Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ПЕРСОНАЛЬНОГО МЕНЕДЖЕРА ДОХОДІВ ТА ВИТРАТ
(тема)

Виконав:
здобувач 4 року навчання,

групи ІТІНФ-21-1

Тетюшев І. А.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О. В.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Тетюшеву Ігорю Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка персонального менеджера доходів та витрат

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 02 червня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотеки та фремверк для розробки.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів використання вебзастосунку персонального менеджера доходів та витрат.
2. Огляд технологій для розробки вебзастосунку.
3. Розробка вебзастосунку.
4. Дослідження розробленого вебзастосунку.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)
Схема взаємозв'язку між компонентами MVC (структурна схема) та реалізована в застосунку структура MVC (функціональна схема), зображення реалізації технологій при розробці вебзастосунку.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тітова О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 54 с., 4 табл., 28 рис., 33 джерел.

ПЕРСОНАЛЬНІ ФІНАНСИ, ВЕБЗАСТОСУНОК, ОБЛІК ФІНАНСІВ, ФІНАНСОВИЙ МЕНЕДЖМЕНТ, SPRING BOOT, SPRING MVC.

Об’єктом роботи є технології та засоби автоматизації обліку персональних фінансів за допомогою вебзастосунків.

Метою роботи є розробка функціонального вебзастосунку персонального менеджера доходів та витрат, який дозволяє користувачам ефективно контролювати власні фінанси, аналізувати структуру витрат і планувати бюджет.

У роботі розглянуто основні етапи розробки: поставка задачі, вибір архітектури вебзастосунку, проектування бази даних, реалізація серверної та клієнтської частини, а також тестування функціональності

У результаті роботи здійснена програмна реалізація персонального менеджера доходів та витрат.

PERSONAL FINANCE, WEB APPLICATION, FINANCIAL ACCOUNTING, FINANCIAL MANAGEMENT, SPRING BOOT, SPRING MVC.

The object of this work is the technologies and tools for automating personal finance management using web applications.

The purpose of the work is to develop a functional web application for a personal income and expense manager that allows users to effectively control their finances, analyze their spending structure, and plan their budget.

The main stages of development: task definition, selection of web application architecture, database design, implementation of the server and client parts, as well as functionality testing was performed in the work.

As a result of the work, a software implementation of the personal income and expense manager was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

VPN – Virtual Private Network (віртуальна приватна мережа)

URL – Uniform Resource Locator (єдиний вказівник на ресурс)

AI – Artificial Intelligence (штучний інтелект)

PWA – Progressive Web App (прогресивний вебзастосунок)

MVC – Model-View-Controller (модель-представлення-контролер)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

HTML – Hypertext Markup Language (стандартизована мова розмітки документів для перегляду вебсторінок у браузері)

API – Application Programming Interface (інтерфейс програмування застосунків)

ORM – Object-Relational Mapping (об'єктно-реляційне відображення)

SQL – Structured Query Language (структурована мова запитів)

БД – база даних

СУБД – система управління базами даних

SDK – Software Development Kit (комплект для розробки програмного забезпечення)

IDE – Integrated Development Environment (інтегроване середовище розробки)

ВСТУП

В сучасних умовах економічної нестабільності важливою складовою фінансової грамотності є вміння контролювати свої доходи та витрати. Розробка вебзастосунку персонального менеджера для ведення фінансового обліку дозволяє користувачам ефективно планувати бюджет, відслідковувати витрати та доходи, що сприяє підвищенню фінансової дисципліни.

Розвиток інформаційних технологій має великий вплив на управління особистими фінансами, оскільки технології допомагають зробити процеси більш зручними, доступними та ефективними. Це реалізується через додатки для бюджетування, завдяки яким користувачі можуть відслідковувати свої витрати, створювати бюджети та аналізувати фінансові звички в реальному часі.

Використання інформаційних технологій не тільки робить фінансове управління доступнішим, а й дозволяє більш ефективно контролювати і планувати свої витрати та доходи.

На сьогодні існує численні продукти програмного забезпечення, які вирішують проблему управління особистими фінансами. Проте більшість з них не набули масового визнання та широкого використання через складний і не інтуїтивний інтерфейс, перевантаження функціоналом чи високу вартість.

Актуальність роботи полягає у створенні вебзастосунку, який надає користувачам можливість зручно та ефективно управляти своїми фінансами, забезпечуючи інтуїтивно зрозумілий інтерфейс та функціональність для фінансового планування та відстеження доходів і витрат, які сприяють аналізу фінансових результатів.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ ВИКОРИСТАННЯ ВЕБЗАСТОСУНКУ ПЕРСОНАЛЬНОГО МЕНЕДЖЕРА ДОХОДІВ ТА ВИТРАТ

1.1 Значення фінансового менеджменту у розробці застосунку фінансового менеджера

Невід’ємною умовою становлення та розвитку успішної людини є фінансова грамотність.

Фінансова грамотність складається з набору компонентів (складання бюджету, інвестування, кредитування, оподаткування, управління власними фінансами) та навичок, які дозволяють людині отримувати знання щодо ефективного управління грошовими коштами та боргами. Фінансова безпека забезпечується за рахунок збалансованого поєднання фінансових компонентів для зміцнення та збільшення інвестицій та заощаджень при одночасному зменшені кредитів та боргів [1, 2].

Особливо важливими є навички ефективного управляти грошима, уникання боргів і досягнення фінансової стабільності. Обізнаність в фінансових питаннях дозволяє людині планувати бюджет і контролювати витрати; розуміти основи і значення таких категорій як кредити, депозити, податки, інвестиції; приймати обґрунтовані фінансові рішення; заощаджувати для реалізації цілей та планів, захищатися від шахрайства та фінансових ризиків.

Відсутність зручного інструментарію, який би дозволив людині наочно бачити та аналізувати проведені операції з метою визначення фінансового результату та планування транзакцій в подальшому, приводить до відсутності системного фінансового менеджменту власних фінансів. А як результат гроші витрачаються необачно, знижується платоспроможність людини та можливість досягнення фінансових цілей.

Фінансовий менеджмент розглядається як система управління формуванням, розподілом і використанням фінансових ресурсів суб'єкта господарювання та оптимізації обороту його грошових коштів [3].

Фінансовий менеджмент є однією з ключових складових при розробці інформаційних систем, орієнтованих на управління особистими фінансами. В умовах цифровізації економіки та зростання фінансової самостійності населення, попит на ефективні інструменти для обліку доходів і витрат постійно зростає. Саме тому інтеграція принципів фінансового менеджменту на етапі проєктування та реалізації таких застосунків є необхідною умовою їхньої практичної цінності.

Фінансовий менеджмент забезпечує основу для структурування облікової системи застосунку. Користувачу необхідно мати змогу класифікувати власні доходи та витрати за категоріями, аналізувати грошові потоки, встановлювати бюджети, а також контролювати їх дотримання. Для цього застосовуються концепції управлінського обліку, касового методу обліку та бюджетування [3].

Важливим аспектом є планування особистих фінансів. Завдяки застосуванню фінансових інструментів, таких як аналіз доходів та витрат, прогнозування залишку коштів, розрахунок заощаджень чи інвестицій, користувач отримує реальні механізми для прийняття раціональних фінансових рішень. Ці рішення базується на загальновизнаних підходах у фінансовому менеджменті – наприклад, принципі 50/30/20, оцінці платоспроможності тощо [3,4].

Ефективний застосунок повинен забезпечувати аналітичні функції, що дозволяють користувачу бачити динаміку витрат, відхилення від запланованих бюджетів, оцінювати структуру доходів, та на основі цього приймати обґрунтовані рішення. Фінансовий менеджмент у цьому випадку виступає джерелом для розробки ключових фінансових індикаторів, які формують основу таких звітів.

Крім того, фінансовий менеджмент впливає на проєктування системи безпеки. Захист особистих фінансових даних, резервне копіювання, шифрування,

багаторівневий доступ – усе це пов’язане з принципами управління фінансовими ризиками, що є важливою складовою сучасного фінансового менеджменту [3].

Знання фінансового менеджменту також впливає на вибір моделі монетизації застосунку. Залежно від цільової аудиторії, структури функціоналу та рівня конкуренції, розробник може обрати модель передплати, рекламу або партнерську інтеграцію. Кожна з цих моделей потребує фінансового аналізу доцільності та потенційного прибутку.

Отже, інтеграція принципів фінансового менеджменту у розробку застосунку персонального менеджера доходів і витрат не лише підвищує його функціональність, а й сприяє формуванню цінності продукту для кінцевого користувача, забезпечуючи ефективність управління особистими фінансами.

1.2 Дослідження актуальності розробки застосунку менеджера доходів та витрат

В сучасних умовах стрімкого розвитку цифрових технологій та зростання рівня фінансової свідомості населення набувають актуальність інструменти для ефективного управління особистими фінансами.

Застосунки виконують роль не лише інструментів, але й стають для користувачів помічниками в досягнені цілей. Застосунки допомагають керувати складними завданнями та знаходити оптимальні шляхи для виконання рутинних задач. Застосунки персонального фінансового менеджменту дозволяють користувачам контролювати власні доходи та витрати, планувати бюджет, формувати фінансові цілі та приймати обґрунтовані рішення на основі фінансової аналітики. При цьому для прийняття правильних і своєчасних рішень, інформація повинна бути актуальною, повною та зрозумілою.

Важливою перевагою подібних застосунків є доступність. Вони можуть використовуватися з будь-якого пристрою. Єдиною умовою є підключення до

мережі Інтернет. Це робить їх надзвичайно зручними для використання в будь-якому місці в будь-який час.

За результатами досліджень міленіали та представники покоління Z – це найактивніші користувачі застосунків мобільного банкінгу. 99% представників покоління Z і 98% представників покоління Y використовують застосунки мобільного банкінгу для широкого спектру задач [5]. Попри те, що подібні застосунки вкрай популярні серед молоді, 86,5% представників покоління X та 69,5% представників покоління бумерів також регулярно користуються всіма сучасними зручностями, які надають застосунки для планування бюджету чи економії грошових коштів [6].

Незважаючи на те, що банківські застосунки є корисним інструментарієм, вони не є універсальним засобом управління власними фінансами, оскільки обмежені інформацією щодо стану рахунків тільки в одному банку. Якщо користувач має декілька рахунків в різних банківських установах, щоб мати можливість цілісного управління своїми коштами, доцільно використовувати застосунки менеджера доходів та витрат.

Однією з ключових причин актуальності розробки застосунку менеджера доходів та витрат є зростаюча складність фінансових операцій у повсякденному житті. Особливо це важливо в умовах швидкості, зручності та ефективності використання ресурсів.

Наразі кожен з нас має багато карток та рахунків відкритих в різних банках. Частина розрахунків проводиться людиною заплановано, частина витрат є результатом імпульсу чи зумовлена терміновою необхідністю. Крім того, є такі операції як перерозподіл між рахунками, знаття готівки чи поповнення рахунку, що відображають рух грошових коштів без фактичної зміни фінансового результату.

Розширення спектру платіжних інструментів (банківські картки, електронні гаманці, онлайн-банкінг), різноманіття підписок, кредитів і мікроплатежів ускладнюють ручне ведення обліку витрат. В умовах економічної

нестабільності, інфляції та зниження купівельної спроможності людини зростає потреба у систематичному контролі особистих фінансів.

Таким чином, розробка зручного, функціонального та інтуїтивно зрозумілого застосунку для управління особистими фінансами є актуальною відповіддю на реальні потреби сучасного користувача. Такий продукт сприятиме не лише підвищенню фінансової дисципліни, але й формуванню стабільних економічних рішень на рівні побуту, що загалом позитивно впливає на фінансову систему суспільства.

1.3 Аналіз застосунків конкурентів

Ідея розробки застосунку персонального менеджера доходів і витрат не є новою. Останні роки кількість подібних застосунків збільшується. Вони відрізняються за функціональністю, зручністю використання, ціновою політикою. Різноманіття функціональних можливостей застосунків спонукає користувачів віднайти найзручніший само для себе.

Функціональні відмінності застосунків для управління доходами та витратами зазвичай полягають у рівні автоматизації, аналітики, синхронізації інформації з банківськими установами, у наявності додаткових функцій.

Доцільно виділити основні категорії відмінностей існуючих застосунків:

– метод введення даних: ручне ведення, коли користувач самостійно записує всі доходи та витрати; автоматичне імпортування транзакцій, коли застосунок підключається до банківських рахунків та карток і автоматично фіксує їх;

– аналітика та візуалізація: базова аналітика передбачає формування простих графіків доходів, витрат, категорій; розширена аналітика (побудова звітів, діаграм, трендів по місяцях, прогнозів);

– бюджетування: фіксовані бюджети (встановлення ліміту на категорії витрат); динамічне бюджетування (бюджети адаптуються до змін у доходах або

залишках);

– мультивалютність та підтримка рахунків: підтримка декількох валют та ведення кількох рахунків, включно з готівкою, банківськими, електронними;

– цілі та заощадження: можливість створення фінансових цілей (накопичення, резерви); нагадування та відстеження прогресу;

– сумісність і синхронізація: кросплатформеність (синхронізація між смартфоном, планшетом та ПК) та спільне використання (підтримка кількох користувачів, наприклад, для сімейного бюджету);

– конфіденційність і безпека: захист паролем, біометрією; локальне або хмарне збереження даних.

Вибір застосунку доволі суб'єктивний процес. Застосунок може виявитися незручним у використані. Десь недостатньо категорій, десь незрозуміла логіка розподілу доходів та витрат. Деякі графічно перенавантажені одночасно всіма категоріями на екрані. Деякі передбачають складний підхід до заповнення сум чи рознесення по категоріях.

Використання застосунку передбачає необхідність постійної роботи з ним. Це має бути щоденний процес, який триватиме не один місяць. І якщо щось в застосунку не влаштовує чи дратує користувача, то ймовірність його подальшого використання мала. Отже, застосунок не зможе виконувати покладені функції аналізу та систематизації інформації про доходи та витрати. Адже, тільки довгострокове використання подібних застосунків може дати потрібний результат: дисциплінує користувача, забезпечить зрозумілою та системної інформацією та запропонує підказки по оптимізації фінансових потоків.

Для того щоб обрати найкращий для власного використання застосунок менеджера доходів та витрат, слід дотримуватися такої схеми:

Крок 1. Визначити, для чого потрібен застосунок (аналіз витрат, контроль заощаджень).

Крок 2. Перевірити функціональні можливості. Звернути увагу на наявність таких функцій: синхронізація з банківськими рахунками, підтримка

різних валют, автоматичне категоризування витрат, інтеграція з іншими платформами, створення бюджету.

Крок 3. Оцінити простоту використання. Застосунок має містити зрозумілий інтерфейс, з інтуїтивно зрозумілим використанням ключових функцій.

Крок 4. Безпека та захист даних. Надавати перевагу варто застосункам які пропонують: захист паролем, двофазну авторизацію [7-10].

Розглянемо деякі найбільш популярні застосунки.

Застосунок Money Manager Expense & Budget (рис. 1.1).

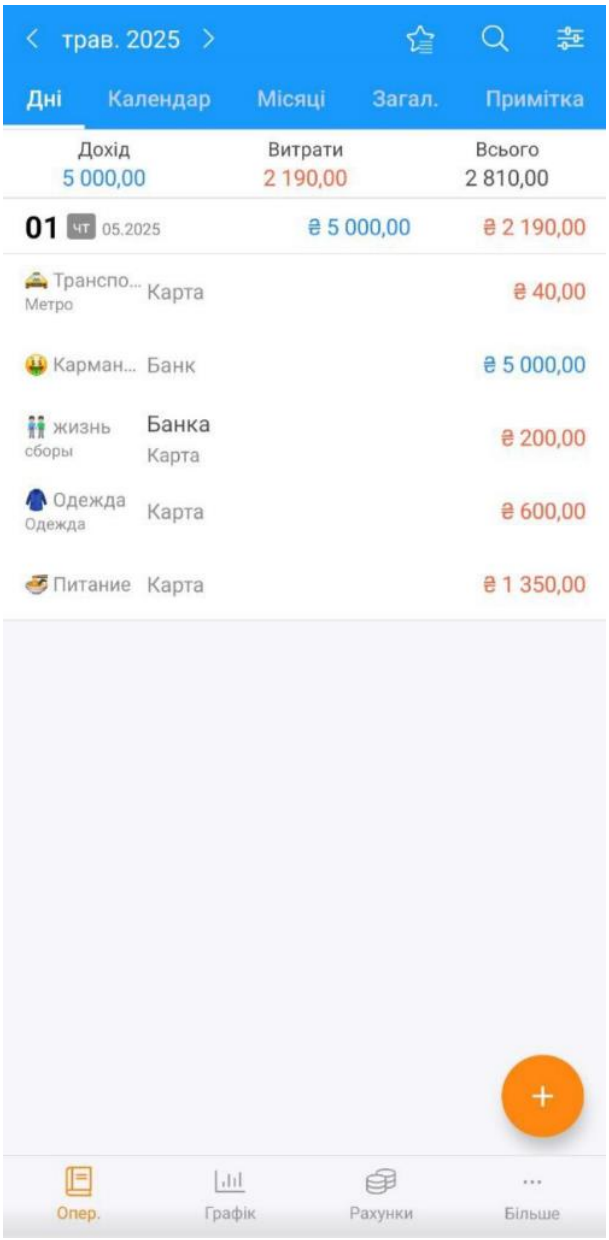

Рисунок 1.1 – Застосунок «Money Manager Expense & Budget»

Застосунок відображає доход, витрати за день, тиждень, місяць. Має зрозумілий інтерфейс, який можна налаштувати: стилі та режими відображання, зміна кольорів доходів, витрат. Має категорії та підкатегорії, які можна редагувати чи створювати свої. Графік у вигляді діаграми систематизує дані доходів та витрат за підкатегоріями. Є можливість експортувати дані в Excel. Щоб додати транзакцію треба вказати суму, категорію, рахунок, за якою була проведена операція, можна додати фото чека або квитанції. Всі транзакції можна переглядати у вигляді календаря, що дозволяє побачити в які дні користувач витрачав більше або менше. Окремо можна фільтрувати всі операції за типом, категорією, рахунком.

Інформацію з програми можна вивести на екран як віджет. Найчастіші операції можна віднести до «Обране» та проводити автоматично. Для кредитних платежів можна налаштувати «Нагадування» та відслідковувати суми майбутньої оплати.

Усі витрати фіксуються на головній сторінці у вигляді списку (якщо витрат багато може бути незручно). Основним недоліком є велика кількість реклами, яку можливо прибрати якщо вибрати платну версію.

Розглянемо один із популярних застосунків – Monefy (рис. 1.2). Він має зручний інтерфейс. Усі категорії витрат представлені на головному екрані у вигляді картинок. Додавання витрат та доходів здійснюється легко, в одну дію. Є можливість мати примітки до кожної операції. Розподіл витрат за категоріями представлено у вигляді діаграми. Можна переглядати всі фінансові операції за окремою категорією чи приміткою за вказаний період. Можна переглядати бюджети за день, тиждень, місяць, рік, весь час або обрати конкретний інтервал часу чи дату.

Можливість додавати власні категорії, синхронізуватися з кількома пристроями, додавати різні валюти. Обрати українську мову можливо лише в платній версії. Неможливо додати дані за попередній період (день). Якщо не встигли чи забули ввести дані за минулий день, всі нові записи зараховуються в

день внесення. Не підтримує планування регулярних платежів. Відсутня синхронізації з банками. Безкоштовна версія має вкрай обмежений функціонал.

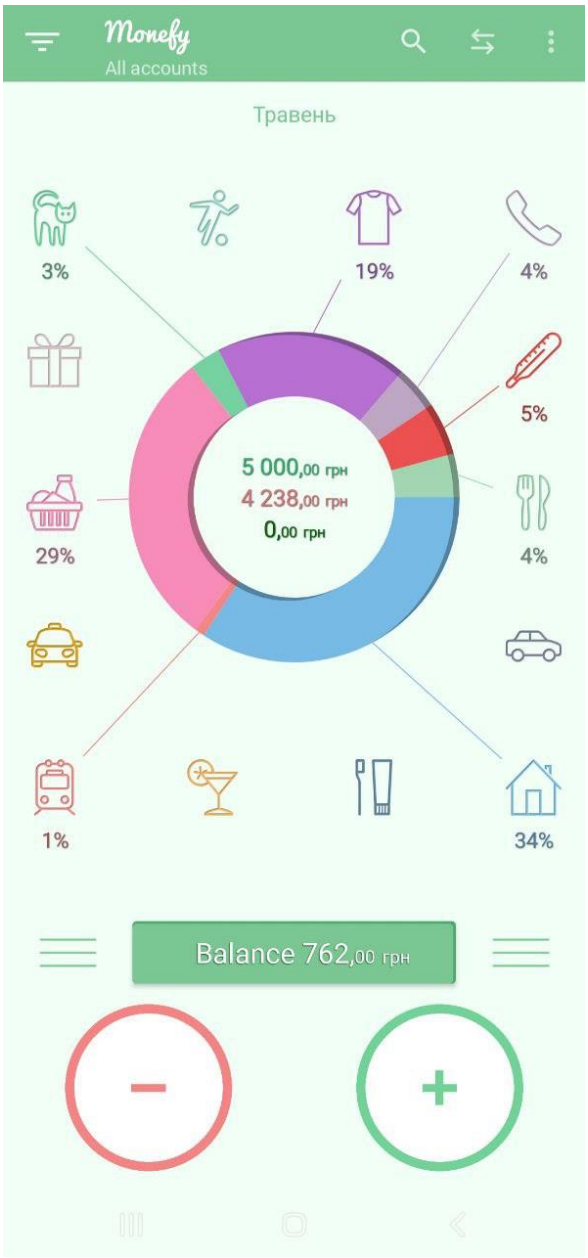

Рисунок 1.2 – Застосунок «Monefy»

Застосунок «Saldo» (рис. 1.3) надає безкоштовний доступ для українських користувачів (в термін до скасування військового стану). Передбачає можливість вести облік як особистих, так і бізнес-фінансів. Дає можливість отримувати дані по транзакції у Монобанку у реальному часі. Декларується, що незабаром буде інтеграція з ПриватБанк, Ощадбанк. Є можливість додавати доходи і витрати,

створювати категорії та підкатегорії вручну. Деталізована статистика. Є можливість запрошувати інших людей для ведення бюджету з налаштуванням прав доступу.

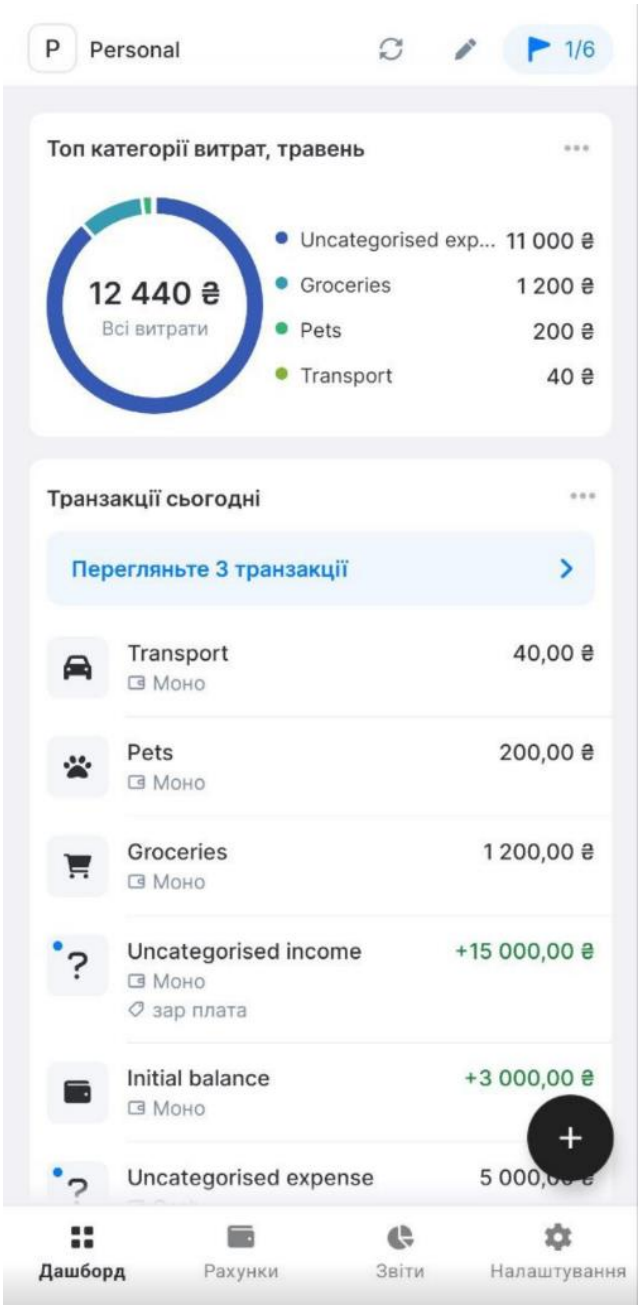

Рисунок 1.3 – Застосунок «Saldo»

Серед недоліків можна виділити обмежену інтеграцію з банками та обмеження в функціональності для користувачів за межами України.

Порівняння характеристик застосунків наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння застосунків менеджерів доходів та витрат

Характеристики	**Money Manager**	**Monefy**	**Saldo**
Рейтинг	4,9	4,1	4,7
Платформа	Android та OIS	Android та OIS	Android та OIS
Вебверсія	Є	Немає	Немає
Кількість завантажень	понад 10 млн	понад 5 млн	понад 10 тис.

Порівняння функціональності застосунків наведено у таблиці 1.2.

Таблиця 1.2 – Порівняння застосунків менеджерів доходів та витрат

Функція застосунку	**Money Manager**	**Monefy**	**Saldo**
Підтримка кількох рахунків	Є	Немає	Є
Підтримка кількох валют	Є	Є	Є
Автоматичний імпорт транзакції	Немає	Немає	Є
Планування бюджету	Є	Немає	Є
Нагадування про платежі	Є	Немає	Є
Безкоштовна версія	Є	Є	Є

Обережно треба використовувати, а краще виключити застосування російських застосунків. Адже є ризик використання РФ персональних даних користувачів, а використання застосунків означитиме спонсорування держави-терориста. Це такі застосунки, як Moneon, Money Manager, Expenses OK, MoneyFlow, CoinKeeper, MoneyPro, MoneyBox, Simpler Expense.

На основі проведеного аналізу актуальності використання застосунку персонального менеджера доходів та витрат, а також аналізу функціональних можливостей різноманіття існуючих застосунків, що представлено на ринку, було прийнято рішення створення власного вебзастосунку.

1.4 Постановка задачі

У сучасну цифрову епоху ефективне управління особистими фінансами стає все більш актуальним. Багато людей стикаються з труднощами у веденні обліку доходів та витрат, аналізі фінансових звичок та плануванні бюджету. Існуючі рішення часто не відповідають вимогам користувачів через обмежену функціональність, складність у користуванні або відсутність персоналізації. Тож, незважаючи на численні застосунки, що представлені на ринку, розробка вебзастосунку персонального менеджера доходів та витрат є актуальною.

Об’єктом роботи є технології та засоби автоматизації обліку персональних фінансів за допомогою вебзастосунків.

Метою роботи є розробка функціонального вебзастосунку персонального менеджера доходів та витрат, який дозволяє користувачам ефективно контролювати власні фінанси, аналізувати структуру витрат і планувати бюджет.

Для досягнення мети необхідно вирішити наступні задачі:

– проаналізувати існуючі рішення для управління особистими фінансами;

– сформувати вимоги до системи та визначити ключові функції застосунку;

– розробити архітектуру вебзастосунку;

– спроектувати реляційну базу даних для збереження фінансової інформації користувача;

– реалізувати серверну та клієнтську частини застосунку;

– провести тестування функціональності для перевірки надійності та зручності використання.

2 ТЕХНОЛОГІЇ ТА МЕТОДИ РОЗРОБКИ ВЕБЗАСТОСУНКУ

2.1 Загальна архітектура системи

Вебзастосунок представляє собою застосунок, до якого можна отримати доступ за допомогою веббраузера. В порівнянні з власне застосунками, вважається, що вебзастосунки легше розробляти та оновлювати. Час завантаження вебзастосунків швидший, що перш за все пов’язано з особливістю архітектури вебзастосунків [14].

Вебзастосунки та застосунки (мобільні чи десктопні) мають різні характеристики, особливості використання та переваги [15].

Вебзастосунки характеризуються тим, що запускаються через браузер, не потребують встановлення на пристрій, працюють через інтернет (за виключенням PWA – це прогресивні вебзастосунки з офлайн-можливостями), оновлюються централізовано на сервері. Це зумовлює переваги: доступ з будь-якого пристрою з браузером, менше залежить від платформи (Windows, iOS, Android тощо), легке розгортання та оновлення (табл. 2.1).

Таблиця 2.1 – Функціональні відмінності вебзастосунків та застосунків

Вебзастосунок	**Застосунок (мобільний чи десктопний)**
Кросплатформенність	Особлива платформа
Установка не потрібна	Потрібна установка
Зручність використання в автономному режимі	Висока продуктивність
Низька вартість	Висока вартість
Низьке споживання даних	Вимагає оновлень

Застосунки (мобільні чи десктопні) характеризуються тим, що встановлюються на пристрій, працюють як онлайн, так і офлайн (залежить від функціоналу), створюються під конкретну платформу або з використанням кросплатформених технологій. Застосунки мають переваги: повний доступ до апаратних можливостей пристрою, швидша і стабільніша робота, можливість працювати без інтернету. Недоліками застосунків є те, що вони потребують інсталяції, оновлення треба проходити через магазини додатків, залежать від платформи (iOS, Android, Windows) (табл. 2.1).

Архітектура вебзастосунку – це структура, яка визначає взаємодію між різними компонентами вебзастосунку, їх структуру та спосіб взаємодії. Вона описує, як розподіляються завдання та функції між частинами програми, з метою забезпечення ефективної роботи та простоти масштабування, а також сприяє підтримці кода та забезпечує його легке оновлення в майбутньому [16].

2.2 Компоненти системи

Серверна частина вебзастосунку реалізована з використанням фреймворку Spring Boot.

Spring Boot – це фреймворк на Java, який дозволяє швидко та легко створювати автономні, готові до продуктивного використання додатки на основі Spring Framework. Spring Boot значно спрощує налаштування та конфігурацію додатків, дозволяючи зосередитися на розробці бізнес-логіки замість складного налаштування середовища [17].

Основні особливості Spring Boot наведено нижче.

Автоматична конфігурація: Spring Boot автоматично налаштовує багато параметрів додатка на основі залежностей, що додаються, і не вимагає великої кількості конфігураційних файлів.

Вбудовані сервери: Spring Boot дозволяє запускати додатки з вбудованими серверами, такими як Tomcat, Jetty чи Undertow, без необхідності розгортати

додаток на зовнішньому сервері.

Простота використання: Мінімальна кількість налаштувань і конфігурацій, що дозволяє зосередитися на функціональності додатка.

Готовність до продуктивного використання: Включає інструменти для моніторингу, управління здоров'ям додатка, а також інтеграцію з системами управління конфігурацією.

Мікросервіси: Spring Boot широко використовується для створення мікросервісів завдяки своїй легкості та можливості швидкого налаштування.

Вебзастосунок реалізовується за допомогою шаблону Model–View–Controller (MVC).

MVC (Model-View-Controller) – це архітектурний шаблон проєктування програмного забезпечення, який широко використовується в розробці вебзастосунків. Він забезпечує розділення логіки представлення, обробки запитів та бізнес-логіки. Підхід дозволяє логічно розділити відповідальність між компонентами (рис. 2.1)

Model (модель) – відповідає за представлення даних застосунку; роботу з базою даних (зчитування, збереження, оновлення, видалення); бізнес-логіку та правила обробки даних.

View (представлення) – відповідає за відображення інформації користувачу; формування інтерфейсу, який бачить користувач; не містить логіки обробки даних, лише їх виведення.

Controller (контролер) – відповідає за отримання запитів від користувача; взаємодію з моделлю (отримання/оновлення даних); вибір відповідного представлення (View) для відображення результатів.

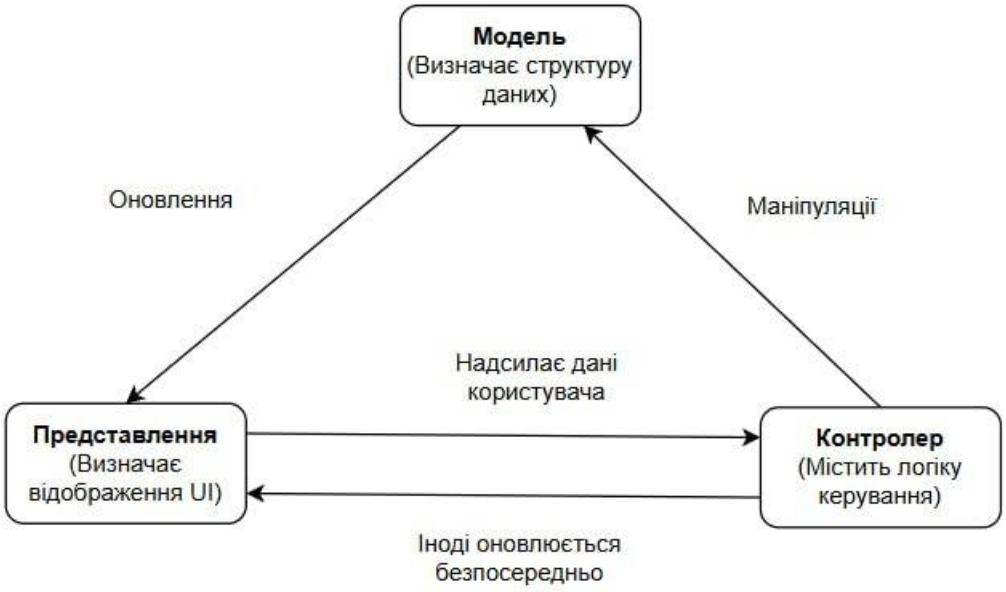

Рисунок 2.1 – Взаємозв’язок між компонентами MVC

Разом MVC працює таким чином:

Крок 1. Користувач виконує дію (наприклад, відкриває сторінку).

Крок 2. Controller отримує запит, обробляє його.

Крок 3. Controller звертається до Model, щоб отримати або змінити дані.

Крок 4. Після обробки, Controller передає дані до View.

Крок 5. View формує інтерфейс, який бачить користувач.

Серед переваг MVC можна виділити: розділення обов’язків, що дозволяє легше розвивати та тестувати застосунок; масштабованість, завдяки якої легко додавати нові функції та повторне використання коду, оскільки одна модель може використовуватись у кількох представленнях.

Компоненти в застосунку, що розроблявся реалізовано наступним чином (реалізація Spring Web MVC) [18].

Перший компонент – Model (модель), а саме класи, які представляють дані застосунку (пакет ua.nure.finance.model) (рис. 2.2).

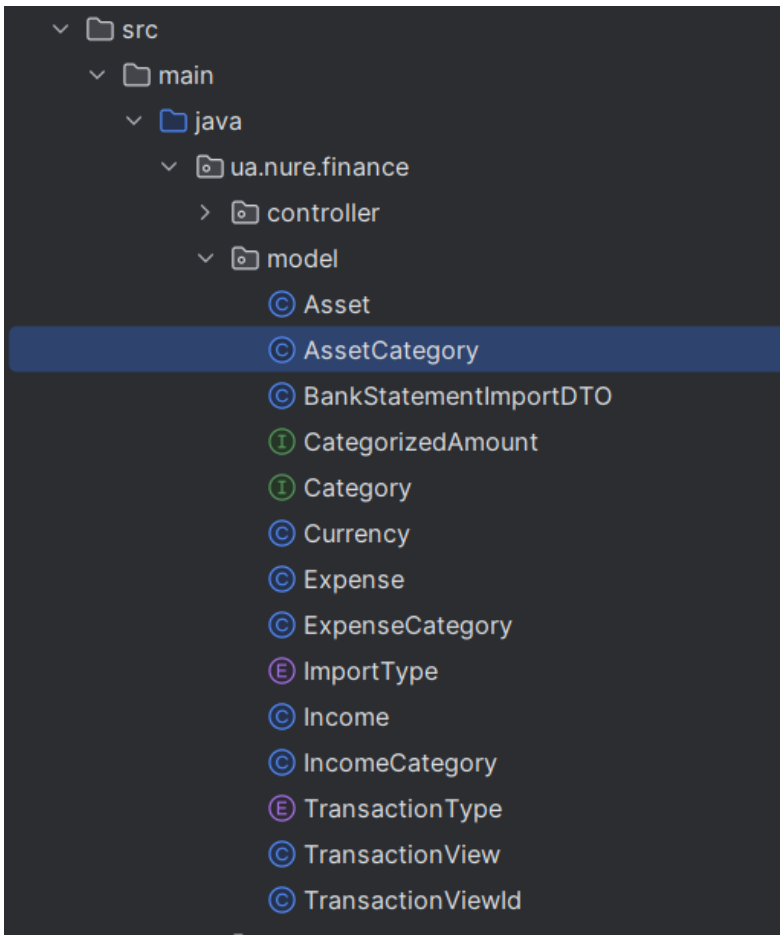

Рисунок 2.2 – Моделі розробленого застосунку

Використовуються анотації jakarta.persistence.Entity та jakarta.persistence.Table для означення entity та таблиці бази них відповідно (рис. 2.3).

```
@Entity  ihortetiushev *
@Table(name = "assets")
@Getter
@Setter
public class Asset {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    ....
```

Рисунок 2.3 – Приклад Entity-класу в розробленому застосунку

Наступний компонент – View (представлення): у Spring Boot використовувався Thymeleaf в якості шаблонізатора для представлення HTML-сторінок, який безпосередньо інтегрується зі Spring MVC

Thymeleaf – це сучасний серверний механізм шаблонів Java як для веб-, так і для автономних середовищ. Його головною метою є привнесення природних шаблонів у робочий процес розробки HTML, який може коректно відображатися в браузерах, а також працювати як статичні прототипи. Thymeleaf дозволяє створювати динамічні HTML-сторінки та вставляти дані з Java-класів у шаблони [19].

Також використано HTML5, CSS3, jQuery, Google charts та базові можливості Boots trap для забезпечення адаптивного дизайну [20].

jQuery – це швидка та багатофункціональна бібліотека JavaScript. Вона значно спрощує такі речі, як переміщення та маніпулювання HTML-документами, обробка події, завдяки простому у використанні API, який працює в багатьох браузерах [21].

Google charts – це інструменти обробки даних для діаграм, що є простими в використанні, з широким набором параметрів, мають крос-браузерну сумісність, не потребує плагінів [22].

Thymeleaf шаблони у розробленому вебзастосунку розташовані у папці src/main/resources/templates (рис. 2.4).

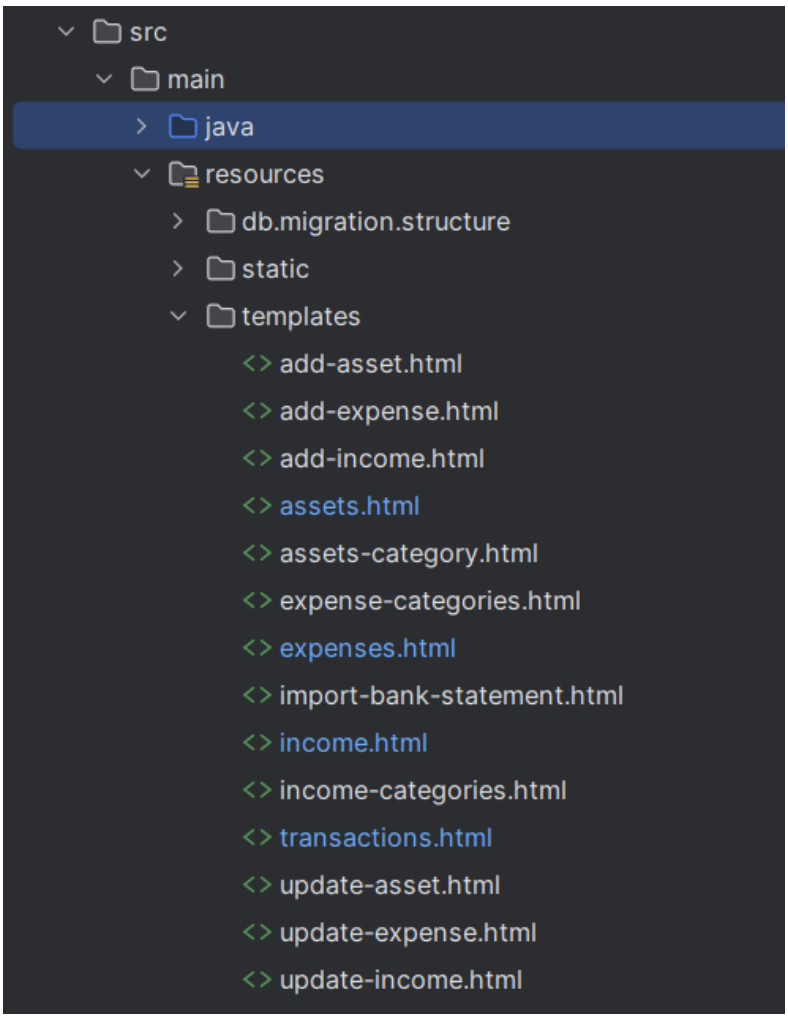

Рисунок 2.4 – Thymeleaf шаблони у розробленому вебзастосунку

Третій компонент – Controller (контролер), він обробляє HTTP запити та зв’язує модель із представленням (пакет ua.nure.finance.controller) (рис. 2.5).

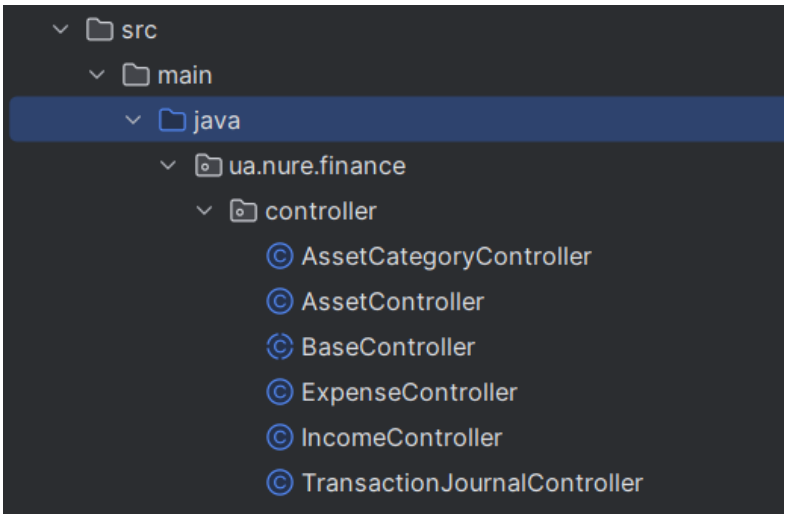

Рисунок 2.5 – Контролери в розробленому вебзастосунку

Для позначення контролера використовується анотація org.springframework.stereotype.Controller (рис. 2.6).

```
@Controller  1 usage  ihortetiushev
@RequestMapping("/assets-categories")
public class AssetCategoryController {

    private final AssetCategoryRepository categoryRepository;  8 usages

    public AssetCategoryController(AssetCategoryRepository categoryRepository) {  no usages  ihortetiushev
        this.categoryRepository = categoryRepository;
    }

    // Show all categories
    @GetMapping  no usages  ihortetiushev
    public String listCategories(Model model) {
        model.addAttribute( attributeName: "categories", categoryRepository.findAll());
        model.addAttribute( attributeName: "category", new AssetCategory()); // for add form
        return "assets-category"; // Thymeleaf template name
```

Рисунок 2.6 – Приклад контролеру розробленого вебзастосунку

Таким чином, маємо наступну структуру проєкту Spring Boot MVC на рисунку 2.7.

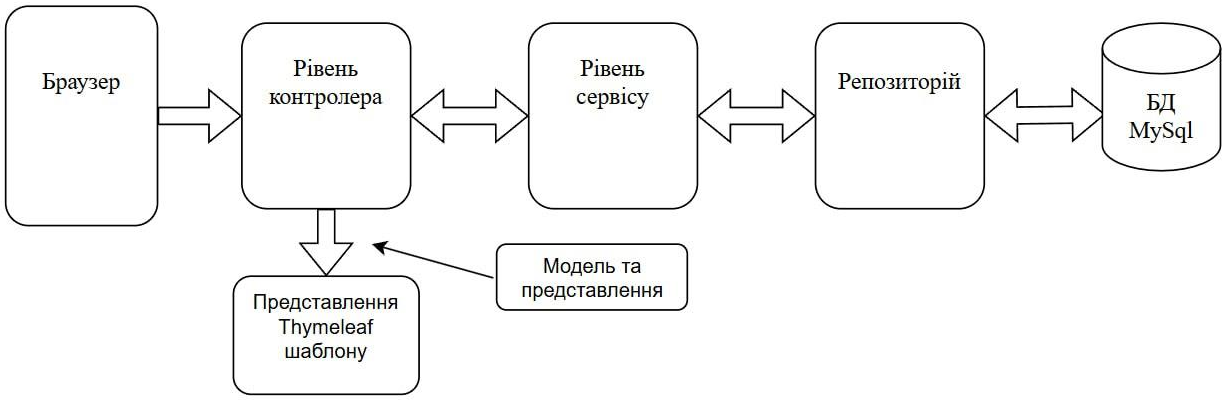

Рисунок 2.7 – Реалізована в застосунку структура MVC в Spring Boot

Spring Data JPA використовується для доступу до бази даних за допомогою об’єктно-реляційного відображення (ORM). Spring Data JPA представляє собою підмодуль Spring Data, який спрощує роботу з базами даних через Java Persistence API (JPA). Фактично це обгортка навколо JPA (наприклад, Hibernate), яка

автоматизує створення репозиторіїв та запитів. Його основною метою є максимальне зменшення кількості шаблонного коду при взаємодії з реляційною базою даних [23].

В розробленому застосунку репозиторії JPA розташовані в пакеті ua.nure.finance.repository (рис. 2.8).

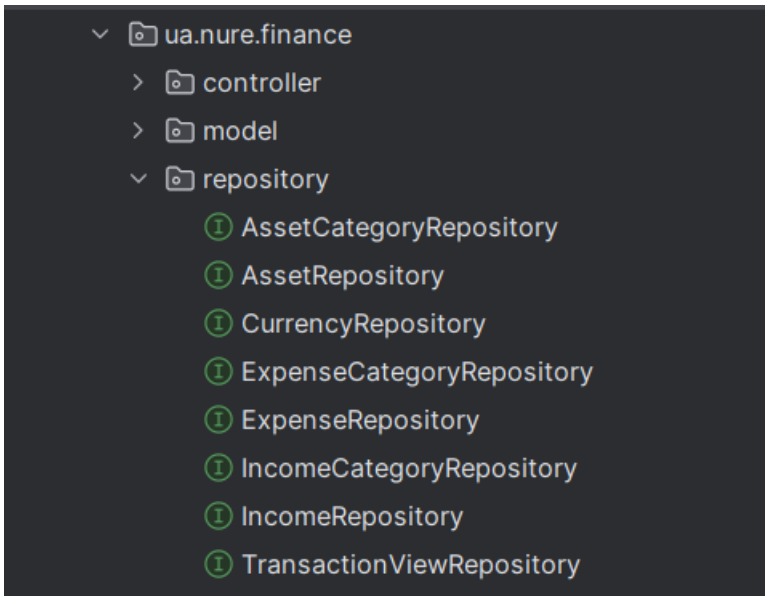

Рисунок 2.8 – Репозиторії JPA розробленого вебзастосунку

Приклад репозиторію JPA розробленого вебзастосунку (рис. 2.9).

```
package ua.nure.finance.repository;

import ...

public interface AssetRepository extends JpaRepository<Asset, Long> {  ihortetiushev
    List<Asset> findByStatus(Asset.Status status, Sort sort);  6 usages  ihortetiushev
    List<Asset> findByStatusAndCategory_Name(Asset.Status status, String categoryName);  4 usages  ihortetiushev

}
```

Рисунок 2.9 – Приклад репозиторію JPA розробленого вебзастосунку

MySQL – це одна з найбільш популярних реляційних систем управління базами даних з відкритим вихідним кодом. Вона базується на структурованій мові запитів SQL (Structured Query Language) та використовується для зберігання, обробки та управління даними в різноманітних програмних рішеннях [24].

Вся логіка збереження, оновлення, видалення та отримання даних реалізована через сервіси, що взаємодіють з репозиторіями JPA.

Для зберігання даних застосовується база даних MySQL. Робота з нею реалізується через Spring Data JPA, що дозволяє: автоматично генерувати SQL-запити; працювати з базою даних через об'єкти (Entity-класи); легко масштабувати та підтримувати систему [23].

Міграція схеми БД реалізована за допомогою Flyway [25].

Flyway – це інструмент управління версіями баз даних, який дозволяє безпечно і послідовно застосовувати зміни до структури бази даних (міграції). Його основної метою є забезпечення контролю за історією змін бази даних у рамках процесу розробки.

В розробленому застосунку Flyway-скріпти розташовані в папці src/main/resources/db/migration/structure (рис. 2.10).

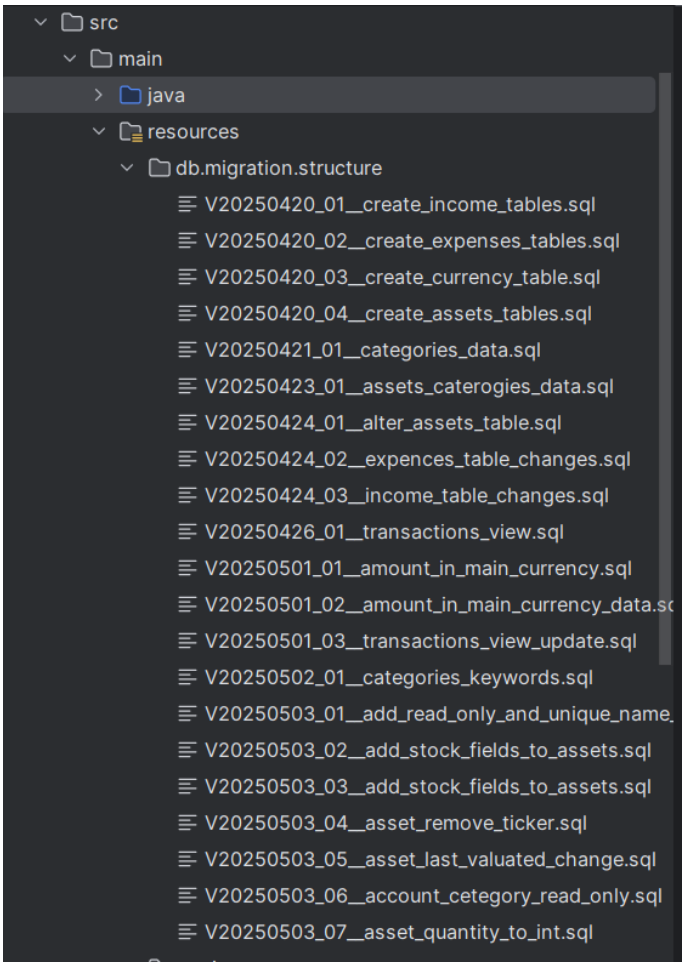

Рисунок 2.10 – Flyway-скріпти в розробленому вебзастосунку

2.3 Інструменти та середовище розробки

IntelliJ IDEA – це основне інтегроване середовище розробки, що спеціалізовано на розробці на Java, проте підтримує багато інших мов і фреймворків (Scala, JavaScript, Python, SQL тощо).

Серед основних особливостей IntelliJ IDEA можна виділити:

– розумне автодоповнення: пропонує найрелевантніші варіанти підставлення коду; розуміє контекст, типи даних і залежності між об’єктами;

– аналіз та рефакторинг коду: постійно аналізує код та підказує помилки або недоліки ще до компіляції; використовує потужні інструменти для рефакторингу (перейменування, виділення методу, зміна змінної тощо);

– інтеграція з системами контролю версій: підтримує Git, GitHub, SVN, має вбудований інтерфейс для комітів, злиття, перегляду історії та вирішення конфліктів;

– підтримка фреймворків і мов: Java (Spring, Java EE, Maven), Kotlin, Groovy, Scala; JavaScript, TypeScript, Angular, React, Vue; SQL, Python, PHP (через плагіни);

– інструменти для баз даних: підключення до СУБД напряму з IDE, інтерактивний перегляд і редагування таблиць, запуск SQL-запитів, автодоповнення SQL-коду;

– розробка для Android: підтримує Android SDK, сумісна з Android Studio (на базі IntelliJ IDEA Community Edition);

– плагіни: широка бібліотека плагінів, які розширюють функціональність IDE.

Git – це розподілена система контролю версій (Version Control System, VCS), яка дозволяє розробникам зберігати історію змін у коді, працювати над різними версіями проєкту, зручно співпрацювати з іншими та відстежувати зміни.

Основними особливостями Git є:

– розподіленість: кожен розробник має повну копію репозиторію з усією

історією змін, можна працювати офлайн, а потім синхронізувати зміни з віддаленим репозиторієм (наприклад, GitHub);

– історія комітів: кожна зміна зберігається як commit із повідомленням, автором, датою; можна повертатися до будь-якої попередньої версії проєкту;

– гілки (Branches): дозволяють паралельно розробляти нові функції або виправлення; гілки можна легко зливати (merge) або комбінувати (rebase);

– система індексації (Staging area): перед додаванням змін до репозиторію файли потрапляють у staging area (область підготовки); це дозволяє вибірково комітити частини змін.

GitHub – хостинг для репозиторію з вихідним кодом. Це веб-сервіс для зберігання, управління та спільної роботи над проєктами, які використовують систему контролю версій Git. Він надає інтерфейс для роботи з репозиторіями Git у браузері та безліч інструментів для спільної розробки, автоматизації та DevOps.

Основними функціями GitHub є:

– хостинг Git-репозиторіїв: публічні або приватні репозиторії з підтримкою Git; можливість клонувати, комітити, пушити й пулити прямо з IDE або терміналу;

– pull requests (PR): основний інструмент для внесення змін у проєкт; дозволяє переглянути, обговорити і протестувати зміни перед злиттям у основну гілку;

– issues: система для трекінгу задач, багів, фіч; підтримка тегів, призначення відповідальних, дедлайнів;

– GitHub Actions: CI/CD-платформа для автоматизації задач: збірка, тестування, деплой; працює на основі YAML-файлів у .github/workflows;

– wiki і документація: кожен репозиторій може мати власну wiki, підтримка Markdown у README-файлах для опису проєкту;

– code review: вбудований інструмент для рецензування коду;

– підсвічування синтаксису, коментарі до конкретних рядків;

– Fork: створення власної копії чужого репозиторію; дозволяє вносити

зміни без впливу на оригінал і надсилати pull request назад;

– GitHub Pages: хостинг статичних сайтів прямо з репозиторію; часто використовується для портфоліо, документації або демо.

Maven – система управління залежностями та зборки проєкту. Основна задача Maven полягає в допомозі розробнику найшвидше зрозуміти повний стан розробки.

Для досягнення цієї мети Maven вирішує питання спрощення процесу зборки, забезпечує єдину систему, надає якісну інформацію про проєкт.

Зведений перелік технологій, що використовувалися при розробці вебзастосунку персонального менеджера доходів та витрат наведено у таблиці 2.2.

Початковий код розробленого вебзастосунку персонального менеджера доходів та витрат знаходиться на GitHub https://github.com/ihortetiushev/diploma.git

Таблиця 2.2 Технології розробки вебзастосунку

Технологія	**Призначення**
1	**2**
Spring Boot	Фреймворк для швидкої розробки Java-застосунків, автоматизація конфігурації
Spring MVC	Побудова структури проєкту за шаблоном Model–View–Controller
Spring Data JPA	Робота з базою даних через ORM, спрощене управління запитами
Thymeleaf	Шаблонізатор для генерації динамічних HTML-сторінок

Продовження таблиці 2.2

1	2
MySQL	Реляційна база даних для зберігання інформації про користувача, доходи та витрати
Flyway	Контроль за історією змін бази даних
HTML5, CSS	Розмітка та стилізація вебсторінок
Bootstrap	Фреймворк для адаптивної верстки та інтерфейсу
JQuery	Багатофункціональна бібліотека JavaScript
Google Charts	Інструменти обробки даних для діаграм
IntelliJ IDEA	Середовище розробки Java-додатків
Git	Система контролю версій для відстеження змін у коді
Maven	Управління залежностями та збіркою проєкту

3 ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО ВЕБЗАСТОСУНКУ ПЕРСОНАЛЬНОГО МЕНЕДЖЕРА ДОХОДІВ ТА ВИТРАТ

3.1 Огляд розробленого вебзастосунку

3.1.1 Особливості побудові системи

Застосунок, що розробляється представлено у вигляді вебзастосунку. Одна з переваг вебзастосунку полягає в тому, що на клієнтський пристрій (смартфон, ПК, планшет, ноутбук тощо) не потрібно встановлювати додаткове програмне забезпечення. Оскільки застосунок працює у браузері, який встановлюється за замовчуванням у більшість операційних систем.

Тож вебзастосунок має додаткові переваги: підтримка різних операційних систем (Android, IOS, Windows) без необхідності написання/адаптування програмного коду та підтримка різних типів девайсів (чи то смартфон, ПК або будь-який девайс, який може запускати браузер).

На відміну від вебзастосунків класичні застосунки потребують адаптування або створення відповідного застосунку «з нуля» окремо для різних платформ і девайсів.

Але є і суттєвий недолік: для того щоб вебзастосунок працював, необхідні серверні ресурси, безпосередньо на які встановлюється застосунок.

В нашому випадку, основний кейс передбачає використання наявних домашніх серверів користувача (наприклад, міні ПК). Є варіант розміщення застосунку в хмарних провайдерах. Але у поточній версії, це неможливо без доопрацювання механізму аутентифікації.

Система не має паролю для входу, тому що це локальна система, яка встановлюється на ресурсах (серверах) користувача в його локальній мережі. Для доступу зовні передбачається використання VPN або сервісів які надають такий доступ, наприклад, Tailscale, Cloudflare tunn’el та ін. Якщо є потреба

використовувати хмарні провайдери для встановлення системи, необхідно інтегрувати систему аутентифікації.

Здійснюючи управління фінансами, користувач оперує активами. Мета застосунку полягає в тому, щоб користувач в будь-який час міг побачити наявні активи та мати уявлення про їх вартість.

Під активами розуміється відображення банківського рахунку, готівкових коштів в різних валютах, депозитів, зобов'язань третіх осіб, інвестицій, наявне у власності майно тощо.

Кожна витрата або доход змінює вартість/залишок визначеного активу. Витрата зменшує вартість активу, а дохід його збільшує. Система не контролює залишки. Наприклад, якщо залишок дорівнював нулю і було проведено витрати, отриманий результат має від'ємне значення. Це прийнятно для системи. Фінансовий зміст залишається на відповідальності користувача.

Для спрощення роботи застосунку, система не веде історію. Тобто залишок на вибраний день в минулому побачити неможливо. Вартість на поточну дату визначається як поточна в момент оновлення інформації.

Деякі активи мають початкову вартість, тобто вартість на момент занесення у систему (придбання цього активу), та поточну вартість, яка складається з переоцінки автоматичної (використовується для акцій) або мануальної. Для рахунків це не має сенсу. Ця умова використовується для акцій.

3.1.2 Функціональні особливості системи

Для запуска застосунку треба відкрити браузер та ввести URL (ім'я або ip-адресу) сервера на якому розгорнуто застосунок та порт 8080. Наприклад, http://192.168.1.225:8080 або http://localfinance:8080

Головна сторінка застосунку представлена вкладками (sidebar): витрати, доходи, активи, журнал транзакцій (рис. 3.1).

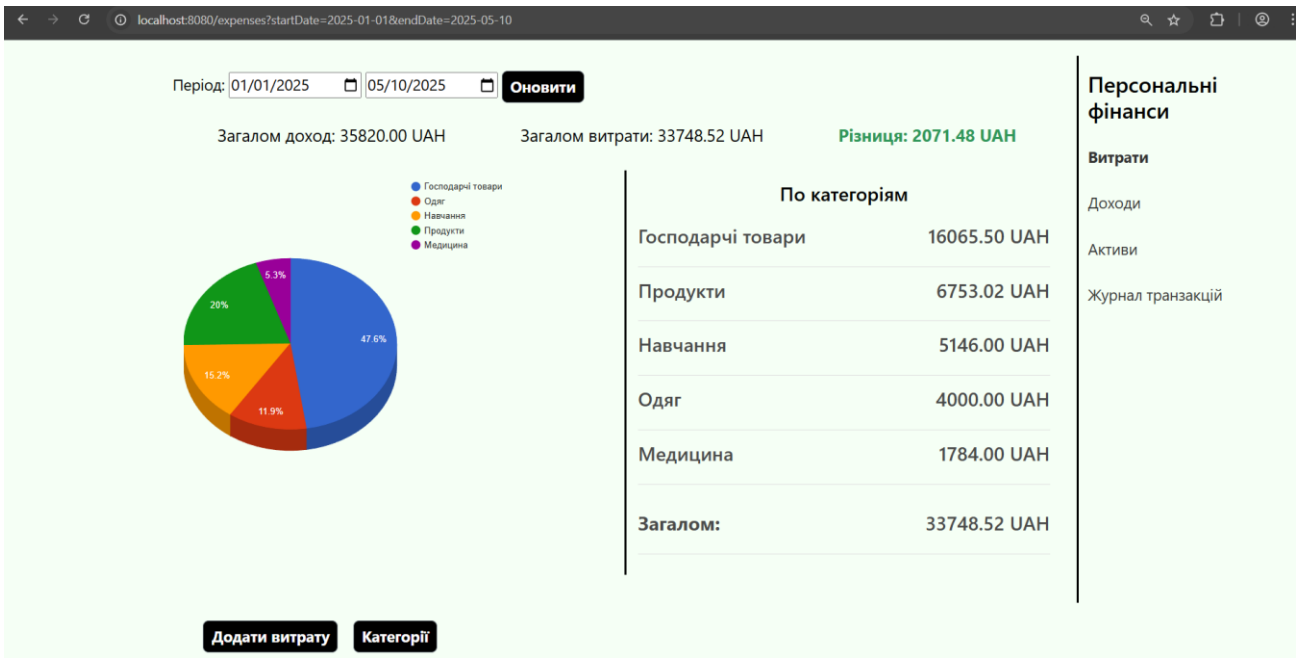

Рисунок 3.1 – Головна сторінка розробленого вебзастосунку

Відкривши програму, користувач одразу попадає на вкладку «Витрати». Задача цієї вкладки передбачає структурування витрат за категоріями та наочне відображення відповідних категорій в структурі витрат у вигляді діаграми. Користувач може задати будь-який період для відображання (день, тиждень, місяць, весь час користування застосунком), визначивши його в представленому календарі. За вказаний період буде надано зведену інформацію щодо отриманих доходів та проведених витрат з визначенням отриманого фінансового результату. У випадку залишку вільних коштів (доходу) різниця буде підсвічена зеленим кольором. У випадку, коли витрати переважають отримані за цей період доходи, сума буде підсвічена червоним кольором.

В програмі є категорії (як витрат, так і доходів), які встановлюються за замовчуванням. За визначений період всі витрати буде розділено по категоріям (наприклад: продукти, одяг, медицина, тварини, авто, навчання, розваги, комунальні платежі, інше). Відповідно до вказаних сум, наочно структура витрат буде відображена на круговій діаграмі з визначенням суми та питомої ваги категорії в структурі витрат.

При натисканні на кнопку «Категорії» відображаються категорії, існуючі в системі та ключові слова, які використовуються при імпортуванні даних з

банківської виписки для віднесення витрат до тієї чи іншої категорії (рис. 3.2).

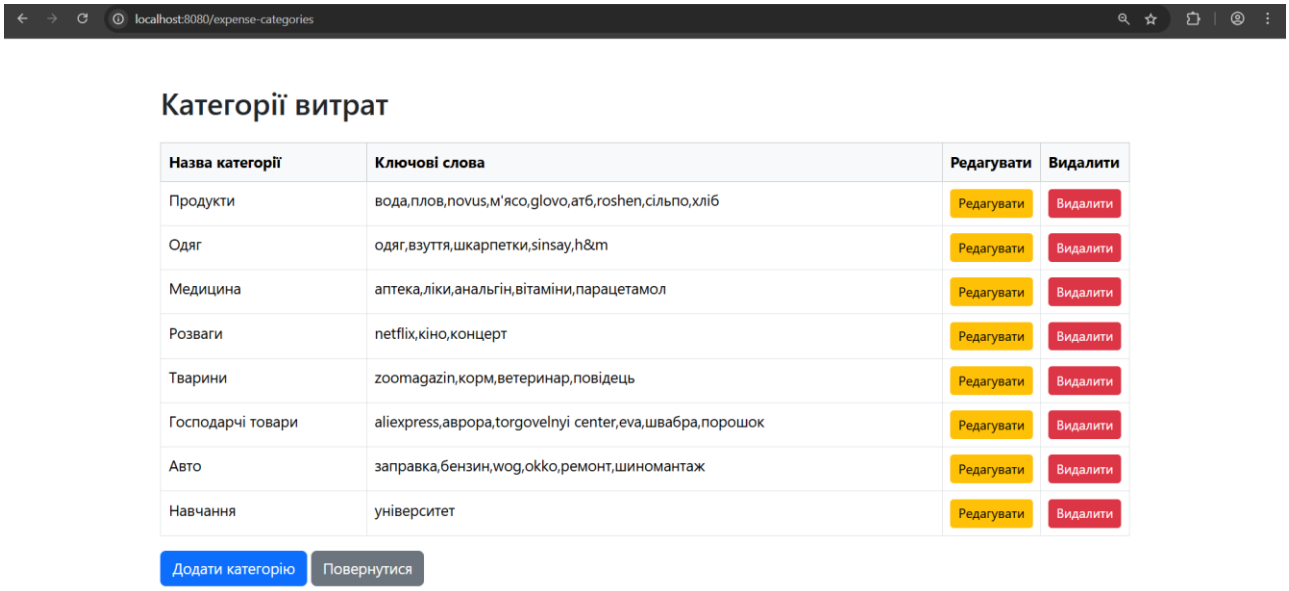

Рисунок 3.2 – Категорії витрат

Наприклад, задані слова, що зустрічаються в описі операції виписки банківської установи, такі як aliexpress, аврора, torgivelnyi center, eva, порошок, відносять витрати до категорій «Господарчі товари». Кожну категорію можна додавати, редагувати чи видаляти відповідно до своїх потреб. Редагування категорії передбачає зміну назви та ключових слів, що відносять до неї.

Якщо в системі існують витрати, які відносяться до певної категорії, то таку категорію видалити неможливо. При спробі видалити категорію, система видає помилку (рис. 3.3).

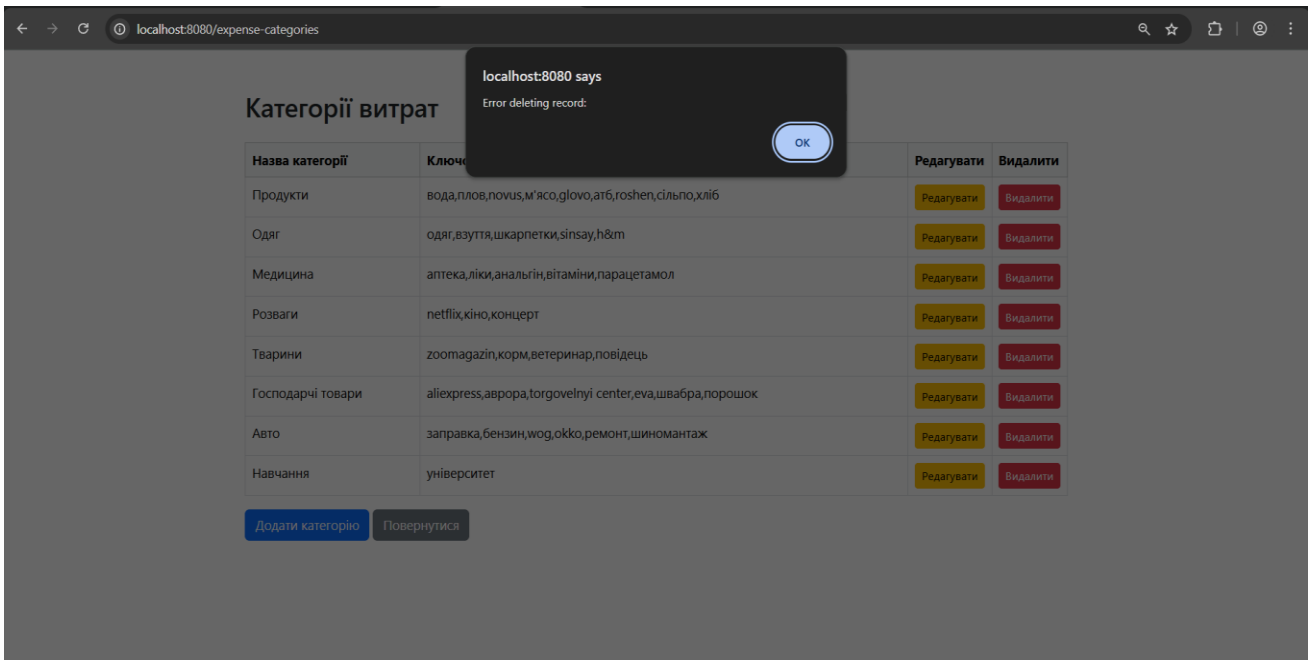

Рисунок 3.3 – Діалогове вікно при спробі видалення категорії

Застосунок передбачає можливість ручного введення витрат (рис. 3.4).

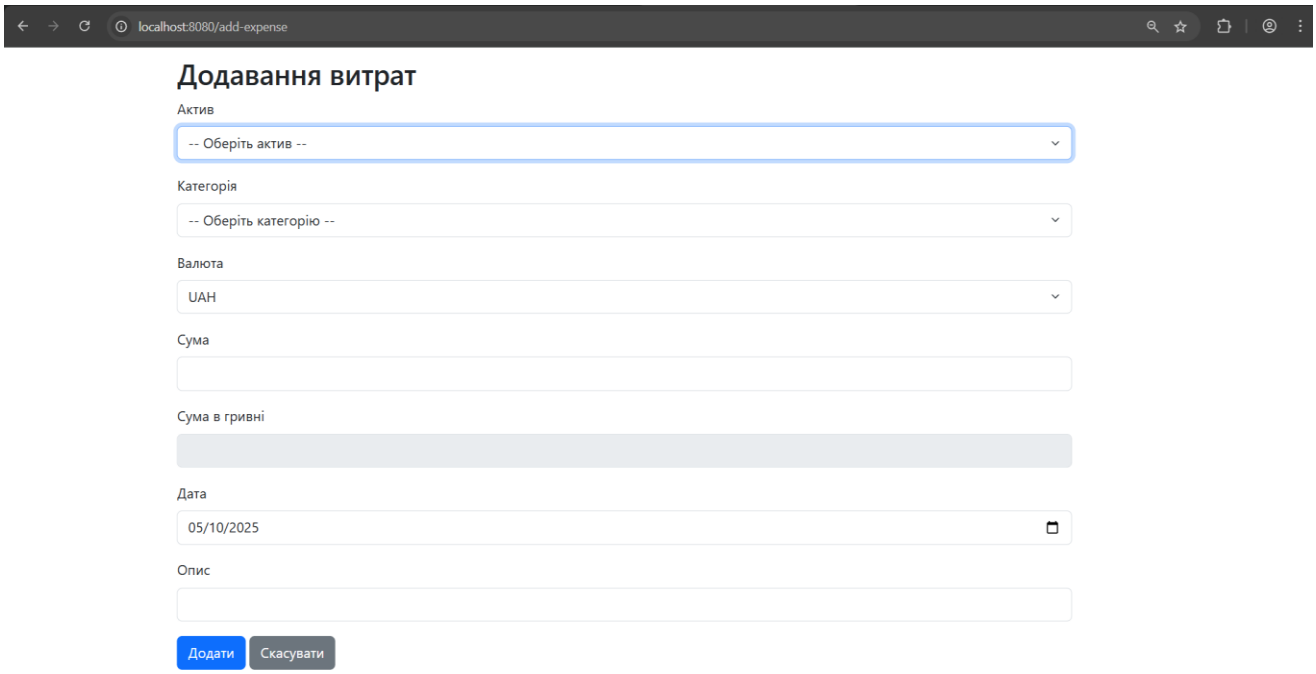

Рисунок 3.4 – Вкладка для додавання витрат

Мануальне додавання витрат можливо у разі здійснення, наприклад, операцій з готівкою. Можна вибрати вид активу, категорії, валюти, дати. Вказати суму та надати опис транзакцій.

Вкладка «Доходи» працює аналогічно вкладці «Витрати» (рис. 3.5).

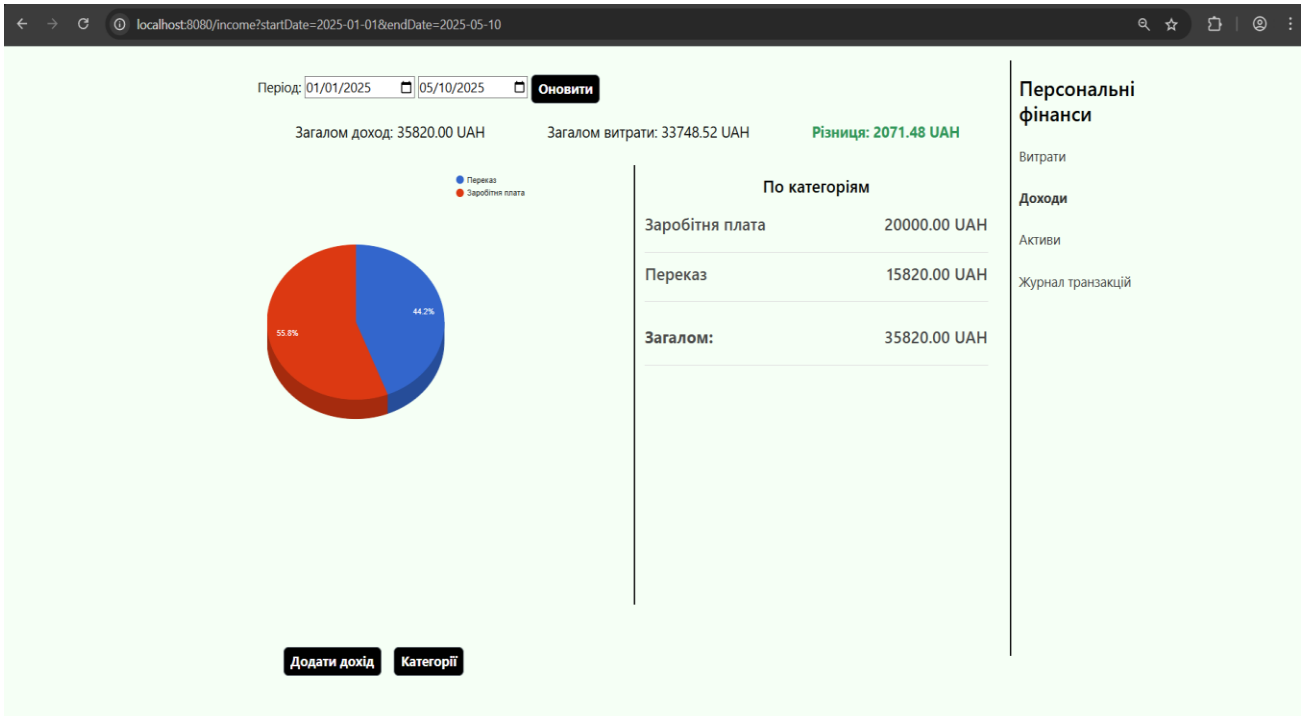

Рисунок 3.5 – Вкладка доходів

Задача вкладки передбачає відображення отриманих доходів за вказаний користувачем період. Це реалізується через групування сум по категоріям: наприклад, заробітна плата, переказ та інше.

Наочне відображення структури доходів реалізується через кругову діаграму з визначенням суми та питомої ваги кожної категорії в загальних доходах за вказаний період.

При натисканні на кнопку «Категорії» відображаються категорії, що існують в системі та ключові слова, які використовуються при імпортуванні даних з банківської виписки для віднесення доходів до тієї чи іншої категорії. Наприклад: зарплата, заробіток, контракт, переказ, повернення (рис. 3.6).

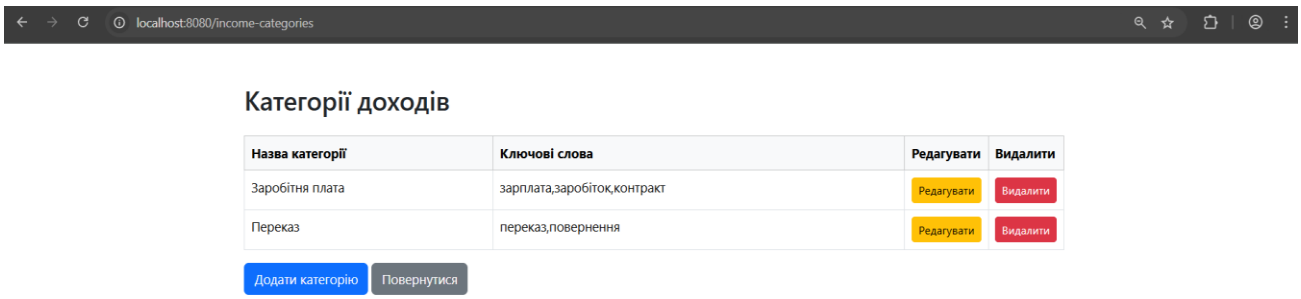

Рисунок 3.6 – Категорії доходів

За умови можливих змін в отримані доходу, категорії можуть бути додані, змінені або видалені.

Застосунок передбачає можливість ручного введення доходів. Додати дохід можливо натиснувши кнопку «Додати дохід» на вкладці «Доходи» (рис. 3.7).

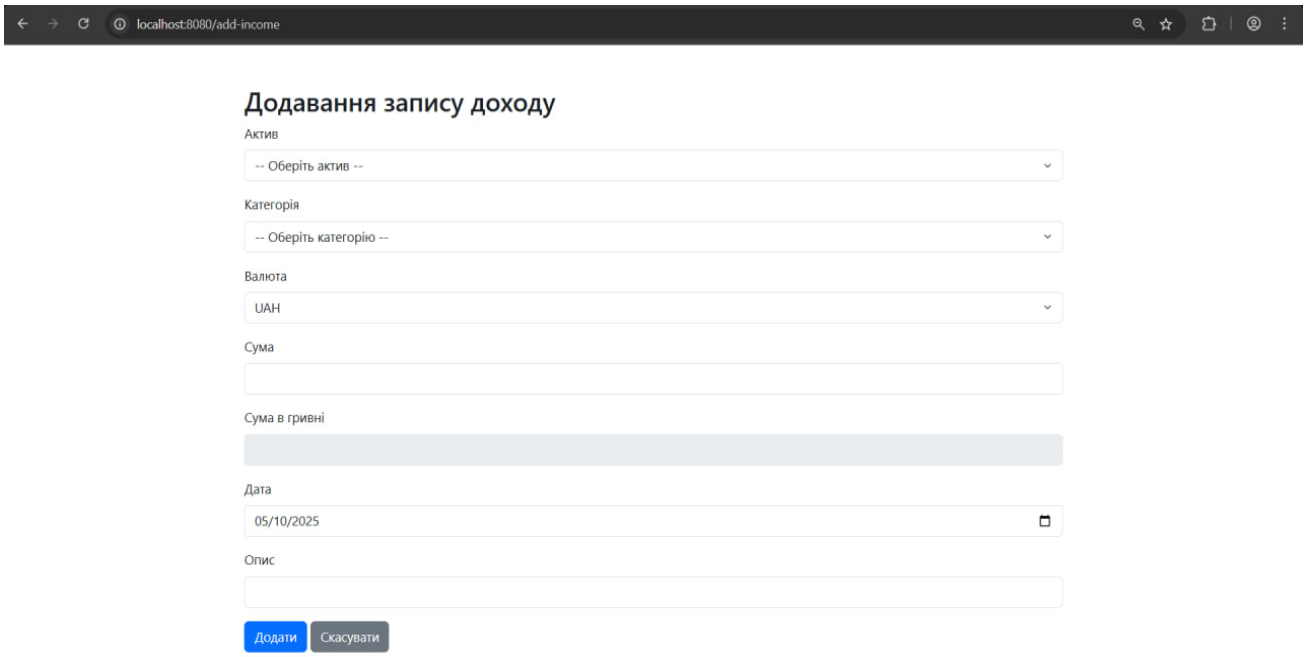

Рисунок 3.7 – Додавання доходів

Мануальне додавання доходів можливе у разі здійснення, наприклад, операцій з готівкою. Можна вибрати вид активу, категорії, валюти, дати. Вказати суму та надати опис транзакцій.

Вкладка «Активи» представлена у вигляді таблиці, що надає інформацію про наявні активи (рис. 3.8).

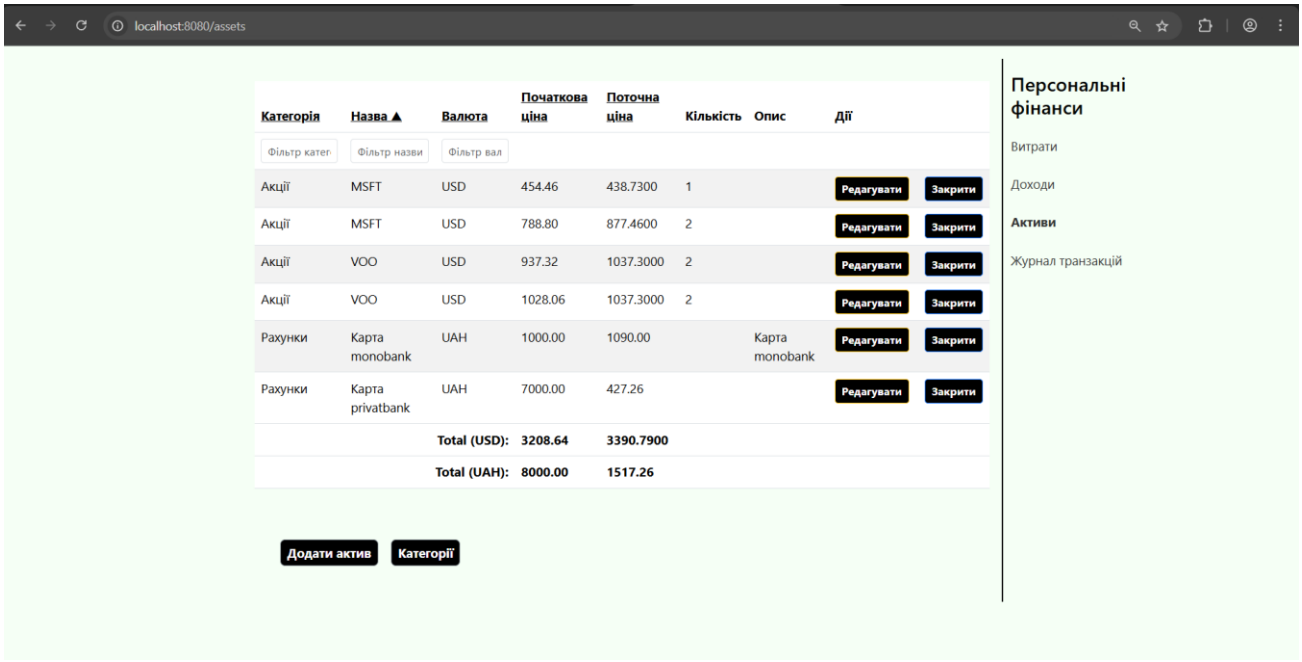

Рисунок 3.8 – Вкладка наявних активів

На цій сторінці є можливість фільтрування або пошуку по стовпчиках: категорії, назва, валюта. Для цього у вікні введення даних над відповідним стовпчиком вносяться дані для пошуку. Після чого треба натиснути Enter. Якщо у фільтрі вказати неіснуючі дані, система зазначає, що таких даних немає.

По стовпчиках, які позначені як посилання (категорія, назва, валюта, початкова вартість, поточна вартість) можливе сортування.

Порядок сортування (по зростанню або зменшенню) по стовпчиках означений трикутником.

Кожну позицію можна редагувати та закривати.

При закритті актив не видаляється фізично, а залишається в системі зі статусом «closed». Закрити можна будь-який актив, незалежно від наявних

залишків. Користувач повністю несе відповідальність за ці дії (рис.3.9).

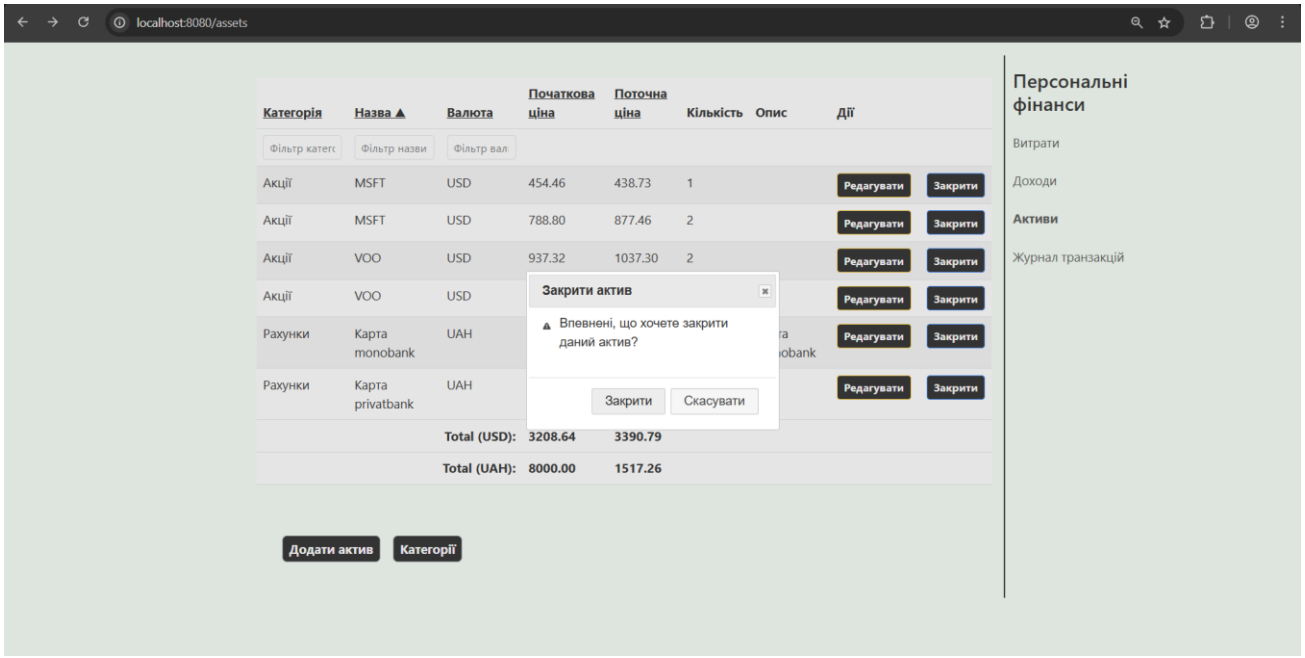

Рисунок 3.9 – Вікно підтвердження закриття активу

Якщо актив був закритий випадково, можна звернутися до адміністратора для зміни (повернення) статусу активу в «active».

В подальших планах, для розширення функціоналу, є надання користувачу можливість працювати зі статусом «closed»: переглядати та змінювати статус.

Кожен актив треба віднести до відповідної категорії. Є категорії (рахунки, акції), які вбудовані в систему. Їх не можна редагувати чи видалити. При наведені курсору на кнопку редагування або видалення з’являється відповідний текст для інформування користувача.

Інші категорії можна додавати, редагувати або видаляти за бажанням (рис. 3.10).

Рисунок 3.10 – Категорії активів

Активи можна додавати. Активи можна занести за категоріями, наприклад: рухоме майно (авто), рахунки (готівка в різних валютах, розрахункові, депозитні та карткові рахунки), акції тощо (рис 3.11).

Якщо актив віднесено до категорії, її неможливо видалити. Редагувати можна як завгодно.

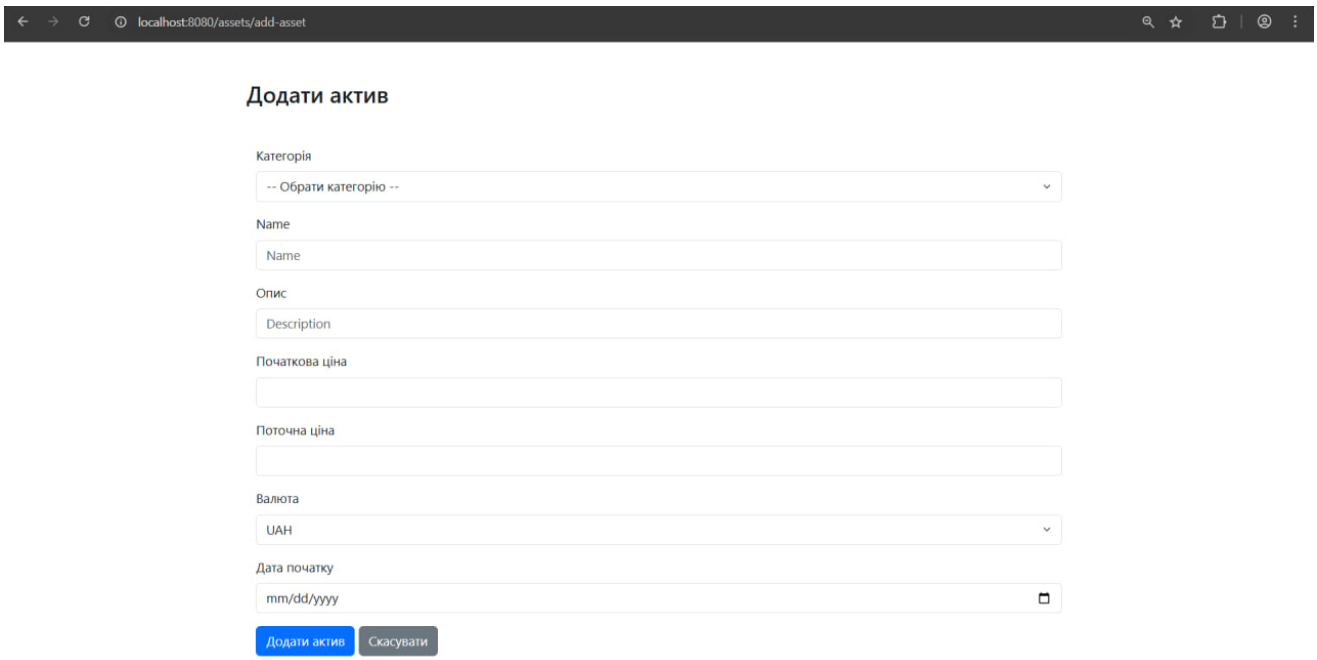

Рисунок 3.11 – Додавання активу

При мануальному додаванні активів треба вибрати категорію активу, назву, надати опис, встановити початкову ціну (ціну покупки), поточну ціну (ціну переоцінки на сьогодні) валюту, дату.

Особливістю розробленого застосунку є опрацювання категорії «Акції» у структурі активів (рис. 3.12).

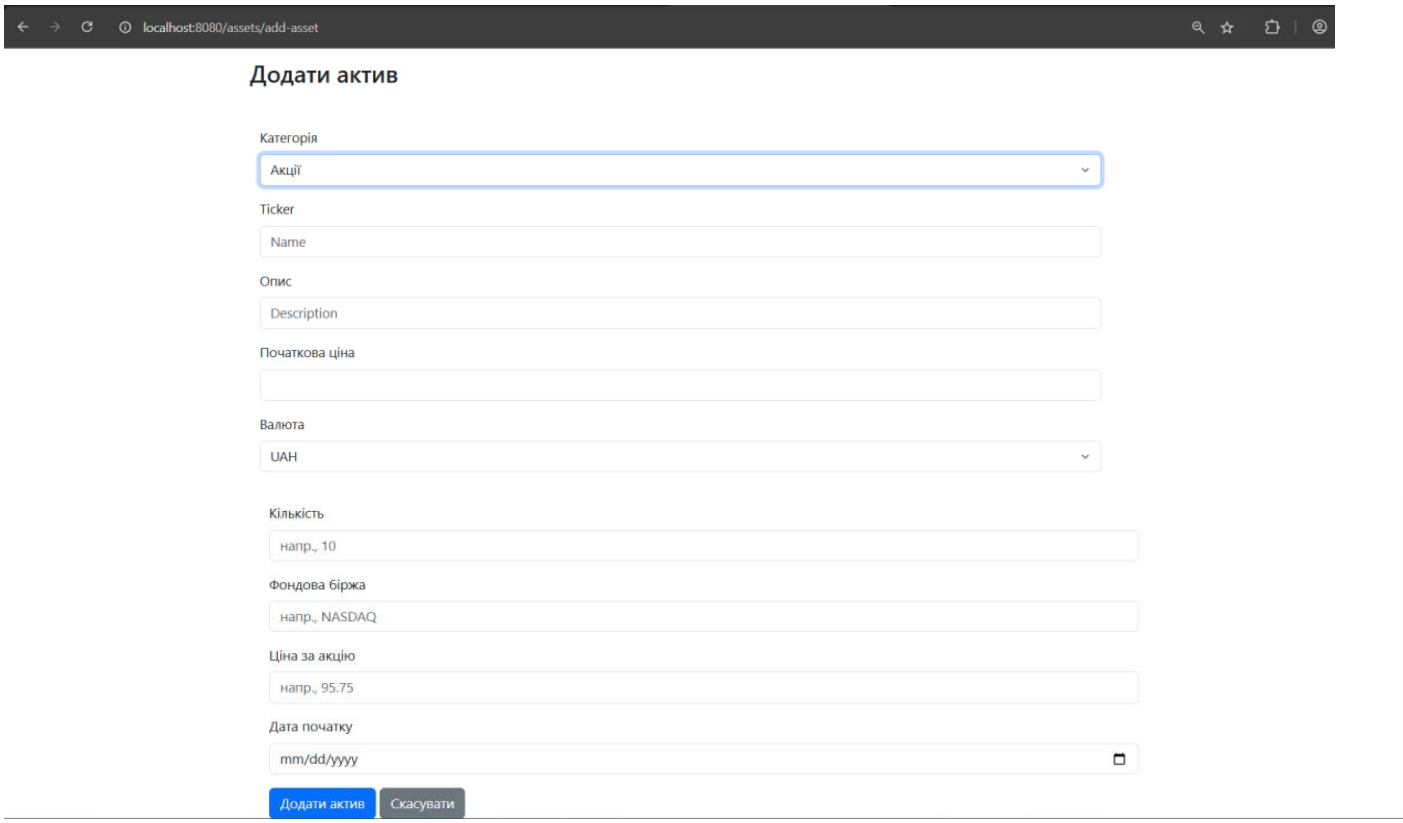

Рисунок 3.12 – Додавання акцій

Це новий функціональний інструментарій, який не зустрічався в подібних застосунках персональних фінансів. Відстеження обліку вартості акцій доволі кропітка та витратна за часом операція. Це пов’язано із тим, що важко відслідковувати зміну окремих траншей в загальному портфелі цінних паперів. У зв’язку із чим облік акцій ведеться за двома сумами: первісна вартість (сума по якій акції купувалися) та поточна вартість (сума останнього оновлення вкладки).

Вклавши в акції визначену суму коштів, поточна вартість в застосунку відображається на підставі актуального котирування на біржовому ринку. Застосунок розраховує поточну вартість акцій шляхом помноження кількості

акцій в поточному активі на актуальну вартість відповідної акції. Це проводиться автоматично кожного разу, коли вкладка активів відкривається. Ім’я активу типу «Акції» повинно бути тікером відповідного цінного паперу. Наприклад, VOO, MSFT, AAPL та ін.

Треба зазначити, що для калькулювання поточної вартості акцій використовується public API, який має обмеження 25 запитів на добу. При перевищені ліміту запитів наступне оновлення вартості може бути лише на наступний день.

Ідея обліку акцій в застосунку передбачає, що однакові акції треба заносити в систему окремим активом по вартості купівлі (рис. 3.13). Бо однакові акцій, що купувалися в різний час, можуть мати різну вартість.

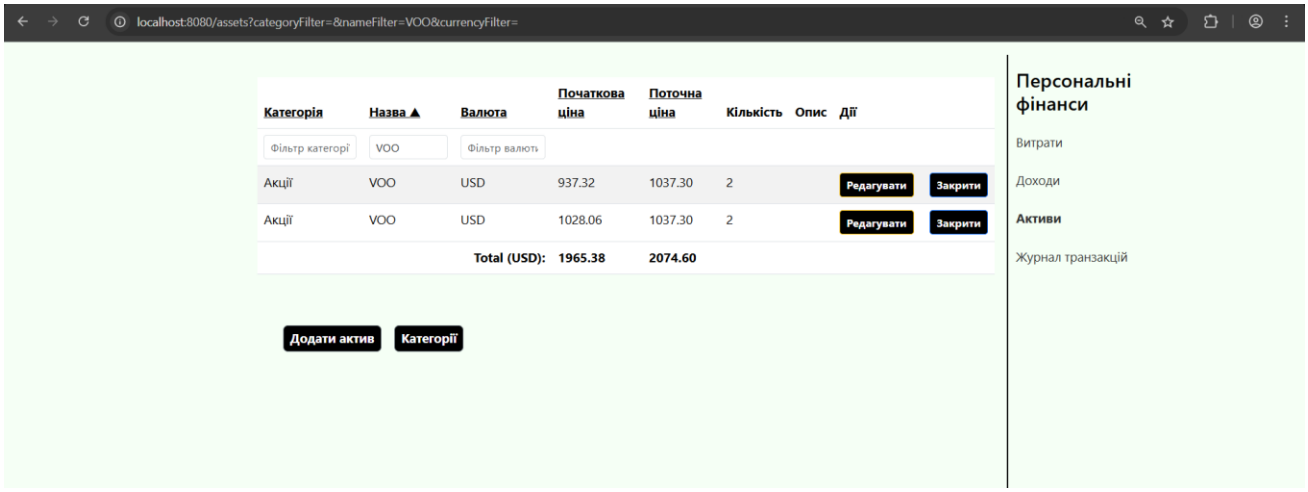

Рисунок 3.13 – Декілька траншей з фільтром однієї назви акцій

Використовуючи фільтри по типу активу («акції») та назві активу (тікер), можна бачити поточний фінансовий результат по обраним акціям.

Особливістю обліку коштів на поточних або карткових рахунках є можливість користування кредитними коштами (овердрафт). У цьому випадку поточна вартість буде мати від’ємне значення.

Загальна сума активів складається за різними валютами касовим методом (для визначення фінансового результати доходи складаються, а витрати віднімаються у момент їх фіксації). Тобто залишок заборгованості за

овердрафтом (від'ємне значення) за картковим рахунком віднімається від суми отриманих доходів для визначення поточного стану активів (рахунків).

Поточна вартість інших активів (наприклад, авто, нерухомості) переглядається користувачем довільно і періодично редагується. Це зроблено з метою наочного розуміння загальної вартості активів.

Журнал транзакцій представляє собою перелік всіх операцій (рис. 3.14)

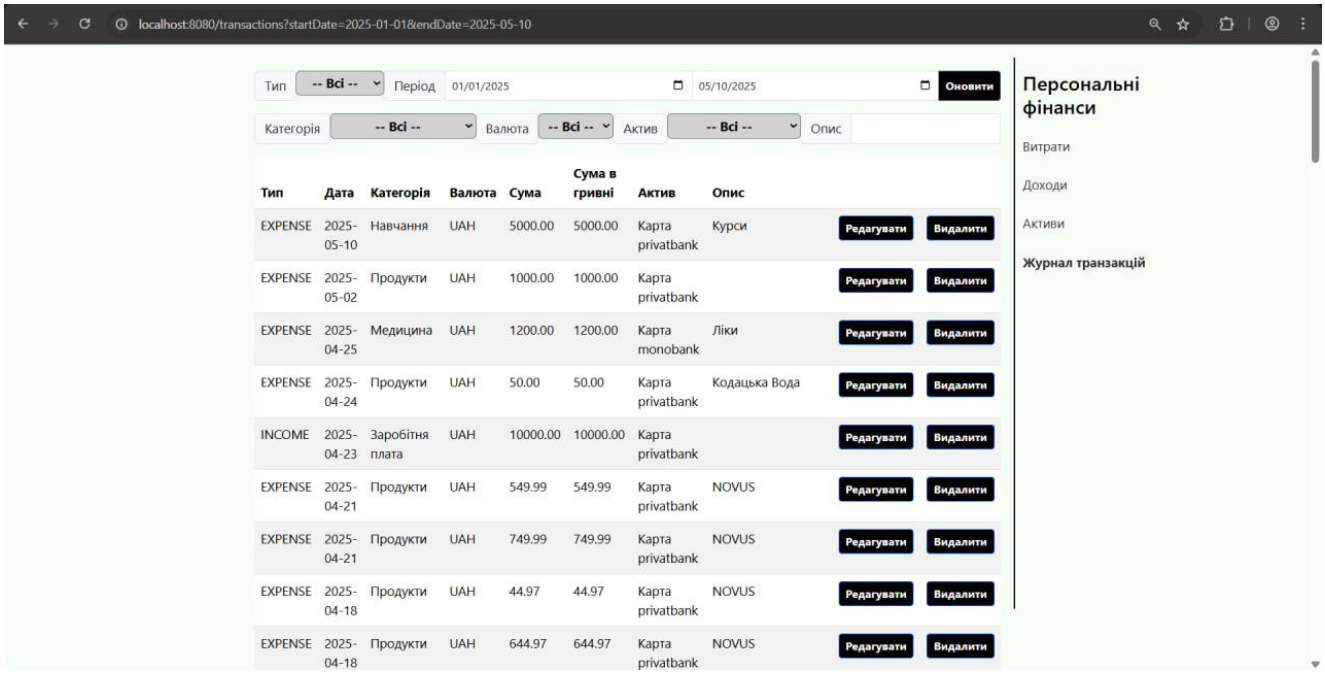

Рисунок 3.14 – Журнал транзакцій

Тип транзакцій відповідає признаку доходів або витрат (EXPENCE, INCOME), категорія відображає приналежність транзакції до відповідної категорії доходів або витрат.

Сума в гривні автоматично не перераховує валютні залишки. Еквівалент в гривні визначається користувачем при введені/редагуванні відповідної операції.

Опис та інші поля можна редагувати. Наприклад, поповнення карткового рахунку можна змінити на «подарунок на ДН».

Особливістю застосунку є можливість редагування даних без будь-яких обмежень. Тобто операцію, яка була в минулому, можна відобразити у будь-який час. Це робить застосунок гнучким. Проте накладає на користувача додаткову

відповідальність за контроль даних.

Варто зазначити, що видалення транзакції має незворотній характер і не може бути скасовано. Тому користувач має бути уважним з цією функцією.

Визначенні в описі транзакції назви торгівельних мереж, в яких здійснювалися витрати, за характером покупок автоматично відносять їх до відповідних категорій. Наприклад, NOVUS, Кодацька вода – до продуктів, Аврора, Eva – до господарчих товарів.

Період для відображення транзакцій в журналі операцій задається користувачем.

Транзакції в журналі операцій можна фільтрувати за: типом операції, категорією, валютою, видом активу, описом (часткове співпадіння).

Значною перевагою застосунку є те, що є можливість імпортувати дані з банківських виписок (рис. 3.15). Наразі використовуються виписки з Монобанк та Приватбанк (ці банки розглядаються як найбільш поширені банківські установи серед користувачів). При зацікавленості чи початку обслуговування користувача в інших банках, імпортування виписок по рахунках цих банків, буде додано в наступних версіях застосунку.

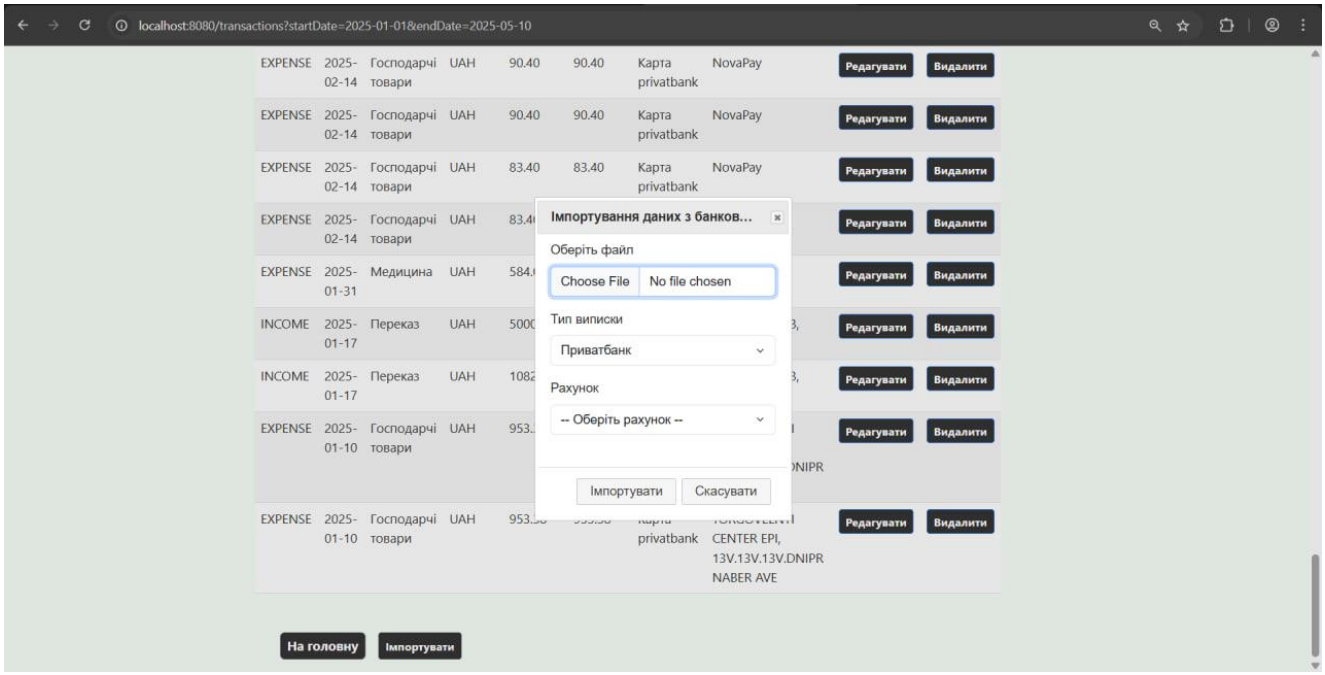

Рисунок 3.15 – Діалогове вікно імпортування банківської виписки

При внесенні в систему валютних витрат або доходів, суми фіксуються як в валюті, так і в еквіваленті в гривні. Відповідно в вкладках доходів та витрат такі суми обліковуються в еквіваленті в гривні. Система курси валют не зберігає та не синхронізує курс з зовнішніми джерелами.

3.2 Вдосконалення розробленого вебзастосунку

Для того, щоб застосунок завжди відповідав викликам часу, його потрібно постійно вдосконалювати. Це можуть бути як вдосконалення існуючих функціональних можливостей, так і додавання принципово нових можливостей і функцій.

Надалі буде продовжено роботу по забезпеченню зручного інтерфейсу, що могло б покращити інтуїтивне використання застосунку.

Визначимо наступні напрямки функціонального вдосконалення розробленого застосунку:

– запровадження аутентифікації передбачає не лише перевірку особи користувача, але і сукупні функції: зберігання облікових даних користувача, забезпечення механізму входу та двофакторної аутентифікації, вихід, тайм-аут, управління сесіями;

– багатокористувацький режим передбачає одночасне функціонування однієї програми для певного числа користувачів. Це накладає ролі на користувачів (адміністратор з повним доступом, менеджер для операційного управління означеним колом ресурсів та власне користувачі, які мають обмежений доступ лише до своїх даних). Такий функціонал потребує ізоляції даних та можливості їх фільтрації на основі ID, контролю доступу, аудиту дій;

– масштабування застосунку потребує наявності зручного доступу користувачами. Можливо перенесення його в хмарні сервіси. Крім того, в залежності від числа користувачів, застосунок потребує масштабування без погіршення операційних можливостей (наприклад, часу відгуку тощо);

– налаштування та адаптивний дизайн для мобільних девайсів;

– покращення або додавання валідації даних. Опрацювання відповідної бізнес-логіки. Наприклад, при видалені категорії активу проведення перевірки коректності та доцільності таких дій;

– впровадження різних типів діаграм. В розробленому застосунку використовується лише кругова діаграма, яка ілюструє чисельні пропорції. Для аналізу динаміки показників, трендів, запровадження можливості прогнозування, доцільно використовувати діаграми-лінії, стовпчасті діаграми, тощо;

– додати історію ведення активів. Визначення вартості на будь-яку минулу дату. Це дозволить візуалізувати динаміку та наочно бачити зміни;

– вдосконалити налаштування застосунку, щоб зробити його як візуально привабливим, так і функціонально корисним. Запровадити можливість вибору теми, мови для користувача. А також налаштування полів виписки, додавання інтеграцій з банківськими установами, підтримка банків, що працюють через API без необхідності імпортування виписки;

– підтримка курсів валют. Використання, зберігання, забезпечення історії курсів на момент операції;

– підтримка трансферу. Спрощення процедури перерозподілу активів без зміни фінансового результату (конвертація суми в іншу валюту, перерозподіл між готівкою та безготівковими залишками);

– робота з закритими активами. Зараз в застосунку при закритті активу зникає можливість перегляду історії дій з ним;

– можливість експортувати/імпортувати всі дані;

– використання AI для аналізу доходів та структури витрат для отримання персоналізованих рекомендацій з оптимізації фінансів.

Всі доопрацювання та вдосконалення застосунку мають робити фінансове управління доступним та спонукати користувача до більш ефективного контролювання і планування власних фінансів.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено вебзастосунок персонального менеджера доходів та витрат, призначений для допомоги користувачам у ведені особистих фінансів. В результаті проведеного аналізу існуючих аналогів та вимог користувачів було сформовано основні функціональні вимоги до системи.

В якості оптимального рішення було обрано вебзастосунок. Таке рішення не потребує встановлення додаткового програмного забезпечення, підтримується різними операційними системами та на різних типах девайсів.

Застосунок дозволяє імпортувати дані з банківських виписок з Монобанк та Приватбанк чи проводити будь-які дії (додавання/редагування/видалення) з транзакціями мануально, автоматично категорізувати доходи та витрати за ключовими словами, вести їх облік та відображати категорії в структурі доходів та витрат у вигляді діаграм. Новим для існуючих на ринку застосунків є облік акції та їх автоматична переоцінка вартості при відкриті вкладки на підставі актуального котирування на біржовому ринку.

Застосунок має відкритий код. Початковий код розробленого вебзастосунку персонального менеджера доходів та витрат знаходиться на GitHub

https://github.com/ihortetiushev/diploma.git

Застосунок легко може доопрацьовувати. Це можуть бути як вдосконалення існуючих функціональних можливостей, так і додавання принципово нових можливостей і функцій.

Використання застосунку дозволить користувачу мати наочну, актуальну та повну інформацію про наявні активи та їх вартість, що дозволить вдосконалити фінансовий менеджмент власних активів та оптимізує їх використання.

Результати роботи апробовано у вигляді тез доповідей під час Міжнародного молодіжного форуму» РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [1].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Тетюшев І.А. (2025). Розробка вебзастосунку персонального менеджера доходів та витрат. Радіоелектроніка та молодь у XXI столітті: 29 міжнародний молодіжний форум. С. 143-144

2. Financial Literacy: Importance, Benefits and How to Increase Your Financial Literacy. Corporate Finance Institute. URL: https://corporatefinanceinstitute.com/resources/wealth-management/financial-literacy/ (дата звернення 11.05.2025).

3. Крупка, М. І., Коваленко, В. М., & Ковалюк, О. М. (2019). Фінансовий менеджмент: підручник. Львів: ЛНУ імені Івана Франка.

4. Von Tobel, A. (2013). Financially fearless: The LearnVest Program for taking control of your money. Crown Currency.

5. WHITE, Alexandria. Millennials and Gen Z are the most likely to use mobile banking apps–here's why, plus budgeting tips. CNBC. URL: https://www.cnbc.com/select/why-millennials-gen-z-use-mobile-banking-apps/ (дата звернення 11.05.2025).

6. Financial Apps are Becoming More Popular Among Younger Generations. Spave – Make your own world of difference. URL: https://www.spave.io/resources/financial-apps-are-becoming-more-popular-among-younger-generations (дата звернення 11.05.2025).

7. Топ-5 мобільних додатків для управління фінансами у 2024 році. Фінансист – Останні новини про валюту, криптовалюти та фінанси в Україні. URL: https://www.finansist.org/news/finansy/top-5-dodatkiv-upravlinnya-finansamy-2024?utm_source=chatgpt.com (дата звернення 11.05.2025).

8. Обираємо додаток для обліку доходів і витрат- investnews.com.ua. URL: https://investnews.com.ua/analityka/2207-poriadok-u-finansakh-iak-pravylno-obraty-zastosunok-dlia-kontroliu-dokhodu-ta-vytrat/?utm_source=chatgpt.com (дата звернення 11.05.2025)

9. Wallet by BudgetBakers review. Finder UK. URL: https://www.finder.com/uk/budgeting/wallet-budgetbakers-review?utm_source=chatgpt.com (дата звернення 11.05.2025).

10. П’ять українських додатків для контролю витрат. URL: https://rubryka.com/article/dlya-kontrolyu-vytrat/ (дата звернення 11.05.2025).

11. Тітов, С. В., Тітова, О. В., & Чорна, О. С. (2022). Опис нескоротних наборів ознак в приблизних множинах з використанням систем числення. *Збірник наукових праць Харківського національного університету Повітряних Сил*, (1 (71)), 106-110.

12. Chorna, O., Didyk, P., Titov, S., & Titova, O. (2024). Usage of Clustering Algorithms for Automating Route Planning in Transportation Routing Tasks. *Системи обробки інформації*, (1), 176.

13. Євтушенко, Д., & Творошенко, І. С. (2024). Особливості застосування методів оптимізації бізнес-процесів, реалізованих засобами машинного навчання та нейронних мереж.

14. The Decline Of The Native App And The Rise Of The Web App. Forbes. URL: https://www.forbes.com/sites/victoriacollins/2019/04/05/why-you-dont-need-to-make-an-app-a-guide-for-startups-who-want-to-make-an-app/#54e0fd826e63 (дата звернення 11.05.2025).

15. PWA vs Native: Which Approach to Choose in 2025 Brainhub software engineering company. URL: https://brainhub.eu/library/pwa-vs-native. (дата звернення 11.05.2025).

16. Web Application Architecture. URL: https://www.spaceotechnologies.com/blog/web-application-architecture (дата звернення 11.05.2025).

17. Spring Boot. Spring Boot. URL: https://spring.io/projects/spring-boot (дата звернення 11.05.2025).

18. Spring Web MVC :: Spring Framework. URL: https://docs.spring.io/spring-framework/reference/web/webmvc.html (дата звернення 11.05.2025).

19. Thymeleaf. URL: https://www.thymeleaf.org/ (дата звернення 11.05.2025).

20. Bootstrap. The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/ (дата звернення 11.05.2025).

21. jQuery. URL: https://jquery.com (дата звернення 11.05.2025).

22. Charts | Google for Developers. Google for Developers. URL: https://developers.google.com/chart (дата звернення 11.05.2025).

23. Spring Data JPA. Spring Data JPA. URL: https://spring.io/projects/spring-data-jpa (дата звернення 11.05.2025).

24. MySQL. MySQL. URL: https://www.mysql.com/ (дата звернення 11.05.2025).

25. GitHub - flyway/flyway: Flyway by Redgate • Database Migrations Made Easy. GitHub. URL: https://github.com/flyway/flyway (дата звернення 11.05.2025).

26. Тітов, С. В., & Тітова, О. В. (2015). Оцінка юзабіліті освітніх сайтів: методи і технології. *Вісник Харківської державної академії культури. Серія: Соціальні комунікації*, (47), 127-134.

27. Ситніков, Д. Е., Ситнікова, П. Е., Тітов, С. В., & Тітова, О. В. (2021). Фільтрація результуючого набору асоціативних правил з точки зору оцінки цікавості. *Системи обробки інформації*, (1 (164)), 83-88.

28. Гороховатський В.О., Творошенко І.С. (2022) Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124 с.

29. Tvoroshenko, I., & Maksimenko, H. (2021). Research of regression and modular testing of web applications.

30. Iryna, T., & Maksym, K. (2021). ABOUT THE ROLE OF TESTING IN PROCESS OF MOBILE APPLICATION DEVELOPMENT. *Problems of modern science and practice*, *1*, 416.

31. Sitnikov, D., Titova, O., Minukhin, S., Kovalenko, A., & Titov, S. (2018, October). Informativity of association rules from the viewpoint of information theory. In *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)* (pp. 595-598). IEEE.

32. Гороховатський В.О., Творошенко І.С. (2021) Методи інтелектуального

аналізу та оброблення даних: навч. посібник. Харків: ХНУРЕ, 92

33. Творошенко І.С. (2021) Технології прийняття рішень в інформаційних системах: навч. посібник. Харків: ХНУРЕ, 120 с.