

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Розрахунково-графічна робота**

з дисципліни

**«Бази даних і засоби управління»**

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

Виконав: студент ІІI курсу

ФПМ групи КВ-23

Зінєдін Шайдін

Перевірив:

Київ – 2024

*https://github.com/9koto/Lab2-Zinedin-Shaidin*

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

*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 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).

**Виконання роботи**

Нормалізована база даних з першої лабораторної роботи

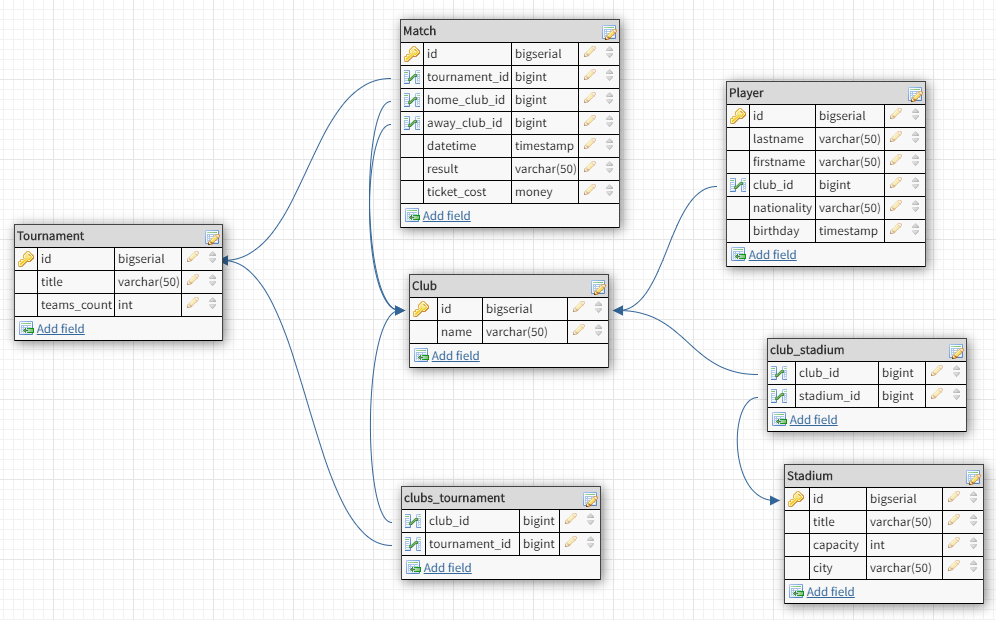


Рисунок 1 - Нормалізована база даних.

**Опис програми**

**Структура программи**

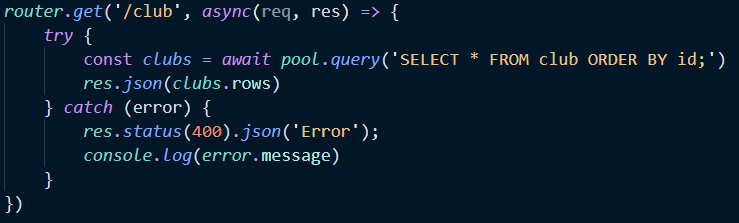
Програма складається з 4 основних модулів.

1. Файл index.js, який запускає сервер для прийому запитів клієнта;
2. Файли роутингу на серверній частині(для кожної таблиці свій окремий), вони обробляють запити, які надійшли на сервер;
3. Файл Table.js, модуль, для роботи в браузері з таблицею, а саме – пошук, створення, редагування, видалення даних з таблиці бази даних;
4. Файл routes.js для навігації користувача по програмі;
5. Модуль підключення до БД.

Навігація по модулях

1. Сервер
   1. [Запуск сервера.](#index)
2. Файли роутингу
   1. [Для таблиці club](#clubRoutes);
   2. [Для таблиці stadium;](#stadiumRoutes)
   3. [Для таблиці player;](#playerRoutes)
   4. [Для таблиці tournament;](#tournamentRoutes)
   5. [Для таблиці match;](#matchRoutes)
   6. [Для таблиці clubs\_stadiums;](#clubs_tournaments_routes)
   7. [Для таблиці clubs\_tournametns.](#clubs_tournaments_routes)
3. Модулі для роботи з таблицями.
   1. [Задання сутностей;](#entity)
   2. [Модуль, який відображає їх у браузері.](#table)
4. Навігація користовуча
   1. [Модуль навігації;](#routes)
   2. [Відбраження навігації(Меню).](#menu)
5. [Модуль для підключення до БД.](#db)

**Обробка помилок**

Для цього завдання кожен SQL запит було виконанно в контсрукції try\_catch, приклад:  


Коли при запиті виникає якась помилка, то сервер поперне користувача відповідь зі статусом помилки(400) і текстом “error”. І коли користувач на свій запит до сервера отримує статус 400 йому на єкран буде виведено повідомлення про помилку. Приклад:   


Коли, користувач отримує відповідь, яка має статус 400(тобто помилки), функція window.alert виведе на єкран, що сталася помилка.

**Робота delete запитів**

В своїй роботі, при видаленні рядків, що мають посилання на зовнішні ключі, я встановлював значення NULL в рядку(якщо він дозволенний в таблиці), на який посилався данних зовнішній ключ, якшо NULL не дозволенно – видалява рядок.

Приклад з моеї програми, при видалення рядку з таблиці “club”:

*router.delete*('/club/:id', *async*(req, res) => {

*try* {

        const{*id*}=*req.params*

        const*qu*=`*DELETE FROM club\_stadium WHERE club\_id =*${id}*;*

*DELETE FROM clubs\_tournaments WHERE club\_id =*${id}*;*

*DELETE FROM match WHERE home\_club\_id =*${id}*OR away\_club\_id =*${id}*;*

*UPDATE player SET club\_id = NULL WHERE club\_id =*${id}*;*

*DELETE FROM club WHERE id =*${id}*;*`

        const*response*=*await pool.query*(qu)

*res.json*('Successfuly deleted')

    } *catch* (error) {

*res.status*(400)*.json*('Error');

*console.log*(*error.*message)

    }

})

Тут можна побачити, що в поле club\_id таблиці player встановлюється значення NULL, а в інших випадках видаляються. Тому, що наприклад, футболіст може існувати без клубу, а футбольний матч, без цього клубу – вже не може.

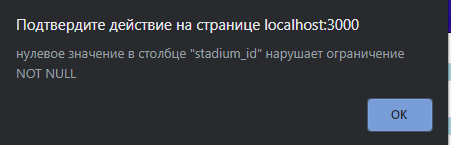
**Дослідження режимів обмеження ON DELETE**

*1. Режим CASCADE*

При видаленні запису з таблиці Stadium, запис з таблиці Club\_Stadium видаляється.

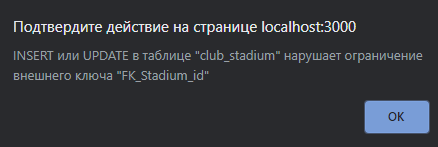
*2. Режим SET NULL*

При видаленні запису з таблиці Stadium, stadium\_id запис з таблиці Club\_Stadium встановлюється в null. Якщо в налаштуваннях таблиці вказати, що stadium\_id не може бути null, то перехоплюємо повідомлення про помилку.



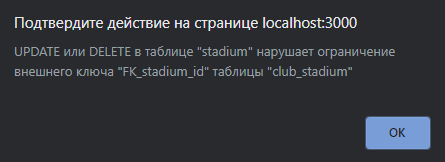
*3. Режим SET DEFAULT (значення за замовчуванням = 0)*

При видаленні запису з таблиці Stadium, перехоплюємо повідомлення про помилку, так як стадіону з id = 0 не існує.



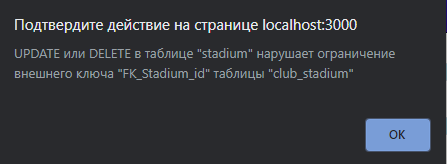
*4. Режим NO ACTION*

При видаленні запису з таблиці Stadium, виникає помилка; це поведінка за замовчуванням.



*5.Режим RESTRICT*

При видаленні запису з таблиці Stadium, виникає помилка; це пояснюється тим, що режим RESTRICT не дає можливості видалити батьківський рядок, якщо в нього є дочірні.



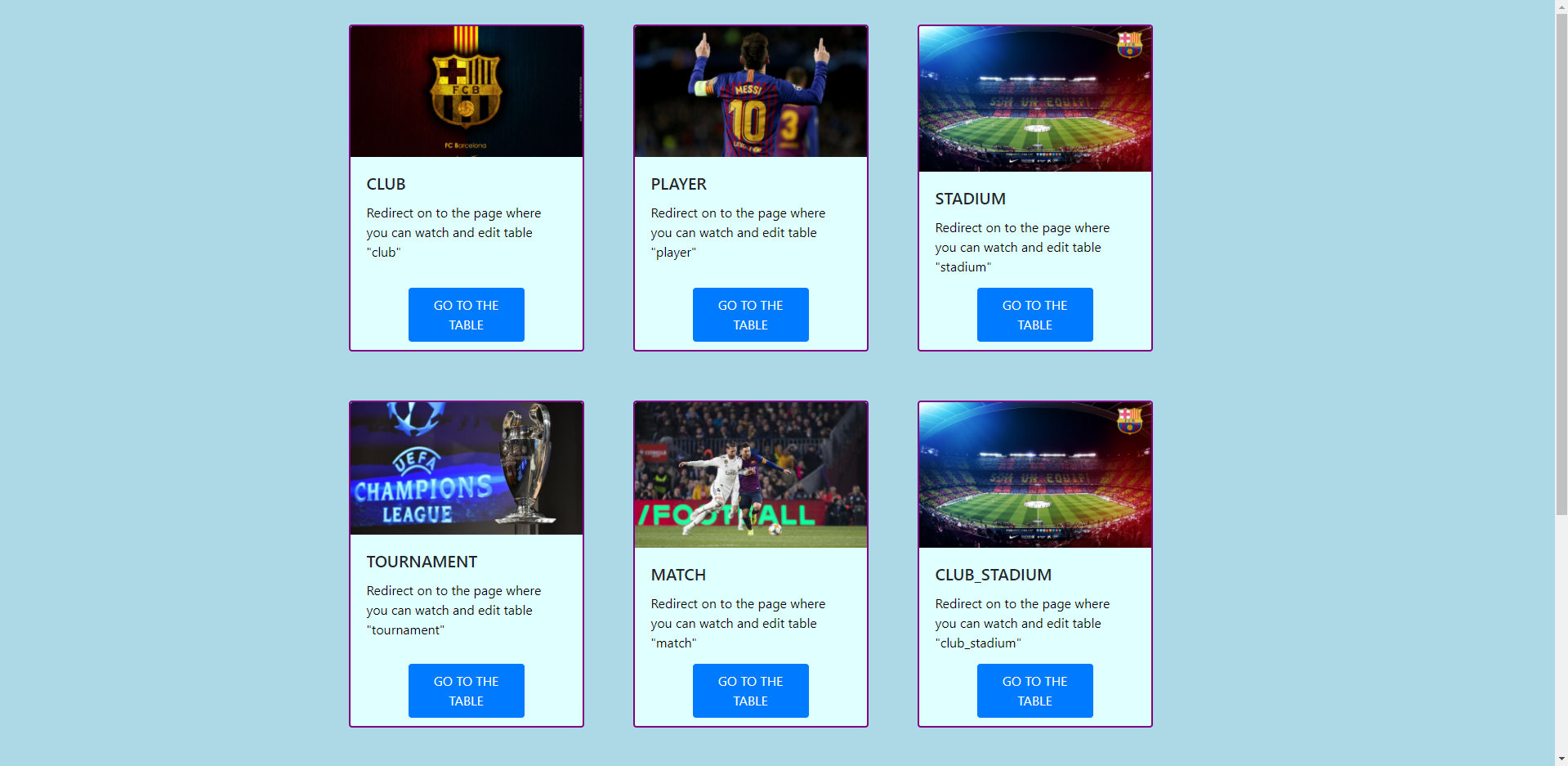
Головне меню програми  


Рисунок 2 - Меню

**Інтерфейс програми**

При виконані графічної роботи для всіх таблиць бази даних було розробленно однаковий інтерфейс. Розглянемо його на прикладі найбільшої з них.

При переході на сторінку таблиці відображаються усі її записи.

Рисунок 3 - Приклад Таблиці.

Далі кожен запис можна видалити за допомогою кнопки “Delete”.  


Рисунок 4 - Кнопка видалення.  
Або можна редагувати натиснувши кнопку “Edit”.  

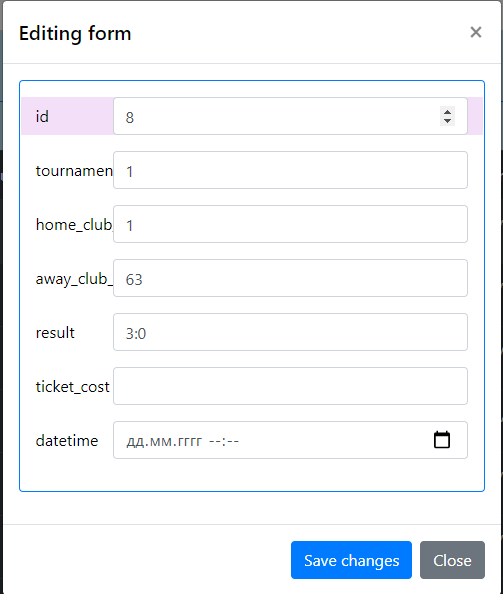

Рисунок 5 - Кнопка редагування.  
Після чого відкриється невелике віконце де можна ввести нові дані для цього рядка.  


Рисунок 6 - Форма редагування даних.  
Також вверху сторінки є форма для пошуку або створення нових записів.  


Рисунок 7 - Пошук рядків.  
Для того, щоб почати пошук – необхідно ввести інформацію хоча б в одне поле і натиснути кнопку “Find”.  

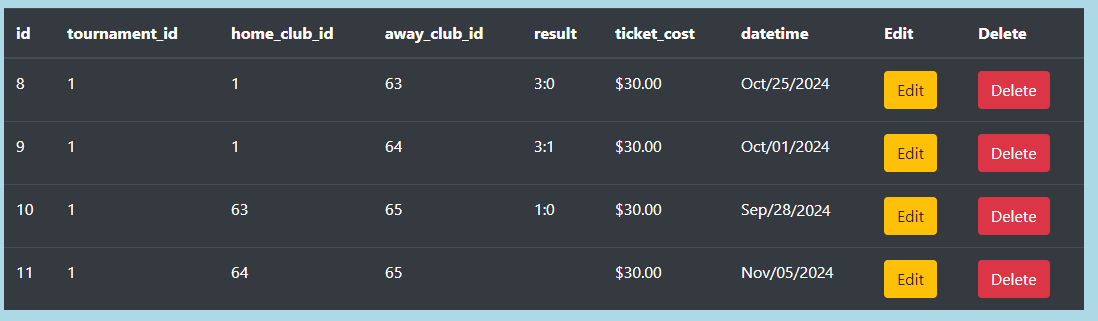

Рисунок 8 - Кнопка пошуку.  
Після чого, в таблиці будуть значення, які відповідають значенням в цій формі. Для прикладу, знайдемо всі рядки, в яких tournament\_id = 1. Для цього в поле tournament\_id введемо 1 і натиснемо кнопку “Find”. І отримаємо…  


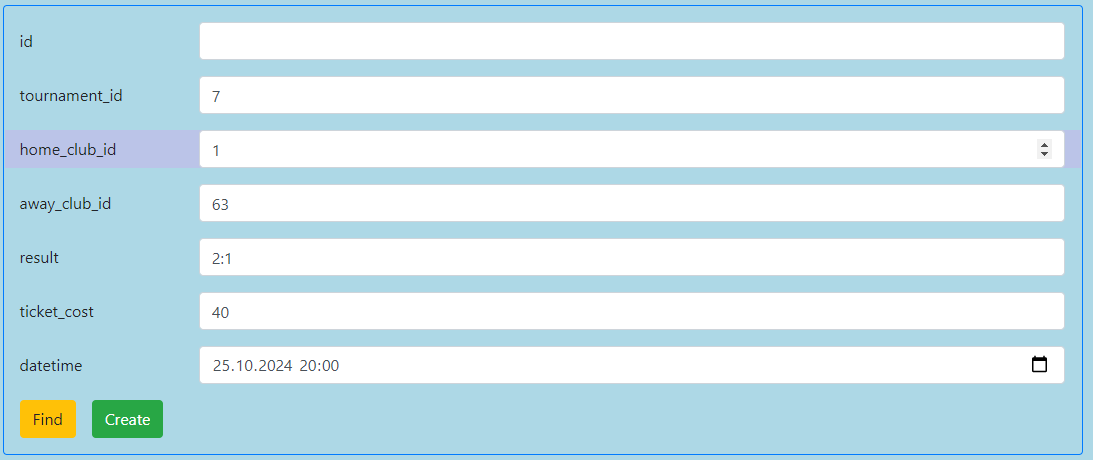
Рисунок 9 - Приклад пошуку.  
  
Далі для прикладу створимо, нехай такий рядок.  


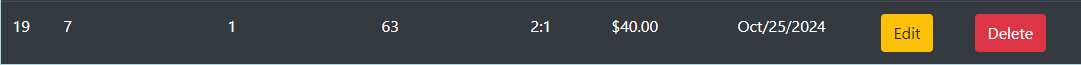
Рисунок 10 - Приклад створення.  
Після чого натиснемо кнопку “Create” і він допишеться в кінець таблиці.  


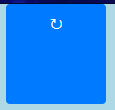
Рисунок 11 - Створений рядок.  
Також є кнопка, щоб видалити параметри пошуку, що виведе на екран всі рядки цієї таблиці.  


Рисунок 12 - Кнопка видалення пошуку.

Для виконання цієї графічної роботи, я використовував середовище розробки VS Code. Для підключення бази даних використовувалась платформа Node.js та додатково бібліотека “PG”.

Файл для підключення до БД.  
db.js

const*Pool*=*require*('pg')*.Pool*

//*connecting to the local database*

const*pool*=newPool({

*user*:"postgres",

*password*:"1928sfsf",

*host*:"localhost",

*port*:5432,

*database*:"footballtournaments",

});

module*.*exports = pool;

Код програми написан мовою програмування JavaScript.