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 №2

із дисципліни *«Технології розроблення програмного забезпечення»*

Тема: «ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»

Виконала:
Студентка групи ІА-24
Сіденко Д.Д.

Перевірив:
Мягкий М.Ю.

Київ-2024

Тема лабораторних робіт:

Варіант 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.).

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.

2. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторної.

3. Намалюйте діаграму класів для реалізованої частини системи.

4. Виберіть 3 прецеденти і напишіть на їх основі прецеденти.

5. Розробити основні класи і структуру системи баз даних.

6. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.

7. Підготувати звіт про хід виконання лабораторних робіт. Звіт, що подається повинен містити: діаграму прецедентів, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Зміст

Хід роботи:

Крок 1. Аналіз теми та шаблонів проектування.

Перед початком роботи над лабораторною роботою, варто проаналізувати надані шаблони проектування та визначитись з варіантами їх використання.

Шаблон	Суть	Варіант застосування
State	Шаблон дозволяє об'єкту змінювати свою поведінку в залежності від свого стану, забезпечуючи динамічну зміну класів.	Стани обробки періодичних платежів ("активні", "призупинені", "скасовані")
Prototype	Дозволяє створювати нові об'єкти копіюванням існуючих. Використовується, коли створення об'єкта є складним або ресурсозатратним.	Створення нових транзакцій на основі існуючих (шаблони регулярних витрат/прибутків)
Decorator	Дозволяє динамічно додавати функціонал до об'єкта без зміни його структури, використовуючи обгортки.	Обгортки для рахунків, які надають інформацію про залишок, кредитний ліміт або спеціальні умови.
Bridge	Дозволяє розділяти абстракцію і реалізацію так, щоб вони могли	Створення гнучкої структури для експорту даних в різні формати (Excel, CSV) чи

	змінюватися незалежно один від одного.	взаємодії з різними джерелами даних (реляційні бази, локальні файли тощо)
Flyweight	Використовується для зменшення споживання пам'яті шляхом спільного використання однакових об'єктів.	Управління великою кількістю однотипних об'єктів, наприклад, транзакцій або витрат, які можуть мати спільні атрибути (категорії, дати).
SOA	Архітектурний стиль, який розбиває програму на незалежні сервіси, що взаємодіють через чітко визначені інтерфейси.	Програма може бути побудована як набір незалежних сервісів для управління фінансами
Репозиторій	Абстрагування логіки доступу до даних і ізолювати її від решти коду, що дозволяє зробити систему гнучкішою та легшою для підтримки.	Організація доступу до даних.

Крок 2. Створення діаграми прецедентів.

Для створення початкової концептуальної моделі представлення, я взяла за основні функції, що вже були вказані в темі.

Варіанти використання для програми "Особиста бухгалтерія" згідно з діаграмою прецедентів:

Ідентифікація

1)Аутентифікація

- Користувач може надавати свої дані для входу в систему, вказавши нікнейм або пошту, та пароль. Ця інформація зберігається в базі даних. Система автоматично пов’язує дані з його акаунтом та надає доступ до перегляду.

Управління фінансами

1) Додавання фінансової операції:

- Користувач може додати нову витрату до системи, вказавши категорію (наприклад, їжа, транспорт), дату, суму та додаткові коментарі. Ця інформація зберігається в базі даних для подальшого аналізу.
- Користувач може додати новий дохід, вказавши джерело (зарплата, бонуси) та суму. Система автоматично оновлює загальний баланс користувача.

2) Управління періодичними операціями:

- Користувач може створити шаблон для регулярних витрат (наприклад, оренда, комунальні платежі) або доходів (зарплата). Система автоматично додає ці суми до відповідних категорій у визначений період.

Використання даних

1) Введення сканованих чеків:

- Користувач може завантажити зображення чеків, пов’язаних з витратами. Система може зберігати ці зображення і асоціювати їх з відповідними транзакціями, що спростить управління документами.

2) Генерація фінансової статистики:

- Користувач може переглядати фінансові звіти та статистику, включаючи графіки витрат та доходів. Це дозволяє отримати візуальне представлення фінансового стану.

3) Експорт/імпорт даних в Excel:

- Користувач може експортувати свої фінансові дані у форматі Excel для

подальшого аналізу або обробки. Це може включати звіти за певний період або всю історію фінансових операцій.

Управління власним гаманцем

1) Управління рахунками:

- Користувач може додавати, редагувати або видаляти рахунки (дебетові, кредитні). Система повинна відображати баланс кожного рахунку та загальний баланс користувача.

2) Управління фондами:

- Користувач може створювати та управляти фондами (наприклад, фонд для ремонту, автомобіля, відпустки). Система дозволяє користувачу визначати мету фонду, вносити кошти та слідкувати за прогресом накопичення. Користувач може також переглядати, скільки грошей залишилося в фонді, і скільки ще потрібно зібрати для досягнення мети.

3) Керування кредитами/вкладом:

- Користувач може вводити інформацію про свої кредити та вклади, вказуючи умови (відсотки, терміни). Система може надавати звіти про стан кредитів та можливість порівняння з доходами.

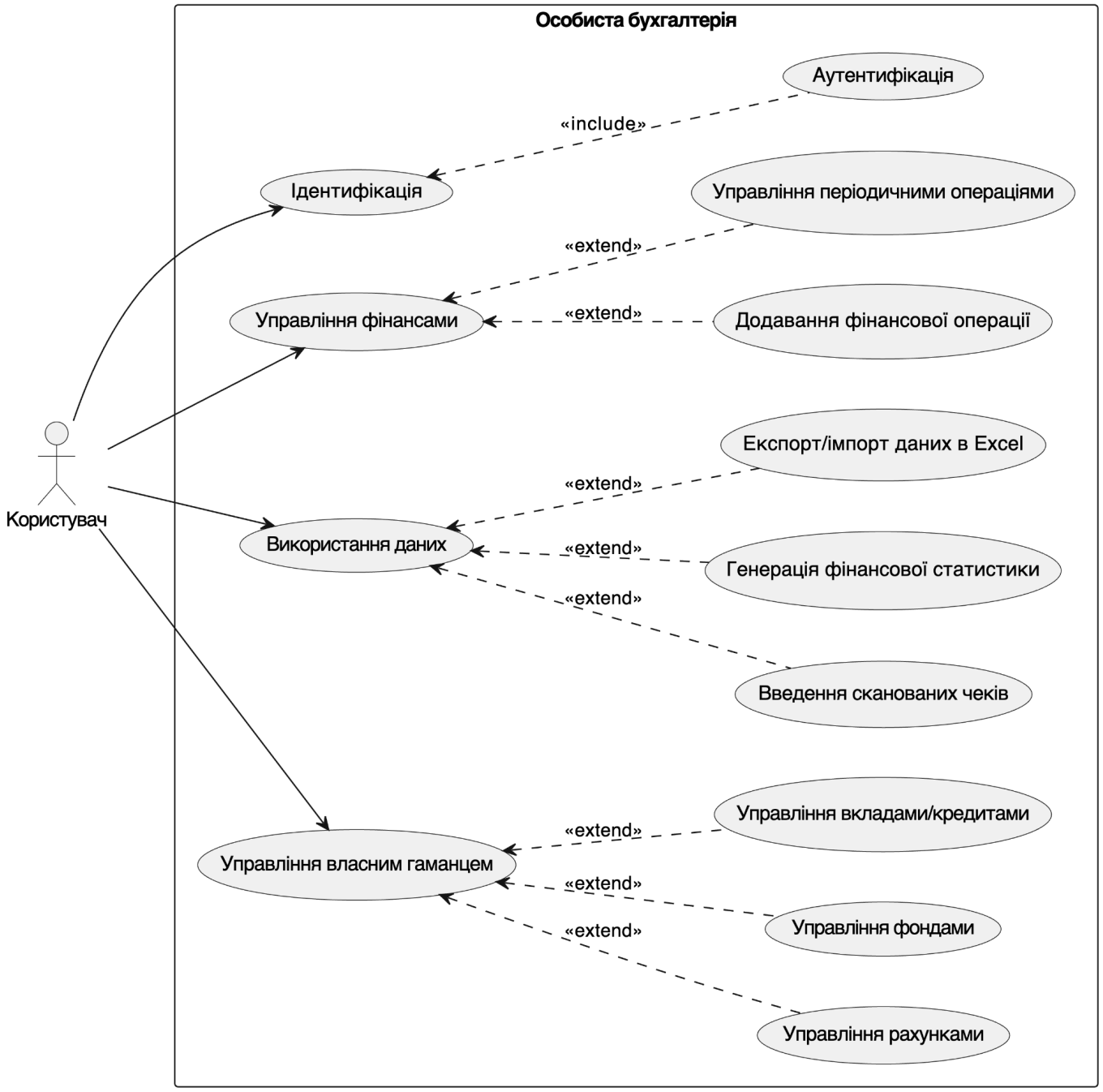

Рис. 1 - Діаграма прецедентів

Крок 3. Опис 3 прецедентів.

1. Прецедент: Додати витрату

Назва: Add Expense

Актори: User

Опис: Користувач може додати нову витрату до системи, заповнивши форму з необхідними даними.

Передумови: Користувач повинен бути автентифікований у системі.

Основний потік:

1. Користувач вибирає опцію "Додати витрату".
2. Система відображає форму для введення інформації про витрату.
3. Користувач заповнює поля: категорія, сума, дата, коментар.
4. Користувач натискає кнопку "Зберегти".
5. Система зберігає витрату у базі даних.
6. Система підтверджує успішне додавання витрати.

Виключення:

4-а. Якщо сума не є числом, система відображає повідомлення про помилку і запитує ввести правильну суму.

2. Прецедент: Переглянути статистику

Назва: View Statistics

Актори: User

Опис: Користувач може переглядати фінансову статистику за обраний період.

Передумови: Користувач повинен бути автентифікований у системі.

Основний потік:

1. Користувач вибирає опцію "Переглянути статистику".
2. Система відображає календар для вибору періоду.
3. Користувач вибирає початкову та кінцеву дати.
4. Користувач натискає кнопку "Показати статистику".
5. Система обробляє запит і формує звіт.
6. Система відображає статистику у формі графіків і таблиць.

Виключення:

3-а. Якщо користувач вибирає неправильні дати (кінцева дата менша за початкову), система відображає повідомлення про помилку.

3. Прецедент: Управління фондами

Назва: Manage Funds

Актори: User

Опис: Користувач може створювати, редагувати або видаляти фонди для накопичення коштів на конкретні цілі.

Передумови: Користувач повинен бути автентифікований у системі.

Основний потік:

1. Користувач вибирає опцію "Управління фондами".
2. Система відображає список існуючих фондів.
3. Користувач може вибрати один із наступних варіантів:

3-a. Додати новий фонд.

4-a. Система відображає форму для введення деталей фонду (назва, мета, сума).

3-b. Редагувати існуючий фонд.

3-c. Видалити фонд.

4. Користувач заповнює поля і натискає "Зберегти".
5. Система зберігає новий фонд у базі даних і підтверджує успішне додавання.

Виключення:

5-a. Якщо одне з обов’язкових полів порожнє, система відображає повідомлення про помилку і запитує ввести дані.

Крок 4. Створення діаграми класів для частини майбутньої системи.

Опис класів:

User - клас який представляє користувача, містить:

- Унікальний ідентифікатор користувача.
- Ім'я користувача для авторизації.
- Пароль користувача для доступу.
- Список рахунків, які належать користувачу.

Account - клас який представляє рахунки користувача, містить:

- Унікальний ідентифікатор рахунка.

- Баланс рахунка, що показує доступні кошти.
- Назва рахунка для ідентифікації.
- Користувач, якому належить рахунок.

Transaction - клас який представляє транзації користувача, містить:

- Унікальний ідентифікатор транзакції.
- Сума транзакції.
- Опис транзакції для пояснення мети.
- Рахунок, з яким пов'язана транзакція.
- Позначення, чи є транзакція періодичною.

Loan - клас який представляє кредити користувача, містить:

- Унікальний ідентифікатор кредиту.
- Сума кредиту, що була отримана.
- Відсоткова ставка за кредитом.
- Рахунок, з яким пов'язаний кредит.

Fund- клас який представляє фонди користувача, містить:

- Унікальний ідентифікатор фонду.
- Назва фонду для його ідентифікації.
- Баланс фонду, що показує загальну суму коштів.
- Список рахунків, пов'язаних з фондом.

Receipt- клас який представляє завантажені чеки користувача, містить:

- Унікальний ідентифікатор чека.
- Шлях до файлу з чеком.
- Транзакція, з якою пов'язаний чек.

Statistics- клас який представляє статистику на основі рахунків користувача, містить:

- Загальний дохід, що обчислюється на основі транзакцій.
- Загальні витрати, що обчислюються на основі транзакцій.
- Список транзакцій, з якими пов'язана статистика.

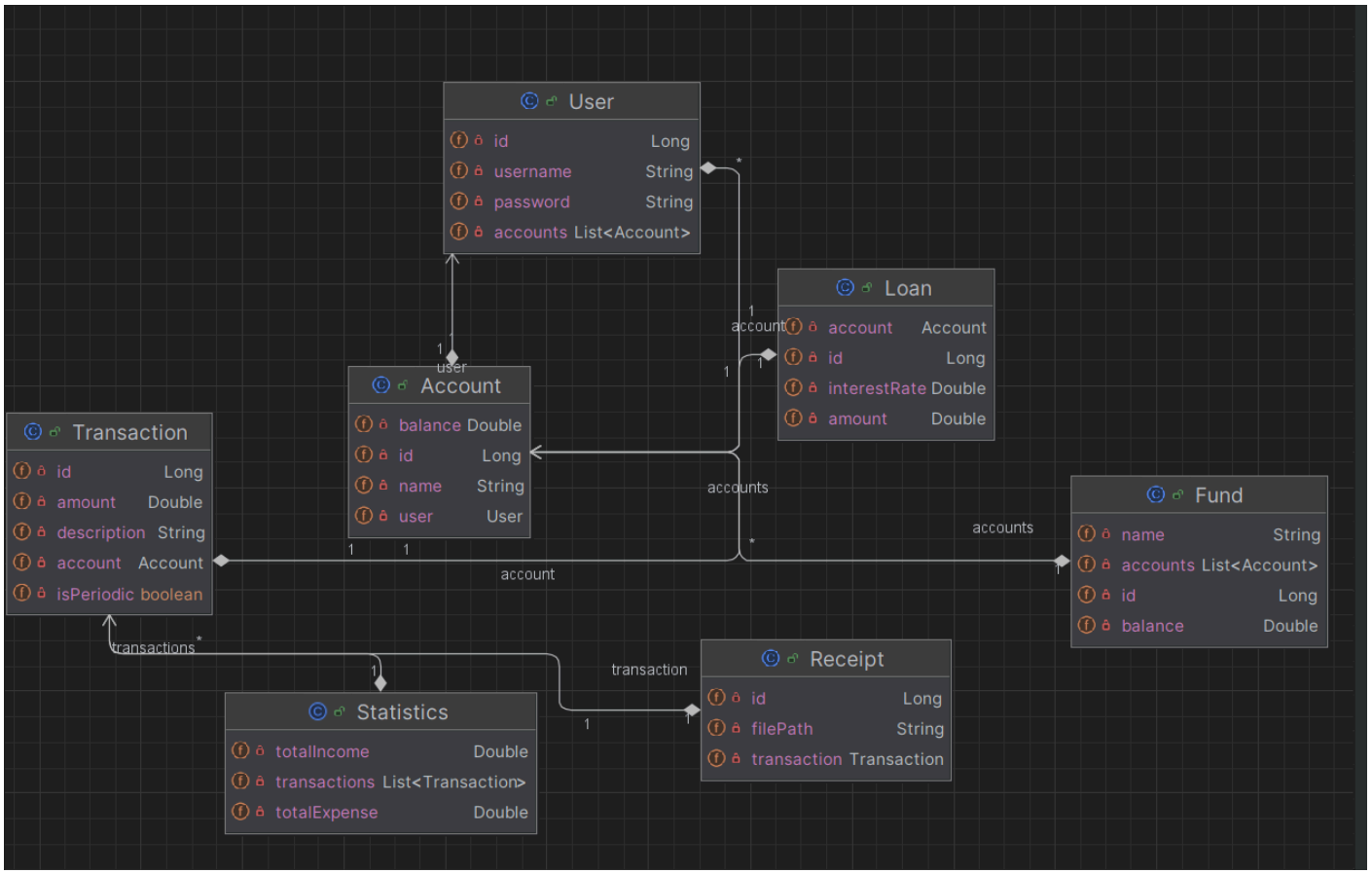

Рис. 2 - Діаграма класів

Код реалізації частини системи можна знайти на гітхабі за посиланням (https://github.com/papiroskada/TRPZ_labs_sidenko_ia-24/tree/lab2/budget-mate)

Крок 5. Представлення схеми таблиць бази даних.

Опис зв'язків між таблицями.

Users до Statistics: Один до Багатьох (Один користувач може мати багато даних статистики)

Users до Accounts: Один до Багатьох (Один користувач може мати багато рахунків)

Accounts до Transactions: Один до Багатьох (Один рахунок може мати багато транзакцій)

Accounts до Funds: Один до Багатьох (Один рахунок може мати багато фондів)

Accounts до Loan: Один до Багатьох (Один рахунок може мати багато кредитів)

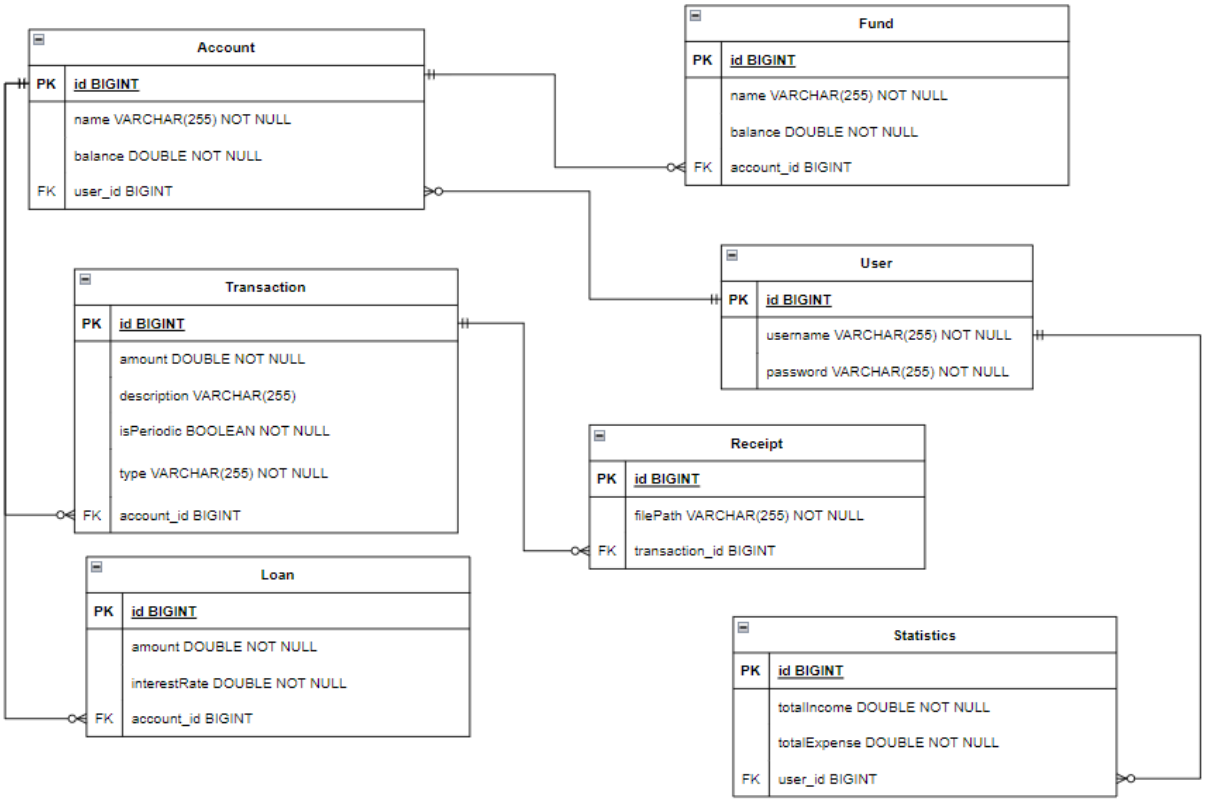

Рис. 3 - Схема таблиць бази даних

Висновок: у даній лабораторній роботі я проаналізувала тему та на основі даних створила діаграму прецедентів та діаграму класів. Також під час виконання даної лабораторної роботи я спроектувала схему бази даних та описала зв’язки між таблицями.