МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка веб-додатку реєстрації на події з використанням фреймворку NestJS**

Виконав: студент 4 курсу, групи 6.1219-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
А. І. Артеменко
(ініціали та прізвище)
Керівник доцент, к.т.н., доцент О. М. Міхайлуца
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалаврський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Т.В. Критська_
"_01_" ___березня__ 2023_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Артеменку Артуру Ігоровичу
(прізвище, ім'я, по батькові)

1.Тема роботи _Розробка веб-додатку реєстрації на події з використанням фреймворку NestJS

керівник роботи ___Міхайлуца Олена Миколаївна, доцент, к.т.н._____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 29.12.2022 №1893-с

2. Строк подання студентом кваліфікаційної роботи_____14.06.2023_____
3. Вихідні дані бакалаврської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми збору та аналізу даних з веб-сторінок;
- створення програмного продукту та його опис;
- дослідження поставленої проблеми та розробка висновків.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Аналіз предметної області	24.04.23	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	27.04.23	виконано
3	Аналіз існуючих методів рішення	28.04-30.04.23	виконано
4	Дослідження засобів реалізації серверної частини застосунку	01.05-03.05.23	виконано
5	Дослідження засобів реалізації фронтендової частини застосунку	04.05-05.05.23	виконано
6	Узгодження подальших дій з науковим керівником	05.05.23	виконано
7	Розгортання бази даних	06.05.23	виконано
8	Програмна реалізація серверної частини застосунку	07.05-12.05.23	виконано
9	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	13.05-15.05.23	виконано
10	Реалізація користувацького інтерфейсу	16.05-20.05.23	виконано
11	Перевірка роботоздатності проекту	21.05-23.05.23	виконано
12	Оформлення звіту	24.05-11.06.23	виконано
13	Оформлення презентації. Отримання рецензій від опонентів.	12.06-15.06.23	виконано

Студент _____ Артеменко А.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Міхайлуца О.М.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціал

АНОТАЦІЯ

Сторінок: 72

Рисунків: 24

Лістингів: 6

Джерел: 16

Артеменко А. І. Дослідження проблеми розробка веб-додатку реєстрації на події з використанням фреймворку NestJS : кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник к.т.н., доцент О.М. Міхайлуца. Запоріжжя : ЗНУ, 2023. 74 с.

У наш час, ефективний механізм реєстрації на кіберспортивні івенти є цінним ресурсом для організації, яка проводить такі змагання. Підвищення ефективності може бути одним з напрямків удосконалення діяльності організацій, що займаються кіберспортом.

Метою дослідження є вивчення особливостей розробки веб-додатку для реєстрації на події, а також надання практичних рекомендацій щодо використання сучасних технологій у веб-додатках такого типу. Досліджено наявні програмні рішення, що використовуються на ринку. Встановлено, що автоматизація робочих процесів є перспективним та важливим напрямом у цих процесах.

Був проведений аналіз технологій для розробки програмного продукту та вже наявних підходів до розробки систем такого типу. На основі цього аналізу було розроблено веб-додаток для реєстрації на кіберспортивні події з використанням сучасних методологій та технологій розробки. Цей додаток може задовольнити потреби організаторів, які прагнуть підвищити ефективність своєї роботи.

Ключові слова: *кіберспорт, фронтед частина, серверана частина, NestJs, React*

ABSTRACT

Pages: 72

Drawings: 24

Listings: 6

Sources: 16

Artemenko A.I. Research of the problem of developing a web application for event registration using the nestjs framework: bachelor's thesis in specialty 121 "Software Engineering" / scientific supervisor Candidate of Technical Sciences, Associate Professor O.M. Mikhailutsa. Zaporizhzhia: ZNU, 2023. 74 p.

Nowadays, an effective mechanism for registering for esports events is a valuable resource for an organization that conducts such competitions. Increasing efficiency can be one of the ways to improve the activities of organizations involved in esports.

The purpose of the study is to study the peculiarities of developing a web application for event registration, as well as to provide practical recommendations on the use of modern technologies in web applications of this type. The existing software solutions used on the market are studied. It is established that workflow automation is a promising and important direction in these processes.

An analysis of technologies for developing a software product and existing approaches to the development of systems of this type was carried out. Based on this analysis, a web application for registering for esports events was developed using modern development methodologies and technologies. This application can meet the needs of organizers seeking to increase the efficiency of their work.

Keywords: *esports, frontend part, backend part, NestJs, React*

ЗМІСТ

ВСТУП

Актуальність теми

За останні роки кіберспорт почав набувати все більшої популярності кожен день. Тому створення сайту, який буде призначений проведенню таких змагань є дуже актуальною темою на сьогоднішній день.

Кіберспорт — це вид спорту, де гравці змагаються у відеоіграх. Це може бути ігрова консоль, комп'ютер або мобільний пристрій, а змагання можуть проходити онлайн або вживу.

Оскільки кіберспорт починає набувати все більшої популярності, то і запит на сайти, які присвячені темі організації турнірів та змагань з кіберспорту поступово зростає. Такі сайти мають можливість зробити більш зручну реєстрацію на змагання, краще підготуватися до своїх супротивників, оцінити можливість виграшу у тої чи іншої команди. Також у деяких сайтів є можливість перегляду записів ігор та коментарів до них.

Кіберспорт має значний комерційний потенціал, оскільки це швидко розвиваючийся та динамічний ринок, що привертає увагу мільйонів глядачів по всьому світу. Таким чином, компанії можуть використовувати кіберспорт як засіб просування своїх продуктів та послуг.

Один з основних способів, яким компанії можуть використовувати кіберспорт для просування своїх брендів - це спонсорство кіберспортивних команд та турнірів. Компанії можуть спонсорувати змагання та команди, надаючи їм фінансову та матеріальну підтримку. У зворотному випадку, команди та змагання можуть рекламувати бренд компанії, що спонсорує їх, на своїх сайтах, рекламних дошках та екранах під час трансляцій.

Крім того, компанії можуть використовувати кіберспортивних гравців та команди для реклами своїх продуктів та послуг. Наприклад, кіберспортивний гравець може бути обличчям рекламної кампанії для відео ігрової консолі, або команда може виступати у рекламі певної компанії.

Крім того, кіберспортивні турніри та змагання також можуть бути майданчиком для реклами та просування продуктів та послуг. Компанії можуть рекламувати свої продукти на екранах під час трансляцій змагань, а також пропонувати свої послуги у зоні коментаторів та під час пауз між іграми.

У загальному, комерційний потенціал кіберспорту є великим, оскільки це швидко розвиваючийся ринок з великою аудиторією.

Можна використовувати гроші інвесторів для того, щоб робити робити власні турніри з великим призовим фондом, а також створення додаткових можливостей для залучення інвесторів й рекламодателів. Таким чином залучення великих капіталів буде позитивно впливати на розвиток сайту.

Можемо зробити висновок, що тема створення сайту, який буде займатися проведенням кіберспортивних змагань є дуже актуальною темою в сучасному світі. Данні сайти можуть бути корисними як і для звичайних гравців, так і для компаній, які мають можливість й хочуть використовувати кіберспорт як засіб просування своїх продуктів.

Мета дослідження

Розробка веб-додатку реєстрації на кіберспортивні події з використанням сучасних методологій розробки та з використанням сучасних технологій розробки.

Завдання дослідження

Завданням дослідження було вивчення та розглядання вже існуючих рішення для проведення кіберспортивних змагань та їх альтернатив, вивчення їх особливостей. Зробити аналіз вже існуючих підходів розробки веб-застосунків та технології, які використовують для написання застосунків даного типу.

Об’єкт дослідження

Об’єктом дослідження є процес створення сайту для проведення кіберспортивних змагань серед команд, які попередньо були зареєстровані на сайті.

Предмет дослідження

Предметом дослідження є необхідність створення власного сайту з проведенням кіберспортивних змагань серед команд та гравців, які були зареєстровані на даному сайті.

Методи дослідження

Теоретичні — дослідження та порівняння технологій, які використовуються для створення інформаційних систем. Класифікація і принципи проектування інформаційних систем, які дозволяють їх ефективно розробляти. Аналіз різних програмних рішень з метою вибору найбільш підходящого для конкретної ситуації.

Практичне значення одержаних результатів

Результати дослідження можуть бути використані для створення веб-додатку реєстрації на кіберспортивні події.

Апробація результатів

Результати роботи було представлено на XVI університетській науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених «Молода наука-2023» [17].

Глосарій

Node.js — це відкрита платформа для розробки серверних додатків на мові JavaScript.

JavaScript — це високорівнева мова програмування, яка використовується для розробки веб-додатків.

Nest.js — це фреймворк для розробки серверних додатків на мові програмування TypeScript.

TypeScript — це мова програмування, яка є розширенням мови JavaScript.

Бекенд (англ. backend) — це частина веб-додатка або програмного забезпечення, яка відповідає за обробку даних та бізнес-логіки.

Фронтенд (англ. frontend) — це частина веб-додатка або програмного забезпечення, яка відповідає за взаємодію з користувачем та візуальне відображення інформації на веб-сторінці.

React.js (або просто React) — це JavaScript бібліотека для розробки користувацьких інтерфейсів.

MongoDB — це документо-орієнтована, NoSQL база даних, яка зберігає дані у вигляді документів у форматі JSON (JavaScript Object Notation).

Mongoose — це бібліотека об'єктно-документного моделювання (ODM) для Node.js та MongoDB.

Чітери — це гравці у комп'ютерні або відеоігри, які використовують заборонені програми, модифікації або інші методи для отримання переваги над іншими гравцями.

Кіберспорт (або електронний спорт) — це форма конкурентної діяльності, в якій гравці змагаються один з одним у відеоіграх на професійному рівні.

Матчмейкінг (Matchmaking) — це процес автоматичного формування пар або груп гравців у відеоіграх для забезпечення рівних або близьких за рівнем навичок та досвіду суперників.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літературних джерел

Питанням огляду та аналізу створення нових веб-застосунків присвячених реєстрації на події присвячено багато статей [1-2]. Зміни в суспільстві, пошук молоддю свого місця у новій цифровій реальності, досягнення та реалізація їх амбіцій незалежно від їх фізичних обмежень — це те, що призвело до зростання кіберспроту та популярності комп’ютерних ігор в цілому. Саме кіберспрот став для багатьох хлопців і дівчат можливістю для самореалізації. За останні десять років відеоігри з простого способу проведення дозвілля переросли в окремий вид спорту, який згодом отримав назву — кіберспорт. На даний момент мільйони глядачів по всьому світу звертають свою увагу на мультиплеєрних змагальних іграх, саме за цим жанром, переважно, проводять змагання з кіберспорту, і для цього навіть достатньо мати лише один смартфон. Крім того, це прирівнює глядачів по всьому світу, оскільки в кіберспорті всі бачать однакову картину, незалежно від свого місця перебування [1].

З початку 2000-х років навколо кіберспортивних змагань була сформовано ціла індустрія і була утворена певна інфраструктура, яка охоплює продаж певних груп товарів: комп'ютерних мишей, навушників, клавіатур а також й іншого обладнання. Також, паралельно з технікою почав розвиватися ринок з продажу одягу з логотипами кіберспортивних команд, сувенірів, які були виготовлені на замовлення улюбленої команди, реклами, прав на трансляцію матчів. У 2020 році даний ринок зміг перевищити дохід в 1 мільярд доларів, також є прогнози, що до 2023 року цей ринок зможе зрости ще на 50%, а саме до рівня у 1,5 мільярда доларів. Саме тому розвиток цієї інфраструктури зміг призвести до залучення численних спонсорів для проведення масштабних турнірів з величезними призовими фондами.

Для того, щоб підтвердити дану гіпотезу, про зацікавленість населення у темі кіберспорту, було проведено дослідження, з використанням сервісу Google Trends. Дослідження базується на пошукових запитах по тематиці “e-sport”. Даний аналіз був проведен для всіх країн світу, а також певних країн які визнали кіберспорт як офіційний вид спорту [2]. Період аналізу охоплював з кінця 2011 року до кінця 2021 року (див. Рис. 1).

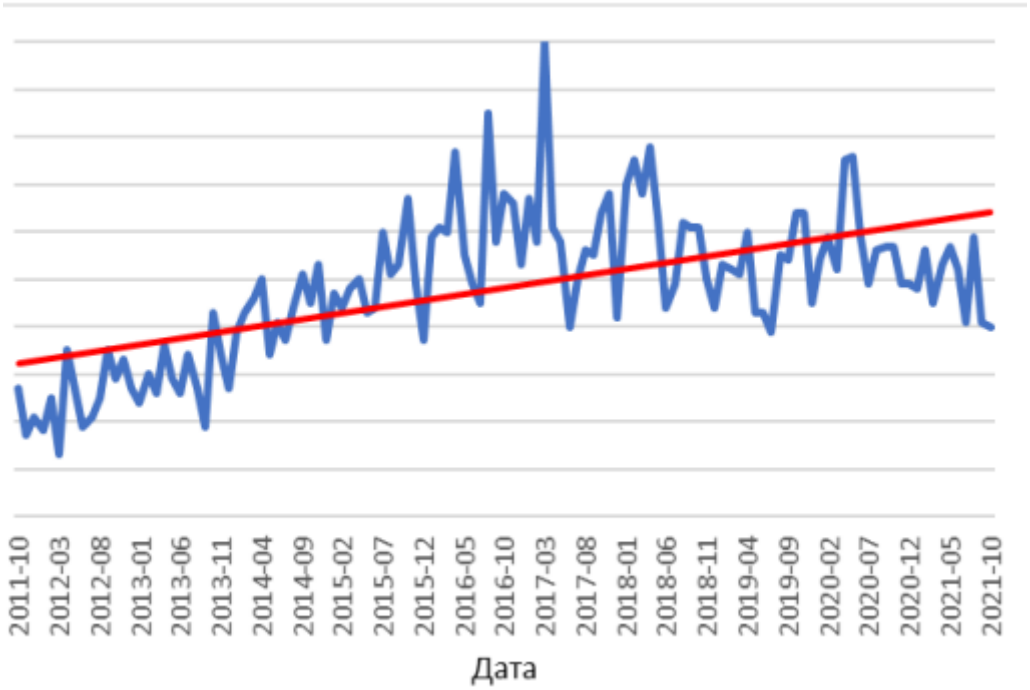

Рис. *1 Аналіз запитів категорії «e-sport» по всьому світу*

На підставі даних, отриманих з рисунку 1, можемо зробити такий висновок, що зацікавленість теми кіберспорту у світі, протягом певного періоду часу, показувала зростаючу тенденцію. Якщо взяти дані з діаграми, то можна побачити, що в 2011 році, в середньому запитів, на тему кіберспорт, на день від користувачів Google Chrome було менше ніж за той самий період в 2021 році. Також можна зазначити, що пік інтересу до теми кіберспорт в світі спостерігався в період між серединою 2016 року та серединою 2017 року, коли був досягнений пік запитів.

Не зважаючи на потужний економічний та цифровий розвиток такої країни як США, активна зацікавленість людей кіберспортом почалась у другій

половині 2013 року (див. Рис. 2). Потім активність, зацікавлених осіб, почала зростати тільки у другій половині 2015 року. Так у цей період кількість запитів була піковою.

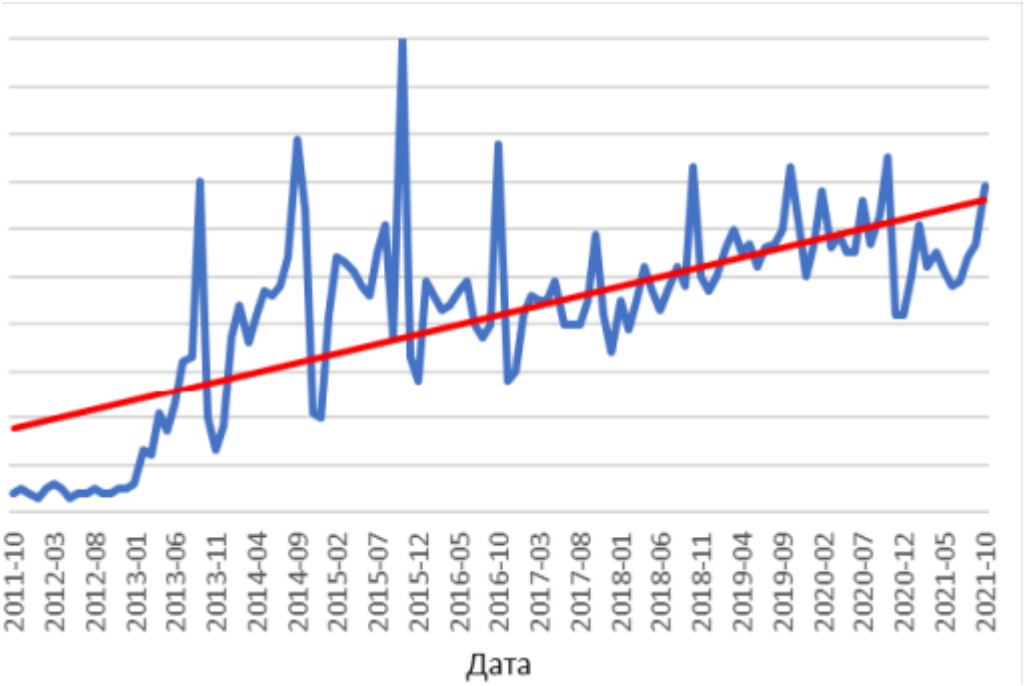

Рис. *2 Аналіз запитів категорії «e-sport» в США*

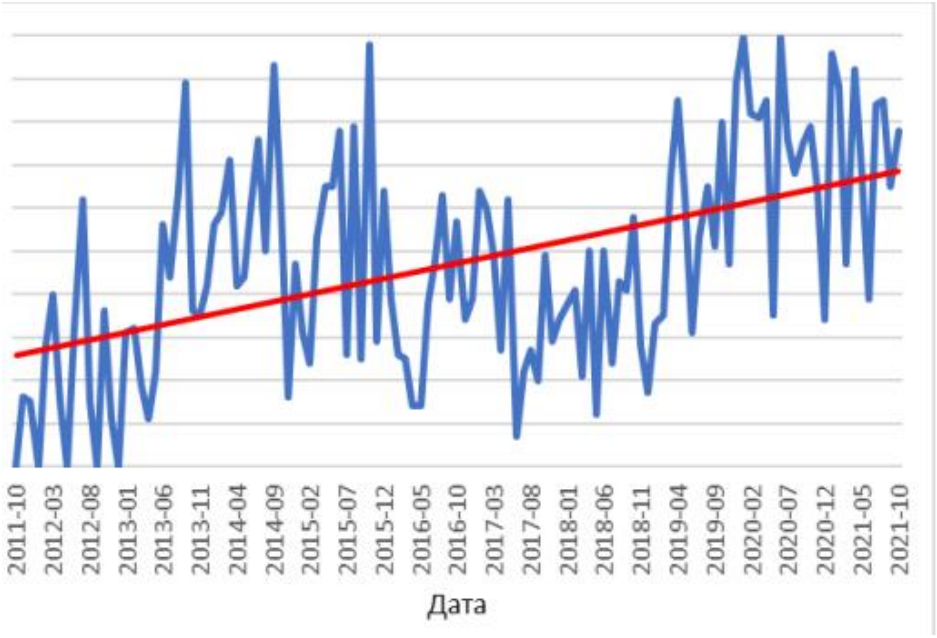

Рис. *3 Аналіз запитів категорії «e-sport» в Україні*

У випадку для України характерна така статистика, яка зображена на рисунку 3. Дана діаграма свідчить про те, що в Україні все більше людей починають цікавитися тематикою кіберспорту й починають дізнаватися про даний вид спорту більш детальну інформацію в мережі Інтернет.

Таким чином, на основі лінії Трента, можна зробити висновок, що спостерігається тенденція активізації популярності кіберспорту.

1.2 Аналіз програмних продуктів-аналогів

HLTV

HLTV — це сайт з новинами, а також сайт, який являє собою форум [3]. Цей сайт охоплює турніри, статистику, а також професійні новини з Counter-Strike: Global Offensive. На рисунку 4 зображена головна сторінка цього сайту.

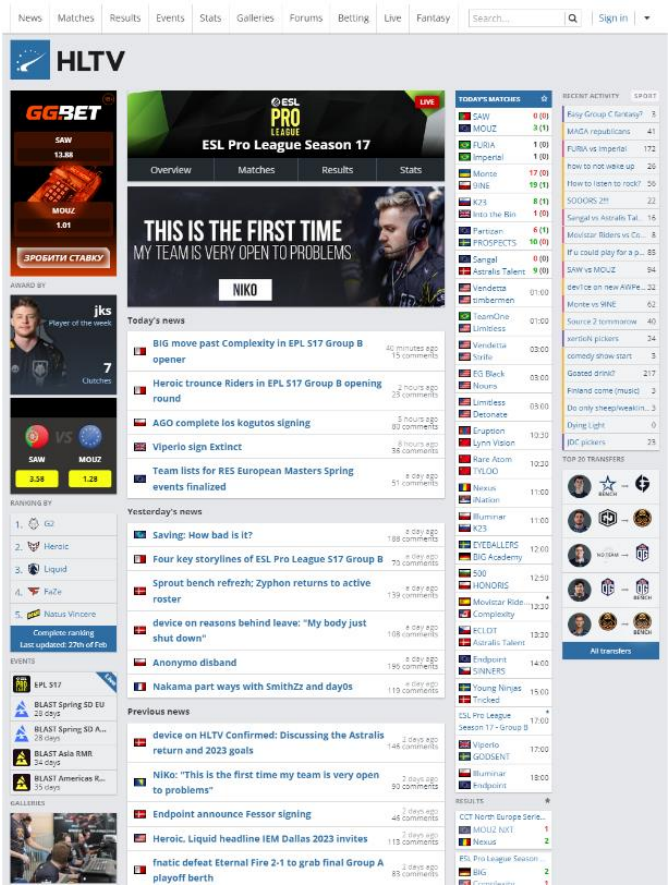

Рис. *4 Головна сторінка сайту HLTV*

Також, цей сайт має власну, дуже розвинену статистику, яка складається з багатьох змінних. Нижче наведені приклади статистики, яка на даний момент існує на сайті:

1. Статистика за матч, до якої входять такі пункти (див. Рис. 5):
    1) статистика в цілому за весь матч;
    2) статистика по кожній окремій карті;
    3) статистика про кожну команду;
    4) детальний опис по кожному гравцю;
    5) статистика зустрічей двох команд один з одним;
    6) статистика попередніх зустрічей кожної команди.

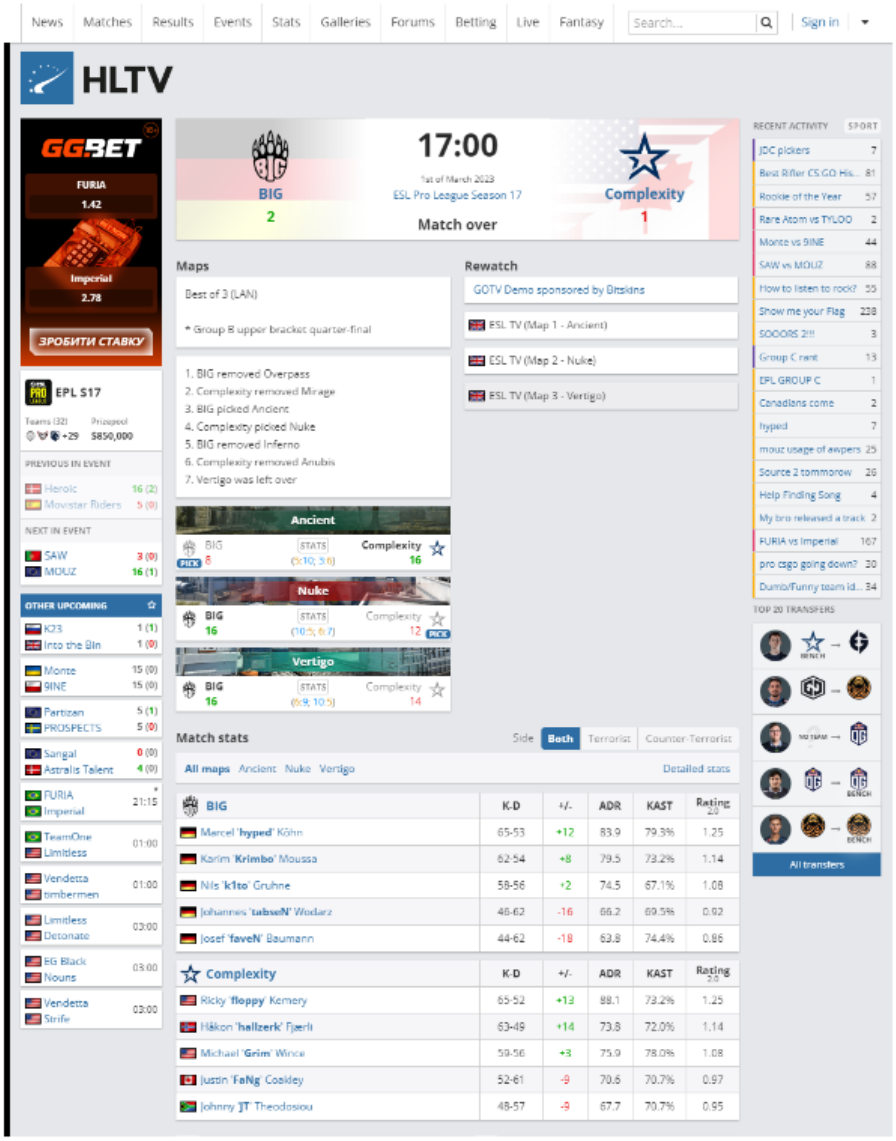

Рис. 5 *Статистика за матч*

2. Статистика кожної команди, в яку входять такі пункти:
    1) статистика обраної команди відносно інших команд у світовому рейтингу найкращих команд;
    2) статистика по матчах команди, які вона провели за весь час існування команди на цьому сайті;
    3) коротка статистика про кожного члена команди загалом;
    4) статистика по турнірах які відвідала дана команда за весь період;
    5) здобутки команди за увесь час.
    6) коротка статистика по картах, які зіграла команда за минулі 3 місяці
3. Статистика гравця, вона має наступні пункти (див. Рис. 6):
    1) нагороди гравця;
    2) повна інформація про гравця, з урахуванням його статистики;
    3) команди за які він грав протягом кар'єри;
    4) нагороди, які отримала команда за весь період гри гравця за команду.

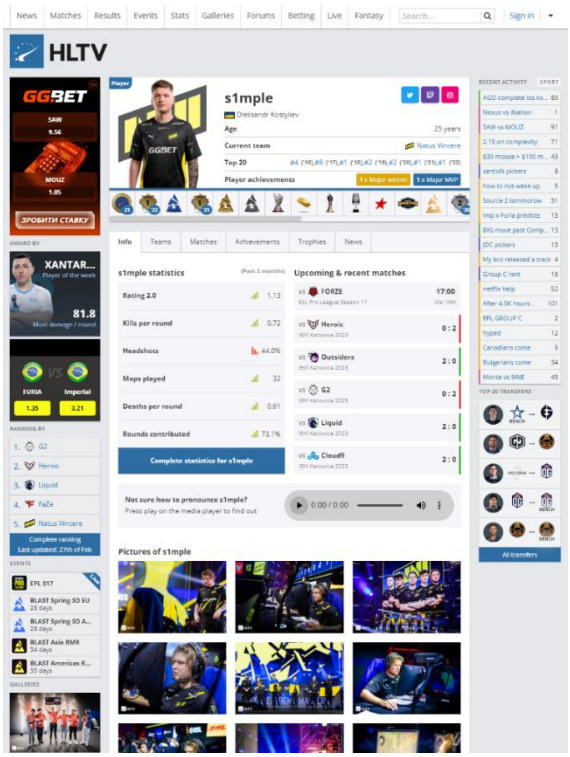

Рис. *6 Статистика гравця*

Даний сайт є одним з найбільших й провідний сайт у цій спільноті з понад 4 мільйонами унікальних відвідувачів кожного місяця. Сайт має гарній адаптив, який може швидко й не причиняючи незручностей користувачеві динамічно змінюватися. Також сайт має дуже швидкі запити до баз даних, що дозволяє якомога швидше отримати бажану інформацію. Якщо у користувача не вистачає часу, або слабке підключення до інтернету, даний сайт може надати можливість, в реальному часі, бачити що відбувається в матчі в реальному часі. Дана функція реалізована за допомогою підключення до сокета. Користувач, у візуальному форматі, бачить що зараз відбувається на карті, з детальною інформацією, статистикою, що дозволяє не сильно навантажувати систему, а також економити інтернет з'єднання.

До переваг даного сервісу можна віднести:

1. Відображення детальної інформації про конкретну гру, конкретної команди в реальному часі.
2. Зручний перелік змагань, які зараз проводяться або будуть проводитися в майбутньому, а також окремий список змагань які вже відбулися.
3. Відображення усіх матчів (які зареєстровані на цьому сайті), які зараз проводяться.
4. Велика база повторів ігор, які можуть бути завантажені будь-яким користувачем.
5. Детальна статистика по кожному матчу, турніру, гравцю.
6. Інтуїтивно зрозумілий інтерфейс для користувача.
7. Швидка обробка запитів.
8. Глобальна статистика по кожному гравцю за всі його змагання, порівняння статистики з іншим гравцем. Також ця статистика має певні фільтри, за якими можна обрати відповідні критерії, за якими буде здійснюватися порівняння гравців й так далі.
9. Сервіс має прямі посилання на показ матчі, які прямо зараз проходять.
10. Адаптивність.

11. Комплексність — HLTV надає інформацію про всі аспекти кіберспортивної індустрії, включаючи турніри, команди, гравців, новини, статистику тощо.

12. Обширна база даних — HLTV має велику базу даних, в якій міститься інформація про багатьох кіберспортивних турнірів та гравців, що дозволяє користувачам отримувати повну та докладну інформацію про турніри та гравців.

13. Популярність — HLTV є одним із найпопулярніших ресурсів кіберспорту у світі, що гарантує йому велику кількість користувачів та безліч джерел інформації.

14. HLTV оновлюється щодня, тому користувачі можуть отримувати останню інформацію про турніри, команди та гравців.

15. Безкоштовний доступ — HLTV є безкоштовним ресурсом, що робить його доступним для всіх користувачів.

До недоліків застосунку відносяться:

1. Застарілий інтерфейс даного сервісу
2. Використання не актуальної версії React.js.
3. Деякі дані можуть бути неточними — HLTV отримує дані від різних джерел, і іноді вони можуть бути неточними, що може призвести до невірної статистики гравців та команд.
4. Обмежена кількість ігор, що підтримуються — HLTV фокусується в основному на грі Counter-Strike, тому не всі кіберспортивні ігри та турніри можуть бути представлені на цьому ресурсі.
5. Не завжди швидке оновлення даних — HLTV іноді не оновлює свою статистику та дані так швидко, як хотілося б користувачам, що може бути проблематичним для тих, хто хоче отримати актуальну інформацію про змагання.

FaceIt

FaceIt — веб-ресурс який збирає поціновувачів певних ігор, на сайті представлений перелік з багатьох найпопулярніших відео ігор таких як: Counter-Strike: Global Offensive, Dota 2, Overwatch, PUBG, LOL, Valorant й так далі. Для того, щоб почати використовувати даний веб-сайт потрібно пройти реєстрацію [4]. Після реєстрації користувачу буде запропоновано обрати гру, можливо прив’язати один аккаунт до декількох ігрових дисциплін. Коли користувач обирає гру, його буде перекинуто на головну сторінку сайту тої гри, яку він обрав (див. Рис. 7).

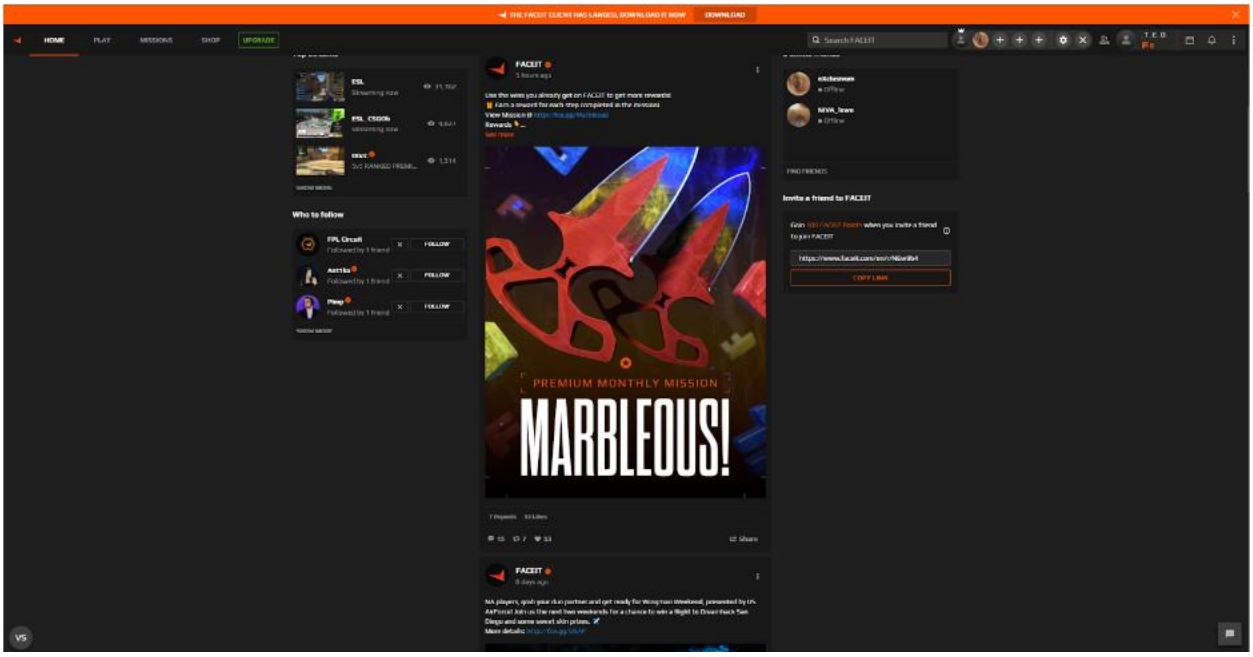

Рис. *7 Головна сторінка сайту*

Також сервіс може запропонувати преміум підписку, приклад такої підписки зображено на рисунку 8. До переваг платної підписки входять деякі привілеї, такі як:

1) найкращі сервери;
2) більш детальна статистика по картам;
3) індивідуальній статистиці;
4) доступ до ліг (при досягненні певного рівня навичок) де грають про гравці;

5) кастомний вибір карт;

6) на аккаунті більше не буде реклами (поки дійсна підписка).

Для більш комфортної гри, та для охоплення великого пласту користувачів компанія орендує сервери по всьому світу. Сервера розташовані так, щоб вони покривали якомога більше користувачів. Вони покривають Європу, частину Азії, а також Америку. Кожного разу, коли гравці починають шукати матч, програма автоматично підготує сервер, який буде оптимальним варіантом для всіх, хто зайшов на матч. Сервера підбираються за допомогою алгоритму який виявляє найбільш стійкий та сервер з найменшою затримкою.

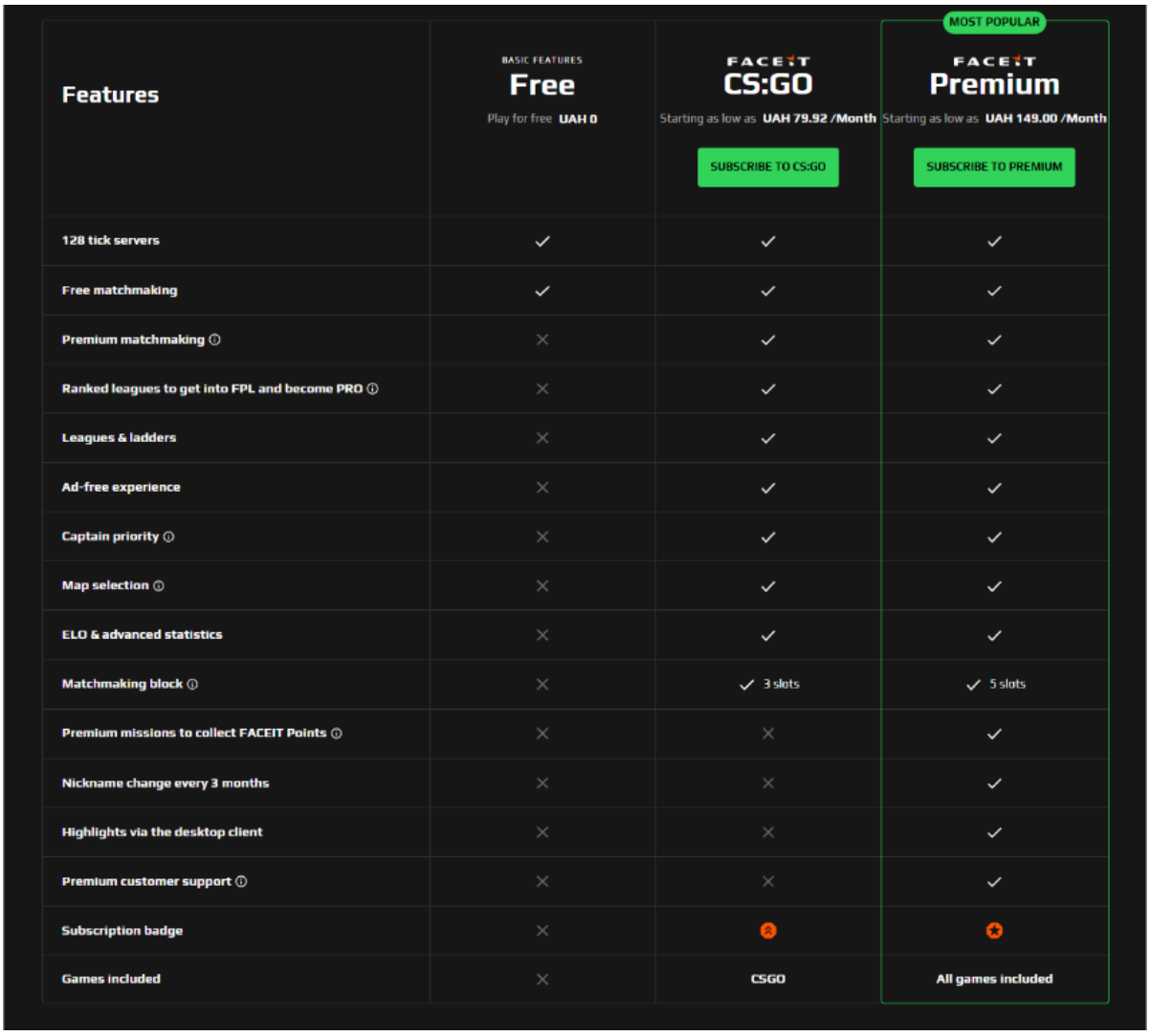

Рис. *8 Платна підписка на сервіс FaceIt*

Перевагами даного сервісу є:

1. Доступний інтерфейс для користувачів, навіть для тих, хто зайшов на даний сервіс в перший раз.
2. Можливість взяти участь в матчах з своїми друзями.
3. Зручне розташування серверів.
4. Зручна технічна підтримка, яка завжди відповідає вчасно. Також є боти, які можуть вже відповідати на питання, які являються шаблонними
5. Боротьба з нечесними гравцями. FaceIt розробила свою власну систему для боротьби з читерами, яка є однією з найкращих систем виявлення стороннього софту.
6. Гарний адаптив сайта.
7. Можливість перегляду повтору своєї гри.
8. Регулярні турніри з допуском будь-якого гравця.
9. Система рейтингів та матчмейкінг — FaceIT має унікальну систему рейтингів, яка допомагає гравцям знаходити суперників зі схожим рівнем гри, що забезпечує більш рівні та цікаві поєдинки.
10. Якісний та стабільний сервіс — FaceIT пропонує стабільну та надійну платформу для організації та участі у змаганнях.
11. Велика кількість гравців та турнірів — завдяки своїй популярності, FaceIT пропонує велику кількість турнірів та гравців з усього світу, що дозволяє знайти гідних суперників та отримати максимальний ігровий досвід.
12. Велика кількість функцій та інструментів — сервіс пропонує безліч функцій та інструментів, які дозволяють налаштовувати ігровий процес та керувати турнірами, наприклад, створювати команди, налаштовувати правила змагань, підключатися до голосового чату та багато іншого.
13. Призові фонди та нагороди — FaceIT пропонує вигравати реальні призи та нагороди за участь у турнірах, що робить гру більш захоплюючою та мотивуючою.

Недоліки цього сервісу:

1. Довге завантаження сторінок.

2. Проблеми з читерами та шахраями — у FaceIt були помічені випадки використання читерських програм та інших шахрайських дій. Це може негативно позначитися на репутації сервісу та досвіді гри для чесних користувачів.

3. Вимога підписки на преміум-акаунт для доступу до деяких функцій — деякі функції та можливості доступні лише для користувачів, які придбали преміум-акаунт, що може стати проблемою для деяких гравців.

ESL Play

ESL Play — це онлайн-платформа для гравців в електронні спортивні ігри, яка надає можливість брати участь у змаганнях, турнірах та матчмейкінгу [5]. Вона є однією з найбільших платформ для проведення турнірів та змагань з різних електронних спортивних ігор у світі. Головна сторінка сайту зображена на рисунку 9.

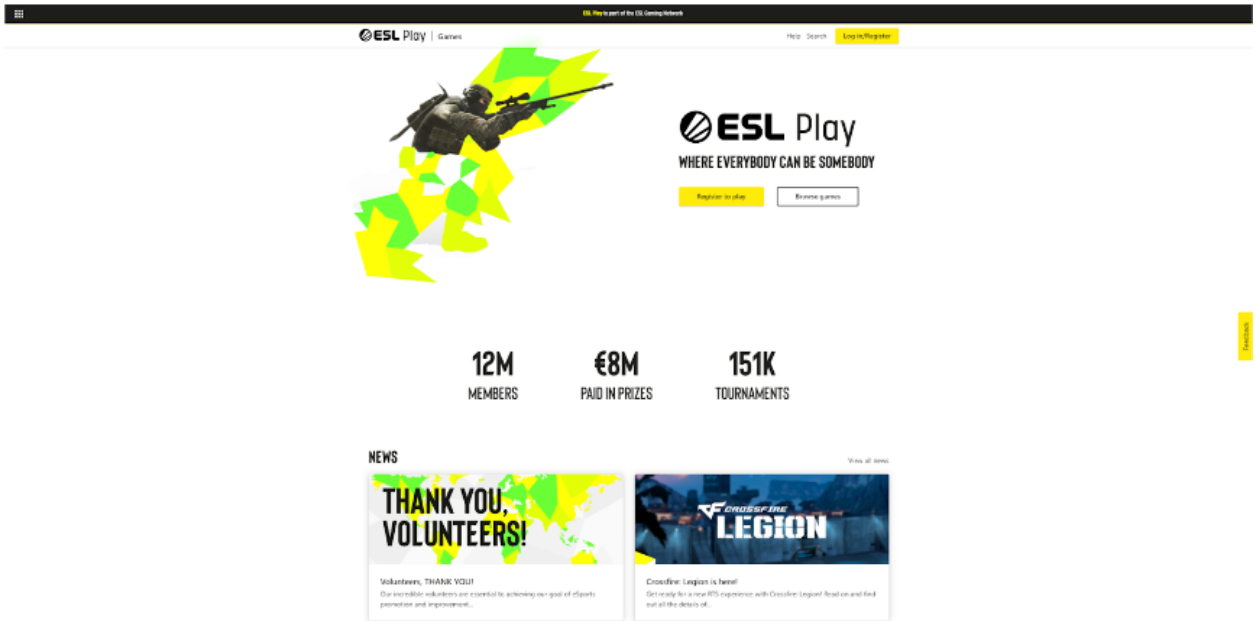

Рис. *9 Головна сторінка сайту ESL Play*

ESL Play дозволяє гравцям брати участь у змаганнях з таких ігор, як Counter-Strike: Global Offensive, Dota 2, League of Legends, FIFA, Hearthstone та багато інших. На платформі також доступні різні формати турнірів,

включаючи поодинокі битви, командні змагання та ліги. Приклад сторінки з турнірами з однієї гри зображено на рисунку 10.

Організатори турнірів та змагань можуть використовувати ESL Play для створення та керування своїми власними турнірами. Вони можуть обирати формати змагань, призові фонди, кількість учасників та багато іншого. Крім того, ESL Play пропонує інструменти для трансляції турнірів, щоб глядачі могли стежити за матчами в режимі реального часу.

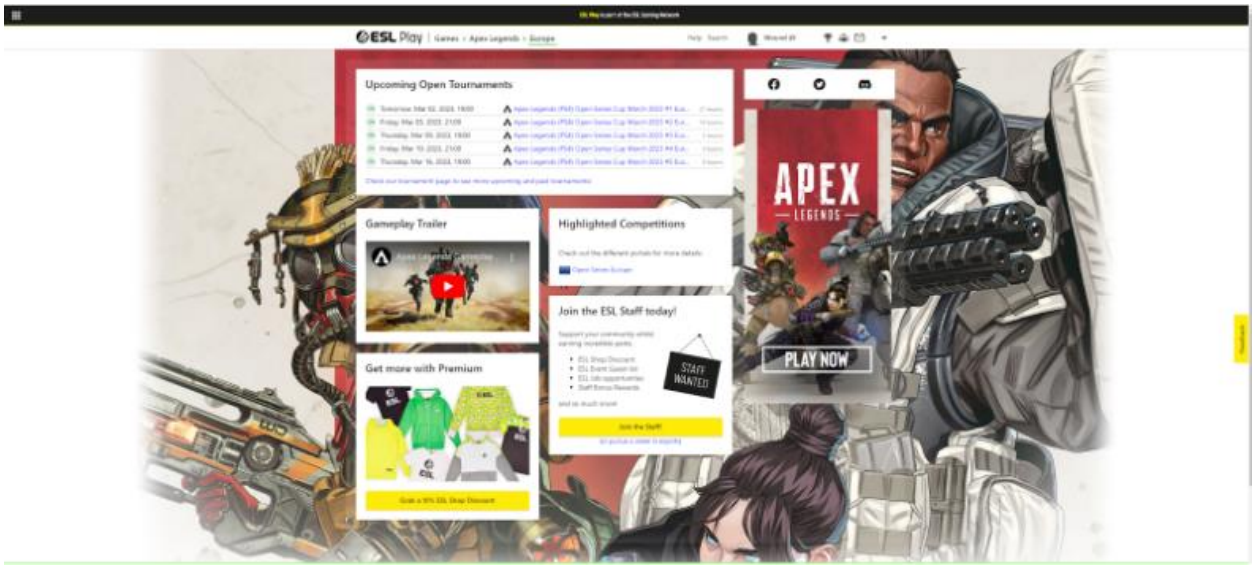

Рис. *10 Сторінка з турнірами*

ESL Play також пропонує систему рейтингів для гравців та команд, які беруть участь у турнірах. Це допомагає гравцям відстежувати свій прогрес та порівнюватися з іншими учасниками. також доступний перегляд сторінки з командами, а також можна подивитися свою команду, з кого вона складається й перейти на профіль користувача. Приклад команди зображений на рисунку 11.

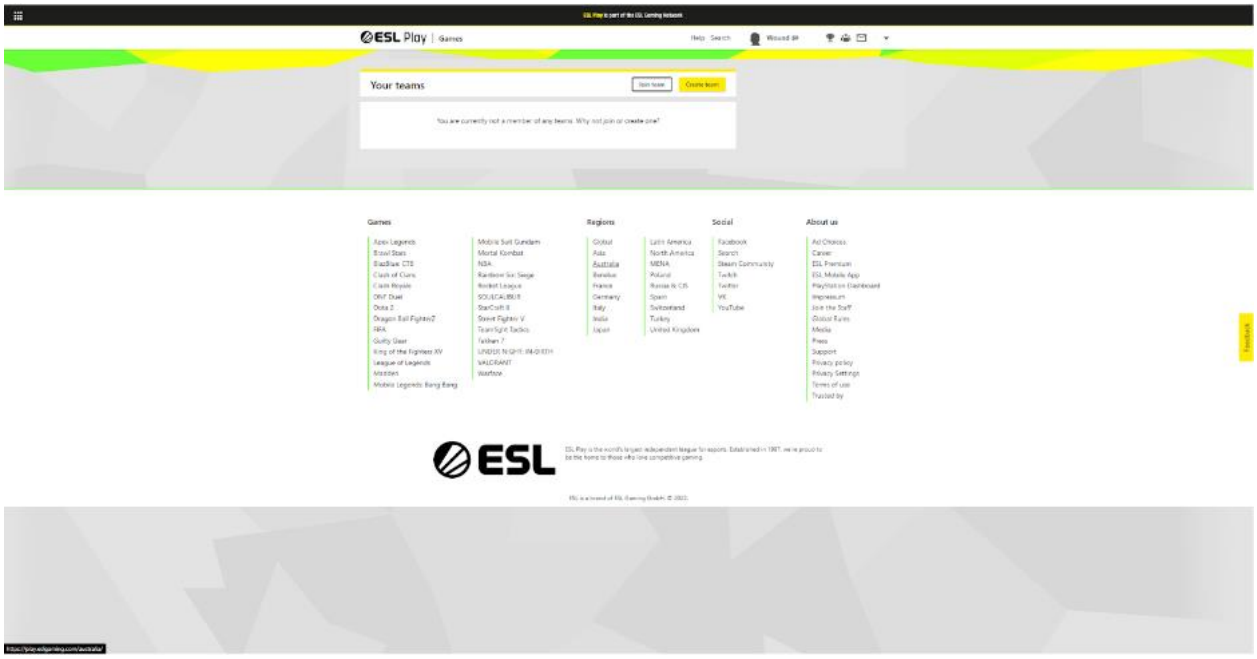

Рис. *11 Сторінка команди*

В цілому, ESL Play є однією з найпопулярніших та найзручніших онлайн-платформ для організації та участі в електронних спортивних змаганнях.

Переваги даного сервісу:

1. Велика кількість ігор та турнірів — ESL Play надає доступ до широкого спектра ігор та турнірів, що робить його дуже привабливим для гравців з усього світу.
2. Зручність використання — інтерфейс та функціональність ESL Play зрозумілі та зручні для використання, що робить його доступним для гравців усіх рівнів.
3. Система рейтингів та рангів — ESL Play має систему рейтингів та рангів, які дозволяють гравцям відстежувати свій прогрес та рівень гри.
4. Можливість створення власних турнірів — користувачі можуть створювати свої турніри та запрошувати учасників.
5. Хороша підтримка — ESL Play має хорошу підтримку, яка швидко та ефективно вирішує проблеми користувачів.

Недоліки даного сервісу:

1. Не завжди найкраща якість з'єднання — іноді користувачі скаржаться на погану якість з'єднання під час використання ESL Play.
2. Проблеми з фальсифікацією результатів — деякі користувачі можуть спробувати обдурити систему, щоб підвищити свій рейтинг чи виграти призові гроші.
3. Обмежені можливості безкоштовної версії — деякі функції та можливості доступні лише для користувачів з передплатою преміум-акаунту.
4. Конкуренція через те, що ESL Play є одним з найпопулярніших сервісів для організації електронних спортивних змагань, конкуренція може бути досить жорсткою, що ускладнює досягнення високих результатів.

1.3 Постановка завдання

Метою кваліфікаційної роботи є створення веб-додатку реєстрації на події. Впровадження даної системи для організацій, які проводять турніри, може полегшити організацію турнірів та проведення матчів.

Для того, щоб досягти поставлено мети — потрібно вирішити певну кількість задач, такі як:

1. Спроектувати схему бази даних, створити базу даних, в нашому випадку це буде база даних MongoDB.
2. Розробити інтерфейс користувача.
3. Створити фронтенд частину застосунку з використанням технології React.js.
4. Створити бекенд сатину застосунку з таким стеком технологій: Node.js, NestJs, mongoose, TypeScript.
5. Розробити систему безпеки, коли користувач не може виконувати певні дії без попередньої реєстрації, або логіну, на веб-сайті. Також задати права доступу для певних роутів, наприклад коли користувач не зможе зробити запит, якщо у нього немає певної ролі, наприклад ролі “Admin”.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Аналіз сучасних платформ для розроботки серверних частин застосунку

На поточний час у ринку IT існують потреби у створені серверних застосунків, які будуть забезпечувати користувач максимальну швидкодію при виконанні запитів, зручно розгоратися на серверах замовника, мати стабільний зв’язок з клієнтом, бути стійким до помилок і найголовніше бути захищеним. Зараз на ринку основними конкурентами є такі платформі для розробки та фреймворки як .Net, Node.js, Django, PHP.

.NET

Технологія .NET є фреймворком розробки, розробленим компанією Microsoft. Вона містить в собі мови програмування, такі як C#, Visual Basic та F#, і надає можливість розробляти серверні застосунки, веб-додатки, мобільні додатки та інше.

.NET часто використовується для створення підприємницьких рішень, корпоративних додатків, веб-серверів і веб-додатків [6].

Переваги використання .NET:

1. Велика підтримка та екосистема, що забезпечує широкий спектр інструментів і бібліотек.
2. Висока продуктивність та швидкодія.
3. Підтримка масштабування та паралельної обробки завдань.
4. Хороша підтримка мови C#, яка вважається елегантною і легко зрозумілою для розробників.

Недоліки у використанні .NET:

1. .NET є платформо залежною і підтримується лише на операційних системах Windows, що обмежує переносимість коду.
2. Інколи вимагає великих ресурсів для роботи.
3. Є платним, хоча є безкоштовна версія .NET Core з відкритим кодом.

Node.js

Технологія Node.js — це виконавче середовище, засноване на движку JavaScript V8 веб-переглядача Google Chrome [7]. Воно дозволяє виконувати JavaScript на серверній стороні та розробляти масштабовані та ефективні серверні застосунки.

Використання: Node.js широко використовується для створення веб-серверів, API, розширень браузера та інших серверних застосунків.

Переваги використання Node.js:

1. Висока продуктивність і швидкодія завдяки однопотоковій не блокуючій I/O моделі.
2. Масштабованість і можливість обробки багатьох одночасних з'єднань.
3. Велика кількість зовнішніх модулів та бібліотек, доступних через пакетний менеджер npm.
4. JavaScript є популярною мовою програмування з великою спільнотою розробників.

Недоліки використання Node.js:

1. Не всі бібліотеки можуть бути оптимізовані для однопотокової моделі, що може обмежувати деякі сценарії використання.
2. Не підтримується безпосередньо багатопотоковість, що може ускладнювати деякі сценарії розробки.

Django

Технологія Django — це високорівневий фреймворк розробки веб-додатків на мові Python [8]. Він надає структуру та компоненти для швидкої розробки веб-додатків.

Використання: Django популярний для розробки веб-сайтів, веб-додатків, електронної комерції та інших проектів, які вимагають веб-функціональності.

Переваги у використані Django:

1. Швидкість розробки завдяки високому рівню абстракції та великому набору готових компонентів.

2. Вбудована адміністративна панель, яка спрощує управління та адміністрування додатків.
3. Є платформо залежним, що забезпечує переносимість на різних операційних системах.
4. Підтримка ORM (Object-Relational Mapping), що спрощує взаємодію з базами даних.

Недоліки у використанні Django:

1. Потребує знання мови програмування Python.
2. Менша продуктивність порівняно з деякими іншими фреймворками.
3. Менша гнучкість, оскільки Django надає структуру та конвенції, які можуть обмежувати відхилення від стандартного підходу.

PHP

Технологія PHP — це скриптова мова програмування, яка спеціально призначена для розробки веб-додатків [9]. Вона виконується на стороні сервера і може використовуватись для створення динамічних веб-сторінок та серверних застосунків.

Використання: PHP використовується для розробки різноманітних веб-додатків, включаючи блоги, електронну комерцію та соціальні мережі.

Переваги використання технології PHP:

1. Легкість вивчення та використання, особливо для розробників, які вже знайомі з HTML.
2. Велика спільнота розробників та велика кількість доступних ресурсів і бібліотек.
3. Хороша підтримка баз даних із популярними системами, такими як MySQL та PostgreSQL.

Недоліки використання технології PHP:

1. Не завжди вважається надійним та безпечним, особливо при недостатній обробці вхідних даних.
2. Не так ефективний та швидкий, як деякі інші мови.
3. Менша гнучкість порівняно з деякими іншими фреймворками.

Висновок:

Хоча кожна з розглянутих платформ має свої переваги та недоліки, Node.js зазвичай виступає як найкращий вибір для багатьох сучасних серверних застосунків, особливо з урахуванням вимог до продуктивності, масштабованості та ефективності.

2.2 Аналіз сучасних фреймворків для написання серверних застосунків на Node.js

На даний момент основними конкурентами при розробці серверних застосунків з використанням технології Node.js є Express і NestJS.

Express

Express — це мінімалістичний та гнучкий фреймворк для розробки веб-додатків на Node.js [10]. Він надає простий інтерфейс та широкі можливості для створення серверів, маршрутизації, обробки запитів та відправки відповідей.

Використання: Express використовується для створення різноманітних веб-додатків, від простих API до повноцінних веб-додатків з рендерінгом шаблонів.

Переваги:

1. Легкий у вивченні та використанні, має мінімалістичний дизайн.
2. Широка спільнота розробників та велика кількість ресурсів для навчання та підтримки.
3. Гнучкий та налаштовуваний, дозволяє реалізовувати різноманітні сценарії розробки.

Недоліки:

1. Відсутність вбудованих функцій для організації проекту та роботи з базами даних.
2. Менша структурованість порівняно з іншими фреймворками.

NestJS

NestJS — це прогресивний фреймворк для розробки на Node.js, який використовує TypeScript і виконується на сервері [11]. Він пропонує модульну архітектуру, підтримку Dependency Injection та вбудовані функції для розробки веб-додатків та API.

Використання: NestJS використовується для створення масштабованих веб-додатків, API та мікросервісів з підтримкою структурованої розробки та організації коду.

Переваги:

1. Заснований на TypeScript, що дозволяє писати безпечний та структурований код з підтримкою сучасних функцій JavaScript.
2. Модульна архітектура та підтримка Dependency Injection сприяють розширенню та тестуванню додатків.
3. Вбудовані функції, такі як міжпроцесова комунікація, автентифікація, валідація даних, обробка запитів та відповідей.
4. Інтеграція з ORM (TypeORM, Sequelize) для роботи з базами даних.
5. Автоматична генерація документації API за допомогою Swagger.

Недоліки:

1. Потребує додаткового часу та зусиль для навчання TypeScript та підходів NestJS порівняно з Express.

Чому краще використовувати NestJS: NestJS є потужним фреймворком, який забезпечує структуровану розробку, підтримку TypeScript, вбудовані функції та легку інтеграцію з базами даних. Він спрощує розробку серверних застосунків, забезпечуючи готові рішення для багатьох типових завдань, таких як маршрутизація, обробка запитів, автентифікація та валідація даних. Завдяки модульній архітектурі та підтримці Dependency Injection, NestJS сприяє покращенню розширюваності та тестування додатків. Тому, для розробки сучасних серверних застосунків, NestJS є більш привабливим вибором порівняно з Express.

2.3 Технологія для написання фронтенд частини застосунку React

React — це бібліотека JavaScript для розробки фронтенд-частини веб-застосунків [12]. Вона була розроблена компанією Facebook з метою спрощення створення інтерактивних та швидких користувацьких інтерфейсів.

Основні використання React включають:

1. Розробка односторінкових додатків (Single-Page Applications, SPA): React дозволяє створювати динамічні та ефективні SPA, де зміни відбуваються без перезавантаження сторінки.
2. Розробка мобільних додатків: За допомогою фреймворків, таких як React Native, можна використовувати React для побудови крос-платформних мобільних додатків.
3. Розробка універсальних додатків: React може бути використаний для рендерингу на сервері, що дозволяє створювати універсальні додатки, що працюють як на клієнті, так і на сервері.

Переваги React порівняно з конкурентами, такими як Vue.js, Next.js та AngularJS, включають [14-16]:

1. Продуктивність: Використання віртуального DOM та ефективного алгоритму оновлення дозволяє забезпечити швидку роботу та ефективне використання ресурсів.
2. Широкий екосистема: React має велику спільноту розробників, що сприяє наявності багатьох сторонніх бібліотек, модулів та інструментів підтримки.
3. Гнучкість: React дає розробникам більше свободи у виборі інших технологій та бібліотек для побудови повноцінного стеку розробки.
4. Легка навчання: Синтаксис React базується на JavaScript та використовує JSX, який дозволяє розробникам працювати з компонентами, що схожі на HTML, зменшуючи відхилення від звичного підходу до розробки веб-інтерфейсів.

2.4 TypeScript для написання фронтенд та бекенд частин застосунку

TypeScript — це мова програмування, що розширює JavaScript, призначена для розробки масштабованих та підтримуваних програм [13].

TypeScript був розроблений командою відділенням Microsoft, зокрема Андерсом Хейлсбергом. Головною метою TypeScript було розширення JavaScript шляхом додавання статичної типізації та нових функцій, які полегшують розробку складних проектів та покращують інструменти розробника.

Переваги використання TypeScript для серверних застосунків:

1. Статична типізація: TypeScript дозволяє оголошувати типи для змінних, параметрів та повернених значень функцій, що допомагає виявляти помилки на ранніх етапах розробки та полегшує рефакторинг коду.
2. Розширена підтримка ООП: TypeScript підтримує об'єктно-орієнтований підхід з класами, успадкуванням, інтерфейсами, модифікаторами доступу та іншими концепціями, що дозволяють писати більш структурований та перевикористовуваний код.
3. Інтеграція з існуючим JavaScript кодом: TypeScript може використовувати існуючий JavaScript код без проблем, що дає можливість поступово впроваджувати TypeScript в існуючі проекти.
4. Інструменти розробника: TypeScript надає розширений функціонал для підтримки розробки, включаючи автодоповнення коду, перевірку типів під час редагування, засоби рефакторингу та інші.

Недоліки використання TypeScript для серверних застосунків:

1. Витрати на час: Використання TypeScript вимагає додаткового часу для написання та оголошення типів, що може призвести до затримок у розробці.
2. Навчання та пороговий рівень: Якщо команда розробників не має досвіду з TypeScript, може знадобитися час для вивчення нових концепцій та правил мови.

3. Великі розміри файлів: Файли TypeScript можуть мати більший обсяг порівняно з відповідними JavaScript-файлами, оскільки вони містять типову інформацію.
4. Переваги використання TypeScript для фронтенд-частини застосунків:
5. Статична типізація: Забезпечує зручне виявлення та усунення помилок на етапі розробки.
6. Компіляція: TypeScript код компілюється до JavaScript, що дозволяє його виконувати на будь-якому сучасному браузері.
7. Розширена підтримка ООП: TypeScript підтримує об'єктно-орієнтовану парадигму, що полегшує розробку складних фронтенд-інтерфейсів.
8. Інструменти та екосистема: TypeScript має багатий вибір засобів розробки, включаючи розширення для редакторів коду, системи збирання модулів та багато іншого.

Недоліки використання TypeScript для фронтенд-частини застосунків:

1. Пороговий рівень та навчання: Як і для серверних застосунків, використання TypeScript може вимагати часу для навчання та ознайомлення зі специфічними правилами та патернами.
2. Збільшений обсяг коду: TypeScript вимагає додаткового написання типів, що може призвести до збільшення обсягу коду порівняно з JavaScript.

Загалом, TypeScript є потужним інструментом для розробки серверних та фронтенд-частини застосунків, але його вибір залежить від вимог проекту, наявності досвіду команди та специфіки роботи з мовою програмування.

3 РОЗРОБКА

3.1 Опис предметної області

Кіберспорт є однією з найшвидше зростаючих галузей розваг і спорту. Він об'єднує професійних гравців, які змагаються у віртуальних світах різних комп'ютерних ігор. Ці змагання включають турніри, ліги та чемпіонати, які залучають мільйони глядачів та прихильників з усього світу.

З огляду на швидкий розвиток кіберспорту, існує потреба в ефективних системах реєстрації та управління учасниками кіберспортивних подій. Ручна обробка реєстраційних форм, управління даними гравців та керування розкладом подій можуть бути складними та часоємкими процесами для організаторів. Тому автоматизація цих процесів є критично важливою для ефективного функціонування турнірів та ігрових заходів.

У даній кваліфікаційній роботі пропонується розробка веб-додатку з використанням фреймворку NestJS для реєстрації на кіберспортивні події. NestJS є прогресивним фреймворком, побудованим на основі Node.js, що надає зручний інструментарій для розробки серверних додатків. Він пропонує модульну архітектуру, засновану на принципах SOLID, і надає велику кількість інструментів для розробки API та веб-додатків.

Розроблений веб-додаток має надати користувачам можливість зареєструватись на кіберспортивні події, вказавши необхідні дані про себе, такі як ім'я, прізвище, електронна пошта, інформація про гру, в якій вони бажають прийняти участь тощо. Додаток повинен забезпечити перевірку правильності введених даних та унікальність користувачів. Після реєстрації, користувачі повинні мати змогу переглянути свою інформацію, редагувати її або відмінити реєстрацію на певну подію.

Розробка веб-додатку на базі фреймворку NestJS дозволить забезпечити надійну та безпечну реєстрацію користувачів на кіберспортивні події. Використання Node.js в поєднанні з NestJS дозволяє створювати швидкі та

масштабовані серверні додатки, які відповідають сучасним стандартам безпеки та ефективності.

Однією з ключових переваг фреймворку NestJS є його модульна архітектура, яка дозволяє розбити додаток на незалежні модулі. Це спрощує розробку, тестування та підтримку коду, оскільки кожен модуль відповідає за свою функціональність. Наприклад, можна створити окремий модуль для автентифікації користувачів, що дозволить забезпечити безпеку даних та контроль доступу до функцій додатку.

Для забезпечення перевірки правильності введених даних та унікальності користувачів, веб-додаток може використовувати валідацію даних на основі вбудованих механізмів NestJS або сторонніх бібліотек, таких як Joi або class-validator. Це дозволить забезпечити консистентність даних та уникнути некоректного введення.

Крім того, розроблений веб-додаток повинен надати зручний інтерфейс для користувачів, щоб вони могли легко переглядати свою інформацію, редагувати її або скасовувати реєстрацію на певну подію. Використання фронтенд-фреймворків, таких як React або Angular, у поєднанні з NestJS, може забезпечити інтуїтивно зрозумілий та добре виглядаючий інтерфейс для користувачів.

3.2 Архітектура системи

Веб-додаток для реєстрації на кіберспортивні івенти можна розбити на декілько частин, на серверну частину й фронтенд частину застосунку.

Серверна частина застосунку, або Back-end, відповідає за обробку запитів від користувачів а також й за взаємодію з базою даних. Дана частина відповідає за логіку пошуку, авторизацію, реєстрацію, обробки запитів а також усіх інших бізнес-процесів, які пов’язані з бізнес моделлю веб-додутку для реєстрації на кіберспортивні івенти. Також серверна частина відповідає за повну взаємодію з базою даних, в якій зберігається вся необхідна інформація

для коректної роботи застосунку, така як список користувачів, команд, матчів й тому подібне. Також серверна сторона повинна забезпечувати ефективне, безперебійне а також надійне управління даними.

Фронтенд частина застосунку відповідає за коректну взаємодію з користувачем, відправка запитів на серверну частину застосунку, отримання відповідної інформації з сервера. Фронтенд частина повинна забезпечувати зручну й інтуїтивно зрозумілу для користувача систему з відповідним дизайном. Фронтенд частина застосунка повинна включати реалізацію такі ключеві моменти як авторизація, реєстрація, перегляд матчів й тому подібне.

3.3 Функціональні вимоги системи

Для того щоб забезпечити коректну роботу веб-застосунку з реєстрації на кіберспортивні івенти з використанням таких технологій як Node.js, NestJs, React а також інтеграції бази даних MongoDB потрібно розробити певні функціональні вимоги, їх список наведений нижче.

1. Аутентифікація користувача:
    1. Веб-застосунок повинен мати можливість реєстрації нового користувача, щоб отримати повноцінний доступ до функцій даного сервісу.
    2. Користувачі повинні мати змогу зайти в вже, попередньо створений аккаунт.
    3. Користувачі повинні мати можливість в будь-який момент мати змогу вийти з свого аккаунту і зайти в інший аккаунт.
2. Система безпеки:
    1. Система повинна не давати допуску до певних роутів, на серверній частині застосунку, якщо користувач попередньо не пройшов реєстрацію на сайті, або не увійшов до свого аккаунта у веб-застосунку.

2. Система повинна не давати допуск до певних роутів, якщо користувач не має відповідної ролі, наприклад якщо у користувача ролі звичайного користувача — він не зможе скористатися тим роутом, який призначений для адміністратора.

3. Редагування профайлу користувача:
    1. Користувач повинен мати змогу змінювати інформацію свого профайлу.
    2. Серверна частина застосунку не повинна дозволяти редагувати користувачу профайл, якщо його токен був прострочений.
    3. Фронтенд частина повинна автоматично оновити інформацію профіля, після того, як користувач оновив дані свого аккаунта.
4. Отримання списку користувачів застосунку:
    1. Серверна частина застосунку повинна надавати доступ до цього роуту, тільки користувачам, які мають роль адміністратора застосунку.
    2. Фронтенд частина застосунку повинна відображати дану сторінку лише в тому випадку, якщо кристувач має роль адмін.
    3. Фронтенд частина повинна відображати список користувачів, який приходить з серверної частини застосунку, з коректним і повним відображенням інформації з сервера.
5. Видалення користувача:
    1. Серверна частина застосунку повинна надавати доступ до цього роуту тільки користувачам, які мають роль адміністратора застосунку.
    2. Фронтенд частина застосунку повинна відображати дану сторінку лише в тому випадку, якщо кристувач має роль адмін.
    3. При натисканні на кнопку видалення користувача й ого видалення без помилок, фронтенд частина повинна надіслати запит на отримання оновленого списку користувачів й коректо його відобразити.
6. Створення команди:

1. Серверна частина застосунку повинна надавати доступ до цього роуту тільки користувачам, які мають роль адміністратора застосунку.
2. Фронтенд частина застосунку відображає даний компонент, якщо кристувач має роль адмін.
3. Після створення команди веб-застосунок насилає запит й отримує оновлений список команд.

7. Отримання списку команд:
    1. Фронтенд частина застосунку відправляє запит на серверну частину з певними параметрами, такими як ліміт на отримання команд, а також з якої команди (за номером) повинно почати рахувати їх ліміт.
    2. Фронтенд частина повинна не давати змогу користувачу отримувати нові команди, якщо команди вже скінчились.
8. Видалення команди:
    1. Серверна частина застосунку повинна надавати доступ до цього роуту тільки користувачам, які мають роль адміністратора застосунку.
    2. Фронтенд частина застосунку відображає даний компонент, якщо кристувач має роль адмін.
    3. При натисканні на кнопку видалити, якщо видалення пройшло успішно, користувача повинно перекинути на сторінку сайту, де відображено список всіх команд.
9. Редагування команди:
    1. Серверна частина застосунку повинна надавати доступ до цього роуту тільки користувачам, які мають роль адміністратора застосунку.
    2. Фронтенд частина застосунку відображає даний компонент, якщо кристувач має роль адмін.

3. При заповнені поля з новою назвою й натискання кнопки оновлення назви команди, фронтенд частина автоматично робить запит до серверної частини й відображає оновлену інформацію про цю команду.

10. Можливість увійти в команду:
    1. Кожен користувач повинен мати змогу увійти до будь-якої команди, якщо вона ще повністю не укомплектована. Якщо в команді присутні вже 5 користувачів, фронтед частина й серверна частина повинна обробити даний випадок й видати помилку.
11. Можливість видалення гравця з команди:
    1. Серверна частина застосунку повинна надавати доступ до цього роуту тільки користувачам, які мають роль адміністратора застосунку.
    2. Фронтенд частина застосунку відображає даний компонент, якщо кристувач має роль адмін.
    3. Після того, як адміністратор видалив користувача, веб-застосунок повинен автоматично оновити інформацію про команду, яку зараз спостерігає адміністратор.
12. Отримання інформації про свою команду:
    1. Фронтенд частина повинна зберігати токен, який приходить при реєстрації, або при логіні на сайті, й надсилати запит на серверну частину, для того, щоб серверна частина розуміла про яку команду потрібно достати інформацію.
13. Створення матчу:
    1. Серверна частина застосунку повинна надавати доступ до цього роуту тільки користувачам, які мають роль адміністратора застосунку.
    2. Фронтенд частина застосунку відображає даний компонент, якщо кристувач має роль адмін.

3. Адміністратор повинен обрати дві команди а також дату проведення матчу, після цього натиснути на кнопку додати матч. Якщо не буде колізій, тобто одна команда не буду в той же час приймати участь в іншому матчі, цей матч повинен відобразитися у списку загальнодоступних матчів.

14. Видалення матчу:
    1. Серверна частина застосунку повинна надавати доступ до цього роуту тільки користувачам, які мають роль адміністратора застосунку.
    2. Фронтенд частина застосунку відображає даний компонент, якщо кристувач має роль адмін.
    3. Після того, як адміністратор натиснув на кнопку видалення.
15. Визначення переможця матчу:
    1. Серверна частина застосунку повинна надавати доступ до цього роуту тільки користувачам, які мають роль адміністратора застосунку.
    2. Фронтенд частина застосунку відображає даний компонент, якщо кристувач має роль адмін.
    3. Адміністратор обирає переможця, після чого веб-застосунок повинен оновити список матчів.

Ці функціональні вимоги відповідають усім ключовим можливостям веб-застосунку з реєстрації на кіберспортивні івенти. Данні вимоги повинні бути реалізовані для того, щоб забезпечити усі базові потреби користувача та для ефективної роботи системи.

3.4 Проектування

Одним із етапів проектування веб-застосунку є створення діаграм використання (Use — Case) для визначення функціональності застосунка. Саме тому для візуалізації функціональності а також взаємодії між акторами

(користувачем та адміністратором) і веб-застосунком були створені відповідні Use — Case діаграми.

На рисунку 12 зображено Use — Case діаграму користувача яка включає виконання таких сценаріїв:

1. Реєстрація: користувач проводить реєстрацію в системі, для цього йому потрібно надати певну про себе інформацію (наприклад надати інформацію про свою електрону пошту, бажаний нікнейм на сайті).
2. Логін: користувач повинен мати змогу зайти на сайт, якщо він попередньо пройшов реєстрацію на сайті.
3. Перегляд профайлу: користувач повинен мати змогу переглядати свій профайл, якщо він попередньо зареєструвався у веб-застосунку.
4. Перегляд своєї команди: користувач повинен мати змогу переглядати свою команду, якщо він вже знаходиться в певній команді, якщо користувач ще не знаходиться в команді, потрібно відобразити напис що користувач зараз не в команді.
5. Отримання списку команд: користувач повинен мати змогу отримати список команд, які зараз існують у веб-застосунку.
6. Отримання списку матчів: користувач повинен мати змогу отримати список матчів, які зараз існують у веб-застосунку.
7. Перегляд інформації про команду: користувач повинен мати змогу отримати інформацію про команду, яку він бажає переглянути.
8. Перегляд інформації про матч: користувач повинен мати змогу отримати інформацію про бажаний матч.
9. Редагування профілю: користувач повинен мати змогу редагувати профайл, а саме змінювати свій нікнейм на сайті, якщо він буде унікальним — веб-застосунок оновить інформацію.
10. Приєднання до команди: користувач повинен мати змогу приєднатися до команди, якщо кількість учасників даної команди менше 5 осіб.

Рис. *12 Use — Case діаграмма користувача*

На рисунку 13 зображено Use — Case діаграму адміністратора яка включає виконання сценаріїв звичайного користувача а також додаткових сценаріїв:

1. Видалення матчів: адміністратор має змогу видаляти матчі, якщо на те є необхідність.
2. Редагування команди: адміністратор має змогу видалити команду, якщо на те є необхідність.
3. Видалення користувача з команди: адміністратор має можливість видаляти користувачів з команди.
4. Додавання нової команди: адміністратор може створити нову команду.
5. Видалення команди: при необхідності, адміністратор може створювати нові команди.

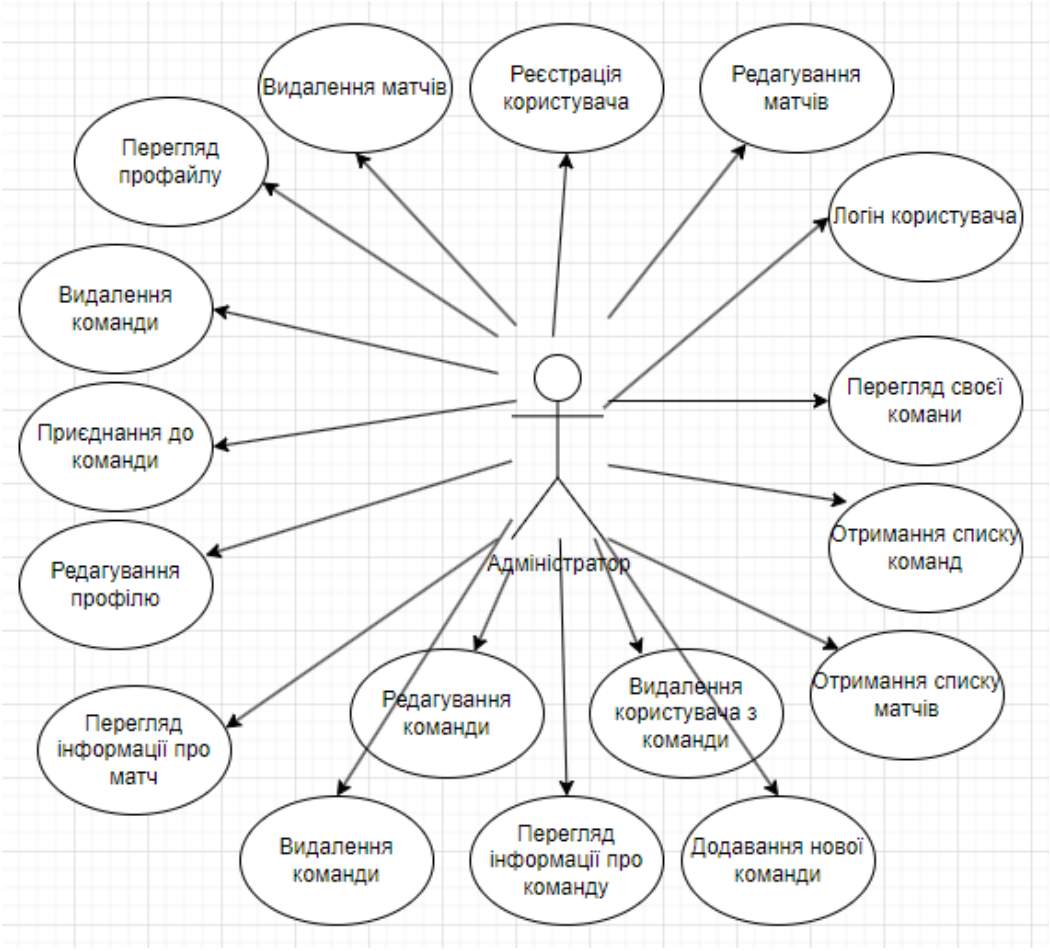

Рис. *13 Use — Case діаграмма користувача*

Розглянемо базу даних, яку використовує даний застосунок, це не реляційна база даних MongoDB. Усього в базі даних представлено 5 колекцій.

1. Колекція «Matches»
    1. Поле «_id»: унікальний ідентифікатор матча.
    2. Поле «firstTeam»: унікальний ідентифікатор першої команди.
    3. Поле «secondTeam»: унікальний ідентифікатор другої команди.
    4. Поле «startDate»:дата початку матча.
    5. Поле «winner»: унікальний ідентифікатор переможця.
2. Колекція «Players»
    1. Поле «_id» : унікальний ідентифікатор гравця.
    2. Поле «nickname»: унікальний никнейм гравця.
    3. Поле «email»: унікальний email гравця.
    4. Поле «password»: захешований пароль користувача.

5. Поле «roleId»: унікальний ідентифікатор ролі.

3. Колекція «Roles»
    1. Поле «_id» : унікальний ідентифікатор ролі.
    2. Поле «name»: унікальна назва ролі.
4. Колекція «Teams»
    1. Поле «_id» : унікальний ідентифікатор команди.
    2. Поле «name»: унікальна назва команди.
5. Колекція «Teammembers»
    1. Поле «_id» : унікальний ідентифікатор проміжної таблиці.
    2. Поле «teamId»: унікальний ідентифікатор команди.
    3. Поле «playerId»: унікальний ідентифікатор гравця.

Діаграма бази даних, яка зображена на рисунку 14, відображає структуру усіх таблиць та поля для збереження інформації. Якщо застосунок буде використовувати цю бази даних, з її поточними таблицями, — користувачам буде забезпечений швидкий доступ до необхідних даних а також швидке їх оновлення.

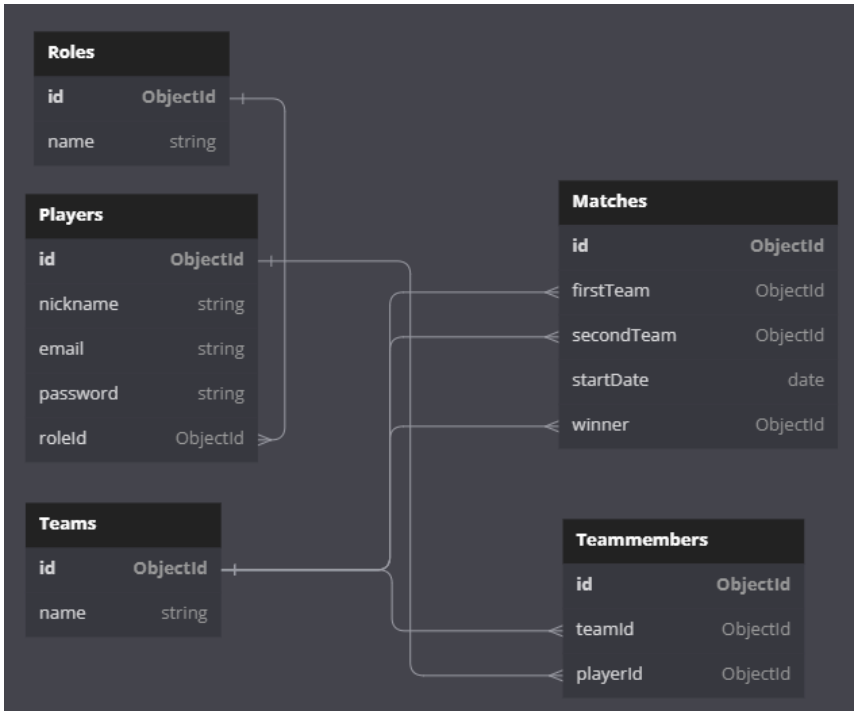

Рис. *14 Файлова структура серверної частини*

3.5 Програмна реалізація застосунка

Розглянемо структуру серверної частини застосунка, серверна частина була розроблена з використанням технології Node.js, з використанням фреймворку NestJS.

Серверна частина складається з деяких доменів, які можна побачити на рисунку 15, а саме:

1. Match domain.
2. Player domain.
3. Team domain.

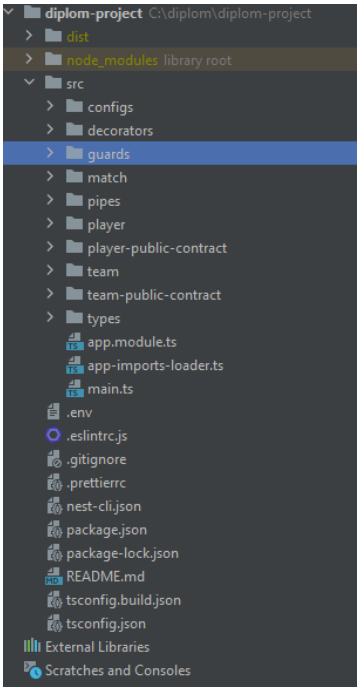

Рис. *15 Файлова структура серверної частини*

Також в структурі є папки, до яких входить напис «public-contact», це зроблено для того, щоб можна було незалежно спілкуватися між різними доменами.

В папці під назвою «guards» зберігаються самописні гуарди. В фреймворку NestJS, guard (охоронець) — це механізм, який дозволяє виконувати фільтрацію та перехоплення запитів перед тим, як вони досягнуть

обробників маршрутів. Гарди дозволяють контролювати доступ до маршрутів та виконувати різні перевірки безпеки перед обробкою запиту.

Guard може бути використаний для виконання різноманітних завдань, таких як перевірка прав доступу, автентифікація, авторизація, валідація даних та багато іншого. Він забезпечує можливість втручатись в обробку запиту до того, як він дійде до відповідного обробника маршруту, і при необхідності виконувати додаткові дії.

Гуарди можуть бути застосовані на рівні модуля, контролера або окремих маршрутів, що надає гнучкості для контролю доступу в різних частинах вашого додатку. Також є можливість використовувати групові гуарди, які дозволяють застосовувати один або декілька гардів до групи маршрутів.

Використання гуардів дозволяє вам забезпечити безпеку додатку, контролювати доступ до ресурсів та виконувати перевірки перед обробкою запитів. Вони підвищують надійність і захищеність додатку, дозволяючи легко впроваджувати та налаштовувати правила безпеки.

Розглянемо файл roles.guard.ts, який відповідає за перевірку прав доступу користувачу по ролі (див. лістинг 1).

Лістинг *1 Перевірка на права доступу користувача по його ролі*

```
@Injectable()
export class RolesGuard implements CanActivate {
  constructor(
    private jwtService: JwtService,
    private reflector: Reflector,
    private playerRepository: PlayerRepository,
  ) {}

  async canActivate(context: ExecutionContext) {
    try {
```

```
            const roles =
this.reflector.getAllAndOverride<string[]>(ROLES_KEY, [
              context.getHandler(),
            ]);
            if (!roles) return true;
            const req = context.switchToHttp().getRequest();
            const token = req.headers.authorization?.split('
')[1];
            if (!token) throw new HttpException('No
authorization', 401);
            const user =
this.jwtService.verify<Express.User>(token);
            if (!user) throw new HttpException('No
authorization', 401);
            const userDB = await
this.playerRepository.getPlayerByEmail(user.email);
            if (!userDB) throw new HttpException('No
authorization', 401);
            const role = await
this.playerRepository.getRoleById(userDB.roleId);
            if (!role) throw new HttpException('No
authorization', 401);
            req.user = user;
            return roles.includes(role.name);
          } catch (error) {
            if (error instanceof HttpException) throw error;
            throw new HttpException('No access', 403);
          }
        }
      }
```

Даний код представляє собою реалізацію RolesGuard, який є кастомним гвардом відповідальним за авторизацію та автентифікацію користувачів на основі їхніх ролей.

Цей гвард використовується для захисту маршрутів в додатку, щоб дозволяти доступ лише користувачам з певними ролями. Якщо користувач має необхідні ролі, йому надається доступ, в іншому випадку генерується виняток, що вказує на відсутність доступу.

Основний метод canActivate перевіряє ролі, використовуючи рефлектор, який отримується в конструкторі. Якщо ролі не визначені для даного маршруту, дозвіл на доступ надається безпосередньо. Якщо ролі визначені, виконується подальша перевірка.

У першу чергу, метод отримує доступ до запиту (request) через context.switchToHttp().getRequest(). Далі, отримується токен авторизації з заголовків запиту. Якщо токен відсутній, генерується виняток HttpException з кодом 401, що вказує на відсутність авторизації.

Далі, метод перевіряє та верифікує токен за допомогою JwtService, який отримується в конструкторі. Якщо токен не валідний або невірний, також генерується виняток HttpException з кодом 401.

Далі, метод отримує користувача з бази даних за його електронною поштою, використовуючи PlayerRepository. Якщо користувач не знайдений, генерується виняток HttpException з кодом 401.

Після отримання користувача з бази даних, метод отримує роль користувача за його ідентифікатором ролі, також використовуючи PlayerRepository. Якщо роль не знайдена, генерується виняток HttpException з кодом 401.

У кінці методу, об'єкт користувача (user) присвоюється до властивості user запиту (req), щоб його можна було використовувати в подальших обробниках запитів. На завершення, перевіряється наявність ролі користувача серед необхідних ролей для доступу до маршруту. Якщо роль присутня, метод повертає true і надає доступ. В іншому випадку, повертається false і

генерується виняток HttpException з кодом 403, що вказує на відсутність доступу.

Цей RolesGuard може бути використаний в роутерах або контролерах для захисту маршрутів та перевірки ролей користувачів перед доступом до них.

Є ще один гард, який знаходиться в файлі is-loged-in.guard.ts, який відповідає за перевірку того, що користувач увійшов під свій аккаунт або ні (див. лістинг 2).

Лістинг *2 Перевірка пройшов користувач логін або ні*

```
@Injectable()
export class IsLogedInGuard implements CanActivate {
  constructor(private jwtService: JwtService) {}

  async canActivate(context: ExecutionContext) {
    try {
      const req = context.switchToHttp().getRequest();
      const token = req.headers.authorization?.split(' 
')[1];
      if (!token) throw new HttpException('No 
authorization', 401);
      const user = 
this.jwtService.verify<Express.User>(token);
      if (!user) throw new HttpException('No 
authorization', 401);
      req.user = user;
      return true;
    } catch (error) {
      if (error instanceof HttpException) throw error;
      throw new HttpException('No authorization', 401);
    }
  }
```

```
}
```

Даний код представляє собою реалізацію IsLoggedInGuard, який є кастомним гвардом для перевірки авторизації користувача.

Цей гвард використовується для захисту маршрутів в додатку і перевірки того, чи користувач має валідний токен авторизації. Якщо у користувача є валідний токен, йому надається доступ до маршруту, в іншому випадку генерується виняток, що вказує на відсутність авторизації.

Основний метод canActivate перевіряє наявність токену авторизації в заголовках запиту. Він отримує доступ до запиту (request) через context.switchToHttp().getRequest(), а потім отримує токен з заголовків запиту. Якщо токен відсутній, генерується виняток HttpException з кодом 401, що вказує на відсутність авторизації.

Далі, метод використовує JwtService (який отримується в конструкторі) для верифікації токена. Виклик this.jwtService.verify<Express.User>(token) перевіряє валідність токена та повертає розшифровані дані користувача. Якщо токен не валідний або невірний, генерується виняток HttpException з кодом 401.

У кінці методу, об'єкт користувача (user) присвоюється до властивості user запиту (req), щоб його можна було використовувати в подальших обробниках запитів. На завершення, метод повертає true, що вказує на те, що користувач є авторизованим та має доступ до маршруту.

Цей IsLoggedInGuard може бути використаний в роутерах або контролерах для захисту маршрутів та перевірки наявності авторизованого користувача перед доступом до них.

Для того, щоб користувач міг змогу отримати певну інформацію з сервера, користувачу потрібно відправити запит на один з роутів, вони знаходяться в контролерах. Для прикладу контролер «Player» містить роут, який відповідає за логін користувача.

Лістинг 3 *Роут для логіну користувача*

```
@Post('/login')
  async login(@Body() dto: LoginDto): Promise<any> {
    const user = await
this.playerService.getPlayerByEmail(dto.email);
    if (!user) throw new HttpException('Incorrect
data', 400);
    const comparePasswords = await bcryptjs.compare(
      dto.password,
      user.password,
    );
    if (!comparePasswords) throw new
HttpException('Incorrect data', 400);
    const role = await
this.playerRepository.getRoleById(user.roleId);
    const token =
this.playerService.generateToken(user);
    return { token, role };
  }
```

У лістингу 3 зазначений код, який представляє функцію login для обробки POST-запиту на шляху '/login'.

Ця функція виконує процес автентифікації користувача. Вона отримує об'єкт dto, який містить електронну пошту (email) та пароль (password) користувача.

Спочатку функція викликає сервіс getPlayerByEmail для отримання користувача за його електронною поштою. Якщо користувач не знайдений, генерується виняток HttpException з кодом 400 та повідомленням "Incorrect data". Це означає, що надані дані (електронна пошта) не співпадають з жодним користувачем у системі.

Далі, функція використовує bcryptjs.compare для порівняння наданого пароля (dto.password) зі збереженим хешем пароля користувача

(user.password). Якщо пароль не співпадає, генерується виняток HttpException з кодом 400 та повідомленням "Incorrect data". Це означає, що наданий пароль не вірний.

Після успішної перевірки електронної пошти та пароля, функція викликає getRoleById для отримання ролі користувача за його roleId. Далі, викликається generateToken сервісу playerService, який генерує JWT-токен для користувача.

На завершення, функція повертає об'єкт, що містить токен (token) та роль (role) користувача. Цей об'єкт буде відправлений у відповідь на клієнтську сторону.

Ця функція відповідає за обробку авторизаційного запиту, перевірку валідності даних користувача та генерацію та повернення токену доступу та ролі користувача.

Розглянемо ще одну роут, представлений у лістингу 4, який відповідає за коннект до команд.

Лістинг 4 *Роут для коннекта у команду*

```
@UseGuards(IsLogedInGuard)
  @Post('/member/add/:playerId/:teamId')
  async addMemberToTeam(
    @Param('playerId', ValidateMongooseIdPipe)
playerId: ObjectId,
    @Param('teamId', ValidateMongooseIdPipe) teamId:
ObjectId | string,
  ) {
    console.log(teamId);
    if (!playerId || !teamId) {
      throw new HttpException('Incorrect playerId or
teamId', 400);
    }
```

```
        const player = await
this.playerPublicService.getPlayerById(playerId);
        if (!player) {
          throw new HttpException('Player with such id does
not exists', 400);
        }
        const team = await
this.teamService.getTeamById(teamId);
        if (!team) {
          throw new HttpException('Team with such id does
not exists', 400);
        }
        const userAlreadyOnTeam = await
this.teamService.userOnTeam(playerId);
        if (userAlreadyOnTeam.length) {
          throw new HttpException('Player already on a
team', 400);
        }
        const membersOnTeam = await
this.teamService.getTeamInfo(team.id);
        const fullTeam = 5;
        if (membersOnTeam.total === fullTeam) {
          throw new HttpException('This team is already
full', 400);
        }
        await this.teamService.addPlayerToTeam(teamId,
playerId);
        return {
          message: `Player added successfully`,
        };
      }
```

Цей код представляє обробник маршруту для додавання гравця до команди.

Цей обробник має прикріплений @UseGuards(IsLogedInGuard), що означає, що доступ до цього маршруту мають тільки автентифіковані користувачі. Перш за все, обробник отримує playerId та teamId як параметри маршруту. Вони валідуються за допомогою ValidateMongooseIdPipe, який перевіряє, чи вони є дійсними ObjectId або рядком, що представляє ObjectId.

У коді виконується наступне:

Перевіряється наявність playerId та teamId. Якщо хоча б один з них відсутній, викидається помилка HttpException зі статусом 400 і повідомленням "Incorrect playerId or teamId".

Отримується гравець за допомогою виклику функції класса playerPublicService playerPublicService.getPlayerById(playerId). Якщо гравець не знайдений, викидається помилка HttpException зі статусом 400 і повідомленням "Player with such id does not exist".

Отримується команда за допомогою teamService.getTeamById(teamId). Якщо команда не знайдена, викидається помилка HttpException зі статусом 400 і повідомленням "Team with such id does not exist".

Перевіряється, чи користувач вже належить до команди, за допомогою teamService.userOnTeam(playerId). Якщо користувач вже належить до команди, викидається помилка HttpException зі статусом 400 і повідомленням "Player already on a team".

Отримується інформація про членів команди за допомогою teamService.getTeamInfo(team.id).

Перевіряється, чи команда вже заповнена (має максимальну кількість гравців). Якщо кількість гравців на команді дорівнює максимальному значенню (у цьому випадку 5), викидається помилка HttpException зі статусом 400 і повідомленням "This team is already full".

Додається гравець до команди за допомогою teamService.addPlayerToTeam(teamId, playerId).

Повертається об'єкт з повідомленням "Player added successfully".

Цей код передбачає валідацію параметрів маршруту, перевірку наявності гравця та команди, а також перевірку різних умов для додавання гравця до команди.

Також даний застосунок має й фронтенд частину, яка була написана за допомогою використання технології React.

Нижче, на лістингу 5 наведено приклад коду однієї з основних компонент, компоненти логіну.

Лістинг 5 *Компонент логіну*

```
import React, {useState} from 'react';
import style from "./modal-login.module.css"
import {useTypedSelector} from
"../../hooks/useTypedSelector";
import {useDispatch} from "react-redux";
const ModalLogin = () => {
    const {isActiveLog} = useTypedSelector(state =>
state.user);
    const dispatch = useDispatch();
    const apiUrl = "http://localhost:5000"
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");

    const handleSubmit = async () => {
        console.log("Email:", email);
        console.log("Password:", password);
        const requestBody = {
            "email":email,
            "password":password
        }
        try {
```

```
                const response = await
fetch(`${apiUrl}/player/login`, {
                    method: "POST",
                    body: JSON.stringify(requestBody),
                    headers: {
                        'Content-Type': 'application/json'
                    },
                })
                const responseToken = await response.json()
                if (responseToken.token) {
                    dispatch({type: "SET_TOKEN", payload:
responseToken.token});
                    dispatch({type: "SET_ROLE", payload:
responseToken.role.name});
                    setEmail("");
                    setPassword("");
                    dispatch({type: "SET_MODAL_Log",
payload: false});
                } else {
                    alert("Ви ввели неправильний пароль або
логін")
                }
            }catch (e){
                alert('Something went wrong' + e);
            }
        };
        return (
            <div>
                {isActiveLog && (
                    <div className={style.body}>
                        <div className={style.main}>
```

```
                                <span onClick={ () =>
dispatch({type: "SET_MODAL_Log", payload: false})}
className={style.close}>X</span>
                                <div
className={style.inputBlock}>
                                    <p
className={style.label}>Email</p>
                                    <input
                                        className={style.input}
                                        type="email"
                                        value={email}
                                        onChange={(e) =>
setEmail(e.target.value)}
                                    />
                                </div>
                                <div
className={style.inputBlock}>
                                    <p
className={style.label}>Password</p>
                                    <input
                                        className={style.input}
                                        type="password"
                                        value={password}
                                        onChange={(e) =>
setPassword(e.target.value)}
                                    />
                                </div>
                                <button onClick={() =>
handleSubmit()} className={style.button}>Увійти</button>
                            </div>
                        </div>
                    )}
                </div>
```

```
    );
};
export default ModalLogin;
```

Даний код, представляє компонент ModalLogin, який відповідає за відображення модального вікна для входу в систему.

Цей компонент використовується для відображення модального вікна з полями для введення електронної пошти та пароля користувача. Після введення даних користувачем та натискання кнопки "Увійти", дані відправляються на сервер для перевірки та отримання токену доступу та ролі користувача.

У компоненті використовуються хуки useState, useTypedSelector та useDispatch з бібліотеки React Redux. За допомогою хука useTypedSelector отримується значення isActiveLog зі стану користувача, яке вказує, чи показувати модальне вікно входу в систему. За допомогою хука useDispatch отримується функція dispatch, яка використовується для відправки дій до Redux store.

Після отримання значень isActiveLog та dispatch, компонент відображає модальне вікно з полями введення для електронної пошти та пароля. При введенні даних, значення зберігаються в стані компонента за допомогою хуків useState. При натисканні на кнопку "Увійти", виконується функція handleSubmit.

У функції handleSubmit дані електронної пошти та пароля об'єднуються в об'єкт requestBody, який потім відправляється на сервер за допомогою fetch методу. Якщо відповідь від сервера містить токен (responseToken.token), він зберігається в Redux store за допомогою dispatch({type: "SET_TOKEN", payload: responseToken.token}), а також зберігається роль користувача (responseToken.role.name) за допомогою dispatch({type: "SET_ROLE", payload: responseToken.role.name}).

Після успішної обробки відповіді, поля введення електронної пошти та пароля очищаються, а модальне вікно приховується за допомогою dispatch({type: "SET_MODAL_Log", payload: false}). У випадку, якщо відповідь від сервера не містить токену, виводиться повідомлення про неправильні дані.

Компонент ModalLogin експортується як основний експорт модуля і може бути використаний у батьківських компонентах для відображення модального вікна входу в систему.

Розглянемо ще один реакт компонент, який собою являє компонент сторінки команди користувача.

Лістинг 6 *Компонент команди користувача*

```
import React, {useEffect, useState} from 'react';
import style from "../team/team.module.css";
import {useTypedSelector} from
"../../hooks/useTypedSelector";
interface IMembers {
    id: string,
    nickname: string,
    email: string
}
interface ITeam {
    id: string,
    name: string
}
interface ITeamInfo {
    team: ITeam,
    members: IMembers[]
}
const MyTeam = () => {
    const {token} = useTypedSelector(state =>
state.user);
```

```
        const apiUrl = "http://localhost:5000";
        const [teamResponse, setTeamResponse] =
useState<ITeamInfo>({
            team: {
                id: "",
                name: ""
            },
            members: [
                {
                    id: "",
                    nickname: "",
                    email: ""
                }
            ],
        });
        async function getTeamInfo() {
            const request = await fetch(`${apiUrl}/team`, {
                method: "GET",
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                },
            });
            const response = await request.json();
            console.log(response)
            setTeamResponse(response)
        }
        useEffect(() => {
            getTeamInfo();
        }, [])
        const teamItems = teamResponse.members.map((item)
=>
```

```
            <span
className={style.member}>{item.nickname}</span>
        );
        return (
            <div className={style.body}>
                <h3 className={style.title}>Твоя команда
{teamResponse.team.name}</h3>
                <div className={style.members}>
                    <span
className={style.members_title}>Члени цієї команди: </span>
                    {teamItems}
                </div>
            </div>
        );
    };
    export default MyTeam;
```

Код, наведений у лістингу 6 представляє компонент MyTeam, який відповідає за відображення інформації про команду користувача.

Цей компонент використовується для відображення інформації про команду користувача, включаючи назву команди та список її членів. Компонент отримує доступ до токену користувача за допомогою хука useTypedSelector з бібліотеки React Redux.

У компоненті використовується стан teamResponse, який містить об'єкт типу ITeamInfo з інформацією про команду та її членів. Початкове значення teamResponse встановлюється за допомогою useState.

Функція getTeamInfo виконує запит на сервер для отримання інформації про команду за допомогою методу GET. Запит включає заголовки, включаючи токен доступу (Authorization: Bearer ${token}). Отриману відповідь перетворюється у формат JSON і зберігається у стані teamResponse за допомогою setTeamResponse.

Ефект useEffect виконує функцію getTeamInfo при монтажі компонента (пустий масив залежностей), щоб отримати інформацію про команду при завантаженні сторінки.

Компонент повертає JSX-розмітку, в якій відображається назва команди (teamResponse.team.name) та список її членів (teamResponse.members). Кожен елемент списку відображається за допомогою teamItems, який мапить масив teamResponse.members на JSX-елементи з нікнеймами членів команди.

Компонент MyTeam експортується як основний експорт модуля і може бути використаний у батьківських компонентах для відображення інформації про команду користувача.

3.6 Розробка інтерфейсу фронтенд частини застосунку

В сучасному цифровому світі, де веб-застосунки займають центральне місце у нашому щоденному житті, користувачі очікують від них не тільки потужних функціональних можливостей, але й інтуїтивно зрозумілого і привабливого інтерфейсу. Щоб задовольнити ці очікування, розробники все частіше обирають фреймворки, які надають їм потужні інструменти для створення сучасних та зручних інтерфейсів.

Одним з найвдаліших виборів для розробки інтуїтивно зрозумілого і привабливого інтерфейсу є фреймворк React. Він пропонує розробникам широкий набір компонентів і інструментів, які допомагають створювати ефективні та динамічні інтерфейси. React дозволяє розбити складну структуру веб-застосунку на менші, легкі для управління компоненти.

Відповідно до усіх функціональних вимог, які потрібно було реалізувати у веб-застосунку, був розроблений веб-сайт, в якому були реалізовані всі згадані функції. Для того, щоб перейти на головну сторінку сайту знову, користувачу потрібно буде просто натиснути на логотип сайту.

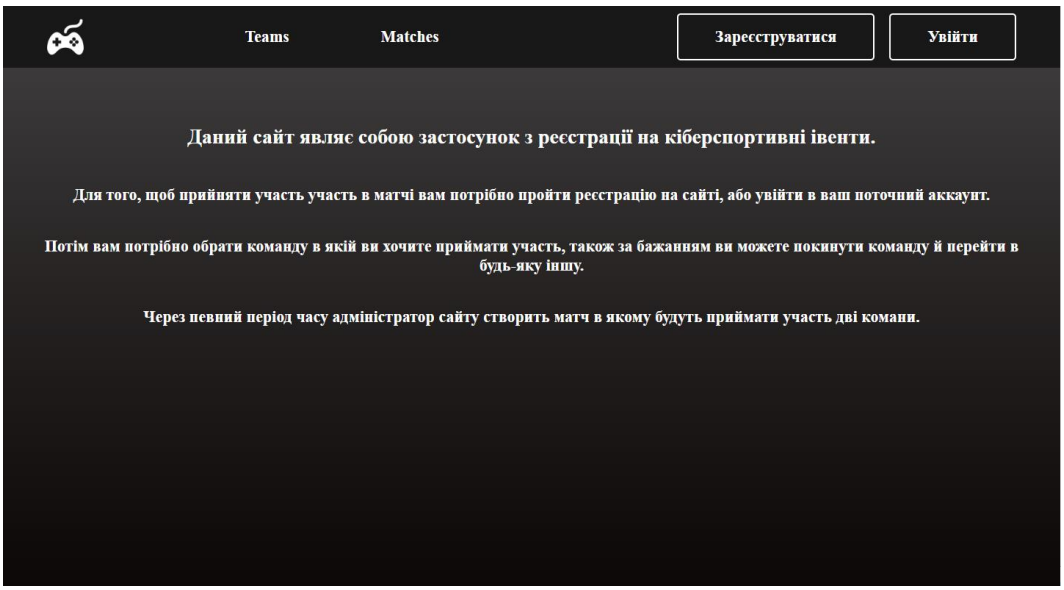

Рис. 16 *Головна сторінка веб-застосунку*

Коли користувач заходить на сайт, він потрапить на головну сторінку, яка буде мати вигляд, який зображено на рисунку 16.

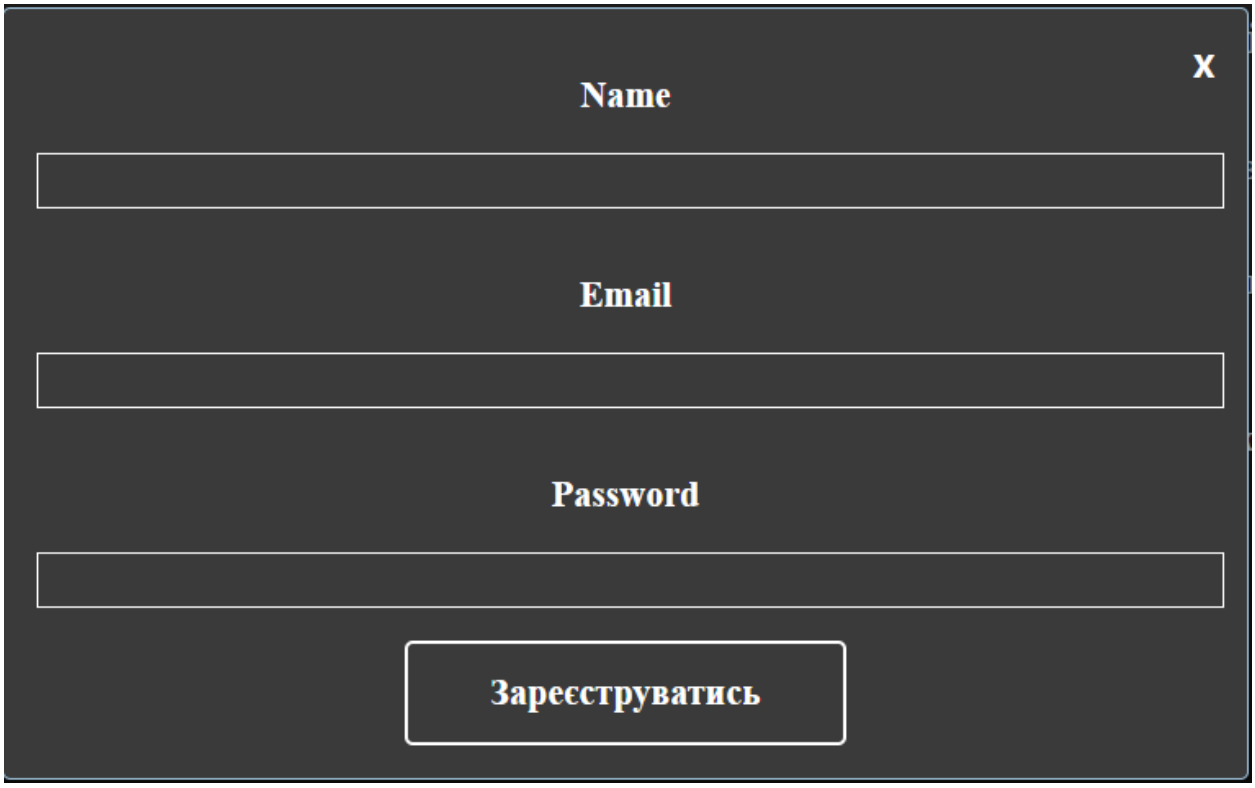

Рис. 17 *Вікно реєстрації нового користувача*

У правому верхньому куті можна побачити дві кнопки, зареєструватися на сайті або увійти до особистого облікового запису користувача. При спробі зареєструватися у даному застосунку, користувачу буде відображено модальне вікно, яке зображено на рисунку 17.

Для того, щоб користувач завершив реєстрацію на даному веб-сайті, йому потрібно ввести унікальне ім’я користувача а також вказати унікальну пошту, потім ввести надійний, на розсуд користувача, пароль. Якщо користувач буде унікальним — він пройде реєстрацію на сайті й зможе користуватися усіма перевагами застосунку.

Якщо користувач вже має аккаунт у застосунку, йому просто потрібно натиснути на копку «Увійти» в правому верхньому куті веб-сайту. Потім буде відображено модальне вікно, де користувачу потрібно буде ввести свій email а також пароль, який він використовував при реєстрації на сайті (див. Рис. 18).

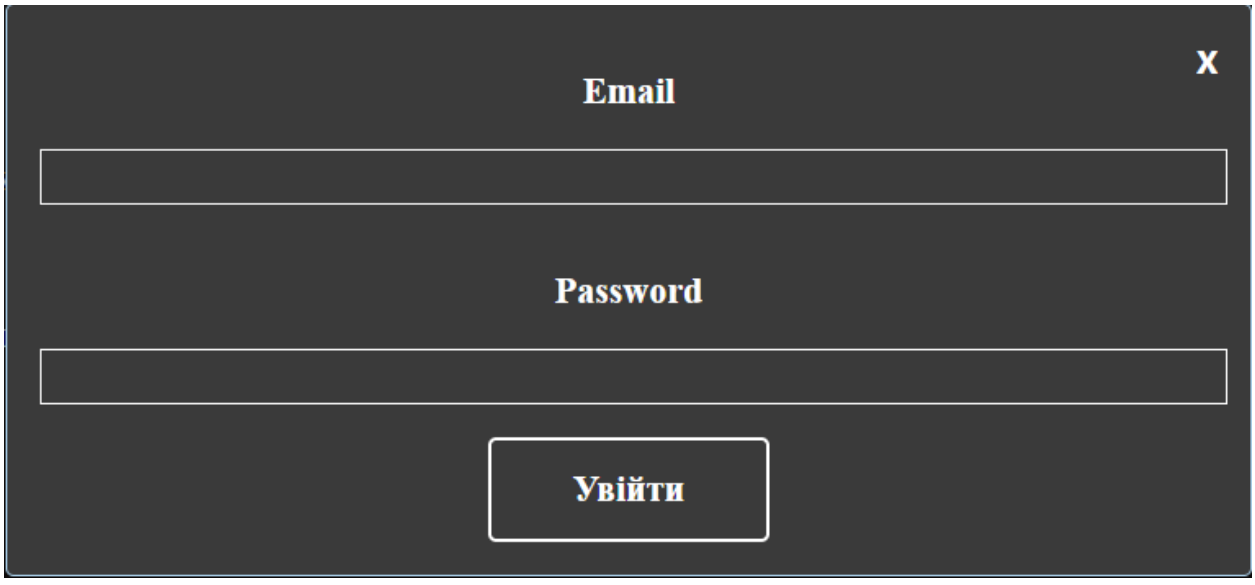

Рис. *18 Вікно логіну зареєстрованого користувача*

Коли користувач увійде до веб-сайту, у верхній частині застосунку, буде відображено додаткову кнопку «Вийти» в правому верхньому куті, дана кнопка буде заміщувати кнопки «Увійти», «Зареєструватися». Також в хедері веб-сайту буде зображено додаткові поля, які можна побачити на рисунку 19.

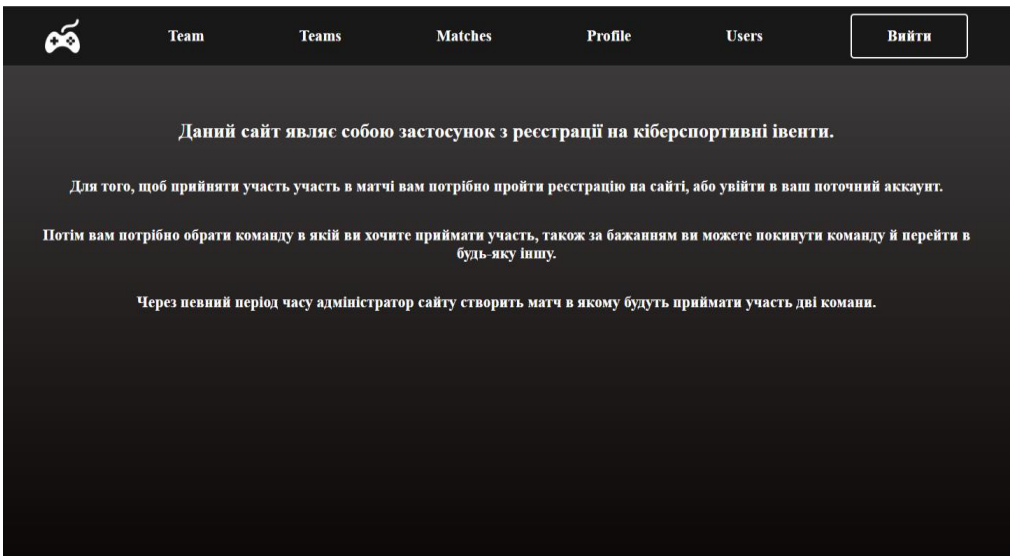

Рис. *19 Головна сторінка сайту після того як користувач пройшов реєстрацію*

Дані кнопки також мають свій функціонал. Наприклад кнопка «Profile» переправить на сторінку з повною інформацією про користувача (див. Рис. 20), а саме його ім’я, пошту та в якій команді перебуває даний користувач. Також є можливість змінити ім’я користувача на нове, у відповідне поле просто потрібно ввести нове ім’я користувача й натиснути кнопку змінити ім’я, якщо воно буде унікальним — веб-застосунок змінить його і потім автоматично відобразить нове ім’я.

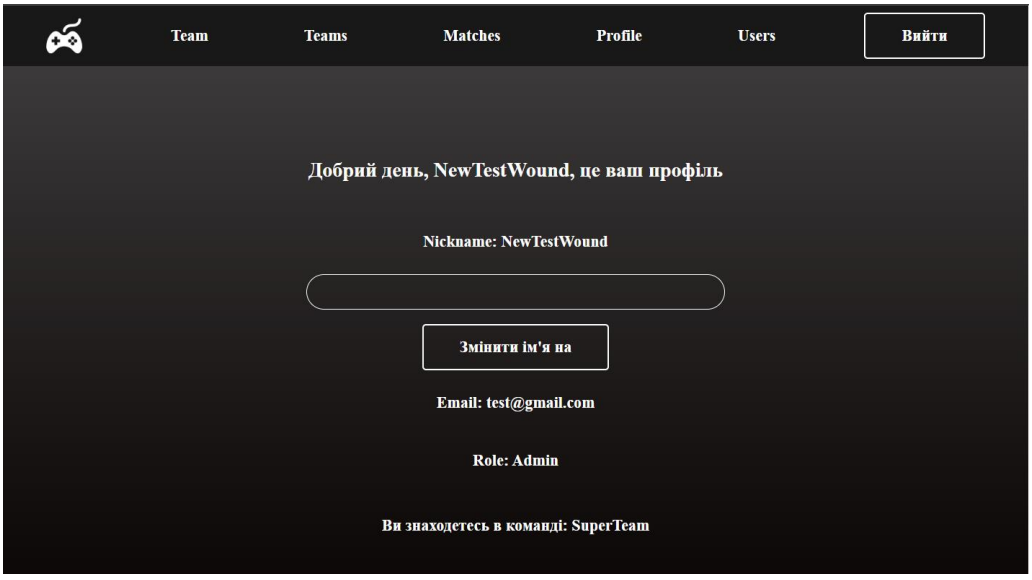

Рис. *20 Сторінка користувача*

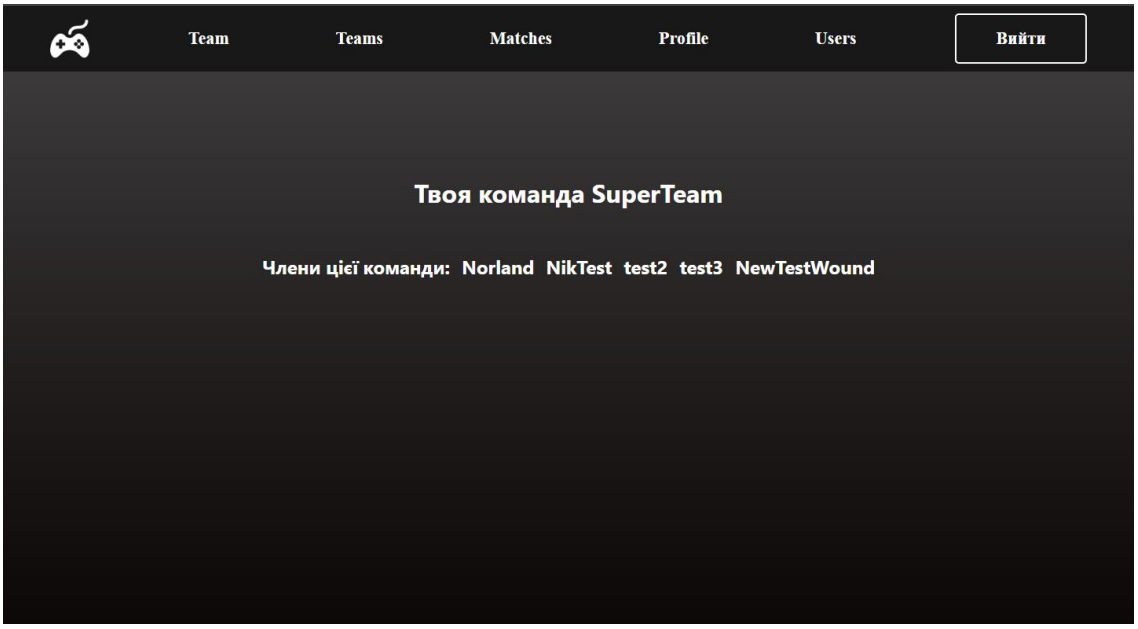

Рис. *21 Сторінка команди користувача*

Кнопка «Team» відображає інформацію про команду (див. Рис. 21), в якій знаходиться користувач. Буде відображено інформацію про назву команди а також список членів даної команди.

Кнопка «Teams» відповідає за те, що відображає список усіх команд, з можливістю пагінації (див. Рис. 22). Спочатку будуть відображенні дві команди , а потім можна натиснути на кнопку вперед, сайт відобразить наступні дві команди, якщо користувач захоче отримати дві попередні команди, потрібно натиснути на кнопку назад. Якщо при спробі користувача отримати наступні або попередні команди, користувач дійде до ліміту команд, веб-сайт заблокує можливість користувача отримати наступні або попередні команди.

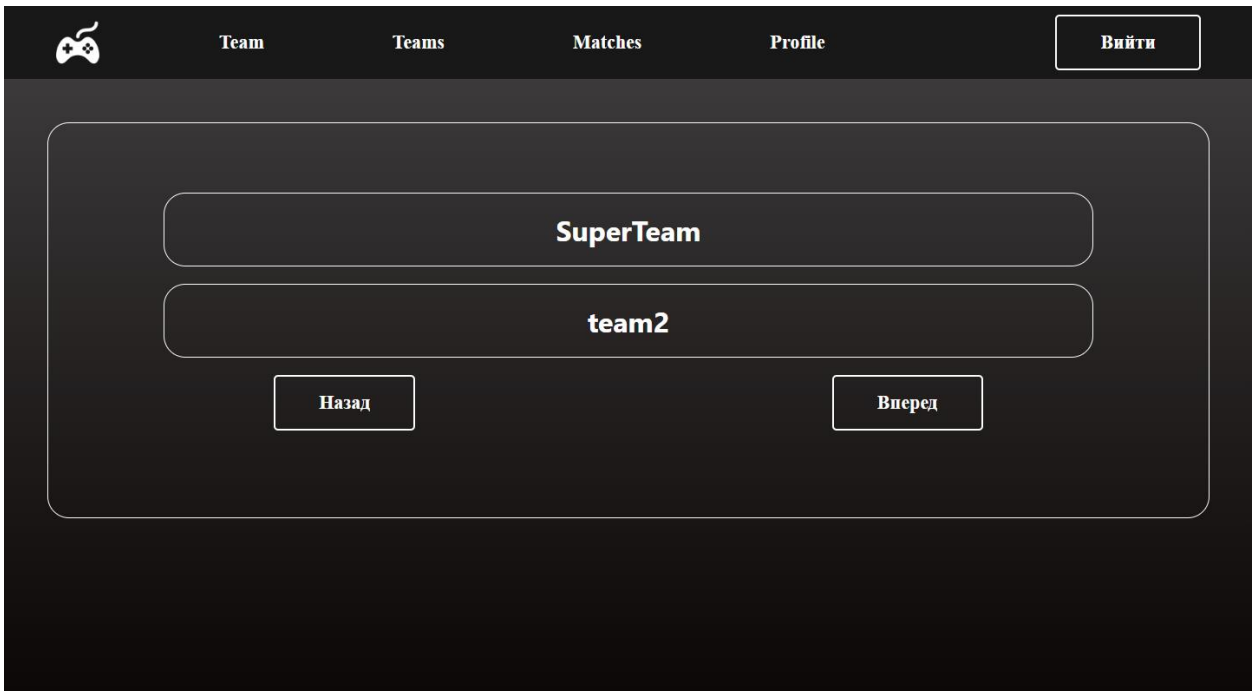

Рис. *22 Сторінка усіх команд сервісу*

Кнопка «Matches» відповідає за відображення матчів, які були зареєстровані на даному веб-сайті, приклад зображений на рисунку 23.

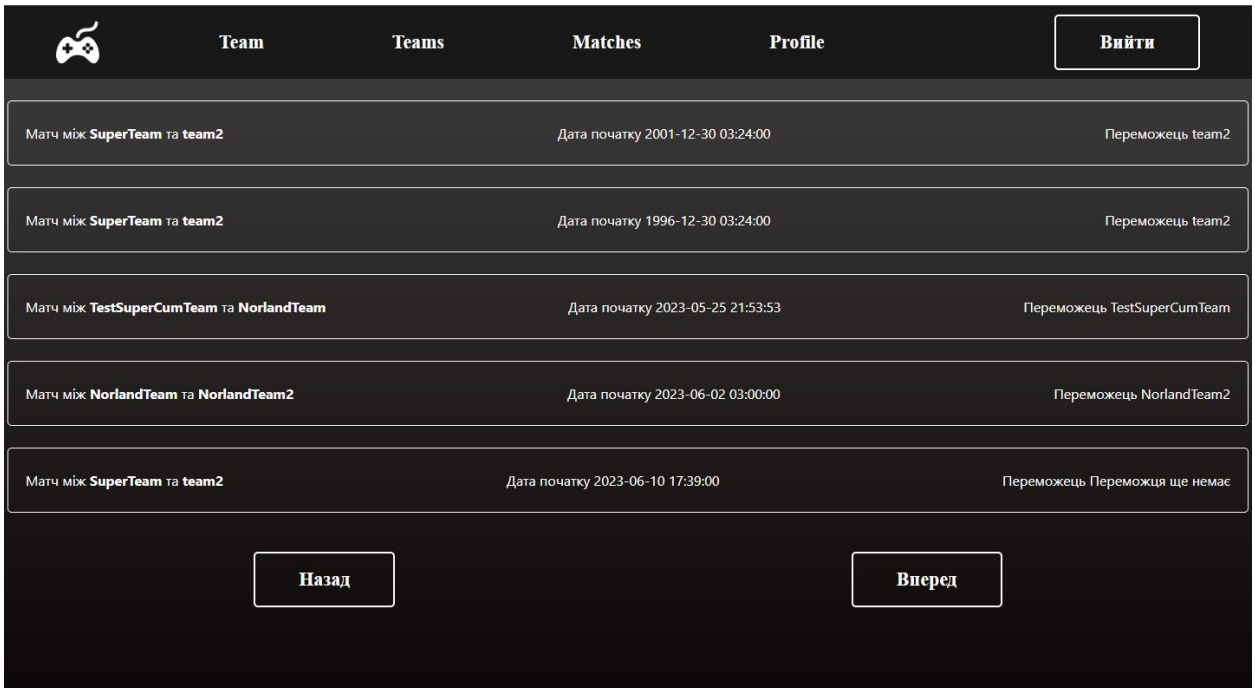

Рис. *23 Сторінка усіх матчів сервісу*

Кнопка «Users» доступна лише для тих користувачів, які мають роль Admin (див. Рис. 24). Ця сторінка відображає всіх користувачів застосунку, з можливістю їх видалення.

Рис. *24 Список користувачів застосунку*

ВИСНОВКИ

1. У ході реалізації кваліфікаційної роботи було опрацьовано літературні джерела за темою веб-застосунців та проаналізовано продукти аналоги, яким надають перевагу при використанні звичайні користувачі.
2. Аналіз доступних наявних інформаційних матеріалів та результатів наукових досліджень, показав, що найбільша увага розробників та дослідників приділяється IT технологіям, зорієнтованим на швидку обробку інформації, її швидке відправлення клієнту та швидку комунікацію з базою даних. Саме тому метою розробки став веб-застосунок для реєстрації на кіберспортивні івенти для задоволення потреб користувачів.
3. Після аналізу поточного стану ситуації, було складено технічне завдання та проведено планування робіт. Перед початком реалізації було проведено проектування архітектури системи та ідентифіковано різні варіанти використання. Розроблена система повністю відповідає всім поставленим функціональним вимогам.
4. Був спроектований та розроблений серверний застосунок, який покривав всі вимоги користувачів. Крім цього було розроблено користувацький інтерфейс.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Імас Є., Петровська Т., Ганага О. Кіберспорт в Україні як сучасний культурний феномен. Теорія і методика фізичного виховання і спорту. 2021. № 1. С. 75—81.

2. Кудряшова Т. І. Місце та ознаки кіберспорту як спортивної дисципліни / Т. І. Кудряшова, Т. І. Лисенко, О. О. Морозова. 2021. С. 511—524.

3. HLTV: веб-сайт. URL: https://www.hltv.org/ (дата звернення: 25.05.2023)

4. FaceIt: веб-сайт. URL: https://www.faceit.com/en (дата звернення: 25.05.2023)

5. ESL Play: веб-сайт. URL: https://play.eslgaming.com/ (дата звернення: 25.05.2023)

6. Adam Freeman Pro ASP.NET MVC 5 (Expert's Voice in ASP.Net) 5-те вид. 2013. 832 с.

7. Shelley Powers Learning Node: Moving to the Server-Side 2-ге вид. 2016. 288 с.

8. Julia Elman Lightweight Django: Using REST, WebSockets, and Backbone 1-ше вид. 2014. 246 с.

9. David Sklar Learning PHP: A Gentle Introduction to the Web's Most Popular Language 1-ше вид. 2016. 416 с.

10. Greg Lim, Daniel Correa Ethan Brown Web Development with Node and Express: Leveraging the JavaScript Stack 2-ге вид. 2018. 326 с.

11. Practical Nest.js: Develop clean MVC web applications 2022. 256 с.

12. Stefanov S. React: up & running: building web applications. 2-ге вид. Sebastopol : O'Reilly Media, 2021. 230 с.

13. Josh Goldberg Learning TypeScript. Enhance Your Web Development Skills Using Type-Safe JavaScript 2022. 318 с.

14.Hanchett E. Vue.js in action first edition. NY : Manning, 2018. 375 с.

15.Fain Y., Moiseev A. Angular Development with TypeScript. 2-ге вид. NY : Manning, 2018. 560 с.

16. Raymond CamdenThe Jamstack Book: Beyond static sites with JavaScript, APIs, and markup, 2022 280 с.

17. Артеменко Артур студент 4 курсу ІННІ ім. Ю.М. Потебні ЗНУ. Наук. кер.: к.т.н., доц. Міхайлуца О.М. «Розробка веб-додатку реєстрації на кіберспортивні змагання з використанням фреймворку nest.js». Збірник наукових праць студентів, аспірантів і молодих вчених «Молода наука-2023» : у 5 т. Запорізький національний університет. Запоріжжя: ЗНУ, 2023. Т.5. С. 73-74.

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Артеменко Артур Ігорович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz19bd-01@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Розробка веб-додатку реєстрації на події з використанням фреймворку NestJS**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 20.06.2023 Підпис _____ Артеменко Артур Ігорович

(студент)

Дата 20.06.2023 Підпис _____ Міхайлуца Олена Миколаївна

(науковий керівник)