

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни

**«**Бази даних і засоби управління**»**

Виконав студент ІII курсу

ФПМ групи КВ-83

Кубай О.Ф.

Перевірив: Петрашенко А. В.

Київ – 2020

**Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL**

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

*Загальне* *завдання* роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з 2-х та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

*Деталізоване завдання*:

1. Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти:
2. контроль при введенні - валідація даних;
3. перехоплення помилок (**try...except**) від сервера PostgreSQL при виконанні відповідної команди SQL.

Особливу увагу варто звернути на дані таблиць, що мають зв’язок 1:N.

З боку батьківської таблиці необхідно контролювати **вилучення** (**ON DELETE**) рядків за умови наявності даних у підлеглій таблиці.

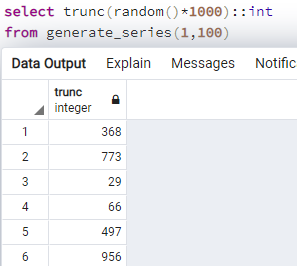
З боку підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні **внесення** до неї нових даних.

Унеможливити виведення програмою на екрані системних помилок PostgreSQL шляхом їх перехоплення і адекватної обробки.

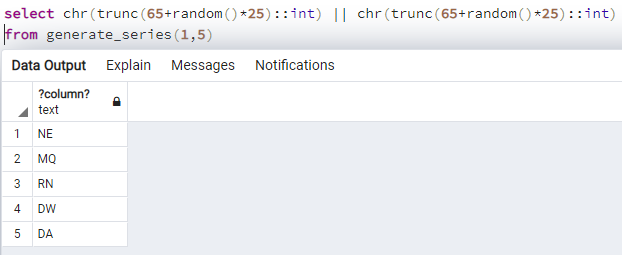
Внесення даних виконується користувачем у консольному вікні програми.

1. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не програмою, а відповідним SQL-запитом**!

Приклад генерації 100 псевдовипадкових чисел:



Приклад генерації 5-ти псевдовипадкових рядків:



Приклад генерації псевдовипадкової мітки часу з діапазону [доступний за посиланням](https://stackoverflow.com/questions/22964272/postgresql-get-a-random-datetime-timestamp-between-two-datetime-timestamp/22965061).

Кількість даних для генерування має вводити користувач з клавіатури.

Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності (foreign key).

1. Для реалізації багатокритеріального пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Після виведення даних вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.
2. Програмний код організувати згідно шаблону Model-View-Controller (MVC). Приклад організації коду згідно шаблону доступний [за даним посиланням](https://www.giacomodebidda.com/mvc-pattern-in-python-introduction-and-basicmodel/). Модель, подання (представлення) та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

Рекомендована бібліотека взаємодії з PostgreSQL Psycopg2: http://initd.org/psycopg/docs/usage.html)

*Вимоги до інтерфейсу користувача*

Використовувати консольний інтерфейс користувача.

*Вимоги до інструментарію*

Середовище для лагодження SQL-запитів до бази даних – PgAdmin4.

Мова програмування – Python 3.6-3.7

Середовище розробки програмного забезпечення – PyCharm Community Edition 2020.

*Вимоги до оформлення звіту лабораторної роботи у електронному вигляді*

Опис (файл README.md) лабораторної роботи у **репозиторії** **GitHub** включає: назву лабораторної роботи, структуру бази даних з лабораторної роботи №1.

Репозиторій має містить файл звіту у форматі PDF та програмний код файлів мовою Python (або іншою).

Звіт у форматі PDF має містити: титульний аркуш, завдання та відповіді на вимоги до звітування щодо пунктів 1-4 деталізованого завдання:

*Вимоги до пункту №1 деталізованого завдання:*

* ілюстрації обробки виняткових ситуацій (помилок) при уведенні/вилучення даних;
* ілюстрації валідації даних при уведенні користувачем.

*Вимоги до пункту №2 деталізованого завдання:*

* копії екрану (ілюстрації) з фрагментами згенерованих даних таблиць.

*Вимоги до пункту №3 деталізованого завдання:*

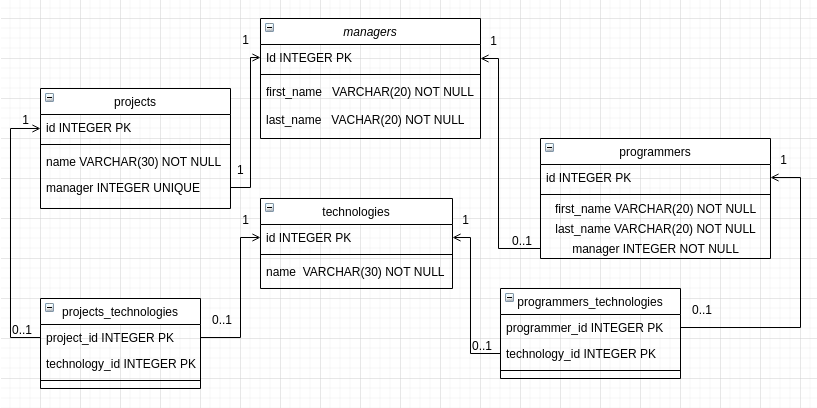
* ілюстрації уведення пошукового запиту та результатів виконання запитів.

*Вимоги до пункту №4 деталізованого завдання:*

* ілюстрації програмного коду з репозиторію Git.

# Нормалізована модель даних

Програма використовує базу даних, розроблену на попередній лабораторній роботі. На рисунку зображена нормалізована модель бази даних.



Опис програми

Програма створена для управління базою даних за допомогою базових операцій СУБД PostgreSQL та реалізовує функціональні вимоги, що наведені у завданні. Програма складається з двох проектів (кожен з яких окремий логічний рівень):

1. DAL – DataAccessLayer (рівень доступу до даних). В ньому знаходяться моделі, репозиторії та інтерфейси репозиторіїв.
2. UI – UserInterface (рівень користувацького інтерфейсу). В ньому знаходяться меню та контролери, які зв’язують меню та репозиторії.

Програма складається з таких пакунків і класів:

1) com.naofi.model.entity.\* - класи відображають дані в таблицях.

2) com.naofi.model.dao.\* - об’єкти доступу до даних (інтерфейси, їх реалізації за допомогою spring jdbc).

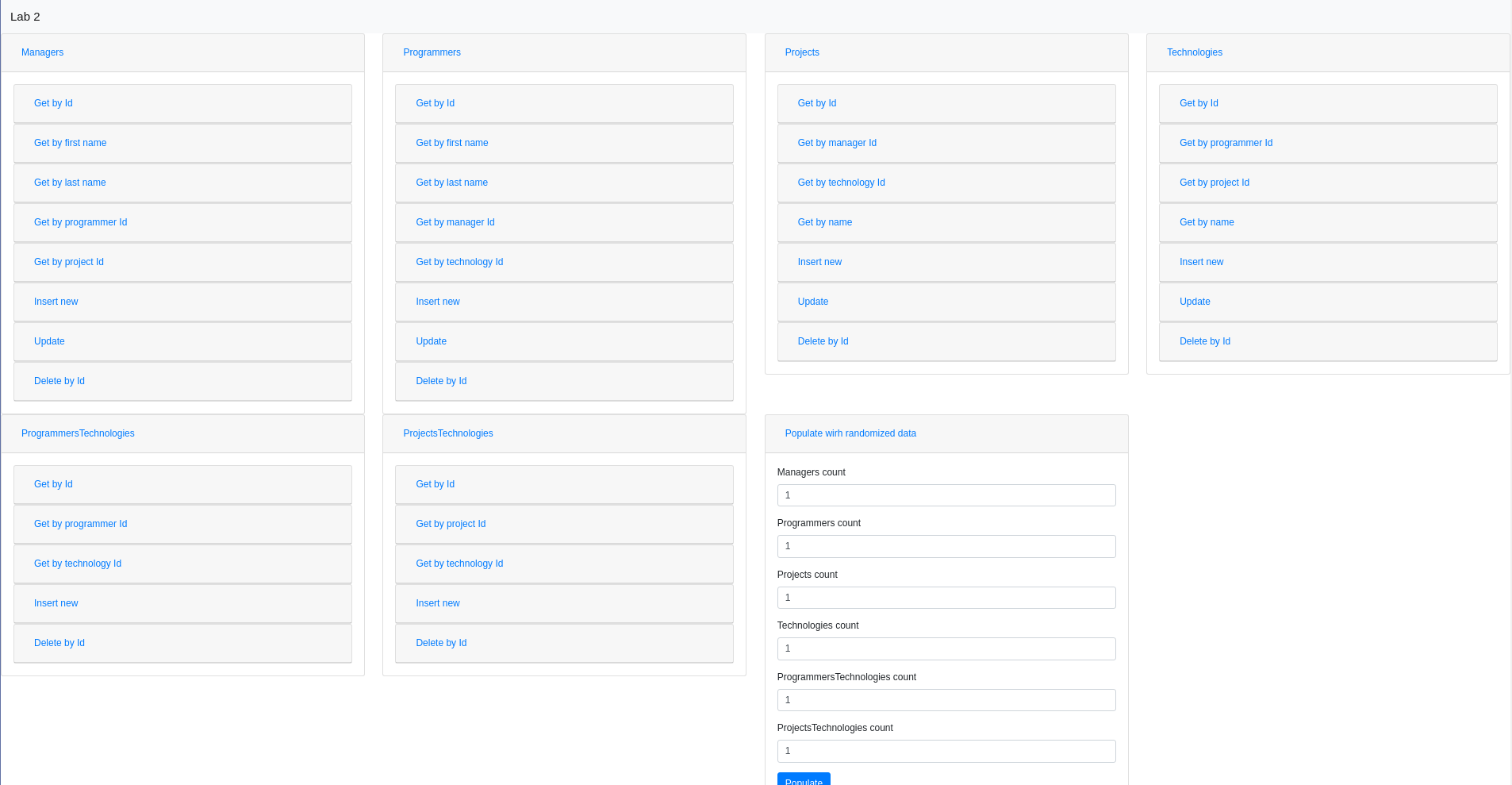
3) com.naofi.controllers.\* - контроллери (rest, або для доступу до jsp)

4) com.naofi.util.IdGenerator — генератор id (для jsp)

5) webapp/\*\* - view

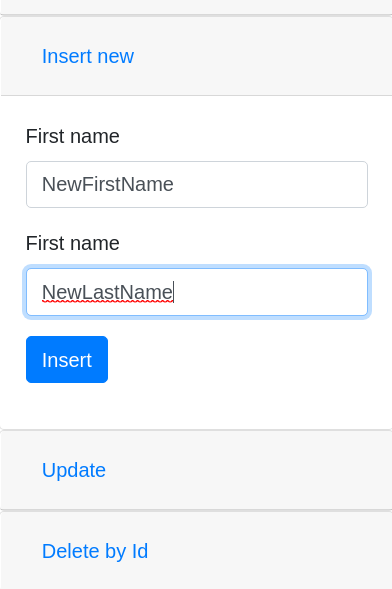
# Результати роботи програми

Запуск програми (необхідний application server, для тестування був використаний glassfish 4):



# Завдання 1

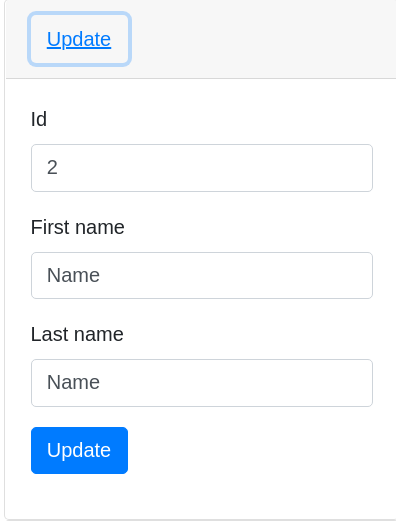
# Додавання даних до БД:



Результат:



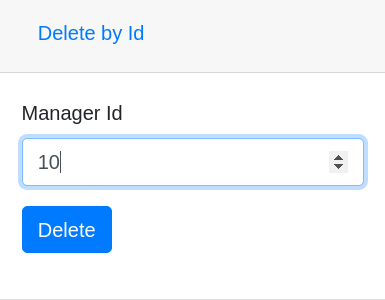
## Редагування даних(id обов’язкове поле):



Результат:



## Видалення даних:



## 

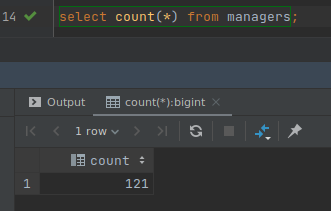
**Валідація відбувається за допомогою відловлення exceptions у файлі ExceptionsProcessor.java. Відбувається звичайне виведенн повідомлень у браузер. Наприклад:**

## 

# Завдання 2. Пакетне генерування даних:

Генератор даних включений у кожен dao обʼєкт (метод populate, що приймає на вхід кількість записів, що необхідно згенерувати)

Перед генеруванням даних:



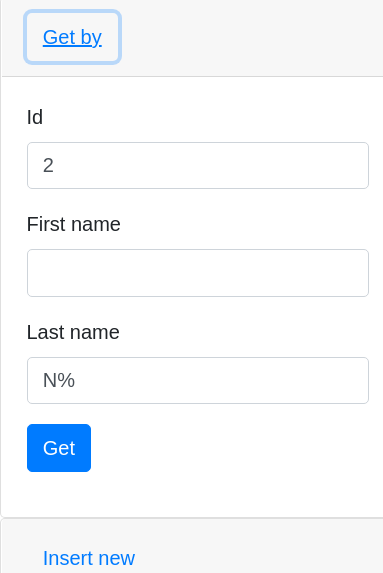
# Генеруємо 1000 записів у таблиці managers:

# 

# Після генерування:

# 

# **Завдання 3. Пошук за двома-трьома атрибутами одночасно**



# Результат:

# 

# Посилання да github:

# <https://github.com/kubayof/databases>