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 Розробки Програмного Забезпечення»

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

Варіант-11

Виконав:
Студент групи ІА-24
Коханчук Михайло Миколайович

Перевірив:
Мягкий Михайло Юрійович

Київ-2024

ЗМІСТ

МЕТА

Метою роботи є проєктування системи шляхом розробки діаграми розгортання, діаграми компонентів та діаграми послідовностей, а також підготовка звіту про виконану роботу відповідно варіанту(JSON Tool).

Теоритичні відомості:

UML (Unified Modeling Language) — це стандартизована мова для візуального моделювання, опису та документування програмного забезпечення, бізнес-процесів чи будь-яких інших систем. UML допомагає розробникам, аналітикам, тестувальникам та іншим учасникам проекту спілкуватися на єдиній мові.

Основні елементи UML:

Актори – зовнішні системи чи користувачі, які взаємодіють із системою.
Класи та об'єкти – описують статичну структуру системи.
Ассоціації, залежності та зв'язки – вказують на взаємодії між елементами.
Активності та стани – описують динамічну поведінку.
Компоненти та вузли – вказують на фізичні аспекти системи.

UML включає 14 типів діаграм, які діляться на *структурні* та *поведінкові*.

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

Хід роботи:

Для початку побудуємо діаграму розгортання для обраної теми роботи – JSON Tool.

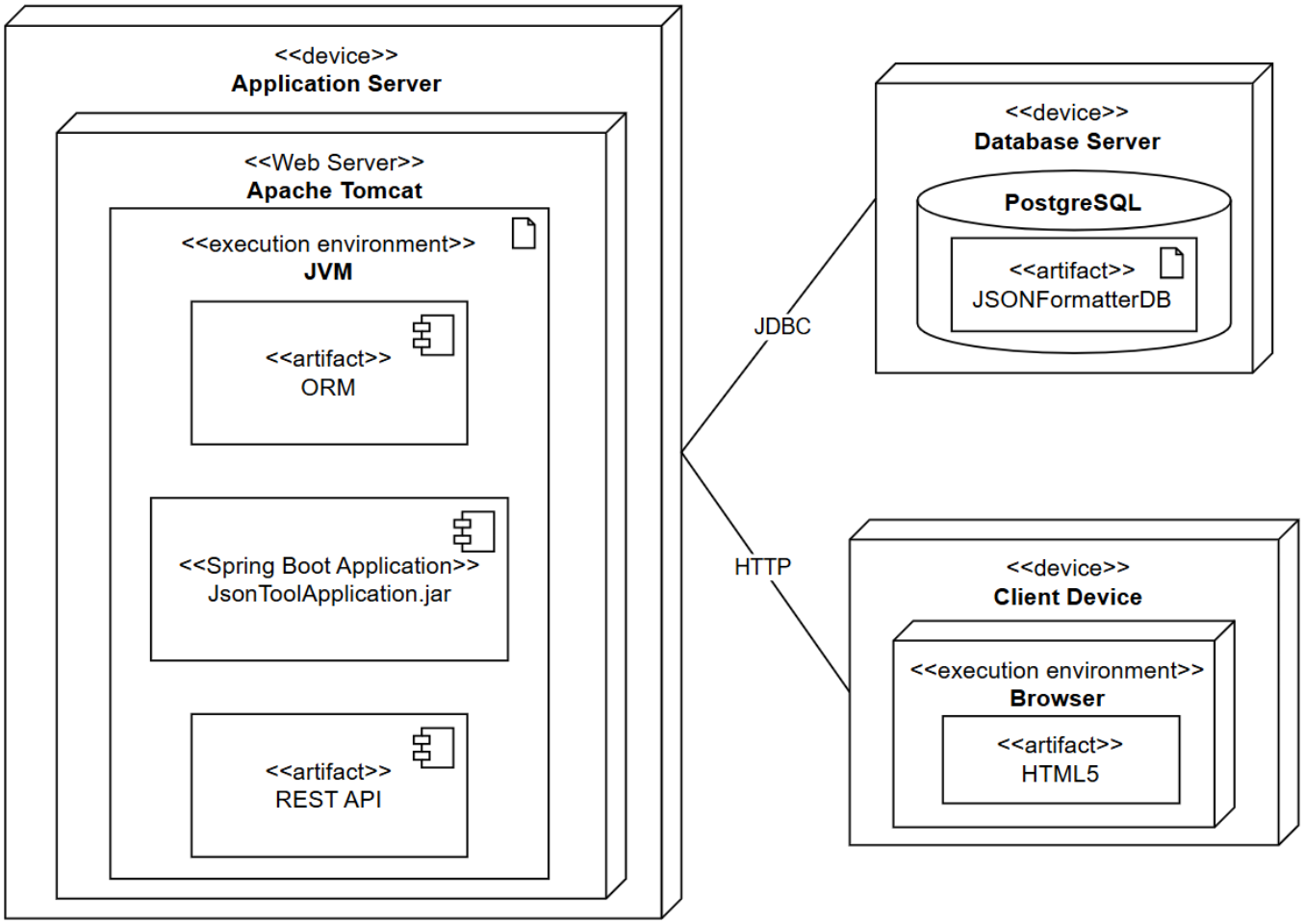

Рис. 1. Діаграма розгортання для JSON Tool

Застосунок розгортається на пристрої Application Server, який обробляє запити через REST API і взаємодіє з Database Server за допомогою JDBC. Клієнт отримує доступ до сервера через браузер по HTTP.

Компоненти діаграми

1. ***Application Server***
   - Web Server: Apache Tomcat виконує роль веб-сервера для обслуговування запитів.
   - Execution Environment: JVM (Java Virtual Machine), на якому працює застосунок.
   - Spring Boot Application:
     - Артефакт JsonToolApplication.jar є основним застосунком.
     - ORM (Object-Relational Mapping) відповідає за взаємодію із базою даних.
     - REST API: Забезпечує HTTP-комунікацію з клієнтами.
2. ***Database Server***
   - PostgreSQL: Використовується як СУБД.
   - Артефакт: JSONFormatterDB, який зберігає JSON-схеми та інші дані.
   - JDBC: Протокол для комунікації між застосунком та базою даних.
3. ***Client Device***
   - Execution Environment: Browser, що працює на клієнтському пристрої.
   - HTML5: Використовується для відображення інтерфейсу користувача.

Зв'язки

- **HTTP**: Комунікація між REST API на сервері та браузером на клієнтському пристрої.
- **JDBC**: Зв'язок між ORM і PostgreSQL для доступу до бази даних.

Далі побудуємо діаграму компонентів розробленої частини застосунку:

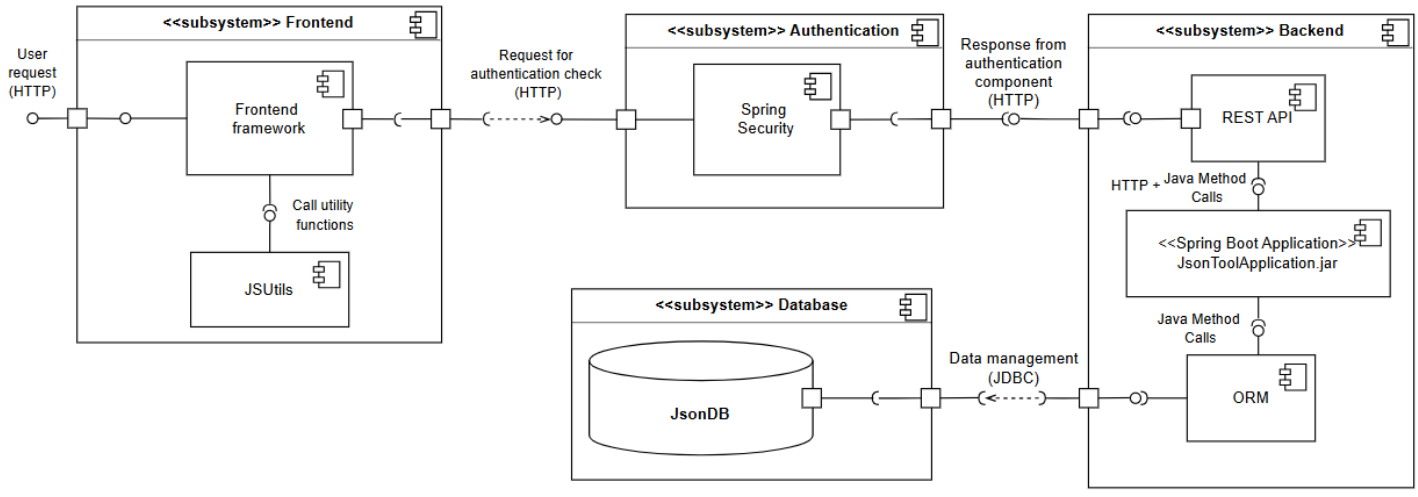

Рис. 2. Діаграма компонентів реалізованої частини системи

Загальний опис діаграми

Діаграма показує, як HTTP-запити користувача проходять через різні компоненти застосунку. Застосунок складається з чотирьох ключових підсистем:

1. Frontend
2. Authentication
3. Backend
4. Database

1. Підсистема: Frontend

Обробляє запити користувача (HTTP) та взаємодіє з підсистемою Backend для виконання функцій форматування JSON.

Компоненти:

- Frontend framework: Основний інтерфейс для обробки запитів користувача.
- JSUtils: JavaScript утиліти для виконання додаткових функцій на фронтенді (наприклад, валідація даних чи маніпуляції з JSON).

Потік даних:

- Користувач надсилає HTTP-запит через інтерфейс (Frontend framework).
- Викликаються допоміжні функції JSUtils для попередньої обробки запиту.
- Далі запит передається до Authentication для перевірки.

2. Підсистема: Authentication

Аутентифікація користувача за допомогою Spring Security.

Компоненти:

- Spring Security: Компонент для обробки HTTP-запитів на аутентифікацію користувача.

Потік даних:

- HTTP-запит передається в перевірку на аутентифікацію.
- Після успішної аутентифікації відповідь повертається назад до Frontend.

3. Підсистема: Backend

Обробляє бізнес-логіку та форматує JSON.

Компоненти:

- REST API: Обробник HTTP-запитів, що надає API для взаємодії з Frontend.
- JsonToolApplication.jar: Основний Spring Boot застосунок, який реалізує функціонал форматування JSON.
- ORM: Служить для взаємодії з базою даних через JDBC.

Потік даних:

- Після успішної аутентифікації, Frontend надсилає запит на REST API.
- REST API викликає відповідні методи в JsonToolApplication.jar для обробки JSON.
- Застосунок взаємодіє з ORM, якщо потрібна робота з даними.

4. Підсистема: Database

Зберігає та управляє даними для застосунку.

Компоненти:

- JsonDB: База даних, яка використовується для збереження JSON-об'єктів або логів.

Взаємодія:

- Використовується JDBC для обробки даних між Backend та базою даних.

Загальний потік даних:

1. Користувач надсилає HTTP-запит через Frontend.
2. Запит передається на Spring Security для аутентифікації.
3. Після успішної аутентифікації запит переходить до Backend.
4. REST API обробляє запит і викликає методи з JsonToolApplication.jar для форматування JSON.
5. Якщо необхідно, ORM виконує операції з JsonDB за допомогою JDBC.
6. Результат повертається назад до користувача через Frontend.

Далі розробимо діаграму послідовностей:

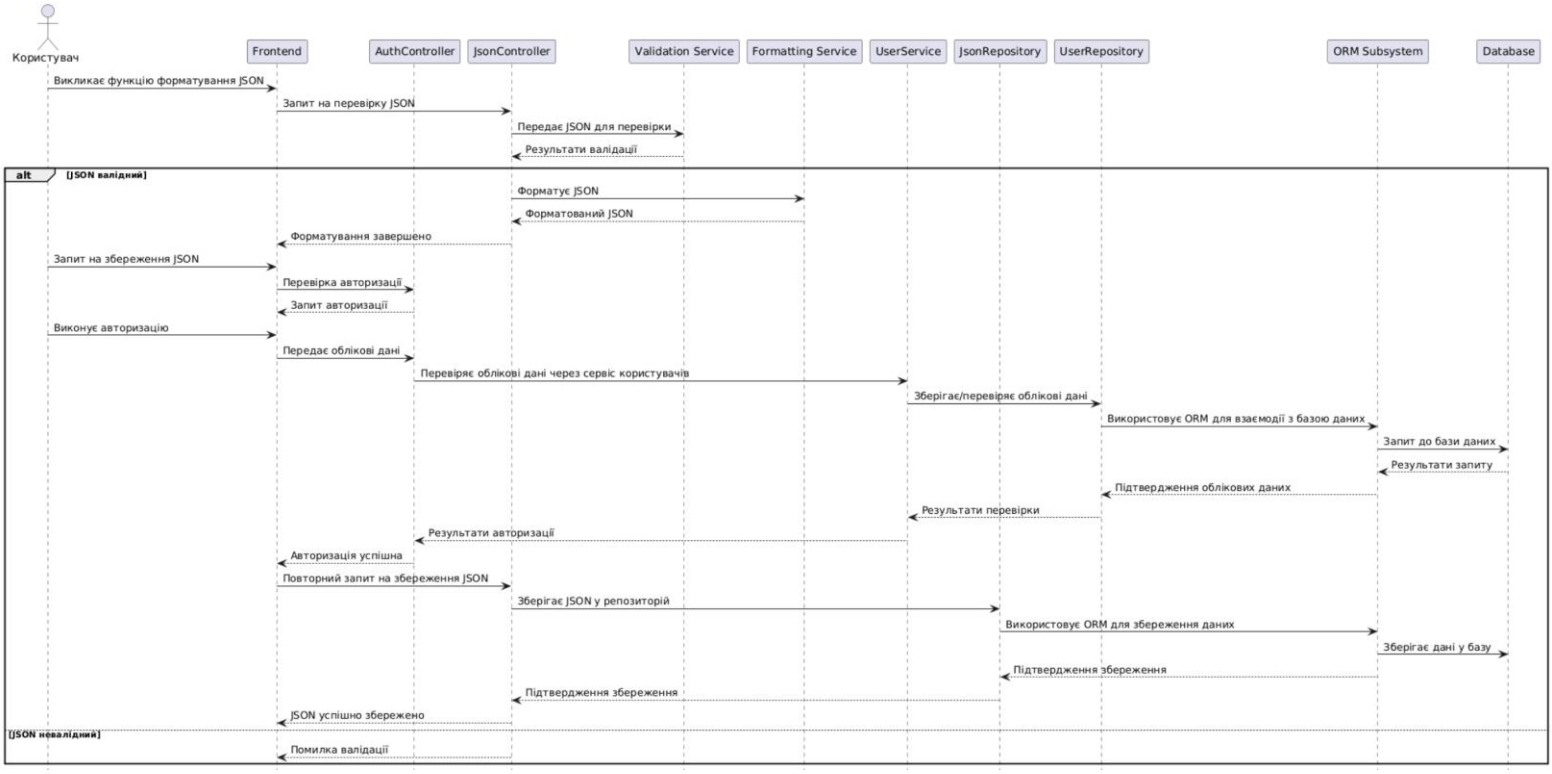

Рис. 3. Діаграма компонентів послідовностей частини системи

Ця діаграма показує, як відбувається обробка JSON, включно з форматуванням, авторизацією користувача та збереженням у базу даних. Вона демонструє взаємодію між усіма основними сервісами, контролерами та сховищами даних.

Основний процес включає два варіанти сценаріїв: успішна валідація JSON та помилка валідації.

Актор «Користувач» ініціює запит на форматування та збереження JSON.

Основні модулі:

1. Frontend – інтерфейс користувача, який надсилає запити до контролерів.
2. AuthController – відповідає за авторизацію користувача.
3. JsonController – основний контролер для обробки запитів щодо JSON.
4. Validation Service – сервіс, який перевіряє валідність JSON.
5. Formatting Service – відповідає за форматування JSON.
6. UserService – сервіс для перевірки облікових даних користувача.
7. JsonRepository – відповідає за збереження JSON у сховище.
8. UserRepository – взаємодіє з базою даних для перевірки облікових даних.
9. ORM Subsystem – для взаємодії з базою даних.
10. Database – кінцеве сховище даних.

Основний сценарій: Успішна валідація JSON

1. Користувач викликає функцію форматування JSON (запит передається через Frontend до JsonController).

2. JsonController:
   - Передає JSON у Validation Service для перевірки валідності.
   - Отримує результат валідації.

3. Якщо JSON валідний:
   - Formatting Service форматує JSON.
   - Форматування завершується, і надходить запит на збереження JSON.

4. Проводиться перевірка авторизації:
   - AuthController надсилає запит до UserService.
   - UserService взаємодіє з UserRepository та ORM Subsystem для перевірки облікових даних у Database.

5. У разі успішної авторизації:
   - Запит на збереження JSON повторно надходить у JsonRepository.
   - JsonRepository використовує ORM для збереження даних у базу.
   - Дані успішно зберігаються, і відправляється підтвердження збереження користувачу.

Альтернативний сценарій: Помилка валідації JSON

1. Якщо JSON не проходить перевірку у Validation Service, користувач отримує помилку валідації.
2. Збереження JSON не виконується.

Основні комунікації:

- Frontend ↔ JsonController: Вхідні запити та результати обробки.
- JsonController ↔ Validation Service: Перевірка JSON.
- JsonController ↔ Formatting Service: Форматування JSON.
- AuthController ↔ UserService: Перевірка авторизації користувача.
- UserService ↔ UserRepository ↔ ORM Subsystem ↔ Database: Доступ до бази даних для перевірки облікових даних.
- JsonRepository ↔ ORM Subsystem ↔ Database: Збереження JSON у базі даних.

ВИСНОВОК

У ході роботи було спроєктовано систему JSON Tool шляхом розробки діаграми послідовностей яка показує взаємодію користувача з компонентами системи під час форматування валідації та збереження JSON, діаграми розгортання яка представлена архітектурою системи з серверами середовищем виконання та комунікацією через HTTP і JDBC та діаграми компонентів яка відображає структуру основних модулів системи та їх функціональність. Розроблені діаграми забезпечують чітке розуміння архітектури та взаємодії системи для подальшої реалізації