Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3

Технології розроблення програмного забезпечення

Тема: «Особиста бухгалтерія»

Виконала:
Студентка групи ІА-34
Дригант А.С
Перевірив:
Мягкий Михайло Юрійович

Київ-2025

Тема: Основи проектування розгортання

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Тема роботи:

27. Особиста бухгалтерія (state, prototype, decorator, bridge, flyweight, SOA) Програма повинна бути наочним засобом для ведення особистих фінансів: витрат і прибутку; з можливістю встановлення періодичних витрат / прибутку (зарплата і орендна плата); введення сканованих чеків з відповідними статтями витрат; побудова статистики; експорт/імпорт в Excel, реляційні джерела даних; різні рахунки; ведення єдиного фонду на всі рахунки (всією сім'єю) – на особливі потреби (ремонт, автомобіль, відпустка); можливість введення вкладів / кредитів для контролю банківських рахунків (звірка нарахованих відсотків з необхідними і т.д.).

Теоритичні відомості

Діаграма розгортання (Deployment Diagram)

Показує фізичне розташування системи: на якому обладнанні чи середовищі виконання запускається ПЗ.

Основні елементи:

- Вузли (nodes): пристрій (device - комп’ютер, сервер) або середовище виконання (execution environment - ОС, вебсервер).
- Зв’язки між вузлами: лінії, які можуть мати назву (HTTP, IPC).
- Артефакти: файли, що розгортаються (exe, dll, jar, скрипти, таблиці БД, документи).

Бувають:

- Описові діаграми - без конкретного обладнання, для початкових етапів.
- Екземплярні діаграми - з реальними ПК, серверами, конкретними

артефактами.

Діаграма компонентів (Component Diagram)

Показує систему як набір модулів (компонентів).

Види:

- Логічні - система як набір автономних модулів.
- Фізичні - залежності між файлами (.exe, .dll) та модулями.
- Виконувані - виконувані файли, HTML-сторінки, БД тощо.

Призначення:

- візуалізація структури коду,
- специфікація виконуваного варіанта,
- багаторазове використання коду,
- опис концептуальної та фізичної БД.

Діаграма послідовностей (Sequence Diagram)

Відображає взаємодію об’єктів у часі (обмін повідомленнями).

Елементи:

- Актори (користувачі чи системи).
- Об’єкти/класи (прямокутники з «лініями життя»).
- Повідомлення (стрілки між об’єктами - виклик методів, повернення результату).
- Активності (прямокутники на лініях життя).
- Контрольні структури - умови (alt), цикли (loop).
- Використання: моделювання бізнес-процесів, проєктування архітектури, тестування.

Хід роботи

Діаграма розгортання

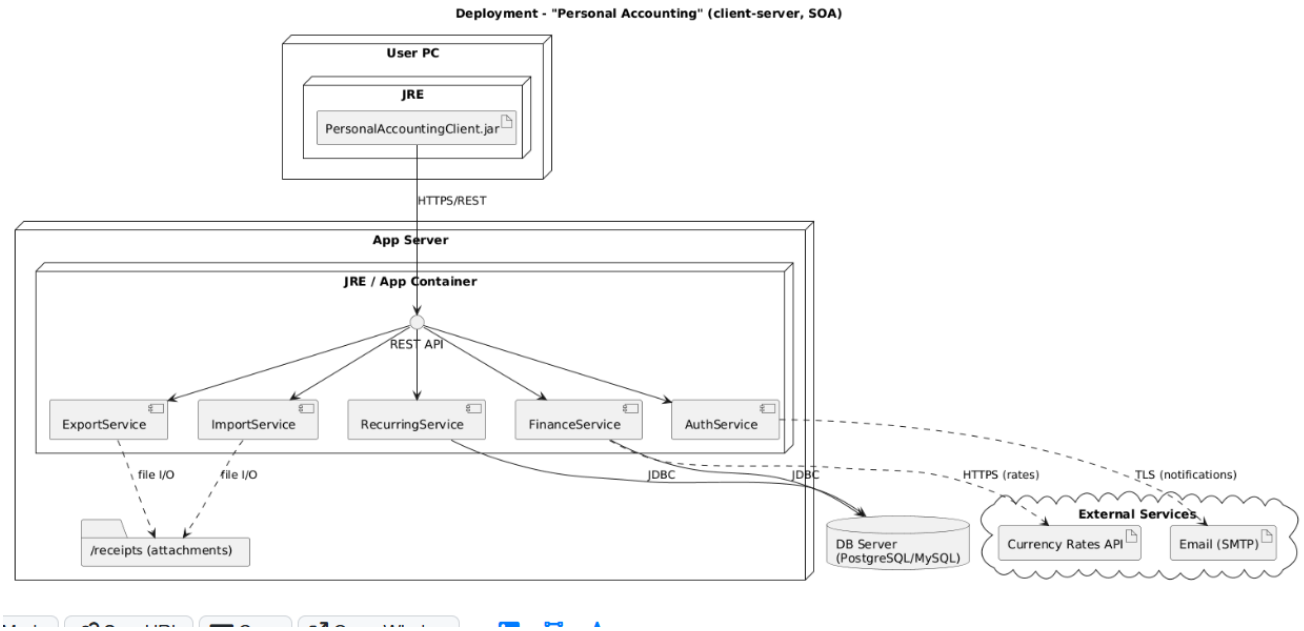


Рисунок 1 – Діаграма розгортання

Дана діаграма відображає архітектуру розгортання програмного комплексу “Personal Accounting”, побудованого за принципом клієнт-серверної взаємодії з використанням сервісно-орієнтованої архітектури (SOA). На комп’ютері користувача у середовищі JRE запускається клієнтський застосунок PersonalAccountingClient.jar, який через протокол HTTPS і REST API взаємодіє із серверною частиною. На стороні застосункового сервера у контейнері JRE розгорнуті окремі сервіси, що відповідають за різні функції системи. До них належать ExportService для експорту даних, ImportService для імпорту та обробки файлів і сканованих чеків, RecurringService для організації періодичних фінансових операцій, FinanceService як основний модуль управління витратами, доходами та рахунками, а також AuthService, який відповідає за автентифікацію та безпеку користувачів. Усі сервіси взаємодіють між собою через REST API та використовують JDBC для доступу до бази даних, що розгорнута на сервері з підтримкою PostgreSQL або MySQL. База даних зберігає інформацію про рахунки, транзакції, користувачів та інші важливі дані. Система також інтегрується із зовнішніми сервісами: Currency Rates API для отримання актуальних валютних

курсів через захищений канал HTTPS та Email-сервісом (SMTP) для надсилання повідомлень і сповіщень через TLS. Таким чином, уся система забезпечує зручне ведення особистих фінансів, підтримку періодичних операцій, роботу з зовнішніми даними та надійну комунікацію з користувачем через клієнтський застосунок.

Діаграма компонентів

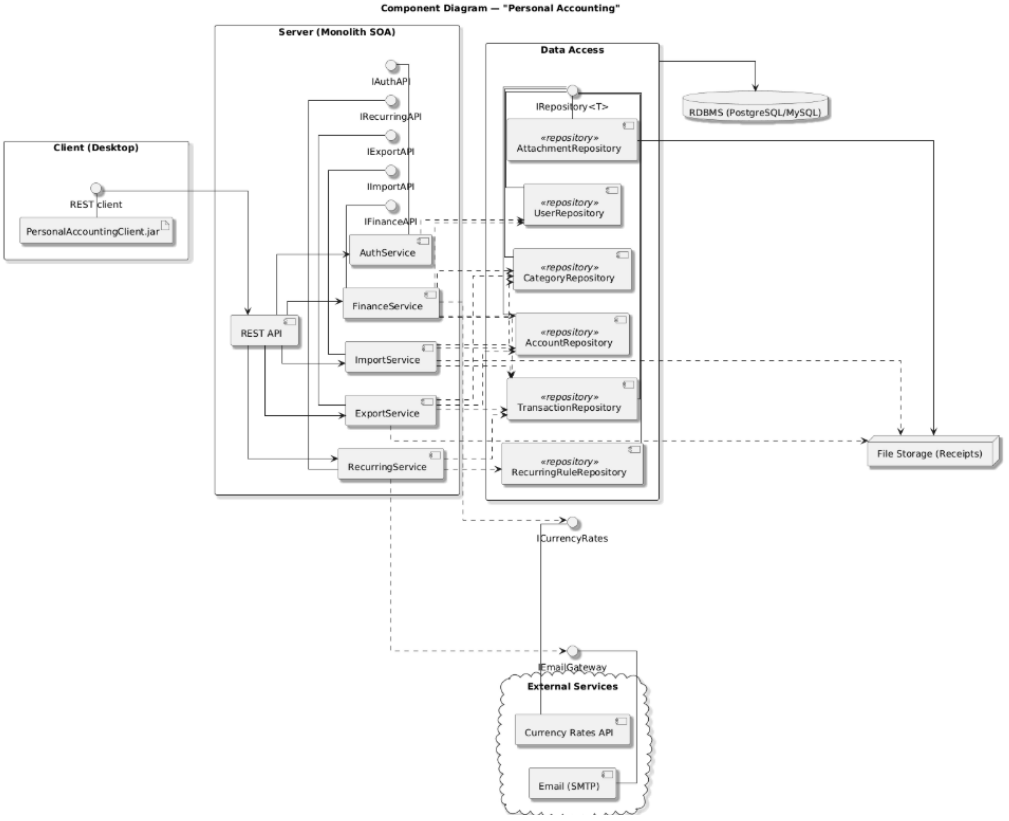


Рисунок 2 –Діаграма компонентів

Ця діаграма відображає компонентну структуру системи “Personal Accounting”, реалізованої у вигляді монолітного серверного застосунку з використанням сервісно-орієнтованих принципів (SOA).

Клієнтська частина системи представлена застосунком PersonalAccountingClient.jar, що запускається на робочому столі користувача та взаємодіє із сервером через REST API. Сервер складається з набору логічних сервісів, кожен із яких виконує окремі бізнес-функції. Серед них виділяються: AuthService для автентифікації користувачів, FinanceService для управління фінансовими операціями та рахунками, ImportService та ExportService для імпорту й експорту даних, а також

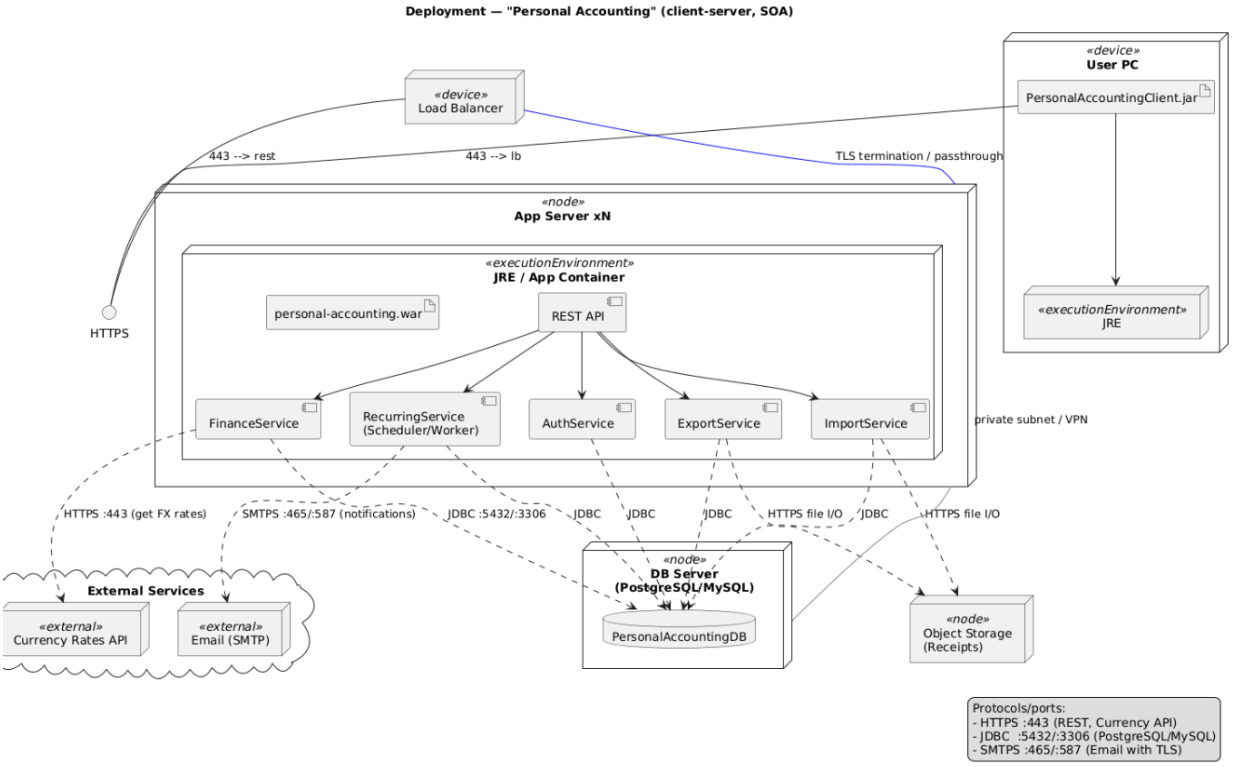
RecurringService для обробки періодичних платежів і доходів. Кожен сервіс використовує відповідні інтерфейси та взаємодіє з іншими через внутрішні виклики.

Зберігання даних реалізовано через шар доступу до даних (Data Access), який включає набір репозиторіїв. До них належать: UserRepository для управління користувачами, AccountRepository для даних рахунків, CategoryRepository для категорій витрат і доходів, TransactionRepository для транзакцій, RecurringRepository для періодичних операцій, а також AttachmentRepository для зберігання сканованих чеків та вкладень. Усі репозиторії взаємодіють з RDBMS (PostgreSQL), що виконує роль центрального сховища даних, тоді як файли чеків зберігаються окремо у файловому сховищі (File Storage).

Крім того, система підключається до зовнішніх сервісів через окремі інтерфейси. Для роботи з курсами валют використовується Currency Rates API, що дозволяє актуалізувати фінансові дані в системі. Для надсилання повідомлень і сповіщень застосовується Email-сервіс (SMTP), який забезпечує комунікацію з користувачами.

Таким чином, діаграма ілюструє структуровану архітектуру, де клієнт взаємодіє з сервером через REST API, сервер організований у вигляді модульних сервісів, що користуються спільним шаром доступу до даних, а інтеграція із зовнішніми сервісами розширює функціонал системи, забезпечуючи її повноцінну роботу як інструменту для особистого фінансового обліку.

Діаграма розгортання для проєктованої системи



Це діаграма розгортання для проєктованої системи “Personal Accounting”.

Вона відображає фізичне розташування компонентів і середовищ виконання у клієнт-серверній архітектурі. Клієнтська частина представлена застосунком PersonalAccountingClient.jar, що запускається на ПК користувача у середовищі JRE та взаємодіє з сервером через HTTPS. Для підвищення продуктивності і відмовостійкості використовується Load Balancer, який розподіляє навантаження між екземплярами застосункового сервера.

На стороні сервера у контейнері JRE розгорнуто веб-застосунок personal-accounting.war, що включає сервіси: FinanceService, RecurringService (з планувальником/воркером), AuthService, ExportService та ImportService. Вони взаємодіють між собою через REST API. Для збереження даних використовується DB Server (PostgreSQL/MySQL) із базою PersonalAccountingDB, а для чеків і вкладень — окремий вузол Object Storage.

Таким чином, діаграма показує, як логічні сервіси системи розгортаються на фізичних вузлах і взаємодіють між собою та з зовнішніми сервісами.

Сценарій 1: Запис доходу/витрати

Передумови:

- Користувач автентифікований у системі
- Існує хоча б один рахунок і одна категорія транзакцій

Постумови:

- У системі створено нову транзакцію.
- Баланс рахунку оновлено

Взаємодіючі сторони:

Користувач, Система.

Короткий опис:

Цей варіант використання описує процес додавання доходу або витрати користувачем у систему.

Основний потік подій:

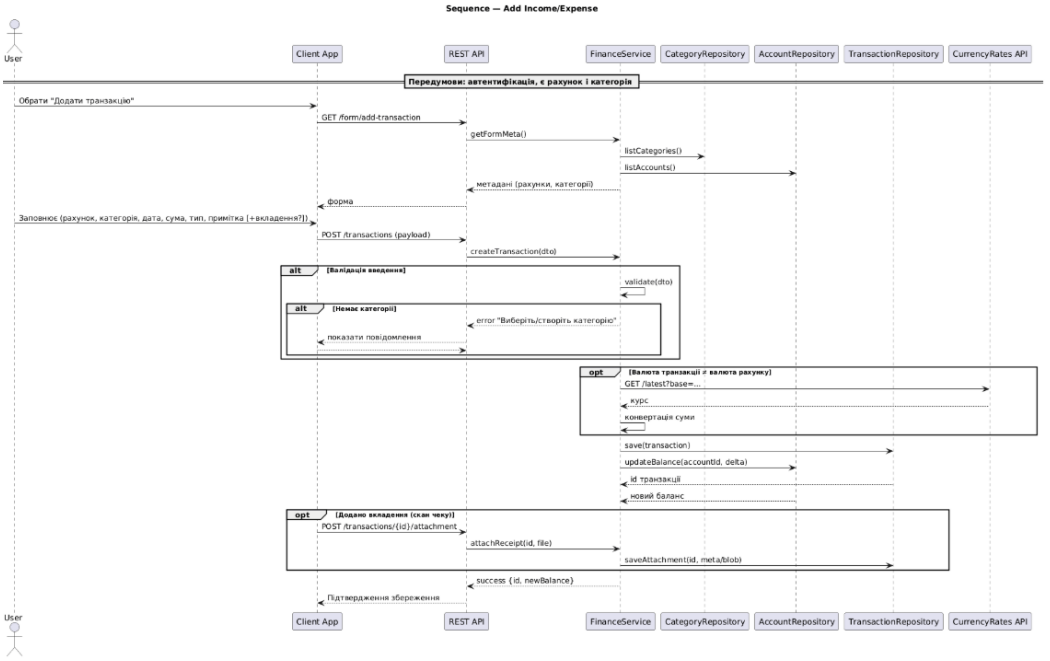
1. Користувач обирає функцію «Додати транзакцію»
2. Система пропонує ввести дані: рахунок, категорію, дату, суму, тип (дохід/витрата), примітку
3. Користувач вводить усі необхідні дані
4. Система створює нову транзакцію, оновлює баланс рахунку та повідомляє про успішне збереження

Винятки:

- Якщо користувач не вказав категорію то система просить вибрати або створити нову
- Якщо валюта рахунку не збігається з введеною транзакцією, то система виконує конвертацію або видає повідомлення про помилку

Примітки:

Можливе додавання вкладення (скан чеку).



Сценарій 2: Планування регулярних платежів

Передумови:

- Користувач увійшов у систему
- Існує хоча б один рахунок і категорія

Постумови:

- У системі збережено нове правило періодичної транзакції
- У зазначений час будуть автоматично створюватися відповідні транзакції

Взаємодіючі сторони:

Користувач, Система.

Короткий опис:

Цей варіант описує створення періодичного платежу (наприклад, щомісячної оплати оренди).

Основний потік подій:

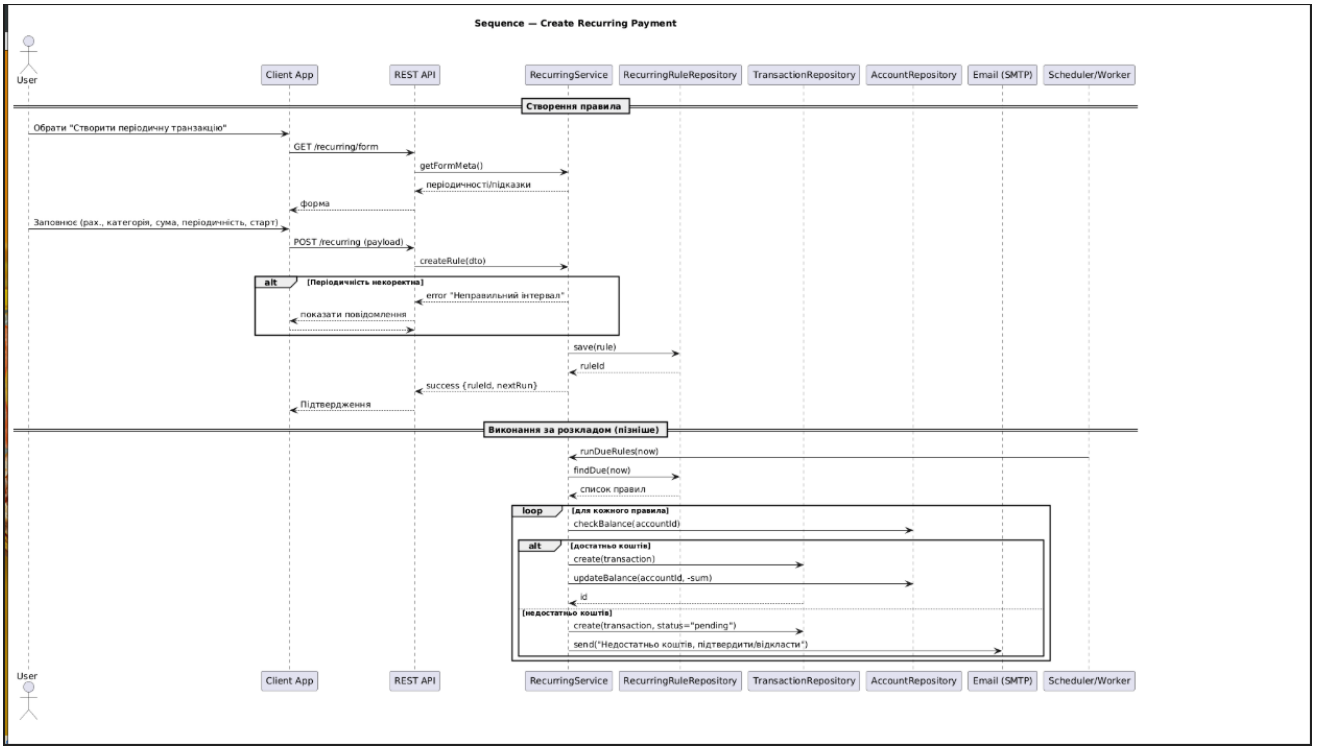
1. Користувач обирає функцію «Створити періодичну транзакцію»
2. Система пропонує ввести дані: рахунок, категорію, суму, періодичність, дату початку
3. Користувач вводить дані й підтверджує створення
4. Система зберігає правило і планує першу дату виконання

Винятки:

- Якщо користувач вводить некоректну періодичність то система видає повідомлення і просить повторити введення
- Якщо на рахунку немає достатньо коштів у момент виконання то система позначає транзакцію як «очікує підтвердження»

Примітки:

Для регулярних доходів (наприклад, зарплата) процес аналогічний.



Сценарій 3: Експорт даних у Excel

Передумови:

- Користувач увійшов у систему.
- У базі є хоча б одна транзакція.

Постумови:

- Згенеровано файл Excel із даними про транзакції.

Взаємодіючі сторони:

Користувач, Система.

Короткий опис:

Цей варіант описує процес експорту фінансових даних у формат Excel для подальшого аналізу.

Основний потік подій:

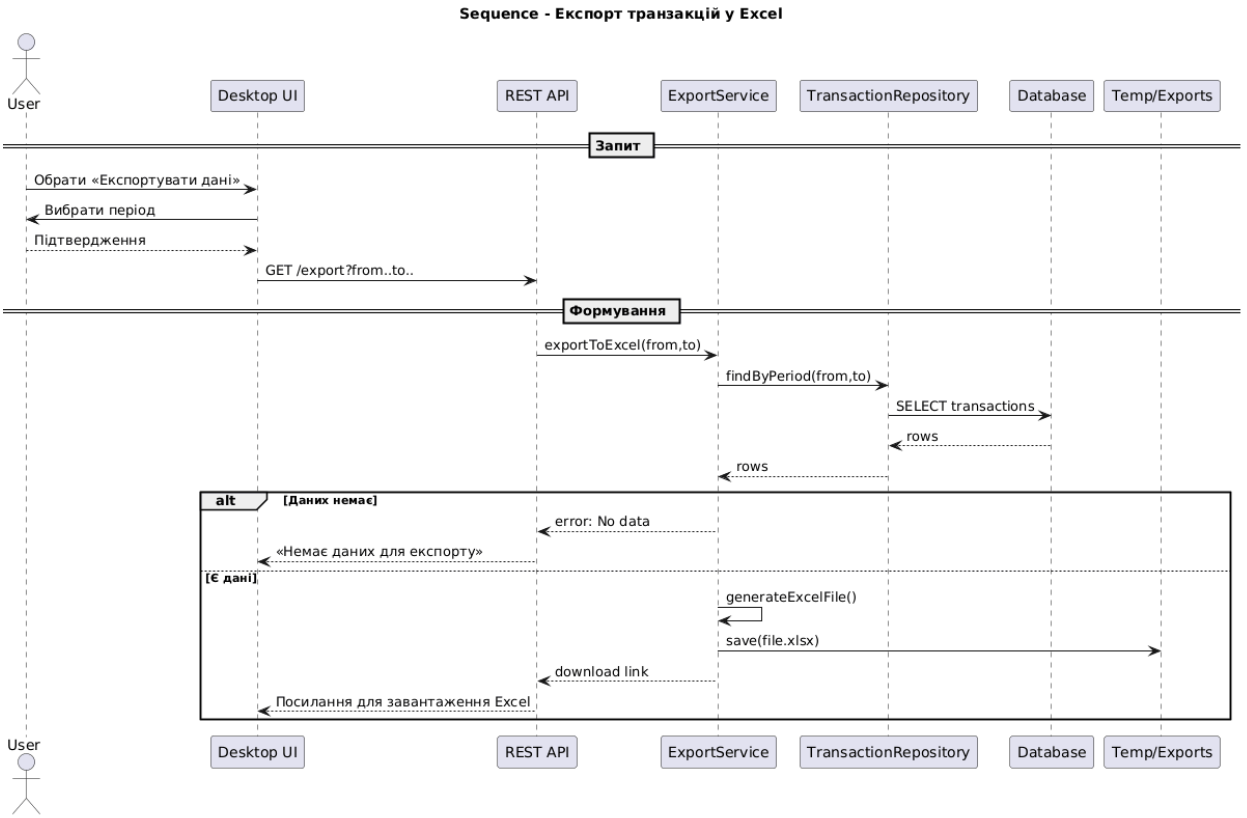
1. Користувач обирає функцію «Експортувати дані»
2. Система пропонує вибрати період (наприклад, за місяць)
3. Користувач підтверджує вибір
4. Система формує Excel-файл та надає можливість його завантажити

Винятки:

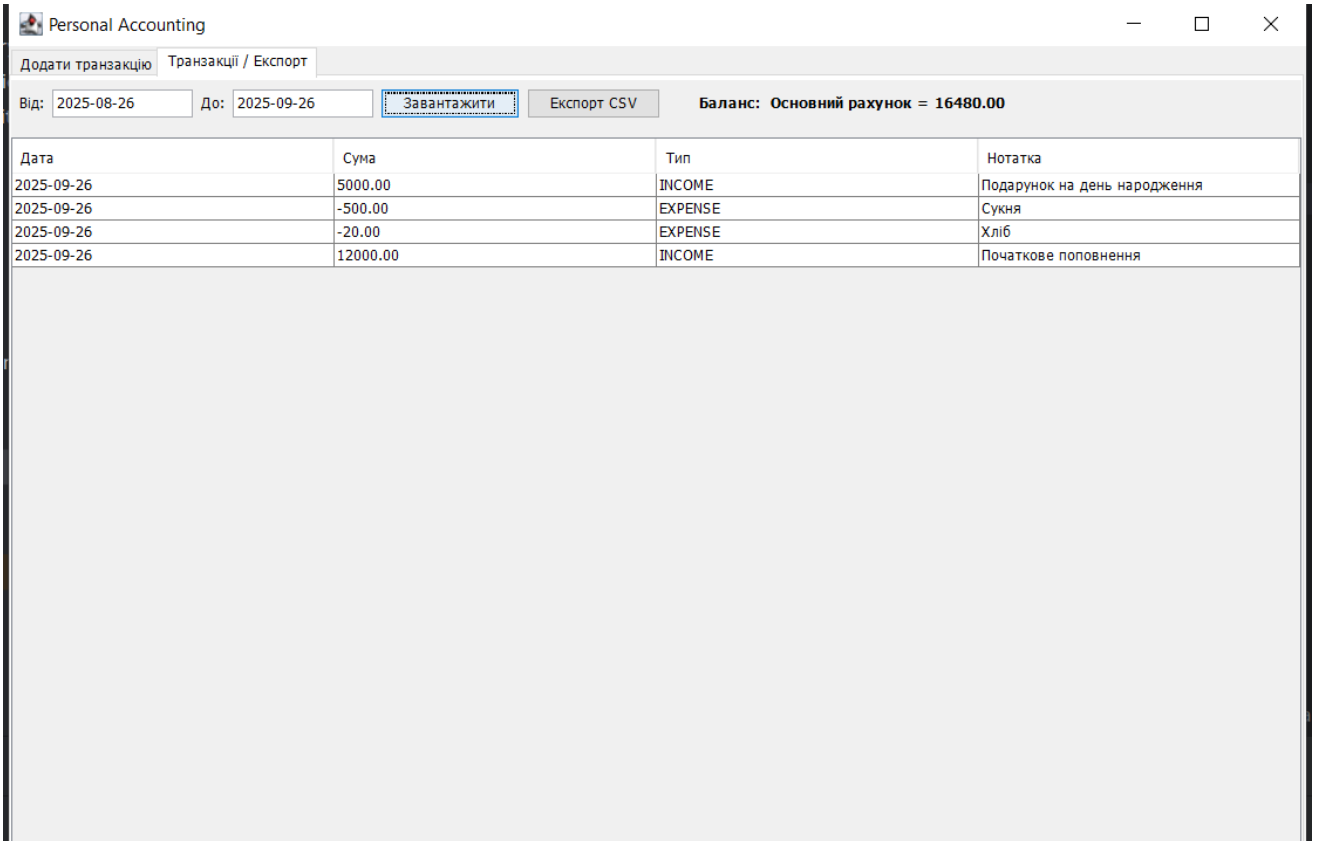
Якщо за вибраний період немає даних → система показує повідомлення «Немає даних для експорту»

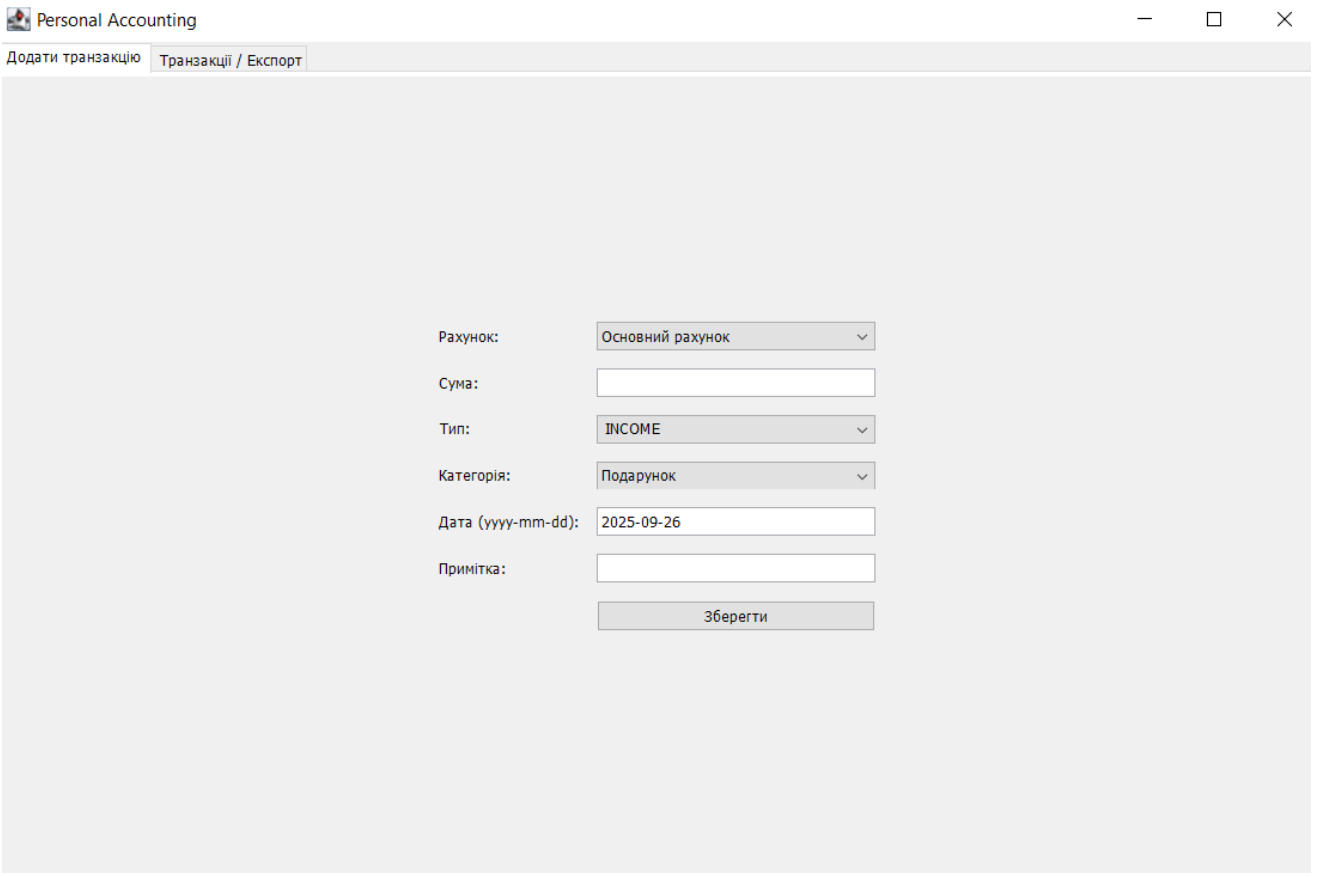
Примітки:

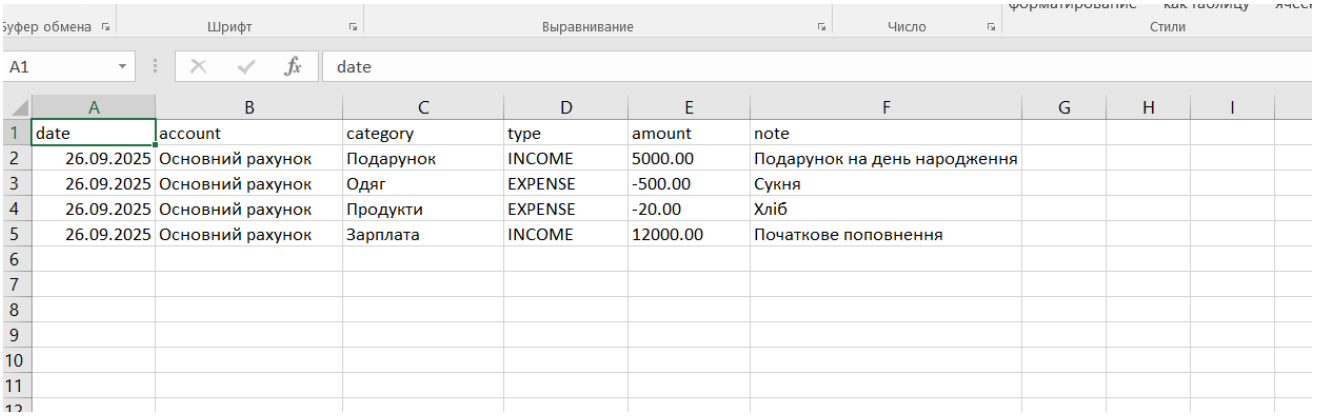
Експорт може виконуватися і у формат CSV


Sequence - Експорт транзакцій у Excel
User
Desktop UI
REST API
ExportService
TransactionRepository
Database
Temp/Exports
Запит
Обрати «Експортувати дані»
Вибрати період
Підтвердження
GET /export?from..to..
Формування
exportToExcel(from,to)
findByPeriod(from,to)
SELECT transactions
rows
rows
alt
[Даних немає]
error: No data
«Немає даних для експорту»
[Є дані]
generateExcelFile()
save(file.xlsx)
download link
Посилання для завантаження Excel

Застосунок


Personal Accounting
Додати транзакцію
Транзакції / Експорт
Від: 2025-08-26
До: 2025-09-26
Завантажити
Експорт CSV
Баланс: Основний рахунок = 16480.00
Дата
Сума
Тип
Нотатка
2025-09-26
5000.00
INCOME
Подарунок на день народження
2025-09-26
-500.00
EXPENSE
Сукня
2025-09-26
-20.00
EXPENSE
Хліб
2025-09-26
12000.00
INCOME
Початкове поповнення





	A	B	C	D	E	F	G	H	I
1	date	account	category	type	amount	note			
2	26.09.2025	Основний рахунок	Подарунок	INCOME	5000.00	Подарунок на день народження			
3	26.09.2025	Основний рахунок	Одяг	EXPENSE	-500.00	Сукня			
4	26.09.2025	Основний рахунок	Продукти	EXPENSE	-20.00	Хліб			
5	26.09.2025	Основний рахунок	Зарплата	INCOME	12000.00	Початкове поповнення			
6									
7									
8									
9									
10									
11									

Висновок

У процесі виконання роботи мною було розроблено настільний застосунок «Особиста бухгалтерія» для ведення обліку доходів та витрат. Система реалізована на основі архітектури з повним циклом роботи з даними: від введення транзакцій на формі до їх збереження у базі даних PostgreSQL та подальшого відображення на графічному інтерфейсі.

Програма підтримує роботу з різними рахунками та категоріями, надає можливість обліку доходів і витрат, формування періодичних вибірок та експорту даних у формат CSV. Це забезпечує зручність у користуванні та

дає змогу отримувати аналітичну інформацію про фінансовий стан користувача.

Результати підтвердили, що система відповідає поставленим вимогам: реалізовано дві основні форми (додавання транзакцій і перегляд/експорт), забезпечено повноцінну інтеграцію з БД, а також перевірено правильність роботи обліку та підрахунку залишків.

Таким чином, поставлене завдання виконано повністю, і розроблений застосунок може використовуватися як базовий інструмент для особистої бухгалтерії.

Контрольні питання

1. Що собою становить діаграма розгортання?

Діаграма розгортання UML відображає фізичну архітектуру системи: як програмні модулі (артефакти) розміщуються на апаратних і програмних вузлах, та як вони між собою взаємодіють у реальному середовищі

2. Які бувають види вузлів на діаграмі розгортання?

Основні типи вузлів - це апаратні (сервери, ПК, мобільні пристрої) та програмні (віртуальні машини, контейнери, середовища виконання). Вузли можуть бути вкладеними один в одного, утворюючи ієрархію

3. Які бувають зв'язки на діаграмі розгортання?

Можуть бути: асоціації (фізичні з'єднання між вузлами), комунікаційні лінії (мережеві зв'язки), залежності (відображають використання одного елемента іншим)

4. Які елементи присутні на діаграмі компонентів?

Компоненти системи, інтерфейси (надавані та потрібні), залежності між ними, а також артефакти, які реалізують ці компоненти. Часто додають підсистеми для структурування

5. Що становлять собою зв'язки на діаграмі компонентів?

Зв'язки показують відношення використання чи залежності: один компонент може використовувати інтерфейс іншого, або бути реалізованим

через певний артефакт. Це дозволяє зрозуміти, як частини системи взаємодіють

6. Які бувають види діаграм взаємодії?

В UML виділяють: діаграми послідовностей, діаграми комунікацій, діаграми часових обмежень (timing), діаграми огляду взаємодій. Всі вони відображають динамічні аспекти системи

7. Для чого призначена діаграма послідовностей?

Вона потрібна для того, щоб показати обмін повідомленнями між об’єктами у часі. Це допомагає зрозуміти порядок виконання операцій і логіку роботи сценарію системи

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Основні елементи: актори, об’єкти чи класи, життєві лінії (lifelines), повідомлення (синхронні та асинхронні виклики), блоки альтернатив та циклів, умови виконання

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Діаграми варіантів використання показують, які функції виконує система з точки зору користувача, а діаграми послідовностей деталізують ці функції, описуючи послідовність взаємодій між об’єктами

10. Як діаграми послідовностей пов’язані з діаграмами класів?

Діаграма класів описує статичну структуру системи, а діаграма послідовностей - динамічну поведінку об’єктів, створених із цих класів. Вона фактично демонструє, як класи співпрацюють у конкретних сценаріях