

Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут»

**Розрахунково-графічна робота**

**з дисципліни**

**«Бази даних та засоби управління»**

**Назва: “Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”**

Виконав студент групи: КВ-12

ПІБ: Сілін І.Д.

Перевірив: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Київ 2023**

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

*Загальне* *завдання* роботи полягає у наступному:

1. Реалізувати функції перегляду, 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 двох 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).

*Варіант:* **Система управління персоналом компанії.**

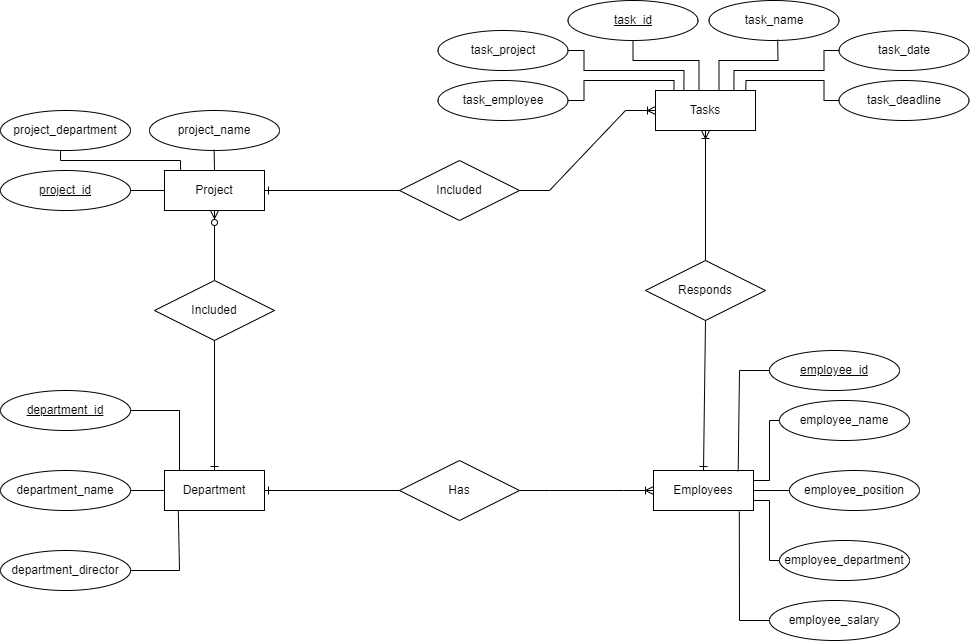
*Репозиторій:* [*https://github.com/canco32/CPMS*](https://github.com/canco32/CPMS)

*Контакт у Телеграм:* [*@IicancoI*](https://t.me/IicancoI)

*Мова розробки: PHP 8.0.28*

*Бібліотеки: Bootstrap, Faker, jQuery.*

Розробка моделі «сутність-зв’язок» предметної галузі бази даних «Система управління персоналом компанії».



*Рис. 1. ER-діаграма побудована за нотацією* «Crow’s foot».

**Сутності з описом та призначенням**

Модель предметної галузі «сутність-зв’язок» бази даних «Система управління персоналом компанії» включає у себе 4 сутності, які містять в собі від 3 до 6 атрибутів:

1. Department (department\_id, department\_name, department\_director).
2. Employees (employee\_id, employee\_name, employee\_position, employee\_department, employee\_salary).
3. Tasks (task\_id, task\_name, task\_project, task\_employee, task\_date, task\_deadline).
4. Project (project\_id, project\_name, project\_department).

Сутність «Department» описує відділ, до якого належать співробітники компанії. «Department» містить атрибути, які описують унікальний номер відділу, назву відділу, а також його керівника.

Сутність «Employees» описує співробітників компанії. «Employees» містить атрибути, які описують унікальний номер співробітника, ім’я співробітника, його посаду, номер відділу, а також його заробітну плату.

Сутність «Tasks» описує завдання, що отримали співробітники компанії. «Tasks» містить атрибути, які описують унікальний номер завдання, унікальний номер співробітника, унікальний номер проекту, до якого належить завдання, назву задачі, а також початок та дедлайн завдання.

Сутність «Project» описує проекти, що існують у компанії. «Project» містить атрибути, які описують унікальний номер завдання, унікальний номер проекту, назву проекту, а також відділ, який займається цим проектом.

**Зв’язки між сутностями**

*Сутність «Department» та «Employees»:*

* Кожен співробітник компанії може належати лише до одного відділу. Тим часом, як відділ може мати 1-го та більше співробітників. Тому зв’язок між сутностями «Department» та «Employees» є **1:N**.

*Сутність «Employees» та* «Tasks»*:*

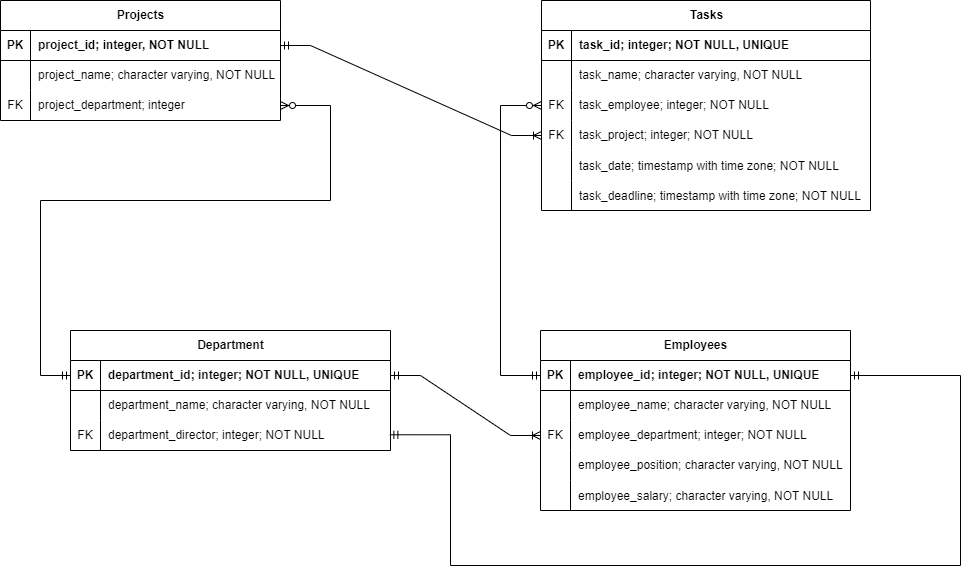
* Зазвичай у компаніях на кожного співробітника виділяється щонайменше одне, а частіше декілька завдань. Тому кожен співробітник може належати до декількох завдань, але кожне завдання належить лише одному відповідному співробітнику. Тому зв’язок між сутностями «Employees» та «Tasks» є **1:N**.

*Сутність «Tasks» та «Project»:*

* Кожне завдання може належати одному проекту. Кількість завдань, що можуть бути у проекті необмежена. Звідси, кожному завданню належить лише один проект, а кожному проекту безліч завдань. Тому зв’язок між сутностями «Tasks» та «Project»є **1:N**.

*Сутність «Project» та «Department»:*

* Кожному проекту компанії належить лише один підпорядкований відділ, тим часом як відділу може належати багато проектів або жодного. Тому сутність між «Project» та «Department»є **1:N**.



*Рис. 2. Логічна модель бази даних.*

**Опис проекту**

Проект побудований на серверні мові програмування PHP та включає у себе реалізацію побудови бази даних для задачі керування компанією, а саме керуванням персоналом та їх задачами та проектами.

Для запуску проекту знадобиться віртуальний локальний сервер PostgreSQL, а також MAMP або XAMPP. Також необхідне підключення до інтернету, адже у програмі використовуються підключення до CDN серверів для надання необхідних бібліотек.

У проекті використовується бібліотека jQuery для підключення стилів за допомогою Bootstrap. Цей пакет бібліотек необхідний для придання проекту приємного зовнішнього вигляду. Також використовується PHP бібліотека Faker з українською локалізацією. Призначення бібліотеки полягає у створенні випадкових даних для заповнення правдоподібним вмістом таблиці бази даних.

Програма включає у себе 4 основні файли з розширенням \*.php, а саме: «index.php», «model.php», «view.php, «controller.php». А також директорію «vendor» та файли «composer.json» і «composer.lock», які призначені для використання у проекті бібліотеки генерації випадкових даних Faker.

**Детальніше про програмні модулі управління програмою:**

*Програмний модуль Model:* У цьому модулі ми генеруємо підключення до бази даних, а також виконуємо деякі дії над таблицями бази, а саме: каскадне очищення, перевірка зовнішнього ключа, а також видалення таблиці.

*Програмний модуль View:* У цьому модулі виконуються основні дії над таблицею, адже всі дії пов’язані з вводом та виводом таблиці. Маємо функцію, що виводить список усіх таблиць, функції перегляду даних усієї таблиці або певної кількості елементів. Також є функція генерації та обробки форм для заповнення, змінення або видалення таблиць.

*Програмний модуль Controller:* Цей модуль організовує зв’язок між класами інших модулів, а також створює та призначає дії для органів керування програмою.

**Опис функціоналу програми. Зв’язок з базою даних**

За допомогою класу PostgreSQLConnection, що розміщено у модуль «Model» можемо створити з`єднання з базою PostgreSQL. У подальшому це з’єднання буде використовуватися у всій програмі.

Конструктор класу, що відповідає за установку з’єднання:

**public** **function** \_\_construct() {

try {

$dsn = "pgsql:host=**{$this->host}**;port=**{$this->port}**;dbname=**{$this->database}**";

$this->connection = **new** PDO($dsn, $this->username, $this->password);

$this->connection->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch (PDOException $e) {

echo "Помилка підключення: " . $e->getMessage();

}

}

**Опис функціоналу програми. Функція перегляду даних**

На головному екрані під час запуску програми з’являється список усіх наявних таблиць та кількість відповідних записів у таблицях. При кліку на назву таблиці буде відкриватися перегляд вмісту даних таблиці, ось як це працює:

Функція, що запускає перегляд списку таблиць та їх вмісту належить до модуля View, а саме до класу Viewer та має назву displayTables.

**public** **function** displayTables($table\_name = **NULL**) {

$result = "";

if ($table\_name == **NULL**) {

$result = "

<h2>Таблиці</h2><br>

<table class=**\"**table**\"**>

<thead>

<tr>

<th>Назва таблиці</th>

<th>Кількість записів</th>

</tr>

</thead>

<tbody>

";

try {

$pdo = $this->dbConnection->getConnection();

$tables = $this->getAllTables($pdo);

foreach ($tables as $table) {

$result .= "<tr>";

$result .= "<td><a href=**\"**index.php?view\_table=**{$table['table\_name']}\"**>**{$table['table\_name']}**</a></td>";

$result .= "<td>**{$table['row\_count']}**</td>";

$result .= "</tr>";

}

$result .= " </tbody>

</table>

";

} catch (PDOException $e) {

echo "<br>Помилка виконання запиту: " . $e->getMessage();

}

} else {

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>**$table\_sanitized**</h2><br>";

try {

$pdo = $this->dbConnection->getConnection();

$columns\_data = $this->getColumnsNames($pdo, $table\_sanitized);

$table\_data = $this->getNecessaryTable($pdo, $table\_sanitized);

$first\_column\_saved = **false**;

$data\_id = "";

$result .= "

<table class=**\"**table**\"**>

<thead>

<tr>

";

foreach ($columns\_data as $column) {

if (!$first\_column\_saved) {

$data\_id = $column['column\_name'];

}

$result .= "<th>**{$column['column\_name']}**</th>";

$first\_column\_saved = **true**;

}

$result .= "<th style=**\"**text-align: center;**\"** colspan=**\"**2**\"**>Дії</th>";

$result .= " </tr>

</thead>

<tbody>

";

foreach ($table\_data as $row) {

$result .= "<tr>";

foreach ($row as $cell) {

$result .= "<td>**{$cell}**</td>";

}

$result .= "<td><a href=**\"**index.php?edit\_table=**$table\_sanitized**&edit=**$row**[**$data\_id**]**\"**>Змінити</a></td>";

$result .= "<td><a href=**\"**index.php?edit\_table=**$table\_sanitized**&delete=**$row**[**$data\_id**]**\"**>Видалити</a></td>";

$result .= "</tr>";

}

$result .= " </tbody>

</table>

";

} catch (PDOException $e) {

echo "<br>Помилка виконання запиту: " . $e->getMessage();

}

}

return $result;

}

Якщо не існує GET-параметра, то запускається перегляд усіх таблиць та кількості записів у них. Якщо ж існує GET-параметр, то він має містити назву таблиці, на яку відбувся клік. В такому ж випадку відкривається вміст відповідної таблиці.

**Опис функціоналу програми. Управління даними. Вилучення, Додавання та Редагування даних**

Для кожного елементу таблиці наявні кнопки управління - *“Редагувати”* та *“Видалити”*. Користувач може змінювати зміст таблиці цими кнопками з урахуванням зовнішнього ключа.

Також маємо 5 кнопок над таблицею:

* *“Усі таблиці”* - повернення на головний екран.
* *“Додати запис”* - додати новий запис до таблиці.
* *“Видалити таблицю”* - видалення таблиці з урахуванням зовнішніх ключів.
* *“Очистити каскадно”* - каскадне очищення поточної таблиці разом з пов’язаними таблицями.
* *“Випадкові поля”* - додавання рандомізованих полів у кількості до 100.000 засобами SQL та бібліотеки Faker.

Усі дії над записами таблиці так само виконуються за допомогою відповідних GET-параметрів, а саме: $\_GET['edit\_table'] та $\_GET['delete'] для вилучення запису з таблиці; $\_GET['add\_note'] для додавання запису до таблиці; $\_GET['edit\_table'] та $\_GET['edit'] для додавання запису до таблиці.

Для додавання запису до таблиці використовується така конструкція виклику функції:

**<?php** if(isset($\_GET['add\_note'])): **?>**

**<?=** $viewer->addNote($\_GET['add\_note']); **?>**

**<?php** endif; **?>**

Функція, що викликається:

**public** **function** addNote($table\_name) {

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>Додання запису до таблиці **\"$table\_sanitized\"**</h2><br>";

try {

$pdo = $this->dbConnection->getConnection();

$columns\_data = $this->getColumnsNames($pdo, $table\_sanitized);

$result .= "<form method=**\"**post**\"**>";

$id = 0;

foreach($columns\_data as $input) {

if($id != 0) {

if (stripos($input['column\_name'], 'date') !== **false** || stripos($input['column\_name'], 'deadline') !== **false**) {

$result.= "

<div class=**\"**form-group**\"**>

<label for=**\"**input**$id\"**>Поле **\"{$input['column\_name']}\"**</label>

<input name=**\"{$input['column\_name']}\"** type=**\"**date**\"** class=**\"**form-control**\"** placeholder=**\"**Введіть значення для поля '**{$input['column\_name']}**'**\"**>

</div>

";

} else {

$result.= "

<div class=**\"**form-group**\"**>

<label for=**\"**input**$id\"**>Поле **\"{$input['column\_name']}\"**</label>

<input name=**\"{$input['column\_name']}\"** type=**\"**text**\"** class=**\"**form-control**\"** placeholder=**\"**Введіть значення для поля '**{$input['column\_name']}**'**\"**>

</div>

";

}

}

$id++;

}

$result.= "

<button type=**\"**submit**\"** class=**\"**btn btn-primary**\"**>Готово</button>

</form>

";

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$this->processForm($pdo, $table\_sanitized, $columns\_data, $\_POST);

}

} catch (PDOException $e) {

echo "<br>Помилка виконання запиту: " . $e->getMessage();

}

return $result;

}

Також ця функція звертається до функції getColumnsNames(PDO $pdo, $table\_name, $limit = NULL). Ця функція призначена для повернення назв колонок відповідної таблиці для дії вставки записів.

**private** **function** getColumnsNames(PDO $pdo, $table\_name, $limit = **NULL**) {

if($limit == **NULL**) {

$sql = "SELECT column\_name, data\_type FROM information\_schema.columns WHERE table\_name = '**$table\_name**' ORDER BY ordinal\_position";

} else {

$sql = "SELECT column\_name, data\_type FROM information\_schema.columns WHERE table\_name = '**$table\_name**' ORDER BY ordinal\_position LIMIT **$limit**";

}

$stmt = $pdo->prepare($sql);

$stmt->execute();

$columns\_names = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $columns\_names;

}

Після виконання цих функцій програма генерує форму з відповідними полями для того, щоб правильно додати та опрацювати вміст нового запису таблиці.

Для операції додавання кінцевою функцією є processForm(PDO $pdo, $table\_name, $columns\_data, $form\_data)

**private** **function** processForm(PDO $pdo, $table\_name, $columns\_data, $form\_data) {

try {

$valid\_data = [];

$current\_date = **new** DateTime();

$one\_week\_later = (clone $current\_date)->add(**new** DateInterval('P7D'))->format('Y-m-d H:i:sP');

foreach ($columns\_data as $column\_info) {

if ($column\_info === $columns\_data[0]) {

continue;

}

$column = $column\_info['column\_name'];

if (is\_scalar($column) && isset($form\_data[$column]) && !empty($form\_data[$column])) {

$valid\_data[$column] = trim(filter\_var($form\_data[$column], FILTER\_SANITIZE\_SPECIAL\_CHARS));

if (isset($form\_data[$column]) && !empty($form\_data[$column])) {

if (stripos($column, 'date') !== **false** || stripos($column, 'deadline') !== **false**) {

$date\_time = DateTime::createFromFormat('Y-m-d', $form\_data[$column]);

$valid\_data[$column] = $date\_time ? $date\_time->format('Y-m-d H:i:sP') : $one\_week\_later;

} else {

$valid\_data[$column] = trim(filter\_var($form\_data[$column], FILTER\_SANITIZE\_SPECIAL\_CHARS));

}

}

} else {

echo "Поле '**$column**' обов'язкове для заповнення.";

return;

}

}

$columns\_str = implode(', ', array\_keys($valid\_data));

$values\_str = implode(', ', array\_fill(0, count($valid\_data), '?'));

$sql = "INSERT INTO **\"$table\_name\"** (**$columns\_str**) VALUES (**$values\_str**)";

$stmt = $pdo->prepare($sql);

$i = 1;

foreach ($valid\_data as $value) {

$stmt->bindValue($i++, $value);

}

$stmt->execute();

echo "Дані успішно додані до таблиці **$table\_name**.";

} catch (PDOException $e) {

echo "Помилка при вставці даних: " . $e->getMessage();

}

}

Ця функція оброблює та перевіряє введені дані на безпечність та додає до вмісту таблиці, як новий запис.

Функція вилучення запису подібна до дій додавання запису, але відрізняється використанням іншої кінцевої функції, а саме використовується функція processDeleteNote(PDO $pdo, $table\_name, $note\_id):

**private** **function** processDeleteNote(PDO $pdo, $table\_name, $note\_id) {

try {

$first\_columnQuery = "

SELECT column\_name

FROM information\_schema.columns

WHERE table\_name = :table\_name

ORDER BY ordinal\_position

LIMIT 1

";

$stmt = $pdo->prepare($first\_columnQuery);

$stmt->bindParam(':table\_name', $table\_name, PDO::PARAM\_STR);

$stmt->execute();

$first\_column = $stmt->fetchColumn();

$delete\_query = "DELETE FROM **\"$table\_name\"** WHERE **$first\_column** = :note\_id";

$stmt = $pdo->prepare($delete\_query);

$stmt->bindParam(':note\_id', $note\_id, PDO::PARAM\_INT);

$stmt->execute();

echo "Запис успішно видалено з таблиці **$table\_name**.";

} catch (PDOException $e) {

$errorCode = $e->getCode();

$errorMessage = $e->getMessage();

if ($errorCode == '23503') {

echo "Помилка зовнішнього ключа: **$errorMessage**";

} else {

echo "Інша помилка: **$errorMessage**";

}

}

}

Ця функція отримує назву таблиці та унікальний номер первинного ключа запису, якій необхідно вилучити. Якщо немає зовнішнього ключа, то функція успішно видалить обраний запис.

Функція оновлення запису подібна до дій додавання запису, але відрізняється використанням іншої кінцевої функції, а саме використовується функція processUpdateNote(PDO $pdo, $table\_name, $columns\_data, $form\_data, $note\_id):

**private** **function** processUpdateNote(PDO $pdo, $table\_name, $columns\_data, $form\_data, $note\_id) {

try {

$update\_query = "UPDATE **\"$table\_name\"** SET ";

$primary\_key\_name = $columns\_data[0]['column\_name'];

foreach ($columns\_data as $input) {

$column\_name = $input['column\_name'];

if ($column\_name != $primary\_key\_name) {

$update\_query .= "**$column\_name** = :**$column\_name**, ";

}

}

$update\_query = rtrim($update\_query, ', ');

$update\_query .= " WHERE **$primary\_key\_name** = :**$primary\_key\_name**";

$stmt = $pdo->prepare($update\_query);

foreach ($columns\_data as $input) {

$column\_name = $input['column\_name'];

if ($column\_name != $primary\_key\_name) {

$input\_value = trim(filter\_var($form\_data[$column\_name], FILTER\_SANITIZE\_SPECIAL\_CHARS));

$stmt->bindValue(":**$column\_name**", $input\_value);

}

}

$stmt->bindValue(":**$primary\_key\_name**", $note\_id);

$stmt->execute();

echo "Дані успішно оновлено у таблиці **$table\_name**.";

} catch (PDOException $e) {

echo "Помилка оновлення даних: " . $e->getMessage();

}

}

Ця функція отримує назву таблиці та унікальний номер первинного ключа запису, якій необхідно змінити. Крім того, вона також отримує назви колонок таблиць та їх оновлений вміст.

**Опис функціоналу програми. Пакетне генерування даних**

Кожну таблицю можна пакетно заповнити випадковими даними, які будуть схожі на справжні. Для цього використовується функція:

**public** **function** dataTableFill($table\_name) {

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>Додання випадкових полів до таблиці **\"$table\_sanitized\"**</h2><br>";

$result .= "

<form method=**\"**post**\"**>

<div class=**\"**form-group**\"**>

<label for=**\"**input**\"**>Введіть кількість нових випадкових полей для таблиці</label>

<input name=**\"**fillTable**\"** type=**\"**number**\"** min=**\"**1**\"** max=**\"**100000**\"** class=**\"**form-control**\"** placeholder=**\"**Введіть ціле значення від 0 до 100000 **\"**>

</div>

<button type=**\"**submit**\"** class=**\"**btn btn-primary**\"**>Створити</button>

</form>

";

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$pdo = $this->dbConnection->getConnection();

$this->randomFillTable($pdo, $table\_sanitized, $\_POST);

}

return $result;

}

Зміст цієї функції полягає у тому, що користувач має ввести кількість випадкових полів для заповнення бажаної таблиці. Кількість може бути від 0 до 100 000, але для додавання вмісту від 25 000 може займати більше 120 секунд, що призведе до помилки. Для цього необхідно редагувати файл сервера php\_ini для збільшення часу очікування.

Наступні дві функції генерують випадкові дані для таблиці, а перша ще вставляє ці дані за допомогою SQL запиту:

**private** **function** randomFillTable($pdo, $table\_name, $form\_data) {

$faker = **new** Faker\Generator();

$faker->addProvider(**new** Faker\Provider\uk\_UA\Person($faker));

$faker->addProvider(**new** Faker\Provider\uk\_UA\Company($faker));

$limit = $form\_data['fillTable'];

$startTime = microtime(**true**);

$columns\_data = $this->getColumnsNames($pdo, $table\_name);

$primary\_key\_name = $columns\_data[0]['column\_name'];

$sql = "INSERT INTO **\"$table\_name\"** (";

foreach ($columns\_data as $column) {

$column\_name = $column['column\_name'];

if ($column\_name !== $primary\_key\_name) {

$sql .= "**$column\_name**, ";

}

}

$sql = rtrim($sql, ", ") . ") VALUES ";

for ($i = 0; $i < $limit; $i++) {

$sql .= "(";

foreach ($columns\_data as $column) {

$column\_name = $column['column\_name'];

if($column\_name !== $primary\_key\_name) {

switch($column\_name) {

case "employee\_department":

$sql .= "(SELECT department\_id FROM **\"**Department**\"** ORDER BY RANDOM() LIMIT 1), ";

break;

case "department\_name":

$fake\_department = $faker->company();

$sql .= "('**$fake\_department**'),";

break;

case "employee\_name":

$fake\_name = $faker->name();

if (strpos($fake\_name, "'") !== **false**) {

$fake\_name = str\_replace("'", "`", $fake\_name);

}

$sql .= "('**$fake\_name**'),";

break;

case "employee\_position":

$fake\_position = $faker->jobTitle();

$sql .= "('**$fake\_position**'),";

break;

case "task\_employee":

$sql .= "(SELECT employee\_id FROM **\"**Employees**\"** ORDER BY RANDOM() LIMIT 1), ";

break;

case "task\_project":

$sql .= "(SELECT project\_id FROM **\"**project**\"** ORDER BY RANDOM() LIMIT 1), ";

break;

case "project\_department":

$sql .= "(SELECT department\_id FROM **\"**Department**\"** ORDER BY RANDOM() LIMIT 1), ";

break;

case "task\_date":

$sql .= "

(SELECT

date\_trunc('day', CURRENT\_DATE - INTERVAL '1 month') +

random() \* (CURRENT\_DATE - date\_trunc('day', CURRENT\_DATE - INTERVAL '1 month')) AS random\_date

),

";

break;

case "task\_deadline":

$sql .= "

(SELECT

CURRENT\_DATE +

random() \* INTERVAL '6 months' AS random\_date

)

";

break;

case "employee\_salary":

$sql .= "('$' || floor(random() \* (10000 - 300 + 1) + 300)::int), ";

break;

default:

$sql .= $this->generateRandomSqlValue($pdo, $column) . ", ";

break;

}

}

}

$sql = rtrim($sql, ", ") . "), ";

}

$sql = rtrim($sql, ", ") . ";";

$stmt = $pdo->prepare($sql);

$stmt->execute();

$endTime = microtime(**true**);

$executionTime = ($endTime - $startTime) \* 1000;

echo "Запит виконано за " . number\_format($executionTime, 2, '.', ' ') . " мс";

}

**private** **function** generateRandomSqlValue($pdo, $column) {

switch ($column['data\_type']) {

case 'integer':

return 'floor(random() \* 100)';

case 'character varying':

return "(SELECT left(md5(random()::text), 10))";

default:

return 'NULL';

}

}

Перша функція randomFillTable($pdo, $table\_name, $form\_data) генерує випадкові дані для таблиці з використанням бібліотеки Faker з українською локалізацією.

Друга функція generateRandomSqlValue($pdo, $column) генерує випадкові числа або набори символів за допомогою мови SQL.

**Опис функціоналу програми. Каскадне вилучення**

Каскадне очищення таблиці передбачає у собі видалення вмісту таблиці з усіма залежними записами з інших таблиць. Ось наступні функції, які це роблять:

**public** **function** cascadeTruncate($table\_name) {

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>Каскадне очищення таблиці **\"$table\_sanitized\"**</h2><br>";

try {

$pdo = $this->dbConnection->getConnection();

$sql = $this->generateTruncateTableSQL($table\_sanitized);

$pdo->exec($sql);

$result .= "<p>Таблиця **\"$table\_sanitized\"** та усі залежні від неї таблиці каскадно очищені.</p>";

} catch (PDOException $e) {

$result .= "<br>Помилка каскадного видалення: " . $e->getMessage();

}

return $result;

}

**private** **function** generateTruncateTableSQL($table\_name) {

return "TRUNCATE **\"$table\_name\"** RESTART IDENTITY CASCADE";

}

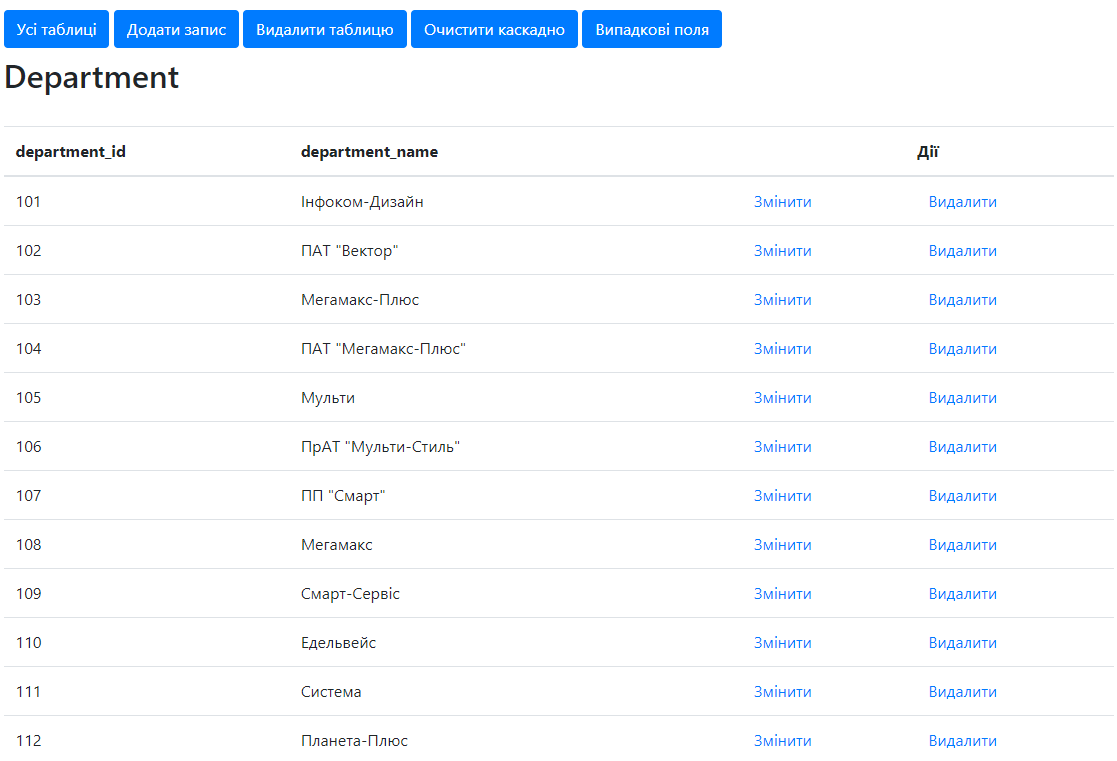
**Ілюстрації роботи програми**

****

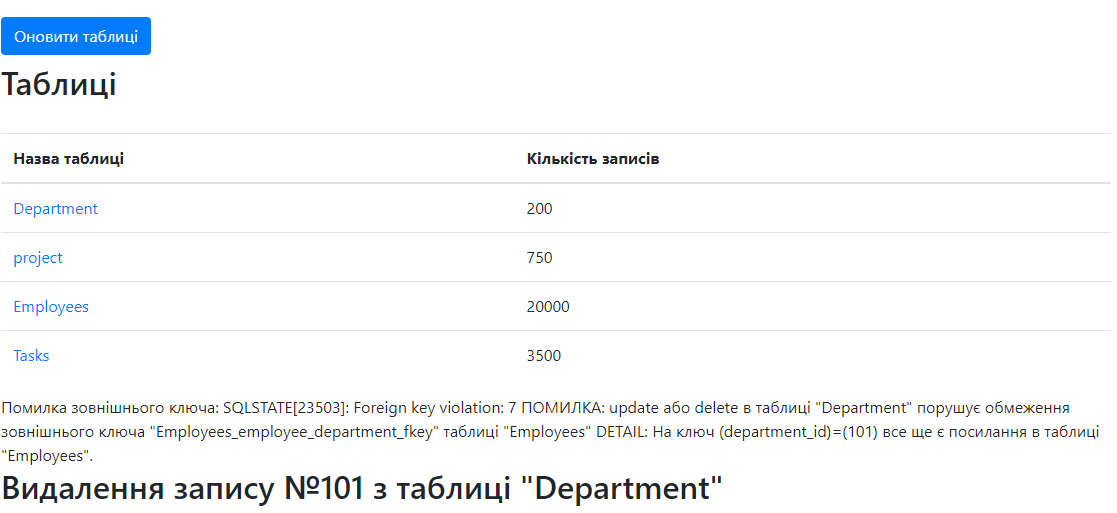
*Рис. 3. Головний екран програми*.



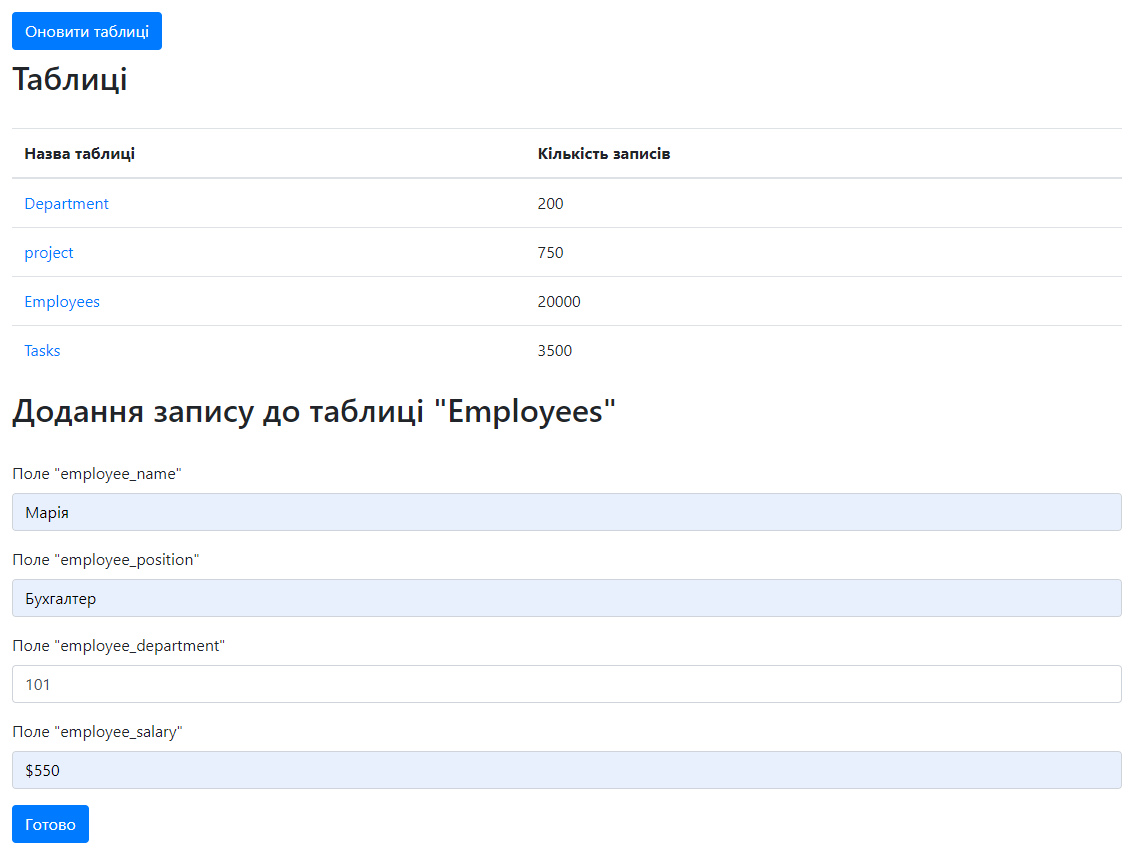
*Рис. 4. Операції з таблицею “Employees”*.



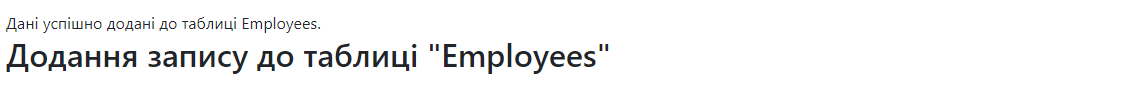
*Рис. 5. Операції з таблицею “Department”*.



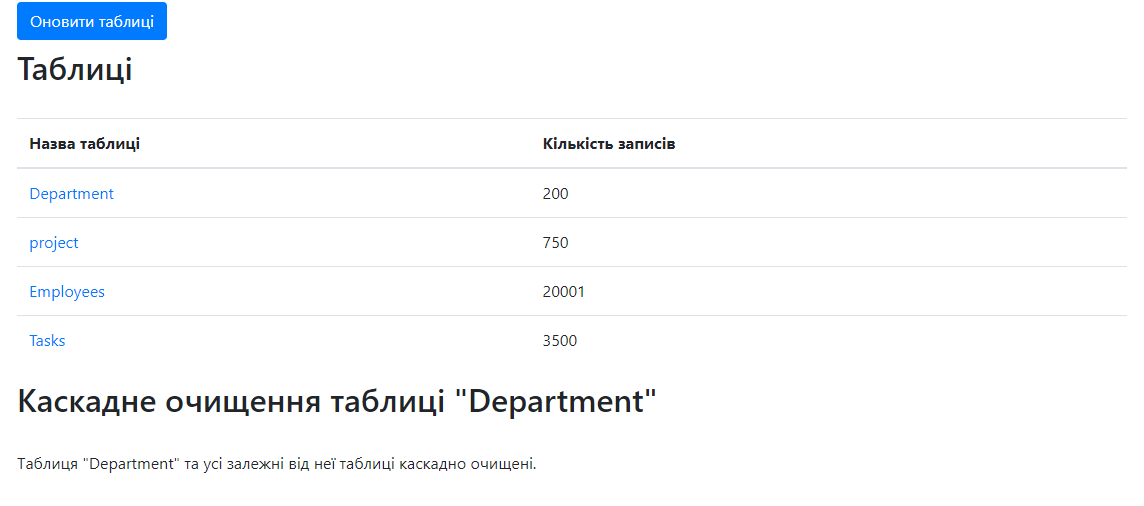
*Рис. 6. Операції з таблицею “Department”*. Спроба видалення залежного запису.



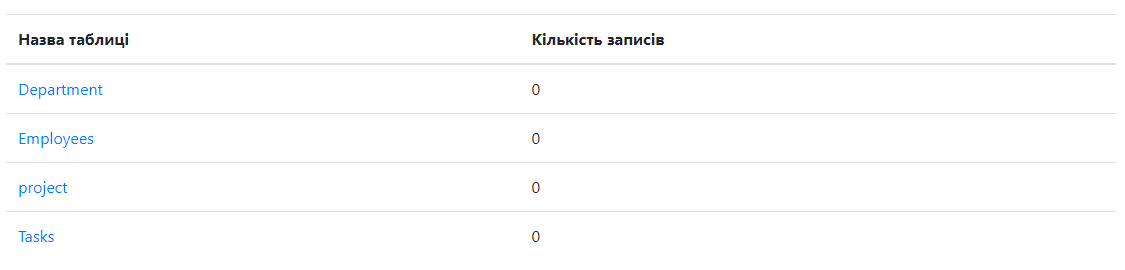
*Рис. 7. Додавання нового співробітника “Employees”*.



*Рис. 8. Повідомлення про успіх додавання співробітника.*

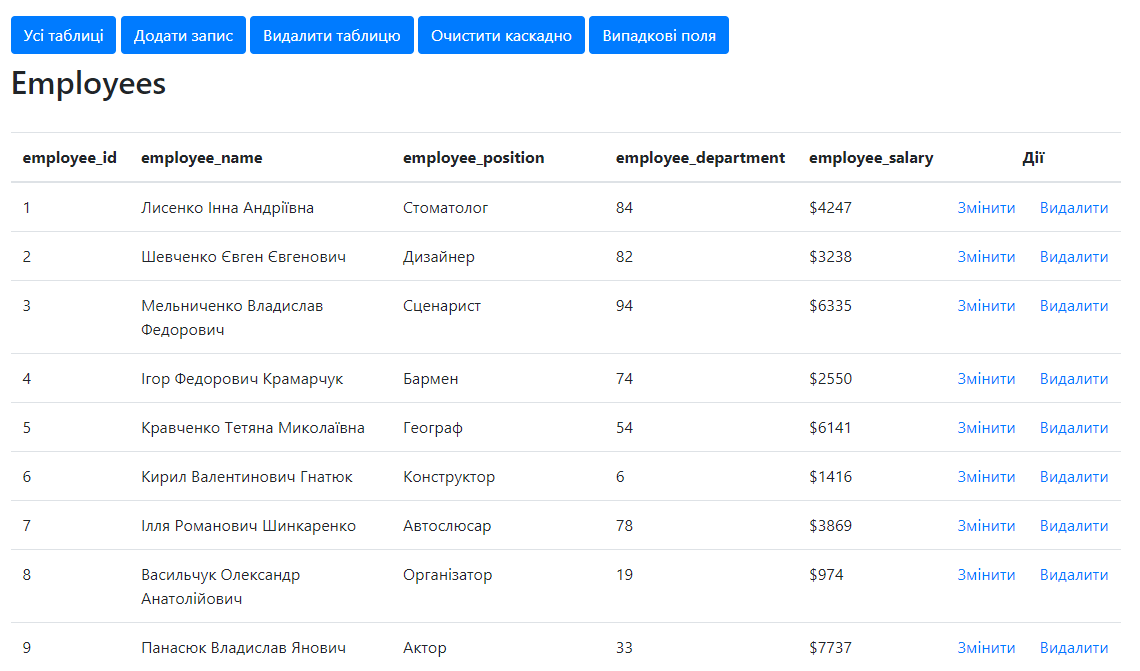


*Рис. 9. Каскадне очищення таблиць пов’язаних з “Department”*.

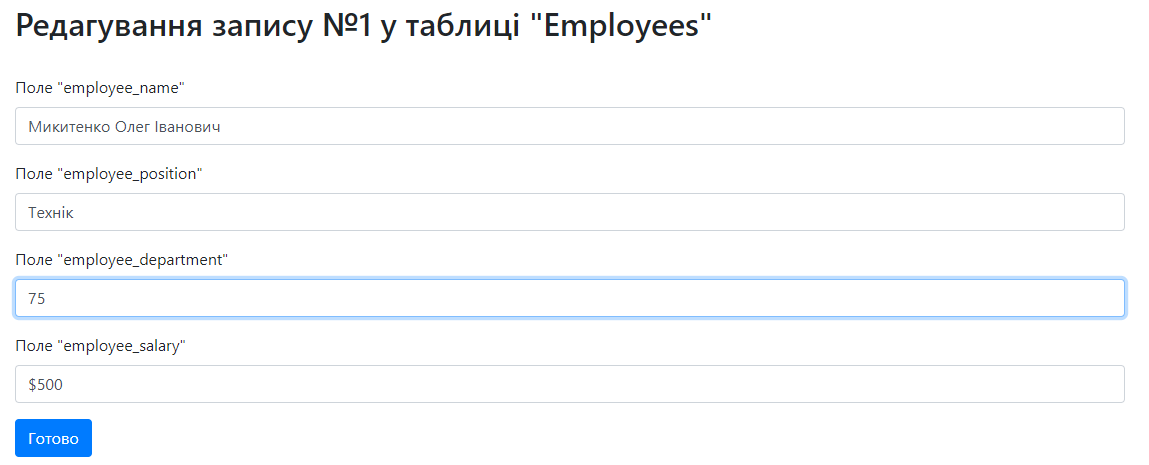


*Рис. 10. Успіх каскадного очищення*.

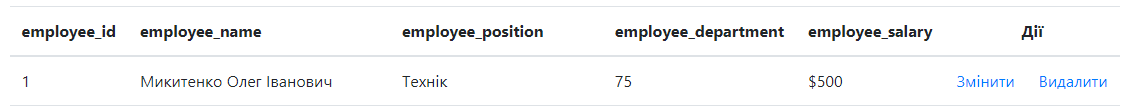
*Рис. 11. Додавання кількості випадкових записів у таблиці «Department»*.



*Рис. 12. Результат додавання випадкових записів.*



*Рис. 12. Редагування запису.*



*Рис. 13. Результат оновлення запису*.

**Програмний код. Model**

**<?php**

**class** PostgreSQLConnection {

**private** $host = "localhost";

**private** $port = 5432;

**private** $database = "postgres";

**private** $username = "postgres";

**private** $password = "1";

**private** $connection;

**public** **function** \_\_construct() {

try {

$dsn = "pgsql:host=**{$this->host}**;port=**{$this->port}**;dbname=**{$this->database}**";

$this->connection = **new** PDO($dsn, $this->username, $this->password);

$this->connection->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch (PDOException $e) {

echo "Помилка підключення: " . $e->getMessage();

}

}

**public** **function** disconnect() {

$this->connection = **null**;

}

**public** **function** getConnection() {

return $this->connection;

}

}

**class** DataBaseActions {

**private** $dbConnection;

**public** **function** \_\_construct(PostgreSQLConnection $dbConnection) {

$this->dbConnection = $dbConnection;

}

**public** **function** dataTableDelete($table\_name) {

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>Видалення таблиці **\"$table\_sanitized\"**</h2><br>";

try {

$pdo = $this->dbConnection->getConnection();

if ($this->checkForeignKey($table\_sanitized)) {

$result .= "<p>Таблиця містить зовнішній ключ. Видаліть зовнішній ключ перед видаленням таблиці.</p>";

} else {

$sql = $this->generateDeleteTableSQL($table\_sanitized);

$pdo->exec($sql);

$result .= "<p>Таблиця **\"$table\_sanitized\"** успішно видалено.</p>";

}

} catch (PDOException $e) {

$result .= "<br>Помилка виконання запиту: " . $e->getMessage();

}

return $result;

}

**public** **function** cascadeTruncate($table\_name) {

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>Каскадне очищення таблиці **\"$table\_sanitized\"**</h2><br>";

try {

$pdo = $this->dbConnection->getConnection();

$sql = $this->generateTruncateTableSQL($table\_sanitized);

$pdo->exec($sql);

$result .= "<p>Таблиця **\"$table\_sanitized\"** та усі залежні від неї таблиці каскадно очищені.</p>";

} catch (PDOException $e) {

$result .= "<br>Помилка каскадного видалення: " . $e->getMessage();

}

return $result;

}

**private** **function** checkForeignKey($table\_name) {

$pdo = $this->dbConnection->getConnection();

$sql = "SELECT COUNT(\*) FROM information\_schema.table\_constraints

WHERE constraint\_type = 'FOREIGN KEY' AND table\_name = :table\_name";

$stmt = $pdo->prepare($sql);

$stmt->bindParam(':table\_name', $table\_name, PDO::PARAM\_STR);

$stmt->execute();

$count = $stmt->fetchColumn();

return $count > 0;

}

**private** **function** generateDeleteTableSQL($table\_name) {

return "DROP TABLE IF EXISTS **\"$table\_name\"**";

}

**private** **function** generateTruncateTableSQL($table\_name) {

return "TRUNCATE **\"$table\_name\"** RESTART IDENTITY CASCADE";

}

}

**?>**

**Програмний код. View**

**<?php**

**class** Viewer {

**private** $dbConnection;

**public** **function** \_\_construct(PostgreSQLConnection $dbConnection) {

$this->dbConnection = $dbConnection;

}

**public** **function** displayTables($table\_name = **NULL**) {

$result = "";

if ($table\_name == **NULL**) {

$result = "

<h2>Таблиці</h2><br>

<table class=**\"**table**\"**>

<thead>

<tr>

<th>Назва таблиці</th>

<th>Кількість записів</th>

</tr>

</thead>

<tbody>

";

try {

$pdo = $this->dbConnection->getConnection();

$tables = $this->getAllTables($pdo);

foreach ($tables as $table) {

$result .= "<tr>";

$result .= "<td><a href=**\"**index.php?view\_table=**{$table['table\_name']}\"**>**{$table['table\_name']}**</a></td>";

$result .= "<td>**{$table['row\_count']}**</td>";

$result .= "</tr>";

}

$result .= " </tbody>

</table>

";

} catch (PDOException $e) {

echo "<br>Помилка виконання запиту: " . $e->getMessage();

}

} else {

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>**$table\_sanitized**</h2><br>";

try {

$pdo = $this->dbConnection->getConnection();

$columns\_data = $this->getColumnsNames($pdo, $table\_sanitized);

$table\_data = $this->getNecessaryTable($pdo, $table\_sanitized);

$first\_column\_saved = **false**;

$data\_id = "";

$result .= "

<table class=**\"**table**\"**>

<thead>

<tr>

";

foreach ($columns\_data as $column) {

if (!$first\_column\_saved) {

$data\_id = $column['column\_name'];

}

$result .= "<th>**{$column['column\_name']}**</th>";

$first\_column\_saved = **true**;

}

$result .= "<th style=**\"**text-align: center;**\"** colspan=**\"**2**\"**>Дії</th>";

$result .= " </tr>

</thead>

<tbody>

";

foreach ($table\_data as $row) {

$result .= "<tr>";

foreach ($row as $cell) {

$result .= "<td>**{$cell}**</td>";

}

$result .= "<td><a href=**\"**index.php?edit\_table=**$table\_sanitized**&edit=**$row**[**$data\_id**]**\"**>Змінити</a></td>";

$result .= "<td><a href=**\"**index.php?edit\_table=**$table\_sanitized**&delete=**$row**[**$data\_id**]**\"**>Видалити</a></td>";

$result .= "</tr>";

}

$result .= " </tbody>

</table>

";

} catch (PDOException $e) {

echo "<br>Помилка виконання запиту: " . $e->getMessage();

}

}

return $result;

}

**public** **function** addNote($table\_name) {

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>Додання запису до таблиці **\"$table\_sanitized\"**</h2><br>";

try {

$pdo = $this->dbConnection->getConnection();

$columns\_data = $this->getColumnsNames($pdo, $table\_sanitized);

$result .= "<form method=**\"**post**\"**>";

$id = 0;

foreach($columns\_data as $input) {

if($id != 0) {

if (stripos($input['column\_name'], 'date') !== **false** || stripos($input['column\_name'], 'deadline') !== **false**) {

$result.= "

<div class=**\"**form-group**\"**>

<label for=**\"**input**$id\"**>Поле **\"{$input['column\_name']}\"**</label>

<input name=**\"{$input['column\_name']}\"** type=**\"**date**\"** class=**\"**form-control**\"** placeholder=**\"**Введіть значення для поля '**{$input['column\_name']}**'**\"**>

</div>

";

} else {

$result.= "

<div class=**\"**form-group**\"**>

<label for=**\"**input**$id\"**>Поле **\"{$input['column\_name']}\"**</label>

<input name=**\"{$input['column\_name']}\"** type=**\"**text**\"** class=**\"**form-control**\"** placeholder=**\"**Введіть значення для поля '**{$input['column\_name']}**'**\"**>

</div>

";

}

}

$id++;

}

$result.= "

<button type=**\"**submit**\"** class=**\"**btn btn-primary**\"**>Готово</button>

</form>

";

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$this->processForm($pdo, $table\_sanitized, $columns\_data, $\_POST);

}

} catch (PDOException $e) {

echo "<br>Помилка виконання запиту: " . $e->getMessage();

}

return $result;

}

**public** **function** editTableNote($table\_name, $note\_id) {

$note\_id = trim(filter\_var($note\_id, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>Редагування запису №**$note\_id** у таблиці **\"$table\_sanitized\"**</h2><br>";

try {

$pdo = $this->dbConnection->getConnection();

$columns\_data = $this->getColumnsNames($pdo, $table\_sanitized);

$columns\_content = $this->getColumnsContentById($pdo, $table\_sanitized, $note\_id);

$result .= "<form method=**\"**post**\"**>";

$id = 0;

foreach($columns\_data as $input) {

if($id != 0) {

$column\_name = $input['column\_name'];

$value = $columns\_content[0][$column\_name];

if (stripos($input['column\_name'], 'date') !== **false** || stripos($input['column\_name'], 'deadline') !== **false**) {

$result.= "

<div class=**\"**form-group**\"**>

<label for=**\"**input**$id\"**>Поле **\"{$input['column\_name']}\"**</label>

<input name=**\"{$input['column\_name']}\"** type=**\"**date**\"** value=**\"$value\"** class=**\"**form-control**\"** placeholder=**\"**Введіть значення для поля '**{$input['column\_name']}**'**\"**>

</div>

";

} else {

$result.= "

<div class=**\"**form-group**\"**>

<label for=**\"**input**$id\"**>Поле **\"{$input['column\_name']}\"**</label>

<input name=**\"{$input['column\_name']}\"** type=**\"**text**\"** value=**\"$value\"** class=**\"**form-control**\"** placeholder=**\"**Введіть значення для поля '**{$input['column\_name']}**'**\"**>

</div>

";

}

}

$id++;

}

$result.= "

<button type=**\"**submit**\"** class=**\"**btn btn-primary**\"**>Готово</button>

</form>

";

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$this->processUpdateNote($pdo, $table\_sanitized, $columns\_data, $\_POST, $note\_id);

}

} catch (PDOException $e) {

echo "<br>Помилка виконання запиту: " . $e->getMessage();

}

return $result;

}

**public** **function** deleteTableNote($table\_name, $note\_id) {

$note\_id = trim(filter\_var($note\_id, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>Видалення запису №**$note\_id** з таблиці **\"$table\_sanitized\"**</h2><br>";

$pdo = $this->dbConnection->getConnection();

$this->processDeleteNote($pdo, $table\_sanitized, $note\_id);

return $result;

}

**public** **function** dataTableFill($table\_name) {

$table\_sanitized = trim(filter\_var($table\_name, FILTER\_SANITIZE\_SPECIAL\_CHARS));

$result = "<h2>Додання випадкових полів до таблиці **\"$table\_sanitized\"**</h2><br>";

$result .= "

<form method=**\"**post**\"**>

<div class=**\"**form-group**\"**>

<label for=**\"**input**\"**>Введіть кількість нових випадкових полей для таблиці</label>

<input name=**\"**fillTable**\"** type=**\"**number**\"** min=**\"**1**\"** max=**\"**100000**\"** class=**\"**form-control**\"** placeholder=**\"**Введіть ціле значення від 0 до 100000 **\"**>

</div>

<button type=**\"**submit**\"** class=**\"**btn btn-primary**\"**>Створити</button>

</form>

";

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$pdo = $this->dbConnection->getConnection();

$this->randomFillTable($pdo, $table\_sanitized, $\_POST);

}

return $result;

}

**private** **function** randomFillTable($pdo, $table\_name, $form\_data) {

$faker = **new** Faker\Generator();

$faker->addProvider(**new** Faker\Provider\uk\_UA\Person($faker));

$faker->addProvider(**new** Faker\Provider\uk\_UA\Company($faker));

$limit = $form\_data['fillTable'];

$startTime = microtime(**true**);

$columns\_data = $this->getColumnsNames($pdo, $table\_name);

$primary\_key\_name = $columns\_data[0]['column\_name'];

$sql = "INSERT INTO **\"$table\_name\"** (";

foreach ($columns\_data as $column) {

$column\_name = $column['column\_name'];

if ($column\_name !== $primary\_key\_name) {

$sql .= "**$column\_name**, ";

}

}

$sql = rtrim($sql, ", ") . ") VALUES ";

for ($i = 0; $i < $limit; $i++) {

$sql .= "(";

foreach ($columns\_data as $column) {

$column\_name = $column['column\_name'];

if($column\_name !== $primary\_key\_name) {

switch($column\_name) {

case "employee\_department":

$sql .= "(SELECT department\_id FROM **\"**Department**\"** ORDER BY RANDOM() LIMIT 1), ";

break;

case "department\_name":

$fake\_department = $faker->company();

$sql .= "('**$fake\_department**'),";

break;

case "employee\_name":

$fake\_name = $faker->name();

if (strpos($fake\_name, "'") !== **false**) {

$fake\_name = str\_replace("'", "`", $fake\_name);

}

$sql .= "('**$fake\_name**'),";

break;

case "employee\_position":

$fake\_position = $faker->jobTitle();

$sql .= "('**$fake\_position**'),";

break;

case "task\_employee":

$sql .= "(SELECT employee\_id FROM **\"**Employees**\"** ORDER BY RANDOM() LIMIT 1), ";

break;

case "task\_project":

$sql .= "(SELECT project\_id FROM **\"**project**\"** ORDER BY RANDOM() LIMIT 1), ";

break;

case "project\_department":

$sql .= "(SELECT department\_id FROM **\"**Department**\"** ORDER BY RANDOM() LIMIT 1), ";

break;

case "task\_date":

$sql .= "

(SELECT

date\_trunc('day', CURRENT\_DATE - INTERVAL '1 month') +

random() \* (CURRENT\_DATE - date\_trunc('day', CURRENT\_DATE - INTERVAL '1 month')) AS random\_date

),

";

break;

case "task\_deadline":

$sql .= "

(SELECT

CURRENT\_DATE +

random() \* INTERVAL '6 months' AS random\_date

)

";

break;

case "employee\_salary":

$sql .= "('$' || floor(random() \* (10000 - 300 + 1) + 300)::int), ";

break;

default:

$sql .= $this->generateRandomSqlValue($pdo, $column) . ", ";

break;

}

}

}

$sql = rtrim($sql, ", ") . "), ";

}

$sql = rtrim($sql, ", ") . ";";

$stmt = $pdo->prepare($sql);

$stmt->execute();

$endTime = microtime(**true**);

$executionTime = ($endTime - $startTime) \* 1000;

echo "Запит виконано за " . number\_format($executionTime, 2, '.', ' ') . " мс";

}

**private** **function** generateRandomSqlValue($pdo, $column) {

switch ($column['data\_type']) {

case 'integer':

return 'floor(random() \* 100)';

case 'character varying':

return "(SELECT left(md5(random()::text), 10))";

default:

return 'NULL';

}

}

**private** **function** getAllTables(PDO $pdo) {

$sql = "

SELECT table\_name, (xpath('/row/cnt/text()', xml\_count))[1]::text::int AS row\_count

FROM (

SELECT table\_name,

query\_to\_xml(format('SELECT COUNT(\*) AS cnt FROM %I.%I', table\_schema, table\_name), false, true, '') AS xml\_count

FROM information\_schema.tables

WHERE table\_schema = 'public'

) AS counts;

";

$stmt = $pdo->prepare($sql);

$stmt->execute();

$tables = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $tables;

}

**private** **function** getNecessaryTable(PDO $pdo, $table\_name, $note\_id = **NULL**, $primary\_key = **NULL**) {

$primary\_key\_table = $this->getColumnsNames($pdo, $table\_name, **NULL**, 1);

$primary\_key\_name = $primary\_key\_table[0]['column\_name'];

if($note\_id != **NULL** && $primary\_key != **NULL**) {

$sql = "SELECT \* FROM **\"$table\_name\"** WHERE **$primary\_key** = '**$note\_id**'";

} else {

$sql = "SELECT \* FROM **\"$table\_name\"** ORDER BY **$primary\_key\_name** ASC";

}

$stmt = $pdo->prepare($sql);

$stmt->execute();

$table\_content = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $table\_content;

}

**private** **function** getColumnsNames(PDO $pdo, $table\_name, $limit = **NULL**) {

if($limit == **NULL**) {

$sql = "SELECT column\_name, data\_type FROM information\_schema.columns WHERE table\_name = '**$table\_name**' ORDER BY ordinal\_position";

} else {

$sql = "SELECT column\_name, data\_type FROM information\_schema.columns WHERE table\_name = '**$table\_name**' ORDER BY ordinal\_position LIMIT **$limit**";

}

$stmt = $pdo->prepare($sql);

$stmt->execute();

$columns\_names = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $columns\_names;

}

**private** **function** processForm(PDO $pdo, $table\_name, $columns\_data, $form\_data) {

try {

$valid\_data = [];

$current\_date = **new** DateTime();

$one\_week\_later = (clone $current\_date)->add(**new** DateInterval('P7D'))->format('Y-m-d H:i:sP');

foreach ($columns\_data as $column\_info) {

if ($column\_info === $columns\_data[0]) {

continue;

}

$column = $column\_info['column\_name'];

if (is\_scalar($column) && isset($form\_data[$column]) && !empty($form\_data[$column])) {

$valid\_data[$column] = trim(filter\_var($form\_data[$column], FILTER\_SANITIZE\_SPECIAL\_CHARS));

if (isset($form\_data[$column]) && !empty($form\_data[$column])) {

if (stripos($column, 'date') !== **false** || stripos($column, 'deadline') !== **false**) {

$date\_time = DateTime::createFromFormat('Y-m-d', $form\_data[$column]);

$valid\_data[$column] = $date\_time ? $date\_time->format('Y-m-d H:i:sP') : $one\_week\_later;

} else {

$valid\_data[$column] = trim(filter\_var($form\_data[$column], FILTER\_SANITIZE\_SPECIAL\_CHARS));

}

}

} else {

echo "Поле '**$column**' обов'язкове для заповнення.";

return;

}

}

$columns\_str = implode(', ', array\_keys($valid\_data));

$values\_str = implode(', ', array\_fill(0, count($valid\_data), '?'));

$sql = "INSERT INTO **\"$table\_name\"** (**$columns\_str**) VALUES (**$values\_str**)";

$stmt = $pdo->prepare($sql);

$i = 1;

foreach ($valid\_data as $value) {

$stmt->bindValue($i++, $value);

}

$stmt->execute();

echo "Дані успішно додані до таблиці **$table\_name**.";

} catch (PDOException $e) {

echo "Помилка при вставці даних: " . $e->getMessage();

}

}

**private** **function** processDeleteNote(PDO $pdo, $table\_name, $note\_id) {

try {

$first\_columnQuery = "

SELECT column\_name

FROM information\_schema.columns

WHERE table\_name = :table\_name

ORDER BY ordinal\_position

LIMIT 1

";

$stmt = $pdo->prepare($first\_columnQuery);

$stmt->bindParam(':table\_name', $table\_name, PDO::PARAM\_STR);

$stmt->execute();

$first\_column = $stmt->fetchColumn();

$delete\_query = "DELETE FROM **\"$table\_name\"** WHERE **$first\_column** = :note\_id";

$stmt = $pdo->prepare($delete\_query);

$stmt->bindParam(':note\_id', $note\_id, PDO::PARAM\_INT);

$stmt->execute();

echo "Запис успішно видалена із таблиці **$table\_name**.";

} catch (PDOException $e) {

$errorCode = $e->getCode();

$errorMessage = $e->getMessage();

if ($errorCode == '23503') {

echo "Помилка зовнішнього ключа: **$errorMessage**";

} else {

echo "Інша помилка: **$errorMessage**";

}

}

}

**private** **function** getColumnsContentById($pdo, $table\_name, $note\_id) {

$first\_element = $this->getColumnsNames($pdo, $table\_name, 1);

foreach($first\_element as $primary\_key) {

$table\_primary\_key = $primary\_key['column\_name'];

break;

}

$result = $this-> getNecessaryTable($pdo, $table\_name, $note\_id, $table\_primary\_key);

return $result;

}

**private** **function** processUpdateNote(PDO $pdo, $table\_name, $columns\_data, $form\_data, $note\_id) {

try {

$update\_query = "UPDATE **\"$table\_name\"** SET ";

$primary\_key\_name = $columns\_data[0]['column\_name'];

foreach ($columns\_data as $input) {

$column\_name = $input['column\_name'];

if ($column\_name != $primary\_key\_name) {

$update\_query .= "**$column\_name** = :**$column\_name**, ";

}

}

$update\_query = rtrim($update\_query, ', ');

$update\_query .= " WHERE **$primary\_key\_name** = :**$primary\_key\_name**";

$stmt = $pdo->prepare($update\_query);

foreach ($columns\_data as $input) {

$column\_name = $input['column\_name'];

if ($column\_name != $primary\_key\_name) {

$input\_value = trim(filter\_var($form\_data[$column\_name], FILTER\_SANITIZE\_SPECIAL\_CHARS));

$stmt->bindValue(":**$column\_name**", $input\_value);

}

}

$stmt->bindValue(":**$primary\_key\_name**", $note\_id);

$stmt->execute();

echo "Дані успішно оновлено у таблиці **$table\_name**.";

} catch (PDOException $e) {

echo "Помилка оновлення даних: " . $e->getMessage();

}

}

}

**?>**

**Програмний код. Controller**

**<?php**

**require\_once('model.php');**

**require\_once('view.php');**

**require\_once ('vendor/autoload.php');**

**$connection = new PostgreSQLConnection();**

**$viewer = new Viewer($connection);**

**$database = new DataBaseActions($connection);**

**?>**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">**

**<title>CPMS, Silin KV-12</title>**

**</head>**

**<body>**

**<div class="container mt-4">**

**<?php if(isset($\_GET['view\_table'])): ?>**

**<a class="btn btn-primary mb-2" href="index.php">Усі таблиці</a>**

**<a class="btn btn-primary mb-2" href="index.php?add\_note=<?=$\_GET['view\_table']?>">Додати запис</a>**

**<a class="btn btn-primary mb-2" href="index.php?delete\_table=<?=$\_GET['view\_table']?>">Видалити таблицю</a>**

**<a class="btn btn-primary mb-2" href="index.php?truncate\_table=<?=$\_GET['view\_table']?>">Очистити каскадно</a>**

**<a class="btn btn-primary mb-2" href="index.php?fill\_table=<?=$\_GET['view\_table']?>">Випадкові поля</a>**

**<?= $viewer->displayTables($\_GET['view\_table']); ?>**

**<?php endif; ?>**

**<?php if(!isset($\_GET['view\_table'])): ?>**

**<a class="btn btn-primary mb-2" href="index.php">Оновити таблиці</a>**

**<?= $viewer->displayTables(); ?>**

**<?php endif; ?>**

**<?php if(isset($\_GET['add\_note'])): ?>**

**<?= $viewer->addNote($\_GET['add\_note']); ?>**

**<?php endif; ?>**

**<?php if(isset($\_GET['edit\_table']) && isset($\_GET['edit'])): ?>**

**<?= $viewer->editTableNote($\_GET['edit\_table'], $\_GET['edit']); ?>**

**<?php endif; ?>**

**<?php if(isset($\_GET['edit\_table']) && isset($\_GET['delete'])): ?>**

**<?= $viewer->deleteTableNote($\_GET['edit\_table'], $\_GET['delete']); ?>**

**<?php endif; ?>**

**<?php if(isset($\_GET['delete\_table'])): ?>**

**<?= $database->dataTableDelete($\_GET['delete\_table']);?>**

**<?php endif; ?>**

**<?php if(isset($\_GET['truncate\_table'])): ?>**

**<?= $database->cascadeTruncate($\_GET['truncate\_table']);?>**

**<?php endif; ?>**

**<?php if(isset($\_GET['fill\_table'])): ?>**

**<?= $viewer->dataTableFill($\_GET['fill\_table']);?>**

**<?php endif; ?>**

**</div>**

**<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>**

**<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js"></script>**

**<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>**

**<?php $connection->disconnect(); ?>**

**</body>**

**</html>**