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 №2

Технології розроблення програмного забезпечення

«ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»

Виконала
студентка групи ІА–22:
Фоменко Альона

Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Тема: Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи

Мета: Проаналізувати тему, намалювати схему прецеденту, діаграму класів, розробити основні класи і структуру бази.

Теоретичні відомості

Діаграма варіантів використання (Use-Cases Diagram)

Діаграма варіантів використання (Use-Cases Diagram) - це UML діаграма за допомогою якої в графічному вигляді можна зобразити вимоги до системи, що розробляється. Діаграма варіантів використання – це вихідна концептуальна модель проектованої системи, вона описує внутрішній устрій системи.

Діаграми варіантів використання призначені для:

1.Визначення загальної межі функціональності проектованої системи;
2.Сформулювати загальні вимоги до функціональної поведінки проектованої системи

3. Розробка вихідної концептуальної моделі системи;

4.Створення основи для виконання аналізу, проектування, розробки та тестування.

Діаграми варіантів використання є відправною точкою при збиранні вимог до програмного продукту та його реалізації. Ця модель будується на аналітичному етапі побудови програмного продукту (збір та аналіз вимог) і дозволяє бізнес-аналітикам отримати більш повне уявлення про необхідне програмне забезпечення та документувати його.

6 Технології розроблення ПЗ

Діаграма варіантів використання складається з низки елементів. Основними елементами є: варіанти використання або прецеденти (use case), актор або дійова особа (actor) та відносини між акторами та варіантами використання (relationship).

Актори (actor)

Актором називається будь-який об'єкт, суб'єкт чи система, що взаємодіє з модельованою бізнессистемою ззовні для досягнення своїх цілей або вирішення певних завдань. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка є джерелом впливу на модельовану систему.

Зображення акторів на діаграмах UML

Ім'я актора має бути достатньо інформативним з точки зору програмного, що розробляється. забезпечення або предметної галузі. Для цієї мети підходять найменування посад у компанії

(Наприклад, продавець, касир, менеджер, президент).

Варіанти використання (use case)

Варіант використання служить для опису служб, які система надає актору. Іншими словами кожен варіант використання визначає набір дій, який здійснюється системою при діалозі з актором. Кожен варіант використання є послідовністю дій, який повинен бути виконаний проектованою системою при взаємодії її з відповідним актором, самі ці події не відображаються на діаграмі.

Варіант використання відображається еліпсом, всередині якого міститься його коротке ім'я з великою літери у формі іменника або дієслова.

Позначення варіанта використання

Приклади варіантів використання: реєстрація, авторизація, оформлення замовлення, переглянути замовлення, перевірка стану поточного рахунку тощо.

Відносини на діаграмі варіантів використання

Ставлення (relationship) – семантичний зв'язок між окремими елементами моделі. Один актор може взаємодіяти з кількома варіантами використання. У цьому випадку цей актор звертається до кількох служб цієї системи. У свою чергу один варіант використання може взаємодіяти з кількома акторами, надаючи всім їх свій функціонал.

7 Технології розроблення ПЗ

Існують такі відносини: асоціації, узагальнення, залежність (складається з включення та розширення).

Асоціація (association) – узагальнене, невідоме ставлення між актором та варіантом використання. Позначається суцільною лінією між актором та варіантом використання.

Спрямована асоціація (directed association) - те саме, що й проста асоціація, але показує, що Випадок використання ініціалізується актором. Позначається стрілкою.

Спрямована асоціація дозволяє запровадити поняття основного актора (він є ініціатором асоціації) та другорядного актора (варіант використання є ініціатором, тобто передає акторові довідкові відомості чи звіт про виконану роботу).

Особливості використання відносини асоціації:

1. Один варіант використання може мати кілька асоціацій з різними акторами.

2.Два варіанти використання, які стосуються одному й тому акторові, неможливо знайти асоційовані, т.к. кожен їх описує закінчений фрагмент функціональності актора.

Сценарії використання

Діаграма варіантів використання надає знання про необхідну функціональності звичайно системи в інтуїтивно-зрозумілому вигляді, проте не несе відомостей про фактичний спосіб її реалізації. Конкретні варіанти

використання можуть звучати занадто спільно і розпливчасто і не є придатними для програмістів.

Для документації варіантів використання у вигляді деякої специфікації та для усунення неточностей та непорозуміння діаграм варіантів використання, як частина процесу збору та аналізу вимог складаються так звані сценарії використання.

Сценарії використання - це текстові уявлення тих процесів, які відбуваються при взаємодії користувачів системи та самої системи. Вони є чітко формалізованими, покроковими інструкціями, що описують той чи інший процес у термінах кроків досягнення мети. Сценарії використання однозначно визначають кінцевий результат.

Діаграми UML. Діаграми класів. Концептуальна модель системи

Діаграми класів використовуються при моделюванні ПС найчастіше. Вони є однією з форм статичного опису системи з погляду її проектування, оказуючи її структуру. Діаграма класів не відображає

Технології розроблення ПЗ динамічна поведінка об'єктів зображених у ній класів. на діаграмах Класи показують класи, інтерфейси і відносини між ними.

Подання класів

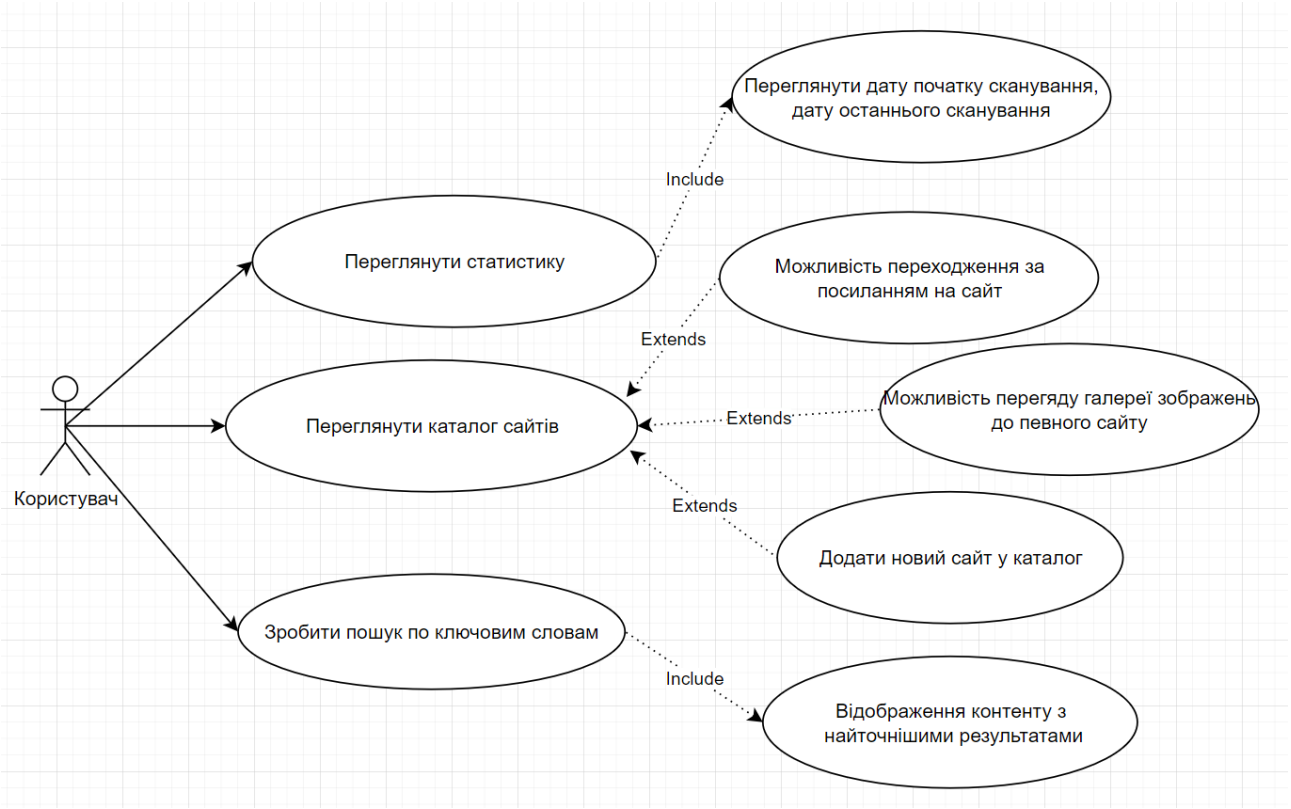
Клас – це основний будівельний блок ПС. Це поняття є і в ГО мовами програмування, то є між класами UML та програмними класами є відповідність, що є основою для автоматичної генерації програмних кодів або для виконання реінжинірингу. Кожен клас має назву, атрибути та операції. Клас на діаграмі показується у вигляді прямокутника, розділеного на 3 області. У верхній міститься назва класу, у середній – опис атрибутів (властивостей), у нижній – назви операцій – послуг, які надаються об'єктами цього класу.

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.

Діаграма прецедентів



Оберемо 3 прецеденти і напишемо для них сценарії використання

Прецедент: Переглянути статистику

Актор: Користувач
Опис: Користувач переглядає статистику відвідувань або взаємодії з вебсайтами.
Основний сценарій:

1. Користувач відкриває розділ зі статистикою.
2. Система завантажує актуальні дані зі статистики.
3. Користувач переглядає дані: кількість відвідувань, тривалість сеансів тощо.
4. Користувач може переглянути дату початку сканування, дату останнього сканування

Альтернативний сценарій:

Якщо дані недоступні (проблеми з мережею або відсутність даних), система відображає повідомлення про помилку та пропонує повторити запит пізніше.

Прецедент: Переглянути каталог сайтів

Актор: Користувач
Опис: Користувач переглядає каталог збережених сайтів, може перейти на будь-який із них, додати новий сайт або подивитися галерею зображень до сайту.
Основний сценарій:

1. Користувач відкриває розділ каталогу сайтів через головне меню або панель навігації.
2. Система завантажує та відображає список сайтів, які збережені в каталозі, разом із коротким описом кожного сайту.
3. Користувач переглядає доступні сайти і може:
    - Сортувати або фільтрувати список за категоріями, популярністю, або іншими критеріями.
    - Переглянути галерею зображень, якщо вона доступна для певного сайту.
4. Користувач обирає сайт для перегляду та натискає на його назву або зображення.
5. Система відкриває вибраний сайт у новій вкладці браузера.
6. Якщо користувач бажає додати новий сайт до каталогу:
    - Користувач натискає на кнопку "Додати новий сайт".
    - Система відкриває форму для введення інформації про новий сайт (назва, URL, опис, категорія, зображення тощо).
    - Користувач заповнює форму та підтверджує додавання.
    - Система зберігає новий сайт у каталозі та відображає повідомлення про успішне додавання.

Альтернативний сценарій:

1. Якщо сайт недоступний:
    - Система відображає повідомлення про те, що сайт недоступний або не відповідає, і пропонує переглянути інші сайти або повторити спробу пізніше.
2. Якщо користувач хоче переглянути галерею зображень до сайту:
    - Користувач натискає на опцію "Переглянути галерею".
    - Система відкриває вікно або сторінку з галереєю зображень, пов'язаних із сайтом.
    - Користувач переглядає зображення та може повернутися до списку сайтів.
3. Якщо додавання сайту не вдалось (через помилку або неправильний формат даних):
    - Система відображає повідомлення про помилку та пропонує користувачу виправити дані (наприклад, неправильний URL) або повторити спробу пізніше.

Прецедент: Зробити пошук за ключовими словами

Актор: Користувач
Опис: Користувач шукає інформацію за ключовими словами в каталозі сайтів або контенті сайтів.
Основний сценарій:

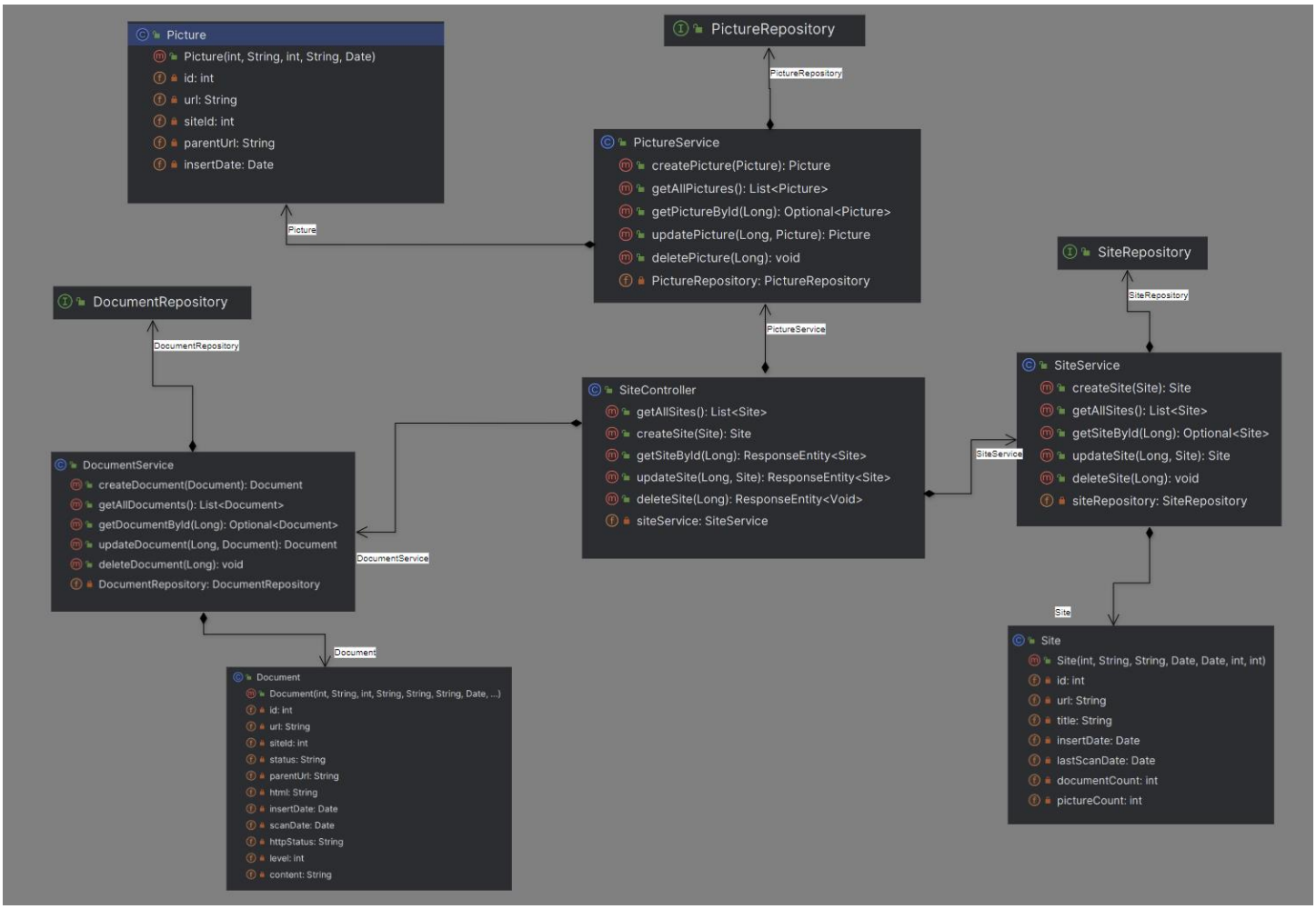
- Користувач відкриває пошукове поле.
- Користувач вводить ключове слово або фразу для пошуку.
- Система обробляє запит та відображає результати пошуку.
- Користувач переглядає результати пошуку.

- Якщо потрібно, користувач може відсортувати або відфільтрувати результати.

Альтернативний сценарій:

- Якщо за запитом не знайдено результатів, система відображає відповідне повідомлення та пропонує користувачу спробувати інші ключові слова.

Діаграма классів



Основні класи:

1. Picture (Зображення) :

- Поля:

- id: int – ідентифікатор зображення.

-url: String – URL зображення.

- iteId: int– ідентифікатор сайту, до якого належить зображення.

- parentUri: String – батьківський URL.

- insertDate: Date – дата додавання зображення.

- Клас використовується для зберігання інформації про зображення.

2. Document (Документ) :

- Поля:

- id: int – ідентифікатор документа.

- url: String – URL документа.

- siteId: int – ідентифікатор сайту, до якого належить документ.

- parentUri: String – батьківський URL.

- insertDate: Date – дата додавання документа.

- Інші поля включають статус HTTP, рівень та зміст документа.

- Цей клас описує документи, пов’язані з певними сайтами.

3. Site (Сайт) :

- Поля:

- id: int – ідентифікатор сайту.

- url: String – URL сайту.

- title: String – назва сайту.

- insertDate: Date – дата додавання сайту.

- Інші поля включають дату останнього сканування, кількість документів і зображень.

- Використовується для представлення інформації про сайт та пов’язані з ним дані.

Репозиторії:

1. PictureRepository :

- Призначений для операцій з об'єктами класу Picture (створення, отримання, оновлення, видалення).

2. DocumentRepository :

- Забезпечує операції над об'єктами класу Document.

3. SiteRepository :

- Виконує операції з об'єктами класу Site.

Сервіси:

1. PictureService :

- Надає методи для роботи із зображеннями, такі як створення, отримання, оновлення і видалення зображень.

- Взаємодіє з PictureRepository.

2. DocumentService :

- Забезпечує функції для роботи з документами, включаючи створення, отримання, оновлення і видалення документів.

- Працює з DocumentRepository.

3. SiteService :

- Містить методи для роботи з сайтами: створення, отримання, оновлення та видалення.

- Взаємодіє з SiteRepository.

Контролер:

1. SiteController :

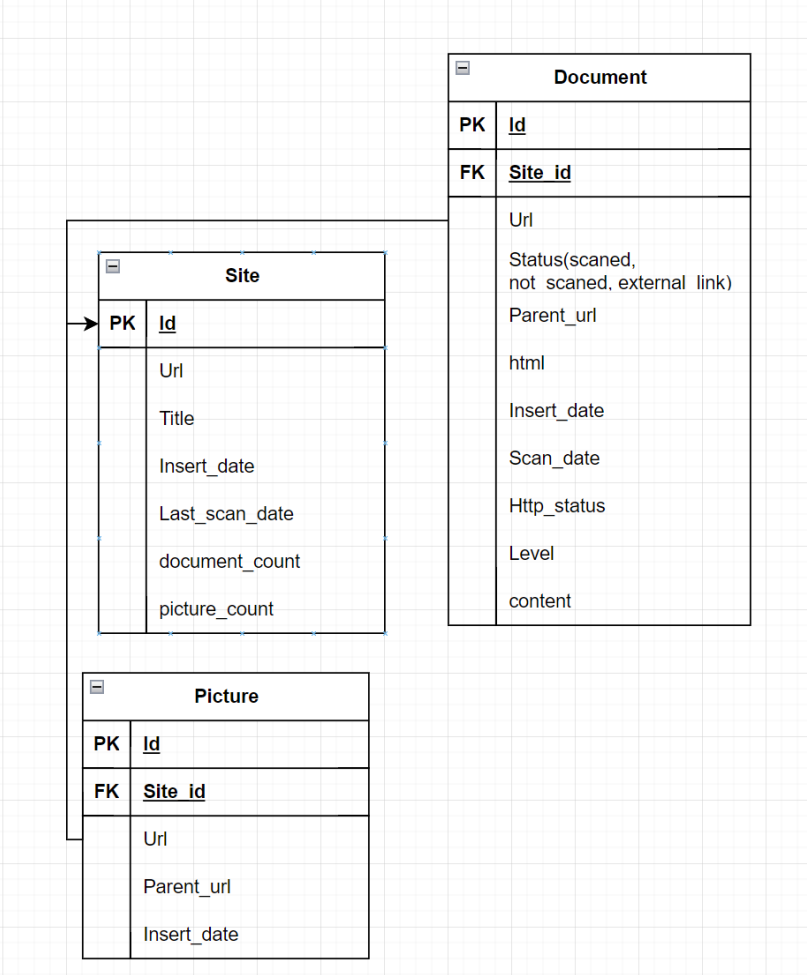
- Надає REST API для операцій з сайтами, таких як створення сайту, отримання всіх сайтів, оновлення і видалення сайтів.

- Використовує SiteService для виконання бізнес-логіки.

Взаємозв'язки:

- Класи Picture, Document та Site пов’язані з відповідними репозиторіями та сервісами, що створює чітку архітектуру, де кожен сервіс відповідає за конкретну бізнес-логіку і взаємодію з базою даних через репозиторій.

Структура бази дани



Опис таблиць:

1. Site (Сайт) :

- Поля:

- Id (PK) – первинний ключ, ідентифікатор сайту.

- Url – URL сайту.

- Title – назва сайту.

- Insert_date – дата додавання сайту в базу даних.

- Last_scan_date – дата останнього сканування сайту.

- document_count – кількість документів, пов’язаних із сайтом.

- picture_count – кількість зображень, пов’язаних із сайтом.

- Таблиця зберігає основну інформацію про сайти, такі як URL, назва та кількість пов'язаних документів та зображень.

2. Document (Документ) :

- Поля:

- Id (PK) – первинний ключ, ідентифікатор документа.

- Site_id (FK) – зовнішній ключ, що посилається на сайт (таблиця Site).

- Url – URL документа.

- Status – статус документа (просканово, не просканово, зовнішнє посилання).

- Parent_url – батьківський URL документа.

- html – HTML-код документа.

- Insert_date – дата додавання документа в базу.

- Scan_date – дата сканування документа.

- Http_status – HTTP-статус відповіді від сервера.

- Level – рівень важливості або інший показник.

- content – вміст документа.

- Таблиця описує документи, пов’язані з певними сайтами, та їхні характеристики, включаючи статус сканування та інші метадані.

3. Picture (Зображення) :

- Поля:

- Id (PK) – первинний ключ, ідентифікатор зображення.

- Site_id (FK) – зовнішній ключ, що посилається на сайт (таблиця Site).

- Url – URL зображення.

- Parent_url – батьківський URL для цього зображення.

- Insert_date – дата додавання зображення в базу.

- Таблиця зберігає інформацію про зображення, які належать певним сайтам, та їхні метадані.

- Таблиці Document і Picture мають зовнішні ключі (Site_id), які пов’язують їх з таблицею Site

- Відносини між таблицями є типовими для реляційних баз даних, де сайт може мати багато документів і зображень, а кожен документ або зображення може належати лише одному сайту.

Висновок: завдяки цій лабораторній роботі ми змогли створити діаграму прецедентів, діаграму класів, а також структурну схему бази данних до майбутнього проекту.