

**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  
Тема: 25. Installer generator**

Виконав:

Студент групи ІА-22

Птачик Р.С.

Перевірив:

Мягкий М.Ю.

**Київ 2024**

## Лабораторна робота №2

Діаграма варіантів використання. Сценарії варіантів використання.  
Діаграми uml. Діаграми класів. Концептуальна модель системи

Хід роботи:

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

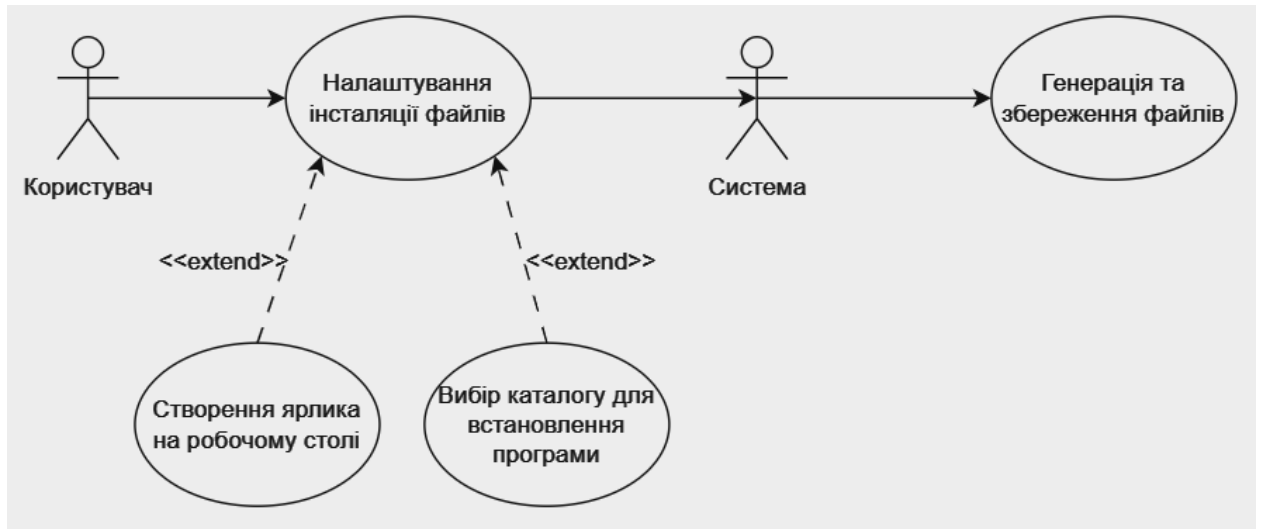


Рис. 1 – Діаграма варіантів використання (прецедентів)

2. Намалюйте діаграму класів для реалізованої частини системи.

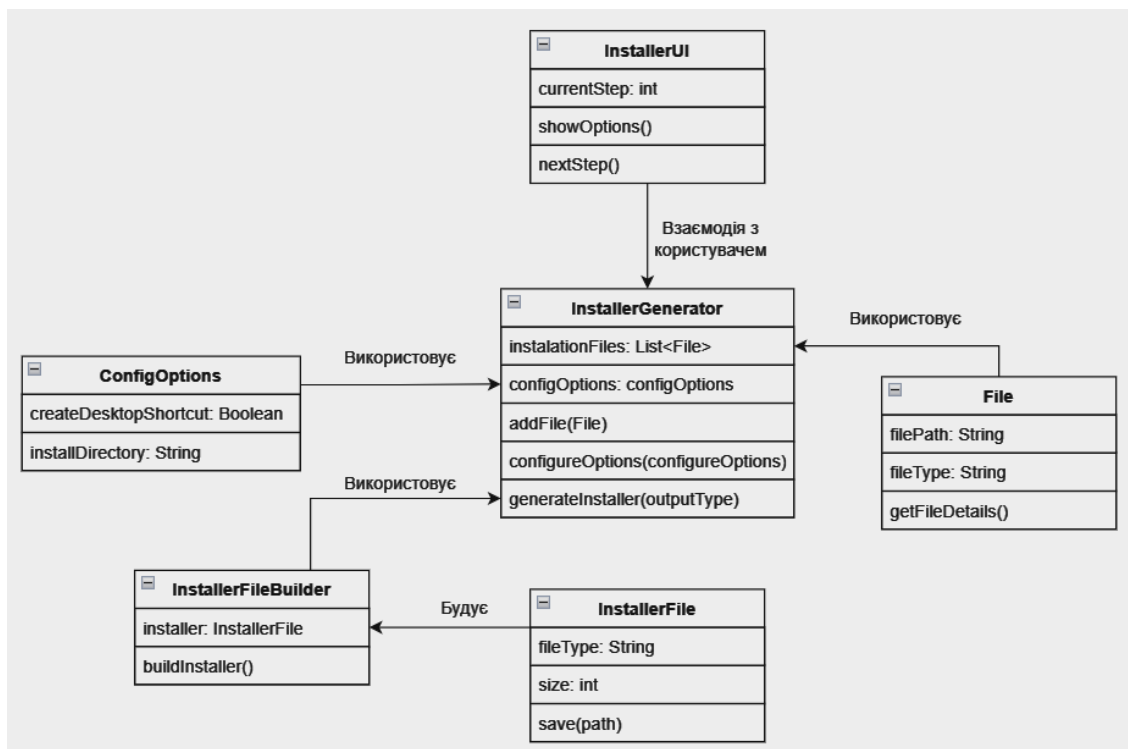


Рис. 2 – Діаграма класів

### **3. Виберіть 3 прецеденти і напишіть на їх основі прецеденти.**

#### **1) Налаштування інсталяційних файлів:**

**Короткий опис:** Користувач налаштовує параметри для інсталяції файлів.

**Основний потік подій:**

##### **Створення ярлика на робочому столі:**

1. Користувач відкриває вікно налаштування інсталяції.
2. Система пропонує користувачу вибрати опцію створення ярлика.
3. Користувач вибирає, чи потрібно створити ярлик.
4. Якщо користувач вибирає опцію створення ярлика, система зберігає цей параметр.

##### **Альтернативні потоки:**

- Користувач може пропустити цей крок, і ярлик не буде створено.

##### **Вибір каталогу для встановлення програми:**

1. Система пропонує користувачу стандартну директорію для встановлення.
2. Користувач може прийняти стандартну директорію або вибрати іншу.
3. Система зберігає вибрану директорію для встановлення програми.

##### **Альтернативні потоки:**

- Якщо користувач не вибирає директорію, система використовує стандартний шлях за замовчуванням.

#### **2) Збереження інсталяційного файлу**

**Короткий опис:** дані налаштування передаються в систему після чого інсталяційний файл зберігається.

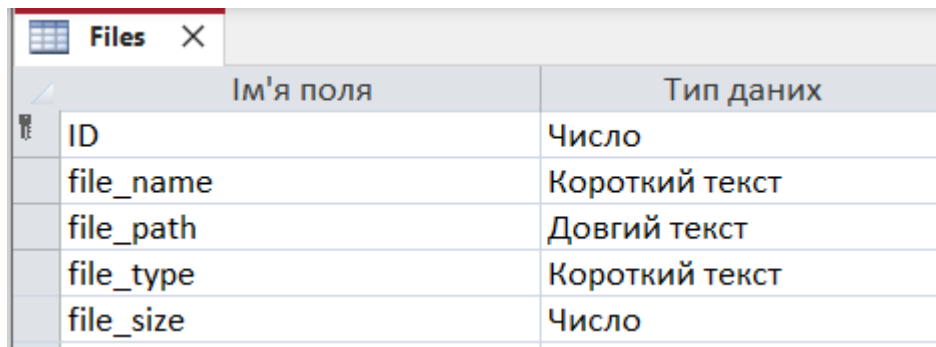
**Основний потік подій:**

##### **Збереження інсталяційного файлу:**

1. Дані користувача про налаштування встановлення файлу передаються в систему.
2. До вибраних налаштувань застосовуються відповідні засоби (створення ярлика та/або зміна директорії для встановлення)
3. Інсталяційний файл зберігається на пристрої користувача.

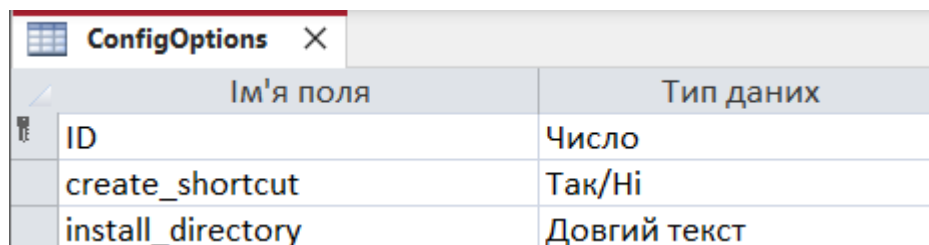
#### 4. Структура системи баз даних.

База даних включає 3 таблиці:



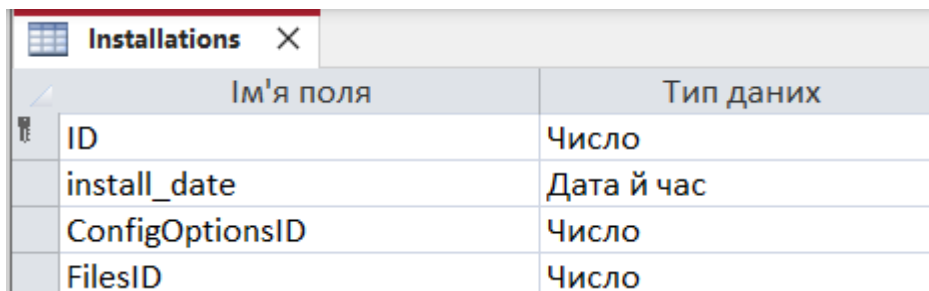
Ім'я поля	Тип даних
ID	Число
file_name	Короткий текст
file_path	Довгий текст
file_type	Короткий текст
file_size	Число

Рис. 3 – Таблиця Files



Ім'я поля	Тип даних
ID	Число
create_shortcut	Так/Ні
install_directory	Довгий текст

Рис. 4 – Таблиця ConfigOptions



Ім'я поля	Тип даних
ID	Число
install_date	Дата й час
ConfigOptionsID	Число
FilesID	Число

Рис. 5 – Таблиця Installations

Нижче наведено структуру бази даних та зв'язки в ній:

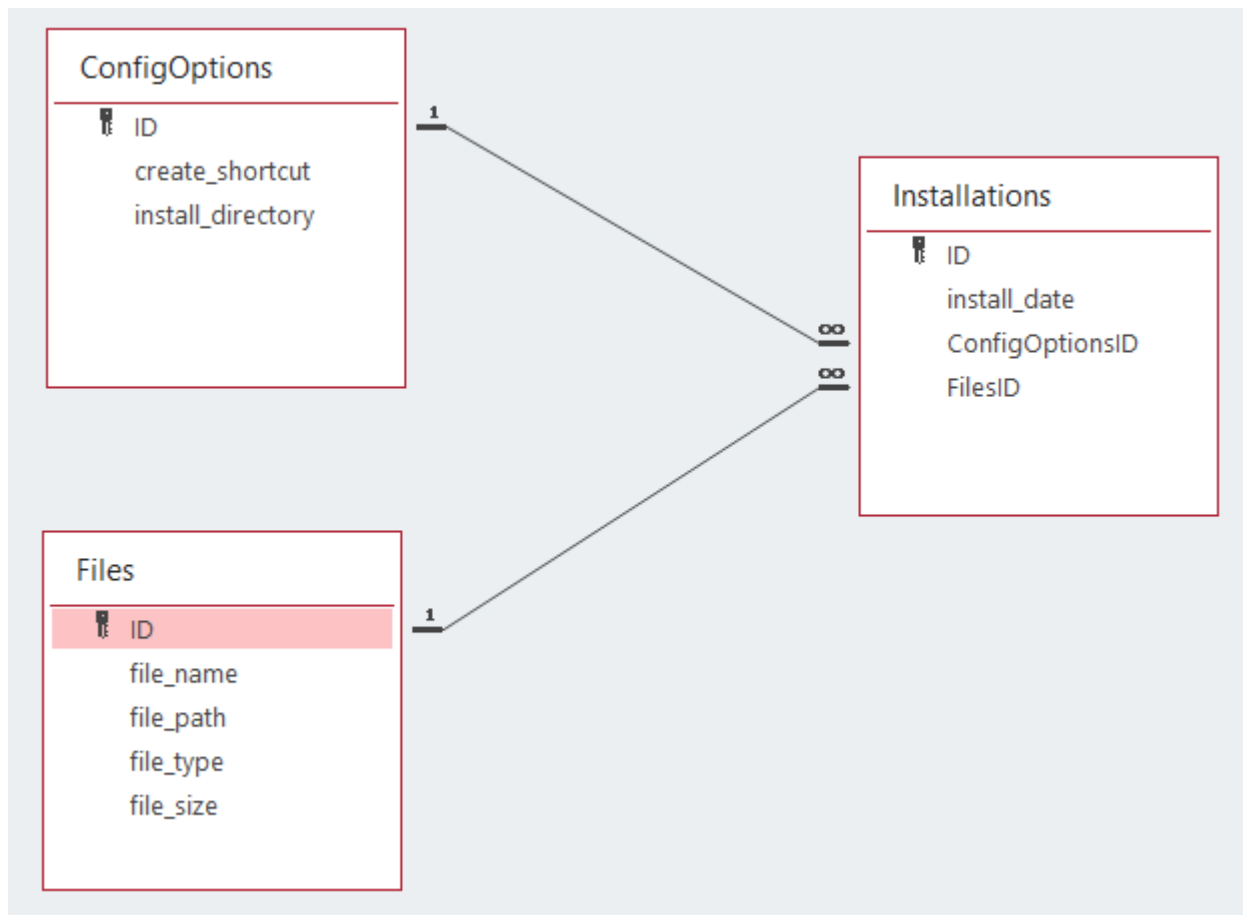


Рис. 6 – Структура БД

**5. Створення основних класів. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.**

**Код:**

**Основні класи які реалізують шаблон репозиторію:**

<https://github.com/FryMondo/TRPZ/blob/master/InstallGenerator/repository.py>

**Тестування зв'язку з БД:**

<https://github.com/FryMondo/TRPZ/blob/master/InstallGenerator/test.py>

```

from database import get_connection

2 usages
class ConfigOptionsRepository:
    def __init__(self):
        self.conn = get_connection()
        self.cursor = self.conn.cursor()

    1 usage
    def add(self, create_shortcut, install_directory):
        sql = "INSERT INTO ConfigOptions (create_shortcut, install_directory) VALUES (%s, %s)"
        values = (create_shortcut, install_directory)
        self.cursor.execute(sql, values)
        self.conn.commit()
        return self.cursor.lastrowid

    1 usage
    def get(self, config_id):
        sql = "SELECT * FROM ConfigOptions WHERE config_id = %s"
        self.cursor.execute(sql, (config_id,))
        return self.cursor.fetchone()

2 usages
class FilesRepository:
    def __init__(self):
        self.conn = get_connection()
        self.cursor = self.conn.cursor()

    1 usage
    def add(self, file_name, file_path, file_type, file_size):
        sql = "INSERT INTO Files (file_name, file_path, file_type, file_size) VALUES (%s, %s, %s, %s)"
        values = (file_name, file_path, file_type, file_size)
        self.cursor.execute(sql, values)
        self.conn.commit()
        return self.cursor.lastrowid

```

```

    def get(self, file_id):
        sql = "SELECT * FROM Files WHERE file_id = %s"
        self.cursor.execute(sql, (file_id,))
        return self.cursor.fetchone()

2 usages
class InstallationsRepository:
    def __init__(self):
        self.conn = get_connection()
        self.cursor = self.conn.cursor()

    1 usage
    def add(self, install_date, config_id, file_id):
        sql = "INSERT INTO Installations (install_date, config_id, file_id) VALUES (%s, %s, %s)"
        values = (install_date, config_id, file_id)
        self.cursor.execute(sql, values)
        self.conn.commit()
        return self.cursor.lastrowid

    1 usage
    def get(self, installation_id):
        sql = "SELECT * FROM Installations WHERE installation_id = %s"
        self.cursor.execute(sql, (installation_id,))
        return self.cursor.fetchone()

```

**Висновок:** на цій лабораторній роботі я створив діаграму варіантів використання (прецедентів), діаграму класів, описав сценарій використання, створив структуру бази даних та створив основні класи які реалізують шаблон Репозиторію та протестував з'єднання з БД.