

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Органайзер з керування витрат»

Виконав студент групи ІПЗ-21-1
КИРИЧЕНКО Олександр Сергійович

Керівник роботи:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КИРИЧЕНКО Олександр Сергійович**

Керівник роботи: **ЛОКТИКОВА Тамара Миколаївна**

Тема роботи: **«Органайзер з керування витрат»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: **об'єктом дослідження є процес автоматизації обліку та планування особистих фінансів, а предметом дослідження - технологічні рішення, алгоритмічні підходи та сервіси, що забезпечують збір, обробку й візуалізацію даних про витрати.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	27.02.2025р.	27.02.2025р.
2	Локтікова Т.М.	27.02.2025р.	27.02.2025р.
3	Локтікова Т.М.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебзастосунку органайзера з керування витрат та пояснювальної записки. Пояснювальна записка містить 66 сторінок текстової частини, 26 ілюстрацій, 20 таблиць, 6 додатків та 42 джерела інформації. Загальний обсяг роботи становить 78 сторінок.

Розроблено сучасну веб-орієнтовану систему для управління особистими фінансами із залученням актуальних технологій та підключенням зовнішніх API для отримання ринкової інформації.

Архітектура побудована на чіткому розділенні клієнтської та серверної частин, які взаємодіють між собою через REST API. Серверна частина реалізована на основі PHP-фреймворку Laravel, а клієнтська частина створена за допомогою інструментів React і Vite, що забезпечує швидкість розробки та високу продуктивність.

Розроблена система охоплює основні аспекти управління особистими фінансами: ведення рахунків у різних валютах, класифікацію транзакцій, планування бюджету, формування аналітичних звітів, а також інтеграцію з інвестиційними сервісами. Окремо реалізовано механізм обробки звітів.

Інтерфейс застосунку є інтуїтивно зрозумілим і адаптивним, що дозволяє користувачам зручно контролювати свої доходи й витрати, планувати бюджет, аналізувати фінансові показники та приймати обґрунтовані рішення щодо власних фінансів.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, КЕРУВАННЯ ВИТРАТАМИ, LARAVEL, REACT, TAILWIND, VITE.

					ІПЗ.КР.Б – 121 – 25 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Органайзер з керування витрат. Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Кириченко О.С						
Перевір.		Локтікова Т.М.					3	66
Керівник		Локтікова Т.М.				ФІКТ Гр. ІПЗ-21-1		
Н. кодтр.		Тичина О.П.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification thesis consists of a web application for expense management organizer and an explanatory note. The explanatory note contains 66 pages of textual part, 26 illustrations, 20 tables, 6 appendices and 42 sources of information. The total volume of the work is 78 pages.

A modern web-oriented system for personal finance management has been developed using current technologies and connecting external APIs to obtain market information.

The architecture is built on a clear separation of client and server parts, which interact with each other through REST API. The server side is implemented based on the PHP Laravel framework, and the client side is created using React and Vite tools, which ensures development speed and high performance.

The developed system covers the main aspects of personal finance management: maintaining accounts in different currencies, transaction classification, budget planning, analytical reporting, as well as integration with investment services. A separate report processing mechanism has been implemented.

The application interface is intuitive and adaptive, allowing users to conveniently control their income and expenses, plan budgets, analyze financial indicators and make informed decisions about their personal finances.

KEYWORDS: WEB APPLICATION, EXPENSE MANAGEMENT, LARAVEL, REACT, TAILWIND, VITE.

		Кириченко О.С			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ОРГАНАЙЗЕРА З КЕРУВАННЯ ВИТРАТАМИ	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Вибір архітектури органайзера	17
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	19
Висновки до розділу 1	23
РОЗДІЛ 2 ПРОЕКТУВАННЯ ОРГАНАЙЗЕРА	24
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	24
2.2 Розробка бази даних системи.....	32
2.3 Проектування та реалізація алгоритмів роботи системи.....	38
2.4 Реалізація функціоналу органайзера	40
Висновки до розділу 2	43
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОРГАНАЙЗЕРОМ.....	44
3.1 Порядок встановлення та налаштування системи.....	44
3.2 Структура інтерфейсу органайзера	47
3.3 Тестування роботи застосунку	56
Висновки до розділу 3	59
ВИСНОВКИ	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	63
ДОДАТКИ.....	66

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HMR - функція, яка дозволяє розробникам оновлювати модулі в запущеному застосунку без необхідності повного перезавантаження сторінки

БД - база даних

UX/UI - User Experience/ User Interface, користувацький досвід/користувацький інтерфейс

НБУ - Національний банк України

SPA - Single Page Application, односторінковий застосунок

DOM - Document Object Model, об'єктна модель документа

API - Application Programming Interface, інтерфейс прикладного програмування

ORM - Object-Relational Mapping, об'єктно-реляційне відображення

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. У сьогоденних умовах швидкого зростання обсягів фінансових операцій кожен користувач стикається з потребою не лише в їх агрегації, а й у своєчасному контролі власних витрат. Саме вирішенню цього питання присвячений розроблений органайзер з керування витратами. Бажання до кращого управління особистими фінансами спонукає шукати прості інструменти, що можуть автоматизувати рутинні дії, давати загальні дані і робити прогнози. Таким чином, тема цієї роботи не тільки важлива для користувачів, але й є важливою також з науково-технічного боку, адже поєднує сучасні підходи до обробки даних у реальному часі зі зручним UX/UI.

Метою цієї роботи стало створення ефективного вебзастосунку органайзера з керування витратами.

Для досягнення поставленої мети необхідним є виконанням таких **завдань**: аналіз предметної області та огляд й аналіз існуючих аналогів програмного продукту, обґрунтування вибору технологій для реалізації вебзастосунку, проектування структури бази даних і API ендпоінтів, реалізація клієнтської та серверної частин, а також проведення тестування розробленого органайзера.

Об'єктом дослідження виступає процес автоматизації обліку та планування особистих фінансів.

Предметом дослідження є технологічні рішення, алгоритмічні підходи та сервіси, що забезпечують збір, обробку й візуалізацію даних про витрати.

У роботі використовуватимуться методи системного аналізу для визначення функціональних вимог, об'єктного проектування для організації коду та сервісів, а також різні підходи до тестування, зокрема за допомогою вбудованих засобів Laravel.

За темою випускної кваліфікаційної роботи бакалавра опубліковані тези: Кириченко О.С., Локтікова Т.М. Органайзер з керування витратами. Тези доповідей Всеукраїнської науково-практичної on-line конференції здобувачів

		Кириченко О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

вищої освіти і молодих учених, присвяченої Дню науки, 12-17 травня 2025 року. Житомир: «Житомирська політехніка», 2025. С. 61-62 [1].

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1 АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ОРГАНАЙЗЕРА З КЕРУВАННЯ ВИТРАТАМИ

1.1 Постановка задачі

Метою випускної кваліфікаційної роботи бакалавра є створення фінансового застосунку з обліку витрат, що поєднає інтуїтивно зрозумілий інтерфейс для користувачів із високою продуктивністю, також потрібно врахувати, щоб застосунок було легко масштабувати під будь-які вимоги.

Планування бюджету - добра справа, незалежно від мети. Тримати все під контролем корисно, хоч би й через цікавість, навчання грамотному розподілу коштів або прагнення раціональної економії.

Органайзер має представляти собою декілька основних програмних модулів, що забезпечують основний функціонал та можливості для ефективного управління рахунками та планування бюджету.

Модуль транзакцій повинен мати можливість дозволяти створювати записи про доходи та витрати з детальною категоризацією, встановлювати теги для швидкого пошуку та групування операцій. Основною його функцією має бути можливість вести облік операцій по різних рахунках та валютах.

Модуль аналітики повинен надати засоби для побудови звітів про фінансові потоки за різними періодами. Це дозволить створювати графіки та діаграми для візуалізації всієї фінансової діяльності, аналізувати тенденції та виявляти закономірності у витратах користувача.

Модуль експорту та імпорту даних має забезпечувати експорт фінансових даних у різних форматах, а саме Excel, CSV, PDF, OFX, для подальшого аналізу або звітності. Також має бути реалізовано функцію імпорту банківських виписок.

Модуль інтеграції з зовнішніми сервісами дозволить отримувати інформацію в реальному часі, наприклад для отримання курсу валют з

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

офіційних джерел, або котирування акцій компаній на біржах з детальною історією для аналізу, в що краще вкласти зекономлені кошти, та прогнозовану дохідність.

Для досягнення поставленої задачі необхідно виконати наступні етапи розробки: потрібно передбачити проведення аналізу предметної області та дослідити існуючі аналоги для виявлення їхніх переваг та недоліків. На цьому етапі також має бути сформовано детальні вимоги до функціоналу; потрібно спроектувати архітектуру системи з урахуванням принципів модульності та масштабованості. На цьому етапі створюється структура БД, визначаються API інтерфейси для взаємодії між компонентами та розробка загальної концепції інтерфейсу користувача; передбачається реалізація спроектованої системи з поступовою розробкою кожного модуля. Особлива увага приділятиметься забезпеченню якості коду та дотриманню встановлених архітектурних принципів; також здійснюватиметься комплексне тестування розробленої системи, перевірка продуктивності при роботі з великими обсягами даних та налагодження виявлених помилок.

В результаті повинно бути створено застосунок, який дозволить користувачу ефективно керувати своїми витратами та планувати бюджет. Також важливою складовою є модульність архітектури застосунку, оскільки всі компоненти системи взаємодіятимуть через чітко визначені інтерфейси, що забезпечуватиме легке масштабування та додавання нового функціоналу.

1.2 Аналіз аналогів програмного продукту

Сучасний ринок застосунків для моніторингу за особистим бюджетом представлений широким спектром рішень, що відрізняються функціоналом та технічними підходами. Для розуміння ринку та виявлення можливостей для покращення було проведено детальний аналіз найпопулярніших аналогів, які підходять для українського ринку.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

Wallet є поширеним застосунком з керування витратами, якій підтримує різні валюти та має хмарну синхронізацію з різними платформами [39]. На рис. 1.1 представлено головну сторінку платформи Wallet від budgetbakers.

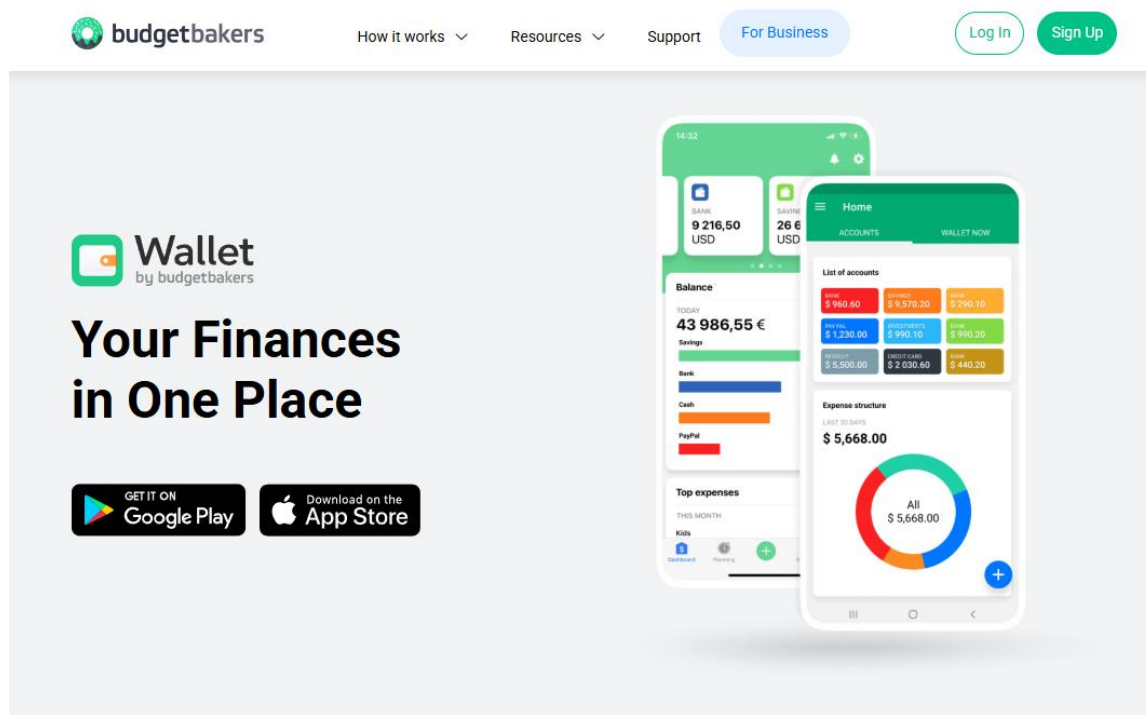


Рисунок 1.1 - Головна сторінка Wallet

Даний продукт розроблений компанією budgetbakers, має як версії для Android та IOS, так і веб версію. Застосунок розрахований на ручне введення транзакцій з можливістю синхронізації між пристроями через хмарні сервіси. Хоч і має можливість інтеграції з банками, але це в преміум підписці.

Система використовує систему монетизації з базовим безкоштовним функціоналом та розширеним за щомісячну підписку.

Основними перевагами Wallet є: інтуїтивний інтерфейс з чіткою категоризацією транзакцій, що дозволяє користувачам швидко адаптуватись до інтерфейсу, навіть людям, які ніколи не користувались подібними застосунками; наявність системи підтримки валют з автоматичним конвертуванням за поточним курсом; застосунок дозволяє створювати різні типи рахунків (готівкові, банківські, кредитні карти); відстеження прогресу; автоматичний розрахунок всіх необхідних щомісячних платежів.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Особливостями є фотографування чеків з автоматичним розпізнаванням суми; система також надає додаткову аналітику з графіками трендів, порівняння періодів; також є можливість імпорту даних через документи.

Серед недоліків Wallet можна виділити: обмеження функціоналу в безкоштовній версії - користувачі мають обмежену к-ть рахунків, мають доступ до базової аналітики, тоді як найбільш потрібна аналітика буде в платній версії; неможливість встановлювати складні бюджети з підкатегоріями; відсутній імпорт і експорт даних у популярних та зручних форматах без преміум підписки.

Spendee є фінансовим застосунком нового покоління з акцентом на візуальне представлення даних та естетику інтерфейсу [36]. На рис. 1.2 представлено головну сторінку платформи Spendee.

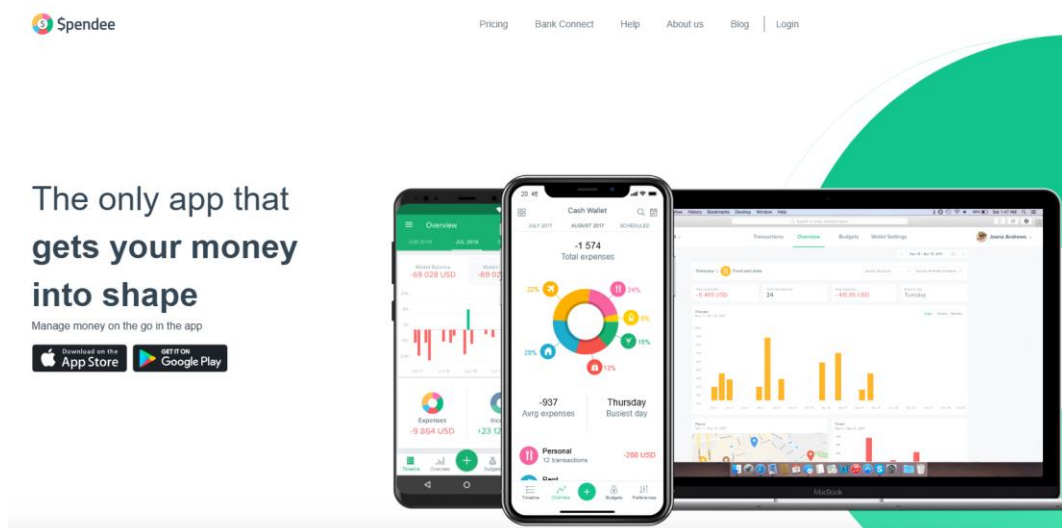


Рисунок 1.2 - Головна сторінка Spendee

Даний продукт розроблений з особливою увагою до дизайну та простоти використання, намагаючись зробити процес ведення фінансового обліку більш захоплюючим та мотивуючим. Застосунок доступний як мобільні версії для Android та iOS, так і веб-версію з синхронізацією між пристроями

Система використовує freemium модель монетизації з базовим безкоштовним функціоналом та розширеним преміум-доступом за щомісячну підписку.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Основними перевагами Spendee є: сучасний дизайн з використанням принципів Material Design, яскравими кольорами та плавними анімаціями; автоматична генерація красивих звітів з детальною аналітикою через кругові діаграми, стовпчасті графіки та тренди; функція спільних гаманців для сімейного бюджету з можливістю одночасного ведення обліку кількох користувачами; підтримка геолокації для автоматичного додавання місця покупки; гнучка система категоризації з можливістю створення власних категорій для швидкого розпізнавання типів витрат.

Серед недоліків Spendee можна виділити: акцент на візуальну частину часто йде на шкоду функціональності - деякі базові операції перенасичені візуальними ефектами, що ускладнює роботу з великою кількістю транзакцій; критично обмежений функціонал безкоштовної версії - майже всі важливі функції як створення рахунків, додавання транзакцій, експорт даних та аналітика доступні лише після сплати підписки; безкоштовна версія має настільки мінімальний функціонал, що практично непридатна для повноцінного використання.

MoneyManager представляє мінімалістичний підхід до керування фінансами, орієнтований на швидке введення транзакцій та базову аналітику [22]. На рис. 1.3 представлено інтерфейс застосунку MoneyManager.

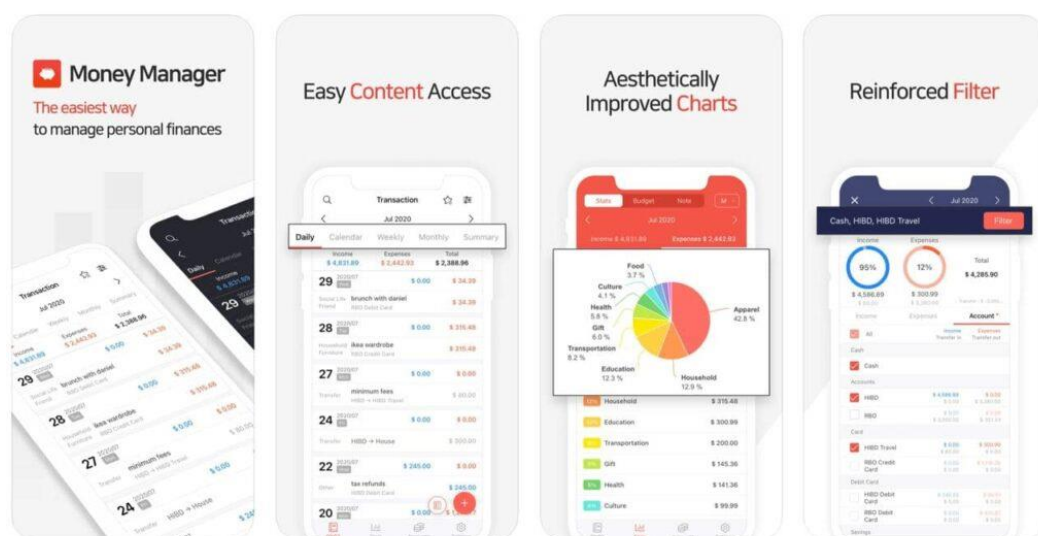


Рисунок 1.3 - Представлення MoneyManager

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Даний продукт розроблений з урахуванням потреб користувачів, які цінують простоту та ефективність над розширеним функціоналом. Застосунок доступний переважно як мобільний додаток для Android та iOS без веб-версії та хмарної синхронізації.

Система використовує повністю безкоштовну модель без преміум-підписок, що робить всі функції доступними одразу після встановлення.

Основними перевагами MoneyManager є: видатна швидкодія та інтуїтивна простота використання - додавання транзакції займає лише кілька секунд завдяки продуманому інтерфейсу; система запам'ятовування часто використовуваних операцій з можливістю повторення за шаблоном; інтуїтивно зрозумілий інтерфейс навіть для нових користувачів без досвіду роботи з подібними застосунками; підтримка великої кількості валют з автоматичним конвертуванням між рахунками; вбудований калькулятор для швидких обрахунків без переходу в сторонні сервіси; система створення резервних копій даних для легкого перенесення інформації між пристроями.

Серед недоліків MoneyManager можна виділити: суттєво обмежені можливості аналітики - звіти досить прості та не надають глибокого розуміння фінансових трендів; відсутність функцій прогнозування, порівняння періодів або виявлення аномальних витрат; відсутність веб-версії або десктопного клієнта, що ускладнює роботу з великими обсягами даних; відсутність хмарної синхронізації, що обмежує використання на кількох пристроях одночасно; мінімальні можливості для детального аналізу фінансових звітів порівняно з конкурентами.

Bluescoins є найбільш комплексним рішенням для обліку фінансів серед розглянутих аналогів, з акцентом на детальний облік та глибоку аналітику [6]. На рис. 1.4 представлено інтерфейс застосунку Bluescoins.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		14



Рисунок 1.4 - Представлення Bluecoins

Даний продукт позиціонує себе як професійний інструмент для користувачів та бізнесу з серйозними вимогами до фінансового планування та обліку. Застосунок доступний як мобільні версії для Android та iOS з обмеженою веб-функціональністю.

Система використовує freemium модель з базовим безкоштовним функціоналом та розширеними можливостями за одноразову оплату без щомісячних підписок.

Основними перевагами Bluecoins є: глибина функціоналу з багаторівневою категоризацією витрат та необмеженою кількістю підкатегорій; можливість створення складних звітів за будь-які періоди з фільтрацією по множині параметрів включаючи категорії, рахунки, теги, суми та геолокацію; розвинена система тегів та міток для створення перехресних зав'язків між транзакціями; детальне відстеження боргів та кредитів з автоматичним розрахунком відсотків та графіками погашення; підтримка множинних валют включаючи історію курсів та крипто валюти з інтеграцією до основних бірж; найбільш розвинутий функціонал бюджетування з можливістю створення складних бюджетів з різними типами лімітів та

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

каскадними обмеженнями; підтримка різних типів рахунків включно з інвестиційним портфелем.

Серед недоліків Bluescoins можна виділити: надмірну складність інтерфейсу для користувачів, яким потрібен лише базовий функціонал домашньої бухгалтерії; перевантаженість інтерфейсу опціями налаштувань та меню, що робить його малопридатним для швидкого введення повсякденних витрат; складність базових операцій - навіть додавання звичайної транзакції вимагає заповнення множинних полів; технічні проблеми при синхронізації між пристроями та роботі з великими обсягами даних; повільну роботу при накопиченні тисяч транзакцій та випадкові збої при генерації складних звітів.

Проведений аналіз існуючих рішень виявив характерну проблему ринку фінансових застосунків - відсутність збалансованого рішення, яке б поєднувало простоту використання з достатнім функціоналом для ефективного керування особистими фінансами. У табл. 1.1 наведено порівняльну характеристику оглянутих аналогів програмного продукту за основними параметрами.

Таблиця 1.1

Порівняльна характеристика аналогів програмного продукту

	Wallet	Spendee	MoneyManager	Bluescoins
Модель монетизації	Freemium	Freemium	Безкоштовний	Freemium
Складність використання	Середня	Низька	Дуже низька	Висока
Аналітика та звіти	Хороша	Базова	Обмежена	Професійна
Експорт/Імпорт даних	Платно	Платно	Обмежений	Повний
Синхронізація	Хмарна	Хмарна	Відсутня	Хмарна
Швидкість роботи	Хороша	Середня	Відмінна	Повільна
Веб-версія	Так	Так	Ні	Так

Як випливає з табл. 1.1, основними недоліками всіх розглянутих рішень є обмеження експорту/імпорту даних у безкоштовних версіях. Це не дозволяє користувачам вільно мігрувати між різними системами. Крім того, жоден з аналогів не орієнтований на український ринок, з його особливостями валютного регулювання та банківської системи.

На основі виявлених недоліків обґрунтовано необхідність створення власного рішення, яке поєднає інтуїтивність MoneyManager, аналітичні можливості Bluescoins, візуальну привабливість Spendee та функціональність Wallet, при цьому забезпечуючи повністю безкоштовний доступ до основного функціоналу та орієнтацію на український ринок.

1.3 Вибір архітектури організатора

На основі проведеного аналізу аналогів та визначених технічних вимог було прийнято рішення про використання сучасної веб-орієнтованої архітектури, яка забезпечить гнучкість, масштабованість та високу продуктивність системи. Архітектурні рішення обирались з урахуванням виявлених недоліків аналогів та необхідності створення універсального застосунку для українського ринку.

Головним рішенням при створенні архітектури стало використання роздільної клієнт-серверної архітектури з чітким розподілом відповідальності. Користувацький клієнт реалізовано як SPA [35] на базі JavaScript бібліотеки React, що забезпечує динамічний інтерфейс та швидку взаємодію з користувачем без необхідності перезавантаження сторінок. Сервер побудований як Restful API на фреймворку Laravel, що надає стандартизовані точки доступу до даних та бізнес-логіки.

Цей підхід дозволяє легко масштабувати кожну частину системи незалежно одну від одної, забезпечує можливість створювати мобільні застосунки, десктопні версії для різних систем у майбутньому, та спрощує тестування окремих компонентів. На відміну від монолітних рішень, що

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

використовуються в деяких аналогах, розділена архітектура дозволяє різним командам працювати паралельно над клієнтом та сервером.

Для створення серверної частини було обрано класичну шарову архітектуру з чітким розподілом обов'язків між компонентами. Контролери відповідають виключно за прийманням HTTP-запитів, перевірку вхідних даних та формування відповідей, тоді як вся бізнес-логіка винесена в окремі сервісні класи, що робить код більш тестованим та повторно використовуваним.

Робота з БД організована через Eloquent ORM [17]. Це дозволяє абстрагуватись від конкретної реалізації збереження даних та легко змінювати логіку запитів без впливу на верхні шари. Особливо важливим це рішення є для фінансових операцій, де необхідно забезпечити цілісність.

На клієнтській частині використовується компонентний підхід з функціональними компонентами React та хуками для керування станом. Кожен компонент має чітко визначену відповідальність та мінімальні залежності від інших частин системи. Великі сторінки розбиті на окремі віджети, які можуть використовуватись незалежно та комбінуватися для покращення користувацького досвіду.

Для керування даними використовується Context API [9] замість складніших рішень типу Redux, що спрощує архітектуру та зменшує кількість шаблонного коду. Асинхронні операції організовані через кастомні хуки, що забезпечує частоту компонентів та можливість повторного використання логіки.

Архітектура застосунку спроектована з урахуванням принципу відкритості/закритості - система відкрита для розширення, але закрита для модифікації. Нові функції додаються через створення нових модулів без зміни базової структури. До прикладу, модуль інвестицій реалізований як окремий набір компонентів з власними сервісами та API ендпоінтами.

Всі функціональні модулі мають власні структури файлів та можуть розвиватись незалежно. Особливо це важливо для майбутнього додавання

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

функцій типу планування бюджету, аналітики витрат і т.д. Модульність також спрощує підтримку коду та дозволяє легко відключати непотрібні функції.

API спроектовано згідно принципів REST [32] з використанням стандартних HTTP методів та статус кодів. Всі ендпоінти повертають дані в JSON форматі з дотриманням структури відповідей, що включає статус операцій та даних. Для складних операцій, а саме експорт або імпорт даних, передбачено асинхронну обробку з можливістю відстеження прогресу.

Також приділено значну увагу до обробки помилок - система розрізняє перевірку помилки, логічних обмежень та технічні збоїв, що надає користувачу зрозуміле повідомлення, аби користувач зрозумів, що він може виправити в запиті або звернутись до адміністрації. Всі критичні операції логуються для можливості діагностики проблем.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір технологічного стеку для розробки органайзера з керування витратами здійснювався з урахуванням специфіки фінансових застосунків, необхідності забезпечення високої продуктивності при роботі з великими обсягами даних та вимог до безпеки. Кожна технологія обиралася на основі її зрілості, підтримки спільноти та відповідності архітектурним принципам проекту.

Для реалізації серверної частини було обрано PHP фреймворк Laravel 12 [16] як основу для розробки RESTful API [33]. Це рішення обумовлено кількома ключовими факторами. Laravel забезпечує потужну систему маршрутизації з можливістю групування ендпоінтів та middleware для обробки автентифікації та авторизації. Вбудований Eloquent ORM спрощує роботу з БД, що забезпечує безпечне виконання запитів.

Фреймворк має відмінну підтримку перевірки даних з можливістю створення правил перевірки, що є важливим чинником при виборі БД для

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

фінансових застосунків, де точність даних є пріоритетом. Система міграцій БД дозволяє легко керувати змінами структури даних при деплої на різні середовища, до прикладу в Docker. Окрім того, Laravel має розвинену екосистему та документацію, що прискорює його вивчення та розробку.

Для зберігання даних обрано PostgreSQL [27] - потужну об'єктно-реляційну систему для управління базами даних. Цей вибір зумовлений високою надійністю PostgreSQL, дотриманням ACID-властивостей, що є важливими для забезпечення цілісності фінансових угод, і можливістю виконувати складні запити. В порівнянні з MySQL, PostgreSQL частіше обирають за кращу відповідність стандартам SQL та більшу стабільність при складних операціях. Для фінансового застосунку, для якого точність та узгодженість даних є головними пріоритетами, переваги реляційної системи, а саме PostgreSQL над NoSQL-рішеннями, які можуть надати більшої гнучкості структури, але за рахунок гарантій транзакції, є визначальними.

Також PostgreSQL підтримує JSON типи даних, що дозволяє ефективно зберігати дані транзакцій, налаштування користувачів та конфігурації експорту без необхідності створення додаткових таблиць. Система індексування, включаючи часткові та функціональні індекси, забезпечує швидку обробку аналітичних запитів навіть при мільйонах записів транзакцій. Окрім того, він має систему резервного копіювання, що є важливим для розробки фінансових застосунків.

React [30] було обрано для розробки користувацького інтерфейсу, як основну JavaScript бібліотеку як стандарт для розробки клієнтської частини, з великою екосистемою. React забезпечує декларативний підхід до створення інтерфейсів, що робить код більш передбачуваним та легким для налаштування. Віртуальний DOM оптимізує оновлення інтерфейсу, що особливо важливо при відображенні великих об'ємів даних.

Компонентна архітектура ідеально підходить для подібного роду застосунків, де багато елементів інтерфейсу повторюються. Форми введення

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

транзакцій, таблиці з даними, графіки та діаграми можуть бути реалізовані як незалежні один від одного компонент з власною логікою.

Для стилізації використовується фреймворк Tailwind CSS [37], що забезпечує швидку розробку інтерфейсу з приємним дизайном. Tailwind дозволяє створювати адаптивні компоненти без написання власного CSS коду, використовуючи готові класи для всіх основних стилів. Він запобігає дублюванню CSS коду та забезпечує легку підтримку стилів всіх класів, вони мають зрозумілі назви та передбачувану поведінку. Вбудована система дизайну з палітрою кольорів, типографією та відступами гарантує візуальну цілісність всіх компонентів.

Для побудови графіків та діаграм фінансової аналітики було обрано Chart.js [7] як основну бібліотеку візуалізації. Вона забезпечує широкий спектр типів графіків, від простих кругових діаграм для відображення структури витрат до складних лінійних графіків з множинними осями для аналізу трендів доходів та витрат у часі. Бібліотека має відмінну продуктивність при роботі з великими даними та підтримує інтерактивність, наприклад користувачі можуть наводити курсор на елементи графіків для отримання детальної інформації, масштабувати часові періоди, та фільтрувати дані. Адаптивність Chart.js забезпечує коректне відображення графіків на різних розмірах екранів.

Для організації процесу розробки та збірки клієнтського середовища використовується фреймворк Vite [38]. Він забезпечує швидкий холодний старт при розробці завдяки використанню ES модулів та надзвичайно HMR [15], що значно прискорює цикл розробки. При збірці Vite використовує Rollup [34] для оптимізації коду з tree-shaking та code splitting, які розділяють код на модулі для підвищення продуктивності.

Для контролю якості коду налаштовано ESLint [12] з суворими правилами для React застосунків та Prettier [28] для автоматичного форматування. Це забезпечує цілісність коду та запобігає типовим помилкам.

Для роботи з HTTP запитами використовується Axios [5] як HTTP клієнт з підтримкою модуля перевірки повідомлень, більш відомий як «interceptors»

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

для автоматичної обробки автентифікації та помилок. Бібліотека забезпечує зручну обробку асинхронних операцій та автоматичне перетворення JSON відповідей.

Перевірка форм реалізована за допомогою хуків з інтеграцією Yup [42] для схем перевірки. Це поєднання забезпечує високу продуктивність при роботі з великими формами завдяки мінімальній кількості ре-рендерів та гнучкій перевірці.

Для роботи з датами використовується Day.js [11] як легша альтернатива Moment.js. Бібліотека забезпечує всі необхідні операції з датами для фінансових звітів - форматування, обчислення періодів, часові зони та локалізацію.

Мінімальні вимоги до серверного обладнання для розгортання системи включають 2 ядра CPU з частотою не менше 2.4 GHz для забезпечення швидкої обробки API запитів та виконання складних аналітичних запитів до БД. Оперативна пам'ять має становити не менше 4 GB для комфортної роботи PHP-FPM процесів. Дисковий простір залежить від кількості користувачів та обсягу даних, але рекомендований мінімум становить 50 GB SSD накопичувача для забезпечення швидкого доступу до БД. Для комерційного використання середовища рекомендується використання RAID 1 для забезпечення стійкості даних. Мережевий канал має забезпечувати пропускну здатність не менше 100 МБ/с для обслуговування одночасних запитів користувачів та синхронізації з зовнішніми API курсів валют.

Для комфортної роботи з застосунком користувачам рекомендується використовувати сучасні веб-браузери з підтримкою ES6+. Мінімальні вимоги до оперативної пам'яті клієнтського пристрою становлять 4 GB для забезпечення плавної роботи з великими таблицями транзакцій та складними графіками.

Стабільне інтернет-з'єднання зі швидкістю не менше 1 МБ/с необхідне для синхронізації даних та завантаження динамічного контенту. Застосунок

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

підтримує автономний режим для перегляду раніше завантажених даних, але для створення нових транзакцій необхідне активне з'єднання.

Висновки до розділу 1

У результаті проведеного аналізу було визначено основні напрямки використання інформаційних технологій для створення ефективного органайзера з керування особистими фінансами. Детальне дослідження предметної області дозволило сформулювати чіткі технічні вимоги до системи та обґрунтувати архітектурні рішення.

Аналіз існуючих аналогів виявив основні недоліки сучасних рішень на ринку фінансових застосунків. Wallet by BudgetBakers демонструє потужний функціонал, але має суттєві обмеження у безкоштовній версії. Spendee робить акцент на візуальній привабливості, поступаючись практичності. MoneyManager пропонує простоту використання, але не має достатніх аналітичних можливостей. Bluescoins забезпечує найширший функціонал, але його складність відлякує більшість користувачів.

На основі виявлених недоліків було обґрунтовано необхідність створення збалансованого рішення, яке поєднує простоту використання з потужними аналітичними можливостями, забезпечує безкоштовний доступ до основного функціоналу та орієнтується на потреби українського ринку.

Обраний технологічний стек з Laravel для серверу, React для клієнту, PostgreSQL для БД та Tailwind CSS для стилізації забезпечує створення сучасного, продуктивного та масштабованого рішення. Модульна архітектура дозволить легко розширювати функціонал системи без кардинального переписування коду.

Визначені вимоги до апаратного забезпечення є помірними та доступними для більшості потенційних користувачів, що забезпечить широке впровадження системи серед цільової аудиторії.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2 ПРОЕКТУВАННЯ ОРГАНАЙЗЕРА

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Проектування веб системи з керування особистими фінансами розпочинається з детального аналізу варіантів використання та визначення його об'єктно-орієнтованої структури. Основою для проектування стала діаграма варіантів використання, яка відображає всі можливі сценарії взаємодії користувачів з системою.

Система передбачає взаємодію з двома типами акторів. Користувач являє собою основного актора системи, який має доступ до всього функціоналу керування особистими фінансами - від створення транзакцій до аналізу ринку. Адміністратор системи відповідає за налаштування системи, керування користувачами та забезпечення стабільної роботи застосунку. Діаграма варіантів використання представлена на рис. 2.1 та демонструє організацію функціональності системи у групи відповідно до області використання.

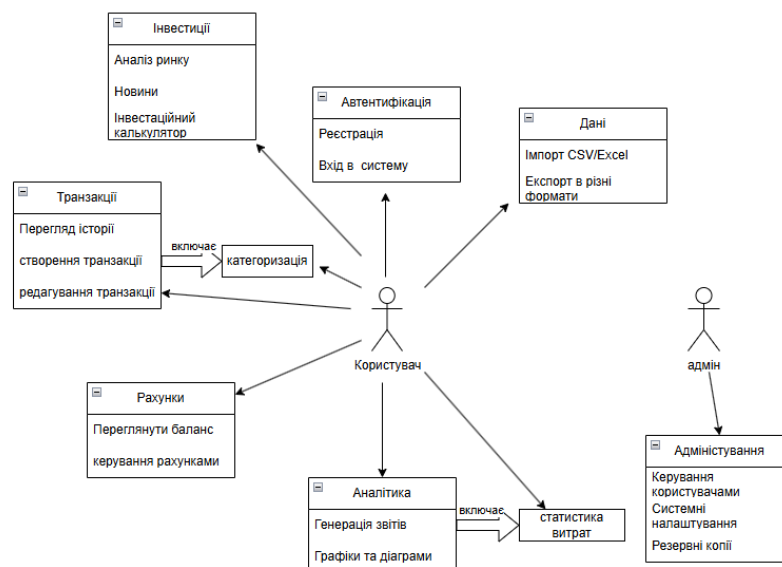


Рисунок 2.1 - Діаграма використання системи

Опис діаграми варіантів використання наведено в табл. 2.1 - 2.8.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Таблиця 2.1

Опис прецеденту «Автентифікація»

Назва варіанту використання	Автентифікація
Основні актори	Користувач, Адміністратор
Короткий опис	Користувачі можуть зареєструватися в системі та увійти до свого облікового запису для доступу до функціоналу керування фінансами

Таблиця 2.2

Опис прецеденту «Керування транзакціями»

Назва варіанту використання	Керування транзакціями
Основні актори	Користувач
Короткий опис	Користувачі можуть створювати, редагувати та переглядати транзакції доходів та витрат з автоматичною категоризацією

Таблиця 2.3

Опис прецеденту «Керування рахунками»

Назва варіанту використання	Керування рахунками
Основні актори	Користувач
Короткий опис	Користувачі можуть створювати різні типи рахунків та переглядати баланси.

Таблиця 2.4

Опис прецеденту «Аналітика та звіти»

Назва варіанту використання	Аналітика та звіти
Основні актори	Користувач
Короткий опис	Система надає аналітичні засоби, генерацію звітів та інтерактивні графіки для аналізу фінансових показників.

Таблиця 2.5

Опис прецеденту «Робота з даними»

Назва варіанту використання	Робота з даними
Основні актори	Користувач
Короткий опис	Користувачі можуть імпортувати дані з CSV/Excel файлів та експортувати свої фінансові дані в різних форматах.

Таблиця 2.6

Опис прецеденту «Інвестиції»

Назва варіанту використання	Інвестиції
Основні актори	Користувач
Короткий опис	Користувачі можуть відстежувати інвестиції, аналізувати ринок, дивитись останні економічні новини та використовувати інвестиційний калькулятор.

Таблиця 2.7

Опис прецеденту «Адміністрування системи»

Назва варіанту використання	Адміністрування системи
Основні актори	Адміністратор
Короткий опис	Адміністратори можуть керувати користувачами, налаштовувати систему та створювати резервні копії даних.

Таблиця 2.8

Опис прецеденту «Категоризація»

Назва варіанту використання	Категоризація
Основні актори	Користувач
Короткий опис	Система забезпечує автоматичну та ручну категоризацію транзакцій для кращого аналізу витрат.

Основна ціль розробки веб-орієнтованої системи полягає в створенні онлайн-інструменту, який дозволяє користувачам ефективно керувати своїми фінансами, аналізувати витрати та планувати бюджет. Система спрямована на забезпечення зручного та надійного механізму для ведення фінансового обліку з можливістю глибокого аналізу та прогнозування. Основна мета полягає у сприянні формуванню фінансової грамотності користувачів шляхом надання інструментів для детального аналізу витрат, планування бюджету та відстеження фінансових цілей. Система надає можливість користувачам вести повний облік доходів та витрат, категоризація транзакції, аналізувати тренди та приймати обґрунтовані фінансові рішення.

Адміністратор має можливість керувати обліковими записами користувачів системи, включаючи створення, редагування та деактивацію

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

облікових записів. Адміністратор налаштовує системні параметри, такі як валюти за замовчуванням, категорії транзакцій та інтеграційні налаштування. Функція створення резервних копій забезпечує збереження критично важливих фінансових даних користувачів. Адміністратор також здійснює моніторинг роботи системи та керує глобальними категоріями транзакцій.

Звичайний користувач має доступ до повного функціоналу системи керування особистими фінансами. Користувач проходить автентифікацію в системі для доступу до своїх даних та створює різні типи транзакцій доходів і витрат з автоматичною категоризацією. Система дозволяє керувати множинними рахунками різних типів, переглядати баланси та здійснювати перекази між рахунками з конвертацією валют. Користувач має можливість генерувати детальні звіти та аналітику, імпортувати дані з зовнішніх джерел та експортувати свої фінансові дані в різних форматах.

Система повинна забезпечувати безпечну автентифікацію користувачів з використанням токенів доступу Laravel Sanctum [18], що гарантує захист персональних фінансових даних. Система надає можливість створення множинних рахунків різних типів з підтримкою різних валют та автоматичною конвертацією за актуальними курсами. Автоматична категоризація транзакцій здійснюється на основі аналізу історичних даних та шаблонів користувача, що значно спрощує процес ведення фінансового обліку. Система генерує інтерактивні звіти та графіки з використанням Chart.js для візуалізації фінансових трендів та аналізу витрат. Підтримка імпорту та експорту даних реалізована для форматів CSV, Excel, PDF та OFX, що забезпечує інтеграцію з іншими фінансовими системами. Інтеграція з зовнішніми API забезпечує отримання актуальних курсів валют та котирувань акцій для інвестиційного модуля.

Система спроектована з урахуванням зручності користування, де час освоєння системи для нового користувача не повинен перевищувати десяти хвилин завдяки інтуїтивному інтерфейсу та логічній структурі навігації. Час відповіді системи на дії користувача оптимізовано до максимум двох секунд

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

для забезпечення комфортної роботи з фінансовими даними. Інтерфейс розроблено як повністю адаптивний з використанням Tailwind CSS, що забезпечує коректне відображення на різних пристроях від смартфонів до великих моніторів. Продуктивність системи розрахована на одночасну роботу до тисячі користувачів з швидкою обробкою великих обсягів транзакцій завдяки оптимізації запитів PostgreSQL та використанню індексів.

Клієнтське середовище вимагає наявності 4 Гігабайтів оперативної пам'яті та сучасного веб-браузера з підтримкою стандартів ES6+. Стабільне інтернет-з'єднання необхідне для синхронізації даних та отримання актуальної інформації про курси валют. Серверне середовище базується на PHP версії 8.1 або вище з фреймворком Laravel 12 для забезпечення сучасної архітектури та безпеки. База даних PostgreSQL версії 13 або вище забезпечує надійне зберігання фінансових даних з підтримкою ACID транзакцій. Сервер повинен мати мінімум вісім гігабайт оперативної пам'яті та SSD накопичувач для забезпечення швидкого доступу до БД.

Аналіз функціональних вимог дозволив виділити основні сутності, що забезпечать програмну реалізацію застосунку. Діаграма класів відображає структуру об'єктно-орієнтованої архітектури системи з чітким розподілом залежності між компонентами (рис. 2.2).

Клас User керує користувачами програми. Він забезпечує реєстрацію нових облікових записів через register(), автентифікацію через login(), завершення сесій через logout(), генерацію API токенів через createToken() та перевірку прав доступу через hasRole().

Клас Currency обробляє валютні операції. Включає getRateFor() для курсів обміну, updateRate() для актуалізації курсу, convertToBaseCurrency() для переведення в основну валюту та convertFromBaseCurrency() для зворотної конвертації.

Клас Role управляє правами користувачів. Надає assignToUser() для призначення статусу користувачеві та hasPermission() для верифікації дозволу.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

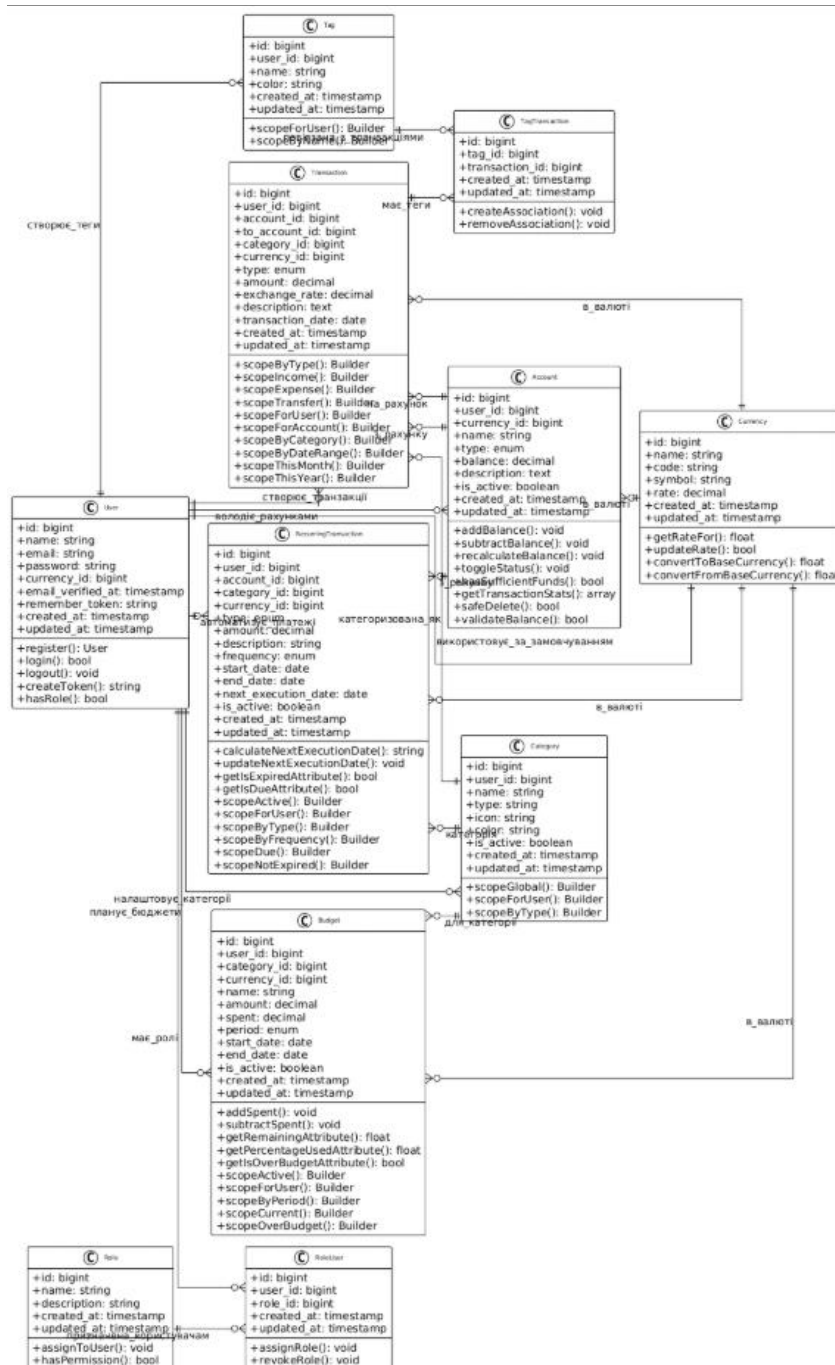


Рисунок 2.2 - Діаграма класів

Клас Account працює з фінансовими рахунками. Містить addBalance() для поповнення, subtractBalance() для списання, recalculateBalance() для оновлення залишку, toggleStatus() для активації/деактивації, hasSufficientFunds() для перевірки наявності коштів, getTransactionStats() для аналітики операцій, safeDelete() для захищеного видалення та validateBalance() для контролю коректності суми.

Клас Transaction обслуговує фінансові операції. Реалізує scopeByType() для групування за видом, scopeIncome() для доходів, scopeExpense() для витрат, scopeTransfer() для переказів, scopeForUser() для операцій користувача, scopeForAccount() для операцій рахунка, scopeByCategory() для категоризації, scopeByDateRange() для часових проміжків, scopeThisMonth() для поточного місяця та scopeThisYear() для поточного року.

Клас Category структурує категорії операцій. Надає scopeGlobal() для загальних категорій, scopeForUser() для персональних категорій та scopeByType() для групування за видом.

Клас Tag організовує мітки операцій. Реалізує scopeForUser() для персональних міток та scopeByName() для пошуку за назвою.

Клас Budget контролює бюджетне планування. Включає addSpent() для додавання витрат, subtractSpent() для зменшення витрат, getRemainingAttribute() для залишку, getPercentageUsedAttribute() для відсотка використання, getIsOverBudgetAttribute() для контролю перевищення, scopeActive() для активних планів, scopeForUser() для персональних бюджетів, scopeByPeriod() для періодів, scopeCurrent() для поточних планів та scopeOverBudget() для перевищених лімітів.

Клас RecurringTransaction автоматизує регулярні операції. Містить calculateNextExecutionDate() для розрахунку наступної дати, updateNextExecutionDate() для оновлення дати, getIsExpiredAttribute() для контролю терміну, getIsDueAttribute() для перевірки необхідності виконання, scopeActive() для активних операцій, scopeForUser() для персональних операцій, scopeByType() для групування за видом, scopeByFrequency() для частоти, scopeDue() для операцій до виконання та scopeNotExpired() для активних операцій.

Допоміжний клас TagTransaction організовує зв'язки міток та операцій через createAssociation() для створення зв'язку та removeAssociation() для його видалення.

Допоміжний клас RoleUser організовує зв'язки ролей та користувачів через assignRole() для призначення статусу та revokeRole() для його скасування.

Запропонована архітектура демонструє комплексну структуру об'єктів програми керування персональними фінансами. Побудована на основі міграцій та об'єктів Laravel. Кожна сутність виконує специфічні функції та містить набір операцій для реалізації логіки предметної сфери. Score функції Laravel забезпечують ефективну фільтрацію інформації, а автоматичне видалення підтримує цілісність даних.

2.2 Розробка бази даних системи

Для ефективного функціонування системи керування особистими фінансами була розроблена реляційна база даних, яка забезпечує надійне зберігання та швидкий доступ до фінансових даних користувачів. Проектування бази даних здійснювалося з урахуванням принципів нормалізації, масштабованості та продуктивності.

База даних системи складається з 11 основних таблиць, які відображають ключові сутності предметної області (табл. 2.9 – 2.19).

Таблиця 2.9

Структура таблиці users

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
name	VARCHAR(255)	NOT NULL	Ім'я
email	VARCHAR(255)	NOT NULL, UNIQUE	пошта
password	VARCHAR(255)	NOT NULL	Хешований пароль
currency_id	BIGINT	FOREIGN KEY	Валюта за замовчуванням
email_verified_at	TIMESTAMP	NULL	Час верифікації електронної пошти
remember_token	VARCHAR(100)	NULL	Токен для запам'ятовування сесії
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.10

Структура таблиці currencies

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
name	VARCHAR(255)	NOT NULL	Назва валюти
code	VARCHAR(3)	NOT NULL, UNIQUE	ISO код валюти
symbol	VARCHAR(10)	NOT NULL	Символ валюти
rate	DECIMAL(10,4)	NOT NULL, DEFAULT 1.0000	Курс обміну до базової валюти
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.11

Структура таблиці roles

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
name	VARCHAR(255)	NOT NULL, UNIQUE	Назва ролі
description	TEXT	NULL	Опис ролі та її повноважень
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.12

Структура таблиці accounts

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
user_id	BIGINT	NOT NULL, FOREIGN KEY	Власник рахунку
currency_id	BIGINT	NOT NULL, FOREIGN KEY	Валюта рахунку
name	VARCHAR(255)	NOT NULL	Назва рахунку
type	ENUM	NOT NULL	Тип рахунку (cash, bank, credit, investment)
balance	DECIMAL(15,2)	NOT NULL, DEFAULT 0.00	Поточний баланс
description	TEXT	NULL	Опис рахунку
is_active	BOOLEAN	NOT NULL, DEFAULT TRUE	Статус активності
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.13

Структура таблиці categories

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
user_id	BIGINT	NULL, FOREIGN KEY	Власник категорії (NULL для глобальних)
name	VARCHAR(255)	NOT NULL	Назва категорії
type	VARCHAR(50)	NOT NULL	Тип категорії (income, expense)
icon	VARCHAR(255)	NULL	Іконка категорії
color	VARCHAR(7)	NULL	Колір категорії в HEX форматі
is_active	BOOLEAN	NOT NULL, DEFAULT TRUE	Статус активності
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.14

Структура таблиці transactions

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
user_id	BIGINT	NOT NULL, FOREIGN KEY	Власник транзакції
account_id	BIGINT	NOT NULL, FOREIGN KEY	Рахунок списання
to_account_id	BIGINT	NULL, FOREIGN KEY	Рахунок зарахування (для переказів)
category_id	BIGINT	NULL, FOREIGN KEY	Категорія транзакції
currency_id	BIGINT	NOT NULL, FOREIGN KEY	Валюта операції
type	ENUM	NOT NULL	Тип операції (income/expense)
amount	DECIMAL(15,2)	NOT NULL	Сума операції
exchange_rate	DECIMAL(10,4)	NOT NULL, DEFAULT 1.0000	Курс обміну на момент операції
description	TEXT	NULL	Опис транзакції
transaction_date	DATE	NOT NULL	Дата проведення операції
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.15

Структура таблиці tags

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
user_id	BIGINT	NOT NULL, FOREIGN KEY	Власник тегу
name	VARCHAR(255)	NOT NULL	Назва тегу
color	VARCHAR(7)	NULL	Колір тегу в HEX форматі
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.16

Структура таблиці budgets

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
user_id	BIGINT	NOT NULL, FOREIGN KEY	Власник бюджету
category_id	BIGINT	NOT NULL, FOREIGN KEY	Категорія бюджету
currency_id	BIGINT	NOT NULL, FOREIGN KEY	Валюта бюджету
name	VARCHAR(255)	NOT NULL	Назва бюджету
amount	DECIMAL(15,2)	NOT NULL	Заплановано до витрати
spent	DECIMAL(15,2)	NOT NULL, DEFAULT 0.00	Фактично витрачено
period	ENUM	NOT NULL	Період (weekly, monthly, quarterly, yearly)
start_date	DATE	NOT NULL	Дата початку періоду
end_date	DATE	NOT NULL	Дата закінчення періоду
is_active	BOOLEAN	NOT NULL, DEFAULT TRUE	Статус активності
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.17

Структура таблиці recurring_transactions

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
user_id	BIGINT	NOT NULL, FOREIGN KEY	Власник транзакції
account_id	BIGINT	NOT NULL, FOREIGN KEY	Рахунок операції
category_id	BIGINT	NULL, FOREIGN KEY	Категорія операції

Продовження табл. 2.17

Поле	Тип даних	Обмеження	Опис
currency_id	BIGINT	NOT NULL, FOREIGN KEY	Валюта операції
type	ENUM	NOT NULL	Тип операції (income, expense)
amount	DECIMAL(15,2)	NOT NULL	Сума операції
description	VARCHAR(255)	NULL	Опис операції
frequency	ENUM	NOT NULL	Частота (daily, weekly, monthly, yearly)
start_date	DATE	NOT NULL	Дата початку
end_date	DATE	NULL	Дата закінчення
next_execution_date	DATE	NOT NULL	Наступна дата виконання
is_active	BOOLEAN	NOT NULL, DEFAULT TRUE	Статус активності
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.18

Структура таблиці role_user

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
user_id	BIGINT	NOT NULL, FOREIGN KEY	Ідентифікатор користувача
role_id	BIGINT	NOT NULL, FOREIGN KEY	Ідентифікатор ролі
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

Таблиця 2.19

Структура таблиці tag_transaction

Поле	Тип даних	Обмеження	Опис
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	ідентифікатор
tag_id	BIGINT	NOT NULL, FOREIGN KEY	Ідентифікатор тегу
transaction_id	BIGINT	NOT NULL, FOREIGN KEY	Ідентифікатор транзакції
created_at	TIMESTAMP	NULL	Час створення запису
updated_at	TIMESTAMP	NULL	Час останнього оновлення

При розробці структури бази даних були враховані наступні принципи та особливості. Нормалізація даних забезпечує приведення усіх таблиць до третьої нормальної форми для мінімізації дублювання інформації та забезпечення цілісності даних. Підтримка різних валют реалізована через те, що кожна фінансова операція містить посилання на валюту та курс обміну на момент створення, що дозволяє користувачам працювати з різними валютами одночасно.

Гнучкість категоризації досягається через підтримку як глобальних категорій для всіх користувачів, так і персональних категорій для індивідуальних потреб через nullable поле `user_id`. Аудит змін забезпечується тим, що усі таблиці містять поля `created_at` та `updated_at` для відстеження часу створення та модифікації записів, що забезпечує можливість аудиту дій користувачів.

Каскадне видалення реалізовано через налаштовані зв'язки між таблицями, які забезпечують автоматичне видалення залежних записів при видаленні основних сутностей для збереження цілісності даних. Підтримка переказів здійснюється через те, що таблиця `transactions` містить поле `to_account_id` для реалізації переказів між рахунками користувачів з автоматичним розрахунком курсів обміну. Регулярні операції підтримуються окремою таблицею `recurring_transactions`, яка дозволяє налаштовувати автоматичні платежі з різною частотою виконання та контролем термінів дії.

Для забезпечення цілісності та безпеки даних у базі реалізовані наступні механізми. Зовнішні ключі з обмеженнями підтримують цілісність між пов'язаними таблицями. Обов'язкові поля для критично важливих атрибутів забезпечуються через NOT NULL обмеження. Унікальні індекси на поля `email` та код валюти запобігають дублюванню записів. Перевірка типів даних та обмежень здійснюється на рівні бази даних через ENUM типи. Індексация зовнішніх ключів оптимізує швидкість виконання запитів. Розроблена структура бази даних забезпечує надійне зберігання фінансової інформації користувачів та підтримує всі необхідні операції для функціонування системи

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

керування особистими фінансами з можливістю масштабування та розширення функціональності.

2.3 Проектування та реалізація алгоритмів роботи системи

Для забезпечення коректного функціонування системи керування особистими фінансами були розроблені та реалізовані ключові алгоритми, які обробляють фінансові операції, розрахунки та автоматизовані процеси. Алгоритми проектувалися з урахуванням точності фінансових розрахунків, продуктивності та масштабованості системи.

Основним алгоритмом системи є розрахунок поточного балансу фінансових рахунків користувачів. Алгоритм враховує всі типи операцій та автоматично оновлює баланси при проведенні нових транзакцій.

Алгоритм працює наступним чином. Спочатку система отримує початковий баланс рахунку з таблиці accounts. Потім виконується вибірка всіх транзакцій для конкретного рахунку з урахуванням типу операції. Для доходних операцій сума додається до балансу, для витратних операцій сума віднімається від балансу. При переказах між рахунками сума списується з рахунку відправника та зараховується на рахунок отримувача з урахуванням курсів валют. Валютна конвертація здійснюється автоматично якщо валюта операції відрізняється від валюти рахунку. Система використовує збережений курс обміну на момент проведення операції для забезпечення точності історичних розрахунків. Оновлений баланс зберігається в полі balance таблиці accounts та використовується для подальших розрахунків.

Система підтримує багато валют через спеціальний алгоритм конвертації, який забезпечує точні розрахунки між різними валютами з урахуванням актуальних курсів обміну.

Алгоритм конвертації спочатку перевіряє чи потрібна конвертація шляхом порівняння кодів валют. Якщо валюти однакові, конвертація не виконується. При необхідності конвертації система отримує актуальний курс

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

цільової валюти з таблиці currencies. Розрахунок виконується через базову валюту системи за формулою результат дорівнює сума поділена на курс вихідної валюти та помножена на курс цільової валюти. Для забезпечення точності всі валютні розрахунки виконуються з використанням типу DECIMAL з чотирма знаками після коми. Система зберігає курс обміну на момент операції в полі exchange_rate таблиці transactions для можливості точного відтворення історичних розрахунків. При зміні курсів валют нові операції використовують оновлені курси, а історичні дані залишаються незмінними.

Для ефективного планування фінансів реалізований алгоритм автоматичного контролю виконання бюджетних планів користувачів з підтримкою різних часових періодів та категорій витрат.

Алгоритм працює наступним чином. При створенні нової витратної операції система автоматично перевіряє чи існують активні бюджети для категорії операції. Для знайдених бюджетів виконується перевірка чи операція потрапляє в часовий період бюджету. Якщо умови виконуються, сума операції додається до поля spent відповідного бюджету. Система розраховує відсоток використання бюджету за формулою відсоток дорівнює витрачена сума поділена на заплановану суму та помножена на сто. При перевищенні запланованої суми бюджету система встановлює відповідний статус та може надсилати повідомлення користувачу. Залишок бюджету розраховується як різниця між запланованою та витраченою сумами з урахуванням валютної конвертації.

Для автоматизації повторюваних фінансових операцій розроблений алгоритм обробки регулярних платежів, який автоматично створює транзакції згідно з налаштованим розкладом.

Алгоритм виконується по розкладу та спочатку вибирає всі активні регулярні платежі з датою виконання не пізніше поточної дати. Для кожного знайденого платежу система перевіряє чи не закінчився термін його дії та чи достатньо коштів на рахунку для виконання операції. При успішній перевірці

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

створюється нова транзакція з параметрами регулярного платежу. Після створення транзакції система розраховує наступну дату виконання залежно від частоти платежу. Для щоденних платежів додається один день, для щотижневих сім днів, для щомісячних один місяць, для щорічних один рік. Розрахована дата зберігається в полі `next_execution_date` для наступного циклу обробки. При виникненні помилок або недостатності коштів система логує інформацію та може призупинити виконання платежу.

Для зручності користувачів реалізований алгоритм автоматичної категоризації фінансових операцій на основі історичних даних та правил системи.

Алгоритм аналізує опис нової транзакції та порівнює його з описами попередніх операцій користувача. Система шукає схожі операції за ключовими словами та сумами платежів. При знаходженні відповідності, алгоритм пропонує найбільш ймовірну категорію на основі частоти використання в минулому. Додатково система підтримує правила автоматичної категоризації для типових операцій. Наприклад платежі в продуктових магазинах автоматично відносяться до категорії продукти харчування, платежі за комунальні послуги до категорії житлово-комунальні послуги. Користувач може створювати власні правила категоризації та корегувати запропоновані системою варіанти.

Розроблені алгоритми забезпечують автоматизацію ключових процесів системи керування особистими фінансами та гарантують точність і надійність фінансових розрахунків при роботі з різними валютами операціями та складними бюджетними планами.

2.4 Реалізація функціоналу органайзера

Програмна реалізація системи здійснювалася з використанням сучасних веб-технологій та фреймворків, які забезпечують високу продуктивність, безпеку та зручність використання. Архітектура системи побудована за

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

принципом розділення клієнтської та серверної частин з взаємодією через REST API.

Серверна частина реалізована з використанням PHP Laravel версії 12, який забезпечує швидку розробку та надійну роботу вебзастосунків. Laravel надає вбудовані засоби для роботи з базами даних, автентифікації користувачів, перевірки даних та маршрутизації запитів. Для роботи з базою даних використовується Eloquent ORM, який дозволяє зручно працювати з реляційними даними через об'єктно-орієнтований підхід. Міграції бази даних забезпечують можливість відкату змін. Автентифікація користувачів реалізована через Laravel Sanctum, який надає можливості створення API токенів для безпечного доступу до ресурсів системи. Паролі користувачів зберігаються в зашифрованому вигляді з використанням bcrypt алгоритму. Реалізована система ролей через модель Role та проміжну таблицю role_user для контролю доступу до різних функцій системи. Система включає спеціалізовані сервіси для роботи з фінансовими даними. Сервіс YahooFinanceService забезпечує отримання котирувань акцій та ринкових новин через Python [29] скрипти.

Користувацький інтерфейс розроблений з використанням React та Vite як системи збірки, що забезпечує швидку розробку та оптимізовану продуктивність. Для стилізації використовується Tailwind CSS, який надає утилітарні класи для створення сучасного та адаптивного дизайну. Маршрутизація в програмі реалізована через React Router, який дозволяє створювати одно сторінковий застосунок з плавними переходами між різними розділами. Компонентна архітектура включає модулі для керування транзакціями, рахунками, аналітикою, інвестиціями та імпорту/експорту даних. Взаємодія з серверною частиною здійснюється через спеціалізований apiClient на базі Axios, який забезпечує централізоване керування HTTP запитами. Реалізована система перехоплення запитів для автоматичного додавання токенів автентифікації та обробки помилок з автоматичним перенаправленням на сторінку входу при закінченні сесії.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

REST API системи побудоване згідно з принципами RESTful архітектури та забезпечує повний набір операцій для керування фінансовими даними. API підтримує стандартні HTTP методи з перевіркою вхідних даних та консистентною структурою JSON відповідей. Система включає розширені можливості для обробки фінансових даних. DataController забезпечує експорт транзакцій у форматах CSV, Excel та OFX з підтримкою фільтрації за рахунками та періодами. InvestmentController надає доступ до ринкової інформації через інтеграцію з Yahoo Finance API (Додаток В). Реалізована підтримка отримання котирувань акцій, історичних даних та новин з автоматичним хешуванням для зменшення навантаження на зовнішні сервіси.

Модуль Overview забезпечує комплексну аналітику фінансових даних з підтримкою різних часових періодів та інтерактивних віджетів. Користувачі можуть налаштовувати дошку згідно з особистими потребами, переміщувати віджети та обирати відображувані дані. Компонент TransactionList надає повнофункціональне керування фінансовими операціями з підтримкою фільтрації за типом, категорією, рахунком та періодом. Реалізована можливість швидкого пошуку та сортування транзакцій з збереженням стану фільтрів. Модуль AccountManager забезпечує керування фінансовими рахунками з підтримкою різних типів валют та автоматичним розрахунком балансів. Система відстежує зміни балансів у реальному часі та надає детальну статистику по кожному рахунку. DataManager реалізує комплексні можливості імпорту та експорту фінансових даних. Підтримуються формати CSV, Excel та OFX. Користувачі можуть налаштовувати правила імпорту та переглядати результати обробки.

Система включає комплекс заходів для забезпечення безпеки користувацьких даних. Всі HTTP запити використовують HTTPS протокол для шифрування передачі даних. Реалізований захист від CSRF атак через Laravel Sanctum токени. Ринкові дані оброблюються через YahooFinanceService, створені оптимізовані індекси для прискорення виконання запитів за датами та типами операцій (Додаток Г). Система логування дозволяє відстежувати

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

помилки та відстежувати продуктивність програмного продукту. Всі критичні операції логуються з деталями для можливості аудиту та відновлення даних у випадку проблем. Розроблена система керування особистими фінансами забезпечує надійне зберігання фінансової інформації користувачів, підтримує всі необхідні операції для ефективного планування бюджету та надає зручний інтерфейс для щоденного використання.

Висновки до розділу 2

У другому розділі було проведено детальне проектування та реалізацію системи керування особистими фінансами. Розроблена діаграма класів відображає основні сутності предметної області та їх взаємозв'язки, забезпечуючи повне покриття функціональних вимог системи через одинадцять взаємопов'язаних класів моделей.

Спроектована структура бази даних включає одинадцять нормалізованих таблиць, які забезпечують надійне зберігання фінансових даних з підтримкою різних валют, категоризації операцій та бюджетного планування. Застосування принципів нормалізації та створення оптимізованих індексів забезпечує високу продуктивність та масштабованість системи. Розроблені алгоритми автоматизують ключові процеси обробки фінансових операцій, включаючи розрахунок балансів, конвертацію валют, контроль бюджету. Алгоритми гарантують точність розрахунків та цілісність даних при виконанні складних валютних операцій. Програмна реалізація виконана з використанням сучасного технологічного стеку Laravel та React з Tailwind CSS, що забезпечує високу якість коду, безпеку даних та зручність використання. Модульна архітектура системи дозволяє легко розширювати функціональність та адаптуватися до змінних вимог користувачів, включаючи інтеграцію з зовнішніми фінансовими сервісами.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОРГАНАЙЗЕРОМ

3.1 Порядок встановлення та налаштування системи

Встановлення та налаштування системи керування особистими фінансами вимагає розгортання всіх її компонентів та налаштування як кожного елементу окремо, так і загальної системи в цілому.

На рис. 3.1 представлено діаграму розгортання системи.

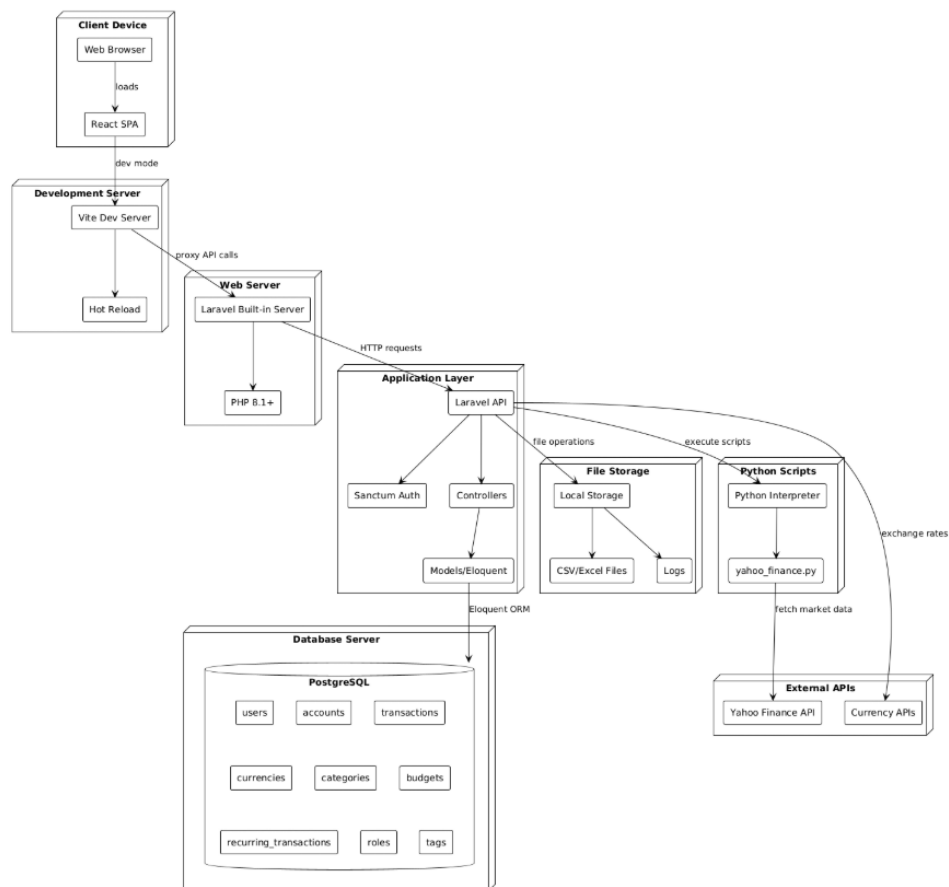


Рисунок 3.1 - Діаграма розгортання системи

Розглянемо детально процес розгортання.

WEB BROWSER - клієнтська частина застосунку, яка завантажується у веб-браузері користувача та забезпечує інтерактивний інтерфейс через React компоненти Dashboard, Overview, Analytics, Transactions, Accounts, Investments та DataManager.

VITE DEV SERVER - система збірки та сервер розробника, який забезпечує швидку розробку, проксування API запитів до Laravel серверу та обслуговування статичних файлів React застосунку.

LARAVEL BUILT-IN SERVER - вбудований PHP сервер Laravel, який працює на порту 8000 через команду «php artisan serve» та забезпечує REST API для всіх операцій системи.

LARAVEL API - серверна частина на PHP Laravel 12, яка включає контролери AuthController, TransactionController, AccountController, DataController, InvestmentController для обробки бізнес-логіки та надання JSON відповідей.

DATABASE SERVER - сервер бази даних PostgreSQL, який зберігає всі дані системи через 11 основних таблиць з підтримкою міграцій Laravel та сідерів для початкових даних.

FILE STORAGE - локальне файлове сховище Laravel у папці storage для обробки завантажених CSV та Excel файлів через UniversalDataExtractor та зберігання логів системи (Додаток А).

PYTHON SCRIPTS - окремий компонент з Python інтерпретатором та скриптом yahoo_finance.py, який виконується Laravel через YahooFinanceService для отримання ринкових даних та новин.

YAHOO FINANCE API [41] - зовнішній сервіс для отримання котирувань акцій, історичних даних та фінансових новин через Python скрипти з можливістю демо даних для демонстрацій (аварійний).

CURRENCY APIS - зовнішні API для отримання актуальних валютних курсів через ExchangeRateService з автоматичним кешуванням та аварійні механізмами з запитами до декількох найпопулярніших українських банків якщо не працює з API НБУ [3] (Додаток Б).

Процес встановлення розпочинається з підготовки серверного середовища. Необхідно встановити PHP [24] версії 8.1 або вище з розширеннями PDO, OpenSSL, JSON та URL. Composer [8] використовується для управління PHP залежностями, Node.js [23] версії 16 або вище для збірки

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

клієнтської частини та Python для роботи з ринковими даними. Серверна частина розгортається через клонування репозиторію та виконання команди «composer install» у папці server для встановлення Laravel framework, Eloquent ORM, Laravel Sanctum та PhpSpreadsheet [25]. У файлі .env налаштовуються параметри підключення до бази даних, ключ застосунку через php artisan key:generate та шлях до Python інтерпретатора через змінну PYTHON_EXECUTABLE. База даних ініціалізується командою «php artisan migrate» для створення всіх 11 таблиць згідно з розробленими міграціями. Команда «php artisan db:seed» наповнює базу початковими даними включаючи ролі Admin та User, глобальні категорії транзакцій та тестові валюти з курсами обміну. Клієнтська частина збирається через «npm install» у папці client для встановлення React, Vite, Tailwind CSS, React Router та Chart.js для візуалізації даних. У файлі .env налаштовується VITE_API_URL що вказує на http://127.0.0.1:8000 для локального розгортання.

Python скрипти налаштовуються через встановлення необхідних пакетів для роботи з Yahoo Finance API. YahooFinanceService виконує yahoo_finance.py скрипти для отримання ринкових даних з автоматичним демо режимом для демонстраційні дані при недоступності зовнішнього API (Додаток Г). Система використовує файлове кешування Laravel замість Redis через налаштування CACHE_DRIVER=file та SESSION_DRIVER=file у файлі .env. Це спрощує розгортання та не вимагає додаткових сервісів для кешування даних.

Запуск системи здійснюється одночасним стартом двох процесів. У папці server виконується «php artisan serve» для підняття API на порту 8000, а у папці client запускається «npm run dev» для старту Vite сервера на порту 3000. Система підтримує імпорт фінансових даних через DataController яка підтримує CSV, Excel та OFX (Додаток Д).

Безпека забезпечується через Laravel Sanctum для автентифікації API, Policy класи для авторизації доступу до ресурсів та перевірки всіх вхідних

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

даних на рівні контролерів. Система логування дозволяє відстежувати всі критичні операції та помилки у папці storage/logs.

3.2 Структура інтерфейсу органайзера

Користувацький інтерфейс системи побудований за принципом одно сторінкового застосунку з інтуїтивною навігацією та адаптивним дизайном. Інтерфейс розроблений з використанням React та Tailwind CSS, що забезпечує сучасний вигляд та швидку роботу на різних пристроях.

При першому заході на сайт користувач перенаправляється на сторінку авторизації або реєстрації (рис. 3.2). Система автентифікації побудована на Laravel Sanctum з підтримкою безпечних API токенів та захистом від CSRF [10] атак.

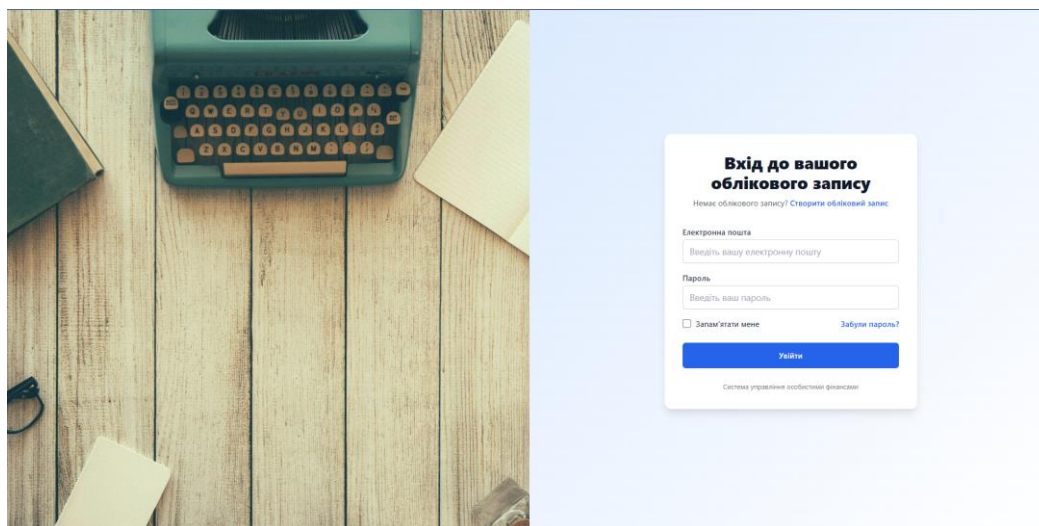


Рисунок 3.2 – Сторінка входу

Форма входу містить поля для введення електронної пошти та пароля з перевіркою на стороні клієнта та сервера. Додатково присутні посилання для відновлення пароля та переходу на сторінку реєстрації нових користувачів.

Для початку роботи з системою створимо профіль з довільними параметрами (рис. 3.3). Процес реєстрації включає введення імені користувача, електронної пошти та безпечного пароля з підтвердженням.

Форма реєстрації забезпечує перевіркою всіх полів у реальному часі з відображенням помилок та підказок для створення надійного пароля. Після

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

успішної реєстрації користувач автоматично перенаправляється на головну сторінку системи.

Create your account

Already have an account? [Sign in](#)

Full name

Olexandr Kyrychenko

Email address

ipz211_kos@student.ztu.edu.ua

Password

.....

Confirm password

.....

Create account

Рисунок 3.3 – Сторінка реєстрації

Після успішного входу в систему користувач потрапляє на головну сторінку Огляду, яка надає загальний огляд фінансового стану через систему віджетів та інтерактивних графіків (рис. 3.4).

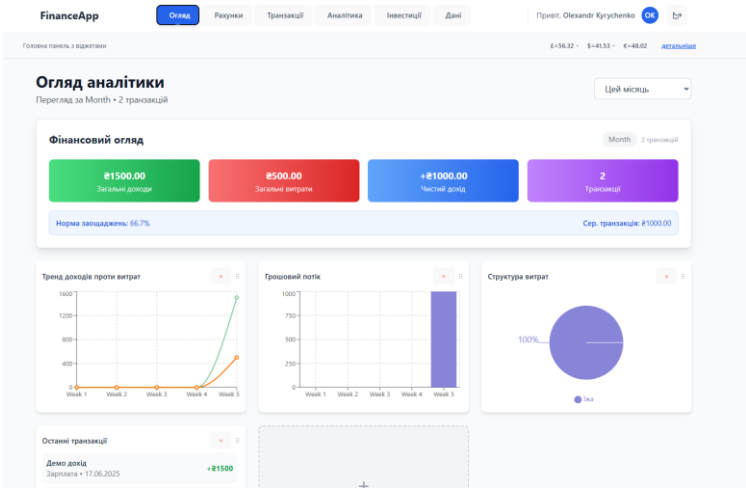


Рисунок 3.4 – Сторінка огляду

Сторінка огляду містить віджети загального балансу всіх рахунків, графік динаміки доходів та витрат за обраний період, діаграму розподілу

витрат по категоріях та список останніх транзакцій. Користувач може налаштовувати відображення віджетів та обирати часові періоди для аналізу.

Перехід до розділу рахунків дозволяє управляти фінансовими рахунками користувача з можливістю створення, редагування та видалення рахунків різних типів (рис. 3.5).

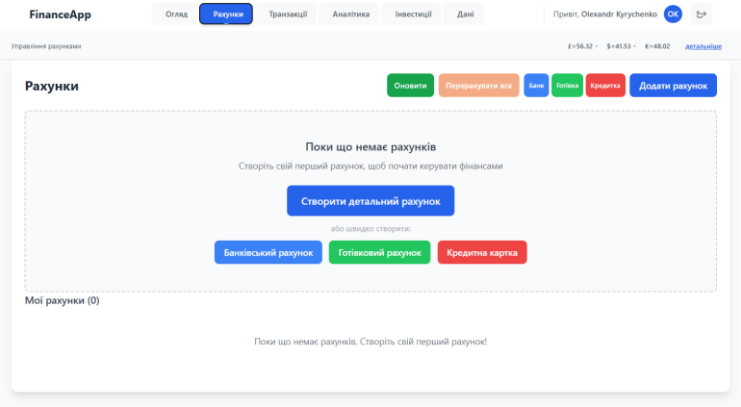


Рисунок 3.5 – Сторінка рахунків

Сторінка рахунків відображає список всіх створених рахунків з інформацією про назву, тип, валюту та поточний баланс. Кожен рахунок має кольорову індикацію типу та іконку для візуального розрізнення готівкових, банківських, кредитних та інвестиційних рахунків.

Створення нового рахунку здійснюється через модальне вікно з формою, що містить всі необхідні поля для налаштування рахунку згідно з потребами користувача (рис. 3.6).

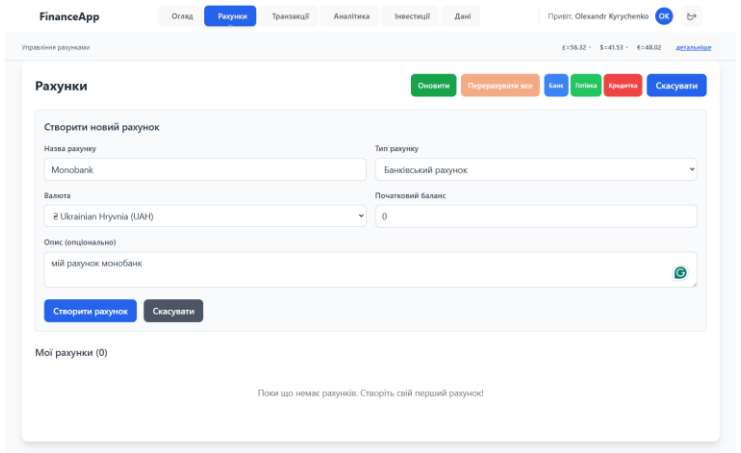


Рисунок 3.6 – Створення рахунку

Форма створення рахунку включає поля для назви, вибору типу рахунку з випадаючого списку, валюти з автодоповненням та опису. Система

автоматично перевіряє введені дані та попереджає про помилки до збереження.

Після створення рахунку він відразу відображається у списку з можливістю подальшого редагування параметрів або видалення при необхідності (рис. 3.7).

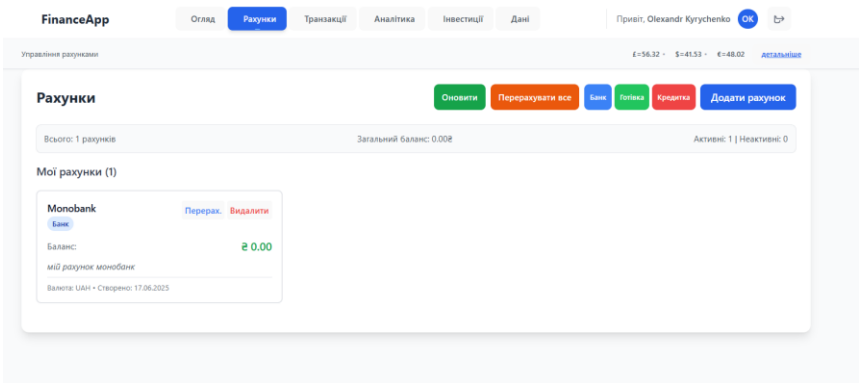


Рисунок 3.7 – Вигляд після створення рахунку

Новий рахунок з’являється у списку з нульовим початковим балансом, який буде автоматично розраховуватися на основі проведених транзакцій.

Розділ Транзакції забезпечує повне управління фінансовими операціями через інтерфейс зі списком транзакцій та розширеними можливостями фільтрації й пошуку (рис. 3.8).

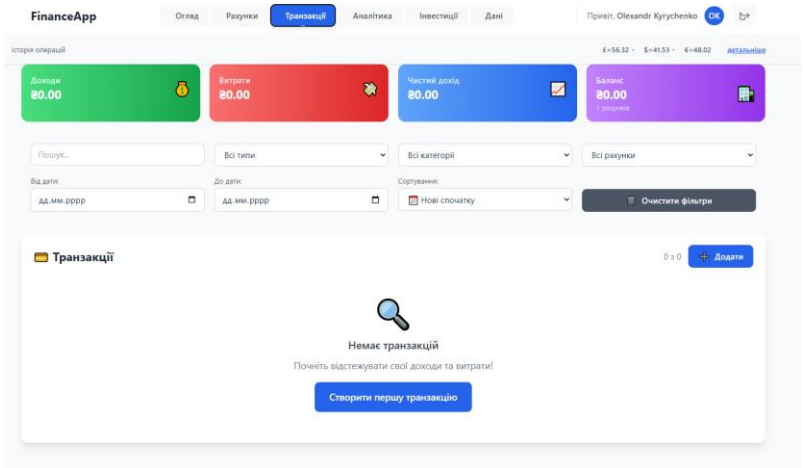


Рисунок 3.8 – сторінка транзакцій

Сторінка транзакцій містить список всіх операцій користувача з можливістю сортування за датою, сумою або категорією. Інтерфейс включає

фільтри за типом операції, періодом, рахунком та категорією для швидкого пошуку потрібних транзакцій.

Під час створення транзакції користувач може додати нову категорію якщо існуючі не підходять для класифікації операції (рис. 3.9).

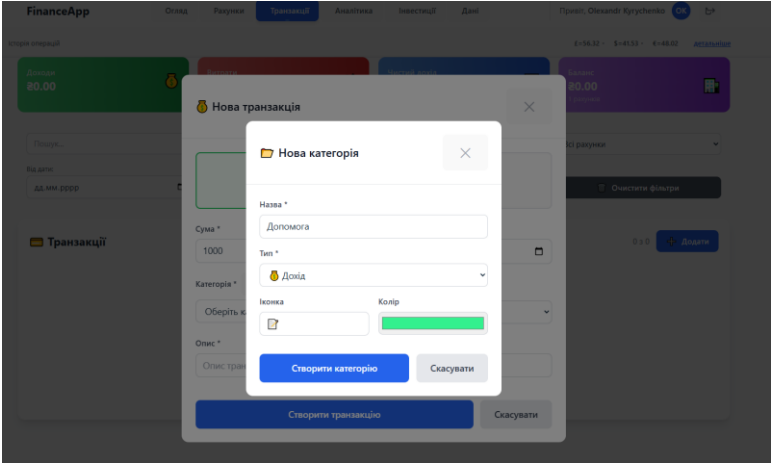


Рисунок 3.9 – створення нової категорії

Форма створення категорії дозволяє вказати назву, тип категорії (дохід або витрата), обрати іконку та колір для візуального оформлення. Створені категорії стають доступними для всіх майбутніх транзакцій користувача.

Створення нової транзакції здійснюється через детальну форму з автоматичним розрахунком курсів валют та підказками для категоризації операцій (рис. 3.10).

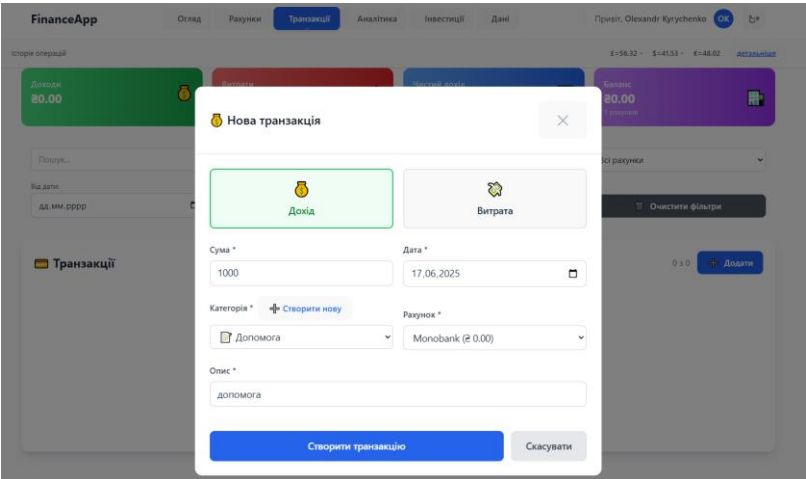


Рисунок 3.10 – Створення транзакції

Форма транзакції включає поля для вибору рахунку з відображенням поточного балансу, категорії операції, введення суми з автоматичною

валютною конвертацією, дати операції та опису. Система автоматично розраховує еквівалент суми в базовій валюті користувача.

Після створення транзакції вона відразу відображається у списку з автоматичним оновленням балансу відповідного рахунку та статистичних показників (рис. 3.11).

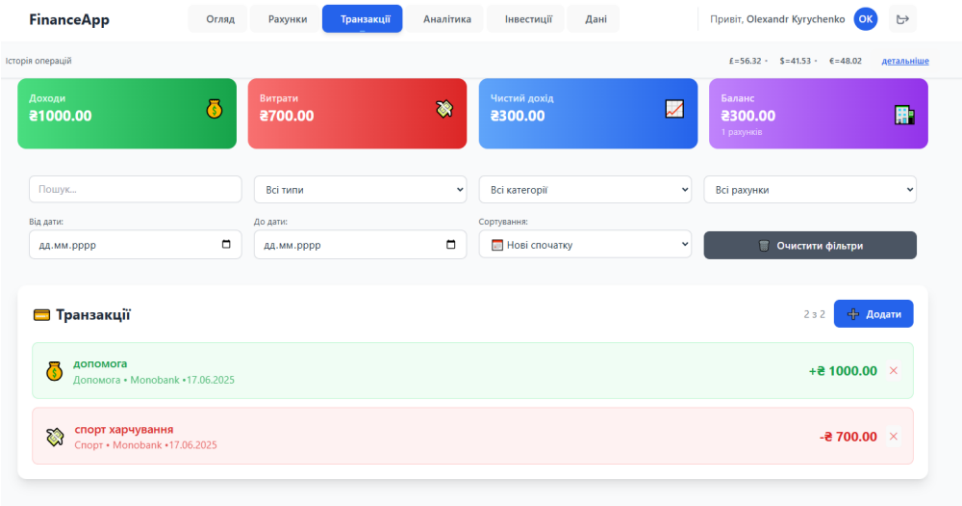


Рисунок 3.11 – Відображення створених транзакцій

Список транзакцій оновлюється у реальному часі з відображенням всіх деталей операції включаючи дату, опис, категорію, суму та рахунок. Кожна транзакція має кольорове кодування за типом операції для швидкого візуального розрізнення доходів та витрат.

Розділ аналітики надає комплексну аналітику фінансових даних через різноманітні графіки, діаграми та звіти з можливістю детального налаштування параметрів відображення.

Верхня частина аналітичної сторінки містить основні КРІ показники включаючи загальний баланс, доходи та витрати за обраний період, а також динаміку зміни балансу у вигляді лінійного графіка з можливістю зміни часового масштабу (рис. 3.12).

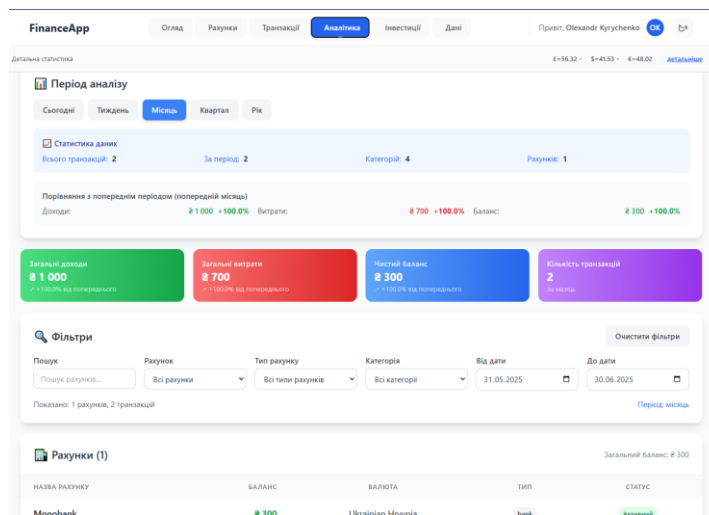


Рисунок 3.12 – Сторінка аналітики (верхня частина)

Нижня частина включає детальну статистику витрат по категоріях у вигляді кругової діаграми та таблиці з процентним розподілом, що дозволяє користувачу аналізувати структуру своїх фінансових потоків та виявляти основні статті витрат (рис. 3.13).

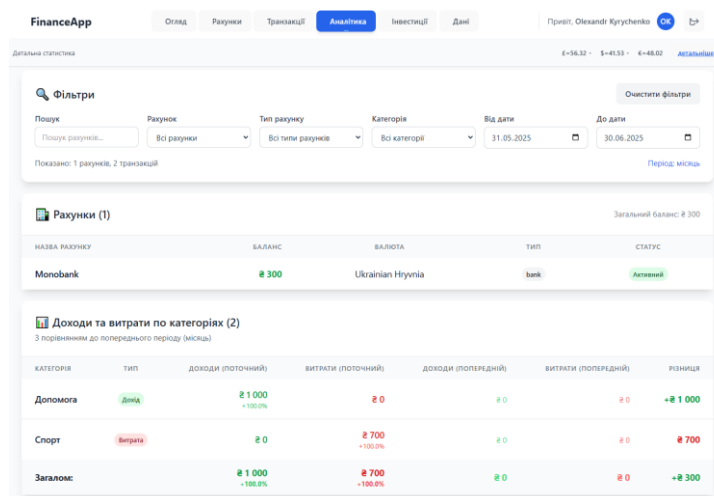


Рисунок 3.13 – Сторінка аналітики (нижня частина)

Розділ інвестиції інтегрований з Yahoo Finance API для відображення актуальних ринкових даних, котирувань акцій та фінансових новин.

Верхня частина інвестиційної сторінки відображає список активів з поточними цінами, зміною за день у відсотках та абсолютних величинах (рис. 3.14). Кожен актив має кольорову індикацію росту або падіння ціни для швидкої візуальної оцінки ринкової ситуації.

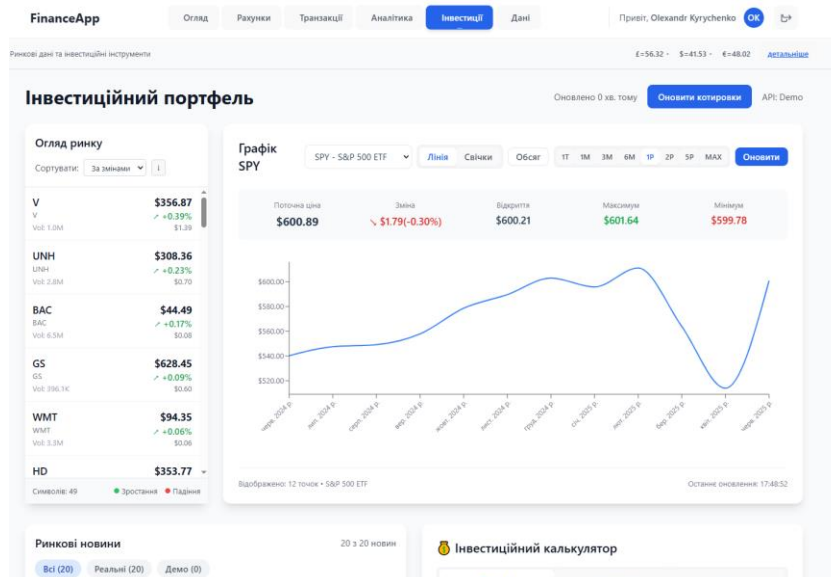


Рисунок 3.14 – Сторінка інвестицій (верхня частина)

Нижня частина містить стрічку актуальних фінансових новин з автоматичним оновленням для інформування користувача про важливі ринкові події, та інвестиційний калькулятор (рис. 3.15).

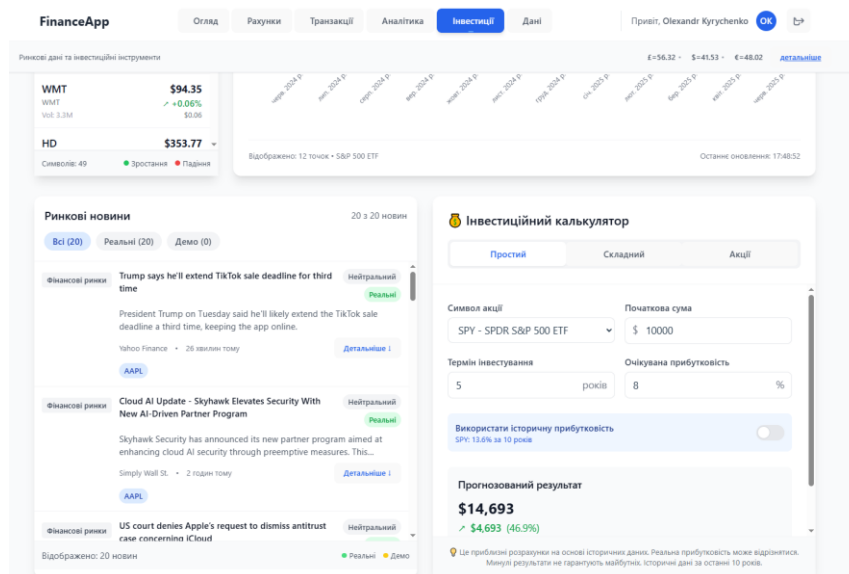


Рисунок 3.15 – Сторінка інвестицій (нижня частина)

Розділ дані забезпечує функціональність імпорту банківських виписок та експорту фінансових даних у різних форматах для інтеграції з зовнішніми системами (рис. 3.16).

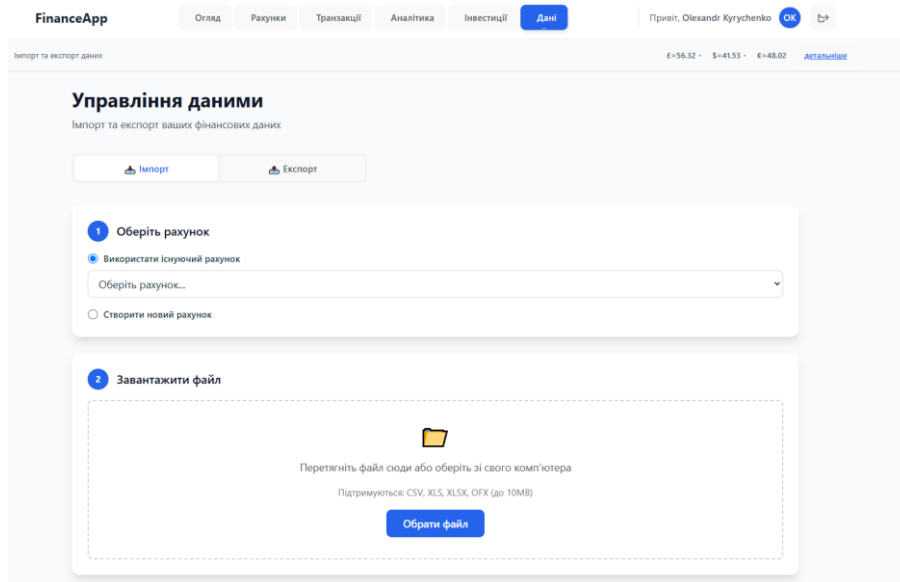


Рисунок 3.16 – Сторінка роботи з даними

Основна сторінка модуля містить дві основні функції: імпорт файлів з банківськими виписками у форматах CSV, Excel та OFX, а також експорт існуючих даних користувача у зручних для аналізу форматах.

Інтерфейс аналізу файлу дозволяє користувачу вручну налаштувати відповідність колонок файлу полям системи, попередньо переглянути результат імпорту та виправити можливі помилки перед остаточним збереженням транзакцій (рис. 3.17).

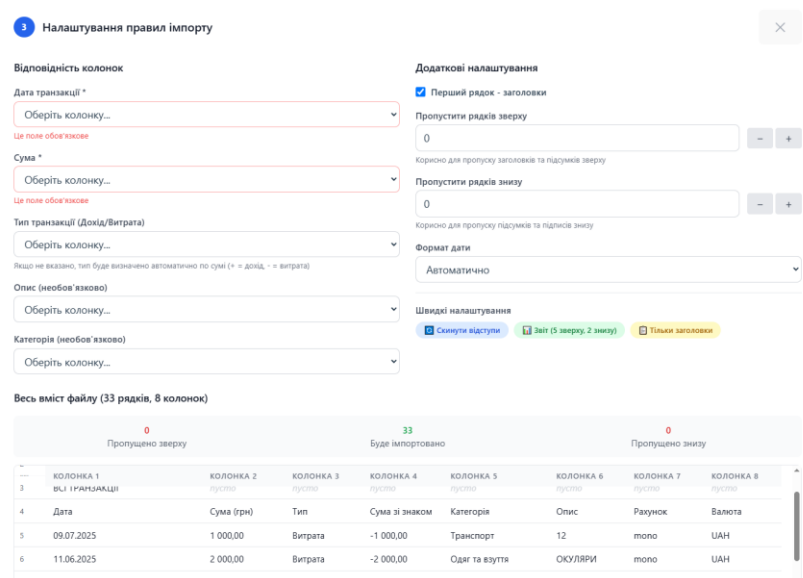


Рисунок 3.17 – Сторінка аналізу файлу після його завантаження

Експорт даних надає гнучкі можливості для вибору формату файлу, часового періоду та фільтрації за рахунками або категоріями для створення персоналізованих звітів (рис. 3.18).

Рисунок 3.18 – Сторінка експорту даних

Форма експорту включає опції вибору формату (CSV, Excel, PDF, OFX), налаштування фільтрів за датами, рахунками та категоріями, а також можливість включення додаткової статистичної інформації у згенерований звіт для комплексного аналізу фінансових даних.

3.3 Тестування роботи застосунку

Тестування системи керування особистими фінансами проводилося з використанням комплексного підходу, що включає автоматизовані тести API, функціональні тести та тести безпеки для забезпечення надійності та коректності роботи всіх компонентів системи (рис. 3.20).

Основне тестування системи здійснювалося через запуск автоматизованих тестів командою «php artisan test tests/Feature/ApiRoutesTest.php». Тестовий набір включає 54 тести, які перевіряють функціональність всіх ключових компонентів системи.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

```

✓ accounts destroy 0.04s
✓ accounts stats 0.03s
✓ recalculate all balances 0.03s
✓ transactions index 0.04s
✓ transactions store 0.03s
✓ transactions stats 0.03s
✓ transactions monthly data 0.03s
✓ categories index 0.03s
✓ categories store 0.03s
✓ categories global 0.03s
✓ budgets index 0.04s
✓ budgets store 0.03s
✓ budgets current 0.03s
✓ tags index 0.03s
✓ tags store 0.03s
✓ tags popular 0.03s
✓ dashboard stats 0.03s
✓ monthly report 0.03s
✓ yearly report 0.03s
✓ data export csv 0.04s
✓ data export excel 0.14s
✓ data import with rules 0.04s
✓ account validation fails without required fields 0.03s
✓ transaction validation fails with invalid amount 0.03s
✓ budget validation fails with invalid dates 0.03s
✓ user cannot access other users accounts 0.03s
✓ user cannot modify other users transactions 0.03s
✓ user cannot delete other users categories 0.03s
✓ api handles invalid json 0.03s
✓ api handles large file uploads 0.05s
✓ api handles unsupported file types 0.03s
✓ api pagination works correctly 0.06s
✓ api filtering works correctly 0.05s
✓ complete transaction workflow 0.04s
✓ complete budget workflow 0.06s

Tests: 54 passed (323 assertions)
Duration: 14.88s

```

PS E:\diploma_project\server>

Рисунок 3.20 – Проходження тестів API

Тестування модуля валют та курсів обміну включає перевірку отримання списку доступних валют, перевірка отримання валют з актуальними курсами, тестування оновлення валютних курсів та перевірку алгоритму конвертації між валютами. Усі тести цього модуля успішно пройдені з часом виконання від 0.11 до 0.70 секунди.

Автентифікація та авторизація тестуються через перевірку процесу реєстрації нових користувачів, перевірку успішного входу з перевіреними даними, тестування захисту від неправильних облікових даних, перевірку процесу виходу з системи та перевірку отримання даних авторизованого користувача. Тести автентифікації виконуються швидко з часом від 0.06 до 0.29 секунди.

Інвестиційний модуль проходить тестування отримання котирувань акцій через Yahoo Finance API, завантаження фінансових новин, перевірку статусу підключення до зовнішніх API, перевірку стабільності з'єднання з ринковими сервісами та тестування виконання Python скриптів для отримання ринкових даних. Ці тести займають найбільше часу від 2.71 до 4.28 секунди через необхідність мережевих запитів до зовнішніх сервісів.

Управління рахунками тестується через перевірку отримання списку рахунків користувача, перевірку створення нових рахунків, тестування отримання деталей конкретного рахунку, перевірку редагування параметрів рахунків, перевірку безпечного видалення рахунків, тестування розрахунку статистики по рахунках та перевірку алгоритму перерахунку балансів. Усі тести модуля рахунків виконуються швидко за 0.03 секунди кожен.

Функціональність транзакцій проходить перевірку отримання списку транзакцій з пагінацією, перевірку створення нових фінансових операцій, тестування розрахунку статистики транзакцій та перевірку отримання місячних звітів. Тести транзакцій також виконуються швидко з часом від 0.03 до 0.05 секунди.

Категорії та бюджети тестуються через перевірку отримання категорій користувача, перевірку створення нових категорій, тестування глобальних категорій системи, перевірку списку бюджетів користувача, перевірку створення нових бюджетних планів та тестування поточного стану бюджетів. Усі тести виконуються за 0.03-0.04 секунди.

Додаткові модулі включають тестування тегів через перевірку отримання тегів користувача, перевірку створення нових тегів та тестування популярних тегів системи. Звітність тестується через перевірку статистики головної панелі, перевірку звітів та тестування річних звітів. Імпорт та експорт проходять перевірку експорту даних у CSV формат, перевірка експорту в Excel формат та тестування імпорту з правилами перевірки.

Перевірка вхідних даних включає перевірку обов'язкових полів рахунків, перевірку сум транзакцій та тестування дат бюджетів. Усі тести підтверджують правильність обробки некоректних даних з поверненням відповідних помилок.

Безпека доступу тестується через перевірку ізоляції даних користувачів, перевірку захисту транзакцій, тестування захисту категорій та перевірку автентифікації для захищених маршрутів. Система успішно блокує доступ до чужих даних та вимагає автентифікацію для всіх захищених операцій.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Обробка помилок проходить через тестування некоректного JSON, великих файлів та непідтримуваних форматів. Система коректно оброблення всіх типів помилок з поверненням інформативних повідомлень.

Пагінація та фільтрація тестуються на коректність роботи з великими обсягами даних та точність застосування фільтрів за різними критеріями. Комплексні тести перевіряють повний цикл роботи з транзакціями та бюджетами від створення до видалення.

Усі 54 тести успішно пройдені з загальним часом виконання 16.80 секунд, що підтверджує стабільність та надійність системи. Покриття тестами включає всі критично важливі аспекти функціональності. Найдовші тести пов'язані з зовнішніми API запитам до Yahoo Finance, що є нормальним для мережових операцій. Основна частина тестів виконується за 0.03-0.14 секунди, що свідчить про оптимізовану продуктивність системи.

Тестування підтвердило коректність роботи всіх основних модулів включаючи автентифікацію користувачів, управління фінансовими рахунками, обробку транзакцій, валютні конвертації, інтеграцію з зовнішніми API, імпорт та експорт даних, а також систему безпеки. Результати автоматизованого тестування гарантують надійність системи керування особистими фінансами та її готовність для використання в продуктивному середовищі з підтримкою всіх заявлених функціональних вимог.

Висновки до розділу 3

У третьому розділі детально розглянуто інтерфейс та порядок роботи з розробленою системою керування особистими фінансами, а також проведено комплексне тестування функціональності.

Розроблено повний цикл встановлення та налаштування системи, що включає розгортання серверної частини на Laravel, клієнтської частини на React з Vite та налаштування бази даних PostgreSQL. Створена діаграма

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

розгортання демонструє чітке розділення архітектурних компонентів та їх взаємодію через REST API.

Проаналізовано структуру користувацького інтерфейсу, який побудований за принципом односторінкового застосунку з адаптивним дизайном. Інтерфейс включає шість основних модулів: огляд фінансового стану, управління рахунками, роботу з транзакціями, аналітичну звітність, інвестиційний портфель та модуль імпорту/експорту даних.

Реалізовано систему автентифікації на базі Laravel Sanctum. Функціональність управління рахунками забезпечує підтримку різних типів рахунків. Система транзакцій включає розширені можливості фільтрації та категоризації операцій.

Аналітичний модуль надає візуалізацію фінансових даних через графіки Chart.js. Інвестиційний модуль інтегрований з Yahoo Finance API через Python скрипти для отримання ринкових даних. Модуль роботи з даними реалізує імпорт банківських виписок у форматах CSV, Excel та OFX.

Проведено комплексне тестування системи з використанням 54 автоматизованих тестів, які покривають всі ключові компоненти. Усі тести успішно пройдені з загальним часом виконання 16.80 секунд, що підтверджує стабільність системи. Час відповіді основних операцій складає 0.03-0.14 секунди, що забезпечує високу продуктивність.

Розроблена система повністю відповідає функціональним вимогам та забезпечує комплексне рішення для управління особистими фінансами з сучасним інтерфейсом, надійною архітектурою та високим рівнем безпеки.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

ВИСНОВКИ

У процесі виконання випускної кваліфікаційної роботи бакалавра був успішно розроблений та реалізований органайзер з керування витратами, який забезпечує комплексне рішення для обліку доходів та витрат, планування бюджету та аналізу фінансового стану користувачів.

Проведений аналіз предметної області показав актуальність розробки сучасних інструментів для управління особистими фінансами у контексті зростаючої потреби населення в ефективному плануванні бюджету та контролі витрат. Дослідження існуючих рішень виявило недоліки наявних систем та визначило основні функціональні вимоги до розроблюваної системи.

Архітектурне проектування системи здійснено з використанням сучасних підходів та технологій. Розроблена діаграма класів включає одинадцять основних сутностей предметної області з чітко визначеними зв'язками. Структура бази даних спроектована за принципами нормалізації та забезпечує надійне зберігання фінансових даних з підтримкою інших валют та складних аналітичних запитів.

Реалізовані алгоритми обробки фінансових операцій гарантують точність розрахунків балансів рахунків, коректність валютних конвертацій, автоматизацію бюджетного контролю та обробку регулярних платежів. Особлива увага приділена алгоритмам автоматичної категоризації транзакцій та інтелектуального імпорту банківських виписок.

Програмна реалізація виконана з використанням сучасного технологічного стеку включаючи PHP Laravel 12 для серверної частини та React з Vite для клієнтського застосунку. Архітектура побудована за принципом розділення клієнтську частину та сервер з взаємодією через REST API, що забезпечує масштабованість та можливість подальшого розширення функціональності.

Користувацький інтерфейс розроблений з урахуванням принципів зручності використання та адаптивності для різних типів пристроїв. Система включає модулі для управління рахунками, транзакціями, бюджетами,

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

аналітики, інвестиційного портфеля та роботи з даними. Інтеграція з зовнішніми API забезпечує отримання актуальних ринкових даних та валютних курсів.

Комплексне тестування системи включало 54 автоматизованих тести, які покривають всі основні функції та сценарії використання. Результати тестування підтвердили коректність роботи алгоритмів, надійність системи безпеки, правильність перевірки даних та стабільність роботи під навантаженням.

Система забезпечує повний життєвий цикл управління особистими фінансами від реєстрації користувача до створення детальних аналітичних звітів. Функціональність імпорту та експорту даних дозволяє інтегрувати систему з існуючими банківськими сервісами та зовнішніми аналітичними інструментами.

Розроблена система відповідає всім поставленим цілям та завданням, забезпечуючи користувачам зручний та надійний інструмент для управління особистими фінансами. Модульна архітектура та використання сучасних технологій створюють основу для подальшого розвитку та додавання нових функціональних можливостей відповідно до потреб користувачів.

Практична значущість роботи полягає у створенні готового до використання програмного продукту, який може бути впроваджений як самостійний сервіс або інтегрований у існуючі фінансові платформи. Результати дослідження можуть бути використані для подальших практичних робіт у сфері розробки фінансових застосунків та систем управління особистими фінансами.

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кириченко О.С., Локтікова Т.М. Органайзер з керування витратами. Тези доповідей Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених, присвяченої Дню науки, 12-17 травня 2025 року. Житомир: «Житомирська політехніка», 2025. С. 61-62
2. API для розробникві Монобанк [Електронний ресурс] – <https://api.monobank.ua/docs/index.html>
3. API для розробникві НБУ [Електронний ресурс] – <https://bank.gov.ua/ua/open-data/api-dev>
4. API для розробникві Приват [Електронний ресурс] – <https://api.privatbank.ua/>
5. Axios [Електронний ресурс] – Режим доступу до ресурсу: <https://axios-http.com/docs/intro>
6. Bluecoins [Електронний ресурс] – <https://www.bluecoinsapp.com/>
7. Chart.js [Електронний ресурс] – Режим доступу до ресурсу: <https://www.chartjs.org/docs/latest>
8. Composer [Електронний ресурс] – Режим доступу до ресурсу: <https://getcomposer.org/doc>
9. Context API [Електронний ресурс] – <https://legacy.reactjs.org/docs/context.html>
10. CSRF [Електронний ресурс] – <https://developer.mozilla.org/en-US/docs/Web/Security/Attacks/CSRF>
11. Day.js [Електронний ресурс] – <https://day.js.org/en/>
12. ESLint [Електронний ресурс] – <https://eslint.org/docs/latest/>
13. Financial Data Formats OFX [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ofx.net/downloads.html>
14. Guide how to Build an Effective Personal Finance Application [Електронний ресурс] – <https://appinventiv.com/blog/how-to-build-personal-finance-app/>

15. Hot Module Replacement [Електронний ресурс] –
<https://webpack.js.org/concepts/hot-module-replacement/>
16. Laravel [Електронний ресурс] – Режим доступу до ресурсу:
<https://laravel.com/docs>
17. Laravel Eloquent ORM [Електронний ресурс] – Режим доступу до ресурсу:
<https://laravel.com/docs/12.x/eloquent>
18. Laravel Sanctum [Електронний ресурс] – Режим доступу до ресурсу:
<https://laravel.com/docs/12.x/sanctum>
19. Laravel Testing Guide [Електронний ресурс] – Режим доступу до ресурсу:
<https://laravel.com/docs/12.x/testing>
20. MDN Web Docs JavaScript [Електронний ресурс] – Режим доступу до
ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
21. Microservices Architecture Pattern [Електронний ресурс] – Режим доступу
до ресурсу: <https://microservices.io/patterns/microservices.html>
22. MoneyManager [Електронний ресурс] – <https://www.realbyteapps.com/>
23. Node.js [Електронний ресурс] – Режим доступу до ресурсу:
<https://nodejs.org/en/docs>
24. PHP [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.php.net/manual/en>
25. PhpSpreadsheet [Електронний ресурс] – Режим доступу до ресурсу:
<https://phpspreadsheet.readthedocs.io>
26. PIP [Електронний ресурс] – <https://pypi.org/project/pip/>
27. PostgreSQL [Електронний ресурс] – <https://www.postgresql.org/docs/>
28. Prettier [Електронний ресурс] – <https://prettier.io/docs/>
29. Python [Електронний ресурс] – <https://www.python.org/doc/>
30. React [Електронний ресурс] – Режим доступу до ресурсу:
<https://react.dev/learn>
31. React Router [Електронний ресурс] – Режим доступу до ресурсу:
<https://reactrouter.com/en/main>

- 32.REST API [Електронний ресурс] – Режим доступу до ресурсу:
<https://restfulapi.net>
- 33.RESTful Web Services [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.tutorialspoint.com/restful/index.htm>
- 34.Rollup [Електронний ресурс] – <https://rollupjs.org/introduction/>
- 35.Single Page Application [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.mozilla.org/en-US/docs/Glossary/SPA>
- 36.Spendee [Електронний ресурс] – <https://www.spendee.com/>
- 37.Tailwind CSS [Електронний ресурс] – Режим доступу до ресурсу:
<https://tailwindcss.com/docs>
- 38.Vite [Електронний ресурс] – Режим доступу до ресурсу:
<https://vitejs.dev/guide>
- 39.Wallet [Електронний ресурс] – <https://web.budgetbakers.com/>
- 40.Web Storage API [Електронний ресурс] – Режим доступу до ресурсу:
https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API
- 41.Yahoo Finance API [Електронний ресурс] – Режим доступу до ресурсу:
<https://finance.yahoo.com/lookup>
- 42.Yup [Електронний ресурс] – <https://formspre.io/blog/yup-validation/>

ДОДАТКИ

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

```

private function extractFromCsv(UploadedFile $file): array
{
    Log::info(«Processing CSV file», ['file' => $file-
>getClientOriginalName()]);

    $tempPath = $file->store('temp');
    $fullPath = storage_path('app/' . $tempPath);

    try {
        $transactions = [];
        $accounts = [];
        $categories = [];

        if (($handle = fopen($fullPath, 'r')) !== false) {
            $headers = fgetcsv($handle);

            if (!$headers) {
                throw new Exception('CSV файл порожній або пошкоджений');
            }

            // Нормалізуємо заголовки
            $headers = array_map(function($header) {
                return $this->normalizeHeader(trim($header));
            }, $headers);

            Log::info(«CSV headers found», ['headers' => $headers]);

            $rowCount = 0;
            while (($row = fgetcsv($handle)) !== false && $rowCount < 10000)
            { // Обмежуємо 10k рядків
                $rowCount++;

                if (count($row) !== count($headers)) {
                    continue; // Пропускаємо невірні рядки
                }

                $data = array_combine($headers, $row);

                // Парсимо транзакцію
                $transaction = $this->parseTransactionFromRow($data);
                if ($transaction) {
                    $transactions[] = $transaction;

                    // Збираємо унікальні рахунки та категорії
                    $this->collectAccountAndCategory($transaction, $accounts,
$categories);
                }
            }
        }
    }
}

```

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

        fclose($handle);
    }

    Log::info(«CSV processing completed», [
        'transactions' => count($transactions),
        'accounts' => count($accounts),
        'categories' => count($categories)
    ]);

    return [
        'transactions' => $transactions,
        'accounts' => array_values($accounts),
        'categories' => array_values($categories)
    ];

} finally {
    if (file_exists($fullPath)) {
        unlink($fullPath);
    }
}
}

```

Додаток Б

```

private function fetchRatesFromMultipleSources(): array
{
    // Спробуємо NBU
    $rates = $this->fetchFromNBU();
    if (!empty($rates)) {
        Log::info('Exchange rates fetched from NBU');
        $this->updateCurrencyRatesInDatabase($rates);
        return $rates;
    }

    // Якщо NBU не працює, спробуємо PrivatBank
    $rates = $this->fetchFromPrivatBank();
    if (!empty($rates)) {
        Log::info('Exchange rates fetched from PrivatBank');
        $this->updateCurrencyRatesInDatabase($rates);
        return $rates;
    }

    // Якщо i PrivatBank не працює, спробуємо Monobank
    $rates = $this->fetchFromMonobank();
    if (!empty($rates)) {
        Log::info('Exchange rates fetched from Monobank');
        $this->updateCurrencyRatesInDatabase($rates);
        return $rates;
    }
}

```

		Кириченко О.С		
		Локтікова Т.М.		
Змн.	Арк.	№ докум.	Підпис	Дата

```

        Log::error('Failed to fetch exchange rates from all sources');
        return $this->getFallbackRates();
    }

    private function fetchFromNBU(): array
    {
        try {
            Log::info('Attempting to fetch rates from NBU...');
            $response = $this->getHttpClient()->get($this->apiSources['nbu']);

            if (!$response->successful()) {
                Log::error('NBU API returned non-successful status: ' .
                    $response->status());
                return [];
            }

            $data = $response->json();
            $rates = ['UAH' => 1.0];

            Log::info('NBU API response received', ['count' => count($data)]);

            foreach ($data as $currency) {
                $code = $currency['cc'] ?? null;
                $rate = $currency['rate'] ?? null;

                if ($code && $rate && in_array($code, ['USD', 'EUR', 'GBP',
                    'PLN'])) {
                    $rates[$code] = 1 / (float)$rate;
                    Log::info("Added rate for {$code}: " . $rates[$code]);
                }
            }

            Log::info('NBU rates processed', ['rates' => $rates]);
            return $rates;
        } catch (\Exception $e) {
            Log::error('NBU API error: ' . $e->getMessage());
            return [];
        }
    }
}

```

Додаток В

```

public function getQuotes(Request $request): JsonResponse
{
    try {
        $symbols = $request->get('symbols');
        $forceRefresh = $request->boolean('force_refresh', false);

        Log::info('📊 Investment quotes request', [
            'symbols' => $symbols,
            'force_refresh' => $forceRefresh,

```

		Кириченко О.С			ІІЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

        'ip' => $request->ip()
    });

    // Валідація символів
    if (empty($symbols)) {
        return response()->json([
            'success' => false,
            'error' => 'No symbols provided'
        ], 400);
    }

    // Конвертуємо в масив якщо це строка
    if (is_string($symbols)) {
        $symbols = array_map('trim', explode(',', $symbols));
    }

    // Фільтруємо порожні символи
    $symbols = array_filter($symbols, function($symbol) {
        return !empty(trim($symbol));
    });

    if (empty($symbols)) {
        return response()->json([
            'success' => false,
            'error' => 'No valid symbols provided'
        ], 400);
    }

    // Обмежуємо кількість символів
    if (count($symbols) > 100) {
        $symbols = array_slice($symbols, 0, 100);
        Log::warning('⚠ Limited symbols to 100', ['original_count' =>
count($symbols)]);
    }

    Log::info('🔍 Processing symbols', ['symbols' => $symbols, 'count'
=> count($symbols)]);

    // Отримуємо котирунки
    $quotes = $this->yahooFinanceService->getMultipleQuotes($symbols,
$forceRefresh);

    Log::info('✅ Quotes retrieved successfully', [
        'requested' => count($symbols),
        'received' => count($quotes),
        'symbols' => array_keys($quotes)
    ]);

    return response()->json([
        'success' => true,

```

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

        'data' => $quotes,
        'summary' => [
            'requested' => count($symbols),
            'received' => count($quotes),
            'failed' => count($symbols) - count($quotes),
            'symbols' => array_keys($quotes)
        ],
        'timestamp' => now()->toISOString()
    ]);

} catch (\Exception $e) {
    Log::error('✖ Error in getQuotes', [
        'error' => $e->getMessage(),
        'file' => $e->getFile(),
        'line' => $e->getLine(),
        'symbols' => $request->get('symbols')
    ]);

    return response()->json([
        'success' => false,
        'error' => 'Failed to fetch quotes: ' . $e->getMessage(),
        'timestamp' => now()->toISOString()
    ], 500);
}
}

```

Додаток Г

```

private function executePythonScript($action, $symbols = [])
{
    try {
        $pythonCommand = $this->findWorkingPython();

        if (!$pythonCommand) {
            Log::warning("⚠ No working Python found, using mock data");
            return $this->getMockDataForAction($action, $symbols);
        }

        $scriptPath = $this->scriptPath;
        $arguments = implode(' ', $symbols);

        // Створюємо команду як масив
        $commandArray = [$pythonCommand, $scriptPath, $action];
        if (!empty($arguments)) {
            $commandArray = array_merge($commandArray, explode(' ',
str_replace(' ', '', $arguments)));
        }

        Log::info("🚀 === EXECUTING PYTHON COMMAND ===", [
            'command' => implode(' ', $commandArray),
            'python' => $pythonCommand,

```

		Кириченко О.С			ІІЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

        'script_exists' => file_exists($scriptPath)
    ]);

    // Налаштовуємо змінні середовища для правильного UTF-8
    $env = array_merge($_ENV, [
        'PYTHONIOENCODING' => 'utf-8',
        'PYTHONUNBUFFERED' => '1',
        'LC_ALL' => 'C.UTF-8',
        'LANG' => 'C.UTF-8',
        'CHCP' => '65001' // UTF-8 для Windows CMD
    ]);

    // Для Windows додатково встановлюємо кодування
    if (PHP_OS_FAMILY === 'Windows') {
        // Виконуємо команду через cmd з UTF-8
        $cmdCommand = 'chcp 65001 >nul 2>&1 && ' . implode(' ',
array_map('escapeshellarg', $commandArray));
        $process = Process::fromShellCommandline($cmdCommand,
base_path(), $env);
    } else {
        $process = new Process($commandArray, base_path(), $env);
    }

    $process->setTimeout(30);
    $process->setIdleTimeout(15);

    Log::info("▶ Starting Python process...");
    $process->run();

    $exitCode = $process->getExitCode();
    $output = $process->getOutput();
    $errorOutput = $process->getErrorOutput();

    Log::info("▣ === PROCESS RESULT ===", [
        'exit_code' => $exitCode,
        'stdout_length' => strlen($output),
        'stderr_length' => strlen($errorOutput),
        'has_json' => strpos($output, '{') !== false,
        'output_preview' => substr($output, 0, 200)
    ]);

    if ($exitCode !== 0) {
        Log::error("✗ Python process failed", [
            'exit_code' => $exitCode,
            'stderr' => $errorOutput,
            'stdout' => $output
        ]);

        return $this->getMockDataForAction($action, $symbols);
    }
}

```

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

// Обробляємо вивід
$cleanOutput = $this->extractJsonFromOutput($output);

if (empty($cleanOutput)) {
    Log::warning("⚠ No JSON found in output, using mock data");
    return $this->getMockDataForAction($action, $symbols);
}

// Декодуємо JSON
$decodedData = json_decode($cleanOutput, true, 512,
JSON_INVALID_UTF8_SUBSTITUTE);

if (json_last_error() !== JSON_ERROR_NONE) {
    Log::error("✖ JSON decode error", [
        'error' => json_last_error_msg(),
        'clean_output_preview' => substr($cleanOutput, 0, 500)
    ]);

    return $this->getMockDataForAction($action, $symbols);
}

Log::info("✅ === PYTHON SUCCESS ===", [
    'result_success' => $decodedData['success'] ?? false,
    'data_count' => is_array($decodedData['data']) ?? null) ?
count($decodedData['data']) : 0
]);

return $decodedData;

} catch (\Exception $e) {
    Log::error("✖ Exception in Python execution", [
        'error' => $e->getMessage(),
        'trace' => $e->getTraceAsString()
    ]);

    return $this->getMockDataForAction($action, $symbols);
}
}

```

Додаток Г

```

def get_historical_data(symbol, period):
    """Отримує історичні дані символу"""
    try:
        yahoo_period, interval = get_period_params(period)

        ticker = yf.Ticker(symbol.upper())
        hist = ticker.history(period=yahoo_period, interval=interval)

```

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

if hist.empty:
    raise ValueError(f"No historical data available for {symbol}")

# Конвертуємо в наш формат
chart_data = []
for date, row in hist.iterrows():
    chart_data.append({
        'date': date.strftime('%Y-%m-%d'),
        'price': round(float(row['Close']), 2),
        'open': round(float(row['Open']), 2),
        'high': round(float(row['High']), 2),
        'low': round(float(row['Low']), 2),
        'volume': int(row['Volume']) if not pd.isna(row['Volume']) else 0
    })

# Сортуюмо по даті
chart_data.sort(key=lambda x: x['date'])

# Отримуємо поточну котировку
quote_data = format_quote_data(ticker, symbol)

# Розраховуємо зміни за період
if len(chart_data) > 1:
    first_price = chart_data[0]['price']
    current_price = quote_data['price']
    period_change = current_price - first_price
    period_change_percent = (period_change / first_price * 100) if
first_price != 0 else 0

    quote_data['periodChange'] = round(period_change, 2)
    quote_data['periodChangePercent'] = round(period_change_percent, 2)

return {
    'success': True,
    'data': {
        'symbol': symbol.upper(),
        'name': quote_data['name'],
        'quote': quote_data,
        'chartData': chart_data,
        'period': period,
        'period_info': {
            'requested_period': period,
            'yahoo_period': yahoo_period,
            'interval': interval,
            'start_date': chart_data[0]['date'] if chart_data else None,
            'end_date': chart_data[-1]['date'] if chart_data else None,
            'points': len(chart_data)
        },
    },
    'isDemo': False,

```

		Кириченко О.С			ІІЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

        'lastUpdated': datetime.now().isoformat(),
        'source': 'Yahoo Finance'
    },
    'timestamp': datetime.now().isoformat()
}

except Exception as e:
    return {
        'success': False,
        'error': str(e),
        'symbol': symbol.upper(),
        'period': period,
        'timestamp': datetime.now().isoformat()
    }

```

Додаток Д

```

public function importWithRules(Request $request)
{
    try {
        Log::info('=== IMPORT WITH RULES START ===');
        Log::info('Request data:', [
            'has_file' => $request->hasFile('file'),
            'account_id' => $request->input('account_id'),
            'new_account_name' => $request->input('new_account_name'),
            'create_new_account' => $request->input('create_new_account'),
            'import_rules' => $request->input('import_rules'),
        ]);

        // Базова валідація
        if (!$request->hasFile('file')) {
            return response()->json([
                'success' => false,
                'error' => 'Файл не знайдено'
            ], 422);
        }

        $file = $request->file('file');
        $importRulesJson = $request->input('import_rules');
        $createNewAccount = filter_var($request->input('create_new_account'),
FILTER_VALIDATE_BOOLEAN);
        $accountId = $request->input('account_id');
        $newAccountName = $request->input('new_account_name');

        // Парсимо правила імпорту
        $importRules = json_decode($importRulesJson, true);
        if (!$importRules) {
            return response()->json([

```

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

        'success' => false,
        'error' => 'Невірний формат правил імпорту'
    ], 422);
}

Log::info('Parsed import rules:', $importRules);

// ВИПРАВЛЕНА ПЕРЕВІРКА ОБОВ'ЯЗКОВИХ ПОЛІВ
$dateColumn = $importRules['dateColumn'] ?? '';
$amountColumn = $importRules['amountColumn'] ?? '';

// Перевіряємо що колонки вказані і є числами або числовими рядками
if ($dateColumn === '' || $dateColumn === null ||
!is_numeric($dateColumn)) {
    return response()->json([
        'success' => false,
        'error' => 'Колонка дати є обов'язковою. Оберіть колонку для
дати.'
    ], 422);
}

if ($amountColumn === '' || $amountColumn === null ||
!is_numeric($amountColumn)) {
    return response()->json([
        'success' => false,
        'error' => 'Колонка суми є обов'язковою. Оберіть колонку для
суми.'
    ], 422);
}

// Додаткова перевірка що це різні колонки
if ($dateColumn === $amountColumn) {
    return response()->json([
        'success' => false,
        'error' => 'Колонки дати та суми мають бути різними'
    ], 422);
}

// Перевіряємо рахунок
$account = null;
if ($createNewAccount) {
    if (empty($newAccountName)) {
        return response()->json([
            'success' => false,
            'error' => 'Назва нового рахунку не може бути порожньою'
        ], 422);
    }

    // Створюємо новий рахунок
    $account = auth()->user()->accounts()->create([

```

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

        'name' => $newAccountName,
        'type' => 'bank',
        'balance' => 0,
        'currency_id' => 1, // UAH
        'description' => 'Створено при імпорті',
        'is_active' => true
    ]);

    Log::info('Created new account:', ['id' => $account->id, 'name' =>
$account->name]);
    } else {
        if (empty($accountId)) {
            return response()->json([
                'success' => false,
                'error' => 'Оберіть існуючий рахунок або створіть новий'
            ], 422);
        }

        $account = auth()->user()->accounts()->find($accountId);
        if (!$account) {
            return response()->json([
                'success' => false,
                'error' => 'Рахунок не знайдено'
            ], 422);
        }

        Log::info('Using existing account:', ['id' => $account->id, 'name' =>
$account->name]);
    }

    // Аналізуємо файл
    $analysisResponse = $this->analyzeCsvFile($file);
    $analysisData = json_decode($analysisResponse->getContent(), true);

    if (!$analysisData['success']) {
        return response()->json([
            'success' => false,
            'error' => 'Помилка аналізу файлу: ' . $analysisData['error']
        ], 422);
    }

    $allRows = $analysisData['all_rows'];
    Log::info('File analysis complete', [
        'total_rows' => count($allRows),
        'rules' => $importRules
    ]);

    // Обробляємо дані згідно з правилами
    $result = $this->processImportWithRules($allRows, $importRules,
$account);

```

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

```

        Log::info('Import completed:', $result);

        return response()->json([
            'success' => true,
            'message' => 'Імпорт успішно завершено',
            'data' => $result,
            'account_name' => $account->name,
            'created_account' => $createNewAccount ? $account : null
        ]);

    } catch (\Exception $e) {
        Log::error('Import with rules error', [
            'error' => $e->getMessage(),
            'trace' => $e->getTraceAsString()
        ]);

        return response()->json([
            'success' => false,
            'error' => 'Помилка імпорту: ' . $e->getMessage()
        ], 500);
    }
}

```

		Кириченко О.С			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		78