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
ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ.
ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ
Особиста Бухгалтерія

Виконала:
Студентка групи ІА-22
Лахоцька С. О.

Перевірив:
Мягкий М. Ю

Київ 2024

Зміст

Тема: Діаграма розгортання, діаграма компонентів, діаграма послідовностей
Мета: Проаналізувати тему, створити діаграму розгортання та діаграму компонентів

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграма розгортання відображає фізичну структуру системи, вказуючи, на якому обладнанні або середовищах виконуються її компоненти. Основними елементами діаграми розгортання є вузли (nodes), що представляють фізичне обладнання або програмне забезпечення, яке може містити інші елементи системи. Вузли поділяються на два типи: пристрої (наприклад, сервери, комп'ютери або мобільні пристрої) і середовища виконання (такі як веб-сервери або операційні системи). Кожен вузол може містити програмні артефакти, такі як виконувані файли, бібліотеки, бази даних або інші компоненти, необхідні для функціонування системи.

Діаграма розгортання також містить зв'язки між вузлами, що представляють собою канали обміну даними, такі як HTTP-з'єднання або інші протоколи комунікації. Ці зв'язки дозволяють зрозуміти, як вузли взаємодіють між собою і яким чином забезпечується потік інформації між компонентами системи. Діаграми розгортання часто використовуються для планування фізичного розташування компонентів на етапах розгортання та налагодження системи, що допомагає уникнути проблем з продуктивністю і забезпечити відповідність системи вимогам до мережевої інфраструктури та обробки навантажень.

Існують два типи діаграм розгортання: описові та екземплярні. Описові діаграми зображають структуру системи загалом, вказуючи на необхідне обладнання та програмні вимоги. Екземплярні ж конкретизують дані про окремі вузли (наприклад, про певний сервер), що важливо на етапі фінального розгортання системи.

Діаграма компонентів (Component Diagram)

Діаграма компонентів відображає програмні модулі системи та їхні взаємозв'язки, що дозволяє спроектувати систему як набір взаємозамінних частин. Основними елементами діаграми є компоненти (components), що представляють собою незалежні модулі системи. Кожен компонент може взаємодіяти з іншими за допомогою інтерфейсів, що визначають функціональні можливості модуля. Наприклад, в системі управління даними окремими компонентами можуть бути модулі для обробки даних, інтерфейси взаємодії з користувачем, сервіси бази даних тощо.

Діаграма компонентів використовується для моделювання структури програмного коду та опису архітектури системи. Вона дозволяє розробникам визначити, які частини програми можна створювати незалежно, а також забезпечує можливість повторного використання коду за рахунок чіткого розмежування функціональних частин. Компоненти можуть також розподілятися між різними фізичними вузлами в діаграмі розгортання, що забезпечує гнучкість у розробці розподілених або компонентно-орієнтованих систем.

Розробка діаграми компонентів є ключовою для забезпечення масштабованості і зручності обслуговування, особливо у великих проєктах. Вона дозволяє відстежувати залежності між модулями і знижувати ризик конфліктів під час інтеграції.

Діаграма послідовностей (Sequence Diagram)

Діаграма послідовностей описує порядок взаємодії між об'єктами системи у рамках певного сценарію або процесу. Вона допомагає візуалізувати послідовність обміну повідомленнями між об'єктами під час виконання операцій, що є особливо корисним для розуміння алгоритмів та процесів взаємодії в системі. На діаграмі послідовностей зображаються об'єкти системи (actor), повідомлення, які вони обмінюються, і порядок цих повідомлень. Основні елементи включають учасників (actors), повідомлення (messages), що передаються між ними, та часові лінії, які відображають порядок виконання дій.

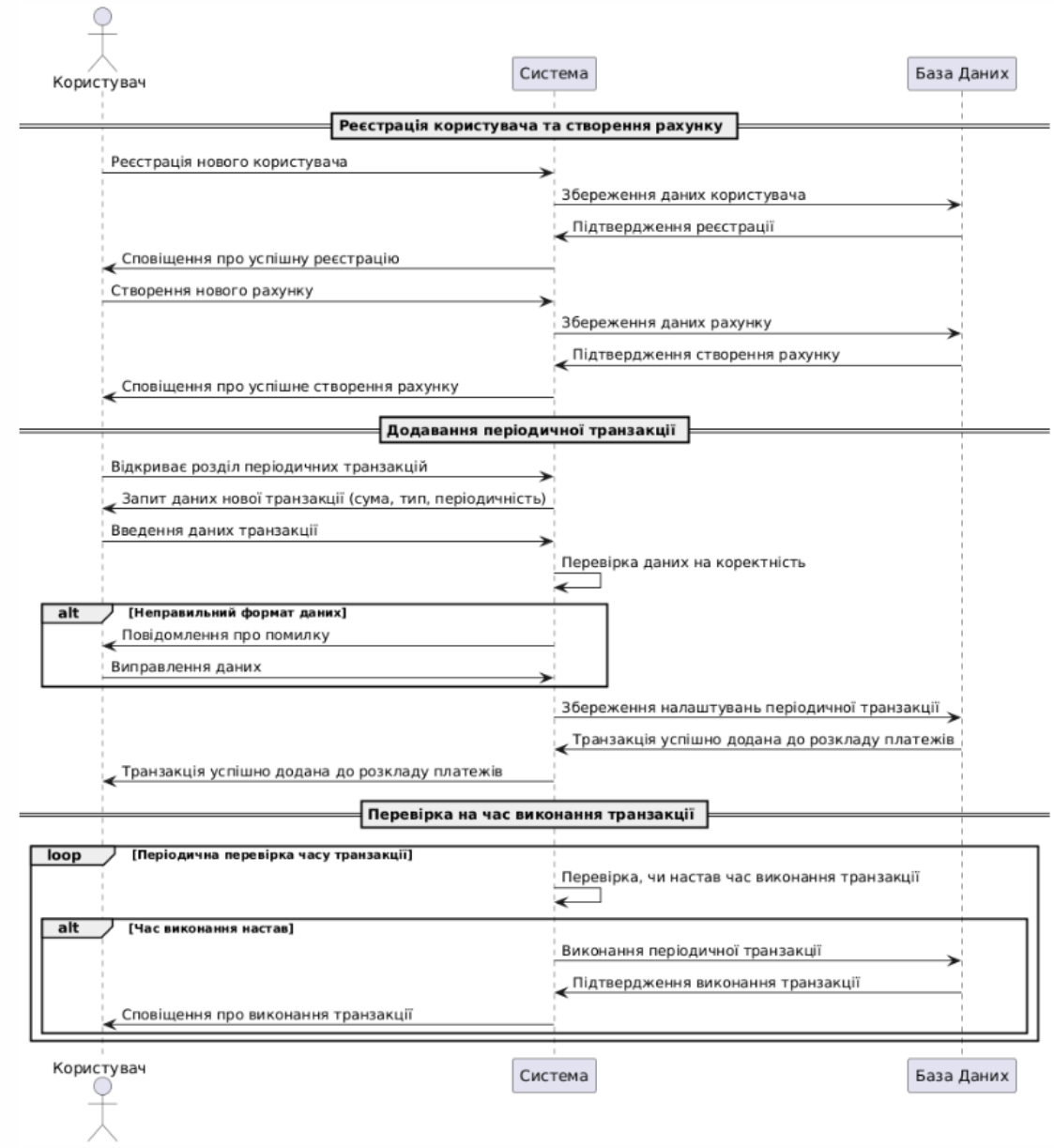
Завдяки діаграмам послідовностей можна точно визначити основний потік виконання функцій та альтернативні сценарії, які виникають при певних умовах. Наприклад, в системі інтернет-магазину діаграма послідовностей може відображати процес оформлення замовлення: від вибору товарів і заповнення інформації про доставку до підтвердження замовлення і надсилання його на склад. Кожен з цих кроків представлений як окреме повідомлення на діаграмі, що дозволяє чітко побачити, як дані передаються між компонентами системи і в якому порядку відбуваються операції.

Використання діаграм послідовностей є важливим етапом у розробці систем, які мають складні алгоритми або потребують точного контролю над послідовністю операцій. Вони дозволяють спростити тестування і налагодження, оскільки допомагають передбачити можливі збої в логіці роботи системи та зрозуміти взаємодію компонентів на рівні процесів.

Хід роботи

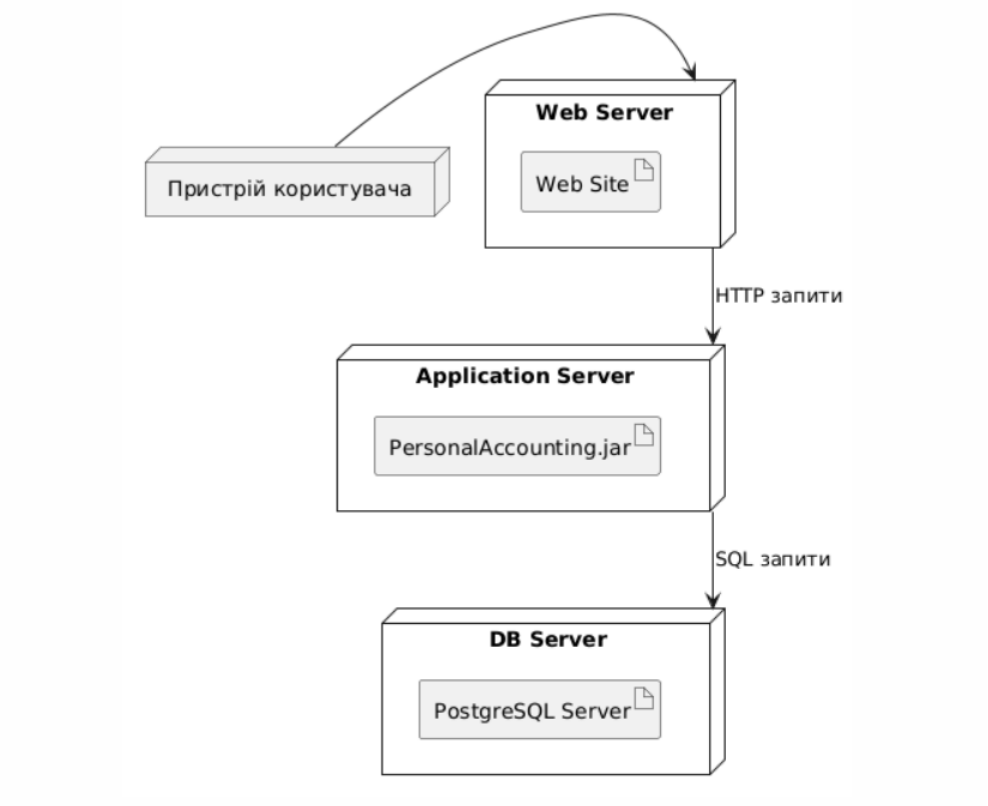
Схема послідовностей

Ця діаграма послідовностей показує процес реєстрації користувача, створення рахунку, додавання періодичної транзакції та автоматичного виконання цих транзакцій у встановлений час. Спочатку користувач реєструється в системі, і його дані зберігаються в базі даних. Після успішної реєстрації користувач створює новий рахунок, дані якого також зберігаються. Далі користувач додає нову періодичну транзакцію, вводячи необхідні дані, які система перевіряє на коректність. Якщо формат даних неправильний, користувач отримує повідомлення про помилку і може виправити дані. Після успішного збереження налаштувань транзакції система періодично перевіряє, чи настав час для виконання транзакції. Коли час настає, транзакція виконується, і користувач отримує сповіщення про це.



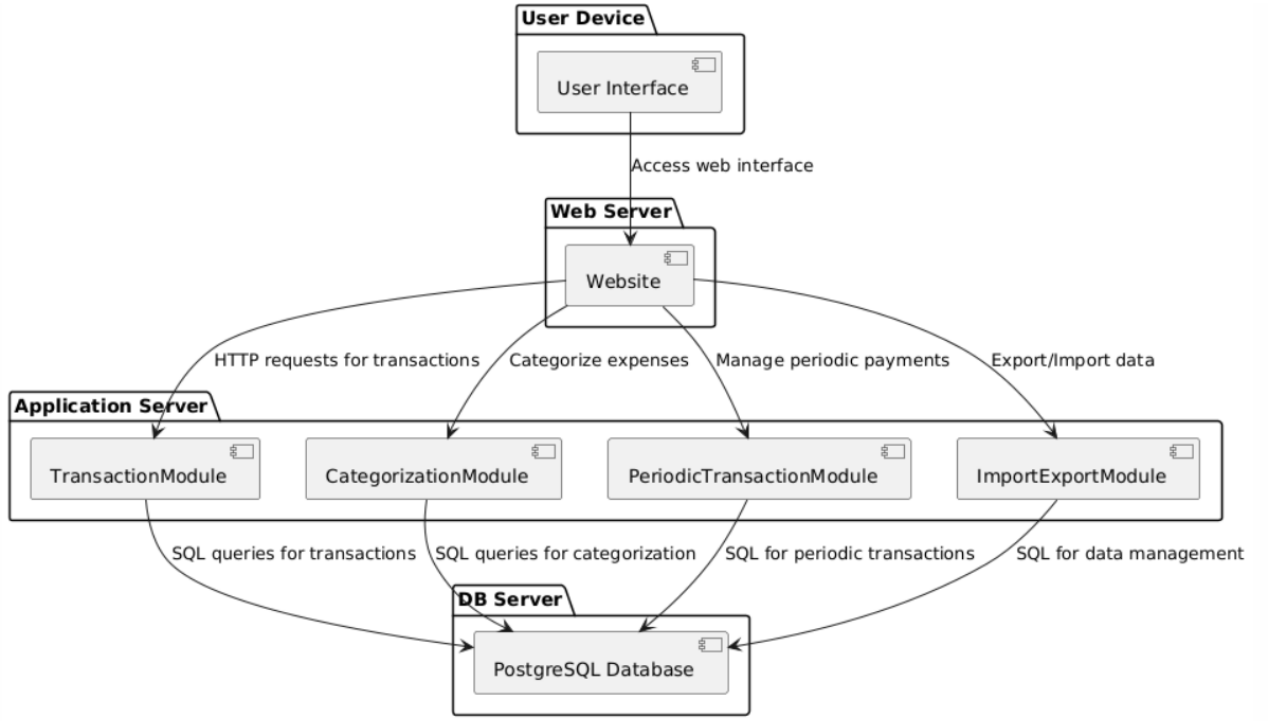
Діаграма Розгортання

Ця діаграма представляє тришарову архітектуру серверної системи. Користувацький пристрій (позначений як "Пристрій користувача") надсилає HTTP-запити до веб-сервера, на якому розміщений вебсайт. Веб-сервер передає ці запити до серверу додатків, де розташований файл PersonalAccounting.jar для обробки бізнес-логіки. Сервер додатків надсилає SQL-запити до серверу баз даних (DB Server), на якому встановлений PostgreSQL Server для збереження та обробки даних.



Діаграма компонентів

Ця діаграма показує компоненти системи керування фінансами з багатошаровою архітектурою. Користувач взаємодіє з інтерфейсом через веб-сайт, розміщений на веб-сервері. Веб-сервер обробляє HTTP-запити для фінансових операцій та передає їх на сервер додатків, який складається з чотирьох модулів: TransactionModule (для обробки транзакцій), CategorizationModule (для категоризації витрат), PeriodicTransactionModule (для керування періодичними платежами) та ImportExportModule (для експорту/імпорту даних). Кожен модуль сервера додатків надсилає SQL-запити до сервера баз даних PostgreSQL, де зберігаються та обробляються відповідні дані.



Висновки

У цій лабораторній роботі було створено діаграми розгортання, компонентів і послідовностей для особистої бухгалтерії. Діаграми відображають архітектуру та взаємодію між веб-сервером, сервером додатків і сервером баз даних, що забезпечує ефективну обробку даних.

Вихідний код

У роботі розглянуто діаграми розгортання, компонентів та послідовностей, які дозволяють моделювати фізичну структуру системи, зв'язки між компонентами та порядок виконання дій. Хоча теоретично проаналізовано всі три види діаграм, на практиці було реалізовано лише одну з них. Реалізована діаграма демонструє основну архітектуру системи «Особиста Бухгалтерія» і показує, як фізично взаємодіють компоненти під час роботи програми. Це забезпечило прозорість планування і надійність майбутньої розробки.

```
// file: src/main/java/org/example/accounting/api/service/QrCodeService.java

package org.example.accounting.api.service;


import com.google.zxing.BinaryBitmap;

import com.google.zxing.MultiFormatReader;

import com.google.zxing.NotFoundException;

import com.google.zxing.client.j2se.BufferedImageLuminanceSource;

import com.google.zxing.common.HybridBinarizer;

import org.example.accounting.api.web.dto.ReceiptInfo;

import org.springframework.stereotype.Service;

import org.springframework.web.multipart.MultipartFile;


import javax.imageio.ImageIO;

import java.io.IOException;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.Arrays;

import java.util.stream.Collectors;
```

```
@Service
public class QrCodeService {

    private final MultiFormatReader multiFormatReader = new MultiFormatReader();

    public String readQrCode(MultipartFile file) throws IOException, NotFoundException {
        BinaryBitmap binaryBitmap = new BinaryBitmap(new HybridBinarizer(
                new BufferedImageLuminanceSource(ImageIO.read(file.getInputStream()))));

        return multiFormatReader.decode(binaryBitmap).getText();
    }

    public ReceiptInfo parseDataFromReceiptQrCode(MultipartFile file) throws NotFoundException, IOException {
        var qrCodeData = readQrCode(file);

        var data = Arrays.stream(qrCodeData.split("&"))
                .map(s -> s.split("="))
                .collect(Collectors.toMap(s -> s[0], s -> s[1]));

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd'T'HHmm");
        LocalDateTime dateTime = LocalDateTime.parse(data.get("t"), formatter);
        double amount = Double.parseDouble(data.get("s"));
```

```
        return new ReceiptInfo(dateTime, amount);
    }
}

package org.example.accounting.api.web.dto;

import java.time.LocalDateTime;

public record ReceiptInfo (LocalDateTime dateTime, double amount) {
}
```