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

Технології розроблення програмного забезпечення

«ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ. ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ »

Виконала
студентка групи ІА–22:
Фоменко Альона

Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Зміст

Тема: ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ. ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ

Мета: Розробити діаграму розгортання для проектованої системи. Розробити діаграму компонентів для проектованої системи. Розробити діаграму послідовностей для проектованої системи.

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграми розгортання представляють фізичне розташування системи, показуючи, на якому фізичному устаткуванні запускається та чи інша складова програмне забезпечення.

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 може включати інше програмне забезпечення, наприклад, операційну систему або процес-контейнер (наприклад, веб-сервер).

Між вузлами можуть стояти зв'язки, які зазвичай зображуються у вигляді прямої лінії.

Як і на інших діаграмах, зв'язки можуть мати атрибути множинності (для показання, наприклад, підключення 2х і більше клієнтів до одного сервера) та назва.

У назві зазвичай міститься спосіб зв'язку між двома вузлами — це може бути назва протоколу (http, IPC) чи використовувана технологія забезпечення взаємодії вузлів (. Remoting, WCF).

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

Артефакти можна зображати у вигляді прямокутників класів або перераховувати їхні імена усередині вузла. Якщо ви показуєте ці елементи у вигляді прямокутників класів, можна додати значок документа або ключове слово «artifact». Можна супроводжувати вузли або артефакти значеннями у вигляді міток, щоб вказати різну цікаву інформацію про сайт, наприклад постачальника, операційну систему, місце - загалом, все, що прийде вам на думку.

Часто у вас буде безліч фізичних вузлів для вирішення однієї і тієї ж логічного завдання. Можна відобразити цей факт, намалювавши безліч прямокутників вузлів або поставивши число у вигляді значення-мітки.

Артефакти часто є реалізацією компонентів. Це можна показати, задавши значення-мітки всередині прямокутників артефактів.

Основні види артефактів:

вихідні файли;

• виконувані файли;

• сценарії;

• таблиці баз даних;

• документи;

• результати розробки, UML-моделі.

Можна також деталізувати артефакти, що входять у вузол; наприклад, додатково всередині файлу, що розгортається вказати, які туди входять компоненти чи класи. Така деталізація, як правило, не має сенсу на діаграмах розгортання, оскільки може зміщувати фокус уваги від моделі розгортання програмного забезпечення для його внутрішнього пристрою, однак іноді може бути корисною. При цьому, можливо встановлювати зв'язки між компонентами/класами межах різних вузлів.

Діаграми розгортань розрізняють двох видів: описові та екземплярні. На діаграмах описової форми вказуються вузли, артефакти та зв'язку між вузлами без зазначення конкретного обладнання чи програмного забезпечення, необхідного для розгортання. Такий вид діаграм корисний на ранніх етапах розробки для розуміння, які впринципі фізичні пристрої необхідні для функціонування системи або для опису процесу розгортання у загальному ключі.

Діаграми екземплярної форми несуть у собі екземпляри обладнання, артефактів та зв'язків між ними. Під екземплярами розуміються конкретні елементи - ПК з відповідним набором характеристик та встановленим ПЗ; цілком можливо, в межах однієї організації це може бути якийсь конкретний вузол (наприклад, ПК тестувальника Василя). Діаграми екземплярної форми розробляються на завершальних стадіях розробки ПЗ - коли вже відомі і сформульовано вимоги до програмного комплексу, обладнання закуплено та все готове до розгортання. Діаграми такої форми є скоріше планом розгортання у графічному вигляді, ніж модель розгортання.

Хід роботи

Тема 11: Web crawler (proxy, chain of responsibility, memento, template method, composite, p2p)

Веб-сканер повинен вміти розпізнавати структуру сторінок сайту, переходити за посиланнями, збирати необхідну інформацію про зазначений термін, видаляти не семантичні одиниці (рекламу, об'єкти javascript і т.д.), зберігати знайдені дані у вигляді структурованого набору html файлів вести статистику відвіданих сайтів і метадані.

Частина функціональності системи

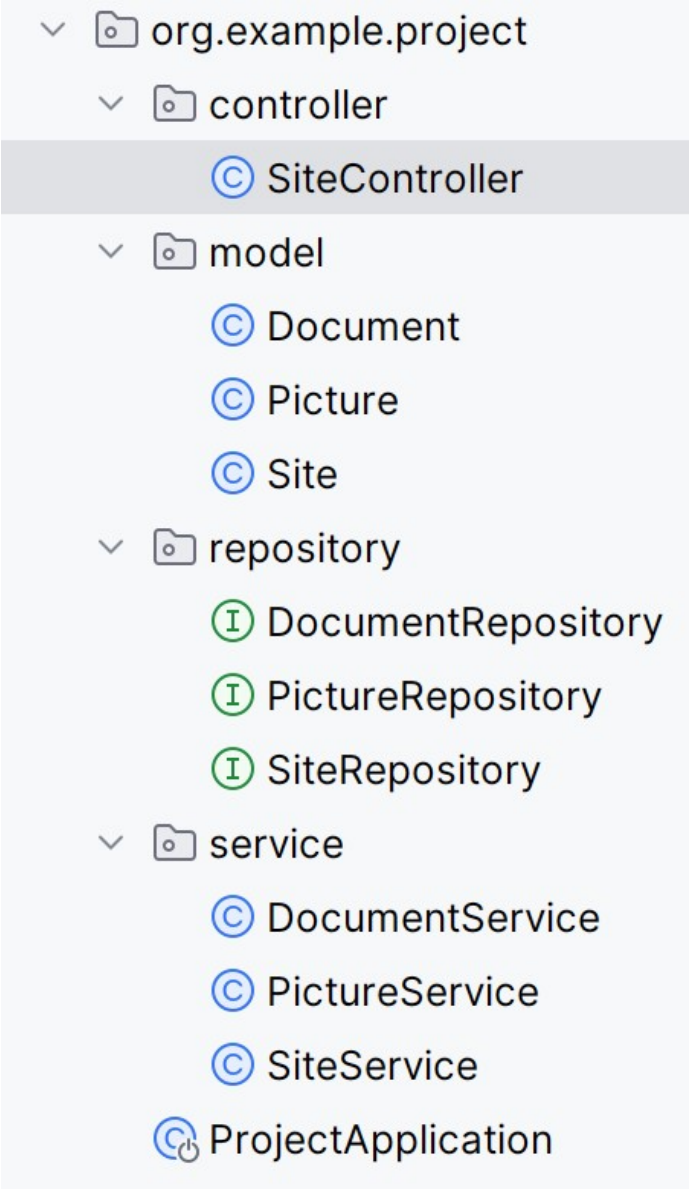

Рис. 1: Частина функціональної системи

Ця структура проекту організована по пакетах у наступний спосіб:

1. controller
    - Містить клас SiteController, який відповідає за обробку вхідних HTTP-запитів і передає їх відповідним сервісам для подальшої обробки. Це рівень контролера в архітектурі, який взаємодіє з клієнтами через API.
2. model
    - Містить класи Document, Picture та Site, які представляють основні сутності або об'єкти даних у системі. Ці класи, зазвичай, містять атрибути та методи, необхідні для моделювання об'єктів бази даних.
3. repository
    - Містить інтерфейси DocumentRepository, PictureRepository та SiteRepository, які забезпечують доступ до бази даних. Ці репозиторії використовуються для збереження, оновлення, видалення та отримання даних про документи, зображення та сайти.
4. service
    - Містить сервіси DocumentService, PictureService та SiteService, які реалізують бізнес-логіку системи. Вони працюють з відповідними репозиторіями для виконання операцій над даними та надають функціонал, який використовується контролером.
5. ProjectApplication
    - Основний клас програми, який запускає додаток. Зазвичай, цей клас ініціалізує контекст програми та запускає вбудований сервер для обробки запитів.

Діаграма розгортання

На цій діаграмі розгортання можна зобразити архітектуру краулера, що складається з клієнтської та серверної частин.

Клієнтська частина представлена як пристрій (ClientPC), який містить артефакт `jarClient`, що позначає виконуваний файл клієнтської частини краулера. Цей компонент відповідає за відправлення HTTP-запитів до сервера з метою отримання даних для аналізу.

Серверна частина представлена як пристрій (Server), що містить артефакт `jarServer` — виконуваний файл серверної частини краулера, а також базу даних, позначену як Database (DB). Основні функції серверної частини включають обробку HTTP-запитів від клієнта, виконання SQL-запитів до бази даних та надсилання HTTP-відповідей з результатами обробки запитів назад до клієнта.

Взаємодія між компонентами включає:

- Надсилання HTTP-запитів клієнтом до сервера, щоб ініціювати процес збору або обробки даних.
- Отримання HTTP-відповідей від сервера, які можуть містити інформацію про стан виконання запиту або результати краулінгу.
- Виконання сервером SQL-запитів до бази даних для збереження, оновлення або отримання інформації, необхідної для подальшого аналізу.

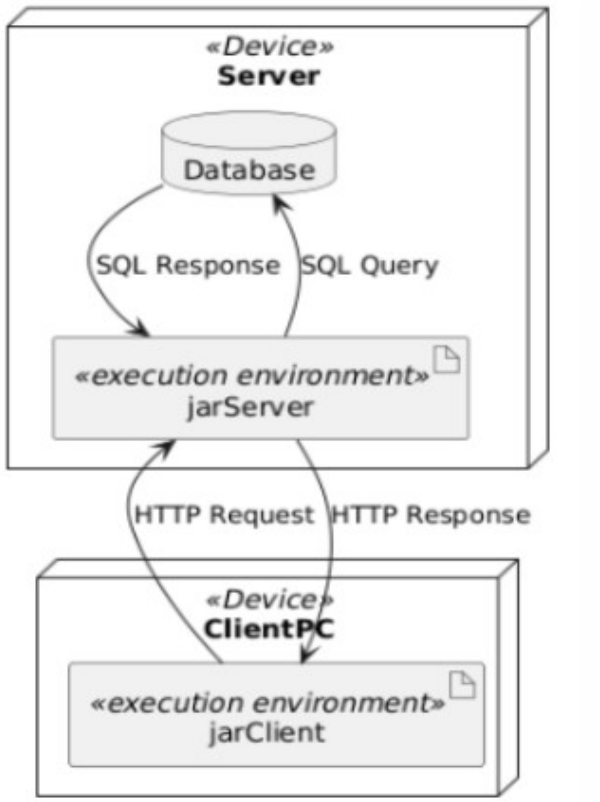

Рис. 2: Діаграма розгортання

Діаграма послідовностей

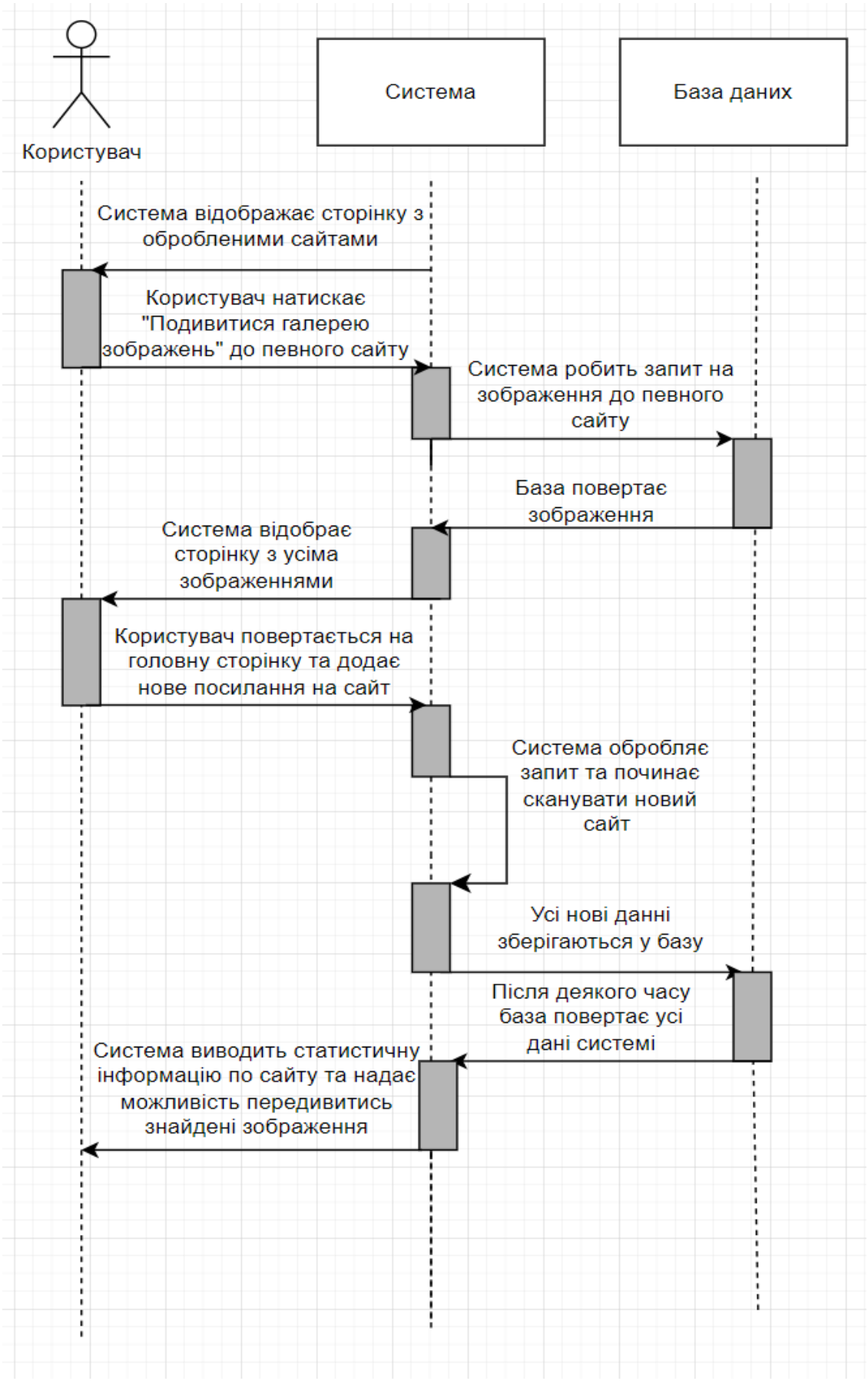

Рис. 3: Діаграма послідовностей

Відображення сторінки з обробленими сайтами:

- Система показує користувачу сторінку, де перераховані сайти, з яких вже зібрано зображення.

Запит на перегляд галереї зображень:

- Користувач натискає на кнопку "Подивитися галерею зображень" для вибраного сайту.
- Система робить запит до бази даних, щоб отримати зображення з обраного сайту.

Отримання зображень із бази даних:

- База даних повертає зображення, і система відображає їх користувачу у вигляді галереї.

Додавання нового посилання на сайт:

- Користувач повертається на головну сторінку та додає нове посилання на сайт, який потрібно сканувати для зображень.

Сканування нового сайту:

- Система приймає запит, починає обробку та сканує новий сайт на наявність зображень.
- Усі знайдені зображення зберігаються в базі даних.

Оновлення даних у базі:

- Після завершення процесу сканування база даних повертає оновлені дані системі.

Виведення статистики та зображень:

- Система виводить статистичну інформацію по обробленому сайту та надає користувачу можливість переглянути знайдені зображення.

Діаграма компонентів

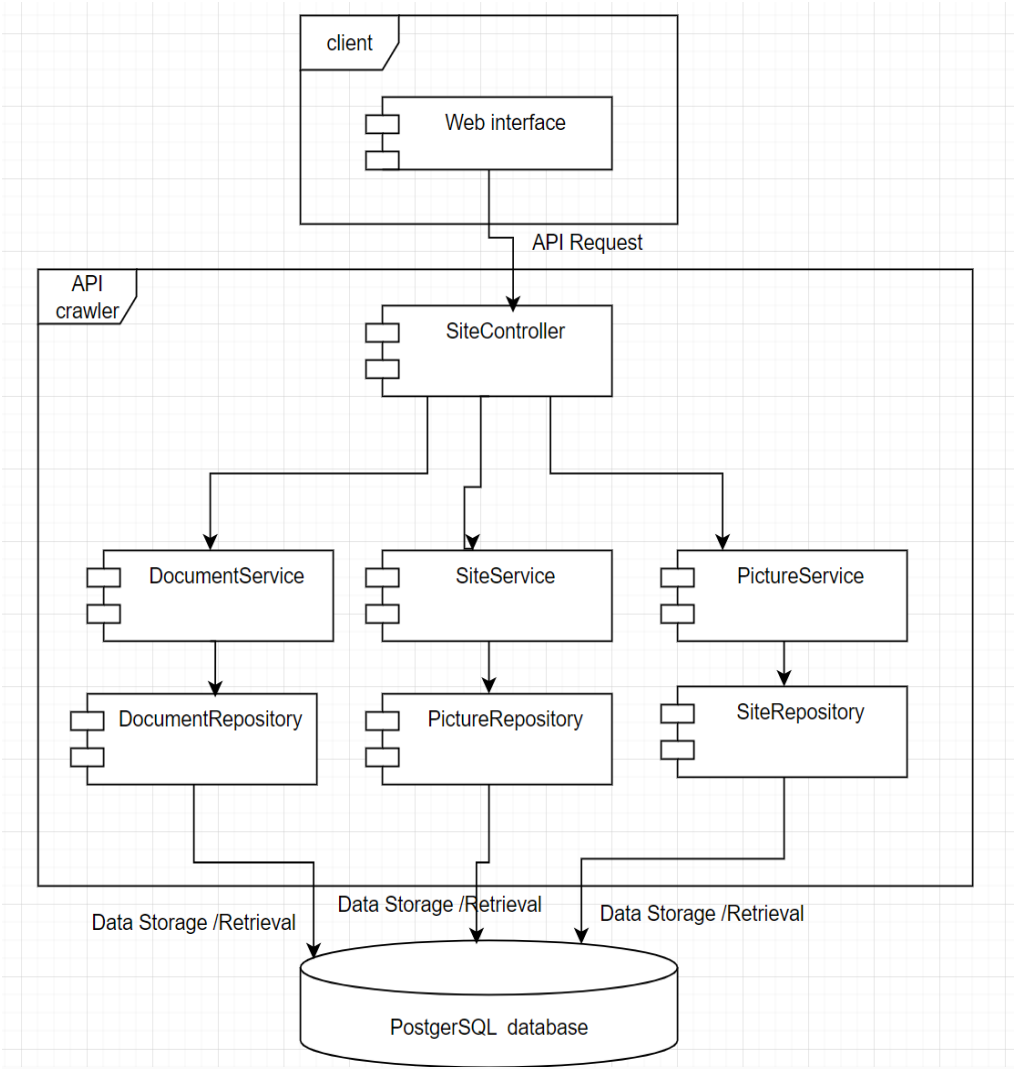

Рис. 4: Діаграма компонентів

Ця діаграма ілюструє архітектуру системи краулера з використанням API, яка складається з кількох компонентів для збору, збереження та обробки даних про сайти, документи та зображення.

Основні компоненти:

1. Client (Клієнт):
    - Має доступ до Web interface (веб-інтерфейсу), через який користувач може надсилати запити до API краулера.
2. API Crawler (API краулер):

- SiteController: Основний контролер, що приймає запити від клієнта через веб-інтерфейс. Контролер керує потоком даних між клієнтом і сервісами.
- DocumentService, SiteService, PictureService: Ці сервіси обробляють дані відповідно до їхніх специфічних задач:
    - DocumentService працює з документами сайту.
    - SiteService відповідає за інформацію про сайт.
    - PictureService займається зображеннями.
- DocumentRepository, SiteRepository, PictureRepository: Репозиторії для кожного типу даних забезпечують зберігання та доступ до відповідних даних, обслуговуючи сервіси.

3. PostgreSQL Database (База даних PostgreSQL):
    - Служить для зберігання даних, таких як інформація про сайти, документи та зображення.

Висновок: розробка діаграм розгортання, компонентів та послідовностей допомагає створити чітке уявлення про архітектуру, логіку та взаємодію елементів системи. Це дозволяє оптимізувати розподіл ресурсів, структурувати компоненти, перевірити послідовність операцій та уникнути потенційних помилок. У результаті, така моделююча робота підвищує ефективність проєктування, сприяє злагодженій роботі команди та забезпечує надійність і масштабованість майбутньої системи.