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
Технології розроблення програмного забезпечення
« Основи проектування розгортання»

Тема: Особиста бухгалтерія

Виконала:
студентка групи ІА-32
Коляда М. С.

Перевірила:
Мягкий М.Ю.

Київ 2025

Зміст

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.

Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу.

• Розробити діаграму компонентів для проєктованої системи.

• Розробити діаграму розгортання для проєктованої системи.

• Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.

• На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму розгортання з описом, діаграму компонентів системи з описом, діаграми послідовностей, а також вихідний код системи, який було додано в цій лабораторній роботі.

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграми розгортання представляють фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення. Головними елементами діаграми є вузли, пов'язані інформаційними шляхами. Вузол (node) – це те, що може містити програмне забезпечення. Вузли бувають двох типів. Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний із системою. Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер (наприклад, вебсервер). Між вузлами можуть стояти зв'язки, які зазвичай зображують у вигляді прямої лінії. Як і на інших діаграмах, у зв'язків можуть бути атрибути множинності (для показання, наприклад, підключення 2х і більше клієнтів до одного сервера) і назва. У назві, як правило, міститься спосіб зв'язку між двома 40 вузлами – це може бути назва протоколу (HTTP, IPC) або технологія, що використовується для забезпечення взаємодії вузлів (.NET Remoting, WCF). Вузли можуть містити артефакти (artifacts), які є фізичним уособленням програмного забезпечення; зазвичай це файли. Такими файлами можуть бути виконувані файли (такі як файли .exe, двійкові файли, файли DLL, файли JAR, збірки або сценарії) або файли даних, конфігураційні файли, HTML-документи тощо. Перелік артефактів усередині вузла вказує на те, що на даному вузлі артефакт розгортається в систему, що запускається. Артефакти можна зображати у вигляді прямокутників класів або перераховувати їхні імена всередині вузла. Якщо ви показуєте ці елементи у вигляді прямокутників класів, то можете додати значок документа або ключове слово «artifact». Можна супроводжувати вузли або артефакти значеннями у вигляді міток, щоб вказати різну цікаву інформацію про вузол, наприклад постачальника, операційну систему, місце розташування – загалом, усе, що спаде вам на

думку. Часто у вас буде безліч фізичних вузлів для розв'язання однієї й тієї самої логічної задачі. Можна відобразити цей факт, намалювавши безліч прямокутників вузлів або поставивши число у вигляді значення-мітки. Діаграми розгортань розрізняють двох видів: описові та екземплярні. На діаграмах описової форми вказуються вузли, артефакти і зв'язки між вузлами без вказівки конкретного обладнання або програмного забезпечення, необхідного для розгортання. Такий вид діаграм корисний на ранніх етапах розроблення для розуміння, які взагалі фізичні пристрої необхідні для функціонування системи або для опису процесу розгортання в загальному ключі. Діаграми екземплярної форми несуть у собі екземпляри обладнання, артефактів і зв'язків між ними. Під екземплярами розуміють конкретні елементи – ПК із відповідним набором характеристик і встановленим ПЗ; цілком може бути, у межах однієї організації це може бути якийсь конкретний вузол (наприклад, ПК тестувальника Василя). Діаграми екземплярної форми розробляють на завершальних стадіях розроблення ПЗ – коли вже відомі та сформульовані вимоги до програмного комплексу, обладнання закуплено і все готово до розгортання. Діаграми такої форми являють собою скоріше план розгортання в графічному вигляді, ніж модель розгортання.

Діаграма компонентів

Діаграма компонентів UML є представленням проєктованої системи, розбитої на окремі модулі. Залежно від способу поділу на модулі розрізняють три види діаграм компонентів: логічні; фізичні; виконувані. Коли використовують логічне розбиття на компоненти, то у такому разі проєктовану систему віртуально уявляють як набір самостійних, автономних модулів (компонентів), що взаємодіють між собою. Коли на діаграмі представляють фізичне розбиття, то в такому разі на діаграмі

компонентів показують компоненти та залежності між ними. Залежності показують, що класи в з одного компонента використовують класи з іншого компонента. Фізична модель використовується для розуміння які компоненти повинні бути зібрані в інсталяційний пакет. Також така діаграма показує зміни в якому компоненті будуть впливати на інші компоненти. У цьому випадку на діаграмі показани фізичні файли (.exe та .dll), показана залежність між ними, а також всі файли робиті на три блоки: компоненти для серверної частини, компоненти для клієнтської та компоненти, умовно названі middleware, які повинні бути як на серверній так і на клієнтській стороні, тому 43 що в них або є загальна бізнес-логіка, або є загальні інтерфейси методів та даних якими обмінюються між собою клієнт та сервер. Компоненти можуть поділятися за фізичними одиницями – окремі вузли розподіленої системи – набір комп'ютерів і серверів; на кожному з вузлів можуть бути встановлені різні виконувані компоненти. Такий вид діаграм компонентів застарів і зазвичай замість нього використовують діаграму розгортань. На діаграмах компонентів з виконуваним поділом компонентів кожен компонент являє собою деякий файл – виконувані файли (.exe), файли вихідних кодів, сторінки html, бази даних і таблиці тощо. У цьому разі діаграма схожа на діаграму класів, але на більш верхньому рівні – рівні виконуваних файлів або процесів. Такий підхід дає інший розріз представлення системи.

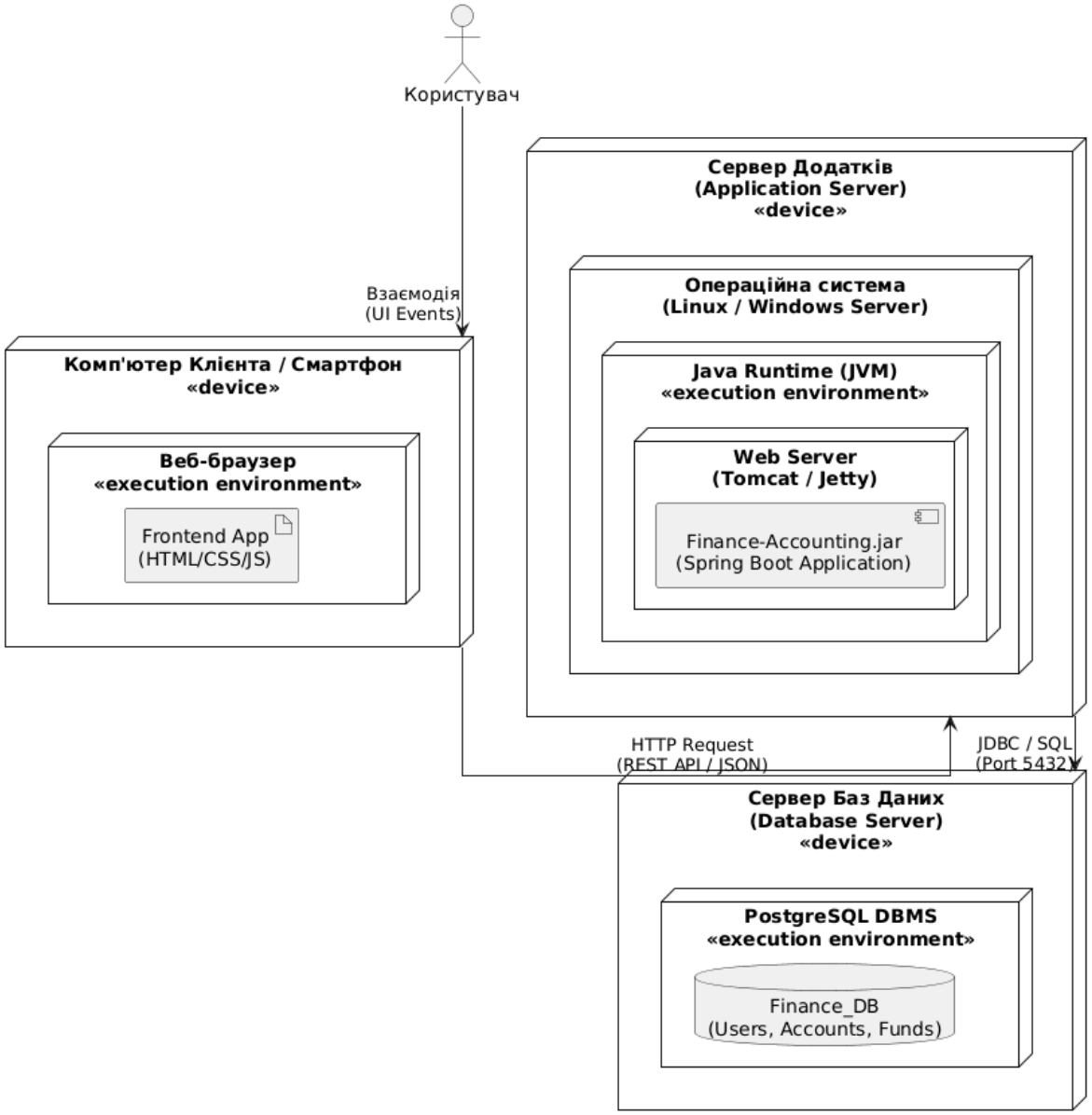
Діаграми послідовностей

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML (Unified Modeling Language), який використовується для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона відображає, як об'єкти обмінюються повідомленнями, показуючи порядок і логіку виконання операцій. Діаграма складається з

таких основних елементів: Актори (Actors): Зазвичай позначаються піктограмами або назвами. Це користувачі чи інші системи, які взаємодіють із системою. Актори можуть бути зовнішніми стосовно моделювання системи.Об'єкти або класи: Розміщуються горизонтально на діаграмі. Вони позначаються прямокутниками з іменем об'єкта або класу під прямокутником. Кожен об'єкт має «життєвий цикл», який представлений вертикальною пунктирною лінією (лінія життя). Повідомлення: Це лінії зі стрілками, які з'єднують об'єкти. Вони показують передачу повідомлень чи виклик методів. Стрілка може бути синхронною (звичайна стрілка) або асинхронною (лінія з відкритим трикутником) та з пунктирною лінією, що показує повернення результату. Активності: Вказують періоди, протягом яких об'єкт виконує певну дію. На діаграмі це позначається прямокутником, накладеним на лінію життя. Контрольні структури: Використовуються для відображення умов, циклів або альтернативних сценаріїв. Наприклад, блоки "alt" (альтернатива) або "loop" (цикл). Основні кроки створення діаграми послідовностей: визначити акторів і об'єкти, які беруть участь у сценарії; побудувати їхні лінії життя; розробити послідовність передачі повідомлень між об'єктами; додати умовні блоки або цикли за необхідності. Діаграми послідовностей є корисними для моделювання бізнес-процесів, проєктування архітектури систем і тестування. Вони дають змогу візуалізувати логіку взаємодії компонентів та виявити потенційні проблеми ще на етапі проєктування.

Хід роботи

Діаграма розгортання



Опис діаграми розгортання

Вузол Клієнта: Пристрій користувача (ПК або смартфон), на якому у середовищі веб-браузера виконується Frontend-частина (HTML/CSS/JS).

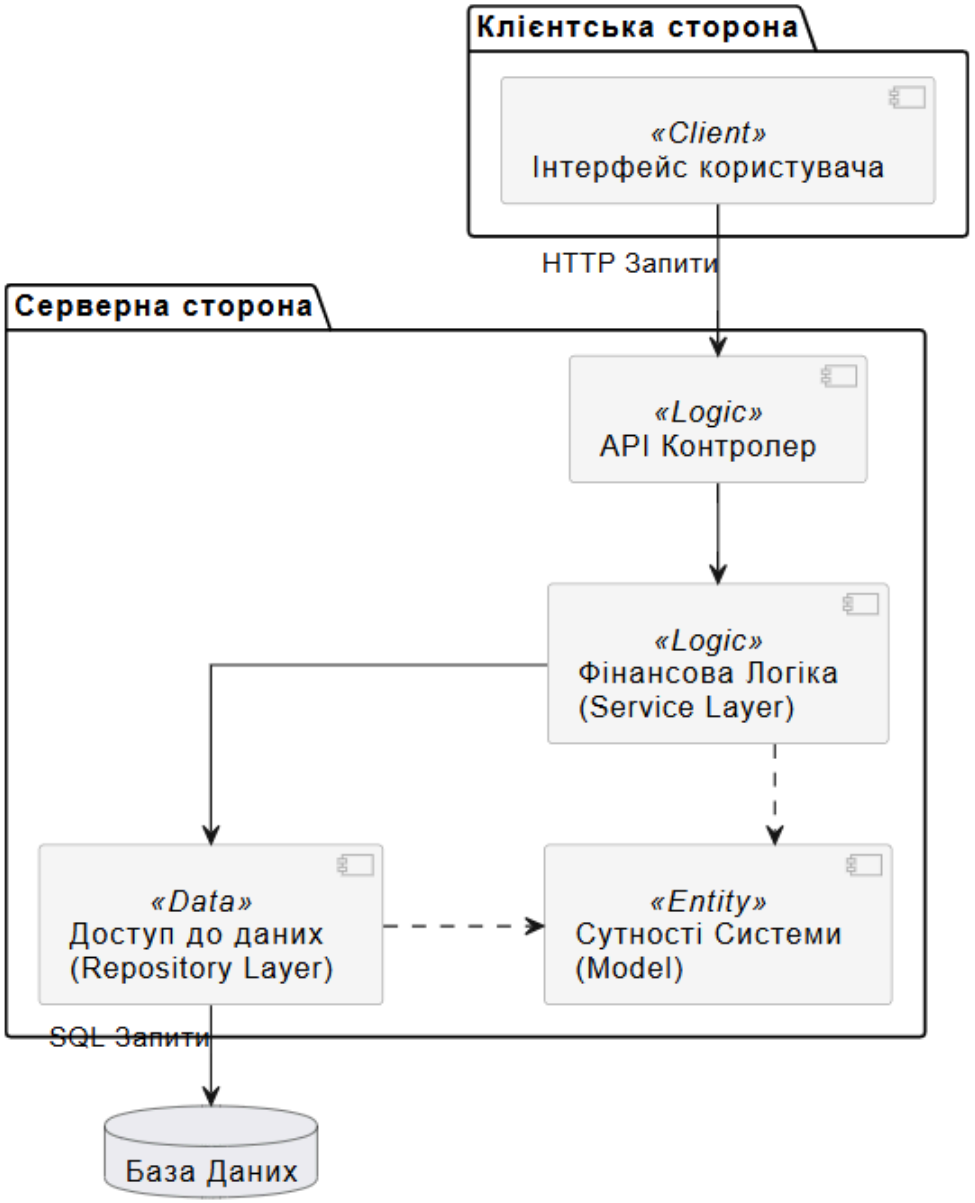
Сервер Додатків: Виділений сервер під управлінням ОС (Linux/Windows). У середовищі JVM та веб-сервера Tomcat розгорнуто виконуваний файл бекенду – Finance-Accounting.jar (Spring Boot).

Сервер Баз Даних: Окремий вузол із СКБД PostgreSQL, де фізично розміщена база даних Finance_DB.

Протоколи взаємодії:

- Клієнт – Сервер: Обмін даними через HTTP (REST API / JSON).
- Сервер – БД: Виконання SQL-запитів через протокол JDBC (порт 5432).

Діаграма компонентів



Опис компонентів :

Клієнтська сторона («Client»): Відповідає за взаємодію з користувачем, візуалізацію форм та відправку запитів на сервер.

Серверна сторона:

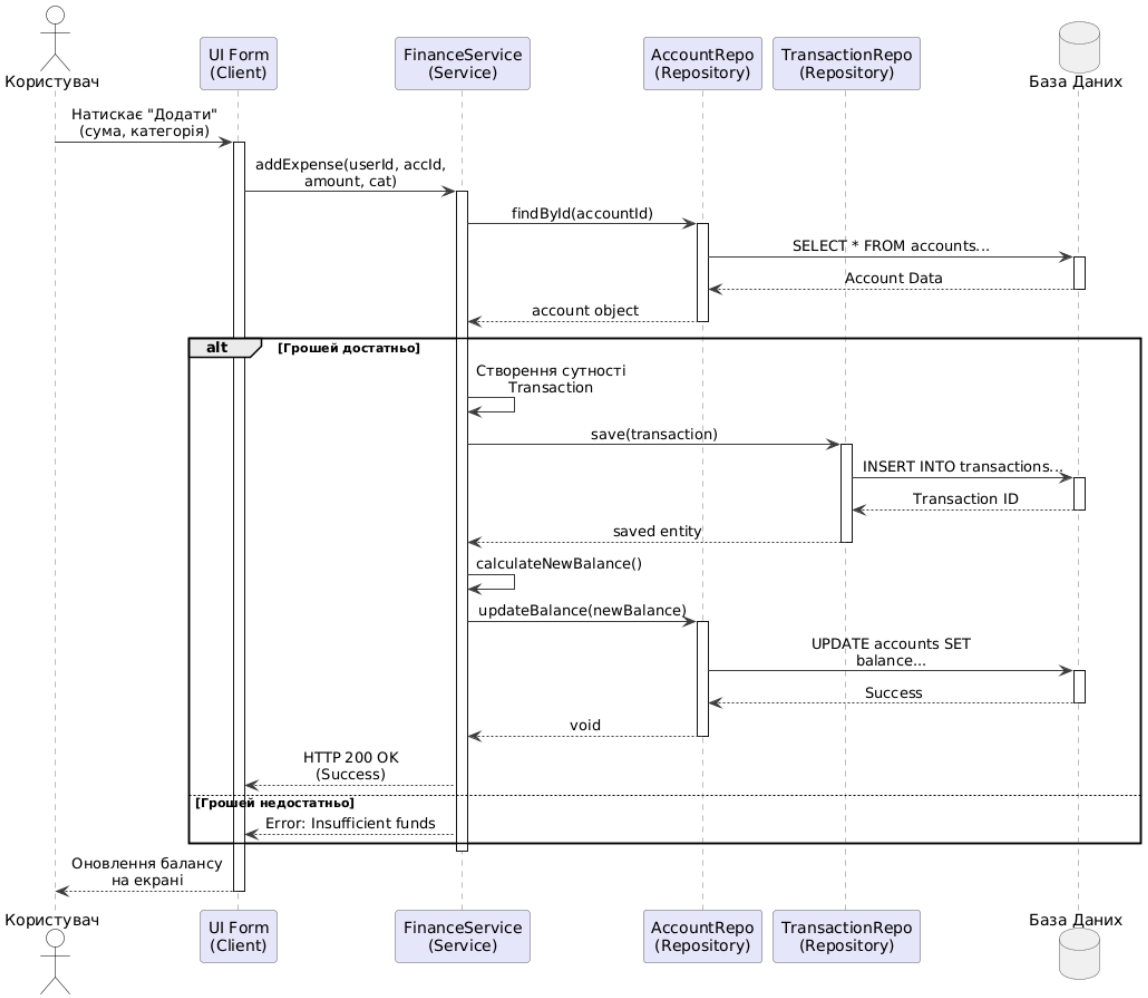
- API Контролер: Приймає HTTP-запити та передає їх на обробку.
- Фінансова Логіка («Logic»): Ядро системи; виконує бізнес-операції (розрахунки, транзакції).
- Доступ до даних («Data»): Шар репозиторіїв, що забезпечує зв'язок із базою даних.

Сутності («Entity»): Спільні моделі даних (User, Account), які використовуються всіма компонентами для обміну інформацією.

База Даних: Зовнішнє сховище для фізичного збереження інформації через SQL-запити.

Діаграми послідовностей

Додавання витрати

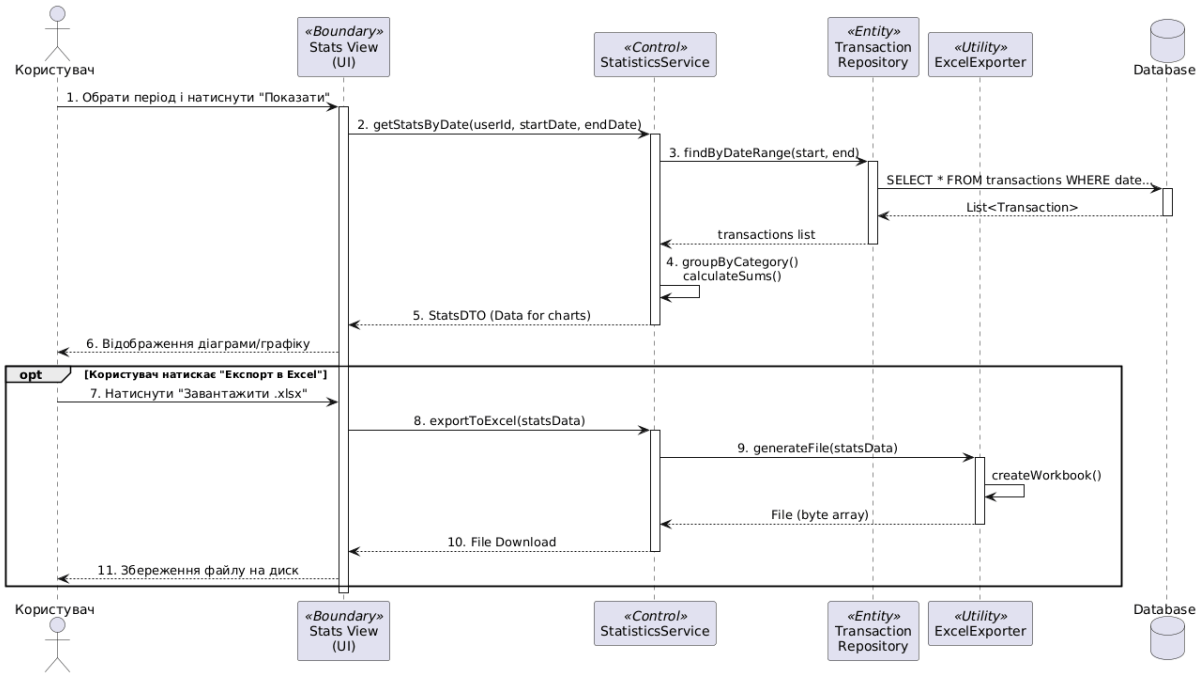


Сценарій 1: Додавання витрати

1. Користувач вводить суму та категорію і натискає кнопку «Додати витрату» в інтерфейсі.
2. UI Form передає введені дані у Finance Service для обробки бізнес-логіки.
3. Сервіс звертається до Account Repo, щоб знайти відповідний рахунок і отримати його поточний стан із Бази Даних.
4. Якщо коштів достатньо, сервіс зберігає нову транзакцію через Transaction Repo, створюючи запис (INSERT) у БД.

5. Finance Service автоматично перераховує залишок коштів і оновлює баланс рахунку через Account Repo (UPDATE).
6. Сервер повертає підтвердження успішного виконання операції (HTTP 200 OK).
7. Клієнт оновлює інтерфейс, відображаючи нову транзакцію та актуальний баланс користувачу.

Побудова статистики



Сценарій 2: Побудова статистики та Експорт в Excel

1. Користувач обирає часовий період (дати початку та кінця) і натискає кнопку «Показати» в інтерфейсі (Stats View).
2. UI відправляє запит getStatsByDate до контролера бізнес-логіки StatisticsService.
3. Сервіс звертається до Transaction Repository (метод findByDateRange), щоб отримати список транзакцій за вказаний проміжок часу.

4. Репозиторій виконує SQL-запит (SELECT * ...) до Бази Даних і повертає отриманий список об'єктів транзакцій сервісу.
5. StatisticsService виконує внутрішню обробку даних: групує транзакції за категоріями та підсумовує витрати (calculateSums).
6. Сервіс повертає сформований об'єкт StatsDTO (готовий набір даних для візуалізації) на клієнтську частину.
7. Інтерфейс відображає користувачу графік або діаграму витрат.

Вихідний код:

https://github.com/mariiakoliada/TRPZ/tree/master/src/ia32/koliada/finance

Питання до лабораторної роботи

1. Що собою становить діаграма розгортання?

Діаграма розгортання показує, на яких фізичних вузлах (сервери, ПК, мобільні пристрої) розгортаються програмні артефакти (компоненти, виконувані файли).

2. Які бувають види вузлів на діаграмі розгортання?

Фізичні пристрої (сервер, комп’ютер, телефон), віртуальні вузли (VM, контейнер), а також підвузли всередині них (CPU, сервіс, підсистема).

3. Які бувають зв’язки на діаграмі розгортання?

Переважно це канали зв’язку/комунікації між вузлами, які показують мережеві з’єднання та обмін повідомленнями.

4. Які елементи присутні на діаграмі компонентів?

Компоненти, інтерфейси (надавані й вимаганні), залежності між компонентами, пакети й, за потреби, артефакти.

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

Це залежності й інтерфейсні зв’язки, які показують, який компонент використовує (імпортує) інший, через який інтерфейс і в якому напрямку.

6. Які бувають види діаграм взаємодії?

Діаграма послідовностей, діаграма комунікації, діаграма часових характеристик і діаграма загальної взаємодії.

7. Для чого призначена діаграма послідовностей?

Щоб показати в часі обмін повідомленнями між об’єктами/акторами для реалізації конкретного сценарію (варіанту використання).

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Актори та об’єкти (lifelines), повідомлення між ними, активності (activation bars), фрагменти управління (alt, loop, opt), а також створення/знищення об’єктів.

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Кожен варіант використання можна деталізувати однією або кількома діаграмами послідовностей, які показують, як саме реалізується сценарій.

10. Як діаграми послідовностей пов’язані з діаграмами класів?

Діаграма послідовностей показує динамічну взаємодію екземплярів класів, а діаграма класів — їх статичну структуру; об’єкти на послідовності зазвичай є екземплярами класів із діаграми класів.

Висновок: під час виконання лабораторної роботи, навчилася проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробила діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.