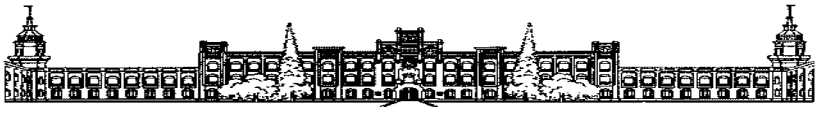


Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4

Технології розроблення програмного забезпечення

ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Особиста бугалтерія

Виконав
студент групи ІА – 21:
Ліщинський Б. В.

Перевірив:
Амонс О. А.

Київ 2024

Тема: шаблони «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Мета: Реалізувати частину функціоналу робочої програми та застосувати один з розглянутих шаблонів при реалізації програми.

Хід роботи

Згідно з варіантом завдання та діаграмою класів, побудованою у ході виконання лабораторної роботи №2(рис. 1), при реалізації рівня сервісів програми було використано шаблон проектування “State”.

Патерн State (Стан) належить до поведінкових патернів і використовується для управління станами об'єктів у програмі. Він дає змогу змінювати поведінку об'єкта в залежності від його поточного стану без використання складних умовних конструкцій (if-else, switch-case). Основна ідея полягає у виділенні кожного стану об’єкта в окремий клас і делегуванні йому відповідної поведінки.

Основні елементи патерну

1. Контекст — клас, який має внутрішній стан і виконує конкретні дії, делегуючи їх відповідному стану. Контекст може мати змінну, що вказує на поточний стан, і метод для зміни станів.
2. Інтерфейс стану — визначає загальні методи, які реалізують конкретні стани. Наприклад, якщо об'єкт змінює поведінку залежно від поточного стану, то інтерфейс визначає загальні методи для кожного з них.
3. Конкретні стани — окремі класи, що реалізують специфічну поведінку для кожного стану. Кожен клас стану реалізує методи інтерфейсу, змінюючи поведінку залежно від стану. Такі класи можуть змінювати стан Контексту, якщо виникає потреба в переході до іншого стану.

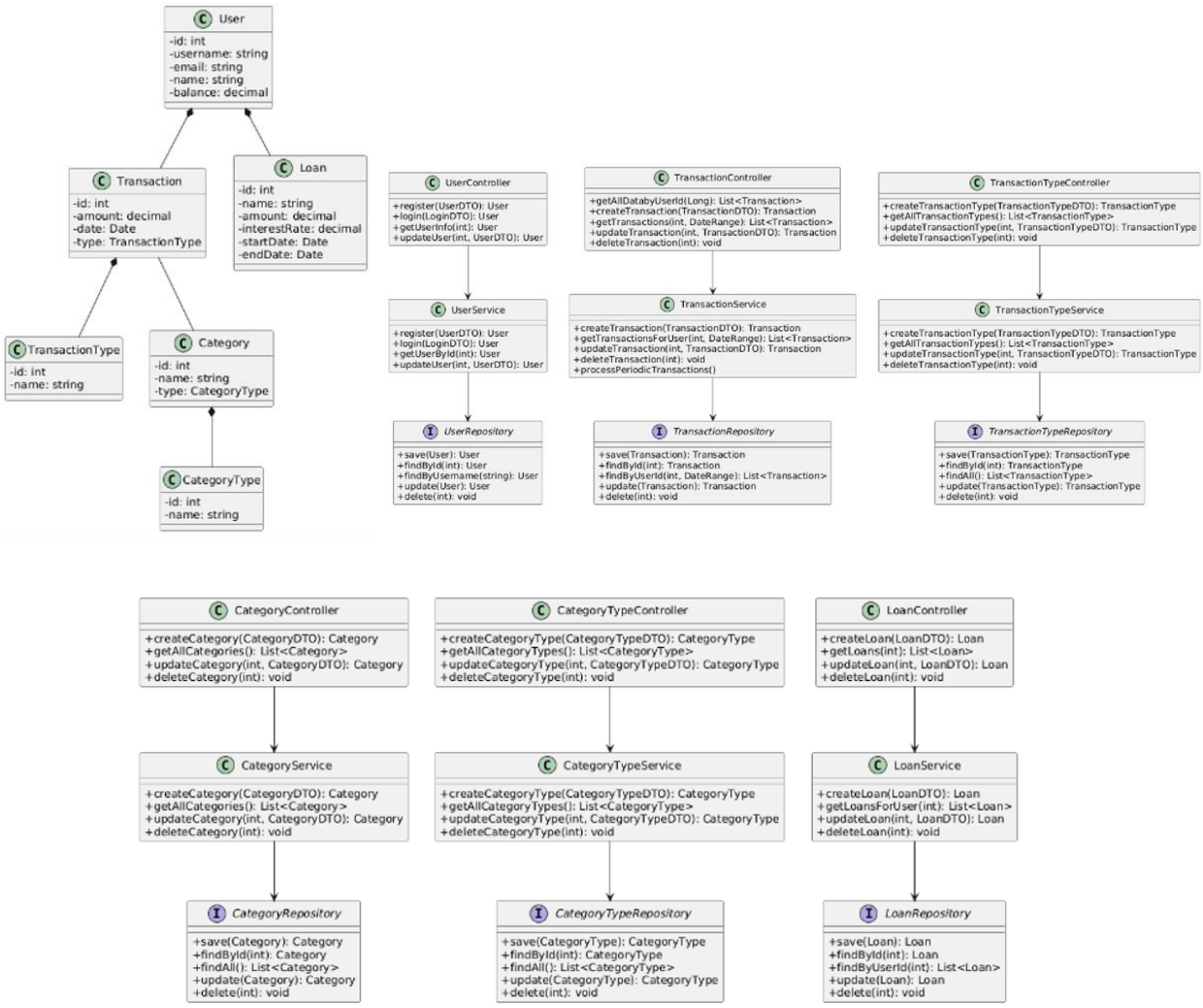


Рисунок 1 – Діаграма класів з ЛР №2

У ході виконання лабораторної роботи ця частина програми була розроблена(рис. 2).

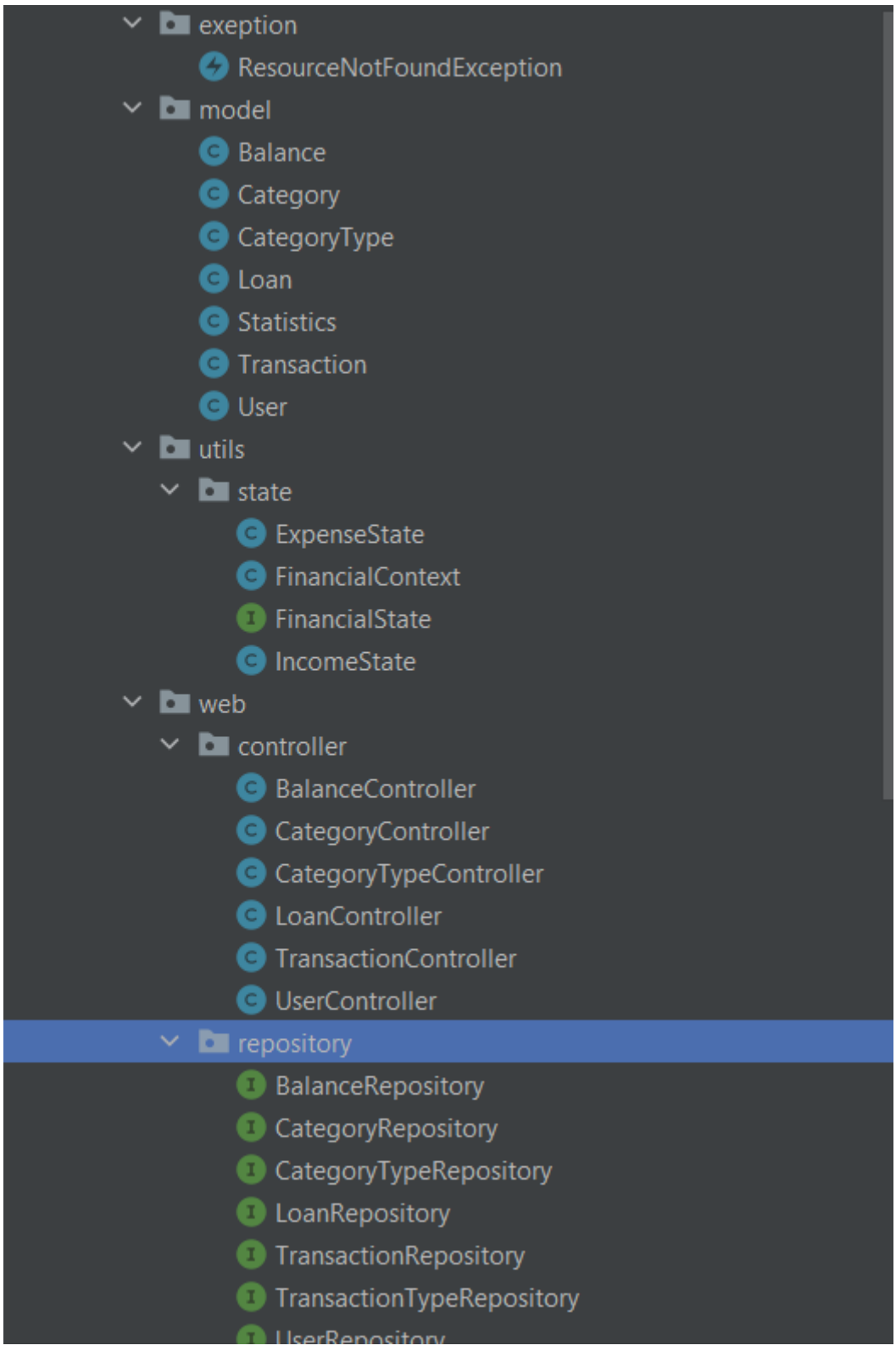


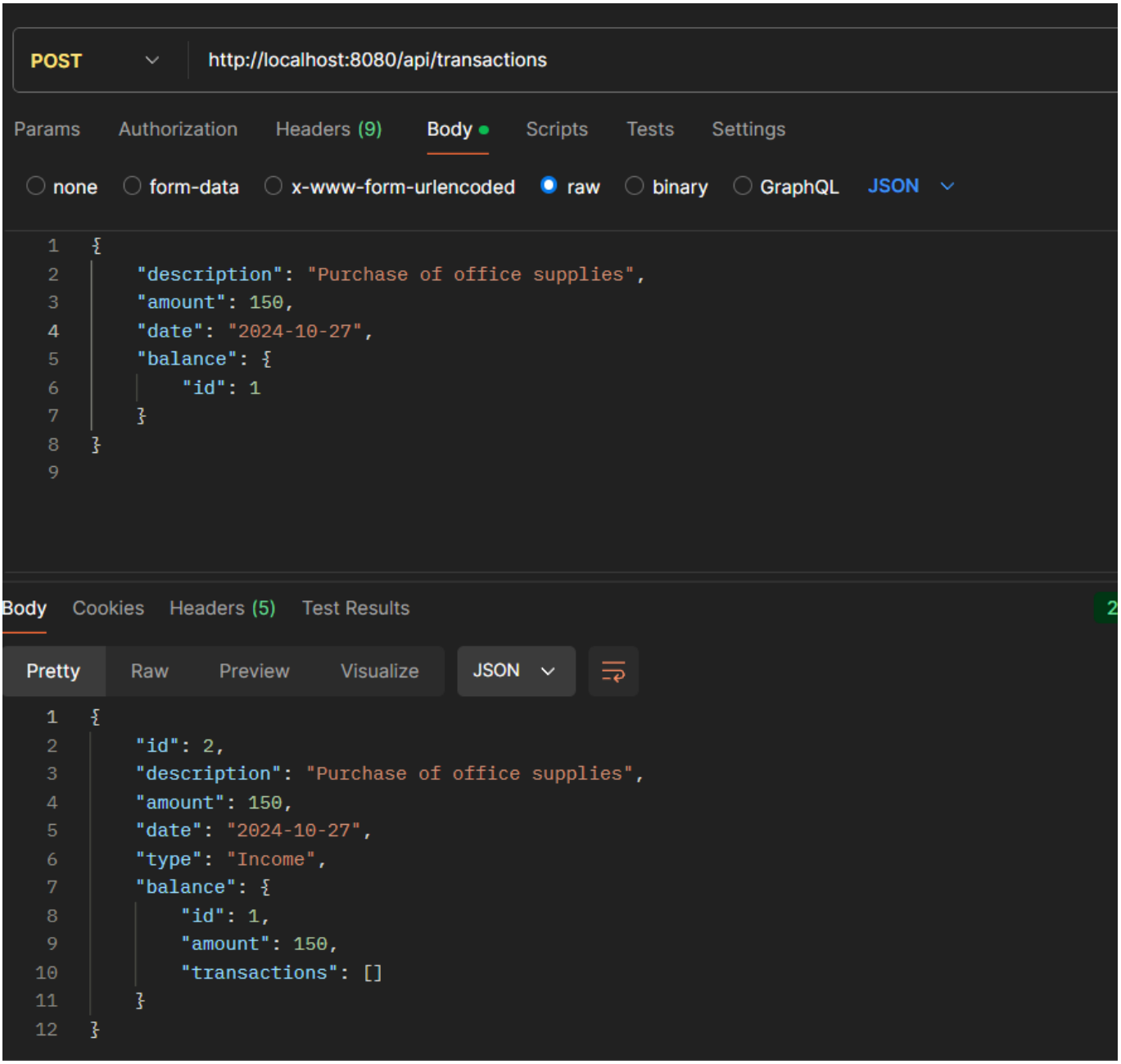
Рисунок 2 – Дерево програмного коду проекту

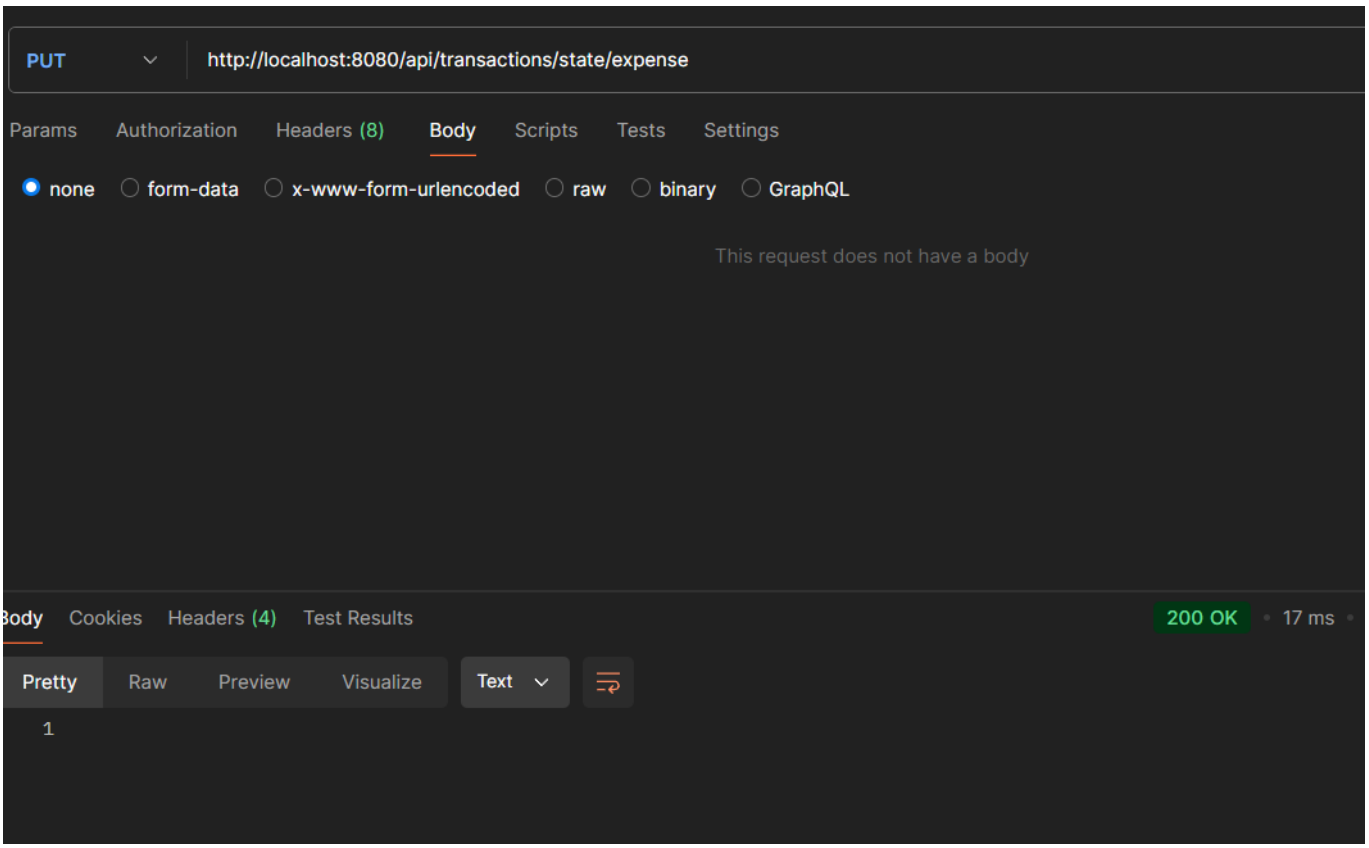
Приклад коду:

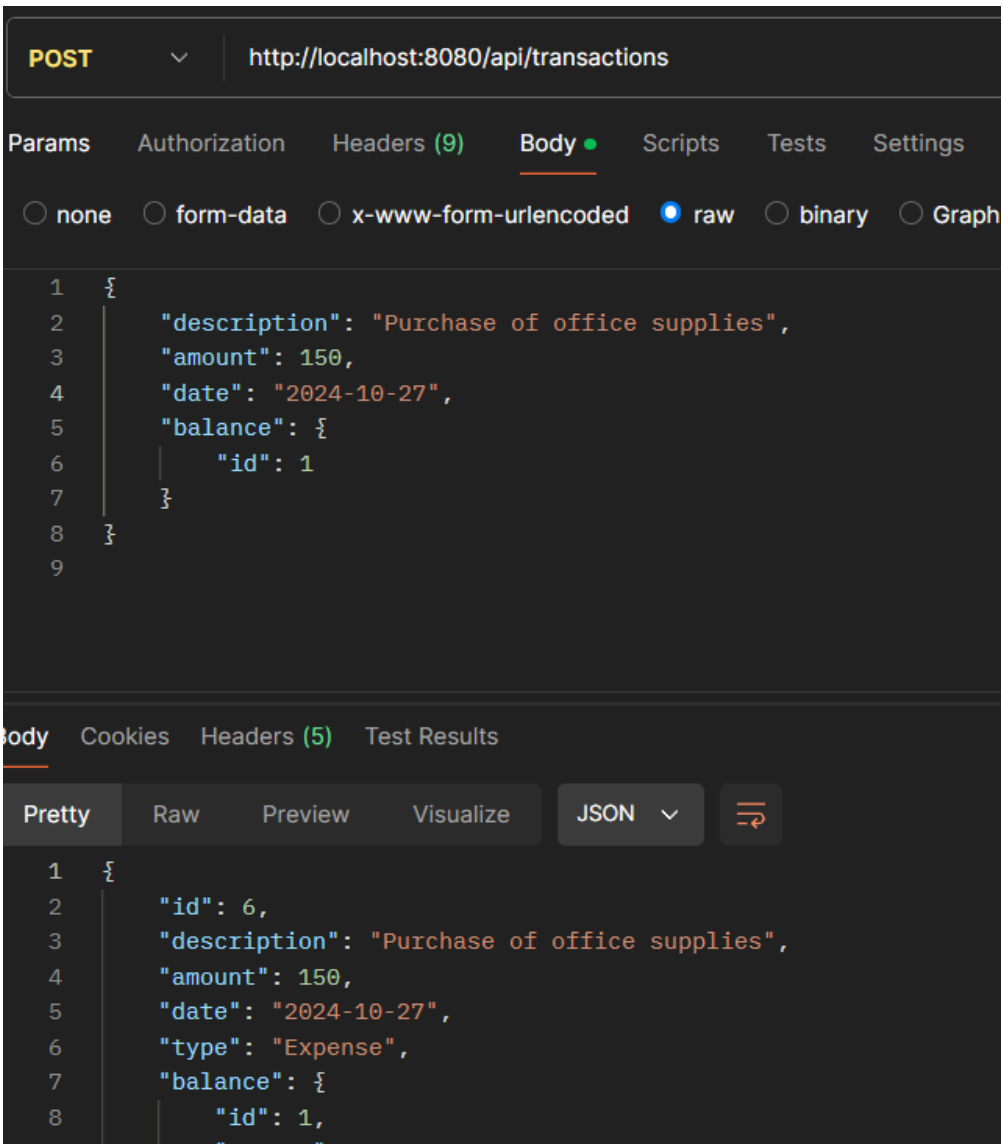
```
@Getter
@Setter
@Component
public class FinancialContext {
```

```
    private FinancialState currentState;

   public FinancialContext(TransactionService transactionService, BalanceService
balanceService) {
    this.currentState = new IncomeState(transactionService, balanceService);
    }

    public Transaction addTransaction(Transaction transaction) {
        return currentState.addTransaction(transaction);
    }

    public void exportToExcelByState() {
        currentState.exportToExcelByState();
    }

    public void importFromExcelByState() {
        currentState.importFromExcelByState();
    }

    public void setCurrentState(FinancialState newState) {
        this.currentState = newState;
    }
}
public interface FinancialState {
    Transaction addTransaction(Transaction transaction);

    void exportToExcelByState();

    void importFromExcelByState();
}

@AllArgsConstructor
public class ExpenseState implements FinancialState {
    private TransactionService transactionService;
    private BalanceService balanceService;

    @Override
    public Transaction addTransaction(Transaction transaction) {
        transaction.setType("Expense");
        Balance updatedBalance = transaction.getBalance();
        updatedBalance.setAmount(updatedBalance.getAmount() -
transaction.getAmount());
        balanceService.updateBalance(updatedBalance.getId(), updatedBalance);

        return transactionService.addTransaction(transaction);
    }
...

@PutMapping("/state/{newState}")
public ResponseEntity<Void> changeState(@PathVariable String newState) {
    FinancialState state = stateMap.get(newState.toLowerCase());
    if (state == null) {
        return ResponseEntity.badRequest().build();
    }
    financialContext.setCurrentState(state);
    return ResponseEntity.ok().build();
}
@PostMapping
public ResponseEntity<Transaction> addTransaction(@RequestBody Transaction
transaction) {
    Transaction createdTransaction = financialContext.addTransaction(transaction);
    return new ResponseEntity<>(createdTransaction, HttpStatus.CREATED);
}
```

Приклад використання:




PUT
http://localhost:8080/api/transactions/state/expense
Params
Authorization
Headers (8)
Body
Scripts
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body
Body
Cookies
Headers (4)
Test Results
200 OK
17 ms
Pretty
Raw
Preview
Visualize
Text
1



Висновки: я реалізував частину функціональності системи, застосувавши шаблон проектування.