Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ПРОВЕДЕННЯ СТУДЕНТСЬКИХ СПОРТИВНИХ ЗМАГАНЬ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-1
Корсун В.О.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Корсун Вікторії Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку для проведення студентських спортивних змагань

затверджена наказом по університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 2 червня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, відкрита JavaScript бібліотека для розробки користувацького інтерфейсу вебзастосунків React.js, мова програмування JavaScript, відкрите середовище виконання Node.js, реляційна база даних PosgreSQL.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналітичний огляд вебтехнологій для розробки вебзастосунків.
2. Моделювання вебзастосунку для проведення студентських спортивних змагань.
3. Програмна реалізація вебзастосунку для проведення студентських спортивних змагань.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розробки вебзастосунків, постановка задачі, математичні моделі, інформаційна система вебзастосунку, реалізація вебзастосунку, опис прототипу вебзастосунку.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-15.04.23	
3	Аналіз літератури з досліджуваної проблеми	16.04.23-23.04.23	
4	Аналіз технічних засобів	24.04.23-30.04.23	
5	Розробка методу	01.05.23-11.05.23	
6	Програмна реалізація	12.05.23-26.05.23	
7	Оформлення пояснювальної записки	27.05.23-2.06.23	
8	Перевірка на плагіат	09.06.23	
9	Рецензування	09.06.23	
10	Підготовка презентації та доповіді	10.06.23	
11	Занесення роботи в електронний архів	10.06.23	
12	Попередній захист атестаційної роботи	11.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Кобилін О.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 65 с., 2 табл., 25 рис., 39 джерел.

ВЕБЗАСТОСУНОК, СПОРТИВНІ ЗМАГАННЯ, ПЛАВАННЯ, REACT, NODE.JS, POSTGRESQL, FRONTEND, BACKEND, АЛГОРИТМИ, ПРОТОКОЛИ.

Об’єктом роботи є вебзастосунок для проведення студентських спортивних змагань із плавання.

Метою роботи є розробка вебзастосунку для проведення змагань за плавання з можливістю реєструвати заявки спортсменів, організовувати розподіл учасників по доріжкам за допомогою алгоритмів та генерувати протоколи для визначення переможців.

У ході роботи проведений аналітичний огляд існуючих технологій для аналізу розробки вебзастосунків, змодельована інформаційна система застосунку, використані сучасні технології та фреймворки, такі як React для розробки застосунку, Node.js для розробки серверної частини та PostgreSQL для зберігання даних.

WEB APPLICATION, SPORTS, SWIMMING, REACT, NODE.JS, POSTGRESQL, FRONTEND, BACKEND, ALGORITHMS, PROTOCOLS.

The object of the work is a web application for conducting student sports competitions in swimming.

The purpose of the work is to develop a web application for swimming competitions with the ability to register athletes' applications, organize the distribution of participants on lanes using algorithms, and generate protocols for determining winners.

During the study, an analytical review of existing technologies for the analysis of web application development was carried out, the information system of the application was modeled, and modern technologies and frameworks were used, such as React for application development, Node.js for backend development, and PostgreSQL for data storage.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Фронтенд – частина веброзробки, яка займається розробкою клієнтської частини вебсайтів і застосунків

Бекенд – частина застосунку або програми, яка забезпечує обробку запитів та взаємодію з базою даних, що знаходиться на сервері

СУБД – система управління базами даних

ІС – інформаційна система

ОС – операційна система

API – Application Programming Interface (прикладний програмний інтерфейс)

ПО – предметна область

DOM – Document Object Model (об’єктна модель документа)

SPA – Simple Page Application (односторінковий застосунок)

ВСТУП

У сучасному світі спорт відіграє важливу роль у формуванні та розвитку молоді. Спортивні змагання серед студентів не тільки сприяють фізичному здоров'ю, але й сприяють формуванню командного духу, лідерських навичок та загального розвитку особистості. Організація таких змагань вимагає ефективного управління, аналізу та обробки інформації, що включає реєстрацію учасників, планування змагань, облік результатів та спілкування з учасниками та глядачами.

Основними завданнями кваліфікаційної роботи є аналіз вимог та потреб учасників спортивних змагань для розробки відповідного функціоналу вебзастосунку; проєктування інформаційної системи, включаючи структуру бази даних, архітектуру програмного забезпечення та інтерфейс користувача; розробка та реалізація вебзастосунку, який включатиме функції реєстрації учасників, планування змагань, обліку результатів та спілкування з учасниками та глядачами; тестування та валідація розробленого вебзастосунку для забезпечення його правильної роботи та відповідності вимогам.

Очікується, що розроблений вебзастосунок спростить та оптимізує процес організації та проведення студентських спортивних змагань. Він надасть зручність та доступність для учасників, адміністраторів та глядачів, забезпечуючи ефективне управління, аналітику результатів та покращену комунікацію.

Актуальність роботи полягає у розробці вебзастосунку, який забезпечує ефективне проведення студентських спортивних змагань. Цей вебзастосунок буде виконувати широкий спектр функцій, що дозволить автоматизувати багато аспектів організації та управління змаганнями, забезпечуючи зручність для учасників, адміністраторів та глядачів.

1 АНАЛІТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКІВ

1.1 Розробка вебзастосунків

Зв’язок між клієнтом та сервером є важливим аспектом веброзробки, і тому дуже важливо вибрати правильні технології для розробки клієнтської та серверної частин вебзастосунків. У даному розділі розглянуто найпопулярніші фреймворки та бібліотеки для розробки клієнтської та серверної частин вебзастосунків, порівняно їх та визначено найкращі для використання в проєкті.

Розробка вебзастосунків – це процес створення програмного забезпечення, яке може бути доступне через веббраузер. Вебзастосунки дозволяють користувачам взаємодіяти з інформацією, виконувати різноманітні завдання і зберігати дані в онлайн-середовищі.

При розробці вебзастосунків використовуються різні технології, мови програмування та фреймворки. Фронтенд-розробка займається створенням користувацького інтерфейсу [1, 2], включаючи HTML, CSS і JavaScript. Бекенд-розробка відповідає за обробку запитів користувача, взаємодію з базами даних та бізнес-логікою застосунку. Вона може використовувати мови програмування, такі як Python, Java, Ruby або PHP, а також фреймворки, які спрощують розробку.

Головна функція браузера – відкриття вебсторінок. Сторінки вебсайту складаються з коду, який браузер отримує з сервера, де розташований сайт. Цей спосіб передачі інформації між користувачем і сервером називається «клієнт-сервер», де кілька комп’ютерів обмінюються інформацією, наприклад, вводом даних у базу даних або спілкуванням у чаті. При введенні інформації «клієнт» – це програма, яку користувач використовує для

введення даних, а «сервер» – програма, яка використовується для зберігання, обробки або передачі інформації.

Найчастіше вебзастосунок являє собою вебсайт, на якому розміщені сторінки з частково або повністю несформованим вмістом. Остаточний вміст формується тільки після того, як відвідувач сайту запросить сторінку з вебсервера (рис. 1.1).

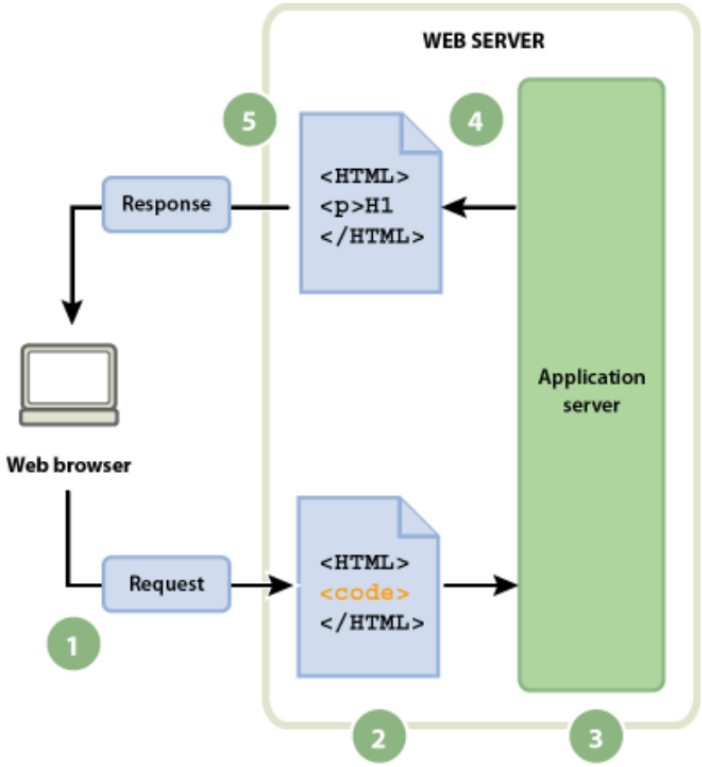


Рисунок 1.1 – Зображення принципу роботи вебзастосунків

Під час розробки вебзастосунків важливо враховувати безпеку, масштабованість і продуктивність. Також може знадобитися інтеграція з іншими системами або сторонніми сервісами.

Розробка вебзастосунків може включати такі етапи як визначення вимог, проєктування інтерфейсу, розробка функціональності, тестування, розгортання та підтримка.

Завдяки вебзастосункам користувачі можуть отримувати доступ до різноманітних послуг і функцій через Інтернет, незалежно від операційної системи або пристрою, що вони використовують. Вебзастосунки широко

застосовуються у різних галузях, включаючи електронну комерцію, соціальні мережі, фінанси, освіту та інші.

1.2 Огляд технологій клієнтської частини

Найбільш популярні фреймворки та бібліотеки для розробки клієнтської частини вебзастосунків на сьогоднішній день – це React.js, Angular та Vue.js. Кожен з них має свої переваги та недоліки, що необхідно враховувати при виборі технології для проєкту. React.js, наприклад, відрізняється високою продуктивністю, Angular – зручною структурою проєкту та багатим функціоналом, а Vue.js – легкістю та розробки та високою продуктивністю. Більш того, React.js відкриває двері для використання різноманітних бібліотек та інструментів, що можуть допомогти під час розробки вебзастосунків.

Крім React.js, Angular та Vue.js, існують і інші фреймворки та бібліотеки для розробки клієнтської частини вебзастосунків, такі як Ember.js, Backbone.js, Knockout.js та інші. Кожен з них має свої особливості та переваги, які потрібно враховувати при виборі технології для проєкту.

1.2.1 React.js

React.js – це відкрита JavaScript бібліотека, яка використовується для розробки інтерфейсів користувача. Вона дозволяє створювати вебзастосунки, які можуть змінюватися без перезавантаження сторінки. React працює на основі компонентів, які можна створювати знову та знову, щоб забезпечити гнучкість та повторне використання коду.

React дає змогу динамічно оновлювати вміст на сторінці без необхідності перезавантаження всієї сторінки. Крім того, React використовує віртуальний DOM (Document Object Model), що дозволяє швидко оновлювати

інформацію на сторінці та зменшує кількість зайвих операцій з маніпулюванням DOM [3].

Переваги React.js:

– легкість в освоєнні: React простий у використанні та має детальну документацію, яка допомагає розробникам зрозуміти його концепції та особливості;

– гнучкість: React дозволяє легко переносити компоненти з одного застосунку до іншого, що робить його ідеальним для масштабування проєктів;

– широкі можливості: React має велику кількість сторонніх бібліотек та інструментів, що робить його ідеальним для будь-яких проєктів веброзробки;

– швидкість та продуктивність: React дозволяє побудувати швидкі та продуктивні застосунки, що допомагає залучати користувачів.

Недоліки React.js:

– висока складність: хоча React простий у використанні, він має високу складність, яка може призвести до великої кількості багів та проблем;

– розподіленість: React не включає в себе механізми для розподілення застосунків на різних серверах, що може бути проблемою для проєктів з великою кількістю користувачів;

– невідповідність стандартам: React не завжди відповідає стандартам веброзробки, що може бути проблемою для проєктів з високими вимогами до стандартів.

1.2.2 Angular

Angular є фреймворком від компанії Google для розробки вебзастосунків з високою продуктивністю та масштабованістю. Ця

технологія використовує мову програмування TypeScript для розробки клієнтської частини застосунку.

До переваг Angular можна віднести високу швидкість та продуктивність завдяки оптимізації віртуального DOM. Також цей фреймворк має вбудовану підтримку маршрутизації, валідації даних та деяких інших функцій, що спрощують процес розробки. Крім того, Angular є повністю компонентним, що дозволяє створювати чітко визначені та самодостатні компоненти для будь-яких цілей.

Недоліками Angular є складність вивчення та висока поріг входу для початківців, особливо для тих, хто не має досвіду роботи з TypeScript або з іншими фреймворками. Також, у порівнянні з React.js, Angular може бути менш гнучким, що затруднює розширення застосунку в майбутньому.

1.2.3 Vue.js

Vue.js – це прогресивний JavaScript фреймворк для розробки інтерфейсів користувача. Він використовується для створення вебзастосунків та SPA (односторінкових застосунків).

Основні переваги Vue.js:

– легкість вивчення і використання: Vue.js має простий синтаксис і легкий у використанні;

– реактивність: зміни в даних автоматично відображаються в DOM, що дозволяє створювати більш динамічні інтерфейси;

– простота інтеграції: Vue.js може інтегруватися з іншими бібліотеками та фреймворками;

– модульність: Vue.js дозволяє розбивати вебзастосунок на маленькі компоненти для більш легкого управління;

– висока продуктивність: завдяки своїй легкості та реактивності, Vue.js забезпечує високу продуктивність вебзастосунків.

Основні недоліки Vue.js:

– відносно невелика спільнота порівняно з Angular і React;

– недостатній набір інструментів для тестування та підтримки коду;

– відсутність зміни в синтаксисі від версії до версії, що може призвести до проблем при оновленні старих проєктів.

1.2.4 Вибір найкращого фреймворку для клієнтської частини

React.js є одним з найпопулярніших JavaScript фреймворків для розробки вебзастосунків, і він має кілька вагомих переваг, які роблять його привабливим вибором для розробки вебзастосунків для проведення студентських спортивних змагань. React.js надає гнучкість та масштабованість для будь-якої складності застосунків, має велике співтовариство та екосистему з багатьма ресурсами, бібліотеками та інструментами. Використання віртуального DOM дозволяє ефективно обробляти зміни в стані застосунку та проводити швидку перерендерінг. Крім того, React.js надає широкі можливості для тестування. Враховуючи ці переваги, React.js є прекрасним вибором для розробки вебзастосунків для проведення студентських спортивних змагань.

1.3 Огляд технологій серверної частини

Для розробки серверної частини вебзастосунків існує також багато різних фреймворків, які допомагають забезпечити високу продуктивність та ефективність розробки. Найбільш популярні фреймворки для розробки серверної частини на сьогоднішній день – це Java Spring, Python Django, Node.js Express та .NET Core.

1.3.1 Java Spring

Java Spring – це один з найпопулярніших фреймворків для розробки серверної частини вебзастосунків. Він надає широкий функціонал для створення серверних застосунків, таких як автентифікація, робота з базами даних, робота з вебсервісами та багато іншого. Spring також дозволяє використовувати різні технології, такі як JPA, Hibernate, Thymeleaf та інші.

Spring Framework можна розглядати як колекцію менших фреймворків або фреймворків усередині фреймворку (рис. 1.2). Він складається з різних компонентів, кожен з яких має свою спеціалізацію і може бути використаний в багатьох сценаріях розробки. Ці компоненти не залежать один від одного у плані архітектури, що надає гнучкість при використанні фреймворку.

Крім того, Spring Framework має високу рівень інтеграції з багатьма сторонніми системами, що дозволяє легко поєднувати його з іншими інструментами та технологіями. Така інтеграція сприяє покращенню ефективності розробки та забезпечує можливість використання різноманітних розширень та модулів для вирішення різних завдань.

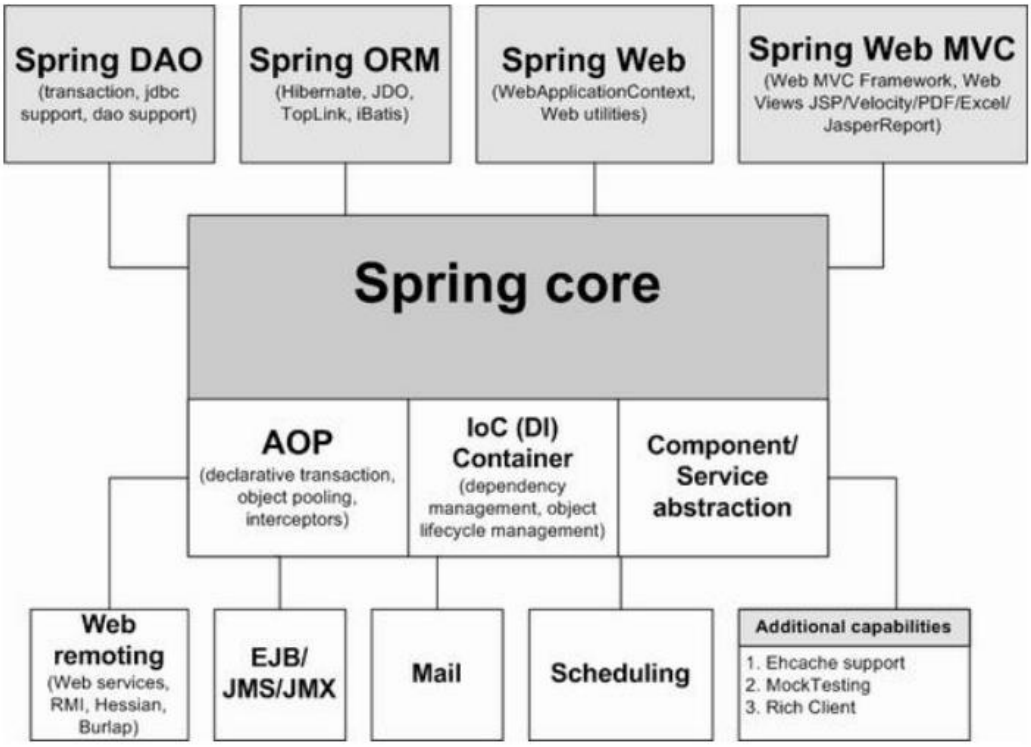


Рисунок 1.2 – Огляд фреймворку Java Spring

1.3.2 Python Django

Python Django – це фреймворк для розробки серверної частини вебзастосунків, який також знаходиться на популярності. Django забезпечує швидку та ефективну розробку вебзастосунків, особливо з використанням шаблонів та моделей. Він має вбудований ORM (Object-Relational Mapping), який дозволяє легко працювати з базами даних, такі як SQLite, MySQL та PostgreSQL. Django також забезпечує підтримку вебсервісів та різних типів медіа-файлів.

1.3.3 Node.js

Node.js – це відкрите середовище виконання JavaScript, яке дозволяє розробникам виконувати JavaScript-код на сервері. Він побудований на базі двигуна V8, який використовується в браузері Google Chrome. Одна з ключових особливостей Node.js – це його асинхронний та подієвий підхід до обробки запитів (рис. 1.3), що дозволяє ефективно обробляти багато запитів одночасно без блокування виконання інших операцій.

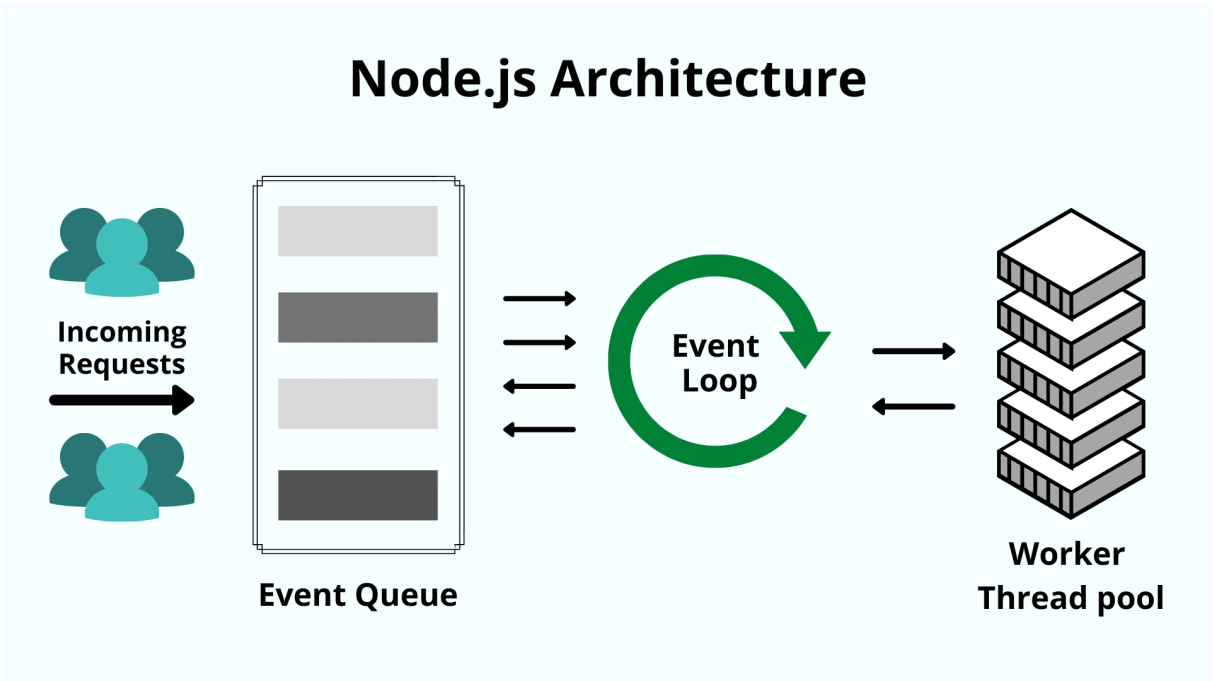


Рисунок 1.3 – Архітектура Node.js

Node.js дозволяє розробникам створювати бекенд застосунки за допомогою JavaScript, що спрощує спільне використання коду між фронтендом та бекендом. Він також має широкий вибір модулів та бібліотек, що полегшують розробку та розширення функціональності.

Node.js підтримує розробку високопродуктивних застосунків, що масштабуються горизонтально, що дозволяє розподіляти навантаження на кілька серверів та забезпечувати високу доступність. Він також володіє потужним набором інструментів для тестування та налагодження, що полегшує розробку та підтримку вебзастосунків.

1.3.4 Вибір найкращої технології для серверної частини

JavaScript як мова програмування є найбільш оптимальною, адже це забезпечує:

– підвищення ефективності розробки завдяки використанню однієї мови для як фронтенд, так і бекенд, а також можливості повторного використання коду;

– використання найбільшого пакетного менеджера npm;

– спрощений пошук розробників, оскільки JavaScript входить до списку найпопулярніших мов програмування.

З огляду на переваги мови JavaScript, згадані вище, Node.js є найбільш вдалим вибором для розробки серверної частини вебзастосунку з наступних причин:

– ефективна обробка багатьох одночасних запитів: Node.js працює на принципі однопотокового, подієвого та неблокуючого вводу/виводу, що дозволяє обробляти багато запитів одночасно без блокування виконання інших операцій. Це робить його ефективним для ситуацій з великим навантаженням і високою швидкістю відповіді;

– широкий вибір бібліотек та модулів: в екосистемі Node.js існує велика кількість готових бібліотек та модулів, які значно спрощують

розробку бекенду. Це дозволяє розробникам ефективно використовувати готові рішення та прискорювати процес розробки;

– спільне використання коду: Node.js дозволяє використовувати один і той самий код на фронтенді та бекенді, що полегшує обмін даними та взаємодію між клієнтом і сервером. Це зменшує зусилля, необхідні для розробки та підтримки застосунків;

– швидкість розробки: Node.js має простий та зрозумілий синтаксис JavaScript, що дозволяє розробникам швидко створювати функціональність бекенду. Крім того, велика кількість готових модулів та фреймворків сприяє прискоренню процесу розробки.

1.4 Огляд технологій для управління базами даних

1.4.1 MySQL

MySQL – це одна з найпопулярніших та використовуваних в світі систем управління базами даних (СУБД). Вона пропонує широкі можливості для зберігання, організації та керування структурованою інформацією в базах даних.

Переваги MySQL:

– надійність та стабільність: MySQL відома своєю надійністю та стабільністю. Вона використовуєся в багатьох критичних застосунках, де надійність даних є вирішальним фактором;

– простота використання: MySQL має добре документовану синтаксис та простий інтерфейс, що полегшує розробку та адміністрування баз даних. Навіть новачки можуть швидко вивчити основи MySQL та почати використовувати його;

– висока продуктивність: MySQL забезпечує швидку роботу з базами даних, особливо при оптимізації запитів та використанні правильних

індексів. Вона може обробляти великі обсяги даних та велику кількість одночасних запитів;

– гнучкість: MySQL підтримує різні типи даних та дозволяє розробникам налаштовувати бази даних залежно від вимог проєкту. Вона підтримує реляційну модель даних та може працювати зі складними зв’язками між таблицями.

Недоліки MySQL:

– складність масштабування: при роботі з дуже великими обсягами даних та високим навантаженням можуть виникнути проблеми з масштабуванням MySQL. Незважаючи на наявність рішень для горизонтального масштабування, це може бути складним завданням;

– обмежена підтримка JSON: хоча MySQL підтримує роботу з JSON-даними, це не є його сильною стороною порівняно з деякими іншими СУБД;

– відсутність деяких сучасних функцій: MySQL може відставати за функціональністю порівняно з деякими іншими СУБД. Наприклад, він може мати обмежену підтримку аналітичних запитів та недостатній набір вбудованих функцій для обробки даних. Однак, це можна компенсувати за допомогою зовнішніх розширень або використанням додаткових інструментів.

Незважаючи на недоліки, MySQL є потужним та надійним рішенням для управління базами даних. Вибір між MySQL та іншими СУБД залежить від конкретних вимог проєкту, його масштабу, швидкодії та комплексності.

1.4.2 PostgreSQL

PostgreSQL – це вільно поширювана об’єктно-реляційна система управління базами даних, є найбільш розвиненої з відкритих СУБД в світі і є реальною альтернативою для комерційних баз даних.

Архітектура PostgreSQL базується на моделі «клієнт-сервер», де клієнти звертаються до сервера PostgreSQL за доступом до бази даних. Внутрішня архітектура PostgreSQL включає ядро (core), що виконує основні операції, та різні модулі розширення, що надають додаткові функції та можливості. PostgreSQL підтримує транзакційний механізм ACID (атомарність, консистентність, ізольованість, довіреність), що забезпечує надійність та цілісність даних.

Переваги PostgreSQL:

– надійність та стабільність: PostgreSQL відомий своєю надійністю та здатністю до оптимальної роботи навіть при великому обсязі даних та високих навантаженнях;

– розширюваність та гнучкість: PostgreSQL має широкий набір вбудованих та сторонніх модулів, які дозволяють розширити його функціональність та придатність для виконання специфічних завдань;

– підтримка географічних та геопросторових даних: PostgreSQL надає вбудовану підтримку для зберігання та обробки географічних та геопросторових даних, що є важливим для багатьох геолокаційних застосунків.

Недоліки PostgreSQL:

– складність налаштування: Налаштування та оптимізація PostgreSQL можуть бути складними завданнями, особливо для непрофесійних користувачів;

– обмежена підтримка реплікації: Хоча PostgreSQL підтримує механізми реплікації, їх налаштування та управління можуть бути складними порівняно з деякими іншими СУБД.

У порівнянні з деякими іншими СУБД, такими як MySQL, PostgreSQL може мати меншу спільноту користувачів та розробників, а також менші ресурси, такі як документація та підручники.

1.4.3 Microsoft SQL Server

Microsoft SQL Server – система керування базами даних (РСУБД), розроблена корпорацією Microsoft. Основний використовуваний мову запитів – Transact-SQL, створений спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI/ISO структурованої мови запитів (SQL) з розширеннями. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства; конкурує з іншими СУБД в цьому сегменті ринку [4].

Розглянемо основні можливості та переваги Microsoft SQL Server:

– SQL Server 7.0 масштабується в діапазоні від портативних комп’ютерів з Windows 95 або Windows 98 до мультипроцесорних кластерів, що працюють під управлінням Windows NT® Server Enterprise Edition;

– SQL Server 7.0 містить заново спроєктований процесор запитів, який забезпечує підтримку баз даних дуже великого обсягу та обробку складних запитів, серед його нових особливостей – використання складових індексів, нові алгоритми хешування і злиття, множинні тригери, а також обробка гетерогенних, розподілених і паралельних запитів [5];

– збільшений до 8 КБ розмір сторінок сприяє швидкому вилученню даних, дозволяє використовувати такі рядки і стовпці більшого розміру, що відкриває можливість ефективного зберігання складних, докладних даних;

– менеджер блокувань динамічно адаптує алгоритм використання ресурсів у великих базах даних, що робить продукт найбільш придатним для інтерактивної обробки транзакцій (online transaction processing – OLTP) і створення сховищ даних;

– виконання багатьох рутинних завдань адміністрування тепер автоматизовано, алгоритми управління пам'яттю і блокуванням адаптуються динамічно, розмір файлів автоматично збільшується і скорочується, крім того, кошти автоматичного налаштування динамічно налаштовують

алгоритми використання ресурсів в залежності від робочого навантаження [6];

– готовність до використання в Інтернеті, інтрамережі і для електронної комерції;

– SQL Server 7.0 підтримує лінгвістичний пошук, дозволяючи створювати спеціальні індекси ключових слів або фраз для обраних стовпців або таблиць;

– SQL Server пропонує широкий спектр можливостей реплікації, що забезпечують автоматичну синхронізацію змін, в тому числі і вироблених в автономному режимі, SQL Server 7.0 підтримує програми, що використовують технологію активних серверних сторінок (Active Server Pages – ASP);

– Access 2000 може безпосередньо звертатися до SQL Server, дозволяючи організувати прозору взаємодію клієнт-сервер [7];

– електронні таблиці, діаграми і зведені таблиці можуть бути безпосередньо пов'язані з SQL Server або службами, що надає користувачам можливості перегляду та аналізу даних за допомогою оглядача.

1.4.4 Вибір найкращої СУБД

Порівняльний аналіз MySQL, PostgreSQL та Microsoft SQL Server [8] показано у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика СУБД

Критерії	**MySQL**	**PostgreSQL**	**Microsoft SQL Server**
1	**2**	**3**	**4**
Відкритість	Так	Так	Ні
Надійність	Висока	Висока	Висока

Продовження таблиці 1.1

1	2	3	4
Масштабованість	Добра	Висока	Добра
Підтримка ACID	Так	Так	Так
Реплікація	Так	Так	Так
Підтримка геопросторових даних	Обмежена	Так	Обмежена
Функціональні можливості	Середні	Високі	Високі
Налаштування та оптимізація	Середня	Складна	Середня
Спільнота користувачів	Велика	Велика	Велика

Враховуючи цю порівняльну таблицю, PostgreSQL видається найкращою технологією для багатьох випадків, зокрема він має відкритий код, високу надійність, масштабованість, підтримку ACID, реплікацію, а також багатий функціонал та широку спільноту користувачів.

1.5 Постановка задачі

Таким чином, розробка вебзастосунку для проведення студентських спортивних змагань, а саме змагань з плавання різного типу, є актуальним завданням для підвищення ефективності організації змагань, розподілення учасників, аналізу та протоколювання результатів як для тренерів і суддів, так і для самих учасників. Цей вебзастосунок буде виконувати широкий спектр функцій, що дозволить автоматизувати багато аспектів організації та управління змаганнями, забезпечуючи зручність для учасників, адміністраторів та глядачів.

Об’єктом роботи є вебзастосунок для проведення студентських спортивних змагань із плавання.

Метою роботи є розробка вебзастосунку для проведення змагань за плавання з можливістю реєструвати заявки спортсменів, організовувати розподіл учасників по доріжкам за допомогою алгоритмів та генерувати протоколи для визначення переможців.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз вимог до інформаційної системи;

– розробити інформаційну систему, визначити бізнес правила та ролі системи;

– реалізувати можливість реєструвати заявки спортсменів та авторизувати суддів і тренерів;

– розробити та реалізувати алгоритм генерації стартового протоколу;

– реалізувати зручний інтерфейс для головного судді, щоб він швидко міг ввести зареєстрований час кожного спортсмена у кожній дисципліні;

– реалізувати алгоритм генерації підсумкових (або фінальних) протоколів для визначення переможців.

2 МОДЕЛЮВАННЯ СИСТЕМИ ТА ОПИС АЛГОРИТМІВ

2.1 Проєктування інформаційної системи

Проєктування системи – це процес створення архітектури та деталей системи з метою вирішення конкретної задачі або задоволення потреби користувачів. Цей процес включає кроки аналізу вимог, проєктування структури бази даних, розробку функціональності системи, тестування, впровадження та підтримку [9].

Основні етапи проєктування інформаційної системи:

− аналіз вимог: встановлення вимог до системи шляхом спілкування зі зацікавленими сторонами, вивчення бізнес-процесів та потреб користувачів. Результатом цього етапу є формулювання функціональних та нефункціональних вимог до системи;

− проєктування архітектури: розробка загальної структури системи, визначення компонентів, модулів та їх взаємозв’язків. На цьому етапі також вирішуються питання про технології, платформи та інфраструктуру, які будуть використовуватися;

− проєктування компонентів: процес створення детального опису кожного компонента системи, включаючи його функціональність та інтерфейси. На цьому етапі визначається, як кожен компонент буде виконувати свої завдання та взаємодіяти з іншими компонентами;

− проєктування бази даних: створення схеми бази даних, включаючи таблиці, поля та зв’язки між ними. Враховується необхідність зберігання та організації даних, що відповідають вимогам системи;

− проєктування інтерфейсу: процес розробки користувацького інтерфейсу системи, включаючи його організацію, навігацію та взаємодію з користувачем. Головною метою проєктування інтерфейсу є створення простого використання та зручного середовища для користувача. Під час проєктування інтерфейсу враховуються такі аспекти, як розташування

елементів на екрані, організація меню та навігаційних елементів, використання підказок та повідомлень для користувача, а також забезпечення зручної взаємодії через кнопки, поля введення, списки тощо. Важливо, щоб інтерфейс був легким у сприйнятті, інтуїтивно зрозумілим та відповідав потребам та очікуванням користувача;

– розробка функціональності: розробка програмного коду, який виконує необхідні операції та функції системи. Цей етап включає розробку різних модулів, компонентів, інтерфейсів та бізнес-логіки системи;

– тестування та валідація: перевірка правильності роботи системи, виявлення та виправлення помилок. Тестування може включати функціональне тестування, тестування продуктивності, безпеки та інші види тестування для переконання в якості системи перед її впровадженням;

– впровадження: впровадження системи в робоче середовище. Цей етап включає установку, налаштування та інтеграцію системи, навчання користувачів, перехід від попередньої системи до нової.

Проєктування системи – це складне завдання, що вимагає комплексного підходу та розуміння потреб користувачів, технічних обмежень і принципів дизайну та архітектури. Цей процес включає глибокий аналіз, творчий підхід і розробку оптимальних рішень для досягнення поставлених цілей проєкту. Крім того, враховується майбутній розвиток та масштабування системи, щоб забезпечити її ефективність та гнучкість. Також велику увагу приділяється безпеці даних та захисту системи від потенційних загроз. Весь цей процес вимагає від команди проєктувальників глибоких знань, аналітичних навичок та технічної експертизи для успішної реалізації інформаційної системи.

Кожен етап проєктування інформаційної системи має свої завдання та вимоги, і вони часто взаємодіють між собою. Цей процес вимагає аналітичних, проєктних та програмних навичок, а також комунікації з різними сторонами, щоб забезпечити успішну розробку та впровадження інформаційної системи [10].

2.1.1 Специфікація вимог

Аналіз вимог системи є ключовим етапом у розробці будь-якого проєкту. Він дозволяє зрозуміти потреби та очікування користувачів і визначити, які функціональні та нефункціональні вимоги повинна виконувати система. Нефункціональні вимоги відіграють важливу роль у визначенні якості, безпеки, ефективності та надійності системи.

Таким чином, приступаючи до аналізу вимог для даної системи проведення студентських спортивних змагань, зокрема плавання, необхідно врахувати потреби користувачів, їх очікування та вимоги до функціональності.

Інформаційна система повинна дозволяти виконувати наступні функції:

а) створення змагання і програму цього змагання, яка включає будь-які дисципліни і в який день змагання будуть проводиться;

б) створення команди і прив’язувати їх до тренерів;

в) створення спортсменів і реєстрація їх на участь в змаганнях від однієї з команд;

г) генерація стартового протоколу для кожного змагання на основі даних про реєстрацію спортсменів в ті чи інші дисципліни;

д) можливість ручного коректування стартового протоколу головним суддею змагання;

е) введення результатів (підсумкового часу) для кожного спортсмена для кожного його запливу;

ж) генерація фінальних протоколів, які визначають переможця;

з) відстеження результатів спортсменів і аналітика їх результатів в розрізі різних часових відрізків;

и) редагування довідкових даних і реєстрація нових користувачів в системі.

В інформаційній системі повинно існувати 3 ролі користувачів:

– адміністратор: доступні функції – 1, 2, 5, 9;

– тренер: доступні функції – 3, 8;

– суддя: доступні функції – 1, 4, 5, 6, 7.

Ролі користувачів та їхні функції зображені на рисунку 2.1.

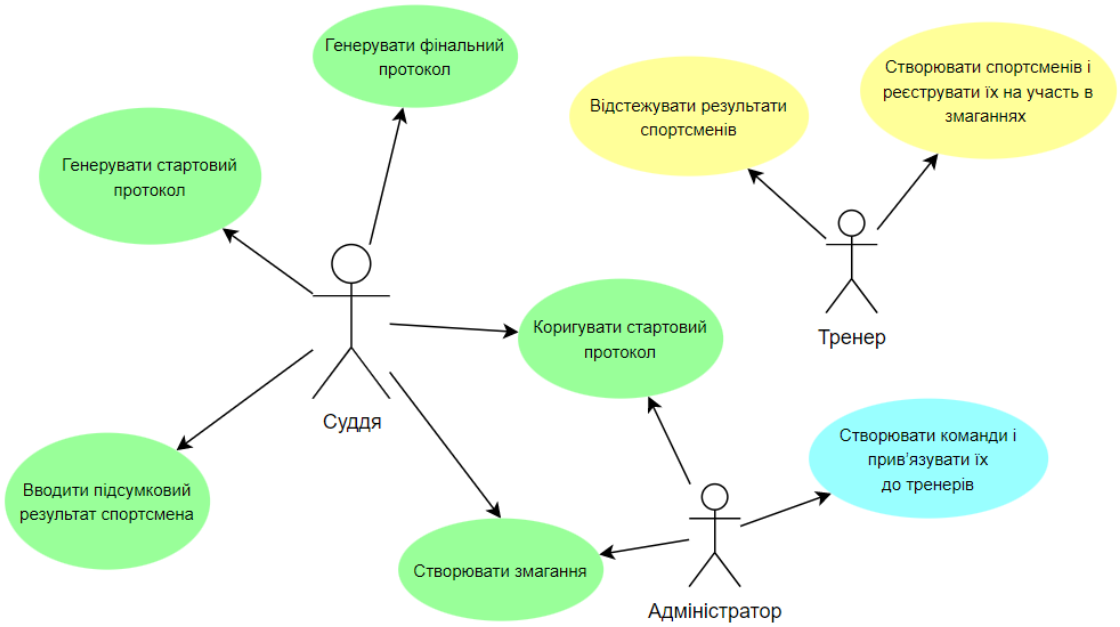


Рисунок 2.1 – Ролі та функції користувачів

2.1.2 Розробка бізнес правил БД

Бізнес правила:

– інформація про змагання містить ім’я, дату початку і програму змагання;

– кожне змагання може бути відкрито для реєстрації спортсменів, а може бути закрито за певний час до початку змагання;

– змагання може проводитися кілька днів;

– у кожен день змагання може проводиться кілька дисциплін, які суддя може виставляти у вільній послідовності;

– у кожної дисципліни є своя назва і список нормативів, які враховуються при підрахунку фінальних (підсумкових) протоколів;

– базова інформація про кожного спортсмена містить: ПІБ, дату народження, стать, країну народження;

– базова інформація про тренера містить ПІБ;

– базова інформація про команду – це її назва і тренер, який тренує цю команду;

– при реєстрації на змагання спортсмена для кожної дисципліни, в яких цей спортсмен братиме участь, тренер вказує заявлений час, категорію спортсмена;

– при формуванні стартового протоколу за допомогою алгоритму, який враховує заявлений час, кожному спортсмену в кожній дисципліні в рамках змагання присвоюється номер запливу і номер доріжки, на якій він буде плисти;

– при розрахунку фінальних протоколів використовується результат спортсмена (підсумковий час) в кожній дисципліні з урахуванням типів рейтингів, в яких він брав участь;

– результати кожного змагання повинні зберігатися і повинні доступні в будь-який момент для перегляду Тренером або Суддею, а також для повторної генерації фінальних протоколів;

– у рамках змагання в кожній дисципліні спортсмен може виступати за різні команди, але в рамках однієї дисципліни тільки за одну;

– фінальні (підсумкові протоколи) не повинні зберігатися в базі даних, вони кожен раз генеруються за запитом.

2.1.3 Проєктування схеми бази даних

База даних є центральною складовою будь-якої інформаційної системи. Вона зберігає структуровані дані, які використовуються для збереження, організації та отримання інформації. Бази даних використовуються для збереження великих обсягів даних і забезпечення швидкого та ефективного доступу до них.

У даній системі планується використовувати PostgreSQL. PostgreSQL є популярною вільною об’єктно-реляційною системою управління базами даних.

PostgreSQL дозволяє нам зберігати дані про спортсменів, їхні дані, участь у змаганнях, результати, команди, протоколи та інші важливі дані, необхідні для роботи даної системи. Вона підтримує швидкий доступ до даних, запити і індексацію, що дозволяє ефективно виконувати операції з базою даних.

Крім того, використання PostgreSQL дозволяє нам працювати з гнучкою схемою даних. Ми можемо додавати нові поля до документів без необхідності зміни всієї структури бази даних. Це дає нам можливість легко розширювати функціональність системи і адаптуватися до змін вимог.

Загалом, база даних PostgreSQL допомагає нам ефективно зберігати і керувати даними в даній системі, забезпечуючи швидкий доступ та гнучкість в роботі з інформацією [11, 12].

На основі специфікацій і бізнес правил була побудована логічна і концептуальні схеми бази даних для вебзастосунку. Ці схеми допомогли визначити структуру даних, яку необхідно буде зберігати для оптимальної роботи вебзастосунку. Були також враховані взаємозв’язки між різними сутностями і визначені ключові поля для кожної сутності (рис. 2.2).

Були визначені всі необхідні ключі, включаючи первинні, і типи сутностей та зв’язків у базі даних були визначені за допомогою концептуальної моделі в синтаксисі Чена. При перетворенні концептуальної моделі в логічну модель не виникло потреби внесення змін, оскільки не було складних або рекурсивних зв’язків або надмірності інформації. Логічна модель була перевірена згідно з правилами нормалізації баз даних і потім перенесена на логічну ER-модель. Оскільки модель не містила складних або рекурсивних зв’язків, ER-модель була побудована на папері, без використання спеціального програмного забезпечення.

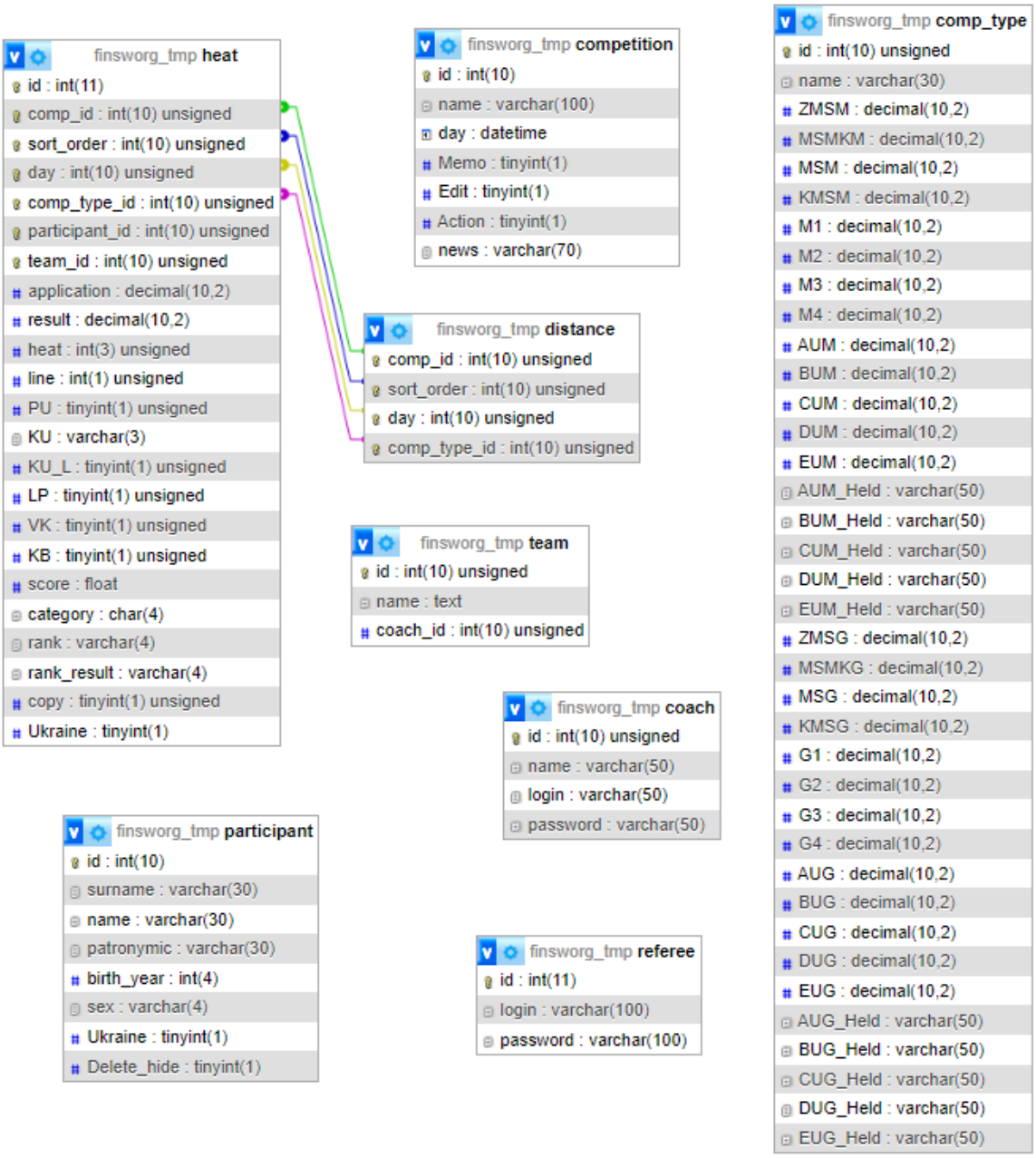


Рисунок 2.2 – Схема бази даних для вебзастосунку

2.2 Архітектура системи

Проєктування архітектури системи для вебзастосунку з проведення спортивних змагань є критичним етапом, який охоплює розробку структури, компонентів і їх взаємодії. Враховуючи описані сторінки та їх

функціональність, ми можемо розглянути загальну структуру архітектури системи.

Архітектура даної системи може бути заснована на клієнт-серверній моделі, де вебсайт виступає як клієнт, а сервер є головним компонентом системи, відповідальним за обробку запитів, зберігання даних та взаємодію з базою даних [13].

Клієнтська частина:

– фронтенд: він включає в себе користувацький інтерфейс, який дозволяє користувачам переглядати розклади змагань, реєструватися на них, переглядати результати та виконувати інші дії пов’язані зі спортивними змаганнями. В даній роботі використано React разом з Next.js для побудови користувацького інтерфейсу. Next.js надасть можливість серверного рендерингу (SSR) та статичну генерацію (SSG) сторінок для покращення продуктивності та SEO;

– бібліотека компонентів: використання Material-UI для швидкої розробки стилізованих інтерфейсів. Material-UI надає багато готових компонентів та можливостей для стилізації;

– керування станом: використання бібліотеки Redux для керування станом даних та управління запитами до сервера. Redux дозволяє ефективно кешувати дані та автоматично оновлювати їх при необхідності.

Серверна частина:

− бекенд: він забезпечує обробку запитів від клієнта. Бекенд включає в себе сервер, який отримує запити від клієнта, взаємодіє з базою даних та обробляє дані для надання відповіді клієнту. В серверній частині реалізовані різні компоненти, такі як маршрутизатори, контролери та сервіси, які забезпечують логіку застосунку та взаємодію з базою даних; використання Node.js для побудови серверної частини застосунку;

− вебфреймворк: використання Express для реалізації вебсервера та обробки запитів;

− база даних: використання PostgreSQL для зберігання даних.

Комунікація між клієнтом та сервером в проєкті може бути забезпечена наступними засобами:

– API для взаємодії: для забезпечення обміну даними між клієнтом та сервером можна використовувати REST або GraphQL API. REST надає стандартні API-ендпоінти для виконання різноманітних операцій, тоді як GraphQL дозволяє клієнту запитувати лише необхідні дані;

– захист API: для забезпечення безпеки та захисту даних користувачів використовуються механізми авторизації та контролю доступу до API. Це включає в себе використання аутентифікації, де користувачі підтверджують свою ідентичність, і авторизації, де визначається, які ресурси та операції можуть бути доступні для конкретного користувача. Застосування правильних механізмів захисту допомагає запобігти несанкціонованому доступу до API та зловживанням даними.

Інші технології:

– типізація: використання Typescript для покращення роботи з кодом, забезпечення типізації та зменшення можливих помилок;

– тестування: використання фреймворку для тестування, такого як Jest, для написання автоматизованих тестів, які перевірятимуть функціональність та стабільність системи.

Архітектура системи була спроєктована з урахуванням декількох аспектів, щоб забезпечити ефективну роботу, масштабованість та зручність для користувачів. Використання сучасних технологій та передових практик гарантує розробку стабільної та функціональної системи, яка повністю задовольняє потреби користувачів у зручній та доступній організації спортивних студентських змагань. Враховуючи цільові аудиторію та їх вимоги, були обрані оптимальні рішення, що сприяють зручності використання та покращенню користувацького досвіду. Результатом є система, яка готова задовольнити потреби користувачів та забезпечити успішну організацію спортивних змагань студентів.

2.2.1 Опис сторінок та основних компонентів

Створення сторінок для сайту – це процес розробки та реалізації вебсторінок, які відображають контент та функціонал сайту. Це включає в собі створення дизайну та макету сторінок, розміщення контенту, використання різних елементів інтерфейсу, таких як заголовки, текстові блоки, зображення, кнопки та форми введення даних.

Під час розробки сторінок, важливо враховувати потреби користувачів, забезпечити легку навігацію та зручний користувацький досвід. Розміщення контенту має бути логічним та інтуїтивно зрозумілим, щоб користувачі легко знайшли потрібну інформацію. Важливо також використовувати привабливий та сучасний дизайн, що відповідає бренду або концепції сайту.

Крім того, сторінки потрібно оптимізувати для пошукових систем, використовуючи відповідні метатеги, ключові слова та оптимізований контент. Це допоможе забезпечити більшу видимість сайту в пошукових системах і залучити більше відвідувачів. Також важливо перевірити сумісність сторінок з різними браузерами та пристроями, щоб забезпечити доступність сайту для широкої аудиторії.

В цілому, створення сторінок сайту вимагає комбінації технічних та дизайнерських навичок, а також уважного планування та тестування. Від якості розроблених сторінок залежить ефективність та привабливість сайту для користувачів, що впливає на загальний успіх проєкту [14].

Зважаючи на опис функціональних та нефункціональних вимог проєкту, можна створити систему з наступними компонентами: «Логін», «Панель Адміністратора», «Інтерфейс тренера» та «Заявки та результати», «Панель судді», «Редагування програми змагань», «Стартовий протокол».

На сторінці «Логін» користувач може ввести свій логін та пароль, після чого система перевіряє введені дані та здійснює аутентифікацію користувача. Залежно від вхідних даних система може визначити одну з трьох ролей користувача: Адміністратор, Суддя чи Тренер. Ця сторінка створена для того,

щоб користувачі могли ввійти до системи шляхом введення своїх облікових даних. Вона перевіряє достовірність введених даних і надає доступ до особистого облікового запису. У разі успішного входу, користувач перенаправляється на відповідну сторінку до своєї ролі.

«Панель адміністратора». В рамках своєї панелі керування, діаграма якої зображена на рисунку 2.3, адміністратор повинен мати можливість редагувати будь-які записи бази даних через простий, зрозумілий і відповідає вимогам безпеки інтерфейс. Сюди також відноситься і редагування довідкових даних (рекорди, нормативи, тощо), а також додавання/видалення нових користувачів (суддів і тренерів) в вебзастосунок.

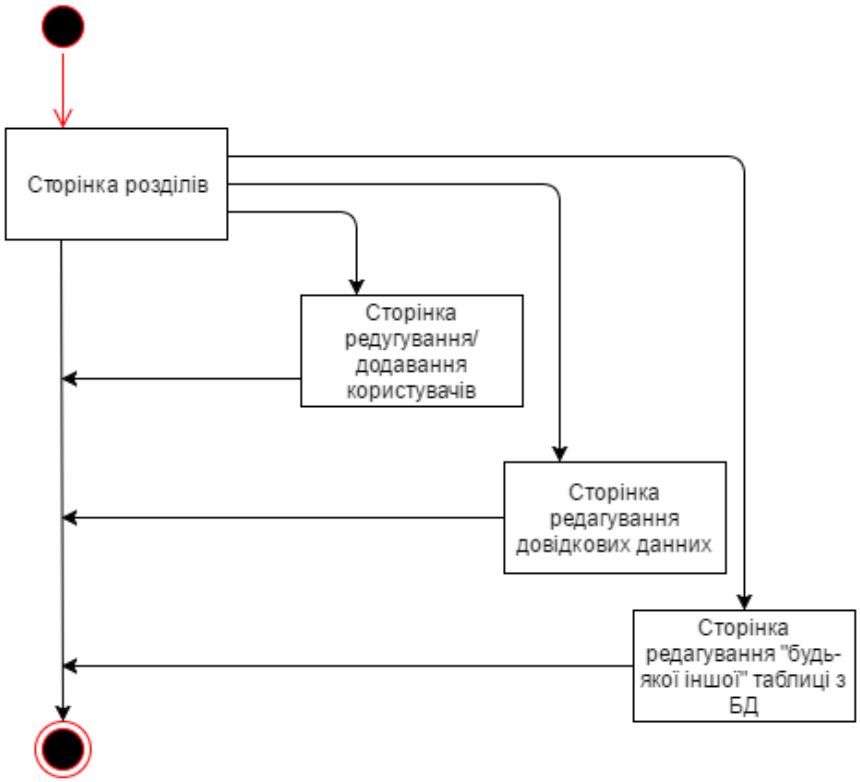


Рисунок 2.3 – Діаграма сторінки Адміністратора

«Інтерфейс Тренера». В рамках своєї панелі керування тренер повинен мати можливість:

– додавати спортсменів в систему;

– реєструвати спортсменів на змагання;

– переглядати заявки по всім спортсменам за всіма своїми командам по кожному доступному змагання;

– переглядати результати по всім спортсменам за всіма своїми командам по кожному доступному змагання;

– мати інтерфейс відомості даних по своїм командам, спортсменам і їх результатів в зручному для аналізу вигляді.

На основі необхідного функціоналу була побудована діаграма переходу між сторінками панелі Тренера, показана на рисунку 2.4.

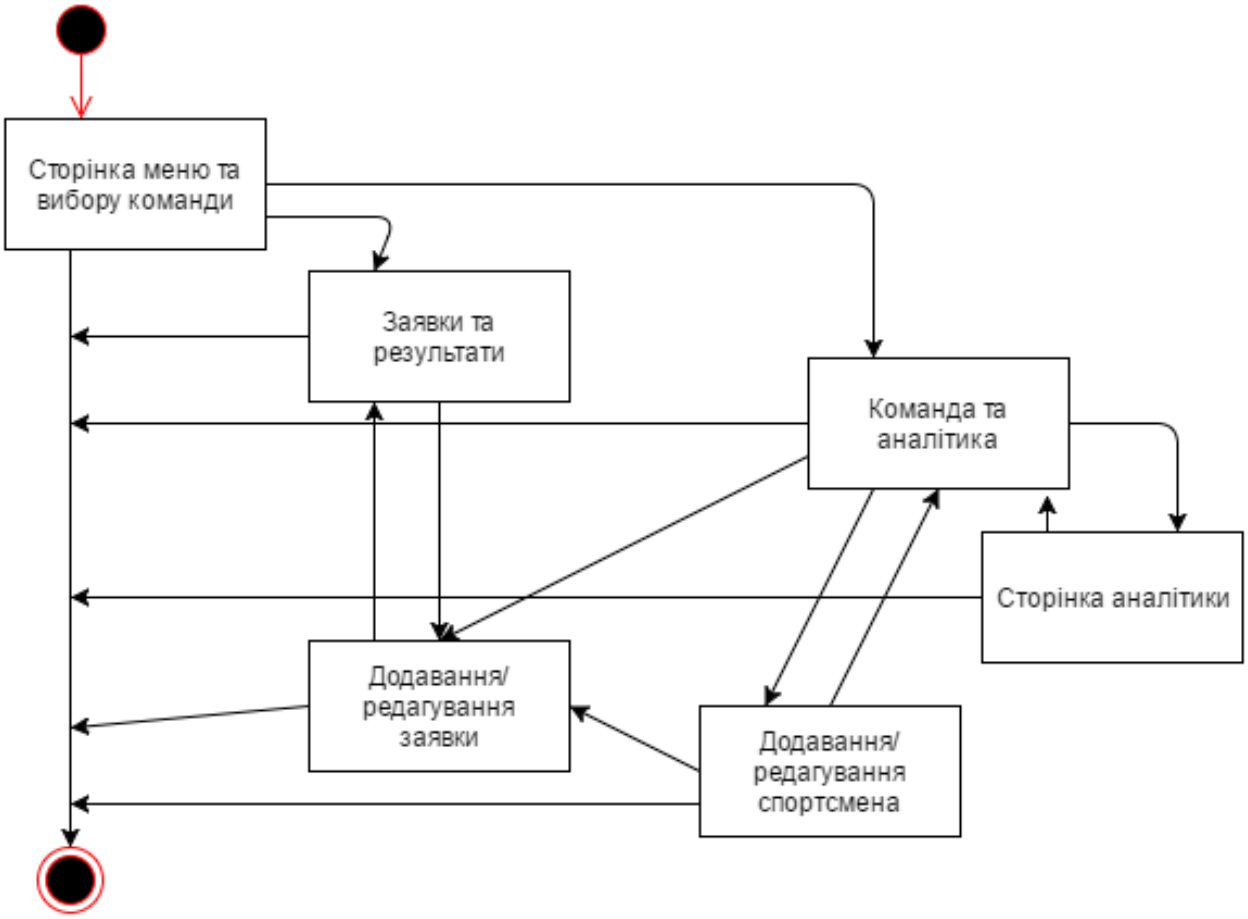


Рисунок 2.4 – Діаграма сторінки Тренера

«Заявки та результати». Стартова сторінка для всіх тренерів – це сторінка заявок і результатів. Тому саме з цієї сторінки почалася промальовування прототипу. А завдяки діаграмі переходу між сторінками було легко визначити, які функціональні елементи повинні знаходиться в тому чи іншому розділі і прототип якої сторінки потрібно малювати наступним після поточного. Нижче наведено приклад прототипу стартової сторінки в панелі Тренера.

«Панель судді». В рамках своєї панелі керування, діаграма якої зображена на рисунку 2.5, суддя повинен мати можливість:

– створювати і редагувати основну інформацію про змагання (назва, дата початку, тощо);

– запускати автоматичну генерацію стартового протоколу;

– створювати і редагувати для кожного змагання програму, що складається з дисциплін, які є в довідкових таблицях створеної бази даних, та днів, у які ці дисципліни будуть проводитись;

– редагувати вручну стартових протокол (переставляти спортсменів в інші запливи та/або на інші доріжки);

– вводити результати (підсумковий час) для кожного спортсмена;

– запускати автоматичну генерацію фінальних протоколів.

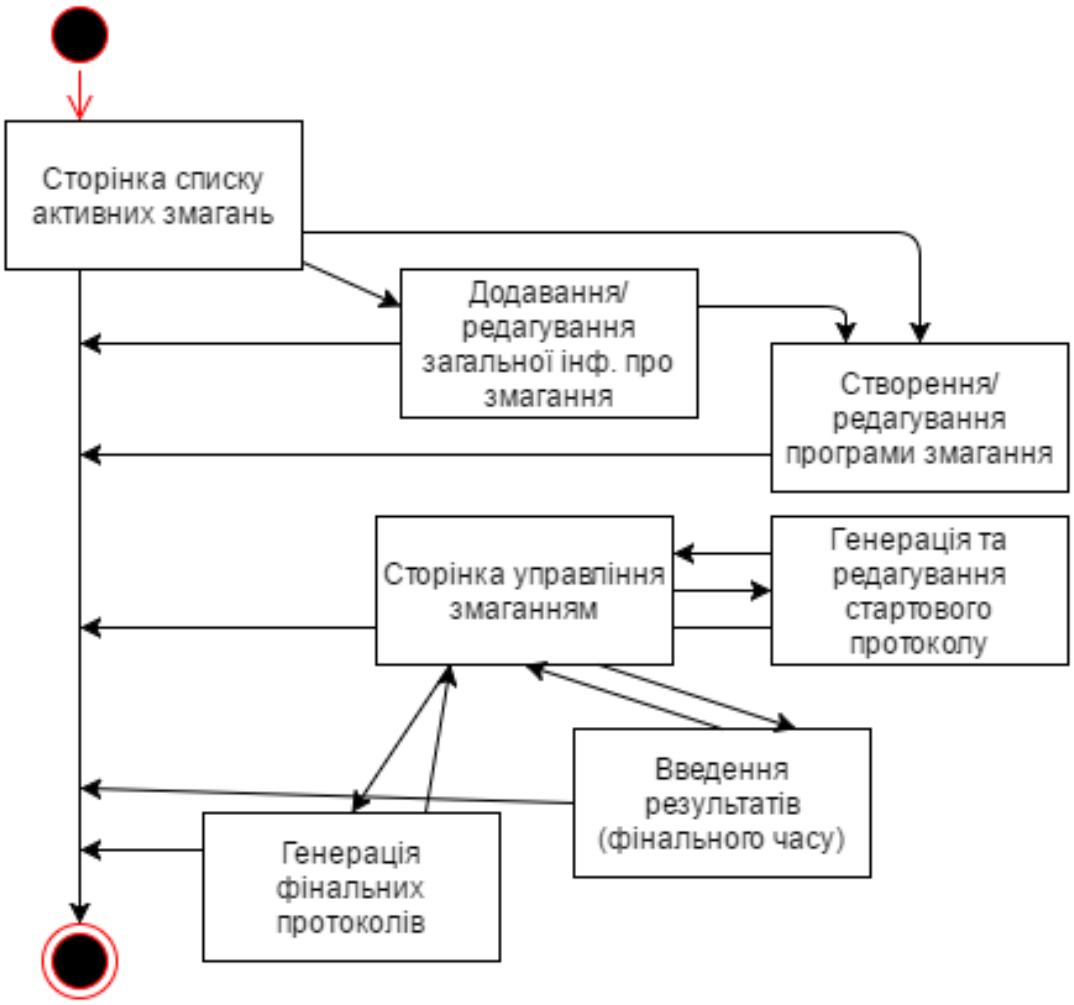


Рисунок 2.5 – Діаграма сторінки Судді

Важливим розділом панелі судді є аналітика. В рамках кваліфікаційної роботи не було завдання реалізувати можливість глибокої аналітики, але необхідно було закласти фундамент для подальшого розвитку проєкту та показати, що такий розділ, як аналітика також спростить життя кінцевим користувачам.

У вебзастосунок була впроваджена можливість перегляду інформації по кожному спортсмену в розрізі часу і різних дисциплін. Це означає, що кожен тренер може по кожному своєму спортсмену подивитися зведені таблиці і графіки, які покажуть окремо по кожній дисципліні:

– який заявлений час і реальний фінальний час були у спортсмена на кожному змаганні;

– на скільки фінальний час відрізняється від заявленого часу;

– як змінюється фінальний час від змагання до змагання;

– на скільки виступи даного спортсмена в даній дисципліні відповідають планам тренера (якщо у вибірці змагань 50% і більше виступів завершились з рівним (або вище) заявленим часом, то вважається, що спортсмен виконує «план тренера» в даній дисципліні).

Після практичного тестування кінцевими користувачами, буде також братися зворотний зв’язок по розділу аналітики з метою дізнатися, які дані тренерам цікаво переглядати і в якому розрізі, враховуючи, що система може генерувати зрізи, таблиці та графіки тільки по тій інформації, яка є в базі даних вебзастосунку.

2.2.2 Структурна мапа вебсайту

Карта сайту – це візуальне представлення структури вебсайту. Вона показує ієрархічні зв’язки між різними сторінками або розділами сайту. Карта сайту допомагає організувати і систематизувати вміст, навігацію та структуру вебсайту. Зазвичай карта сайту відображає сторінки на рівні

дерева, де головна сторінка є коренем, а наступні сторінки розміщуються як гілки чи листочки. Це дозволяє візуально представити розташування сторінок і їх взаємозв'язки. Мапа сайту допомагає розуміти загальну структуру сайту, забезпечує зручну навігацію для користувачів і полегшує роботу з розробниками і дизайнерами при створенні і підтримці вебсайту [3].

У даному проєкті також була підготовлена карту сайту, яка допоможе нам краще зорієнтуватись у структурі даної системи. Карта сайту включає всі основні сторінки та функціональні елементи, що присутні на вебсайті. Вона надає нам відображення загального зображення системи та розташування її компонентів. Подальше представлення карти сайту дозволить нам крок за кроком розглянути кожну сторінку, її функціональність та взаємозв'язки з іншими сторінками. Це допоможе нам краще зрозуміти внутрішню логіку системи та взаємодію між її компонентами.

На рисунку 2.6 зображена мапа поточного вебзастосунку з організації та проведення студентських спортивних змагань.

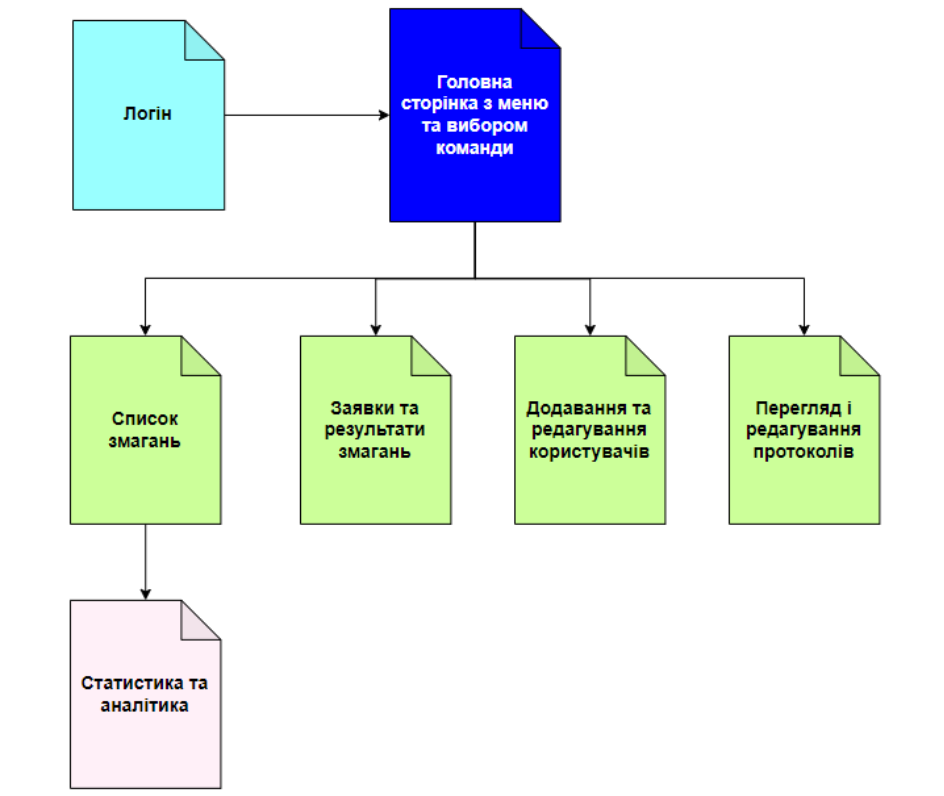


Рисунок 2.6 – Мапа вебсайту

2.3 Опис основних алгоритмів системи

2.3.1 Алгоритм генерації стартового протоколу

Перед початком змагання Суддя генерує «Стартовий протокол» [15] (формування запливів відповідно до алгоритму розстановки по доріжках). Цей протокол генерується на основі алгоритму, який словами по пунктам описаний нижче:

Крок 1. Спочатку йде розбивка по днях. У кожному дні запливи нумеруються від 1 до *n*.

Крок 2. В рамках однієї дисципліни спочатку йдуть запливи Жінок, потім Чоловіків.

Крок 3. Спочатку йдуть запливи категорії «E», потім «D», потім «C», «B», «A».

Крок 4. У кожній категорії «найсильніший заплив» (з кращим заявленим часом) повинен бути повністю заповнений (на кожній з 8 доріжок хтось пливе).

Крок 5. Спортсмени розставляються по доріжках від кращого заявленого часу до гіршого в наступній послідовності: [4, 5, 6, 3, 2, 7, 1, 8]. Тобто спортсмена з найкращим (найсильнішим) заявленим часом ми ставимо на четверту доріжку, спортсмена з заявленим часом трохи гірше на п’яту доріжку і т.д.

Крок 6. У випадку, коли в одній категорії в рамках однієї дисципліни кількість спортсменів перевищує восьмеро, необхідно застосовувати спеціальне розподілення на запливи. Відповідно до цього правила, якщо, наприклад, в категорії виступає 10 спортсменів, то дев’ятий і десятий учасники (враховуючи їх заявлений час) будуть розташовані на 4-й і 5-й доріжках в сусідньому запливі. Такий підхід гарантує справедливість та оптимальне використання доріжок у випадках, коли кількість учасників перевищує обмеження.

2.3.2 Алгоритм генерації фінального протоколу

У підводному швидкісному плавання, для якого даний вебзастосунок був розроблений, існує кілька типів фінальних протоколів:

– кубок України;
– першість України;
– особистий залік;
– командна боротьба;
– поза конкурсом.

Всі вони були успішно реалізовані в рамках кваліфікаційної роботи. При цьому кожен протокол являє собою окремо працюючий алгоритм та окрему сторінку для демонстрації результатів змагань.

Протокол «Кубок України» є єдиним з усіх протоколів, який має свої особливості і рахується трохи інакше, ніж інші типи протоколів, тому що кожний спортсмен отримує певну кількість очок залежно від свого місця. У таблиці 2.1 вказані місця і відповідні їм очки. Очки отримують лише перші 20 місць, інші спортсмени у рамках «Кубку України» отримують по 0 очок.

Таблиця 2.1 – Розподіл очок для протоколу Кубку України

Місце	**Очки**	**Місце**	**Очки**
1	25	11	10
2	22	12	9
3	20	13	8
4	18	14	7
5	16	15	6
6	15	16	5
7	14	17	4
8	13	18	3
9	12	19	2
10	11	20	1

Алгоритм генерації підсумкового протоколу для «Кубка України»:

Крок 1. Для кожної дисципліни ми розбиваємо всіх учасників на «Дівчат», «Юнаків», «Жінок» та «Чоловіків».

Крок 2. У кожній категорії робиться сортування за «Результатом» (фінальним часом спортсмену) та розподіляються бали відповідно до таблиці 2.1.

Крок 3. Для кожного спортсмена виконується перевірка відповідності фінального часу до розряду або звання, яке вказав тренер на етапі подачі заявки.

Крок 4. Для кожного фінального часу проводиться перевірка на всеукраїнський рекорд відносно тих даних, які є в довідковій таблиці.

Алгоритм генерації підсумкового протоколу для всіх інших типів протоколів, крім «Кубка України» працює однаково. Тому для них усіх інших протоколів реалізований єдиний спосіб роботи описаний далі:

Крок 1. Для кожної дисципліни ми розбиваємо усіх учасників за категоріями «E», «D», «C», «B», «A».

Крок 2. Всередині категорій розбиває всіх учасників на «Жінок» та «Чоловіків».

Крок 3. У кожній категорії робиться сортування за «Результатом» (підсумковим фінальним часом) за спаданням, але балів, які розподілялися в «Кубку України», вже немає.

Крок 4. Для кожного спортсмена йде перевірка відповідності фінального часу до розряду або звання, яке вказав тренер на етапі подачі заявки.

Крок 5. Для кожного фінального часу проводиться перевірка на всеукраїнський рекорд відносно тих даних, які є в довідковій таблиці.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ДЛЯ ПРОВЕДЕННЯ СТУДЕНТСЬКИХ СПОРТИВНИХ ЗМАГАНЬ

3.1 Обґрунтування вибору технологій та середовища програмної реалізації

У рамках кваліфікаційної роботи було обрано наступні технології: React.js, Node.js Express, PostgreSQL. Для середовища програмної реалізації було обрано JetBrains WebStorm (рис. 3.1) – інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблене на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей.

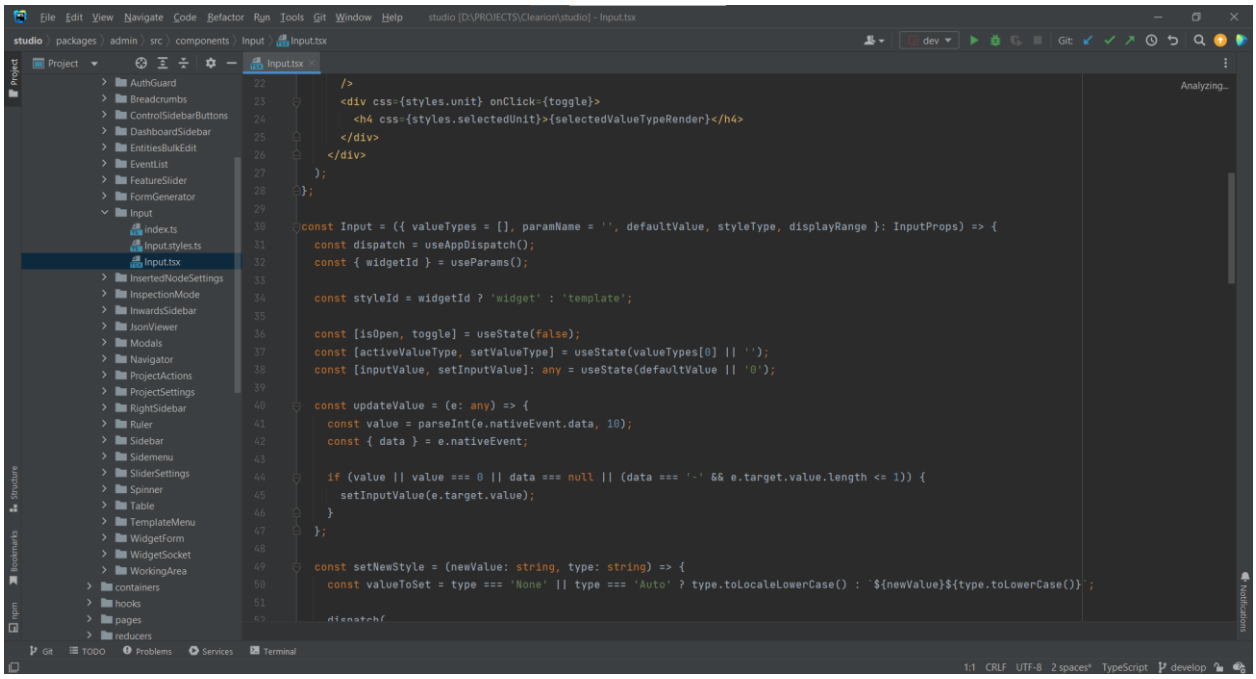


Рисунок 3.1 – Приклад інтерфейсу IDE Webstorm

Для збереження інформації про спортсменів, категорії змагань та створені протоколи змагань буде використана СУБД PostgreSQL, тому що вона має потужні і надійні механізми транзакцій і реплікації, а також легку розширюваність.

Express є фреймворком для розробки серверної частини застосунків на Node.js. Він допомагає встановити маршрутизацію, обробку запитів та створення API для взаємодії з базою даних та передачі даних між клієнтом і сервером.

React.js є бібліотекою для розробки користувацького інтерфейсу на клієнтській стороні. Він дозволяє створювати ефективні, інтерактивні та модульні користувацькі інтерфейси з використанням компонентного підходу. React робить дану систему більш динамічною та відзивчивою, забезпечуючи багатофункціональний та зручний інтерфейс для користувачів.

Node.js є середовищем виконання JavaScript, яке дозволяє запускати JavaScript-код на сервері. Він дозволяє створювати серверну частину застосунків, обробляти запити, взаємодіяти з базою даних та забезпечувати функціональність, необхідну для роботи даної системи.

Обрані технології дозволяють створити повноцінний застосунок, в якому можливо взаємодіяти з базою даних, обробляти запити, створювати динамічний інтерфейс та забезпечувати потрібну функціональності для користувачів системи.

PostgreSQL забезпечує нам гнучкість та швидкий доступ до даних, Express допомагає створити потрібні маршрути та API для обробки запитів, React дає змогу створювати інтерактивні та зручні користувацькі інтерфейси, а Node.js забезпечує зв’язок між фронтендом та базою даних, обробку запитів та виконання логіки на серверній стороні. Комбінація цих технологій дозволяє нам створити ефективний застосунок, який задовольняє потреби користувачів та забезпечує зручну та роботу з даними [16-19].

Також для спрощеної розробки вебзастосунку було розглянуто вебплатформу cPanel [20]. cPanel є комерційною вебхостинговою панеллю керування, розробленою компанією cPanel, L.L.C. Вона пропонує широкий спектр функціональних можливостей, що дозволяють власникам вебсайтів і адміністраторам серверів ефективно керувати своїми хостинговими акаунтами.

Основні можливості cPanel включають:

– керування файлами: cPanel надає файловий менеджер, який дозволяє завантажувати, видаляти, переміщувати та редагувати файли на сервері безпосередньо через вебінтерфейс;

– бази даних: cPanel підтримує керування реляційними базами даних, такими як MySQL та PostgreSQL. Можна створювати бази даних, керувати користувачами, виконувати резервне копіювання та відновлення даних;

– пошта: cPanel надає інструменти для керування електронною поштою. Можна налаштовувати поштові скриньки, пересилати пошту, налаштовувати захист від спаму та використовувати вебпошту;

– установка CMS та інших програм: cPanel має модуль Softaculous, який дозволяє швидко та легко встановлювати популярні вебзастосунки, такі як WordPress, Joomla, Drupal та багато інших (рис. 3.2);

– налаштування сервера: cPanel дозволяє настроювати параметри сервера, такі як PHP-налаштування, доступ до журналів сервера, налаштування SSL-сертифікатів та багато іншого;

– статистика вебсайту: cPanel надає статистику вебсайту, включаючи інформацію про відвідуваність, трафік, використання ресурсів та інше.

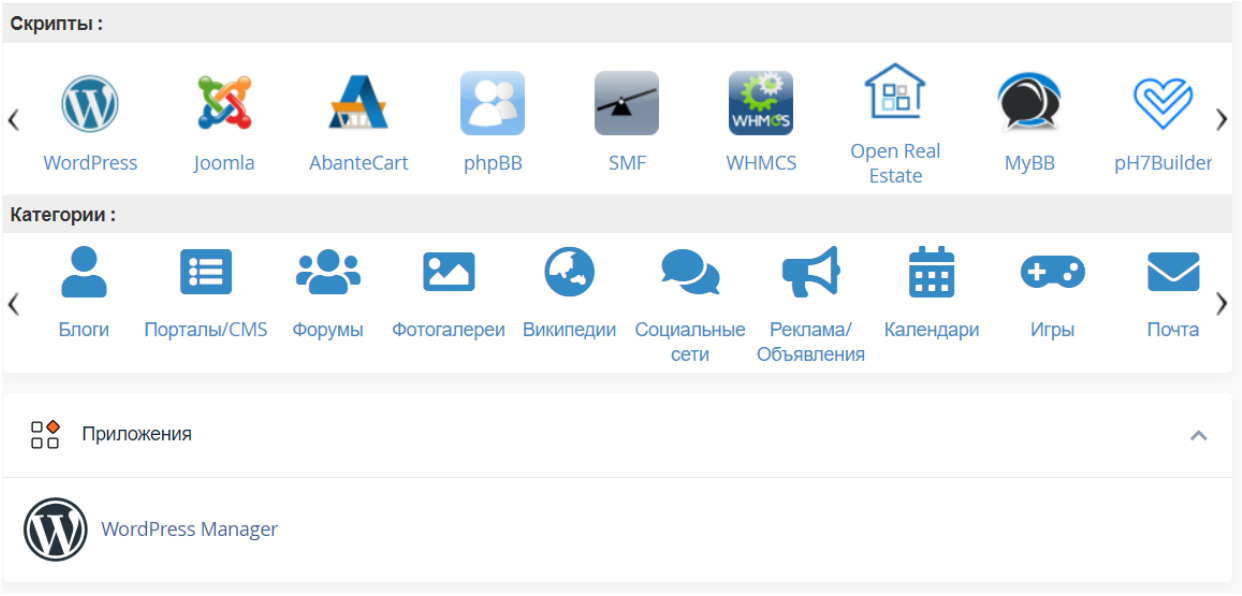


Рисунок 3.2 – Можливості платформи cPanel

Загалом, cPanel є потужним і простим у використанні інструментом для керування вебхостингом. Він дозволяє користувачам зосередитися на розвитку вебпроєктів, надаючи легкий доступ до необхідних функцій і налаштувань.

Переваги cPanel:

– інтуїтивно зрозумілий і легкий у використанні: cPanel має зрозумілий і простий інтерфейс, що дозволяє навіть новачкам легко управляти своїми вебсайтами і серверами;

– багатофункціональність: cPanel надає широкий спектр функціональних можливостей, таких як керування файлами, базами даних, поштою, налаштування сервера, статистика вебсайту, установка CMS та інших програм тощо;

– розширюваність: cPanel підтримує велику кількість додаткових модулів і розширень, що дозволяє розширити його функціональність та забезпечити додаткові можливості;

– зручний доступ до технічної підтримки: cPanel надає зручний доступ до документації та технічної підтримки, що допомагає вирішувати проблеми та отримувати допомогу у випадку потреби.

Недоліки cPanel:

– вартість: деякі хостинг-провайдери вимагають додаткову плату за використання cPanel, що може збільшити загальну вартість хостингу, тому ця платформа не є найкращим вибором з цієї точки зору;

– залежність від хостинг-провайдера: cPanel є пропрієтарним рішенням, тому ви залежите від політики та оновлень, які здійснює ваш хостинг-провайдер. Це може обмежити вашу свободу вибору і контролю над сервером;

– обмежена масштабованість: хоча cPanel підтримує багато функцій, він може бути обмежений для великих підприємств або веброзробників, які потребують високого рівня настроювання та контролю.

Варто зазначити, що існують інші альтернативні панелі керування, такі як Plesk і DirectAdmin, які також мають свої переваги та недоліки. Вибір панелі керування залежить від ваших потреб, вмінь та вимог.

3.2 Створення серверної частини

Для створення серверної частини та API було обрано середовище Node.js, яке дозволяє створювати високопродуктивні серверні застосунки.

Для забезпечення роботи з даними була використана база даних PostgreSQL, яка забезпечує швидку та ефективну роботу з даними.

Для створення API була використана технологія REST API, яка дозволяє передавати дані між клієнтом та сервером в стандартному форматі JSON [21].

База даних є одним із основних складових будь-якого сервера. PostgreSQL є однією з найбільш популярних та найбільш надійних баз даних, яка забезпечує гнучкість та швидкість при роботі з даними.

Аутентифікація та авторизація є важливою частиною більшості застосунків та сервісів. Аутентифікація означає перевірку ідентифікації користувача, тобто підтвердження того, що користувач є тим, за кого він себе вважає. Авторизація визначає, які дії має дозволено виконувати аутентифікованому користувачу в залежності від його ролі та прав доступу [22].

NextAuth – це бібліотека для Node.js, яка дозволяє швидко додати аутентифікацію та авторизацію до застосунків. Вона підтримує багато провайдерів авторизації, таких як Google, Facebook, Twitter, GitHub, а також дозволяє створювати власні провайдери. NextAuth забезпечує безпеку за допомогою JWT-токенів, які можуть бути використані для авторизації запитів до сервера та для зберігання даних про користувача на клієнтському пристрої. Для використання NextAuth потрібно встановити його на сервері та

на клієнтській стороні, налаштувати провайдерів авторизації та обробники авторизації, які дозволяють зберігати та отримувати інформацію про користувача. NextAuth також дозволяє налаштувати різні ролі та дозволи доступу, щоб забезпечити безпеку застосунку [7].

У створеному онлайн сервісі авторизація відбувається наступним чином:

Крок 1. Сервер під’єднується до бази даних та робить запит до колекції Користувачів.

Крок 2. Якщо користувач бажає зайти у вже створений обліковий запис, то сервер намагається знайти його логін-дані з-поміж інших. Якщо логін та пароль сходяться, створюється нова сесія користувача, яка зберігає у собі JWT-токен.

Крок 3. Якщо користувач бажає створити новий обліковий запис, то сервер зробить запит до бази даних зі створенням нового користувача з усією наданою інформацією. Після успішної реєстрація, користувач має можливість спробувати функціональність описану у Кроці 2 (вхід у вже створений обліковий запис).

Крок 4. При кожному запиті на будь-яку сторінку, де потрібна авторизація, JWT-токен розбирається та витягується необхідна інформація для подальшого відображення онлайн сторінок.

Дуже важливо те, що пароль є зашифрованим і доступ до бази даних ніяким чином не надає шахраям цінні дані користувача.

На рисунку 3.3 можна побачити структуру файлів авторизації.

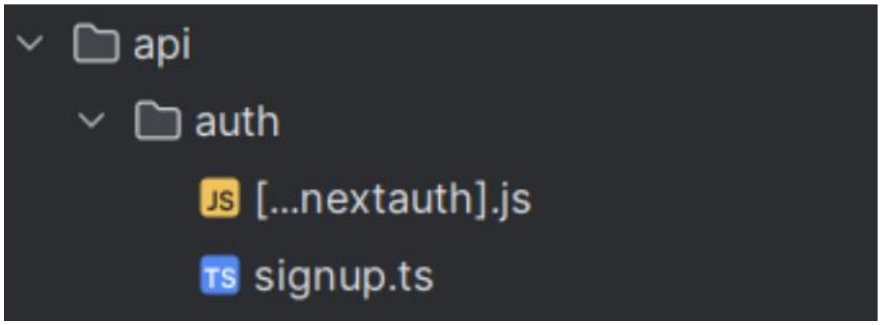


Рисунок 3.3 – Структура файлів авторизації

Файл […nexauth].js відповідає за вхід до вже створеного акаунту користувача, в той час як signup.ts – за створення нового облікового запису. Також, у кожному зі способів авторизації є дуже важлива валідація даних, яка запобігає створення нового аккаунту на вже зареєстровану електронну адресу, без пароля, тощо.

На лістингу 3.1 можна побачити код щодо цих дій.

Лістинг 3.1 Валідація реєстрації користувача:

```
if (
    !email ||
    !email.includes("@") ||
    !password ||
    password.trim().length < 6
) {
    res.status(422).json({
        message:
          "Invalid input - password should also be at least 6 characters
    long.",
    });
return;
}
```

Після успішного входу в обліковий запис, користувачу відкривається відповідна функціональність застосунку з проведення та протоколювання спортивних змагань до його ролі. У вебзастосунку існує 3 ролі: Адміністратор (для управління і підтримки самого вебзастосунку), Тренер (для управління спортсменами) і суддя (головний суддя, організатор змагання). Оскільки функції у кожної ролі різні і практично не перетинаються, то інтерфейси у всіх ролей будуть кардинально відрізнятися.

Код з файлу signin.js, що забезпечує можливість користувача залогінитись наведений у лістингу 3.2.

Лістинг 3.2 Вхід користувача з визначеною роллю:

```
({access: 'public',
method: async ({ login, password }) => {
const user = await api.auth.provider.getUser(login);
const hash = user ? user.password : undefined;
const valid = await metarhia.metautil.validatePassword(password, hash);
if (!user || !valid)
    throw new Error('Incorrect login or password');
    console.log(`Logged user: ${login}`);
    const token = await context.client.startSession(user.accountId);
    return {
        status: 'logged', token
        };
    },
});
```

3.3 Створення бази даних

У рамках кваліфікаційної роботи було розроблено базу даних для проведення студентських спортивних змагань. База даних містить наступні таблиці та їх взаємозв’язки: таблиця «Тренер», «Змагання», «Тип змагання», «Дистанція», «Турнір», «Суддя», «Команда» (рис. 3.4).

Таблиця «Тренер» має такі поля (рис. 3.5):

– id – унікальний ідентифікатор тренера;
– name – ім’я тренера;
– login – логін тренера;

– password – пароль тренера.

	Table	Action						
☐	coach	☆	Browse	Structure	Search	Insert	Empty	Drop
☐	competition	☆	Browse	Structure	Search	Insert	Empty	Drop
☐	comp_type	☆	Browse	Structure	Search	Insert	Empty	Drop
☐	distance	☆	Browse	Structure	Search	Insert	Empty	Drop
☐	heat	☆	Browse	Structure	Search	Insert	Empty	Drop
☐	participant	☆	Browse	Structure	Search	Insert	Empty	Drop
☐	referee	☆	Browse	Structure	Search	Insert	Empty	Drop
☐	team	☆	Browse	Structure	Search	Insert	Empty	Drop
	8 tables	Sum						

Рисунок 3.4 – Структура бази даних в таблицях

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	id 🔑	int(10)		UNSIGNED	No	*None*		AUTO_INCREMENT
☐	2	name	varchar(50)	utf8_general_ci		No	*None*		
☐	3	login	varchar(50)	utf8_general_ci		No	*None*		
☐	4	password	varchar(50)	utf8_general_ci		No	*None*		

Рисунок 3.5 – Структура таблиці «Тренер»

Таблиця «Змагання» має такі поля (рис. 3.6):

– id – унікальний ідентифікатор змагання;

– name – назва змагання;

– day – день проведення.

	#	Name	Type	Collation	Attributes	Null	Default	Comments
☐	1	id 🔑	int(10)			No	0	
☐	2	name	varchar(100)	utf8_general_ci		Yes	*NULL*	
☐	3	day	datetime			Yes	*NULL*	

Рисунок 3.6 – Структура таблиці «Змагання»

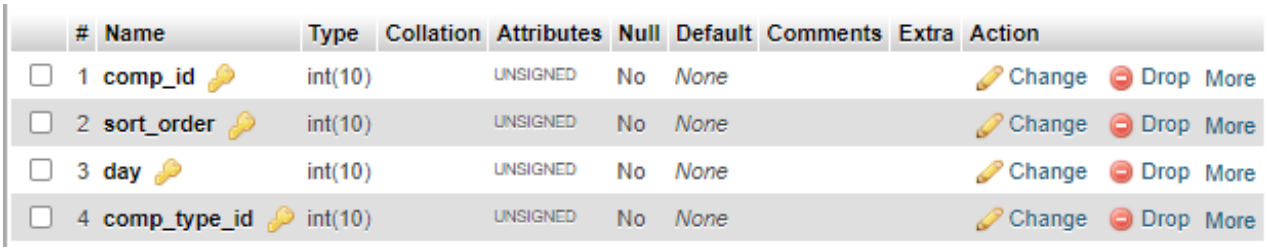
Таблиця «Дистанція» має такі поля (рис. 3.7):

– id – унікальний ідентифікатор дистанції;

– sort_order – порядок розподілу учасників;

– day – день проведення змагання на дистанції;

– comp_type_id – унікальний ідентифікатор типу змагання (зовнішній ключ до таблиці «Тип змагання»).



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	comp_id	int(10)		UNSIGNED	No	None			Change Drop More
2	sort_order	int(10)		UNSIGNED	No	None			Change Drop More
3	day	int(10)		UNSIGNED	No	None			Change Drop More
4	comp_type_id	int(10)		UNSIGNED	No	None			Change Drop More

Рисунок 3.7 – Структура таблиці «Дистанція»

Таблиця «Турнір» має такі поля (рис. 3.8):

– id – унікальний ідентифікатор турніра;

– comp_id – унікальний ідентифікатор змагання (зовнішній ключ до таблиці «Змагання»);

– sort_order – порядок розподілу учасників;

– day – день проведення змагання на дистанції;

– comp_type_id – унікальний ідентифікатор типу змагання (зовнішній ключ до таблиці «Тип змагання»);

– participant_id – унікальний ідентифікатор учасника (зовнішній ключ до таблиці «Учасник»);

– team_id – унікальний ідентифікатор команди (зовнішній ключ до таблиці «Команда»);

– application – заявка на участь;

– result – результат турніру.

Таблиця «Учасник» має такі поля (рис. 3.9):

– id – унікальний ідентифікатор турніра;

– surname – прізвище;

– name – ім’я;

– patronymic – по-батькові;

– birth_date – дата народження;

– sex – стать.

#	Name	Type	Collation	Attributes	Null	Default
1	id	int(11)			No	None
2	comp_id	int(10)		UNSIGNED	No	None
3	sort_order	int(10)		UNSIGNED	No	None
4	day	int(10)		UNSIGNED	No	None
5	comp_type_id	int(10)		UNSIGNED	No	None
6	participant_id	int(10)		UNSIGNED	No	None
7	team_id	int(10)		UNSIGNED	No	None
8	application	decimal(10,2)			Yes	NULL
9	result	decimal(10,2)			Yes	NULL

Рисунок 3.8 – Структура таблиці «Турнір»

#	Name	Type	Collation	Attributes	Null	Default
1	id	int(10)			No	0
2	surname	varchar(30)	utf8_general_ci		Yes	NULL
3	name	varchar(30)	utf8_general_ci		Yes	NULL
4	patronymic	varchar(30)	utf8_general_ci		Yes	NULL
5	birth_year	int(4)			Yes	NULL
6	sex	varchar(4)	utf8_general_ci		Yes	NULL

Рисунок 3.9 – Структура таблиці «Учасник»

Таблиця «Суддя» має такі поля (рис. 3.10):

– id – унікальний ідентифікатор судді;

– login – логін судді;

– password – пароль судді.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(11)			No	None		AUTO_INCREMENT
2	login	varchar(100)	utf8_general_ci		No	None		
3	password	varchar(100)	utf8_general_ci		No	None		

Рисунок 3.10 – Структура таблиці «Суддя»

Таблиця «Команда» має такі поля (рис. 3.11):

– id – унікальний ідентифікатор команди;

– name – ім’я команди;

– coach_id – унікальний ідентифікатор тренера команди (зовнішній ключ до таблиці «Тренер»).

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(10)		UNSIGNED	No	None		AUTO_INCREMENT
2	name	text	utf8_general_ci		No	None		
3	coach_id	int(10)		UNSIGNED	No	None		

Рисунок 3.11 – Структура таблиці «Команда»

Таблиці у базі даних студентських спортивних змагань мають наступні зв’язки. Таблиця «Тренер» має зв’язок з таблицею «Команда» через поле coach_id, яке є зовнішнім ключем до ідентифікатора тренера. Цей зв’язок вказує, що один тренер може бути пов’язаний з багатьма командами, але кожна команда може мати лише одного тренера.

Таблиця «Змагання» використовується в таблиці «Дистанція» через поле comp_id, яке є зовнішнім ключем до ідентифікатора змагання. Цей зв’язок означає, що одне змагання може мати багато дистанцій, але кожна дистанція належить лише до одного змагання.

Таблиця «Турнір» має кілька зв’язків з іншими таблицями:

а) використовує таблицю «Змагання» через поле comp_id, яке є зовнішнім ключем до ідентифікатора змагання. Це означає, що один турнір пов’язаний з одним змаганням;

б) використовує таблицю «Дистанція» через поле comp_type_id, яке є зовнішнім ключем до ідентифікатора типу змагання. Цей зв’язок вказує на тип змагання, який відповідає даному турніру;

в) таблиця «Турнір» має зв’язок з таблицею «Учасник» через поле participant_id, яке є зовнішнім ключем до ідентифікатора учасника. Цей

зв’язок означає, що один турнір може мати багато учасників, а кожен учасник може брати участь лише в одному турнірі;

г) з таблицею «Турнір» також пов’язана таблиця «Команда» через поле team_id, яке є зовнішнім ключем до ідентифікатора команди. Цей зв’язок вказує, що один турнір може мати багато команд, але кожна команда може брати участь лише в одному турнірі.

Таблиця «Команда» має зв’язок з таблицею «Тренер» через поле coach_id, яке є зовнішнім ключем до ідентифікатора тренера. Цей зв’язок вказує на те, що кожна команда має свого тренера, і кожен тренер може бути пов’язаний з багатьма командами.

Ці зв’язки допомагають управляти та пов’язувати дані між різними таблицями у базі даних студентських спортивних змагань.

Загалом, ця база даних дозволяє зберігати і взаємодіяти з інформацією про учасників, команди, тренерів, змагання, типи змагань та дистанції. Вона надає можливість ефективно вести облік та аналізувати дані, пов’язані зі студентськими спортивними змаганнями.

3.4 Створення клієнтської частини

Створення клієнтської частини для сайту з проведення спортивних змагань включає кілька етапів. Перш за все, проводиться аналіз вимог, де вивчаються потреби користувачів та функціональність, яку вони очікують від сайту.

Наступним кроком є проєктування інтерфейсу, де створюються дизайн та інтерфейс, які забезпечують зручну навігацію та ефективну взаємодію з користувачем. Розробка функціональності включає реалізацію різних функцій, таких як реєстрація та авторизація користувачів, перегляд інформації про змагання, учасників, команд, можливість подачі заявок на участь в змаганнях та перегляд результатів. Для цього використовуються

вебтехнології, такі як HTML, CSS і JavaScript, а також можуть використовуватися фреймворки, наприклад, React або Angular. Розробка адаптивності забезпечує коректне відображення сайту на різних пристроях та розмірах екранів. Нарешті, перед запуском сайту проводиться тестування, щоб перевірити правильність роботи функціоналу та гарантувати задоволення від користування сайтом.

Для створення компонентів сторінки ми можемо використовувати бібліотеки, такі як React, який дозволяє створювати компоненти зі складових частин. У даному випадку, було використано Next.js, що є фреймворком для React, який надає додаткові можливості для розробки застосунків [9].

Next.js – це фреймворк, який спрощує створення статичних та динамічних сторінок вебсайту з невеликими зусиллями. Він підтримує рендерінг як на стороні сервера, так і на стороні клієнта, що дозволяє покращити продуктивність застосунку і забезпечити кращий досвід користувача [23].

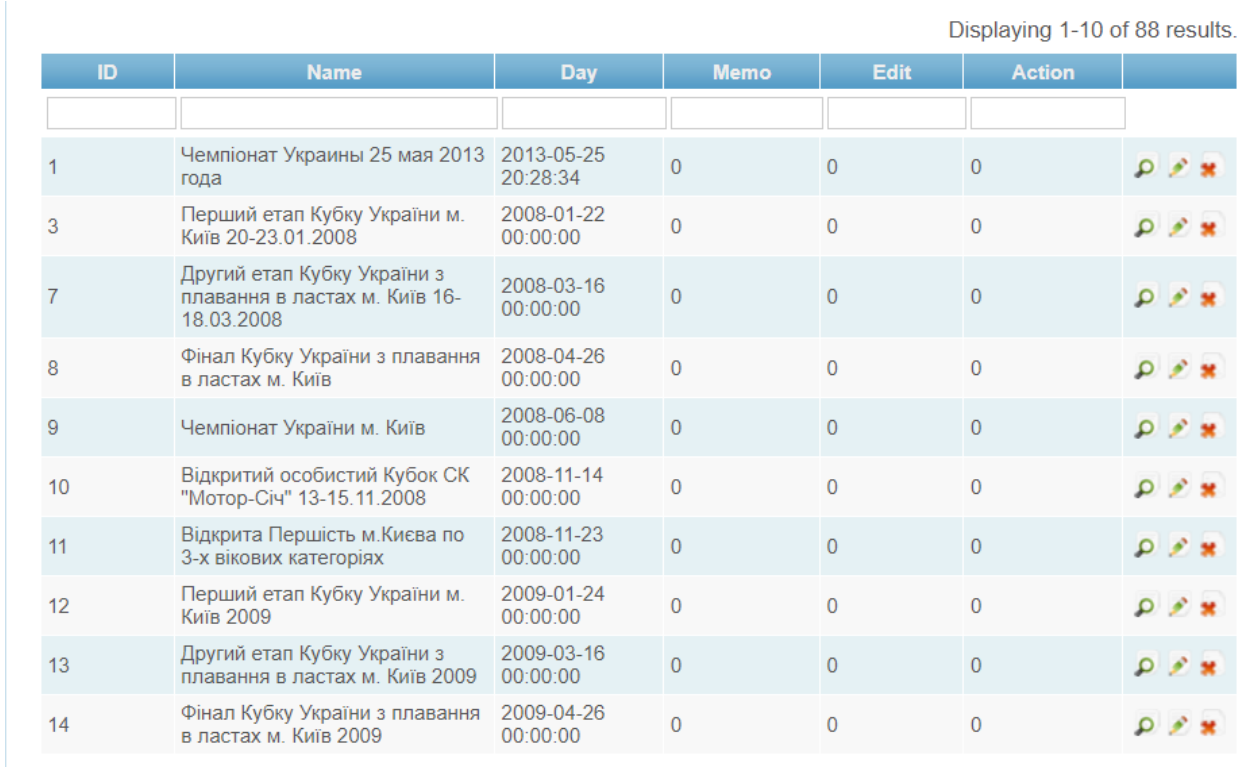
Для зв’язку між компонентами та серверною частиною використовується API, яке дозволяє передавати дані з клієнта на сервер і отримувати відповіді з сервера. Для реалізації аутентифікації та авторизації був використаний NextAuth. Цей інструмент дозволяє користувачам зареєструватися та увійти на сайт з використанням свого облікового запису [24-28].

Таким чином, використання Next.js разом із API та NextAuth дозволяє створити вебсайт зі зручною аутентифікацією та авторизацією, що полегшує роботу з застосунком та покращує взаємодію користувача з ним.

На рисунку 3.12 можна побачити програмну реалізацію одного з компонентів вебсайту – екран панелі Адмімністратора.

Стартова сторінка для всіх тренерів – це сторінка заявок і результатів. Тому саме з цієї сторінки почалася промальовування прототипу. А завдяки діаграмі переходу між сторінками було легко визначити, які функціональні елементи повинні знаходиться в тому чи іншому розділі і прототип якої

сторінки потрібно малювати наступним після поточного. На рисунку 3.13 наведено приклад прототипу сторінки команд.



Displaying 1-10 of 88 results.

ID	Name	Day	Memo	Edit	Action	
1	Чемпіонат Украины 25 мая 2013 года	2013-05-25 20:28:34	0	0	0	
3	Перший етап Кубку України м. Київ 20-23.01.2008	2008-01-22 00:00:00	0	0	0	
7	Другий етап Кубку України з плавання в ластах м. Київ 16-18.03.2008	2008-03-16 00:00:00	0	0	0	
8	Фінал Кубку України з плавання в ластах м. Київ	2008-04-26 00:00:00	0	0	0	
9	Чемпіонат України м. Київ	2008-06-08 00:00:00	0	0	0	
10	Відкритий особистий Кубок СК "Мотор-Січ" 13-15.11.2008	2008-11-14 00:00:00	0	0	0	
11	Відкрита Першість м.Києва по 3-х вікових категоріях	2008-11-23 00:00:00	0	0	0	
12	Перший етап Кубку України м. Київ 2009	2009-01-24 00:00:00	0	0	0	
13	Другий етап Кубку України з плавання в ластах м. Київ 2009	2009-03-16 00:00:00	0	0	0	
14	Фінал Кубку України з плавання в ластах м. Київ 2009	2009-04-26 00:00:00	0	0	0	

Рисунок 3.12 – Панель Адміністратора

Катран-плюс Київськ.обл. ТСОУ

ФИО спортсмена	Пол	Год рождения	Категория	Разряд	День 1					День 2					День 3		День 6		День 5
					50 м плавання в ластах	200 м плавання в ластах	100 м плавання в	400 м плавання в ластах	100 м підводне плавання	100 м плавання в ластах	50 м плавання в	200 м плавання в	50 м пірнання	400 м підводне плавання	800 м плавання в ластах	800 м підводне плавання	1500 м плавання в ластах	Ест. 4x200 в	3000 м марафон
Тютюн Євгенія	Жін.	1996	А	1р.	00:29.00	01:15.00					00:30.00								
Белоброва Катерина	Жін.	1995	А	1р.	00:25.00	01:10.00				00:55.00	00:28.00								
Шайда Ірина	Жін.	1994	А	1р.	00:28.00						00:29.00								
Полевая Олена	Жін.	1993	А	1р.	00:26.00	01:10.00				00:55.00	00:29.50								

"Сігма-Богатир" м.Кр. Ріг

ФИО спортсмена	Пол	Год рождения	Категория	Разряд	День 1					День 2					День 3		День 6		День 5
					50 м плавання в ластах	200 м плавання в ластах	100 м плавання в	400 м плавання в ластах	100 м підводне плавання	100 м плавання в ластах	50 м плавання в	200 м плавання в	50 м пірнання	400 м підводне плавання	800 м плавання в ластах	800 м підводне плавання	1500 м плавання в ластах	Ест. 4x200 в	3000 м марафон
Шама Вікторія	Жін.	1996	А	3р.		01:19.00					00:33.00								
Кліменкова Надія	Жін.	1993	А	1р.	00:23.80						00:27.50								
Волк Діана	Жін.	1992	А	1р.	00:21.50	00:56.50							00:20.00						
Щербак Олександра	Жін.	1992	А	МС		00:53.50					00:23.80	01:58.00							
Пікінер Ірина	Жін.	1990	А	МСМК		00:53.00					00:23.50	01:56.00	00:21.00						

Рисунок 3.13 – Сторінка команд

Панель для ролі Тренера (рис. 3.14) має безліч вкладених сторінок зі складною схемою переходу, що вимагало ретельного планування та

розробки. Початково був створений прототип всіх запланованих сторінок, який служив основою для подальшої програмної розробки інтерфейсу та функціоналу. В процесі розробки було використано Bootstrap, що дозволило створити панель тренера з адаптивною верстkою. Завдяки цьому, користувачі можуть зручно використовувати вебзастосунок навіть зі своїх смартфонів, за умови, що вони тримають горизонтальну орієнтацію екрану. Такий підхід забезпечує зручність і доступність використання системи на різних пристроях, забезпечуючи зручність роботи для тренерів у будь-який час та з будь-якого пристрою [29-34].

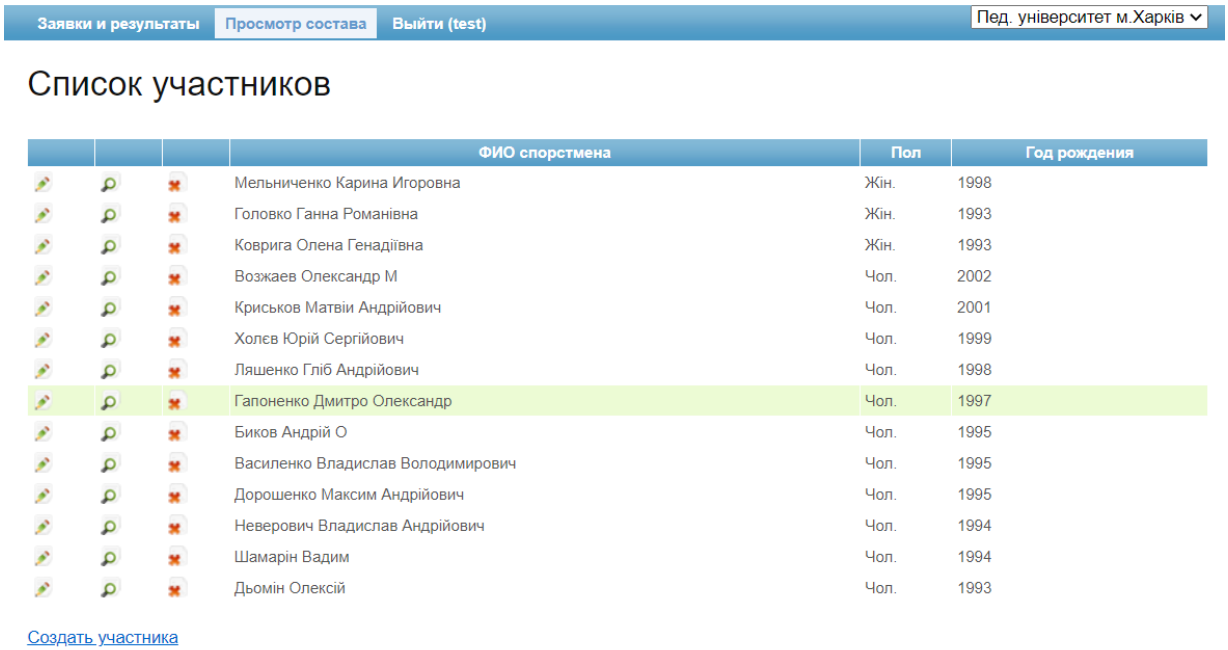


Рисунок 3.14 – Сторінка панелі Судді

Алгоритм генерації стартового протоколу, описаний в другому розділі, представлений на сайті на рисунку 3.15. Спортсмени розставляються по доріжках від кращого заявленого часу до гіршого в наступній послідовності: [4, 5, 6, 3, 2, 7, 1, 8]. Тобто спортсмена з найкращим (найсильнішим) заявленим часом ми ставимо на 4-у доріжку, спортсмена з заявленим часом трохи гірше на 5-у доріжку і т.д.

1 день змагання

50 м плавання в ластах

Доріжка	Разряд	Призвище	Им'я	Категорія	Заявка	Команда
1 заплив						
2	юн.	Денисюк	Оксана	E	00:38.00	ДЮСШ "Аквалідер" - ЦСТК м.Київ
3	3р.	Стукало	Олена	E	00:30.00	ДЮСШ Україна м.Київ
4	2р.	Юречко	Марта	E	00:26.00	ДЮСШ "Аквалідер" - ЦСТК м.Київ
5	2р.	Галицька	Вікторія	E	00:26.70	ДЮСШ "Аквалідер" - ЦСТК м.Київ
6	юн.	Кайдан	Мілана	E	00:30.00	ДЮСШ "Аквалідер" - ЦСТК м.Київ
2 заплив						
1	юн.	Чала	Ярина	D	00:30.00	ДЮСШ "Аквалідер" - ЦСТК м.Київ
2	3р.	Іванова	Анастасія	D	00:28.75	ДЮСШ "Аквалідер" - ЦСТК м.Київ
3	2р.	Кухар	Катерина	D	00:26.40	ДЮСШ Україна м.Київ
4	2р.	Сафонова	Валерія	D	00:25.50	Сігма-Богатир м.Кривий Ріг
5	2р.	Черкасова	Ганна	D	00:25.80	ДЮСШ "Аквалідер" - ЦСТК м.Київ
6	2р.	Мухай	Валерія	D	00:26.50	ДЮСШ-21 м.Київ
7	3р.	Левченко	Ілларія	D	00:29.50	Дарниця-303 м.Київ
8	юн.	Савченко	Дарина	D	00:36.00	Дарниця-303 м.Київ
3 заплив						
1	2р.	Сахарова	Марина	D	00:25.40	Дарниця-303 м.Київ
2	2р.	Шарейко	Дарина	D	00:24.00	ДЮСШ "Аквалідер" - ЦСТК м.Київ
3	1р.	Петрущенко	Дар'я	D	00:23.60	ДЮСШ "Аквалідер" - ЦСТК м.Київ
4	2р.	Почтаренко	Дарина	D	00:23.00	ДЮСШ "Аквалідер" - ЦСТК м.Київ
5	2р.	Больботенко	Маргарита	D	00:23.53	ДЮСШ "Аквалідер" - ЦСТК м.Київ
6	2р.	Яцив	Валерія	D	00:23.90	ДЮСШ "Аквалідер" - ЦСТК м.Київ
7	2р.	Артеменко	Вікторія	D	00:24.40	Сігма-Богатир м.Кривий Ріг
8	2р.	Брагіна	Анастасія	D	00:25.40	ДЮСШ "Аквалідер" - ЦСТК м.Київ

Рисунок 3.15 – Вигляд «Стартового протоколу» в панелі Судді

У поточному вебзастосунку даний алгоритм був реалізований спочатку з допомогою SQL запиту, а після був відображений на синтаксис Yii фреймворка.

Крім реалізації безпосередньо самого алгоритму, який запускається лише 1 раз для кожного змагання, була реалізована можливість в створеному стартовому протоколі вручну для кожного спортсмена змінювати:

– номер запливу в рамках однієї дисципліни і категорії;
– доріжка, по якій буде плисти спортсмен;
– заявлений час, яке вказав тренер при реєстрації.

При ручному редагуванні стартового протоколу, були добавлені перевірки, які запобігають людській помилці. Зокрема з’явилася перевірка, яка перевіряє, щоб на одній доріжці було не більше одного спортсмена у рамках одного запливу.

При подачі заявки для кожного спортсмена Тренер зазначає типи рейтингів, в яких бере участь даний спортсмен, за допомогою чекбоксів (рис. 3.16).

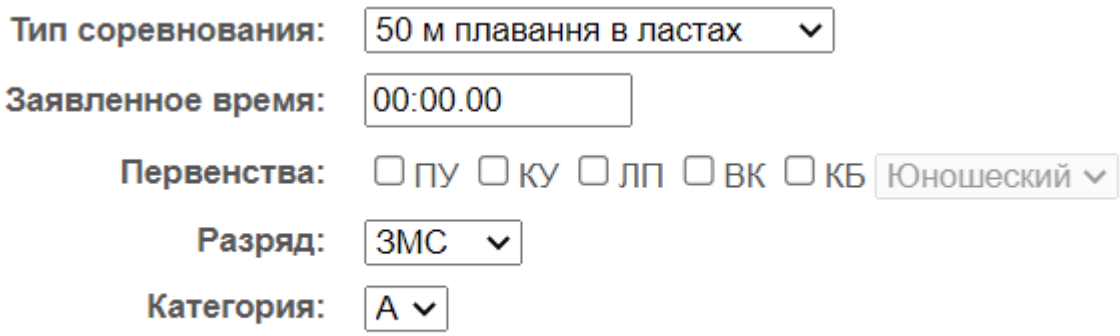


Рисунок 3.16 – Область сторінки подачі заявки

Тому при підрахунку фінальних (підсумкових) протоколів, алгоритм повинен враховувати для кожного протоколу тільки тих спортсменів, у яких за цим типом протоколу відзначено значення «True» в заявці.

У поточному вебзастосунку усі алгоритми фінальних протоколів спочатку були реалізовані за допомогою SQL запитів [35-39]. Після цього були протестовані на коректність роботи.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований вебзастосунок з проведення студентських спортивних змагань.

Метою кваліфікаційної роботи було створення зручного та ефективного інструменту для проведення змагань за плавання різного типу, де не має можливості запровадити повністю автоматичні системи хронометражу та реєстрації часу. Під час розробки були використані сучасні технології та фреймворки, такі як React для розробки застосунку, Node.js для розробки серверної частини та PostgreSQL для зберігання даних. Застосунок був ретельно спроєктований, використовуючи інформаційну систему, математичні моделі та алгоритми, що дозволило ефективно керувати та організовувати спортивні змагання.

Використання React.js для фронтенду дозволило створити інтуїтивний та динамічний інтерфейс користувача, який забезпечує зручну навігацію та взаємодію зі спортивними даними.

Node.js було обрано для розробки бекенду, що забезпечило швидке та масштабоване виконання запитів, а також можливість використання JavaScript як мови програмування як на фронтенді, так і на бекенді, що спростило розробку та підтримку коду.

Використання PostgreSQL як системи управління базами даних дозволило ефективно зберігати та керувати спортивними даними, забезпечуючи швидкий доступ та надійність операцій.

В результаті реалізації вебзастосунку студентські спортивні змагання стали більш організованими, ефективними та доступними для учасників та організаторів. Користувачі можуть легко реєструватися на змагання, переглядати розклад, результати та статистику, а організатори мають зручні інструменти для керування змаганнями та обробки даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. D'Angelo, J., & Little, S. K. (1998). Successful web pages: what are they and do they exist?. *Information technology and libraries*, *17*(2), 71.

2. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

3. Banks, A., & Porcello, E. (2020). *Learning React: modern patterns for developing React apps*. O'Reilly Media.

4. What is Microsoft SQL Server? A definition from Whatls.com. URL: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server (дата звернення 21.04.2023).

5. Microsoft SQL Server. (2018). *Admin Kit.* Birn, Jeffrey. M.: LORI, 211.

6. Zheng, Q., Jiao, J., Cao, Y., & Lau, R. W. (2018). Task-driven webpage saliency. *In Proceedings of the European conference on computer vision (ECCV)* (pp. 287-302).

7. NextAuth.js Documentation. URL: https://next-auth.js.org/getting-started/introduction (дата звернення 21.04.2023).

8. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management System. URL: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems (дата звернення 08.05.2023).

9. Kikuchi, K., Otani, M., Yamaguchi, K., & Simo-Serra, E. (2021, October). Modeling Visual Containment for Web Page Layout Optimization. In *Computer Graphics Forum* (Vol. 40, No. 7, pp. 33-44).

10. Регламент змагань з плавання та бігу. URL: https://swimrun.if.ua/index.php/event-program (дата звернення 21.04.2023).

11. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image

classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

12. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (pp. 665-670). IEEE.

13. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

14. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video Clustering via Multidimensional Time-Series Analysis. In *Proceedings of the 9th International Conference on Information Management and Engineering* (pp. 60-63). ACM.

15. Zain, S. B. M., & Zain, S. M. (2020). JAVA GUI BUNDLE pp. 4-23.

16. Loy, M., Eckstein, R., Wood, D., Elliott, J., & Cole, B. (2020). Java swing. " O'Reilly Media, Inc." pp. 4-10.

17. FRĄSZCZAK, D., BUGAJEWSKI, D., & MAGDZIARZ, K. (2022). Swing Dynamic GUI. Proceedings of the 40th International Business Information Management Association (IBIMA), 23, 24.

18. SWT on the Eclipse website. URL: https://www.eclipse.org/swt/ (дата звернення 19.04.2023).

19. JavaFX documentation. URL: https://openjfx.io/ (дата звернення 21.04.2023).

20. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – 4701-4706.

21. Rosati, C., Weiss, G., Lund, S. K. K., & Smith, E. (2019). New Java frameworks for building next generation EPICS applications.

22. Alkhars, A., & Mahmoud, W. (2019). Cross-Platform Desktop Development (JavaFX vs. Electron) pp. 7-15.

23. Pedersen, S. B., & Jackson, S. (2019). Graphical User Interface Programming Challenges Moving Beyond Java Swing and JavaFX. Proc. ICALEPCS'19.

24. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (pp. 665-670). IEEE.

25. Daniels, J. (2013). Daniels' running formula. Human Kinetics pp. 1-90.

26. Рожков, В. О. (2022). Особливості застосування навантажень під час занять здобувачів вищої освіти в секціях з легкої атлетики. Матеріали III Всеукраїнської науково-практичної конференції, С. 87.

27. Junaydulloevich, A. M., & Istamovich, A. K. (2021). Basic laws and descriptions of ways to develop technical skills in boxing. Web of Scientist: International Scientific Research Journal, 2(05), pp. 15-26.

28. Costill, D. L., Thomas, R., Robergs, R. A., Pascoe, D., Lambert, C., Barr, S., & Fink, W. J. (1991). Adaptations to swimming training: influence of training volume. Medicine & Science in Sports & Exercise, 23(3), 371-377.

29. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *5*(57).

30. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2018, August). Representative Based Clustering of Long Multivariate Sequences with Different Lengths. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* (pp. 545-548). IEEE.

31. Bodyanskiy, Y., Kobylin, I., Rashkevych, Y., Vynokurova, O., & Peleshko, D. (2018, February). Hybrid fuzzy-clustering algorithm of unevenly and asynchronously spaced time series in computer engineering. In *2018 14th*

International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 930-935). IEEE.

32. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23-28.

33. Lyashenko, V., Matarneh, R., Kobylin, O., & Putyatin, Y. (2016). Contour detection and allocation for cytological images using Wavelet analysis methodology.

34. Гузій, О. В., Романчук, О. П., & Магльований, А. В. (2020). Сенсомоторні показники як критерії впливу інтенсивних фізичних навантажень на організм спортсмена. Український журнал медицини, біології та спорту, 5(3), С. 351-358.

35. Tursunov Oqil Amrilloevich. (2022). Methods of Increasing the Effectiveness of Training of Athletes Engaged in Primary Training Groups. Texas Journal of Multidisciplinary Studies, 6, pp. 134–139.

36. Kobylin, O., & Lyashenko, V. (2016). Contrast Modification as a Tool to Study the Structure of Blood Components.

37. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

38. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2019, June). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In *International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications* (pp. 210-230). Springer, Cham.

39. Kinoshenko, D., Kobylin, O., Mashtalir, S., & Stolbovyi, M. (2019, March). Metric video retrieval speedup by irrelevant data elimination. In *Eleventh International Conference on Machine Vision (ICMV 2018)* (Vol. 11041, pp. 176-183).