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

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування розгортання»

Тема роботи: 3. Текстовий редактор

Виконав
студент групи ІА–33
Супик Андрій Олександрович

Київ 2025

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.

ТемаРоботи:

3. **Текстовий редактор** (strategy, command, observer, template method, flyweight, SOA)

Текстовий редактор повинен вміти розпізнавати текстові файли в будь-якій кодуванні, мати розширені функції редагування: макроси, сніппети, підказки, закладки, перехід на рядок / сторінку, підсвічування синтаксису (для однієї мови програмування або розмітки на розсуд студента).

Посилання на репозиторій: https://github.com/insxlll/trpzlab

Короткі теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонентів.

Діаграма компонентів (Component diagram)

Діаграма компонентів – в UML, діаграма, на якій відображаються компоненти, залежності та зв'язки між ними. Діаграма компонент відображає залежності між компонентами програмного забезпечення,

включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути представлено як компоненту. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонентів відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Діаграма послідовностей (Sequence diagram)

Діаграма послідовностей Діаграма послідовності – різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність надісланих повідомлень.

Діаграма розгортання

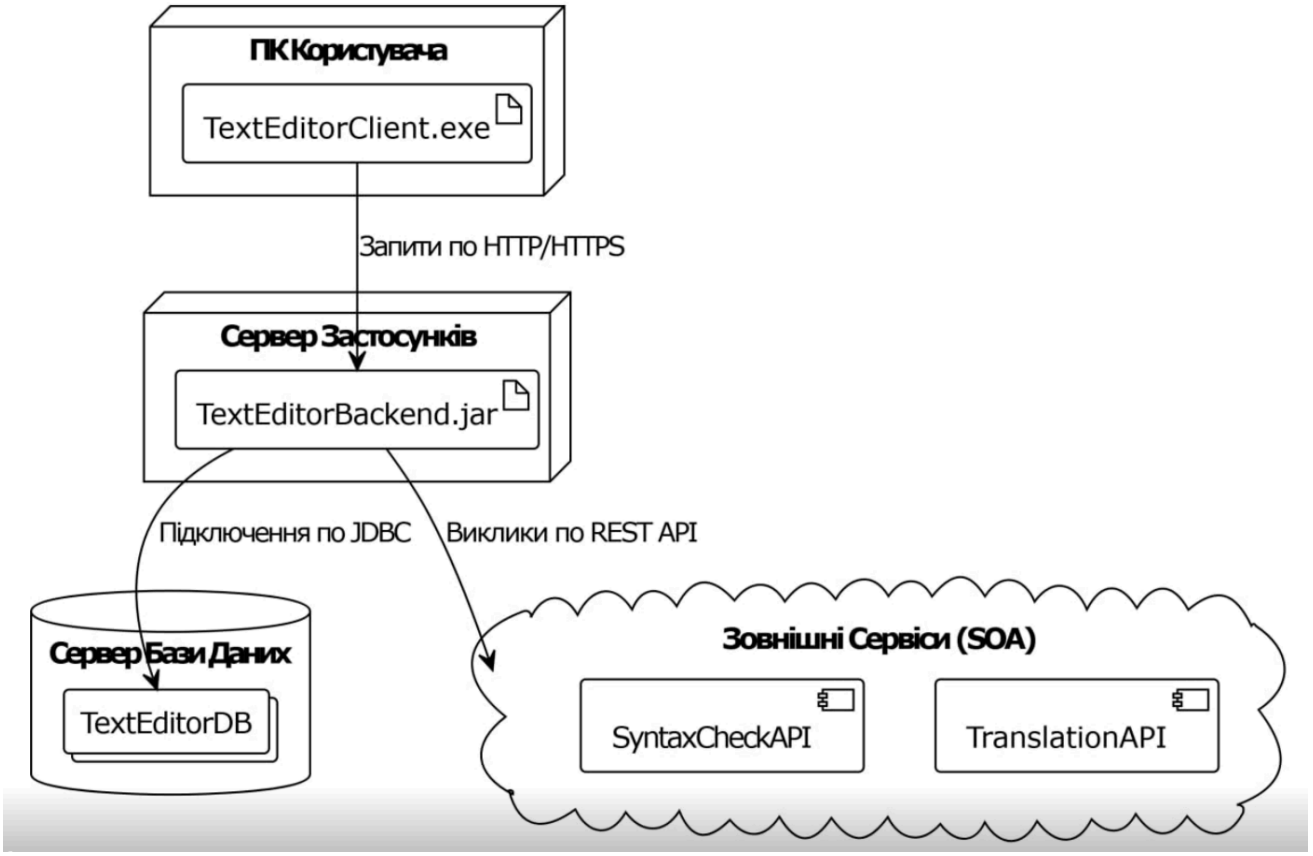

Рисунок 1. - Діаграма розгортання застосунку

Ця діаграма розгортання показує компоненти системи для текстового редактора і те, як вони розгорнуті в різних середовищах.

ПК Користувача: a) Вузол «ПК Користувача» містить клієнтський додаток TextEditorClient.exe. Це середовище, де користувач безпосередньо взаємодіє з текстовим редактором. b) TextEditorClient.exe відповідає за надання графічного інтерфейсу, обробку дій користувача (наприклад, відкриття, редагування та збереження файлів) та надсилання запитів до сервера застосунків.

Сервер Застосунків: a) Вузол «Сервер Застосунків» містить серверний компонент TextEditorBackend.jar. Це вказує на те, що бекенд системи, який містить бізнес-логіку, реалізований на платформі Java. b) TextEditorBackend.jar обробляє запити від клієнта, взаємодіє з базою даних для збереження та отримання даних (використовуючи патерн Repository), а також інтегрується із зовнішніми сервісами.

Сервер Бази Даних: a) Вузол «Сервер Бази Даних» представляє сервер, на якому розміщено базу даних TextEditorDB. b) TextEditorDB відповідає за зберігання та управління інформацією системи, такою як дані про користувачів (User) , документи (Document) , макроси (Macro) , сніпети (Snippet) , закладки (Bookmark) та історію змін (History).

Зовнішні Сервіси (SOA): a) Вузол «Зовнішні Сервіси (SOA)» представляє сторонні сервіси, з якими інтегрується система. На діаграмі він містить компоненти SyntaxCheckAPI та TranslationAPI. b) Ці API надають розширені функції: TranslationAPI використовується для автоматичного перекладу тексту , а SyntaxCheckAPI — для перевірки синтаксису.

Протоколи зв'язку: a) HTTP/HTTPS використовується для зв'язку між TextEditorClient.exe та TextEditorBackend.jar для надсилання запитів та отримання відповідей. b) JDBC (Java Database Connectivity) використовується сервером застосунків для підключення та виконання операцій з базою даних

TextEditorDB. c) REST API використовується сервером застосунків для викликів зовнішніх сервісів SyntaxCheckAPI та TranslationAPI.

Діаграма компонентів

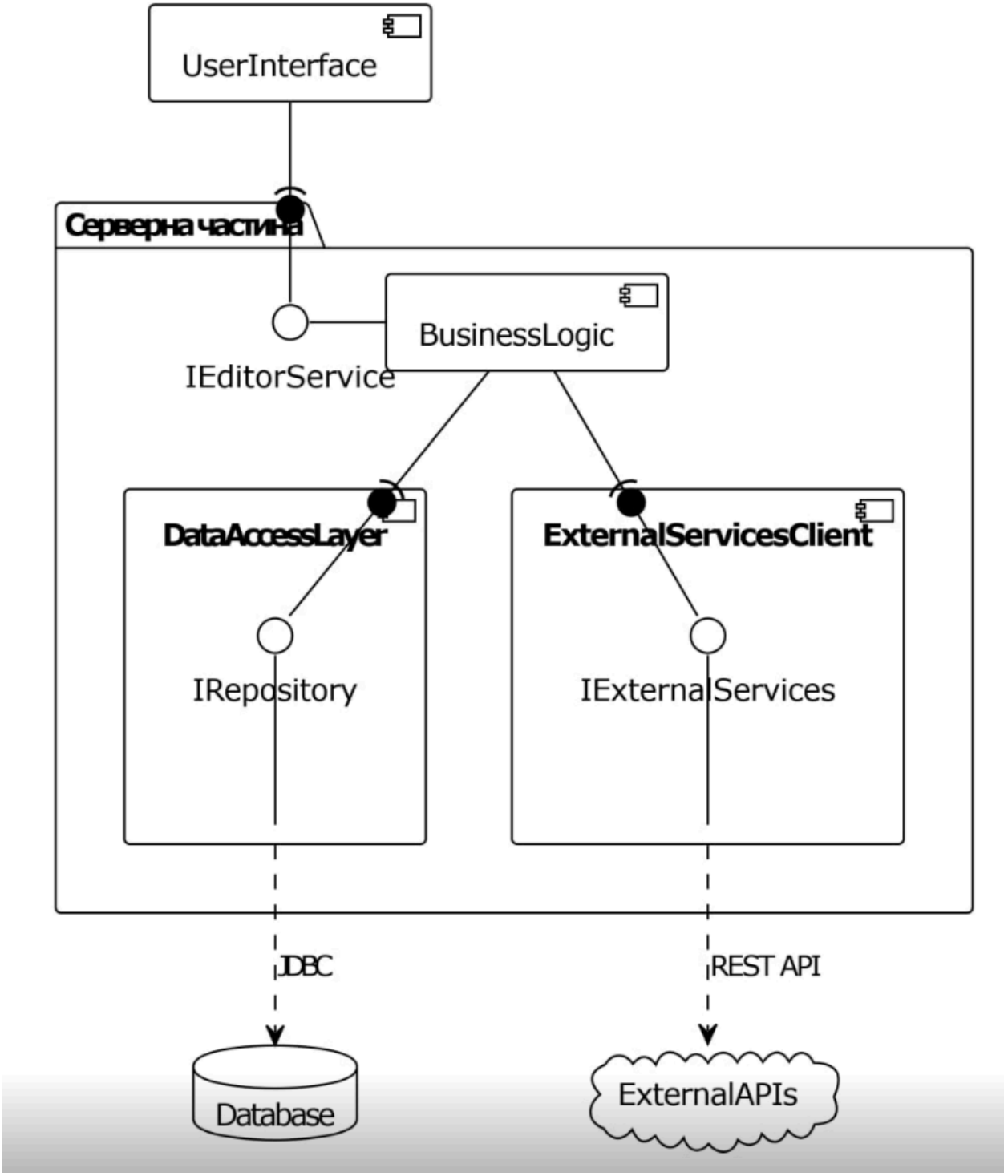

Рисунок 2. - Діаграма компонентів застосунку

У даній діаграмі компонентів представлена архітектура системи «Текстовий редактор», яка побудована за моделлю клієнт-сервер.

Перший рівень – це компонент UserInterface. Це клієнтська частина, через яку користувач взаємодіє із системою. Користувач може відкривати файли, редагувати текст, використовувати макроси, шукати слова та виконувати інші функції. Усі ці дії надсилаються у вигляді запитів до серверної частини через її чітко визначений інтерфейс IEditorService.

Другий рівень – це «Серверна частина», де реалізована основна логіка роботи програми. Ця частина містить три ключові компоненти:

BusinessLogic: Цей компонент реалізує інтерфейс IEditorService і містить основну бізнес-логіку редактора (наприклад, логіку редагування, керування документами, виконання макросів).

DataAccessLayer: Використовується компонентом BusinessLogic для доступу до даних. Він надає інтерфейс IRepository , який абстрагує логіку роботи з базою даних від бізнес-логіки.

ExternalServicesClient: Також використовується BusinessLogic для інтеграції зі сторонніми сервісами. Він надає інтерфейс IExternalServices для таких функцій, як автопереклад або перевірка синтаксису.

Третій рівень – це зовнішні залежності системи.

Database: Забезпечує зберігання всієї інформації. Тут містяться дані про користувачів, документи, макроси, закладки та історію змін. Компонент DataAccessLayer звертається до бази даних через протокол JDBC.

ExternalAPIs: Це зовнішні сервіси (SOA), з якими взаємодіє система. Компонент ExternalServicesClient звертається до них через REST API для отримання послуг перекладу чи перевірки синтаксису.

Діаграма демонструє чіткий поділ відповідальності між компонентами. Використання інтерфейсів (IEditorService, IRepository, IExternalServices) забезпечує низьку зв'язність системи, що робить її гнучкою, масштабованою та зручною для подальшого розвитку та тестування.

Діаграма послідовностей

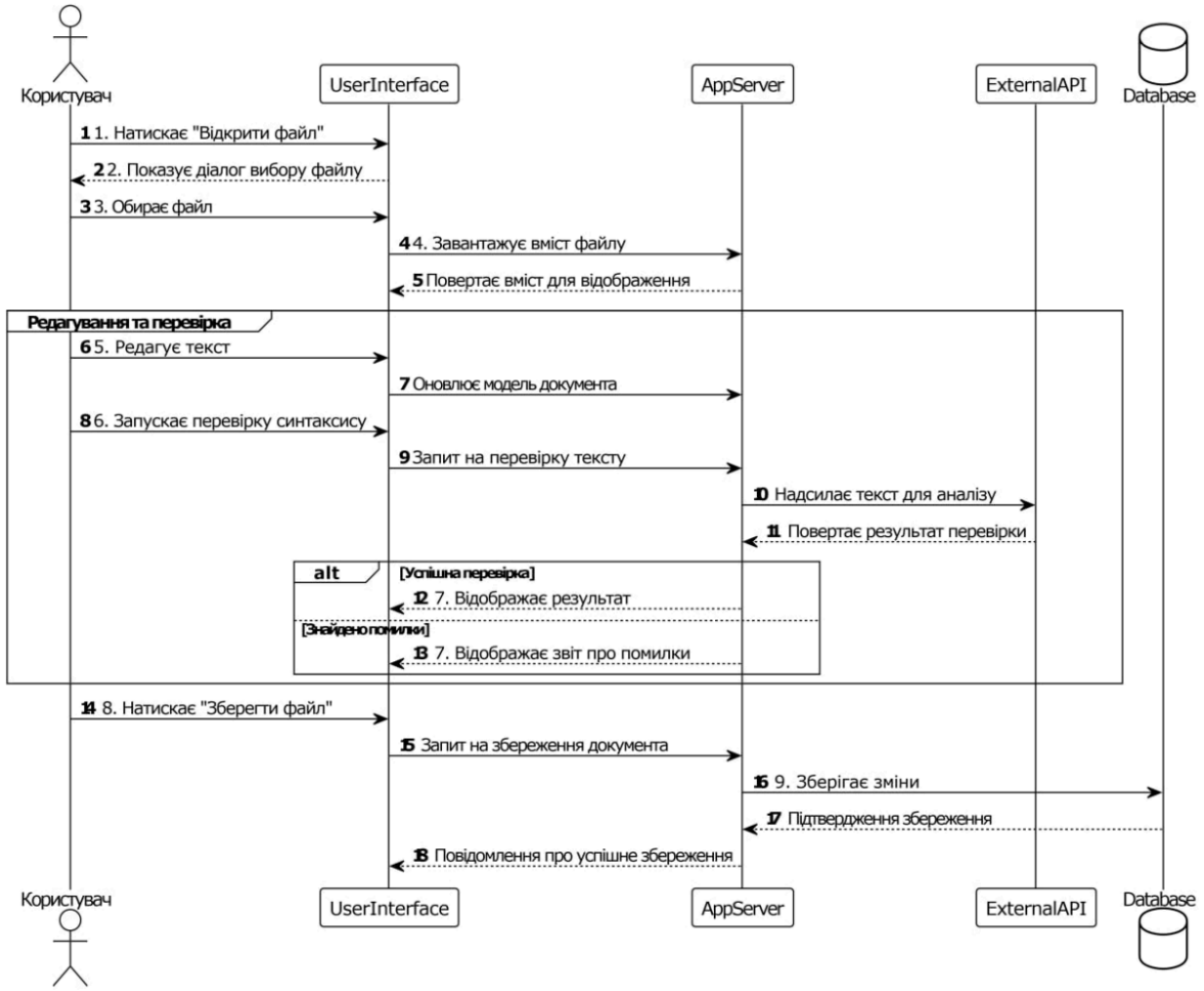

Рисунок 3. - Діаграма послідовностей для сценарію «Відкриття та редагування файлу»

Діаграма послідовностей для «Сценарію №1: Відкриття та редагування файлу» ілюструє взаємодію між п'ятьма основними учасниками: Користувачем, UserInterface (клієнтським додатком), AppServer (сервером застосунків), ExternalAPI (зовнішнім сервісом) та Database (базою даних).

Користувач ініціює процес, натискаючи «Відкрити файл» на UserInterface. Інтерфейс, після вибору файлу користувачем , надсилає запит на завантаження вмісту до AppServer, який повертає дані для відображення у редакторі.

На етапі «Редагування та перевірка» кожна зміна тексту користувачем призводить до оновлення моделі документа на AppServer. Коли користувач запускає перевірку синтаксису, UserInterface надсилає запит до AppServer. Сервер, у свою чергу, делегує це завдання, надсилаючи текст для аналізу до ExternalAPI (що відповідає інтеграції із зовнішніми сервісами). ExternalAPI обробляє запит і повертає результат перевірки.

Діаграма також враховує виняткові ситуації за допомогою блоку alt. У випадку успішної перевірки AppServer відображає результат. Якщо ж виявлено помилки («З виключенням»), система інформує користувача, відображаючи звіт про помилки.

На завершення, коли користувач натискає «Зберегти файл», UserInterface надсилає запит на збереження на AppServer. AppServer обробляє його, зберігаючи зміни в Database. Після отримання підтвердження від бази даних, сервер повідомляє UserInterface про успішне збереження.

Таким чином, діаграма демонструє логіку взаємодії компонентів системи, чітко розділяючи відповідальність: UserInterface керує взаємодією з користувачем, AppServer координує бізнес-логіку, ExternalAPI надає спеціалізовані функції, а Database відповідає за надійне зберігання даних.

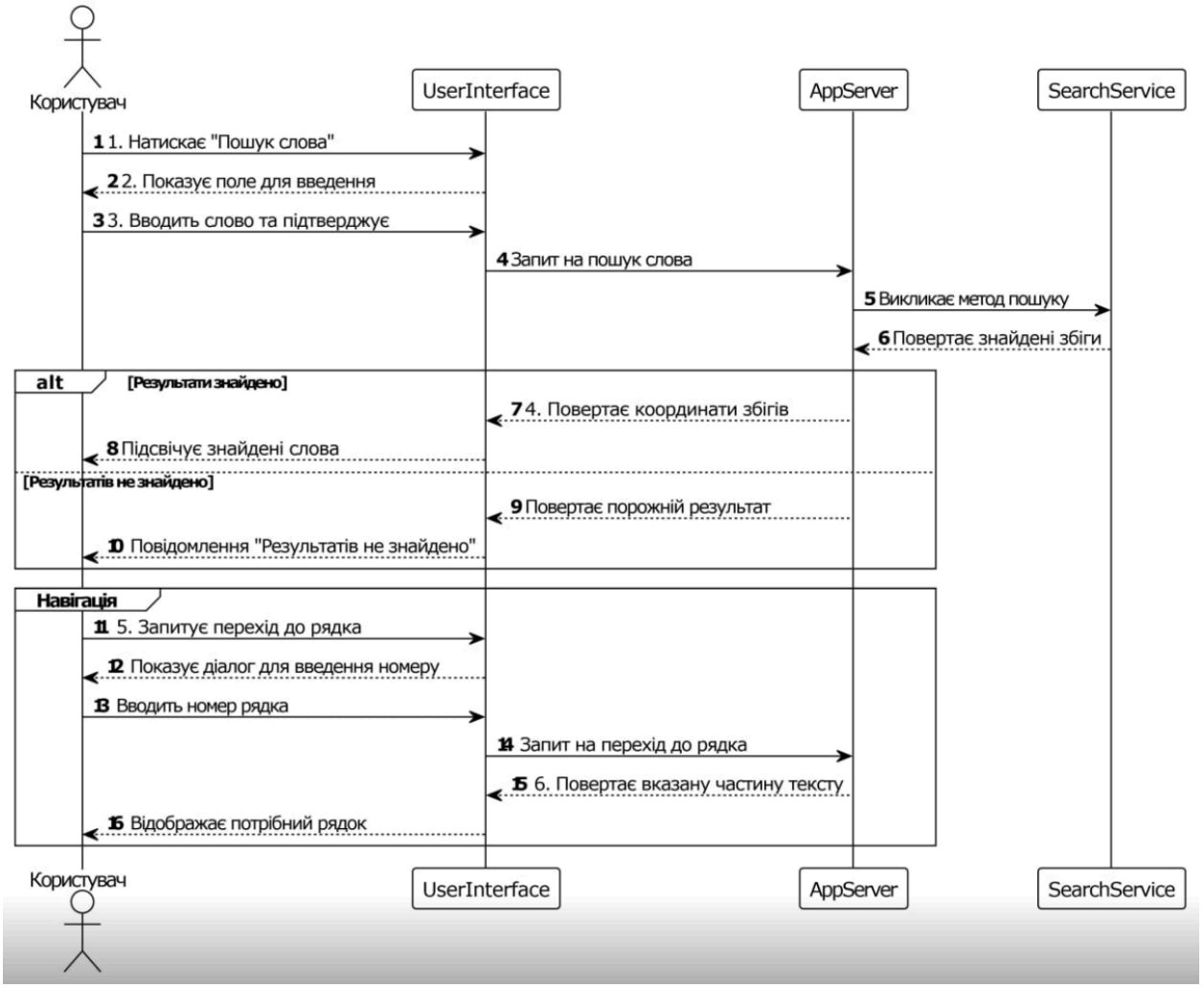

Рисунок 4. - Діаграма послідовностей для сценарію «Пошук і навігація по тексту»

Діаграма послідовностей для «Сценарію №2: Пошук і навігація по тексту» ілюструє взаємодію між чотирма учасниками: Користувачем, UserInterface (клієнтським додатком), AppServer (сервером застосунків) та SearchService (спеціалізованим сервісом пошуку).

Процес починається, коли Користувач натискає «Пошук слова». UserInterface відображає поле для введення, і після того, як користувач вводить слово та підтверджує , інтерфейс надсилає запит на пошук до AppServer. AppServer не виконує пошук самостійно, а делегує це завдання, викликаючи відповідний метод у SearchService. SearchService обробляє запит і повертає знайдені збіги на AppServer.

Далі діаграма використовує блок alt для обробки двох можливих результатів. Якщо «Результати знайдено», AppServer повертає координати збігів , і UserInterface підсвічує їх для Користувача. Якщо ж «Результатів не знайдено», AppServer повертає порожній результат , і UserInterface інформує про це Користувача відповідним повідомленням.

Окремо показано потік «Навігація». Користувач ініціює перехід до рядка , вводить номер у діалоговому вікні , і UserInterface надсилає запит на AppServer. AppServer обробляє цей запит і повертає вказану частину тексту , яку UserInterface відображає Користувачу.

Таким чином, діаграма демонструє чіткий розподіл обов'язків: UserInterface відповідає за взаємодію з користувачем, AppServer виступає в ролі координатора логіки, а SearchService інкапсулює складну логіку пошуку.

Фото роботи програми

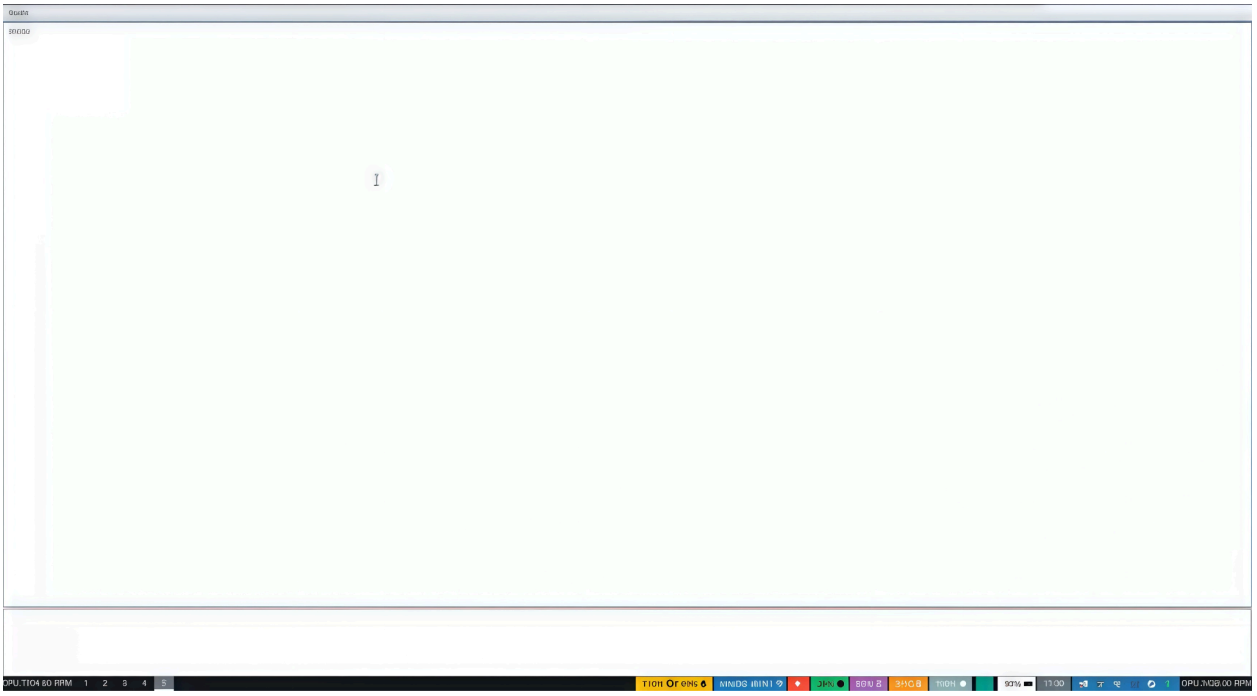

Рисунок 5.

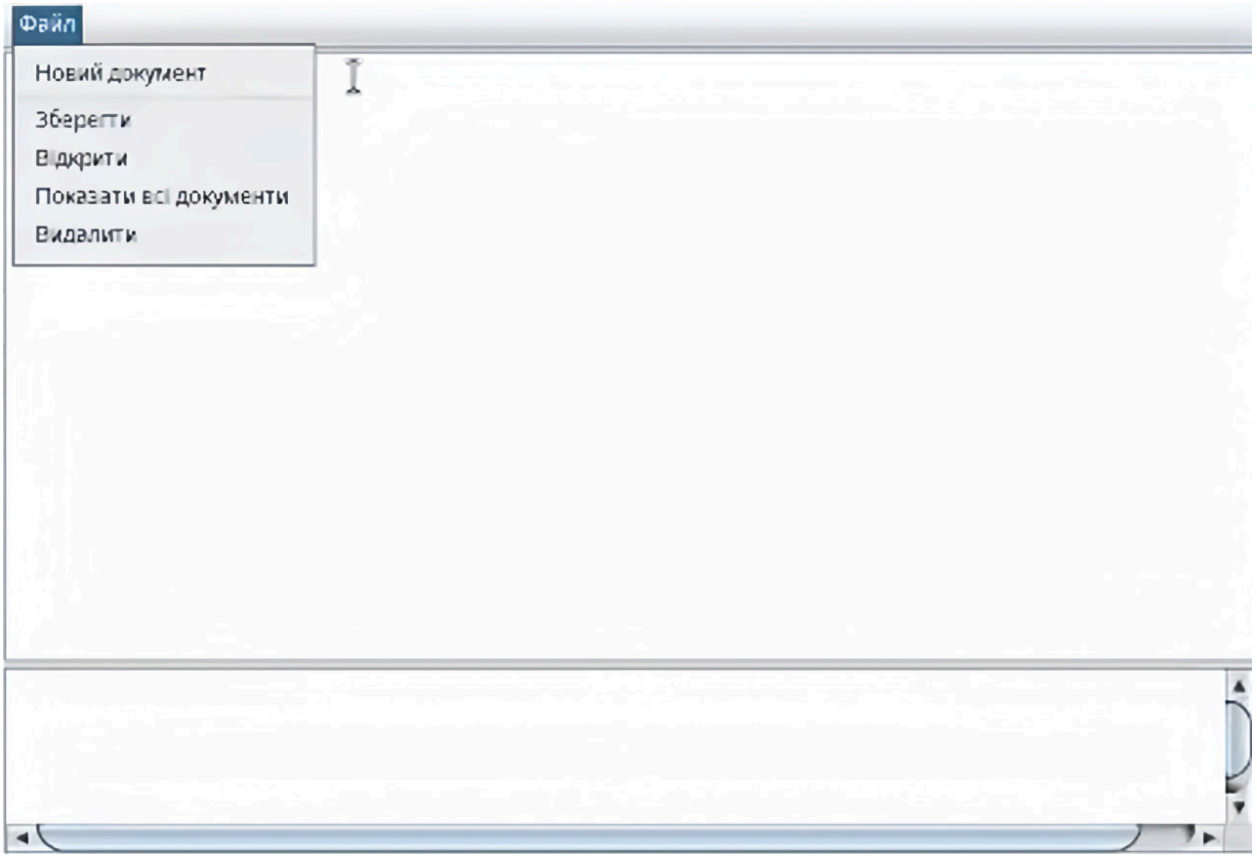

Рисунок 6. - Функціонал

Рисунок 7.

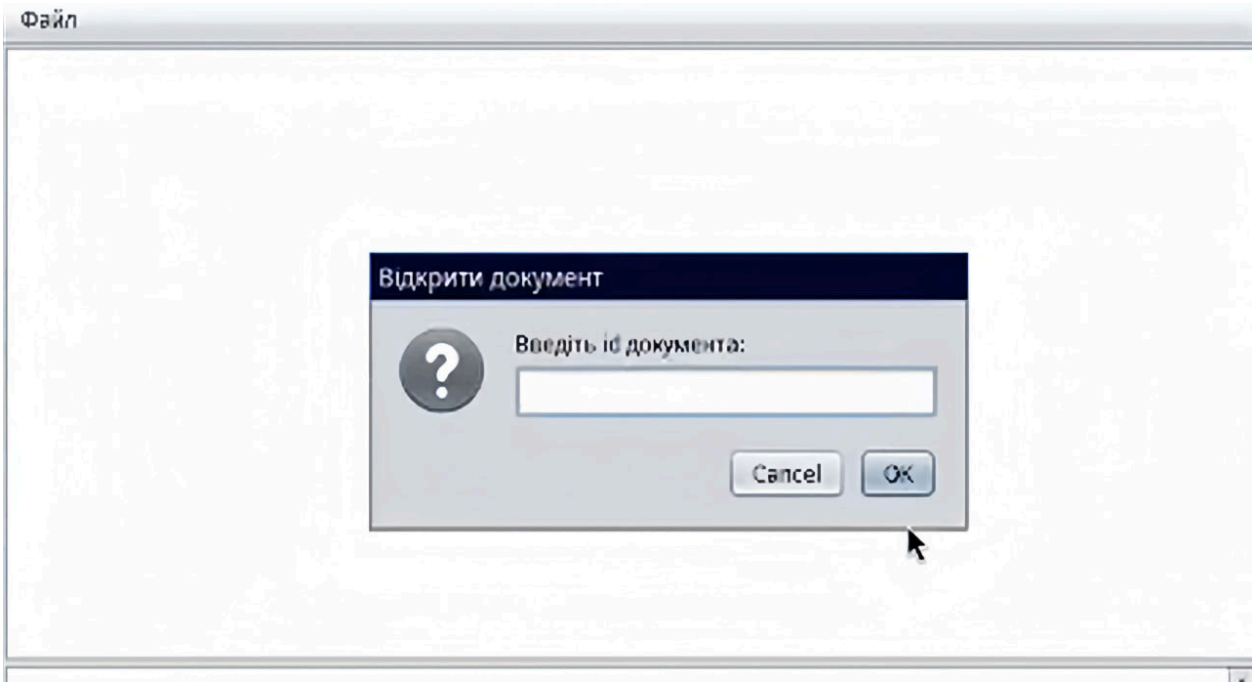

Рисунок 8.

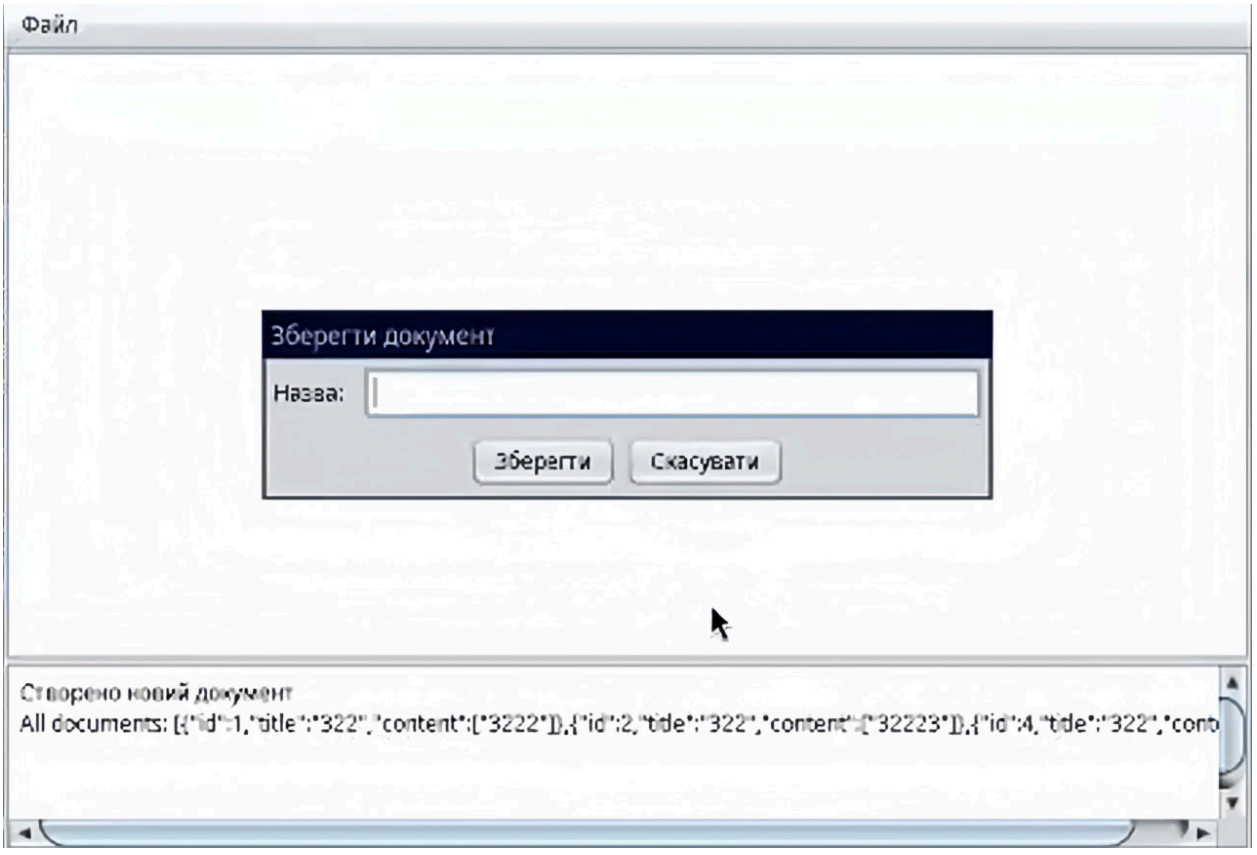

Рисунок 9.

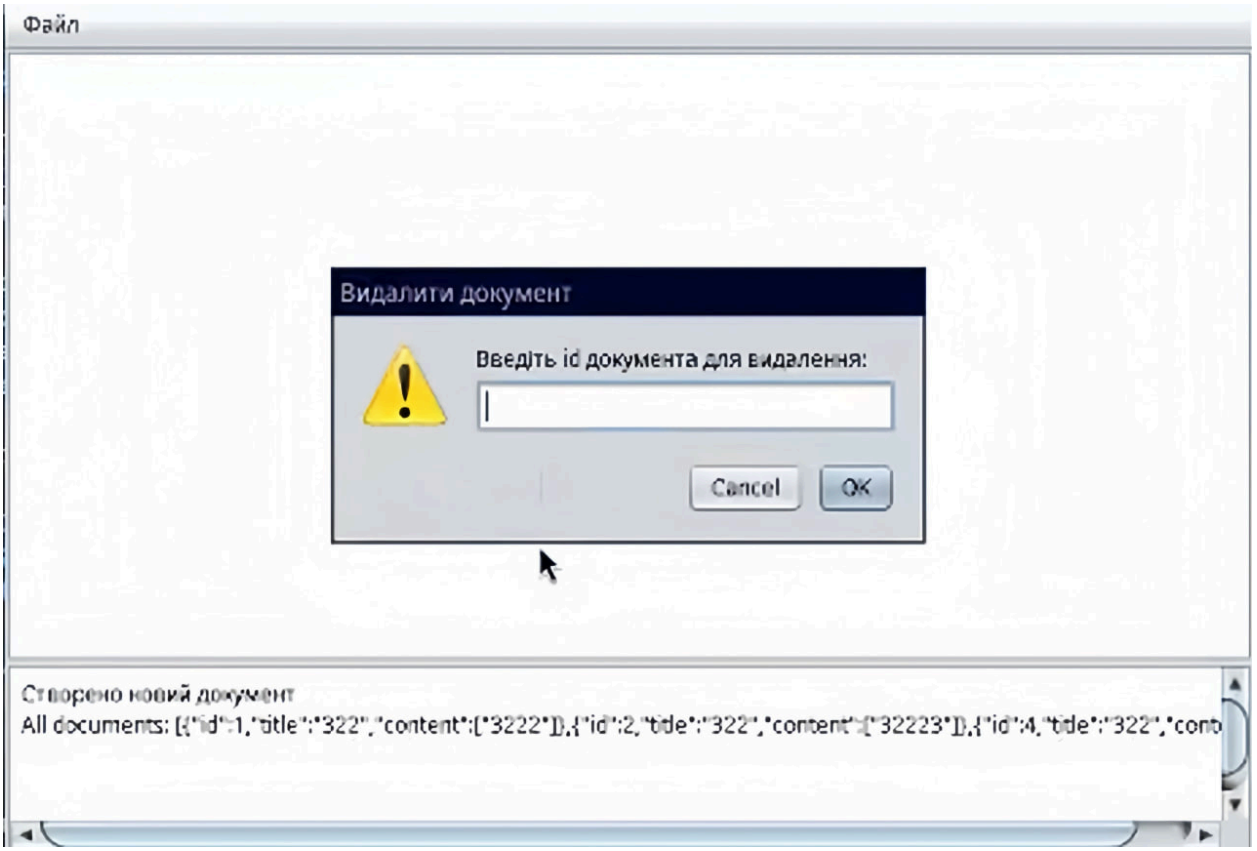

Рисунок 10.

Висновок: У ході лабораторної роботи було успішно застосовано принципи об'єктно-орієнтованого аналізу та проєктування для створення повної архітектури «Текстового редактора».

Було проведено аналіз вимог, що включав побудову діаграми варіантів використання та розробку детальних сценаріїв. На основі цього була спроєктована логічна модель у вигляді діаграми класів , а також розроблена схема бази даних з використанням патерну Repository для відокремлення логіки доступу до даних.

Архітектура системи була візуалізована за допомогою діаграми компонентів, а її динамічна поведінка — за допомогою діаграм послідовностей. На завершення, діаграма розгортання визначила фізичну клієнт-серверну архітектуру.

Результатом роботи є повний, структурований та готовий до подальшої

реалізації архітектурний проєкт текстового редактора.

Контрольні запитання

1. Що собою становить діаграма розгортання? Діаграма розгортання — це тип діаграми UML, який показує фізичне розміщення програмних компонентів системи на апаратних пристроях. Вона відображає, на яких серверах або комп’ютерах розгорнуті частини програми та як між ними здійснюється зв’язок. Така діаграма використовується для опису архітектури системи й показує, де саме працюють окремі компоненти.
2. Які бувають види вузлів на діаграмі розгортання? Пристрій — це фізичний елемент системи, наприклад комп’ютер, сервер, смартфон або маршрутизатор. Він представляє апаратне забезпечення, на якому може виконуватись програмне забезпечення. Виконавче середовище — це програмна платформа, яка працює всередині пристрою й забезпечує виконання програм. Прикладом може бути операційна система, віртуальна машина Java або контейнер.
3. Які бувають зв’язки на діаграмі розгортання? Зв’язок комунікації показує мережеву взаємодію між вузлами — наприклад, обмін даними між сервером і клієнтом. Такий зв’язок відображає, як пристрої або середовища з’єднані між собою. Зв’язок розміщення показує, який програмний компонент або артефакт розгорнуто на певному вузлі. Він відображає фізичне розташування програмних частин системи на пристроях.
4. Які елементи присутні на діаграмі компонентів? Компонент — це основний елемент діаграми, який представляє окрему частину системи, наприклад модуль, бібліотеку або службу. Інтерфейс показує, які функції або сервіси надає чи використовує компонент Залежність відображає відношення між компонентами, коли один із них потребує функцій іншого. З’єднання показують взаємодію між компонентами через їхні інтерфейси.
5. Що становлять собою зв’язки на діаграмі компонентів? Зв’язки на діаграмі компонентів показують взаємозалежність і взаємодію між різними

компонентами системи. Вони відображають, як один компонент використовує або надає функціональність іншому. Найпоширенішим типом зв’язку є залежність, яка показує, що один компонент потребує інтерфейс або сервіс іншого для своєї роботи. Також можуть бути з’єднання через інтерфейси, які відображають реальну взаємодію між компонентами під час виконання програми.

6. Які бувають види діаграм взаємодії? Діаграма послідовності показує обмін повідомленнями між об’єктами у часі — тобто хто, коли і в якій послідовності викликає певні операції. Діаграма кооперації (комунікації) відображає ті самі взаємодії, але зосереджується не на часовій послідовності, а на структурних зв’язках між об’єктами, які беруть участь у взаємодії.

7. Для чого призначена діаграма послідовностей? Діаграма послідовностей призначена для відображення процесу взаємодії між об’єктами у часі. Вона показує, які об’єкти беруть участь у певному сценарії системи, які повідомлення вони надсилають одне одному та в якій послідовності це відбувається. За допомогою такої діаграми можна наочно простежити логіку виконання операцій, порядок викликів методів і реакцію системи на певні події. Діаграма послідовностей використовується під час аналізу або проєктування програм, щоб описати поведінку системи у динаміці.

8. Які ключові елементи можуть бути на діаграмі послідовностей? На діаграмі послідовностей основними елементами є об’єкти, лінії життя, повідомлення та активації. Об’єкти представляють учасників взаємодії, які надсилають або отримують повідомлення. Лінія життя показує існування об’єкта протягом часу — вона зображується вертикально під об’єктом. Повідомлення відображають виклики методів або обмін даними між об’єктами та показуються стрілками. Активація позначає період, коли об’єкт виконує певну дію у відповідь на отримане повідомлення.

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання? Діаграми послідовностей тісно пов’язані з діаграмами варіантів використання, оскільки вони деталізують поведінку системи,

описану у варіантах використання. Діаграма варіантів використання показує, які функції виконує система та які актори взаємодіють із нею. Діаграма послідовностей конкретизує ці сценарії, показуючи покрокову взаємодію об’єктів і обмін повідомленнями, необхідними для реалізації певного варіанту використання.

10. Як діаграми послідовностей пов’язані з діаграмами класів? Діаграми послідовностей пов’язані з діаграмами класів тим, що вони описують взаємодію об’єктів конкретних класів у часі. Діаграма класів визначає структуру системи, показуючи класи, їхні атрибути та методи, а також зв’язки між класами. Діаграма послідовностей показує, як об’єкти цих класів взаємодіють один з одним під час виконання певного сценарію, які методи викликаються і в якому порядку.