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 розробки програмного забезпечення
«ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ.
ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ»

Виконала:
студенка групи ІА-24
Орловська А. В.
Перевірив:
Мягкий М. Ю.

Київ 2024

Зміст

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

Мета: Розробити діаграми розгортання, компонентів, взаємодій та послідовностей для обраної тема проекту

Короткі теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Мета:

Діаграма розгортання в UML використовується для моделювання фізичного розташування програмного забезпечення і апаратного забезпечення. Вона показує, як компоненти системи розгортаються на вузлах (сервери, клієнти тощо).

Ключові елементи:

- **Вузли (Nodes):** Фізичні пристрої або віртуальні сервери, де розгортається програмне забезпечення.
- **Артефакти (Artifacts):** Програмні модулі (наприклад, файли, бази даних), які розгортаються на вузлах.
- **Зв'язки (Communication Paths):** Лінії між вузлами, що показують обмін даними.

Приклад застосування:

- Розподіл серверних компонентів між веб-сервером, базою даних і клієнтськими пристроями.

Діаграма компонентів (Component Diagram)

Мета:

Діаграма компонентів використовується для представлення структури системи з точки зору її модулів або компонентів. Вона показує, як ці компоненти взаємодіють один з одним і з зовнішнім середовищем.

Ключові елементи:

- **Компоненти (Components):** Логічні модулі, що виконують певні функції.
- **Інтерфейси (Interfaces):** Визначають точки взаємодії компонентів.
- **Залежності (Dependencies):** Показують, як один компонент залежить від іншого.

Приклад застосування:

- Визначення зв'язків між мікросервісами у розподіленій архітектурі.

Діаграма взаємодій (Interaction Diagram)

Мета:

Діаграма взаємодій відображає, як об'єкти системи взаємодіють між собою для виконання певного сценарію.

Типи:

1. **Діаграма послідовностей (Sequence Diagram):**
    - **Мета:** Показує порядок взаємодії об'єктів у часі.
    - **Ключові елементи:**
        - **Актори (Actors):** Ініціюють взаємодії.
        - **Об'єкти (Objects):** Елементи системи, які беруть участь у процесі.
        - **Повідомлення (Messages):** Вказують передачу даних між об'єктами.
    - **Приклад:** Авторизація користувача в системі.
2. **Діаграма комунікацій (Communication Diagram):**
    - **Мета:** Фокусується на зв'язках між об'єктами.
    - **Ключові елементи:**
        - **Вузли (Nodes):** Об'єкти, які взаємодіють.
        - **Повідомлення:** Лінії, які показують передачу даних між вузлами.

Приклад застосування:

- **Діаграма послідовностей:** Моделювання процесу оформлення замовлення.
- **Діаграма комунікацій:** Представлення зв'язків між сервісами в мікросервісній архітектурі.

Ці діаграми в поєднанні допомагають зрозуміти архітектуру системи, її компоненти, фізичну інфраструктуру та взаємодію між різними елементами в часі.

Хід роботи

***Система для колективних покупок*(proxy, builder, decorator, façade, composite).** Система дозволяє створити список групи для колективної покупки, список що потрібно купити з орієнтовною вартістю кожної позиції та орієнтовною загальною вартістю, запланувати хто що буде купляти. Щоб користувач міг відмітити що він купив, за яку суму, з можливістю прикріпити чек. Система дозволяє користувачу вести списки бажаних для нього покупок, з можливістю позначати списки, які будуть доступні для друзів (як списки, що можна подарувати користувачеві). Система дозволяє добавляти інших користувачів в друзі.

Діаграма розгортання для проектованої системи

На діаграмі розгортання зображаємо фізичні компоненти, необхідні для роботи системи. Це сервер, на якому буде встановлено Backend та базу данних

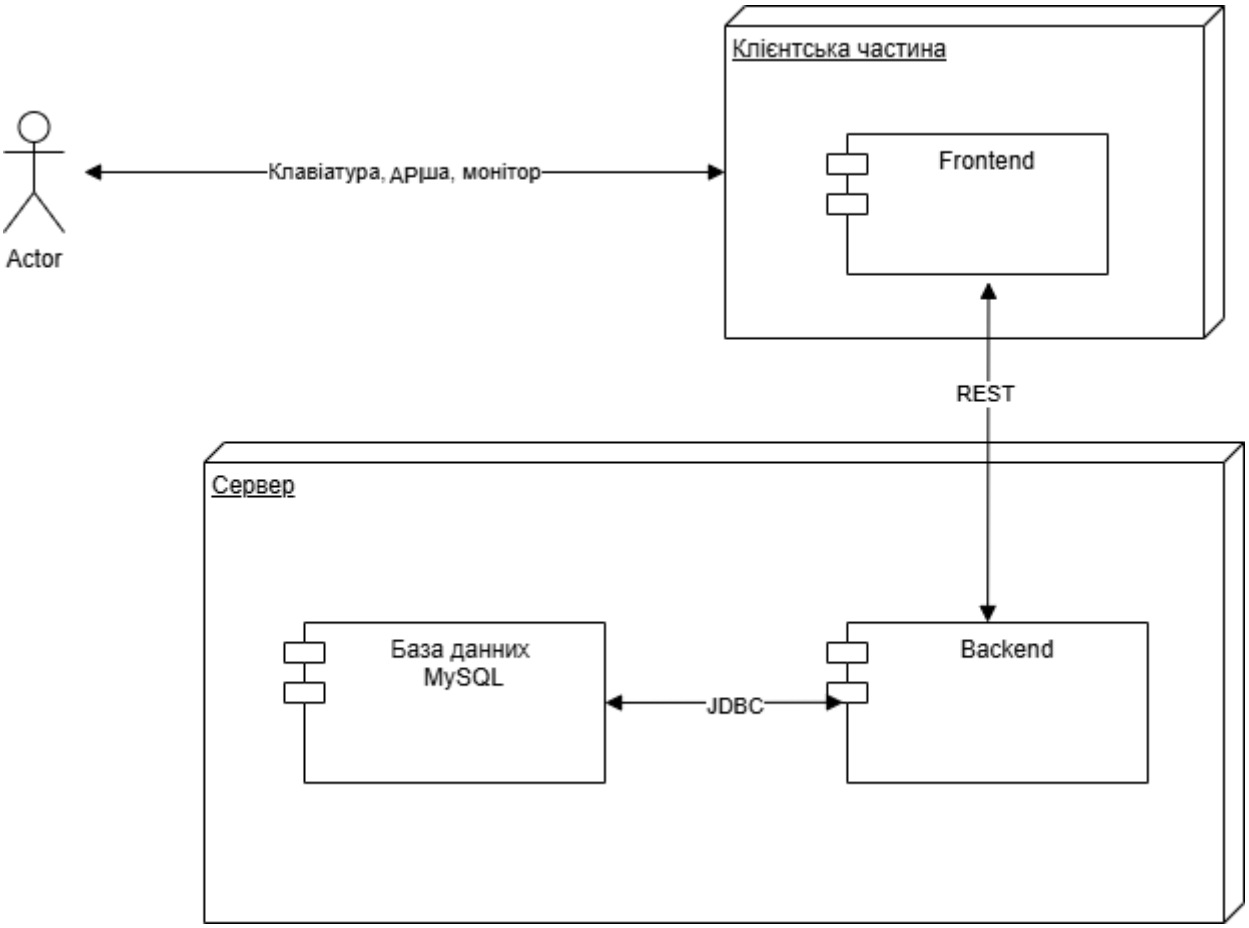

Рисунок 1 — Діаграма розгортання

Діаграма зображає архітектуру програмного забезпечення, поділену на клієнтську частину (Frontend) і серверну частину (Backend), а також взаємодію між користувачем і різними компонентами системи.

1. **Клієнтська частина**:

- **Frontend** — відповідає за взаємодію з користувачем і забезпечує користувацький інтерфейс. Вона надсилає запити до сервера через API (REST) і отримує відповіді.

2. **Серверна частина**:
   - **Backend** — реалізує бізнес-логіку застосунку. Приймає запити від клієнтської частини через REST API та обробляє їх.
   - **База даних MySQL** — зберігає дані застосунку. Backend взаємодіє з базою даних через JDBC (Java Database Connectivity) для виконання операцій із даними.
3. **Актор (Actor)**:
   - Уособлює користувача або систему, які взаємодіють із клієнтською частиною через API.

Взаємодія компонентів:

- Користувач (Actor) ініціює запит через клієнтську частину.
- Frontend надсилає запит до Backend через REST API.
- Backend взаємодіє з базою даних MySQL через JDBC для виконання операцій із даними.
- Після обробки запиту Backend повертає результат клієнтській частині, яка передає його користувачеві.

Діаграма компонентів для розроблюваної системи

На діаграмі компонентів зображаємо основні компоненти системи та їхні інтерфейси, через які вони взаємодіятимуть один з одним

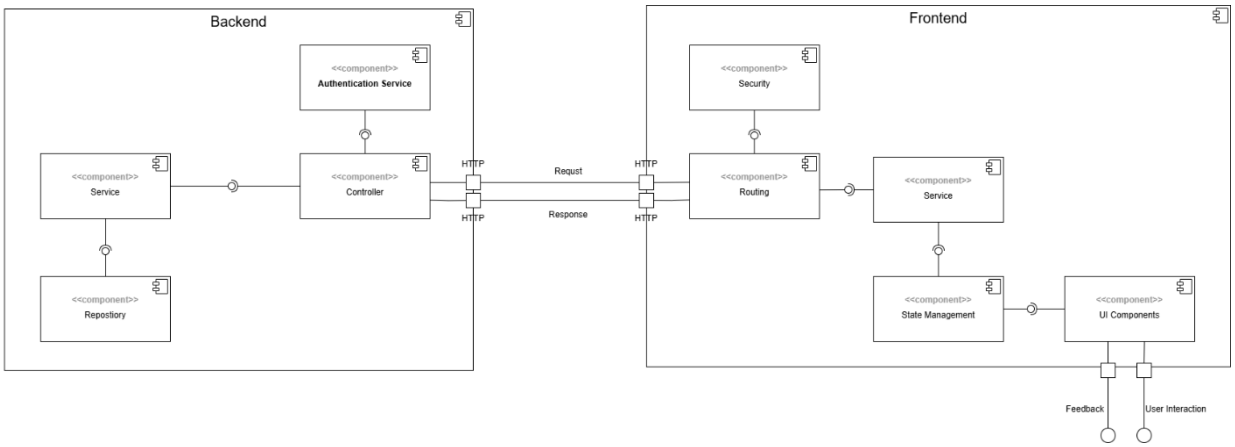

Рисунок 2 — Діаграма компонентів

Компоненти:

Backend (Серверна частина):

Серверна частина складається з компонентів, які відповідають за бізнес-логіку, зберігання даних і аутентифікацію:

1. **Authentication Service (Сервіс аутентифікації):**
   - Відповідає за завдання, пов’язані з аутентифікацією користувачів: вхід, вихід, перевірка користувача.
   - З'єднаний із компонентом Controller, імовірно, для перевірки запитів.
2. **Controller (Контролер):**
   - Виконує роль точки входу для HTTP-запитів.
   - Координує взаємодію між сервісами та надсилає відповіді (HTTP responses) до клієнтської частини.
3. **Service (Сервіс):**
   - Містить основну бізнес-логіку.
   - Взаємодіє з компонентом Repository для отримання або збереження даних.
4. **Repository (Репозиторій):**
   - Відповідає за роботу з базою даних.
   - Надає дані на рівень сервісів.

Frontend (Клієнтська частина):

Клієнтська частина відповідає за користувацький інтерфейс і логіку на стороні клієнта:

1. **Security (Безпека):**
    - Забезпечує функції безпеки на клієнтській стороні, наприклад, управління токенами або перевірку прав доступу.
2. **Routing (Маршрутизація):**
    - Управляє маршрутизацією додатку, співвідносячи URL-адреси із конкретними компонентами або сторінками.
    - З'єднаний із компонентами Security та Service.
3. **Service (Сервіс):**
    - Реалізує логіку для взаємодії з серверною частиною через HTTP-запити.
4. **State Management (Управління станом):**
    - Відповідає за збереження стану клієнтського додатку, наприклад, дані сесії користувача або глобальні змінні.
5. **UI Components (Компоненти інтерфейсу користувача):**
    - Відображають інтерфейс користувача.
    - Реагують на дії користувачів і надають їм відповідний зворотний зв'язок.

Взаємодія між Backend і Frontend:

- Зв’язок між Frontend і Backend здійснюється через HTTP за допомогою механізмів Request (запит) і Response (відповідь).
- Це дозволяє клієнтській частині викликати серверні сервіси та отримувати дані для відображення.

Взаємодія з користувачем:

- У нижній частині клієнтської частини показано взаємодію користувача із UI Components (компонентами інтерфейсу).

Діаграма послідовностей для проектованої системи

Спроєктувати діаграму послідовностей для одного із процесів розроблюваної системи.

На діаграмі послідовностей зображаємо процес завантаження чеку

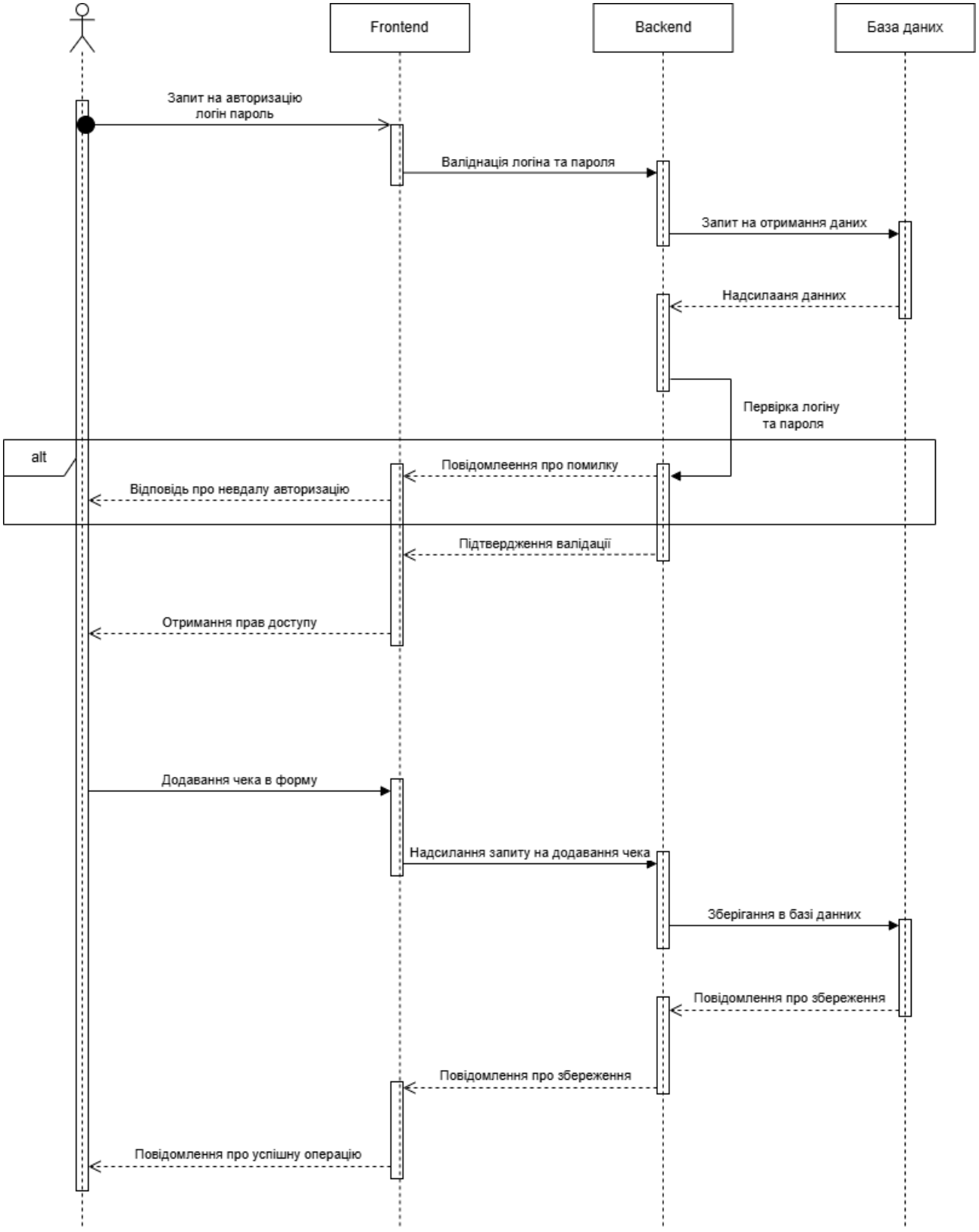

Рисунок 3 — Діаграма послідовностей

Діаграма послідовностей демонструє процес взаємодії користувача (Actor), бекенду (Backend), фронтенду(Frontend) і бази даних (Database) під час виконання операцій автентифікації та додавання нового чеку. Ось опис етапів:

1. Авторизація:

- Користувач вводить логін і пароль, які відправляються на Frontend.
- Frontend передає запит на авторизацію до Backend.
- **Backend:**
    - Виконує перевірку логіна і пароля, звертаючись до бази даних.
    - База даних надсилає відповідні дані для перевірки.
    - Backend визначає, чи введені дані є коректними:
        - У разі невдачі відправляється повідомлення про помилку на Frontend.
        - У разі успіху надсилається підтвердження, і користувач отримує права доступу.

2. **Додавання чека:**

- Після успішної авторизації користувач додає чек у форму.
- Frontend формує запит на додавання чека і передає його Backend.
- Backend зберігає дані в базі даних:
    - База даних підтверджує успішне збереження.
- Після цього Frontend отримує повідомлення про успішну операцію, яке відображається користувачу.

Альтернативний сценарій (alt):

- Якщо логін або пароль некоректні, Backend надсилає повідомлення про помилку авторизації на Frontend, і доступ не надається.

Посилання на репозиторій: https://github.com/annaorlovskaaa/TRPZ_labs.git

Висновок: У даній лабораторній роботі ми створили діаграми послідовностей, розгортання та компонентів, що дає нам змогу краще зрозуміти процеси системи в часі, план розгортання та внутрішню структуру