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

із дисципліни *«Технології розроблення програмного забезпечення»*

Тема: «ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ. ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ.»

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. Розробити діаграму розгортання для проектованої системи.
3. Розробити діаграму компонентів для проектованої системи.
4. Розробити діаграму послідовностей для проектованої системи.
5. Скласти звіт про виконану роботу

Зміст

Хід роботи:

Крок 1: Короткі теоретичні відомості

У даній лабораторній роботі я маю на меті створити такі діаграми: діаграма розгортання, діаграма компонентів, діаграма взаємодій та послідовностей. Вони використовуються для візуалізації різних аспектів системи.

Діаграма розгортання показує фізичне розташування системи, показує, на якому фізичному обладнанні працює той чи інший компонент програмне забезпечення.

Вона відображає, як програмні компоненти, вузли (сервери, клієнтські машини тощо) і ресурси взаємодіють у реальному середовищі. Важливими елементами є вузли (сервери, пристрої), артефакти (програмні файли), а також з'єднання між ними. Вузли представляють фізичні пристрої або середовища, де розміщуються компоненти системи. Артефакти - це програмні одиниці, які розгортаються на вузлах. Зв’язки між вузлами для передачі даних представленні у вигляді ліній з’єднань.

Діаграма компонентів представляє програмні компоненти системи та їхні залежності. Вона фокусується на логічній структурі системи, а не на фізичному середовищі. Даного типу діаграм існує три види: логічна, фізична, виконавча. Під час її побудови використовують такі основні елементи: компоненти (логічні частини), інтерфейси(точки взаємодії), залежності (зв’язки)

Діаграма взаємодій демонструє, як об'єкти або компоненти системи взаємодіють між собою під час виконання певного процесу. Вона може відображати потік повідомлень між об'єктами в певному контексті.

На діаграмі взаємодій відображається логіка та послідовність переходів від однієї діяльності до іншої, а увага аналітика фокусується на результатах. Результат діяльності може призвести до зміни стану системи чи поверненню деякого значення.

Основні компоненти діаграми взаємодії можуть бути різними в залежності від конкретного типу. Але до загальних можна віднести: об’єкти або учасники, лінії життя, повідомлення, активність, потоки управління, обмеження, повернення.

Одним із різновиду діаграми взаємодії є *діаграма послідовності.* Вона показує, як об'єкти системи взаємодіють у певному порядку, відображаючи послідовність повідомлень у певному процесі.

Крок 2: Діаграма розгортання

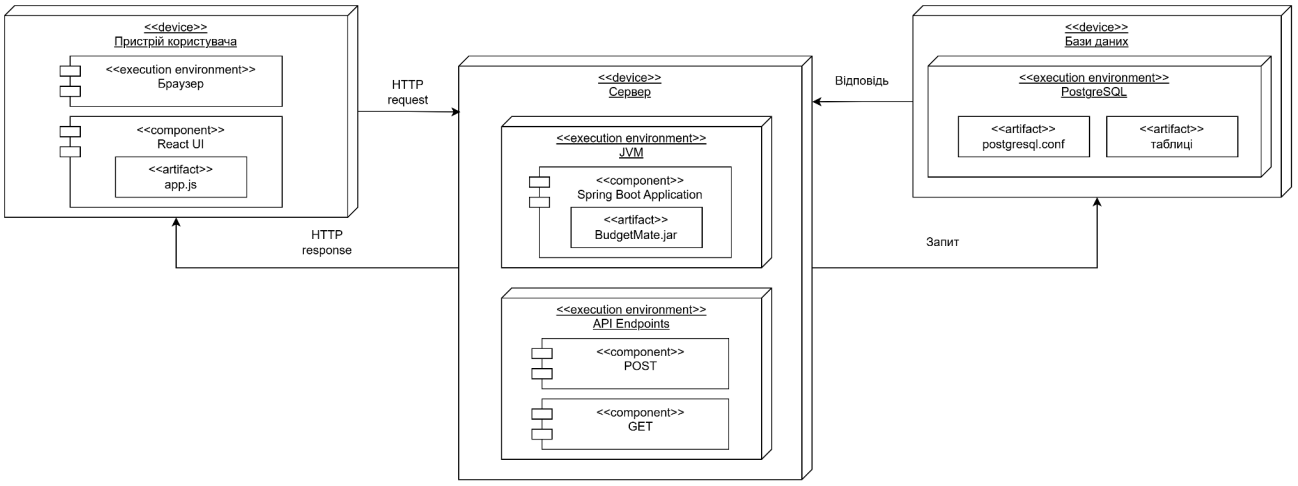


Рис. 1 - Діаграма розгортання

Діаграма розгортання описує архітектуру системи, яка складається з трьох основних компонентів: пристрою користувача, сервера та бази даних. Ось детальний опис кожного компонента:

Пристрій користувача:

Містить середовище виконання <<execution environment>> "Браузер", де виконується клієнтська частина інтерфейсу.

У цьому середовищі розміщений компонент <<component>> "React UI" з артефактом <<artifact>> "app.js", який представляє собою основний файл фронтенду, розроблений за допомогою бібліотеки React.

Сервер:

Містить два середовища виконання:

<<execution environment>> "JVM" (Java Virtual Machine), в якому розміщено компонент <<component>> "Spring Boot Application". Цей компонент містить артефакт <<artifact>> "BudgetMate.jar", який є згенерованим файлом програми, створеним за допомогою Spring Boot.

<<execution environment>> "API Endpoints" – тут розміщено два компоненти <<component>> "POST" та "GET", які представляють різні кінцеві точки API для взаємодії з клієнтським інтерфейсом.

База даних:

Містить середовище виконання <<execution environment>> "PostgreSQL".

У цьому середовищі знаходяться два артефакти:

<<artifact>> "postgresql.conf", що є конфігураційним файлом PostgreSQL.

<<artifact>> "таблиці", які представляють структуру даних у базі (таблиці) для зберігання інформації, з якою працює додаток.

Крок 3: Діаграма компонентів

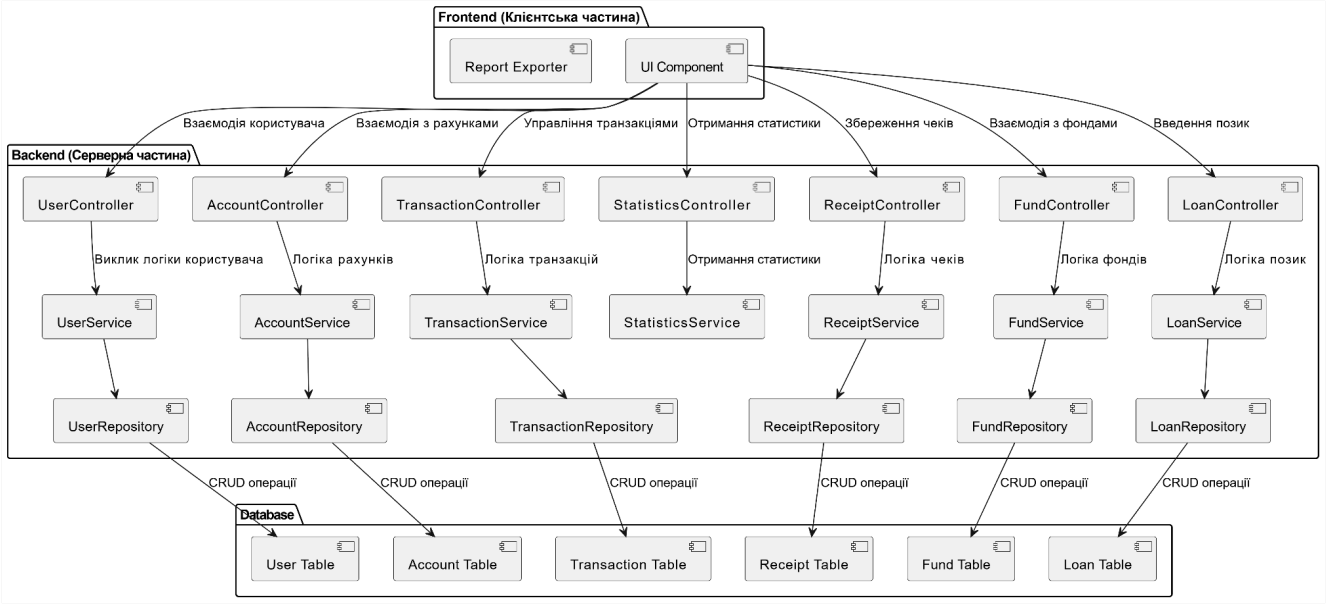


Рис.2 - Діаграма компонентів

Frontend (Клієнтська частина):

UI Component: Графічний інтерфейс користувача для взаємодії з системою, який дозволяє користувачу переглядати, вводити та редагувати інформацію про витрати, прибутки, рахунки, фонди тощо.

Report Exporter: Компонент для експорту та імпорту звітів у різні формати (наприклад, Excel), а також для відображення графіків і звітів за витратами та прибутками.

Backend (Серверна частина):

Сервіси:

UserService: Сервіс для керування користувачами, забезпечує реєстрацію, автентифікацію та роботу з особистими даними користувачів.

AccountService: Сервіс для управління рахунками користувача, включаючи оновлення балансу і підтримку різних типів рахунків.

TransactionService: Сервіс для обробки транзакцій (витрат і прибутків). Підтримує періодичні транзакції, такі як заробітна плата чи орендна плата.

StatisticsService: Сервіс для розрахунку статистики витрат і доходів користувача. Використовується для побудови звітів та графіків.

ReceiptService: Сервіс для збереження та обробки сканованих чеків, зв'язує чеки з відповідними транзакціями.

FundService: Сервіс для управління фондами, які можуть бути спільними для всієї сім'ї та підтримувати різні цілі (наприклад, ремонт, автомобіль, відпустка).

LoanService: Сервіс для управління позиками та кредитами, включаючи розрахунок і перевірку нарахованих відсотків.

Контролери:

UserController: Контролер для управління запитами, пов’язаними з користувачами.

AccountController: Контролер для обробки запитів, пов’язаних з рахунками.

TransactionController: Контролер для роботи з транзакціями, включаючи витрати та прибутки.

StatisticsController: Контролер для запитів, що стосуються побудови статистики.

ReceiptController: Контролер для керування запитами, пов'язаними з чеками.

FundController: Контролер для обробки запитів, пов'язаних з фондами.

LoanController: Контролер для обробки запитів, пов'язаних з позиками.

Репозиторії:

UserRepository: Репозиторій для зберігання та доступу до даних користувачів.

AccountRepository: Репозиторій для взаємодії з таблицею рахунків.

TransactionRepository: Репозиторій для доступу до даних про транзакції.

StatisticsRepository: Репозиторій для взаємодії з даними про статистику.

ReceiptRepository: Репозиторій для зберігання даних про чеки.

FundRepository: Репозиторій для управління даними про фонди.

LoanRepository: Репозиторій для роботи з даними про позики.

Database (База даних):

User Table: База даних для зберігання інформації про користувачів.

Account Table: База даних для зберігання інформації про рахунки.

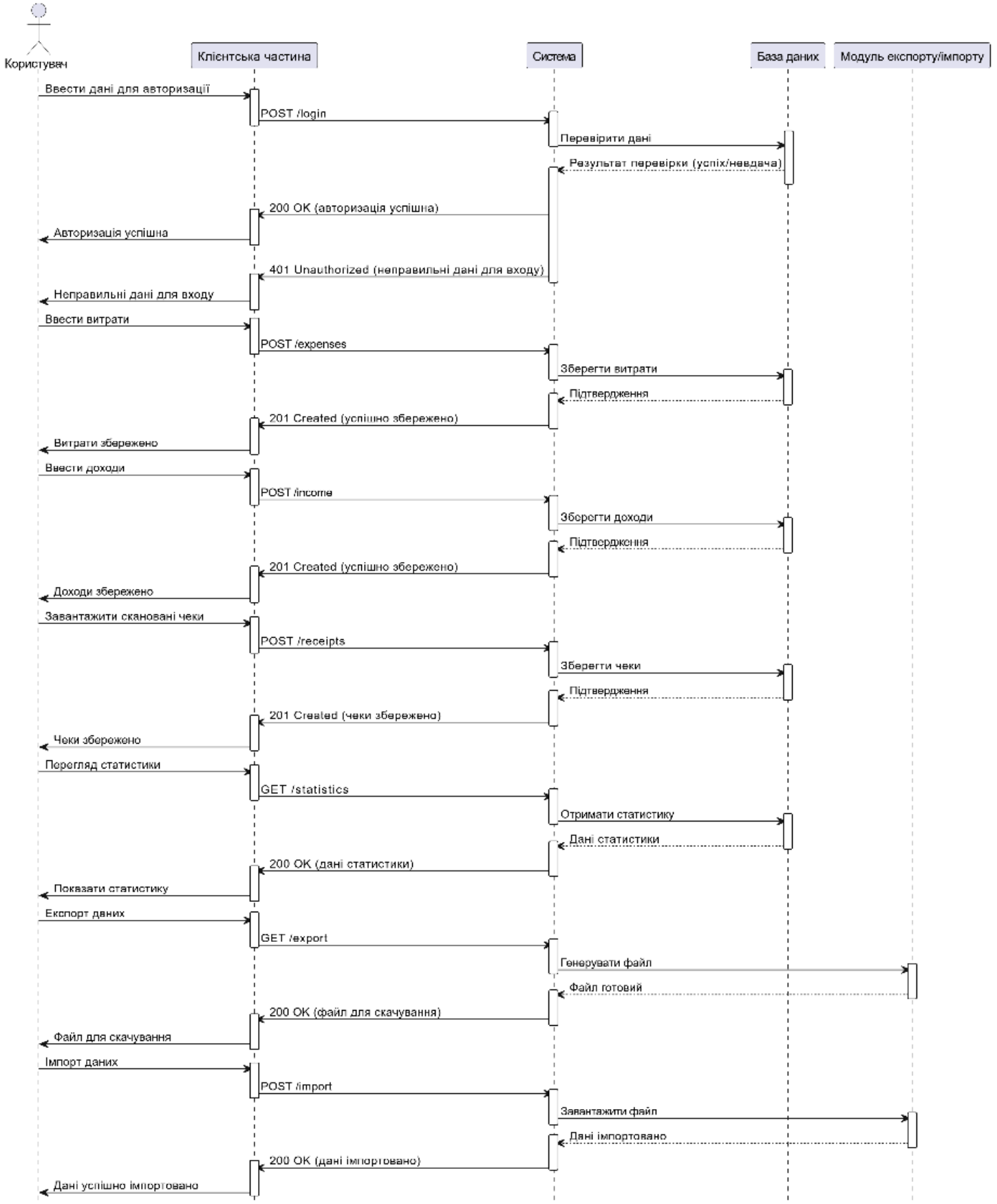
Transaction Table: База даних для зберігання інформації про транзакції.

Receipt Table: База даних для зберігання сканованих чеків.

Fund Table: База даних для зберігання інформації про фонди.

Loan Table: База даних для зберігання інформації про позики.

Крок 4: Діаграма послідовності



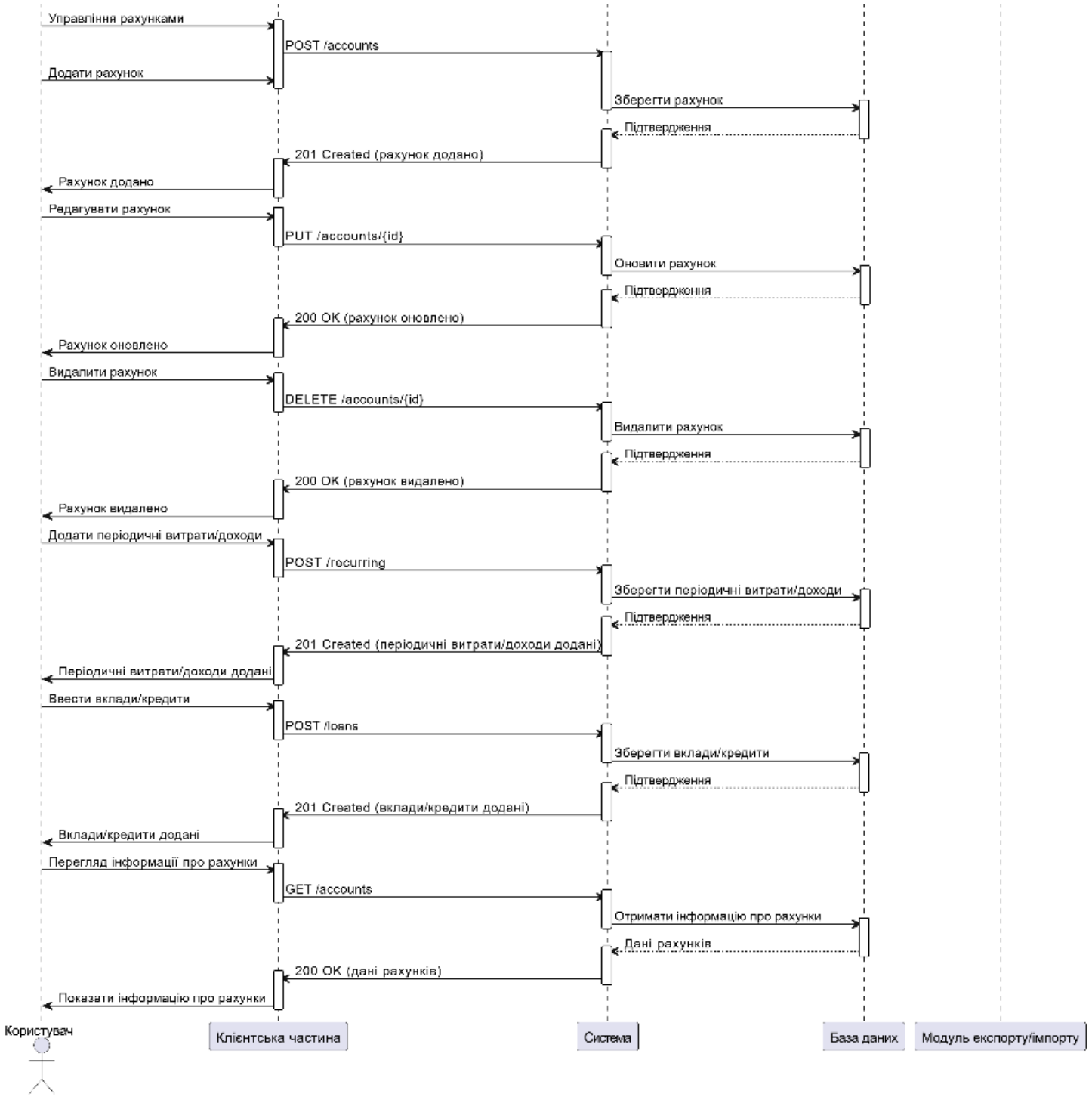


Рис.3 - Діаграма послідовностей

Опис діаграми послідовностей

Авторизація:

Користувач вводить свої дані для входу, які передаються до системи через POST-запит. Система перевіряє дані у базі даних і відповідає про успішність або невдачу авторизації.

Введення витрат і доходів:

Користувач може вводити витрати та доходи через POST-запити, які зберігаються в базі даних.

Завантаження сканованих чеків:

Користувач завантажує чеки, які зберігаються через POST-запит.

Перегляд статистики:

Користувач може переглядати статистику через GET-запит, який отримує дані з бази даних.

Експорт/імпорт даних:

Дані можуть бути експортовані через GET-запит або імпортовані через POST-запит.

Управління рахунками:

Користувач може додавати, редагувати та видаляти рахунки, використовуючи POST, PUT і DELETE запити відповідно.

Додавання періодичних витрат і вкладів/кредитів:

Користувач може додавати періодичні витрати та інформацію про вклади/кредити через POST-запити.

Перегляд інформації про рахунки:

Користувач може переглядати деталі своїх рахунків за допомогою GET-запиту.

Висновок: у рамках виконання лабораторної роботи щодо проектування системи "Особиста бухгалтерія" було здійснено комплексний підхід до розробки та моделювання її архітектури. На початковому етапі було проведено аналіз теоретичних основ, що стосуються проектування інформаційних систем, зокрема, важливості UML-діаграм для візуалізації та документування архітектури програмного забезпечення. Розроблені діаграми стали основою для подальшої реалізації проекту, що забезпечить ефективне управління особистими фінансами користувачів.