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

Тема: Основи проектування розгортання.

Виконала
студентка групи ІА–32:
Слюсарева А.А.

Київ 2025

ЗМІСТ

ЛАБОРАТОРНА РОБОТА № 3

Тема: Основи проектування розгортання.

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

3.1. Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу.

• Розробити діаграму компонентів для проєктованої системи.

• Розробити діаграму розгортання для проєктованої системи.

• Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.

• На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму розгортання з описом, діаграму компонентів системи з описом, діаграми послідовностей, а також вихідний код системи, який було додано в цій лабораторній роботі.

3.2. Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграма розгортання показує фізичне розташування системи та відображає, на якому обладнанні працюють програмні компоненти. Основними елементами є вузли (node), які можуть бути:

- Пристроями (device) – фізичне обладнання: ПК, сервер, мобільний пристрій.
- Середовищами виконання (execution environment) – програмні контейнери, наприклад ОС, вебсервер.

Між вузлами зображують зв’язки, які можуть мати назви (HTTP, IPC) та множинність.

Всередині вузлів можуть міститися артефакти – фізичні файли: .exe, .dll, .jar, скрипти, конфігурації, дані.

Діаграми розгортання бувають двох типів:

- Описові – без прив’язки до конкретного обладнання.
- Екземплярні – відображають реальні сервери, ПК та розгорнуті на них артефакти.

Діаграма компонентів (Component Diagram)

Діаграма компонентів показує структуру системи, поділену на функціональні модулі, та залежності між ними.

Існують три види:

- Логічні – показують логічні модулі та їхню взаємодію.
- Фізичні – відображають реальні файлові компоненти (.exe, .dll) та залежності між ними.

- Виконувані – показують виконувані файли, сторінки, бази даних, таблиці.

Такі діаграми застосовують для:
• візуалізації структури коду;
• формування виконуваного варіанта системи;
• аналізу залежностей та можливості повторного використання коду;
• опису фізичної структури даних.

Діаграма послідовностей (Sequence Diagram)

Діаграма послідовностей моделює взаємодію між об’єктами у хронологічному порядку й показує, які повідомлення вони передають одне одному.

Основні елементи:

- Актори – користувачі або зовнішні системи.
- Об’єкти/класи – учасники взаємодії з лініями життя.
- Повідомлення – виклики методів або відповіді (синхронні, асинхронні).
- Активності – періоди виконання операцій.
- Контрольні структури – блоки alt, loop для умов та циклів.

Етапи побудови:

1. Визначення акторів та об’єктів.
2. Створення ліній життя.
3. Визначення послідовності повідомлень.
4. Додавання умов чи циклів за потреби.

Діаграми послідовностей використовують для моделювання бізнес-процесів, проектування архітектури та підготовки тестових сценаріїв.

3.3 Хід роботи

• Розробити діаграму компонентів для проєктованої системи.

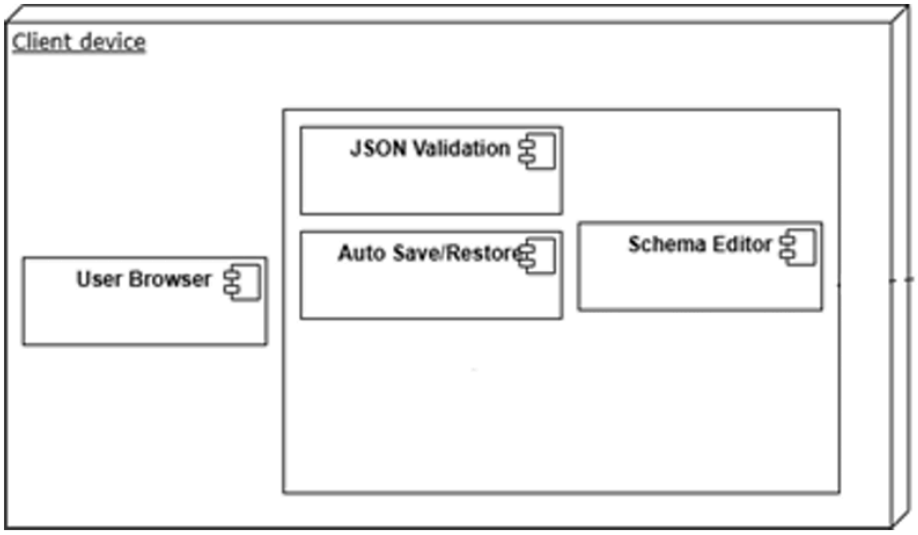

Рис 3.3.1 Діаграма компонентів

Основні модулі системи

1. JSON Validation

   Модуль відповідає за перевірку коректності JSON-даних відповідно до визначеної JSON-схеми. Він виявляє синтаксичні помилки, невідповідності типів, пропущені обов’язкові поля та інші неточності, одразу інформуючи користувача про знайдені помилки. Це забезпечує високу точність даних та запобігає виникненню збоїв під час подальшої обробки.
2. Auto Save / Restore

   Система автоматичного збереження забезпечує постійне фіксування

змін користувача без необхідності ручного збереження. У разі раптового закриття браузера, оновлення сторінки або збою пристрою редактор автоматично відновлює останній стан роботи. Це підвищує надійність та захищає користувача від втрати важливої інформації.

3. Schema Editor

   Інструмент для створення, редагування та налаштування JSON-схем. Завдяки візуальному та структурованому підходу Schema Editor дозволяє легко формувати моделі даних, додаючи нові властивості, визначаючи типи, обмеження, вкладені структури та інші параметри, що використовуються модулем валідації.

• Розробити діаграму розгортання для проєктованої системи.

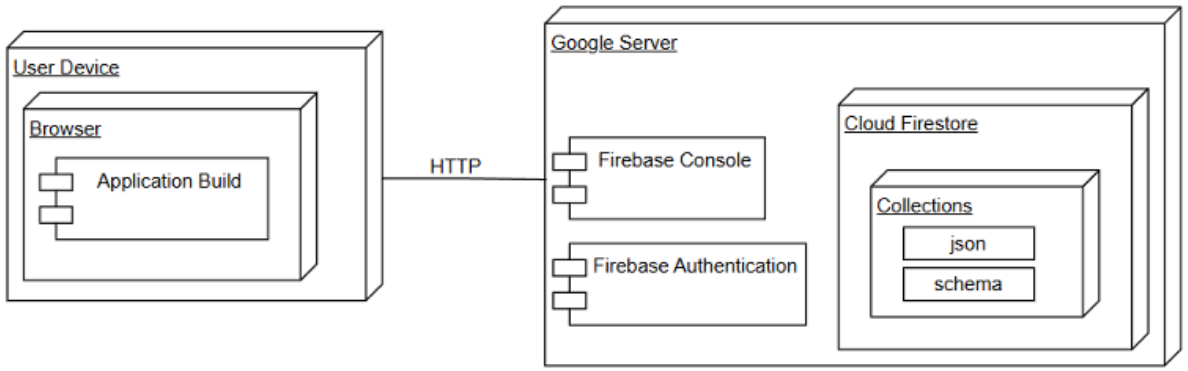

Рис 3.3.2 Діаграма розгортання

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:

   2.1 Firebase Console: використовується для управління конфігурацією та налаштуваннями додатка.

2.2. Firebase Authentication: відповідає за автентифікацію користувачів.

2.3. 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.

- Розробити діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.

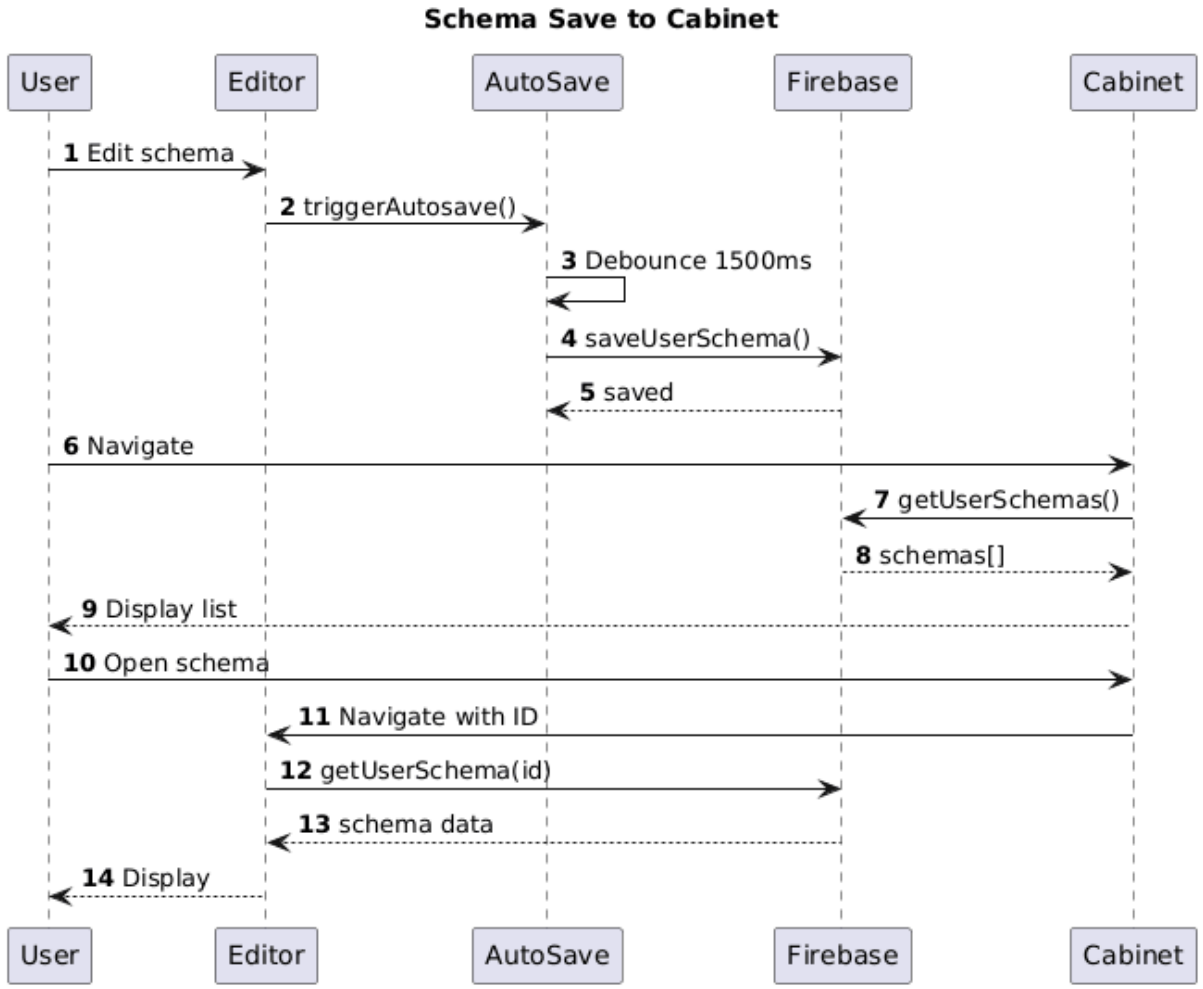

Рис 3.3.3 Діаграма послідовностей

Діаграма послідовностей описує процес автоматичного збереження схеми, перегляду списку схем у кабінеті та завантаження конкретної схеми для

редагування. Взаємодія відбувається між користувачем (User) та чотирма компонентами системи: Editor, AutoSave, Firebase (база даних) та Cabinet.

1. Користувач редагує схему в інтерфейсі редактора (Editor).
2. Editor ініціює процес автозбереження, викликаючи функцію triggerAutosave() у компоненті AutoSave.
3. AutoSave обробляє запит:
    - Виконує затримку (Debounce) у 1500 мс, щоб уникнути надмірної кількості запитів.
    - Викликає saveUserSchema() для збереження даних у Firebase.
    - Отримує підтвердження saved від Firebase про успішне збереження.
4. Користувач переходить (Navigate) до розділу Cabinet.
5. Cabinet завантажує дані:
    - Звертається до Firebase із запитом getUserSchemas() для отримання списку схем.
    - Отримує масив схем (schemas[]) від Firebase.
    - Відображає список схем користувачеві (Display list).
6. Користувач відкриває схему (Open schema) зі списку.
7. Система виконує перехід до редагування:
    - Cabinet перенаправляє до Editor, передаючи ідентифікатор схеми (Maps with ID).
    - Editor запитує дані конкретної схеми у Firebase через метод getUserSchema(id).
    - Firebase повертає дані схеми (schema data), які Editor відображає користувачеві.

3.4 Висновок

В ході виконання лабораторної роботи було спроєктовано діаграми розгортання та компонентів для системи що проєктується, а також розроблено діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі

3.5. Питання до лабораторної роботи

1. **Що собою становить діаграма розгортання?**
   Діаграма, що показує фізичне розміщення програмних компонентів на апаратних вузлах.

2. **Які бувають види вузлів на діаграмі розгортання?**
   Апаратні вузли, віртуальні вузли, вузли середовищ виконання, сховища даних.

3. **Які бувають зв’язки на діаграмі розгортання?**
   Комунікаційні шляхи та залежності між вузлами.

4. **Які елементи присутні на діаграмі компонентів?**
   Компоненти, інтерфейси, порти, артефакти, пакети, залежності.

5. **Що становлять собою зв’язки на діаграмі компонентів?**
   Вони показують взаємодію компонентів та використання інтерфейсів.

6. **Які бувають види діаграм взаємодії?**
   Послідовностей, комунікації, таймінг-діаграми, огляду взаємодій.

7. **Для чого призначена діаграма послідовностей?**
   Для відображення порядку повідомлень між об’єктами у часі.

8. **Які ключові елементи можуть бути на діаграмі послідовностей?**
   Учасники, повідомлення, активності, відповіді, фрагменти (alt, loop, opt).

9. **Як діаграми послідовностей пов’язані з діаграмами варіантів використання?**
   Вони деталізують сценарії, описані у варіантах використання.

10. **Як діаграми послідовностей пов’язані з діаграмами класів?**
    Вони показують динамічну взаємодію між об’єктами, що базується на статичній структурі класів.