Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота №3

із дисципліни «Технології розроблення програмного забезпечення»
Тема**:** «Діаграма розгортання, компонентів, взаємодії та послідовностей»

Виконав:
Студент групи ІА-24
Боднар А. Д.

Перевірив:
Мягкий М.Ю.

Київ-2024

Завдання:
1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми

Хід роботи

Теоретичні відомості

Діаграма розгортання (Deployment Diagram):

- Діаграма розгортання показує фізичне розташування програмних компонентів на апаратних пристроях. Вона демонструє, як і де програмні модулі встановлені в інфраструктурі, зображаючи пристрої (наприклад, сервери та клієнтські машини), середовища виконання, з'єднання між ними, а також програмні артефакти, які розгорнуті на цих пристроях. Це важливо для розуміння, як система працює в реальних умовах і які ресурси використовує для обробки даних.

Діаграма компонентів (Component Diagram):

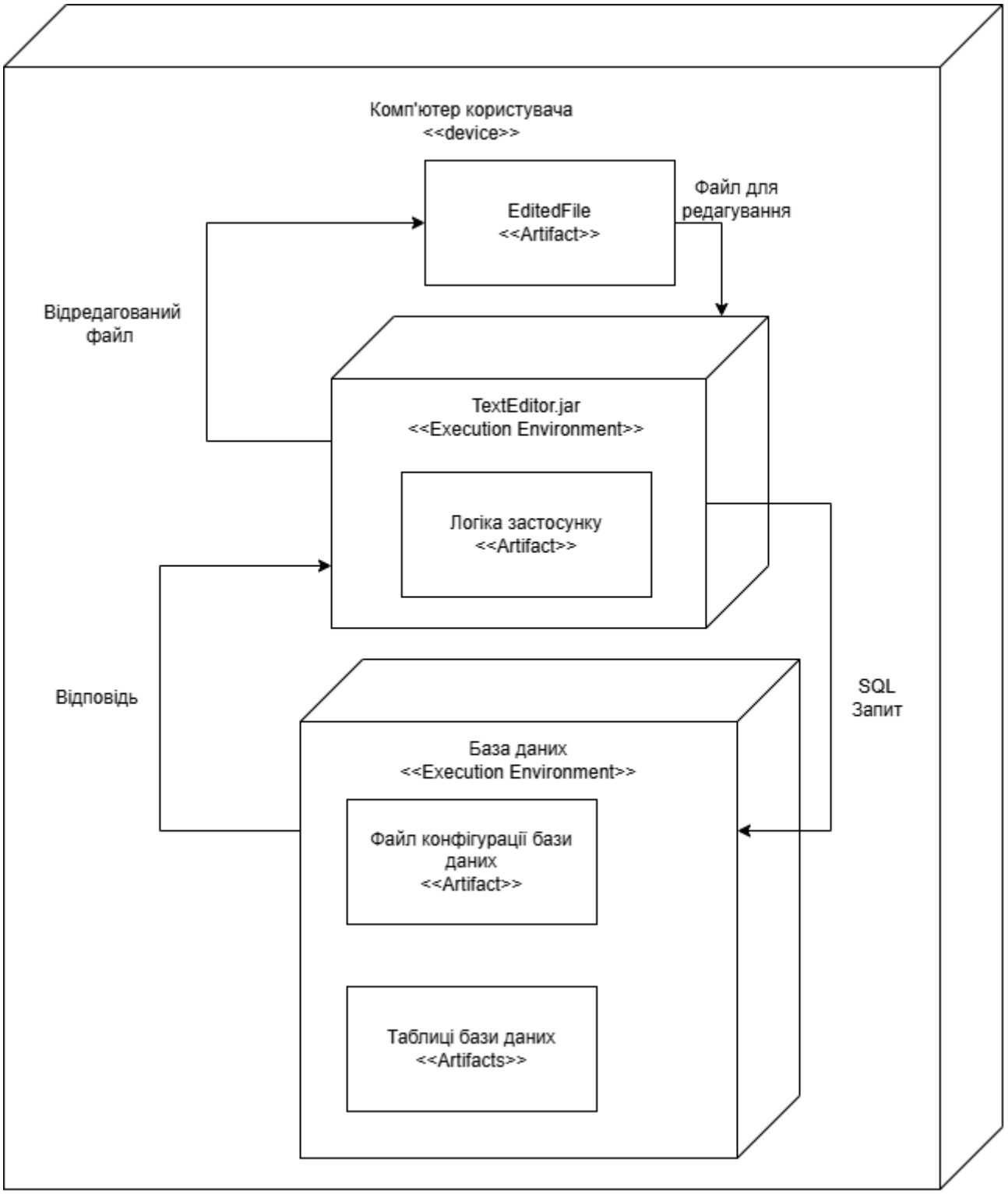
- Діаграма компонентів зображає структуру програмного забезпечення на рівні модулів або компонентів, показуючи взаємодії та залежності між ними. Вона описує, як різні частини програми (наприклад, контролери, сервіси, репозиторії) співпрацюють для реалізації функцій системи. Діаграма компонентів використовується для моделювання логічної структури коду та зрозумілого розподілу функціональності, що полегшує розробку, тестування та підтримку системи.

Діаграма послідовностей (Sequence Diagram):

- Діаграма послідовностей показує порядок виконання дій між учасниками або компонентами системи у вигляді послідовності повідомлень. Вона відображає, які об'єкти взаємодіють між собою, які повідомлення передаються, і в якому порядку. Це допомагає візуалізувати сценарій використання системи, розуміти взаємодію між компонентами в реальному часі, а також деталізувати логіку для реалізації конкретних процесів або функцій.

Тема “Текстовий редактор”

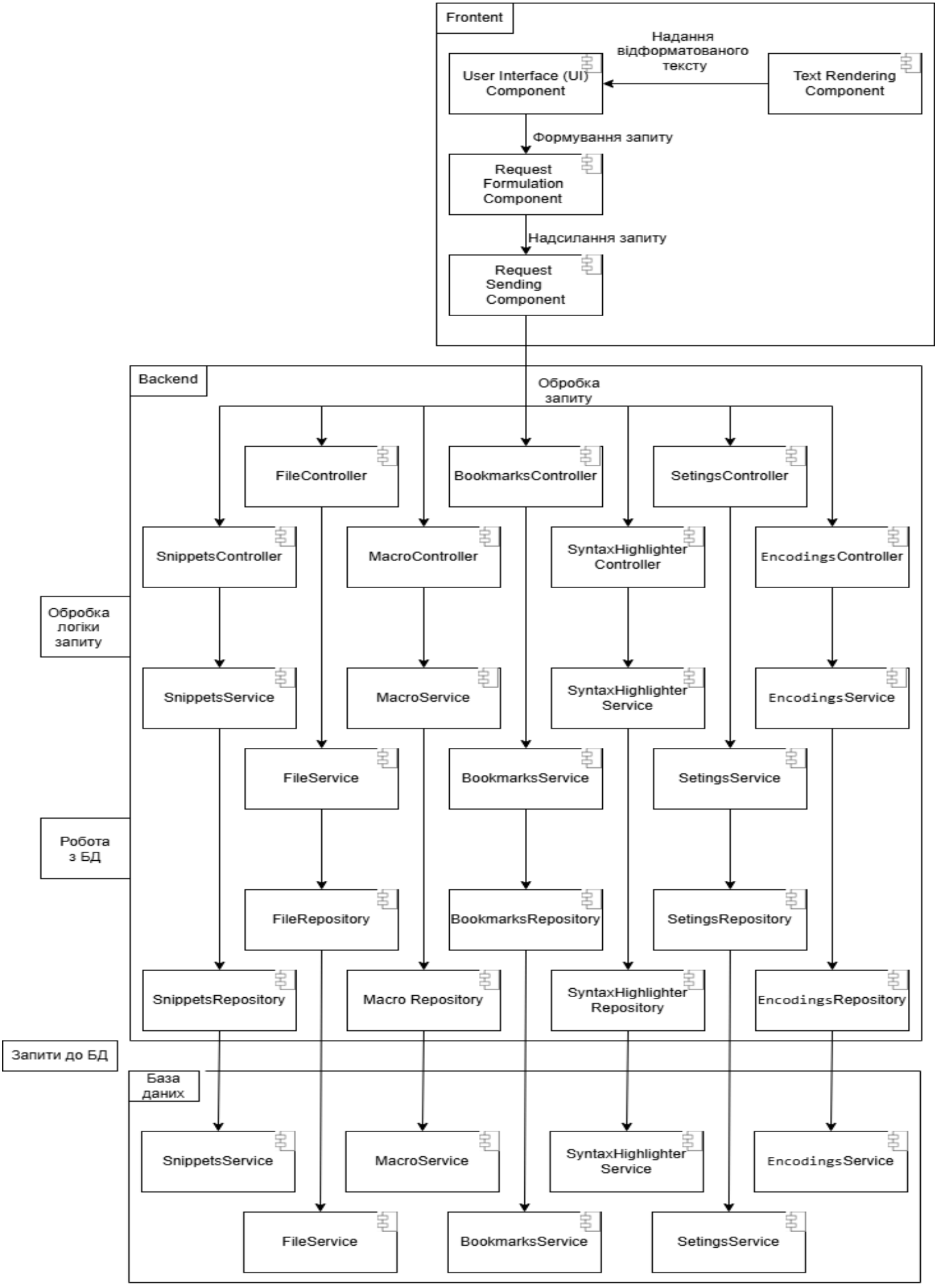
Діаграма розгортання



Опис структури діаграми розгортання:

1. **Комп'ютер користувача** (<<device>>):
    - Основний пристрій, де користувач взаємодіє з текстовим редактором.
    - Містить артефакт EditedFile (<<Artifact>>), який представляє відредагований файл користувача.
    - Файл для редагування передається до текстового редактора для обробки.
2. **TextEditor.jar** (<<Execution Environment>>):
    - Виконавче середовище для запуску програми текстового редактора.
    - Містить артефакт Логіка застосунку (<<Artifact>>), який відповідає за обробку дій користувача, виконання логіки програми, формування SQL-запитів до бази даних та повернення результатів.
3. **База даних** (<<Execution Environment>>):
    - Виконавче середовище для управління збереженими даними текстового редактора.
    - Містить два ключові артефакти:
        - Файл конфігурації бази даних (<<Artifact>>): Використовується для налаштування параметрів підключення до бази даних.
        - Таблиці бази даних (<<Artifact>>): Зберігають дані, такі як сніпети, історія редагування та інші інформаційні елементи.
4. **Зв’язки між компонентами**:
    - Файл для редагування: Передається з пристрою користувача до текстового редактора (TextEditor.jar) для обробки.
    - SQL-запит: Логіка застосунку надсилає SQL-запити до бази даних для отримання або оновлення даних.
    - Відповідь: База даних надсилає результати запитів (записи з таблиць чи підтвердження операцій) назад до Логіки застосунку.
    - Відредагований файл: Передається від текстового редактора назад до користувача після завершення роботи.

Діаграма компонентів



Frontend(Користувацький інтерфейс)

1. **Text Rendering Component**
    - **Опис**: Цей компонент відповідає за обробку та відображення тексту у вікні редактора. Він забезпечує коректне відображення тексту, застосування стилів і підсвітки синтаксису (якщо це необхідно). Передає відформатований текст до **User Interface (UI) Component**.
2. **User Interface (UI) Component**
    - **Опис**: Основний компонент інтерфейсу, що забезпечує взаємодію користувача із системою. Відображає вікна, меню та елементи керування, надає можливість запуску різних функцій редактора (як-от робота з файлами, закладками, сніплетами).
3. **Request Formulation Component**
    - **Опис**: Компонент формування запиту. Відповідає за збір даних, необхідних для виконання певної дії (наприклад, редагування, збереження файлів, вставка сніпетів), і формування запиту, який передається до **Request Sending Component**.
4. **Request Sending Component**
    - **Опис**: Компонент надсилання запиту. Після формування запиту передає його до відповідного контролера в бекенді для подальшої обробки, виконуючи роль зв’язку між фронтендом і бекендом.

Backend

Controllers (7 контролерів)

- **Опис для всіх контролерів**: Кожен контролер обробляє запити, що надходять із фронтенду, і передає їх у відповідний сервіс для виконання бізнес-логіки. Контролери забезпечують логічний поділ відповідно до сутностей у базі даних:

- **FileController**: обробляє запити для роботи з файлами.
- **SnippetController**: обробляє запити для роботи зі сніплетами.
- **MacroController**: обробляє запити для виконання макросів.
- **BookmarkController**: обробляє запити для роботи з закладками.
- **SyntaxHighlighterController**: обробляє запити для підсвітки синтаксису.
- **EncodingController**: обробляє запити для роботи з кодуванням файлів.
- **SettingsController**: обробляє запити для управління налаштуваннями редактора.

Services (7 сервісів)

- **Опис для всіх сервісів**: Кожен сервіс реалізує бізнес-логіку для своєї сутності та взаємодіє з відповідним репозиторієм для отримання та збереження даних у базі даних:
  - **FileService**: логіка для обробки файлів (відкриття, збереження, закриття).
  - **SnippetService**: логіка для роботи зі сніплетами (додавання, видалення, оновлення).
  - **MacroService**: логіка для управління макросами (виконання, запис).
  - **BookmarkService**: логіка для роботи з закладками (створення, видалення, редагування).
  - **SyntaxHighlighterService**: логіка для обробки та застосування підсвітки синтаксису.
  - **EncodingService**: логіка для роботи з кодуванням файлів (читання та перетворення кодувань).
  - **SettingsService**: логіка для роботи з налаштуваннями текстового редактора.

Repositories (7 репозиторіїв)

- **Опис для всіх репозиторіїв**: Кожен репозиторій відповідає за збереження та отримання даних у відповідній таблиці бази даних. Репозиторії служать проміжним шаром між сервісами та базою даних, забезпечуючи доступ до інформації:
  - **FileRepository**: взаємодія з таблицею файлів.
  - **SnippetRepository**: збереження та отримання сніпетів.
  - **MacroRepository**: збереження та отримання макросів.
  - **BookmarkRepository**: робота із закладками.
  - **SyntaxHighlighterRepository**: отримання даних для підсвітки синтаксису.
  - **EncodingRepository**: збереження даних про кодування.
  - **SettingsRepository**: збереження та отримання налаштувань.

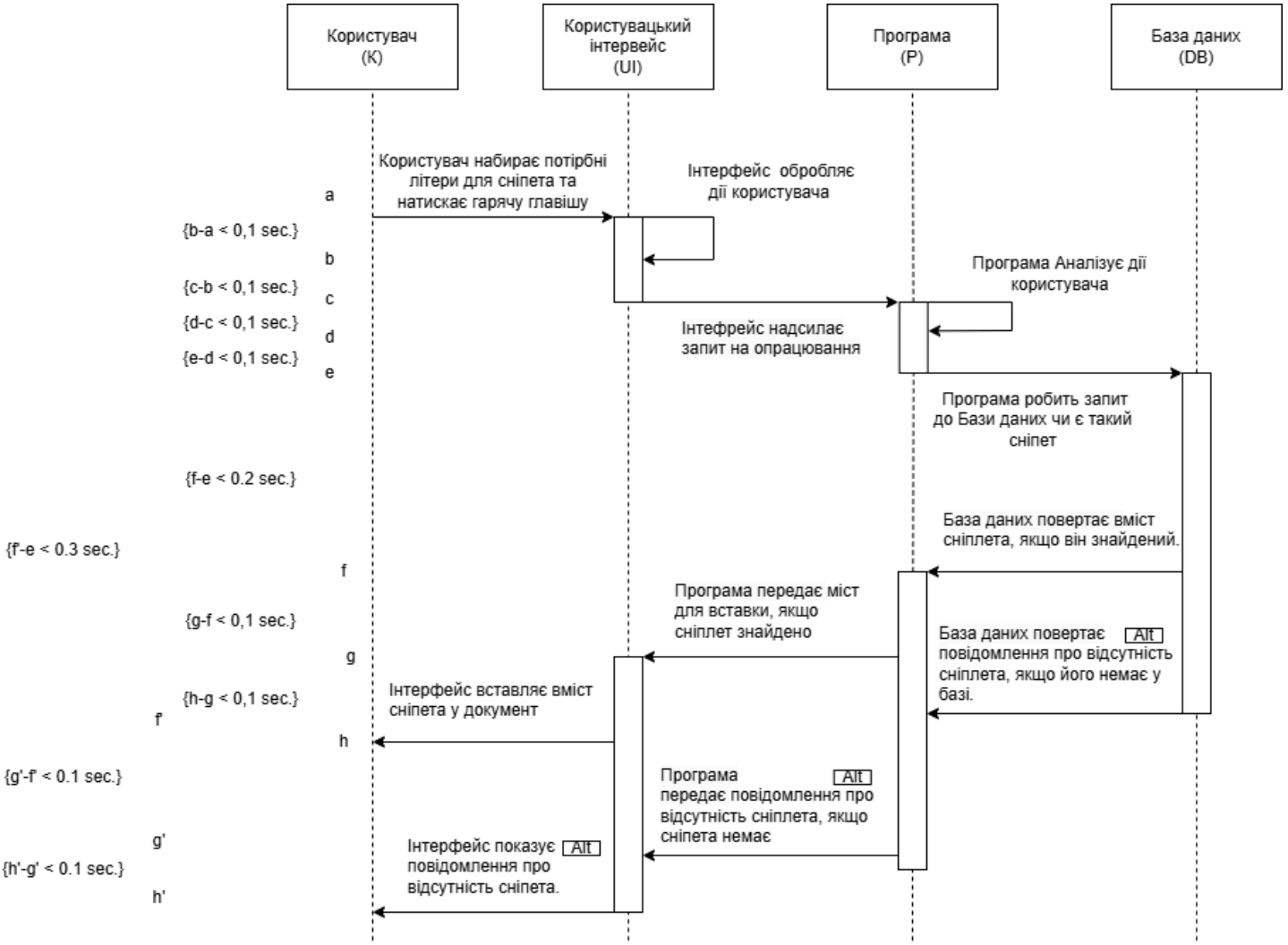
Database (База даних)

Tables (7 таблиць)

- **Опис для всіх таблиць**: Кожна таблиця зберігає інформацію для певної сутності, що дозволяє системі функціонувати і зберігати дані для різних частин текстового редактора:
  - **FileTable**: зберігає дані файлів користувача.
  - **SnippetTable**: містить дані зі сніпетами, доступними для вставки.
  - **MacroTable**: зберігає макроси користувача.
  - **BookmarkTable**: містить дані про закладки, створені користувачем.
  - **SyntaxHighlighterTable**: містить правила та параметри для підсвітки синтаксису.
  - **EncodingTable**: зберігає інформацію про підтримувані кодування.
  - **SettingsTable**: містить дані про налаштування редактора для користувача.

Діаграма послідовностей

Процес використання сніпету



Висновки

У цій лабораторній роботі було проведено проектування текстового редактора з використанням діаграм, що охоплюють структуру системи, фізичне розгортання та послідовність дій для ключових функцій. Це дозволило отримати чітке уявлення про архітектуру програми, визначити взаємодію між компонентами та продумати основні процеси роботи редактора.

1. **Діаграма розгортання** показала, як текстовий редактор буде працювати на комп'ютері користувача, зображуючи фізичні пристрої, середовища виконання та розташування компонентів. Це дозволяє розуміти, які ресурси система використовує для забезпечення своєї роботи.
2. **Діаграма компонентів** надала структуроване уявлення про основні модулі програми, включаючи контролери, сервіси та репозиторії для кожної сутності. Це дозволило побачити, як різні частини системи взаємодіють одна з одною та як дані передаються між компонентами.
3. **Діаграма послідовностей** деталізувала кроки виконання однієї з функцій редактора (наприклад, використання сніплета), що допомогло візуалізувати обробку запитів від користувача та забезпечити чітке розуміння взаємодії між інтерфейсом, логікою та базою даних.