****

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 №9  
РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ:

CLIENT-SERVER, PEER-TO-PEER,

SERVICE-ORIENTED ARCHITECTURE

Виконав Перевірив:

студент групи ІА – 13: Мягкий М. Ю

Тарасовець Максим

**Завдання:**

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів і їх взаємодій для

досягнення конкретних функціональних можливостей.

3. Реалізувати взаємодію програми в одній з архітектур відповідно до обраної теми.

**Хід роботи**

Архітектура SOA (Service-Oriented Architecture або Архітектура, орієнтована на сервіси) - це підхід до розробки програмного забезпечення, де функціональність поділяється на незалежні компоненти, які називаються сервісами. Ці сервіси можуть виконувати конкретні завдання і можуть взаємодіяти один з одним через стандартизовані інтерфейси, зазвичай на основі веб-сервісів.

Архітектура складається з сервісів для роботи з рахунками, транзакціями, статистикою та операціями з депозитами та кредитами.

app.py: Основний скрипт, відповідальний за запуск та управління кількома сервісами за допомогою multiprocessing. Визначає клас ServiceState для управління станом (запущений чи зупинений) кожного сервісу.

**transactions\_service.py**: Реалізує сервіс для обробки операцій, пов'язаних із транзакціями, таких як додавання транзакцій, обробка регулярних транзакцій та отримання даних про транзакції.

**statistic\_service.py**: Реалізує сервіс для обробки операцій, пов'язаних із статистикою, таких як отримання різних статистичних показників (загальні витрати, загальний дохід і т. д.) та експорту статистики в Excel.

**deposit\_loan\_service.py**: Реалізує сервіс для обробки операцій, пов'язаних із депозитами та кредитами, таких як додавання депозитів та кредитів, розрахунок відсотків, оновлення балансу рахунку та відображення балансів протягом часу.

**accounts\_handler.py**: Визначає клас (AccountsHandler) для обробки рахунків. Включає методи для додавання рахунків та отримання інформації про рахунок. Цей клас може використовувати різні реалізації, і в цьому випадку він використовує DatabaseHandler для взаємодії з базою даних SQLite.