



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих
комп'ютерних систем

Лабораторна робота №2

з дисципліни
«Бази даних і засоби управління»

Виконав: студент 3 курсу

ФПМ групи КВ-83

Проценко Владислав

Перевірів: Павловський В.І.

Варіант 20

Загальне завдання роботи полягає у наступному:

1. Забезпечити можливість введення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв'язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати **вилучення** рядків за умови наявності даних у підлеглий таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні **внесення** нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.
2. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не мовою програмування, а відповідним SQL-запитом!**

Нормалізована база даних з першої лабораторної роботи

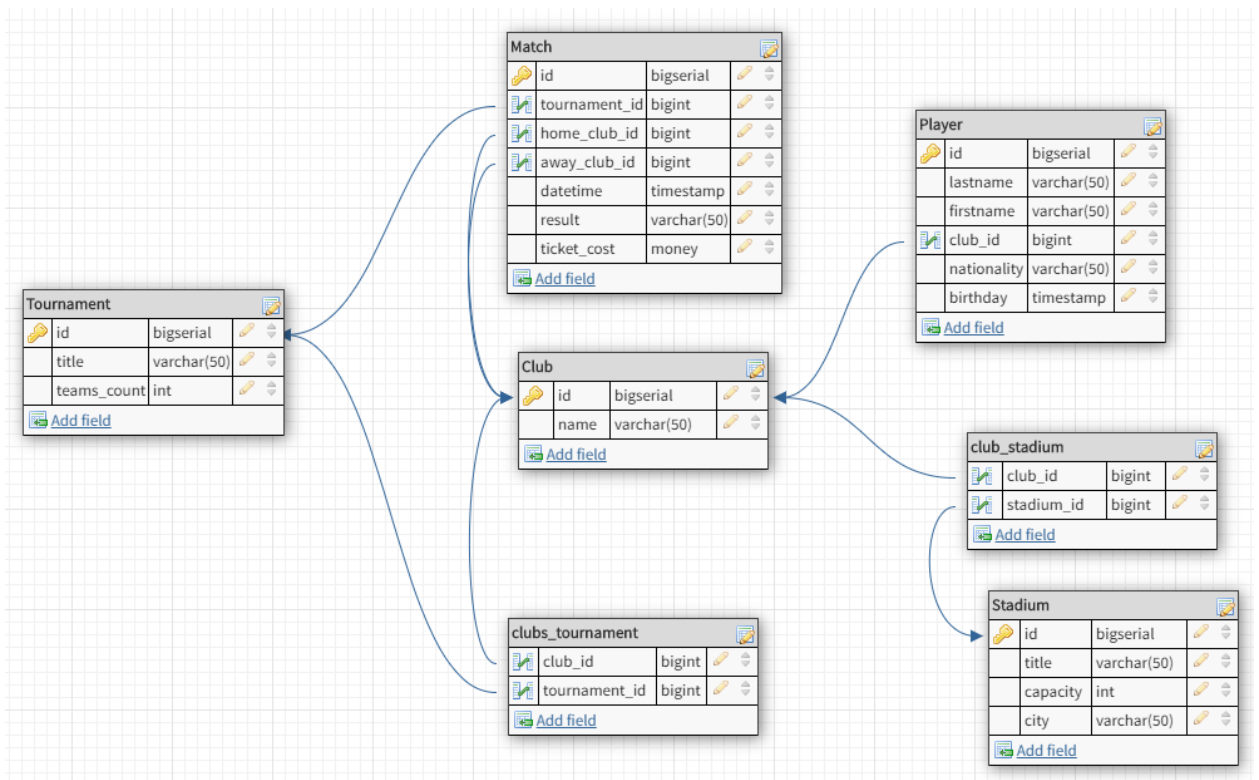


Рисунок 1 - Нормалізована база даних.

Опис програми

Структура програми

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
 - 1.1. [Запуск сервера.](#)
2. Файли роутингу
 - 2.1. [Для таблиці club;](#)
 - 2.2. [Для таблиці stadium;](#)
 - 2.3. [Для таблиці player;](#)
 - 2.4. [Для таблиці tournament;](#)
 - 2.5. [Для таблиці match;](#)
 - 2.6. [Для таблиці clubs_stadiums;](#)
 - 2.7. [Для таблиці clubs_tournamets.](#)
3. Модулі для роботи з таблицями.
 - 3.1. [Задання сутностей;](#)
 - 3.2. [Модуль, який відображає їх у браузері.](#)
4. Навігація користувача
 - 4.1. [Модуль навігації;](#)
 - 4.2. [Відображення навігації\(Меню\).](#)
5. [Модуль для підключення до БД.](#)

Обробка помилок

Для цього завдання кожен SQL запит було виконано в конструкторі try_catch, приклад:

```
router.get('/club', async(req, res) => {  
  try {  
    const clubs = await pool.query('SELECT * FROM club ORDER BY id;')  
    res.json(clubs.rows)  
  } catch (error) {  
    res.status(400).json('Error');  
    console.log(error.message)  
  }  
})
```

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 екран буде виведено повідомлення про помилку. Приклад:

```
//makes GET request to server and getting all rows of a table  
const getRows = async (tableName) => {  
  try {  
    const response = await fetch(  
      `${serverUrl}/${tableName}`, {  
        method: 'GET',  
      }  
    )  
  
    const jsonData = await response.json()  
    if(response.status === 400) {  
      window.alert(jsonData)  
      return  
    }  
    setRows(jsonData)  
  } catch (error) {  
    console.log(error.message);  
  }  
}
```

Коли, користувач отримує відповідь, яка має статус 400(тобто помилки), функція window.alert виведе на екран, 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 дозволений в таблиці), на який посилався даних зовнішній ключ, якщо NULL не дозволено – видаляв рядок.

Приклад з моєї 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_cl
ub_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('Successfully 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.

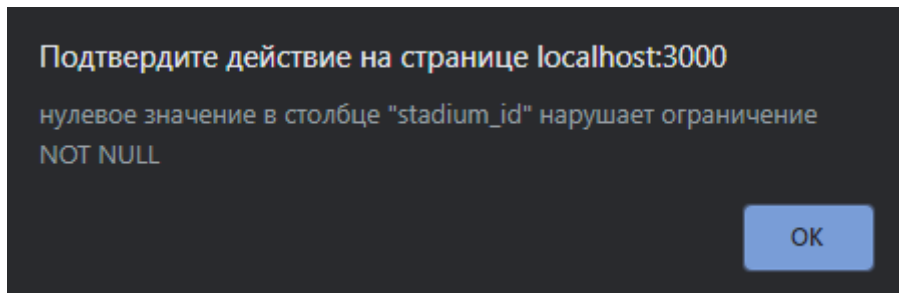
Дослідження режимів обмеження ON DELETE

1. Режим CASCADE

При видаленні запису з таблиці Stadium, запис з таблиці Club_Stadium видалюється.

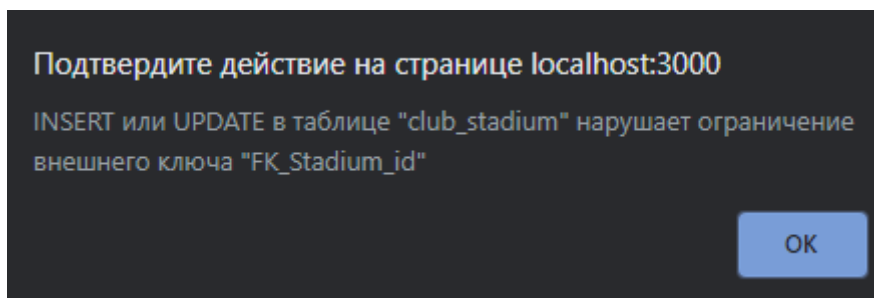
2. Режим SET NULL

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.



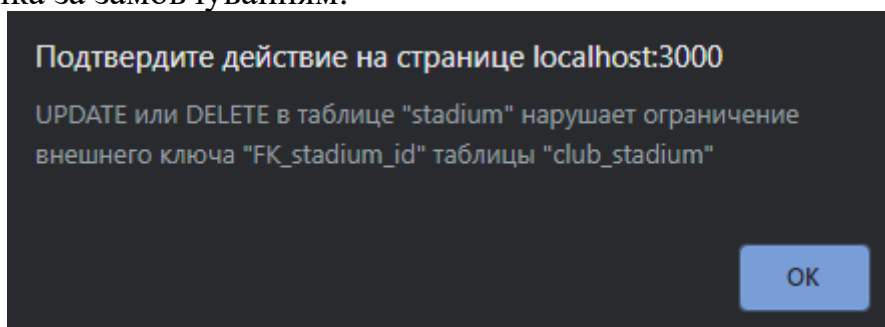
3. Режим SET DEFAULT (значення за замовчуванням = 0)

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



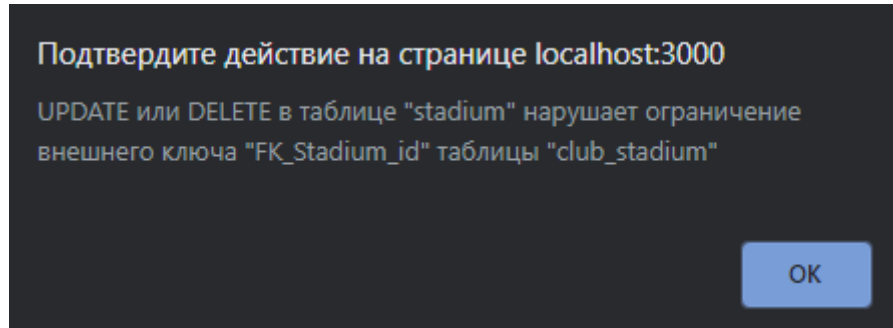
4. Режим NO ACTION

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

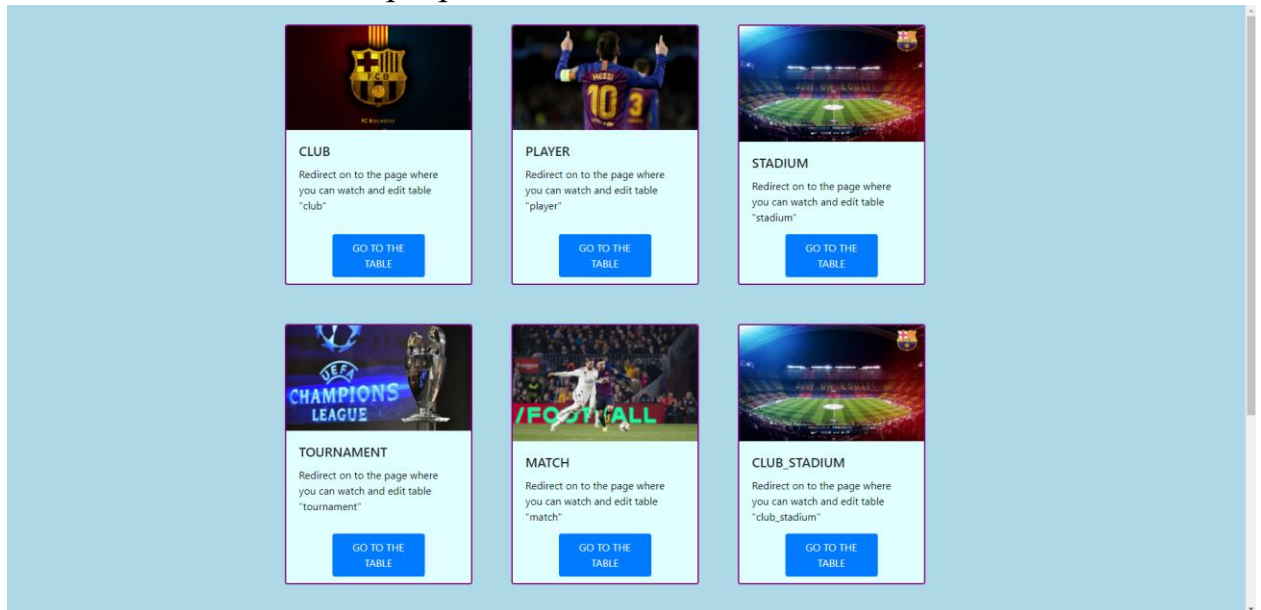


Рисунок 2 - Меню

Інтерфейс програми

При виконанні лабораторної роботи для всіх таблиць бази даних було розроблено однаковий інтерфейс. Розглянемо його на прикладі найбільшої з них.

При переході на сторінку таблиці відображаються усі її записи.

id	tournament_id	home_club_id	away_club_id	result	ticket_cost	datetime	Edit	Delete
8	1	1	63	3:0	\$30.00	Oct/25/2020	Edit	Delete
9	1	1	64	3:1	\$30.00	Oct/01/2020	Edit	Delete
10	1	63	65	1:0	\$30.00	Sep/28/2020	Edit	Delete
11	1	64	65		\$30.00	Nov/05/2020	Edit	Delete
12	2	65	66	1:0	\$40.00	Oct/22/2020	Edit	Delete
13	2	66	67	2:4	\$30.00	Oct/29/2020	Edit	Delete
14	2	67	68	4:4	\$40.00	Oct/22/2020	Edit	Delete
15	2	68	69	3:2	\$40.00	Oct/22/2020	Edit	Delete
16	2	69	66	3:1	\$40.00	Oct/25/2020	Edit	Delete
17	7	1	70	2:2	\$45.00	Oct/13/2020	Edit	Delete
18	7	70	71	3:0	\$45.00	Oct/22/2020	Edit	Delete

Рисунок 3 - Приклад Таблиці.

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

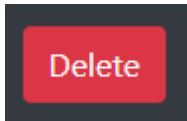


Рисунок 4 - Кнопка видалення.
Або можна редагувати натиснувши кнопку “Edit”.



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

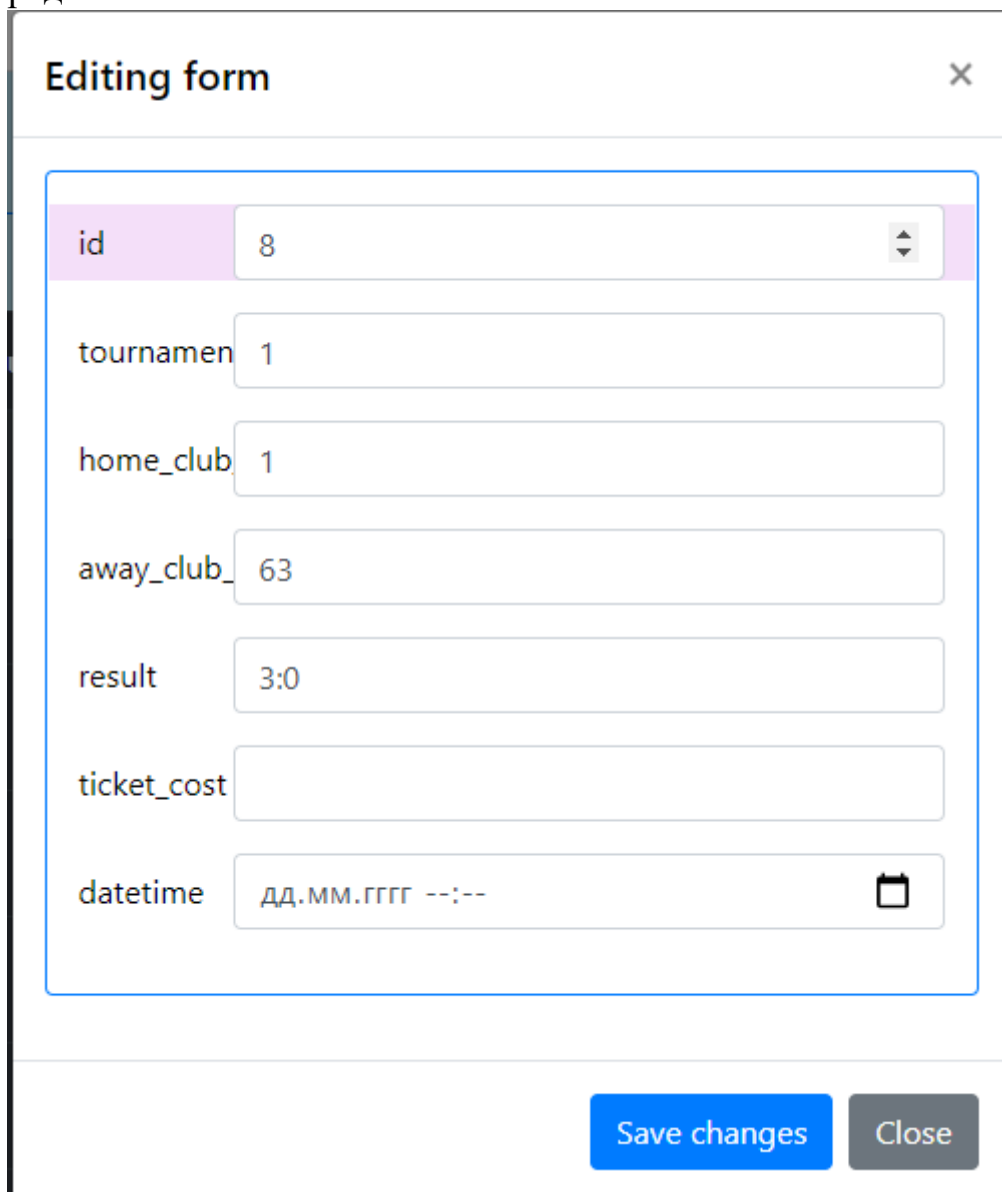
A screenshot of a web application's "Editing form" dialog box. The form has a title bar with "Editing form" and a close button (X). Inside, there are several input fields: "id" (value 8), "tournamen" (value 1), "home_club" (value 1), "away_club_" (value 63), "result" (value 3:0), "ticket_cost" (empty), and "datetime" (value DD.MM.YYYY --:-- with a calendar icon). At the bottom, there are two buttons: "Save changes" (blue) and "Close" (grey).

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

Searching/Creating form

id	<input type="text"/>
tournament_id	<input type="text"/>
home_club_id	<input type="text"/>
away_club_id	<input type="text"/>
result	<input type="text"/>
ticket_cost	<input type="text"/>
datetime	<input type="text" value="ДД.ММ.РРРР --:--"/>

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

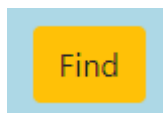


Рисунок 8 - Кнопка пошуку. Після чого, в таблиці будуть значення, які відповідають значенням в цій формі. Для прикладу, знайдемо всі рядки, в яких tournament_id = 1. Для цього в поле tournament_id введемо 1 і натиснемо кнопку “Find”. І отримаємо...

id	tournament_id	home_club_id	away_club_id	result	ticket_cost	datetime	Edit	Delete
8	1	1	63	3:0	\$30.00	Oct/25/2020	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
9	1	1	64	3:1	\$30.00	Oct/01/2020	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
10	1	63	65	1:0	\$30.00	Sep/28/2020	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
11	1	64	65		\$30.00	Nov/05/2020	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

Рисунок 9 - Приклад пошуку.

Далі для прикладу створимо, нехай такий рядок.

id	<input type="text"/>
tournament_id	<input type="text" value="7"/>
home_club_id	<input type="text" value="1"/>
away_club_id	<input type="text" value="63"/>
result	<input type="text" value="2:1"/>
ticket_cost	<input type="text" value="40"/>
datetime	<input type="text" value="25.10.2020 20:00"/>
<input type="button" value="Find"/> <input type="button" value="Create"/>	

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

19	7	1	63	2:1	\$40.00	Oct/25/2020	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
----	---	---	----	-----	---------	-------------	-------------------------------------	---------------------------------------

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

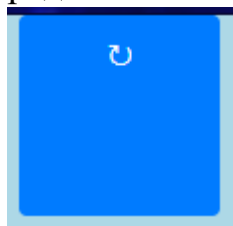


Рисунок 12 - Кнопка видалення пошуку.

Для виконання цієї лабораторної 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 = new Pool({
  user: "postgres",
  password: "1928sfsf",
  host: "localhost",
  port: 5432,
  database: "footballtournaments",
});

module.exports = pool;
```

Код програми написан мовою програмування JavaScript.

Клієнтська частина

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
serviceWorker.unregister();
```

App.js

```
import React from 'react';
```

```

import {BrowserRouter as Router} from 'react-router-dom'
import { Routes } from './Routes';

function App() {
  const routes = Routes()
  return (
    <div className="container">
      <Router>
        {routes}
      </Router>
    </div>
  );
}

export default App;

```

Routes.js

```

import React from 'react'
import {Switch, Route, Redirect} from 'react-router-dom'
import Table from './components/Table'
import { MainMenu } from './components/MainMenu'
import {types} from './components/Table'

export const Routes = () => {

  //creating structure of my tables
  //it is are arrays of objects, those objects represents columns
  //first value in object is name of the column, second - it is type
  const clubTable = [
    {field: 'id', type: types.number},
    {field: 'name', type: types.string},
  ]

  const clubsTournamentsTable = [
    {field: 'id', type: types.number},
    {field: 'club_id', type: types.number},
    {field: 'tournament_id', type: types.number},
  ]

```

```
]

const clubStadiumTable = [
  {field: 'club_id', type: types.number},
  {field: 'stadium_id', type: types.number},
]

const tournamentTable = [
  {field: 'id', type: types.number},
  {field: 'title', type: types.string},
  {field: 'teams_count', type: types.number},
]

const playerTable = [
  {field: 'id', type: types.number},
  {field: 'firstname', type: types.string},
  {field: 'lastname', type: types.string},
  {field: 'club_id', type: types.number},
  {field: 'birthday', type: types.date},
]

const stadiumTable = [
  {field: 'id', type: types.number},
  {field: 'title', type: types.string},
  {field: 'capacity', type: types.number},
  {field: 'city', type: types.string},
]

const matchTable = [
  {field: 'id', type: types.number},
  {field: 'tournament_id', type: types.number},
  {field: 'home_club_id', type: types.number},
  {field: 'away_club_id', type: types.number},
  {field: 'result', type: types.string},
  {field: 'ticket_cost', type: types.money},
  {field: 'datetime', type: types.datetime},
]
```

```

    return(
      //creating routes, it is set up component(s) for each possible api of the application
      <Switch>
        <Route path="/" exact>
          <MainMenu />
        </Route>

        <Route path="/club" exact>
          <Table table={clubTable} tableName="club" randomize={true}/>
        </Route>

        <Route path="/player" exact>
          <Table table={playerTable} tableName="player" randomize={false}/>
        </Route>

        <Route path="/stadium" exact>
          <Table table={stadiumTable} tableName="stadium" randomize={true}/>
        </Route>

        <Route path="/match" exact>
          <Table table={matchTable} tableName="match" randomize={false}/>
        </Route>

        <Route path="/tournament" exact>
          <Table table={tournamentTable} tableName="tournament" randomize={true}/>
        </Route>

        <Route path="/clubs_tournaments" exact>
          <Table table={clubsTournamentsTable} tableName="clubs_tournaments" randomize={false}/>
        </Route>

        <Route path="/club_stadium" exact>

```

```

        <Table table={clubStadiumTable} tableName="club
_stadium" randomize={false}/>
      </Route>

      <Redirect to="/" />
    </Switch>
  )}

```

Table.js

```

import React, {Fragment, useState, useEffect} from 'react'
import { EditRow } from './EditRow'

export const serverUrl = 'http://192.168.0.20:5000'

export const types = {
  date: 'date',
  datetime: 'timestamp',
  money: 'money',
  number: 'integer',
  string: 'varchar',
}

const Table = (props) => {

  //array of all records, which need to be shown for viewer
  const [rows, setRows] = useState([])
  //hold data, for filtering information
  const [searchRow, setSearchRow] = useState(null)
  //contain count of records which should be randomly created
  const [randomRows, setRandomRows] = useState(0)
  //variable, used for updating edit window(presented like modal)
  const [toChange, setToChange] = useState(false)

  //makes GET request to server and getting all rows of a table
  const getRows = async (tableName) => {

```



```

    try {
      const response = await fetch(
        `${serverUrl}/${tableName}`, {
          method: 'GET',
        }
      )

      const jsonData = await response.json()
      if(response.status === 400) {
        window.alert(jsonData)
        return
      }
      setRows(jsonData)

    } catch (error) {
      console.log(error.message);
    }
  }

  // loading all rows of the table
  useEffect( () => {

    getRows(props.tableName)

  }, [props.tableName])

  //looking is it need to updat modal window
  useEffect( () => {
    setToChange(true)
  }, [rows])

  //called when typing in search form
  const searchInputHandler = event => {
    // set value of field with name == to it is input name
    setSearchRow({
      ...searchRow,
      [event.target.name]: event.target.value,
    })
  }
}

```

//called when trying to send data on server for filltering records

```
const toSearch = async () => {
  try {
    const body = searchRow
    const response = await fetch(
      `${serverUrl}/${props.tableName}/search`, {
        method: 'POST',
        headers: {'Content-
Type': 'application/json'},
        body: JSON.stringify(body)
      })

    const data = await response.json()
    if(response.status === 400) {
      window.alert(data)
      return
    }
    setRows(data)
    setSearchRow(null)
  } catch (error) {
    console.log(error.message);
  }
}
```

//Sends POST request on server to create new row in the table

```
const createRow = async () => {
  try {
    const body = {}
    props.table.forEach( (row, index) => {
      body[row.field] = document.getElementById(`search-create-${row.field}`).value
    })

    const response = await fetch(
      `${serverUrl}/${props.tableName}/new`, {
        method: 'POST',
```

```

        headers: {'Content-
Type': 'application/json'},
        body: JSON.stringify(body)
    }
)

const data = await response.json()
if(response.status === 400) {
    window.alert(data)
    return
}
data.forEach(element => {
    if(Object.keys(element).includes( props.table[0
].field )) rows.push(element)
})

props.table.forEach( (row) => {
    document.getElementById(`search-create-
${row.field}`).value = ''
})
setSearchRow (null)
setRows(rows)

} catch (error) {
    console.log(error.message)
}
}

//deleting row from rows array by it index
const deleteRow = async row_id => {

    try {
        const response = await fetch(
            `${serverUrl}/${props.tableName}/${rows[row_id]
[props.table[0].field]}`, {
            method: 'DELETE',
        }
    )
}

```

```

        if(response.status === 400) {
            window.alert(await response.json())
            return
        } else {
            setRows(rows.filter((row, index) => index !== row_i
d))
        }
        //editingField(false)
    } catch (error) {
        console.log(error.message)
    }
}

//load all records from current table
const reload = (tableName) => {
    setSearchRow({})
    getRows(tableName)
}

//update how many random records should be created
const countInputHandler = (e) => {
    setRandomRows(e.target.value)
}

//send request on the server to create randoms records
const createRandomRows = async () => {
    try {
        const body = {count: randomRows}

        const response = await fetch(
            `${serverUrl}/${props.tableName}/rand`, {
                method: 'POST',
                headers: {'Content-
Type': 'application/json'},
                body: JSON.stringify(body)
            }
        )

        const data = await response.json()

```

```

        if(response.status === 400) {
            window.alert(data)
            return;
        }
        let _ = [...rows]
        data.forEach(element => {
            if(Object.keys(element).includes( props.table[0
].field )) _.push(element)
        })
        setRows(_)
    } catch (error) {
        console.log(error.message)
    }
}

return (

    <div className="container">
        <div >
            <span onClick={() => reload(props.tableName)} c
lassName="position-absolute btn btn-
primary " style={{right: '20px', width: '100px', height: '100px
'}} >#8635;</span>
        </div>

        /* block for filltering rows or creating records */
    </div>

    <div className="search-create-form">
        <h1 className="text-
center">Searching/Creating form</h1>

        /* div play role of form */
        <div className="p-3 rounded border border-
primary">
            {
                props.table.map( (row, index) => {
                    let type

```

```

        switch (row.type) {
            case types.date:
                type='date'
                break
            case types.datetime:
                type='datetime-local'
                break
            case types.money:
                type='number'
                break
            case types.string:
                type='text'
                break
            case types.number:
                type='number'
                break
            default:
                type='text'
                break;
        }
        return (
            <div className="form-
group row" key={index}>
                <label className="col-sm-2 col-
form-label" name={row.field} htmlFor={`search-create-
${row.field}`}>{row.field}</label>
                /* block for typing data */
                <div className="col-sm-
10"><input className="form-control" id={`search-create-
${row.field}`} name={row.field} type={type} value={searchRow ?
searchRow[row.field] : ''} onChange={ e => searchInputHandler(e
)} /></div>
                </div>
            )
        })
    }
    /* start searching for rows */
    <button className="btn btn-warning mr-
3" onClick={toSearch} >Find</button>
    /* creates record */

```

```

        <button className="btn btn-
success" onClick={createRow} >Create</button>
    </div>
</div>

<table className='table mt-5 table-hover table-
dark'>
    {/* headers of the table */}
    <thead>
        <tr>
            {
                props.table.map( (row, index) => {
                    return (
                        <Fragment key={index}>
                            <th >{row.field}</th>
                        </Fragment>
                    )
                })
            }
            <th>Edit</th>
            <th>Delete</th>
        </tr>
    </thead>
    <tbody>
        {
            // converting rows array into html
            rows.map( (row, index) => {
                return (
                    <tr key={index}>
                        {
                            props.table.map( (entityRow
, ind) => {
                                return (
                                    <td key={ind} >{row[ent
ityRow.field]}</td>
                                )
                            })
                        }
                    )
                })
            })
        }
    </tbody>
</table>

```

```

        }
        <td><EditRow table={props.table} update={getRows} tableName={props.tableName} entity={{...rows[index]}} changeRow={toChange}/></td>
        <td><button className="btn btn-danger" onClick={() => deleteRow(index)}>Delete</button></td>
      </tr>
    )
  })
}
</tbody>
</table>

  /* block for creating random data if it possible for this table*/
  {
    props.randomize ?
      (<div className="form-inline">
        <div className="form-group mt-2" >
          <input className="form-control mr-2" type="number" value={randomRows} onChange={e => countInputHandler(e)} />
          <button onClick={createRandomRows} className="btn btn-success">Create random rows</button>
        </div>
      </div>)
    : null
  }
</div>
)
}

export default Table

```

MainMenu.js

```

import React, {Fragment} from 'react'
import './styles.css'

```



```

import { Card } from './Card'

export const MainMenu = () => {
  return (
    <Fragment>
      <div className="d-flex flex-wrap">
        <Card link="/club" tableName="club" img="Barca"
        />
        <Card link="/player" tableName="player" img="Messi2" />
        <Card link="/stadium" tableName="stadium" img="Stadium" />
        <Card link="/tournament" tableName="tournament"
        img="UCL" />
        <Card link="/match" tableName="match" img="match" />
        <Card link="/club_stadium" tableName="club_stadium" img="Stadium" />
        <Card link="/clubs_tournaments" tableName="clubs_tournaments" img="UCL" />
      </div>
    </Fragment>
  )
}

```

Card.js

```

import React, {Fragment} from 'react'
import { NavLink } from 'react-router-dom'
import './components/styles.css'

export const Card = (props) => {

  return (
    <Fragment>
      <div className="card div-card" style={{width: '18rem'}}>
        <img className="card-img-top" src={process.env.PUBLIC_URL + `/images/${props.img}.jpg`} alt={props.img} />
        <div className="card-body">

```

```

        <h5 className="card-title text-
uppercase">{props.tableName}</h5>
        <p className="card-
text" style={{color: 'black'}} >Redirect on to the page where y
ou can watch and edit table "{props.tableName}"</p>
        <NavLink className="nav-link btn btn-
primary card-btn" to={props.link}>
            GO TO THE TABLE
        </NavLink>
    </div>
</div>

</Fragment>
)
}

```

EditRow.js

```

import React, {Fragment, useState, useEffect} from 'react'
import {types} from './Table'
import {serverUrl} from './Table'

export const EditRow = (props) => {

    const [entity, setEntity] = useState({...props.entity})

    useEffect(() => {
        setEntity({...props.entity})
    }, [props.toChange, props.entity])

    const editRowHandler = async () => {
        try {
            console.log('aaaaaaa');
            const body = entity
            const response = await fetch(
                `${serverUrl}/${props.tableName}/${entity[props
.table[0].field]}`, {
                method: 'PUT',
                headers: {'Content-
Type': 'application/json'},
                body: JSON.stringify(body)
            }
        )
    }
}

```

```

    }
  )

  await response.json()

  props.update(props.tableName)
  setEntity({})
} catch (error) {
  console.log(error.message)
}
}

const editInputChangeHandler = event => {
  console.log('aaaaaa, event', event.target.value);
  setEntity({...entity, [event.target.name]: event.target
.value})
  console.log(entity)
}

return (
  <Fragment>
    <button
      type="button"
      className="btn btn-warning"
      data-toggle="modal"
      data-target={`#edit-modal-
${entity[props.table[0].field]}`}>Edit</button>

    <div style={{color: 'black'}} className="modal" id=
{`edit-modal-${entity[props.table[0].field]}`} tabIndex="-
1" role="dialog">
      <div className="modal-dialog" role="document">
        <div className="modal-content">
          <div className="modal-header">
            <h5 className="modal-
title">Editing form</h5>

```



```

                                <div className="form-
group row" key={index}>
                                <label className="c
ol-sm-2 col-form-label" name={row.field} htmlFor={`#edit-
${row.field}-
${entity[props.table[0].field]}`}>{row.field}</label>

                                <div className="col
-sm-10"><input className="form-control" id={`edit-${row.field}-
${entity[props.table[0].field]}`} name={row.field} type={type}
onChange={e => editInputChangeHandler(e)} value={entity[row.fie
ld]} /></div>

                                </div>

                                )
                                })
                                }

                                </div>
                                { /* ending of editing form */}
                                </div>

                                { /* enging of editing form inside of mo
dal */}

                                </div>
                                <div className="modal-footer">
                                    <button type="button" onClick={editRowH
andler} data-dismiss="modal" className="btn btn-
primary">Save changes</button>
                                    <button type="button" onClick={() => set
Entity(props.entity)} className="btn btn-secondary" data-
dismiss="modal">Close</button>
                                </div>
                                </div>
                                </div>
                                </Fragment>
                                )

```

```
}
```

Серверна частина програми

index.js

```
const express = require('express')
const cors = require('cors')
const app = express()

app.use(cors())
app.use(express.json())
app.use('/', require('./routes/clubRoutes'))
app.use('/', require('./routes/playerRoutes'))
app.use('/', require('./routes/stadiumRouter'))
app.use('/', require('./routes/matchRoutes'))
app.use('/', require('./routes/tournamentRoutes'))
app.use('/', require('./routes/clubs_tournaments'))
app.use('/', require('./routes/club_stadium'))

app.listen(5000, () => {
  console.log('Server has started on port 5000');
})
```

clubRoutes.js

```
const {Router} = require('express')
const pool = require('../db')
const router = Router()

//returns all records in the table
router.get('/club', async(req, res) => {
  try {
    const clubs = await pool.query('SELECT * FROM club ORDER BY id;')
    res.json(clubs.rows)
  } catch (error) {
    res.status(400).json('Error');
    console.log(error.message)
  }
})
```

```

//returns all rows which match to given paramatres
router.post('/club/search', async(req, res) => {
  try {
    const {name} = req.body
    const qu = `SELECT * FROM club WHERE LOWER(name) LIKE \
'${name}%\`
    const response = await pool.query(qu)
    res.json(response.rows)
  } catch (error) {
    res.status(400).json('Error');

    console.log(error.message);
  }
})

//create new record into the table by givin information
router.post('/club/new', async(req, res) => {
  try {
    const {name} = req.body
    let qu = `INSERT INTO club(name) values ('${name}')`;
    let response = await pool.query(qu)
    qu = `SELECT * FROM club ORDER BY id DESC FETCH FIRST 1
ROW ONLY;`
    response = await pool.query(qu)
    res.json(response.rows)
  } catch (error) {
    res.status(400).json('Error');
    console.log(error.message);
  }
})

//create random record
router.post('/club/rand', async(req, res) => {
  try {
    const {count} = req.body
    let qu = `INSERT INTO club(name) select getrandomstring
(10) as name from generate_series(1, ${count});`
    let response = await pool.query(qu)
    qu = `SELECT * FROM club ORDER BY id DESC FETCH FIRST $
{count} ROW ONLY;`

```

```

        response = await pool.query(qu)
        res.json(response.rows)
    } catch (error) {
        res.status(400).json('Error');

        console.log(error.message);
    }
})

//changes record with given id
router.put('/club/:id', async(req, res) => {
    try {
        const {id} = req.params
        const {name} = req.body
        let qu = `UPDATE club SET name = \`${name}\` WHERE id = ${id};`

        let response = await pool.query(qu)
        qu = `SELECT * FROM club WHERE id = ${id};`
        response = await pool.query(qu)
        res.json(response.rows[0])
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

//delete record by given id
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('Successfully deleted')
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message)
    }
})

module.exports = router

```

playerRoutes.js

```

const {Router} = require('express')
const pool = require('../db')
const router = Router()

const selectAll = 'SELECT id, firstname, lastname, club_id, FOR
MAT(birthday::varchar, \'d\', \'en-US\') as birthday FROM'

//returns all records in the table
router.get('/player', async(req, res) => {
    try {
        const players = await pool.query(`${selectAll} player;`
    )
        res.json(players.rows)
    } catch (error) {
        res.status(400).json('Error');

        console.log(error.message)
    }
})

//create random record
router.post('/player/rand', async(req, res) => {
    try {
        const {count} = req.body
        let qu = `INSERT INTO player(name) select getrandomstri
ng(10) as firstname, getrandomstring(10) as lastname, getrandom
date() as birthday, from generate_series(1, ${count});`

```

```

        let response = await pool.query(qu)
        qu = `${selectAll} player ORDER BY id DESC FETCH FIRST
${count} ROW ONLY;`
        response = await pool.query(qu)
        res.json(response.rows)
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

//returns all rows which match to given paramatres
router.post('/player/search', async(req, res) => {
    try {
        let {lastname, firstname, club_id, birthday} = req.body
        let qu = `${selectAll} player WHERE TRUE `
        if(lastname !== undefined && lastname.length !== 0) {
            qu += `AND LOWER(lastname) LIKE \'%${lastname}%\' `
        }
        if(firstname !== undefined && firstname.length !== 0) {
            qu += `AND LOWER(firstname) LIKE \'%${firstname}%\' `
        }
        if(club_id !== undefined && club_id.length !== 0) {
            qu += `AND club_id = ${club_id} `
        }
        if(birthday !== undefined && birthday.length !== 0) {
            qu += `AND birthday = '${birthday}':::date`
        }
        qu += ';'

        const response = await pool.query(qu)
        res.json(response.rows)
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

```

```

router.post('/player/new', async(req, res) => {
  try {
    let {lastname, firstname, club_id, birthday} = req.body
    let qu = `INSERT INTO player(firstname, lastname, club_
id, birthday) values (\`${firstname}\`, \`${lastname}\`, ${club_
_id}, \`${birthday}\`::date);`

    let response = await pool.query(qu)
    qu = `${selectAll} player ORDER BY id DESC FETCH FIRST
1 ROW ONLY;`
    response = await pool.query(qu)
    //response = await pool.query(qu)
    res.json(response.rows)
  } catch (error) {
    res.status(400).json('Error');
    console.log(error.message);
  }
})

```

```

//changes record with given id
router.put('/player/:id', async(req, res) => {
  try {
    const {id} = req.params
    const {firstname, lastname, club_id, birthday} = req.bo
dy

    let qu = ``
    if(lastname !== undefined && lastname.length !== 0) {
      qu += `lastname = '${lastname}\`, `
    }
    if(firstname !== undefined && firstname.length !== 0) {
      qu += `firstname = '${firstname}\`, `
    }
    if(club_id !== undefined && club_id.length !== 0) {
      qu += `club_id = ${club_id}, `
    }
    if(birthday !== undefined && birthday.length !== 0) {
      qu += `birthday = '${birthday}'::date`
    }
    if(qu[qu.length - 1] === ',') qu[qu.length - 1] = ' '
  }
}

```

```

        let query2 = `UPDATE player SET ${qu} WHERE id = ${id}`;

        let response = await pool.query(query2)
        qu = `${selectAll} player WHERE id = ${id}`;
        response = await pool.query(qu)
        res.json(response.rows[0])
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

//delete record by given id
router.delete('/player/:id', async(req, res) => {
    try {
        const {id} = req.params
        const qu = `DELETE FROM player WHERE id = ${id}`;
        const response = await pool.query(qu)

        res.json('Successfully deleted')
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message)
    }
})

module.exports = router

```

stadiumRoutes.js

```

const {Router} = require('express')
const pool = require('../db')
const router = Router()

//returns all records in the table
router.get('/stadium', async(req, res) => {
    try {

```

```

        const stadiums = await pool.query(`SELECT * FROM stadium;`)

        res.json(stadiums.rows)
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message)
    }
})

//create random record
router.post('/stadium/rand', async(req, res) => {
    try {
        const {count} = req.body
        let qu = `INSERT INTO stadium(title, city, capacity) select getrandomstring(10) as title, getrandomstring(10) as city, randomnum(100000) as capacity from generate_series(1, ${count});`

        let response = await pool.query(qu)
        qu = `SELECT * FROM stadium ORDER BY id DESC FETCH FIRST ${count} ROW ONLY;`
        response = await pool.query(qu)
        res.json(response.rows)
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

//returns all rows which match to given paramatres
router.post('/stadium/search', async(req, res) => {
    try {
        let {title, city, capacity} = req.body
        let qu = `SELECT * FROM stadium WHERE TRUE `
        if(title !== undefined && title.length !== 0) {
            qu += `AND LOWER(title) LIKE \'%${title}%\' `
        }
        if(city !== undefined && city.length !== 0) {
            qu += `AND LOWER(city) LIKE \'%${city}%\' `
        }
    }
})

```

```

        //qu += `AND LOCATE('${city}', LOWER(city)) > 0`
    }
    if(capacity !== undefined && capacity.length !== 0) {
        qu += `AND capacity = ${capacity}`
    }
    qu += ';'

    const response = await pool.query(qu)
    res.json(response.rows)
} catch (error) {
    res.status(400).json('Error');
    console.log(error.message);
}
})

//creates new record by give data
router.post('/stadium/new', async(req, res) => {
    try {
        let {title, city, capacity} = req.body
        let qu = `INSERT INTO stadium(city, title, capacity) va
Lues ('${city}', '${title}', ${capacity});`

        let response = await pool.query(qu)
        qu = `SELECT * FROM stadium ORDER BY id DESC FETCH FIRS
T 1 ROW ONLY;`
        response = await pool.query(qu)
        //response = await pool.query(qu)
        res.json(response.rows)
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

//edit record with given id by replacing it with new data
router.put('/stadium/:id', async(req, res) => {
    try {
        const {id} = req.params
        const {city, title, capacity} = req.body
        let qu = ``

```

```

        if(title !== undefined && title.length !== 0) {
            qu += `title = '${title}\\', `
        }
        if(city !== undefined && city.length !== 0) {
            qu += `city = '${city}\\', `
        }
        if(capacity !== undefined && capacity.length !== 0) {
            qu += `capacity = ${capacity} `
        }

        if(qu[qu.length - 1] === ',') qu[qu.length - 1] = ' '

        let query2 = `UPDATE stadium SET ${qu} WHERE id = ${id}`
;`

        let response = await pool.query(query2)
        qu = `SELECT * FROM stadium WHERE id = ${id};`
        response = await pool.query(qu)
        res.json(response.rows[0])
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

//delete record by given id
router.delete('/stadium/:id', async(req, res) => {
    try {
        const {id} = req.params
        const qu = `DELETE FROM club_stadium WHERE stadium_id =
${id};`
        DELETE FROM stadium WHERE id = ${id};`
        const response = await pool.query(qu)

        res.json('Successfully deleted')
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message)
    }
})

```

```
module.exports = router
```

matchRoutes.js

```
const {Router} = require('express')
const pool = require('../db')
const router = Router()

const selectAll = 'SELECT id, tournament_id, home_club_id, away_club_id, to_char(datetime, \'Mon/DD/YYYY\') as datetime, result, concat(\'$\', ticket_cost::numeric) as ticket_cost FROM'

//returns all records in the table
router.get('/match', async(req, res) => {
  try {
    const matchs = await pool.query(`${selectAll} match;`)
    res.json(matchs.rows)
  } catch (error) {
    res.status(400).json('Error');

    console.log(error.message)
  }
})

//returns all rows which match to given paramatres
router.post('/match/search', async(req, res) => {
  try {
    let {result, tournament_id, home_club_id, away_club_id, ticket_cost, datetime} = req.body
    let qu = `${selectAll} match WHERE TRUE `
    if(result !== undefined && result.length !== 0) {
      qu += `AND LOWER(result) LIKE LOWER(\'%${result}%\') `
    }
    if(home_club_id !== undefined && home_club_id.length !== 0) {
      qu += `AND home_club_id = ${home_club_id} `
    }
  }
})
```



```

        if(away_club_id !== undefined && away_club_id.length !==
= 0) {
            qu += `AND away_club_id = ${away_club_id} `
        }
        if(tournament_id !== undefined && tournament_id.length
!== 0) {
            qu += `AND tournament_id = ${tournament_id} `
        }
        if(ticket_cost !== undefined && ticket_cost.length !==
0) {
            qu += `AND ticket_cost::integer = ${ticket_cost} `
        }
        if(datetime !== undefined && datetime.length !== 0) {
            qu += `AND datetime::date = '${datetime}'::date`
        }
        qu += ';'

        const response = await pool.query(qu)
        res.json(response.rows)
        //res.json('Server has got a data')
    } catch (error) {
        res.status(400).json('Error');

        console.log(error.message);
    }
})

//create new record into the table by givin information
router.post('/match/new', async(req, res) => {
    try {
        let {result, tournament_id, home_club_id, away_club_id,
ticket_cost, datetime} = req.body
        let qu = `INSERT INTO match(tournament_id, result, away
_club_id, home_club_id, datetime, ticket_cost) values (${tourna
ment_id}, '${result}', ${away_club_id}, ${home_club_id}, '${${
datetime}}'::timestamp, '${ticket_cost}'::money);`

        let response = await pool.query(qu)
        qu = `${selectAll} match ORDER BY id DESC FETCH FIRST 1
ROW ONLY;`
    }

```

```

        response = await pool.query(qu)
        //response = await pool.query(qu)
        res.json(response.rows)
    } catch (error) {
        console.log(error.message);
    }
})

//changes record with given id
router.put('/match/:id', async(req, res) => {
    try {
        const {id} = req.params
        const {result, tournament_id, home_club_id, away_club_id, ticket_cost, datetime} = req.body
        let qu = ``

        if(result !== undefined && result.length !== 0) {
            qu += `result = '${result}' `
        }
        if(home_club_id !== undefined && home_club_id.length !== 0) {
            qu += `home_club_id = ${home_club_id} `
        }
        if(away_club_id !== undefined && away_club_id.length !== 0) {
            qu += `away_club_id = ${away_club_id} `
        }
        if(tournament_id !== undefined && tournament_id.length !== 0) {
            qu += `tournament_id = ${tournament_id} `
        }
        if(ticket_cost !== undefined && ticket_cost.length !== 0) {
            qu += `ticket_cost = '${ticket_cost}'::money `
        }
        if(datetime !== undefined && datetime.length !== 0) {
            qu += `datetime = '${datetime}'::timestamp`
        }
        if(qu[qu.length - 1] === ',') qu[qu.length - 1] = ' '
    }

```

```

        let query2 = `UPDATE match SET ${qu} WHERE id = ${id};`
        let response = await pool.query(query2)
        qu = `${selectAll} match WHERE id = ${id};`
        response = await pool.query(qu)
        res.json(response.rows[0])
    } catch (error) {
        res.status(400).json('Error');

        console.log(error.message);
    }
})

//delete record by given id
router.delete('/match/:id', async(req, res) => {
    try {
        const {id} = req.params
        const qu = `DELETE FROM match WHERE id = ${id};`
        const response = await pool.query(qu)

        res.json('Successfully deleted')
    } catch (error) {
        res.status(400).json('Error');

        console.log(error.message)
    }
})

module.exports = router

```

tournamentRoutes.js

```

const {Router} = require('express')
const pool = require('../db')
const router = Router()

//returns all records in the table
router.get('/tournament', async(req, res) => {
    try {

```

```

        const tournaments = await pool.query('SELECT * FROM tournament;')

        res.json(tournaments.rows)
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message)
    }
})

//returns all rows which match to given paramatres
router.post('/tournament/search', async(req, res) => {
    try {
        const {title, teams_count} = req.body
        let qu = `SELECT * FROM tournament WHERE TRUE `
        if(teams_count !== undefined && teams_count.length !== 0) {
            qu += `AND teams_count = ${teams_count} `
        }
        if(title !== undefined && title.length !== 0) {
            qu += ` AND LOWER(title) LIKE LOWER(\'%${title}%\')`
        }
        qu += ';';
        ;
        const response = await pool.query(qu)
        res.json(response.rows)
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

//create new record into the table by givin information
router.post('/tournament/new', async(req, res) => {
    try {
        const {title, teams_count} = req.body
        let qu = `INSERT INTO tournament(title, teams_count) values (\`${title}\`, ${teams_count});`
    }
})

```

```

    let response = await pool.query(qu)
    qu = `SELECT * FROM tournament ORDER BY id DESC FETCH F
IRST 1 ROW ONLY;`
    response = await pool.query(qu)
    res.json(response.rows)
  } catch (error) {
    res.status(400).json('Error');
    console.log(error.message);
  }
})

//create random record
router.post('/tournament/rand', async(req, res) => {
  try {
    const {count} = req.body
    let qu = `INSERT INTO tournament(title, teams_count) se
lect UPPER(getrandomstring(4)) as title, (10 + (randomnum(10)))
as teams_count from generate_series(1, ${count});`

    let response = await pool.query(qu)
    qu = `SELECT * FROM tournament ORDER BY id DESC FETCH F
IRST ${count} ROW ONLY;`
    response = await pool.query(qu)
    res.json(response.rows)
  } catch (error) {
    res.status(400).json('Error');
    console.log(error.message);
  }
})

//changes record with given id
router.put('/tournament/:id', async(req, res) => {
  try {
    const {id} = req.params
    const {title, teams_count} = req.body
    let qu = `UPDATE tournament SET title = '${title}', t
eams_count = ${teams_count} WHERE id = ${id};`

    let response = await pool.query(qu)

```

```

        qu = `SELECT * FROM tournament WHERE id = ${id}`;
        response = await pool.query(qu)
        res.json(response.rows[0])
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

//delete record by given id
router.delete('/tournament/:id', async(req, res) => {
    try {
        const {id} = req.params
        const qu = `DELETE FROM tournament where id = ${id}`;
        const response = await pool.query(qu)

        res.json('Successfully deleted')
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message)
    }
})

module.exports = router

```

club_stadium.js

```

const {Router} = require('express')
const pool = require('../db')
const router = Router()

//returns all records in the table
router.get('/club_stadium', async(req, res) => {
    try {
        const club_stadiums = await pool.query('SELECT * FROM club_stadium;')
        res.json(club_stadiums.rows)
    } catch (error) {
        res.status(400).json('Error');
    }
})

```

```

        console.log(error.message)
    }
})

//returns all rows which match to given paramatres
router.post('/club_stadium/search', async(req, res) => {
    try {
        const {club_id, stadium_id} = req.body
        let qu = `SELECT * FROM club_stadium WHERE TRUE `
        if(club_id !== undefined && club_id.length !== 0) {
            qu += `AND club_id = ${club_id} `
        }
        if(stadium_id !== undefined && stadium_id.length !== 0)
        {
            qu += ` AND stadium_id = ${stadium_id} `
        }
        qu += ';'
        const response = await pool.query(qu)
        res.json(response.rows)
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

//create new record into the table by givin information
router.post('/club_stadium/new', async(req, res) => {
    try {
        const {club_id, stadium_id} = req.body
        let qu = `INSERT INTO club_stadium(club_id, stadium_id)
values (${club_id}, ${stadium_id});`
        let response = await pool.query(qu)
        qu = `SELECT * FROM club_stadium ORDER BY club_id DESC
FETCH FIRST 1 ROW ONLY;`
        response = await pool.query(qu)
        //response = await pool.query(qu)
        res.json(response.rows)
    } catch (error) {
        res.status(400).json('Error');
        console.log(error.message);
    }
})

```

```

    }
  })

  //changes record with given id
  router.put('/club_stadium/:id', async(req, res) => {
    try {
      const {id} = req.params
      const {club_id, stadium_id} = req.body
      let qu = `UPDATE club_stadium SET club_id = ${club_id},
teams_count = ${stadium_id} WHERE id = ${id};`

      let response = await pool.query(qu)
      qu = `SELECT * FROM club_stadium WHERE id = ${id};`
      response = await pool.query(qu)
      console.log(response.rows);
      res.json(response.rows[0])
    } catch (error) {
      res.status(400).json('Error');
      console.log(error.message);
    }
  })

  //delete record by given id
  router.delete('/club_stadium/:id', async(req, res) => {
    try {
      const {id} = req.params
      const qu = `DELETE FROM club_stadium where id = ${id};`
      const response = await pool.query(qu)

      res.json('Successfully deleted')
    } catch (error) {
      res.status(400).json('Error');
      console.log(error.message)
    }
  })
})

module.exports = router

```


clubs_tournaments.js

```
const {Router} = require('express')
const pool = require('../db')
const router = Router()

//returns all records in the table
router.get('/clubs_tournaments', async(req, res) => {
  try {
    const clubs_tournamentss = await pool.query('SELECT * FROM clubs_tournaments;')
    res.json(clubs_tournamentss.rows)
  } catch (error) {
    res.status(400).json('Error');
    console.log(error.message)
  }
})

//returns all rows which match to given paramatres
router.post('/clubs_tournaments/search', async(req, res) => {
  try {
    const {club_id, tournament_id} = req.body
    let qu = `SELECT * FROM clubs_tournaments WHERE TRUE`
    if(club_id !== undefined && club_id.length !== 0) {
      qu += `AND club_id = ${club_id}`
    }
    if(tournament_id !== undefined && tournament_id.length !== 0) {
      qu += ` AND tournament_id = ${tournament_id}`
    }
    qu += ';'
    const response = await pool.query(qu)
    res.json(response.rows)
  } catch (error) {
    res.status(400).json('Error');
    console.log(error.message);
  }
})
```

```

//create new record into the table by givin information
router.post('/clubs_tournaments/new', async(req, res) => {
  try {
    const {club_id, tournament_id} = req.body
    let qu = `INSERT INTO clubs_tournaments(club_id, tournament_id) values (${club_id}, ${tournament_id});`
    let response = await pool.query(qu)
    qu = `SELECT * FROM clubs_tournaments ORDER BY id DESC FETCH FIRST 1 ROW ONLY;`
    response = await pool.query(qu)
    //response = await pool.query(qu)
    res.json(response.rows)
  } catch (error) {
    res.status(400).json('Error');
    console.log(error.message);
  }
})

//changes record with given id
router.put('/clubs_tournaments/:id', async(req, res) => {
  try {
    const {id} = req.params
    const {club_id, tournament_id} = req.body
    let qu = `UPDATE clubs_tournaments SET club_id = ${club_id}, teams_count = ${tournament_id} WHERE id = ${id};`

    let response = await pool.query(qu)
    qu = `SELECT * FROM clubs_tournaments WHERE id = ${id};`

    response = await pool.query(qu)
    res.json(response.rows[0])
  } catch (error) {
    res.status(400).json('Error');
    console.log(error.message);
  }
})

//delete record by given id
router.delete('/clubs_tournaments/:id', async(req, res) => {
  try {

```

```
const {id} = req.params
const qu = `DELETE FROM clubs_tournaments where id = ${
id};`

const response = await pool.query(qu)

res.json('Successfully deleted')
} catch (error) {
  res.status(400).json('Error');
  console.log(error.message)
}
})

module.exports = router
```