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 №2

із дисципліни «Технології Розробки Програмного Забезпечення»

Тема: «ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»

Варіант-11

Виконав:
Студент групи ІА-24
Коханчук Михайло Миколайович

Перевірив:
Мягкий Михайло Юрійович

Київ-2024

ЗМІСТ

МЕТА

Метою роботи є проєктування та реалізація системи для роботи з JSON-схемами з використанням бази даних, шаблону Репозиторію та об’єктно-орієнтованого підходу. Задача полягає у створенні схеми прецедентів, що відповідає функціональним вимогам системи, розробці діаграми класів для реалізованої частини системи, виборі та детальному опрацюванні трьох прецедентів, проєктуванні основних класів системи та структури бази даних, а також реалізації класів, що використовують шаблон Репозиторію для взаємодії з базою даних. Результатом роботи стане звіт із діаграмами прецедентів, класів, вихідними кодами та структурою бази даних.

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. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторії.
3. Намалюйте діаграму класів для реалізованої частини системи.
4. Виберіть 3 прецеденти і напишіть на їх основі прецеденти.
5. Розробити основні класи і структуру системи баз даних.
6. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.
7. Підготувати звіт про хід виконання лабораторних робіт. Звіт, що подається повинен містити: діаграму прецедентів, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних

Хід роботи:

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

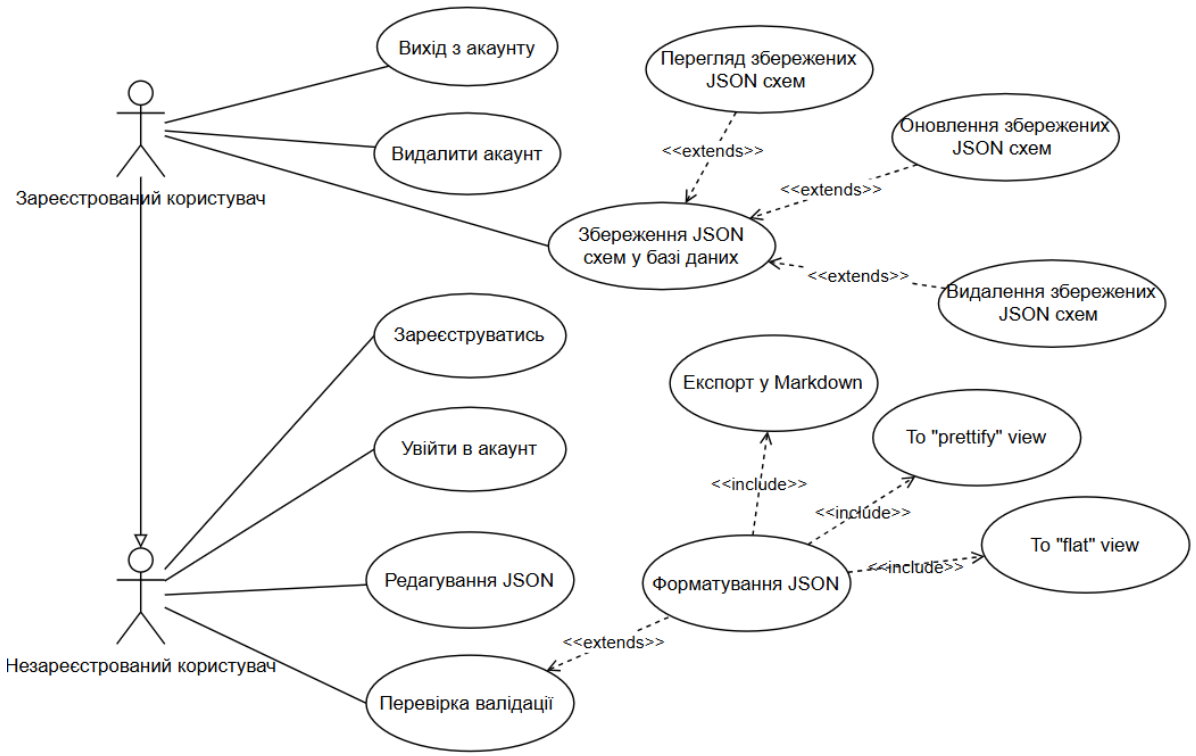

Рис. 1. Діаграма прецедентів для JSON Tool

Далі опишемо 3 прецеденти діаграми згідно завдання.

Прецедент 1: *Незареєстрований користувач реєструється*

Передумови:

- Незареєстрований користувач не має існуючого акаунта в системі.
- Незареєстрований користувач знаходиться на сторінці реєстрації.

Післяумови:

- Незареєстрований користувач стає зареєстрованим.
- Створено новий запис у базі даних користувачів.
- Незареєстрований користувач автоматично перейшов на сторінку логіну у систему після реєстрації.

Основний хід подій:

1. Незареєстрований користувач відкриває форму реєстрації.
2. Незареєстрований користувач заповнює необхідні поля (ім'я, email, пароль).
3. Незареєстрований користувач натискає кнопку "Зареєструватися".
4. Система перевіряє введені дані на коректність і унікальність email.
5. Якщо все коректно, система створює новий акаунт і авторизує користувача.
6. Незареєстрований користувач стає зареєстрованим і отримує доступ до функцій, доступних тільки для зареєстрованих користувачів.

Виключення:

- Якщо email уже використовується, система видає помилку про існуючий акаунт і пропонує увійти в систему або використати інший email.
- Якщо введені дані некоректні (наприклад, неправильний формат email або слабкий пароль), система повідомляє про помилку і пропонує виправити дан

Прецедент 2: *Зареєстрований користувач зберігає Json схему*

Передумови:

- Користувач вже зареєстрований і увійшов у систему.
- Користувач створив або завантажив JSON-схему у редакторі.

Післяумови:

- JSON-схема збережена в базі даних.
- Користувач може переглядати, оновлювати або видаляти збережену JSON-схему.

Основний хід подій:

1. Зареєстрований користувач відкриває редактор JSON-схем.
2. Користувач вносить зміни до схеми або створює нову.
3. Користувач натискає кнопку "Зберегти".
4. Система перевіряє валідність JSON-схеми.
5. Якщо схема валідна, система зберігає її у базі даних, повідомляючи користувача про успішне збереження.

Виключення:

- Якщо JSON-схема не відповідає вимогам валідності, система повідомляє про помилку і пропонує виправити її.
- Якщо виникає збій під час збереження (наприклад, проблема з базою даних), система повідомляє про помилку та пропонує спробувати пізніше.

Прецедент 3: *Користувач форматує Json схему*

Передумови:

- Користувач знаходиться на сторінці редактора JSON.

Післяумови:

- JSON-схема форматована та відображається у вибраному вигляді (prettify або flat).

Основний хід подій:

1. Користувач завантажує або створює JSON-схему в редакторі.
2. Користувач обирає один із варіантів форматування ("Prettify view" або "Flat view")
3. Користувач натискає кнопку форматування.
4. Система аналізує JSON-схему та застосовує вибраний стиль форматування.
5. Система відображає відформатовану JSON-схему користувачу.

Виключення:

- Якщо JSON-схема містить помилки (наприклад, синтаксичні), система повідомляє про помилку і пропонує виправити схему перед форматуванням.
- Якщо обраний режим форматування недоступний через технічний збій, система повідомляє про проблему і пропонує спробувати інший режим.

Побудуємо діаграму класів розробленої частини застосунку:

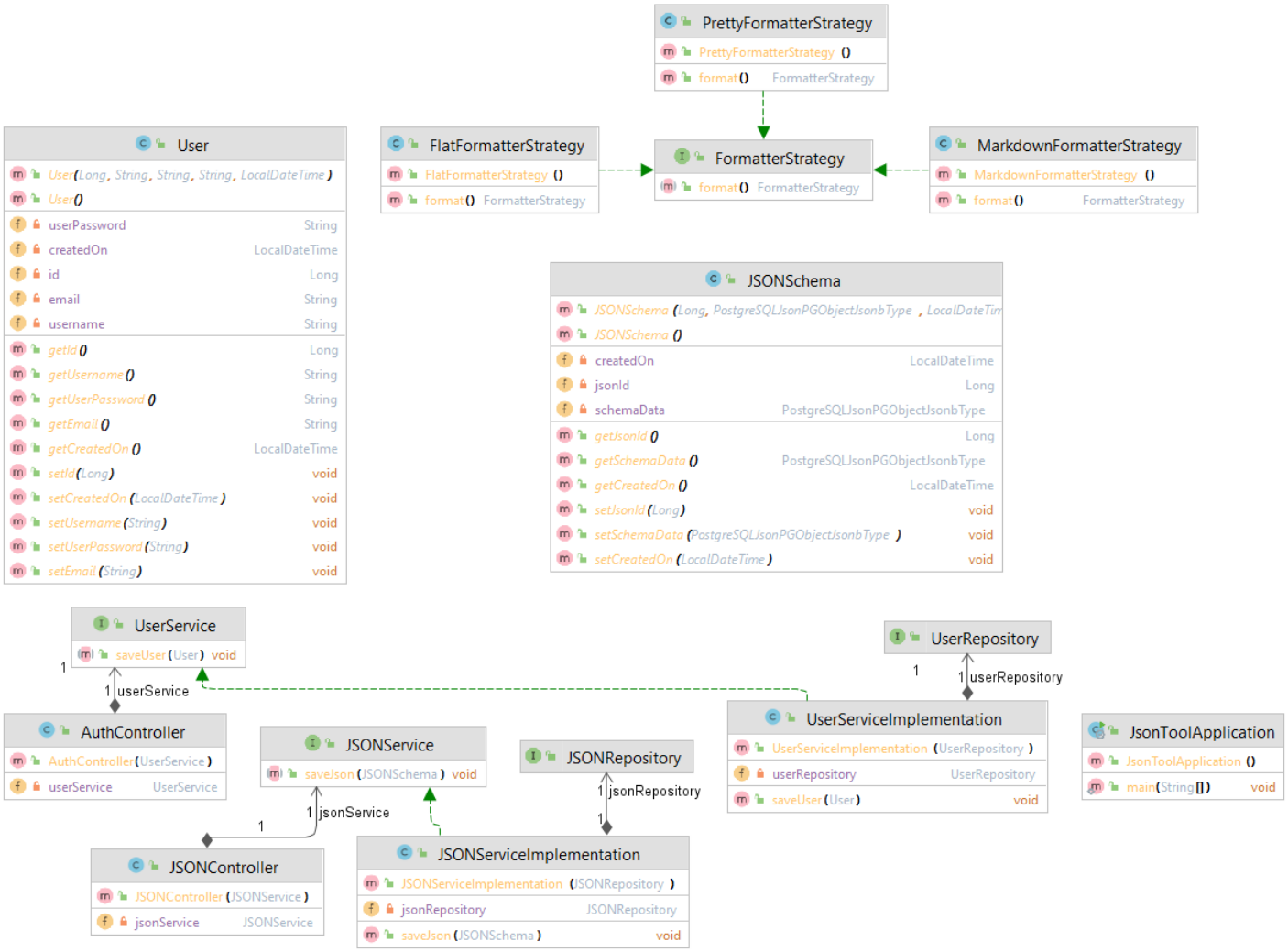

Рис. 2. Діаграма класів реалізованої частини системи

Архітектура системи

Система побудована на N-рівневій архітектурі, що передбачає чітке розділення на рівні:

Модель (Model) – містить сутності, що представляють дані(`User`, `JSONSchema`).

Контролер (Controller) – обробляє запити користувачів та передає їх у бізнес-логіку.

Сервіс – надає бізнес-логіку. В рамках рівня сервісу реалізовано патерн Стратегія для форматування JSON.

Репозиторій – надає доступ до даних.

Основні класи системи

1. User (Модель)

Поля:

id: Long – унікальний ідентифікатор.

username: String – ім'я користувача.

email: String – пошта користувача.

userPassword: String – пароль.

createdOn: LocalDateTime – дата створення.

Методи:

Гетери/сетери для всіх полів.

Конструктор для ініціалізації.

2. JSONSchema (Модель)

Поля:

jsonId: Long – ідентифікатор JSON-об'єкта.

schemaData: PostgreSQLJsonPGobjectJsonbType – дані JSON-схеми.

createdOn: LocalDateTime – дата створення.

Методи:

Гетери/сетери для доступу до полів.

Конструктор для створення JSONSchema.

3. FormatterStrategy (Інтерфейс патерну Стратегія) Інтерфейс

format(): void – метод для форматування JSON.

Реалізації:

PrettyFormatterStrategy – форматування для "prettify" view.

FlatFormatterStrategy – форматування для "flat" view.

MarkdownFormatterStrategy – експорт у Markdown.

4. UserService та UserServiceImplementation

saveUser(User user): void – зберігає користувача через UserRepository.

5. JSONService та JSONServiceImplementation

saveJson(JSONSchema jsonSchema): void – зберігає JSON-схему через JSONRepository.

6. Контролери

AuthController – отримує UserService для обробки автентифікації.

JSONController – отримує JSONService для обробки JSON-запитів.

Ключові зв'язки між класами:

Контролери інжектують сервіси.

Сервіси залежать від репозиторіїв.

Реалізації FormatterStrategy наслідують інтерфейс для форматування.

Далі розробимо структуру бази даних. Для цього побудуємо даталогічну модель бази даних:

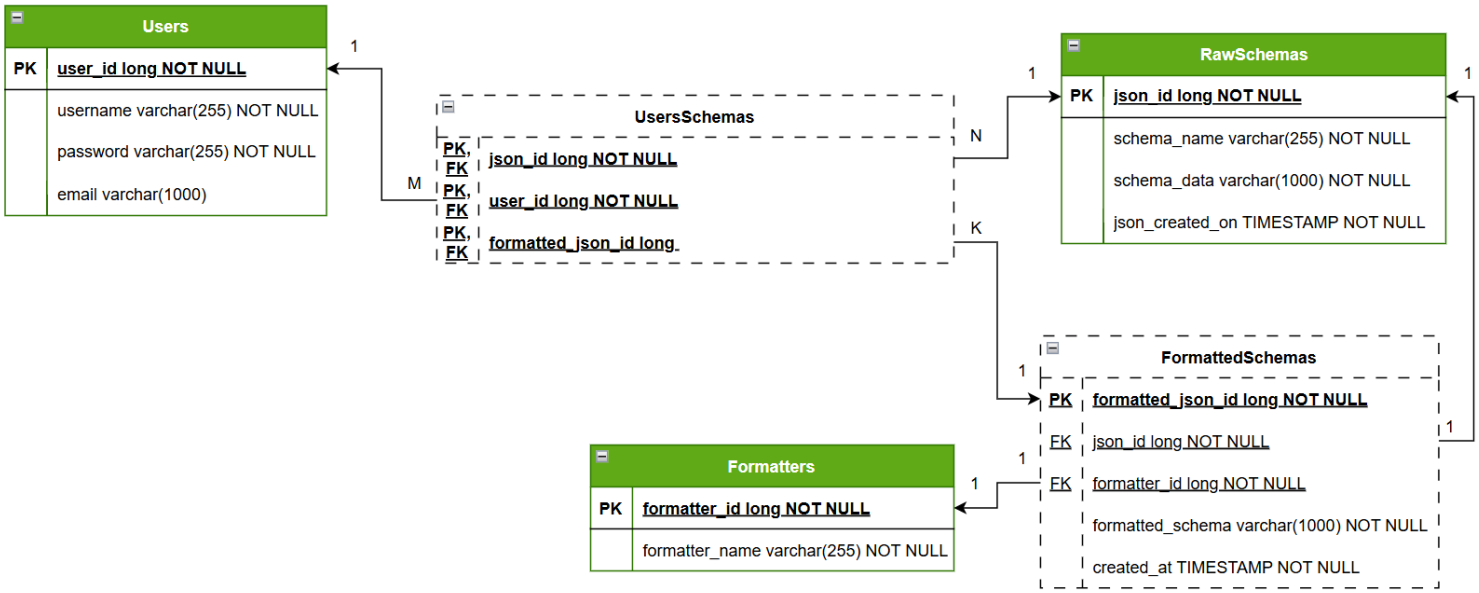

Рис. 3. Зображення структури бази даних для JSON Tool

Опис структури бази даних

База даних складається з 5 основних таблиць, що зберігають дані про користувачів, їх JSON-схеми, форматовані версії схем та історію змін.

Таблиця Users

Зберігає інформацію про користувачів.

Поле	Тип даних	Опис
user_id	long (PK)	Унікальний ідентифікатор користувача.
username	varchar(255)	Ім'я користувача.
password	varchar(255)	Пароль користувача.
email	varchar(1000)	Електронна адреса.

Таблиця JSONSchemas

Зберігає JSON-схеми користувачів та їх статус актуальності.

Поле	Тип даних	Опис
json_id	long (PK)	Унікальний ідентифікатор JSON-схеми.
schema_data	Varchar(1000)	Вміст JSON-схеми.
schema_name	Varchar(255)	Назва JSON-схеми.
json_created_on	Timestamp	Дата створення JSON-схеми.

Зв'язок:
Один користувач (Users) може мати багато JSON-схем**.**

Таблиця FormattedJSONs

Зберігає форматовані версії JSON-схем.

Поле	Тип даних	Опис
formatted_json_id	long (PK)	Унікальний ідентифікатор форматованої схеми.
json_id	long (FK)	Посилання на JSON-схему (JSONSchemas).
formatter_id	long (FK)	Посилання на форматер (Formatters).
formatted_schema	Varchar(1000)	Дані форматованої JSON-схеми.
created_at	Timestamp	Дата створення форматованої версії.

Зв'язки:

- Кожна форматована версія належить одній JSON-схемі.
- Використовується один форматер для форматування.

Таблиця Formatters

Зберігає інформацію про форматери, що використовуються для обробки JSON-схем.

Поле	Тип даних	Опис
formatter_id	long (PK)	Унікальний ідентифікатор форматера.
formatter_name	varchar(255)	Назва форматера (наприклад, Pretty, Flat).

Зв'язок:

- Один форматер може використовуватися для форматування багатьох JSON-схем.

Таблиця UsersSchemas

Асоціативна таблиця(зв'язок) між Users, FormattedJSONs та JSONSchemas

Поле	Тип даних	Опис
json_id	long (PK, FK)	Посилання на JSON-схему (JSONSchemas).
user_id	long (PK, FK)	Посилання на користувача (Users).
formatted_json_id	long (FK)	Посилання на форматовану JSON-схему.

Зв'язки:

- Один користувач може мати багато записів історії.
- Кожен запис історії пов'язаний із конкретною JSON-схемою та (опційно) форматованою версією.

Код реалізованої частини системи:

https://github.com/mykh3398/trpz_labs/tree/64fdc6feec8d636455d0aefda5cc8d0fd0d266ed/JsonTool

ВИСНОВОК

У результаті виконання роботи було спроєктовано та проаналізовано структуру бази даних для зберігання JSON-схем, їх форматованих версій. Основною метою було забезпечити логічну цілісність даних, оптимізувати зв'язки між таблицями та спростити доступ до актуальних версій схем. Також було розроблено діаграму класів та прецедентів даної системи.