Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 3

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Особиста бухгалтерія»

Виконала:
студентка групи ІА-33
Самойленко Анастасія

Перевірив:
асистент кафедри ІСТ
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.).

Вступ

Проєктування програмних систем є важливим етапом у процесі їх розроблення, адже саме на цьому етапі визначається архітектура, структура та способи взаємодії між компонентами програмного забезпечення. Для ефективного відображення цих аспектів широко застосовується UML (Unified Modeling Language) — уніфікована мова моделювання, яка дає змогу створювати візуальні діаграми, що описують як логічну, так і фізичну структуру системи.

У межах даної лабораторної роботи увага зосереджена на вивченні основ побудови діаграм розгортання та компонентів, які дозволяють відобразити архітектуру системи з точки зору її фізичного розміщення та взаємодії між складовими. Крім того, важливою частиною роботи є розробка діаграм послідовностей, що демонструють обмін повідомленнями між об’єктами на основі сценаріїв, створених у попередній лабораторній роботі.

Виконання цієї роботи сприятиме формуванню практичних навичок архітектурного моделювання та аналізу систем, що є необхідною складовою підготовки фахівців у галузі інформаційних технологій та програмної інженерії.

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)
Діаграма розгортання відображає фізичне розміщення програмних компонентів системи на апаратному забезпеченні. Основними елементами є вузли (nodes), які з'єднані інформаційними зв'язками. Вузли можуть бути двох типів:

- Пристрої (device) – фізичні елементи, наприклад сервер або комп'ютер;
- Середовища виконання (execution environment) – програмне забезпечення, у якому виконуються додатки (наприклад, ОС або вебсервер).
  У вузлах можуть бути розміщені артефакти – виконувані файли, скрипти, бази даних чи конфігураційні файли. Діаграми розгортання бувають описові (загальне уявлення) та екземплярні (конкретне розміщення компонентів).

Діаграма компонентів (Component Diagram)
Діаграма компонентів показує структуру програмної системи у вигляді взаємопов'язаних модулів (компонентів).
Вона використовується для відображення:

- логічної структури програмного коду;
- залежностей між компонентами;
- розподілу функцій між клієнтською та серверною частинами.
  Компоненти можуть бути виконуваними файлами (.exe, .dll), бібліотеками або модулями. Така діаграма допомагає візуалізувати архітектуру системи, визначити залежності між модулями та забезпечити багаторазове використання коду.

Діаграма послідовностей (Sequence Diagram)
Діаграма послідовностей моделює взаємодію між об'єктами системи у певній часовій послідовності.
Основні елементи діаграми:

- Актори (Actors) – користувачі або зовнішні системи;
- Об'єкти/класи – учасники взаємодії, для яких показано лінії життя;
- Повідомлення – обмін викликами методів між об'єктами;
- Активності – періоди виконання дій;
- Контрольні структури – умовні або циклічні блоки.
  Діаграми послідовностей допомагають описати сценарії роботи системи, зрозуміти логіку обміну повідомленнями між компонентами та виявити можливі помилки на етапі проєктування.

Хід роботи

1. Проаналізувавши створені в попередній лабораторній роботі діаграми (зокрема діаграму компонентів і діаграму класів), було спроєктовано діаграму розгортання, яка відображає фізичну структуру системи та взаємодію між її елементами під час виконання. (рис. 1)

Діаграма розгортання демонструє фізичну архітектуру програмного комплексу «Personal Accounting», який реалізує функціональність ведення особистих фінансів, контролю витрат і доходів, роботи з періодичними операціями, а також обробки сканованих чеків та інтеграції із зовнішніми сервісами.

Архітектура включає кілька фізичних вузлів:

1. Комп’ютер користувача

На стороні користувача розміщується десктопний клієнтський застосунок *PersonalAccountingClient.jar*, який виконується у середовищі JRE (Java Runtime Environment).
Програма має графічний інтерфейс користувача (наприклад, створений за допомогою JavaFX) і забезпечує введення, перегляд та аналіз фінансових операцій. Клієнт взаємодіє із серверною частиною системи через REST API з використанням захищеного протоколу HTTPS.

2. Сервер застосунків

На цьому вузлі розгорнуто середовище виконання JRE / APP-контейнер, у якому працюють окремі сервіси системи:

- AuthService — відповідає за автентифікацію користувачів і безпеку доступу;
- FinanceService — основний компонент, що керує рахунками, транзакціями, бюджетами та фінансовими звітами;
- RecurringService — забезпечує облік періодичних операцій, таких як орендна плата або зарплата;
- ImportService — обробляє вхідні файли та скановані чеки, інтегруючи їх з відповідними категоріями витрат;
- ExportService — реалізує експорт даних у формати Excel або реляційні джерела.

Усі сервіси взаємодіють між собою через REST API, що підвищує масштабованість і спрощує підтримку системи.

3. Сервер бази даних

Використовується MySQL як реляційна система керування базами даних.
База зберігає інформацію про користувачів, рахунки, транзакції, категорії витрат, періодичні операції та інші довідкові дані.

Доступ до бази здійснюється за допомогою JDBC, що забезпечує уніфіковану взаємодію між сервісами та сховищем даних.

4. Зовнішні сервіси (Інтернет / Хмара)

Цей вузол представлений у вигляді хмарного середовища, яке містить зовнішні інтегровані сервіси:

- Rates Provider Module — отримання актуальних валютних курсів через HTTPS-з’єднання;
- Email-сервіс (SMTP) — надсилання сповіщень і звітів користувачам за допомогою захищеного каналу TLS;
- Scan Import Service — обробка завантажених сканів чеків і квитанцій, що дозволяє автоматично розпізнавати та класифікувати витрати.

Взаємодія між сервером застосунків і хмарними сервісами відбувається через інтернет за безпечними протоколами зв’язку.

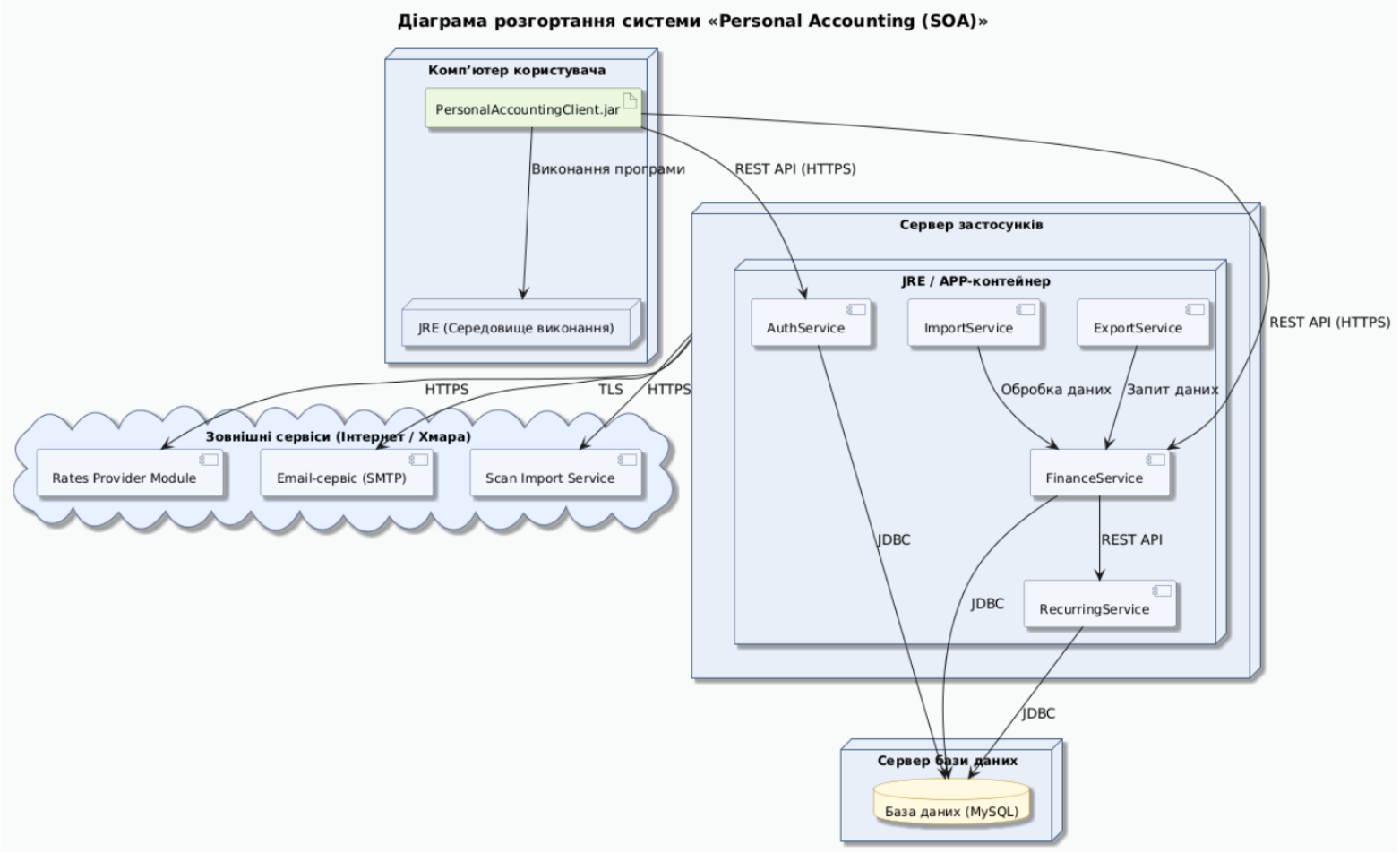


Рисунок 1 – Діаграма розгортання (Deployment Diagram)

2. Для візуалізації структури проєктованої системи «Personal Accounting» була розроблена UML-діаграма компонентів (рис.2), яка відображає логічне та технічне розбиття системи на окремі модулі. Діаграма дозволяє зрозуміти, як взаємодіють ключові компоненти, які інтерфейси використовуються, а також як організовано доступ до даних та взаємодія із зовнішніми сервісами.
   Основні блоки системи, представлені на діаграмі:
   1. Клієнтська частина:
      - Виконує функції інтерфейсу користувача через компонент PersonalAccountingClient.
      - Взаємодіє з серверною частиною через блок REST API, обмінюючись даними у форматі JSON.
   2. REST API:
      - Містить інтерфейси сервісів (IAuthAPI, IFinanceAPI, IRecurringAPI, IImportAPI, IExportAPI).
      - Виконує роль шлюзу між клієнтською частиною та сервісами серверної частини, забезпечуючи чітко визначені контракти взаємодії.
   3. Серверна частина (Бізнес-логіка):
      - Містить основні сервіси системи: AuthService, FinanceService, RecurringService, ImportService, ExportService.
      - Сервіси реалізують відповідні інтерфейси REST API і взаємодіють між собою для обробки бізнес-логіки (наприклад, FinanceService використовує RecurringService для обробки повторюваних транзакцій).
   4. Data Access Layer:
      - Використовує паттерн Repository для доступу до даних.
      - Загальний інтерфейс IRepository<T> реалізується конкретними репозиторіями: UserRepository, AccountRepository, TransactionRepository, RecurringRuleRepository, AttachmentRepository.
      - Репозиторії забезпечують взаємодію із реляційною базою даних та сховищем файлів.
   5. Зберігання даних:
      - RDBMS (MySQL) для зберігання структурованих даних (користувачі, рахунки, транзакції).
      - File Storage для зберігання файлів документів та квитанцій, що прикріплюються до транзакцій.
   6. Зовнішні сервіси:
      - Email Service (SMTP) – для відправки повідомлень користувачам.
      - Rates Provider Module – отримання актуальних курсів валют.
      - Scan Import Service – для завантаження та обробки фінансових документів.

   Особливості та переваги розробленої діаграми компонентів:

- Чітко показує взаємодію між клієнтом, REST API та серверними сервісами.
- Відображає використання інтерфейсів сервісів, що забезпечує модульність і можливість багаторазового використання компонентів.
- Показує зв’язки з Data Access Layer, включаючи загальні інтерфейси репозиторіїв, що спрощує підтримку та розширення системи.
- Включає взаємодію із зовнішніми сервісами та сховищем файлів, що відображає реалістичну архітектуру системи.

Розроблена діаграма компонентів слугує основою для подальшого проектування та впровадження системи, дозволяє оцінити вплив змін у одному компоненті на інші.

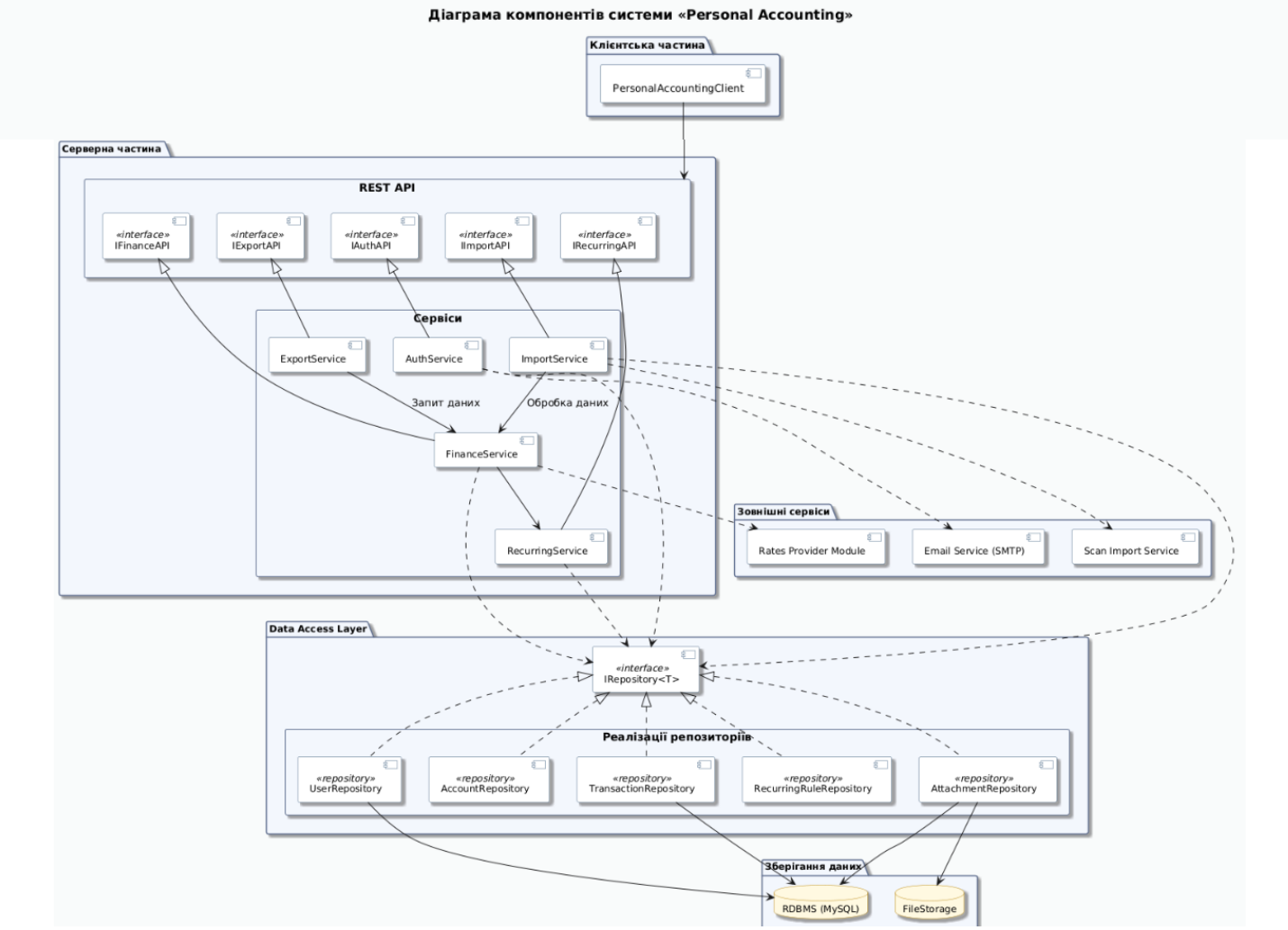


Рисунок 2 – Діаграма компонентів

3. Ми доопрацювали та розробили діаграму розгортання для проєктованої системи «Personal Accounting» (рис.3). Діаграма показує фізичну архітектуру системи, розподіл компонентів по вузлах та їх взаємодію. Клієнтська частина представлена виконуваним артефактом на комп’ютері користувача, серверна частина містить бізнес-сервіси в контейнері JRE, а сервер бази даних включає

MySQL та сховище файлів для чеків і квитанцій. Додано балансувальник навантаження для розподілу запитів між серверами застосунків. На діаграмі вказані протоколи та стандартні порти для взаємодії компонентів та зовнішніх сервісів, що дозволяє чітко бачити шляхи комунікації та фізичне розміщення системи. Розроблена діаграма є структурованою, наочною та може бути використана для планування розгортання, масштабування та тестування системи.

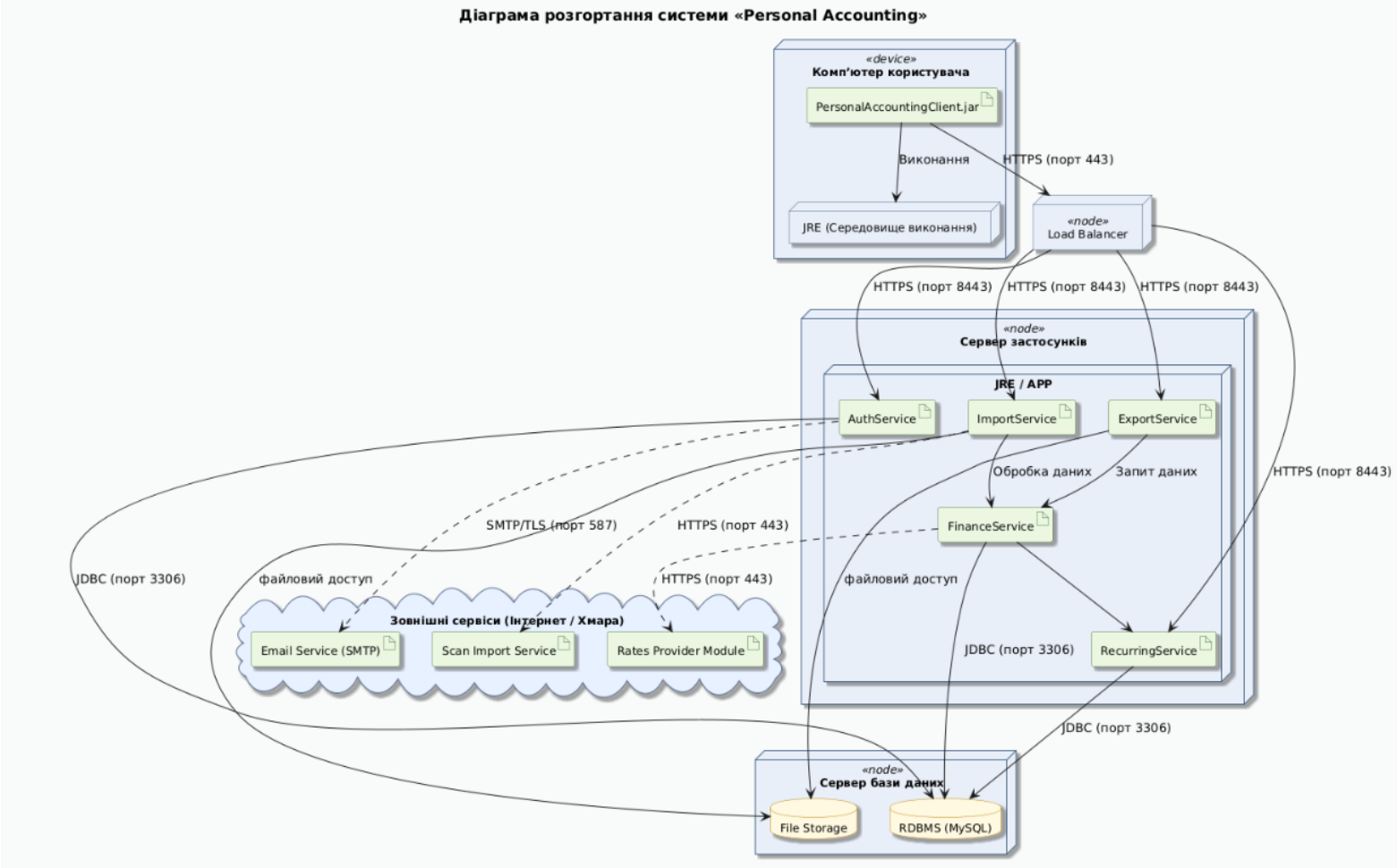


Рисунок 3 – Діаграма розгортання для проєктованої системи

4. Далі ми розробили три діаграми послідовності для основних сценаріїв нашої проєктованої системи: додавання нової транзакції, планування регулярного платежу та формування фінансового звіту. Для кожного сценарію визначено взаємодію користувача із системою, задіяні сервіси та репозиторії, що відповідають архітектурі системи. Діаграми демонструють порядок дій користувача, обробку даних сервісами (AuthService, FinanceService, RecurringService, ImportService, ExportService), взаємодію з репозиторіями (TransactionRepository, RecurringRuleRepository, AccountRepository), а також обробку винятків і повідомлень для користувача. Результатом роботи є три завершені діаграми послідовності, які наочно показують логіку виконання основних функцій системи та взаємодію між компонентами (рис. 5–7).

Сценарій 1: Додавання нової транзакції (дохід / витрата)

Передумови:

- Користувач увійшов у систему.
- Створено хоча б один рахунок і категорію.
- Система працює у штатному режимі.

Постумови:

- У базі даних з’являється новий запис транзакції.
- Баланс рахунку оновлюється відповідно до типу транзакції (дохід/витрата).
- Система зберігає оновлену аналітику.

Взаємодіючі сторони:

Користувач, Система.

Короткий опис:

Користувач додає нову фінансову операцію, вказуючи тип, суму, категорію, опис і дату. Система перевіряє дані, зберігає їх у базі та оновлює баланс рахунку.

Основний потік подій:

1. Користувач відкриває розділ «Транзакції».
2. Обирає опцію «Додати транзакцію».
3. Система запитує: тип операції (дохід/витрата), рахунок, категорію, суму, дату, опис.
4. Користувач заповнює поля та натискає «Зберегти».
5. Система перевіряє коректність введених даних (позитивна сума, вибрані рахунок і категорія).
6. Якщо перевірка успішна, система створює об’єкт Transaction і додає його до вибраного Account.
7. Баланс рахунку оновлюється, а транзакція з’являється у списку операцій.
8. Система показує повідомлення «Транзакцію успішно додано».

Винятки:

- Некоректна сума (нуль або від’ємне значення) → повідомлення «Сума некоректна».
- Не вибрано рахунок або категорію → система просить уточнити.
- Відсутній зв’язок із базою → повідомлення «Помилка збереження даних».

Примітки:

Може бути викликано з основного меню або через коротке посилання «+ Додати транзакцію» на головному екрані.

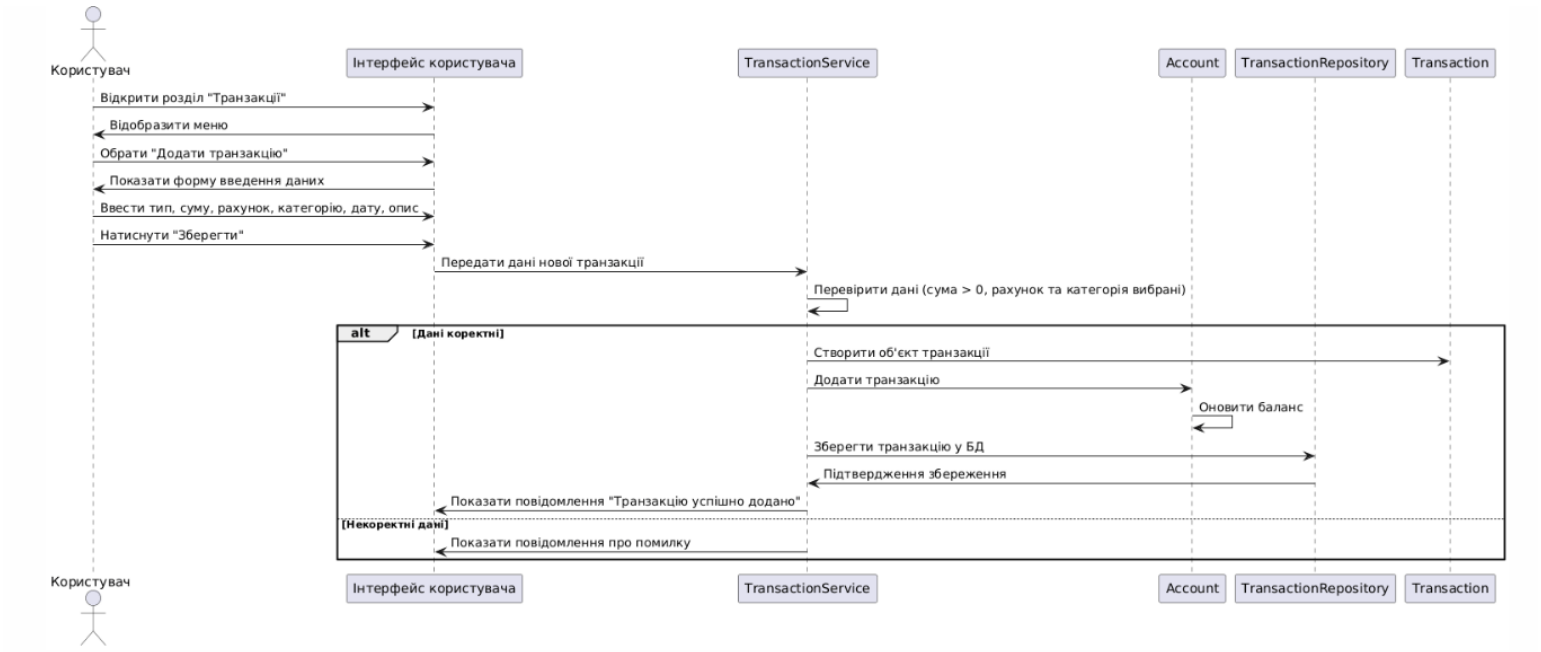


Рисунок 4 – Діаграма послідовностей для сценарію №1

Сценарій 2: Планування регулярного платежу (зарплата, оренда, кредит)

Передумови:

- Користувач авторизований.
- Існує хоча б один рахунок і категорія.
- Система синхронізована за часом.

Постумови:

- Створено запис у таблиці RegularPayment.
- Збережено шаблон транзакції для автоматичного створення.
- Налаштовано нагадування або автозапуск транзакцій.

Взаємодіючі сторони:

Користувач, Система.

Короткий опис:

Користувач створює регулярний платіж (наприклад, оренда чи заробітна плата). Система зберігає шаблон, додає його до розкладу й автоматично створює транзакції у зазначені дати.

Основний потік подій:

1. Користувач відкриває розділ «Регулярні платежі».
2. Обирає опцію «Додати новий платіж».
3. Система пропонує ввести назву, тип (дохід/витрата), суму, періодичність, дату початку, рахунок і категорію.
4. Користувач заповнює поля й натискає «Підтвердити».
5. Система перевіряє дані (усі поля заповнені, коректна дата, позитивна сума).

6. Система створює об’єкт RegularPayment і додає його до бази даних.
7. Запускається сервіс SchedulerService, який планує майбутні транзакції.
8. Система відображає повідомлення «Регулярний платіж створено».

Винятки:

- Некоректна дата → повідомлення «Дата повинна бути в майбутньому».
- Відсутня періодичність → система просить уточнити інтервал.
- Помилка при збереженні → система пропонує повторити.

Примітки:

При кожному запуску застосунку система перевіряє, чи настав час для автоматичного виконання регулярних платежів.

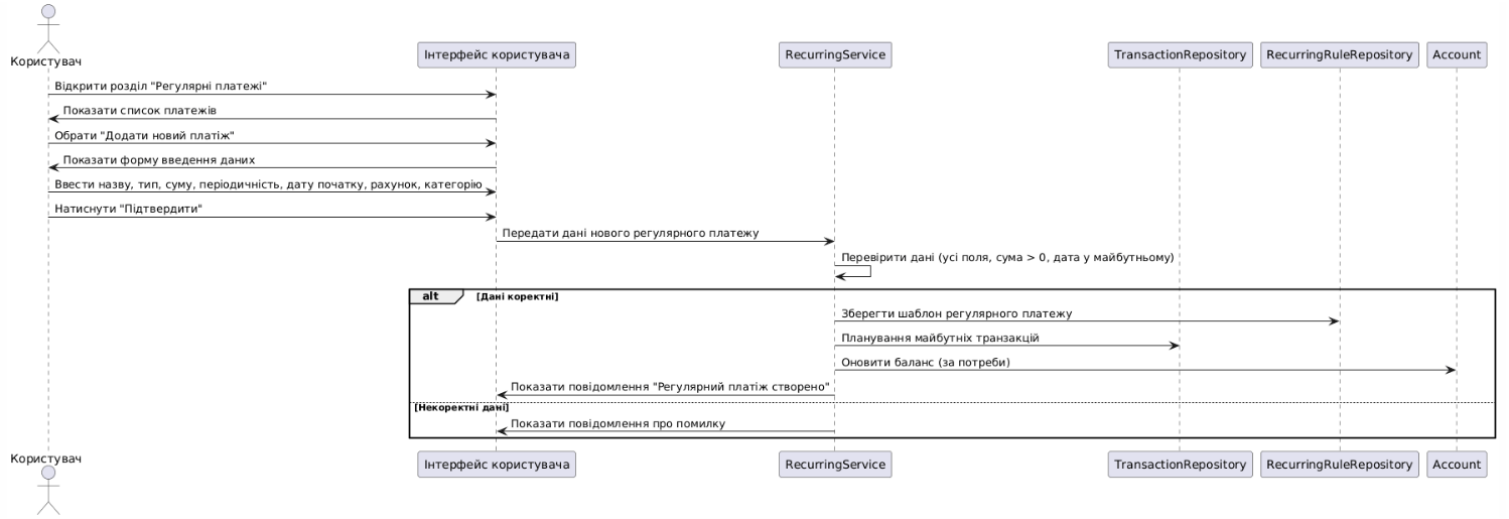


Рисунок 5 – Діаграма послідовностей для сценарію №2

Сценарій 3: Формування фінансового звіту

Передумови:

- Користувач увійшов у систему.
- Є транзакції у вибраному періоді.

Постумови:

- Сформовано фінансовий звіт у вигляді таблиці або діаграми.
- Звіт можна зберегти, експортувати або надрукувати.

Взаємодіючі сторони:

Користувач, Система.

Короткий опис:

Користувач формує фінансовий звіт за вибраний період. Система виконує вибірку з бази даних, обчислює підсумки та будує графічне відображення доходів і витрат.

Основний потік подій:

1. Користувач переходить до розділу «Аналітика і звіти».
2. Обирає тип звіту (доходи, витрати, баланс).

3. Вказує період (початкова й кінцева дати).
4. Система формує запит до бази транзакцій.
5. TransactionService отримує дані та передає їх у ReportGenerator.
6. ReportGenerator створює аналітичний звіт і повертає його системі.
7. Система відображає звіт користувачу у вигляді таблиці чи діаграми.
8. Користувач за потреби натискає «Експортувати в Excel».
9. IntegrationService генерує файл і пропонує зберегти його.

Винятки:

- У періоді відсутні транзакції → повідомлення «Дані для звіту відсутні».
- Помилка експорту → система повідомляє «Не вдалося зберегти файл».

Примітки:

Система може зберігати останні параметри звіту для швидкого повторного формування.

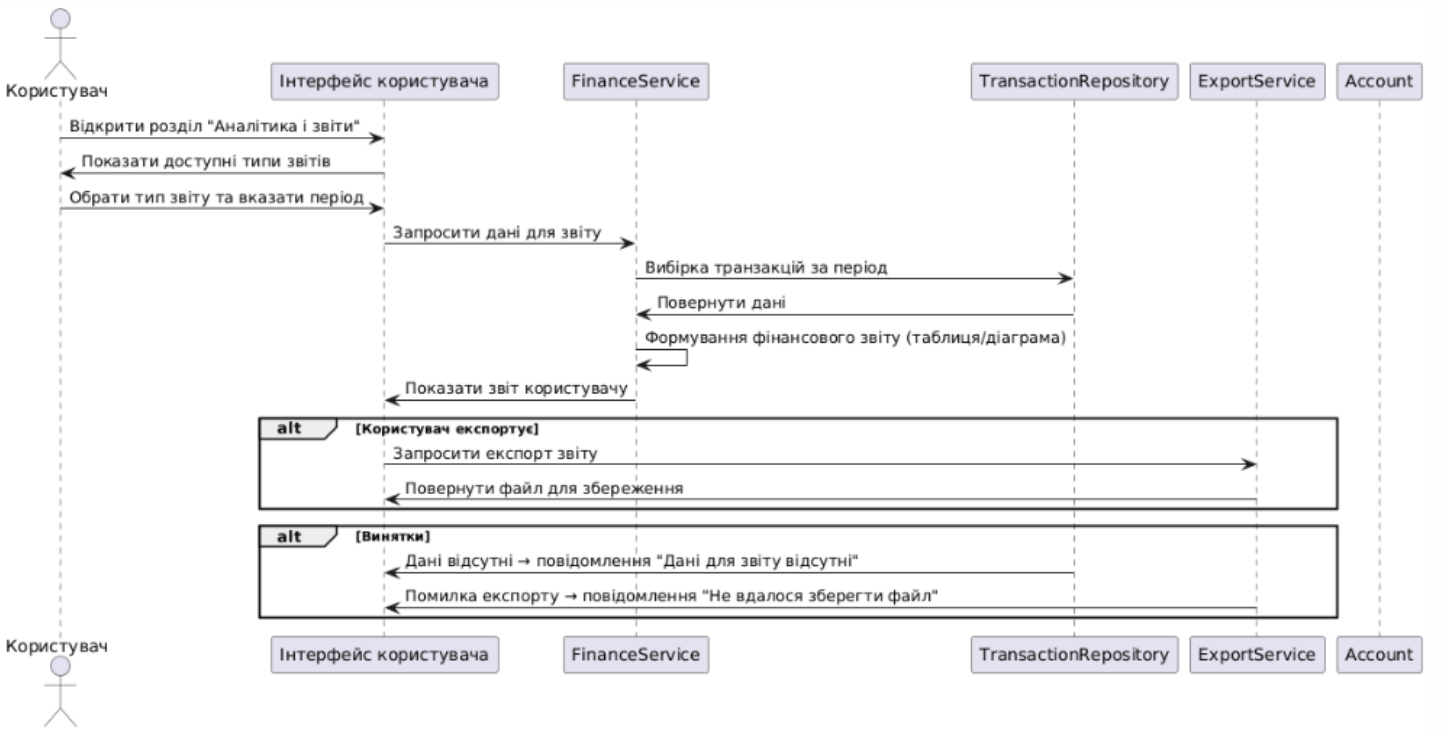
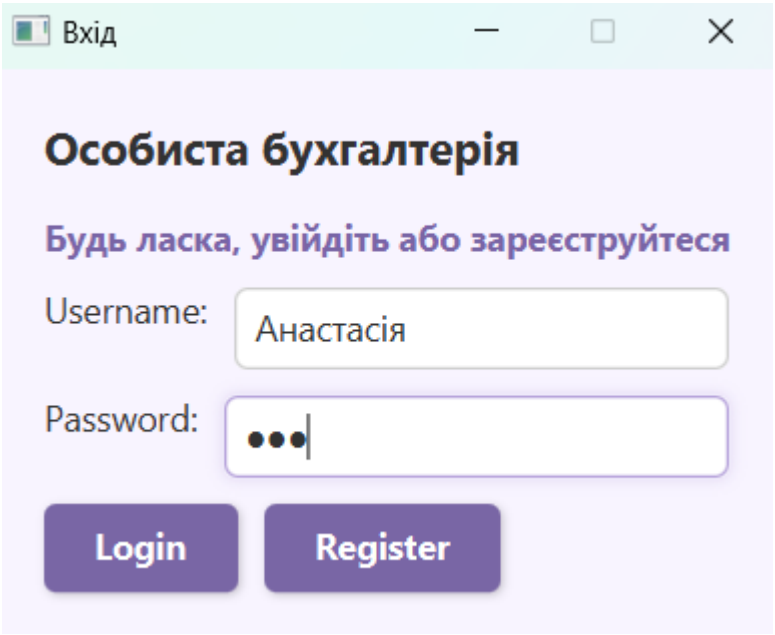
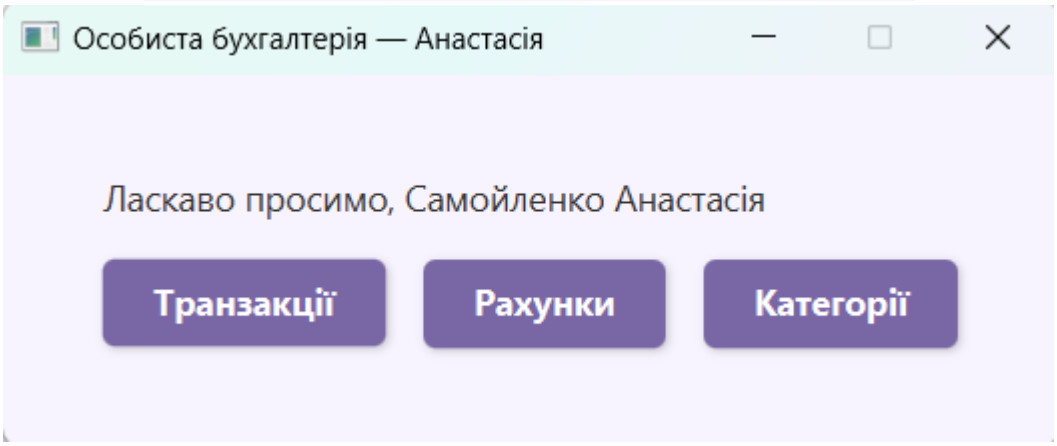
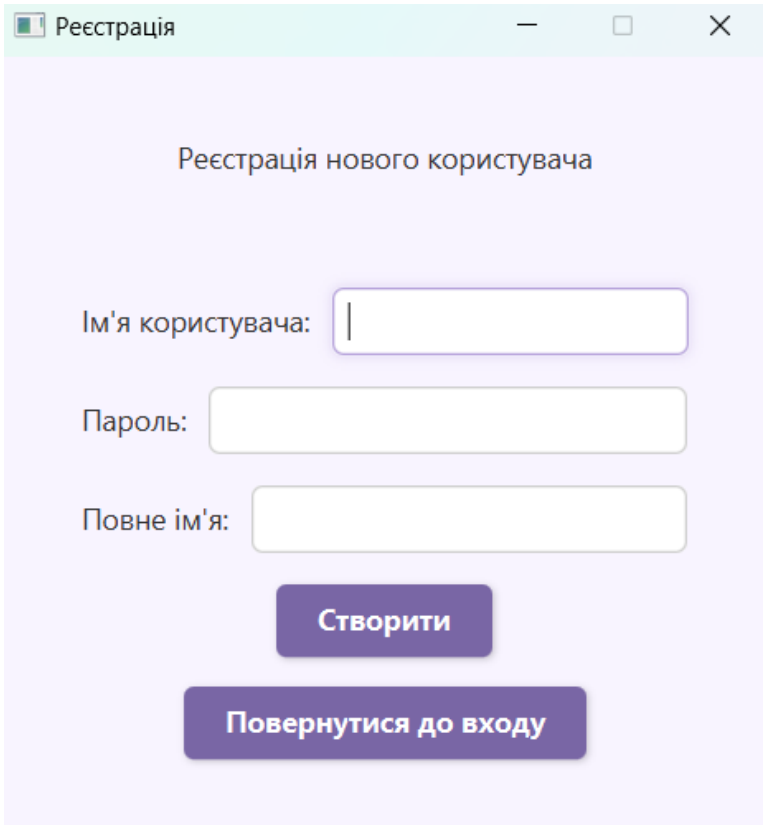


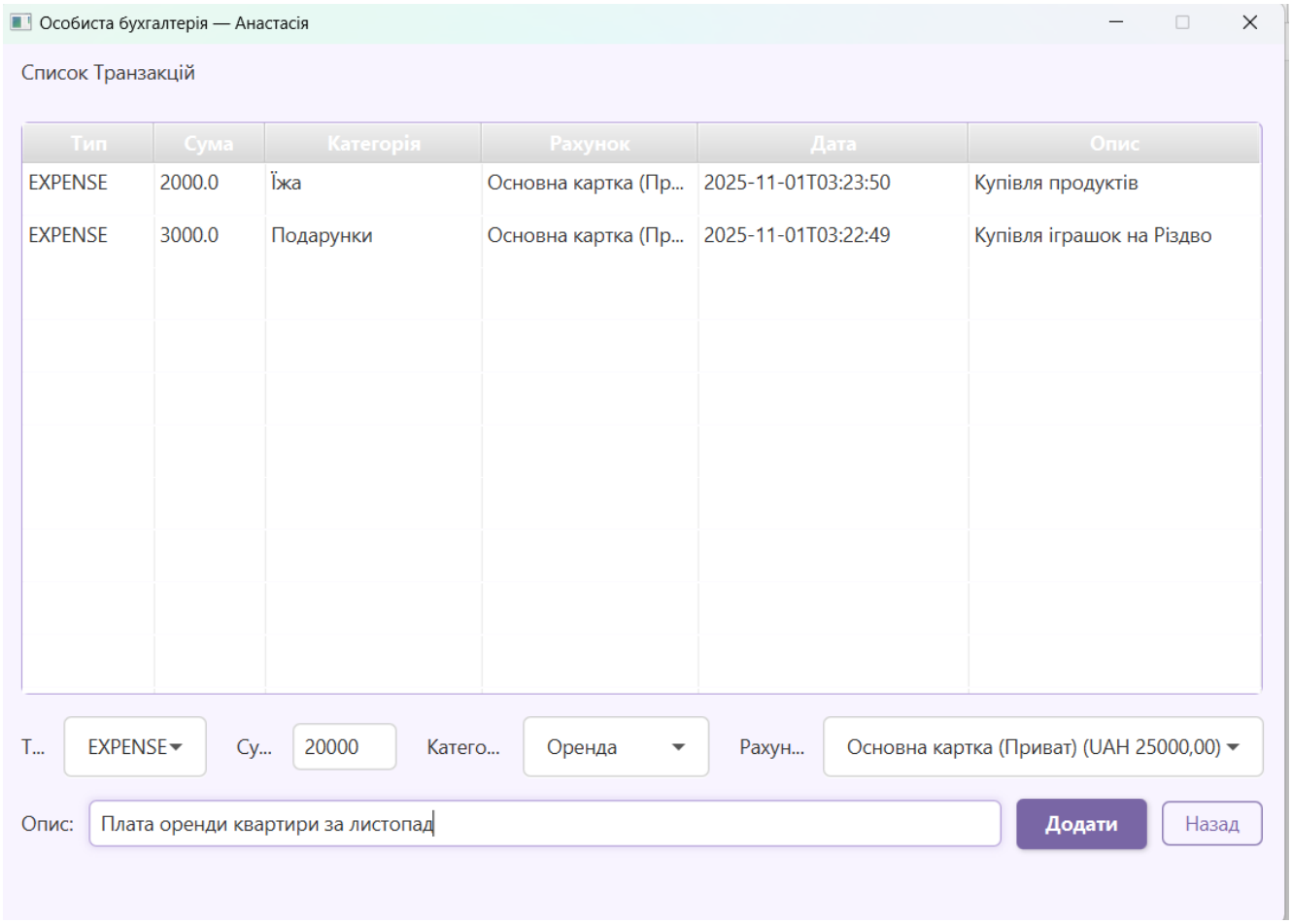
Рисунок 6 – Діаграма послідовностей для сценарію №3

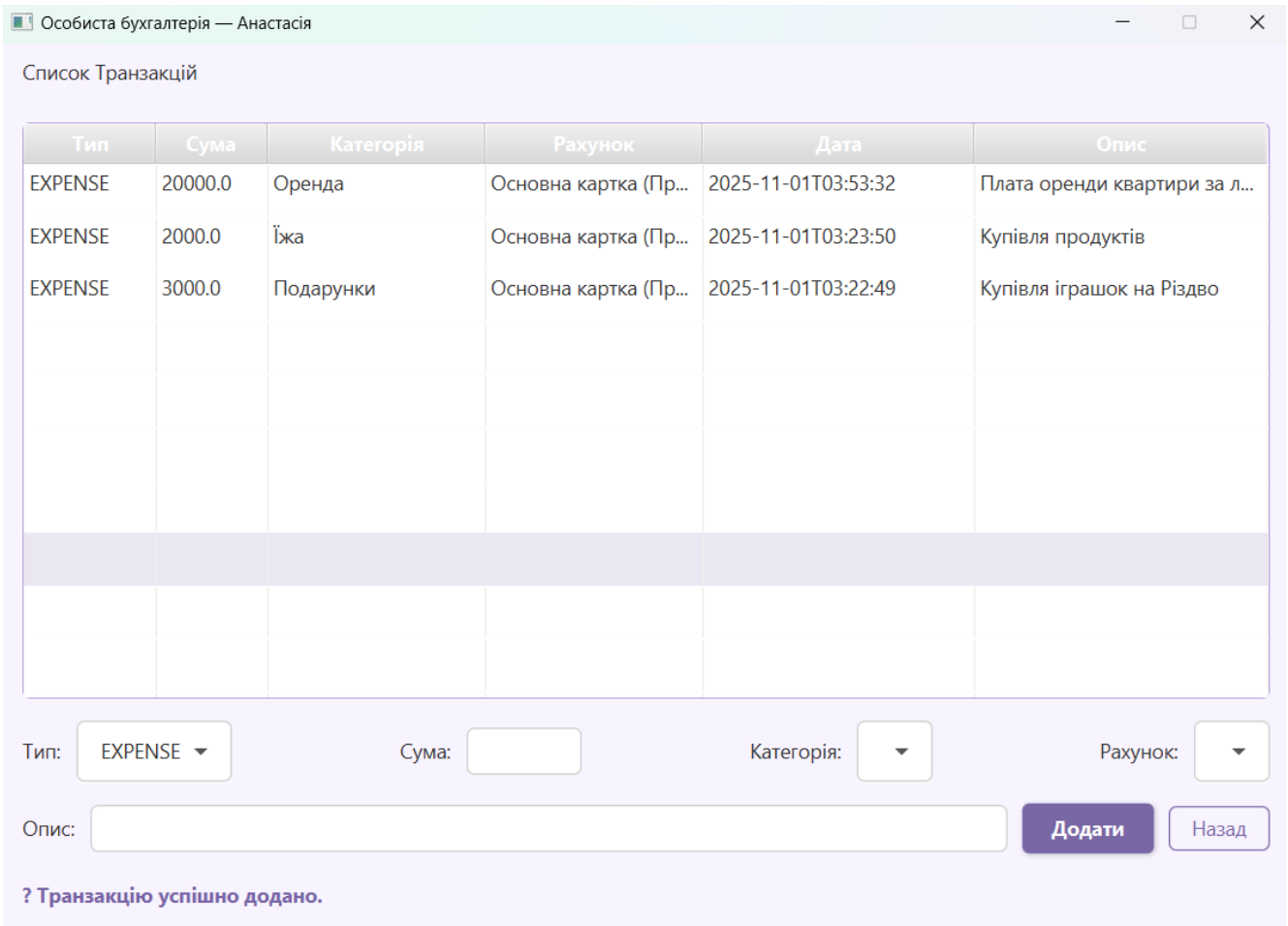
5. На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи (рис.7-13).

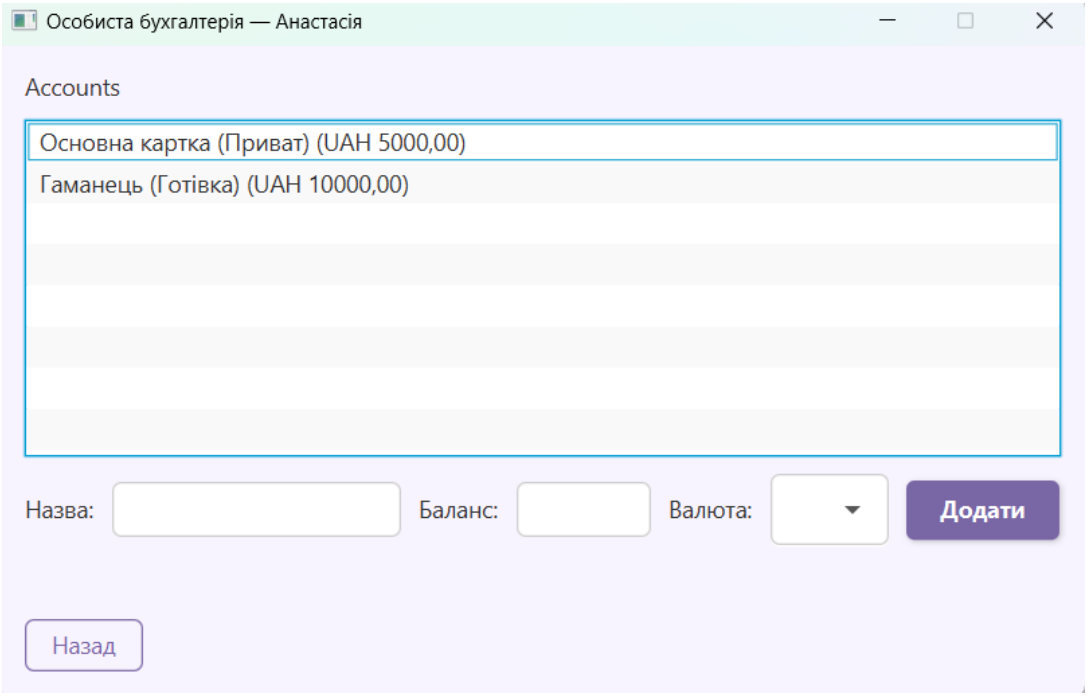

Вхід
Особиста бухгалтерія
Будь ласка, увійдіть або зареєструйтеся
Username:
Анастасія
Password:
Login
Register


Особиста бухгалтерія — Анастасія
Ласкаво просимо, Самойленко Анастасія
Транзакції
Рахунки
Категорії


Реєстрація
Реєстрація нового користувача
Ім'я користувача:
Пароль:
Повне ім'я:
Створити
Повернутися до входу


Особиста бухгалтерія — Анастасія
Список Транзакцій
Тип
Сума
Категорія
Рахунок
Дата
Опис
EXPENSE
2000.0
Їжа
Основна картка (Пр...
2025-11-01T03:23:50
Купівля продуктів
EXPENSE
3000.0
Подарунки
Основна картка (Пр...
2025-11-01T03:22:49
Купівля іграшок на Різдво
Т...
EXPENSE
Су...
20000
Катего...
Оренда
Рахун...
Основна картка (Приват) (UAH 25000,00)
Опис:
Плата оренди квартири за листопад
Додати
Назад


Особиста бухгалтерія — Анастасія
Список Транзакцій
Тип
Сума
Категорія
Рахунок
Дата
Опис
EXPENSE
20000.0
Оренда
Основна картка (Пр...
2025-11-01T03:53:32
Плата оренди квартири за л...
EXPENSE
2000.0
Їжа
Основна картка (Пр...
2025-11-01T03:23:50
Купівля продуктів
EXPENSE
3000.0
Подарунки
Основна картка (Пр...
2025-11-01T03:22:49
Купівля іграшок на Різдво
Тип:
EXPENSE
Сума:
Категорія:
Рахунок:
Опис:
Додати
Назад
? Транзакцію успішно додано.



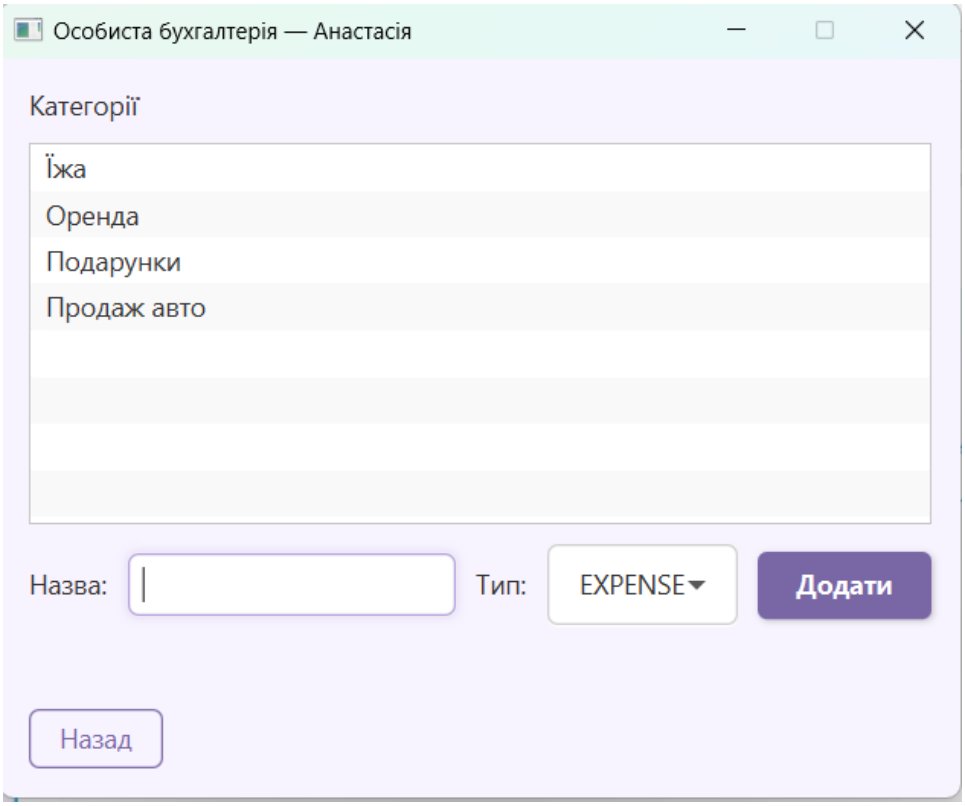


Рисунок 7-13 - Застосунок

Та зберігання даних в БД:

id	username	password_hash	full_name	email	created_at
1	Анастасія	123	Самойленко Анастасія	as@gmail.com	2025-11-01 01:50:10
NULL	NULL	NULL	NULL	NULL	NULL

id	user_id	name	type	currency	balance	created_at
1	1	Основна картка (Приват)	CASH	UAH	5000.0000	2025-11-01 03:11:04
2	1	Гаманець (Готівка)	CASH	UAH	10000.0000	2025-11-01 03:11:31

id	user_id	name	type	created_at
1	1	Подарунки	EXPENSE	2025-11-01 03:05:30
2	1	Їжа	EXPENSE	2025-11-01 03:06:51
3	1	Продаж авто	INCOME	2025-11-01 03:07:23
4	1	Оренда	EXPENSE	2025-11-01 03:07:38
NULL	NULL	NULL	NULL	NULL

id	account_id	category_id	amount	currency	description	trans_date	created_at	recurring_rule_id	user_id
1	1	1	3000.0000	UAH	Купівля іграшок на Різдво	NULL	2025-11-01 03:22:49	NULL	1
2	1	2	2000.0000	UAH	Купівля продуктів	NULL	2025-11-01 03:23:50	NULL	1
3	1	4	20000.0000	UAH	Плата оренди квартири за листопад	NULL	2025-11-01 03:53:32	NULL	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Відповіді на контрольні запитання:

1. Що собою становить діаграма розгортання?

Діаграма розгортання являє собою візуалізацію фізичної архітектури програмної системи. Вона показує, на яких реальних апаратних засобах (серверах, пристроях) фізично розташовані та виконуються окремі програмні компоненти (артефакти).

2. Які бувають види вузлів на діаграмі розгортання?

На діаграмі розгортання вузол (node), що є контейнером для програмного забезпечення, може бути одного з двох основних типів:

- Пристрій (device): Будь-яке фізичне обладнання, як-от комп'ютери, сенсори або інша апаратура, що є частиною системи.
- Середовище виконання (execution environment): Програмна платформа, яка функціонує в межах пристрою і здатна розміщувати інше ПЗ. Прикладами є операційна система (ОС) або контейнерний процес (наприклад, вебсервер чи середовище JVM).

3. Які бувають зв’язки на діаграмі розгортання?

Зв'язки на цій діаграмі описують, як вузли з'єднані між собою:

- Асоціації: Позначають фізичні з'єднання або пряму взаємодію між вузлами.
- Комунікаційні лінії: Відображають мережеві канали чи інші шляхи передачі даних, через які вузли обмінюються інформацією.
- Залежності: Показують, що один вузол використовує ресурси або функціональність іншого.

4. Які елементи присутні на діаграмі компонентів?

Діаграма компонентів відображає будівельні блоки системи. Основні елементи включають:

- Компоненти: Самодостатні, замінювані частини системи.

- Інтерфейси: Які надаються (реалізуються компонентом) та потрібні (використовуються компонентом) для взаємодії.
- Залежності: Відносини, що показують використання чи зв'язок між компонентами.
- Артефакти: Конкретні файли чи об'єкти, які втілюють ці компоненти.
- Для кращої організації часто використовуються підсистеми як логічні групи компонентів.

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

Зв'язки на цій діаграмі визначають взаємовідносини між частинами системи. Вони вказують на:

- Використання інтерфейсу: Коли один компонент звертається до функціональності (інтерфейсу), яку надає інший компонент.
- Реалізацію: Зв'язок, що демонструє, який артефакт (наприклад, виконуваний файл) фактично містить код певного компонента. Це дає змогу зрозуміти, як функціональні одиниці системи співпрацюють і залежать одна від одної.

6. Які бувають види діаграм взаємодії?

UML класифікує динамічні діаграми взаємодії, які відображають поведінку системи у часі, на такі типи:

- Діаграми послідовностей (Sequence): Фокусуються на хронології обміну повідомленнями.
- Діаграми комунікацій (Communication): Акцентують увагу на структурі зв'язків між об'єктами.
- Діаграми часових обмежень (Timing): Детально показують часові обмеження та зміни стану.
- Діаграми огляду взаємодій (Interaction Overview): Поєднують діаграми активностей із фрагментами інших діаграм взаємодії.

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей слугує для візуалізації динамічного виконання сценарію. Її основне призначення — показати, у якому порядку та в який час об'єкти системи обмінюються повідомленнями (викликами). Це дозволяє проаналізувати хронологію операцій і логіку роботи конкретного процесу чи функції.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

До основних складових, що формують діаграму, належать:

- Учасники: Актори (зовнішні користувачі чи системи) та Об'єкти/Класи, які беруть участь у взаємодії.
- Життєві лінії (Lifelines): Вертикальні лінії, що позначають існування об'єкта протягом сценарію.
- Повідомлення: Горизонтальні стрілки, що показують виклики між об'єктами (можуть бути синхронними чи асинхронними).
- Фрагменти (Blocks): Елементи для відображення альтернативних шляхів, циклів та умов виконання (наприклад, alt, loop, opt).

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?
Ці діаграми доповнюють одна одну:

- Діаграми варіантів використання описують високорівневі функції системи з точки зору кінцевого користувача ("ЩО" робить система).
- Діаграми послідовностей є детальним описом одного конкретного варіанта використання, показуючи покроковий обмін повідомленнями між внутрішніми об'єктами ("ЯК" система реалізує цю функцію).

10. Як діаграми послідовностей пов’язані з діаграмами класів?
Зв'язок між ними такий:

- Діаграма класів є статичною моделлю, що визначає структуру системи (які класи існують та як вони пов'язані).
- Діаграма послідовностей відображає динамічну поведінку, демонструючи, як екземпляри (об'єкти), створені на основі цих класів, взаємодіють між собою під час виконання певного сценарію. Вона оживляє статичну модель класів.

Висновки

В ході виконання лабораторної роботи ми ознайомилися з основами проєктування розгортання програмних систем, навчилися створювати діаграми розгортання, що показують фізичне розміщення компонентів та обладнання, на якому виконується програмне забезпечення, розглянули типи вузлів — пристрої та середовища виконання, і визначили їх роль у системі, вивчили типи зв’язків між вузлами та компонентами для опису фізичних підключень, комунікаційних ліній та залежностей, опанували створення діаграм компонентів з компонентами, інтерфейсами, артефактами та залежностями, а також структурування підсистем, розробили діаграми взаємодії, зокрема послідовностей, для відображення обміну повідомленнями між об’єктами та реалізації сценаріїв системи, зрозуміли взаємозв’язок між діаграмами варіантів використання, які визначають функції системи, та діаграмами послідовностей, що деталізують їх реалізацію, і таким чином закріпили практичні навички проєктування розгортання, компонентної структури та динамічної поведінки системи через UML-діаграми.