємною частиною і диктується законом України, а саме її Податковим Кодексом. Інвестор, який декларує дохід і сплачує податки:

- захищає свій дохід;
- підтримує країну фінансово через наповнення бюджету;
- дотримується європейських цінностей;

З 2024 року інвестору не вдасться приховати свої закордонні рахунки, бо у серпні 2022 року Україна приєдналася до Багатосторонньої угоди компетентних органів про автоматичний обмін інформацією про фінансові рахунки. Інвестор, що ухиляється від сплати податків отримує наступне:

- його дохід не захищений і нелегалізований;
- штрафні санкції за несвоєчасну подачу декларації;
- пеню з доходу у разі несплати податку.

Процес розрахунку прибутку і декларування непростий, рутинний і доволі затратний зі сторони часу. Таким чином розробка програмного рішення для автоматизації процесу формування податкового звіту є актуальною, бо:

- зменшить кількість помилок в розрахунках до нуля;
- зменшить час на розрахунок від 1-3 днів до 5 хвилин;

2 ОГЛЯД ТЕХНОЛОГІЙ JAVA ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ

2.1 Мова програмування Java

Java — це широко використовувана об’єктно-орієнтована мова програмування та програмна платформа, яка працює на мільярдах пристроїв, включаючи ноутбуки, мобільні пристрої, ігрові консолі, медичні пристрої та багато інших. Правила і синтаксис Java засновані на мовах C і C++.

Однією з основних переваг розробки програмного забезпечення за допомогою Java є його переносимість. Після того, як ви написали код для програми на Java на ноутбуці, його дуже легко перемістити на мобільний пристрій. Коли в 1991 році Джеймсом Гослінгом із Sun Microsystems (пізніше придбаної Oracle) цю мову винайшов, головною метою була можливість «написати один раз і запустити будь-де».

Також важливо розуміти, що Java значно відрізняється від JavaScript. Javascript не потрібно компілювати, тоді як код Java потрібно компілювати. Крім того, Javascript працює лише у веб-браузерах, тоді як Java можна запускати будь-де.

Нові та вдосконалені інструменти розробки програмного забезпечення надходять на ринок із надзвичайною швидкістю, витісняючи існуючі продукти, які раніше вважалися незамінними. У світлі цього постійного обороту довговічність Java вражає; Більш ніж через два десятиліття після свого створення Java все ще залишається найпопулярнішою мовою для розробки програмного забезпечення — розробники продовжують обирати її замість таких мов, як Python, Ruby, PHP, Swift, C++ та інших. Як результат, Java залишається важливою вимогою для конкуренції на ринку праці.

Перш ніж досліджувати причини незмінної популярності Java, давайте детальніше розглянемо, що таке Java і її важливість для розробки корпоративних програм.

Java — це технологія, що складається як з мови програмування, так і з програмної платформи. Щоб створити програму за допомогою Java, вам потрібно завантажити Java Development Kit (JDK), який доступний для Windows, macOS і Linux. Ви пишете програму на мові програмування Java, потім компілятор перетворює програму на байт-код Java — набір інструкцій для віртуальної машини Java (JVM), яка є частиною середовища виконання Java (JRE). Байт-код Java працює без змін у будь-якій системі, яка підтримує JVM, що дозволяє виконувати ваш код Java будь-де.

Програмна платформа Java складається з JVM, Java API і повного середовища розробки. JVM аналізує та запускає (інтерпретує) байт-код Java. Java API складається з великого набору бібліотек, включаючи базові об'єкти, функції мережі та безпеки; Генерація розширюваної мови розмітки (XML); і веб-сервіси. Разом мова Java і програмна платформа Java створюють потужну, перевірену технологію для розробки корпоративного програмного забезпечення.

Якщо ви розробник корпоративних програм, ви вже знаєте, що таке Java, і цілком імовірно, що у вашій організації вже є тисячі, навіть мільйони рядків робочого коду, написаних на Java. Ймовірно, вам знадобиться певний рівень знань Java, щоб ви могли виправляти неполадки, підтримувати та оновлювати існуючу кодову базу.

Однак було б помилкою розглядати Java лише з точки зору застарілих програм. Мова Java є серцевиною операційної системи Android, яка забезпечує найбільшу кількість смартфонів у світі. Java також є однією з найпопулярніших мов для програм машинного навчання та науки про дані. Його надійність, простота використання, кросплатформні можливості та безпека роблять Java мовою вибору для інтернет-рішень у багатьох корпоративних магазинах.

Зокрема, технологія Java є ідеальною структурою для розробки веб-додатків, основою для цифрового бізнесу в будь-якій галузі. Сервери додатків Java — це веб-контейнери для компонентів Java, XML і веб-служб, які взаємодіють із базами даних і надають динамічний веб-вміст. Сервери додатків Java утворюють стабільне середовище розгортання для корпоративних додатків із такими можливостями, як

керування транзакціями, безпека, кластеризація, продуктивність, доступність, підключення та масштабованість.

Коли справа доходить до вибору мови програмування та середовища для вашої наступної корпоративної програми, є вагомі технічні причини розглянути Java, включаючи взаємодію, масштабованість і адаптивність.

Основна філософія, яка лежить в основі його створення, — взаємодія між різними пристроями — залишається найвагомішим аргументом на користь Java для нових корпоративних програм. Об’єктно-орієнтована архітектура Java дозволяє створювати модульні програми та багаторазовий код, скорочуючи цикли розробки та подовжуючи довговічність корпоративних програм.

Масштабованість платформи є ключовим атрибутом Java. З Java ви можете використовувати одну єдину систему в широкому діапазоні випадків використання. Існуючі настільні програми можна легко адаптувати для роботи на невеликих пристроях з обмеженими ресурсами. Ви також можете перенести програми з мобільних пристроїв на настільні, розробляючи бізнес-програми для платформи Android, а потім інтегруючи їх у поточне програмне забезпечення для настільних ПК, минаючи тривалі та дорогі цикли розробки.

Java також отримує бали зі стратегічного планування за свою здатність адаптуватися до нових випадків використання. Наприклад, Java вважається ідеальною платформою для Інтернету речей (IoT). Типова програма IoT об’єднує велику кількість різних пристроїв, завдання, яке значно спрощується тим, що мільярди пристроїв працюють на Java. Крім того, широка екосистема розробників Java постійно розробляє та ділиться новими бібліотеками з функціями, спеціально націленими на розробку додатків IoT.

Технічні аргументи на користь Java є переконливими, але бізнес-причини для вибору Java не менш вагомі: великий резерв талантів, короткий період навчання та широкий спектр інтегрованих середовищ розробки (IDE).

Оскільки все більше компаній використовують підключені пристрої, алгоритми машинного навчання та хмарні рішення, попит на кваліфікованих розробників продовжує зростати. Багато аналітиків передбачають дефіцит програмістів старшого рівня в найближчому майбутньому, що ускладнить укомплектування нових програмних ініціатив. Попит на розробників мобільних додатків незабаром може легко перевищити наявну пропозицію.

Велика кількість талантів розробників Java є вагомою причиною для заснування основних програмних ініціатив на Java. Коли менеджери з персоналу публікують вакансії для Java-розробників, вони можуть очікувати отримання багатьох кваліфікованих резюме та зайняти ці посади відносно швидко. Менеджери також можуть використовувати контрактні ресурси, щоб доповнити внутрішній персонал для виконання конкретних завдань без збільшення штату.

Окрім розробників старшого рівня, великі програмні ініціативи також потребують великої кількості молодших учасників. Хоча Java залишається популярною початковою мовою програмування в університетських навчальних програмах з інформатики, багатьом випускникам не вистачає навичок, щоб бути продуктивними в перший день. Java легше вивчати та опанувати, ніж багато інших мов програмування, що призводить до коротшого навчання та швидшого підвищення продуктивності. Велика онлайн-спільнота Java, яка складається з форумів розробників, посібників і груп користувачів, допомагає початківцям швидко освоїтися, а досвідченим програмістам надає ефективні, перевірені інструменти для вирішення проблем.

У сфері інструментів програмування Java пропонує низку IDE. Досвідчені Java-розробники можуть швидко освоїти нове середовище, що дає менеджерам з розробки можливість вибрати IDE, яка найкраще відповідає типу проекту, бюджету, методології розробки та рівню навичок програміста. Багато досвідчених Java-програмістів вважають NetBeans, Eclipse і IntelliJ IDEA трьома найкращими IDE для розробки

корпоративних програм. Але є випадки, коли найкращим вибором є більш легка IDE, наприклад DrJava, BlueJ, JCreator або Eclipse.

Питання, яке хвилює багатьох розробників програмного забезпечення, полягає в наступному: що можна зробити за допомогою програмування на Java? Окрім розробки технологій великих даних та створення рішень IoT (Інтернет речей), Java допомагає створювати різні типи програм. Нижче ми описали топ-11 випадків використання мови Java.

Мобільні додатки Java є офіційною мовою програмування сучасних мобільних додатків. Здається, ця мова сумісна з такими програмними рішеннями, як Android Studio або Kotlin. Завдяки Java програмісти можуть створювати повні мобільні програми, які працюють на смартфонах або планшетах.

Веб-додатки. Крім мобільних додатків, Java широко використовується для створення веб-додатків. Крім того, він забезпечує ефективну підтримку таких програм за допомогою сервлетів, Struts і JSP. Завдяки певним технологіям розробники можуть створювати різні види веб-додатків, які їм потрібні.

Хмарні програми – це ті, які працюють з інформацією, обробленою на відповідному хмарному сервері. Сьогодні існують різні типи веб-сервісів, які використовують технології хмарних обчислень. І Java є однією з найкращих мов програмування для створення хмарних програм. Це тому, що він забезпечує високу продуктивність і масштабованість.

Чат-боти. Багато організацій використовують технологію ШІ (штучного інтелекту) для створення ефективних чат-ботів. Тим часом багато з цих чат-ботів також написані на Java. Вони допомагають вирішувати виникаючі питання та проблеми клієнтів і надають необхідну інформацію. Це робить чат-боти неймовірно корисним інструментом обслуговування клієнтів.

Ігри. Що можна створити за допомогою Java, крім згаданих рішень? Так, ігри. Програмування на Java є ідеальним варіантом для розробки ігор, особливо для

програм Android. Оскільки Java забезпечує високу продуктивність, вона забезпечує ефективну роботу таких програм у відповідних операційних системах.

Додатки для охорони здоров'я. Серед найвидатніших речей, які ви можете робити з Java, це створення програм для охорони здоров’я. Наприклад, послуги розробки на Java дозволяють вашій організації створювати програми для електронних медичних записів, які можуть перевіряти страховий статус пацієнтів. Крім того, мобільні додатки для охорони здоров’я можуть допомогти користувачам створювати, передавати та зберігати необхідні медичні дані.

Логістичні програми. Відповідаючи на питання «що можна робити з Java», варто згадати логістичні програми. За допомогою логістичних настільних додатків або відповідних мобільних версій клієнти можуть перевірити логістичний процес після бронювання своїх замовлень і відстежувати доставку. Крім того, застосування Java допомагає надати деякі важливі функції, такі як виставлення рахунків або клієнтська база даних.

Корпоративні програми. Java часто є найбільш підходящим варіантом для створення корпоративних програм. Він пропонує спеціальну платформу під назвою Java Enterprise Edition (JavaEE), яка забезпечує API та середовище виконання, необхідні для створення сценаріїв. Наприклад, JavaEE служить основою для різноманітних банківських програм.

Наукові додатки. Багато розробників програмного забезпечення вважають Java найкращим варіантом кодування різних математичних операцій і наукових розрахунків. Ці наукові програми завжди неймовірно безпечні та швидкі. Вони забезпечують вищий рівень мобільності та низькі витрати на обслуговування. Крім того, багато наукових програм використовують Java для створення інтерактивного інтерфейсу користувача.

Бекенд-обробка. Отже, для чого використовується Java, крім різних програм? Численні організації застосовують його для серверної обробки, особливо в завданнях

пакетної обробки в непікові години. Це допомагає зменшити вплив користувачів у звичайний робочий час.

Вбудовані системи. Використання мови програмування Java також є ефективним підходом до створення вбудованих систем. Вони служать комп'ютерними системами спеціального призначення, розробленими для виконання одного або кількох обмежених завдань. Такі системи відомі своїми мінімальними розмірами, енергоспоживанням і ціною.

2.2 Фреймворки для розробки підприємницьких додатків

Існує багато фреймворків для розробки підприємницьких додатків на основі Java. В цьому розділі розглянемо найпопулярніші та їх альтернативи.

У сучасному програмуванні на Java фреймворки займають особливе місце. Фреймворк – це набір готових бібліотек та інструментів, які спрощують та прискорюють розробку програмного забезпечення. Використання фреймворків дозволяє розробникам зосереджуватися на бізнес-логіці додатків, а не на створенні базової інфраструктури.

На нашому курсі Java програмування ми детально розглядаємо фреймворки Spring та Hibernate, а в цьому матеріалі зробимо їх короткий огляд.

Фреймворк Spring з’явився у 2002 році, як відповідь на складність розробки корпоративних додатків на Java. Spring розроблений командою, яку очолював Род Джонсон, що впослідстві заснував компанію SpringSource (потім була придбана компанією VMware).

Spring використовується в різних проектах, починаючи від невеликих веб-додатків до великих корпоративних систем. За останні роки він став одним з найпопулярніших фреймворків для розробки на Java. Spring використовується в компаніях різного рівня, від стартапів до відомих компаній, таких як Netflix, Spotify, і Amazon.

Основні модулі фреймворка Spring:

Spring Core: ядро фреймворка, яке включає в себе контейнер залежностей та забезпечує інверсію керування (IoC) та аспектно-орієнтоване програмування (AOP).

Spring MVC: модуль для побудови веб-додатків з використанням моделі MVC. Він включає в себе широкий спектр веб-компонентів, таких як контролери, перехоплювачі, валідатори та ін.

Spring Data: модуль, який надає шаблони для роботи з різними системами зберігання даних, такими як реляційні бази даних, NoSQL бази даних, кешування, і пошукові системи.

Spring Security: модуль для захисту вашого додатку на рівні аутентифікації та авторизації, забезпечує можливість реалізувати систему контролю доступу до ресурсів додатку.

Spring Boot: модуль, який спрощує процес налаштування та розгортання Spring-додатків, надаючи автоматичну конфігурацію та вбудовані залежності для швидкого старту проекту.

Spring Cloud: модуль, який допомагає розробникам працювати з різними хмарними сервісами та платформами. Він надає інструменти для налаштування, моніторингу, маршрутизації, автоматичного масштабування, безпеки та інших характеристик хмарно-орієнтованих додатків.

Spring Integration: модуль, який спрощує інтеграцію Spring-додатків з іншими системами та сервісами через абстракції на рівні коду та конфігурації.

Spring Batch: модуль, який забезпечує роботу з пакетною обробкою даних, дозволяючи ефективно обробляти великі обсяги даних та забезпечувати їхній обмін між різними системами.

Spring Test: модуль, який допомагає розробникам писати та виконувати юніт-тести для Spring-додатків, надаючи інструменти для тестування компонентів, конфігурації, і інтеграції з іншими системами.

Усі ці модулі можуть бути використані окремо або в комбінації з іншими, забезпечуючи гнучкість та розширюваність для розробки різноманітних додатків на основі Java.

Фреймворк Hibernate є одним з ключових гравців у світі Java. Він був створений для спрощення роботи з базами даних, використовуючи техніку Object-Relational Mapping (ORM).

Цей фреймворк було створено в 2001 році. З того часу він активно розвивається та вдосконалюється. Гавін Кінг та Крістіан Бауер є головними розробниками Hibernate. Вони створили фреймворк, що допомагає розробникам працювати з базами даних у набагато більш спрощеній формі.

Hibernate широко використовується в різноманітних проектах Java, включаючи веб-додатки, корпоративні системи та інші. Це сталося завдяки його гнучкості, продуктивності та здатності легко інтегруватися з іншими технологіями.

Основні модулі фреймворка Hibernate: Hibernate містить декілька модулів, зокрема:

- Hibernate ORM: Це основний модуль, який забезпечує взаємодію з базами даних через ORM.
- Hibernate Validator: Модуль для валідації даних на стороні сервера.
- Hibernate Search: Модуль для повнотекстового пошуку та індексації.
- Hibernate Envers: Модуль для аудиту змін у даних, що зберігаються в базі даних.
- Hibernate OGM: Модуль для роботи з NoSQL базами даних.

Hibernate є потужним фреймворком для розробки Java-додатків, який забезпечує широкі можливості та гнучкість при роботі з базами даних. Завдяки своїй модульній структурі, Hibernate дозволяє розробникам вибирати тільки ті компоненти, які потрібні для конкретного проекту.

Spring та Hibernate вже довгий час використовуються в реальних проектах, включаючи веб-додатки, мобільні додатки та корпоративні системи. Ці фреймворки

допомагають розробникам швидко створювати надійні та масштабовані рішення, враховуючи специфіку кожного проекту.

Нижче наведені кілька практичних прикладів використання Spring та Hibernate. Використання Spring Boot для створення веб-додатка:

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
   public static void main(String[] args) {
      SpringApplication.run(Application.class, args);
   }
}
```

Цей код ініціалізує Spring Boot додаток з автоматичною конфігурацією та стартом вбудованого сервера.

Використання Spring Data JPA для роботи з базою даних:

```
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository {
}
```

Тут Spring Data JPA автоматично створює реалізацію JpaRepository для класу User, надаючи CRUD-операції без необхідності писати код.

Використання Hibernate для збереження даних у базі даних:

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateExample {
```

```
    public static void main(String[] args) {
        SessionFactory sessionFactory = new Configuration()
                                        .configure()
                                        .addAnnotatedClass(User.class)
                                        .buildSessionFactory();

        try (Session session = sessionFactory.openSession()) {
            User newUser = new User("John", "Doe");
            session.beginTransaction();
            session.save(newUser);
            session.getTransaction().commit();
        } finally {
            sessionFactory.close();
        }
    }
}
```

У цьому прикладі коду використовується Hibernate для створення нового користувача та збереження його в базі даних із використанням об’єкту сесії.

Використання Spring MVC для створення веб-контролера:

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {
    @GetMapping("/")
    public String home(Model model) {
        model.addAttribute("message", "Welcome to our website!");
```

```
        return "home";
    }
}
```

У цьому прикладі використовується Spring MVC для створення контролера, який обробляє HTTP-запити і повертає відповідний веб-шаблон з даними.

Ці приклади демонструють, як Spring та Hibernate можуть допомогти розробникам швидко та ефективно реалізовувати різні аспекти програмного забезпечення. Важливо враховувати, що можливості цих фреймворків значно перевищують наведені приклади, і їх використання може знайти застосування в різних типах проектів.

Вибір фреймворка для розробки Java-додатків в значній мірі залежить від вимог та специфіки вашого проекту. Spring та Hibernate є відмінним вибором для багатьох сценаріїв розробки, проте інші фреймворки також можуть запропонувати цінні можливості та підходи.

В рамках нашого проекту ми будемо їх використовувати, бо вони дозволять легко підняти серверний застосунок з доступом через протокол HTTP та взаємодію даними з базою даних.

2.3 Детальніше про Spring Framework

Spring виник у 2002 році у вигляді полегшеного аналога платформи для корпоративних програм Java Enterprise. Він позиціонується як проста у використанні платформа для веб-застосунків. Spring підтримує кілька мов JVM: Java, Kotlin та Groovy.

Spring складається з великої кількості модулів. Серед них є модулі-стартери, без яких Spring-додаток просто не запуститься, а є також допоміжні проекти, які додають у додаток певну функціональність: наприклад, Spring Data Flow для потокової обробки даних, Security для безпеки або Cloud для розподілених систем. Така структура

дозволяє розробникам ефективно створювати та підтримувати програми, використовуючи лише потрібні інструменти.

Зупинимося докладніше на основних модулях Spring і дізнаємося, навіщо їх можна використовувати.

Декілька слів про Spring Framework. Spring Framework – ядро платформи Spring. Framework неявно використовується іншими компонентами Spring, наприклад, MVC і WebFlux, забезпечуючи фундаментальні інструменти для різних архітектур додатків. Ми не розглядатимемо його як окремий модуль Spring, але перерахуємо основні характеристики та особливості.

Spring Framework включає:

- Основні технології: використання залежностей, події, ресурси, i18n, перевірка, зв'язування даних, перетворення типів, SpEL, AOP.
- Інструменти тестування: mock-об'єкти, TestContext, Spring MVC Test, WebTestClient.
- Доступ до даних: транзакції, підтримка DAO, JDBC, ORM, Marshalling XML.
- Інтеграцію: дистанційна взаємодія, JMS, JCA, JMX, електронна пошта, завдання, планування, кеш.

Розглянемо модуль Spring Boot. Це комплексний фреймворк для створення та запуску додатків з мінімальними зусиллями та налаштуваннями. Цей модуль ділиться на два стеки: заснований на API сервлетів Spring MVC та реактивний Spring WebFlux.

Spring WebFlux — веб-платформа, створена, щоб максимально використовувати переваги сучасних багатоядерних процесорів і обробляти величезну кількість одночасних підключень.

Spring MVC побудований на API сервлетів та використовує архітектуру синхронного блокуючого вводу-виводу з моделлю «один запит на потік».

У Spring Boot можна також опціонально підключити бібліотеку Reactor для створення реактивних систем на JVM.

Характеристики:

- Вбудовування контейнерів Tomcat, Jetty або Undertow без розгортання WAR-файлів.
- Готові стартові залежності, що спрощують конфігурацію збирання.
- Можливість конфігурувати проект прямо у браузері за допомогою Spring Initializr.
- Автоматичне налаштування сторонніх бібліотек (по можливості).
- Готові до роботи функції, такі як збирання метрик, перевірка працездатності та використання зовнішньої конфігурації.
- Немає кодогенерації і не потрібна конфігурація XML – все конфігурується через анотації.

Для чого використовується:

- Разом із Spring Boot у проектах зазвичай використовуються Spring Security та Cloud. За допомогою Spring Boot можна створювати:
- мікросервіси;
- реактивні системи;
- веб-програми.

Spring Data, модуль забезпечує додаткам доступ до даних через реляційні та нереляційні бази даних (БД), map-reduce фреймворки та хмарні послуги. Spring Data містить безліч підпроектів, призначених для певних СУБД. Серед них є, наприклад, MySQL, MongoDB, Redis та багато інших. Також можна використовувати підмодулі, розроблені спільнотою Spring для більш специфічних баз даних на кшталт ArangoDB, Google Datastore, Microsoft Azure Cosmos DB та інших.

Основний механізм, що реалізується в Spring Data - репозиторій. Це набір інтерфейсів, що використовують JPA Entity для взаємодії даних.

Характеристики:

- Відображення сутностей, що настроюється, в БД на Java-об'єкти.

- Створення динамічних запитів до бази даних через сигнатуру методу інтерфейсу репозиторію.
- Базові класи для різноманітних завдань.
- Прозорий аудит об'єктів.
- Можливість інтегрувати власний код репозиторію.
- Проста інтеграція зі Spring через JavaConfig, а також кастомні простори імен XML.
- Розширена інтеграція із контролерами Spring MVC.

Spring Data використовується скрізь, де потрібний доступ до даних, і легко інтегрується з іншими модулями Spring.

Зі Spring Cloud ви зможете легко і швидко створювати шаблони в розподілених системах. З прикладів таких шаблонів: управління конфігурацією, виявлення сервісів, інтелектуальна маршрутизація, мікропроксі, одноразові токени та багато іншого.

Шаблони, створені за допомогою Spring Cloud, добре працюватимуть у будь-якому розподіленому середовищі, включаючи ваш власний ноутбук, центри обробки даних та PaaS-платформи, такі як Cloud Foundry.

Spring Cloud також складається з багатьох підпроектів для різних цілей. Так, Spring Cloud Azure інтегрує Spring зі службами Azure, Spring Cloud Stream використовується для створення керованих подіями мікросервісів (event-driven microservices) і таке інше.

Характеристики:

- Розподілена/версійна конфігурація.
- Реєстрація та виявлення сервісів.
- Маршрутизація.
- Зв'язок між сервісами (service-to-service calls).
- Балансування навантаження.
- Вибір лідера та стан кластера.

- Розподілений обмін повідомленнями.

Для чого використовується? – Spring Cloud містить багато корисних інструментів для мікросервісів та розподілених систем.

Spring Cloud Data Flow потрібен програмам, у яких використовується потокова передача та пакетна обробка даних.

Фреймворк підтримує низку готових кейсів обробки даних, серед яких ETL, потокова обробка подій та прогнозна аналітика.

Характеристики:

- Розгортання програм на платформах Cloud Foundry та Kubernetes.
- Готові програми для різних сценаріїв інтеграції та обробки даних.
- Програми, що настроюються, орієнтовані на сполучне ПЗ або служби даних.
- Простий потоковий конвеєр DSL, щоб вказати, які програми розгортати і як підключати виходи та входи.
- Графічний редактор для інтерактивної побудови конвеєрів даних та моніторингу метрик за допомогою Wavefront, Prometheus, Influx DB або інших систем.
- REST API для створення та розгортання конвеєрів даних з можливістю роботи з командного рядка.

Для чого використовується – Spring Cloud Data Flow підійде для створення конвеєрів потокової обробки даних - наприклад, щоб пересилати будь-які дані в базу і зручно аналізувати їх.

Spring Security – середовище автентифікації, авторизації та контролю доступу. Це стандартний фреймворк, який використовується для захисту програм на основі Spring.

Spring Security надає базові функції безпеки, які можна легко розширити для ваших потреб.

Характеристики:

- Аутентифікація та авторизація користувачів.
- Захист від атак, таких як фіксація сесії, клікджекінг, підробка міжсайтових запитів тощо.
- Можливість інтеграції із Servlet API.
- Модуль Spring Web MVC, що опціонально підключається.

Для чого використовується? Модуль Security потрібен, щоб забезпечити перевірку безпеки та захистити програму від атак.

Spring Integration дозволяє полегшити обмін повідомленнями у додатках на основі Spring, підтримує інтеграцію із зовнішніми системами та дає інструменти для обробки даних із різних джерел. Один з підпроектів Spring Cloud, Spring Cloud Stream, використовує Spring Integration як двигун для мікросервісів, керованих подіями.

Характеристики:

- Багато шаблонів для інтеграції додатків підприємства.
- Інтеграція із зовнішніми системами.
- Веб-сервіси (SOAP та REST).
- Широка підтримка JMX.
- MBeans-компоненти.

Спосіб використання наступни – Spring Integration підключається до проекту, якщо потрібно зв'язати POJO (Plain Old Java Object) за допомогою парадигми обміну повідомленнями без впровадження залежностей (DI). Також Integration дозволяє взаємодіяти із зовнішніми системами за допомогою адаптерів каналів та шлюзів. Адаптери каналів використовуються для односторонньої інтеграції (надсилання або отримання), а шлюзи - для сценаріїв запиту / відповіді (вхідного або вихідного).

Spring Batch – платформа для розробки пакетних програм. Spring Batch підійде як для простих, так і більш складних проектів — платформа легко масштабується і може обробляти великі обсяги інформації.

Характеристики:

- Управління транзакціями.

- Обробка з урахуванням фрагментів даних.
- Декларативне введення/висновок.
- Веб-інтерфейс адміністрування (Spring Cloud Data Flow).

Для чого застосовується? – Spring Batch підійде для програм з багаторазово використовуваними функціями, щоб обробляти великі обсяги записів. Серед таких функцій — ведення логів та трасування, управління транзакціями, статистика обробки завдань, перезапуск та пропуск завдань, управління ресурсами та інші.

Ми перерахували лише ключові модулі Spring. Насправді їх набагато більше: наприклад, є ще Spring для Android для створення Android-додатків, Spring CredHub для взаємодії з CredHub-сервером, Spring LDAP та багато інших.

З повним списком проектів Spring можна ознайомитись на офіційному сайті.

2.4 Використання технологій Java у фінансовій сфері

Фінтех-проекти вимагають надійного, безпечного та масштабованого технологічного стеку для обробки складних фінансових операцій і вимог до обробки даних.

Java є ключовою частиною багатьох успішних фінтех-рішень, пропонуючи унікальне поєднання надійності, масштабованості та продуктивності. Важливість вибору Java — або будь-якого іншого стеку технологій — виходить за рамки лише технічних атрибутів; він також відіграє важливу роль у бізнес-стратегії, управлінні ризиками та відповідності.

Отже, для чого саме використовується Java? Fintech (таке позначення у фінансовій сфері в англомовному світі) — це не лише фінансові операції; мова йде про їх виконання з точністю, швидкістю та безпекою, забезпечуючи дотримання різноманітних нормативних умов. Давайте дослідимо, як Java відповідає цим конкретним вимогам.

Дотримання регіональних і міжнародних норм є ключовим у діяльності Fintech. Ці правила стосуються конфіденційності даних, автентичності транзакцій, виявлення шахрайства тощо.

Java надає різноманітні інструменти, які полегшують моніторинг відповідності та звітування. Такі бібліотеки, як Bouncy Castle, пропонують криптографічні алгоритми для безпечного зберігання та передачі даних, тоді як фреймворки, такі як Apache Shiro, пропонують комплексні рішення безпеки, гарантуючи дотримання правил захисту даних.

Окрім захисту даних, ключовим аспектом відповідності нормативним вимогам у Fintech є відстеження та підзвітність. Java пропонує інструменти та фреймворки, які можуть допомогти у створенні надійних журналів аудиту, які є важливими для відстеження та перевірки транзакційної діяльності з метою відповідності.

Фреймворки, такі як Log4j або SLF4J, полегшують детальне журналювання дій програми. Однак важливо розуміти, що просте використання цих інструментів не гарантує дотримання вимог. Правильна конфігурація, керування журналами та додаткова обробка є обов’язковими для забезпечення відповідності цих журналів певним нормативним стандартам. Подібним чином, хоча Java може взаємодіяти з безліччю баз даних, які підтримують політики збереження даних, сама мова програмування безпосередньо не забезпечує відповідності збереження даних.

Замість цього він надає можливості, а розробники та організації покладають відповідальність за правильне впровадження та дотримання нормативних вимог. Ця всеохоплююча структура дає змогу додаткам Fintech не лише прагнути до відповідності, але й бути добре підготовленими до регуляторних запитів або аудитів.

Суть Fintech полягає в обробці грошових транзакцій, де навіть найменша помилка може мати серйозні наслідки.

Вбудовані функції Java, такі як автоматичне керування пам’яттю та збирання сміття, зменшують ризик збоїв системи або витоку пам’яті, забезпечуючи плавну обробку транзакцій. Крім того, екосистема Java має надійні інфраструктури, такі як

Java Transaction API (JTA) для керування та координації транзакцій, забезпечуючи цілісність даних.

Користувачі отримують доступ до програм Fintech з різних пристроїв, починаючи від мобільних телефонів і закінчуючи настільними комп'ютерами, і навіть спеціалізованого банківського обладнання.

Кросплатформна природа Java, укладена у філософії «Напишіть один раз, запустіть будь-де», означає, що програми Fintech, розроблені цією мовою програмування, забезпечують узгоджену роботу користувачів на різних пристроях, зменшуючи розбіжності та потенційне розчарування користувачів.

Сучасні фінтех-платформи – це більше, ніж просто центри транзакцій; вони служать джерелами розуміння, аналітики та прогнозних моделей як для користувачів, так і для компаній. Java надає надійний набір бібліотек, призначених для різноманітних аналітичних потреб. Такі бібліотеки, як Weka, є безцінними для таких завдань, як дослідження даних, моделювання та офлайн-аналіз.

Для більшої обробки даних у реальному часі у Fintech Java може похвалитися низкою інших інструментів і фреймворків. Крім того, завдяки таким бібліотекам, як JScience та Apache Commons Math, складні фінансові розрахунки стають безперебійними. Разом ці інструменти дозволяють платформам Fintech пропонувати розширені послуги, такі як фінансове прогнозування, моделювання ризиків і інвестиційні рекомендації.

Якщо коротко про вищесказане, то:

- Java надає інструменти та бібліотеки, які полегшують дотримання суворих нормативних вимог у секторі фінансових технологій.
- Його властиві функції та допоміжні структури забезпечують надійність обробки транзакцій.
- Кросплатформні можливості Java забезпечують постійну доступність і взаємодію з користувачем на різних пристроях.

- Завдяки багатому набору бібліотек Java платформи Fintech можуть запропонувати розширену аналітику, статистичні дані та фінансові обчислення.

Для чого використовується Java у Fintech? Визначність Java у цій галузі пояснюється її технічними характеристиками, які ідеально відповідають потребам фінансової галузі.

З моменту створення Java завоювала репутацію надійності та стійкості. У Fintech, де точність транзакцій і цілісність даних є найважливішими, така репутація суттєво впливає на рішення про впровадження технологій.

Окрім своєї репутації, Java оснащена багаторівневою системою безпеки. Такі функції, як перевірка байт-коду, перевірки безпеки під час виконання та механізм ізольованого програмного середовища Java гарантують, що додатки залишаються захищеними від потенційних загроз. Його безпека на основі інфраструктури відкритих ключів (PKI) забезпечує зашифровану передачу даних, необхідну для конфіденційності фінансових даних.

Сучасні фінтех-платформи часто відчувають спорадичні транзакційні навантаження. Технологія, що підтримує ці платформи, повинна динамічно адаптуватися до цих коливань без шкоди для оперативності.

Архітектура Java, що базується на віртуальній машині Java (JVM), дозволяє спрощено розподіляти ресурси, оптимізуючи використання ЦП і пам’яті. Крім того, він підтримує багатопотоковість, уможливлюючи одночасну обробку, що є життєво важливим для аналітики в реальному часі та обробки транзакцій у Fintech.

Традиційні монолітні додатки можуть стати громіздкими для оновлення та обслуговування, особливо в швидко мінливому середовищі, як Fintech. Мікросервіси пропонують більш модульний підхід, розбиваючи програми на менші керовані компоненти.

Фреймворки Java, зокрема Spring Boot і Micronaut, полегшують розробку мікросервісів, пропонуючи інструменти для виявлення сервісів, балансування

навантаження та розподіленого трасування. Ці можливості гарантують, що в міру розвитку додатків Fintech їх можна буде періодично оновлювати без масштабних збоїв у системі.

Крім основної мови, корисність будь-якої платформи програмування часто залежить від інструментів, бібліотек і фреймворків, які її підтримують.

Величезна екосистема Java включає такі бібліотеки, як JFreeChart для візуалізації фінансових даних, і такі інструменти, як QuantLib для фінансової математики. Такі фреймворки, як Hibernate, спрощують роботу з базами даних, необхідну для ведення записів транзакцій і аналізу даних у Fintech.

Додатки Fintech доступні з безлічі пристроїв і операційних систем. Незалежно від того, чи це мобільний додаток на смартфоні, веб-портал на настільному комп'ютері чи навіть кіоск у банку, додаток має функціонувати послідовно.

Принцип «Напишіть один раз, запустіть будь-де»: інтерпретація байт-коду Java за допомогою JVM гарантує, що той самий код виконується узгоджено на різних пристроях і операційних системах. Для фінтех-компаній, які прагнуть обслуговувати користувачів на різноманітних технологічних платформах, ця функція спрощує розгортання та забезпечує однакову взаємодію з користувачем.

Підсумовуючи маємо наступні переваги:

- Надійність і багатогранні функції безпеки Java роблять її надійним вибором у секторі фінансових технологій.
- Його здатність динамічно масштабуватися та обробляти одночасну обробку відповідає вимогам сучасних фінтех-платформ у режимі реального часу.
- Багатий набір фреймворків Java підтримує архітектуру мікросервісів, додаючи можливості адаптації до програм Fintech.
- Велика екосистема надає інструменти та бібліотеки, розроблені для різних функціональних можливостей Fintech, від візуалізації даних до складних фінансових обчислень.

- Він забезпечує стабільну продуктивність додатків на різних пристроях і операційних системах, що є ключовим для рішень Fintech, націлених на широку базу користувачів.

Де Java може не впоратися? Хоча, як і будь-яка технологія, вона пропонує безліч переваг для програм Fintech, вона не позбавлена потенційних недоліків. Важливо розуміти ці обмеження, щоб приймати зважені рішення:

Накладні витрати на продуктивність: Програми Java працюють на віртуальній машині Java (JVM), вводячи рівень абстракції між скомпільованим кодом і апаратним забезпеченням. Незважаючи на те, що це забезпечує незалежність від платформи, іноді це може призвести до невеликих витрат на продуктивність порівняно з мовами, які компілюються безпосередньо в машинний код.

Споживання пам'яті: Управління пам'яттю JVM, включаючи збір сміття, є однією з сильних сторін Java. Однак програми Java, як правило, можуть споживати більше пам’яті, ніж програми, написані на таких мовах, як C або C++, які пропонують більш прямий контроль пам’яті.

Час запуску: Програми Java, особливо великі, можуть мати довший час запуску через процеси ініціалізації JVM і завантаження класів. У сценаріях, коли швидкий запуск має вирішальне значення, це можна враховувати.

Інтеграція застарілих систем: Незважаючи на те, що Java надає численні інструменти та бібліотеки для інтеграції, взаємодія з деякими старими, пропрієтарними системами може створити проблеми. Це особливо вірно для деяких фінансових установ із IT-інфраструктурою, яка існує десятиліттями.

Проте варто зазначити, що активна спільнота Java та її велика екосистема постійно працюють над вирішенням цих проблем. Оптимізація в останніх версіях JVM, наприклад, скоротила час запуску та покращила продуктивність.

Ось такий підсумок можна зробити по мінусам використання Java:

- Це може викликати накладні витрати на продуктивність через рівень JVM.

- Програми Java можуть споживати більше пам’яті, ніж програми на деяких інших мовах.
- Час запуску програм Java, особливо складних, може бути довшим.
- Його інтеграція з певними застарілими системами може спричинити труднощі.

Фінтех — це складна галузь, яка потребує поєднання швидкості, надійності, відповідності та адаптивності. Java з її універсальним набором функцій, надійними механізмами безпеки та розгалуженою екосистемою надзвичайно добре відповідає цим вимогам.

Можливості Java виходять далеко за межі простого виконання коду. Його внесок у Fintech проявляється у формі безпечних, масштабованих і адаптованих рішень, які обслуговують широкий спектр фінансових операцій і послуг. Тож для чого використовується Java у Fintech? Відповідь зрозуміла: від безпечних фінансових транзакцій до складної аналітики даних, від архітектури мікросервісів до крос-платформної сумісності, утиліта Java у Fintech настільки ж обширна, як і ефективна.

3 ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО РІШЕННЯ

3.1 Функціональні та технічні вимоги до програмного забезпечення

Як ми уже раніше вияснили, в українського інвестора, який інвестує на фондовій біржі, є один і більше брокерських рахунків(інвестиційних рахунків). Ці рахунки можуть бути відкриті в наступних брокерських компаніях:

- Freedom Broker;
- Interactive Brokers.

Щороку тисячі інвесторів здійснюють торгівельну активність на фондових біржах зарубіжних країн, і в залежності від професійних вмінь – отримуючи прибутки чи збитки.

В цілому існують наступні види доходу:

- Від купівлі/продажу цінних паперів (акції, облігації, деривативи і т.д.);
- Нарахування за використання активів (процентні виплати);
- Депозитні нарахування;
- Дивідендні нарахування.

Інвестиційні інструменти, якими в основному торгують українські інвестори на фондовій біржі:

- Акції;
- Облігації;
- Опціони;
- Ф’ючерси;
- Форвардні контракти.

Закон України Про Податковий Кодекс зобов’язує щороку декларувати свої прибутки через подання Податкової Декларації Про Майновий Стан та Доходи. В цій декларації окремими пунктами вказується інвестиційний прибуток з додатком Ф1 «РОЗРАХУНОК податкових зобов'язань з податку на доходи фізичних осіб та

військового збору з доходів, отриманих від операцій з інвестиційними активами» та дивідендний.

Щоб порахувати загальний інвестиційний прибуток та податок на нього, інвестор повинен по кожному виду доходу та інвестиційному інструменту окремо підрахувати дохід, затрати та прибуток.

Для цього інвестор по кожному рахунку працює зі звітом в якому містяться усі операції по активам.

Переважно звіт формується з моменту відкриття рахунку і до кінця податкового періоду, що декларується. Це пов’язано з тим, що дохід з купівлі/продажу акцій, наприклад, це різниця доходу і затрат з продажу і купівлі відповідно, а це дві окремі операції. Ці операції можуть бути розтягнуті в часі, наприклад, купівля могла статися в 2021 році, продаж в 2022, бо інвестор намагався отримати максимально задовільний для нього прибуток і тому зачекав аж до 2022 року, щоб продати. В даному випадку розрахунок прибутку за 2022 включатиме операції за 2021 рік. В цій ситуації варто згадати про принцип FIFO, який ми описували в попередніх розділах.

Ще важливо відмітити те, що інвестор повинен перевести всі операції в національну валюту.

Після виконання розрахунків інвестор заповнює декларацію та додаток Ф1 в податковому кабінеті, підписує за допомогою КЕП і відправляє. Щоб не виконувати цю операцію вручну, податковий кабінет надає можливість сформувати файл для імпортування в XML форматі. Цим ми також скористаємося, щоб зменшити кількість дій де інвестор може допустити помилку.

Так як, проект виконується з подальшою коммерціалізацією необхідно продумати і можливість прийому оплати.

Отже, функціональні вимоги можна виокремити в наступному списку:

- Змога завантажити, опрацювати та зберегти брокерські звіти інвесторів;
- Розрахунок прибутку по кожному окремому інвестиційному рахунку (згідно із правилами розрахунку);

- В рамках магістерської роботи здійснюватиметься розрахунок тільки прибутку від торгівлі акціями;

- Сформувати річну декларацію та додаток Ф1 по інвестиційних рахунках;
  - В рамках магістерської роботи здійснюватиметься генерування додатка Ф1, який відобразить усі акції по, яким було зафіксовано прибуток;
- Прийняти оплату від інвестора за виконання розрахунків та формування декларацій;

Процес формування податкової декларації зображено на рисунку 3.1.

Рисунок 3.1 – Процес формування податкової декларації

Перейдемо до нефункціональних вимог. До них можна віднести питання безпеки, а саме:

- Можливість авторизації та аутентифікації користувача;
- Розмежування доступу між даними користувачів платформи;
- Шифрування даних;

Попередні вимоги забезпечать достатній рівень безпеки для першої версії платформи.

Також до нефункціональних вимог відносяться розгортання серверної частини в хмарі.

В підсумок хочу підкреслити, що функціональні і технічні вимоги відповідають поставленим завданням і навіть розширили їх в контексті прийому оплати від клієнтів платформи.

3.2 Проектування архітектури системи

Враховуючи поставлену мету архітектура буде клієнт-серверна. Це є розподіленим структурним підходом до розробки програмного забезпечення, де функції програми розділені між сервером (який забезпечує послуги) і клієнтом (який користується цими послугами). Існують різні типи клієнт-серверних архітектур, в залежності від способу організації і взаємодії між клієнтом і сервером. Ось деякі з них:

1. **Товстий клієнт (Fat Client) - Тонкий сервер (Thin Server):**
    - У цьому випадку, клієнт виконує значну частину обчислень і має велику кількість функцій, а сервер використовується головним чином для збереження даних і обробки деяких запитань.
2. **Тонкий клієнт (Thin Client) - Товстий сервер (Fat Server):**
    - Тут клієнт виконує обмежені функції, в основному відображає інформацію та передає дії користувача серверу, який виконує більшість обчислень і обробки.
3. **Централізована архітектура:**
    - У цьому випадку, весь обчислювальний процес відбувається на центральному сервері, а клієнти просто отримують доступ до результатів.
4. **Розподілена архітектура:**
    - Обчислення розподілені між кількома серверами, і клієнти можуть взаємодіяти з різними серверами для різних функцій.
5. **Клієнт-сервер з трьома рівнями (Three-Tier Architecture):**

- Розділення функцій на три рівні: клієнтський рівень (інтерфейс користувача), рівень бізнес-логіки (процеси обробки даних), і рівень даних (збереження та отримання даних).

6. **Сервер-серверна архітектура:**
   - Використовується для взаємодії між різними серверами, де кожен сервер може надавати специфічні послуги і обмінюватися інформацією з іншими серверами.
7. **Пірингова (P2P) архітектура:**
   - У цьому випадку всі взаємодіють як рівноправні учасники, і вони можуть функціонувати як клієнти і як сервери одночасно.

Ці типи архітектур можуть комбінуватися або модифікуватися відповідно до конкретних вимог і характеристик проекту. В рамках роботи було поєднано варіанти 2, 3, 5 і 6.

Архітектура взаємодії компонентів системи (див. рис. 3.2) складається з наступних елементів:

1. Auth Service – сервіс аутентифікації та авторизації. Використовується open-source рішення Keycloak.
2. React Client – веб-клієнт, дає змогу кінцевому користувачеві отримувати послугу розрахунку інвестиційного прибутку та формування декларації за допомогою зручного візуального інтерфейсу. Використовує Spring Boot server API. Не розглядається в рамках магістерської роботи.
3. Spring Boot Service – серверна частина проекту, центральний елемент, який виконує усю бізнес-логіку та інтеграції з третіми системами.
4. PostgreSQL – реляційна база даних. Використовується для збереження операцій зі звітів, розрахунків та інших даних користувача.
5. Fondy – платіжна система, дозволяє створювати платежі та отримувати оплату від користувача.

6. API «НБУ» – сервіс Національно Банку України для отримання курсів валют на запитувану дату.

AWS S3 – хмарне сховище файлів. В цьому сховищі зберігаються зашифровані файли звітів брокера, що завантажують користувачі.

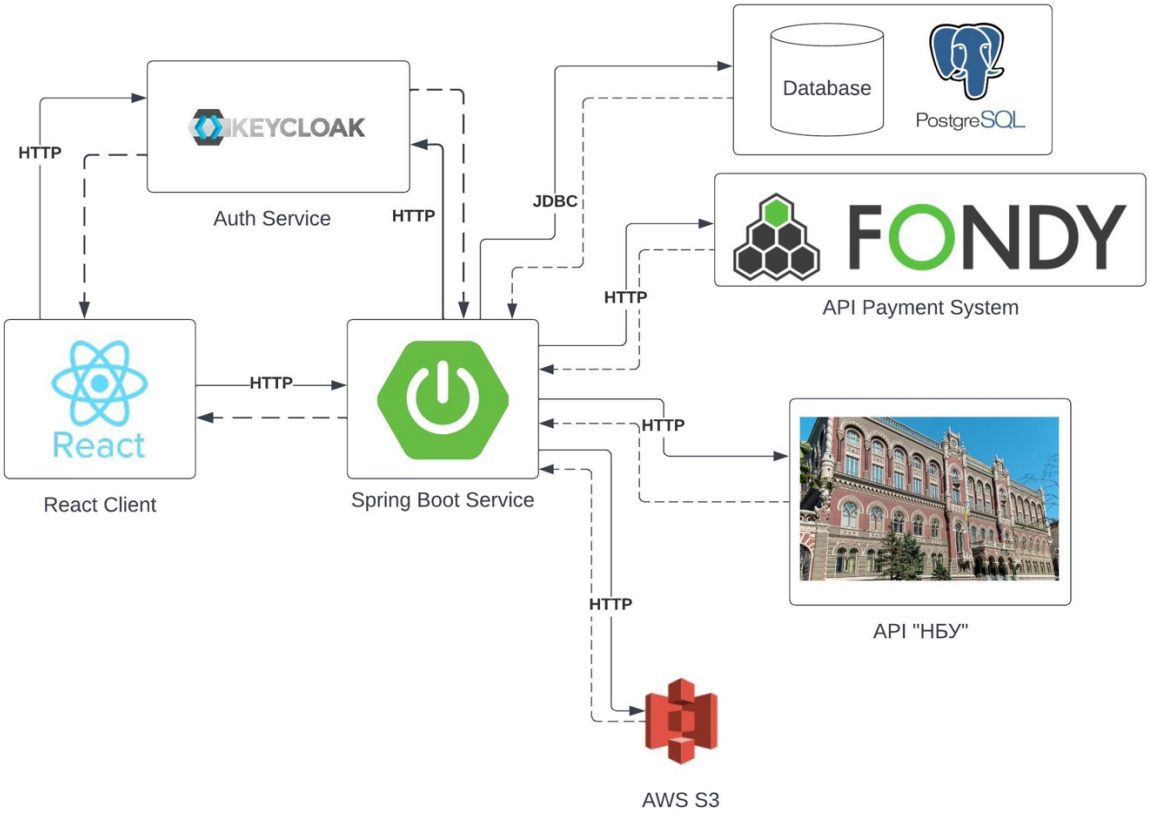


Рисунок 3.2 – Архітектура взаємодії компонентів системи.

Зі схеми на рисунку 3.2 можемо виділити, що в основному взаємодія відбувається за допомогою HTTP протоколу, окрім взаємодії з базою даних.

Архітектура серверної частини монолітна, бо її буде достатньо для розробки і перевірки прототипу.

Вибір Spring Framework для серверної частини дозволить навіть при монолітній архітектурі горизонтально масштабувати, а в подальшому поділити на сервіси через гексагональний підхід до оформлення класів і пакетів проекту.

3.3 Модель даних для збереження операцій торгової активності інвестора зі звітів брокерів

Користувач має можливість завантажити звіти про свої рахунки, які надають брокери. Freedom Broker дає можливість завантажити звіти різних форматів: pdf, xls, json, xml. Найзручнішим і найсучаснішим із них є json (це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною).

Зі звіту Freedom Broker(див. рис. 3.3) необхідно прочитати дані по наступним ключам (включаючи вкладення): date_start, date_end, plainAccountInfoData, userLanguage, userReception, trades.detailed, account_at_start, account_at_end, securities_in_outs, cash_in_outs, corporate_actions.detailed.

```
{
    "date_start": "2021-03-04 23:59:59",
    "date_end": "2023-06-10 23:59:59",
    "plainAccountInfoData": {
        "account_type": " Особистий рахунок фізичної особи ",
        "client_name": "Nazarii Babii",
        "client_code": "183391",
        "base_currency": "USD",
        "tariff_name": "All inclusive in USD",
        "client_date_open": "2021-03-05",
        "activation_date": "2021-03-16"
    },
    "accountInfo": [
    ],
    "userLanguage": "uk",
    "userReception": 35,
    "trades": {
        "detailed": [
            {
                "trade_id": 122636782,
                "date": "2021-03-16 22:16:30",
                "short_date": "2021-03-16",
                "pay_d": "2021-03-18",
                "instr_nm": "IEMG.US",
                "instr_type": 1,
                "instr_kind": "фонд\/ETF",
                "issue_nb": "US46434G1031",
                "operation": "buy",
                "p": 65.48,
                "curr_c": "USD",
                "q": 1,
                "summ": 65.48,
                "turnover": "0.00",
                "profit": 0,
                "fifo_profit": "0.000000",
                "repo_operation": null,
                "mkt_id": 30000000001,
                "order_id": "119117852",
                "office": 35,
                "commission": 2.02,
                "commission_currency": "USD",
                "comment": " Market: usa, security type: stocks, commission currency: USD, service plan:
                    USD.  Total additional commissions: 2 USD ",
                "transaction_id": 496492030,
                "isin": "US46434G1031",
                "offbalance": null,
                "otc": 0,
                "is_dvp": 0,
                "stamp_tax": null,
                "smat": 0,
                "forts_exchange_fee": null,
                "trade_nb": "das_20210316_-T226208",
                "mkt_name": "NYSE\/NASDAQ",
                "id": "122636782\/119117852"
            },
            {
```


Рисунок 3.3 – Приклад брокерського звіту від Freedom Broker

Ці елементи містять в собі дані необхідні для розрахунків. Наприклад, trades.detailed містить операції купівлі чи продажу цінних паперів.

На рисунку 3.4 схематично зображено базу даних операцій по Freedom Broker рахунку. Схема містить 4 таблиці:

- ff_broker_report – головна таблиця, що описує сам файл звіту. На неї посилаються усі інші таблиці на схемі;
- ff_trade – таблиця угод купівлі чи продажу;
- ff_split – таблиця корпоративних дій, а саме дроблення(пряме чи зворотнє), що застосовувалися до цінних паперів та впливають на кількість акцій;

- ff_dividend – таблиця дивідендних нарахувань.

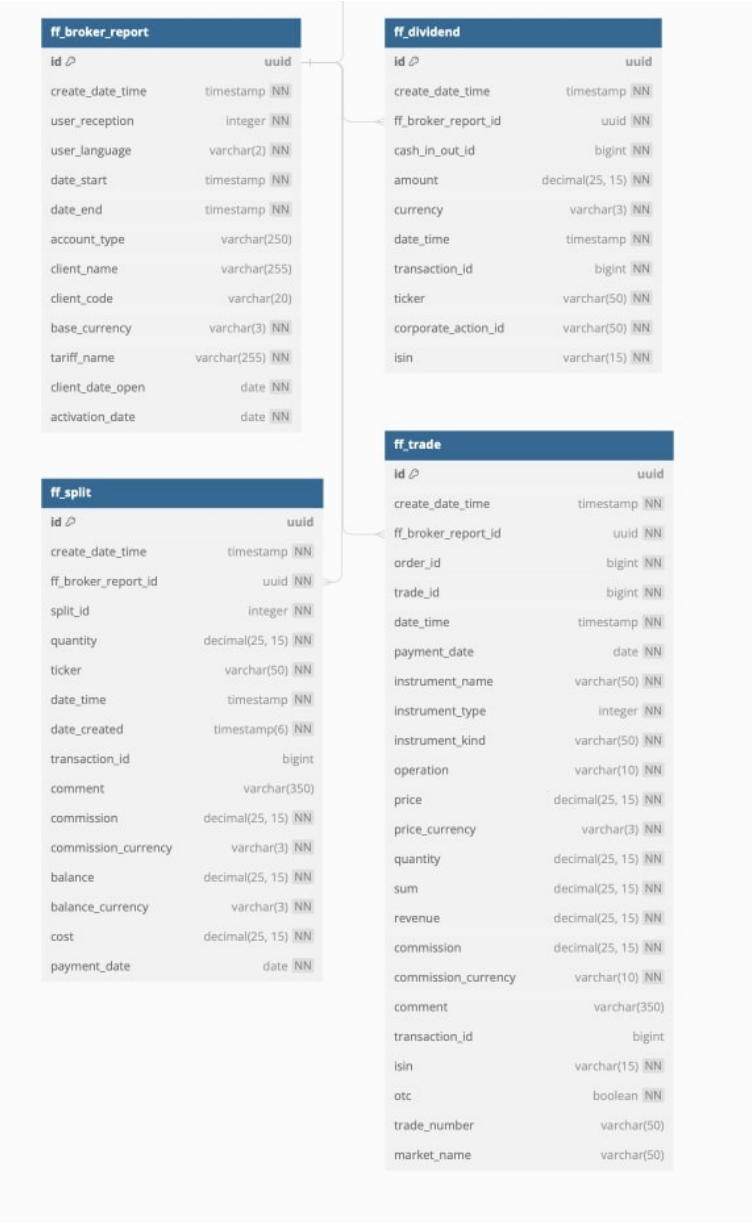


Рисунок 3.4 – Схема бази даних для операцій зі звіту Freedom Broker

У звітах Interactive Brokers найдоступніший звіт, який найпростіше завантажити – це Activiti Statement Report (див. рис. 3.5 і 3.6) в форматі CSV.

```
Net Stock Position Summary,Data,Stocks,USD,PYPL,PAYPAL HOLDINGS INC,86,0,0,86
Net Stock Position Summary,Data,Stocks,USD,VIPS,VIPSHOP HOLDINGS LTD - ADR,868,0,0,868
Trades,Header,DataDiscriminator,Asset Category,Currency,Symbol,Date/Time,Quantity,T. Price,C. Price,Proceeds,Comm/Fee,Basis,Realized P/L,MTM P/L,Code
Trades,Data,Order,Stocks,USD,AAPL,"2021-01-11, 12:40:39",925,129.916886486,128.98,-120173.12,-5.303979563,120178.423979563,0,-866.62,O;P
Trades,Data,Order,Stocks,USD,AAPL,"2021-01-13, 13:37:31",-925,130.891859459,130.89,121074.97,-6.3707114,-120178.423979,890.175309,1.72,C;P
Trades,Data,Order,Stocks,USD,AAPL,"2021-01-26, 15:52:10",209,143.18,143.16,-29924.62,-0.773837653,29925.393837653,0,-4.18,O
Trades,Data,Order,Stocks,USD,AAPL,"2021-01-27, 09:42:48",-209,142.445809569,142.06,29771.1742,-1.276651602,-29925.393837,-155.496289,80.6342,C;P
Trades,SubTotal,,Stocks,USD,AAPL,,0,,,748.4042,-13.725180218,0.000001216,734.67902,-788.4458,
Trades,Data,Order,Stocks,USD,ABT,"2021-01-26, 15:52:53",284,114.829478873,114.73,-32611.572,-0.82153059,32612.39353059,0,-28.252,O;P
Trades,Data,Order,Stocks,USD,ABT,"2021-01-27, 09:39:55",-284,115.25,114.29,32731,-1.80868169,-32612.39353,116.797788,272.64,C
Trades,SubTotal,,Stocks,USD,ABT,,0,,,119.428,-2.63021228,0.00000059,116.797788,244.388,
Trades,Data,Order,Stocks,USD,AMZN,"2021-05-25, 09:30:02",15,3266,3259.05,-48990,-0.37575725,48990.37575725,0,-104.25,O
Trades,Data,Order,Stocks,USD,AMZN,"2021-07-06, 10:39:37",-15,3612.91,3675.74,54193.65,-0.631429865,-48990.375757,5202.642813,-942.45,C
Trades,SubTotal,,Stocks,USD,AMZN,,0,,,5203.65,-1.007187115,0.00000025,5202.642813,-1046.7,
Trades,Data,Order,Stocks,USD,ANSS,"2021-06-02, 15:22:59",29,336.41,336.41,-9755.89,-0.30965725,9756.19965725,0,0,O
Trades,Data,Order,Stocks,USD,ANSS,"2021-08-04, 14:03:59",-29,373.29,372.57,10825.41,-0.414717841,-9756.199657,1068.795625,20.88,C
Trades,SubTotal,,Stocks,USD,ANSS,,0,,,1069.52,-0.724375091,0.00000025,1068.795625,20.88,
Trades,Data,Order,Stocks,USD,ARRY,"2021-03-24, 09:57:21",685,29.18,27.65,-19988.3,-2.536262163,19990.836262163,0,-1048.05,O
Trades,Data,Order,Stocks,USD,ARRY,"2021-03-24, 10:04:35",172,29.03,27.65,-4993.16,-0.63684247,4993.79684247,0,-237.36,O
Trades,Data,Order,Stocks,USD,ARRY,"2021-04-28, 15:13:36",-857,29.2,29.25,25024.4,-1.603012073,-24984.633104,38.163884,-42.85,C;P
Trades,SubTotal,,Stocks,USD,ARRY,,0,,,42.94,-4.776116706,0.000000633,38.163884,-1328.26,
Trades,Data,Order,Stocks,USD,BABA,"2021-07-27, 09:30:27",82,184,186.07,-15088,-0.44865725,15088.44865725,0,169.74,O
Trades,Data,Order,Stocks,USD,BABA,"2021-08-19, 09:31:54",60,165.492325,160.55,-9929.5395,-0.42125725,9929.96075725,0,-296.5395,O;P
Trades,Data,Order,Stocks,USD,BABA,"2021-08-19, 09:31:54",30,165.494666667,160.55,-4964.84,-0.38525725,4965.22525725,0,-148.34,O;P
Trades,Data,Order,Stocks,USD,BABA,"2021-09-20, 14:58:52",33,150.22,151.49,-4957.26,-0.35685725,4957.61685725,0,41.91,O
Trades,Data,Order,Stocks,USD,BABA,"2021-12-09, 09:43:33",118,126.75,123.91,-14956.5,-0.436903555,14956.936903555,0,-335.12,O
Trades,SubTotal,,Stocks,USD,BABA,,323,,,-49896.1395,-2.048932555,49898.188432555,0,-568.3495,
Trades,Data,Order,Stocks,USD,BNGO,"2021-01-27, 09:41:16","2,913",10.984507381,11.04,-31997.87,-10.785593692,32008.655593692,0,161.65,O;P
Trades,Data,Order,Stocks,USD,BNGO,"2021-02-04, 09:30:10","-2,913",11.492403021,11.18,33477.37,-14.56299057,-32008.655593,1454.151413,910.03,C;P
Trades,Data,Order,Stocks,USD,BNGO,"2021-02-04, 14:35:38","2,240",11.159732143,11.18,-24997.8,-8.0177624,25005.8177624,0,45.4,O;P
Trades,Data,Order,Stocks,USD,BNGO,"2021-02-04, 14:38:13","2,228",11.219486086,11.18,-24997.015,-8.30633153,25005.32133153,0,-87.975,O;P
Trades,Data,Order,Stocks,USD,BNGO,"2021-02-10, 09:33:59","-4,468",11.637162041,11.38,51994.84,-28.847871893,-50011.13909,1954.853034,1149,C;P
Trades,Data,Order,Stocks,USD,BNGO,"2021-02-17, 12:20:25","1,756",14.23,14.54,-24987.88,-5.59171731,24993.47171731,0,544.36,O;P
Trades,Data,Order,Stocks,USD,BNGO,"2021-02-25, 09:32:14","-1,756",10.74,10.02,18859.44,-3.119264454,-24993.471718,-6137.150982,1264.32,C
Trades,SubTotal,,Stocks,USD,BNGO,,0,,,-2648.915,-79.231531849,0.000003932,-2728.146535,3986.785,
Trades,Data,Order,Stocks,USD,CGC,"2021-01-13, 13:41:47",940,32.378,32.18,-30435.32,-3.48041815,30438.80041815,0,-186.12,O
Trades,Data,Order,Stocks,USD,CGC,"2021-01-15, 09:35:59",-940,33.733191489,33.39,31709.2,-6.21305147,-30438.800416,1264.186529,322.6,C;P
Trades,SubTotal,,Stocks,USD,CGC,,0,,,1273.88,-9.69346962,0.00000215,1264.186529,136.48,
Trades,Data,Order,Stocks,USD,CNBS,"2021-03-12, 14:02:36",611,31.066543372,31.16,-18981.658,-2.082271798,18983.740271798,0,57.102,O;P
Trades,Data,Order,Stocks,USD,CNBS,"2021-03-16, 09:55:50",-611,31.92,31.43,19503.12,-1.151346709,-18983.740273,518.228382,299.39,C;P
Trades,SubTotal,,Stocks,USD,CNBS,,0,,,521.462,-3.233618507,-0.000001202,518.228382,356.492,
Trades,Data,Order,Stocks,USD,COIN,"2021-08-12, 15:49:26",3,257.68,256.5,-773.04,-0.35085725,773.39085725,0,-3.54,O
Trades,Data,Order,Stocks,USD,COIN,"2021-09-20, 15:00:41",21,233.4,236.53,-4901.4,-0.35445725,4901.75445725,0,65.73,O
Trades,Data,Order,Stocks,USD,COIN,"2021-10-27, 09:30:02",48,309.313697917,311.67,-14847.0575,-0.43035725,14847.48785725,0,113.1025,O;P
Trades,Data,Order,Stocks,USD,COIN,"2021-11-04, 13:58:52",29,344.06,344.45,-9977.74,-0.35605725,9978.09605725,0,11.31,O
Trades,Data,Order,Stocks,USD,COIN,"2021-11-22, 10:42:45",-31,328.34,315.48,10178.54,-0.412056804,-7840.37938,2337.748564,398.66,C
Trades,SubTotal,,Stocks,USD,COIN,,70,,,-20320.6975,-1.903785804,22660.349849,2337.748564,585.2625,
Trades,Data,Order,Stocks,USD,CRM,"2021-06-08, 09:30:01",62,238.773951613,236.42,-14803.985,-0.38515725,14804.37015725,0,-145.945,O;P
Trades,Data,Order,Stocks,USD,CRM,"2021-06-14, 10:30:45",40,245.795,246.26,-9831.8,-0.35825725,9832.15825725,0,18.6,O
Trades,Data,Order,Stocks,USD,CRM,"2021-06-14, 14:39:39",-102,245.94,246.26,25085.88,-0.517738383,-24636.528414,448.833847,-32.64,C
Trades,SubTotal,,Stocks,USD,CRM,,0,,,450.095,-1.261152883,0.0000005,448.833847,-159.985,
Trades,Data,Order,Stocks,USD,FFM,"2021-06-01, 09:59:53",357.55,97859944.56,-19984.36,-1.356018383,19985.716018383,0,7.64,O;P
```

Рисунок 3.5 – Сирий CSV формат звіту Interactive Brokers

Net Stock Position Summary	Data	Stocks	USD	PYPL	PAYPAL HOLDIN(	86	0	0	86	
Net Stock Position Summary	Data	Stocks	USD	VIPS	VIPSHOP HOLDI	868	0	0	868	
Trades	Header	DataDiscriminator	Asset Category	Currency	Symbol	Date/Time	Quantity	T. Price	C. Price	Proceeds
Trades	Data	Order	Stocks	USD	AAPL	2021-01-11, 12:40	925	129.916886486	128.98	-120173.12
Trades	Data	Order	Stocks	USD	AAPL	2021-01-13, 13:37	-925	130.891859459	130.89	121074.97
Trades	Data	Order	Stocks	USD	AAPL	2021-01-26, 15:52	209	143.18	143.16	-29924.62
Trades	Data	Order	Stocks	USD	AAPL	2021-01-27, 09:42	-209	142.445809569	142.06	29771.1742
Trades	SubTotal		Stocks	USD	AAPL		0			748.4042
Trades	Data	Order	Stocks	USD	ABT	2021-01-26, 15:52	284	114.829478873	114.73	-32611.572
Trades	Data	Order	Stocks	USD	ABT	2021-01-27, 09:39	-284	115.25	114.29	
Trades	SubTotal		Stocks	USD	ABT		0			119.428
Trades	Data	Order	Stocks	USD	AMZN	2021-05-25, 09:30	15	3266	3259.05	
Trades	Data	Order	Stocks	USD	AMZN	2021-07-06, 10:39	-15	3612.91	3675.74	54193.65
Trades	SubTotal		Stocks	USD	AMZN		0			5203.65
Trades	Data	Order	Stocks	USD	ANSS	2021-06-02, 15:22	29	336.41	336.41	-9755.89
Trades	Data	Order	Stocks	USD	ANSS	2021-08-04, 14:03	-29	373.29	372.57	10825.41
Trades	SubTotal		Stocks	USD	ANSS		0			1069.52
Trades	Data	Order	Stocks	USD	ARRY	2021-03-24, 09:57	685	29.18	27.65	-19988.3
Trades	Data	Order	Stocks	USD	ARRY	2021-03-24, 10:04	172	29.03	27.65	-4993.16
Trades	Data	Order	Stocks	USD	ARRY	2021-04-28, 15:13	-857	29.2	29.25	25024.4
Trades	SubTotal		Stocks	USD	ARRY		0			42.94
Trades	Data	Order	Stocks	USD	BABA	2021-07-27, 09:30	82	184	186.07	
Trades	Data	Order	Stocks	USD	BABA	2021-08-19, 09:31	60	165.492325	160.55	-9929.5395
Trades	Data	Order	Stocks	USD	BABA	2021-08-19, 09:31	30	165.494666667	160.55	-4964.84
Trades	Data	Order	Stocks	USD	BABA	2021-09-20, 14:58	33	150.22	151.49	-4957.26
Trades	Data	Order	Stocks	USD	BABA	2021-12-09, 09:43	118	126.75	123.91	-14956.5
Trades	SubTotal		Stocks	USD	BABA		323			-49896.1395
Trades	Data	Order	Stocks	USD	BNGO	2021-01-27, 09:41	2,913	10.984507381	11.04	-31997.87
Trades	Data	Order	Stocks	USD	BNGO	2021-02-04, 09:30	-2,913	11.492403021	11.18	33477.37
Trades	Data	Order	Stocks	USD	BNGO	2021-02-04, 14:35	2,240	11.159732143	11.18	-24997.8
Trades	Data	Order	Stocks	USD	BNGO	2021-02-04, 14:38	2,228	11.219486086	11.18	-24997.015
Trades	Data	Order	Stocks	USD	BNGO	2021-02-10, 09:33	-4,468	11.637162041	11.38	51994.84
Trades	Data	Order	Stocks	USD	BNGO	2021-02-17, 12:20	1,756	14.23	14.54	-24987.88

Рисунок 3.6 – Табличний вигляд CSV звіту Interactive Brokers

Для Interactive Brokers застосовується схожа структура бази даних, як у Freedom Broker (див. рис. 3.7).

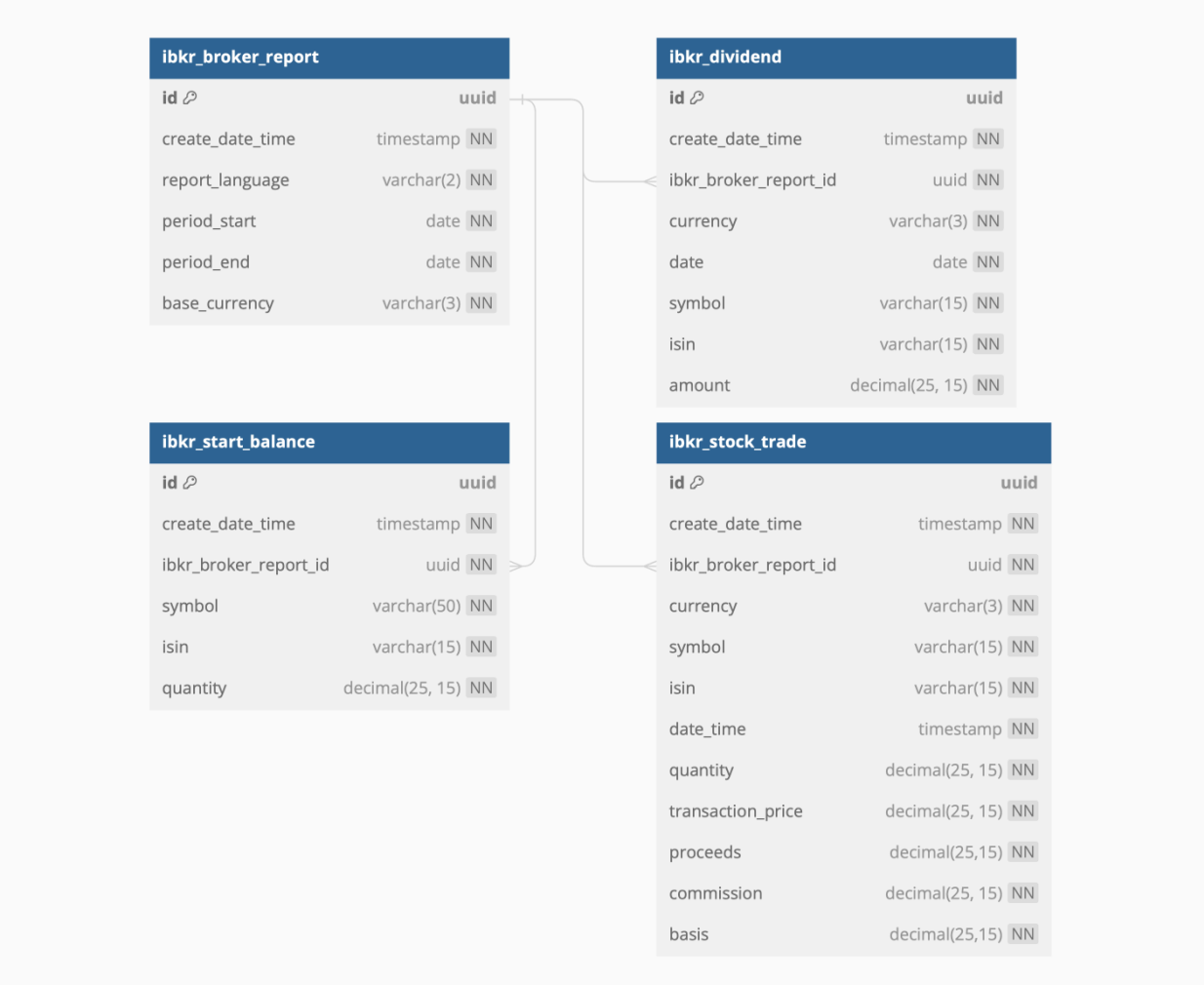


Рисунок 3.7 – Схема бази даних для операцій зі звіту Interactive Brokers

3.4 Збагачення даними операцій з брокерських звітів актуальними обмінними курсами валют НБУ

Щоб збагатити значення торгівельних операцій даними про курс валют на дату виконання було розроблено модуль інтеграції з API Національного Банку України. На рисунку 3.8 зображено клас, який керує доступом до запитів в НБУ. В ньому реалізовано кешування даних для зменшення кількості запитів по мережі. Якщо курс для конкретної пари на конкретну дату уже був отриманий, то повторний запит не відбувається, а дані отримуються з кешу.

```java
@RequiredArgsConstructor
@Component
public class NbuAdapter implements FetchExchageRateNbuPort {

    private final NbuConnector nbuConnector;

    private final HashMap<String, HashMap<LocalDate, ExchangeRate>> hashMap = new HashMap<>();

    private final List<String> currencies = List.of("USD", "EUR", "HKD", "GBP");

    @PostConstruct
    public void init() {
        currencies.forEach(c -> {
            final var exchangeRates =
                    nbuConnector.getExchangeRates(LocalDate.of( year: 2019,  month: 1,  dayOfMonth: 1), LocalDate.now(), c);
            final HashMap<LocalDate, ExchangeRate> ratesByDays = new HashMap<>();
            for (ExchangeRate rate : exchangeRates) {
                ratesByDays.put(rate.getExchangeDate(), rate);
            }
            hashMap.put(c, ratesByDays);
            try {
                Thread.sleep( millis: 300);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        });
    }

    @Override
    public ExchangeRate fetch(LocalDate date, String fromCurrency) {
        if (hashMap.containsKey(fromCurrency)) {
            final var ratesByDays = hashMap.get(fromCurrency);
            if (ratesByDays.containsKey(date))
                return ratesByDays.get(date);
        }
        return makeRequest(date, fromCurrency);
    }
```

Рисунок 3.8 – Адаптер модуля курсів валют

На рисунку 3.9 зображено клас запиту до НБУ за курсом валют і його параметри, а рисунок 3.10 відображає модель даних, що повертається в масиві відповіді від НБУ.

```java
import ...

@RequiredArgsConstructor
@Component
public class NbuConnector {
//    https://bank.gov.ua/NBU_Exchange/exchange_site?start=20220115&end=20230315&valcode=usd&sort=exchangedate&order=desc&json

    @Qualifier("nationalBankRestTemplate")
    private final RestTemplate restTemplate;


    public ExchangeRate[] getExchangeRates(LocalDate start, LocalDate end, String currency) {

        final var url = UriComponentsBuilder.fromHttpUrl("https://bank.gov.ua/NBU_Exchange/exchange_site")
                .queryParam( name: "start", DateUtils.changeFormat(start.toString(), DateUtils.yyyyMMdd))
                .queryParam( name: "end", DateUtils.changeFormat(end.toString(), DateUtils.yyyyMMdd))
                .queryParam( name: "valcode", currency.toLowerCase())
                .queryParam( name: "sort",  ...values: "exchangedate")
                .queryParam( name: "order",  ...values: "desc")
                .queryParam( name: "json",  ...values: true)
                .build().toString();

        return restTemplate.getForEntity(url, ExchangeRate[].class).getBody();
    }

}
```

Рисунок 3.9 – Клас для запитів до API НБУ

```java
@Jacksonized
@Getter
@ToString
@Builder
public class ExchangeRate {

    @JsonProperty("exchangedate")
    @JsonFormat(pattern = "dd.MM.yyyy")
    private final LocalDate exchangeDate;
    @JsonProperty("r030")
    private final String r030;
    @JsonProperty("cc")
    private final Currency currency;
    @JsonProperty("txt")
    private final String nameUa;
    @JsonProperty("enname")
    private final String nameEN;
    @JsonProperty("rate")
    private final BigDecimal rate;
    @JsonProperty("units")
    private final Integer units;
    @JsonProperty("rate_per_unit")
    private final BigDecimal ratePerUnit;
    @JsonProperty("group")
    private final String group;
    @JsonProperty("calcdate")
    @JsonFormat(pattern = "dd.MM.yyyy")
    private final LocalDate calculatedDate;

}
```

Рисунок 3.10 – Клас моделі даних відповіді курсу валют від НБУ

3.5 Алгоритм розрахунку інвестиційного прибутку

З дивідендами алгоритм простий: необхідно отримати операції нарахувань, перевести значення в національну валюту і просто накопичити суму. Далі застосувати ставки податкової і розрахунок готовий.

Операції з цінними паперами потребують більшої і складнішої підготовки, а саме розподіл операцій за принципом FIFO, що згадувався в розділі 1.

Отже, нам дано перелік угод в хронологічному порядку по різним емітентам. Це частина нашого брокерського звіту. По-факту, угоди над цінними паперами змінюють кількість на балансі тих чи інших активів. Угода купівлі – додає певну кількість цінних паперів на баланс, а натомість зменшує залишок грошей за які було придбано цінні папери. Угода продажу, навпаки, зменшує кількість активів та збільшує кількість грошей. Це якщо говорити про лонгові угоди (такі, що були виконані з використанням наших власних активів).

Також існують шортові угоди. Наприклад, ми позичили у брокера 5 акцій компанії AAPL.US та продали по 100$ за штуку. У нас на залишку балансу -5 акцій AAPL.US та +500$. З балансом відбулося теж саме, що ми описали вище про угоду продажу. Проте тепер нам прийдеться відкупити 5 акцій AAPL.US та повернути їх брокеру, але це уже інша тема – коротких позицій.

Повернімося до нашого переліку угод і спробуємо за принципом FIFO співставити купівлі і продажі, щоб вирахувати прибуток по кожному паперу.

Прибуток будемо рахувати по кожному емітенту окремо, а потім підсумуємо. Для прикладу випишемо тільки угоди по компанії AAPL.US (див. таблицю 3.1). Угоди де кількість від’ємна – продаж, а де позитивна – купівля. За хронологію відповідає стовпчик «№».

Порахуємо за принципом FIFO прибуток по компанії AAPL.US. Починаємо з першої угоди в хронологічному порядку.

Перша угода добавляє кількість акцій та зменшує кількість грошей, отже, це угода купівлі. Ми придбали 5 акцій по ціні 100$ за штуку з комісією 2,10 $. Наші загальні витрати на купівлю становлять 502,10 $.

Таблиця 3.1 – Усі угоди перед розподілом

№	Кількість	Ціна	Сума	Комісія
1	+5	100 $	-500 $	-2.10 $
2	-5	110 $	+550 $	-2.10 $
3	+5	105 $	-525 $	-2.10 $
4	-2	105 $	+210 $	-2.04 $
5	-3	120 $	+360 $	-2.06 $

Тепер ми повинні шукати наступну угоду, але продажу. Наступна угода продажу – це угода №2. Ми продали 5 акцій по ціні 110$ за штуку з комісією 2,10 $. Наші загальні витрати на продаж становлять 2,10 $, а дохід 550 $.

Так як кількість активів першої і другої угоди рівні по модулю – ми можемо їх об’єднати умовно в пакет і закрити його підрахувавши загальний прибуток і витрати. Отже, загальний прибуток (беремо дані з таблиці) дорівнюють +550 $ + (-500 $) + (-2.10 $) + (-2.10 $) = +45.80 $.

Угоди 1 і 2 прибираємо з переліку. Вітаю перший пакет закрито і наш перелік має наступний вигляд (див. таблицю 3.2).

Таблиця 3.2 – Угоди після першої ітерації алгоритму FIFO

№	Кількість	Ціна	Сума	Комісія
3	+5	105 $	-525 $	-2.10 $
4	-2	105 $	+210 $	-2.04 $
5	-3	120 $	+360 $	-2.06 $

Тепер повторюємо таку ж ітерацію. Наступна перша хронологічна угода має №3. Ми придбали 5 акцій по ціні 105$ за штуку з комісією 2,10 $. Наші загальні витрати на купівлю становлять 527,10 $.

Тепер ми шукаємо наступну угоду, яка змінює баланс в протилежну сторону. Якщо у нас спершу була купівля (баланс +), то тепер шукаємо продаж (баланс -). Наступна угода продажу – це угода №4. Ми продали 2 акцій по ціні 105$ за штуку з комісією 2,10 $. Наші загальні витрати на продаж становлять 2,04 $, а дохід 210 $.

У нас не сходиться кількість, бо $|+5| \neq |-2|$. Що ж нам робити?

Відповідь – дробити одну з угод, а саме ту, яка по модулю має більше значення, брати з неї частинку рівну угоді, яка по модулю має менше значення, а залишок залишити в переліку угод.

Отже, угода №3 дробиться на наступні дві угоди (див. таблицю 3.3).

Таблиця 3.3 – Угоди, які роздробилися

№	Кількість	Ціна	Сума	Комісія
3.1	+2	105 $	-210 $	-1.04 $
3.2	+3	105 $	-315 $	-1.06 $

Тепер угода № 3.1 рівна нашій угоді № 2 по кількості взятій по модулю. Їх ми об’єднаємо в пакет. Ми умовно позначили їх 3.1 і 3.2, щоб була різниця, але насправді вони хронологічно рівні, тобто в реальності у них ідентичні дата і час виконання. Це зроблено тут для наглядності.

Отже, загальний прибуток (беремо дані з таблиці) = +210 $ + (-210 $) + (-2.10 $) + (-1.04 $) = -3.14 $. Так як прибуток від’ємний ми маємо збиток по цій парі угод. Вітаю другий пакет закрито і наш перелік має наступний вид (див. таблицю. 3.4).

Таблиця 3.4 – Список угод після другої ітерації

№	Кількість	Ціна	Сума	Комісія
3.2	+3	105 $	-315 $	-1.06 $
5	-3	120 $	+360 $	-2.06 $

Тепер повторюємо таку ж ітерацію. Наступна перша хронологічна угода має №3.2 (по-факту 3 враховуючи наші попередні дії). Ми придбали 3 акції по ціні 105$ за штуку з комісією 2,10 $. Наші загальні витрати на купівлю становлять 316,06 $.

Тепер ми шукаємо наступну угоду, яка змінює баланс в протилежну сторону. Якщо у нас перша була купівля (баланс +), то тепер шукаємо продаж (баланс -). Наступна угода продажу – це угода №5. Ми продали 3 акції по ціні 120$ за штуку з комісією 2,06 $. Наші загальні витрати на продаж становлять 2,06 $, а дохід 360 $.

Так як кількість активів першої і другої угоди рівні по модулю – ми можемо їх об’єднати умовно в пакет і закрити його підрахувавши загальний прибуток і витрати. Отже, загальний прибуток (беремо дані з таблиці) = +360 $ + (-315 $) + (-2.06 $) + (-1.06 $) = +41.88 $

Угоди 3.2 і 5 прибираємо з переліку. Вітаю третій пакет закрито і наш перелік тепер пустий. Отже, на балансі у нас не залишилося жодної акції AAPL.US.

Якщо б у нас була б ще якась угода на +5 чи -4 по кількості, то у нас би баланс був +5 і -4 відповідно, а так як більше угод ми не мали б, то не могли б сформувати пакет. І це так би залишився залишок на балансі.

Пропоную загальний алгоритм:

- Отримуємо угоди по сортовані за датою по зростанню;
- Ділимо їх по операціям (два масиви) та конвертуємо знаки для кількості і суми:
- sell “-” – мінус акції, плюс гроші;
- buy “+” – плюс акції, мінус гроші;

Послідовно виконуємо наступні дії поки у нас не закінчаться елементи в обох масивах або буде неможливо закрити пакет по кількості (є залишок на балансі).

1. Беремо перші значення з обох масивів і порівнюємо їх за датою.
2. Із двох беремо старішу угоду. Це буде базова угода пакету. (Угода відкриття позиції)
3. Добавляємо її в пакет.

4. По знаку к-сті визначаємо яка це позиція:
    - лонгова – “+”
    - шортова – “-”
5. Якщо пакет в позиції
    - Лонговій – беремо з масиву sell найближчу угоду по даті.
    - Шортовій – беремо з масиву buy найближчу угоду по даті.
6. Додаємо кількість вибраної угоди до базової,
    - якщо результат, для лонгової позиції:
        - = 0, то закриваємо пакет.
        - < 0, то дробимо вибрану угоду, залишок кладемо назад і закриваємо пакет.
        - > 0, то дробимо базову угоду, залишок кладемо назад і закриваємо пакет.
    - якщо результат, для шортової позиції:
        - = 0, то закриваємо пакет.
        - > 0, то дробимо вибрану угоду, залишок кладемо назад і закриваємо пакет.
        - < 0, то дробимо базову угоду, залишок кладемо назад і закриваємо пакет.
7. Повертаємося на крок 1, якщо елементи закінчилися в обох масивах або неможливо закрити пакет по кількості (є залишок на балансі), то розподіл закінчено без залишку або з відповідно.

Код з розподілом по пакетах винесено в Додаток А.

3.6 Механізм генерації вихідних файлів для зручного імпортування в податковий кабінет

API податкового кабінету надає XSD-схеми для валідації і генерації XML файлів декларацій, що можуть бути імпортовані (див. рис. 3.11).

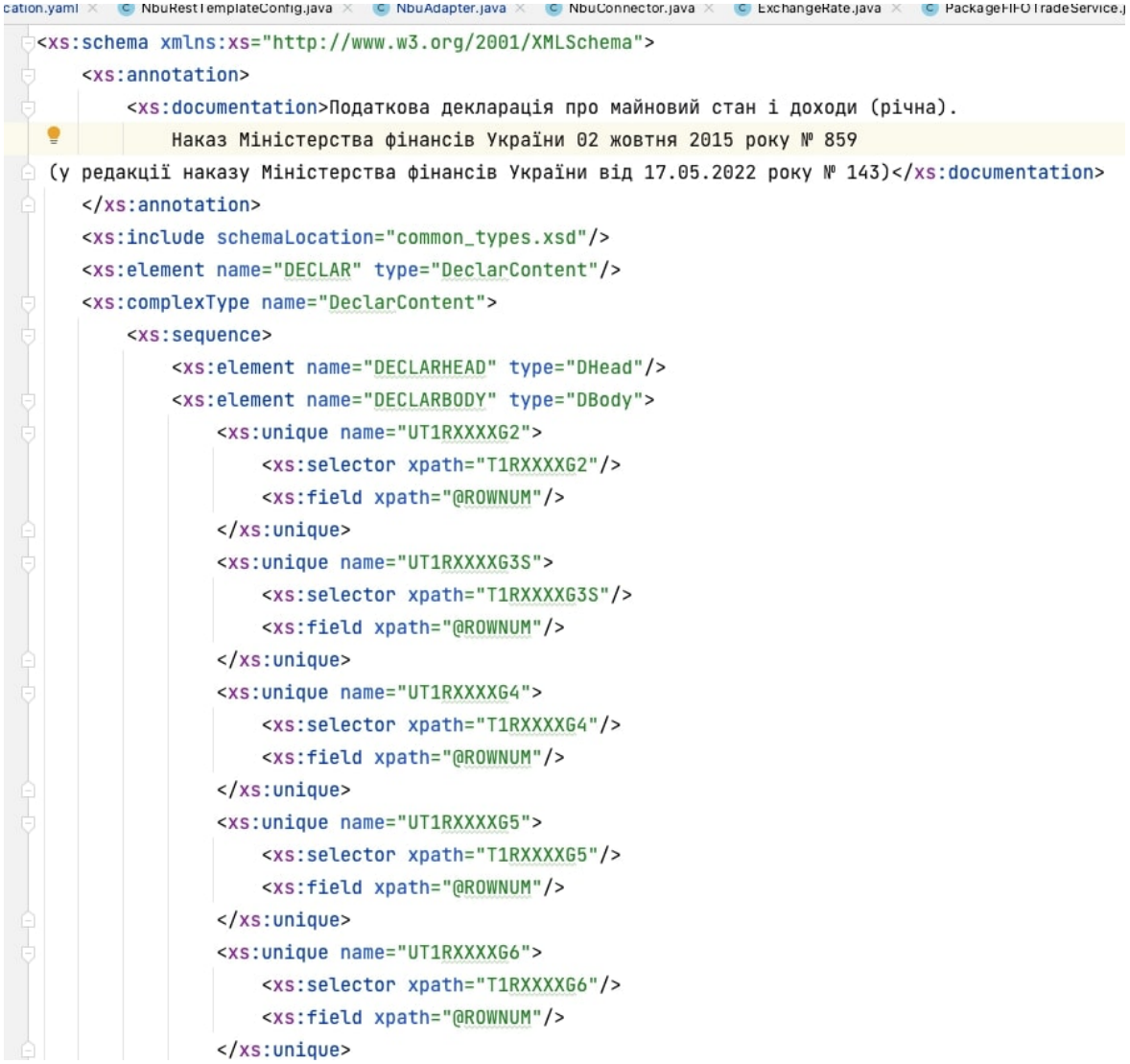


Рисунок 3.11 – Приклад частини XSD-схеми для декларації

Для генерації класів підключимо плагін jaxb2-maven-plugin(див. рис. 3.12).

```xml
</plugin>
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>jaxb2-maven-plugin</artifactId>
    <version>3.1.0</version>
    <executions>
        <execution>
            <id>F0100213-xsd2classes</id>
            <phase>generate-sources</phase>
            <goals>
                <goal>xjc</goal>
            </goals>
            <configuration>
                <xjbSources>
                    <xjbSource>src/main/resources/bindings.xjb</xjbSource>
                </xjbSources>
                <sources>
                    <source>src/main/resources/static/xsd/F0100213.xsd</source>
                </sources>
                <clearOutputDir>false</clearOutputDir>
                <outputDirectory>${basedir}/target/generated-sources/jaxb2</outputDirectory>
                <packageName>ua.gov.tax.cabinet.domain.f0100213</packageName>
                <extension>true</extension>
            </configuration>
        </execution>
        <execution>
```

Рисунок 3.12 – Maven конфігурація плагіну генерації Java-класів з XSD-схеми

Завдяки згенерованим класам ми можемо їх перетворити в XML файл, приклад якого зображено на рисунку 3.13.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <DECLAR xsi:schemaLocation="F0121213.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <DECLARBODY>
      <HZY>0</HZY>
      <T1RXXXXG2 ROWNUM="1">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="2">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="3">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="4">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="5">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="6">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="7">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="8">2</T1RXXXXG2>
      <T1RXXXXG3S ROWNUM="1">S_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="2">TOST_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="3">TASK_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="4">PAY_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="5">MQ_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="6">DOCS_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="7">HLTH_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="8">DLO_IPO.US</T1RXXXXG3S>
      <T1RXXXXG4 ROWNUM="1">1302.27</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="2">-120.20</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="3">2896.19</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="4">107.35</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="5">-43.42</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="6">1527.20</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="7">-114.15</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="8">3712.96</T1RXXXXG4>
      <T1RXXXXG5 ROWNUM="1">-3017.75</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="2">-1126.64</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="3">-1988.45</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="4">-608.87</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="5">-1548.37</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="6">-756.04</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="7">-895.12</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="8">-1836.75</T1RXXXXG5>
      <T1RXXXXG6 ROWNUM="1">1302.27</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="2">-120.20</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="3">2896.19</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="4">107.35</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="5">-43.42</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="6">1527.20</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="7">-114.15</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="8">3712.96</T1RXXXXG6>
    </DECLARBODY>
  </DECLAR>
```

Рисунок 3.13 – Вихідний XML-файл для генерації

Після успішної генерації, XML-файл можна імпортувати в податковий кабінет і усі дані проставляться автоматично(див. рис. 3.14).

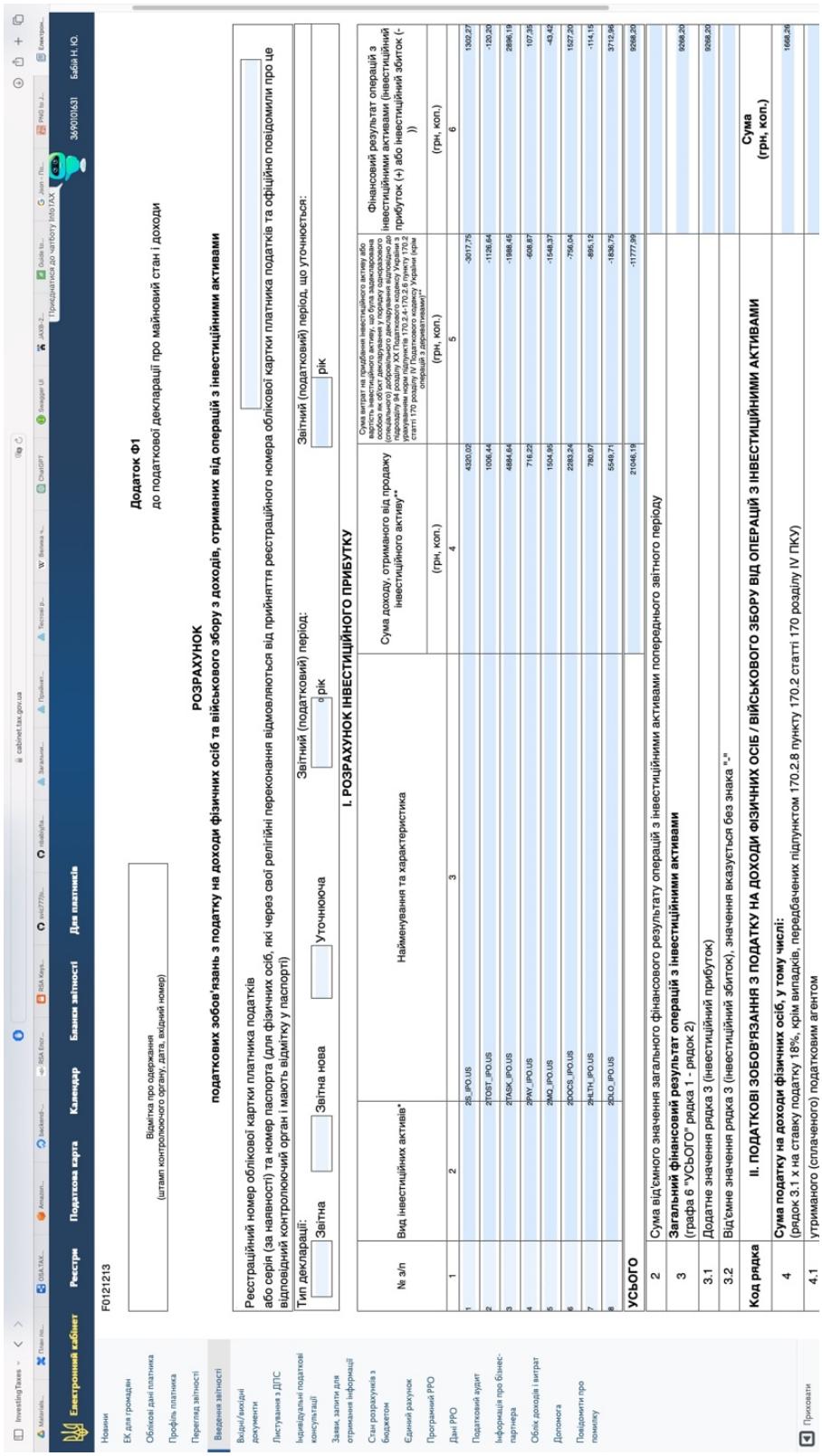


Рисунок 3.14 – Імпортований XML-файл додатку Ф1 в податковий кабінет

3.7 Шифрування звітів брокерів та вихідних файлів для забезпечення безпеки передачі даних між серверним і клієнтським застосунками

Шифрування файлів при їх передачі від React Client на сервер є важливим етапом для забезпечення конфіденційності та безпеки інформації. RSA і AES – це асиметричний та симетричний алгоритми шифрування відповідно, і їх можна комбінувати для забезпечення високого рівня безпеки.

Нижче подано загальний опис способу шифрованої передачі файлів з React Client на сервер за допомогою RSA і AES:

1. **Генерація ключів RSA:**
    a. На React Client генерується пара ключів RSA – публічний і приватний ключі. Приватний ключ зберігається безпечно на React Client, а публічний ключ відправляється на сервер.
2. **Шифрування файлу AES:**
    a. Коли користувач вибирає файл для передачі, на сервері генерується випадковий ключ (symmetric key) для алгоритму AES. Цей ключ буде використовуватися для шифрування та розшифрування самого файлу.
    b. Файл шифрується алгоритмом AES, який був розшифрований приватним ключем, на React Client з використанням випадкового ключа.
3. **Передача шифрованого файл:**
    a. Шифрований файл передаються на сервер по безпечному каналу зв'язку.
4. **Розшифрування на бекенді:**
    a. Згенерований випадковий ключ використовується для розшифрування самого файлу, який був шифрований алгоритмом AES на React Client.

5. **Збереження розшифрованого файлу:**
    a. Розшифрований файл може бути збережений на сервері або використовуватися для подальших операцій, залежно від конкретних вимог системи.

Цей процес забезпечує безпеку файлу під час передачі, оскільки навіть якщо хтось перехопить шифрований файл та ключ, без приватного ключа RSA дешифрувати файл буде вкрай складно. Також, використання симетричного шифрування (AES) для конфіденційного шифрування файлу дозволяє ефективно захищати дані під час транспорту.

На рисунку 3.15 зображено реалізований процес завантаження.

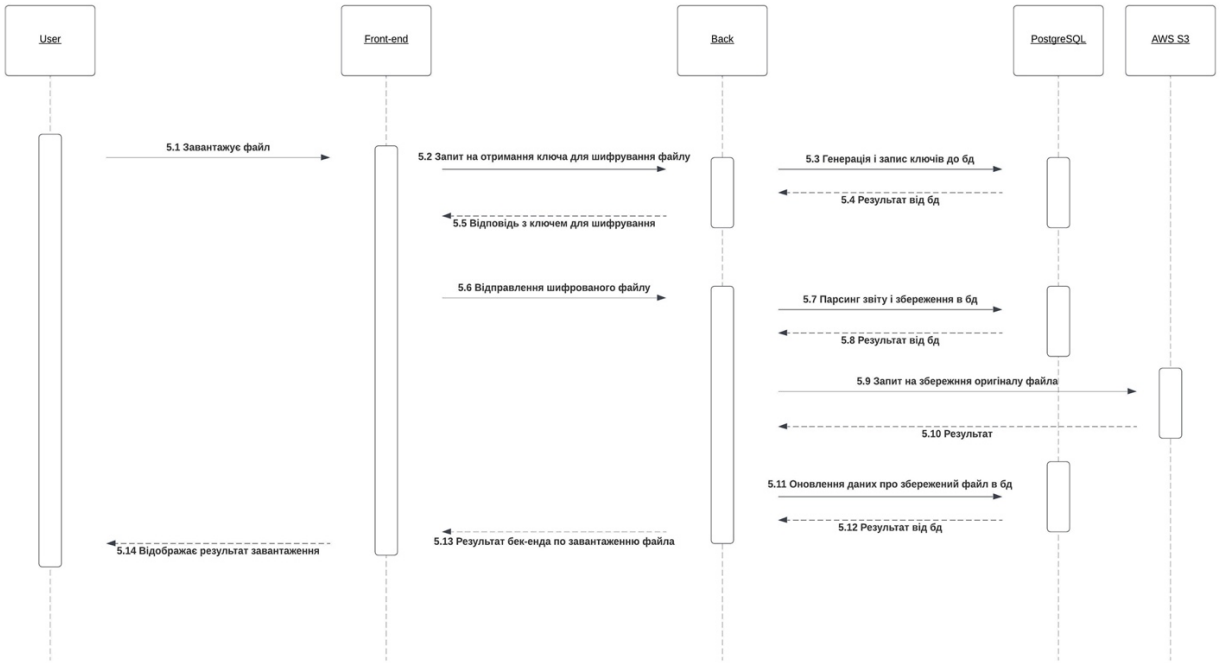


Рисунок 3.15 – UML Sequence діаграма процесу шифрування і завантаження звіту

Діаграма є частиною завантаження користувачем звітів брокера і розподілу їх по рахункам, тому нумерація кроків на рисунку була збережена.

Крок 5.1, користувач завантажує один або декілька файлів через елемент “drag and drop”.

Крок 5.2, React Client повинен згенерувати пару RSA ключів розміром 2048 бітів. Ця пара потрібна, щоб зашифрувати секретний ключ і вектор ініціалізації (елемента AES шифрування), яким буде шифруватися сам файл зі звітом брокера. RSA пару необхідно зберегти до моменту отримання AES ключа і вектора ініціалізації. Приклад коду зображену на рисунку 3.16.

```
const forge = require('node-forge');

// Generate a new 2048-bit RSA key pair
const rsaKeyPair = forge.pki.rsa.generateKeyPair(2048);

// Convert the private key to PEM format
const privateKeyPEM = forge.pki.privateKeyToPem(rsaKeyPair.privateKey);

// Convert the public key to PEM format
const publicKeyPEM = forge.pki.publicKeyToPem(rsaKeyPair.publicKey);

console.log('Private Key (PEM format):');
console.log(privateKeyPEM);

console.log('\nPublic Key (PEM format):');
console.log(publicKeyPEM);
```

Рисунок 3.16 – Генерація RSA ключів

Кроки 5.3 і 5.4, генеруємо пару ключів для шифрування і розшифрування та зберігаємо їх.

Отримуємо з React Client публічний 2048-bits RSA ключ.

Генеруємо 256-bits AES ключ та шифруємо його за допомогою публічного ключа 2048-bits RSA (див. рис. 3.17).

За цим шифруємо ключ за RSA алгоритмом і віддаємо на React Client.

```
import ...

@Slf4j
public final class AESKeyUtil {

    public static final String KEY_ALGORITHM = "AES";
    public static final Integer KEY_SIZE = 256;
    public static final String CIPHER_ALGORITHM = "AES/CBC/PKCS5Padding";

    public static SecretKey generateSecretKey() {
        try {
            var keyGenerator = KeyGenerator.getInstance(KEY_ALGORITHM);
            keyGenerator.init(KEY_SIZE);
            return keyGenerator.generateKey();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    public static byte[] generateInitializationVector() {
        byte[] ivBytes = new byte[16];
        SecureRandom random = new SecureRandom();
        random.nextBytes(ivBytes);
        return ivBytes;
    }
```

Рисунок 3.17 – Генерація AES ключа на Java

Крок 5.5, React Client отримує відповідь з ключем для шифрування. Параметри відповіді:

- id необхідний для передачі на сервер із зашифрованим файлом, щоб ідентифікувати ключ яким буде розшифровуватися файл
- encryptedBase64SecretKey – зашифровайний секретний ключ в base64 кодуванні (256 bits AES)
- encryptedBase64InitializationVector – зашифрований вектор ініціалізації в base64 кодуванні (128 bits)

Секретний ключ і вектор ініціалізації необхідно розшифрувати приватним RSA ключем з тієї пари, що була згенерована раніше.

Крок 5.6, на цьому кроці React Client розшифровує AES ключ і вектор та шифрує ними файл зі звітом.

Для розшифрування використовуємо ту ж бібліотеку node-forge. Ось приклад коду. Після цього з AES-ключем шифруємо файл як показано на рисунку 3.18.

```javascript
const fs = require('fs');
const CryptoJS = require('crypto-js');

// Define the path to the input file and the output file
const inputFile = '183391_2021-03-04 23_59_59_2023-09-07 23_59_59_all.json'; // Replace with your input file
const outputFile = '183391_2021-03-04 23_59_59_2023-09-07 23_59_59_all_encrypted.json'; // Output encrypted file

// Define your AES key and IV as Base64-encoded strings
const base64Key = '3cRXM01Vfrs3BLp+6AWdvcoMaKAzkvL0lO1qRtnihl0='; // Replace with your AES key
const base64IV = 'g8y7gWSi2Ou7CXEnBx8TLA==';   // Replace with your IV

// Read the contents of the input file
const fileData = fs.readFileSync(inputFile);

// Decode the key and IV from Base64
const key = CryptoJS.enc.Base64.parse(base64Key);
const iv = CryptoJS.enc.Base64.parse(base64IV);

console.log(fileData);
console.log(key);
console.log(iv);

// AES encryption with PKCS5Padding
const encrypted = CryptoJS.AES.encrypt(fileData.toString(), key, {
    iv: iv,
    mode: CryptoJS.mode.CBC,
    padding: CryptoJS.pad.Pkcs7,
    blockSize: 16
});

// Save the encrypted data to the output file
fs.writeFileSync(outputFile, encrypted.toString());

console.log('File encrypted and saved to', outputFile);
```

Рисунок 3.18 – Шифрування звіту на стороні клієнта

Шифрований файл відправляємо на сервер. Після чого сервер успішно розшифровує звіт та опрацьовує його відповідно до бізнес-логіки.

3.8 API

Для отримання доступу до серверного застосунку було побудовано REST API. REST (Representational State Transfer) – це архітектурний стиль для розробки веб-сервісів, який базується на принципах і властивостях, описаних у праці Роя Філдінга "Роздільне представлення стану системи" (англ. "Representational State Transfer" або просто "REST"). Основні принципи REST включають:

1. **Роздільне представлення (Separation of Concerns)**: Клієнт і сервер повинні бути незалежними один від одного. Це означає, що зміна в клієнтському інтерфейсі не повинна впливати на сервер і навпаки.

2. **Безстандартність (Statelessness)**: Кожен запит від клієнта до сервера повинен містити всю необхідну інформацію для зрозуміння і обробки запиту. Сервер не повинен зберігати стан клієнта між запитами.
3. **Представлення ресурсів (Resource Representation)**: Ресурси (наприклад, дані чи послуги) представлені у вигляді різних форматів, таких як JSON або XML. Клієнт і сервер взаємодіють через ці ресурси.
4. **Однозначність ідентифікаторів (Uniform Interface)**: Інтерфейс взаємодії між компонентами системи повинен бути однаковий, щоб спростити і робити більш передбачуваним взаємодію між різними частинами системи. Це включає чітко визначені операції (GET, POST, PUT, DELETE) та використання URI для ідентифікації ресурсів.
5. **Легкість кешування (Statelessness)**: Дані можуть бути кешовані для покращення продуктивності, але це повинно відбуватися без взаємодії з сервером, і сервер повинен повідомляти клієнта про те, чи може він використовувати кешовані дані.

REST використовується для побудови веб-сервісів, які є простими, легкими для розуміння та інтеграції. Багато веб-сервісів, які використовують архітектуру REST, використовують протокол HTTP як засіб комунікації між клієнтом і сервером.

На рисунку 3.19 зображено усі ендпоінти, що є доступними веб-застосунку.

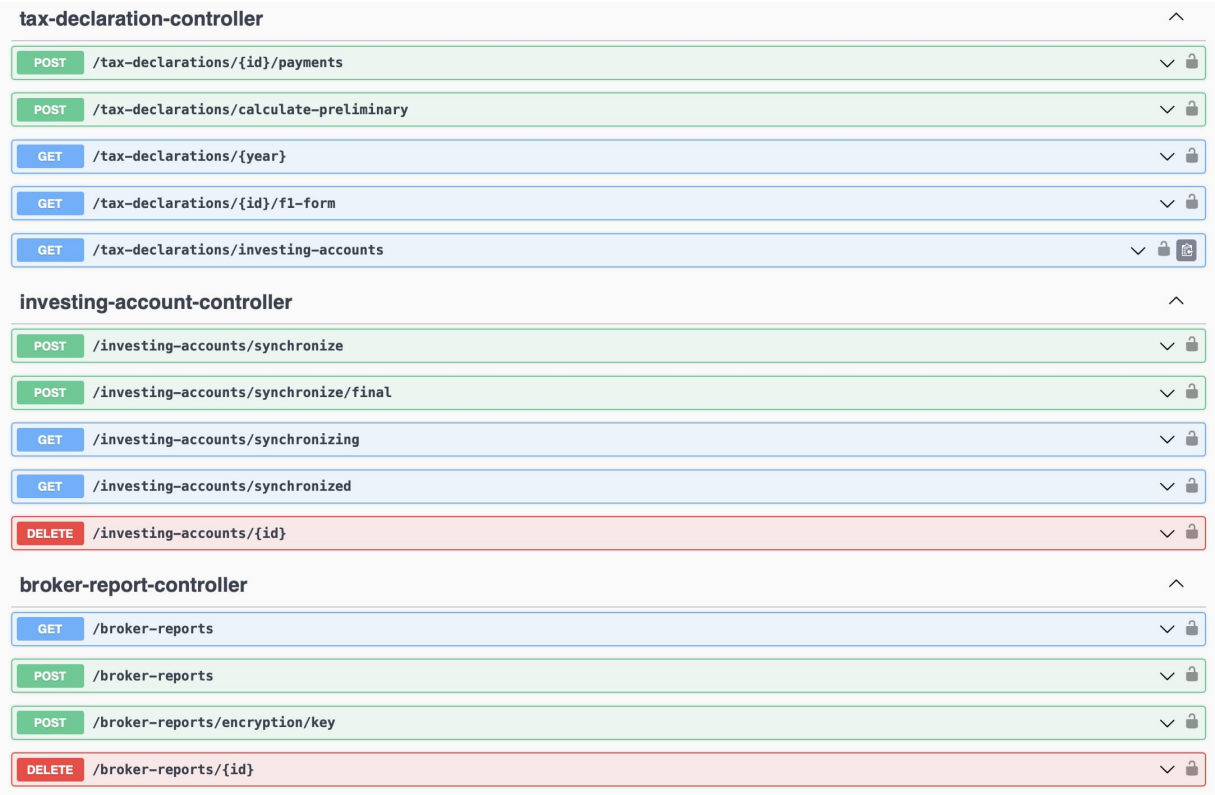


Рисунок 3.19 – API серверного застосунку

Ця візуалізація згенерована з використання стандарту OpenAPI 3.0 та бібліотеки springdoc-openapi-starter-webmvc-ui.

OpenAPI 3.0 (також відомий як Swagger 3.0) – це стандарт для опису веб-служб API. Цей стандарт дозволяє розробникам описувати функціонал API, включаючи доступні ресурси, формати обміну даними, аутентифікацію та інші важливі аспекти.

Основні характеристики OpenAPI 3.0 включають:

1. **Ясна структура YAML або JSON:** OpenAPI використовує YAML або JSON для опису API. Це забезпечує чіткість та зручність у визначенні різних аспектів API.
2. **Опис ресурсів та операцій:** Розробники можуть визначати ресурси (наприклад, URL-шляхи) та операції (GET, POST, PUT, DELETE і т. д.), які підтримуються API.
3. **Схеми даних (Data Schemas):** OpenAPI дозволяє описувати формати обміну даними між клієнтом і сервером за допомогою JSON Schema.

4. **Документація:** З OpenAPI можна автоматично генерувати документацію для API. Це полегшує розуміння та використання API різними розробниками.
5. **Підтримка різних типів безпеки:** OpenAPI дозволяє визначати та документувати різні методи безпеки, такі як OAuth, API ключі, тощо.

OpenAPI є важливим інструментом для спрощення розробки та інтеграції API. Версія 3.0 внесла ряд покращень порівняно з попередніми версіями, спростивши та розширивши можливості опису API.

`springdoc-openapi-starter-webmvc-ui` – це бібліотека для Spring Boot, яка додає підтримку OpenAPI (Swagger) для вашого веб-проекту на базі Spring Boot, що використовує підходи WebMVC. За допомогою цієї бібліотеки ви можете автоматично створювати документацію API на основі ваших контролерів та сервісів.

Давайте розглянемо кілька ключових компонентів, пов'язаних з `springdoc-openapi-starter-webmvc-ui`:

1. **SpringDoc OpenAPI Starter**: Це основний модуль, який надає підтримку OpenAPI для Spring Boot. Він інтегрується в ваш додаток і збирає інформацію про ваші контролери та їхні методи.
2. **WebMVC UI**: Цей модуль надає графічний інтерфейс користувача (UI) для перегляду та тестування документації API. Він дозволяє зручно вивчати та тестувати ваші API-ендпоінти.

Коли ви включаєте `springdoc-openapi-starter-webmvc-ui` у свій проект і налаштовуєте його, ви отримуєте доступ до Swagger UI, який надає веб-інтерфейс для візуалізації та тестування вашого API. Ви також можете переглядати та завантажувати OpenAPI-специфікації у форматі JSON або YAML.

Це допомагає полегшити розробку, тестування та документування API, забезпечуючи зручний спосіб взаємодії з API для розробників та команди, яка використовує серверний застосунок.

4 ВПРОВАДЖЕННЯ ТА ТЕСТУВАННЯ

4.1 Процес впровадження програмного продукту

В наш час клієнт будь-якого веб-сервісу очікує доступність 24/7. Саме тому сервіс потребує розвертання в хмарі. У нас є збудований серверний застосунок у форматі .jar (java archive) за допомогою Apache Maven. Це інструмент для управління проектами у сфері розробки програмного забезпечення (Project Management and Comprehension Tool). Maven допомагає у розгортанні, зборці, документуванні та управлінні залежностями в проектах Java.

Для розвертання було вибрано хостинг Digital Ocean. Він надає можливість для студентів і нових відвідувачів отримати безкоштовні кредити для тестування платформи.

Тепер перед нами постає питання: як доставити наш збудований проект на хостинг і розвернути? Існує варіант орендувати сервер, встановити на нього JVM, по мережі завантажити jar і через віддалений доступ, за допомогою консолі запустити серверний застосунок. Також ще потрібно розгорнути базу даних, але тут ми просто орендуємо готове рішення на хостингу. А як бути з самим серверним застосунком?

Docker – це платформа для розробки, доставки та запуску застосунків у відокремлених, стандартизованих контейнерах. Контейнер – це легкий та стандартизований пакет програмного забезпечення, який містить все необхідне для виконання програми, включаючи код, виконуваний файл, бібліотеки, середовище та конфігураційні файли.

Деякі ключові поняття та характеристики Docker включають:

1. **Контейнеризація:** Docker використовує концепцію контейнерів для упаковки та виконання застосунків з усією їхньою залежністю.

2. **Ізоляція:** Контейнери надають відокремлене середовище для виконання програм, що дозволяє уникнути конфліктів залежностей та конфліктів ресурсів між різними застосунками.
3. **Стандартизація:** Docker контейнери стандартизовані, що дозволяє використовувати один і той же контейнер на різних середовищах розробки, тестування та виробництва.
4. **Легкість використання:** Docker надає простий інтерфейс командного рядка та графічний інтерфейс для створення, управління та взаємодії з контейнерами.
5. **Доставка застосунків:** Docker дозволяє легко розгортати застосунки в різних середовищах без зміни коду або конфігурації.
6. **Контроль версій:** Docker дозволяє версіонувати та керувати версіями контейнерів, що полегшує розгортання та оновлення застосунків.

Docker став популярним інструментом у сфері розробки програмного забезпечення, оскільки він допомагає полегшити розгортання та управління застосунками в різних областях інформаційно-технологічного процесу.

Хостинг DigitalOcean надає дуже зручні опції, а саме оренду docker застосунків. Для цього необхідно збудувати docker образ. На рисунку 4.1 зображено конфігурацію dockerfile для нашого серверного застосунку.

```
FROM alpine:latest as customjre

RUN apk add openjdk17
RUN apk add --no-cache binutils

RUN jlink \
        --verbose \
        --add-modules ALL-MODULE-PATH \
        --strip-debug \
        --no-man-pages \
        --no-header-files \
        --compress=2 \
        --output /customjre

FROM alpine:latest

ENV JAVA_HOME=/jre
ENV PATH="${JAVA_HOME}/bin:${PATH}"

COPY --from=customjre /customjre $JAVA_HOME

ENV JAR_NAME="taxer.jar"
ENV APP_HOME="/app"
ENV JAR_DIRECTORY="target/"

WORKDIR "$APP_HOME"

COPY ./"${JAR_DIRECTORY}${JAR_NAME}" ./

EXPOSE 8080

CMD java -jar -Xms512M -Xmx512M -XX:-OmitStackTraceInFastThrow "${JAR_NAME}"
```

Рисунок 4.1 – Dockerfile

Для побудови образу використаємо наступну команду: docker build -t registry.digitalocean.com/osa-test/taxer:latest -f ./Dockerfile ./ . Після цього за допомогою команди: docker push registry.digitalocean.com/osa-test/taxer:latest – завантажимо образ на хостинг.

Після чого залишиться тільки підключити завантажений образ до docker контейнера за зручною інструкцією наданою хостингом. Тепер серверний застосунок доступний через мережу.

4.2 Тестування та оцінка ефективності

Тестування програмного забезпечення у розробці застосунків на основі Spring включає в себе різні аспекти тестування, які допомагають забезпечити якість коду та функціональність програми. Ось деякі з основних видів та способів тестування Spring-застосунків:

1. **Юніт-тестування (Unit Testing):**
   Опис: Юніт-тестування — це тестування найменших частин програми, таких як окремі методи або класи.
   Інструменти: JUnit часто використовується для юніт-тестування Spring-компонентів.
2. **Інтеграційне тестування (Integration Testing):**
   Опис: Перевірка взаємодії між різними компонентами системи.
   Інструменти: Spring Test Framework (зокрема, **@SpringBootTest** або **@WebMvcTest**) дозволяє легко налаштовувати інтеграційні тести.

Ці підходи та інструменти можуть бути комбіновані для створення комплексної стратегії тестування, яка допоможе забезпечити якість та стабільність Spring-застосунків.

В рамках нашого застосунку реалізовані обидва види тестувань (див. рис. 4.2).

```
                .toBuilder() ProfitCalculationDividendIssue...
                .incomeSumInAssetCurrency(new BigDecimal( val: "3.12"))
                .incomeSumInNationalCurrency(new BigDecimal( val: "102.69"))
                .pdfo(new BigDecimal( val: "9.24"))
                .militaryTax(new BigDecimal( val: "1.54"))
                .build();

        var expectedValue = TaxDeclarationDividendEntity.builder()
                .incomesCount(5)
                .issuersCount(2)
                .incomeTotal(new BigDecimal( val: "364.82"))
                .pdfoTotal(new BigDecimal( val: "32.83"))
                .militaryTaxTotal(new BigDecimal( val: "5.47"))
                .dividendByIssuers(List.of(
                        appleTaxCalculationDividendIssuerEntityExpected,
                        netflixTaxCalculationDividendIssuerEntityExpected
                ))
                .build();
        //then
        var result = DividendProfitCalculator.calculate(dividendOperations);
        //when
        assertThat(result) ObjectAssert<TaxDeclarationDividendEntity>
                .usingRecursiveComparison() RecursiveComparisonAssert<capture of ?>
                .isEqualTo(expectedValue);
    }

}
```

Рисунок 4.2 – Приклад тесту

Також можемо відмітити хорошу швидкодію (див. рис. 4.3).

Name	Domain	Type	Initiator	Transfer Size	Time
synchronized	king-prawn-app-rbn5l.ondigita...	fetch	fetchBaseQuery.ts:281	672 B	154ms
calculate-preliminary	king-prawn-app-rbn5l.ondigita...	fetch	fetchBaseQuery.ts:281	590 B	127ms
payments	king-prawn-app-rbn5l.ondigita...	fetch	fetchBaseQuery.ts:281	512 B	120ms
calculate-preliminary	king-prawn-app-rbn5l.ondigita...	fetch	fetchBaseQuery.ts:281	508 B	129ms
2021	king-prawn-app-rbn5l.ondigita...	fetch	fetchBaseQuery.ts:281	428 B	168ms
investing-accounts	king-prawn-app-rbn5l.ondigita...	fetch	fetchBaseQuery.ts:281	591 B	161ms
payments	king-prawn-app-rbn5l.ondigita...	fetch	fetchBaseQuery.ts:281	395 B	332ms
info	api.fondy.eu	xhr	checkout:1011:26997	1.58 KB	106ms
order	api.fondy.eu	xhr	checkout:1011:26997	884 B	80.4ms
add	api.fondy.eu	xhr	checkout:1011:26997	355 B	94.9ms
update	api.fondy.eu	xhr	checkout:1011:26997	355 B	68.8ms
cards	api.fondy.eu	xhr	checkout:1011:26997	540 B	72.0ms
mobile_pay	api.fondy.eu	xhr	checkout:1011:26997	903 B	77.0ms
update	api.fondy.eu	xhr	checkout:1011:26997	355 B	88.4ms

Рисунок 4.3 – Виклики запитів серверного застосунку з клієнта

В основному запити відпрацьовують до 300 мілісекунд. Завантаження і опрацювання звітів забирає від 1 с до 1,5 с.

ВИСНОВКИ

В рамка роботи було проведено дослідження інвестиційної діяльності українських роздрібних інвесторів на фондовому ринку. Встановлено портрет середньо статистичного інвестора, якими інвестиційними інструментами він користується.

В більшості інвестори здійснюють торгівлю на зарубіжних фондових біржах за допомогою міжнародних брокерських компаній. В Україні найпопулярнішими є:

- Freedom Broker;
- Interactive Brokers.

Проте ці брокерські компанії не є податковими резидентами в Україні, тому інвестору приходиться самостійно готувати, подавати податкову декларацію і сплачувати податки, відповідно з норм описаних в Податковому Кодексі України.

Ми детально дослідили податкову законодавчу базу України щодо оподаткування інвестиційної діяльності. Вияснили правила обчислення та податкові ставки. Важливими елементами є те, що потрібно конвертувати усі операції з іноземної валюти в національну на момент виконання операції, а також з використанням принципу FIFO.

Також Україна вступила в асоціацію про автоматичний обмін даними про фінансові рахунки і тепер зможе отримувати дані про залишки, обороти і доходи на закордонних рахунках і не тільки. Проте інвестор повинен сам здійснювати декларування і якщо він цього не зробить, то тепер з 100% гарантією отримає штраф.

З вище згаданого ми дійшли висновку, що інвестор зацікавлений в правильній подачі податкової декларації, але процес підготовки податкового звіту містить багато нюансів та рутинних операцій, що можуть призвести до помилок в розрахунках і тяганині з інспекторами податкової служби.

Отже, тема автоматизації процесів підготовки податкового звіту з інвестиційної діяльності є справді актуальною. Після цього було прийнято рішення про автоматизацію цього процесу за допомогою мови програмування Java та супутніх технологій таких як: Spring Framework.

Java завдячує своїй широкій популярності у фінансовому секторі не лише надійністю, але й високою ефективністю при обробці великого обсягу даних та забезпеченні високої продуктивності систем. Багато банківських програм та фінансових сервісів, побудованих на Java, мають високий рівень стабільності, що важливо для забезпечення безперебійності фінансових операцій.

Однією з переваг Java є також широкий екосистем, який включає в себе багато бібліотек, фреймворків і інструментів, що полегшують розробку та підтримку великих масштабів фінансових додатків. Це дозволяє командам розробників ефективно впроваджувати нові функції та швидко реагувати на зміни в індустрії.

Я, автор цієї робити наразі є інженером-програмістом в одному з українських банків-лідерів, і основною мовою програмування, що застосовується є Java.

Під час постановки мети було виділено 5 завдань рішення яких допоможе побудувати комплексний прототип серверного застосунку для вирішення задачі формування податкового звіту. Всі 5 завдань були виконані і наше рішення має наступні функціональні характеристики:

- Приймати і опрацьовувати брокерські звіти інвесторів з їхньою торговою діяльністю двох міжнарожних брокерів і для цього була розроблена відповідна модель даних;
- Усі торгові операції, що були здійснені в іноземній валюті конвертуються в національну на день виконання операції завдяки інтеграції з API Національного Банку України;
- Було розроблено програмний алгоритм і модель даних для розподілу операцій за принципом FIFO;

- Вихідний результат, який отримує інвестор – це XML-файл, що можна імпортувати в податковий кабінет в три кліки замість ручного вводу. Це зменшує кількість можливих помилок при вводі до нуля;
- Звіти, що завантажуються шифруються надійними алгоритмами RSA і AES, що надає ще більше безпеки при передачі даних по мережі інтернет.

Програмне рішення являє собою серверний застосунок на SpringFramework, що запакований у docker контейнер і розгорнуто на хостингу DigitalOcean. Серверний застосунок використовує надійну і швидкісну базу даних PostgreSQL, як сервіс хостингу. Клієнтський застосунок отримує доступ завдяки REST API та HTTP протоколу.

Як в підсумок хочу відмітити, що ця робота зменшить кількість помилок при розрахунках і заповненні податкового звіту багатьом інвесторам з 1-3 днів (в середньому за даними Сергія Твердохліба, якщо це виконувати вручну) до 5-10 хвилин користування сервісом(за умови раніше вивантажених звітів з брокерського аккаунту). Також хочу висловити подяку Сергію Твердохлібу, інвестиційному консультанту та трейдеру, за допомогу в дослідженні і аналізу прикладної частини.

Вірю, що впровадження даного рішення, як веб-платформи, сприятиме прозорості в інвестиційній діяльності на території України і як результат наповнення бюджету та зростанню економіки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стаття «Портрет українського інвестора: хто купує акції міжнародних компаній у цифровому банку Sense SuperApp» [Режим доступу: https://minfin.com.ua/ua/2022/01/24/79754893/]

2. Стаття «"Насос" інвестицій для економіки: навіщо Україні фондовий ринок» [Режим доступу: https://apostrophe.ua/ua/article/economy/finansy/2021-06-05/nasos-investitsiy-dlya-ekonomiki-zachem-ukraine-fondovyiy-ryinok/39942]

3. Стаття «Податковий кодекс України [Режим доступу: https://uk.wikipedia.org/wiki/Податковий_кодекс_України]

4. Стаття «Автоматичний обмін податковою інформацією та інші новації податкового законодавства: що змінить проект 8131» [Режим доступу: https://biz.ligazakon.net/analitycs/219372_avtomatichniy-obmn-podatkovoyu-nformatsyu-ta-nsh-novats-podatkovogo-zakonodavstva-shcho-zmnit-proekt-8131]

5. Стаття «FIFO: What the First In, First Out Method Is and How to Use It» [Режим доступу: https://www.investopedia.com/terms/f/fifo.asp]

6. Стаття «What is Java?» [Режим доступу: https://aws.amazon.com/what-is/java/]

7. Стаття «Java (programming language)» [Режим доступу: https://en.wikipedia.org/wiki/Java_(programming_language)]

8. Стаття «What is Java?» [Режим доступу: https://www.ibm.com/topics/java]

9. Стаття «What Can You Do with Java and What is it Used for?» [Режим доступу: https://exoft.net/what-can-you-do-with-java/]

10. Стаття «Популярні фреймворки Java: Spring та Hibernate» [Режим доступу: https://mate.academy/blog/java-development/java-spring-hibernate-frameworks/]

11. Офіційний сайт Spring Framework [Режим доступу: https://spring.io]

12. Стаття «Огляд Spring-компонентів. Частина 1 – Spring Boot та фреймворк» інтеграції [Режим доступу: https://habr.com/ru/articles/674858/]

13. Стаття «What is the Spring Framework?» [Режим доступу: https://www.techtarget.com/searchapparchitecture/definition/Spring-Framework]
14. Стаття «Spring Framework» [Режим доступу: https://en.wikipedia.org/wiki/Spring_Framework]
15. Стаття «What Is Java Used for in Fintech?» [Режим доступу: https://bluebirdinternational.com/what-is-java-used-for-in-fintech/]
16. Національний Банк України. API для розробників [Режим доступу: https://bank.gov.ua/ua/open-data/api-dev]
17. Open API Specification [Режим доступу: https://swagger.io/specification/]
18. Офіційний сайт хостингу DigitalOcean [Режим доступу: https://digitalocean.com]
19. Наказ «Про затвердження Методики визначення інвестиційного прибутку професійним торговцем цінними паперами при виконанні функцій податкового агента» [Режим доступу: https://zakon.rada.gov.ua/laws/show/z0100-12?fbclid=IwAR0iXU7Lp_75FJVN9PBtOAKLI0sCTDemUd9KrPgvZdihd_0gHqRvGfKLx8M#Text]
20. Стаття «Міжнародний автоматичний обмін інформацією за стандартами CRS та CbCR стає реальністю» [Режим доступу: https://kyivobl.tax.gov.ua/media-ark/news-ark/700431.html]
21. Стаття «До уваги фінансових агентів!» [Режим доступу: https://kyivobl.tax.gov.ua/media-ark/news-ark/729265.html]
22. Офіційний сайт OECD [Режим доступу: https://www.oecd.org]
23. Перелік країн, що входять до CRS [Режим доступу: https://www.oecd.org/tax/automatic-exchange/crs-implementation-and-assistance/crs-by-jurisdiction/]
24. Приклади заповнення податкової декларації про майновий стан і доходи [Режим доступу: https://tax.gov.ua/deklaratsiyna-kampaniya-2023/prikladi-zapovnennya-podatkovoi-deklaratsii]
25. Основні випадки подання податкової декларації про майновий стан і доходи

[Режим доступу: https://tax.gov.ua/deklaratsiyna-kampaniya-2023/osnovni-vipadki-podannya-podatkovoi-deklaratsii]

26. Наказ «Про затвердження форми податкової декларації про майновий стан і доходи та Інструкції щодо заповнення податкової декларації про майновий стан і доходи» [Режим доступу: https://ips.ligazakon.net/document/re27743?an=1]

ДОДАТОК А. Програмний код алгоритму розподілу операцій за принципом FIFO

Клас FIFOPackageEntity:

```
package com.osa.taxer.core.model.entity.profitcalculation;

import jakarta.persistence.*;
import lombok.*;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.time.ZonedDateTime;
import java.util.UUID;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@ToString
@EntityListeners({AuditingEntityListener.class})
@Table(name = "fifo_package")
@Entity
public class FIFOPackageEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    private UUID id;

    @CreatedDate
    @Column(updatable = false, nullable = false)
    private ZonedDateTime createDateTime;

    @OneToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name = "open_fifo_trade_id", referencedColumnName = "id",
nullable = false)
    private FIFOTradeEntity openTrade;

    @OneToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name = "close_fifo_trade_id", referencedColumnName = "id",
nullable = false)
    private FIFOTradeEntity closeTrade;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private Type type;
```

```
    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private Kind kind;

    @ToString.Exclude
    @ManyToOne
    @JoinColumn(name = "profit_calculation_stock_issuer_id",
referencedColumnName = "id", nullable = false)
    private ProfitCalculationStockIssuerEntity
profitCalculationStockIssuer;

    public enum Type {
        PHANTOM, CLOSED
    }

    public enum Kind {
        LONG, SHORT
    }

}
```

Клас FIFOTradeEntity:

```
package com.osa.taxer.core.model.entity.profitcalculation;

import com.osa.taxer.core.model.entity.ff.TradeOperationType;
import jakarta.persistence.*;
import lombok.*;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.time.ZonedDateTime;
import java.util.UUID;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@ToString
@EntityListeners({AuditingEntityListener.class})
@Table(name = "fifo_trade")
@Entity
public class FIFOTradeEntity {

    @Id
```

```
@GeneratedValue(strategy = GenerationType.UUID)
private UUID id;

@CreatedDate
@Column(updatable = false, nullable = false)
private ZonedDateTime createDateTime;

@Column(nullable = false)
private LocalDateTime dateTime;

@Enumerated(EnumType.STRING)
@Column(nullable = false)
private TradeOperationType operationType;

@Builder.Default
@Column(nullable = false)
private BigDecimal quantity = BigDecimal.ZERO;

@Builder.Default
@Column(nullable = false)
private BigDecimal sum = BigDecimal.ZERO;

@Column(nullable = false)
private String currency;

@Builder.Default
@Column(nullable = false)
private BigDecimal commission = BigDecimal.ZERO;

@Column(nullable = false)
private String commissionCurrency;

@Builder.Default
@Column(nullable = false)
private BigDecimal sumInNationalCurrency = BigDecimal.ZERO;

@Builder.Default
@Column(nullable = false)
private BigDecimal commissionInNationalCurrency = BigDecimal.ZERO;

@Builder.Default
@Column(nullable = false)
private BigDecimal sumExchangeRate = BigDecimal.ZERO;

@Builder.Default
@Column(nullable = false)
private BigDecimal commissionExchangeRate = BigDecimal.ZERO;

@Column(nullable = false)
private Boolean startBalance;

@Builder.Default
```

```
    @Column(nullable = false)
    private Boolean divided = false;

    @ToString.Exclude
    @ManyToOne
    @JoinColumn(name = "trade_id", referencedColumnName = "id")
    private TradeEntity originalTrade;

    @Transient
    private ProfitCalculationStockIssuerEntity instrument;

}
```

Клас PackageFIFOTradeService(сам алгоритм):

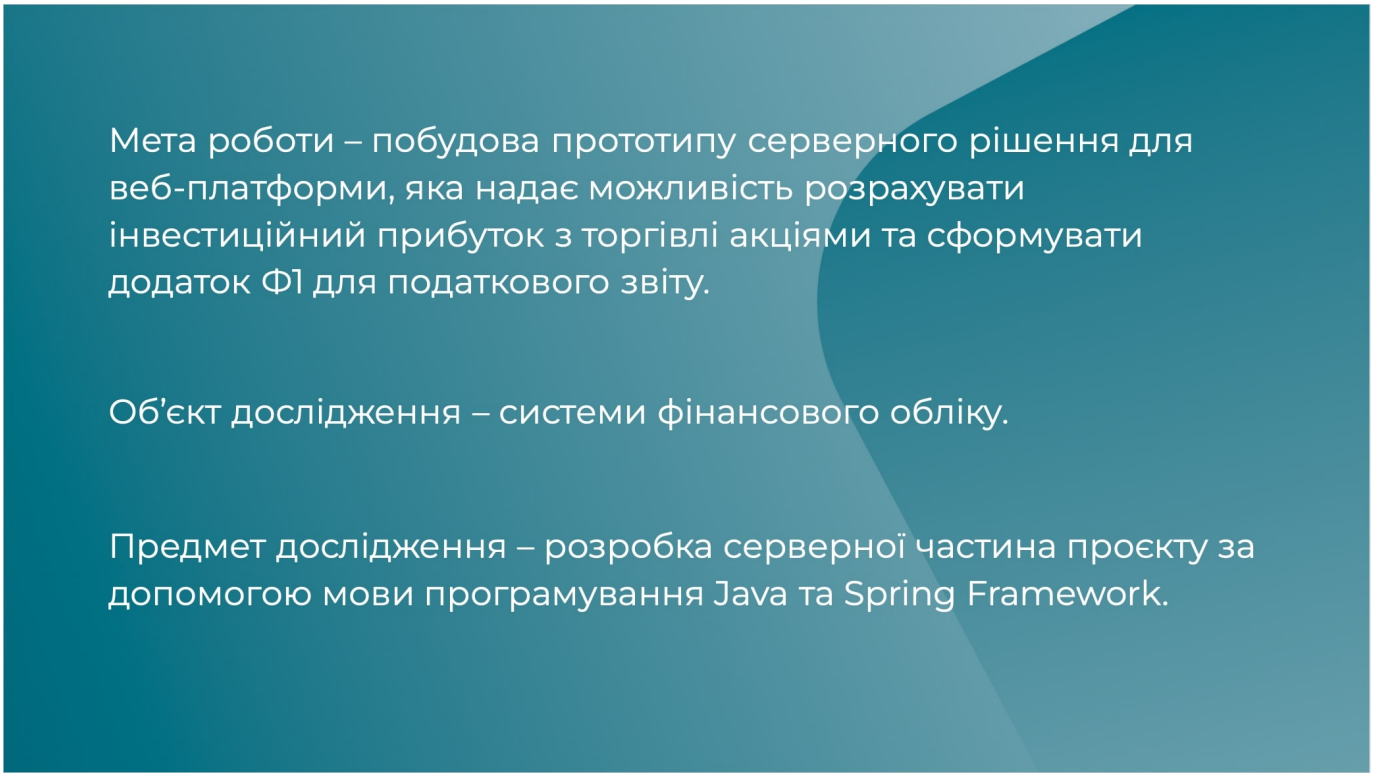
```
package com.osa.taxer.core.service.profitcalculation.stock;

import com.osa.taxer.core.model.entity.ff.TradeOperationType;
import
com.osa.taxer.core.model.entity.profitcalculation.FIFOPackageEntity;
import com.osa.taxer.core.model.entity.profitcalculation.FIFOTradeEntity;
import lombok.RequiredArgsConstructor;
import org.apache.commons.lang3.tuple.Pair;
import org.springframework.stereotype.Component;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;

import static com.osa.taxer.util.BigDecimalUtils.isBiggerThanZero;
import static com.osa.taxer.util.BigDecimalUtils.isLessThanZero;

@RequiredArgsConstructor
@Component
public class PackageFIFOTradeService {

    private final FIFOPackagerMapper mapper;

    /**
     *
     * @param tradeEntities
     * @return Pair: list of fifo-packages and not used trades (end
balance) to use in next iteration
     */
    public Pair<List<FIFOPackageEntity>, List<FIFOTradeEntity>>
divideIntoFifoPackages(List<FIFOTradeEntity> tradeEntities) {
```

```
tradeEntities.sort(Comparator.comparing(FIFOTradeEntity::getDateTime));

        var sellTradeOperations = tradeEntities.stream()
                .filter(tradeOperation ->
TradeOperationType.SELL.equals(tradeOperation.getOperationType()))
                .collect(Collectors.toCollection(ArrayList::new));

        var buyTradeOperations = tradeEntities.stream()
                .filter(tradeOperation ->
TradeOperationType.BUY.equals(tradeOperation.getOperationType()))
                .collect(Collectors.toCollection(ArrayList::new));

        List<FIFOPackageEntity> fifoPackages = new ArrayList<>();

        final int i = 0;
        while (sellTradeOperations.size() > 0 && buyTradeOperations.size()
> 0) {
            FIFOTradeEntity sell = sellTradeOperations.get(i);
            FIFOTradeEntity buy = buyTradeOperations.get(i);

            FIFOPackageEntity fifoPackage = new FIFOPackageEntity();
            fifoPackage.setType(FIFOPackageEntity.Type.CLOSED);
            //define basic trade operation
            BigDecimal quantityRemnant =
sell.getQuantity().add(buy.getQuantity());

            if (buy.getDateTime().isBefore(sell.getDateTime())) {
                fifoPackage.setKind(FIFOPackageEntity.Kind.LONG);
                if (buy.getStartBalance()) {
                    fifoPackage.setType(FIFOPackageEntity.Type.PHANTOM);
                }
                //long buy is open, sell is close
                if (BigDecimal.ZERO.compareTo(quantityRemnant) == 0) {
                    fifoPackage.setOpenTrade(buy);
                    fifoPackage.setCloseTrade(sell);

                    sellTradeOperations.remove(sell);
                    buyTradeOperations.remove(buy);
                } else if (isLessThanZero(quantityRemnant)) {
                    FIFOTradeEntity closeToFifoPackage =
divideOperation(sell, buy, quantityRemnant);

                    fifoPackage.setOpenTrade(buy);
                    fifoPackage.setCloseTrade(closeToFifoPackage);

                    buyTradeOperations.remove(buy);
                } else if (isBiggerThanZero(quantityRemnant)) {
                    FIFOTradeEntity openToFifoPackage =
divideOperation(buy, sell, quantityRemnant);

                    fifoPackage.setOpenTrade(openToFifoPackage);
```

```
                    fifoPackage.setCloseTrade(sell);

                    sellTradeOperations.remove(sell);
                }
            } else {
                fifoPackage.setKind(FIFOPackageEntity.Kind.SHORT);
                if (sell.getStartBalance()) {
                    fifoPackage.setType(FIFOPackageEntity.Type.PHANTOM);
                }
                //short sell is open, buy is close
                if (BigDecimal.ZERO.compareTo(quantityRemnant) == 0) {
                    fifoPackage.setOpenTrade(sell);
                    fifoPackage.setCloseTrade(buy);

                    sellTradeOperations.remove(sell);
                    buyTradeOperations.remove(buy);
                } else if (isBiggerThanZero(quantityRemnant)) {
                    FIFOTradeEntity closeToFifoPackage =
divideOperation(buy, sell, quantityRemnant);

                    fifoPackage.setOpenTrade(sell);
                    fifoPackage.setCloseTrade(closeToFifoPackage);

                    sellTradeOperations.remove(sell);
                } else if (isLessThanZero(quantityRemnant)) {
                    FIFOTradeEntity openToFifoPackage =
divideOperation(sell, buy, quantityRemnant);

                    fifoPackage.setOpenTrade(openToFifoPackage);
                    fifoPackage.setCloseTrade(buy);

                    buyTradeOperations.remove(buy);
                }
            }
            fifoPackages.add(fifoPackage);
        }

        //end balances
        if (buyTradeOperations.size() > 0) {
            return Pair.of(fifoPackages, buyTradeOperations);
        } else if (sellTradeOperations.size() > 0) {
            return Pair.of(fifoPackages, sellTradeOperations);
        }
        return Pair.of(fifoPackages, List.of());
    }

    private FIFOTradeEntity divideOperation(FIFOTradeEntity
operationToDivide, FIFOTradeEntity other, BigDecimal quantityRemnant) {
        FIFOTradeEntity fifo =
mapper.toFIFOTradeEntity(operationToDivide);
        operationToDivide.setDivided(true);
        fifo.setDivided(true);
```

```
        var quantityToFifoPackage = other.getQuantity().negate();
        fifo.setQuantity(quantityToFifoPackage);

        var coefToFifoPackage = BigDecimal.ONE
                .divide(operationToDivide.getQuantity().abs(), 2,
RoundingMode.HALF_DOWN)
                .multiply(quantityToFifoPackage.abs());


fifo.setSum(operationToDivide.getSum().multiply(coefToFifoPackage));

fifo.setCommission(operationToDivide.getCommission().multiply(coefToFifoPa
ckage));

fifo.setSumInNationalCurrency(operationToDivide.getSumInNationalCurrency()
.multiply(coefToFifoPackage));

fifo.setCommissionInNationalCurrency(operationToDivide.getCommissionInNati
onalCurrency().multiply(coefToFifoPackage));
        // remnant
        operationToDivide.setQuantity(quantityRemnant);
        var coefBackToList = BigDecimal.ONE.subtract(coefToFifoPackage);


operationToDivide.setSum(operationToDivide.getSum().multiply(coefBackToLis
t));

operationToDivide.setCommission(operationToDivide.getCommission().multiply
(coefBackToList));

operationToDivide.setSumInNationalCurrency(operationToDivide.getSumInNatio
nalCurrency().multiply(coefBackToList));

operationToDivide.setCommissionInNationalCurrency(operationToDivide.getCom
missionInNationalCurrency().multiply(coefBackToList));
        return fifo;
    }

}
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

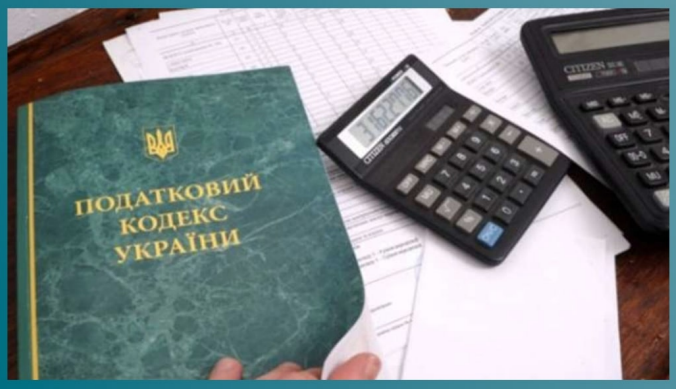


Інвестиційне середовище в Україні

FREEDOM BROKER

InteractiveBrokers

Законодавча податкова база

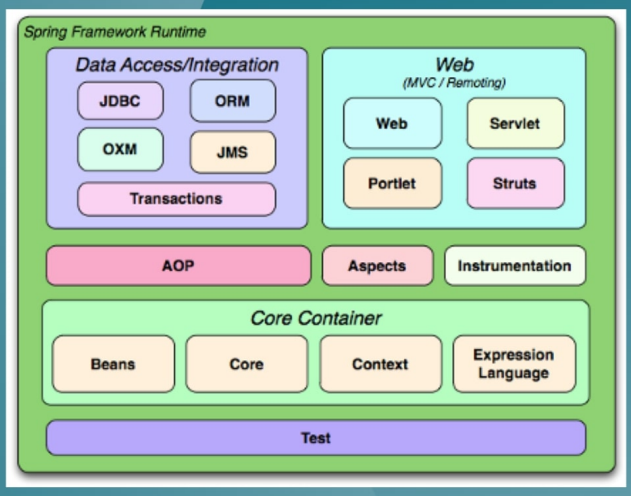


OECD

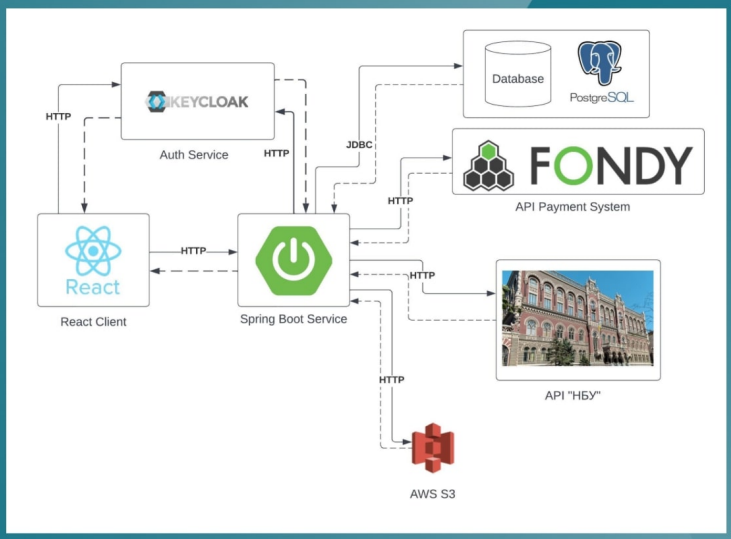
Процес ведення податкової звітності

1. Завантажити брокерський звіт
2. Перевести всі операції в національну валюту
3. Розподілити угоди за принципом FIFO
4. Виконати розрахунок інвестиційного доходу
5. Створити ключ КЕП(кваліфікаційний електронний ключ)
6. Заповнити декларацію «Про майновий стан і доходи» та Форму Ф1
7. Доповнити супровідним листом
8. Підписати і відправити

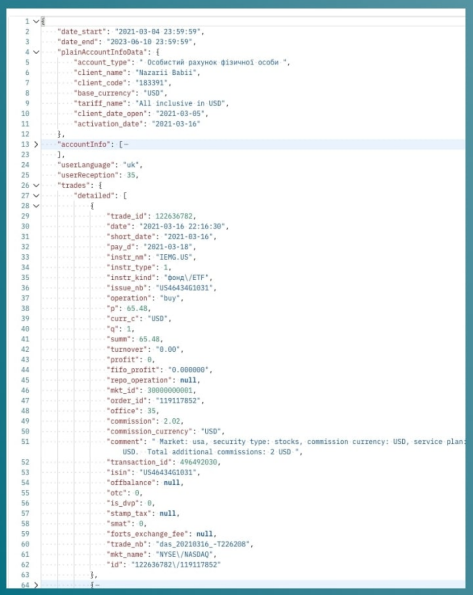
Огляд Java і Spring Framework



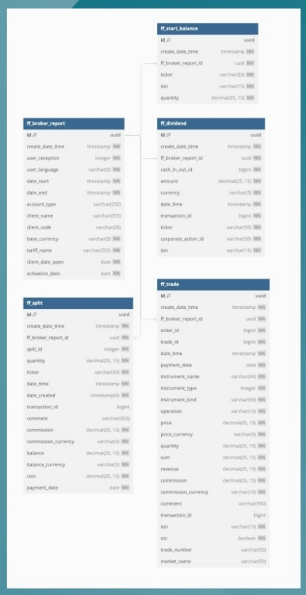
Архітектура



Модель даних для збереження операцій торгової активності інвестора зі звітів брокерів



```
{
  "date_start": "2021-03-04 23:59:59",
  "date_end": "2023-06-10 23:59:59",
  "plainAccountInfoData": {
    "account_type": " Особистий рахунок фізичної особи ",
    "client_name": "Nazarii Babii",
    "client_code": "183391",
    "base_currency": "USD",
    "tariff_name": "All inclusive in USD",
    "client_date_open": "2021-03-05",
    "activation_date": "2021-03-16"
  },
  "accountInfo": {…
  },
  "userLanguage": "uk",
  "userReception": 35,
  "trades": {
    "detailed": [
      {
        "trade_id": 122636782,
        "date": "2021-03-16 22:16:30",
        "short_date": "2021-03-16",
        "pay_d": "2021-03-18",
        "instr_nm": "IEMG.US",
        "instr_type": 1,
        "instr_kind": "фонд\/ETF",
        "issue_nb": "US4643461031",
        "operation": "buy",
        "p": 65.48,
        "curr_c": "USD",
        "q": 1,
        "summ": 65.48,
        "turnover": "0.00",
        "profit": 0,
        "fifo_profit": "0.000000",
        "repo_operation": null,
        "mkt_id": 30000000001,
        "order_id": "119117852",
        "office": 35,
        "commission": 2.02,
        "commission_currency": "USD",
        "comment": " Market: usa, security type: stocks, commission currency: USD, service plan:
          USD.  Total additional commissions: 2 USD ",
        "transaction_id": 496492030,
        "isin": "US4643461031",
        "offbalance": null,
        "otc": 0,
        "is_dvp": 0,
        "stamp_tax": null,
        "smat": 0,
        "forts_exchange_fee": null,
        "trade_nb": "das_20210316_-7226208",
        "mkt_name": "NYSE\/NASDAQ",
        "id": "122636782\/119117852"
      },
      {…
```

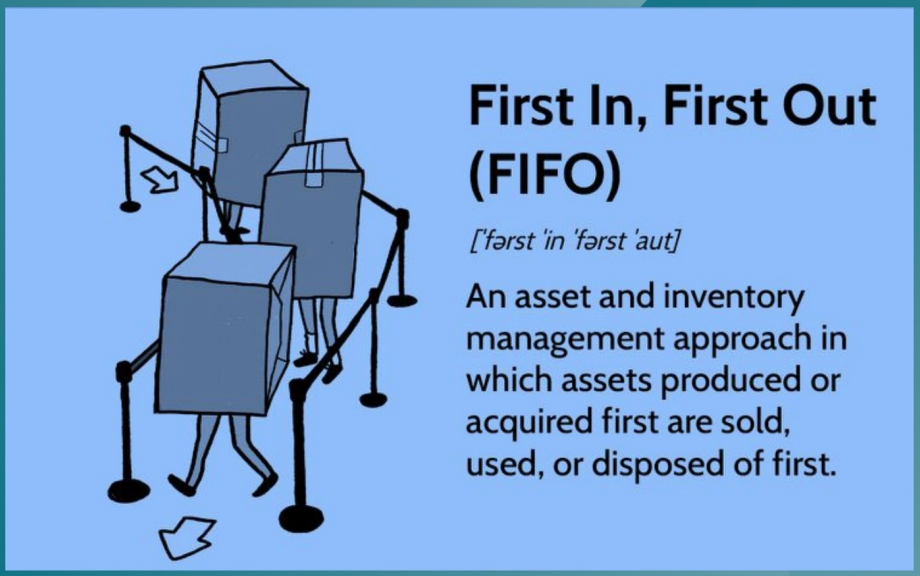


Збагачення даними операцій з брокерських звітів актуальними обмінними курсами валют НБУ

```java
import ...

@RequiredArgsConstructor
@Component
public class NbuConnector {
//    https://bank.gov.ua/NBU_Exchange/exchange_site?start=20220115&end=20230315&valcode=usd&sort=exchangedate&order=desc&json

    @Qualifier("nationalBankRestTemplate")
    private final RestTemplate restTemplate;


    public ExchangeRate[] getExchangeRates(LocalDate start, LocalDate end, String currency) {

        final var url = UriComponentsBuilder.fromHttpUrl("https://bank.gov.ua/NBU_Exchange/exchange_site")
                .queryParam( name: "start", DateUtils.changeFormat(start.toString(), DateUtils.yyyyMMdd))
                .queryParam( name: "end", DateUtils.changeFormat(end.toString(), DateUtils.yyyyMMdd))
                .queryParam( name: "valcode", currency.toLowerCase())
                .queryParam( name: "sort",  ...values: "exchangedate")
                .queryParam( name: "order",  ...values: "desc")
                .queryParam( name: "json",  ...values: true)
                .build().toString();

        return restTemplate.getForEntity(url, ExchangeRate[].class).getBody();
    }

}
```

```java
@RequiredArgsConstructor
@Component
public class NbuAdapter implements FetchExchageRateNbuPort {

    private final NbuConnector nbuConnector;

    private final HashMap<String, HashMap<LocalDate, ExchangeRate>> hashMap = new HashMap<>();

    private final List<String> currencies = List.of("USD", "EUR", "HKD", "GBP");

    @PostConstruct
    public void init() {
        currencies.forEach(c -> {
            final var exchangeRates =
                    nbuConnector.getExchangeRates(LocalDate.of( year: 2019,  month: 1,  dayOfMonth: 1), LocalDate.now(), c);
            final HashMap<LocalDate, ExchangeRate> ratesByDays = new HashMap<>();
            for (ExchangeRate rate : exchangeRates) {
                ratesByDays.put(rate.getExchangeDate(), rate);
            }
            hashMap.put(c, ratesByDays);
            try {
                Thread.sleep( millis: 300);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        });
    }

    @Override
    public ExchangeRate fetch(LocalDate date, String fromCurrency) {
        if (hashMap.containsKey(fromCurrency)) {
            final var ratesByDays = hashMap.get(fromCurrency);
            if (ratesByDays.containsKey(date))
                return ratesByDays.get(date);
        }
        return makeRequest(date, fromCurrency);
    }
```

Алгоритм розрахунку інвестиційного прибутку. FIFO

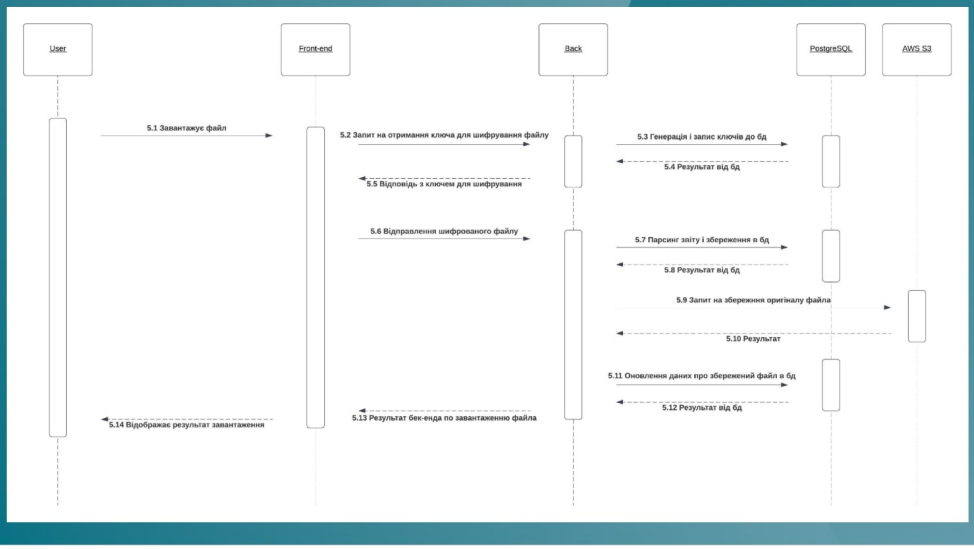


Механізм генерації вихідних файлів для зручного імпортування в податковий кабінет

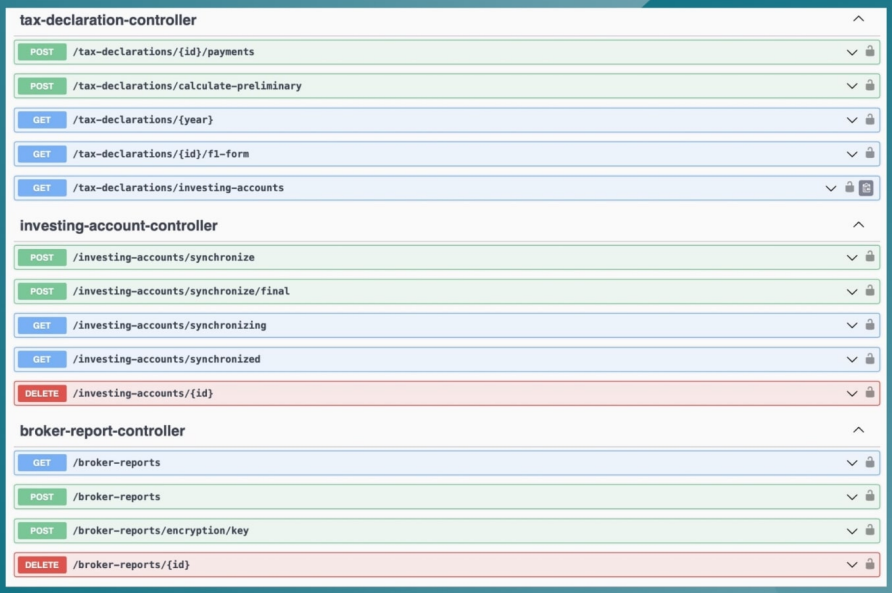
```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:annotation>
        <xs:documentation>Податкова декларація про майновий стан і доходи (річна).
            Наказ Міністерства фінансів України 02 жовтня 2015 року № 859
(у редакції наказу Міністерства фінансів України від 17.05.2022 року № 143)</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="common_types.xsd"/>
    <xs:element name="DECLAR" type="DeclarContent"/>
    <xs:complexType name="DeclarContent">
        <xs:sequence>
            <xs:element name="DECLARHEAD" type="DHead"/>
            <xs:element name="DECLARBODY" type="DBody">
                <xs:unique name="UT1RXXXXG2">
                    <xs:selector xpath="T1RXXXXG2"/>
                    <xs:field xpath="@ROWNUM"/>
                </xs:unique>
                <xs:unique name="UT1RXXXXG3S">
                    <xs:selector xpath="T1RXXXXG3S"/>
                    <xs:field xpath="@ROWNUM"/>
                </xs:unique>
                <xs:unique name="UT1RXXXXG4">
                    <xs:selector xpath="T1RXXXXG4"/>
                    <xs:field xpath="@ROWNUM"/>
                </xs:unique>
                <xs:unique name="UT1RXXXXG5">
                    <xs:selector xpath="T1RXXXXG5"/>
                    <xs:field xpath="@ROWNUM"/>
                </xs:unique>
                <xs:unique name="UT1RXXXXG6">
                    <xs:selector xpath="T1RXXXXG6"/>
                    <xs:field xpath="@ROWNUM"/>
                </xs:unique>
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <DECLAR xsi:schemaLocation="F0121213.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <DECLARBODY>
      <HZY>0</HZY>
      <T1RXXXXG2 ROWNUM="1">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="2">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="3">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="4">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="5">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="6">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="7">2</T1RXXXXG2>
      <T1RXXXXG2 ROWNUM="8">2</T1RXXXXG2>
      <T1RXXXXG3S ROWNUM="1">S_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="2">TOST_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="3">TASK_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="4">PAY_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="5">MQ_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="6">DOCS_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="7">HLTH_IPO.US</T1RXXXXG3S>
      <T1RXXXXG3S ROWNUM="8">DLO_IPO.US</T1RXXXXG3S>
      <T1RXXXXG4 ROWNUM="1">1302.27</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="2">-120.20</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="3">2896.19</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="4">107.35</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="5">-43.42</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="6">1527.20</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="7">-114.15</T1RXXXXG4>
      <T1RXXXXG4 ROWNUM="8">3712.96</T1RXXXXG4>
      <T1RXXXXG5 ROWNUM="1">-3017.75</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="2">-1126.64</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="3">-1988.45</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="4">-688.87</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="5">-1548.37</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="6">-756.04</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="7">-895.12</T1RXXXXG5>
      <T1RXXXXG5 ROWNUM="8">-1836.75</T1RXXXXG5>
      <T1RXXXXG6 ROWNUM="1">1302.27</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="2">-120.20</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="3">2896.19</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="4">107.35</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="5">-43.42</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="6">1527.20</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="7">-114.15</T1RXXXXG6>
      <T1RXXXXG6 ROWNUM="8">3712.96</T1RXXXXG6>
    </DECLARBODY>
  </DECLAR>
```

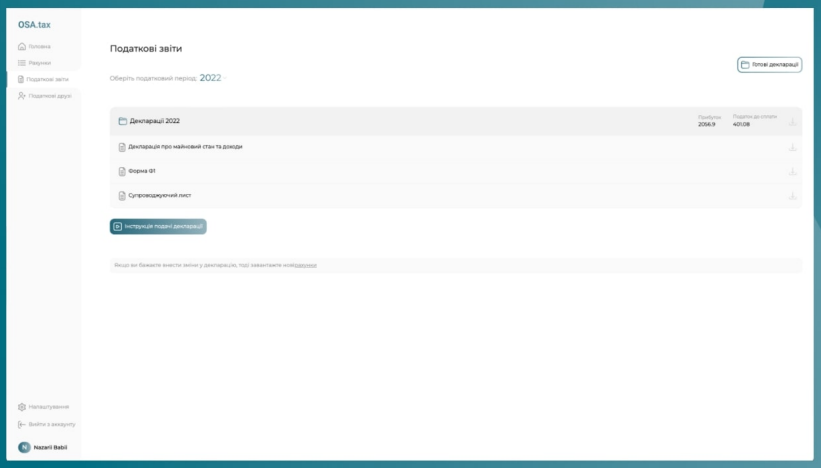
Шифрування звітів брокерів та вихідних файлів для забезпечення безпеки передачі даних між серверним і клієнтським застосунками



API



Демонстрація роботи зі сторони клієнтського застосунку



Висновки

- Побудовано прототип серверного рішення для веб-платформи, яка надає можливість розрахувати інвестиційний прибуток з торгівлі акціями та сформувати додаток Ф1 для податкового звіту
- Клієнтський застосунок отримує доступ завдяки сучасному REST API та HTTP протоколу.
- Кінцевий користувач, а саме український інвестор може розрахувати і сформувати податковий звіт за 5-10 хвилин. Вручну це відбувається від 1-3 днів і більше.

> Вірю, що впровадження даного рішення, як веб-платформи, сприятиме прозорості в інвестиційній діяльності на території України і як результат наповнення бюджету та зростанню економіки.