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. Діаграму компонентів. Діаграму послідовностей»

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

Перевірив:
Мягкий М.Ю.

Київ-2024

Завдання:
Ознайомитися з короткими теоретичними відомостями.
Розробити діаграму розгортання для проектованої системи.
Розробити діаграму компонентів для проектованої системи.
Розробити діаграму послідовностей для одного із процесів розроблюваної системи.
Скласти звіт про виконану роботу.

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):

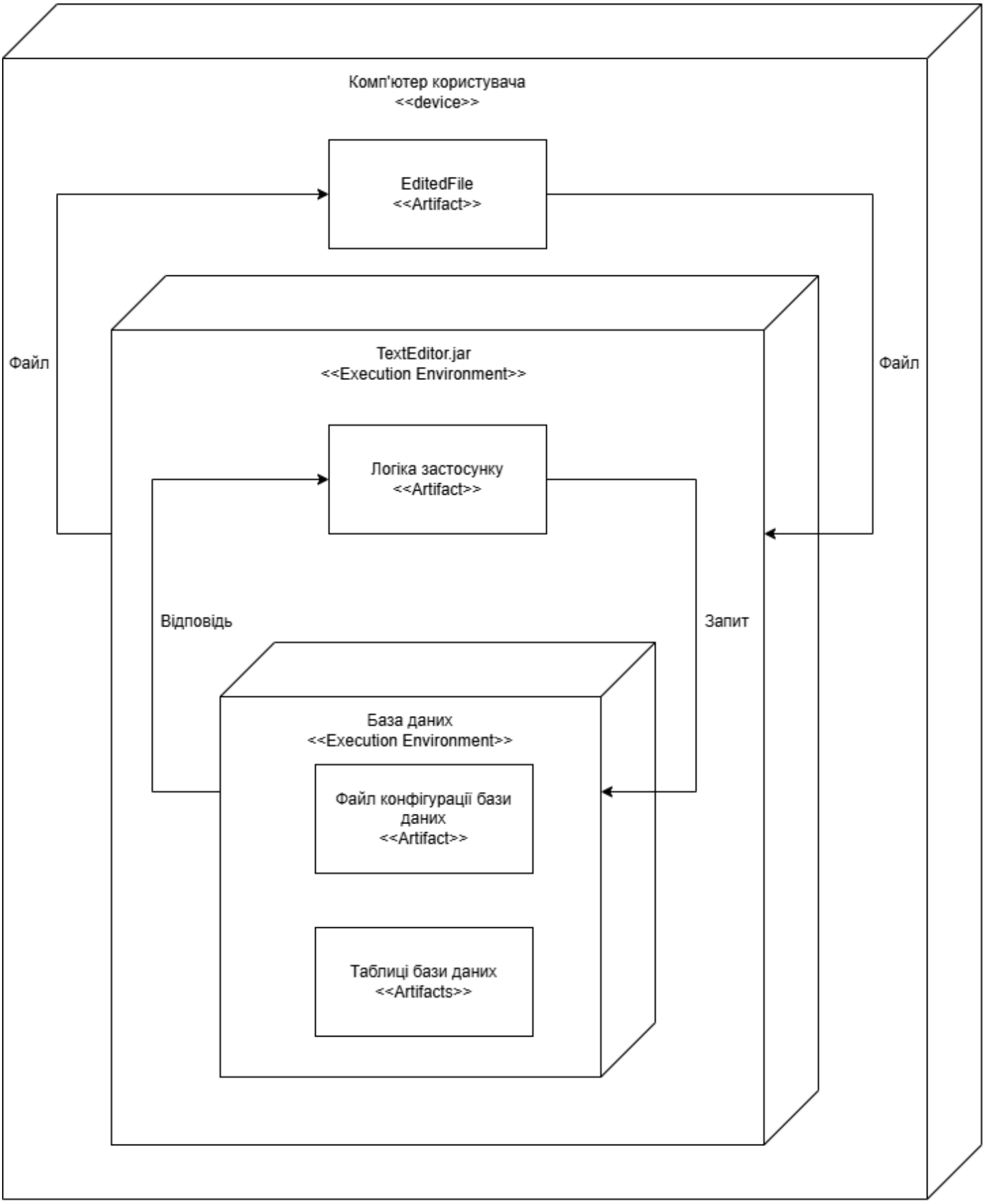
- Діаграма компонентів зображає структуру програмного забезпечення на рівні модулів або компонентів, показуючи взаємодії та залежності між ними. Вона описує, як різні частини програми (наприклад, контролери, сервіси, репозиторії) співпрацюють для реалізації функцій системи. Діаграма компонентів використовується 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

Комп'ютер користувача (<<device>>):

- Основний пристрій, на якому запускається текстовий редактор.
- Містить артефакт **EditedFile (<<Artifact>>)**, який представляє собою файл, що редагується користувачем. Цей артефакт отримує дані з текстового редактора та дозволяє користувачу зберігати редаговані файли.

TextEditor.jar (<<Execution Environment>>):

- Виконавче середовище для текстового редактора у вигляді JAR-файлу.
- Містить артефакт **Логіка застосунку (<<Artifact>>)**, що представляє основний функціональний код програми. Логіка застосунку обробляє запити користувача, взаємодіє з базою даних, обробляє файли та відповідає за передачу відредагованих файлів назад до користувача.

База даних (<<Execution Environment>>):

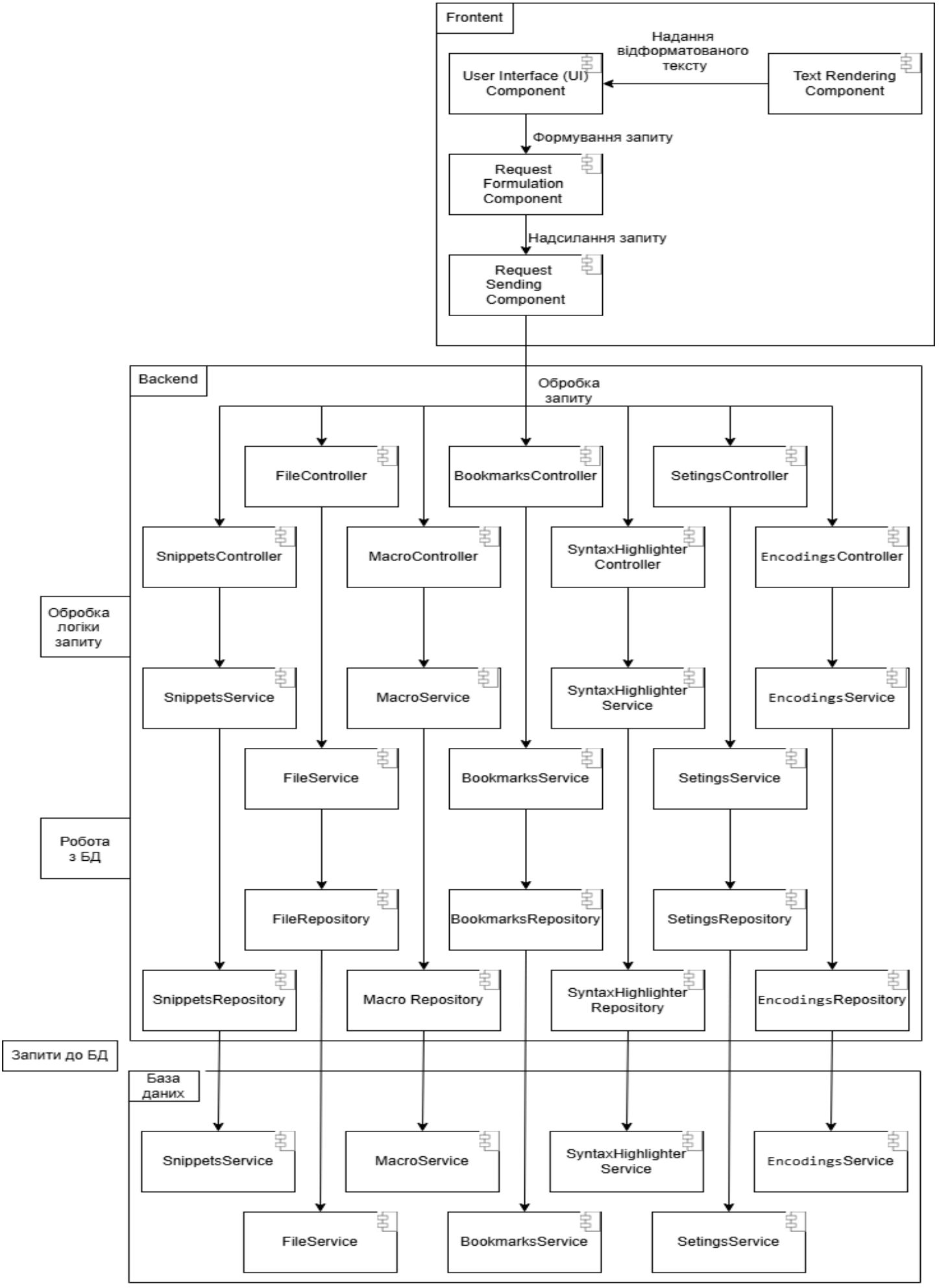
- Виконавче середовище, яке обробляє та зберігає дані для текстового редактора.
- Містить два артефакти:
  - **Файл конфігурації бази даних (<<Artifact>>)**: Налаштовує підключення до бази даних і визначає конфігураційні параметри для її роботи.
  - **Таблиці бази даних (<<Artifact>>)**: Містять фактичні дані, необхідні для роботи програми, такі як сніпети, історія редагування та інші дані, які зберігаються в структурованій формі.

Зв’язки:

- **Запит**: Логіка застосунку надсилає запит до бази даних, щоб отримати необхідні дані (наприклад, сніпети або інші ресурси).

- **Відповідь**: База даних надсилає відповідь назад до Логіки застосунку з результатами запиту (дані або повідомлення про їх відсутність).
- **Файл**: EditedFile обмінюється даними з Логікою застосунку, отримуючи редаговані файли для збереження або передачі їх до користувача.

Діаграма компонентів



Frontend

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.

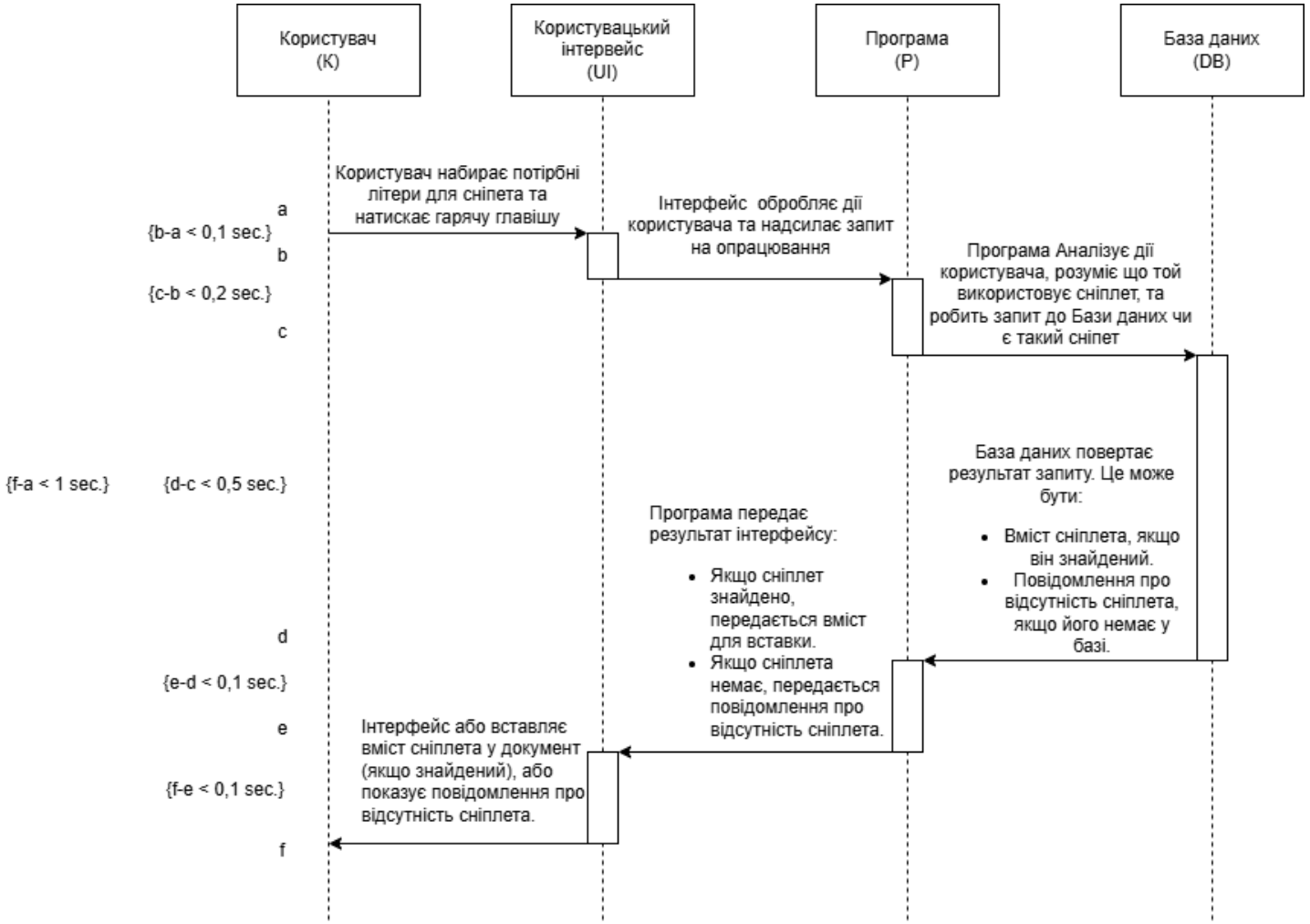
Database (База даних)

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.