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

ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ. ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ.

JSON Tool

Виконав
студент групи ІА–22:
Щаблевський Д. Е.
Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Зміст

Зміст 2
Хід роботи 3
Теоретичні відомості 3
Реалізація частини функціональної системи 4
Діаграма послідовностей 5
Діаграма розгортання 6
Діаграма компонентів 7
Висновок 8

Тема: Діаграма розгортання. Діаграма компнентів. Діаграма взаємодій та послідовностей.

Мета: Проаналізувати тему. Розробити діаграму розгортання, діаграму компонентів та діаграму послідовностей для проектованої системи.

Хід роботи

..28 JSON Tool (ENG) (strategy, command, observer, template method, flyweight)

Display JSON schema with syntax highlight. Validate JSON schema and display errors. Create user friendly table\list box\other for read and update JSON schema properties metadata (description, example, data type, format, etc.). Auto save\restore when edit, maybe history. Can check JSON value by schema (Put schema and JSON = valid\invalid, display errors). Export schema as markdown table. JSON to "flat" view.

Теоретичні відомості

У лабораторній роботі 3 розглядається створення UML-діаграм, які використовуються для моделювання фізичного розгортання, компонування та взаємодії елементів програмної системи.

1. **Діаграма розгортання (Deployment Diagram)** відображає фізичне розташування програмного забезпечення на апаратних вузлах. Основними елементами є вузли, які представляють фізичні пристрої або середовища виконання. Вузли з'єднуються зв’язками, що демонструють шляхи передачі даних.
2. **Діаграма компонентів (Component Diagram)** ілюструє розподіл системи на окремі модулі чи компоненти. Вона показує зв’язки між компонентами, сприяючи структуризації коду та забезпечуючи можливість його повторного використання.

3. **Діаграма послідовностей (Sequence Diagram)** моделює хронологію взаємодії об'єктів. Вона відображає, які об'єкти беруть участь у взаємодії та в якому порядку передаються повідомлення між ними.

Ці діаграми дозволяють детально вивчити архітектуру системи, оптимізувати її структуру та спланувати ефективне розгортання і подальший розвиток.

Реалізація частини функціональної системи

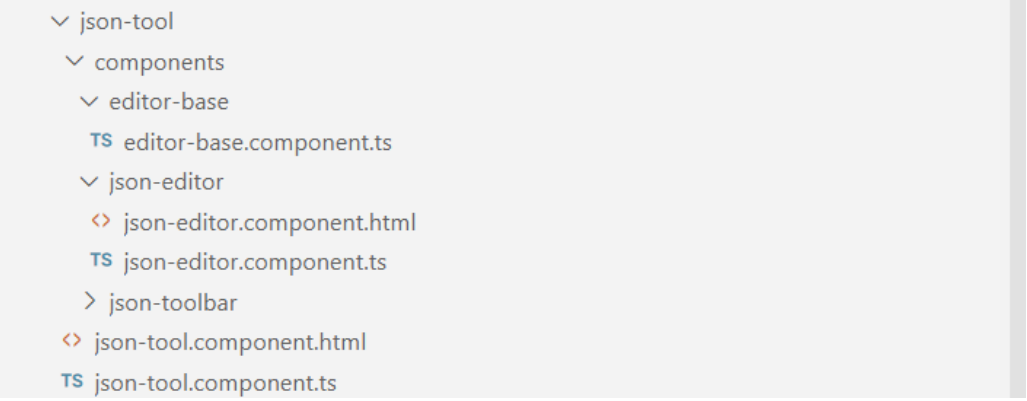

Рисунок 1. – Реалізована частина проекту

Реалізовані компоненти для редагування JSON:

- editor-base.component.ts
  Базовий компонент, який служить основою для редактора.
- json-editor/
  - json-editor.component.html
    HTML-шаблон компонента JSON-редактора.
  - json-editor.component.ts
    Типовий файл для логіки компонента JSON-редактора.
- json-toolbar/
  Директорія для панелі інструментів редактора JSON.
- json-tool.component.html
  Головний HTML-шаблон для JSON-інструменту.
- json-tool.component.ts
  Головний файл логіки для JSON-інструменту.

Діаграма послідовностей

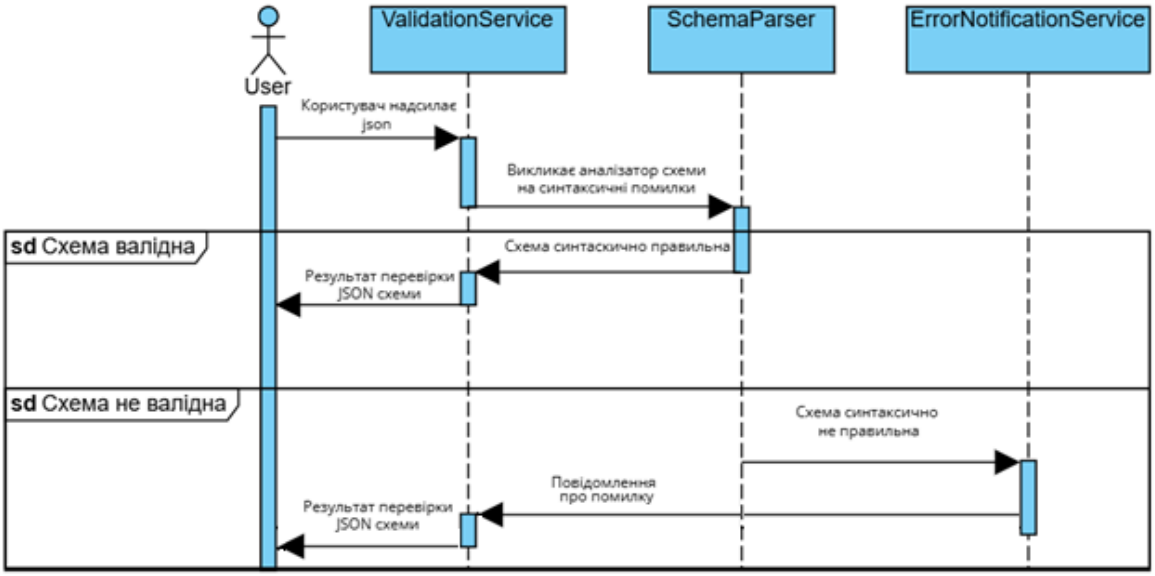

Рисунок 2 - діаграма послідовностей для сценарію «Перевірка JSON»

Діаграма послідовностей описує процес перевірки JSON-схеми на синтаксичну правильність. Взаємодія відбувається між користувачем та трьома компонентами системи: ValidationService, SchemaParser і ErrorNotificationService.

1. Користувач надсилає JSON-схему до системи через редактор.
2. ValidationService передає схему в SchemaParser для аналізу.
3. SchemaParser виконує перевірку:
    - Якщо схема є синтаксично правильною, результат передається назад у ValidationService, який повідомляє користувача про успішну перевірку.
    - Якщо виявлено синтаксичні помилки, SchemaParser повертає інформацію про помилки у схему.

4. У разі помилок ValidationService викликає ErrorNotificationService, який формує повідомлення про проблему та надсилає його користувачеві.

Діаграма розгортання

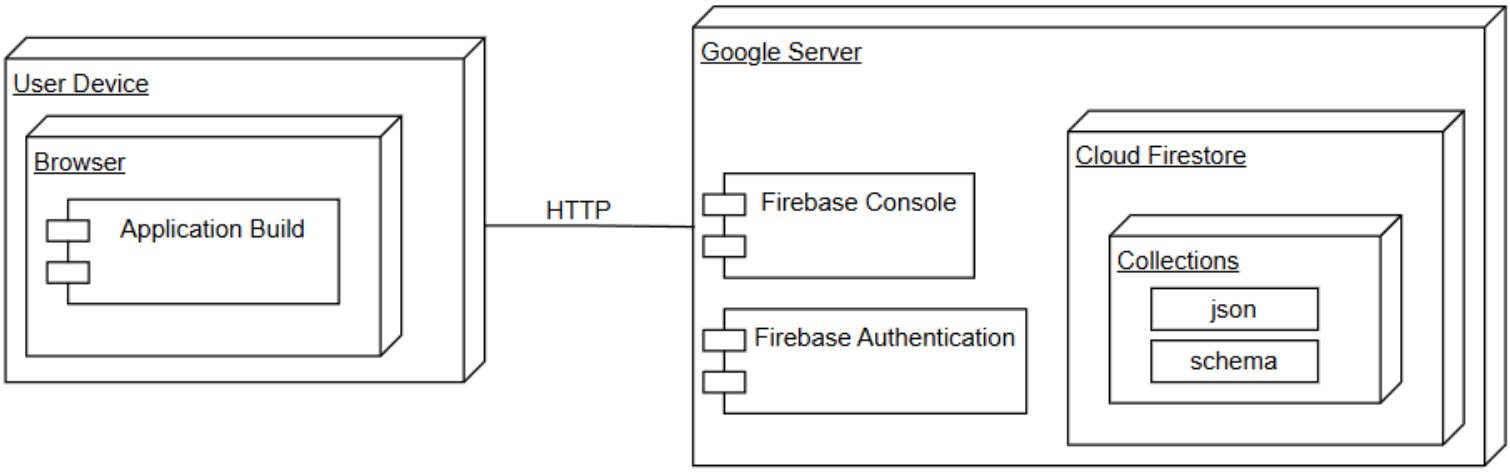

Рисунок 3. - Діаграма розгортання

Дана діаграма складається з двох основних компонентів:

1. **Клієнтський пристрій (User Device)**: Включає браузер, у якому розміщена збірка додатка (Application Build). Клієнтський пристрій взаємодіє із сервером через HTTP-запити.
2. **Google Server**: Містить кілька підсистем:
    - **Firebase Console**: використовується для управління конфігурацією та налаштуваннями додатка.
    - **Firebase Authentication**: відповідає за автентифікацію користувачів.
    - **Cloud Firestore**: база даних, яка організована у вигляді колекцій, що зберігають JSON-об'єкти та їх схеми (schema).

Ці компоненти взаємодіють між собою, забезпечуючи роботу веб-додатка, збереження даних у хмарному сховищі та автентифікацію користувачів.

Діаграма компонентів

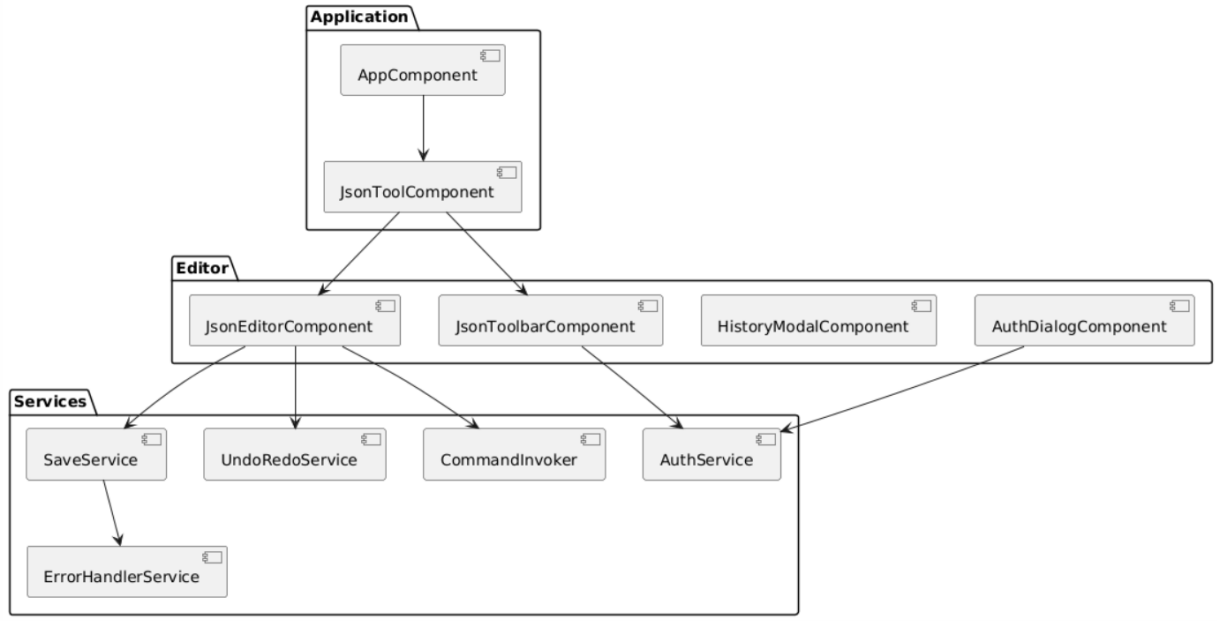

Рисунок. 4. – Діаграма компонентів

Ця діаграма компонентів ілюструє структуру застосунку, розподіленого на три основні рівні:

1. **Application**: Представляє основний інтерфейс користувача. Містить AppComponent (головний компонент), який взаємодіє з JsonToolComponent для надання доступу до інструментів роботи з JSON.
2. **Editor**: Центральна частина для редагування JSON. Включає:
    - JsonEditorComponent для редагування вмісту JSON.
    - JsonToolbarComponent для доступу до функціональних кнопок.
    - HistoryModalComponent для перегляду історії змін.
    - AuthDialogComponent для авторизації користувачів.
3. **Services**: Логіка та функціональні сервіси застосунку:

- SaveService: Відповідає за збереження даних.
- UndoRedoService: Реалізує функціонал скасування та повторення дій.
- CommandInvoker: Забезпечує виконання команд.
- AuthService: Керує авторизацією користувачів.
- ErrorHandlerService: Обробляє помилки, які виникають у системі.

Компоненти на діаграмі пов’язані через стрілки, що відображають їх взаємодію між собою та із сервісами.

Посилання на репозиторій проекту:

https://github.com/neodavis/json-tool

Висновок:
У ході виконання лабораторної роботи було досліджено основи створення діаграм розгортання, компонентів і послідовностей, які є важливими інструментами для моделювання програмних систем. Створені діаграми дозволили детально проаналізувати архітектуру системи, візуалізувати її структуру, а також продемонструвати логіку роботи та взаємодії між її компонентами. Це сприяло кращому розумінню принципів організації системи, оптимізації її функціональних процесів та забезпеченню узгодженої інтеграції між усіма елементами. Робота підтвердила значення UML-діаграм як ефективного засобу для проектування, розробки та підтримки програмного забезпечення.