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»**

Виконав:

студент групи ІА–24

Яблонський Д.Б.

Перевірив:

Мягкий М.Ю.

Київ 2024

Зміст

Мета: навчитися створювати діаграми розгортання, компонентів та послідовностей у нотації UML для моделювання програмних систем. Має бути зображено фізичне розміщення компонентів системи (діаграма розгортання), логічну структуру та взаємодію між модулями (діаграма компонентів), а також відобразити послідовність взаємодій між компонентами під час ключових процесів (діаграма послідовностей).

Теоретичні відомості

Діаграма розгортання

Діаграма розгортання показує фізичне розташування програмних компонентів на апаратному забезпеченні системи. Вона відображає вузли (пристрої або сервери), де запускаються програмні модулі, та зв'язки між ними. Основною метою діаграми розгортання є візуалізація того, як система буде працювати в реальному середовищі. Вона демонструє, на яких пристроях або серверах розміщуються різні частини системи (клієнтська частина, сервер додатків, база даних) та як вони взаємодіють між собою. Це корисно для розуміння фізичної архітектури системи, її масштабованості та ефективності.

Діаграма компонентів

Діаграма компонентів використовується для відображення логічної структури програмного забезпечення та показує, як різні модулі або компоненти системи взаємодіють між собою. Компоненти — це окремі частини програмного забезпечення, які виконують конкретні функції та можуть взаємодіяти через інтерфейси. Ця діаграма допомагає зрозуміти структуру системи, які модулі вона містить, та як ці модулі обмінюються інформацією. Використовується для документування системи на етапі проектування та дає уявлення про логічну організацію коду.

Діаграма послідовностей

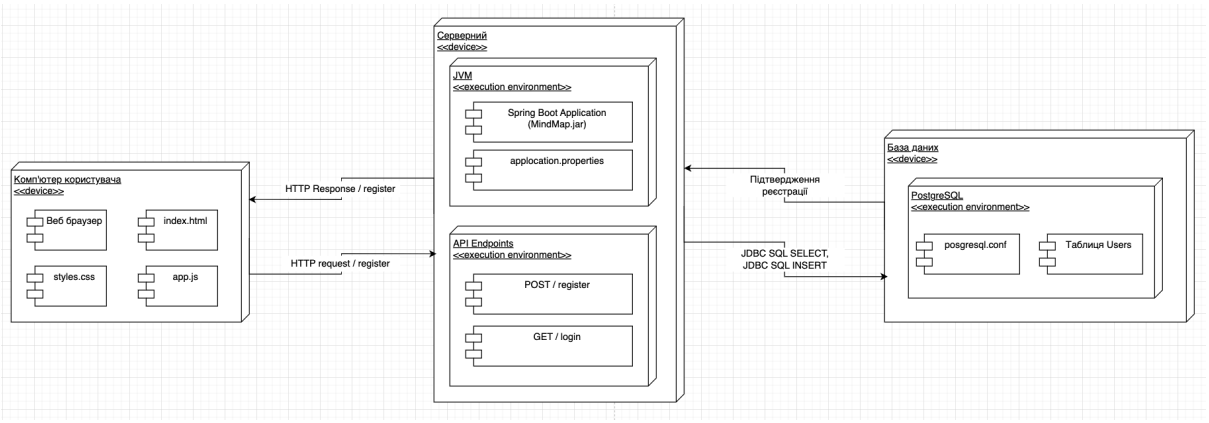
Діаграма послідовностей показує, як об'єкти системи взаємодіють між собою в хронологічному порядку для виконання певної операції. Вона фокусується на тому, які повідомлення передаються між компонентами та в якому порядку. Це корисний інструмент для моделювання динамічних аспектів системи, таких як процес реєстрації користувача або збереження даних. Діаграма допомагає зрозуміти, як відбувається взаємодія між клієнтом, сервером та базою даних у реальному часі, забезпечуючи чітке уявлення про порядок виконання операцій.

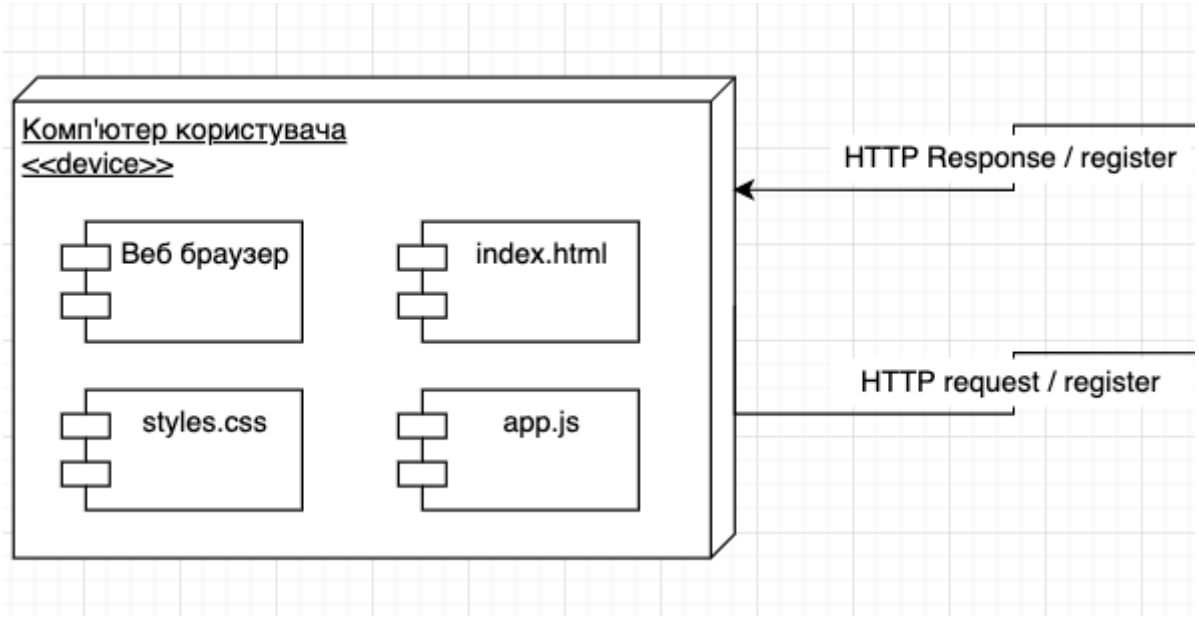
Хід роботи

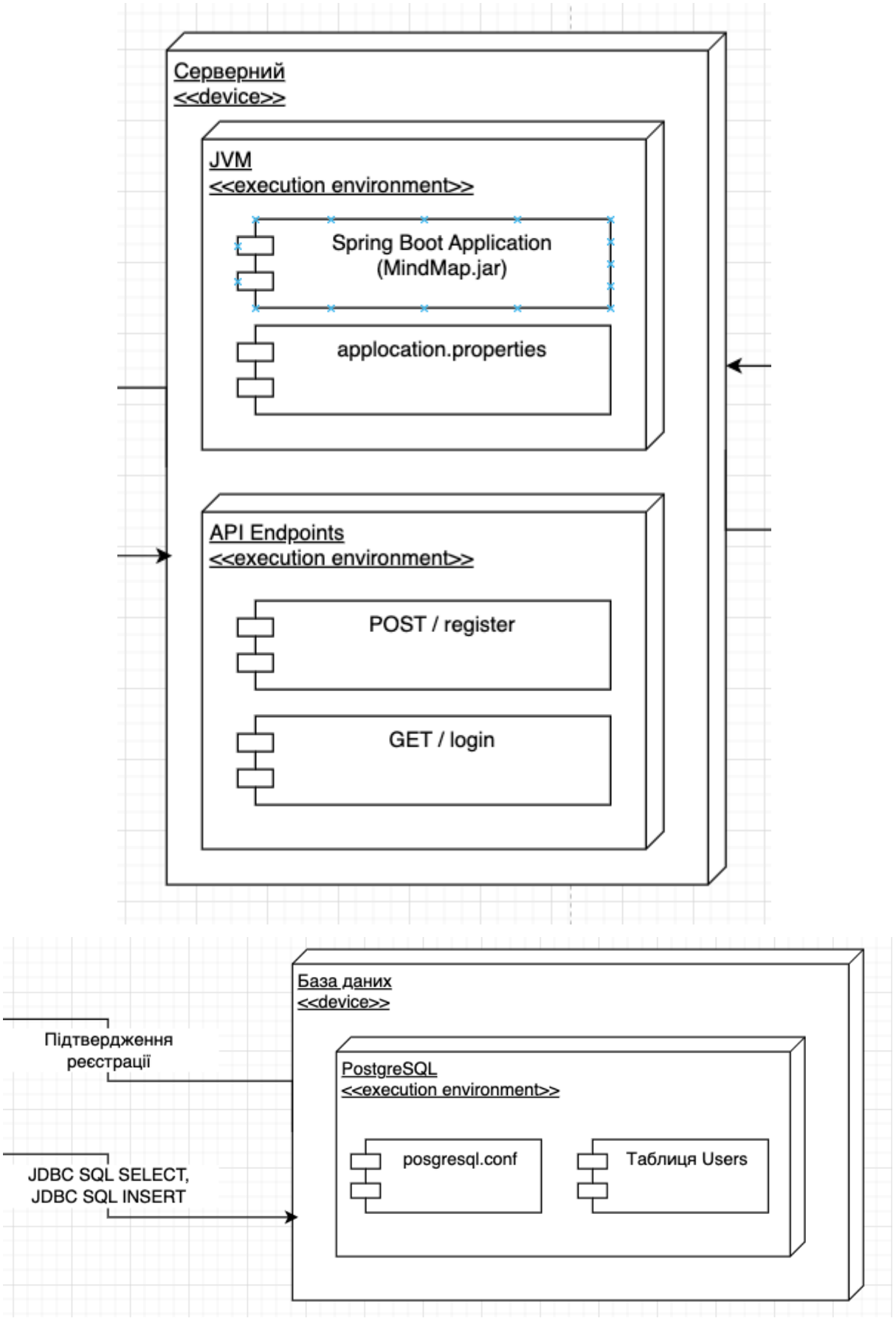
1. Ознайомитися з короткими теоретичними відомостями.
2. Розробити діаграму розгортання для проектованої системи.
3. Розробити діаграму компонентів для проектованої системи.
4. Розробити діаграму послідовностей для проектованої системи.

Тема: Mind-mapping software

Діаграма розгортання (реєстрація користувача)







1. Клієнтський пристрій (Комп'ютер користувача)

- Компоненти:

    - Веб браузер: Інтерфейс, через який користувач взаємодіє з системою.
    - HTML (index.html): Відповідає за структуру сторінки реєстрації, що відображається користувачу.
    - CSS (styles.css): Стилі, які надають інтерфейсу користувача візуальне оформлення.
    - JavaScript (app.js): Обробляє взаємодію користувача з інтерфейсом та відправляє HTTP запити на сервер.
- Взаємодія:
    - HTTP-запит (request): Користувач вводить свої реєстраційні дані у веб-форму (ім'я, email, пароль), після чого клієнтська частина надсилає запит на сервер для реєстрації через POST запит на endpoint /register.

2. Серверний пристрій (Сервер)

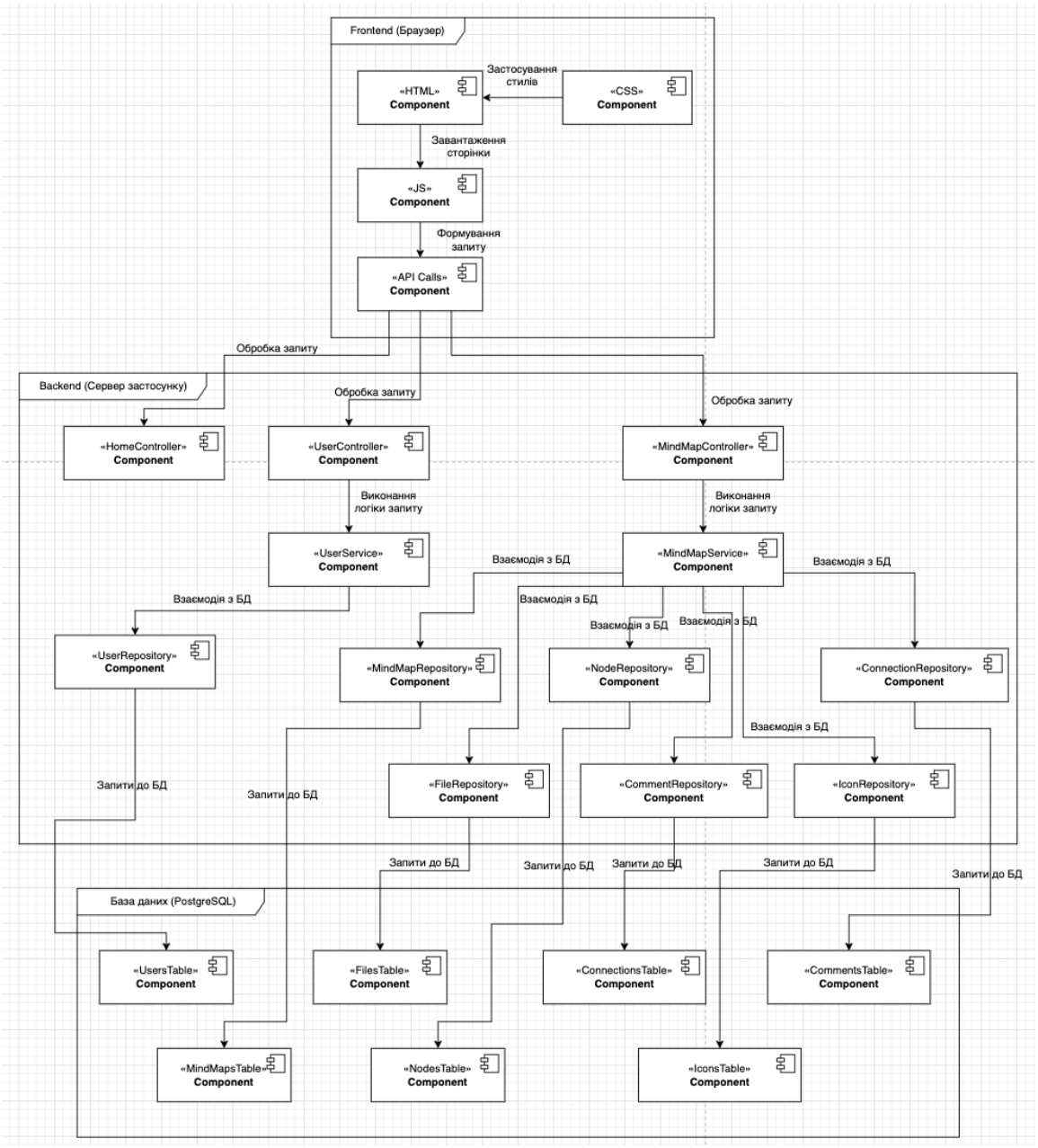
- Середовище виконання:
    - JVM: Виконує програму, написану на Java з використанням Spring Boot Framework.
    - Spring Boot Application (MindMap.jar): Основний серверний додаток, що обробляє всі запити клієнтів. Реалізований на Java із застосуванням Spring Boot.
    - application.properties: Конфігураційний файл для налаштування параметрів додатку (наприклад, підключення до бази даних).
- API Endpoints:
    - POST /register: Обробляє запити реєстрації користувачів. Отримує дані реєстрації від клієнта, передає їх на обробку сервісами для перевірки та збереження користувача в базі даних.
    - GET /login: Цей endpoint використовується для процесу входу в систему (при реєстрації користувач в той же час і входить в акаунт).
- Взаємодія:
    - Сервер обробляє HTTP-запит від клієнта, перевіряє дані користувача, передає запит на базу даних через JDBC для збереження нового користувача.

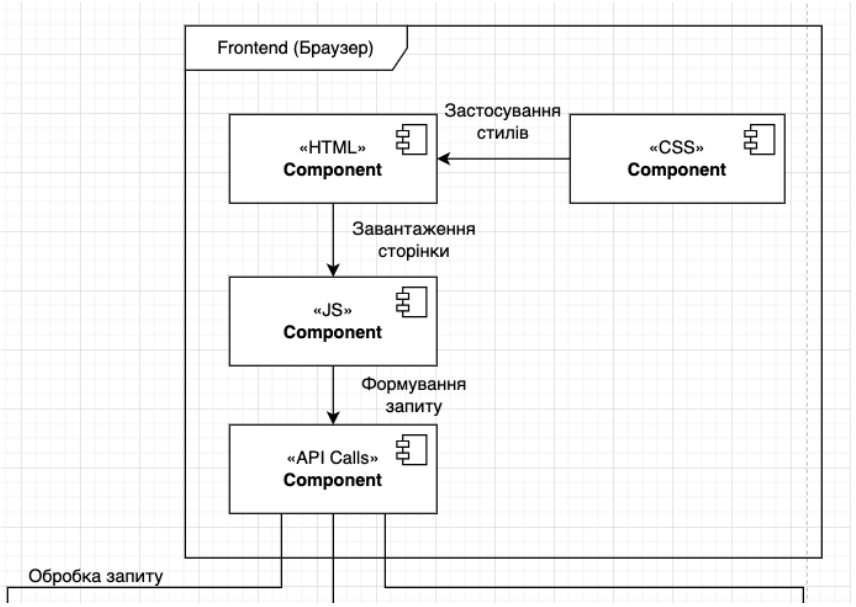
3. Сервер бази даних (PostgreSQL)

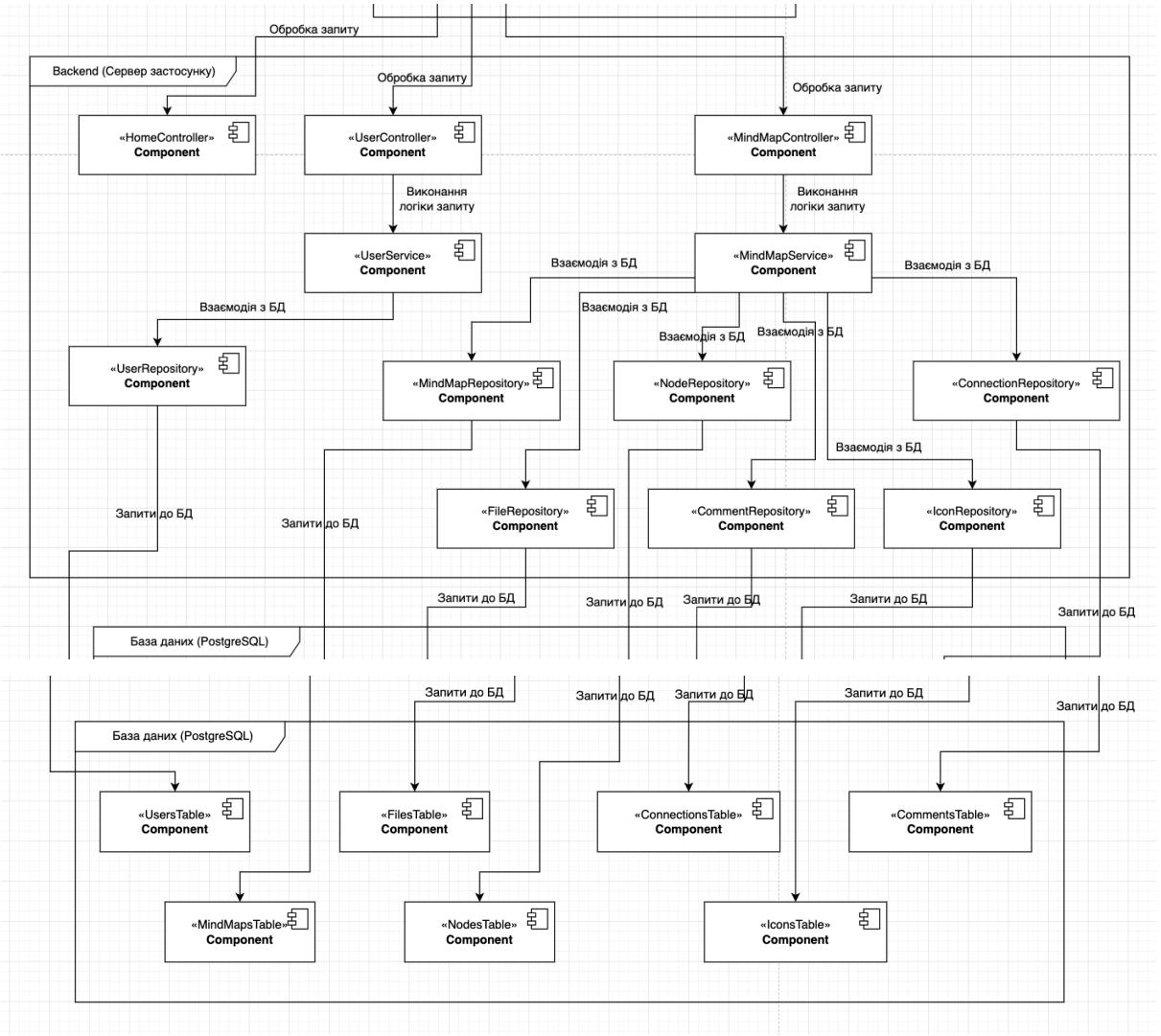
- Середовище виконання:
    - PostgreSQL: База даних, що зберігає інформацію про

користувачів та інші сутності системи.
    - postgresql.conf: Конфігураційний файл для налаштування параметрів бази даних.
    - Таблиця Users: Таблиця, в яку зберігається інформація про користувачів, що зареєстровані в системі (email, ім'я, пароль та інше).
- Взаємодія:
    - Сервер бази даних отримує запити на перевірку існуючих користувачів або на збереження нових даних. В процесі реєстрації відправляється SQL-запит для збереження нових даних користувача (JDBC SQL INSERT) у таблицю Users.

Діаграма компонентів







Frontend (Клієнтська частина):

- HTML Component – відповідає за структуру сторінки, що відображається користувачу.
- CSS Component – забезпечує візуальне оформлення інтерфейсу.
- JS Component – відповідає за інтерактивність і логіку на стороні клієнта.
- API Calls Component – формує HTTP-запити до сервера для взаємодії з бекендом.

Backend (Серверна частина):

- HomeController Component – обробляє основні запити користувача.
- UserController Component – відповідає за операції, пов'язані з користувачами (реєстрація, авторизація).
- MindMapController Component – обробляє запити, пов'язані з картами пам'яті. Оскільки дії пов’язані з вузлами, зв’язками, коментарями, файлами та іконками відносяться до карти і без карти

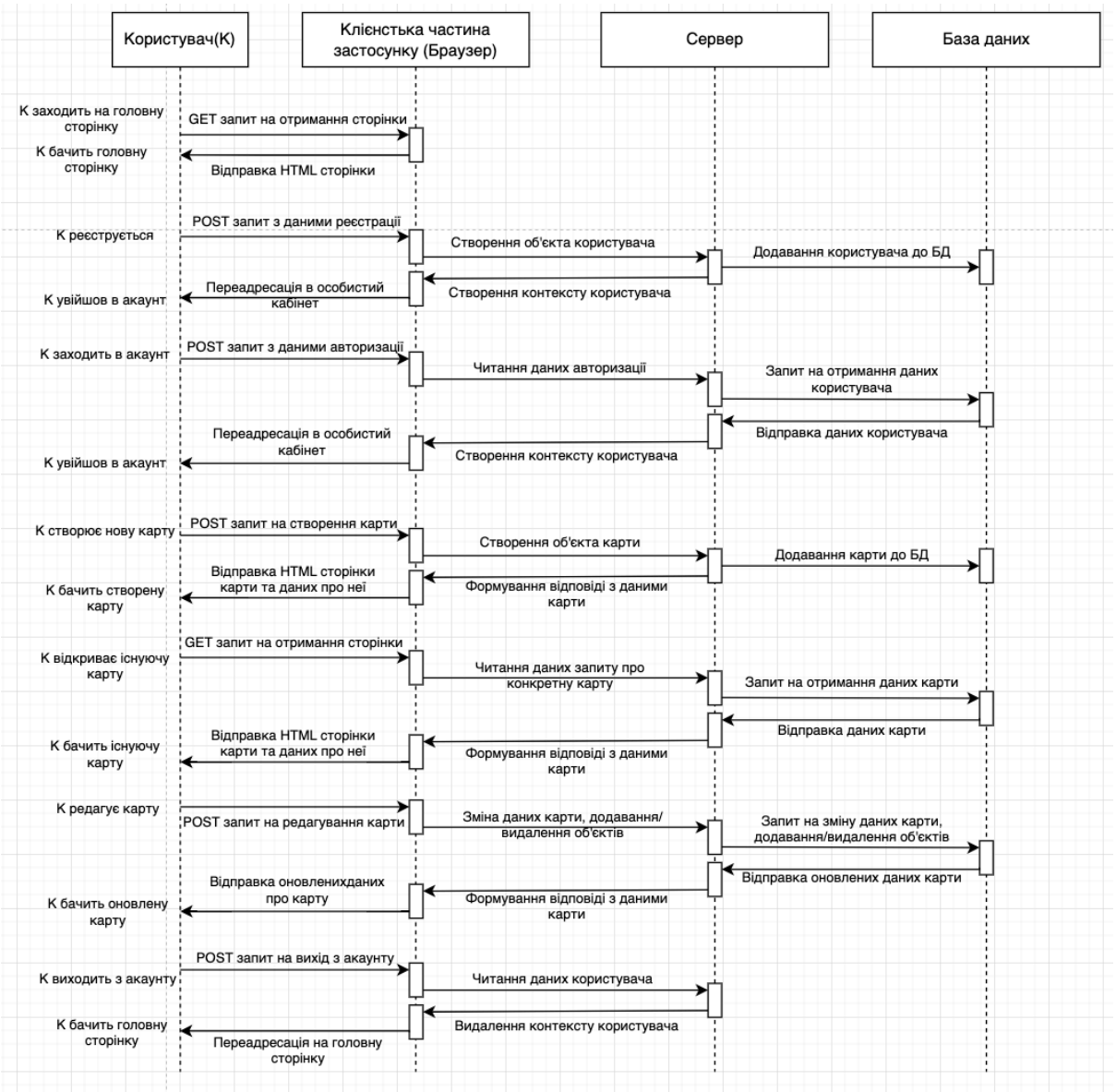
не матимуть сенсу, MindMapController об’єднує обробку запитів щодо всіх цих дій

- UserService Component – виконує бізнес-логіку, пов'язану з користувачами.
- MindMapService Component – реалізує логіку роботи з картами пам'яті. Як вже було згадано в контексті контролерів, всі дії з внутрішніми елементами карти будуть обраблятися сервісом самої карти
- UserRepository Component – інтерфейс для взаємодії з таблицею користувачів в базі даних.
- MindMapRepository Component – інтерфейс для взаємодії з таблицею карт пам’яті в базі даних.
- NodeRepository Component – інтерфейс для взаємодії з таблицею вузлів в базі даних.
- FileRepository Component – інтерфейс для взаємодії з таблицею файлів в базі даних.
- CommentRepository Component – інтерфейс для взаємодії з таблицею коментарів в базі даних.
- ConnectionRepository Component – інтерфейс для взаємодії з таблицею зв’язків в базі даних.
- IconRepository Component – інтерфейс для взаємодії з таблицею іконок в базі даних.

Репозиторії (взаємодія з БД):

- UsersTable Component – таблиця для збереження інформації про користувачів (ім'я, email, пароль).
- MindMapsTable Component – таблиця для зберігання даних про створені карти пам'яті.
- NodesTable Component – містить дані про вузли карт пам'яті.
- FilesTable Component – зберігає дані про файли, які прикріплені до карт пам'яті.
- CommentsTable Component – містить коментарі до карт пам'яті.
- ConnectionsTable Component – зберігає інформацію про зв'язки між вузлами в картах пам'яті.
- IconsTable Component – таблиця для зберігання іконок, що використовуються в картах пам'яті.

Діаграма послідовностей



Процес реєстрації та авторизації:

1. Користувач відкриває головну сторінку, надсилаючи GET запит для її отримання. Сервер відповідає, надсилаючи HTML-сторінку.
2. Користувач реєструється, заповнивши форму, після чого відправляється POST запит з даними реєстрації на сервер.
3. Сервер обробляє запит, створює об'єкт користувача та зберігає його в базі даних.
4. Після реєстрації відбувається переадресація на особистий кабінет користувача.
5. Користувач авторизується, надсилаючи POST запит з даними для входу. Сервер читає дані авторизації з бази даних і створює контекст користувача для сесії.

Робота з картами пам'яті:

1. Користувач створює нову карту пам'яті, відправляючи POST запит на сервер.
2. Сервер створює об'єкт карти пам'яті та додає її до бази даних.
3. Користувач отримує сторінку з новоствореною картою та її даними.
4. При відкритті існуючої карти пам'яті, GET запит надсилається на сервер для отримання даних карти, які зчитуються з бази даних.
5. Сервер формує відповідь і надсилає сторінку з даними карти користувачу.

Редагування карти пам'яті:

1. Користувач редагує карту пам'яті, відправляючи POST запит із новими даними.
2. Сервер обробляє зміни, вносить їх до бази даних (додавання, зміна або видалення об'єктів).
3. Після обробки змін сервер надсилає оновлені дані карт користувачу.

Вихід із системи:

1. Користувач натискає "Вийти", надсилається POST запит на вихід.
2. Сервер видаляє контекст користувача і завершує сесію, перенаправляючи користувача на головну сторінку.

Висновок: У ході виконання цієї лабораторної роботи я доповнив свої знання UML-діаграм, зокрема ознайомившись з діаграмами розгортання, компонентів та послідовностей. Я створив ці діаграми для моделювання архітектури системи, яка включає в себе процеси реєстрації, авторизації та роботи з картами пам'яті. Це дозволило мені чітко зрозуміти, як різні частини системи взаємодіють між собою на різних рівнях, від фронтенду до бази даних.

Завдяки діаграмі розгортання я візуалізував фізичне розташування компонентів і зрозумів, як клієнт, сервер та база даних співпрацюють під час обробки запитів. Діаграма компонентів допомогла мені структурувати логічні частини системи, а діаграма послідовностей показала динамічну взаємодію між ними, зокрема під час ключових процесів, таких як створення та редагування карт пам'яті.

В результаті цієї роботи я зміцнив розуміння архітектури програмного

забезпечення та отримав практичні навички побудови UML-діаграм, які допомагають ефективно планувати та описувати складні програмні системи.