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

Архіватор

Виконав
студент групи ІА – 24:
Любченко І.М

Перевірив:
Мягкий М.Ю

Київ 2024

Зміст

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

Короткі теоритичні відомості Прецеденти (Use Case Diagram)

Діаграма прецедентів використовується для візуалізації функціональних вимог до системи. Вона показує, як користувачі (актор або актори) взаємодіють із системою через певні сценарії використання (прецеденти). Основними елементами діаграми є актори, прецеденти, а також зв'язки між ними. Прецеденти допомагають виявити основні функції системи та забезпечують їх розуміння на високому рівні.

Діаграма класів (Class Diagram)

Діаграма класів використовується для моделювання статичної структури системи. Вона показує класи, атрибути класів, методи (операції), а також зв'язки між класами, такі як асоціації, агрегації, композиції та успадкування. Класи представляють основні компоненти системи, їхні характеристики (атрибути) та поведінку (методи).

Зв'язки показують, як ці класи взаємодіють між собою.

База даних та її структура

База даних - це організований набір даних, які зберігаються в структурованому вигляді, зазвичай у вигляді таблиць. Таблиці в базі даних складаються з рядків (записів) і стовпців (полів), що містять атрибути даних. Структура бази даних визначає, як дані взаємопов'язані між собою. Основними елементами є таблиці, ключі (первинні та зовнішні) і зв'язки між таблицями (один-до-одного, одиндобагатьох, багато-до-багатьох).

Шаблон Репозиторію (Repository Pattern)

Шаблон Репозиторію використовується для абстрагування доступу до даних. Він дозволяє взаємодіяти з базою даних через клас-репозиторій, що інкапсулює всі операції збереження, отримання, оновлення та видалення даних. Це забезпечує слабку залежність між бізнес-логікою та логікою доступу до даних, роблячи систему більш гнучкою для змін або модернізацій.

Вибір прецедентів та аналіз

Прецеденти дозволяють ідентифікувати сценарії використання, що найбільш підходять для вашої системи. Важливо вибрати і проаналізувати кілька подібних систем, щоб побачити, як їх функціональність відповідає вимогам вашого проекту. Це допомагає розробити найбільш ефективні та зручні рішення для користувачів.

Хід роботи

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Проаналізувати тему та намалювати схему прецеденту, що відповідає обраній темі лабораторії.
3. Намалювати діаграму класів для реалізованої частини системи.
4. Вибрати 3 прецеденти і написати на їх основі прецеденти.
5. Розробити основні класи і структуру системи баз даних.
6. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.
7. Підготувати звіт про хід виконання лабораторних робіт. Звіт, що подається повинен містити: діаграму прецедентів, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Схема прецеденту, що відповідає обраній темі, зображена на рисунку 1.

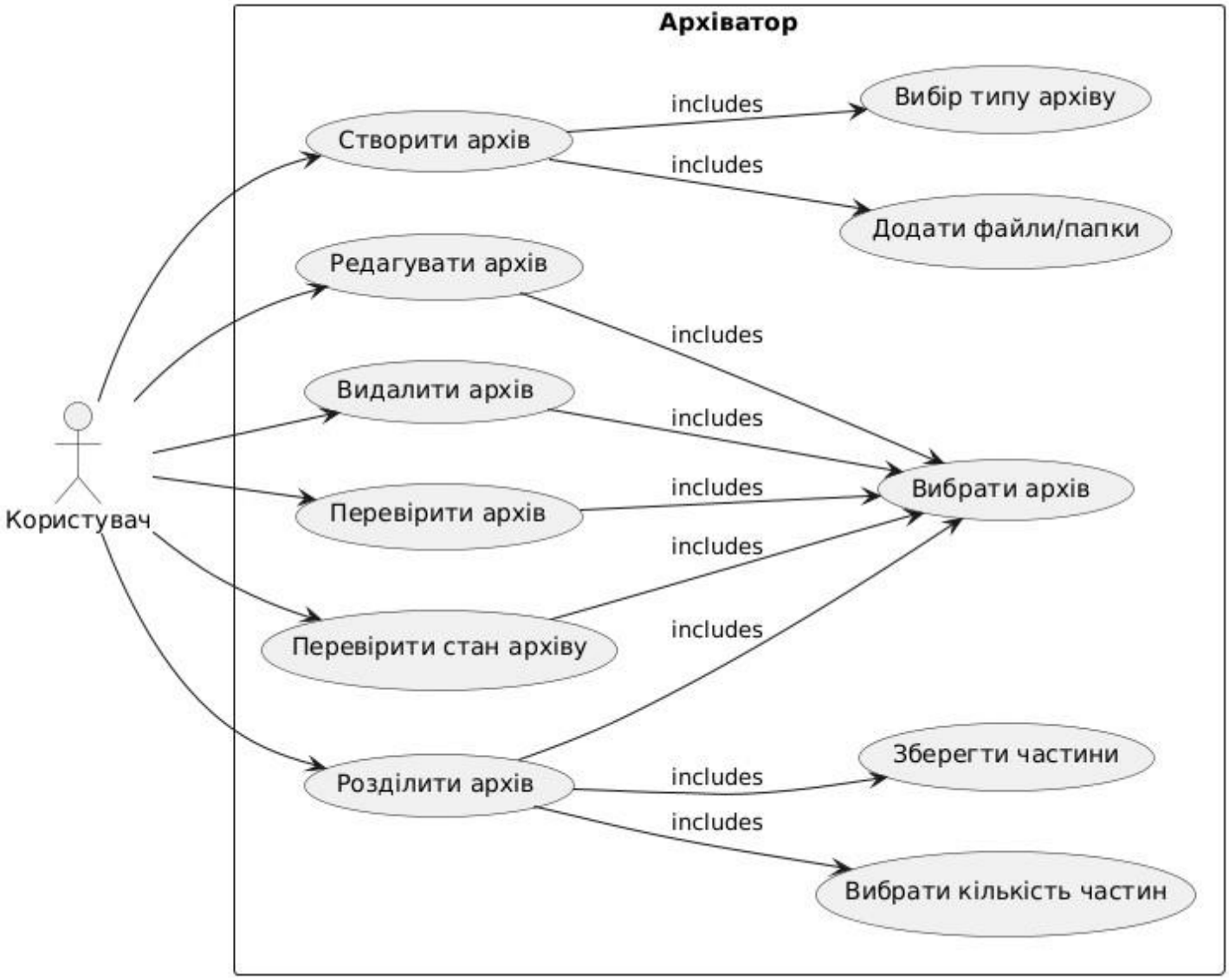

Рисунок 1. - Схема прецеденту

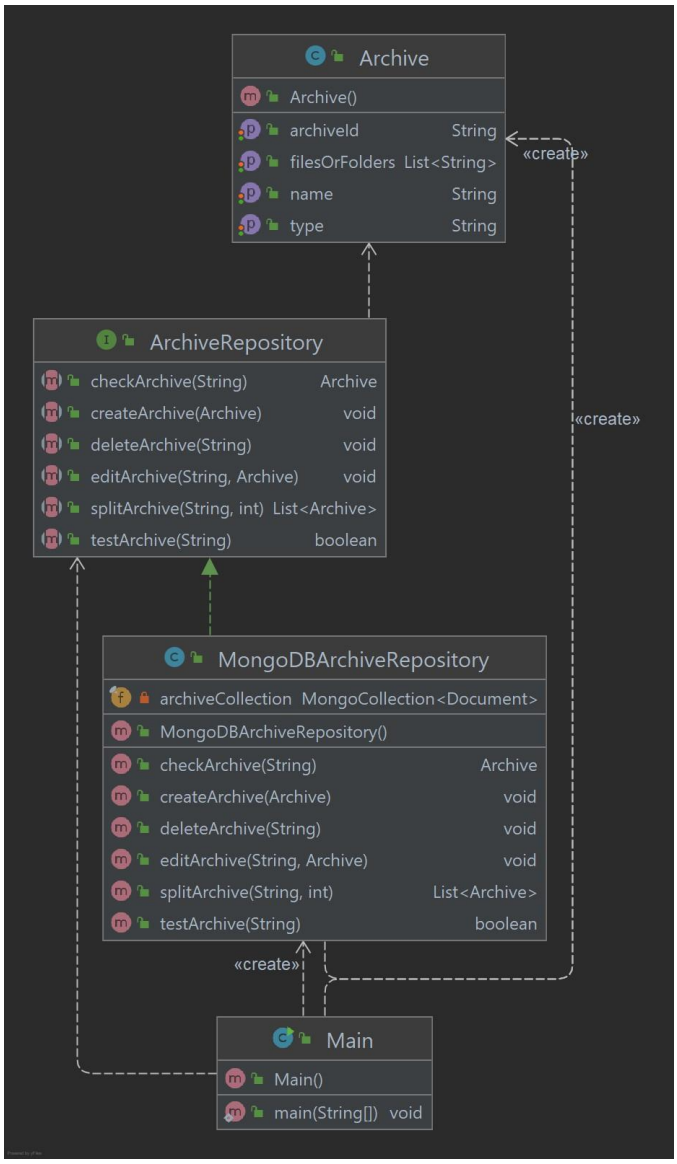

Рисунок 2. - Діаграма Класів

archives

	_id ObjectId	name String	type String	archiveId String	filesOrFolders Array
1	ObjectId('651ddf4251cfa157d22…	"asdas"	"safasf"	"2"	[] 2 elements
2	ObjectId('651de0284be1fc6a668…	"asdas_part1"	"safasf"	"4"	[] 1 elements
3	ObjectId('651de0284be1fc6a668…	"asdas_part2"	"safasf"	"5"	[] 1 elements

Рисунок 3. - Структура системи баз даних

Прецедент 1: Створення нового архіву

Передумови. Відсутні.

Постумови. Створено новий архів з вибраними файлами або папками.

Актори: Користувач

Опис. Користувач хоче створити новий архів з вибраними файлами або папками та зберегти його.

Основний хід подій:

1. Користувач вибирає опцію "Створити архів" у програмі.
2. Система відображає вибір типу архіву (наприклад, .zip, .rar, тощо).
3. Користувач обирає тип архіву та вказує ім'я та розташування для нового архіву.
4. Користувач обирає файли або папки, які він хоче додати до архіву.
5. Користувач підтверджує свій вибір.
6. Система створює новий архів та додає вибрані файли або папки до нього.
7. Система повідомляє користувача про успішне створення архіву.

Альтернатиивний потік подій:

Користувач не обрав тип архіву: Система автоматично вибирає тип архіву за замовчуванням (наприклад, .zip) і продовжує створення архіву.

Користувач обрав файли, які вже містяться в архіві: Система попереджає про дублікати і пропонує користувачу перезаписати існуючі файли або скасувати вибір дубльованих файлів.

Користувач скасував процес створення архіву: Система припиняє створення архіву і видаляє будь-які тимчасові дані. Винятки та примітки. Відсутні.

Винятки та примітки. Відсутні.

Прецедент 2. Редагування архіву

Передумови. Користувач ввійшов у систему та має створений архів.

Постумови. Архів було успішно відредаговано, і зміни збережено.

Актори: Користувач

Опис: Користувач хоче внести зміни до існуючого архіву, включаючи редагування метаданих.

Основний хід подій:

1. Користувач вибирає опцію "Редагувати архів" у програмі.
2. Система відображає список наявних архівів, і користувач обирає потрібний архів для редагування.
3. Користувач може вибрати опцію редагування метаданих архіву (наприклад, назва, опис).
4. Користувач може додавати або видаляти файли або папки в архіві.
5. Після внесення всіх необхідних змін користувач підтверджує свій вибір.
6. Система зберігає зміни у вибраному архіві.
7. Система повідомляє користувача про успішне редагування архіву.

Альтернативний потік подій:

Обраний архів пошкоджений або недоступний: Система повідомляє користувача про проблему і пропонує спробувати відновити архів або обрати інший.

Користувач додає файли, які перевищують максимальний розмір архіву: Система попереджає про обмеження і пропонує зменшити кількість файлів або використовувати інший тип архіву. Користувач відмовився зберігати зміни: Система скасовує редагування і зберігає архів у початковому стані.

Винятки та примітки. Відсутні

Прецедент 3. Перевірка цілісності архіву.

Передумови: Користувач ввійшов у систему та має створений архів.

Постумови: Статус цілісності архіву було перевірено, і користувач отримав відповідну інформацію.

Актори: Користувач.

Опис: Користувач хоче перевірити цілісність (checksum) існуючого архіву.

Основний хід подій:

1. Користувач вибирає опцію "Перевірити архів" у програмі.
2. Система відображає список наявних архівів, і користувач обирає архів для перевірки цілісності.
3. Система обчислює checksum архіву та порівнює його зі збереженим значенням.
4. Система повідомляє користувача про результат перевірки цілісності (чи співпадають checksum).

Альтернативний потік подій:

Контрольна сума архіву не співпадає з очікуваною: Система попереджає про можливе пошкодження архіву і пропонує спробувати відновлення або створити резервну копію.

Перевірка перервана через збій або скасування користувача: Система припиняє перевірку і пропонує повторити операцію.

Архів не містить контрольної суми: Система повідомляє, що перевірка неможлива через відсутність даних, і пропонує створити новий архів з контрольними даними для перевірки в майбутньому.

Винятки. Якщо користувач відмовиться від перевірки архіву на будь-якому етапі, процес завершується без перевірки.

Висновок: У цій лабораторній роботі я створив UML-діаграми для опису системи, включаючи діаграму прецедентів і діаграму класів. Реалізував ключові сценарії роботи системи, такі як створення, перевірка, редагування та розділення архівів. Спроєктував базу даних і класи, використовуючи шаблон Репозиторію для взаємодії з даними. Підготував звіт із діаграмами, кодом і структурою бази даних. Робота поглибила мої знання в проєктуванні та реалізації програмних систем.