

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Інформаційна система аналітики спортивних даних»

Виконав студент групи ПЗ-22-3
СВИСТАНЮК Назарій Олександрович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СВИСТАНЮК Назарій Олександрович**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: **«Інформаційна система аналітики спортивних даних»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., № 92/с**

Вихідні дані для роботи: об'єктом дослідження виступає процес інформаційної взаємодії користувачів із масивами спортивної статистики, що охоплює методи агрегації даних, їх динамічну візуалізацію та персоналізацію контенту в межах вебплатформи; предметом дослідження є архітектурні підходи до проектування мультиспортивних систем, алгоритми обробки показників матчів у потоковому режимі та програмний стек, який забезпечує безперебійне функціонування аналітичного сервісу.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	01.03.2026р.	01.03.2026р
2	Власенко О.В.	01.03.2026р	01.03.2026р
3	Власенко О.В.	01.03.2026р	01.03.2026р

РЕФЕРАТ

Пояснювальна записка до бакалаврської кваліфікаційної роботи на тему «Інформаційна система аналітики спортивних даних» містить 98 сторінок основного тексту. Ілюстративний матеріал представлений 38 рисунками та 24 таблицями. Бібліографічний перелік складається з 42 найменувань, а додатки займають 11 сторінок. Загальний обсяг документа становить 109 сторінок.

Мета дослідження полягає у проектуванні та програмній реалізації мультиспортивної аналітичної платформи. Основне завдання системи – це автоматизація збору й обробки статистичної інформації за допомогою інтерактивних дашбордів, моніторинг результатів спортивних подій та гнучке налаштування персоналізованого контенту в режимі реального часу.

Архітектурні рішення. Клієнтська частина платформи побудована з використанням бібліотеки React, що гарантує високу продуктивність інтерфейсу. Серверна частина реалізована на базі Node.js, забезпечуючи надійну обробку, структурування та зберігання великих масивів спортивних даних. Застосування компонентної архітектури та протоколу WebSockets для потокової передачі даних гарантує миттєве оновлення турнірних таблиць, оптимізацію використання ресурсів браузера та зручність користування, що є вкрай важливим для безперервного аналізу матчів.

Практичне значення. Створено повнофункціональний прототип вебсистеми, орієнтованої як на спортивних аналітиків, так і на широке коло вболівальників. Завдяки використанню сучасних технологій веброзробки досягнуто високої швидкодії платформи, миттєвого оновлення Live-статистики.

КЛЮЧОВІ СЛОВА: АНАЛІТИКА, ВЕБТЕХНОЛОГІЯ, ІНФОРМАЦІЙНА СИСТЕМА, ПРОГНОЗУВАННЯ, СПОРТИВНІ ДАНІ.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Інформаційна система аналітики спортивних даних	Літ.	Арк.	Аркушів
Розроб.		Сви́станюк Н.О.						
Керівник		Власенко О.В.					4	109
Рецензент		Вакалюк Т.А.				Житомирська політехніка, група ІПЗ-22-3		
Зав. каф		Вакалюк Т.А.						

ABSTRACT

The explanatory note for the bachelor's qualification work on the topic "Information System for Sports Data Analytics" contains 98 pages of main text. The illustrative material is represented by 38 figures and 24 tables. The bibliography consists of 42 items, and the appendices occupy 11 pages. The total volume of the document is 109 pages.

The purpose of the study is the design and software implementation of a multisport analytical platform. The main task of the system is the automation of the collection and processing of statistical information using interactive dashboards, monitoring the results of sports events, and flexible configuration of personalized content in real-time.

Architectural solutions. The client side of the platform is built using the React library, which guarantees high interface performance. The server side is implemented based on Node.js, ensuring reliable processing, structuring, and storage of large volumes of sports data. The application of a component architecture and the WebSockets protocol for data streaming guarantees instant updates of tournament tables, optimization of browser resource utilization, and usability, which is crucial for the continuous analysis of matches.

Practical significance. A fully functional prototype of a web system has been created, targeted at both sports analysts and a broad audience of fans. Thanks to the use of modern web development technologies, high platform performance and instant updates of Live statistics have been achieved.

KEYWORDS: ANALYTICS, FORECASTING, INFORMATION SYSTEM, SPORTS DATA, WEB TECHNOLOGY.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	4
ABSTRACT	5
ЗМІСТ	6
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП	8
РОЗДІЛ 1. ПРЕДМЕТНА ОБЛАСТЬ ТА ПОСТАНОВКА ЗАДАЧІ.....	10
1.1 Постановка задачі та огляд існуючих систем-аналогів	10
1.3 Вибір і обґрунтування архітектурного рішення	22
1.4 Вибір стеку технологій та апаратного забезпечення	25
Висновки до першого розділу	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ	30
2.1 Моделювання варіантів використання та поведінки системи	30
2.2 Проєктування логічної структури бази даних	35
2.3 Проєктування компонентної структури та взаємодії модулів	44
2.4 Розробка основних алгоритмів роботи платформи.....	64
Висновки до другого розділу	72
Розділ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЄКТУ	74
3.1 Порядок встановлення та налаштування параметрів системи	74
3.2 Структура інтерфейсу системи аналітики спортивних даних	78
3.3 Тестування роботи програмного продукту	85
Висновки до третього розділу	90
ВИСНОВКИ.....	92
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	94
ДОДАТКИ	98

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

API (Application Programming Interface) – прикладний програмний інтерфейс

CRUD (Create, Read, Update, Delete) – базові функції управління даними (створення, читання, оновлення, видалення)

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

DTO (Data Transfer Object) – об'єкт передачі даних між шарами архітектури

JSON (JavaScript Object Notation) – текстовий формат обміну даними

JWT (JSON Web Token) – відкритий стандарт для безпечної передачі інформації у вигляді JSON-об'єкта

ORM (Object-Relational Mapping) – технологія програмування, яка зв'язує бази даних з об'єктно-орієнтованим кодом

RBAC (Role-Based Access Control) – управління доступом на основі ролей

REST (Representational State Transfer) – архітектурний стиль взаємодії компонентів розподіленого застосунку в мережі

SPA (Single Page Application) – односторінковий вебзастосунок

SQL (Structured Query Language) – мова структурованих запитів до реляційних баз даних

SSR (Server-Side Rendering) – рендеринг сторінок на боці сервера

UI (User Interface) – інтерфейс користувача

UX (User Experience) – користувацький досвід

WS (WebSocket) – протокол повнодуплексного зв'язку клієнта та сервера у реальному часі

ВСТУП

Актуальність теми зумовлена стабільно високий попит на гнучкі інструменти для агрегації спортивної статистики. Сучасним користувачам необхідний зручний доступ до даних та можливість безперервного відстеження результатів матчів у режимі реального часу. Існуючі на ринку продукти часто не задовольняють ці потреби через перевантаженість інтерфейсів або відсутність ефективних алгоритмів персоналізації стрічки новин. Такий надлишок неструктурованої інформації суттєво погіршує користувацький досвід. Розробка нової інформаційної системи дозволяє вирішити ці проблеми шляхом централізації обробки даних, забезпечення інтерактивної взаємодії через динамічні дашборди та миттєвого оновлення турнірних таблиць. З погляду сучасної інженерії програмного забезпечення, проєктування масштабованої платформи з надійним програмним інтерфейсом та оптимізованим клієнтським боком є вкрай актуальним технічним завданням.

Метою роботи є проєктування та програмна реалізація мультиспортивної аналітичної вебплатформи, призначеної для автоматизації збору статистики, візуалізації даних через інтерактивні дашборди, порівняння командних показників та гнучкої персоналізації контенту. Для досягнення поставленої мети необхідно виконати низку завдань. Насамперед потрібно проаналізувати предметну область та існуючі вебрішення у сфері спортивної аналітики для формування функціональних вимог. Далі слід обґрунтувати вибір сучасного технологічного стеку та інструментів розробки. Після цього вимагається спроектувати серверну архітектуру та структуру бази даних. На завершальному етапі необхідно розробити зручний клієнтський інтерфейс із застосуванням алгоритмів потокової передачі даних для живої візуалізації результатів.

Об'єктом дослідження є процес програмної реалізації комплексної інформаційної системи для інтерактивної візуалізації спортивних даних та управління аналітичним контентом. Предметом дослідження є архітектурні моделі, методи та програмні засоби побудови масштабованої мультиспортивної платформи. Дослідження також охоплює питання оптимізації обробки великих

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

масивів інформації та забезпечення коректної синхронізації ігрових показників у браузері користувача за допомогою протоколів зв'язку в реальному часі.

Методи дослідження. У роботі застосовувався комплексний підхід, що базується на методах системного та порівняльного аналізу для дослідження ринку рішень у сфері спортивної аналітики та вибору оптимального програмного стеку. Для побудови архітектури та розробки функціональних модулів застосовувалися принципи об'єктно-орієнтованого проєктування, а для перевірки надійності алгоритмів потокового оновлення даних та зручності користувацького інтерфейсу використовувалися методи навантажувального та емпіричного тестування.

Практичне значення одержаних результатів. Створений програмний продукт має безпосереднє прикладне значення і може слугувати готовим фундаментом для розгортання повноцінного спортивного порталу. Впровадження системи забезпечить експертам та вболівальникам швидкий доступ до аналітики. Гнучка архітектура проєкту дозволяє легко масштабувати платформу, додавати нові види спорту та інтегрувати сторонні джерела даних без необхідності глибокого переписування програмного коду.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ПРЕДМЕТНА ОБЛАСТЬ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Постановка задачі та огляд існуючих систем-аналогів

Предметною областю дипломного проєкту є сфера спортивних інформаційних технологій, зокрема сегмент вебдодатків для аналізу та візуалізації спортивних даних. В епоху цифрових трансформацій миттєвий доступ до точної статистики є фундаментальною потребою аналітиків та вболівальників. Водночас ефективність аналізу прямо визначається якістю застосовуваного інструментарію. Традиційний підхід до збору результатів на різних порталах характеризується фрагментарністю та значними втратами часу. Головними перепонами виступають брак персоналізації та складнощі у візуалізації історичних відомостей. Окрема проблема полягає у затримках оновлення інформації. Без використання алгоритмів потокової передачі даних більшість показників втрачає актуальність відразу після завершення ігрового епізоду. Розуміння цих обмежень формує запит на створення інтелектуальних платформ. У таких системах рутинний збір статистики виконується автоматично, а користувач отримує зручне інтерактивне середовище для роботи.

У сучасному світі вебзастосунки потребують масштабованої архітектури та високої продуктивності. Ринок пропонує широкий спектр інструментів для реалізації найскладніших технічних вимог. Клієнтський бік платформи забезпечує відображення інтерфейсу та взаємодію з аудиторією. Сьогодні золотим стандартом розробки динамічних вебінтерфейсів вважається використання бібліотеки React разом з екосистемою Next.js [1]. Зазначений технологічний стек дає змогу створювати швидкі системи з оптимізованим рендерингом. Побудова інтерфейсу спирається на декларативний підхід та компонентну архітектуру. Це ідеально відповідає концепції створення аналітичних дашбордів. Графіки, турнірні таблиці та модулі поточних рахунків оновлюються динамічно відповідно до перебігу спортивної події.

Серверна частина сучасних програмних рішень відповідає за обробку запитів, реалізацію бізнес-логіки та інтеграцію зовнішніх сервісів. Застосування середовища Node.js [2] гарантує ефективну роботу бекенду та швидку обробку

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

подій за допомогою сокетних з'єднань. Надійність даних і суворі типізація на всіх етапах розробки забезпечуються мовою програмування TypeScript [3]. Збереження історичної статистики матчів, профілів користувачів та параметрів персоналізації найкраще реалізувати за допомогою реляційних баз даних на кшталт PostgreSQL [4]. Також передбачається залучення сучасних локальних браузерних сховищ для ефективного кешування та миттєвого відображення інформації.

Ключовим інженерним рішенням є інтеграція системи глобального управління станом. Вона дозволяє інкапсулювати логіку взаємодії із сервером. Завдяки механізмам реактивної прив'язки даних будь-які зміни рахунку чи статистики, отримані через вебсокети, миттєво синхронізуються з візуальними елементами дашборда. Це повністю усуває необхідність ручного перезавантаження сторінки. Здійснений аналіз технологічного забезпечення підтвердив доцільність використання сучасної екосистеми JavaScript та оптимізованих серверних інструментів. Такий підхід закладає надійний фундамент для створення швидкого, адаптивного та максимально персоналізованого сервісу спортивної аналітики.

Сегмент програмних рішень для спортивної статистики показує стійку тенденцію до зростання. Вболівальники постійно шукають зручні засоби для оперативного доступу до живих результатів. Одним із найяскравіших представників цього напрямку є популярна платформа Flashscore. Робота даного сервісу ґрунтується на швидкому виведенні поточних рахунків та базової статистики матчів.

Головною перевагою Flashscore виступає величезне охоплення спортивних ліг світу та мінімальні затримки в оновленні інформації. Сервіс надає доступ до текстових трансляцій, стартових складів команд та історії попередніх протистоянь. Такий набір функцій здебільшого задовольняє базові запити середньостатистичного вболівальника. Попри це, поглиблений аналіз платформи вказує на низку помітних недоліків з погляду сучасних стандартів користувацького досвіду та дизайну інтерфейсів.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

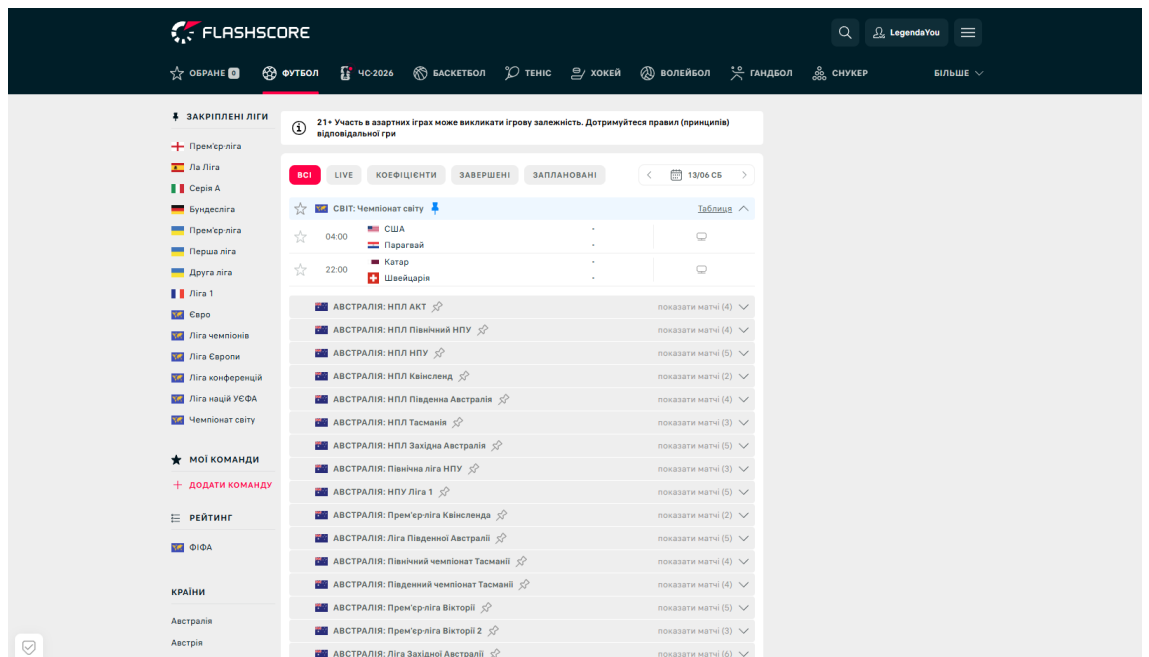


Рис. 1.1 – Інтерфейс головної сторінки сервісу Flashscore

На головній сторінці платформи користувача одразу зустрічає структурований список матчів поточного дня, згрупований за лігами. Функціонально верхня навігаційна панель забезпечує швидкий доступ до вибору видів спорту та календаря подій. Алгоритм подачі контенту реалізований через простий хронологічний список, що дозволяє користувачу миттєво знайти потрібний поєдинок, мінімізуючи когнітивне навантаження та значно знижуючи поріг входу для нових відвідувачів.

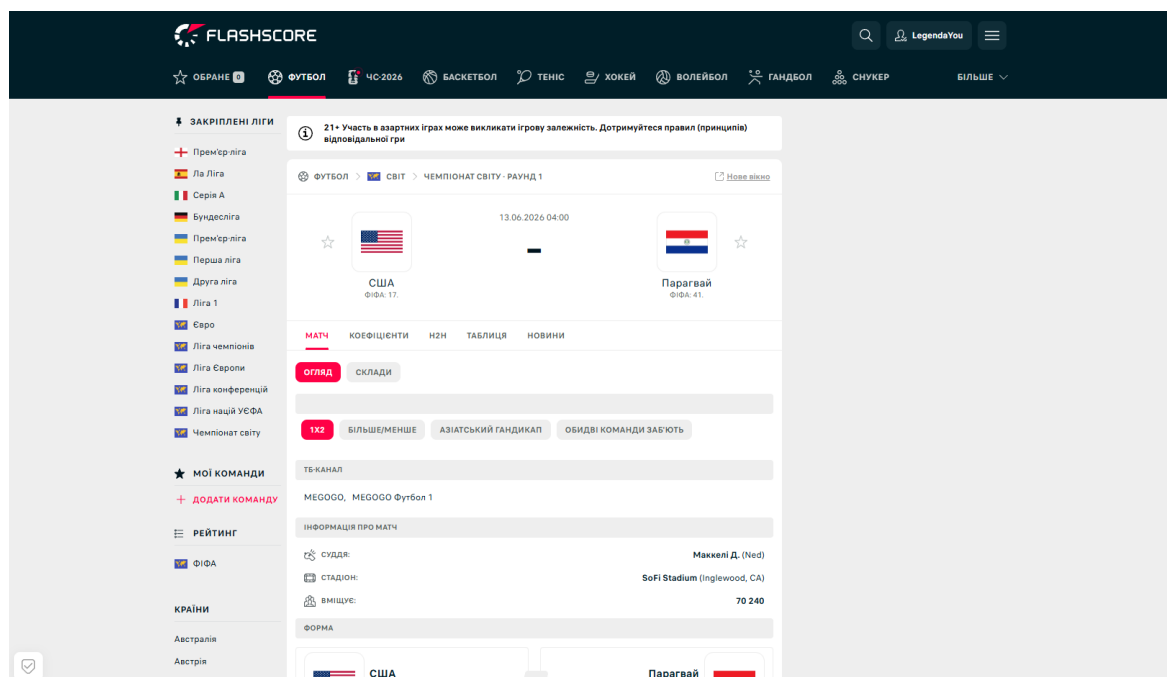


Рис. 1.2 – Сторінка статистики конкретного матчу в системі Flashscore

На сторінці конкретного матчу містить базовий інструментарій для взаємодії зі статистикою. Структура інтерфейсу базується на системі вкладок, які відображають текстовий хід подій, поточні склади та загальні кількісні показники. Ключовим елементом системи виступає модуль миттєвого оновлення результату. Цей компонент динамічно фіксує ігрові події та візуально виділяє зміни безпосередньо в момент їх настання.

Аналіз загального функціонала Flashscore підтверджує здатність системи швидко обробляти інформацію та гарантувати стабільну роботу інтерфейсу. Проте для досвідчених експертів суттєвим недоліком залишається занадто спрощений формат подачі матеріалу. Платформа позбавлена інструментів для детального налаштування дашбордів, відображення просунутих метрик рівня очікуваних голів або створення комплексних порівняльних таблиць. Де-факто сервіс працює переважно як стрічка новин та поточних рахунків. Натомість професійні аналітичні рішення потребують значно вищого рівня гнучкості, розширених графічних інструментів та можливості персоналізації дослідницького простору.

На протигагу масовим продуктам платформа WhoScored позиціонується як вузькоспрямований аналітичний ресурс для агрегації розширеної статистики. На відміну від традиційних застосунків з простою структурою, цей сервіс забезпечує глибокий рівень деталізації даних. Вебресурс функціонує як потужний інструмент для вивчення теплових карт активності гравців, формування об'єктивних рейтингів спортсменів та проведення комплексного тактичного розбору матчів.

Незважаючи на потужний аналітичний інструментарій, подібні вузькоспеціалізовані ресурси також мають очевидні недоліки. Перевантаженість інтерфейсу масивами складних метрик та графічними елементами значно ускладнює навігацію та підвищує поріг входження для звичайних уболівальників.

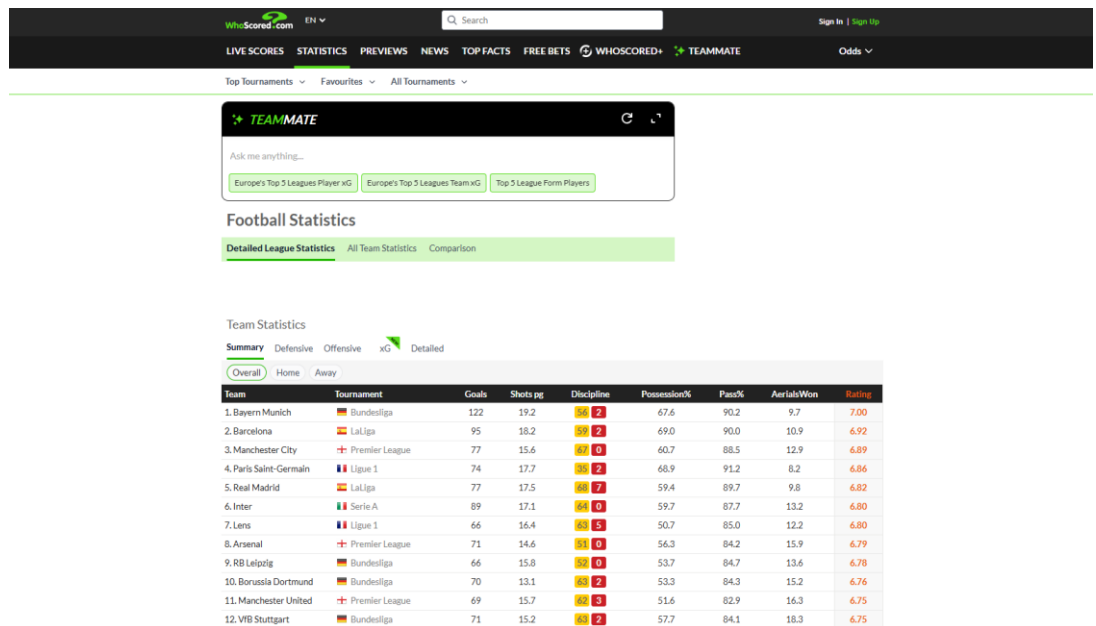


Рис. 1.3 – Головний екран аналітичної платформи WhoScored

На головному екрані платформи користувачу надається доступ до детальних рейтингів ліг та найкращих гравців тижня. Інтерфейс є максимально наповненим і фокусується на складних статистичних показниках: навпроти кожної команди чи гравця відображаються комплексні числові індикатори, які вказують на їх ефективність за допомогою спеціально розроблених алгоритмів рейтингування. Основна логіка системи побудована на математичних моделях, які аналізують кожну дію спортсмена на полі та автоматично розраховують його загальну оцінку матчу. Це забезпечує високу точність об'єктивного аналізу ігрових подій.

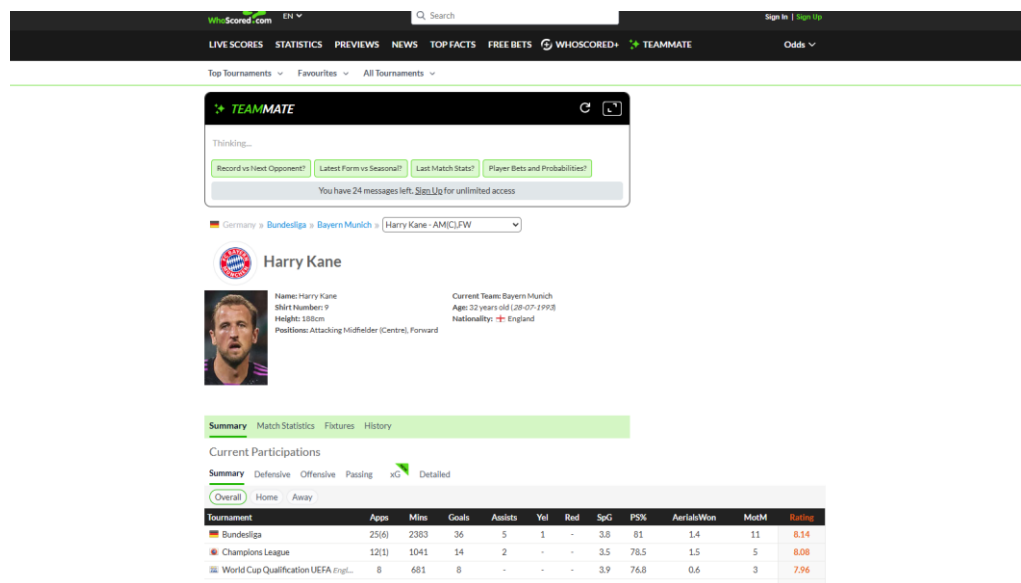


Рис. 1.4 – Інтерфейс порівняння показників гравців в WhoScored

Основним елементом системи виступає інструментарій для деталізованого аналізу. Профіль кожного спортсмена містить потужний набір функцій для глибокого дослідження показників. Користувачі мають змогу вивчати не лише стандартні метрики, а й аналізувати зони ігрової активності через теплові карти, переглядати історію ушкоджень та оцінювати динаміку форми за допомогою складних сезонних графіків. З технічного погляду ресурс характеризується розвиненою системою агрегації та обчислення великих обсягів даних. Однак потужна алгоритмічна база не компенсує вагомих проблем у сфері користувацького досвіду. Застарілий візуальний стиль та надмірна кількість громіздких таблиць суттєво ускладнюють навігацію, роблячи платформу вкрай некомфортною для швидкого повсякденного моніторингу подій.

Альтернативним і вагомим гравцем на ринку аналітичних застосунків виступає сервіс SofaScore. Цей програмний продукт пропонує комплексне бачення, гармонійно поєднуючи оперативне оновлення рахунків із вбудованими механізмами соціальної взаємодії. На противагу виключно інформаційній парадигмі класичних трекерів та вузькопрофільному інструментарію спеціалізованих платформ, SofaScore спирається на сучасні наукові стандарти візуалізації інформації [5].

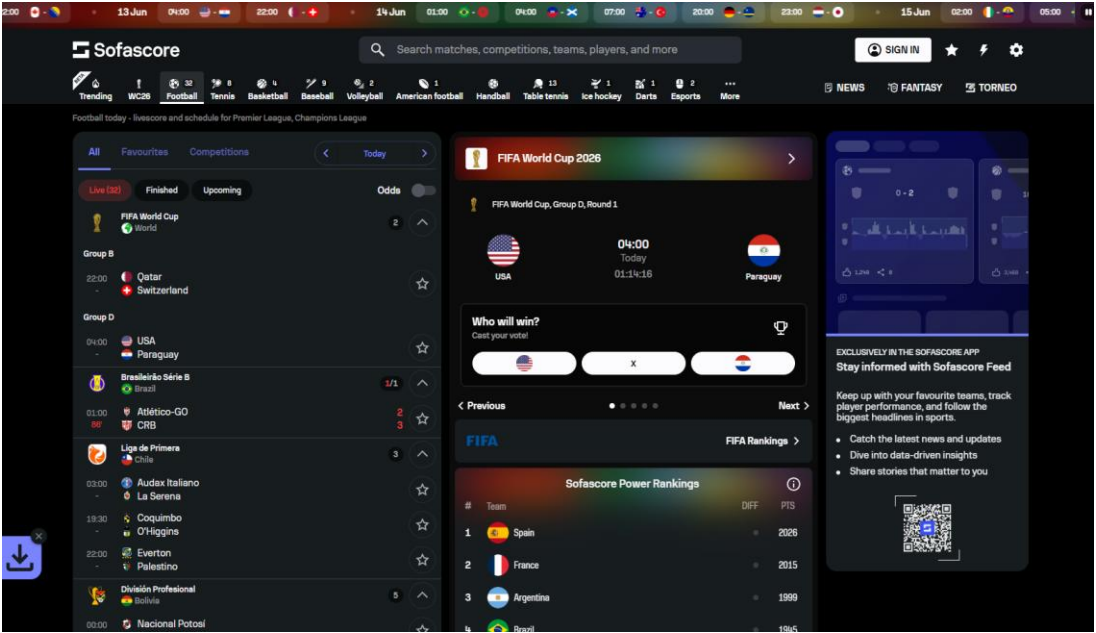


Рис. 1.5 – Головний екран та розклад матчів в додатку SofaScore

На головному екрані застосунку реалізовано логічну архітектуру подачі розкладу у вигляді календаря. Інтерфейс чітко сегментований на види спорту та конкретні ліги. Кожен такий матч інкапсулює у собі комплексні дані склади команд, турнірне становище та графік зміни ігрової переваги з використанням детальної статистики від офіційних провайдерів. З точки зору дизайну інтерфейсу додаток підтримує баланс між інформативністю та візуальним навантаженням: користувач має чітке розуміння поточного стану гри. Окрім цього, система використовує багаторівневу гейміфікацію, де прогнози щодо переможця та індикатори зміни імпульсу матчу інтегровані в інтерфейс без надмірної перевантаженості.

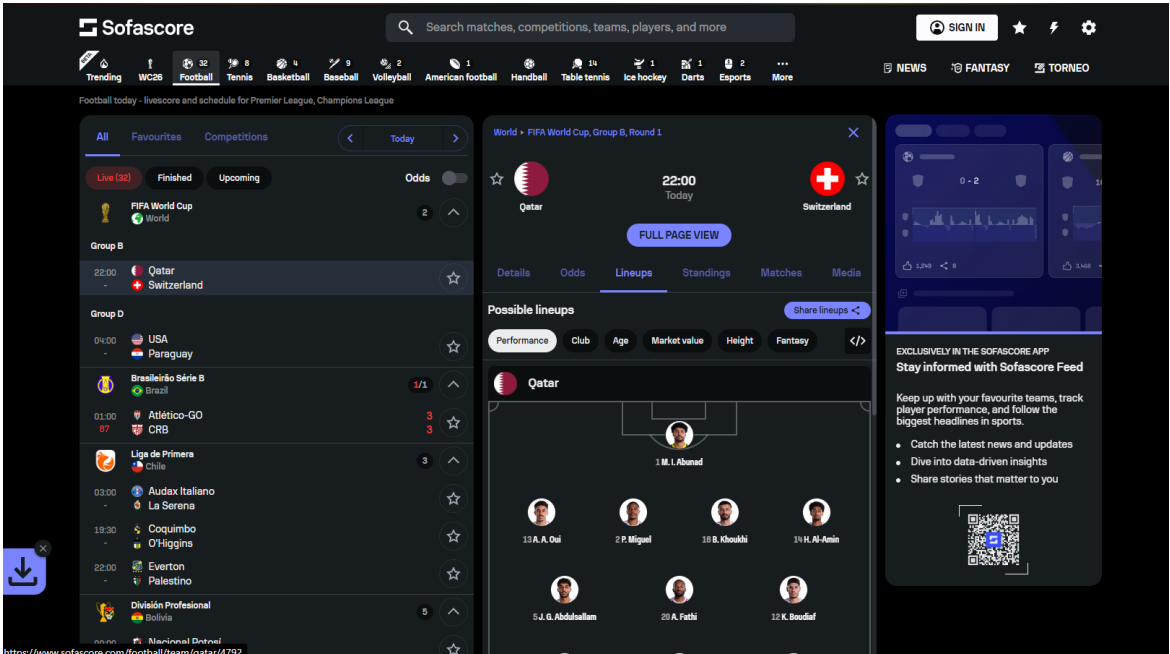


Рис. 1.6 – Інтерфейс модуля соціальної взаємодії та інтерактивних графіків

Унікальним функціональним вузлом архітектури SofaScore є вкладка спільноти та система голосувань. Замість виключної опори на сухі автоматизовані цифри платформа дозволяє користувачам залишати свої коментарі та брати участь в опитуваннях щодо ходу гри спільно з іншими вболівальниками. З інженерної точки зору реалізація такого функціоналу вимагає розробки складної серверної інфраструктури для швидкої маршрутизації повідомлень між мільйонами користувачів. Сучасні оновлення платформи також використовують гібридний підхід: інтеграцію алгоритмів машинного навчання

для генерації графіків тиску на ворота суперника [6] паралельно із соціальним зворотним зв'язком. Загалом архітектура SofaScore ефективно вирішує проблему сухої подачі статистики, додаючи соціальний контекст та інтерактивні графіки, що покращує досвід перегляду. Однак система має і технічні та функціональні недоліки. Зокрема, глибока інтеграція складної анімації графіків вимагає стабільного та швидкого інтернет-з'єднання, а офлайн-режим та розширені інструменти перегляду історичної статистики приховані за монетизаційним бар'єром. Крім того, наявність великої кількості рекламних блоків створює додаткові візуальні перешкоди під час роботи із застосунком.

Для обґрунтування функціональних вимог до розроблюваного застосунку, порівняльні критерії зведено у таблицю 1.1.

Таблиця 1.1

Порівняння інформаційних систем

Критерії порівняння	Flashscore	WhoScored	SofaScore	Система, що розробляється
Інтерактивні графіки та візуалізація	Ні	Ні	Так	Так
Налаштування власних дашбордів	Ні	Ні	Обмежено	Так
Алгоритми глибокої аналітики	Ні	Так	Так	Так
Оновлення подій у реальному часі	Так	Ні	Так	Так
Тип інтерфейсу	Жорстко лінійний	Картковий	Послідовний таймлайн	Модульний адаптивний
Архітектурний підхід	Клієнт-сервер	Клієнт-сервер	Клієнт-сервер	Клієнт-сервер (WebSocket + SPA)
Аналіз складних метрик типу xG	Ні	Ні	Обмежено	Так
Можливість експорту статистики	Ні	Обмежено	Ні	Так

Проведений розгляд програмних аналогів демонструє наявність суттєвих обмежень у наявних на ринку продуктах. Більшість із них або пропонують надто примітивний формат виведення результатів, де корисна статистика ховається за рекламними інтеграціями, або характеризуються застарілим дизайном із перевантаженою логікою подачі інформації. Жодна з досліджених вебплатформ не здатна забезпечити оптимальний баланс між зручним моніторингом поточних

рахунків та потужним професійним інструментарієм для формування персоналізованих аналітичних звітів.

Проектована інформаційна система ефективно усуває виявлені недоліки шляхом застосування сучасних архітектурних рішень. Це гарантує високу швидкодію та стабільну синхронізацію показників у реальному часі в поєднанні з інтуїтивно зрозумілим адаптивним інтерфейсом. Гнучка структура контенту надає користувачам змогу самостійно керувати відображенням статистичних метрик та взаємодіяти з великими масивами спортивних даних. Такий підхід у синергії з алгоритмами миттєвого оновлення перетворює створювану платформу на максимально ефективний засіб для автоматизації аналітичного процесу та комфортного відстеження перебігу турнірів.

1.2 Функціональні та нефункціональні вимоги до системи

Фундаментальним етапом проектування архітектури вебзастосунку виступає визначення цільової аудиторії та дослідження ключових сценаріїв експлуатації системи. Глибоке розуміння запитів кінцевого споживача дає змогу оптимізувати клієнтський інтерфейс, гарантувати позитивний досвід взаємодії та сконцентрувати ресурси на реалізації найбільш затребуваних функцій. Дослідження сегмента спортивних інформаційних технологій дало змогу виокремити три ключові групи потенційних користувачів платформи, кожна з яких висуває власні специфічні вимоги до програмного рішення.

До першого сегмента належать активні вболівальники та шанувальники спорту. Для цієї категорії критичною виступає швидкість отримання результатів під час насичених спортивних вихідних. Головна складність для них полягає в оперативному пошуку рахунків улюблених команд серед величезного масиву змагань. Розроблювана вебплатформа долає цю перешкоду завдяки механізмам персоналізації та системі своєчасних сповіщень. Надання доступу до інтерактивних турнірних таблиць і лаконічних текстових трансляцій дозволяє органічно інтегрувати моніторинг спортивних подій у щоденний ритм життя. Другу групу формують професійні спортивні аналітики та скаути. Цим фахівцям не підходять стандартні новинні стрічки, натомість вони потребують

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

максимальної гнучкості для створення власної бази статистичних знань. Основною вадою конкурентних продуктів для них є обмежений доступ до просунутих метрик. Створюваний сервіс забезпечує потужний інструментарій кастомізації, завдяки якому аналітик здатний самостійно генерувати тематичні дашборди, фільтрувати показники спортсменів та налаштовувати пріоритетність відображення специфічних даних на кшталт очікуваних голів або теплових карт. Третій сегмент охоплює адміністраторів баз даних та розробників контенту, чия відповідальність полягає в інтеграції турнірних таблиць і підтримці актуальності статистики. Для даної групи архітектура системи реалізує принцип єдиного централізованого джерела істини. Будь-які зміни результатів матчів на сервері автоматично синхронізуються з усіма підключеними клієнтами, що гарантує абсолютну достовірність та цілісність аналітичного матеріалу.

Спираючись на сформовані профілі аудиторії, було визначено базові патерни взаємодії користувачів із системою, які підтверджують спроможність платформи задовольняти різноманітні потреби. У межах щоденної експлуатації реалізується сценарій моніторингу поточних подій. Користувач заходить на платформу, після чого система на базі попередньо збережених налаштувань автоматично генерує стрічку релевантних матчів. Цей етап включає перегляд живих рахунків, аналіз інтерактивних графіків ігрової переваги та ознайомлення зі стартовими складами. При зміні ситуації на ігровому полі клієнтський додаток реактивно оновлює локальне сховище даних, миттєво перераховує турнірне положення учасників та синхронізує оновлений статус із сервером. Інший формат взаємодії розкривається в процесі створення персоналізованого аналітичного модуля. Користувач ініціює побудову нового дашборда шляхом вибору необхідних ліг або спортсменів. Система своєю чергою застосовує алгоритми агрегації даних і пропонує найбільш релевантні варіанти візуалізації показників. Збереження створеної конфігурації генерує оптимізовану структуру даних, завдяки чому звіт стає доступним для подальшого використання у будь-який момент часу. Крім цього, для виявлення глобальних трендів передбачено взаємодію з модулем графіків, де система формує діаграми на основі накопиченої

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

командної статистики. За необхідності підтримується сценарій експорту статистичних вибірок у зручних текстових форматах для подальшої обробки в сторонніх програмах, що гарантує відкритість та мобільність акумульованої інформації.

Функціональне наповнення розроблюваного програмного комплексу ґрунтується на чітко визначеному переліку завдань, виконання яких повністю покриває потреби кінцевого споживача. Такий підхід дає змогу вибудувати надійне аналітичне середовище з узгодженою роботою всіх модулів для забезпечення безперебійного потоку спортивної інформації. Архітектурна модель вебзастосунку спирається на систему управління доступом на основі ролей. Зазначений метод гарантує високий рівень безпеки бази даних, надійну авторизацію та гнучку взаємодію клієнтської частини з бекендом через прикладний програмний інтерфейс. Встановлені правила формують суворе розмежування прав та охоплюють увесь життєвий цикл роботи зі спортивним контентом від стадії збору й адміністрування турнірних таблиць до безпосереднього перегляду показників та збереження персоналізованих графіків. Детальний перелік функціональних можливостей для кожної користувачької ролі наведено у таблиці 1.2.

Таблиця 1.2

Функціональні вимоги до мобільного застосунку

Роль користувача	Функціональні можливості у системі
Гість або неавторизований користувач	Перегляд головного екрана результатів. Ознайомлення з базовою статистикою загальнодоступних матчів. Реєстрація нового акаунта та авторизація через електронну пошту або сторонні сервіси.
Користувач системи	Управління профілем та налаштування персональної стрічки новин. Створення власних аналітичних дашбордів. Перегляд інтерактивних графіків поточних матчів. Використання інструментів порівняння спортивних команд.
Менеджер спортивних даних	Управління глобальним каталогом ліг та команд. Модерація розкладу турнірів. Синхронізація історичних результатів з офіційними джерелами. Формування шаблонів складних статистичних метрик через адміністративну панель.
Системний адміністратор	Повний контроль над системою та базою користувачів. Призначення ролей. Аналітика загальної активності аудиторії вебплатформи. Налаштування параметрів безпеки та керування ключами доступу зовнішніх провайдерів спортивної статистики.

На противагу функціональним специфікаціям нефункціональні вимоги визначають якісні атрибути програмного продукту, деталізуючи параметри надійності, швидкодії та безпеки виконання закладених функцій [9]. З метою гарантування найвищого рівня якості створюваної аналітичної платформи було сформовано набір чітких інженерних метрик.

Ці показники регламентують швидкість відгуку системи на запити користувачів, особливо в періоди пікових навантажень під час насичених спортивних вікендів. Враховуючи орієнтацію платформи на миттєву обробку інформації, критичним показником виступає тривалість початкового завантаження сторінки, яка має вкластися у дві секунди. Реакція інтерфейсу на взаємодію аналітика при навігації між вкладками статистики повинна складати не більше ста мілісекунд. Рендеринг інтерактивних графіків перебігу матчу має виконуватися стабільно, гарантуючи візуальну плавність незалежно від обчислювальних потужностей клієнтського браузера.

Зазначені вимоги базуються на необхідності забезпечення безперервного потоку оновлень результатів матчів. Фундаментальною умовою є підтримка стабільного сокетного з'єднання для миттєвої доставки змін рахунку без примусового перезавантаження сторінки з подальшим збереженням даних у локальному кеші. З метою мінімізації мережевого трафіку та зниження навантаження на серверну частину трансляція статистичних масивів повинна відбуватися у стисненому вигляді. При цьому пріоритетним напрямом є інтеграція сучасних патернів управління станом безпосередньо на стороні клієнта.

Безпекові аспекти відіграють критичну роль з огляду на обробку персоналізованих користувацьких налаштувань. Програмне рішення зобов'язане гарантувати абсолютну цілісність конфігурацій дашбордів навіть у випадках раптового розриву з'єднання або випадкового закриття вікна браузера. Процес аутентифікації має спиратися на використання захищених цифрових токенів, тоді як зберігання облікових записів у базі даних повинно підпорядковуватися актуальним протоколам серверного шифрування. Крім того, закладена

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

архітектура має забезпечувати високий ступінь масштабованості для безперешкодного розширення переліку видів спорту шляхом застосування компонентної моделі розробки клієнтського інтерфейсу.

1.3 Вибір і обґрунтування архітектурного рішення

Загальна ефективність створюваної мультиспортивної платформи прямо пропорційна надійності її внутрішньої архітектури, швидкості обробки живих потоків статистики та безпеці зберігання користувацьких конфігурацій. Для формування оптимальної моделі організації програмного коду доцільно провести порівняльний аналіз поширених архітектурних патернів.

Традиційно під час проектування комплексних вебдодатків розробники звертаються до класичних монолітних структур, однак такий підхід рідко задовольняє актуальні вимоги до реактивного масштабування. У специфічних умовах спортивної інформаційної системи виникають непередбачувані пікові навантаження під час топових протистоянь, що вимагає синхронної обробки десятків тисяч подій за секунду. За таких обставин монолітна парадигма демонструє свою неефективність та створює небезпечну єдину точку відмови. Наприклад, надмірне навантаження на модуль оновлення поточних рахунків здатне спровокувати повну зупинку всього сервісу. Додатково сильна зв'язність внутрішніх компонентів гальмує процес розробки, перешкоджаючи ізольованому розгортанню та тестуванню окремих підсистем на зразок блоку візуалізації чи фентезі-драфту. Ці фактори зумовлюють гостру потребу у переході до гнучкіших модульних парадигм.

У цьому контексті архітектурний підхід, запропонований Робертом Мартіном [10], виступає потужною альтернативою. Фундаментальним принципом чистої архітектури є повна незалежність центрального ядра системи та її бізнес-логіки від зовнішньої інфраструктури, систем управління базами даних чи сторонніх програмних фреймворків.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

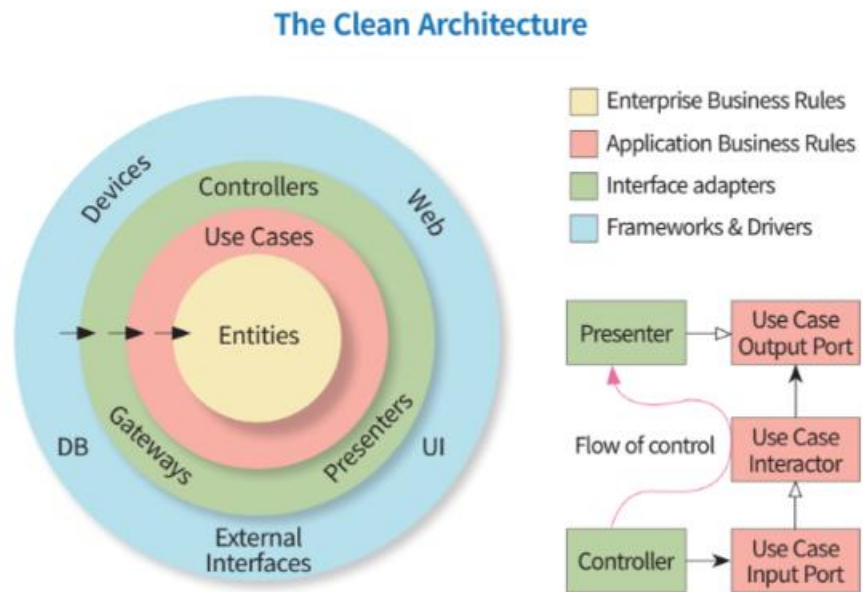


Рис. 1.9 – Концепція Clean Architecture

Згідно з обраною концепцією вектор залежностей завжди спрямований до центру системи. Внутрішній рівень інкапсулює ключові бізнес-правила та фундаментальну логіку. Далі розташовуються сценарії використання, тоді як зовнішні контури відповідають за клієнтський інтерфейс, мережеві комунікації та взаємодію з локальним кешем. Впровадження такої багаторівневої ізоляції виступає найбільш обґрунтованим технічним рішенням для створюваної мультиспортивної платформи. Це гарантує повне відокремлення центрального ядра та алгоритмів обробки потокової інформації від візуального відображення. На рівні клієнтського представлення зазначену парадигму доцільно інтегрувати з патерном глобального управління станом на зразок Redux [11]. Такий симбіоз забезпечить реактивне оновлення дашбордів під час зміни поточних рахунків, а також гарантуватиме гнучкість кодової бази та спростить процес покриття модулів автоматизованими тестами.

Наступним кроком архітектурного проектування виступає визначення методів зберігання динамічних масивів спортивної статистики. Головний інженерний виклик полягає у забезпеченні суворої цілісності турнірних баз одночасно з гнучкістю збереження метрик для складних поєдинків. Аналіз традиційного підходу реляційних баз [12], де кожен показник зберігається в окремій таблиці, виявив його неефективність для аналітичних систем. Значна

кількість операцій об'єднання під час завантаження сторінки матчу створює надмірне серверне навантаження. Це неодмінно призводить до затримок у роботі інтерфейсу та втрати актуальності поточних подій. Натомість застосування суто документо-орієнтованих систем [13] дає змогу швидко завантажувати всю статистику єдиним об'єктом, але породжує ризики порушення консистентності під час оновлення загальних турнірних таблиць через відсутність жорстких зв'язків.

Для вирішення цієї дилеми розробниками обрано гібридну модель збереження інформації. На боці браузера застосовується легковагове локальне сховище [14], яке гарантує швидке кешування персональних налаштувань. На серверному рівні інтегрується потужна реляційна система PostgreSQL [15]. Вона забезпечує надійну транзакційність для роботи з профілями користувачів та історичними масивами результатів. Водночас конфігурації складних ігрових подій зберігаються у текстовому форматі безпосередньо всередині реляційних таблиць. Описане рішення дозволяє розширювати перелік спортивних метрик без постійного коригування схеми бази даних, а також виконувати оперативну синхронізацію через сокети та завантажувати повну структуру результатів туру за допомогою одного мережевого запиту.

З метою забезпечення високого рівня доступності, синхронізації даних у реальному часі та загальної безпеки проекту було прийнято рішення про використання сучасної хмарної інфраструктури. Такий крок дозволяє делегувати технічне обслуговування апаратної частини надійним провайдерам і повністю сконцентруватися на розробці бізнес-логіки. Як базову платформу інтегровано сучасну екосистему інструментів Google Cloud [16]. Спеціалізовані сервіси аутентифікації [17] гарантують безпечну реєстрацію та вхід за допомогою цифрових токенів і підтримують авторизацію через соціальні мережі. Критично важливим компонентом платформи виступає модуль хмарних сповіщень. Він реалізує механізм інформування про забиті м'ячі та зміну рахунку, що відчутно підвищує показники утримання аудиторії. Для оперативного виявлення збоїв в інтерфейсі підключено систему автоматичного моніторингу помилок.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

Зважаючи на неспроможність браузера локально зберігати великі обсяги медіаданих, розміщення важких графічних елементів на основному сервері є нерациональним. Управління статичним візуальним контентом здійснюється через спеціалізовані хмарні сервіси [18]. Вони автоматично оптимізують зображення відповідно до розміру екрана та якості мережевого з'єднання, гарантуючи швидке завантаження дашбордів.

Хостинг серверної логіки та розгортання глобальної бази даних реалізуються з використанням сучасних хмарних платформ [19]. Обрана інфраструктура забезпечує функціонування оптимізованого бекенду, автоматичне налаштування захищених каналів зв'язку та своєчасне масштабування обчислювальних ресурсів у моменти пікових навантажень. Для протидії мережевим атакам та автоматизованим парсерам впроваджено інфраструктуру безпеки Cloudflare [20]. Вона виконує роль захисного проксі-шару та гарантує надійне шифрування трафіку між клієнтом і хмарою. Оскільки дистрибуція та оновлення продукту потребують високої надійності, для автоматизації етапів тестування та розгортання нових версій застосовуються передові інструменти безперервної інтеграції, серед яких конвеєри GitHub Actions [21].

1.4 Вибір стеку технологій та апаратного забезпечення

Визначення інструментальних засобів виступає базовим етапом проєктування, від якого безпосередньо залежить темп розробки, рівень надійності, потенціал масштабування та легкість технічної підтримки готового продукту. У поточному підрозділі здійснено аналіз та аргументацію вибору технологій для клієнтського і серверного рівнів платформи, допоміжних середовищ програмування, а також сформовано вимоги до апаратних потужностей.

Фундаментом для створення вебплатформи слугує технологічний стек, цілком побудований навколо екосистеми JavaScript та її статично типізованого розширення TypeScript [22]. Застосування універсального мовного підходу для фронтенду та бекенду дає змогу значно пришвидшити процес написання коду

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

шляхом повторного використання спільних типів і елементів бізнес-логіки. Цей аспект відіграє вирішальну роль під час оперування складними структурами спортивних даних та алгоритмами агрегації статистики.

Фронтенд виступає найбільш завантаженим компонентом системи, відповідаючи за реалізацію логіки взаємодії з динамічними аналітичними матеріалами. Базовим інструментом розробки інтерфейсу визначено бібліотеку React у поєднанні з фреймворком Next.js [23]. Таке рішення є оптимальним для проєктування комплексних інформаційних систем. Воно спирається на жорсткий компонентний підхід, згідно з яким будь-який візуальний елемент на кшталт турнірної таблиці чи графіка ізолюється в незалежний модуль для полегшення масштабування. Застосування технології серверного рендерингу забезпечує швидке відображення сторінок, тоді як реактивність бібліотеки гарантує плавну зміну показників без примусового оновлення вікна браузера.

Бекенд виконує функції обробки мережових запитів, реалізації бізнес-правил, забезпечення безпечної автентифікації та комунікації з базою даних. Програмною основою сервера обрано середовище Node.js [24]. Враховуючи специфіку платформи з її масовими напливами аудиторії під час резонансних спортивних подій, серверна частина має гарантовано витримувати екстремальні пікові навантаження. Асинхронна подієва архітектура Node.js наділяє бекенд здатністю паралельно обробляти тисячі активних з'єднань. Роль основного вебфреймворку відведено Express.js [25]. Він пропонує гнучкий інструментарій для формування стабільного програмного інтерфейсу, налаштування маршрутизації та безперешкодного впровадження алгоритмів трансляції потокових даних.

Для підтримки високого темпу написання програмного коду залучено спеціалізоване середовище. У ролі базового редактора виступає Visual Studio Code [26]. Перевагами цього продукту є глибока оптимізація, наявність масштабної екосистеми плагінів для зневадження вебпроектів та безшовна інтеграція з лінтерами. Моделювання логічної структури застосунку здійснюється за допомогою PlantUML [27]. Генерація діаграм у цьому сервісі

відбувається через описову мову, що дає можливість зберігати архітектурні схеми безпосередньо в репозиторії системи контролю версій Git [28] поруч із вихідним кодом і зручно контролювати історію редагувань.

Технічні характеристики робочої станції розробника продиктовані потребою паралельного функціонування редактора коду, локального бекенду, контейнерів баз даних та групи браузерів. Це вимагає наявності сучасного багатоядерного процесора та мінімум шістнадцяти гігабайтів оперативної пам'яті. Вимоги до пристроїв кінцевих споживачів залишаються базовими. Для комфортної роботи достатньо будь-якого актуального смартфона або комп'ютера зі свіжою версією браузера та обсягом пам'яті, достатнім для візуалізації аналітичних діаграм. Обов'язковою умовою є наявність стабільного інтернет-з'єднання для підтримки сокетних трансляцій і безперервного доступу до живої статистики турнірів.

Невіддільною складовою архітектури серверного рівня є вибір надійної системи управління базами даних (СУБД) та сучасних засобів взаємодії з нею. Як основне сховище обрано об'єктно-реляційну СУБД PostgreSQL. Її вибір аргументований високою транзакційною надійністю (ACID), потужним механізмом індексації та здатністю ефективно обробляти складні аналітичні запити з множинними об'єднаннями таблиць. Для комунікації між сервером Node.js та базою даних застосовується сучасний інструмент об'єктно-реляційного відображення Prisma ORM. Вибір Prisma зумовлений її здатністю автоматично генерувати строгі типи (Type-Safe API) на основі декларативної схеми даних. Це повністю усуває ризик виникнення помилок невідповідності типів на етапі виконання програми та значно прискорює написання складних міграцій бази даних.

Для реалізації візуальної частини клієнтського інтерфейсу залучено фреймворк Tailwind CSS. На відміну від класичних методологій написання каскадних таблиць стилів (таких як BEM чи OOCSS), Tailwind використовує утилітарний підхід (Utility-First). Це дозволяє стилізувати компоненти безпосередньо в розмітці JSX, що запобігає неконтрольованому розростанню

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

CSS-файлів та виникненню конфліктів селекторів. Поєднання Tailwind CSS із бібліотеками готових доступних компонентів (Radix UI) гарантує кросбраузерність інтерфейсу та його відповідність міжнародним стандартам доступності (a11y).

Оскільки спортивна аналітична платформа працює з фінансовими транзакціями (преміум-підписками) та персональними даними користувачів, обов'язковою інженерною вимогою є імплементація надійної стратегії тестування коду. Для модульного та інтеграційного тестування серверної бізнес-логіки застосовується фреймворк Jest. Він забезпечує швидке виконання тестів у віртуальному середовищі Node.js та підтримує потужні механізми імітації (Mocking) зовнішніх залежностей. Для наскрізного (End-to-End) тестування реальних користувацьких сценаріїв обрано інструмент Playwright, який автоматизує дії в різних браузерних рушіях і перевіряє коректність роботи всього ланцюжка: від клієнтського кліку до збереження даних у PostgreSQL.

З метою забезпечення ідентичності середовищ локальної розробки та промислової експлуатації (Production) використовується технологія контейнеризації Docker. Вона дозволяє надійно ізолювати базу даних та основний вебзастосунок у незалежні легкосприйнятливі контейнери. Застосування файлу конфігурації docker-compose уніфікує процес розгортання інфраструктури, що дозволяє будь-якому розробнику запустити повну копію інформаційної системи однією консольною командою, незалежно від операційної системи його комп'ютера.

Висновки до першого розділу

У першому розділі виконано всебічний аналіз предметної області та доведено актуальність створення нового вебзастосунку для агрегації й аналітики спортивної статистики. Дослідження наявних на ринку програмних рішень виявило гостру потребу в розробці платформи, здатній гармонійно поєднати передові алгоритми обробки потокових даних із високою гнучкістю налаштування персоналізованих дашбордів. Саме такого функціонала критично

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

бракує сучасним закритим аналітичним сервісам та класичним новинним трекерам із занадто спрощеною логікою.

Аналіз потреб цільової аудиторії дав змогу чітко сегментувати майбутніх користувачів та формалізувати ключові патерни їхньої взаємодії з системою. Опираючись на ці дані, було аргументовано вибір сучасного технологічного стеку для побудови масштабованого вебпроєкту. Використання екосистеми React і фреймворку Next.js у синергії з типізацією TypeScript гарантує стабільну продуктивність клієнтського інтерфейсу та високу надійність кодової бази. Для вирішення завдання ефективного управління великими масивами даних впроваджено гібридну архітектурну модель. Вона поєднує механізми локального браузерного кешування для миттєвого доступу до профілів та потужну реляційну базу PostgreSQL на бекенді для надійного збереження історичних зведень.

Також у межах розділу деталізовано функціональні вимоги до програмного продукту на основі рольової моделі управління доступом та визначено ключові нефункціональні показники якості. Підвищену увагу приділено метрикам швидкодії клієнтського боку, безпеці трансляції інформації через хмарну інфраструктуру та збереженню стабільності системи в умовах пікових навантажень під час масових спортивних турнірів.

Сформований комплекс вимог, критичний розгляд архітектурних концепцій, зокрема принципів чистої архітектури для ізоляції бізнес-логіки та патернів управління станом для реактивного рендерингу інтерфейсу, разом із обґрунтованим вибором інструментарію закладають міцний інженерний фундамент. Зібрана теоретико-технічна база гарантує безпечний і плавний перехід до наступної стадії життєвого циклу розробки, яка передбачає логічне проєктування структур баз даних та безпосередню програмну реалізацію всіх аналітичних модулів системи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Моделювання варіантів використання та поведінки системи

Моделювання варіантів використання виступає базовим концептуальним кроком для переходу від теоретичного формування вимог до етапу практичного конструювання архітектури мультиспортивної вебплатформи. На цій стадії об'єктно-орієнтованого аналізу встановлюються межі програмного продукту з виокремленням клієнтського боку та серверної панелі управління. Одночасно з цим відбувається ідентифікація ключових дійових осіб та суворя формалізація сценаріїв їхньої взаємодії із системою. Такий підхід допомагає перетворити розрізнені функціональні запити на цілісну та логічно структуровану архітектурну схему. Вичерпна характеристика кожної виявленої користувацької ролі в межах розроблюваного продукту представлена у таблиці 2.1.

Таблиця 2.1

Ідентифікація акторів системи

Актор	Опис ролі та зона відповідальності
Гість	Неавторизований відвідувач вебзастосунку. Має доступ до загальної інформації про результати матчів та базової статистики команд. Ініціює процедуру реєстрації або безпечного входу.
Аналітик (Авторизований користувач)	Авторизований користувач і основний суб'єкт взаємодії. Наділений повним доступом до аналітичного середовища для перегляду розширеної статистики турнірів, порівняльних графіків та ігрових показників. Керує налаштуваннями власних дашбордів.
Менеджер спортивних даних	Фахівець, що працює через захищену панель адміністратора. Відповідає за наповнення бази даних актуальним спортивним контентом для створення категорій ліг, додавання нових команд та модерації розкладу матчів.
Системний адміністратор	Особа, яка забезпечує технічне управління серверною інфраструктурою. Контролює політику безпеки, управляє правами доступу, моніторить навантаження та конфігурує інтеграції з офіційними статистичними провайдерами та сервісами авторизації.

Згідно зі спроектованою моделлю контролю доступу, архітектура платформи передбачає наявність чотирьох відокремлених акторів із суворо визначеними сферами відповідальності. Зважаючи на архітектурні особливості сучасних вебдодатків, усі доступні сценарії використання структурно розділені на клієнтську браузерну частину та закритий серверний модуль адміністрування.

У відкритому середовищі клієнтського додатка виконуються базові прецеденти перегляду загальної турнірної статистики. Також тут відбуваються процеси реєстрації нових користувачів, безпечної авторизації за допомогою інтеграції із соціальними мережами та відновлення втрачених облікових даних. Зазначені сценарії відіграють роль універсального стартового етапу взаємодії з інформаційною системою.

Цей простір акумулює ключовий інструментарій для поглибленого дослідження спортивних заходів авторизованими користувачами. Даний функціональний блок охоплює моніторинг живих рахунків, аналіз інтерактивних діаграм перебігу матчу, зіставлення командних показників та відстеження просунутої статистики окремих спортсменів у межах турнірів. В окрему категорію винесено прецеденти управління власним профілем, формування списків улюблених клубів та налаштування персонального доступу до спеціалізованих засобів візуалізації інформації.

Закрита частина системи розроблена спеціально для контролю над глобальними ресурсами проєкту Спортивним менеджером та системним Адміністратором. У цьому модулі втілено сценарії управління розкладами змагань у реляційній базі даних, редагування актуальних складів та нагляду за коректністю отримання потокових статистичних результатів. Крім того, адміністративний блок містить системні прецеденти контролю працездатності сервера, управління базою користувачів та налаштування безпекових ключів для інтеграції із зовнішніми джерелами даних.

На основі виявлених акторів та формалізованих прецедентів побудовано розширену діаграму варіантів використання системи подану на (рисунку 2.1).

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

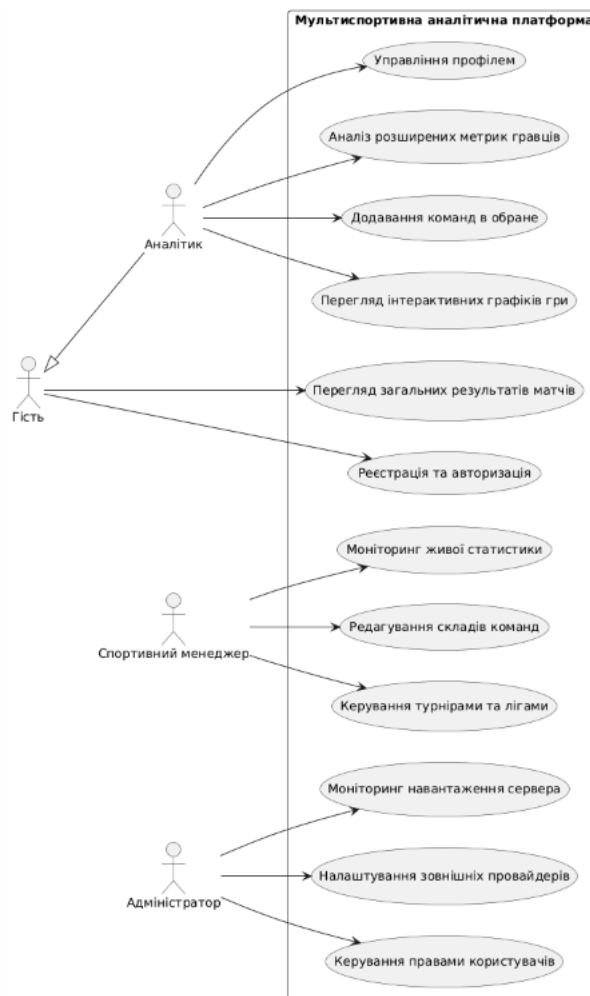


Рис. 2.1 – Діаграма варіантів використання системи

Подана візуалізація чітко розмежовує зони відповідальності: актори клієнтського рівня, такі як Гість та Аналітик, взаємодіють виключно з публічним інтерфейсом вебзастосунку для аналізу спортивних подій та управління власним профілем. Водночас службові ролі наділені ексклюзивними правами доступу до панелі управління для модерації турнірних таблиць та технічного налаштування бекенду. Спроектowana модель слугує надійним фундаментом для подальшого формування об'єктної структури системи та реляційної схеми бази даних.

З метою підготовки до розробки алгоритмів серверного інтерфейсу та клієнтської частини було складено специфікації для ключових прецедентів. Специфікація деталізує кожен крок взаємодії актора із системою, визначає необхідні передумови та описує альтернативні сценарії обробки помилок на зразок збою при підключенні до статистичного провайдера або відхилення

токена авторизації. У таблиці 2.2 наведено специфікацію базового варіанта використання.

Таблиця 2.2

Специфікація прецеденту "Реєстрація та Авторизація"

Атрибут	Опис
Назва прецеденту	Реєстрація та Авторизація
Актор	Гість
Мета	Отримати персоналізований доступ до розширеної статистики, інтерактивних графіків та аналітики команд на платформі.
Передумови	Відвідувач відкрив вебзастосунок у браузері та має активне підключення до мережі Інтернет.
Основний сценарій (Успішний)	1. Гість обирає метод реєстрації або входу через електронну пошту або сторонні сервіси. 2. Система відображає відповідний інтерфейс. 3. Гість вводить облікові дані або підтверджує доступ через соціальний акаунт. 4. Клієнтська частина проводить первинну валідацію. 5. Сервер перевіряє дані в реляційній базі даних або верифікує токен провайдера і генерує безпечний цифровий ключ доступу. 6. Застосунок зберігає сесію та відкриває доступ до головного аналітичного екрана.
Альтернативний сценарій (Помилки)	4а. Некоректний формат даних: система виводить підказку щодо виправлення помилок на зразок невалідного формату пошти. 5а. Помилка на боці сервера: сервер повертає помилку авторизації, вебклієнт пропонує відновити пароль або змінити дані.
Постумови	Користувач переходить у статус авторизованого Аналітика, активується механізм доступу до його персоналізованих дашбордів із хмарним сховищем.

Ключовим функціоналом розроблюваної платформи є можливість розширення базового контенту шляхом формування індивідуальних статистичних дашбордів. У таблиці 2.3 наведено інженерну специфікацію прецеденту «Формування персоналізованого аналітичного модуля», який є найбільш складним з погляду реалізації клієнтської реактивної логіки та синхронізації станів з бекендом. Висока складність цього процесу зумовлена необхідністю безперервної обробки динамічних потоків даних та забезпечення миттєвого оновлення візуальних віджетів без втрати продуктивності інтерфейсу. Тому наведена специфікація детально описує архітектурні вимоги, структуру компонентів та механізми управління станами, що гарантують стабільну та відмовостійку роботу модуля.

Специфікація прецеденту “Формування персоналізованого аналітичного модуля”

Атрибут	Опис
Назва прецеденту	Формування персоналізованого аналітичного модуля
Актор	Аналітик
Мета	Створити власну підбірку статистичного контенту для відстеження улюблених команд та гравців.
Передумови	Аналітик авторизований у вебзастосунку та перейшов до розділу створення персональних дашбордів.
Основний сценарій (Успішний)	1. Аналітик ініціює створення нового дашборда. 2. Клієнт формує базову структуру об'єкта для нового звіту. 3. Користувач додає команди та показники, використовуючи функцію пошуку. 4. Система автоматично інтегрує вибрані сутності у графіки та генерує шаблони візуалізації. 5. Користувач зберігає створений дашборд. 6. Застосунок серіалізує дані та відправляє запит на серверний інтерфейс. 7. Сервер валідує отриману структуру та створює відповідний запис у базі даних PostgreSQL, прив'язуючи його до профілю.
Альтернативний сценарій (Помилки)	6а. Відсутність мережі: застосунок перехоплює помилку, сповіщає користувача та зберігає зміни в локальному браузерному сховищі для подальшої фонові синхронізації. 7а. Помилка валідації на сервері: сервер відхиляє запит, система вказує на некоректно заповнені поля графіків.
Постумови	Оновлена структура персонального модуля збережена в базі даних та доступна для вивчення з будь-якого підключеного пристрою.

Окрім публічного функціоналу, стабільність інформаційної системи підтримується адміністративними процесами модерації та контролю достовірності контенту. Специфікація прецеденту модерації демонструє логіку управління станами турнірів у глобальній базі даних через вебпанель адміністратора. Цей процес передбачає застосування механізмів транзакційної цілісності та безпечної авторизації для захисту від несанкціонованих змін результатів змагань. У наведеній нижче таблиці деталізовано алгоритм взаємодії адміністратора з бекендом, включаючи етапи суворої валідації вхідних даних та безпечне каскадне оновлення пов'язаної спортивної аналітики.

Специфікація прецеденту «Призупинення відображення турніру»

Атрибут	Опис
Назва прецеденту	Призупинення відображення турніру або матчу
Актор	Менеджер спортивних даних, Адміністратор
Мета	Обмежити доступ користувачів до публічного матчу, який містить помилкові дані або скасований.
Передумови	Модератор успішно авторизований у панелі керування та перейшов до розділу управління розкладом.
Основний сценарій (Успішний)	1. Модератор знаходить турнір через систему фільтрації. 2. Обирає дію деактивації відображення. 3. Система вимагає вказати причину призупинення трансляції події. 4. Після підтвердження сервер змінює статус об'єкта в базі даних на призупинений. 5. Система автоматично надсилає оновлення всім підключеним клієнтам про скасування події.
Альтернативний сценарій (Помилки)	4а. Спроба блокування базової ліги: система перевіряє права доступу та відхиляє операцію для захищених глобальних сутностей.
Постумови	При спробі завантаження матчу клієнти отримують статус недоступності, а подія зникає із загального розкладу.

Розроблені специфікації прецедентів визначають чіткі архітектурні контракти між компонентами платформи, куди входять браузерний клієнт, панель адміністрування та сервер. Вони демонструють, що кожна дія актора супроводжується комплексною серверною логікою через безпечну автентифікацію, валідацію прав доступу, управління персональними налаштуваннями, інтеграцію із зовнішніми постачальниками статистики та синхронізацію станів у базі даних PostgreSQL. Ця деталізація слугує фундаментальною базою для наступного етапу логічного проектування об'єктно-орієнтованої структури класів та маршрутів серверного інтерфейсу платформи.

2.2 Проектування логічної структури бази даних

З метою гарантування надійного збереження інформації та підтримки безкомпромісної транзакційної цілісності масивів спортивної статистики було обрано реляційну систему управління базами даних. У межах сучасного підходу до веброзробки ця концепція втілюється через використання локального браузерного кешу [14] та розгортання потужного середовища PostgreSQL на серверному рівні [4]. Інтеграція класичної реляційної моделі повністю задовольняє вимоги щодо надійності обробки транзакцій. Це виступає критично важливим аспектом під час оперування історичними даними спортивних клубів

та збереження персональних конфігурацій користувачів, оскільки будь-яка втрата чи викривлення інформації неминуче призведе до помилок у формуванні турнірних таблиць.

Поряд із цим підсистема персоналізації вимагає опрацювання максимально динамічних структур даних через наявність специфічних метрик у кожному виді спорту. Для ефективного керування такими комплексними конфігураціями матчів архітектура застосунку активно залучає нативний текстовий формат обміну даними безпосередньо всередині таблиць PostgreSQL [15]. Подібний симбіоз забезпечує ідеальний компроміс між надійністю строгих реляційних зв'язків для базових сутностей та гнучкістю документо-орієнтованої парадигми [13] у процесі збереження деталізованої статистики спортсменів. Впроваджене архітектурне рішення суттєво мінімізує необхідність у проведенні складних міграцій бази даних під час інтеграції нових видів спорту та допомагає уникнути поширених інженерних антипатернів [12].

Спроектowana візуальна схема бази даних наочно ілюструє базові предметні сутності аналітичної платформи разом з їхніми внутрішніми атрибутами, а також механізми суворої ідентифікації через первинні та зовнішні ключі. Серед логічних залежностей варто виокремити класичні зв'язки формату один-до-багатьох, які об'єднують облікові записи з їхніми персональними дашбордами, та складні відношення між гравцями і спортивними організаціями. Сформована у такий спосіб архітектурна модель генерує єдиний узгоджений простір даних. Це гарантує надзвичайно швидку вибірку необхідної інформації через серверний програмний інтерфейс [2] для її миттєвої доставки на браузерний клієнт та подальшої синхронізації з хмарною інфраструктурою [19]. Поданий графічний концепт виступає не лише засобом наочного відображення потоків інформації, але й потужним інструментом запобігання виникненню розрахункових аномалій під час турнірів.

Запропонована візуальна схема відношень ілюструє базові сутності аналітичного комплексу разом з їхніми внутрішніми атрибутами, первинними та зовнішніми ключами. Окремо варто виділити класичні зв'язки типу один-до-

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		36

багатьох, які поєднують профілі користувачів із їхніми збереженими статистичними дашбордами, а також складні відношення між гравцями та спортивними клубами. Спроектowana таким чином схема утворює єдиний інформаційний простір, що забезпечує надзвичайно швидку вибірку потрібних даних через серверний інтерфейс [2] для миттєвої передачі на браузерний клієнт та надійну синхронізацію з потужним хмарним середовищем [19].

У підсумку ретельно продумана структура відношень формує надійно узгоджений та оптимізований інформаційний фундамент. Глибоке опрацювання запитів гарантує безпрецедентну швидкість вибірки результатів. Цей фактор є абсолютно критичним для мінімізації мережових затримок та забезпечення візуально плавної роботи інтерактивних графіків у процесі обробки живих потокових оновлень зі спортивних арен.

Поданий графічний концепт ілюструє ключові предметні сутності платформи, їхні внутрішні атрибути, механізми суворої ідентифікації через головні та підпорядковані ключі, а також встановлені між ними логічні залежності. Ця архітектурна модель є не лише наочним візуальним представленням потоків обробки даних, але й дієвим інструментом для запобігання виникненню аномалій розрахунків чи надлишковості інформації під час турнірів. Завдяки застосуванню принципів нормалізації, розроблена схема гарантує високу стійкість до пікових навантажень, що неминуче виникають при масових зверненнях користувачів до статистики в реальному часі. До того ж, закладена гнучкість структури дозволяє безперешкодно масштабувати систему в майбутньому, додаючи нові аналітичні метрики та види спорту без необхідності рефакторингу існуючого ядра бази даних. Такий рівень стандартизації структур даних суттєво спрощує безпечну інтеграцію платформи із зовнішніми постачальниками спортивної статистики через API. Відповідно, команда розробників отримує можливість зосередитися на вдосконаленні бізнес-логіки системи, покладаючись на абсолютно надійний і прогнозований рівень зберігання інформації.

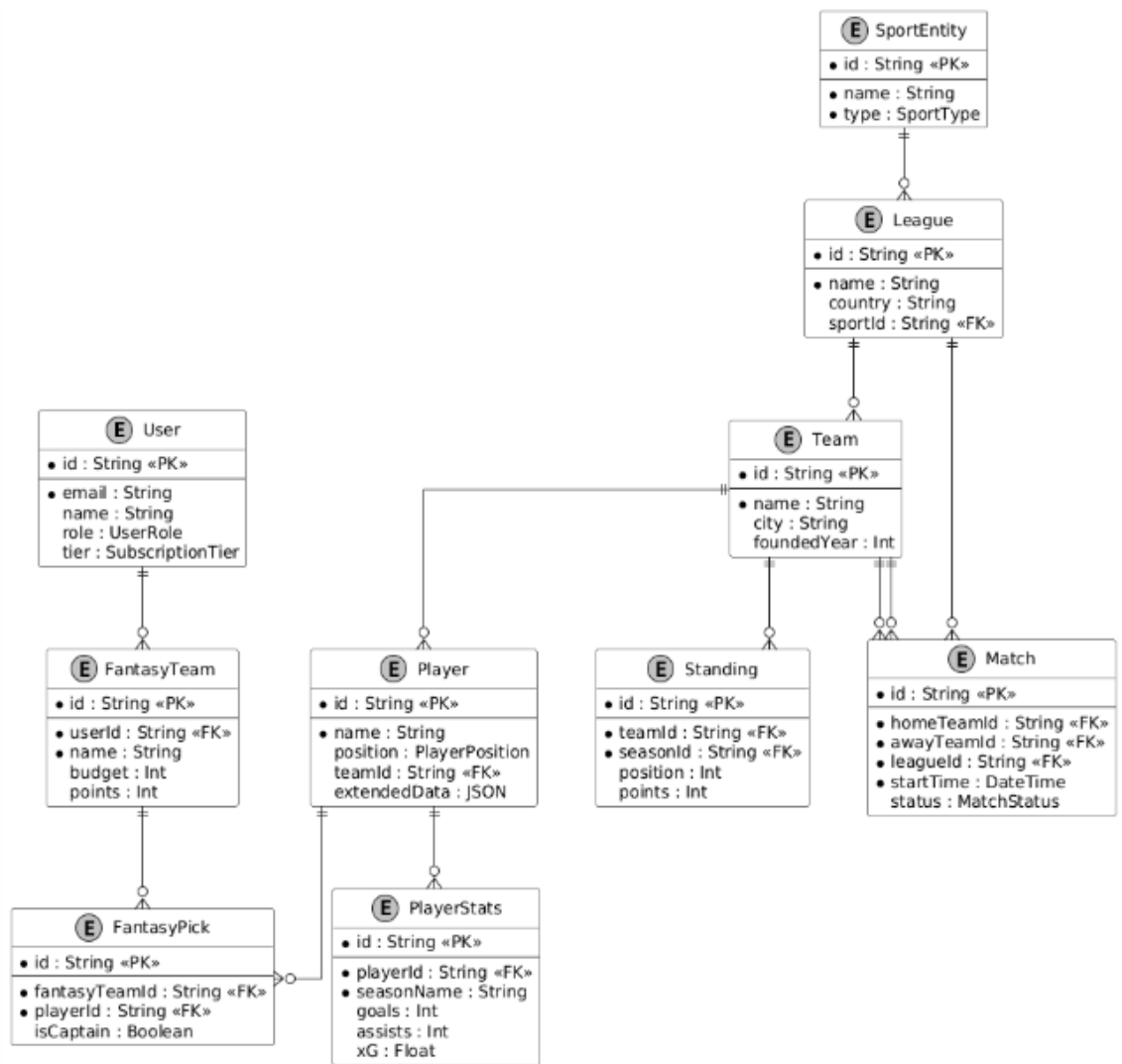


Рис. 2.2 – ЕР-модель реляційної бази даних

Спроектована реляційна схема даних включає строгі фізичні обмеження цілісності (Integrity Constraints) для забезпечення узгодженості та швидкодії. Технічні специфікації ключових обмежень зовнішніх ключів та індексів наведено у таблиці 2.5.

Застосування механізму каскадного видалення гарантує відсутність незв'язаних записів у базі, тоді як В-Tree індексація висококардинальних полів суттєво зменшує час відгуку системи під час генерації аналітичних дашбордів. Це дозволяє підтримувати високу продуктивність та цілісність платформи навіть за умов стрімкого зростання обсягів даних.

Таблиця 2.5

Специфікація фізичних обмежень та індексів бази даних

Сутність (Таблиця)	Цільове поле	Тип обмеження	Опис / Поведінка
User	email	UNIQUE CONSTRAINT	Запобігає реєстрації дублікатів облікових записів.
FantasyTeam	userId	FOREIGN KEY	ON DELETE CASCADE (автоматичне видалення команди при видаленні профілю користувача).
PlayerStat	playerId	FOREIGN KEY	SET NULL (збереження історичної статистики матчів навіть після завершення кар'єри гравця).
Match	homeTeamId, awayTeamId	FOREIGN KEY	Забезпечує строгу цілісність прив'язки учасників до існуючих команд у БД.
Match	startTime, status	B-TREE INDEX	Оптимізація запитів для Live-трекера: прискорення пошуку поточних (LIVE) та запланованих подій.
MatchResult	matchId	UNIQUE CONSTRAINT	Забезпечує відношення "один-до-одного" (1:1) між матчем та його фінальним результатом.
MatchEvent	matchId	FOREIGN KEY	ON DELETE CASCADE (автоматичне очищення подій таймлайну при видаленні матчу).
Standing	teamId, seasonId	COMPOSITE UNIQUE	Обмеження рівня БД: команда може мати лише один запис турнірного становища в конкретному турнірі.
FavoriteMatch	userId, matchId	COMPOSITE PK	Складений первинний ключ таблиці асоціацій, що блокує можливість повторного додавання матчу в "Обране".
FantasyPick	fantasyTeamId, playerId	COMPOSITE UNIQUE	Фізична заборона додавання одного й того ж гравця двічі до складу однієї команди у Fantasy-модулі.
Player	teamId, position	COMPOSITE INDEX	Прискорення складних вибірок ростеру команди за конкретним ігровим амплуа (позицією).

Для глибокої деталізації зв'язків між таблицями та опису архітектури збереження даних було розроблено загальну концептуальну схему. Вона наочно відображає ієрархічну структуру платформи, де центральне місце займає логічна взаємодія між профілем спортивного аналітика та його улюбленими командами або персоналізованими звітами. Процес побудови таких логічних зв'язків повністю базується на фундаментальних принципах реляційної цілісності та

мінімізації інформаційної надлишковості, що у підсумку гарантує стабільну роботу серверної частини при обробці колосальних обсягів турнірної статистики.

Фундаментальні зв'язки між сутностями інформаційної системи спроектовано з дотриманням суворих реляційних стандартів. Наприклад, між обліковими записами користувачів та їхніми фентезі-командами діє відношення один-до-багатьох. Такий підхід надає аналітикам можливість генерувати будь-яку кількість персоналізованих складів. Важливим інженерним рішенням є впровадження механізму каскадного видалення. При знищенні профілю система автоматично стирає всі пов'язані аналітичні конфігурації та турнірні досягнення. Це повністю задовольняє сучасні вимоги до політики конфіденційності та безпеки персональних даних.

Сутності спортивних ліг та турнірів аналогічно пов'язуються з конкретними матчами через залежність один-до-багатьох. Для забезпечення гнучкості при класифікації статистичних матеріалів впроваджено комплексні відношення між спортсменами та їхніми клубами. Завдяки такій структурі користувачі отримують зручний інструмент для фільтрації контенту залежно від спортивної дисципліни за міжнародними стандартами [7] або відповідно до специфічних ігрових амплуа на полі.

Центральне місце в архітектурі бази даних відведено модулю деталізованої ігрової статистики. Цей компонент об'єднує підсумки матчів із показниками конкретних гравців за допомогою спеціальної агрегованої таблиці. Зазначене рішення дає змогу надійно зберігати вичерпну історію виступів кожного спортсмена, фіксувати результативні дії та розраховувати очікувану ефективність на базі передових алгоритмів обчислення метрик. Детальну специфікацію сутності для збереження облікових записів та параметрів доступу користувачів [17] наведено у таблиці 2.6.

Для забезпечення максимальної швидкості доступу до описаних взаємопов'язаних сутностей під час активних спортивних трансляцій структуру бази даних доповнено ефективними стратегіями індексування. Застосування

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

оптимізованих індексів для зовнішніх ключів та полів пошуку значно прискорює виконання складних запитів до агрегованих таблиць статистики.

Таблиця 2.6

Структура таблиці користувачів

Назва поля	Тип даних	Обмеження	Опис
id	VARCHAR(255)	PK	Унікальний текстовий ідентифікатор користувача платформи.
name	VARCHAR(255)	NULL	Публічне ім'я аналітика в системі.
email	VARCHAR(255)	UNIQUE, NOT NULL	Електронна пошта для безпечної авторизації та сповіщень.
password	VARCHAR(255)	NULL	Хешований пароль який може бути порожнім при вході через соціальні мережі.
role	ENUM	NOT NULL	Базова роль доступу у системі.
tier	ENUM	NOT NULL	Поточний рівень підписки на розширену аналітику.

Центральною сутністю підсистеми персоналізації є таблиця фентезі-команд. Вона містить загальні метадані збереженого складу, параметри віртуального бюджету та конфігураційні ключі обраної тактичної побудови. Фізична структура поданої нижче таблиці 2.7 забезпечує гнучке керування збереженими складами та дозволяє інтегрувати їх у загальну аналітичну екосистему розроблюваної спортивної платформи.

Таблиця 2.7

Структура таблиці фентезі-команд

Назва поля	Тип даних	Обмеження	Опис
id	VARCHAR(255)	PK	Унікальний ідентифікатор створеної фентезі-команди.
userid	VARCHAR(255)	FK, NOT NULL	Пряме посилання на власника ігрового складу.
name	VARCHAR(255)	NOT NULL	Персоналізована назва віртуальної команди.
budget	INT	NOT NULL	Залишок віртуального бюджету для трансферів гравців.
formation	VARCHAR(50)	NOT NULL	Обрана тактична схема розстановки гравців на полі.
points	INT	NOT NULL, DEFAULT 0	Загальна кількість зароблених командою рейтингових балів.
created_at	TIMESTAMP	DEFAULT NOW	Час генерації та збереження поточного складу команди.

Для зберігання безпосереднього наповнення фентезі-команди, яке аналітик формує через інтерфейс віртуального драфту, використовується таблиця вибору гравців. Кожен запис у цій таблиці представляє конкретного спортсмена із закріпленою за ним позицією та ігровим статусом у межах створеного складу.

Таблиця 2.8

Структура таблиці вибору гравців

Назва поля	Тип даних	Обмеження	Опис
id	VARCHAR(255)	PK	Унікальний ідентифікатор вибору гравця.
fantasyTeamId	VARCHAR(255)	FK, NOT NULL	Посилання на команду до якої входить гравець.
playerId	VARCHAR(255)	FK, NOT NULL	Посилання на фізичного спортсмена з глобальної бази.
positionIndex	INT	NOT NULL	Індекс тактичної позиції на ігровому полі.
isCaptain	BOOLEAN	DEFAULT FALSE	Параметр що визначає статус капітана з подвійними балами.

Для забезпечення безпеки, аудиту змін у турнірних таблицях та контролю за діями модераторів новин реалізовано підсистему журналювання системних подій. Фізична структура таблиці логів дозволяє відстежувати історію модифікації критичних спортивних даних, що важливо для відновлення станів системи у разі несанкціонованого втручання.

Таблиця 2.9

Структура таблиці match_events

Назва поля	Тип даних	Обмеження	Опис
id	VARCHAR(255)	PK	Унікальний ідентифікатор ігрової події. matchId
matchId	VARCHAR(255)	FK, NOT NULL	Посилання на конкретний матч під час якого відбулася подія.
type	ENUM	NOT NULL	Тип події на зразок гола червоної картки або заміни гравця.
minute	INT	NULL	Хвилина основного часу на якій відбулася подія.
playerId	VARCHAR(255)	NULL	Ідентифікатор головного гравця який виконав дію.
detail	VARCHAR(255)	NULL	Додаткові текстові деталі події або коментар.
created_at	TIMESTAMP	DEFAULT NOW	Точний час фіксації події у базі даних сервера.

Фундаментальним архітектурним рішенням платформи виступає метод збереження глибокої статистики за допомогою текстових схем. Замість генерування надмірної кількості таблиць під кожен новий вид спорту або специфічний формат метрик, для фіксації результатів матчів застосовується нативний формат JSON. Такий підхід гарантує гнучкість, притаманну документо-орієнтованим базам даних, водночас зберігаючи всі переваги суворої реляційної цілісності основного сховища [13]. Завдяки цьому розробники отримують змогу безперешкодно впроваджувати нові аналітичні алгоритми, зокрема машинне обчислення показників очікуваних голів [8], без виконання складних міграцій чи зміни схеми бази даних. Це суттєво пришвидшує загальний цикл розробки та розгортання оновлень вебзастосунку.

Масиви статистичної інформації зберігаються в системі у вигляді впорядкованої послідовності об'єктів. Кожен такий елемент безпосередньо відповідає за певний функціональний блок або аналітичний віджет у браузерному інтерфейсі. Подібна концепція забезпечує максимальний рівень гнучкості під час розширення платформи. Додавання нових різновидів графіків або порівняльних таблиць більше не вимагає втручання в серверну реляційну структуру. Натомість для розширення функціонала достатньо лише оновити логіку обробки цих об'єктів безпосередньо на стороні клієнтського додатка.

Кожен стандартний вузол у межах статистичної схеми характеризується чітко визначеною структурою та містить обов'язковий набір параметрів. Наявність унікального ідентифікатора блоку гарантує коректне функціонування алгоритмів рендерингу та дає змогу динамічно управляти станом візуальних компонентів на екрані. Спеціальний параметр типу діє як константа і вказує фабриці рендерингу на те, який саме віджет необхідно завантажити. Це може бути турнірна таблиця, графік ігрової переваги чи теплова карта активності. Вкладені об'єкти своєю чергою інкапсулюють масиви спортивного контенту та складні конфігурації для візуалізації.

Застосована архітектурна модель забезпечує безпрецедентно високу швидкодію інформаційної системи. Під час відкриття сторінки живого поєдинку

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

серверна частина виконує лише один атомарний запит до бази даних, витягує повну конфігурацію ігрових подій і миттєво транслює її до клієнтського браузера [25]. Фронтенд на базі сучасної бібліотеки React здійснює візуалізацію складної структури інтерфейсу [1], паралельно ініціюючи асинхронні запити для завантаження важких медіафайлів на кшталт клубних логотипів із хмарного сервісу оптимізації [18]. Завдяки цьому досягається абсолютна безперервність користувацького досвіду без жодних відчутних мережових затримок чи тривалих очікувань.

2.3 Проєктування компонентної структури та взаємодії модулів

Діаграма розгортання призначена для моделювання фізичної та інфраструктурної топології аналітичної системи, наочно відображаючи конфігурацію серверних потужностей та хмарних сервісів. Вона дозволяє оцінити розподіл навантаження між окремими компонентами платформи та деталізувати протоколи їхньої взаємодії [9]. Оскільки вебзастосунок проєктується як сучасне хмарне рішення для моніторингу живої статистики, його архітектура повністю базується на принципах розподіленої інфраструктури. Це забезпечує максимальну відмовостійкість, миттєве горизонтальне масштабування та надшвидку доставку важких аналітичних дашбордів до кінцевого користувача [10].

На представлений візуалізації розгортання чітко виділено ключові інфраструктурні вузли платформи. Клієнтський рівень, представлений браузерами звичайних вболівальників та панеллю спортивних модераторів, взаємодіє з бекендом виключно через захищений мережовий шлюз [20]. Основна обробка бізнес-логіки відбувається на масштабованій серверній платформі [19], тоді як збереження структурованих спортивних даних повністю делеговано керованому кластеру реляційної бази. Окремі зовнішні рішення, такі як сервіси оптимізації мультимедійних файлів [18] та офіційні постачальники статистичних показників, інтегровані в єдину захищену мережу, що оптимізує маршрутизацію запитів та гарантує стабільну роботу порталу навіть під час пікових навантажень у дні фіналів великих турнірів.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

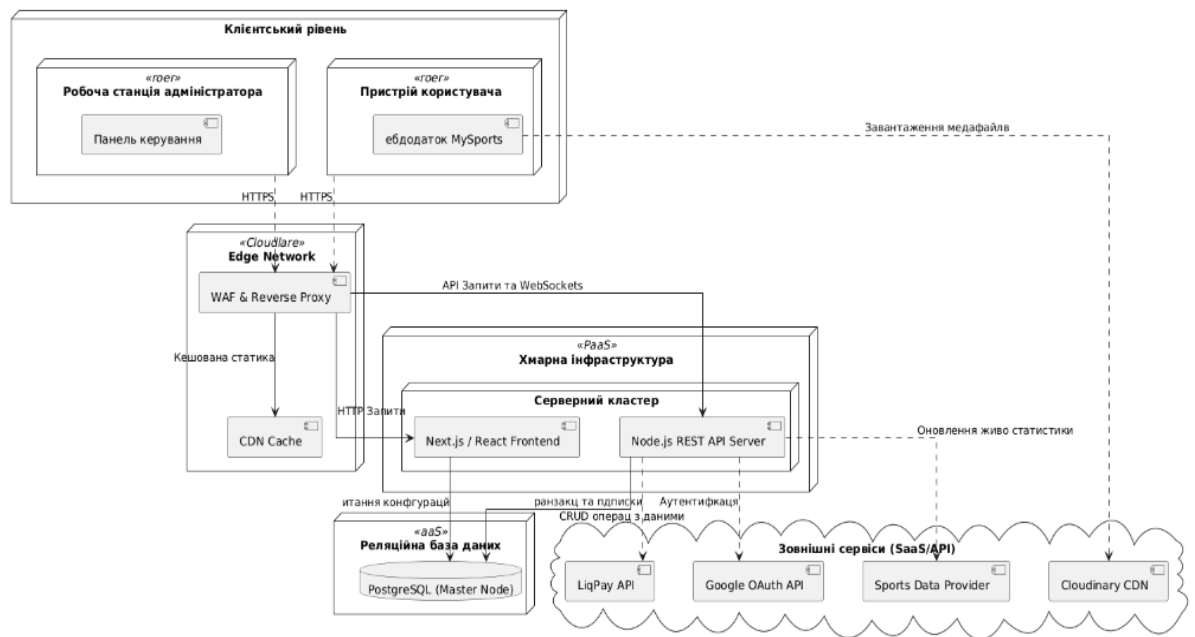


Рис. 2.3 – Діаграма розгортання системи

Архітектурна модель розроблюваної платформи базується на сучасній хмарній парадигмі, яка гарантує стабільне функціонування вебклієнта та надійну двосторонню синхронізацію інформації. Головним вузлом взаємодії виступає пристрій кінцевого користувача, де у середовищі браузера виконується реактивний вебзастосунок, створений на базі фреймворку Next.js та екосистеми React [1]. Зазначений клієнтський компонент відповідає за плавне відображення інтерактивних графіків, керування локальним станом обраних команд завдяки технологіям кешування [14] та забезпечення миттєвого відгуку інтерфейсу на зміну рахунку під час матчів. Окремим функціональним вузлом на боці клієнта є захищена адміністративна панель, яка слугує зручним інструментом для модерації новинного контенту та управління преміальними підписками вболівальників.

Забезпечення глобальної безпеки та інтелектуальна маршрутизація мережевого трафіку реалізуються через розподілену інфраструктуру Cloudflare [20]. Вона виконує роль надійного захисного бар'єра між браузерами відвідувачів та серверним бекендом. Цей компонент працює як потужний зворотний проксі-сервер, що автоматично відсікає шкідливі запити від ботнетів та гарантує суворе шифрування інформації за допомогою передових криптографічних протоколів. Завдяки такій конфігурації легітимні мережеві звернення безпечно

спрямовуються до відповідних хмарних ресурсів, що забезпечує високий рівень доступності аналітичної системи та захищає базу даних від скоординованих кібератак.

Серверна частина розгорнута на спеціалізованій хмарній платформі [19], де у високопродуктивному середовищі Node.js [4] безперервно виконується основна бізнес-логіка інформаційної системи. Цей вузол здійснює миттєву обробку запитів від клієнтських дашбордів [2], відповідає за процеси авторизації користувачів через зовнішні сервіси [31] та керує життєвим циклом підписок за допомогою вітчизняного платіжного шлюзу LiqPay [30]. Також сервер забезпечує надійну транзакційну взаємодію з глобальним сховищем даних. Використана архітектура наділяє бекенд здатністю миттєво масштабувати обчислювальні потужності у разі різкого стрибка активності аудиторії під час трансляцій топових спортивних протистоянь.

Збереження структурованої статистики та календарів турнірів організовано на базі керованого кластера реляційної системи PostgreSQL [29]. Зазначений вузол функціонує в ізольованому віртуальному середовищі, що повністю виключає можливість прямого доступу ззовні та гарантує абсолютну цілісність критично важливого аналітичного матеріалу. Робота з мультимедійними активами, куди входять високоякісні зображення спортсменів та векторні логотипи клубів, здійснюється через інтегровану мережу доставки контенту [18]. Цей сервіс автоматично оптимізує графіку відповідно до розміру екрана конкретного пристрою, що відіграє критичну роль у збереженні мережевого трафіку та швидкому рендерингу складних візуальних елементів платформи.

Будь-який сеанс роботи користувача з аналітичним порталом бере свій початок на клієнтському рівні у момент ініціації первинних мережевих запитів. З метою гарантування максимального рівня захисту від несанкціонованого втручання весь вхідний потік даних автоматично спрямовується на фільтрувальні вузли Cloudflare. Виключно після успішного проходження перевірки на легітимність запити безпечно маршрутизуються до основного хмарного бекенду [19]. Безпосередньо клієнтський інтерфейс побудовано з використанням

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

сучасного компонентного підходу бібліотеки React [1], що дає змогу миттєво перемальовувати динамічні турнірні таблиці без потреби в повному перезавантаженні вебсторінки.

Подолавши захисний мережевий бар'єр, запити потрапляють на обробку до високопродуктивного серверного ядра на базі Node.js [4]. Програмний бекенд аналізує отриману інформацію та делегує її до відповідного модуля бізнес-логіки. Під час формування розкладу поєдинків або завантаження поточного стану фентезі-команди система генерує складні транзакції до реляційного сховища PostgreSQL із застосуванням сучасних інструментів об'єктно-реляційного відображення. Отримані результати вибірок проходять процедуру автоматичної серіалізації у стандартизований текстовий формат JSON та повертаються назад до фронтенду [2].

Дистрибуція графічного контенту програмного комплексу функціонує за окремим оптимізованим алгоритмом. Зважаючи на те, що високоякісні фотографії гравців та емблеми спортивних організацій створюють значне навантаження на канал передачі даних, завдання щодо їхньої доставки повністю покладено на глобальну платформу Cloudinary [18]. Завдяки цьому клієнтські браузери завантажують важкий статичний контент із найближчих кеш-серверів, уникаючи генерації зайвих звернень до основного сервера [9]. У цей час бекенд залишається сфокусованим виключно на виконанні критично важливих фонових процесів, серед яких безперервний парсинг сторонніх спортивних джерел та опрацювання фінансових транзакцій від платіжних систем рівня LiqPay.

Описаний архітектурний поділ відповідальності наділяє систему максимальною стійкістю в умовах пікових навантажень у дні проведення масштабних турнірів. Програмна реалізація серверної інфраструктури базується на передових патернах об'єктно-орієнтованого проєктування, що гарантує низький рівень зв'язності між ізольованими компонентами [10]. Доменна модель застосунку абстрагується від фізичної структури таблиць бази даних та оперує виключно концептуальними сутностями спортивної предметної області. Центральними об'єктами цієї моделі виступають сутності ліги, матчу, команди та

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

гравця, навколо яких концентрується вся подальша логіка збору статистики та моніторингу подій. Для кращого сприйняття механізмів взаємодії цих елементів було розроблено концептуальну діаграму класів предметної області. Ієрархію ключових сутностей розроблюваної аналітичної платформи наочно візуалізовано за допомогою відповідної схеми (рисунок 2.4) [27].

Абстрагування ключової бізнес-логіки від фізичного рівня зберігання даних наділяє команду здатністю безперешкодно розширювати функціонал, інтегрувати нові спортивні дисципліни та впроваджувати складні аналітичні алгоритми без ризику порушення цілісності існуючого коду. Такий інженерний фундамент перетворює розроблювану вебплатформу на потужний і легко масштабований аналітичний центр, який гарантує безперебійне постачання живої статистики та преміальний користувацький досвід незалежно від екстремальних навантажень під час світових спортивних першостей.

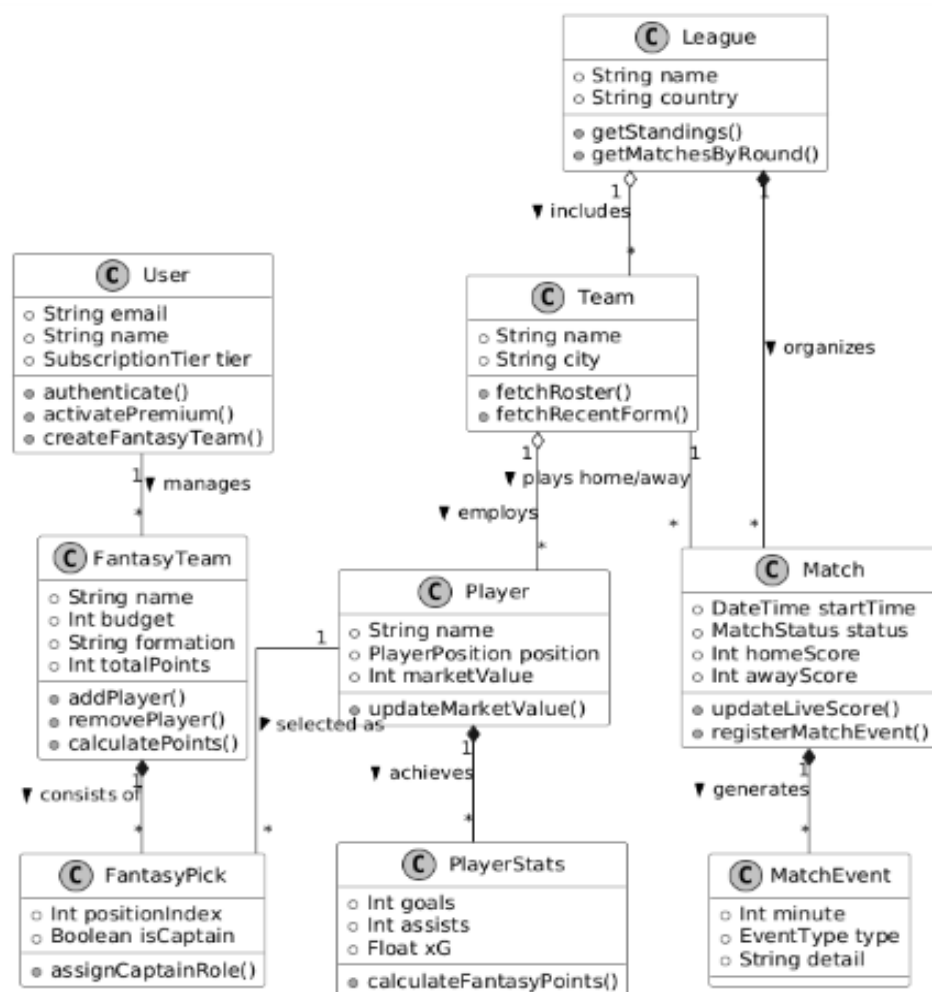


Рис. 2.4 – Діаграма класів предметної області

Фундаментальним елементом предметної області виступає клас профілю користувача. Зазначена сутність перебуває у відношенні композиції з класами фентезі-команд та списками обраних спортивних подій. Зважаючи на те, що ці об'єкти генеруються безпосередньо аналітиком і функціонують виключно в рамках його облікового запису, логічне видалення профілю автоматично запускає процес каскадного очищення всіх пов'язаних даних із глобального сховища платформи.

Зі свого боку клас фентезі-команди відіграє роль ключового агрегатора статистичного контенту та утворює суворий композиційний зв'язок із класами вибору окремих спортсменів. Критично важливими атрибутами подібних вибірок виступають конфігураційні поля у гнучкому форматі JSON [13]. Саме у цих структурах безпосередньо фіксується логіка тактичної розстановки та надійно зберігаються персональні параметри налаштування ігрового складу.

З метою гарантування високого рівня масштабованості програмного коду в умовах хмарної архітектури та упередження небажаного дублювання функцій розробка системи суворо підпорядковується застосуванню перевірених інженерних патернів [10]. Спроектowana концептуальна діаграма класів реалізації наочно демонструє маршрути проходження потоків інформації від браузерного вебзастосунку до реляційної бази даних. Ця модель відображає строгий розподіл зон відповідальності між усіма структурними компонентами системи моніторингу спортивних подій. Деталізовану схему описаної архітектурної взаємодії візуалізовано на (рисунку 2.4).

Для візуалізації динаміки обробки складних транзакційних запитів було розроблено UML діаграму послідовності (Sequence Diagram). Життєвий цикл типового запиту (наприклад, збереження складу фентезі-команди) починається з відправки Payload-даних клієнтом. Далі API-маршрут (Next.js App Router) ініціює обов'язкову валідацію сесії через метод `getSession`. У разі успішного підтвердження прав доступу, сервісний шар перевіряє бізнес-правила та звертається до СКБД PostgreSQL через ORM Prisma для виконання запиту (рисунок 2.5).

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

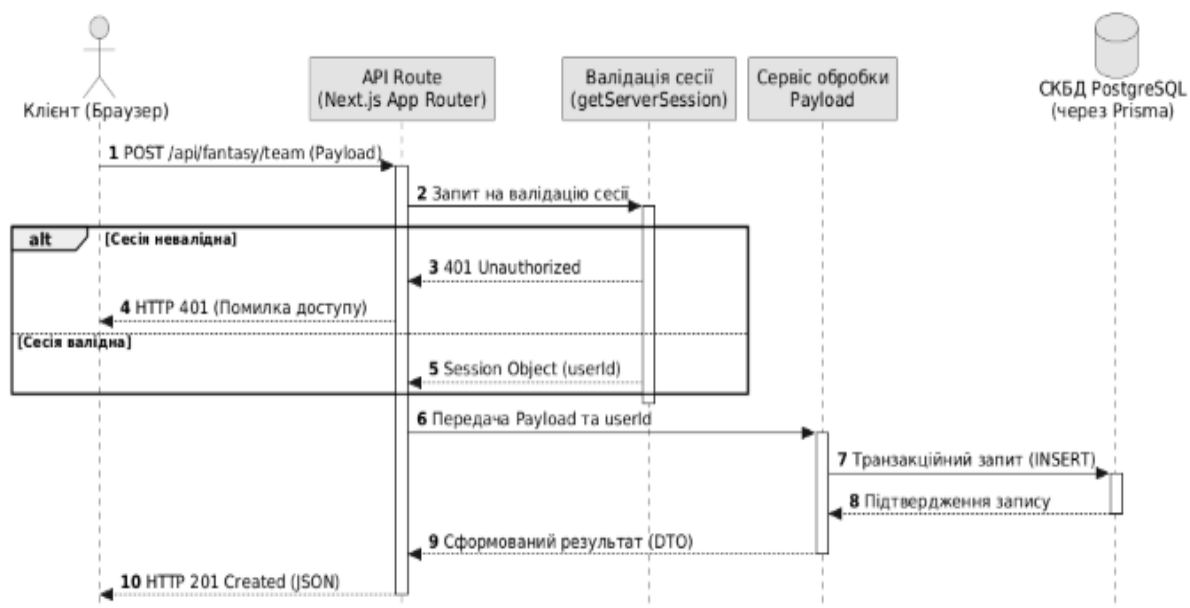


Рисунок 2.5 – Діаграма послідовності обробки транзакційного запиту

Фізичну топологію розгортання платформи відображає UML діаграма розгортання (Deployment Diagram). Програмний комплекс функціонує у трьох ізольованих середовищах: клієнтському браузері, хмарному середовищі виконання Node.js (Vercel), де обробляються API-маршрути, та на віддаленому хостингу бази даних Supabase. Зв'язок між бекендом та СКБД підтримується через постійний пул з'єднань (рисунок 2.6).

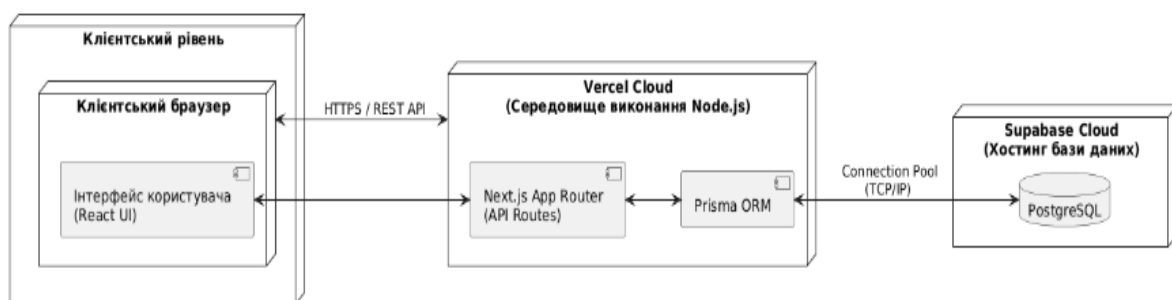


Рисунок 2.6 – Діаграма розгортання програмного комплексу

З метою дотримання принципів SOLID та забезпечення гнучкості програмного коду, архітектуру серверної частини розроблено з використанням класичних патернів об'єктно-орієнтованого проєктування. Деталізовану ієрархію об'єктів, інтерфейсів та їхніх взаємозв'язків відображає діаграма класів реалізації (рисунок 2.7). Зокрема, для доступу до бази даних застосовано патерн «Одинак» (Singleton), який гарантує наявність єдиного екземпляра підключення Prisma Client, запобігаючи вичерпанню пулу з'єднань. Бізнес-логіка повністю

ізолювана в межах сервісного шару (Service Layer), що взаємодіє з контролерами маршрутів виключно через стандартизовані об'єкти передачі даних (DTO). Крім того, для забезпечення реактивного оновлення турнірних таблиць на клієнті імплементовано патерн «Спостерігач» (Observer).

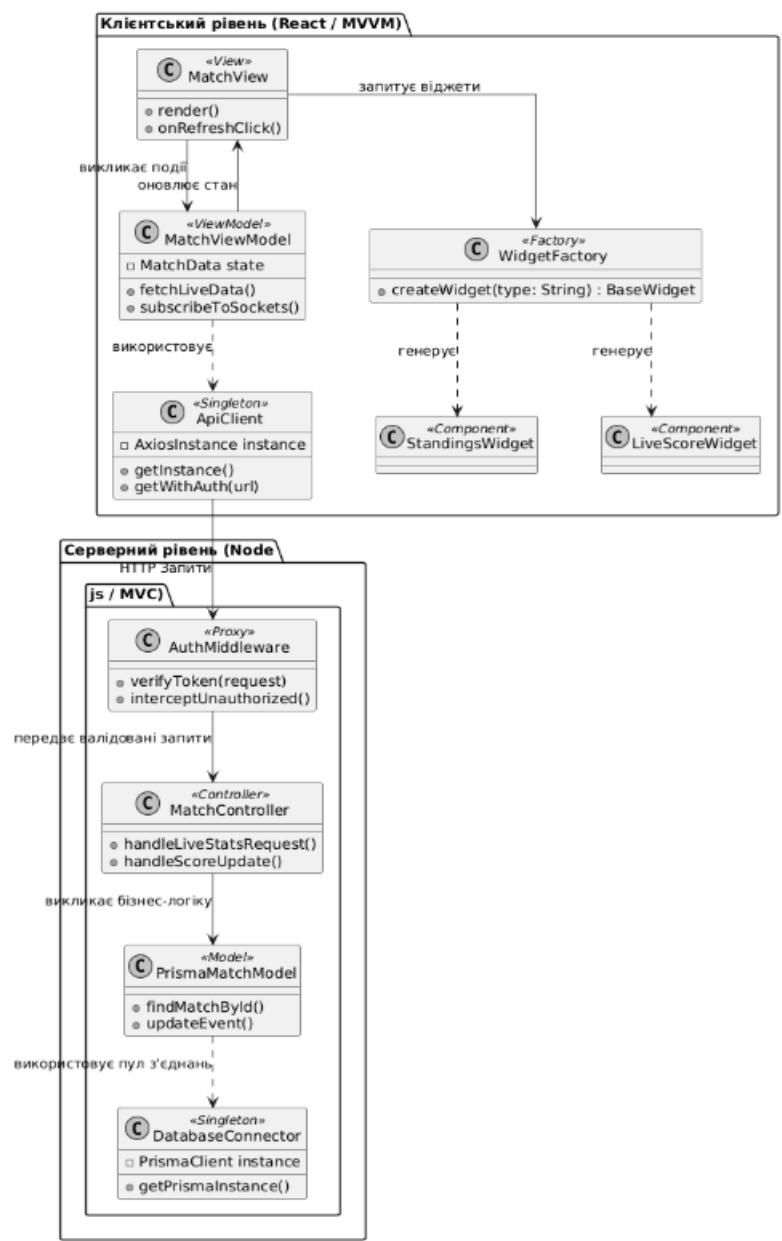


Рис. 2.7 – Діаграма класів реалізації з використанням патернів

В основу проєктування інформаційної системи закладено комплекс перевірених архітектурних патернів, які гарантують ефективний розподіл обчислювальної логіки та високу масштабованість вебдодатка. На серверному рівні імплементовано класичний патерн MVC, у якому контролер маршрутизації

повністю бере на себе управління обробкою запитів, тоді як рівень моделі забезпечує надійну взаємодію з реляційною базою даних за допомогою інструментарію Prisma ORM. Натомість клієнтський бік функціонує за принципами архітектури MVVM [11]. У цій парадигмі безпосереднє відображення інтерфейсу покладено на функціональні компоненти бібліотеки React [1], а логіка управління даними ізольована у відокремлену модель представлення із залученням сучасних менеджерів стану [32]. Впровадження такого підходу забезпечує максимально швидке та реактивне оновлення візуальних елементів під час зміни рахунку безпосередньо в ході матчу.

Для гнучкого формування різноманітних аналітичних віджетів на фронтенді успішно інтегровано патерн Фабрика. Його реалізація відбувається через спеціалізований клас рендерингу компонентів, що дає системі змогу автоматично створювати відповідний графічний блок з огляду на специфіку отриманих даних. Це може бути кругова діаграма володіння м'ячем, теплова карта активності спортсмена чи стандартна турнірна таблиця. Таке інженерне рішення допомагає уникнути жорсткого та неефективного програмування кожного окремого елемента в загальному макеті дашборда.

Управління глобальними сервісами застосунку, які потребують існування виключно в єдиному екземплярі в оперативній пам'яті, базується на патерні Одинак. На боці вебклієнта цю роль виконує спеціально налаштований мережевий клієнт із вбудованими перехоплювачами токенів авторизації [17]. У серверному середовищі цю критично важливу функцію бере на себе клас конектора бази даних. Він ефективно управляє пулом з'єднань до сховища PostgreSQL [4], що надійно запобігає вичерпанню системних ресурсів під час одночасної обробки тисяч користувацьких запитів.

З метою суворого обмеження доступу та перевірки адміністративних прав архітектура платформи використовує патерн Замісник. На клієнтському рівні ця концепція втілюється за допомогою компонентів вищого порядку, які захищають приватні сторінки від несанкціонованого перегляду. На бекенді цей патерн працює як проміжний шар перевірки всередині серверного середовища Node.js

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

[25], гарантуючи надійну валідацію криптографічних токенів перед тим, як надати користувачеві доступ до конфіденційних методів управління системою.

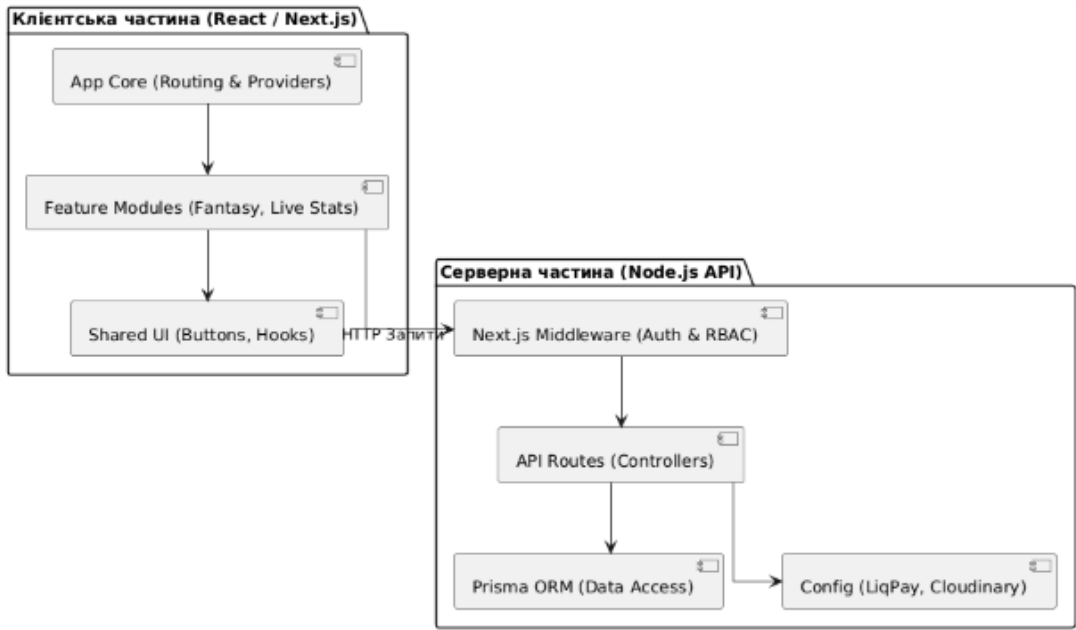


Рис. 2.8 – Діаграма компонентів програмного комплексу

Програмний бекенд вебзастосунку базується на середовищі Node.js [4] та втілює концепцію модульного моноліта. Організація коду спирається на жорсткий розподіл за функціональними рівнями. Спеціалізовані маршрутизатори декларують кінцеві точки прикладного програмного інтерфейсу [2] та спрямовують увесь вхідний мережевий трафік до відповідних контролерів. Глобальний рівень безпеки забезпечується проміжним програмним шаром. Цей компонент діє як надійний бар'єр для перевірки криптографічних токенів та валідації адміністративних прав відповідно до рольової моделі. Усі зовнішні інтеграції винесено в ізольовані конфігураційні модулі для зручності управління. Це повною мірою стосується налаштувань сервісу обробки мультимедіа Cloudinary [18] та конфігурації вітчизняного платіжного шлюзу LiqPay [30].

Клієнтський рівень платформи розроблено мовою TypeScript [5] із залученням потужного інструментарію бібліотеки React [1]. Проєкт спирається на принципи структурної декомпозиції для формування чіткої модульної ієрархії та максимального полегшення подальшої підтримки кодової бази. У рамках цієї концепції весь клієнтський комплекс розділено на три базові логічні рівні. Кожен

із них повністю ізольовано відповідає за виконання власного технічного завдання. Візуалізацію цієї багаторівневої структури у форматі діаграми компонентів програмного комплексу представлено на (рисунку 2.7).

Перший рівень формує глобальне ядро застосунку. Воно інкапсулює базові провайдери стану системи разом із механізмами перевірки сесій та маршрутизації сторінок. Наступний рівень складається з відокремлених предметних модулів для реалізації головної бізнес-логіки платформи. Найскладнішим інженерним компонентом серед них виступає рушій фентезі-драфту. Він бере на себе відповідальність за математичний розрахунок доступного ігрового бюджету, обчислення здобутих балів та динамічний рендеринг тактичної схеми з обраним складом команди. Супутні модулі цього рівня підтримують процеси адміністрування новин та інтеграцію із зовнішніми провайдерами авторизації на зразок Google OAuth [31]. Завершальним елементом виступає спільний робочий простір. Він об'єднує універсальні компоненти інтерфейсу та кастомні хуки для надійної синхронізації живих рахунків через швидкісні вебсокети.

Застосований архітектурний підхід дає розробникам можливість незалежно тестувати та вдосконалювати окремі складові спортивної платформи без загрози порушення роботи суміжних сервісів. Для глибокої деталізації процесів обміну даними між фронтендом та хмарним бекендом було розроблено діаграми взаємодії. Вони наочно демонструють життєвий цикл проходження мережевих запитів, механізми передачі управління між контролерами та безпосередні транзакції до бази даних [13].

Діаграми послідовності відображають суворий хронологічний порядок виклику мережевих методів під час виконання критично важливих сценаріїв використання. Процес створення нової фентезі-команди проілюстровано відповідною діаграмою послідовності (рисунок 2.9). Цей процес виступає одним із найважливіших елементів інтерактивної взаємодії користувача зі спортивним порталом.

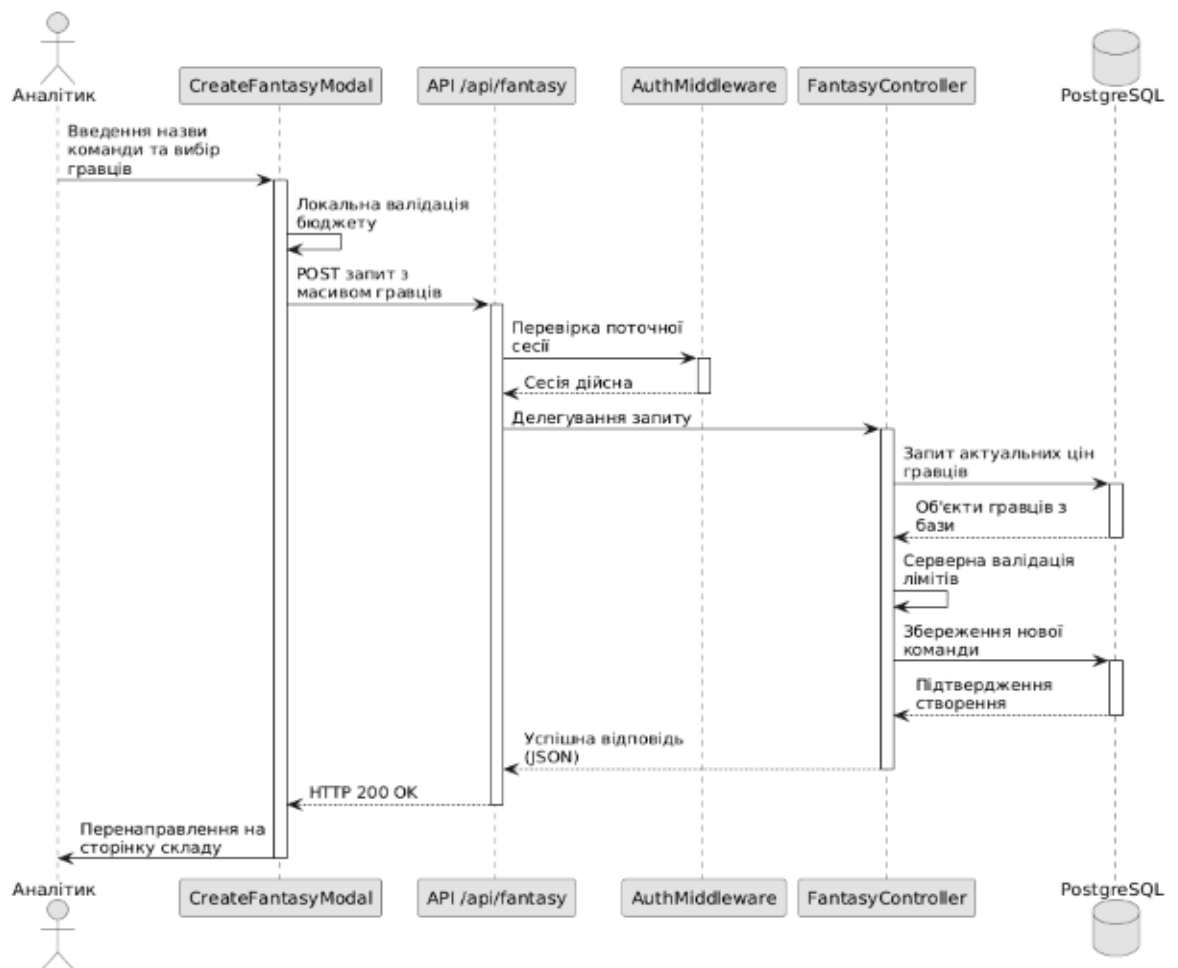


Рис. 2.9 – Діаграма послідовності процесу ініціалізації команди

Процес формування віртуального складу відбувається у суворо визначеній послідовності. Взаємодія розпочинається на боці веббраузера, де користувач через модальне вікно вводить базові параметри, зокрема назву колективу та бажану тактичну схему. Одразу після цього клієнтський компонент здійснює миттєву локальну валідацію. Вона необхідна для контролю того, щоб сумарна вартість обраних спортсменів не перевищувала встановлений бюджетний ліміт [14].

Після підтвердження дії клієнтський додаток генерує асинхронний запит до серверного маршруту та передає конфігураційні дані у вигляді текстового пакета. Цей мережевий запит спочатку проходить через проміжний безпековий шар бекенд-фреймворку [25] для перевірки валідності сесії. Далі він перенаправляється безпосередньо до контролера управління фентезі-лігами. На даному етапі сервер самостійно звертається до таблиці гравців для звірки їхніх

актуальних ринкових вартостей, що повністю унеможлиблює будь-які маніпуляції з ціноутворенням зі сторони клієнта.

За умови успішного проходження всіх лімітних перевірок серверний контролер ініціює створення нового об'єкта. Для цього він передає до бази даних ідентифікатор користувача разом із масивом обраних гравців. Після надійного збереження транзакції у кластері PostgreSQL [29] сервер формує оптимізовану відповідь. Браузерний клієнт перехоплює статус успішного завершення операції, автоматично оновлює локальний кеш та перенаправляє користувача на сторінку новоствореної команди для подальшого керування складом.

Процес створення звернення до контент-менеджерів через клієнтський інтерфейс проілюстровано відповідною діаграмою послідовності (рисунок 2.10). Цей механізм відіграє критичну роль у підтримці стовідсоткової точності турнірної статистики. Він надає користувачам-аналітикам зручний інструментарій для оперативного інформування про можливі неточності у протоколах матчів, помилки у стартових складах команди або для подання пропозицій щодо коригування розрахованих статистичних метрик.

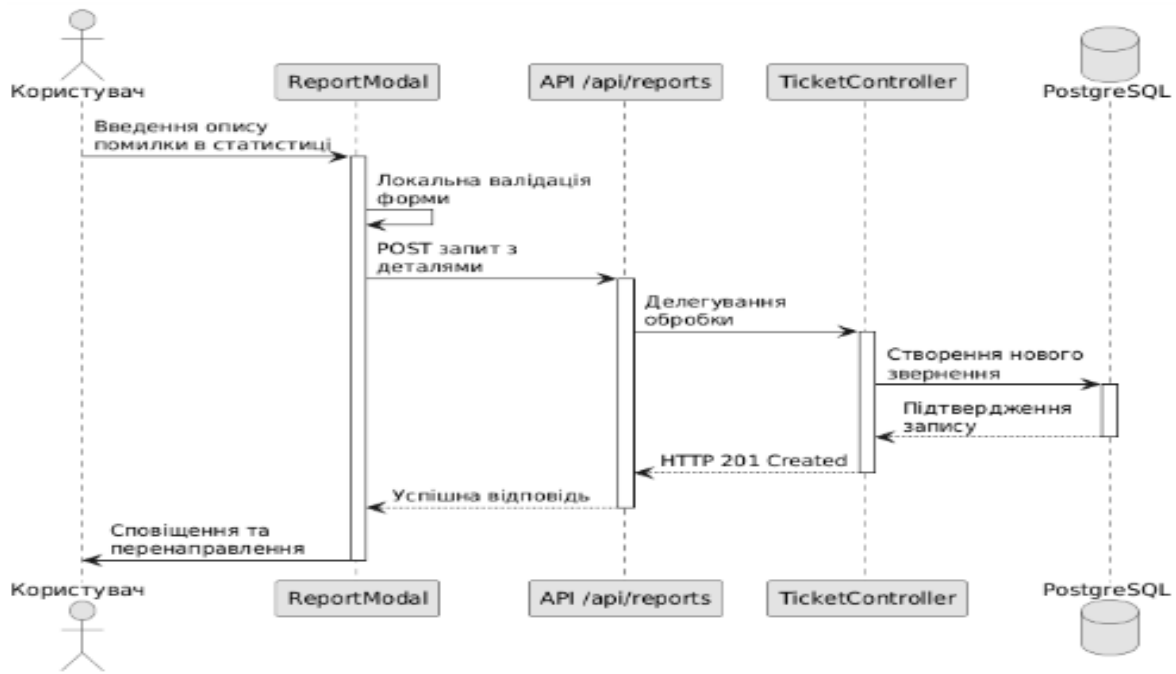


Рис. 2.10 – Діаграма послідовності процесу створення звернення

Логіка обміну даними при формуванні звернення передбачає кілька послідовних етапів. Користувач вводить деталі виявленої помилки у

відповідному розділі порталу, після чого інтерфейс здійснює локальну перевірку полів на наявність обов'язкових посилань на конкретний матч. Користувач підтверджує відправку форми, що ініціює мережевий запит до контролера служби підтримки. Контролер передає отримані дані до ORM-моделі бази даних, яка виконує транзакційний запис у таблицю звернень кластера PostgreSQL [4]. Після успішного завершення операції сервер повертає статус успішного створення, а вебзастосунок повідомляє користувача та перенаправляє його до переліку активних запитів для відстеження статусу модерації.

Складну бізнес-транзакцію процесу модерації та призупинення публічного доступу до новинної публікації чи користувацького дашборда адміністратором платформи відображено на діаграмі (рисунок 2.11).Цей сценарій містить розгалуження та умовні виклики, що забезпечують надійний захист спортивної екосистеми від недостовірного або рекламного наповнення.

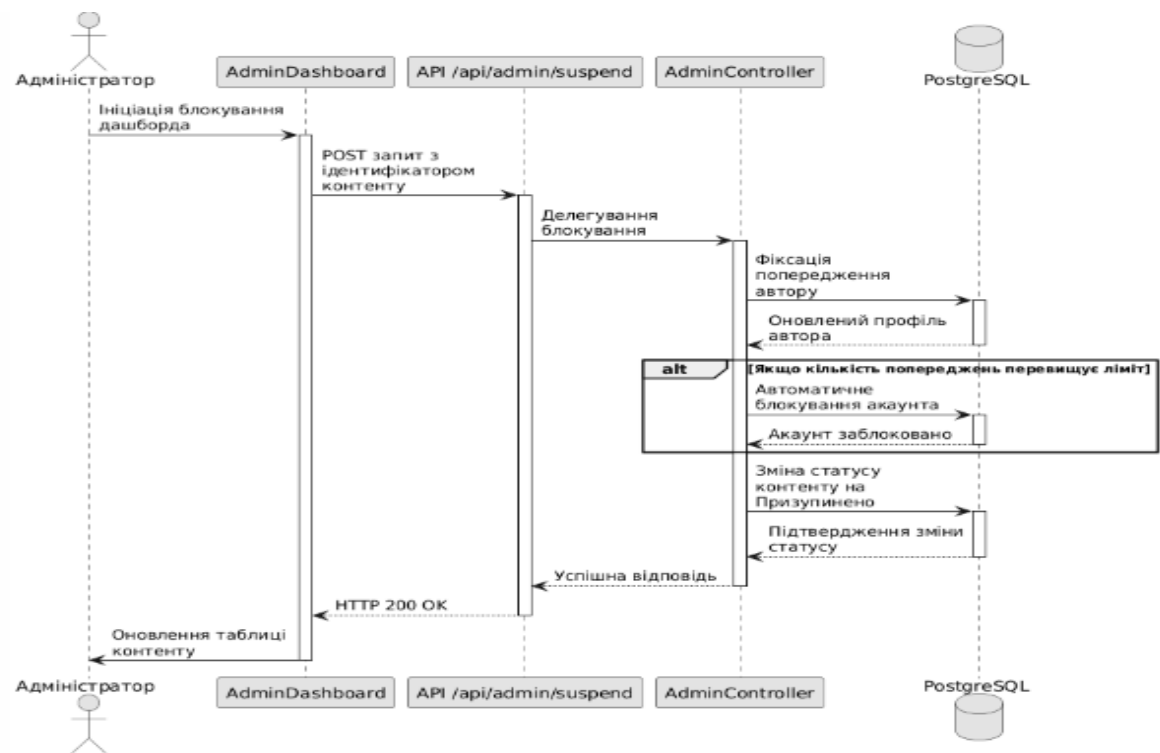


Рис. 2.11 – Діаграма послідовності призупинення аналітичного дашборда

Процедура контролю контенту розпочинається з ініціації блокування сторінки порушника Адміністратором через захищену панель керування. Інформаційна система генерує зашифрований мережевий запит до відповідного контролера. Цей модуль першочергово взаємодіє з моделлю користувачів задля

реєстрації офіційного попередження безпосередньо в профілі автора. Такий підхід дає змогу акумулювати детальну історію інцидентів для подальшого автоматичного прийняття рішень про перманентне блокування облікових записів. Це виступає дієвим інструментом запобігання поширенню недостовірних аналітичних матеріалів [17].

Після успішної реєстрації попередження серверний обробник виконує повторне звернення до бази даних для підрахунку загальної кількості активних порушень конкретного користувача. На даному етапі вступає в дію жорстка умовна логіка. У разі перевищення критичної межі скарг система в автономному режимі повністю обмежує доступ порушника до платформи. Впроваджений багаторівневий механізм надійно протидіє масовому розповсюдженню фейкової спортивної аналітики та систематичному ігноруванню встановлених правил вебпорталу [17].

Незалежно від факту перманентного блокування самого автора контролер в обов'язковому порядку ініціює паралельну команду для негайної зміни статусу проблемної публікації або аналітичного дашборда. Відповідний запис у кластері PostgreSQL [4] отримує маркування призупиненого. Після успішного проведення описаних транзакцій бекенд миттєво повертає клієнтському додатку підтвердження про виконання операції. На завершальній стадії адміністративна панель автоматично синхронізує локальний стан та візуально позначає виявлений матеріал як недоступний для перегляду.

З метою забезпечення глибокого та комплексного розуміння взаємодії об'єктів під час виконання модераційних транзакцій розроблено діаграму кооперації. Зазначена схема фактично відображає ідентичну логіку обміну мережевими повідомленнями. Проте її головний фокус зосереджено не на хронології подій, а безпосередньо на просторовій архітектурі та топології комунікацій між системними модулями застосунку (рисунк 2.12) [10].

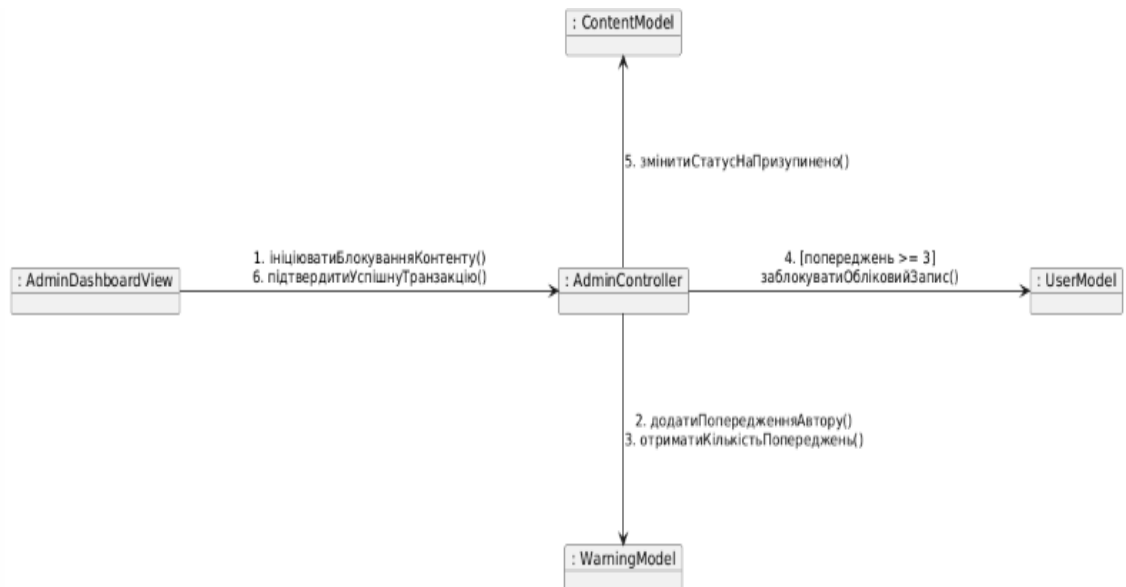


Рис. 2.12 – Діаграма кооперації процесу призупинення контенту

Запропонована комунікаційна діаграма демонструє чітку радіальну топологію управління, де центральним вузлом взаємодії виступає адміністративний контролер платформи. На практиці він виконує ключову функцію фасадного інтерфейсу, надійно ізолюючи клієнтський браузер від прямої взаємодії з шаром збереження даних. Така продумана архітектура дозволяє бекенду ефективно координувати спільну роботу одразу трьох незалежних моделей бази даних: системи страйків, профілів користувачів та агрегованих дашбордів. Впровадження цього патерну гарантує суворе дотримання фундаментального принципу єдиної відповідальності та підтримує бездоганну транзакційну цілісність під час модерації спортивного контенту.

Забезпечення конфіденційності, цілісності та доступності інформації є критичним аспектом проєктування сучасних вебсистем. Архітектуру безпеки розроблюваної платформи побудовано на основі багаторівневого захисту, де ключову роль відіграє підсистема автентифікації та авторизації.

Для реалізації механізму керування сесіями інтегровано спеціалізований фреймворк NextAuth.js. Замість класичного підходу зі збереженням ідентифікаторів сесій у базі даних (Session-based authentication), систему спроєктовано з використанням технології JSON Web Tokens (JWT). Такий stateless-підхід дозволяє серверу не зберігати стан авторизації локально, що

суттєво знижує навантаження на базу даних під час масового напливу користувачів (наприклад, під час онлайн-трансляцій популярних спортивних подій).

Процес автентифікації передбачає генерацію криптографічно підписаного токена, який містить зашифрований Payload із базовими ідентифікаторами користувача. Токен зберігається на клієнті у захищених HttpOnly файлах cookie, що автоматично унеможлиблює його перехоплення через вразливості типу Cross-Site Scripting (XSS).

Окрім внутрішньої безпеки сесій, архітектура передбачає захист від зовнішніх векторів атак. У таблиці 2.10 наведено специфікацію ідентифікованих загроз та спроектованих механізмів їхньої нейтралізації на рівні програмного коду.

Таблиця 2.10

Модель загроз інформаційної безпеки та архітектурні рішення для їх нейтралізації

Вектор атаки (Загроза)	Опис вразливості в контексті платформи	Архітектурний механізм захисту (Мітигація)
XSS (Cross-Site Scripting)	Ін'єкція шкідливих скриптів через поля вводу (наприклад, коментарі до новин або назви команд).	Автоматичне екранування даних на рівні віртуального DOM фреймворку React; валідація вводу через схеми Zod.
CSRF (Cross-Site Request Forgery)	Несанкціоноване виконання транзакцій від імені авторизованого аналітика.	Використання вбудованих Anti-CSRF токенів NextAuth для всіх POST/PUT запитів до API маршрутів.
SQL/NoSQL Injection	Маніпуляція параметрами запитів для несанкціонованого доступу до бази турнірів.	Повна відмова від сирих SQL-запитів. Взаємодія з БД здійснюється виключно через параметризовані методи Prisma ORM.
DDoS (Distributed Denial of Service)	Перевантаження серверних потужностей смітєвим трафіком під час живих матчів.	Делегування фільтрації трафіку інфраструктурі Cloudflare; імплементація механізму Rate Limiting.

Оскільки клієнтська частина платформи є високодинамічним односторінковим застосунком (SPA), що оперує великими масивами спортивної статистики в реальному часі, архітектура управління станом вимагала особливого інженерного підходу. Класичне передавання даних через ієрархію

компонентів (Props Drilling) було відхилено через його неефективність та складність підтримки коду.

Глобальну архітектуру управління станом поділено на два незалежні контури: стан сервера (Server State) та стан клієнта (Client State). Для роботи з серверним станом (асинхронні запити до бази даних, отримання результатів матчів) застосовано патерни кешування на основі SWR / React Query. Цей підхід забезпечує автоматичну фонову ревалідацію даних: якщо під час перегляду користувачем турнірної таблиці на сервері змінюється рахунок матчу, застосунок ініціює точковий ререндеринг лише відповідного компонента інтерфейсу без повного перезавантаження сторінки. У таблиці 2.11 наведено специфікацію рівнів управління станом для ключових модулів інформаційної системи.

Таблиця 2.11

Декомпозиція рівнів управління станом у клієнтському застосунку

Тип стану	Область застосування	Технологія / Інструмент	Механізм оновлення
Локальний (UI State)	Стан розкритих меню, активні вкладки, модальні вікна конструктора команди.	React useState / useReducer	Синхронний точковий ререндеринг компонента.
Глобальний (App State)	Тема оформлення (Dark/Light), налаштування локалізації, загальний об'єкт авторизації.	React Context API / Zustand	Глобальне оповіщення підписників (Observer).
Серверний (Async State)	Каталоги гравців, турнірні таблиці, масиви новин, історія транзакцій LiqPay.	Серверні компоненти (RSC) / SWR	Асинхронне отримання (Fetch) із застосуванням інвалідації кешу.
Реактивний (Live State)	Події матчів у реальному часі, таймери, зміна xG під час гри.	WebSockets	Push-повідомлення від сервера до підписаних клієнтів.

Оскільки інформаційна система передбачає наявність преміальної підписки, критичним етапом розробки стало проектування надійної архітектури взаємодії із зовнішнім платіжним шлюзом LiqPay. Основною інженерною вимогою до цієї підсистеми є забезпечення гарантованої консистентності (узгодженості) даних між банківським еквайрингом та локальною базою даних користувачів.

Процес обробки платежів побудовано на асинхронній моделі комунікації типу Webhook (зворотний виклик). Під час ініціалізації платежу клієнтом, сервер

платформи формує зашифрований пакет даних (Payload), що містить ідентифікатор користувача, суму транзакції та унікальний номер замовлення (Order ID). Пакет підписується криптографічним алгоритмом SHA-1 із використанням закритого приватного ключа. Це унеможливорює модифікацію суми платежу на стороні клієнтського браузера.

Отримання результату транзакції виконується через спеціальний захищений серверний маршрут (Webhook Endpoint), який очікує POST-запити виключно від серверів LiqPay. Для запобігання атакам типу «підробка транзакції» (Spoofing), API-маршрут обов'язково виконує зворотне криптографічне гешування вхідних даних. Лише після успішної верифікації електронного підпису система відкриває транзакцію у базі даних PostgreSQL (через Prisma ORM) та автоматично змінює рівень доступу користувача на «Premium». У таблиці 2.12 наведено специфікацію життєвого циклу транзакцій та механізми реакції бекенду на відповідні статуси платіжного шлюзу.

Таблиця 2.12

Життєвий цикл фінансової транзакції та стани системи

Статус від LiqPay	Опис стану транзакції	Архітектурна дія на сервері (Бекенд)
wait_accept	Гроші заблоковані на картці, очікується підтвердження банку.	Створення запису транзакції зі статусом "Pending".
success	Успішне списання коштів із рахунку користувача.	Верифікація сигнатури; оновлення ролі користувача; надання преміум-доступу.
failure	Помилка обробки (недостатньо коштів або ліміт).	Логування помилки; відправка клієнту повідомлення про відхилення.
reversed	Повернення коштів (Refund) ініційоване адміністратором.	Анулювання транзакції в БД; автоматичне блокування преміального контенту.
error	Критична помилка на боці платіжного провайдера.	Запуск механізму повторних спроб (Retry Pattern) або сповіщення адміністратора.

Враховуючи специфіку роботи спортивного аналітичного порталу, де співвідношення операцій читання до операцій запису становить орієнтовно 95% до 5%, ключовим інженерним рішенням стала імплементація агресивної багаторівневої стратегії кешування. Такий підхід гарантує стабільність системи під час різких стрибків трафіку (Spikes), характерних для періодів проведення вирішальних турнірів.

На рівні інфраструктури застосовується механізм інкрементальної статичної регенерації (Incremental Static Regeneration, ISR). Він дозволяє генерувати публічні сторінки (наприклад, каталоги ліг та профілі команд) безпосередньо під час збирання проєкту на сервері. Надалі ці сторінки роздаються кінцевим користувачам миттєво з вузлів глобальної мережі доставки контенту (CDN).

Для динамічних даних, таких як турнірні таблиці та особисті фентезі-команди, застосовується алгоритм Stale-While-Revalidate (SWR). Згідно з цією парадигмою, користувач миттєво отримує закешовану (стару) версію даних, тоді як клієнтський застосунок асинхронно виконує фоновий запит до бази даних для перевірки оновлень. У разі виявлення розбіжностей, інтерфейс плавно оновлюється без візуального блокування екрана (Non-blocking UI). У таблиці 2.13 описано конфігурації TTL (Time-To-Live) для різних сегментів інформаційної системи.

Таблиця 2.13

Політики кешування та TTL-конфігурації підсистем

Сегмент даних (Підсистема)	Технологія / Патерн	Час життя кешу (TTL)	Умова інвалідації кешу (Очищення)
Глобальний спортивний каталог	Static Generation (SSG)	24 години	Ручний тригер з адмін-панелі (On-Demand).
Сторінки розкладу матчів	ISR (Incremental)	3600 секунд (1 год)	Завершення таймера або зміна статусу матчу.
Турнірні таблиці (Live)	SWR (Stale-While-Revalidate)	60 секунд	Фонova ревалідація при фокусуванні на вкладці.
Профіль та підписки	No-Store (Bypass Cache)	0 секунд (Вимкнено)	Миттєве оновлення при будь-якій транзакції.

Взаємодія між клієнтським застосунком та серверною бізнес-логікою реалізована за допомогою архітектурного стилю REST (Representational State Transfer). Для оптимізації мережевого трафіку та забезпечення надійної типізації всі дані передаються у форматі JSON. У таблиці 2.14 наведено детальну специфікацію основних програмних інтерфейсів (API Endpoints), які обслуговують ключові модулі системи.

Специфікація методів, маршрутів та корисного навантаження (Payload)

REST API

HTTP Метод	Маршрут (Endpoint)	Призначення	Формат Payload (Запит)	Формат відповіді
POST	/api/fantasy-teams	Створення нової фентезі-команди користувача.	{"name": string, "formation": string, "players": array}	201 Created (Об'єкт команди)
GET	/api/matches/live	Отримання результатів поточних матчів.	Немає (Query params: sportId)	200 OK (Масив подій)
POST	/api/payments/webhook	Обробка зворотного виклику від платіжного шлюзу LiqPay.	Унікальний криптографічний хеш + data	200 OK (Статус транзакції)
PUT	/api/users/premium	Оновлення рівня доступу користувача.	{"userId": string, "status": "active"}	200 OK (Оновлений профіль)

Кожен API-маршрут працює в режимі суворої валідації вхідних даних за допомогою бібліотеки zod. У разі невідповідності Payload встановленим правилам (наприклад, якщо масив гравців містить менше 11 елементів), сервер автоматично перериває обробку та повертає статус 400 Bad Request із масивом деталізованих помилок, що гарантує цілісність бази даних.

2.4 Розробка основних алгоритмів роботи платформи

З метою глибшого розуміння внутрішньої логіки створюваної платформи розроблено алгоритми базових бізнес-процесів, які наочно представлено у вигляді діаграм активності. Зазначені моделі детально розкривають динаміку функціонування системи. Вони ілюструють паралельні обчислення, логічні розгалуження та суворий розподіл зон відповідальності між браузером користувача і серверним бекендом за допомогою функціональних доріжок.

Алгоритм формування фентезі-команди визначає чітку послідовність дій системи під час створення нового ігрового ростера. Цей процес характеризується складною транзакційною природою та вимагає багатоетапних перевірок бізнес-логіки на рівні домену задля гарантування чесних змагальних умов. Використання підходу на основі тактичних шаблонів не лише зменшує кількість рутинних дій для аналітика, а й гарантує високу архітектурну гнучкість під час побудови різноманітних розстановок. Завдяки ізоляції бізнес-логіки від

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

візуальної частини система здатна динамічно генерувати інтерактивний інтерфейс спортивного поля без потреби дублювання клієнтського коду.

Безпосередній процес розпочинається з ініціалізації запиту, після чого система направляє його через суворий конвеєр обробки. Даний етап охоплює валідацію вхідних даних, санітизацію текстових параметрів та жорсткий контроль фінансових лімітів віртуального бюджету. Одночасно програмно реалізуються критичні механізми захисту від підробки запитів шляхом криптографічного хешування сесій. Це забезпечує абсолютну цілісність інформації та стабільну роботу додатка навіть за умов перебоїв у мережевому з'єднанні. Додаткове підвищення відмовостійкості цього архітектурного вузла досягається завдяки інтеграції механізмів автоматичних повторних мережових викликів.

Комплексне моделювання потоків управління та даних виступає фундаментальною умовою для архітектурного контролю прихованих асинхронних процесів. Формалізований опис станів системи формує надійну інженерну платформу для подальшого масштабування програмного рішення. Це відкриває шляхи для безперешкодної інтеграції нових спортивних дисциплін та гарантує стабільність роботи в умовах підвищених навантажень під час стрімкого розширення аудиторії активних користувачів. Зокрема, така архітектурна прозорість дозволяє впровадити глибоке автоматизоване тестування, що є критично важливим для безпечного випуску оновлень без ризику деградації існуючого функціоналу. Крім того, чітке визначення асинхронних взаємодій зводить до мінімуму ймовірність виникнення конфліктів при паралельній обробці інформації, гарантуючи безперебійну доставку аналітики в режимі реального часу.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

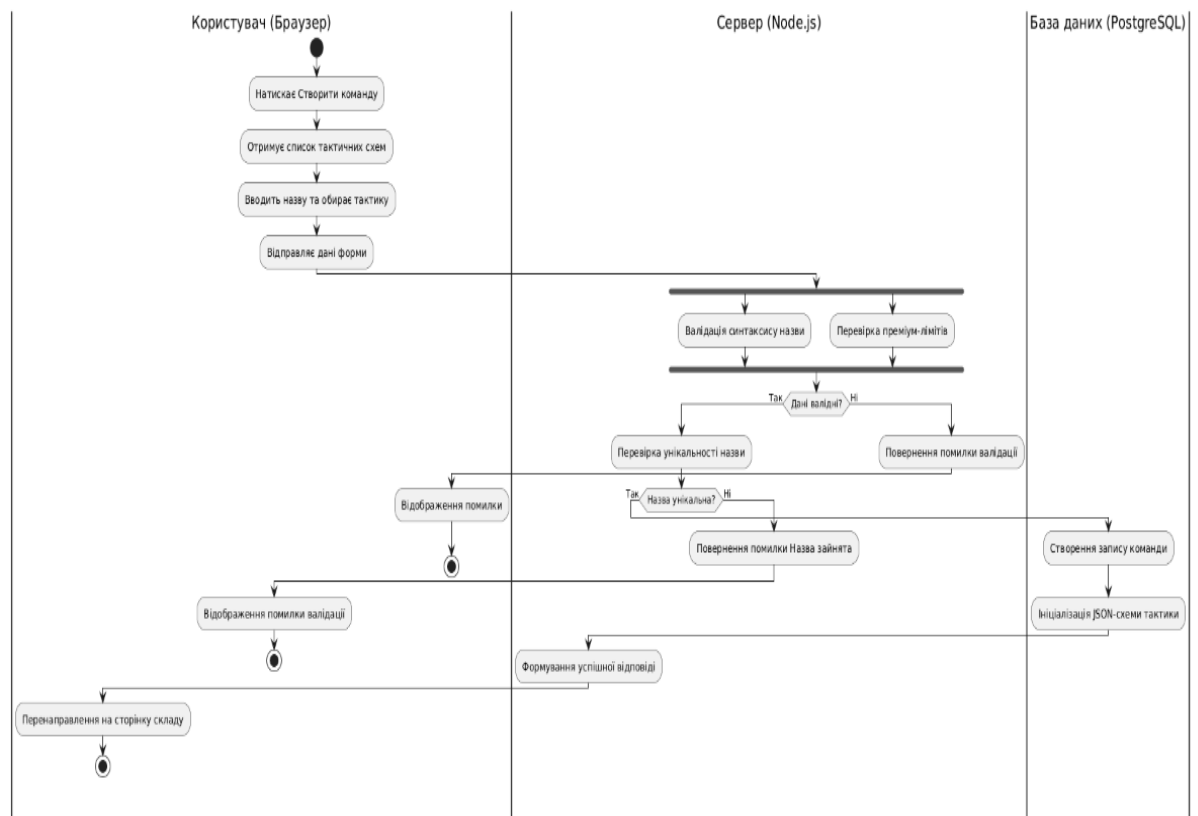


Рис. 2.13 – Діаграма активності процесу створення фентезі-команди

Покрокова генерація команди ініціюється користувачем через звернення клієнтського додатка до прикладного програмного інтерфейсу для отримання масиву доступних тактичних шаблонів. Після завантаження даних аналітик обирає відповідну схему та вказує назву майбутнього клубу. Відправка цих параметрів запускає паралельні процеси валідації на серверному бекенді у середовищі Node.js [25]. Здійснюється перевірка синтаксичної коректності введеної назви та контроль лімітів на кількість команд для поточного облікового запису, що визначається статусом підписки через платіжну систему LiqPay [30]. У разі успішного проходження первинної верифікації сервер звертається до реляційної бази PostgreSQL [4] для підтвердження абсолютної унікальності обраного імені. За умови виконання всіх обмежень у базі даних створюється новий транзакційний запис із подальшим імпортом тактичної конфігурації у форматі JSON [13]. Далі вебклієнт на базі бібліотеки React [1] реактивно перенаправляє користувача до новоствореного робочого простору для початку трансферної кампанії.

Наступний алгоритм регламентує послідовність дій під час безпосереднього управління гравцями на полі. Оскільки статистичні показники прямо впливають на рейтинг користувача, ця процедура супроводжується жорстким контролем цілісності даних. Під час маніпуляцій із картками гравців основне обчислювальне навантаження переноситься на браузерний клієнт, тоді як кількість звернень до сервера зводиться до мінімуму задля забезпечення ідеальної плавності інтерфейсу. Застосований архітектурний підхід дає змогу реалізувати концепцію оптимістичного оновлення. Зміни на віртуальному футбольному полі відображаються миттєво завдяки ефективному управлінню локальним станом [14]. Використання механізмів кешування дозволяє системі обробляти складні маніпуляції зі складом без жодних відчутних мережових затримок. Усі необхідні транзакційні зміни виконуються сервером асинхронно у фоновому режимі шляхом пакетної обробки запитів [2]. Це суттєво зменшує навантаження на серверну інфраструктуру та дозволяє користувачеві максимально сконцентруватися на аналітичному процесі.

Для гарантування абсолютної консистентності інформації між клієнтом і сервером асинхронна фонові синхронізація функціонує в синергії з надійним механізмом вирішення конфліктів. У випадку виникнення помилок на етапі серверної валідації система автоматично ініціює процедуру відкату стану. Це може статися у ситуації, коли сумарна вартість обраних спортсменів несподівано перевищила встановлений віртуальний бюджет або конкретний гравець отримав травму в реальному часі. Завдяки механізму відкату клієнтський застосунок миттєво повертає картку гравця на вихідну позицію та генерує відповідне інформаційне сповіщення. Такий підхід дозволяє коректно обробляти виняткові ситуації без переривання загального сеансу роботи користувача платформи.

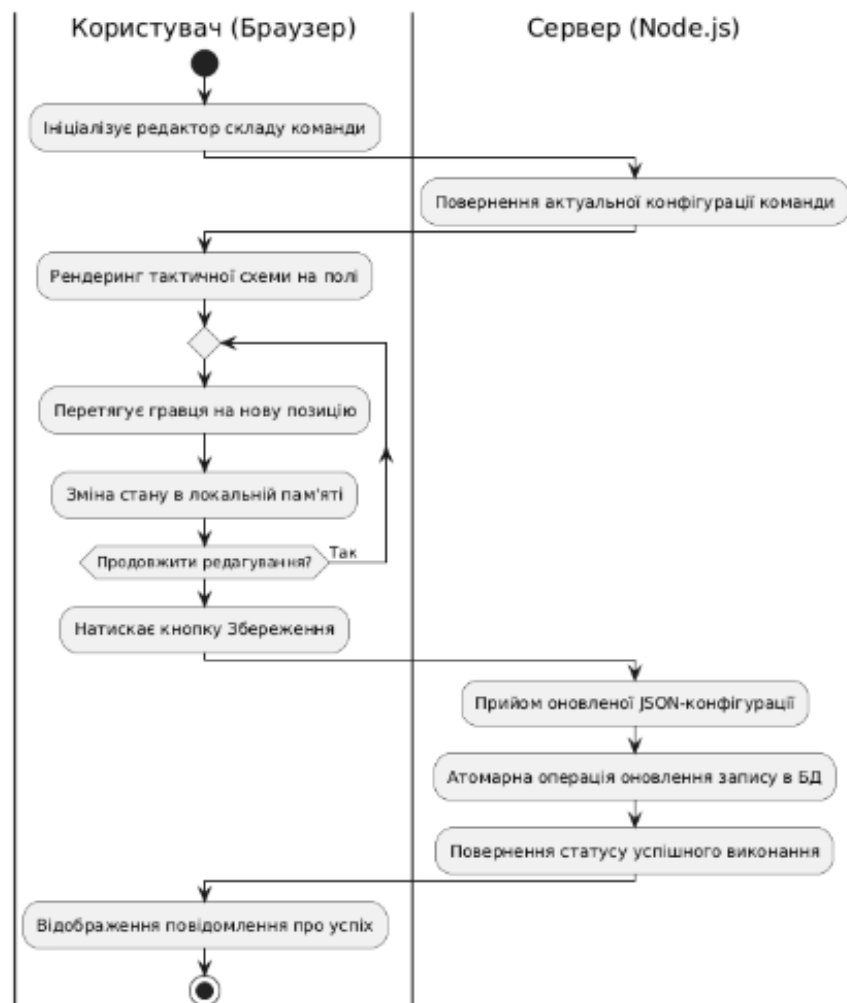


Рис. 2.14 – Діаграма активності процесу редагування тактичного складу

Під час первинного завантаження тактичного редактора серверне ядро на базі платформи Node.js [4] відправляє до браузера актуальний масив обраних гравців та збережені параметри тактичної схеми. Цей процес дозволяє вебклієнту отримати повний зріз стану команди для подальшої локальної маніпуляції об'єктами. Відразу після отримання пакета даних інтерфейс генерує інтерактивне футбольне поле, на якому розташовуються картки футболістів згідно з попередньо обраною розстановкою.

Будь-яка безпосередня взаємодія зі складом команди, така як заміни гравців або перепризначення капітана, виконується виключно в оперативній пам'яті комп'ютера користувача. Використання сучасних бібліотек для глобального керування станом додатка [32] дозволяє системі миттєво реагувати на рухи миші без жодних мережових затримок. Усі внесені тактичні зміни накопичуються локально у вигляді єдиного об'єкта конфігурації.

Лише після натискання кнопки збереження застосунк формують фінальний пакет даних, який описує нову структуру розташування спортсменів на полі [13]. Згенерований пакет передається на серверну частину, де ініціюється атомарна операція оновлення запису у відповідній таблиці реляційної бази даних PostgreSQL [29]. Успішно зафіксувавши транзакцію, бекенд віддає сигнал підтвердження, а користувач бачить зелене сповіщення про надійне збереження нового складу команди.

Наступний алгоритм розкриває логіку взаємодії користувача з платформою під час спроби отримати доступ до ексклюзивних статистичних матеріалів. Сюди входять глибокі метрики очікуваних голів, деталізовані теплові карти переміщень та розширені аналітичні дашборди. Цей процес супроводжується багатоваровою перевіркою прав доступу. Система не лише аналізує поточний рівень облікового запису, але й проводить безпечну верифікацію платіжних статусів для суворого розмежування безкоштовного та преміального контенту.

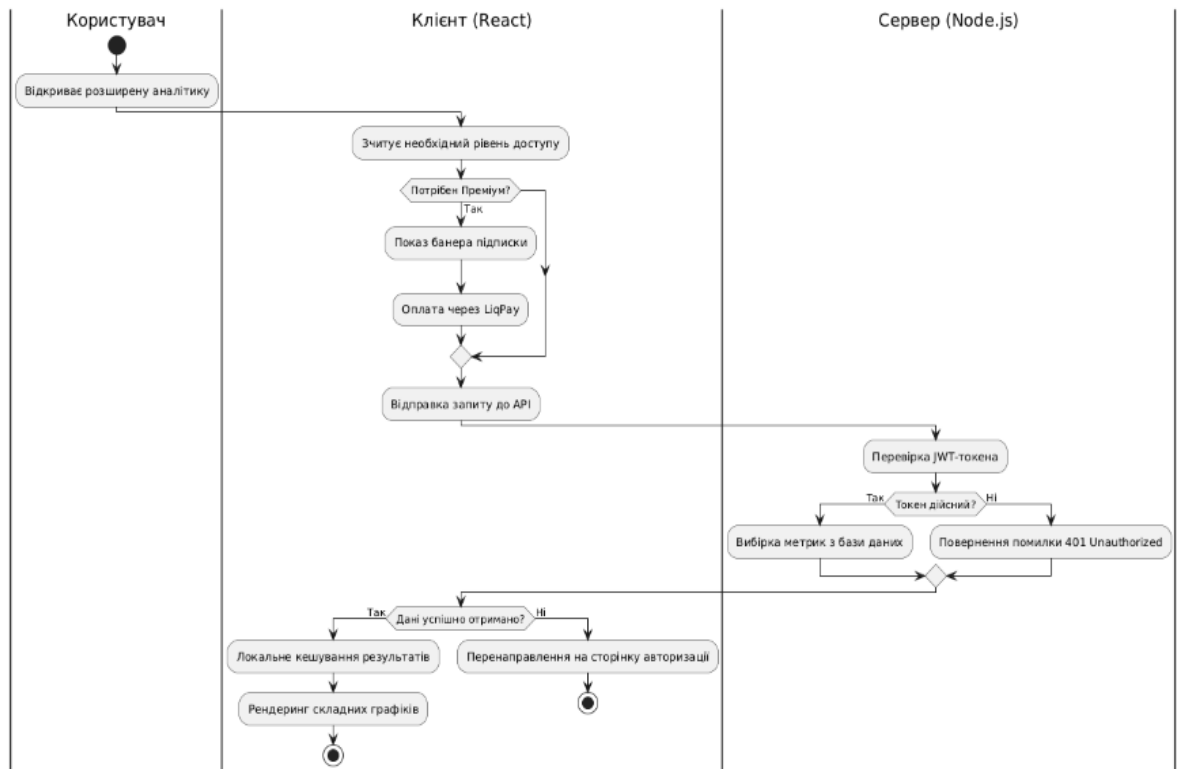


Рис. 2.15 – Діаграма активності процесу отримання розширеної аналітики

Потік виконання ініціюється в момент взаємодії користувача із заблокованим аналітичним віджетом. Клієнтський інтерфейс в автоматичному

режимі перевіряє рівень прав, необхідних для перегляду обраного графіка. У випадку, коли запитуваний матеріал належить до категорії для професійних аналітиків, а відвідувач використовує базовий тарифний план, система миттєво зупиняє рендеринг візуальних компонентів. Замість закритої інформації на екрані з'являється інтерактивний банер із пропозицією оформлення преміальної підписки через захищений вітчизняний платіжний шлюз LiqPay [30].

Під час спроби завантажити закриті масиви даних серверний бекенд першочергово здійснює валідацію криптографічного токена авторизації [17]. За умови його відсутності або вичерпання терміну дії система ініціює примусове перенаправлення користувача на сторінку автентифікації. Якщо верифікація завершується успішно, відповідний контролер підтверджує наявність преміум-статусу в профілі та запускає процес виконання складних аналітичних вибірок з бази даних. Отримані результати одразу зберігаються в локальному кеші браузера [14] задля мінімізації кількості повторних мережевих запитів. Лише після проходження всіх цих етапів на екрані пристрою плавно відмальовуються комплексні порівняльні діаграми.

Завершальний алгоритм описує механізм приховування небажаних публікацій з боку адміністрації платформи. Впровадження такого інструмента модерації виступає критично важливою умовою для підтримки бездоганної репутації інформаційного ресурсу. Враховуючи специфіку роботи зі спортивними новинами, де рівень достовірності прямо корелює з довірою аудиторії, вебзастосунок зобов'язаний миттєво реагувати на будь-які порушення. Цей процес глибоко інтегровано із серверними тригерами. При досягненні критичної маси скарг на певний аналітичний дашборд чи статтю проблемний контент автоматично приховується від публічного доступу та переводиться в статус призупиненого до моменту ухвалення остаточного рішення командою модераторів.

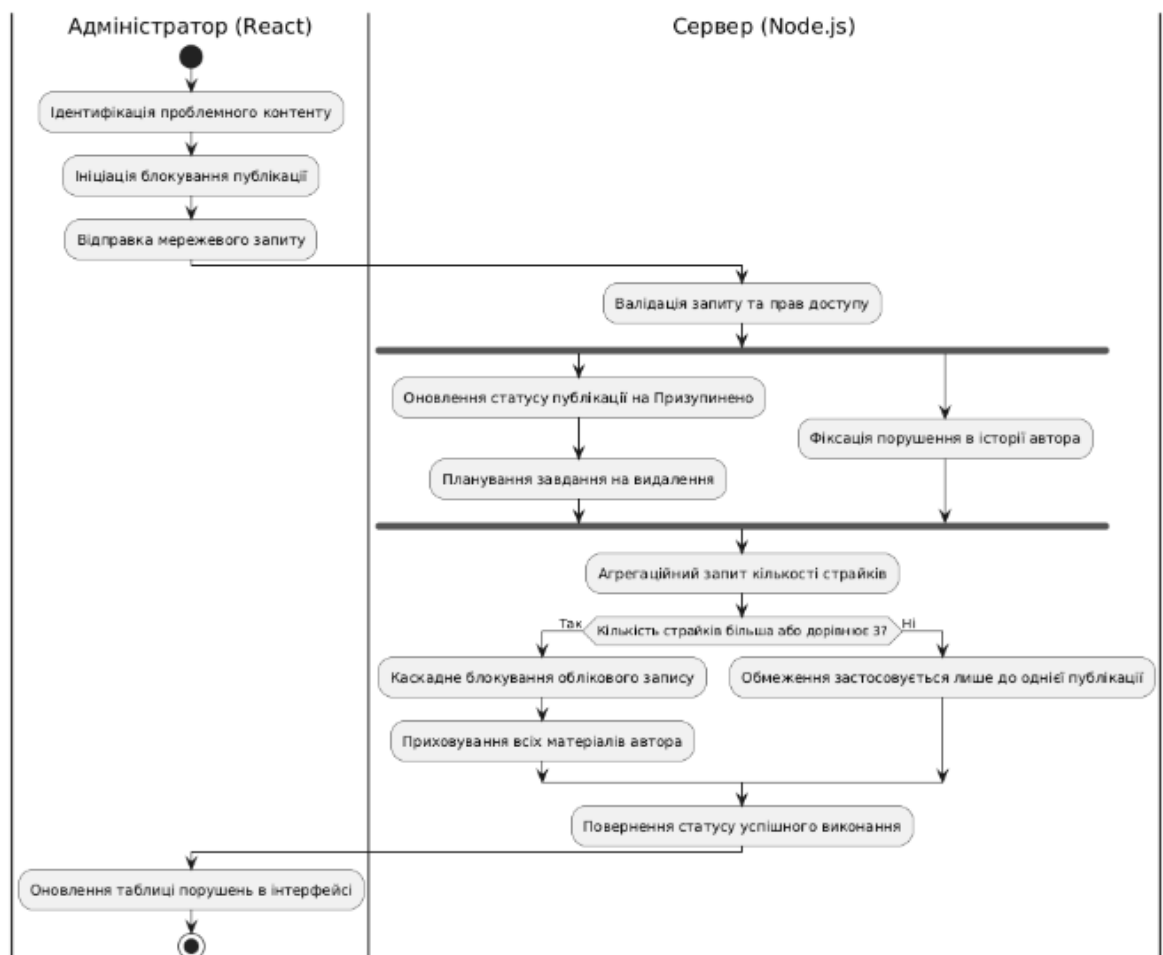


Рис. 2.16 – Діаграма активності " Призупинення публікації на рівні адміністратора "

Процес призупинення публікації на рівні адміністратора проілюстровано відповідною діаграмою активності (рисунок 2.16). Процес модерації розпочинається з етапу ініціації, під час якого через спеціалізовану вебпанель керування виконується ідентифікація аналітичного дашборда або новинної статті, що порушує правила спортивної спільноти, з подальшим запуском процедури блокування. Відповідний мережевий запит надійно передається на хмарний бекенд аналітичної платформи.

Після отримання та валідації дії серверне середовище Node.js [4] одночасно запускає два паралельні процеси обробки. З одного боку, система оновлює статус проблемної публікації на призупинений, автоматично плануючи фонове завдання щодо її безповоротного видалення через тридцять днів для своєчасного очищення реляційної бази даних [13]. З іншого боку, паралельно формується

новий транзакційний запис у таблиці попереджень кластера PostgreSQL [29] для фіксації порушення безпосередньо в історії автора контенту.

Наступним кроком система автоматично виконує агрегаційний запит до бази даних для підрахунку загальної кількості активних страйків у автора даного матеріалу. На основі отриманих даних спрацьовує ключове логічне розгалуження алгоритму. Якщо сумарна кількість попереджень досягла або перевищила критичну межу, бекенд ініціює каскадне блокування всього облікового запису порушника та превентивно приховує всі його публічні матеріали із загального каталогу платформи. У випадку, якщо встановлений ліміт порушень не перевищено, застосовані обмеження стосуються виключно поточного виявленого порушення [17]. На фінальному етапі адміністративна панель синхронізується з сервером, візуально позначаючи контент як заблокований.

Висновки до другого розділу

У межах другого розділу здійснено комплексне проектування програмної архітектури інформаційної системи, призначеної для агрегації та глибокого аналізу спортивної статистики. Процес розробки охопив усі фундаментальні етапи від створення високорівневої моделі серверної інфраструктури до деталізації алгоритмів обробки потокових даних і синхронізації турнірного прогресу між браузерним клієнтом та хмарним бекендом.

Під час роботи було сформовано надійну модель розгортання програмного продукту. Обґрунтовано доцільність впровадження розподіленого хмарного підходу із залученням передових рішень для хостингу платформи на базі Node.js. Управління базою даних покладено на потужний кластер PostgreSQL, тоді як для глобальної оптимізації та миттєвої доставки об'ємних мультимедійних файлів інтегровано мережу Cloudinary. Запропонована інфраструктурна конфігурація повністю гарантує безперебійне функціонування сервісу в умовах пікових навантажень під час масових спортивних заходів.

Важливим етапом стала чітка формалізація варіантів використання аналітичної платформи. Було визначено ключові ролі користувачів, серед яких Аналітик, Контент-менеджер та Адміністратор, а також розроблено вичерпні

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

інженерні специфікації для базових прецедентів взаємодії з дашбордами. Окремий фокус спрямовано на логіку побудови персоналізованих фентезі-команд та механізми багаторівневої модерації новинного контенту, що виступає запорукою дотримання високих стандартів достовірності інформації.

З метою реалізації комплексних бізнес-вимог спроектовано надійну об'єктно-орієнтовану структуру вебдодатка. Створені діаграми класів наочно демонструють практичне впровадження класичних інженерних патернів. Застосування модульної архітектури для фронтенду на основі екосистеми React надає можливість ізольовано вдосконалювати аналітичні віджети та ігрові механіки віртуального драфту при повному збереженні загальної швидкодії системи.

Паралельно було розроблено оптимізовану реляційну модель збереження інформації. Логічна структура бази гармонійно поєднує переваги суворих зв'язків для управління профілями та фінансовими транзакціями через шлюз LiqPay із гнучкістю текстового формату JSON для фіксації динамічних тактичних конфігурацій. Застосований гібридний підхід із використанням інструментарію об'єктно-реляційного відображення гарантує здатність платформи до оперативної адаптації під нові види спорту без проведення ризикованих міграцій базової схеми.

На завершальному етапі докладно описано динамічну поведінку інформаційної платформи. Залучення діаграм активності, послідовності та кооперації дозволило глибоко деталізувати бізнес-логіку обробки живої статистики, модерації ресурсів та управління складами команд. Було успішно візуалізовано хронологію комунікації між вебклієнтом, панеллю керування, серверними контролерами та шаром бази даних у рамках сучасної архітектури прикладних програмних інтерфейсів.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЄКТУ

3.1 Порядок встановлення та налаштування параметрів системи

Аналітична платформа MySports розгортається у розподіленому хмарному середовищі, а процес від написання коду до появи оновленої версії на робочому сервері є повністю автоматизованим. Розробник не виконує жодних ручних дій для публікації релізу. Для цього достатньо інтегрувати перевірені зміни до основної гілки репозиторію. Усю подальшу роботу бере на себе сучасний конвеєр безперервної інтеграції та розгортання платформи Vercel [21].

Програмний комплекс використовує спеціалізовані хмарні служби, детальний перелік яких наведено у таблиці 3.1. Кожна з них відповідає за окремий критичний аспект роботи платформи і здатна масштабуватися незалежно від решти компонентів інфраструктури.

Таблиця 3.1

Хмарна інфраструктура системи

Служба	Платформа	Призначення
Vercel Cloud Platform	Vercel	Хостинг основного серверного середовища та клієнта застосунку
Neon Serverless Postgres	Хмарна платформа Neon	Хмарне зберігання реляційної спортивної статистики та профілів
LiqPay API	ПриватБанк	Безпечна обробка фінансових транзакцій та преміум-підписок
Google Cloud Console	Google	Система авторизації користувачів через провайдера інтеграцій OAuth
GitHub Actions	GitHub	Реєстр вихідного коду та інтеграція для безперервного розгортання

Клієнтські браузерні застосунки функціонують у режимі суворої ізоляції, де будь-які прямі транзакції від інтерфейсу користувача до бази даних фізично блокуються на архітектурному рівні. Єдиним дозволеним каналом взаємодії є серверні маршрути фреймворку Next.js [2], які розгорнуті на хмарних потужностях Vercel [19]. Зазначеному вузлу відведено роль головного оркестратора. Через нього здійснюється управління структурованими реляційними записами у безсерверному кластері Neon [29] та перенаправлення фінансових операцій до платіжного шлюзу LiqPay [30]. Маршрут доставки готового програмного коду пролягає виключно через систему

контролю версій. За допомогою вбудованого конвеєра платформи Vercel забезпечується автоматична компіляція та тестування проєкту після кожного збереження оновлень. На цьому етапі особлива увага приділяється безпомилковому мапінгу змінних середовища для коректної маршрутизації бекенд-запитів, що гарантує стабільність розгортання.

Для забезпечення незалежного локального розгортання та ізольованого тестування інфраструктури серверна частина пакується у віртуальний образ за допомогою технології багатоетапної збірки Docker. На початковому етапі завантажуються всі необхідні бібліотеки та виконується транспіляція вихідного коду з TypeScript [5] у JavaScript. На наступному кроці до фінального контейнера переносяться лише скомпільовані артефакти та виключно робочі залежності, залишаючи важкі інструменти розробки поза межами збірки. Разом із основним застосунком автоматично піднімаються локальні копії бази PostgreSQL та кеш-сервера Redis, що забезпечує повну ідентичність із виробничим середовищем. Фундаментом для такої віртуальної збірки слугує мінімально оптимізований образ із середовищем виконання Node.js [4].

Конвеєр безперервної доставки реалізовано за допомогою інструментарію GitHub Actions, (рисунок 3.1). Цей механізм поділено на послідовні етапи із жорсткими залежностями. Логіка роботи конвеєра є абсолютно безкомпромісною. Будь-який провал перевірки типів або виявлення помилок під час модульного тестування миттєво скасовує подальше розгортання. Додатковим запобіжником виступає строга фільтрація операцій за гілками репозиторію. Платформа також синхронізує стан віддаленого репозиторію, враховуючи чистоту індексованого кешу файлів, щоб уникнути конфліктів при злитті. Решта гілок проходять лише етапи аудиту якості та ізольованого тестування, жодним чином не впливаючи на робоче середовище хмарної платформи Vercel.

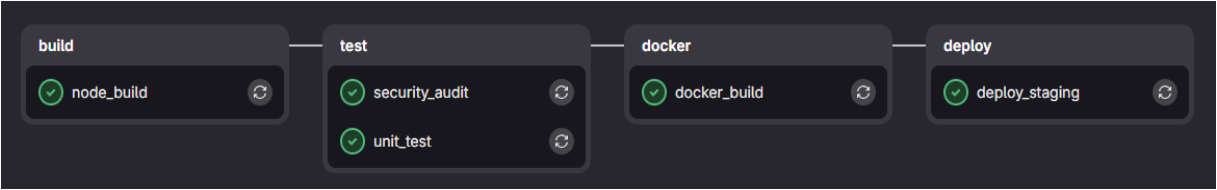


Рис. 3.1 – Діаграма конвеєра безперервного розгортання

Стадії конвеєра

Стадія	Завдання	Зміст
1	2	3
build	npm run build	Компіляція статичних файлів та оптимізація застосунку через фреймворк Next.js
test	npm run test	Виконання модульних тестів бази коду та аудит безпеки встановлених залежностей
docker	docker build	Збірка віртуального образу бази даних та кешу Redis для ізолизованого середовища
deploy	Vercel CLI	Завантаження оновленої версії вебзастосунку на глобальний хмарний хостинг Vercel

Стадія збірки перевіряє компіляцію коду і виявляє потенційно небезпечні конструкції ще до запуску перевірок. Модульні тести перевіряють доменну логіку, наприклад алгоритми обчислення фентезі-балів чи логіку парсингу матчів, без прямого підключення до продакшен-бази даних. Стадія автоматичного розгортання гарантує, що нова версія коду безшовно замінює попередню без зупинки роботи порталу.

Чутливі параметри конфігурації – рядок підключення до бази даних Neon, ключі підпису сесій, облікові дані служб LiqPay та Google категорично не зберігаються у вихідному коді. Вони безпечно зберігаються у вигляді криптографічних секретів і встановлюються безпосередньо у панелі керування Vercel. Ці захищені змінні не відображаються у логах конвеєра навіть при явному виведенні, оскільки платформа автоматично маскує їхні значення.

Параметри конфігурації виробничого середовища

Параметр	Призначення
DATABASE_URL	Рядок підключення до безсерверного кластера бази даних Neon Postgres
NEXTAUTH_SECRET	Криптографічний ключ для підпису токенів безпеки та сесій користувачів
LIQPAY_PRIVATE_KEY	Секретний ключ для управління підписками через платіжний шлюз ПриватБанку [30]
GOOGLE_CLIENT_ID	Ідентифікатор застосунку для забезпечення безпечної авторизації через Google [31]
NEXT_PUBLIC_APP_URL	Глобальна адреса платформи для коректної маршрутизації клієнтських запитів

Для розробки та локального тестування система запускається без підключення до хмарного хостингу. Розробнику достатньо створити локальний

конфігураційний файл оточення та виконати базову команду старту в робочій директорії, яка ініціює роботу локального сервера.

Команда запускає три контейнери, що утворюють повну локальну копію аналітичної системи платформи. Детальний перелік цих сервісів наведено у таблиці 3.4.

Таблиця 3.4

Сервіси локального середовища

Контейнер	Образ	Порт	Призначення
mysports_app	локальна збірка з Dockerfile	3000	Основний клієнтський та серверний вебзастосунок платформи
mysports_db	postgres:15-alpine	5432	Локальна реляційна база даних для збереження спортивної статистики
mysports_redis	redis:7-alpine	6379	Система кешування даних та підтримки швидкісних вебсокетів

Процедура локального розгортання автоматично запускає виконання скриптів для наповнення бази даних початковим тестовим контентом, що включає інформацію про команди та спортсменів. Цей процес реалізується за допомогою інструментарію Prisma. Після успішного формування базових колекцій записів клієнтський інтерфейс порталу стає доступним за адресою локального порту 3000. Застосоване архітектурне рішення дозволяє перетворити робочу станцію розробника на абсолютно точну структурну копію робочого хмарного сервера. Завдяки цьому повністю усувається ризик появи критичних розбіжностей між середовищем розробки та виробничою інфраструктурою.

Взаємодія із зовнішніми постачальниками спортивних даних на етапі розробки здійснюється в повністю автономному режимі. Локальні конфігураційні файли примусово переводять усі сторонні інтеграції у формат безпечної пісочниці. Для прикладу замість виконання реальних фінансових транзакцій серверна частина імітує успішні відповіді від тестового середовища платіжного шлюзу, тоді як синхронізація статистичних зведень може спиратися на локальні файли-заглушки. Зазначений підхід надає розробникам можливість безпечно перевіряти працездатність фінансових модулів без залучення реальних коштів та запобігає вичерпанню доступних лімітів запитів до сторонніх прикладних програмних інтерфейсів.

Щодо демонтажу локальної інфраструктури, то повне очищення середовища, включно зі знищенням контейнерів та прив'язаних томів збереження даних pgdata, ініціюється єдиним консольним запитом:

```
docker compose down -v
```

Під час наступного ініціалізаційного запуску інформаційної системи скрипти первинного наповнення застосовуються повторно. У результаті вебплатформа розпочинає функціонування з абсолютно чистою реляційною базою PostgreSQL. Такий механізм гарантує високу стабільність етапу тестування та повністю виключає негативний вплив будь-яких залишкових артефактів із попередніх сеансів розробки.

3.2 Структура інтерфейсу системи аналітики спортивних даних

Інфраструктура платформи MySports включає повноцінну адаптивну вебчастину, яка однаково добре функціонує як на десктопних, так і на мобільних пристроях. Головна сторінка порталу виконує презентаційну та навігаційну функції. Вона оформлена у сучасному мінімалістичному стилі з використанням масивної типографіки, темної спортивної теми та динамічних градієнтних акцентів. Заголовок із текстом Збирай команду мрії трансліює основний емоційний посил бренду. На сторінці розміщено інтерактивні елементи для переходу до каталогу, навігаційне меню для вибору виду спорту та точку входу до закритої адміністративної панелі.

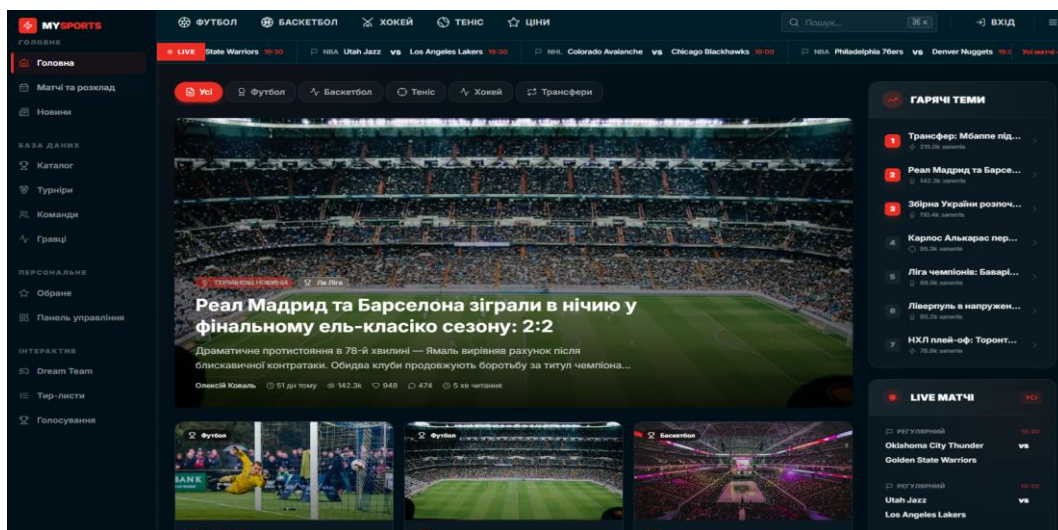


Рис. 3.2 – Головна сторінка вебсайту платформи MySports

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		78

Інтерфейс клієнтського застосунку MySports спроектовано з урахуванням сучасних вимог до взаємодії з користувачем, що забезпечує швидку навігацію масивами спортивної статистики. На етапі першого відвідування платформи неавторизованому профілю відображається екран привітання. Його центральним елементом виступає тематична ілюстрація, спрямована на візуалізацію динаміки спортивних змагань. Використання оптимізованої графіки сприяє забезпеченню високої чіткості відображення на екранах мобільних пристроїв із різною щільністю пікселів без втрати швидкодії платформи. На екрані також розміщено відповідні елементи управління для ініціалізації процесу формування нової команди або проходження процедури авторизації вже зареєстрованими аналітиками.

Під час переходу до базового розділу системи відбувається відображення головного екрана спортивного каталогу. Його структуру спроектовано з акцентом на центральний інформаційний заголовок, що підкреслює масштабність зібраної статистики та задає професійний інформаційний тон. Використання оптимізованої типографіки та карток категорій забезпечує високу чіткість відображення інтерфейсу на моніторах різної роздільної здатності без втрати загальної швидкодії платформи. Для швидкої категоризації даних за видами спорту (футбол, баскетбол, теніс, хокей) на екрані розміщено відповідні навігаційні блоки, а також елементи управління для прямого переходу до загальних реєстрів спортивних клубів та профілів гравців. Таке просторове компонування інтерфейсу не лише мінімізує когнітивне навантаження на користувача, але й дозволяє інтуїтивно орієнтуватися у великих масивах аналітичної інформації. Додатково, модульна побудова головного екрана передбачає можливість легкого масштабування каталогу новими спортивними дисциплінами без необхідності перероблення загального макета сторінки. Крім того, застосування принципів адаптивного дизайну та механізмів відкладеного завантаження (lazy loading) графічних ресурсів гарантує безперервний та якісний користувацький досвід на мобільних пристроях. Це рішення не лише підвищує

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

загальну ергономіку системи, але й суттєво оптимізує споживання мережевого трафіку під час первинного рендерингу сторінки.

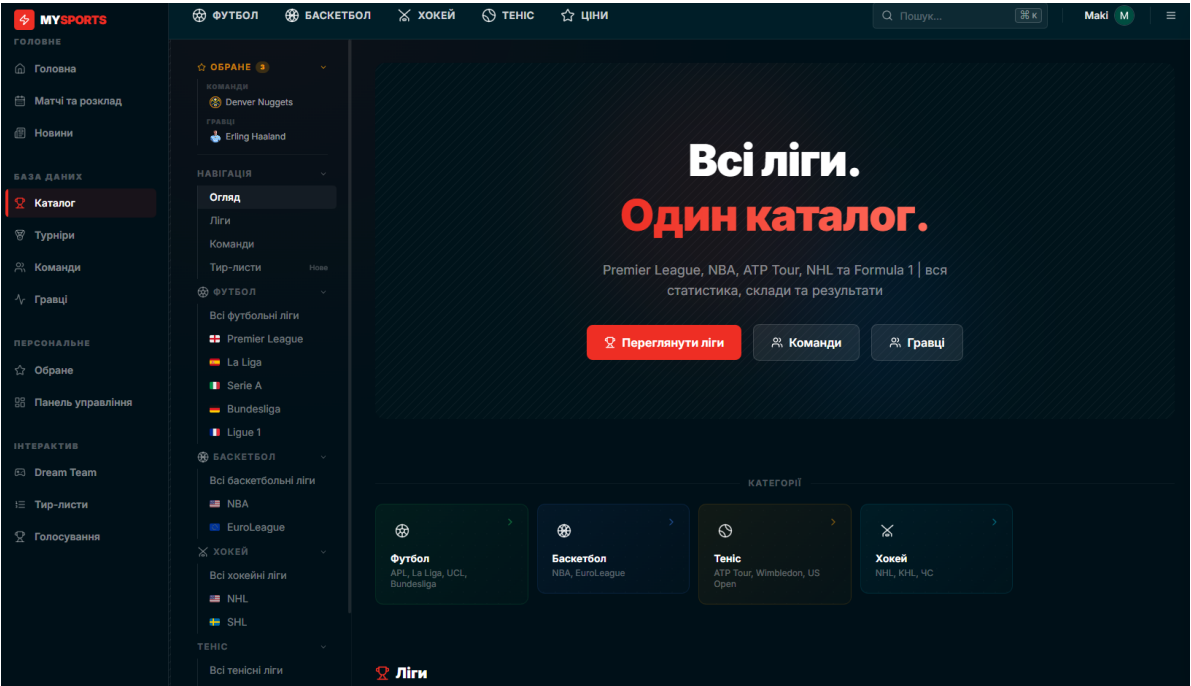


Рис. 3.3 – Головний екран спортивного каталогу

Основою оперативного моніторингу спортивних турнірів на платформі MySports є розділ розкладу матчів. На екрані календаря відображається горизонтальна стрічка дат із можливістю швидкого перемикання між минулими, поточними та майбутніми подіями. Ключовою особливістю підсистеми є розширений механізм миттєвого фільтрування контенту. Для зручності аналізу передбачено можливість візуального сортування видачі за конкретним видом спорту або статусом поєдинку з використанням відповідних елементів управління. Завдяки імплементації технології асинхронних запитів оновлення списку трансляцій та результатів відбувається плавно і без необхідності повного перезавантаження сторінки. Такий архітектурний підхід суттєво зменшує навантаження на серверну частину, оскільки обробляються та передаються лише цільові пакети даних, необхідні для оновлення конкретних компонентів інтерфейсу. Як наслідок, користувачі забезпечуються безперебійним доступом до актуальної інформації в режимі реального часу, що є вкрай важливим під час одночасного проведення кількох масштабних спортивних подій.

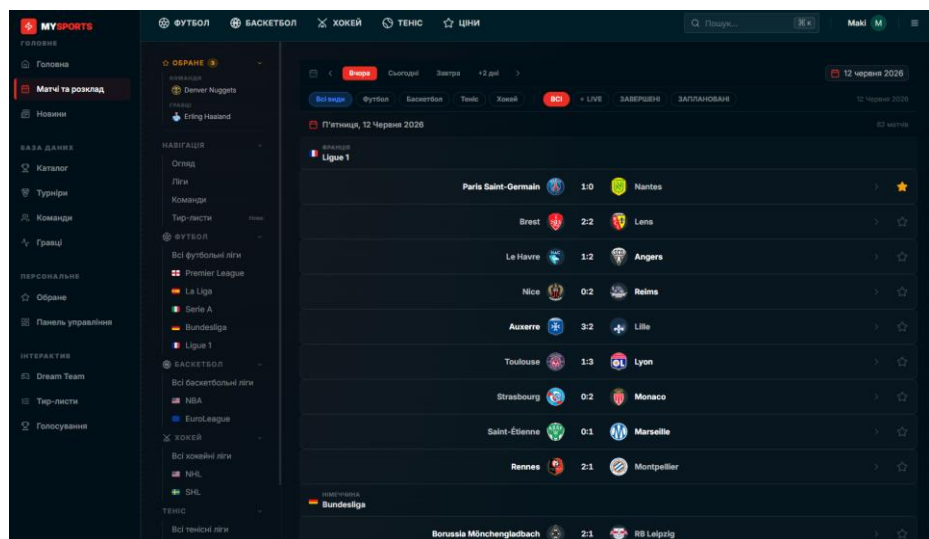


Рис. 3.4 – Інтерактивний календар та розклад матчів

Особистий кабінет користувача оформлено у вигляді персональної аналітичної панелі управління (дашборда). У межах цього інтерфейсу виконується агрегація ключової статистики активності профілю: кількості сформованих команд мрії, доданих до обраного клубів, відстежуваних гравців та збережених розкладів матчів. У центральній частині екрана передбачено спеціальний навігаційний компонент для прямого переходу до модуля конструктора віртуального тактичного складу. Додатково у правій частині екрана відображається компактний блок профілю із зазначенням системної ролі та поточного рівня підписки, що супроводжується меню швидкої навігації.

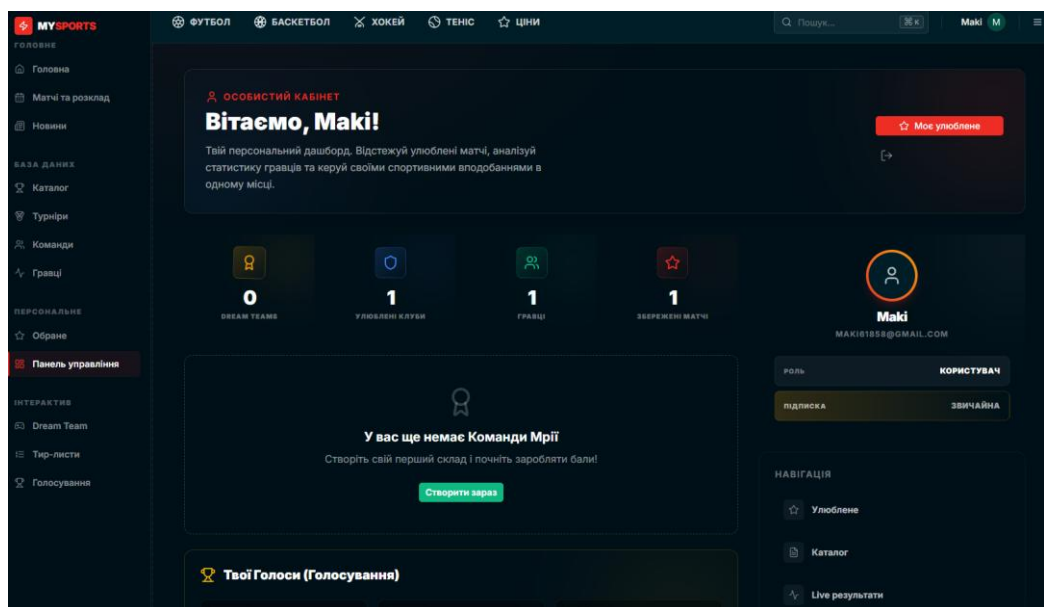


Рис. 3.5 – Персональна панель управління аналітиком

Робочий простір контент-менеджера спроектовано для максимальної оптимізації процесів наповнення бази даних. Дашборд відображає персональну статистику співробітника: кількість доданих турнірів, профілів гравців та оброблених матчів, а також його поточний ранг у системі. Ключовим елементом є спеціалізована панель керування публікаціями, де розміщено кнопки для створення новин за готовими шаблонами та управління вже існуючими матеріалами. Додатково виводяться лічильники статусів статей для зручного контролю модерації.

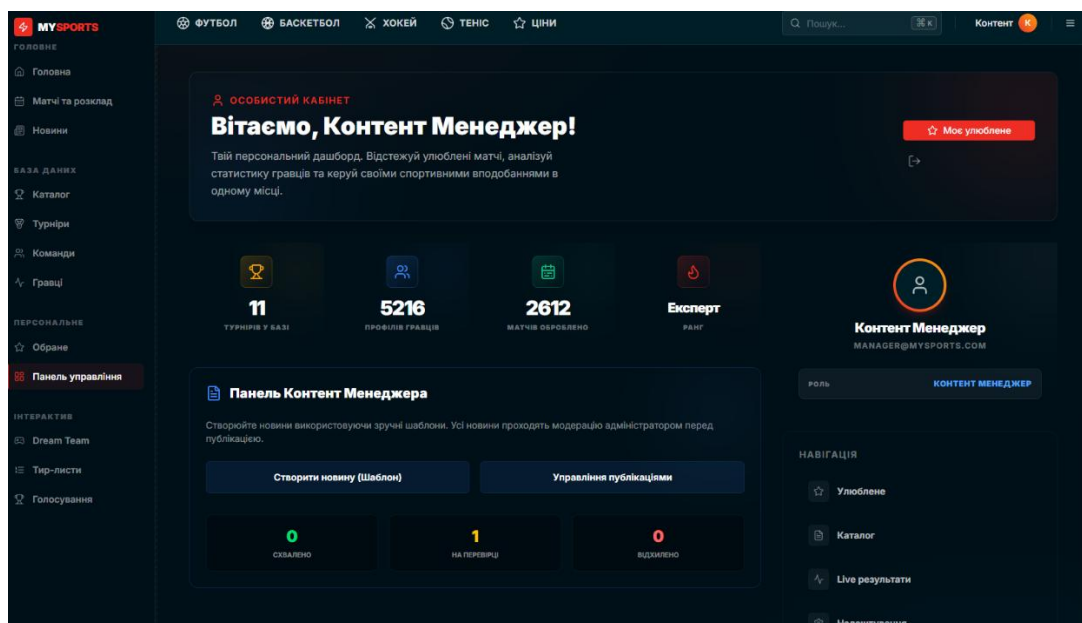


Рис. 3.6 – Персональна панель управління контент-менеджера

Глобальне керування платформою здійснюється через панель головного адміністратора, за допомогою якої забезпечується розширений доступ до всіх підсистем проекту. Адміністративний блок містить інструментарій для управління розкладом матчів, редагування критичної статистики, контролю облікових записів користувачів та кінцевої модерації новин. Окремо виділено зону системних задач, де в режимі реального часу генеруються сповіщення про наявність матеріалів, що очікують на перевірку, що гарантує швидке реагування адміністративного персоналу.

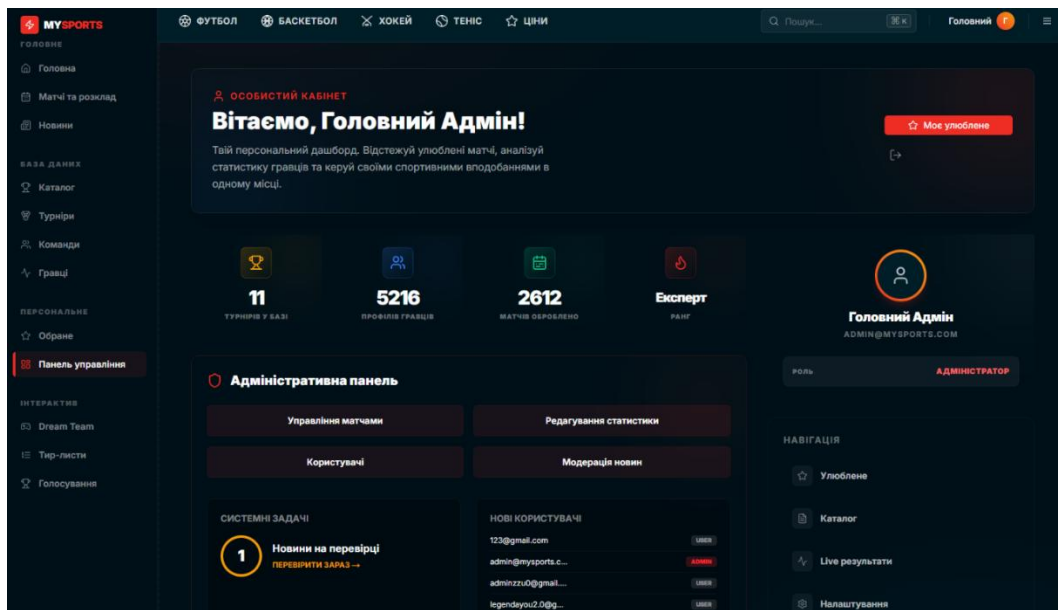


Рис. 3.7 – Глобальна панель управління головного адміністратора

Ключовим інтерактивним модулем платформи є конструктор віртуальної футбольної команди. Інтерфейс стилізовано під тактичне поле, у межах якого передбачено позиційні слоти для вибору гравців відповідно до обраної схеми розстановки. На верхній панелі налаштувань реалізовано елементи управління для зміни виду спорту, тактичної формації, а також модуль контролю ліміту трансферного бюджету. Картки обраних спортсменів містять їхні фотографії, інформацію щодо поточної вартості та індикатори ігрових позицій, що забезпечує високий рівень візуалізації процесу драфту.

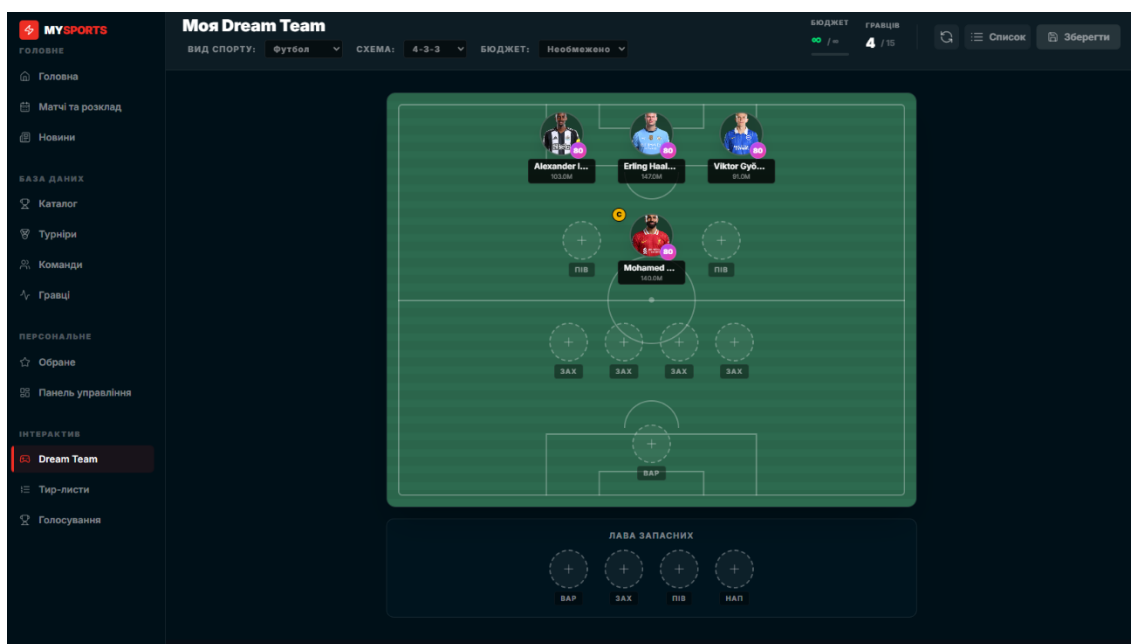


Рис. 3.8 – Інтерфейс тактичного конструктора футбольної команди

Додатковим інструментом залучення аудиторії є модуль створення персональних рейтингів. Головна сторінка цього розділу зустрічає користувача масштабним типографічним заголовком Створюй ідеальні Рейтинги. Інтерфейс пропонує зручну систему навігації з можливістю перемикання між глобальними спортивними лігами та конкретними клубами. Для швидкого доступу передбачено інтерактивний рядок пошуку та набір попередньо згенерованих карток найпопулярніших світових чемпіонатів.

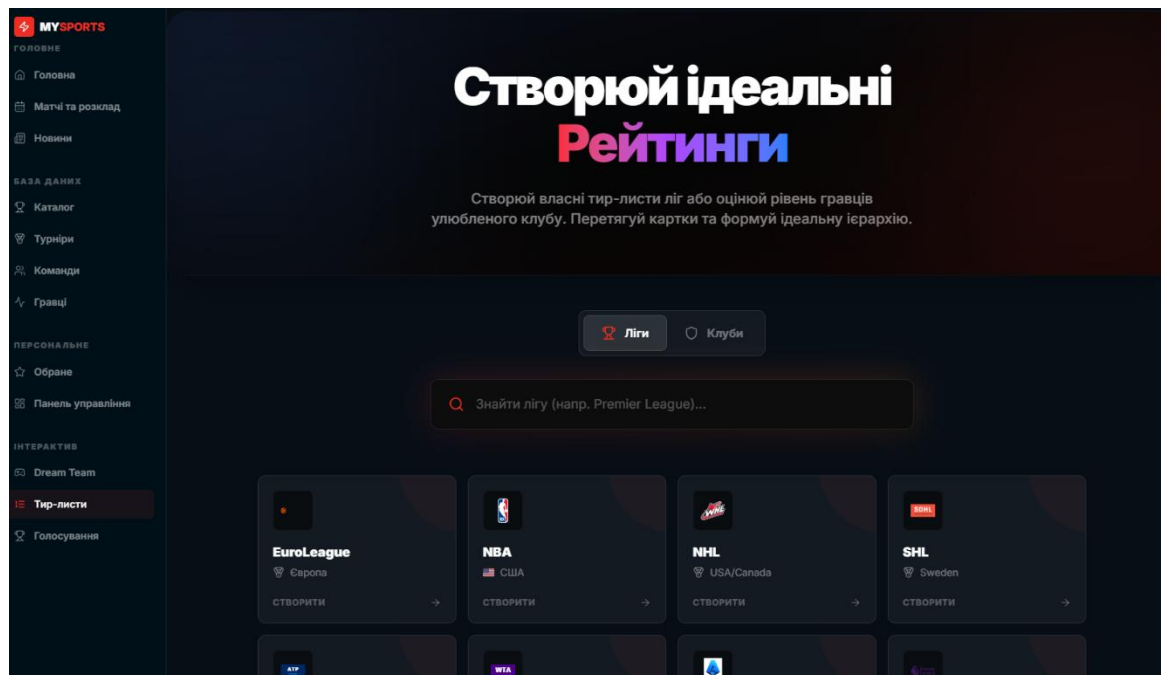


Рис. 3.9 – Головний екран модуля створення рейтингів

Безпосередній процес формування рейтингу реалізовано через гнучкий візуальний редактор тир-листів. Основна робоча область розділена на класичні рівні від найвищого S до найнижчого D, кожен з яких виділено окремим кольором. У нижній частині екрана розташовано горизонтальну панель із нерозподіленими елементами, у даному випадку логотипами команд. Користувач взаємодіє з інтерфейсом за допомогою механіки вільного перетягування об'єктів мишею або сенсорним екраном, що забезпечує високий рівень інтерактивності та плавності роботи.

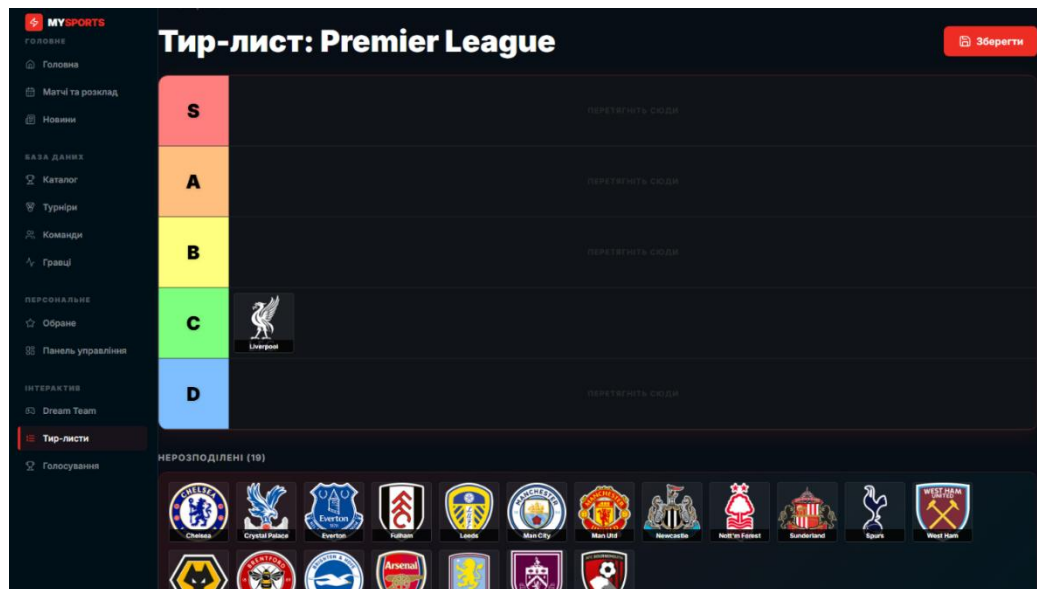


Рис. 3.10 – Інтерактивний редактор тир-листів

3.3 Тестування роботи програмного продукту

Тестування інформаційної системи MySports охоплює три взаємопов'язані рівні: модульні та інтеграційні тести серверної логіки за допомогою фреймворку Jest, модульні тести клієнтських інтерфейсів з використанням React Testing Library, а також перевірку взаємодії з реляційною базою даних через ізольоване тестове середовище PostgreSQL за допомогою інструментарію Prisma. Кожен рівень вирішує свою задачу і навмисно не дублює інші. Серверна частина покрита десятками модульних тестів, які успішно проходять під час кожної автоматизованої збірки. Клієнтська частина містить тестові набори, що охоплюють ключові інтерактивні компоненти, такі як тактичний конструктор та сторінки авторизації.

З метою забезпечення відмовостійкості та безпеки програмного комплексу, процес верифікації було чітко класифіковано на модульне (Unit) та інтеграційне тестування. Модульні тести, реалізовані з використанням фреймворку Jest, ізольовано перевіряють бізнес-логіку системи (наприклад, калькуляцію спортивної статистики). Інтеграційні тести перевіряють роботу REST API маршрутів у зв'язці з тестовим середовищем бази даних.

Окрему увагу приділено тестуванню контуру безпеки платформи. Оскільки система використовує бібліотеку NextAuth для stateless-автентифікації на базі JWT-токенів (JSON Web Tokens), тестові сценарії API-маршрутів залучають

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		85

механізм jest.Mock для імітації поведінки функції getSession. Це дозволяє ефективно перевіряти коректність відхилення неавторизованих запитів (обробка статусу 401 Unauthorized) без необхідності ініціалізації реальної бази даних сесій.

Для систематизації процесу верифікації та підтвердження функціоналу розроблено матрицю відповідності вимогам (Traceability Matrix), яку наведено у таблиці 3.5.

Таблиця 3.5

Матриця відповідності вимогам (Traceability Matrix)

Функціональна вимога	Класифікація тесту	Інструментарій	Очікуваний результат (Assert)
Валідація JWT-авторизації	Інтеграційне	Jest + Supertest	Отримання HTTP 200 та валідного об'єкта сесії
Розрахунок балів фентезі-команди	Модульне (Unit)	Jest	Точний математичний збіг суми балів
Обмеження доступу без токена	Інтеграційне	Supertest	Відхилення запиту з кодом 401 Unauthorized

```

PASS  __tests__/sports-domain.test.ts
PASS  __tests__/domain/liqpay.test.ts
-|-----|-----|-----|-----|-----|
| % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s |
-|-----|-----|-----|-----|-----|
|   91.89 |    28.57 |     100 |   91.89 |                   |
|   91.89 |    28.57 |     100 |   91.89 | 8-10               |
-|-----|-----|-----|-----|-----|

Test Suites: 2 passed, 2 total
Tests:       32 passed, 32 total
Snapshots:   0 total
Time:        19.896 s
Ran all test suites matching /domain/i.

```

Рис. 3.11 – Результати виконання модульних тестів серверної частини

Доменні тести виступають ключовою складовою серверного покриття, забезпечуючи гарантію безпомилкового виконання базових бізнес-правил. Перевірка інваріантів сутностей здійснюється без прив'язки до маршрутизації чи користувацького інтерфейсу. Зокрема, на рівні серверних дій відбувається автоматичне блокування спроб створення профілю з некоректним форматом електронної пошти або перевищення трансферного бюджету під час формування

фентезі-команди. Крім того, публікація спортивних новин є неможливою без попередньої перевірки, що повністю виключає доступ користувачів до немодерованого матеріалу. Програмні алгоритми також гарантують, що призупинений контент не передається клієнтам під час обробки HTTP-запитів. Завдяки комплексному тестуванню формується надійний рівень безпеки, який забезпечує можливість безпечного проведення рефакторингу програмного коду.

Таблиця 3.6

Тести доменного рівня

Клас	Тест	Що перевіряється
1	2	3
AuthActions Tests	Register_InvalidEmail_ShouldThrowError	Некоректний формат пошти недопустимий під час створення профілю
FantasyDraft Tests	AddPlayer_OverBudget_ShouldReject	Додавання спортсмена блокується при перевищенні трансферного ліміту
NewsModerationTests	Publish_WithoutApproval_ShouldThrow	Публікація новинної статті без резолюції адміністратора заборонена
TierListTests	AddTeam_DuplicateInTier_ShouldIgnore	Подвійне додавання одного клубу до рейтингу ігнорується системою
PremiumTests	AccessDashboard_WithoutSub_Returns403	Запит до преміальної аналітики блокується без активної підписки
AdminTests	SuspendMatch_ShouldHideFromClients	Заблокований матч ігнорується клієнтським запитом розкладу

Тестування серверних маршрутів та прикладних інтерфейсів влаштовано з використанням інструментарію симуляції запитів до середовища Next.js. Перевіряється, що алгоритм отримання турнірів правильно фільтрує контент за обраним видом спорту, а аналітичний дашборд повертає достовірні статистичні агрегації (таблиця 3.7).

Таблиця 3.7

Тестові набори серверних маршрутів

Тестовий набір	Кількість тестів	Що охоплює
auth.route.test.ts	15	Реєстрація, безпечна авторизація, валідація токенів доступу
catalog.route.test.ts	24	Отримання списків команд, фільтрація за лігою, пошук гравців
fantasy.action.test.ts	12	Синхронізація складу команди, розрахунок балів, перевірка лімітів

Тести клієнтської частини написані на базі Jest з використанням екосистеми React Testing Library. Ключовий елемент тестування захищені маршрути та глобальний стан користувача. Перевіряється, що неавтентифікований гість примусово перенаправляється на екран входу, а авторизований менеджер безперешкодно бачить свою персональну панель управління. Окремо тестується поведінка інтерфейсу: наприклад, поява червоної модальної панелі при спробі додати занадто дорогого гравця (на рисунку 3.12, таблиці 3.8).

```
PASS tests/components/Paywall.test.tsx
Paywall Component
  ✓ renders the title and description correctly (34 ms)
  ✓ contains a link to the pricing page (43 ms)
  ✓ displays the premium feature badge (4 ms)

-----
... | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
... | 100     | 100     | 100     | 100     |
... | 100     | 100     | 100     | 100     |
... | 100     | 100     | 100     | 100     |
... | 100     | 100     | 100     | 100     |
... | 100     | 100     | 100     | 100     |
-----
Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:  0 total
```

Рис. 3.12 – Результати виконання тестів клієнтської частини

Таблиця 3.8

Тестові файли клієнтської частини

Файл	Що перевіряється
ProtectedRoute.test.tsx	Захист маршрутів між анонімним станом та авторизованим профілем
SportSelector.test.tsx	Візуалізація списку видів спорту та збереження активного вибору
DraftPitch.test.tsx	Логіка перетягування карток гравців на тактичне поле команди
Paywall.test.tsx	Рендеринг екрана підписки та блокування доступу до преміуму
useSession.test.ts	Робота глобального стану авторизації та безпечне збереження сесії

Інтеграційні тести з базою даних є останнім рівнем надійності. Інструментарій Prisma при кожному запуску тестового циклу підключається до ізольованої тестової бази PostgreSQL та застосовує актуальні схеми міграцій. Кожен тест отримує чисту транзакцію, а залишки від попередніх операцій автоматично відкочуються і не впливають на результат. Перевіряється збереження складних реляційних зв'язків між турнірами та командами, а також коректність роботи складних запитів для збору персональної статистики.

```

PASS  __tests__/domain/liqpay.test.ts
PASS  __tests__/sports-domain.test.ts
PASS  __tests__/components/Paywall.test.tsx
PASS  __tests__/integration/schedule.test.ts
...  |-----|-----|-----|-----|-----|
...  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s |
...  |-----|-----|-----|-----|-----|
...  | 79.82   | 35.71    | 62.5     | 79.82   |                   |
...  | 11.11   | 100      | 0        | 11.11   | 6-45              |
...  | 100     | 100      | 100      | 100     |                   |
...  | 100     | 100      | 100      | 100     |                   |
...  | 94.73   | 30.76    | 66.66    | 94.73   |                   |
...  | 91.89   | 28.57    | 100      | 91.89   | 8-10              |
...  | 95.55   | 20       | 33.33    | 95.55   | 85-86,89-90       |
...  | 100     | 100      | 100      | 100     |                   |
...  |-----|-----|-----|-----|-----|
Test Suites: 1 skipped, 4 passed, 4 of 5 total
Tests:       2 skipped, 36 passed, 38 total
Snapshots:   0 total
Time:        3.779 s

```

Рис. 3.13 – Результати виконання інтеграційних тестів та покриття коду

Зведені дані щодо обраних інструментів та результатів покриття коду зібрано нижче (табл. 3.8).

Таблиця 3.9

Загальне покриття тестами

Рівень	Інструмент	Результат
Бізнес-логіка застосунку	Jest	Пройдено
Серверні маршрути	Jest	Пройдено
Компоненти клієнта	Jest та React Testing Library	Пройдено
Інтеграція з БД	Prisma Client та PostgreSQL Test DB	Пройдено

Загалом для перевірки працездатності проєкту написано великий масив автоматизованих тестів. Такий комбінований підхід дозволив протестувати систему з усіх боків. Серверна частина перевірялася фреймворком Jest. Щоб тестувати запити ізольовано від хмарного продакшен-середовища, але максимально наближено до реальності, залежності опиралися на окремий ізольований екземпляр бази PostgreSQL. На клієнтській стороні застосовано зв'язку Jest та Testing Library. Цей набір перевірів правильність візуалізації компонентів, роботу інтерактивних конструкторів команди та надійність захисту сторінок. Оскільки під час прогону жоден тест не видав критичної помилки, поточну версію інформаційної системи можна вважати стабільною та готовою до експлуатації.

Висновки до третього розділу

Програмний комплекс MySports розгортається в межах сучасної хмарної інфраструктури. Архітектуру вебплатформи побудовано на основі єдиного монорепозиторію фреймворку Next.js, який гармонійно поєднує реактивну клієнтську частину та швидкісні серверні дії для безпечної обробки бізнес-логіки. Для надійного збереження масивів спортивної статистики залучено безсерверну реляційну базу даних PostgreSQL від хмарного провайдера Neon, тоді як для миттєвого оновлення результатів розгорнуто підсистему кешування на базі Redis. З метою автоматизації та прискорення випуску оновлень налаштовано конвеєр безперервної інтеграції за допомогою інструментарію GitHub Actions. Кожен коміт до головної гілки репозиторію автоматично ініціює лінтерну перевірку якості коду, запуск модульних тестів та фінальну збірку проєкту. Після успішного проходження всіх етапів пайплайну нова версія системи миттєво розгортається на глобальній хостинговій платформі Vercel. Для забезпечення комфортної роботи розробника на локальній машині передбачено конфігурацію Docker Compose, що дає змогу швидко розгортати ізольовані екземпляри баз даних.

На боці клієнтського застосунку реалізовано суворе розмежування рівнів доступу. Графічний інтерфейс та доступний функціонал динамічно адаптуються залежно від адміністративної ролі або наявності преміальної підписки. Взаємодія базового авторизованого профілю відбувається переважно в межах публічного простору з метою моніторингу глобального каталогу, перегляду розкладу матчів та формування персональної команди мрії на інтерактивному тактичному полі. Натомість для потреб модерації та адміністрування передбачено використання спеціалізованої закритої вебпанелі, яка забезпечує інструментарій для глобального управління турнірами, валідації новин та редагування критично важливої спортивної статистики. Завдяки такому архітектурному підходу екосистему проєкту логічно розділено на зручне середовище для споживання контенту та захищену зону для професійного управління.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

Висока якість програмного коду та загальна стабільність архітектури підтверджуються багаторівневим автоматизованим тестуванням. Серверна логіка застосунку покрита вичерпними модульними тестами на базі фреймворку Jest для суворої перевірки виконання бізнес-правил, серед яких валідація трансферних бюджетів та захист закритих маршрутів. На клієнтському рівні набір тестів контролює коректність алгоритмів рендерингу тактичних конструкторів та поведінку інтерактивних елементів із використанням екосистеми React Testing Library. Критично важливим етапом роботи виступає інтеграційне тестування за допомогою інструментарію Prisma. Воно дає змогу перевірити реальне виконання запитів у тестовому середовищі бази PostgreSQL без жодних ризиків для виробничих даних. Увесь масив перевірок проходить успішно, що беззаперечно доводить надійність обраних технологічних рішень та повну готовність інформаційної системи до високих експлуатаційних навантажень.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Результатом виконання дипломної роботи є створена інформаційна екосистема MySports. Система призначена для глибокого моніторингу спортивних даних та інтерактивної взаємодії користувачів за допомогою віртуального конструктора складу. Вона переводить у сучасний цифровий формат увесь цикл аналітики: від базового ознайомлення зі статистикою турнірів у глобальному каталозі до фінального створення персональних рейтингів та гейміфікованої участі у фентезі-драфтах.

Огляд існуючого програмного забезпечення виявив низку проблем у цій ніші. Наявні на ринку аналоги або є надмірно монетизованими з агресивною рекламою, або мають застарілий та перевантажений інтерфейс, який ускладнює навігацію під час аналізу великих масивів спортивної статистики. Крім того, більшість розглянутих сервісів не має гнучкого механізму, що об'єднував би стрічку новин, календар трансляцій та інтерактивні ігрові модулі в єдину платформу. Це стало головним аргументом для розробки власного комплексного рішення.

Архітектура проєкту базується на сучасних патернах із використанням монорепозиторного підходу фреймворку Next.js. В основу бізнес-логіки покладено сувору реляційну модель на базі системи управління базами даних PostgreSQL. Хмарна інфраструктура надійно контролює збереження прогресу, синхронізацію трансферних бюджетів команд та захищає систему від некоректних дій, наприклад, спроб штучного перевищення фінансових лімітів клубу, незалежно від джерела мережевого запиту. Завдяки сучасному підходу до розгортання, релізи нових версій клієнтського застосунку та інтегрованої адміністративної панелі виконуються повністю автоматично через налаштований конвеєр безперервної інтеграції.

Головна практична цінність створеної системи полягає у формуванні єдиного централізованого спортивного хабу. Забезпечено безшовний досвід роботи зі статистикою у зручному адаптивному форматі з наданням доступу до розкладу матчів, модулів голосування та інтерактивного тактичного

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						92
Змн.	Арк.	№ докум.	Підпис	Дата		

поля для формування команди мрії. Водночас на рівні адміністрування реалізовано можливість у реальному часі публікувати новини, редагувати результати матчів та керувати рівнями доступу через спеціалізовану вебпанель, без необхідності розгортання оновлень програмного коду на сервері. Стабільність платформи підтверджено практично: для перевірки надійності розроблено комплекс автоматизованих тестів, які успішно покривають бізнес-правила доменних моделей, інтеграційний рівень серверних маршрутів та візуальні компоненти клієнтського інтерфейсу.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						93
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Next.js: The React Framework for the Web : офіційна документація. URL: <https://nextjs.org> (дата звернення: 13.06.2026).
2. Node.js: JavaScript runtime built on Chrome's V8 JavaScript engine : офіційна документація. URL: <https://nodejs.org> (дата звернення: 13.06.2026).
3. TypeScript: Typed JavaScript at Any Scale : офіційна документація. URL: <https://www.typescriptlang.org> (дата звернення: 13.06.2026).
4. PostgreSQL: The World's Most Advanced Open Source Relational Database : офіційна документація. URL: <https://www.postgresql.org> (дата звернення: 13.06.2026).
5. Perin C., Vuillemot R., Stolper C. D., Stasko J. T., Wood J., Carpendale S. State of the Art of Sports Data Visualization. Computer Graphics Forum. 2018. Vol. 37, No 3. P. 663–686. DOI: 10.1111/cgf.13447.
6. Spearman W. Beyond Expected Goals. Proceedings of the 12th MIT Sloan Sports Analytics Conference. 2018. P. 1–17.
7. FIFA Football Language : офіційний ресурс. URL: <https://www.fifa.com/technical/football-language> (дата звернення: 13.06.2026).
8. Lucey P., Bialkowski A., Carr P., Morgan S., Matthews I., Sheikh Y. Representing and Discovering Adversarial Team Behaviors Using Player Roles. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR. 2013. P. 2706–2713.
9. Chung L., Nixon B. A., Yu E. Non-functional requirements in software engineering : monograph. Boston : Springer, 2000. 476 p.
10. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.
11. Redux: A Predictable State Container for JS Apps : офіційна документація. URL: <https://redux.js.org> (дата звернення: 13.06.2026).
12. Dinu V., Nadkarni P. Guidelines for the effective use of entity-attribute-value modeling for biomedical databases. International Journal of Medical

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						94
Змн.	Арк.	№ докум.	Підпис	Дата		

- Informatics. 2007. Vol. 76, No 11. P. 769–779. DOI: 10.1016/j.ijmedinf.2006.09.023.
13. Survey on NoSQL database / J. Han, E. Hailong, G. Le, J. Jian. Proceedings of the 6th International Conference on Pervasive Computing and Applications. 2011. P. 363–366. DOI: 10.1109/ICPCA.2011.6106531.
14. MDN Web Docs: Window.localStorage : офіційна документація. URL: <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage> (дата звернення: 13.06.2026).
15. Codd E. F. A relational model of data for large shared data banks. Communications of the ACM. 1970. Vol. 13, No 6. P. 377–387. DOI: 10.1145/362384.362685.
16. Google Cloud: Cloud Computing Services : офіційна документація. URL: <https://cloud.google.com> (дата звернення: 13.06.2026).
17. NextAuth.js: Authentication for Next.js : офіційна документація. URL: <https://next-auth.js.org> (дата звернення: 13.06.2026).
18. Cloudinary: Image and Video Upload, Storage, Optimization : електронний ресурс. URL: <https://cloudinary.com> (дата звернення: 13.06.2026).
19. Vercel: Develop. Preview. Ship : офіційний сайт. URL: <https://vercel.com> (дата звернення: 13.06.2026).
20. Cloudflare: The Web Performance & Security Company : електронний ресурс. URL: <https://www.cloudflare.com> (дата звернення: 13.06.2026).
21. How do software developers use GitHub actions to automate their workflows? / N. Kinsman, M. Wessel, M. A. Gerosa, C. Treude. Empirical Software Engineering. 2021. Vol. 26, No 4. P. 1–35. DOI: 10.1007/s10664-021-09956-1.
22. Bierman G., Abadi M., Torgersen M. Understanding typescript. European Conference on Object-Oriented Programming, ECOOP. 2014. P. 257–281. DOI: 10.1007/978-3-662-44202-9_11.
23. React: The library for web and native user interfaces : офіційна документація. URL: <https://react.dev> (дата звернення: 13.06.2026).

24. Tilkov S., Vinoski S. Node.js: Using JavaScript to build high-performance network programs. IEEE Internet Computing. 2010. Vol. 14, No 6. P. 80–83. DOI: 10.1109/MIC.2010.145.
25. Express: Fast, unopinionated, minimalist web framework for Node.js : електронний ресурс. URL: <https://expressjs.com> (дата звернення: 13.06.2026).
26. Visual Studio Code: Code editing. Redefined : офіційна документація. URL: <https://code.visualstudio.com> (дата звернення: 13.06.2026).
27. PlantUML: Open-source tool that uses simple textual descriptions to draw UML diagrams : електронний ресурс. URL: <https://plantuml.com> (дата звернення: 13.06.2026).
28. Chacon S., Straub B. Pro Git. 2nd ed. New York : Apress, 2014. 456 p.
29. Neon: Serverless Postgres built for the cloud : офіційна документація. URL: <https://neon.tech> (дата звернення: 13.06.2026).
30. LiqPay API Reference : офіційна документація. URL: <https://www.liqpay.ua/documentation/api> (дата звернення: 13.06.2026).
31. Hardt D. The OAuth 2.0 Authorization Framework : RFC 6749. Internet Engineering Task Force. 2012. 76 p. URL: <https://datatracker.ietf.org/doc/html/rfc6749> (дата звернення: 13.06.2026).
32. Redux Toolkit: The official, opinionated, batteries-included toolset for efficient Redux development : офіційна документація. URL: <https://redux-toolkit.js.org> (дата звернення: 13.06.2026).
33. Garousi V., Felderer M. Worlds apart: industrial and academic focus areas in software testing. IEEE Software. 2017. Vol. 34, No 5. P. 38–45. DOI: 10.1109/MS.2017.3571565.
34. Jest: Delightful JavaScript Testing : офіційна документація. URL: <https://jestjs.io> (дата звернення: 13.06.2026).
35. Playwright: Fast and reliable end-to-end testing for modern web apps : офіційна документація. URL: <https://playwright.dev> (дата звернення: 13.06.2026).

36. React Testing Library : офіційна документація. URL: <https://testing-library.com/docs/react-testing-library/intro> (дата звернення: 13.06.2026).
37. Testcontainers for Node.js : офіційна документація. URL: <https://testcontainers.com> (дата звернення: 13.06.2026).
38. Testing with Prisma : офіційна документація. URL: <https://www.prisma.io/docs/guides/testing> (дата звернення: 13.06.2026).
39. Jones M., Bradley J., Sakimura N. JSON Web Token, JWT : RFC 7519. Internet Engineering Task Force. 2015. 30 р. URL: <https://datatracker.ietf.org/doc/html/rfc7519> (дата звернення: 13.06.2026).
40. GitHub Actions Documentation : офіційний сайт. URL: <https://docs.github.com/en/actions> (дата звернення: 13.06.2026).
41. Tailwind CSS: Rapidly build modern websites without ever leaving your HTML : офіційна документація. URL: <https://tailwindcss.com> (дата звернення: 13.06.2026).
42. Merkel D. Docker: lightweight linux containers for consistent development and deployment. Linux Journal. 2014. Vol. 2014, No 239. P. 2. URL: <https://dl.acm.org/doi/10.5555/2600239.2600241> (дата звернення: 13.06.2026).

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						98
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток А

Діаграма варіантів використання



Рис. А.1 – Діаграма варіантів використання

Додаток Б

Діаграма компонентів

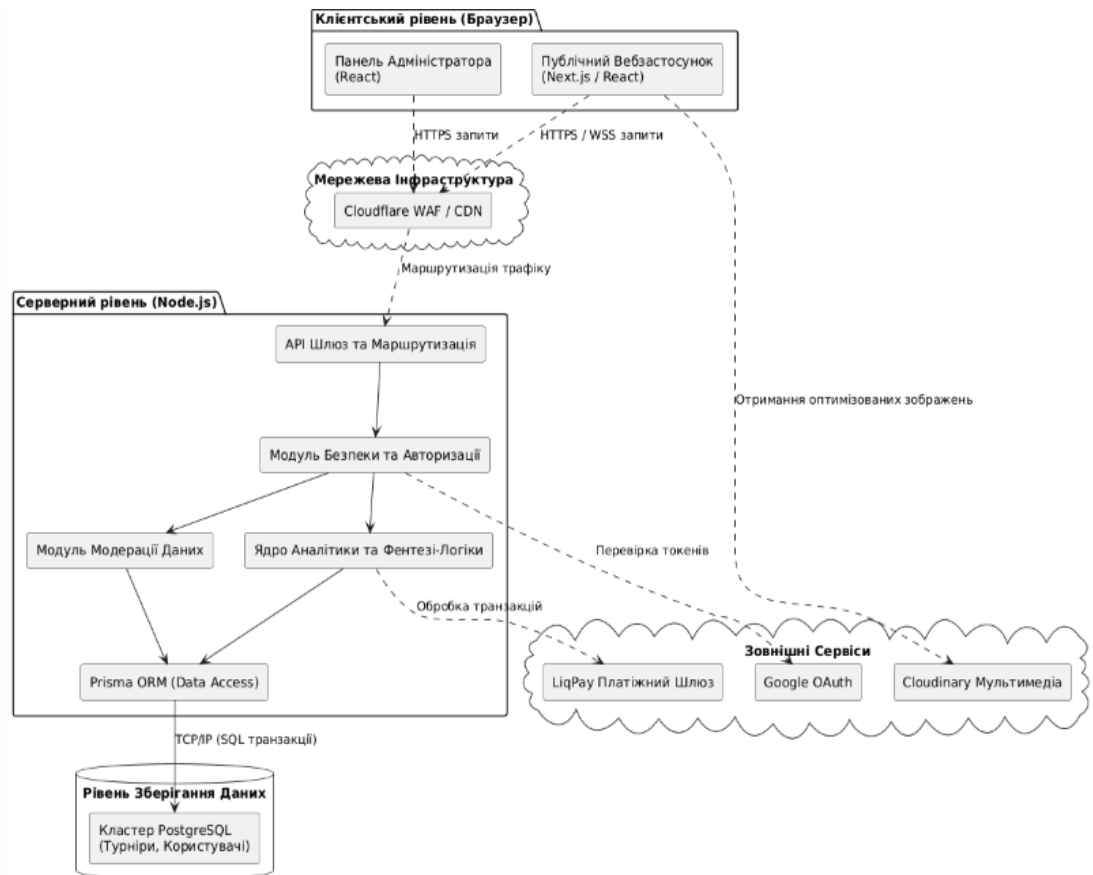


Рис. Б.1 – Діаграма компонентів

Додаток В

Діаграма шару застосунку (Application Layer)

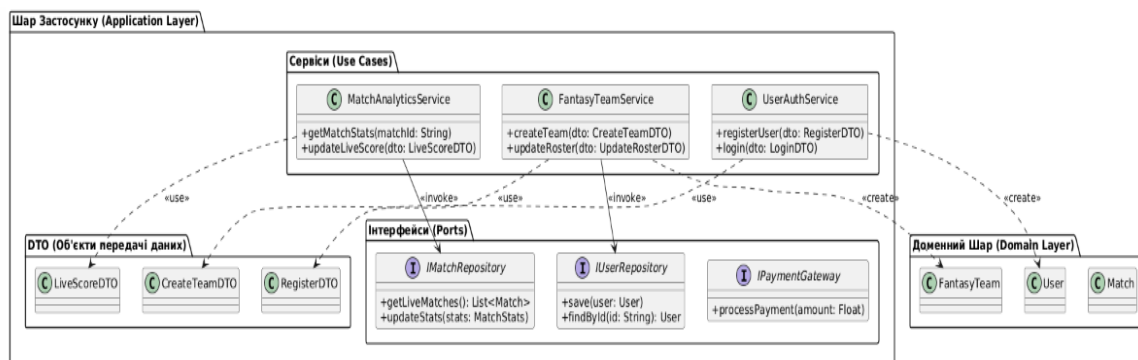


Рис. В.1 – Діаграма шару застосунку

Додаток Г

Діаграма шару інфраструктури (Infrastructure Layer Diagram)

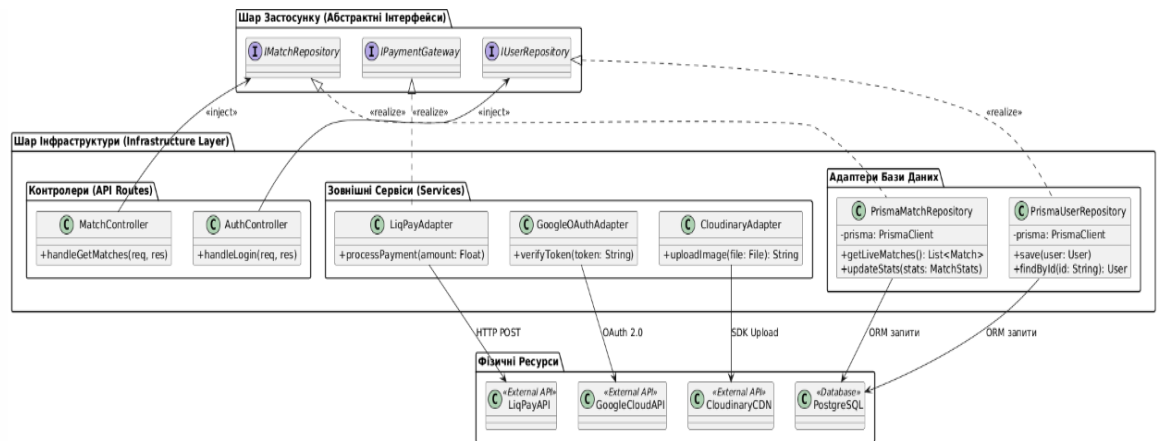


Рис. Г.1 – Діаграма шару інфраструктури

Додаток Д

Діаграма послідовності формулювання аналітичного дашборда

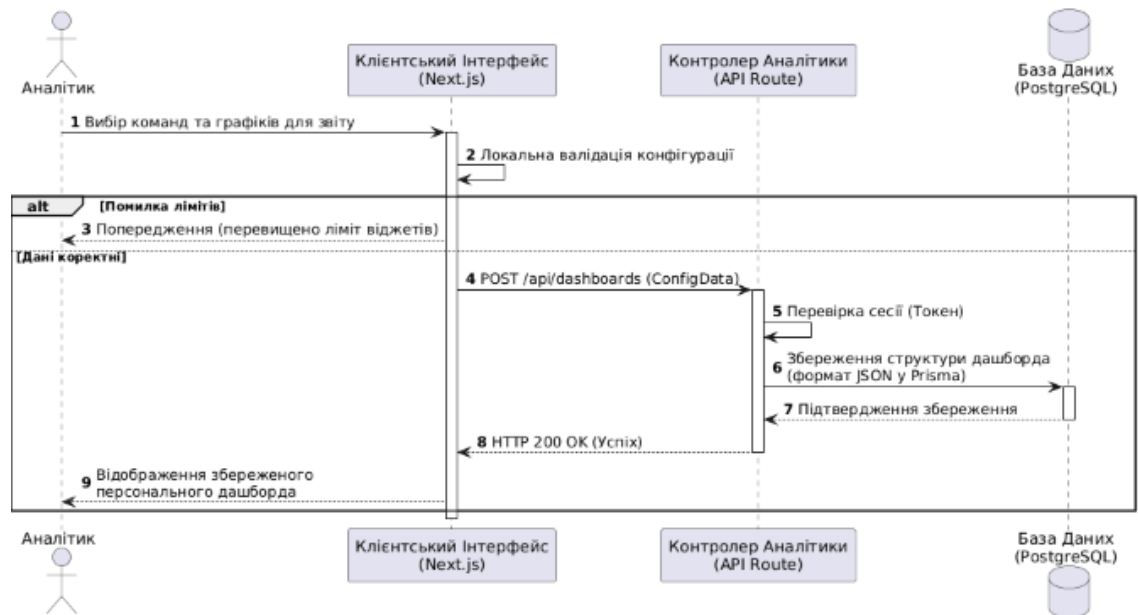


Рис. Д.1 – Діаграма послідовності створення дашборду

Додаток Е

Діаграма послідовності модерації розкладу турніру

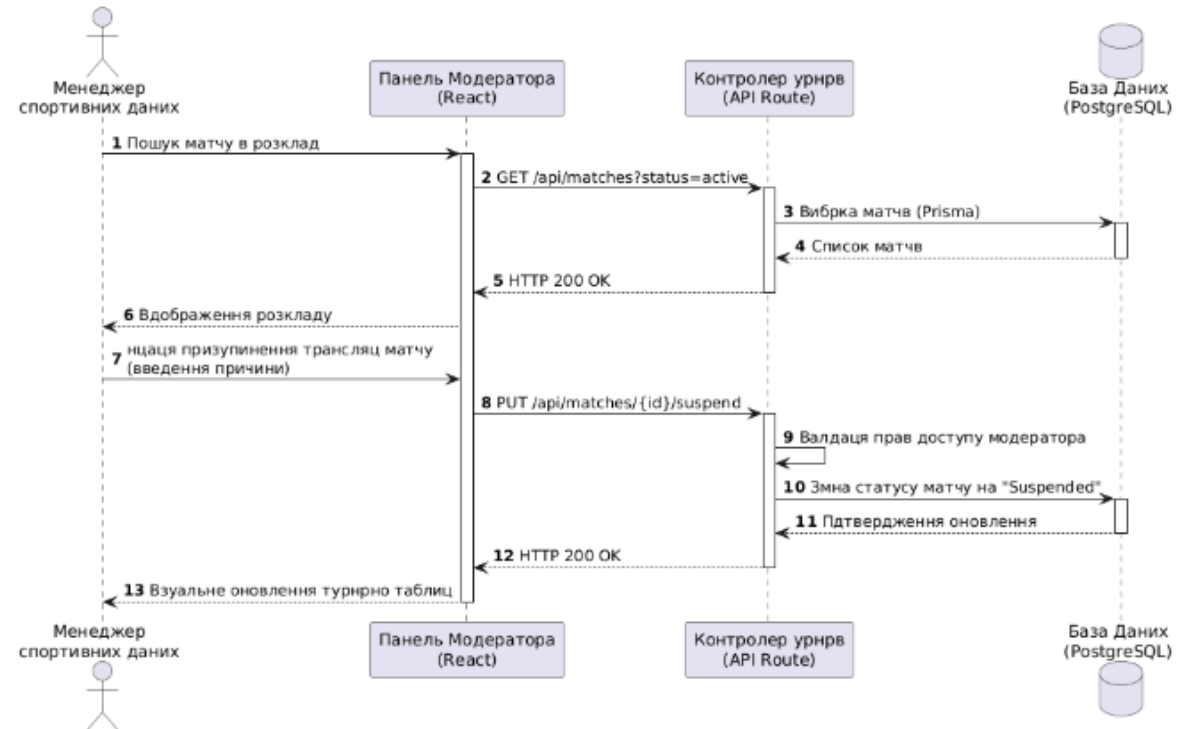


Рис. Е.1 – Діаграма послідовності призупинення матчв

Фрагменти коду серверної частини

Ж.1 Контракт створення фентезі-команди (DTO)

```
export interface CreateFantasyTeamDto
{
    userId: string;

    name: string;

    formation: string;

    captainId: string;

    players: Array<{
        playerId: string;

        position: string;

        isStarter: boolean;
    }>;
}
```

Ж.2 Обробник (Service) логіки збереження команди (Prisma ORM)

```
import { prisma } from '@lib/prisma';
import { ValidationError } from '@lib/exceptions';
import { CreateFantasyTeamDto } from './types';

export class FantasyTeamService {
    public async createTeam(data: CreateFantasyTeamDto) {
        // Валідація бізнес-правил предметної області
        if (!data.name || data.players.length !== 11) {
            throw new ValidationError('Команда повинна мати назву та містити рівно 11 гравців стартового складу.');
```

```
        }

        // Створення транзакції збереження в базі даних (Prisma)
        const newTeam = await prisma.fantasyTeam.create({
            data: {
                userId: data.userId,
                name: data.name,
                formation: data.formation,
                captainId: data.captainId,
                players: {
                    create: data.players.map(player => ({
                        playerId: player.playerId,
                        position: player.position,
                        isStarter: player.isStarter
                    })))
            }
        })
    }
}
```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						105
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    },
    include: {
      players: true
    }
  });

  return newTeam;
}
}

```

Ж.3 Валідатор HTTP-запиту на створення команди (схема Zod)

```

import { z } from 'zod';

export const createFantasyTeamSchema = z.object({
  name: z.string().min(3, 'Назва має містити щонайменше 3 символи'),
  formation: z.string().regex(/^\d-\d-\d$/, 'Формат формації має бути, наприклад, 4-4-2'),
  captainId: z.string().uuid('Недійсний ID капітана команди'),
  players: z.array(
    z.object({
      playerId: z.string().uuid('Недійсний ID гравця'),
      position: z.string().min(1, 'Позиція є обов\'язковою'),
      isStarter: z.boolean().default(true),
    })
  ).length(11, 'Необхідно додати рівно 11 гравців'),
});

```

Ж.4 Метод маршруту (App Router) для обробки POST-запиту

```

import { NextResponse } from 'next/server';
import { getServerSession } from 'next-auth/next';
import { authOptions } from '@/lib/auth';
import { FantasyTeamService } from '@/services/FantasyTeamService';
import { createFantasyTeamSchema } from '@/schemas/fantasy';

const teamService = new FantasyTeamService();

export async function POST(req: Request) {
  try {
    // Перевірка авторизації через NextAuth
    const session = await getServerSession(authOptions);
    if (!session?.user?.id) {
      return NextResponse.json({ error: 'Необхідна авторизація' }, { status: 401 });
    }

    const body = await req.json();

    // Суворі валідації схеми запиту
    const validatedData = createFantasyTeamSchema.parse(body);

    const teamData = { ...validatedData, userId: session.user.id };
    const createdTeam = await teamService.createTeam(teamData);

    return NextResponse.json({
      success: true,
      data: createdTeam,
    }, { status: 201 });
  } catch (error: any) {

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						106
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (error.name === 'ZodError') {
      return NextResponse.json({ errors: error.errors }, { status: 400 });
    }
    return NextResponse.json({ error: error.message }, { status: 500 });
  }
}

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						107
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестовий клас для перевірки доменного рівня (Unit Tests з Jest)

```

import { FantasyTeamService } from '@services/FantasyTeamService';
import { ValidationError } from '@lib/exceptions';
import { prismaMock } from '@test-utils/prismaMock';

describe('FantasyTeamService - Unit Tests', () => {
  let teamService: FantasyTeamService;

  beforeEach(() => {
    teamService = new FantasyTeamService();
  });

  it('повинен викидати помилку, якщо кількість гравців не дорівнює 11', async ()
=> {
    const invalidData = {
      userId: 'user-123',
      name: 'Dream Team',
      formation: '4-4-2',
      captainId: 'player-1',
      players: [], // Порожній масив порушує бізнес-логіку фентезі-спорту
    };

    await
    expect(teamService.createTeam(invalidData)).rejects.toThrow(ValidationError);
    await expect(teamService.createTeam(invalidData)).rejects.toThrow(/рівно 11
гравців/);
  });
});

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						108
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестовий клас обробника API маршрутів (Integration Tests)

```
import { POST } from '@app/api/fantasy-teams/route';
import { getServerSession } from 'next-auth/next';

jest.mock('next-auth/next');

describe('Fantasy Team API Endpoints', () => {

  beforeEach(() => {
    (getServerSession as jest.Mock).mockResolvedValue({
      user: { id: 'user-123', name: 'Test User' }
    });
  });

  it('повинен успішно створювати нову команду при валідних даних', async () => {
    const newTeamPayload = {
      name: 'My Best XI',
      formation: '4-4-2',
      captainId: '550e8400-e29b-41d4-a716-446655440000',
      players: Array(11).fill({
        playerId: '550e8400-e29b-41d4-a716-446655440000',
        position: 'MID',
        isStarter: true
      })
    };

    const req = new Request('http://localhost:3000/api/fantasy-teams', {
      method: 'POST',
      body: JSON.stringify(newTeamPayload)
    });

    const response = await POST(req);
    const json = await response.json();

    expect(response.status).toBe(201);
  });
});
```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						109
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    expect(json.success).toBe(true);

    expect(json.data.name).toBe('My Best XI');

  });

  it('повинен повертати помилку 401 (Unauthorized) без валідної сесії', async ()
=> {

    (getServerSession as jest.Mock).mockResolvedValue(null); // Симуляція
відсутності токена

    const req = new Request('http://localhost:3000/api/fantasy-teams', {
      method: 'POST',
      body: JSON.stringify({ name: 'Test' })
    });

    const response = await POST(req);
    expect(response.status).toBe(401);

  });
});

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						110
Змн.	Арк.	№ докум.	Підпис	Дата		