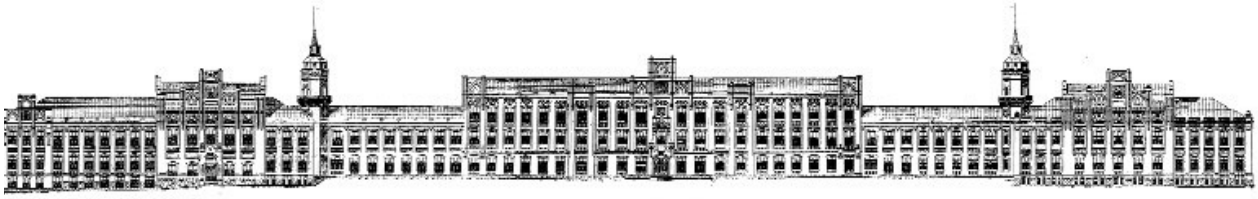

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3

з дисципліни «Технології розроблення

програмного забезпечення»

Тема: «Основи проєктування розгортання»

Виконала
студентка групи ІА-32:
Павлюк В. О.

Зміст

Вступ

Темою роботи є «Основи проєктування розгортання». Мета цієї лабораторної це навчитися проєктувати діаграми розгортання та компонентів для системи, що проєктується, а також розробляти діаграми послідовностей на основі сценаріїв, створених у попередній лабораторній роботі.

Як об’єкт проєктування обрано тему «Web crawler»: програмний сканер, що розпізнає структуру сторінок сайту, переходить за посиланнями, збирає інформацію за заданим терміном, видаляє несемантичні елементи (рекламу, скрипти тощо), зберігає очищені дані у вигляді структурованого набору HTML-файлів і веде статистику відвідань та метадані.

1 Теоретичні відомості

1.1. Діаграма розгортання (Deployment Diagram)

Діаграми розгортання представляють фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення [3]. Головними елементами діаграми є вузли, пов'язані інформаційними шляхами. Вузол (node) – це те, що може містити програмне забезпечення. Вузли бувають двох типів. Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний із системою. Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер (наприклад, вебсервер).

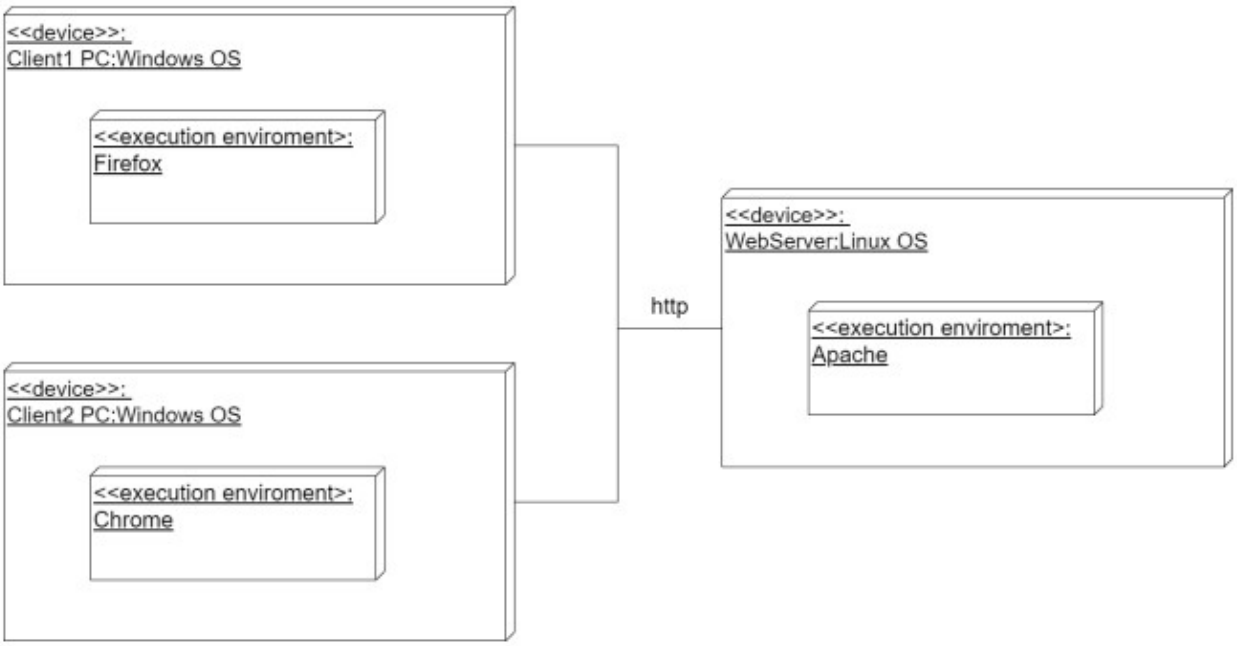

Рисунок 1.1. Діаграма розгортання системи

Між вузлами можуть стояти зв'язки, які зазвичай зображують у вигляді прямої лінії. Як і на інших діаграмах, у зв'язків можуть бути атрибути множинності (для показання, наприклад, підключення 2х і більше клієнтів до одного сервера) і назва. У назві, як правило, міститься спосіб зв'язку між двома 40 вузлами – це може бути назва протоколу (HTTP, IPC) або технологія, що використовується для забезпечення взаємодії вузлів (.NET Remoting, WCF).

Вузли можуть містити артефакти (artifacts), які є фізичним уособленням програмного забезпечення; зазвичай це файли. Такими файлами можуть бути виконувані файли (такі як файли .exe, двійкові файли, файли DLL, файли JAR, збірки або сценарії) або файли даних, конфігураційні файли, HTML-документи тощо. Перелік артефактів усередині вузла вказує на те, що на даному вузлі артефакт розгортається в систему, що запускається.

Артефакти можна зображати у вигляді прямокутників класів або перераховувати їхні імена всередині вузла. Можна супроводжувати вузли або артефакти значеннями у вигляді міток, щоб вказати різну цікаву інформацію про вузол, наприклад постачальника, операційну систему, місце розташування.

1.2. Діаграма компонентів

Діаграма компонентів UML є представленням проєктованої системи, розбитої на окремі модулі [3]. Залежно від способу поділу на модулі розрізняють три види діаграм компонентів: • логічні; • фізичні; • виконувані. Коли використовують логічне розбиття на компоненти, то у такому разі проєктовану систему віртуально уявляють як набір самостійних, автономних модулів (компонентів), що взаємодіють між собою. Коли на діаграмі представляють фізичне розбиття, то в такому разі на діаграмі компонентів показують компоненти та залежності між ними. Залежності показують, що класи в з одного компонента використовують класи з іншого компонента. Фізична модель використовується для розуміння які компоненти повинні бути зібрані в інсталяційний пакет. Також така діаграма показує зміни в якому компоненті будуть впливати на інші компоненти. Приклад такої діаграми наведено на рисунку 1.2.

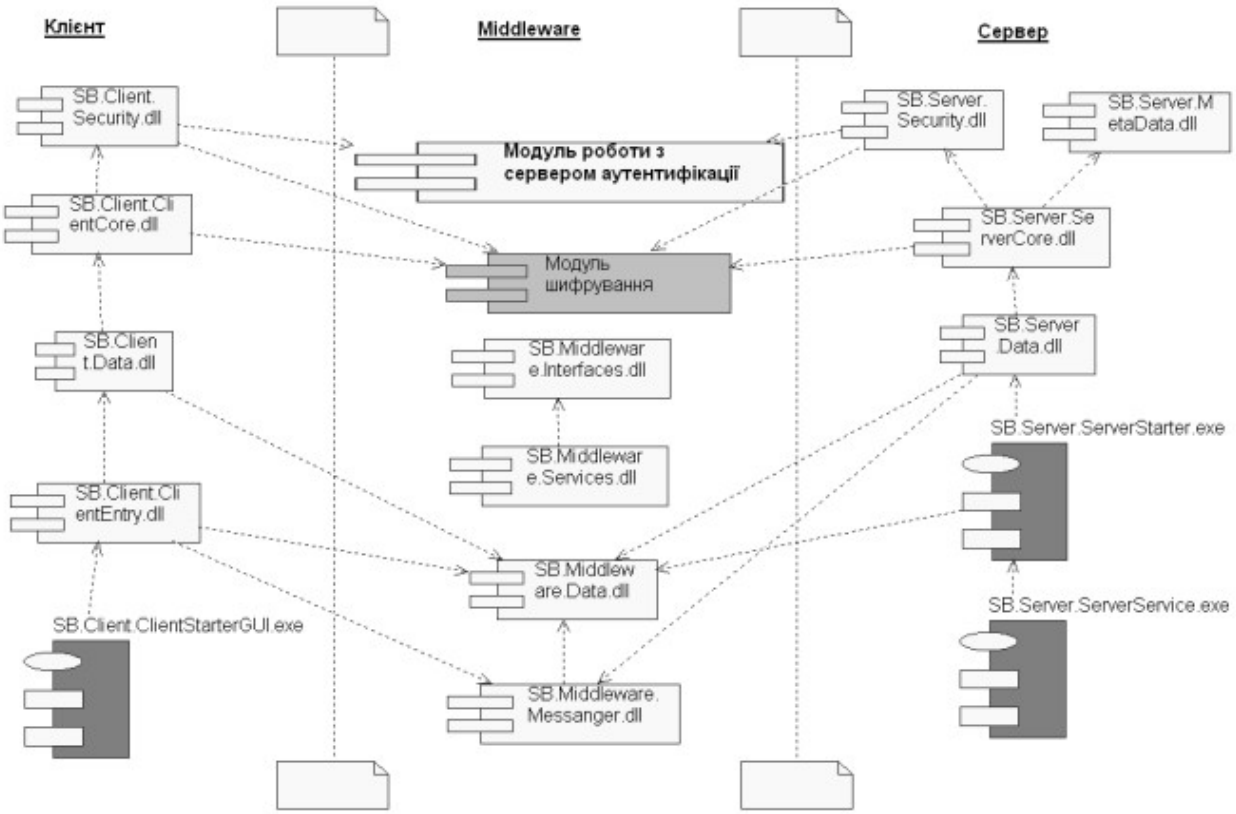

Рисунок 1.2. Приклад діаграми компонентів

1.3. Діаграми послідовностей

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML (Unified Modeling Language), який використовується для моделювання взаємодії між об'єктами системи у певній послідовності часу.

Вона відображає, як об'єкти обмінюються повідомленнями, показуючи порядок і логіку виконання операцій. Діаграма складається з таких основних елементів: Актори (Actors): Зазвичай позначаються піктограмами або назвами. Це користувачі чи інші системи, які взаємодіють із системою. Актори можуть бути зовнішніми стосовно моделювання системи. Об'єкти або класи: Розміщуються горизонтально на діаграмі. Вони позначаються прямокутниками з іменем об'єкта або класу під прямокутником. Кожен об'єкт має «життєвий цикл», який представлений вертикальною пунктирною лінією (лінія життя). Повідомлення: Це лінії зі стрілками, які з'єднують об'єкти. Вони показують передачу повідомлень чи виклик методів. Стрілка може бути синхронною (звичайна стрілка) або асинхронною (лінія з відкритим трикутником) та з пунктирною лінією, що показує повернення результату. Активності: Вказують періоди, протягом яких об'єкт виконує певну дію. На діаграмі це позначається прямокутником, накладеним на лінію життя. Контрольні структури: Використовуються для відображення умов, циклів або альтернативних сценаріїв. Наприклад, блоки "alt" (альтернатива) або "loop" (цикл). Приклад діаграми послідовності зображений на рисунку 1.3.

Рисунок 1.3. Діаграма послідовності «Запит остатку на рахунку»

2 Хід роботи

2.1. Діаграма компонентів

Основні компоненти:

- UI Module
    - Форма налаштувань (SettingsForm).
    - Форма перегляду результатів (ResultsForm).
- Crawler Core
    - CrawlerService — керується логікою сканування.
    - UrlScheduler — планувальник URL.
    - PageFetcher — завантаження сторінок.
    - ContentProcessor — очищення і вилучення тексту.
- Data Access Layer
    - ProfileRepository
    - SessionRepository
    - PageRepository
- Statistics Module
    - StatisticsService
- External Services
    - HttpClient (бібліотека для HTTP-запитів).
    - Database (СУБД, наприклад PostgreSQL).

Зв’язки:

- UI Module використовує Crawler Core та Statistics Module.

- Crawler Core використовує Data Access Layer та HttpClient.
- Data Access Layer взаємодіє з Database.

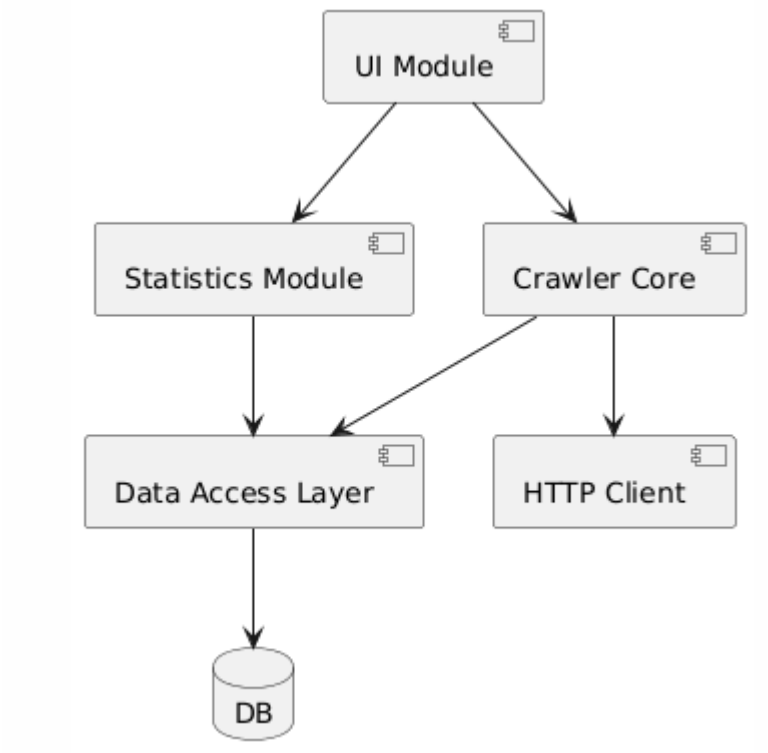

Рисунок 2.1. - Діаграма компонентів

2.2 Діаграма розгортання

Вузли:

1. Client Machine
    - Компоненти: UI Module.
2. Application Server
    - Компоненти: Crawler Core, Data Access Layer, Statistics Module.
3. DB Server
    - Компонент: Database.

Між вузлами:

- Між Client Machine та Application Server — з’єднання по HTTP/HTTPS або іншому протоколу (наприклад, REST API).

- Між Application Server та DB Server — з’єднання по протоколу СУБД (TCP/IP).

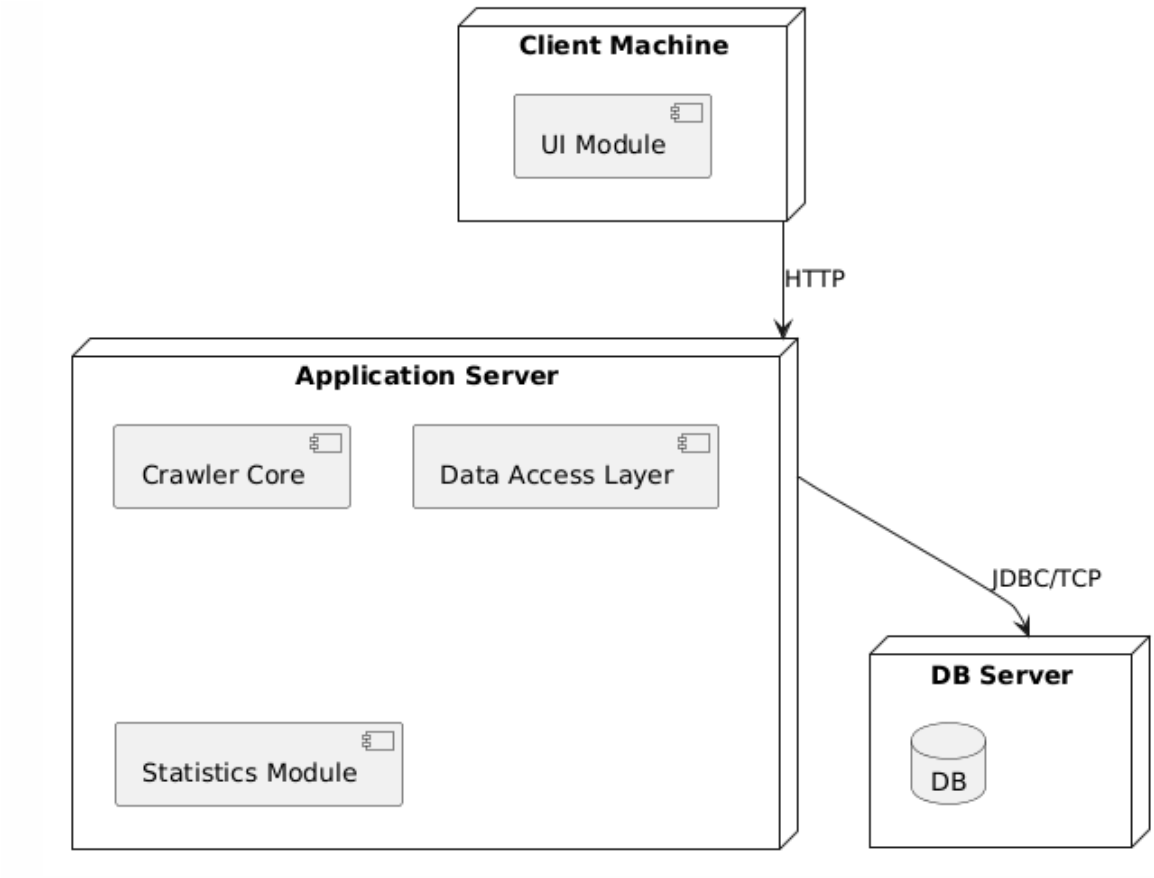

Рисунок 2.2. - Діаграма розгортання

2.3. Діаграми послідовностей

2.3.1. «Запуск сканування»

Учасники: User, SettingsForm, CrawlerService, ProfileRepository, UrlScheduler, PageFetcher, ContentProcessor, PageRepository, SessionRepository.

Основна послідовність:

1. User → SettingsForm: запит на запуск сканування.

2. SettingsForm → ProfileRepository: getProfile(profileId).

3. ProfileRepository → SettingsForm: повертає CrawlProfile.

4. SettingsForm → CrawlerService: startScan(profile).

5. CrawlerService → SessionRepository: createSession(profile).
6. CrawlerService → UrlScheduler: enqueueStartUrls(profile.startUrls).
7. Повторювана частина:
    - CrawlerService → UrlScheduler: nextUrl().
    - UrlScheduler → CrawlerService: URL або null.
    - CrawlerService → PageFetcher: fetch(url).
    - PageFetcher → CrawlerService: HtmlPage.
    - CrawlerService → ContentProcessor: process(page).
    - ContentProcessor → CrawlerService: ParsedPage.
    - CrawlerService → PageRepository: save(parsedPage).
8. Після завершення:
    - CrawlerService → SessionRepository: closeSession(sessionId).

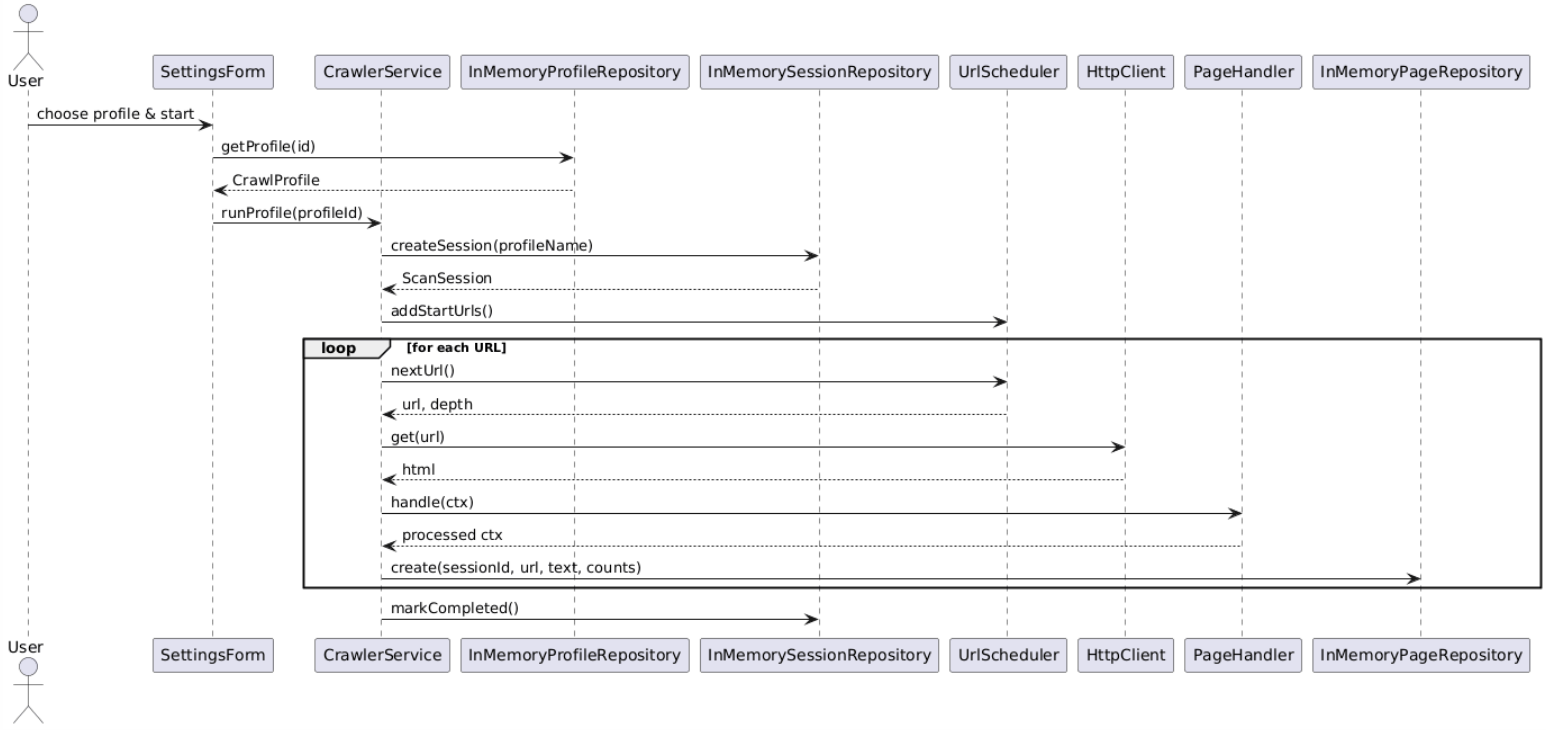

Рисунок 2.3.1. - Діаграма послідовності.

«Запуск сканування»

2.3.2. «Перегляд результатів»

Учасники: User, ResultsForm, SessionRepository, PageRepository, StatisticsService.

1. User → ResultsForm: вибір сеансу сканування.
2. ResultsForm → SessionRepository: getSessionDetails(sessionId).
3. SessionRepository → ResultsForm: деталі сеансу.
4. ResultsForm → PageRepository: getPagesBySession(sessionId).
5. PageRepository → ResultsForm: список сторінок.
6. Опційно:
   ResultsForm → StatisticsService: calculateStats(sessionId)
   StatisticsService → ResultsForm: агреговані статистичні дані.

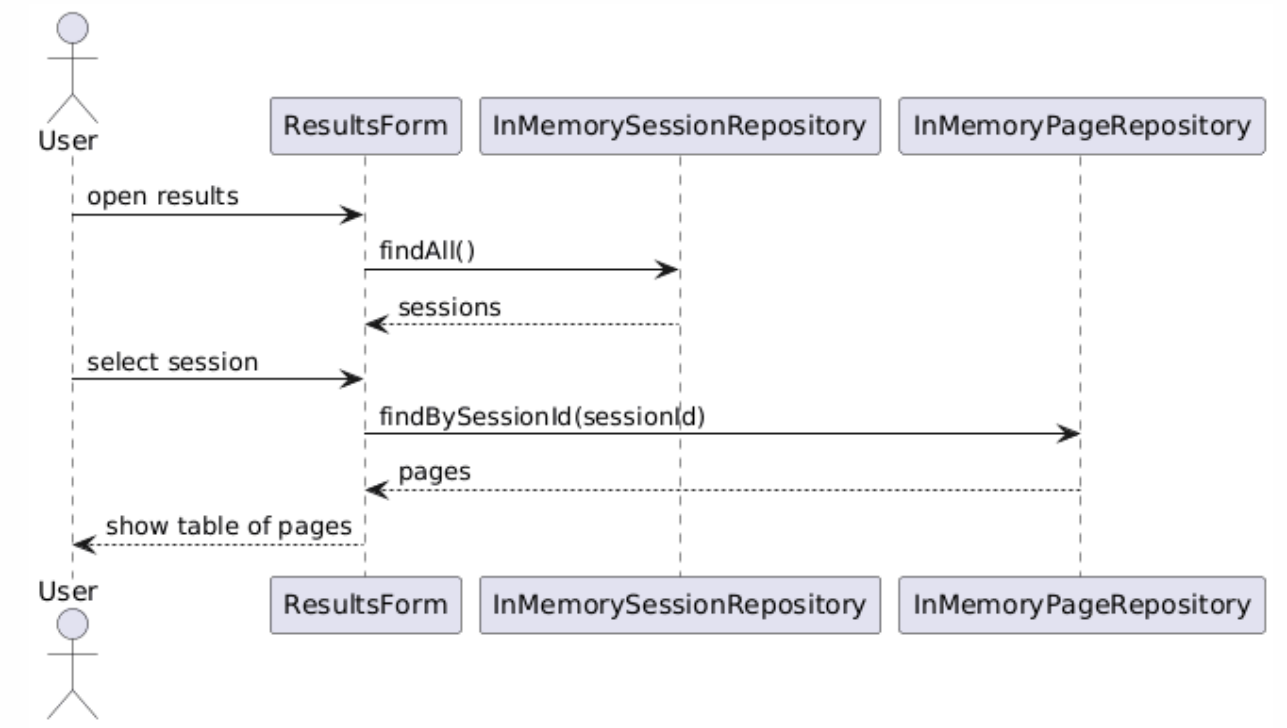

Рисунок 2.3.2. - Діаграма послідовності.

«Перегляд результатів»

2.4. Візуальні форми

Реалізовано два візуальні інтерфейси (наприклад, JavaFX / Swing / WPF):

1. Форма налаштувань сканування:
    - Поля для стартових URL, глибини, фільтру за доменом, ключових слів.
    - Кнопки: «Зберегти профіль», «Запустити сканування».
2. Форма перегляду результатів:
    - Список сеансів сканування.
    - Табличний список сторінок (URL, статус, фрагмент тексту).
    - Панель з деталями вибраної сторінки та кнопкою «Експорт».

2.5 Код

ResultsFrame

```
import javax.swing.*;
import java.awt.*;

public class ResultsFrame extends JFrame {

  private final JTable table;

  public ResultsFrame() {
    super("Crawler Results");
    setSize(600, 400);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    String[] columns = {"URL", "Keyword", "Count"};
    Object[][] data = {};
    table = new JTable(data, columns);

    getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);
  }
}
```

SettingsFrame

```
import javax.swing.*;
import java.awt.*;

public class SettingsFrame extends JFrame {

  public SettingsFrame() {
    super("Crawler Settings");
    setSize(400, 300);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    JPanel panel = new JPanel(new GridLayout(0, 2, 5, 5));
    panel.add(new JLabel("Start URL:"));
    panel.add(new JTextField("https://example.org/"));
    panel.add(new JLabel("Max depth:"));
    panel.add(new JTextField("1"));
    panel.add(new JLabel("Domain filter:"));
    panel.add(new JTextField("example.org"));
    panel.add(new JLabel("Keywords:"));
    panel.add(new JTextField("example,domain"));

    JButton btnSave = new JButton("Save profile");
    JButton btnRun = new JButton("Run crawl");

    JPanel buttons = new JPanel();
    buttons.add(btnSave);
    buttons.add(btnRun);

    getContentPane().add(panel, BorderLayout.CENTER);
    getContentPane().add(buttons, BorderLayout.SOUTH);
  }
}
```

Main

```
package ua.kpi.webcrawler;

import ua.kpi.webcrawler.core.CrawlerService;
import ua.kpi.webcrawler.core.DefaultContentProcessorFactory;
import ua.kpi.webcrawler.http.HttpClient;
import ua.kpi.webcrawler.http.HttpClientProxy;
import ua.kpi.webcrawler.http.RealHttpClient;
import ua.kpi.webcrawler.model.CrawlProfile;
import ua.kpi.webcrawler.repository.InMemoryPageRepository;
import ua.kpi.webcrawler.repository.InMemoryProfileRepository;
import ua.kpi.webcrawler.repository.InMemorySessionRepository;
```

```java
import java.util.List;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) throws Exception {
    System.out.println("Web Crawler Labs Project (Topic 11: Web crawler)");
    InMemoryProfileRepository profileRepo = new InMemoryProfileRepository();
    InMemorySessionRepository sessionRepo = new InMemorySessionRepository();
    InMemoryPageRepository pageRepo = new InMemoryPageRepository();

    // Create default profile
    CrawlProfile defaultProfile = new CrawlProfile(
        1L,
        "Default profile",
        List.of("https://example.org/"),
        1,
        "example.org",
        List.of("example", "domain")
    );
    profileRepo.save(defaultProfile);

    HttpClient realClient = new RealHttpClient();
    HttpClient proxyClient = new HttpClientProxy(realClient, "proxy.example.local",
8080);

    CrawlerService crawlerService = new CrawlerService(
        proxyClient,
        new DefaultContentProcessorFactory(),
        profileRepo,
        sessionRepo,
        pageRepo
    );

    Scanner scanner = new Scanner(System.in);
    while (true) {
      System.out.println();
      System.out.println("1. Run crawl with default profile");
      System.out.println("2. Show sessions");
      System.out.println("3. Exit");
      System.out.print("Choose option: ");
      String line = scanner.nextLine();
      if ("1".equals(line)) {
        crawlerService.runProfile(defaultProfile.getId());
      } else if ("2".equals(line)) {
        System.out.println("Sessions:");
        sessionRepo.findAll().forEach(s -> {
          System.out.println("Session " + s.getId() + " profile=" +
s.getProfileName()
              + " status=" + s.getStatus());
        });
      } else if ("3".equals(line)) {
        break;
```

```
        }
    }
    System.out.println("Bye.");
  }
}
```

Висновки

У межах лабораторної роботи було спроєктовано діаграму компонентів, діаграму розгортання та діаграми послідовностей для системи web crawler. Отримані артефакти дозволили уточнити структуру програмних компонентів, способи їхнього фізичного розміщення на вузлах та детальну взаємодію між об’єктами під час виконання основних сценаріїв. Було визначено склад і ролі серверної частини та клієнтського інтерфейсу, а також підготовлено основу для подальшої реалізації функціоналу.

Контрольні запитання

1. Що собою становить діаграма розгортання?

Діаграма розгортання (Deployment Diagram) — це UML-діаграма, що показує фізичне розміщення програмних компонентів на апаратних вузлах та серверній інфраструктурі. Вона описує, де саме виконуються компоненти системи.

2. Які бувають види вузлів на діаграмі розгортання?

На діаграмі можуть бути такі вузли:

- Апаратні вузли (сервер, комп'ютер, мобільний пристрій).
- Віртуальні вузли (контейнери, віртуальні машини).
- Вкладені вузли (вузли всередині інших вузлів — наприклад, Docker у сервері).
- Виконувані середовища (JVM, .NET CLR, веб-сервери).

3. Які бувають зв'язки на діаграмі розгортання?

Основні типи зв'язків:

- Комунікаційний зв'язок (communication path) — показує мережеву взаємодію між вузлами.
- Зв'язок розгортання (deployment) — показує, який компонент розгорнуто на якому вузлі.

4. Які елементи присутні на діаграмі компонентів?

Діаграма компонентів містить:

- Компоненти (модулі, підсистеми, бібліотеки).
- Інтерфейси (надавані та потрібні).
- Пакети або підсистеми.
- Залежності між компонентами.
- Артефакти, якщо вони пов'язані зі збиранням системи.

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

Зв'язки показують залежність компонентів один від одного:

- хто використовує інтерфейс іншого;
- хто надає функціональність;

- які модулі взаємодіють між собою логічно або структурно.

6. Які бувають види діаграм взаємодії?

До діаграм взаємодії належать:

- Діаграма послідовностей (Sequence Diagram)
- Діаграма комунікації (Communication/Collaboration Diagram)
- Діаграма часових характеристик (Timing Diagram)
- Діаграма взаємодій (Interaction Overview Diagram)

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей показує порядок взаємодії об’єктів у часі, тобто який об’єкт і коли відправляє повідомлення іншому. Вона демонструє логіку виконання одного сценарію використання.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Основні елементи:

- Актори
- Об’єкти або класи
- Життєві лінії (lifelines)
- Повідомлення (messages)
- Активізації (activation boxes)
- Умовні або альтернативні блоки (alt, opt, loop)
- Повернення результатів

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

Діаграми послідовностей деталізують кожен сценарій з діаграми варіантів використання: кожен use case → має свою sequence diagram, яка показує, як саме виконується сценарій.

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

Діаграми послідовностей показують які об’єкти взаємодіють та які методи вони викликають, а діаграма класів описує структуру цих об’єктів.
Тобто sequence diagram допомагає визначити:

- які методи мають бути у класах;
- які зв’язки потрібні між класами;
- які об’єкти створюються під час виконання сценарію.