



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
із дисципліни «Технології розроблення програмного забезпечення»
Тема: «Основи проектування розгортання»

Виконав:
студент групи ІА-31:
Кунда А.П.

Перевірив:
Мягкий М.Ю.

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.

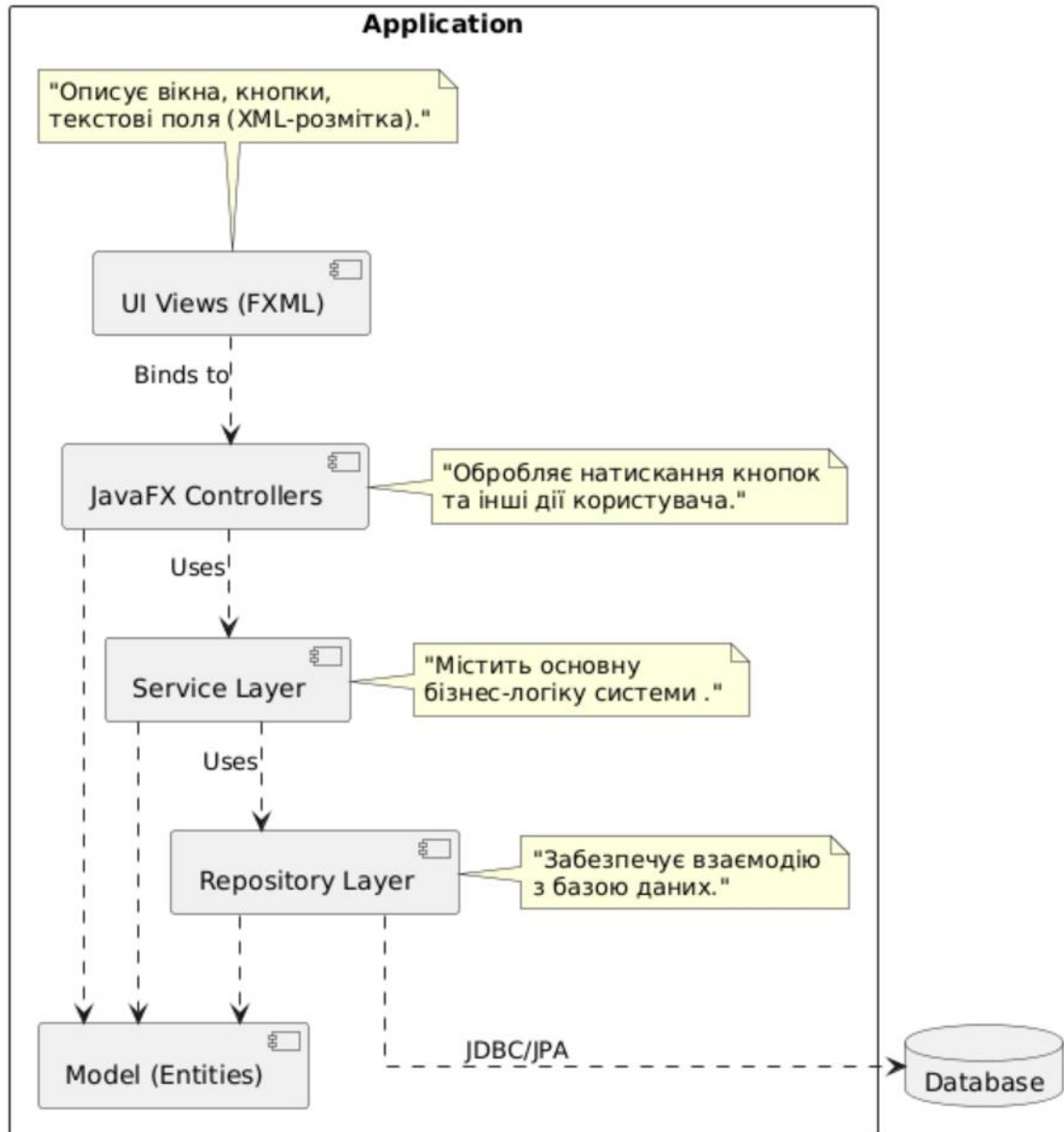
3.1. Завдання

- Ознайомитись з короткими теоретичними відомостями.
- Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроектувати 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 спроектованих 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.

Виконання завдань

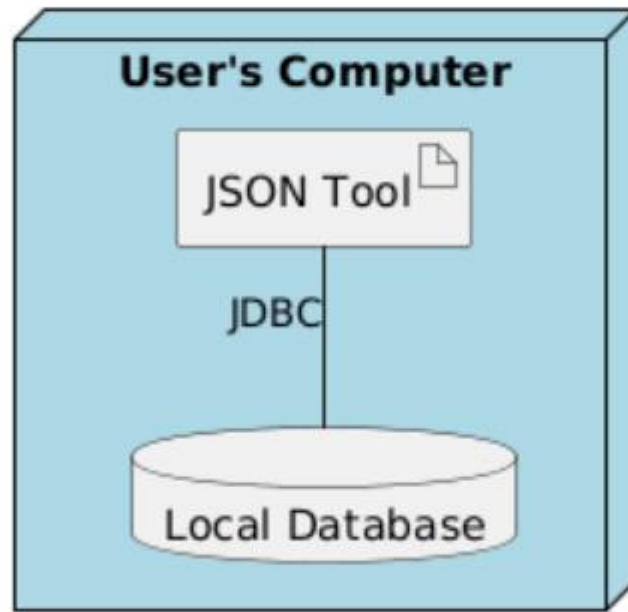
Розробити діаграму компонентів для проєктованої системи

Діаграма Компонентів: JSON Tool (JavaFX)



Розробити діаграму розгортання для проєктованої системи.

Діаграма Розгортання: JSON Tool (JavaFX)



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

Сценарій 1. "Створення нового JSON документа"



- 1) Користувач вводить дані у Browser і ініціює збереження.
- 2) Browser надсилає HTTP POST запит до ProductController.
- 3) Controller (контролер) приймає запит, але не обробляє його сам, викликаючи метод createDocument() у :DocumentService.
- 4) Service (сервіс) виконує всю бізнес-логіку: перевіряє дані на коректність і створює з них об'єкт-сутність JSONDocument.
- 5) Далі Service просить :JSONDocumentRepository зберегти цей об'єкт.
- 6) Repository (репозиторій) формує SQL-запит INSERT і відправляє його до :Database.
- 7) Після успішного збереження результат повертається назад: DB -> Repository -> Service -> Controller -> Browser

Сценарій 2. Перегляд історії операцій"



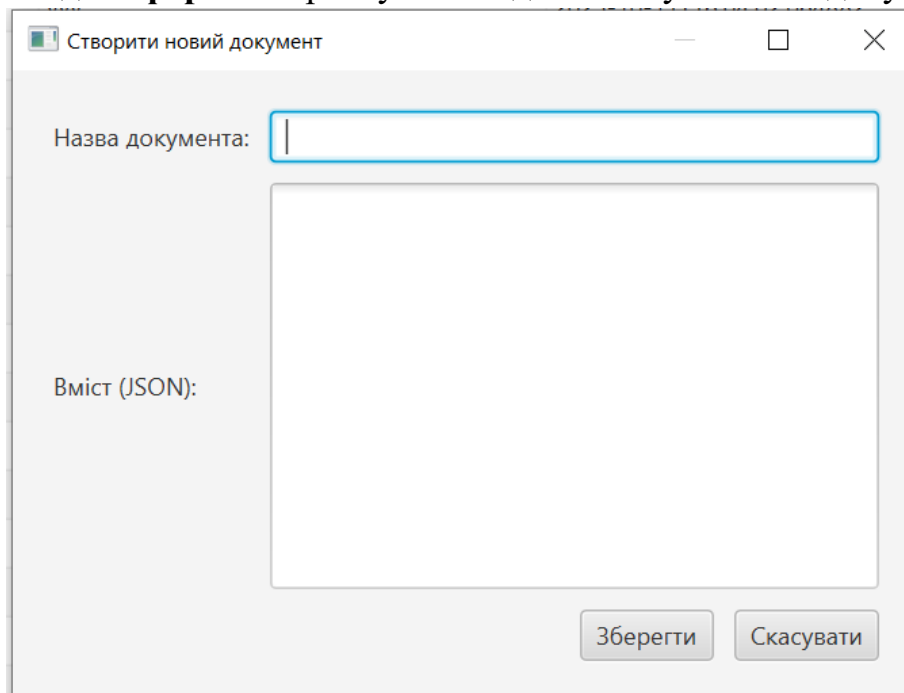
- 1) Користувач ініціює дію в Browser.
- 2) Browser надсилає GET запит до :HistoryController для отримання даних.
- 3) Controller викликає відповідний метод у :HistoryService, щоб отримати історію для поточного користувача.
- 4) Service звертається до :OperationHistoryRepository з проханням знайти всі записи, пов'язані з цим користувачем.
- 5) Repository виконує SELECT запит до :Database.
- 6) База даних повертає список знайдених записів.
- 7) Дані повертаються по ланцюжку: Repository -> Service. На шарі Service відбувається перетворення об'єктів бази даних (Entities) у об'єкти для передачі даних (DTO), щоб відправити на фронтенд тільки потрібну інформацію.
- 8) Controller отримує готові дані та надсилає їх у відповіді Browser, який відображає їх у вигляді зручної таблиці для Користувача.

На основі спроектованих 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).

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).

Ввід на формі: Користувач вводить назву та вміст документа у вікні



Збереження в БД: По натисканню кнопки "Зберегти" контролер викликає сервіс який зберігає новий об'єкт

Після закриття вікна створення на основній формі виконується запит SELECT запит до бази даних, отримує оновлений список всіх документів і відображає його в таблиці.

1	Тест 1	2025-10-11T16:03:28.893038
2	ууу	2025-10-11T16:04:02.064882
3	тест 1	2025-10-11T16:09:22.600576

Контрольні питання

1. Що собою становить діаграма розгортання?

Діаграма розгортання (Deployment Diagram) — це структурна діаграма UML, яка моделює фізичне розміщення програмних компонентів системи на апаратному забезпеченні. Вона показує, які програмні артефакти (наприклад, .exe або .jar файли) встановлюються на які фізичні вузли (наприклад, сервери чи комп'ютери) і як ці вузли з'єднані між собою.

- Аналогія: Це як план будівлі, де показано, в якій кімнаті (сервері) стоїть яка техніка (додаток) і як кімнати з'єднані коридорами (мережа).

2. Які бувають види вузлів на діаграмі розгортання?

Вузли (Nodes) — це основні елементи, що представляють обчислювальні ресурси. Існує два основних види:

- Вузол пристрою (Device Node): Представляє фізичне обладнання, таке як сервер, персональний комп'ютер, мобільний телефон або будь-який інший апаратний пристрій.
- Вузол середовища виконання (Execution Environment Node): Представляє програмне середовище, яке розміщує та виконує інші програмні компоненти. Приклади: сервер додатків (Tomcat), віртуальна машина Java (JVM), операційна система або Docker-контейнер.

3. Які бувають зв'язки на діаграмі розгортання?

Основний тип зв'язку — це шлях комунікації (Communication Path). Він зображується у вигляді суцільної лінії між двома вузлами і показує, що вони можуть обмінюватися інформацією. Цей зв'язок часто доповнюють стереотипом, щоб вказати протокол зв'язку, наприклад <<HTTP>>, <<JDBC>> або <<TCP/IP>>.

4. Які елементи присутні на діаграмі компонентів?

- Компонент (Component): Модульна, взаємозамінна частина системи. Це може бути бібліотека (.dll, .jar), виконуваний файл (.exe) або логічний блок коду (наприклад, "Service Layer").
- Інтерфейс (Interface): Описує набір операцій, які компонент надає іншим (provided interface, "lollipop") або які йому потрібні від інших (required interface, "socket").
- Порт (Port): Точка взаємодії між компонентом та його оточенням.

5. Що становлять собою зв'язки на діаграмі компонентів?

- Залежність (Dependency): Показує, що один компонент залежить від іншого (наприклад, використовує його класи або інтерфейси). Зміни в одному компоненті можуть вплинути на інший. Зображується пунктирною стрілкою.
- З'єднувач збірки (Assembly Connector): "З'єднує" необхідний інтерфейс ("socket") одного компонента з наданим інтерфейсом ("lollipop") іншого. Це показує, як компоненти "підключаються" один до одного для спільної роботи.

6. Які бувають види діаграм взаємодії?

Діаграми взаємодії (Interaction Diagrams) показують, як об'єкти співпрацюють з часом. Основні види:

- Діаграма послідовностей (Sequence Diagram)

- Діаграма комунікації (Communication Diagram)
- Діаграма огляду взаємодії (Interaction Overview Diagram)
- Діаграма синхронізації (Timing Diagram)

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей призначена для візуалізації взаємодії об'єктів у хронологічному порядку. Вона детально показує, яку послідовність повідомлень (викликів методів) об'єкти надсилають один одному для виконання конкретного завдання або сценарію. Головний акцент робиться на часі та порядку викликів.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

- Актор (Actor): Користувач або зовнішня система, що ініціює взаємодію.
- Лінія життя (Lifeline): Вертикальна пунктирна лінія, що представляє існування об'єкта протягом часу.
- Активация (Activation Bar): Вузкий прямокутник на лінії життя, що показує період, коли об'єкт виконує дію.
- Повідомлення (Message): Стрілка між лініями життя, що позначає виклик методу або передачу інформації.
- Фрагмент (Fragment): Рамка, що дозволяє моделювати цикли (loop), умови (alt, opt) та інші складні взаємодії.

9. Як діаграми послідовностей пов'язані з діаграмами варіантів використання?

Вони пов'язані як "що" і "як":

- Діаграма варіантів використання показує, **ЩО** система робить для користувача (наприклад, "Створити документ").
- Діаграма послідовностей показує, **ЯК** система це робить всередині, деталізуючи один конкретний сценарій цього варіанту використання (наприклад, покроковий процес створення документа).

10. Як діаграми послідовностей пов'язані з діаграмами класів?

Вони пов'язані як "структура" і "поведінка":

- Діаграма класів показує статичну структуру системи — які класи існують та які методи вони мають (креслення акторів).
- Діаграма послідовностей показує динамічну поведінку — як об'єкти (екземпляри) цих класів взаємодіють між собою, викликаючи методи один одного в реальному часі (сцена, де актори грають свої ролі).