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

Технологія розроблення програмного забезпечення

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

Варіант 25

Виконав:

Студент групи ІА-23 Перевірив

Калина С.О. Мягкий М.Ю.

Київ 2024

Тема: ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.

2. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторії.

3. Намалюйте діаграму класів для реалізованої частини системи.

4. Виберіть 3 прецеденти і напишіть 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.

Варіант:

..25 Installer generator (iterator, builder, factory method, bridge, interpreter, client-server)

Генератор інсталяційних пакетів повинен мати якийсь спосіб налаштування файлів, що входять в установку, установки вікон з інтерактивними можливостями (галочка - створити ярлик на робочому столі; ввести в текстове поле деякі дані, наприклад, ліцензійний ключ і т.д.). Генератор повинен вивести один файл .exe або .msi.

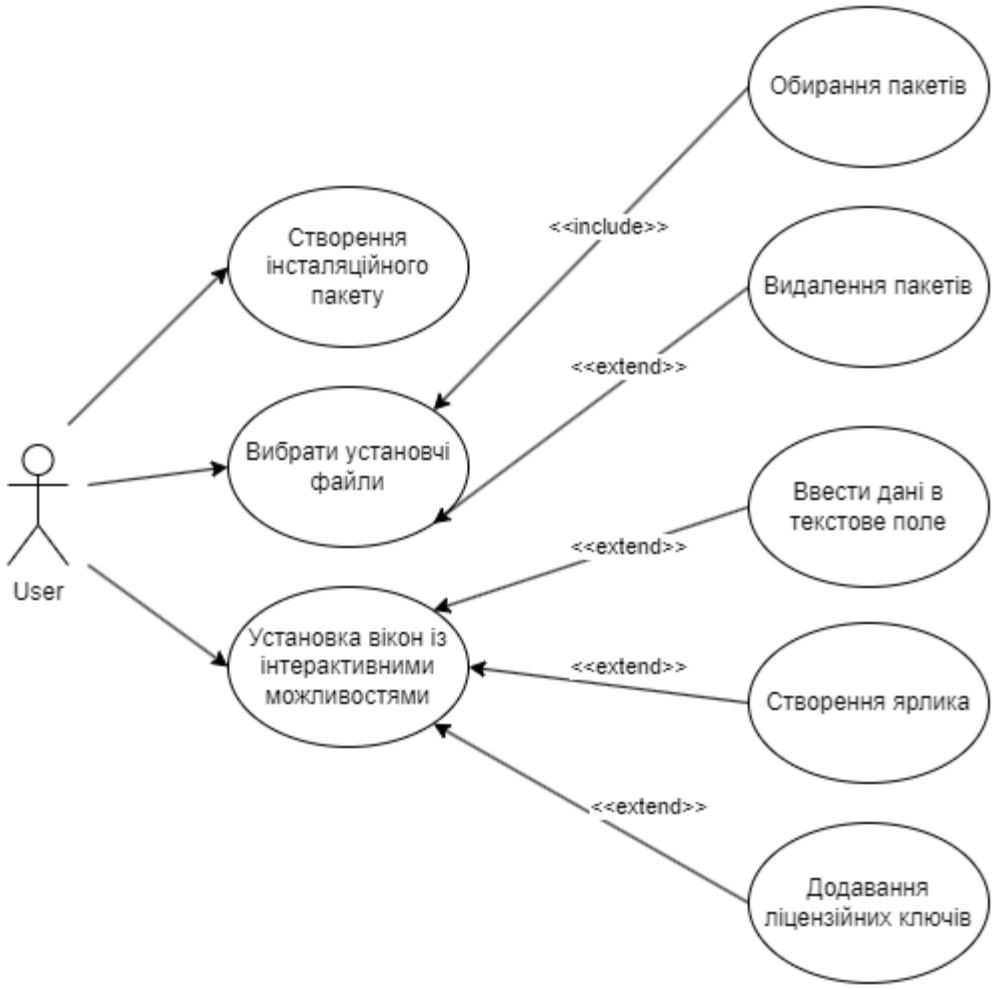
Теоретичні відомості

- **Діаграма варіантів використання (Use Case Diagram)** – це тип діаграми UML, що описує функціональність системи з точки зору її користувачів (акторів) і взаємодії між ними та системою. Вона показує, які дії (варіанти використання) можуть виконуватися користувачами, але не вдається у внутрішні механізми їх реалізації.
- **Сценарії варіантів використання (Use Case Scenarios)** – це текстовий опис варіантів використання, де детально викладається, як система повинна реагувати на дії користувачів у кожній конкретній ситуації. Включає в себе основний потік подій та альтернативні шляхи розвитку сценарію.
- **Діаграма класів (Class Diagram)** – це структура, яка моделює класи системи, їх властивості, методи, а також зв'язки між ними. Класи представляють основні об'єкти системи, які мають атрибути та операції, а також відображають взаємодію між різними компонентами.
- **Концептуальна модель системи** – це абстрактне представлення об'єктів та зв'язків між ними, що відображає ключові аспекти системи з точки зору бізнесу або предметної області. Вона описує основні компоненти, їх взаємодію та структуру, але не деталізує технічну реалізацію.

Ці діаграми дозволяють аналізувати вимоги до системи та планувати її розробку.

Хід роботи

Діаграма прецедентів:



Користувач запускає генератор інсталяції та може:

- створити інсталяційний пакет (діють інструкції по інсталяції)
- вибрати установчі файли (додає/видаляє файли, які мають бути включені до інсталятора)
- установка вікон із інтерактивними можливостями (додає можливість створення ярлика, введення ліцензійного ключа, тощо)

Предеценти на основі трьох прецедентів:

Прецедент 1: Створення інсталяційного пакету

- **Передумови:** Користувач має доступ до необхідних файлів і конфігурацій для створення інсталяційного пакету.
- **Постумови:** Система створює інсталяційний пакет на основі вибраних файлів. У випадку помилок система повідомляє про проблему. Якщо неможливо створити інсталяційний пакет, стан системи не змінюється.
- **Сторони взаємодії:** Авторизований користувач, система управління пакетами.
- **Короткий опис:** Цей варіант використання описує процес створення інсталяційного пакету.

Основний потік подій:

1. Користувач вибирає опцію створення інсталяційного пакету.
2. Система запитує вибір необхідних файлів та параметрів конфігурації.
3. Користувач вводить відповідні дані.
4. Система приймає введені дані та починає створення інсталяційного пакету.
5. Після завершення створення інсталяційного пакету система повідомляє користувача про успішність операції або про виникнення помилок (Виняток №1).

Винятки:

- **Виняток №1:** Неправильний вибір файлів або некоректні параметри конфігурації. Якщо система не може створити пакет через помилки в даних, вона повідомляє про це користувача і пропонує виправити введені дані або скасувати процес.

Прецедент 2: Вибір установчих файлів

- **Передумови:** Користувач має доступ до файлів, необхідних для створення інсталяційного пакету.
- **Постумови:** Файли успішно додані до списку установчих елементів. У разі помилки система повідомляє користувача.
- **Сторони взаємодії:** Авторизований користувач, система управління файлами.
- **Короткий опис:** Цей варіант використання описує процес вибору установчих файлів для подальшого створення інсталяційного пакету.

Основний потік подій:

1. Користувач вибирає опцію вибору установчих файлів.
2. Система надає доступ до файлової системи для вибору потрібних файлів.
3. Користувач вибирає файли для інсталяції.
4. Система приймає вибрані файли і додає їх до списку для подальшого створення інсталяційного пакету.
5. У випадку помилки система повідомляє користувача (Виняток №1).

Винятки:

- **Виняток №1:** Некоректний вибір файлів або відсутність файлів у системі. Система повідомляє про це користувача і дозволяє повторно вибрати файли або скасувати процес.

Прецедент 3: Установка вікон із інтерактивними можливостями

- **Передумови:** Користувач створює інсталяційний пакет з інтерактивними можливостями для вікон встановлення.
- **Постумови:** Система успішно налаштовує вікна встановлення з інтерактивними елементами. Якщо помилка, система повідомляє про проблему.
- **Сторони взаємодії:** Авторизований користувач, система для створення інсталяційних пакетів.
- **Короткий опис:** Цей варіант використання описує процес налаштування вікон встановлення з інтерактивними елементами.

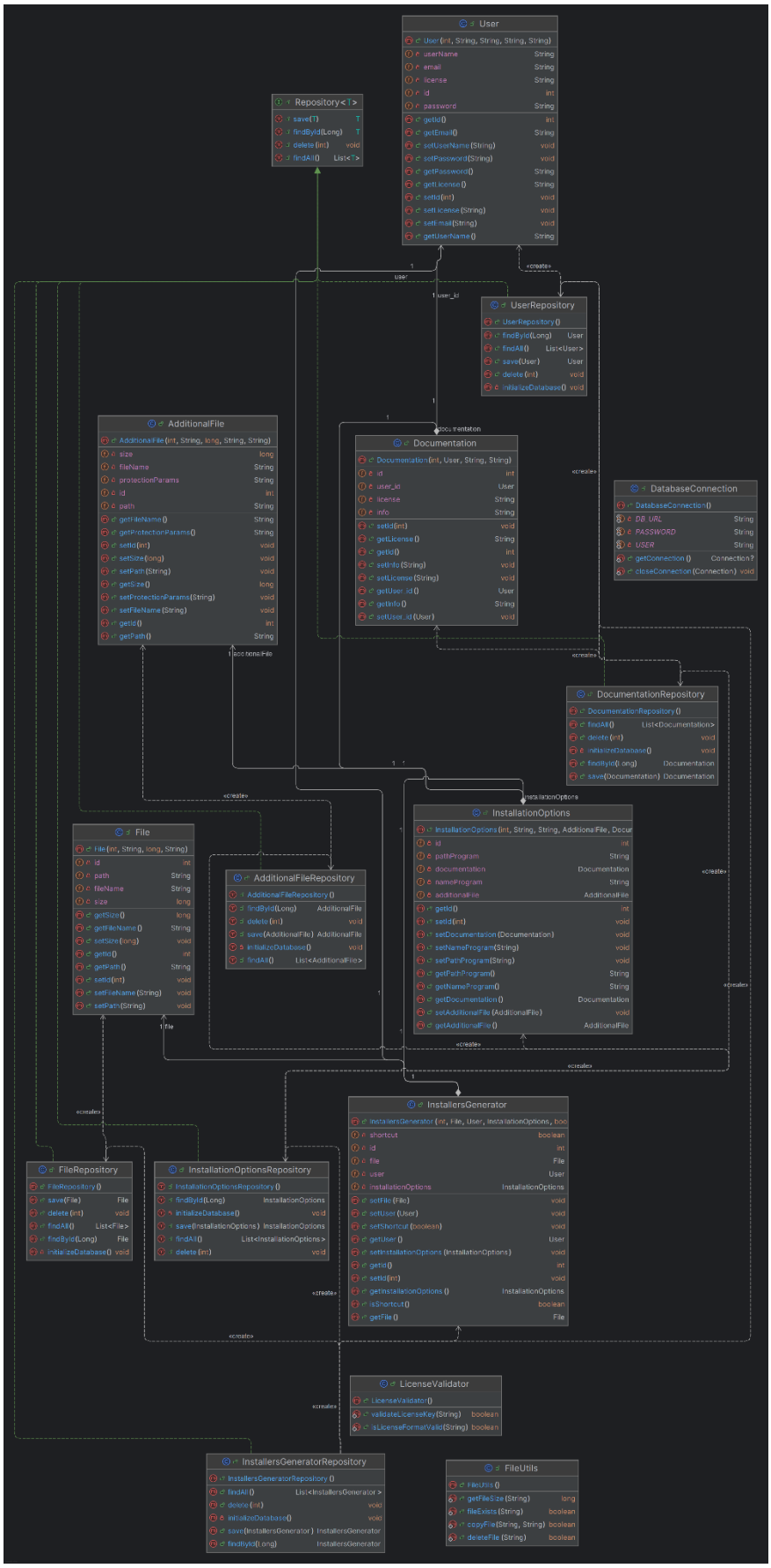
Основний потік подій:

1. Користувач вибирає опцію налаштування вікон із інтерактивними можливостями.
2. Система запитує параметри для вікон встановлення (текстові поля, кнопки, ярлики).
3. Користувач вводить необхідні дані і вибирає елементи.
4. Система приймає введені дані і налаштовує вікна встановлення.
5. Після налаштування система повідомляє про успіх або помилки (Виняток №1).

Винятки:

- **Виняток №1:** Некоректні або неповні дані для налаштування вікон. Система повідомляє про це користувача і пропонує виправити помилки або скасувати процес.

Діаграма класів:



Основні компоненти діаграми:

1. **Repository Pattern**:
   - **Repository<T>** — це інтерфейс, який оголошує базові методи для роботи з базою даних: save(T t), findById(Long id), delete(int id), і findAll(). Цей інтерфейс забезпечує базові операції CRUD (Create, Read, Update, Delete) для всіх моделей.
   - Конкретні репозиторії, такі як **UserRepository**, **DocumentationRepository**, **FileRepository**, реалізують цей інтерфейс, додаючи додаткову логіку для кожної моделі.
2. **Моделі**:
   - **User** — представляє користувача системи з такими полями, як userName, email, license, password. Цей клас містить базові геттери і сеттери для кожного з полів, а також методи для роботи з користувачем.
   - **AdditionalFile** — модель, яка зберігає інформацію про додаткові файли, що включаються в інсталяційні пакети. Поля включають fileName, size, protectionParams, і path.
   - **Documentation** — модель, яка зберігає документацію з ліцензійними ключами. Вона пов'язана з користувачами через поле user.
   - **File** — модель для опису файлів, які додаються до інсталяційних пакетів. Вона містить такі поля, як fileName, size, і path.
   - **InstallationOptions** — зберігає опції інсталяції, такі як шлях до програми, ім'я програми, пов'язані документація та додаткові файли.
   - **InstallersGenerator** — це модель, що зберігає всю інформацію про інсталяційні пакети, які створюються користувачами. Поля включають: file, user, installationOptions, і shortcut, що вказує, чи потрібно створювати ярлик.
3. **Зв'язки між класами**:
   - **User**, **File**, **Documentation**, **AdditionalFile** та інші моделі мають асоціації з класом **InstallersGenerator**, що показує, що ці сутності пов'язані з процесом генерації інсталяційного пакета.
   - **User** асоціюється з **Documentation**, що відображає зв'язок між користувачем і його документацією (ліцензією).
   - **InstallersGenerator** залежить від **InstallationOptions**, яка в свою чергу залежить від **Documentation** і **AdditionalFile**. Це відображає залежність інсталяційного пакету від налаштувань інсталяції, ліцензії та додаткових файлів.
   - **File** та **AdditionalFile** пов'язані з **InstallersGenerator**, що вказує на необхідність цих моделей для формування інсталяційного пакета.
4. **Репозиторії**:
   - **UserRepository** — реалізує методи для роботи з користувачами: findById(Long id), save(User user), delete(int id) і метод для ініціалізації бази даних initializeDatabase().

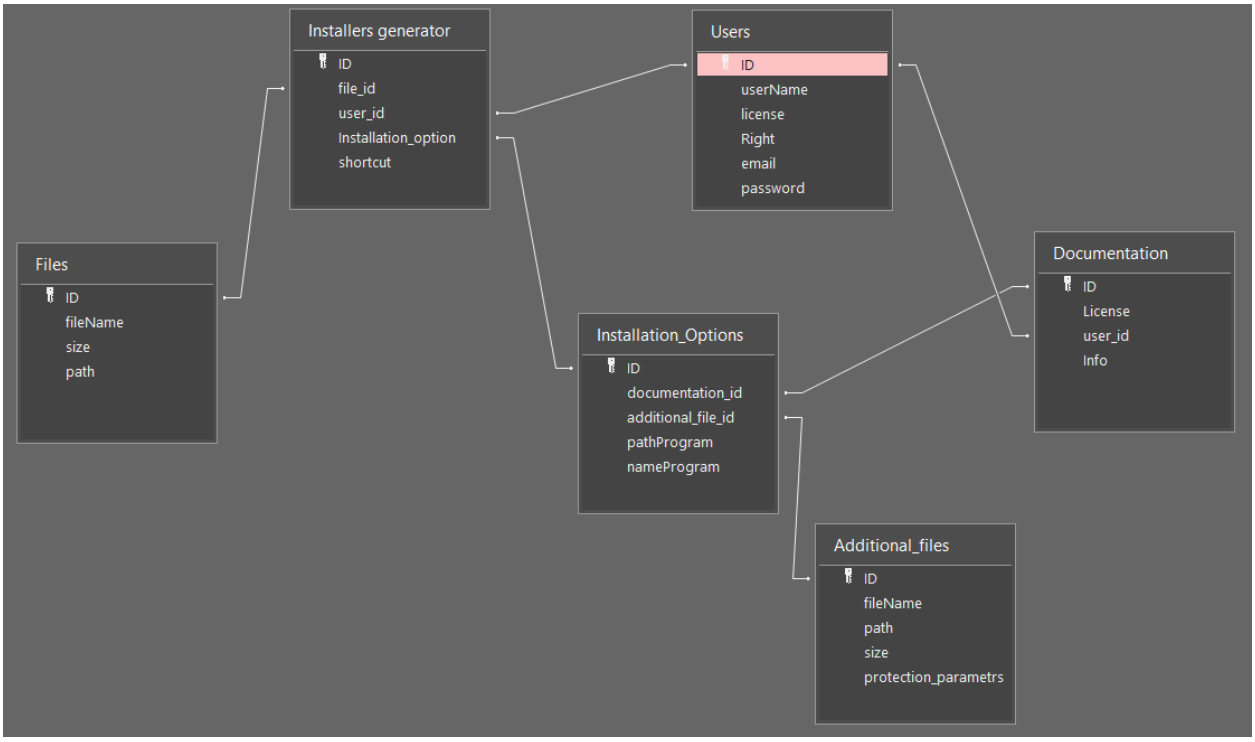
- **DocumentationRepository** — реалізує методи для роботи з документацією: findById(Long id), save(Documentation documentation), і метод для ініціалізації бази даних.
- **FileRepository**, **AdditionalFileRepository**, **InstallersGeneratorRepository**, **InstallationOptionsRepository** мають подібну структуру для роботи зі своїми відповідними моделями. Вони забезпечують збереження, пошук, оновлення та видалення даних у базі.

5. **Utility Classes**:
   - **LicenseValidator** — клас для перевірки ліцензійних ключів. Містить методи validateLicenseKey(String key) для перевірки валідності ліцензії і isLicenseFormatValid(String key), що перевіряє формат ліцензійного ключа.
   - **FileUtils** — допоміжний клас для роботи з файлами, включаючи методи для перевірки існування файлу, отримання розміру, копіювання та видалення файлів.
6. **База даних та з'єднання**:
   - **DatabaseConnection** — цей клас відповідає за встановлення та закриття з'єднання з базою даних. Містить методи getConnection(), який повертає активне з'єднання з базою, та closeConnection(Connection connection), який закриває з'єднання.

Загальна структура:

- **Моделі** пов'язані одна з одною через об'єкти (асоціації), що відображає реальні відносини між об'єктами у програмі.
- **Репозиторії** забезпечують CRUD-операції для кожної з моделей, зберігаючи при цьому розділення логіки доступу до даних і бізнес-логіки.
- **Утиліти**, такі як **LicenseValidator** і **FileUtils**, допомагають у виконанні спеціальних операцій, таких як валідація даних або маніпуляції з файлами.

Структура бази даних:


Installers generator
ID
file_id
user_id
Installation_option
shortcut
Users
ID
userName
license
Right
email
password
Files
ID
fileName
size
path
Documentation
ID
License
user_id
Info
Installation_Options
ID
documentation_id
additional_file_id
pathProgram
nameProgram
Additional_files
ID
fileName
path
size
protection_parametrs

Висновок: отже, у ході виконання лабораторної роботи було проведено ознайомлення з теоретичними відомостями та розроблено прецеденти та діаграми класів для системи керування завданнями. Окрім того, підготовлений звіт включає всі необхідні компоненти, що відображають структуру розробленої системи.