МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА» ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Курсова робота

# з дисципліни «Бази даних»

Виконав: ст. гр. АС-212

Гудименко Євгеній Анатолійович

Перевірила Зіноватна С. Л.

Одеса – 2023

ЗМІСТ

ВСТУП 3

1. [Аналіз предметної області 4](#_TOC_250018)
   1. [Правила предметної області 4](#_TOC_250017)
   2. [Опис вимог 4](#_TOC_250016)
   3. [Опис об’єктів та їх атрибутів 5](#_TOC_250015)
2. [Проєктування бази даних 7](#_TOC_250014)
   1. [Аналіз аномалій, нормалізація 7](#_TOC_250013)
   2. [Обґрунтування первинних ключів 7](#_TOC_250012)
   3. [Концептуальна модель даних 7](#_TOC_250011)
   4. [Реляційна модель даних 8](#_TOC_250010)
3. [Запити 10](#_TOC_250009)
   1. [Опис запитів 10](#_TOC_250008)
4. [Користувачі системи 13](#_TOC_250007)
   1. [Опис груп користувачів 13](#_TOC_250006)
   2. [Засоби керування доступом 13](#_TOC_250005)
   3. [Функціональні можливості 13](#_TOC_250004)
5. [Інтерфейс інформаційної системи 15](#_TOC_250003)
   1. [Опис інтерфейсу 15](#_TOC_250002)
   2. [Керівництво користувача 17](#_TOC_250001)
   3. [Контрольний приклад 19](#_TOC_250000)

ВИСНОВКИ 21

СПИСОК ЛІТЕРАТУРИ 22

ДОДАТОК А 23

ВСТУП

# Завдання на виконання:

**Предметна область:** "Футбольна ліга".

**Обрані види діяльності:** облік проведених командою ігор у різних змаганнях; облік результатів й опис голів (хто й коли забив); облік тренерської роботи різних тренерів, переходів спортсменів; облік отриманих травм.

**Обрана мова програмування:** Node JS.

# 1 Аналіз предметної області

# Правила предметної області

На футбольній лізі може бути декілька команд. Прийнято, що команда - це група гравців, які грають разом в одному чемпіонаті. Кожна команда може відноситись до певної ліги. Одна ліга може містити багато команд, але одна команда може належати лише до однієї ліги. Один гравець може грати лише за однією командою на певний момент, але одна команда може мати багато гравців. Один тренер може тренувати багато команд, але одна команда може мати лише одного тренера.

Кожна команда може мати свій гравецьський склад, який включає імена гравців, їх вік, позицію на полі та інші характеристики. Один гравець може мати лише одну позицію в складі команди. Кожна команда також може мати тренера, який керує командою під час тренувань та матчів.

Крім того, в футбольній лізі можуть бути матчі між командами. Кожен матч має свою дату, місце проведення, рахунок та інші деталі. Один матч може включати дві команди, які змагаються одна проти одної.

Також у футбольній лізі можуть проводитись трансфери гравців. Один клуб може купувати або продавати гравців іншим клубам за певну суму грошей. Кожен трансфер має свого постачальника (клуб-продавця) та одного або більше покупця (клуб-покупця).

# Опис вимог

У таб. 1.2.1 представлено опис вимог до проєктованої системи.

Таблиця 1.2.1 – Опис вимог

|  |  |
| --- | --- |
| **Вимога** | **Опис** |
| Облік проведених командою ігор у різних змаганнях | Виводити список проведених ігор для кожної команди. |
| Облік результатів й опис голів | Виводити список голів із вказанням футболіста що їх забив. |

Продовження таблиці 1.2.1

|  |  |
| --- | --- |
| **Вимога** | **Опис** |
| Облік тренерської роботи різних тренерів, переходів спортсменів | Виводити список викликів тренувань відповідних команд та список трансферів |
| Облік отриманих травм | Виводити список травм та дати їх отримання для ведення документації |

# Опис об’єктів та їх атрибутів

У таб. 1.3.1 представлено опис об’єктів та їх атрибутів.

Таблиця 1.3.1 – Опис об’єктів та їх атрибутів

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вимоги замовника | № | Технічні вимоги |
| 1 | Облік проведених командою ігор у різних змаганнях | 1.1 | БД повинна зберігати дані про гру кожної команди у футбольній лізі |
|  |  | 1.2 | БД повинна зберігати дані про участь кожної команди у різних змаганнях та зберігати їх результат |
| 2 | Облік результатів й опис голів (хто й коли забив) | 2.1 | При внесенні гри до БД, треба записати дані про голи, час забиття голу та бомбардир голу |
|  |  | 2.2 | Можливість пошуку голів по гравцю |
| 3 | Облік тренерської роботи різних тренерів | 3.1 | У БД повинен бути прямий доступ до виграшів та поразок, щоб обчислювати відсоток успіху кожного тренеру всередині програми |
|  |  | 3.2 | Можливість пошуку тренера команди, що виграла |

Продовження таблиці 1.3.1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 3.3 | У БД повинна бути створена сутність тренера, що зберігає дані про команди що він тренує |
| 4 | Переходів спортсменів | 4.1 | У БД повинна бути сутність, відповідаюча за трекінг з якої та до якої команди перейшов кожний спортсмен |
|  |  | 4.2 | Можливість переглянути в якій команді зараз присутній кожен із спорсменів |
| 5 | Облік участі спортсменів у зборах | 5.1 | У БД повинна бути сутність що відповідає за зберігання даних про відвідування кожного спортсмену у кожних зборах своєї команди |
|  |  | 5.2 | У бд повинна бути сутність що відповідає за збереженню інформації про збори |
| 6 | Облік отриманих травм | 6.1 | У бд повинна бути сутність що зберігає травми та дату нанесення цієї травми для кожного футболіста |
|  |  | 6.2 | Можливість пошуку футболіста за травмою |

# 2 Проєктування бази даних

# Аналіз аномалій, нормалізація

**Аномалії видалення.** При видаленні команди втрачається інформація про матчі в яких вони перебували . При видалені команди втрачаємо інформацію про тренера.

**Аномалія введення.** Ми не можемо ввести інформацію про гравців та їх статистику до того, як ми введемо інформацію ро команди, до якої вони належать. Ми не можемо ввести інформацію про матчі, допоки ми не додаємо запис про змагання де вони присутні.

**Рішення:** винесення зв’язків тренера та команди, команди та спортсмена в окремі об’єкти.

# Обґрунтування первинних ключів

Для особ (тренер, спортсмен) первинним ключем неможливо обрати ПІБ, так як ПІБ не може бути унікальним; необхідно ввести унікальний ІД. Для відстеження матчів, голів, травм у кожного з них повинен бути унікальний ІД. Для кожної команди потрібен бути унікальний ІД, так як назва команди не може бути унікальною, як і назва змагання, або трави.

# Концептуальна модель даних

На рис. 2.3.1 представлено концептуальну модель даних.

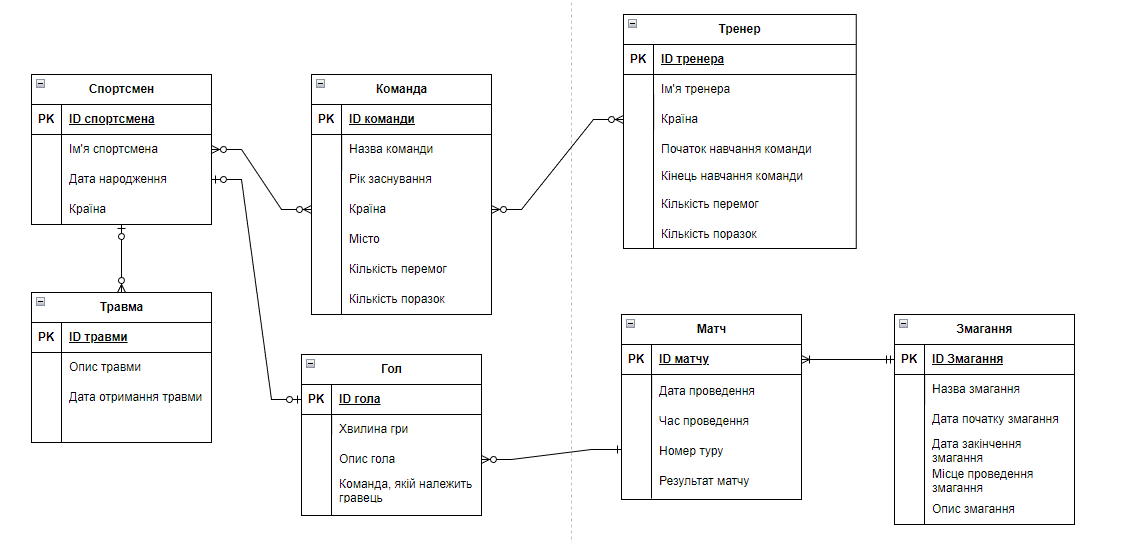


Рисунок 2.3.1 – Концептуальна модель даних

# Реляційна модель даних

Спортсмен(**IDспортсмена**, ПІБСпортсмена, ДатаНародження, Країна)

Травма(**IDТравми**, IDСпортсмена ОписТравми, ДатаОтриманняТравми)

Команда**(IDкоманди**, НазваКоманди, РікЗаснування, Країна, Місто, КількістьПеремог, КількістьПоразок)

Гол(**IDГола**, ХвилинаГри, ОписГола, КомандаГравця)

Тренер(**IDТренера**, ПІБТренера, Країна, КількістьПеремог, КількістьПоразок)

Матч(**IDМатчу**, IDЗмагання, ДатаПроведення, ЧасПроведення, НомерТуру, РезультатМатчу)

Змагання(**IDЗмагання**, НазваЗмагання, ДатаПочаткуЗмагання, ДатаЗакінченняЗмагання, МісцеПроведенняЗмагання, ОписЗмагання)

Трансфер(**IDТрансферу**, IDКомандиВідкуди, IDКомандиКуди, IDГравця)

СпорсменКоманда(**IDСпортсменКоманда**, IDСпортсмен, IDКоманда)

ТренерКоманда(**IDТренерКоманда**, IDТренер, IDКоманда, ПочатокНавчанняКоманди, КінецьНавчанняКоманди)

# 3 Запити

# 3.1 Опис запитів

# Вибір з декількох таблиць із сортуванням.

Список переломів спорстменів

SELECT ПІБСпортсмена

FROM Спортсмен

JOIN Травма ON Спортсмен.IDспортсмена = Травма.IDСпортсмена

WHERE Травма.ОписТравми LIKE 'Зламана%'

ORDER BY Спортсмен.ПІБСпортсмена;

# Завдання умови відбору з використанням предиката LІKE.

Знайти всіх Спортсменів, прізвище яких починається на “А”.

SELECT \*

FROM Спортсмен

WHERE Спортсмен.ПІБСпортсмена LIKE 'А%';

# Завдання умови відбору з використанням предиката BETWEEN.

Список травм, отриманих у період між 2022-01-01 та 2022-12-33

SELECT \*

FROM Травма

WHERE Травма.ДатаОтриманняТравми BETWEEN '2022-01-01' AND '2022-12-31';

# Агрегатна функція без угруповання.

Скільки спортсменів старше 20 років?

SELECT COUNT(\*)

FROM Спортсмен

WHERE Спортсмен.ДатаНародження >= DATE\_SUB(CURDATE(), INTERVAL 20 YEAR);

# Агрегатна функція з угрупованням.

# Список травм що були отриманні спортсменами з зв’язком з їх тренерами

# SELECT Тренер.IDТренера, Тренер.ПІБТренера, COUNT(\*) AS Тренуючі\_команди FROM Тренер JOIN СпортсменКоманда ON Тренер.IDТренера = СпортсменКоманда.IDТренера JOIN Травма ON СпортсменКоманда.IDСпортсмен = Травма.IDСпортсмена GROUP BY Тренер.IDТренера, Тренер.ПІБТренера;

# Використання предиката ALL або ANY.

Тренер що має найбільшу кількість травмованих спортсменів

SELECT Тренер.IDТренера, Тренер.ПІБТренера

FROM Тренер

WHERE Тренер.IDТренера = ALL (SELECT Тренер.IDТренера

FROM Тренер

JOIN СпортсменКоманда ON Тренер.IDТренера = СпортсменКоманда.IDТренера

JOIN Травма ON СпортсменКоманда.IDСпортсмен = Травма.IDСпортсмена

GROUP BY Тренер.IDТренера

ORDER BY COUNT(\*) DESC

LIMIT 1);

# Корельований підзапит.

Цей запит виводить інформацію про тренерів, які мають максимальну кількість спортсменів у командах, з якими вони пов'язані.

SELECT ТренерКоманда.IDТренера, Тренер.ПІБТренера, COUNT(\*) AS Кількість\_пацієнтів

FROM ТренерКоманда

JOIN Тренер ON ТренерКоманда.IDТренера = Тренер.IDТренера

JOIN СпортсменКоманда ON ТренерКоманда.IDКоманда = СпортсменКоманда.IDКоманда

GROUP BY ТренерКоманда.IDТренера, Тренер.ПІБТренера

HAVING COUNT(\*) = (SELECT MAX(CountPatients)

FROM (SELECT COUNT(\*) AS CountPatients

FROM ТренерКоманда

JOIN СпортсменКоманда ON ТренерКоманда.IDКоманда = СпортсменКоманда.IDКоманда

GROUP BY ТренерКоманда.IDТренера) AS T);

# Запит на заперечення.

Тренери, що не тренують ні однієї команди

SELECT Тренер.IDТренера, Тренер.ПІБТренера

FROM Тренер

LEFT JOIN ТренерКоманда ON Тренер.IDТренера = ТренерКоманда.IDТренера

WHERE ТренерКоманда.IDТренерКоманда IS NULL;

SELECT IDТренера, ПІБТренера

FROM Тренер

WHERE IDТренера NOT IN (SELECT IDТренера FROM ТренерКоманда);

SELECT Тренер.IDТренера, Тренер.ПІБТренера

FROM Тренер

WHERE NOT EXISTS (SELECT \*

FROM ТренерКоманда

WHERE Тренер.IDТренера = ТренерКоманда.IDТренера);

# Операція об'єднання UNІON із включенням коментарю в кожен рядок.

Операція об’єднання з коментарем про Тренера “має максимальну кількість команд” “не має в цей час команд

SELECT ПІБТренера, 'Має максимальну кількість хворих' AS Коментар

FROM Тренер

JOIN СпортсменКоманда ON Тренер.IDТренера = СпортсменКоманда.IDТренера

JOIN Травма ON СпортсменКоманда.IDСпортсмен = Травма.IDСпортсмена

GROUP BY Тренер.IDТренера, Тренер.ПІБТренера

UNION

SELECT ПІБТренера, 'Не має в цей час хворих' AS Коментар

FROM Тренер

LEFT JOIN СпортсменКоманда ON Тренер.IDТренера = СпортсменКоманда.IDТренера

LEFT JOIN Травма ON СпортсменКоманда.IDСпортсмен = Травма.IDСпортсмена

WHERE Травма.IDТравми IS NULL;

# Відновлення даних, умова відбору формується з використанням підзапита з іншої таблиці.

Додаткові відомості про працівника – “заключено найбільшу кількість договорів”

UPDATE Тренер

SET ДодатковіВідомості = “Навчив найбільшу кількість команд”

WHERE IDТренера = (SELECT IDТренера

FROM (SELECT Тренер.IDТренера, COUNT(\*) AS Кількість\_пацієнтів

FROM Тренер

JOIN СпортсменКоманда ON Тренер.IDТренера = СпортсменКоманда.IDТренера

JOIN Травма ON СпортсменКоманда.IDСпортсмен = Травма.IDСпортсмена

GROUP BY Тренер.IDТренера) AS T

ORDER BY Кількість\_пацієнтів DESC

LIMIT 1);

# 4 Користувачі системи

# Опис груп користувачів

У ході аналізу предметної області було виділено наступні групи користувачів: працівник, адміністратор та гість. Працівник повинен мати права на таблиці, пов’язані з його роботою. Адміністратор має доступ до всіх таблиць, включно з даними про логіни та дії на сервері. Гість може переглядати лише основні таблиці предметної області.

# Засоби керування доступом

Керування доступом здійснюється через таблицю користувачів. Вона зберігає юзернейми, паролі та рівень доступу користувачів, які зареєстровані у системі. Переглядати та видаляти користувачів може лише адміністратор.

# Функціональні можливості

У таблиці 4.3.1 наведено права користувачів на таблиці бази даних.

Таблиця 4.3.1 – Права користувачів на таблиці бази даних

|  |  |  |
| --- | --- | --- |
| **Назва таблиці** | **Право на перегляд** | **Право на зміни** |
| Користувачі | Адміністратор | |
| Журнал дій | Адміністратор | |
| Змагання | Адміністратор, працівник, гість | Адміністратор, працівник |
| Матч | Адміністратор, працівник, гість | Адміністратор, працівник |
| Гол | Адміністратор, працівник, гість | Адміністратор, працівник |
| Команда | Адміністратор, працівник, гість | Адміністратор, працівник |
| Тренер | Адміністратор, працівник, гість | Адміністратор, працівник |
| Спортсмен | Адміністратор, працівник, гість | Адміністратор, працівник |

Продовження таблиці 4.3.1

|  |  |  |
| --- | --- | --- |
| **Назва таблиці** | **Право на перегляд** | **Право на зміни** |
| Трансфер | Адміністратор, працівник, гість | Адміністратор, працівник |
| Травма | Адміністратор, працівник, гість | Адміністратор, працівник |
| Тренування | Адміністратор, працівник, гість | Адміністратор, працівник |

# 5 Інтерфейс інформаційної системи

# Опис інтерфейсу

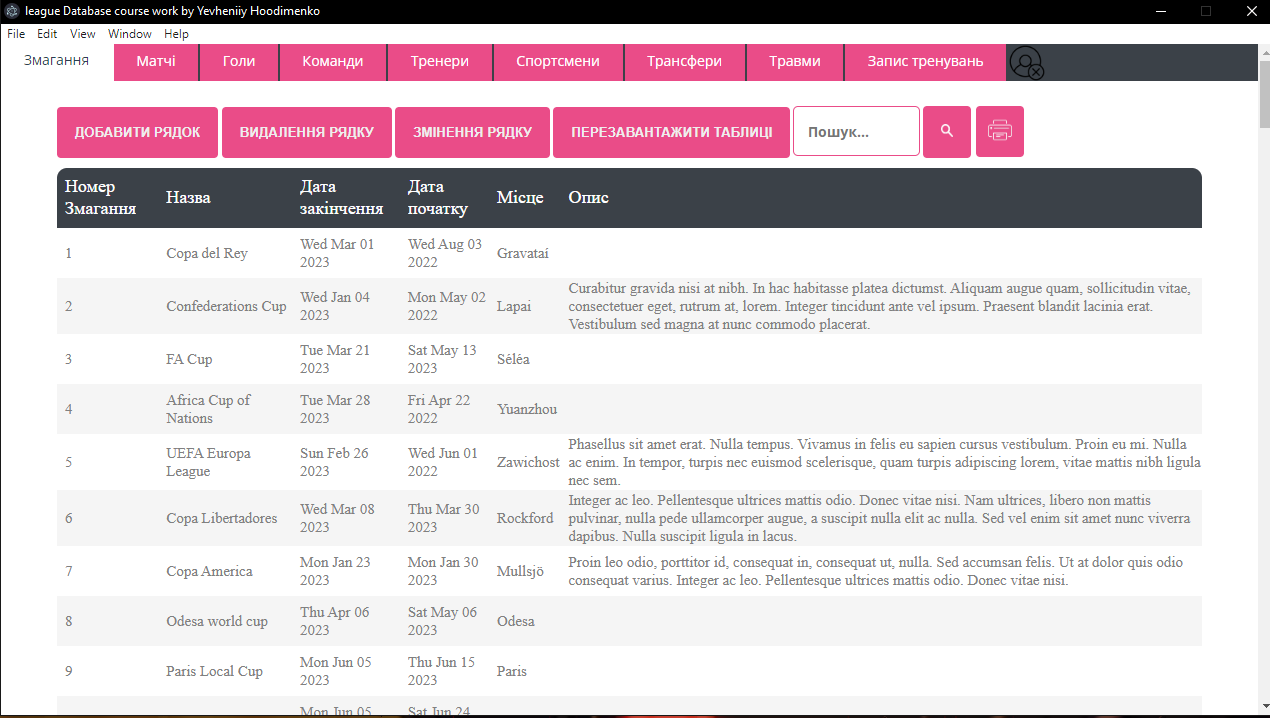
Інтерфейс має біло-сіро-рожеве забарвлення, маніпуляції проводяться за допомогою кнопок, полей вводу, випадаючих списків та чек-боксів. 

Рисунок 5.1.1 – Інтерфейс

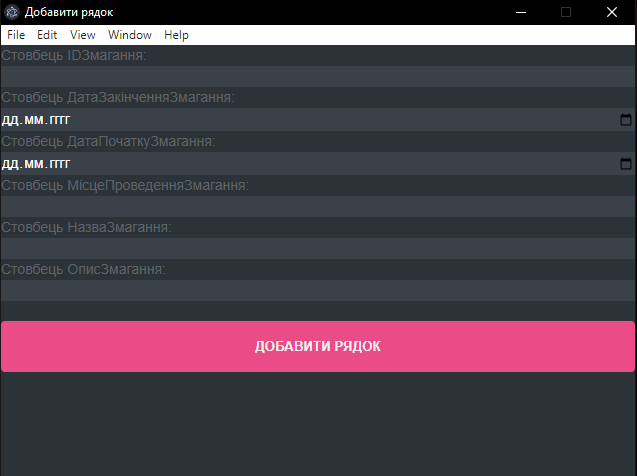
Керування інформацією виконується за допомогою полей вводу та випадаючих списків, кнопок для збереження інформації на нових сторінках, що відкриваються після натискання відповідних кнопок. 

Рисунок 5.1.2 – Заповнення інформації

Інформація про об’єкти розміщено у таблицях. У певних таблицях можливе сортування за кожною колонкою за допомогою відповідного натискання на неї, та повторного натискання, щоб сортувати в оберненому порядку.

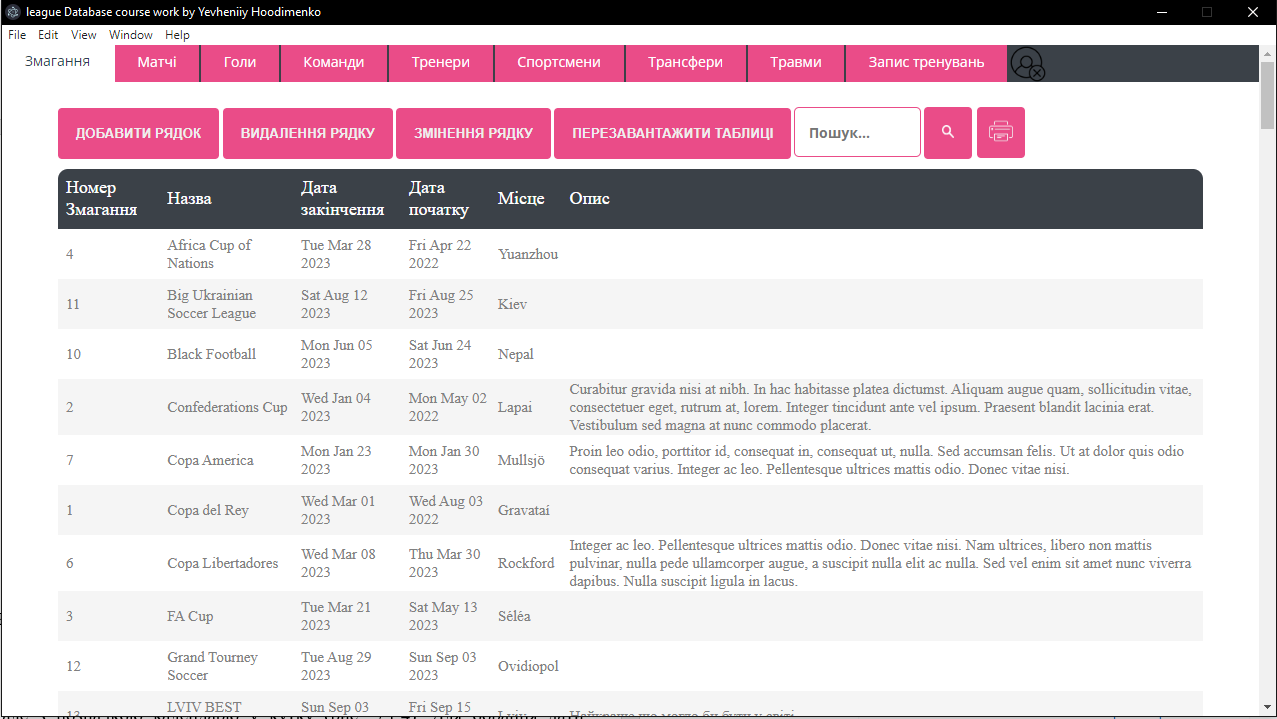


Рисунок 5.1.3 – Сортування в алфавітному порядку за стовпцем Назва

Аби вибрати дату, використовується елемент input типу date. Він схожий на поле для вводу, але з позначкою календарю у кутку (рис. 5.1.4). Для обрання дати натисніть на календар та оберіть дату.

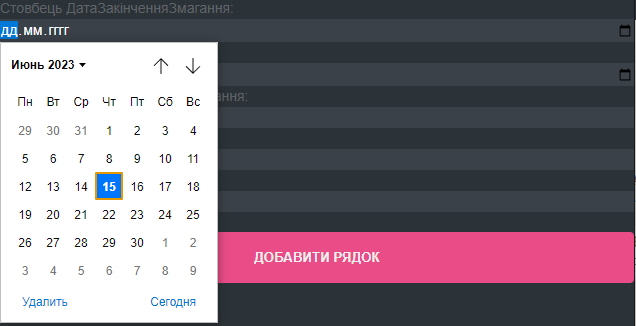


Рисунок 5.1.4 – Input типу Date

# Керівництво аккаунтами та модифікатори доступу

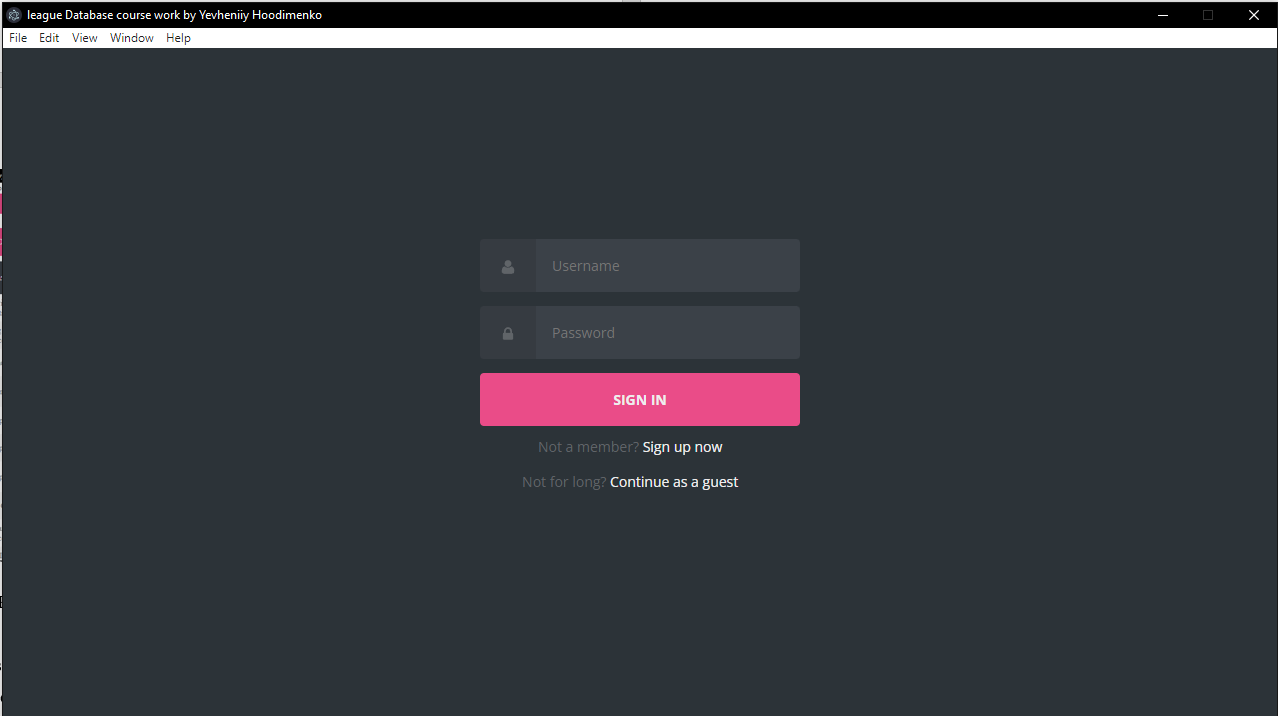
Після запуску серверу, ми можемо запустити застосунок що написаний на NodeJS Electron. Першим чином ми повинні пройти авторизацію, зареєструватись або зайти у якості гістя.

Рисунок 5.2.1 – Форма авторизації

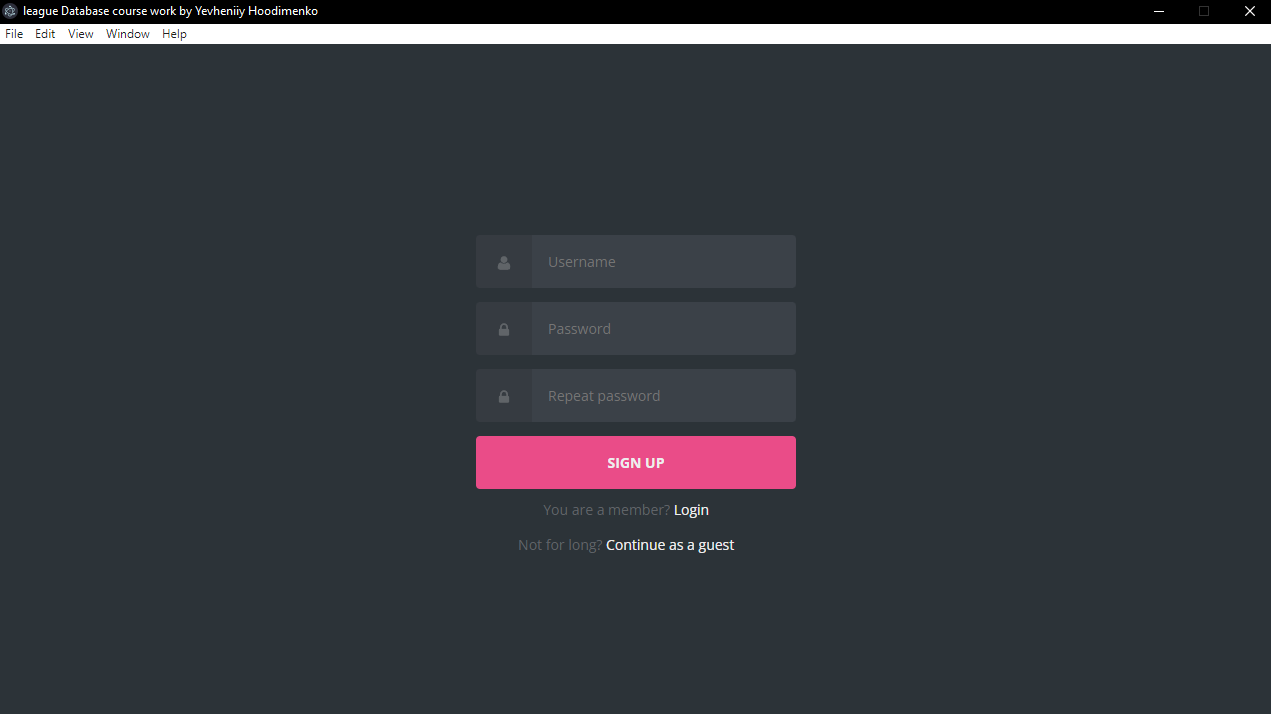
Для того, аби зареєструвати нового користувача, потрібно вибрати відповідне посилання на реєстраційну форму на сторінці логіну.

Рисунок 5.2.2 – Форма реєстрації нового користувача

Для того, щоб змінити дані користувача, або переглянути дії на сервері, повинен авторизуватися адмін з відповідними даними для входу

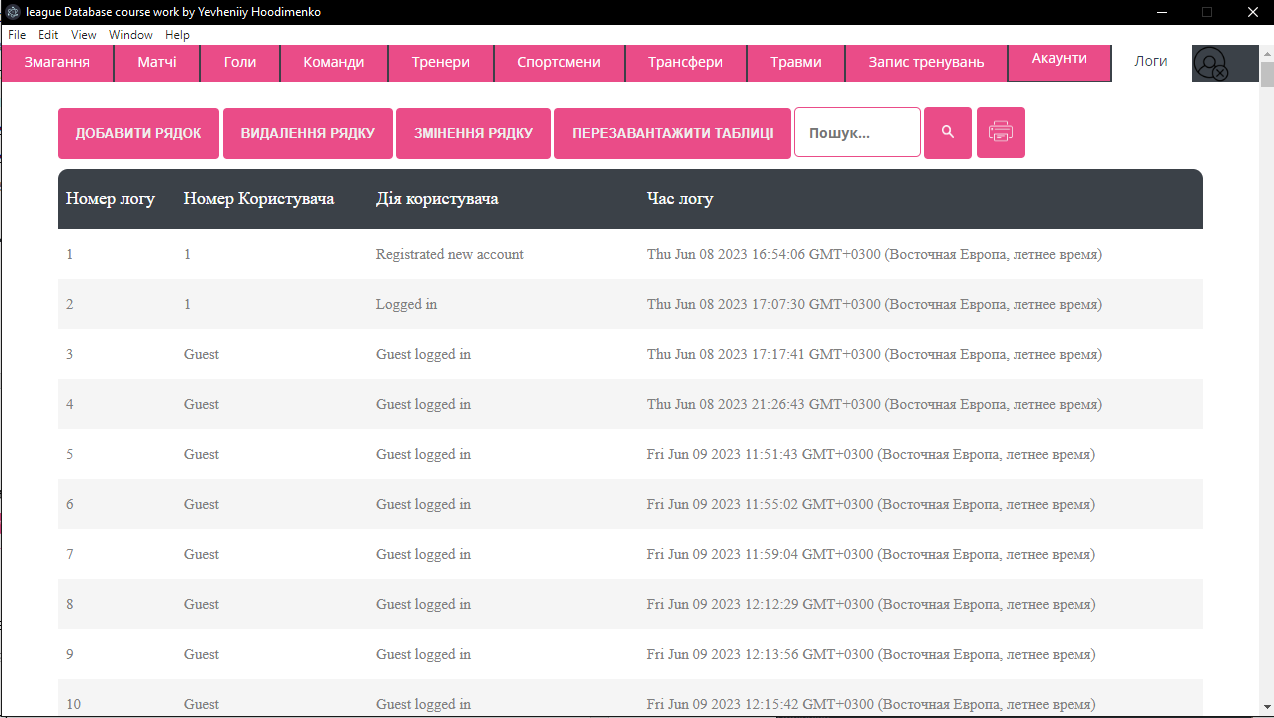


Рисунок 5.2.3 – Таблиці, що доступні лише адміну

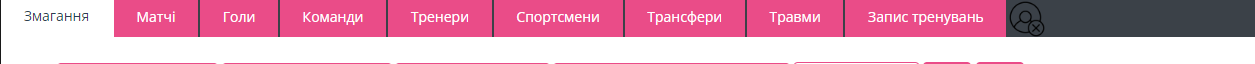
Для навігації по таблицям використовуються вкладки зверху у застосунку.

Рисунок 5.2.4 – Навігація по таблицямм

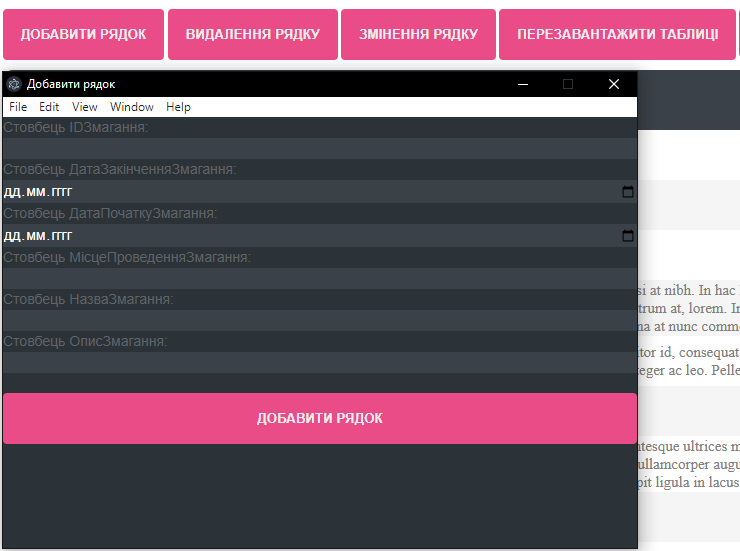
Для додавання запису натискаємо відповідну кнопку та уводимо необхідні дані та/або обираємо їх з відповідних випадаючих списків у новому вікні та натискаємо кнопку “Добавити рядок”. Запис одразу ж додається у таблицю (рис. 5.2.4 – рис. 5.2.5)

Рисунок 5.2.5 – Введення даних

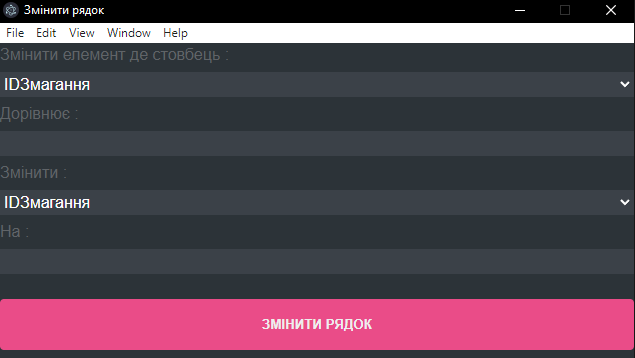
Для редагування даних навпроти рядку, що необхідно відредагувати, проводимо аналогічні дії після натискання кнопки “Змінення рядку”. Поля заповнюються відповідними даними.

Рисунок 5.2.6 – Редагування запису

Після цього натискаємо на “Змінити рядок” навпроти рядку, аби зберегти зміни.

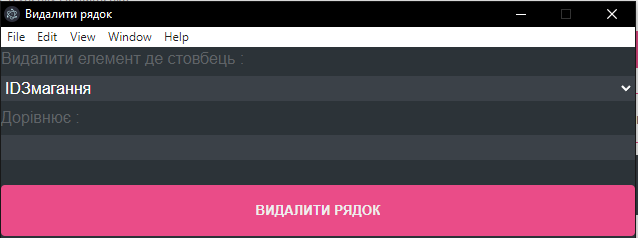
Для видалення запису достатньо лише натиснути “вид” навпроти рядку, що необхідно видалити.

Рисунок 5.2.7 – Запис видалено

Для пошуку можна скористатись полем для пошуку, що розташовано над таблицею:

Рисунок 5.2.8 – Пошук по таблиці

Уведемо необхідні дані для пошуку та натиснемо “знайти”.

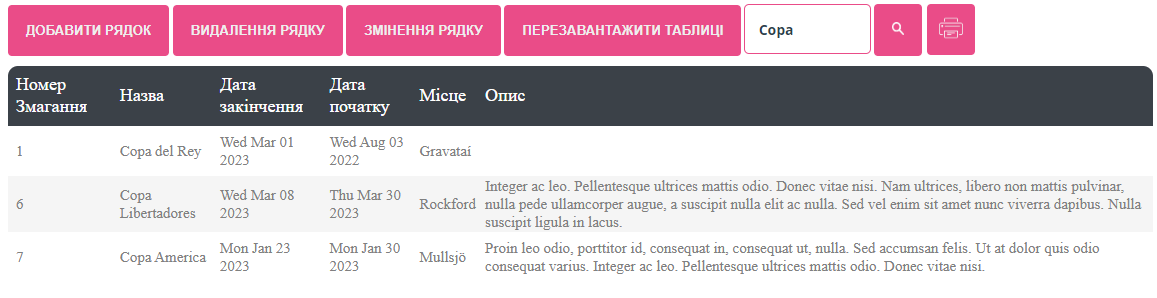


Рисунок 5.2.9 – Результат фільтрування

# Контрольний приклад

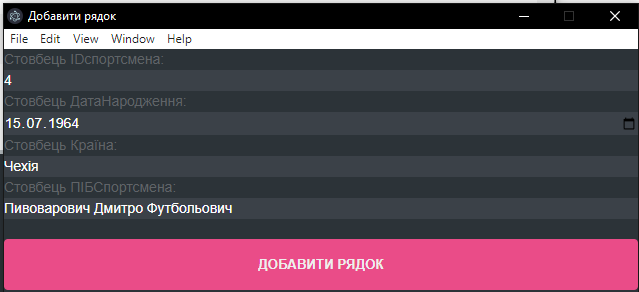
Спробуємо додати запис до таблиці спортсменів. Відкриємо вкладку за допомогою кнопки та додамо нові данні.

Рисунок 5.3.1 – Внесення зв’язаних даних

У коді передбачені поведомлення про всі можливі розвитки роботи програми та повідомлення користувача про них – у цьому разі все пройшло успішно.

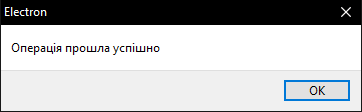


Рисунок 5.3.2 – Повідомлення про статус операції

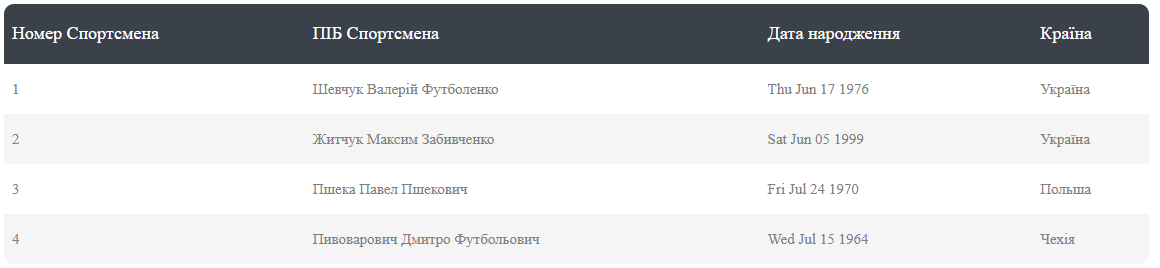
Операція успішна – рядок добавлений

Рисунок 5.3.3 – Результат операції

Тестуємо видалення зв’язаних даних. Для цього спробуємо видалити команду номер 8 з таблиці команд:

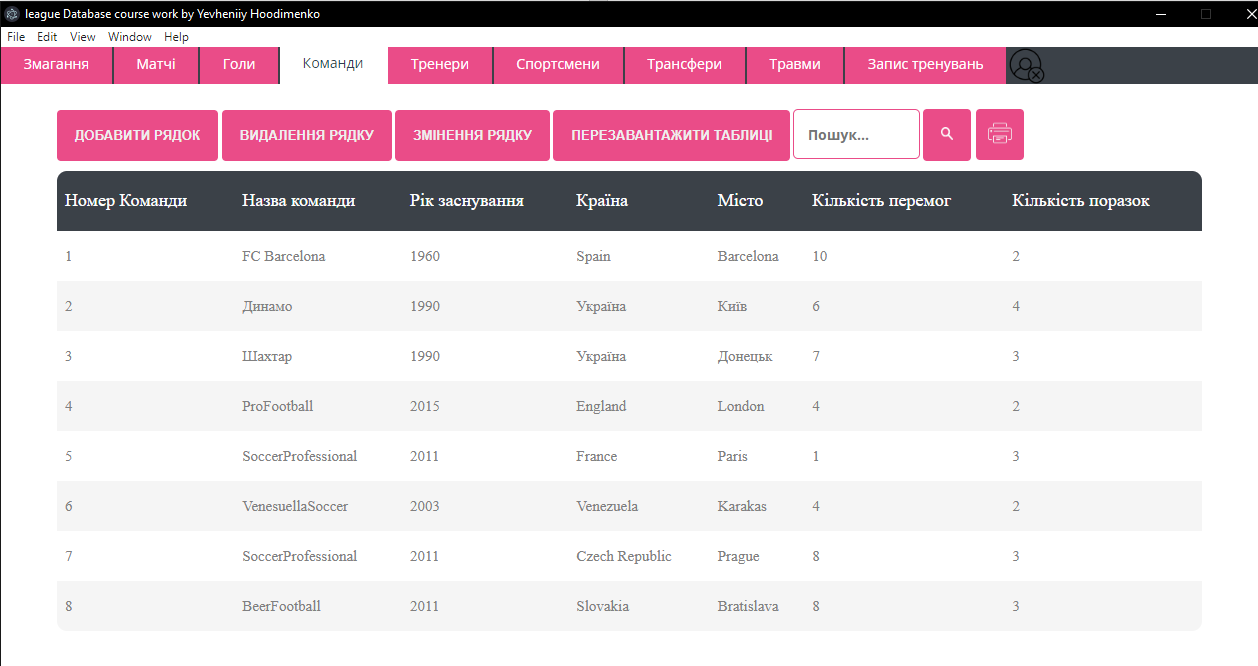


Рис. 5.4.3 – Таблиця до видалення

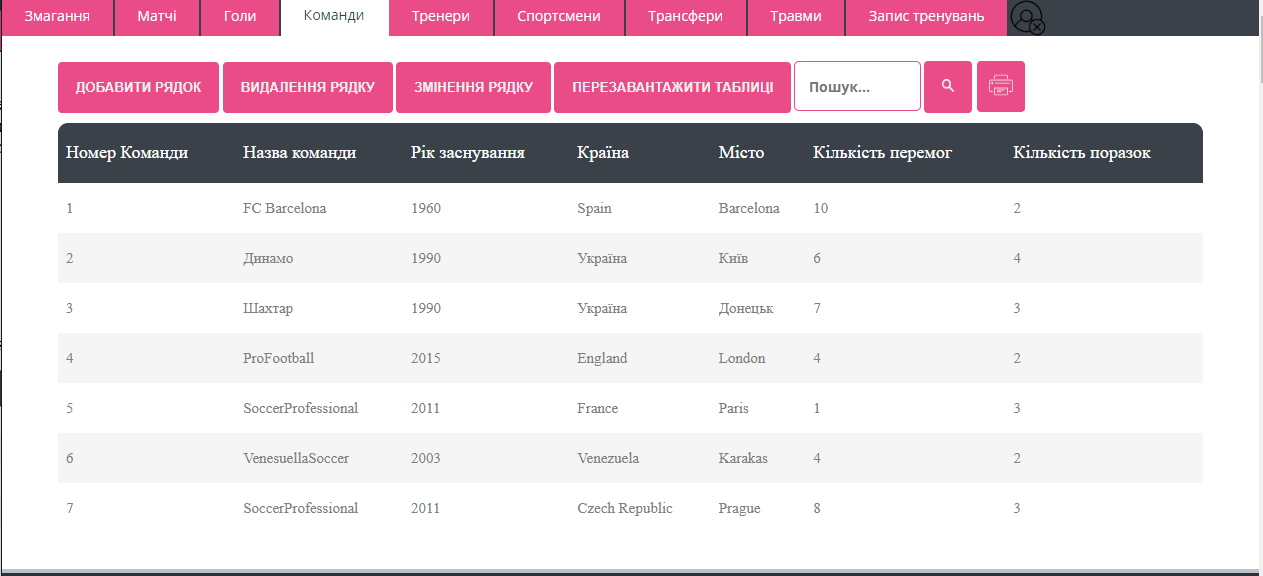


Рисунок 5.4.4 – Оновлена таблиця команд

ВИСНОВКИ

Результатом виконання курсової роботи є розроблена функціонуюча система з базою даних. У ході виконання я ознайомився з методами аналізу предметної області та інструментами керування реляційними базами даних.

У розділі “Аналіз предметної області” приведено сформовані правила предметної області, опис вимог до системи, опис об’єктів та їх атрибутів.

У розділі “Проєктування бази даних” наведено етапи аналізу структури даних: аномалії, їх нормалізація, виявлення первинних ключів, концептуальна та реляційні моделі даних.

У розділі “Запити” описано основні запити до бази даних.

У розділі “Користувачі системи” описані користувачі системи та їх права.

У розділі “Інтерфейс інформаційної системи” наведено опис інтерфейсу, керівництво користувача та тестування функціонування програми.

Було використано 3 таблиці та 20 рисунків.

СПИСОК ЛІТЕРАТУРИ

1. Методичні вказівки до курсової роботи з дисципліни «Бази даних» для студентів спеціальності 121 – «Інженерія програмного забезпечення» / Укл. С.Л. Зіноватна.

- Одеса; ОНПУ, 2020. - 19 с.

1. Конспект лекцій з дисципліни «Бази даних» для студентів спеціальності 121 – Інженерія програмного забезпечення / Укл. С. Л. Зіноватна, – Одеса: ОНПУ, 2019.

- 70 с.

1. Берко, А. Ю. Системи баз даних та знань. Кн. 1. Організація баз даних та знань : навч. посібник / А.Ю. Берко, О.М. Верес, В.В. Пасічник; за заг. ред. В.В. Пасічника .- Л. : Магнолія 2006, 2008.- 456 с.

ДОДАТОК А

У додатку А наведено фрагмент коду що відповідає за скріпт відрендереного вікна, більшість його коду відповідає за за керування базами даних.

const { app } = require('electron/main');

const mysql = require('mysql');

const { BrowserWindow, remote, ipcRenderer } = require('electron');

var user\_id = "NULL";

var extrawindow;

var con;

var tablerows = 0;

var tempsqlargs = '';

var tempsqlstring = '';

let headtblvals = [

["змагання", ["Номер Змагання", "Назва", "Дата закінчення", "Дата початку", "Місце", "Опис"]],

["матч", ["Номер Матчу", "Номер Змагання", "Дата проведення", "Час проведення", "Номер туру", "Результат матчу"]],

["гол", ["Номер Голу", "Номер Матчу", "Хвилина гри", "Опис гола", "Номер команди гравця"]],

["команда", ["Номер Команди", "Назва команди", "Рік заснування", "Країна", "Місто", "Кількість перемог", "Кількість поразок"]],

["тренер", ["Номер Тренера", "ПІБ Тренера", "Країна", "Кількість перемог", "Кількість поразок"]],

["спортсмен", ["Номер Спортсмена", "ПІБ Спортсмена", "Дата народження", "Країна"]],

["трансфер", ["Номер Трансферу", "Номер попередньої команди", "Номер теперішьої команди", "Номер гравця"]],

["травма", ["Номер травми", "Номер спортсмена", "Опис травми", "Дата отримання травми"]],

["тренеркоманда", ["Номер зборів", "Номер тренера", "Номер команди", "Дата початку навчання команди", "Дата кінця навчання команди"]],

["user\_list", ["Номер користувача", "Логін", "Пароль", "Модифікатор доступу"]],

["user\_actions", ["Номер логу", "Номер Користувача", "Дія користувача", "Час логу"]],

]

let orderBy = 1;

asc\_desc = "asc"

var canBeReturned = []

const connectToDbPromise = new Promise((resolve, reject) => {

con = mysql.createConnection({

host: 'localhost',

user: 'root',

password: "password",

database: 'футбольналіга'

});

con.connect(function (err) {

if (err) throw err;

resolve()

console.log("Connected to db!");

});

});

async function checkMainFile() {

await connectToDbPromise;

var params = new URLSearchParams(window.location.search);

var table\_name = params.get("table\_name");

var typeOfAction = params.get("typeOfAction");

if (document.getElementById('add\_row\_container')) {

try {

con.query(

`SELECT COLUMN\_NAME

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = '${table\_name}';

`,

function (err, result, fields) {

if (err) {

throw err

}

let button = document.createElement("button");

let container = document.getElementById("add\_row\_container");

tablerows = result.length;

for (let i = 0; i < tablerows; i++) {

let label = document.createElement("label");

let input = document.createElement("input");

let br = document.createElement("br");

label.setAttribute("for", `myInput${i}`);

label.textContent = `Стовбець ${result[i].COLUMN\_NAME}: `;

if (i > 0) {

tempsqlargs += ","

}

tempsqlargs += result[i].COLUMN\_NAME

let inputType = "text"

if (result[i].COLUMN\_NAME.includes("Дата")) {

inputType = "date"

}

input.setAttribute("type", inputType);

input.setAttribute("id", `myInput${i}`);

container.appendChild(label);

container.appendChild(input);

container.appendChild(br);

}

button.innerText = "Добавити рядок"

button.setAttribute("onClick", `extrawin\_submit( '${typeOfAction}')`)

container.appendChild(button);

}

);

} catch (error) {

console.error('Помилка пошуку таблиці для добавлення:', error);

}

} else if (document.getElementById('del\_row\_container')) {

try {

con.query(

`SELECT COLUMN\_NAME

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = '${table\_name}';

`,

function (err, result, fields) {

if (err) {

throw err

}

let button = document.createElement("button");

let container = document.getElementById("del\_row\_container");

let label = document.createElement("label");

label.setAttribute(`for`, `${table\_name}`);

label.textContent = `Видалити елемент де стовбець : `;

container.appendChild(label);

let select = document.createElement("select")

select.setAttribute("name", table\_name)

select.setAttribute("id", table\_name)

select.setAttribute("onchange", "extrawin\_onChange()")

container.appendChild(select);

tablerows = result.length;

for (let i = 0; i < tablerows; i++) {

let option = document.createElement("option")

option.setAttribute("value", result[i].COLUMN\_NAME)

option.innerText = result[i].COLUMN\_NAME

select.appendChild(option);

}

label = document.createElement("label");

label.setAttribute(`for`, `deleteVal`);

label.textContent = `Дорівнює : `;

container.appendChild(label);

input = document.createElement("input")

input.setAttribute("type", "text");

input.setAttribute("id", `deleteVal`);

container.appendChild(input)

button.innerText = "Видалити рядок"

button.setAttribute("onClick", `extrawin\_submit( '${typeOfAction}')`)

container.appendChild(button);

}

);

} catch (error) {

console.error('Помилка пошуку таблиці для видалення:', error);

}

} else if (document.getElementById('upd\_row\_container')) {

try {

con.query(

`SELECT COLUMN\_NAME

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = '${table\_name}';

`,

function (err, result, fields) {

if (err) {

throw err

}

let button = document.createElement("button");

let container = document.getElementById("upd\_row\_container");

let label = document.createElement("label");

label.setAttribute(`for`, `${table\_name}`);

label.textContent = `Змінити елемент де стовбець : `;

container.appendChild(label);

let select = document.createElement("select")

select.setAttribute("name", table\_name)

select.setAttribute("id", table\_name)

select.setAttribute("onchange", `extrawin\_onChange("del")`)

container.appendChild(select);

tablerows = result.length;

for (let i = 0; i < tablerows; i++) {

let option = document.createElement("option")

option.setAttribute("value", result[i].COLUMN\_NAME)

option.innerText = result[i].COLUMN\_NAME

select.appendChild(option);

}

label = document.createElement("label");

label.setAttribute(`for`, `deleteVal`);

label.textContent = `Дорівнює : `;

container.appendChild(label);

input = document.createElement("input")

input.setAttribute("type", "text");

input.setAttribute("id", `deleteVal`);

container.appendChild(input)

label = document.createElement("label");

label.setAttribute(`for`, `upd\_whatVal`);

label.textContent = `Змінити : `;

container.appendChild(label);

select = document.createElement("select")

select.setAttribute("name", `upd\_whatVal`)

select.setAttribute("id", `upd\_whatVal`)

select.setAttribute("onchange", `extrawin\_onChange("upd")`)

container.appendChild(select);

for (let i = 0; i < tablerows; i++) {

let option = document.createElement("option")

option.setAttribute("value", result[i].COLUMN\_NAME)

option.innerText = result[i].COLUMN\_NAME

select.appendChild(option);

}

label = document.createElement("label");

label.setAttribute(`for`, `updVal`);

label.textContent = `На : `;

container.appendChild(label);

input = document.createElement("input")

input.setAttribute("type", "text");

input.setAttribute("id", `updVal`);

container.appendChild(input)

button.innerText = "Змінити рядок"

button.setAttribute("onClick", `extrawin\_submit( '${typeOfAction}')`)

container.appendChild(button);

}

);

} catch (error) {

console.error('Помилка пошуку таблиці для зміни:', error);

}

}

}

function toRegistration() {

document.getElementById("login\_\_username").value = ""

document.getElementById("login\_\_password").value = ""

document.getElementById("login\_form").style.display = "none";

document.getElementById("registration\_form").style.display = "block";

}

function toLogin() {

document.getElementById("login\_\_username").value = ""

document.getElementById("login\_\_password").value = ""

document.getElementById("login\_form").style.display = "block";

document.getElementById("registration\_form").style.display = "none";

}

function logOut() {

document.getElementById("main\_app").style.display = "none";

toLogin()

user\_id = "NULL";

}

function login() {

let username = document.getElementsByClassName("logLogin")[0].value

let password = document.getElementsByClassName("logPass")[0].value

con.query("SELECT \* FROM user\_list", function (err, result, fields) {

if (err) throw err;

for (let i = 0; i < result.length; i++) {

if (result[i].user\_login == username &&

result[i].user\_password == password) {

console.log("Вхід успішний")

user\_id = result[i].user\_id

dataLogUserActions(user\_id, 'Logged in')

app\_connect(true);

return

}

}

alert("Данні для входу невірні")

});

}

async function registrate() {

let username = document.getElementsByClassName("regLogin")[0].value;

let password = document.getElementsByClassName("regPass")[0].value;

let password2 = document.getElementsByClassName("repeatPassword")[0].value;

if (password === password2) {

try {

const findUserPromise = new Promise((resolve, reject) => {

con.query(

`SELECT \* FROM user\_list WHERE user\_login = '${username}'`,

function (err, result, fields) {

if (err) {

reject(err);

return;

}

if (result.length > 0) {

alert("Користувач з таким логіном вже є")

reject(new Error('Пользователь уже существует'));

} else {

resolve();

}

}

);

});

await findUserPromise;

const insertUserPromise = new Promise((resolve, reject) => {

con.query(

`INSERT INTO user\_list (user\_login, user\_password, user\_restrictions)

VALUES ('${username}', '${password}', 1)`,

function (err, result, fields) {

if (err) {

reject(err);

return;

}

console.log('Регистрация прошла успешно');

resolve(result.insertId);

}

);

});

userId = await insertUserPromise;

con.query(

`INSERT INTO user\_actions (user\_id, user\_action)

VALUES ('${userId}', 'Registrated new account')`,

function (err, result, fields) {

if (err) throw err;

console.log('Запись в лог успешно добавлена');

app\_connect(true);

}

);

} catch (error) {

console.error('Ошибка регистрации:', error);

}

}

}

function asAGuest() {

try {

dataLogUserActions("NULL", 'Guest logged in')

user\_id = "NULL";

app\_connect(false)

}

catch (error) {

console.error('Ошибка входа:', error);

}

}

//await waitOneSecond();

function waitOneSecond() {

return new Promise((resolve) => {

setTimeout(() => {

resolve();

}, 1000);

});

}

async function app\_connect(isLogged) {

document.getElementById("login\_form").style.display = "none";

document.getElementById("registration\_form").style.display = "none";

document.getElementById("main\_app").style.display = "block";

let getUserStatus = new Promise((resolve, reject) => {

con.query("SELECT \* FROM user\_list", function (err, result, fields) {

if (err) {

throw err

resolve(false)

}

if (isLogged) {

for (i = 0; i < document.getElementsByClassName("delButton").length; i++) {

document.getElementsByClassName("addButton")[i].disabled = false

document.getElementsByClassName("delButton")[i].disabled = false

document.getElementsByClassName("updButton")[i].disabled = false

}

for (let i = 0; i < result.length; i++) {

if (user\_id == result[i].user\_id) {

if (result[i].user\_restrictions != 2) {

canBeReturned.push(document.getElementById("tab-10").parentNode)

canBeReturned.push(document.getElementById("tab-11").parentNode)

if (document.getElementById("tab-10").parentNode) {

document.getElementById("tab-10").parentNode.remove()

document.getElementById("tab-11").parentNode.remove()

}

} else {

if (canBeReturned.length) {

document.getElementsByClassName("tabs")[0].appendChild(canBeReturned[0])

document.getElementsByClassName("tabs")[0].appendChild(canBeReturned[1])

canBeReturned = []

}

}

break

}

}

} else {

canBeReturned.push(document.getElementById("tab-10").parentNode)

canBeReturned.push(document.getElementById("tab-11").parentNode)

if (document.getElementById("tab-10").parentNode) {

document.getElementById("tab-10").parentNode.remove()

document.getElementById("tab-11").parentNode.remove()

}

for (i = 0; i < document.getElementsByClassName("delButton").length; i++) {

document.getElementsByClassName("addButton")[i].disabled = true

document.getElementsByClassName("delButton")[i].disabled = true

document.getElementsByClassName("updButton")[i].disabled = true

}

}

resolve(true)

});

})

if (await getUserStatus) {

reloadTables()

}

for (let i = 1; document.getElementById("tab-" + i) != undefined; i++) {

let p = document.getElementById("tab-" + i);

p.onclick = function (event) { defaultOrder() };

let search = document.getElementsByClassName("searchInput")[i-1]

search.setAttribute("id", "searchInput" + i)

}

}

function dataLogUserActions(id\_of\_user, action\_text) {

try {

con.query(

`INSERT INTO user\_actions (user\_id, user\_action)

VALUES (${id\_of\_user}, '${action\_text}')`,

function (err, result, fields) {

if (err) throw err;

console.log('Запись в лог успешно добавлена');

}

);

}

catch (error) {

console.error('Ошибка лога:', error);

}

}

function rowAction(table\_name, typeOfAction) {

extrawindow = window.open(`${typeOfAction}row.html?table\_name=` + encodeURIComponent(table\_name)

+ '&typeOfAction=' + encodeURIComponent(typeOfAction)

+ '&user\_id=' + encodeURIComponent(user\_id)

);

}

async function extrawin\_submit(typeOfAction) {

var params = new URLSearchParams(window.location.search);

var table\_name = params.get("table\_name");

var typeOfAction = params.get("typeOfAction");

var user\_id = params.get("user\_id")

var action;

switch (typeOfAction) {

case "add":

action = new Promise((resolve, reject) => {

try {

for (let i = 0; i < tablerows; i++) {

if (i > 0) {

tempsqlstring += ','

}

let temp = document.getElementById(`myInput${i}`).value

tempsqlstring += goToSqlString(temp)

}

con.query(

`INSERT INTO ${table\_name} (${tempsqlargs})

VALUES (${tempsqlstring});

`,

function (err, result, fields) {

if (err) {

throw err;

resolve(false);

}

resolve(true);

}

);

} catch (error) {

resolve(false);

console.log(error)

}

});

break;

case "del":

action = new Promise((resolve, reject) => {

try {

where = document.getElementById(`${table\_name}`).value

what = document.getElementById(`deleteVal`).value

con.query(

`DELETE FROM ${table\_name} WHERE ${where} = ${goToSqlString(what)} LIMIT 1`,

function (err, result, fields) {

if (err) {

resolve(false);

}

resolve(true);

}

);

} catch (error) {

resolve(false);

}

});

break;

case "upd":

action = new Promise((resolve, reject) => {

try {

where = document.getElementById(`${table\_name}`).value

what = document.getElementById(`deleteVal`).value

fromWhere = document.getElementById("upd\_whatVal").value

toWhat = document.getElementById(`updVal`).value

con.query(

`UPDATE ${table\_name} SET ${fromWhere} = ${goToSqlString(toWhat)} WHERE ${where} = ${goToSqlString(what)} LIMIT 1`,

function (err, result, fields) {

if (err) {

resolve(false);

}

resolve(true);

}

);

} catch (error) {

resolve(false);

}

});

break;

}

if (await action) {

alert("Операція прошла успішно")

dataLogUserActions(user\_id, `Змінення таблиці ${table\_name} : ${typeOfAction}`)

window.close()

} else {

console.error('Помилка при зміненні таблиці:', error);

alert("При виконанні операції сталася помилка")

dataLogUserActions(user\_id, `Невдала спроба змінення таблиці ${table\_name} : ${typeOfAction}`)

window.close()

}

}

function goToSqlString(str) {

if (/^\d+$/.test(str)) {

return str;

} else {

return "'" + str + "'";

}

}

async function extrawin\_onChange(typeAction) {

var params = new URLSearchParams(window.location.search);

var table\_name = params.get("table\_name");

if (typeAction) {

let inputType = "text"

if (document.getElementById(table\_name).value.includes("Дата")) {

inputType = "date"

}

input = document.getElementById("deleteVal")

input.setAttribute("value", "");

input.setAttribute("type", inputType);

} else {

let inputType = "text"

if (document.getElementById(table\_name).value.includes("Дата")) {

inputType = "date"

}

input = document.getElementById("updVal")

input.setAttribute("value", "");

input.setAttribute("type", inputType);

}

}

function reloadTables() {

try {

// Запрос на отримання списку таблиць з бд

const getTables = new Promise((resolve, reject) => {

con.query(

`SELECT table\_name

FROM INFORMATION\_SCHEMA.tables

WHERE table\_schema = "футбольналіга"`,

function (err, result, fields) {

if (err) {

reject(err);

} else {

resolve(result);

}

}

);

});

getTables

.then((tables) => {

// Обработка списку таблиць

const promises = tables.map((table) => {

const tableName = table.TABLE\_NAME;

const table\_head = document.getElementById("table\_head-" + tableName);

if (table\_head) {

let tr\_head = document.getElementById("tr-" + tableName);

tr\_head.innerHTML = '';

let table\_body = document.getElementById("table\_body-" + tableName);

table\_body.innerHTML = '';

// Запрос на отримання списку стовбців

const getColumns = new Promise((resolve, reject) => {

con.query(

`SELECT COLUMN\_NAME

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = '${tableName}'`,

function (err, result, fields) {

if (err) {

reject(err);

} else {

resolve(result);

}

}

);

});

return getColumns.then((columns) => {

let pos;

for (let p = 0; p < headtblvals.length; p++) {

if (tableName == headtblvals[p][0]) {

pos = p;

break;

}

}

for (let j = 0; j < columns.length; j++) {

let th = document.createElement("th");

th.setAttribute("class", "column" + j);

th.setAttribute("onclick", "placeOrderBy(" + j +")");

if (headtblvals[pos][1][j] != undefined) {

tr\_head.appendChild(th);

th.innerHTML = headtblvals[pos][1][j];

}

}

// Запрос отримання даних з таблиці

const getTableData = new Promise((resolve, reject) => {

con.query(

`SELECT \* FROM ${tableName} order by ${orderBy} ${asc\_desc}`,

function (err, result, fields) {

if (err) {

reject(err);

} else {

resolve(result);

}

}

);

});

return getTableData.then((data) => {

for (let k = 0; k < data.length; k++) {

let tr = document.createElement("tr");

table\_body.appendChild(tr);

rowData = Object.values(data[k]);

for (let h = 0; h < rowData.length; h++) {

let td = document.createElement("td");

td.setAttribute("class", "column" + h);

tr.appendChild(td);

rowString = rowData[h] + " "

if (rowString.includes("GMT") && tableName != "user\_actions"){

rowData[h] = rowString.substring(0,15)

}

if (tableName == "user\_actions") {

if (rowData[h] == null) {

rowData[h] = "Guest"

}

}

td.innerHTML = rowData[h];

// Допрацювати це гівно

}

}

});

});

} else {

return Promise.resolve();

}

});

return Promise.all(promises);

})

.catch((error) => {

console.error('Помилка завантаження таблиць:', error);

});

} catch (error) {

console.error('Помилка завантаження таблиць:', error);

}

}

function placeOrderBy(pos) {

if (orderBy == pos + 1) {

if (asc\_desc == "asc") {

asc\_desc = "desc"

} else {

asc\_desc = "asc"

}

} else {

orderBy = pos + 1

}

}

function defaultOrder() {

orderBy = 1;

asc\_desc = "asc"

}

function searchBy(temp\_table\_name) {

var table\_name\_like = [ ["змагання", "НазваЗмагання"],

["матч", "РезультатМатчу"],

["гол", "ОписГола"],

["команда", "НазваКоманди"],

["тренер", "ПІБТренера",],

["спортсмен", "ПІБСпортсмена",],

["трансфер", "IDГравця"],

["травма", "ОписТравми"],

["тренеркоманда", "Номер команди"],

["user\_list", "user\_login"],

["user\_actions","user\_action"]

]

let pos;

let like;

for (var i = 0; i < headtblvals.length; i++) {

if (headtblvals[i][0] == temp\_table\_name) {

like = document.getElementById("searchInput" + (i + 1)).value

pos = i

}

}

const getTableData = new Promise((resolve, reject) => {

con.query(

`SELECT \* FROM ${temp\_table\_name} where ${table\_name\_like[pos][1]} like "%${like}%" order by ${orderBy} ${asc\_desc}`,

function (err, result, fields) {

if (err) {

reject(err);

} else {

resolve(result);

}

}

);

});

let table\_body = document.getElementById("table\_body-" + temp\_table\_name);

table\_body.innerHTML = '';

getTableData.then((data) => {

for (let k = 0; k < data.length; k++) {

let tr = document.createElement("tr");

table\_body.appendChild(tr);

rowData = Object.values(data[k]);

for (let h = 0; h < rowData.length; h++) {

let td = document.createElement("td");

td.setAttribute("class", "column" + h);

tr.appendChild(td);

rowString = rowData[h] + " "

if (rowString.includes("GMT") && temp\_table\_name != "user\_actions") {

rowData[h] = rowString.substring(0, 15)

}

if (temp\_table\_name == "user\_actions") {

if (rowData[h] == null) {

rowData[h] = "Guest"

}

}

td.innerHTML = rowData[h];

}

}

});

}

function printElement(temp\_table\_name) {

let elementToPrint

for (var i = 0; i < headtblvals.length; i++) {

if (headtblvals[i][0] == temp\_table\_name) {

elementToPrint = document.getElementsByClassName("print")[i];

}

}

if (elementToPrint) {

const printWindow = window.open('', '\_blank');

printWindow.document.write('<html><head><title>Print</title></head><body>');

printWindow.document.write(elementToPrint.innerHTML);

printWindow.document.write('</body></html>');

printWindow.document.close();

printWindow.print();

printWindow.close();

} else {

console.error('Элемент с ID ' + elementId + ' не найден.');

}

}

checkMainFile();