

БАКАЛАВРСЬКА РОБОТА

КБР.ІСТ – 16.00.00.000 ПЗ

Група ІСТ-22-1

Роман СЕМЕНЮК

2026

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Факультет інформаційних технологій
Кафедра інформаційно-телекомунікаційних технологій та систем

Семенюк Роман Олегович
(прізвище, ім'я, по батькові)

УДК 004.41:004.774:794.8
(індекс)

БАКАЛАВРСЬКА РОБОТА

Проектування та розробка інформаційної системи управління контентом онлайн-платформи у сфері кіберспорту.

(назва роботи)

Інформаційні системи та технології

(назва освітньої програми)

126 – Інформаційні системи та технології

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня Семенюк Р. О.
(підпис, ініціали та прізвище здобувача)

Науковий керівник к.т.н., доцент Волинський О. І.
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
В. о. завідувача кафедри

Заміховська О. Л.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківський національний технічний університет нафти і газу

(повне найменування закладу вищої освіти)

Факультет інформаційних технологій

Кафедра інформаційно-телекомунікаційних технологій та систем

Освітній рівень бакалавр

Спеціальність 126 – Інформаційні системи та технології

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТТС

Заміховська О. Л.

« ____ » _____ 2026 року

**ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ**

Семенюку Роману Олеговичу

(прізвище, ім'я, по батькові)

1. Тема роботи Проектування та розробка інформаційної системи управління контентом онлайн-платформи у сфері кіберспорту
керівник роботи, доц. каф. Волинський О. І.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від "07" квітня 2026 року № 163/З

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи Спеціалізована інформаційна система на базі React із серверною частиною Node.js та фреймворком NestJS; реляційна база даних PostgreSQL із застосуванням Prisma ORM; управління медіаконтентом та динамічне генерування турнірних сіток; каскадний перерахунок ігрової KDA-статистики матчів; JWT-автентифікація та криптографічне хешування паролів; розмежування доступу на основі багаторівневої рольової моделі (RBAC); підтримка принципів транзакційної цілісності (ACID)

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити)

1 Аналіз аналогів та постановка завдання; 2 Розробка структури реляційної бази даних; 3 Програмна реалізація функціоналу, розробка інтерфейсу користувача веб-орієнтованої інформаційної системи.

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

БР.ІСТ - 16.00.00.000 С1 Діаграма класів;

БР.ІСТ - 16.00.00.000 С2 С4 контекстна діаграма

6. Консультації розділів роботи

Розділ	Консультант	Підпис, дата
1	Волинський О.І	28.04.2026
2	Волинський О.І	15.05.2026
3	Волинський О.І	25.05.2026

7. Дата видачі завдання – 10.04.2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів	Примітка
1.	Аналіз предметної області кіберспортивних медіа та огляд існуючих систем управління контентом	01.04.2026 - 08.04.2026 р.	Виконано
2.	Моделювання бізнес-процесів та формування вимог до системи	01.04.2026 - 08.04.2026 р.	Виконано
3.	Архітектурне проєктування та розробка структури бази даних	08.04.2026 - 15.04.2026 р.	Виконано
4.	Програмна реалізація серверної частини (Node.js, NestJS REST API) та JWT-автентифікації	5.05.2026 - 10.05.2026 р.	Виконано
5.	Розробка клієнтського інтерфейсу на базі React	12.05.2026 - 18.05.2026 р.	Виконано
6.	Реалізація багаторівневої рольової моделі (RBAC) та алгоритмів перерахунку KDA-статистики	19.05.2026 - 25.05.2026 р.	Виконано
7.	Впровадження механізмів безпеки (валідація DTO, CORS), тестування та інтеграція компонентів	25.05.2026 - 29.05.2026 р.	Виконано
8.	Оформлення розрахунково-пояснювальної записки та підготовка графічного матеріалу	01.06.2026 - 03.06.2026 р.	Виконано
9.	Подання роботи на рецензування та підготовка до захисту	04.06.2026 - 08.06.2026 р.	

Студент

(підпис)

Семенюк Р. О.

(прізвище та ініціали)

Керівник роботи

(підпис)

Волинський О. І.

(прізвище та ініціали)

АНОТАЦІЯ

В рамках бакалаврської роботи розроблено спеціалізовану інформаційну систему управління контентом (CyberCore CMS) для кіберспортивної онлайн-платформи на базі середовища Node.js, фреймворку NestJS та інструментарію Prisma ORM.

В результаті було створено функціональний вебдодаток за архітектурою Single Page Application із рольовим доступом для адміністраторів, менеджерів турнірів, журналістів та читачі. Система дозволяє журналістам створювати публікації за допомогою вбудованого редактора та семантичного тегування, а менеджерам турнірів — динамічно генерувати турнірні сітки, вносити результати матчів та здійснювати фоновий каскадний перерахунок індивідуальної KDA-статистики гравців. Інтерфейс користувача є інтерактивним, адаптивним та має сучасну темну колірну схему.

Для реалізації back-end частини використовувалися мова TypeScript та реляційна СУБД PostgreSQL із підтримкою транзакційності, що гарантує надійність збереження пов'язаних сутностей і унеможливорює розсинхронізацію даних турнірів. Front-end частина реалізована за допомогою бібліотеки React та інструментів маршрутизації клієнтського рівня, що дозволило створити швидкодіючий кросплатформний інтерфейс без перезавантаження сторінок у браузері.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, КІБЕРСПОРТ, УПРАВЛІННЯ КОНТЕНТОМ, CMS, ТУРНІРНА СІТКА, ВЕБДОДАТОК, NESTJS, REACT, POSTGRESQL

ANNOTATION

Within the framework of the bachelor's thesis, a specialized content management information system (CyberCore CMS) for an esports online platform was developed based on the Node.js environment, the NestJS framework, and the Prisma ORM toolkit.

As a result, a functional web application was created using the Single Page Application architecture with role-based access for administrators, tournament managers, journalists, and readers. The system allows journalists to create publications using a built-in editor and semantic tagging, while tournament managers can dynamically generate tournament brackets, enter match results, and perform a background cascading recalculation of individual players' KDA statistics. The user interface is interactive, responsive, and features a modern dark color scheme.

To implement the back-end part, the TypeScript language and the PostgreSQL relational DBMS with transaction support were used, which guarantees the reliable storage of related entities and prevents the desynchronization of tournament data. The front-end part was implemented using the React library and client-side routing tools, allowing the creation of a fast cross-platform interface without page reloads in the browser.

Keywords: INFORMATION SYSTEM, ESPORTS, CONTENT MANAGEMENT, CMS, TOURNAMENT BRACKET, WEB APPLICATION, NESTJS, REACT, POSTGRESQL

ЗМІСТ

ВСТУП.....	7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА МОДЕЛЮВАННЯ БІЗНЕС ПРОЦЕСІВ.....	9
1.1 Аналіз предметної області.....	9
1.2 Узагальнена функціональна модель.....	17
1.3 Моделювання потоків даних та процесів.....	20
1.4 Моделювання бізнес процесів кіберспортивної онлайн-платформи BPMN 2.0.....	24
1.5 Формування вимог до інформаційної системи CyberCore CMS.....	28
2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА КОМПОНЕНТІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	31
2.1 UML діаграми та User Journey Map.....	31
2.2 Специфікація бази даних CyberCore CMS	38
2.3 Архітектурна специфікація CyberCore CMS	45
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ CYBERCORE CMS.....	50
3.1. Організація середовища розробки та базове налаштування технологічного стека.....	50
3.2 Програмна реалізація бізнес-логіки серверної частини	52
3.3 Розробка інтерфейсу користувача та клієнтської частини.....	55
3.4 Забезпечення інформаційної безпеки та тестування системи	58
ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62
ДОДАТКИ.....	64

					КРБ.ІСТ-16.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Проекткування та розробка інформаційної системи управління контентом онлайн-платформи у сфері кіберспорту	Літ.	Арк.	Аркушів
Розроб.		Семенюк Р. О.				Н	5	63
Перевір.		Волинський О.І						
Реценз.								
Н. Контр.		Возний А.В						
Затверд.		Заміховська О.Л				ІФНТУНГ ІСТ-22-1		

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ACID – Atomicity, Consistency, Isolation, Durability (атомарність, узгодженість, ізольованість, довговічність)

ADR – Architectural Decision Records (формат документування архітектурних рішень).

BPMN – Business Process Model and Notation (стандарт графічної формалізації та моделювання бізнес-процесів).

CMS – Content Management System (система управління контентом).

CORS – Cross-Origin Resource Sharing (механізм контролю міждомених запитів).

JWT – JSON Web Token (механізм автентифікації за допомогою токенів).

KDA – Kills, Deaths, Assists (кіберспортивна статистика: вбивства, смерті, допомоги).

ORM – Object-Relational Mapping (технологія об'єктно-реляційного відображення).

RBAC – Role-Based Access Control (рольова модель розмежування доступу).

SPA – Single Page Application (односторінковий веб-застосунок).

WYSIWYG – What You See Is What You Get (візуальний редактор тексту, де вміст відображається одразу у фінальному вигляді).

ІС – Інформаційна система.

СУБД / СКБД – Система управління (керування) базами даних.

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Індустрія кіберспорту швидко масштабується, перетворюючись на глобальну екосистему, що супроводжується експоненційним зростанням обсягів медійного, аналітичного та статистичного контенту. Сучасні кіберспортивні онлайн-платформи вимагають ефективних інструментів для динамічної обробки інформації та синхронізації взаємодії між різними групами користувачів. Використання традиційних універсальних систем управління контентом (CMS) у цій предметній галузі є неефективним через відсутність вбудованого інструментарію для структурування специфічних даних: турнірних сіток, профілів команд і гравців, розкладів матчів та ігрової аналітики. Ручне оновлення статичних сторінок та інтеграція розрізнених зовнішніх віджетів уповільнюють процеси публікації, що призводить до втрати актуальності даних та зниження залученості аудиторії. Відповідно, розробка спеціалізованої інформаційної системи управління контентом для кіберспортивної онлайн-платформи є актуальною науково-технічною задачею, вирішення якої дозволить автоматизувати бізнес-процеси медіаресурсів та забезпечити користувачів консолідованим джерелом інформації.

Метою бакалаврської роботи є системний аналіз та проєктування спеціалізованої інформаційної системи CyberCore CMS для кіберспортивної онлайн-платформи, що забезпечить автоматизацію процесів публікації та структурування специфічних даних індустрії.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати предметну область, визначити основні проблеми існуючих підходів до управління кіберспортивним контентом та сформулювати вимоги до нової системи;
- виконати системний та структурний аналіз бізнес-процесів взаємодії користувачів із медіаплатформою;

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

- розробити логічну та фізичну моделі реляційної бази даних для ефективного зберігання інформації про турніри, статті та ігрову статистику;
- спроектувати архітектуру програмного забезпечення та визначити оптимальний технологічний стек для реалізації клієнт-серверної взаємодії;
- здійснити програмну реалізацію бекенд-частини системи для обробки бізнес-логіки та фронтенд-інтерфейсів для різних категорій користувачів;
- провести функціональне тестування інформаційної системи та впровадити механізми захисту даних;

Об'єктом дослідження є процес проєктування, функціонування та архітектурної побудови інформаційних систем для управління контентом та специфічними даними в медійному середовищі.

Предметом дослідження є архітектура, моделі процесів, структура даних та інтерфейсні рішення кіберспортивної інформаційної системи CyberCore CMS.

Для вирішення поставлених завдань у роботі використано комплекс методів системного аналізу та інженерії програмного забезпечення:

- метод порівняльного аналізу;
- структурне моделювання за методологіями IDEF0, DFD, IDEF3 та стандартом BPMN 2.0 ;
- об'єктно-орієнтований підхід та мову UML;
- методи реляційного проєктування баз даних;
- методи проєктування користувацького досвіду;
- архітектурне моделювання за методологією C4;

Робота складається зі вступу, трьох основних розділів, висновків, списку використаних джерел та додатків. Загальний обсяг пояснювальної записки становить 63 сторінки основного тексту. Робота містить необхідний графічний та табличний матеріал для візуалізації проєктних рішень.

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА МОДЕЛЮВАННЯ БІЗНЕС ПРОЦЕСІВ

1.1 Аналіз предметної області

Предметна галузь дослідження охоплює сферу цифрових медіа, управління інформаційними ресурсами та агрегації динамічних даних кіберспортивної індустрії. В умовах стрімкого розвитку електронного спорту виникає об'єктивна необхідність у спеціалізованих інструментах для оперативного структурування та трансляції контенту великим аудиторіям. Практичне застосування такої системи передбачає публікацію новинних матеріалів, генерацію турнірних сіток, ведення статистичних баз даних команд і гравців, відстеження трансферів та взаємодію зі стрімінговими сервісами. Проєктована інформаційна система CyberCore CMS виступає єдиним середовищем, де неструктуровані масиви ігрової та аналітичної інформації перетворюються на зручні інтерактивні блоки для кінцевих споживачів.

Коло зацікавлених сторін проєкту включає декілька ключових груп користувачів із різними функціональними потребами. Головні редактори та журналісти профільних видань використовують систему для швидкого створення матеріалів із мінімізацією рутинних процесів макетування. Водночас кіберспортивні аналітики та організатори турнірів потребують інструментарію для ведення статистичного обліку, налаштування розкладів та адміністрування змагальних сіток. Кінцевими споживачами інформаційного продукту є вболівальники та гравці, для яких критично важливим є своєчасний доступ до консолідованих результатів і медіаматеріалів у наочному форматі.

На сучасному етапі розвитку галузі ключовою проблемою залишається розрізненість даних, за якої статистичні показники з ігрових програмних інтерфейсів, розклади із зовнішніх ресурсів та текстова аналітика функціонують ізольовано. Зазначені недоліки безпосередньо впливають на зниження залученості аудиторії, уповільнюють внутрішні процеси

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

медіакоманд та створюють суттєві обмеження для подальшого масштабування кіберспортивних онлайн-платформ.

Для формування об'єктивних вимог до проєктованої CMS було проведено аналіз провідних платформ у сфері кіберспорту, які є індустріальними стандартами щодо роботи зі специфічним контентом (статистикою, розкладами турнірів та новинами).

HLTV.org – це провідна у світі медіа платформа та найповніша база статистичних даних, присвячена дисципліні Counter Strike. Вона є глобальним стандартом для відстеження рейтингів команд та результатів матчів. Архітектурне рішення – високонавантажений Web додаток. Технологічний стек – базується на кастомних рішеннях із використанням PHP та Node.js для серверної частини, реляційних баз даних (MySQL/MariaDB) та інтенсивному використанні In memory баз даних (Redis) для швидкого кешування результатів у реальному часі [1].

1) Функціональні особливості:

- відстеження матчів у реальному часі (live score, kill feed) завдяки інтеграції з ігровими API (рис. 1.1);
- ведення глобального рейтингу команд (World Ranking) за складними математичними алгоритмами;
- агрегація новин, інтерв'ю та аналітичних матеріалів;
- розгалужений форум для взаємодії спільноти;

Team	Player	K	A	D	ADR
R: 2 - mirage	mytsai	2	0	1	100.0
	Marix	1	0	1	66.5
	Altekez	0	0	1	14.5
	L00m1	0	0	1	13.5
	k1to	0	0	1	0.0
AM	megashort	3	0	1	139.0
	virree	1	0	0	65.0
	zen	1	0	0	25.0
	robiin	0	0	1	11.0
	mogv	0	0	1	7.5

Рисунок 1.1 – Приклад відстежування матчу у реальному часі

2) Сильні та слабкі сторони:

2.1) Сильні сторони:

- велика кількість обчислюваних метрик (рис 1.2);
- висока швидкість оновлення інформації під час турнірів;
- високий рівень довіри професійної спільноти.

Statistics			
Total kills	15342	Rounds played	20477
Headshot %	36.9%	Kills / round	0.75
Total deaths	11894	Assists / round	0.11
K/D Ratio	1.29	Deaths / round	0.58
Damage / Round	75.2	Saved by teammate / round	0.09
Grenade dmg / Round	3.1	Saved teammates / round	0.12
Maps played	857	Impact rating	1.13

Рисунок 1.2 – Перелік обчислюваних метрик

2.2) Слабкі сторони:

- візуально перевантажений інтерфейс, застарілий UX дизайн;
- відсутність зручних інструментів для кастомізації контенту користувачами;

ESL Pro Tour – офіційна інформаційна платформа одного з найбільших у світі кіберспортивних турнірних операторів, орієнтована на висвітлення власної екосистеми змагань. Архітектурне рішення – сучасний Web додаток, побудований за концепцією Headless CMS (чітке розділення серверного сховища контенту та клієнтського інтерфейсу). Технологічний стек – Клієнтська частина базується на сучасних JavaScript фреймворках (React / Next.js) для забезпечення високої інтерактивності. Для управління контентом використовуються корпоративні Headless/API first рішення з хмарним збереженням медіаданих [2]

1) Функціональні особливості:

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

- інтерактивні турнірні сітки та розклади із підтримкою різних форматів (рис 1.3);
- інтеграція з Twitch/YouTube для прямої трансляції матчів безпосередньо на сторінці;
- деталізовані сторінки подій із правилами, розподілом призового фонду та інформацією про кваліфікації;

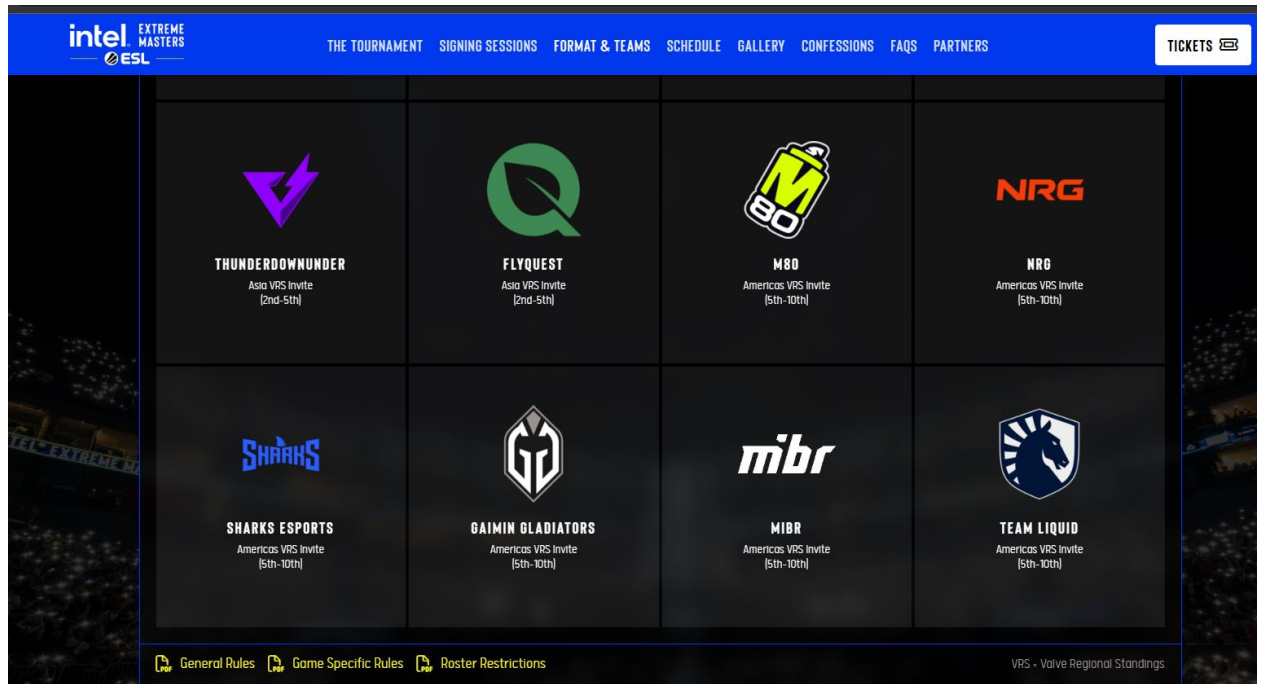


Рисунок 1.3 – Приклад турнірної сітки зі списком команд

2) Сильні та слабкі сторони:

2.1) Сильні сторони:

- високоякісний, ергономічний та сучасний візуальний дизайн;
- відмінна оптимізація для мобільних пристроїв;
- швидкість розгортання нових сторінок під конкретні івенти (Intel Extreme Masters, ESL Pro League).

2.2) Слабкі сторони:

- вузька спеціалізація платформи виключно на власних турнірах оператора;

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

- відсутність глибокої індивідуальної статистики гравців (акцент робиться на розважальний контент та розклад);

Liquipedia – найбільша у світі краудсорсингова кіберспортивна енциклопедія, яка підтримується та наповнюється самою спільнотою за принципами Вікіпедії. Архітектурне – рішення Монолітний Web додаток енциклопедичного типу. Технологічний стек – Базується на відкритому рушії MediaWiki. Використовує PHP для бекенду, реляційні бази (MySQL/MariaDB) для збереження даних та розширення Semantic MediaWiki для структурування бази знань. Для кешування застосовуються Varnish та Memcached [3].

1) Функціональні особливості:

- ведення історії всіх значущих кіберспортивних турнірів;
- детальне відстеження трансферів гравців (рис 1.4);
- відкрита система редагування з багаторівневою модерацією;
- використання складних шаблонів для генерації турнірних таблиць;

Latest Player Transfers (edit)				
Date	Player	Old		New
2026-05-17	Aureo		⇌	None
2026-05-17	Brylu PeTeRoOo	None	⇌	
2026-05-17	Frontsiderr SpavaQ		⇌	None
2026-05-15	Levi_3R ciordaleless kickdown Zaha zeking-		⇌	
2026-05-15	BrazilBob		⇌	None
2026-05-14	Walco		⇌	
2026-05-14	KmZ		⇌	None

Рисунок 1.4 – Приклад відстеження трансферів гравців

2) Сильні та слабкі сторони:

2.1) Сильні сторони:

- найбільш вичерпна база даних про історію команд, навіть у нижчих (Tier 3/Tier 4) дивізіонах;
- можливість спільного редагування, що забезпечує швидку появу інформації про зміни у складах команд;
- відкритість даних та структурованість завдяки семантичним тегам;

2.2) Слабкі сторони:

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

- високий поріг входу для нових авторів контенту;
- застарілий текстовий формат подачі інформації, який не підходить для публікації оперативних новин чи медіа статей;
- відсутність повної автоматизації live даних (більшість результатів оновлюється модераторами вручну);

Об'єктом розробки виступає спеціалізована інформаційна система управління контентом CyberCore CMS. Її основне призначення полягає у забезпеченні редакцій кіберспортивних медіа, турнірних менеджерів та аналітиків консолідованим інструментарієм для створення, структурування й оперативної трансляції контенту. Програмний продукт дозволяє організувати ефективну взаємодію під час підготовки текстових матеріалів, а також автоматизувати облік специфічних галузевих даних: генерацію турнірних сіток, адміністрування розкладів матчів, ведення профілів команд та KDA-статистики гравців.

Головною бізнес-метою впровадження системи є підвищення оперативності роботи медіакоманди та забезпечення високої точності статистичної інформації завдяки єдиному централізованому середовищу управління. Такий підхід дозволяє уникнути фрагментарності під час оновлення інформації про ігрові події, нівелювати ризики виникнення помилок через людський фактор при ручному перенесенні результатів та максимально скоротити часовий розрив між завершенням матчу і публікацією підсумкової аналітики. Відповідно, ключовою метрикою успіху (критерієм ефективності) проєкту визначено вимірюване скорочення часу на повний цикл обробки результату кіберспортивної події – від зміни статусу матчу та автоматичного перерахунку турнірної сітки до випуску відповідного новинного матеріалу, у порівнянні з традиційними підходами ручного редагування сторінок.

Функціональні межі розроблюваної системи утворюють закритий внутрішній контур, що охоплює комплекс взаємопов'язаних модулів. До них належить підсистема управління обліковими записами із жорстким

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

розмежуванням прав доступу на основі ролей (адміністратор, журналіст, менеджер турнірів, гравець), модуль багатофункціонального текстового редагування медіапублікацій із підтримкою семантичного тегування, а також інструментарій для конструювання багатоетапних змагань із підтримкою швейцарської системи та сіток формату Single/Double Elimination. Додатково межі проєкту включають ведення реляційної бази даних кіберспортивних організацій, трансферів та механізми інтеграції для оперативного моніторингу результатів матчів (live score).

Функціональні можливості розроблюваної інформаційної системи CyberCore CMS охоплюють комплекс взаємопов'язаних підсистем, що забезпечують автоматизацію основних процесів кіберспортивної медіаплатформи. Відповідно до вимог оформлення бакалаврських робіт, матеріал викладено грамотним науковим стилем без використання надмірної фрагментації тексту.

У сфері управління публікаціями та медіаконтентом реалізовано інструментарій для створення, редагування, видалення та планування випуску матеріалів. Цей процес забезпечується через вбудований інтерактивний редактор із підтримкою гнучкої системи семантичного тегування, що дозволяє логічно пов'язувати публікації з конкретними турнірами, командами чи гравцями.

Для організації змагального процесу спроектовано модуль конструювання та менеджменту турнірних сіток. Він надає засоби для створення, конфігурації та візуалізації турнірних структур різних типів, зокрема швейцарської системи, а також сіток формату Single Elimination. Модуль автоматизує формування розкладу, фіксацію результатів матчів та алгоритмічне переміщення команд між етапами змагань.

Підсистема ведення бази даних кіберспортивної статистики відповідає за систематизацію профілів організацій, команд та індивідуальних гравців. Її функціонал включає облік актуальних ростерів, відстеження трансферів та

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

збереження деталізованих ігрових метрик (насамперед KDA-статистики) за підсумками завершених матчів.

Безпека та структурованість робочого процесу в системі досягаються шляхом впровадження жорсткої рольової моделі розмежування доступу. Учасникам внутрішнього контуру призначаються визначені ролі (адміністратор, журналіст, менеджер турнірів, гравець) із чітко регламентованими рівнями повноважень. Це охоплює права на створення текстового контенту, модерацію інформації, підтвердження результатів матчів та адміністрування доступу інших користувачів.

Крім того, програмний продукт включає механізми моніторингу та інтеграції даних у реальному часі. Реалізовано функціонал оперативного оновлення статусів та рахунків матчів (live score) з подальшою архівацією результатів у базі даних.

Для визначення акторів у контексті проєктованої системи CyberCore CMS доцільно використати розширену систему ролей:

- адміністратор це користувач із максимальним рівнем системних привілеїв, який здійснює глобальне управління платформою. До його компетенції належить абсолютна модерація контенту, а також управління рівнями доступу та адміністрування облікових записів персоналу;
- менеджер турнірів це спеціаліст, який забезпечує функціонування та супровід кіберспортивної інфраструктури проєкту. Цей актор відповідає за генерацію турнірних сіток, оновлення статистичних профілів команд і гравців, реєстрацію трансферів, а також фіксацію підсумкових результатів матчів та KDA-статистики;
- журналіст це користувач внутрішнього контуру, відповідальний за інформаційне наповнення системи. Наділений необхідними правами для створення, форматування, семантичного тегування та безпосередньої публікації новинних і аналітичних матеріалів.

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

- авторизований користувач (Гравець / Вболівальник) це кінцевий споживач інформаційного продукту, який взаємодіє з публічним (клієнтським) інтерфейсом системи. Базовий рівень передбачає перегляд медіаконтенту та турнірної інформації, а після проходження процедури автентифікації користувач здобуває розширені права для інтерактивної взаємодії на платформі;

1.2 Узагальнена функціональна модель

Побудова функціональної моделі за методологією IDEF0 спрямована на детальний опис процесів функціонування інформаційної системи CyberCore CMS за допомогою раніше розглянутих аналогів. Відповідно до принципів зазначеного стандарту, загальний алгоритм роботи формалізується через сукупність функціональних блоків, що відтворюють чітку логічну послідовність операцій. Сутність діяльності системи полягає у цілеспрямованому перетворенні вхідних інформаційних потоків на вихідний структурований та інтерактивний контент. Ця трансформація здійснюється під впливом визначених керуючих регламентів та із залученням необхідних виконавчих механізмів.

Точка зору в розробленій моделі відповідає позиції системного аналітика. Такий ракурс дозволяє комплексно дослідити програмний продукт, концентруючи увагу на його архітектурній побудові та внутрішній логіці обробки інформації. Як наслідок, забезпечується найбільш повне і структуроване представлення вимог щодо автоматизації ключових бізнес-процесів проєктованої медіаплатформи [4].

Контекстна діаграма системи, яка відображає головну функцію «Керувати контентом та турнірними даними кіберспортивної онлайн-платформи», наведена на рисунку 1.5. Зазначена діаграма репрезентує найбільш загальний абстрактний рівень функціонування інформаційної системи CyberCore CMS, чітко окреслюючи її функціональні межі та

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

специфіку взаємодії із зовнішнім середовищем. Використання методології функціонального моделювання IDEF0 дає змогу формалізувати процеси через строгу структурування вхідних ресурсів, керуючих впливів, виконавчих механізмів та вихідних результатів.

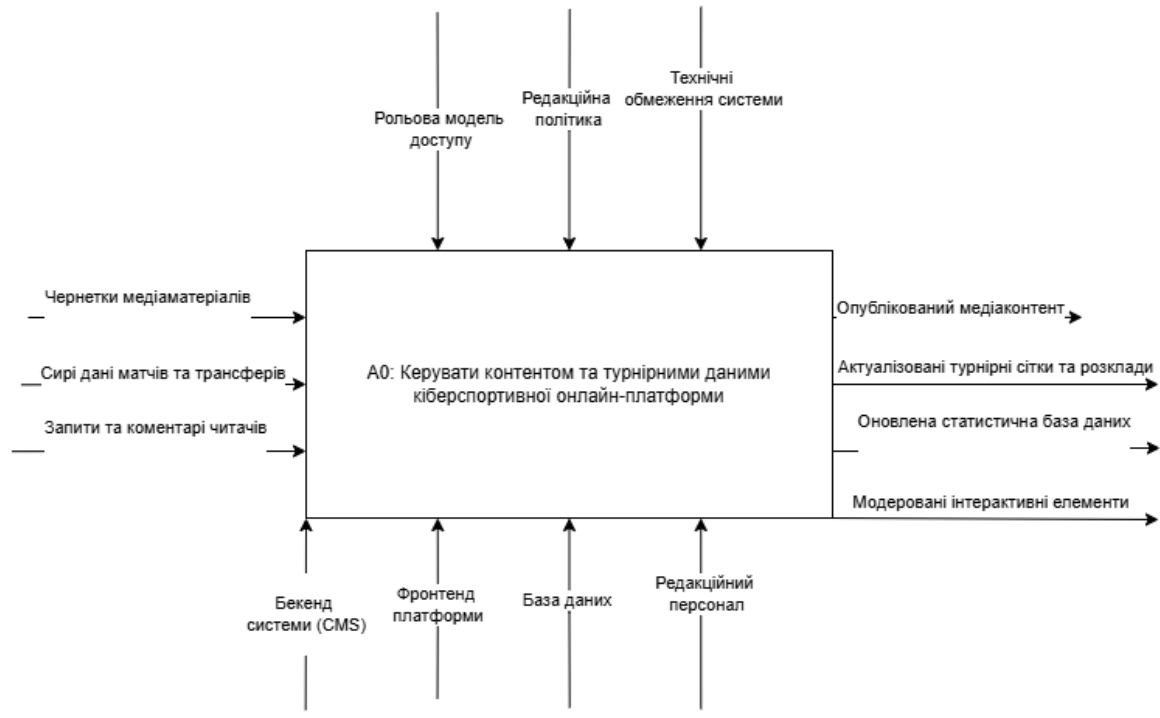


Рисунок 1.5 – Контекстна діаграма А-0

Блок А0 визначає базовий процес трансформації неструктурованих вхідних потоків – чернеток медіаматеріалів, сирих даних матчів і трансферів, а також запитів та коментарів читачів – у кінцевий інформаційний продукт. Результатом цієї трансформації є опублікований медіаконтент, актуалізовані турнірні сітки з розкладами, оновлена статистична база даних KDA-показників та модеровані інтерактивні елементи. Керуючими впливами, що регламентують алгоритми виконання операцій та визначають повноваження персоналу внутрішнього контуру, виступають розроблена багаторівнева рольова модель доступу, редакційна політика платформи та сукупність технічних обмежень.

Практичне виконання зазначеної функції забезпечується комплексом виконавчих механізмів, до складу якого входять програмні компоненти

(бекенд-частина на базі NestJS, клієнтський фронтенд-застосунок на React, реляційна база даних PostgreSQL) та безпосередньо апарат редакційного персоналу. Розроблена контекстна модель є концептуальним підґрунтям та відправною точкою для подальшої структурної декомпозиції бізнес-процесів системи на нижчі ієрархічні рівні.

На рисунку 1.6 наведено діаграму декомпозиції першого рівня A0, що розкриває внутрішню логіку функціонування системи CyberCore CMS. Головну функцію декомпозовано на чотири взаємопов'язані блоки, які відображають основні етапи роботи кіберспортивної редакції. Перший блок «Авторизувати персонал та розподілити права» забезпечує ідентифікацію користувачів та управління безпекою доступу. Другий блок «Створювати та редагувати медіаматеріали» відповідає за підготовку текстового контенту. Паралельно реалізується третій блок «Обробляти турнірні дані та статистику» для ведення ігрових показників. Підсумковий четвертий блок «Модерувати та публікувати контент» консолідує результати попередніх етапів.

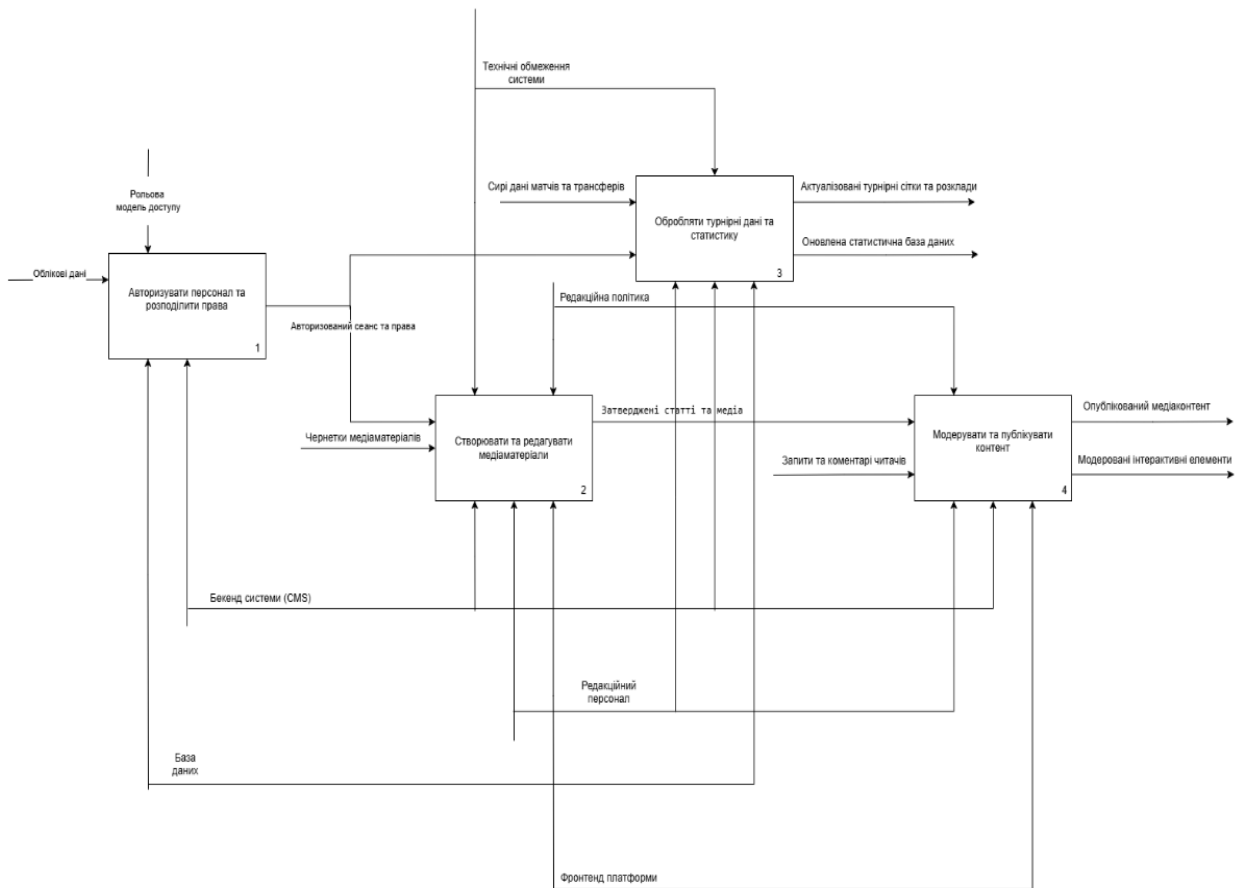


Рисунок 1.6 – Діаграма декомпозиції A0

1.3 Моделювання потоків даних та процесів

Діаграма прецедентів (Use Case) мови UML виступає базовим інструментом для формалізації функціональних вимог до проєктованої інформаційної системи. Вона описує очікувану поведінку програмного продукту на концептуальному рівні, абстрагуючись від деталей технічної реалізації. Головна перевага такого моделювання полягає у можливості проєктування системи крізь призму потреб кінцевих користувачів, що забезпечує відповідність розроблюваного функціоналу реальним бізнес-вимогам [5].

На рисунку 1.7 наведено діаграму прецедентів розроблюваної системи управління контентом CyberCore CMS. Згідно з системною специфікацією, архітектура взаємодії передбачає наявність чотирьох ключових акторів: Адміністратора, Менеджера турнірів, Журналіста та Читача.



Рисунок 1.7 – Use case діаграма

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

Деталізовані базові сценарії (Main Flow) для найважливіших прецедентів, які відображають критичні аспекти функціонування CyberCore CMS, наведено у таблицях 1.1 та 1.2.

Таблиця 1.1 – Сценарій прецеденту «Оновлювати результати матчів»

Крок	Дія актора (Менеджер турнірів)	Відповідь системи
1	Користувач обирає активний матч у турнірній сітці та вносить результати.	Система валідує введені дані (рахунок по картах, статистика гравців).
2	Актор ініціює збереження результатів.	Система фіксує інформацію в базі даних, викликає обов'язковий прецедент автоматичного перерахунку статистики та актуалізує відображення турнірної сітки на публічному порталі

Таблиця 1.2 – Сценарій прецеденту «Створювати текстові матеріали»

Крок	Дія актора (Журналіст)	Відповідь системи
1	Актор створює нову публікацію, вводить текстовий контент та прикріплює медіафайли.	Система зберігає чернетку матеріалу.
2	Актор призначає обов'язкові семантичні теги та надсилає матеріал на публікацію.	Система перевіряє наявність тегів переводить матеріал у статус «Опубліковано» та інтегрує його у загальну стрічку новин.

Діаграма потоків даних (DFD) є інструментом візуалізації руху інформації в межах будь-якого процесу або системи. У ній застосовуються

стандартизовані графічні символи (прямокутники, кола, стрілки) та лаконічні текстові ідентифікатори для позначення вхідних і вихідних потоків, точок зберігання та маршрутів передачі даних. Моделі DFD можуть варіюватися від концептуального огляду процесів до глибокої багаторівневої декомпозиції, яка розкриває механізми обробки інформації [6].

На рисунку 1.8 наведено діаграму потоків даних контекстного рівня для розроблюваної системи CyberCore CMS, яка формалізує її інформаційну взаємодію із зовнішнім середовищем. Головний процес «Інформаційна система CyberCore CMS» обмінюється даними з чотирма зовнішніми сутностями, що відповідають базовим системним ролям: Адміністратором, Журналістом, Менеджером турнірів та Читачем.

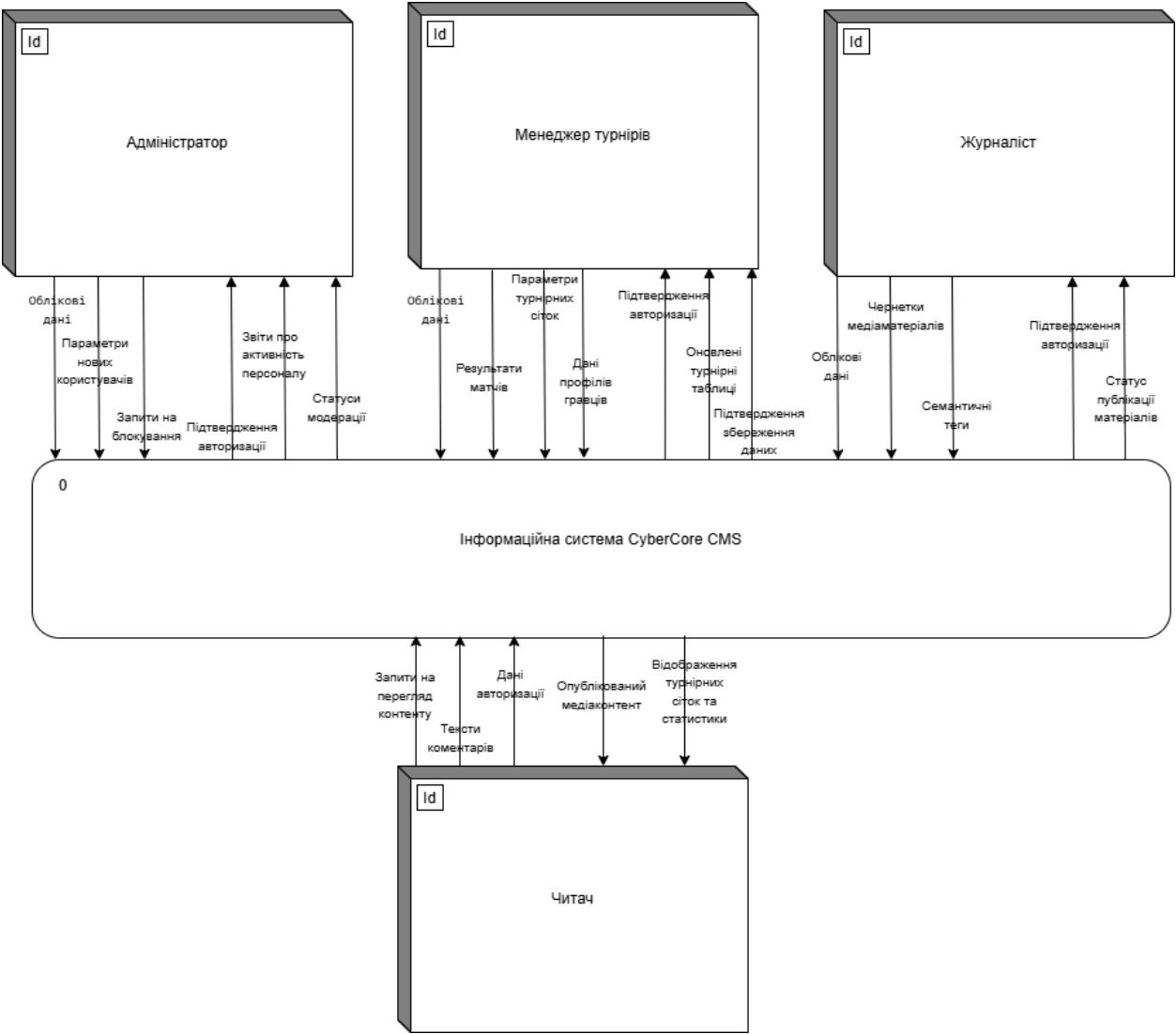


Рисунок 1.8 – Діаграма потоків даних (DFD рівня 0)

Базовим вхідним потоком для персоналу внутрішнього контуру виступають облікові дані авторизації, після чого інформаційні маршрути диференціюються. Журналіст постачає системі чернетки медіаматеріалів та семантичні теги, отримуючи статуси публікації. Менеджер турнірів передає результати матчів, статистику гравців і параметри сіток, натомість отримуючи оновлені турнірні таблиці. Адміністратор оперує параметрами облікових записів та запитами на блокування, приймаючи звіти про активність персоналу. Зовнішня сутність «Читач» ініціює запити на перегляд і надсилає тексти коментарів, отримуючи готовий опублікований медіаконтент та візуалізовані турнірні сітки. Усі потоки на діаграмі іменовані іменниками, що концептуально відображає об'єкти інформаційного обміну без прив'язки до конкретних алгоритмів програмної реалізації.

Для деталізації логіки виконання операцій у системі CyberCore CMS розроблено діаграму процесів у нотації IDEF3 (рисунок 1.9). Наведена модель дає змогу формалізувати точну послідовність дій користувача та системних процесів, ураховуючи розгалуження та паралельне виконання завдань під час обробки турнірних даних. На відміну від DFD-моделей, які орієнтовані виключно на маршрутизацію інформаційних потоків, методологія IDEF3 відображає причинно-наслідкові та хронологічні зв'язки між конкретними одиницями роботи (Unit of Work).

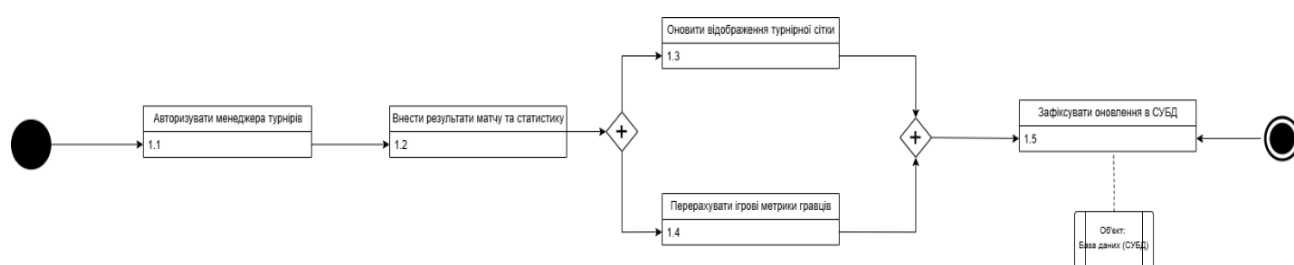


Рисунок 1.9 – Діаграма робочих потоків IDEF3

Основний сценарій, проілюстрований на діаграмі, розкриває внутрішню логіку прецеденту «Оновлювати результати матчів». Алгоритм розпочинається з блоку авторизації Менеджера турнірів у панелі

адміністрування та валідації його прав доступу на бекенді. Після успішної ідентифікації виконується дія введення підсумків кіберспортивного матчу та індивідуальної KDA-статистики гравців. Далі в алгоритмі застосовується шлюз паралельного розгалуження (синхронне «I»). Він демонструє, що після підтвердження форми серверна частина на базі NestJS одночасно ініціює два незалежні процеси: генерацію оновленого відображення турнірної сітки для клієнтського застосунку (React) та фоновий перерахунок ігрових метрик у профілях команд і кіберспортсменів.

Після завершення обох паралельних гілок вони об'єднуються через синхронний шлюз злиття, який дає команду на фінальний запис усіх модифікованих масивів інформації. Для семантичного зв'язку технологічних процесів із логічними сутностями на діаграмі використано елемент Referent (Посилання), що безпосередньо вказує на реляційну базу даних PostgreSQL як на кінцеве сховище стану системи. Такий архітектурний підхід, реалізований за допомогою інструментарію Prisma ORM, гарантує транзакційність: турнірна сітка та статистика гравців оновлюються і фіксуються одночасно, що повністю унеможливорює розсинхронізацію даних на онлайн-платформі.

1.4 Моделювання бізнес процесів кіберспортивної онлайн-платформи BPMN 2.0

Моделювання бізнес-процесів за стандартом BPMN 2.0 виступає уніфікованим інструментом графічної формалізації послідовності виконання операцій та обміну даними між структурними компонентами чи зовнішніми суб'єктами системи. На противагу методологіям IDEF0 та IDEF3, що орієнтовані переважно на ієрархічну декомпозицію функцій і локальну хронологію, нотація BPMN 2.0 забезпечує строге розмежування зон відповідальності через використання пулів (Pools) і доріжок (Lanes), а також надає розширені можливості для опису алгоритмів обробки подій, повідомлень та виняткових ситуацій [7].

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

У контексті проєктування архітектури CyberCore CMS застосування даного стандарту є об'єктивно необхідним. Такий підхід дає змогу комплексно візуалізувати наскрізні сценарії інформаційної взаємодії між персоналом внутрішнього контуру управління, автоматизованими сервісами системи та кінцевими користувачами публічної платформи.

З метою забезпечення наскрізності та логічної послідовності процесу проєктування інформаційної системи CyberCore CMS сформовано аналітичну таблицю (табл. 1.3). Застосування такого підходу дає змогу детально обґрунтувати доцільність впровадження кожного автоматизованого сервісу, користувацької дії чи логічного розгалуження в архітектурі системи, виходячи з об'єктивних інженерних та бізнес-потреб кіберспортивної медіаплатформи. активічно, ця таблиця виконує функцію сполучної ланки між попереднім аналізом предметної області та етапом безпосереднього проєктування програмної архітектури.

Таблиця 1.3 – Трансформація кроків UJM у завдання бізнес процесу

Етап шляху	Дія користувача (UJM)	Точка контакту	Проблема (Біль)	Проектована функція ІС (BPMN Task)
Авторизація	Здійснення процедури ідентифікації в адміністративній панелі.	Форма авторизації.	Загроза несанкціонованого доступу сторонніх осіб до внутрішнього контуру медіаплатформи.	Service Task: Автоматизована валідація облікових даних та верифікація токена сесії на серверній частині.
Робота з контентом	Формування нової аналітичної статті або оперативної новини.	Текстовий редактор із підтримкою форматування (WYSIWYG).	Ризик втрати введених даних унаслідок непередбачуваного закриття сесії	Service Task: Динамічне збереження поточного стану чернетки в базі даних.

Продовження таблиці 1.3

Менеджмент турнірів	Внесення підсумкових результатів завершеного кіберспортивного матчу.	Інтерфейс управління турнірними сітками.	Вплив людського фактора: імовірність виникнення помилок під час ручного обчислення KDA-статистики та актуалізації таблиці.	Parallel Gateway: Синхронний запуск процесів візуального оновлення сітки та фоновий перерахунок ігрових метрик.
Модерація	Валідація та оприлюднення коментарів кінцевих користувачів.	Модуль адміністрування коментарів.	Перенавантаження інформаційної стрічки спамом та деструктивними повідомленнями	Exclusive Gateway: Алгоритмічна фільтрація повідомлень

У межах проєктування логіки CyberCore CMS розроблено три типи діаграм у нотації BPMN 2.0. Вони формалізують різні рівні системної взаємодії: від базових лінійних алгоритмів до комплексних міжмодульних інтеграцій із застосуванням пулів та обміну повідомленнями.

На рисунку 1.10 наведено діаграму базового бізнес-процесу, яка регламентує процедуру ініціалізації та первинного конфігурування нового кіберспортивного турніру. Актором у цьому процесі виступає Менеджер турнірів, що створює подію через адміністративну панель. Модель відображає алгоритм валідації вхідних параметрів та подальшу автоматизацію технічних етапів. Застосування ексклюзивного шлюзу дозволяє коректно обробляти виняткові ситуації, пов'язані з введенням некоректних даних або конфліктом дат, забезпечуючи цілісність загального розкладу на платформі.

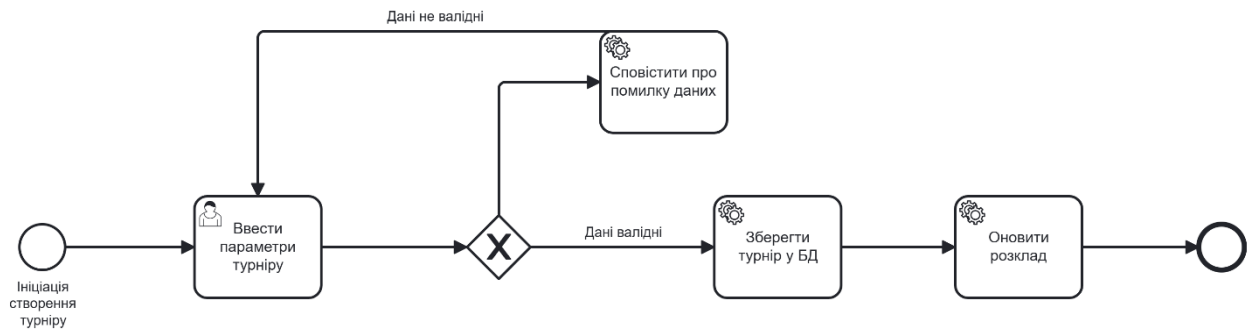


Рисунок 1.10 – BPMN Створення нового турніру

Рисунок 1.11 репрезентує діаграму процесу підготовки та публікації медіаконтенту, логічно розділену на три доріжки: Журналіст, Адміністратор та Сервіс CMS. Ця модель ілюструє наскрізний цикл взаємодії редакційного персоналу. Журналіст формує текстовий матеріал, призначає семантичні теги для інтеграції контенту з базою знань та ініціює відправку на модерацію. На етапі перевірки Адміністратором використовується XOR-шлюз для ухвалення рішення: затвердити публікацію або повернути її на доопрацювання через цикл зворотного зв'язку. У разі погодження програмний сервіс автоматично переводить матеріал у статус «Опубліковано» та здійснює його рендеринг у публічній новинній стрічці.

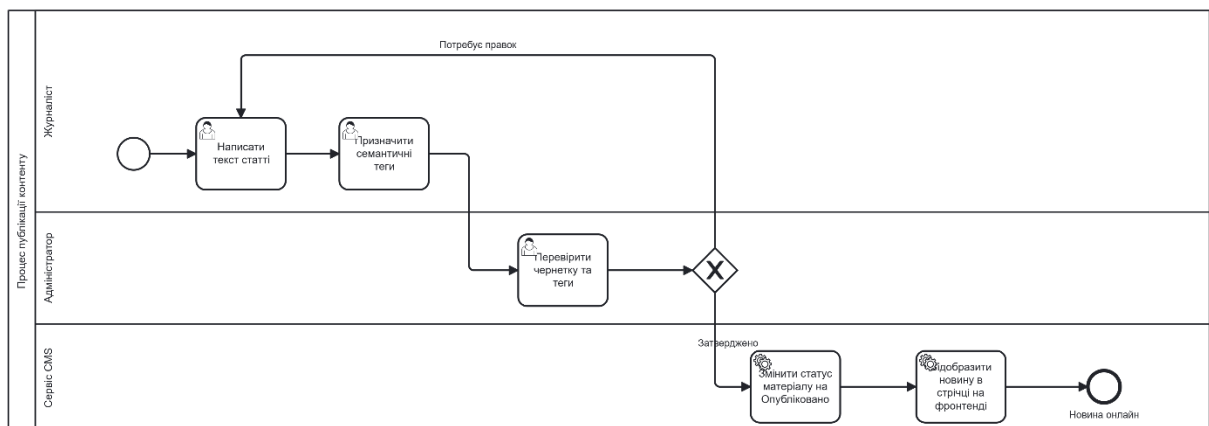


Рисунок 1.11 – BPMN Публікація медіаконтенту

На рисунку 1.12 відображено діаграму комплексного бізнес-процесу обробки результатів матчів та перерахунку ігрової статистики. Архітектура моделі побудована на взаємодії двох незалежних пулів. Комунікація між ними реалізується через потоки повідомлень, що імітують асинхронні API-запити.

Завдяки шлюзу паралельного виконання серверна частина синхронно оновлює візуальну структуру турнірної сітки та ініціює фонові завдання для актуалізації індивідуальних показників кіберспортсменів. Завершується цикл транзакції відправкою підтверджуючого повідомлення на клієнтську частину інтерфейсу.

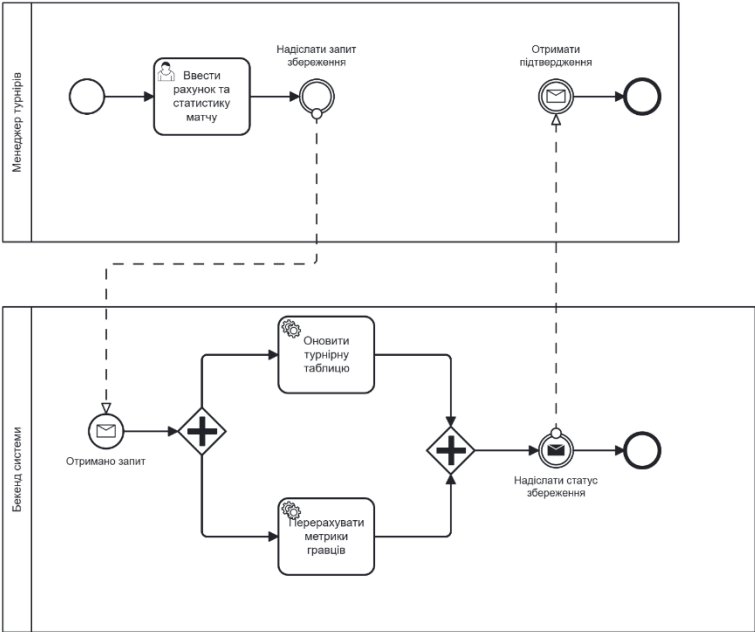


Рисунок 1.12 – BPMN Обробка результатів матчу

1.5 Формування вимог до інформаційної системи CyberCore CMS

На основі проведеного системного аналізу предметної області, виявлених недоліків існуючих рішень та побудованих моделей бізнес-процесів (за методологіями IDEF0 та BPMN 2.0) сформовано комплексний перелік вимог до розроблюваної інформаційної системи. Ці вимоги слугують технічним завданням для подальшого етапів проєктування архітектури та програмної реалізації медіаплатформи.

Усі вимоги до системи логічно поділено на дві основні категорії: функціональні (що саме повинна робити система) та нефункціональні (якими експлуатаційними характеристиками вона повинна володіти).

1) Функціональні вимоги:

- управління доступом та ролями: Система повинна підтримувати механізм авторизації та рольову модель доступу (RBAC) із розподілом повноважень між чотирма категоріями користувачів: Адміністратор, Журналіст, Менеджер турнірів та Читач;
- управління медіаконтентом: Платформа має забезпечувати інструментарій для створення, редагування та публікації новин за допомогою інтегрованого WYSIWYG-редактора, а також підтримувати механізм додавання семантичних тегів для зручної фільтрації контенту;
- управління кіберспортивними турнірами: Система повинна надавати функціонал для створення змагань із динамічним генеруванням форматів (групові стадії, плей-оф), ведення турнірних сіток та фіксації результатів окремих матчів;
- обробка ігрової статистики: Платформа повинна підтримувати збереження та каскадний перерахунок індивідуальних показників гравців (зокрема KDA – Kills, Deaths, Assists) після завершення кожного матчу;
- взаємодія з кінцевим користувачем: Система має генерувати публічну інтерактивну стрічку новин та турнірні таблиці, доступні для перегляду та пошуку без необхідності обов'язкової реєстрації;

2) Нефункціональні вимоги:

- вимоги до архітектури та технологічного стека: Клієнтська частина повинна бути реалізована за архітектурою Single Page Application (SPA), а серверна – мати подієво-орієнтовану мікромодульну структуру на базі середовища Node.js для забезпечення високої швидкості обробки паралельних запитів;
- вимоги до збереження даних: Для унеможливлення розсинхронізації пов'язаних сутностей (наприклад, статусу матчу та статистики гравців) база даних має бути реляційною та підтримувати принципи транзакційної цілісності (ACID);

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		29

- вимоги до інформаційної безпеки: Система повинна забезпечувати криптографічне хешування паролів, захист від XSS-атак, блокування несанкціонованих міждомених запитів (CORS) та валідацію всіх вхідних даних на рівні DTO-моделей. Сесії користувачів мають захищатися за допомогою JWT-токенів;
- масштабованість: Архітектура повинна підтримувати незалежне горизонтальне масштабування клієнтського та серверного контейнерів на випадок пікових навантажень під час великих кіберспортивних турнірів;

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		30

2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА КОМПОНЕНТІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 UML діаграми та User Journey Map

Створення карти шляху користувача (User Journey Map, UJM) виступає ефективним методом структурно-візуального аналізу, спрямованим на дослідження та документування процесів взаємодії користувачів з інтерфейсами та сервісами системи. Цей підхід дозволяє змоделювати користувацький досвід у вигляді формалізованого наративу, що відображає когнітивні, емоційні та поведінкові реакції актора під час його контакту з різними точками інформаційної системи.

У таблиці 2.1 наведено UJM для актора «Журналіст». Модель фокусується на дослідженні повного циклу підготовки, верстки та оприлюднення медійних матеріалів у внутрішньому контурі управління CyberCore CMS.

Таблиця 2.1 – User Journey Map для актора «Журналіст»

Рядок (Параметр)	Фаза 1: Усвідомлення інфоприводу	Фаза 2: Авторизація в системі	Фаза 3: Написання матеріалу	Фаза 4: Тегування та відправка	Фаза 5: Завершен ня та публікація
Дії користува ча	Здійснює моніторинг кіберспортивних подій, агрегує первинні дані про трансфери чи результати матчів.	Заповнює форму авторизації в адміністративній панелі медіаплатформ и.	Ініціює створення новини/статті, верстає текстовий блок, завантажує медіафайли.	Призначає обов'язкові семантичні теги та надсилає матеріал на затвердження.	Здійснює візуальний контроль фінального вигляду публікації у стрічці новин на фронтенді.

Продовження таблиці 2.1

Точки контакту	Зовнішні інформаційні ресурси, стрімінгові платформи, офіційні соцмережі команд.	Сторінка входу (Sign-in form) адмін-панелі CyberCore CMS.	Вбудований текстовий WYSIWYG-редактор, модуль завантаження медіа.	Модуль семантичного тегування контенту, інтерфейсна кнопка «На модерацію».	Публічна стрічка новин клієнтської частини онлайн-платформи.
Думки	«Цей трансфер має значний резонанс, необхідно оперативно підготувати матеріал».	«Сподіваюсь на швидке проходження двофакторної автентифікації».	«Чи передбачено автоматичне збереження чернетки на випадок втрати з'єднання?».	«Важливо коректно тегнути обох гравців для інтеграції новини в їхні профілі».	«Матеріал зверстано вдало, відображення на сайті є коректним».
Емоції	Професійний азарт, відчуття терміновості.	Зосередженість, стан очікування.	Творче натхнення, легке хвилювання.	Відчуття відповідальності.	Професійне задоволення, спокій.
Біль (Pain Point)	Складність та часові затрати на оперативний пошук валідованих першоджерел.	Необхідність повторного введення складних облікових даних у разі розриву сесії.	Критичний ризик втрати значного обсягу незбереженого тексту при програмному збої вкладки браузера.	Надмірна кількість ручних маніпуляцій для встановлення зв'язків статті зі спорідненими сутностями.	Можлива затримка відображення опублікованого матеріалу внаслідок процесів кешування.

Продовження таблиці 2.1

Можливості (Opportunity)	Інтеграція спеціалізованих парсерів офіційних анонсів організацій у систему.	Впровадженн ^я механізмів довготривалих сесій безпеки (Remember me) та протоколу OAuth.	Реалізація функції динамічного фонового збереження чернетки в базі даних.	Впровадженн ^я алгоритмів автоматичного пропонуванн ^я тегів на базі аналізу текстового масиву статті.	Розробка механізму миттєвої інвалідації кешу (Cache invalidation) при зміні статусу матеріалу в CMS.
Крива емоцій	Нейтрально (0)	Трохи вниз (-1)	Вгору (+1)	Стабільно (+1)	Пік вгору (+2)

Аналіз кривої емоцій для Журналіста (рис. 2.1) фіксує характерний спад на етапі авторизації, що зумовлено необхідністю виконання рутинних процедур безпеки. Проте, починаючи з фази безпосереднього створення контенту, графік демонструє стабільну висхідну динаміку. Свого максимуму крива досягає на фінальній фазі, коли користувач фіксує успішний результат своєї роботи на публічній платформі без виникнення технічних затримок.

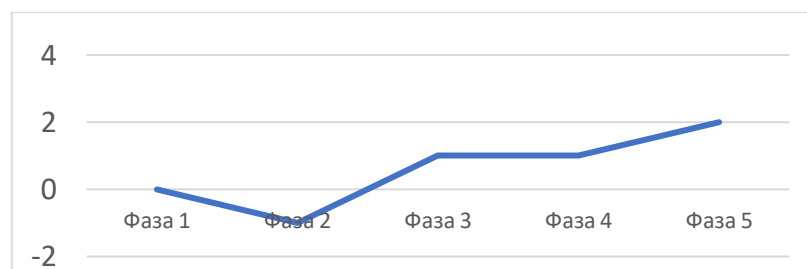


Рисунок 2.1 – Крива емоцій Журналіста

Сценарій взаємодії для актора «Менеджер турнірів» (табл. 2.2) має кардинально іншу специфіку, оскільки його діяльність сфокусована на роботі зі спортивною інфраструктурою та даними в реальному часі (live). Пріоритетним завданням для цієї ролі є швидке внесення значних масивів

кількісних показників із гарантією подальшої глобальної синхронізації даних на платформі.

Таблиця 2.2 – User Journey Map для актора «Менеджер турнірів»

Рядок (Параметр)	Фаза 1: Отримання результатів	Фаза 2: Входження в контекст	Фаза 3: Активне внесення даних	Фаза 4: Синхронізація	Фаза 5: Фіксація стану системи
Дії користувача	Агрегує фінальні сирі дані за підсумками завершеного кіберспортивного матчу.	Запускає конструктор турнірних сіток та локалізує відповідний матч.	Вносить підсумковий рахунок за картами та деталізовану індивідуальну статистику гравців (KDA).	Очікує на завершення фонових процесів оновлення таблиць та алгоритмічного просування команд по сітці.	Здійснює верифікацію доступності оновленої статистики та сітки для кінцевих користувачів сайту.
Точки контакту	Офіційні live-протоколи, ігрові клієнти та API (наприклад, Steam/Riot API).	Інтерфейс адміністрування турнірних подій та списку матчів у CMS.	Модальне вікно внесення матч-поінтів та CRUD-форма статистики.	Системний статус-бар обробки паралельних завдань.	Публічна сторінка турніру та профілі кіберспортсменів (фронтенд).
Думки	«Матч завершено, аудиторія очікує на оперативне оновлення»	«Цей матч належить до групи Б, необхідно швидко відкрити»	«Вводжу статистику для першої карти, переходжу до другої»	«Чи коректно спрацював тригер автоматичного перерахунку»	«Турнірна сітка успішно оновилася, переможці кваліфікувалися до плей-оф».

Продовження таблиці 2.2

Емоції	Стан зосереджено сті.	Оперативна активація уваги.	Напруга, максимальний рівень концентрації.	Стан очікування та надії на коректну обробку.	Емоційне полегшення, професійний спокій.
Біль (Pain Point)	Наявність великої кількості розрізнених джерел з фінальними протоколами матчів.	Ускладнена навігація у масштабних турнірних структурах (наприклад, сітках на 32 і більше команд).	Вплив людського фактора: високий ризик опечатки при ручному введенні значної кількості метрик.	Потреба в ручному запуску процедур оновлення для суміжних модулів статистики.	Технічний ризик розсинхронізації баз даних (наприклад, оновлення рахунку без відповідної зміни позиції в сітці).
Можливості (Opportunity)	Пряма вебхук-інтеграція з офіційними турнірними API для повної автоматизації збору.	Розробка інтерактивної системи пошуку та динамічного масштабування для великих турнірних сіток.	Впровадження алгоритмів валідації "на льоту" із підсвічуванням аномально високих/низьких значень.	Реалізація алгоритмів автоматичного каскадного оновлення зв'язаних сутностей.	Забезпечення повної транзакційності на рівні бази даних (ACID-принципи).
Крива емоцій	Нейтрально (0)	Стабільно (0)	Сильний спад вниз (-1)	Нейтрально (0)	Високий пік вгору (+2)

Динаміка емоційного стану Менеджера турнірів (рис. 2.2) характеризується різким висхідним трендом на завершальному етапі взаємодії. Найбільш стресовою фазою є процес активного введення даних, що пояснюється високим рівнем відповідальності за достовірність статистичної

інформації. Максимальне значення кривої на фінальному кроці свідчить про високий рівень задоволеності ефективністю автоматизації: користувач пересвідчується, що єдина ініційована дія запустила комплексний ланцюжок каскадних змін, коректно оновивши як турнірну структуру, так і індивідуальні профілі кіберспортсменів.

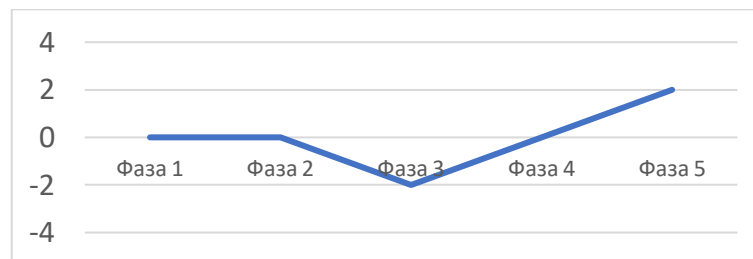


Рисунок 2.2 – Крива емоцій Менеджера турнірів

Результати аналізу побудованих карт шляху користувача свідчать, що ключовими проблемними зонами (Pain Points) для обох акторів є вплив людського фактора під час ручної обробки масивів інформації та об'єктивна потреба в оперативності системної синхронізації. Виявлені фактори були класифіковані як пріоритетні архітектурні вимоги при проєктуванні бек-офісу CyberCore CMS. Їх практичне вирішення досягається шляхом реалізації механізмів фонового збереження чернеток, впровадження інтерактивних інструментів валідації та забезпечення паралельного виконання фонових процесів оновлення пов'язаних сутностей бази даних.

Діаграма класів виступає базовим інструментом об'єктно-орієнтованого проєктування, що візуалізує статичну структуру інформаційної системи. Вона формалізує типи об'єктів, їхні внутрішні атрибути, поведінкові методи та фундаментальні зв'язки. У контексті розробки CyberCore CMS побудована діаграма класів (рисунок 2.3) виконує роль концептуальної основи для подальшого проєктування реляційної бази даних та формування архітектури серверної частини. Концептуальна модель системи логічно сегментована на три взаємопов'язані предметні домени: управління користувачами, адміністрування медіаконтенту та обробка турнірних даних.

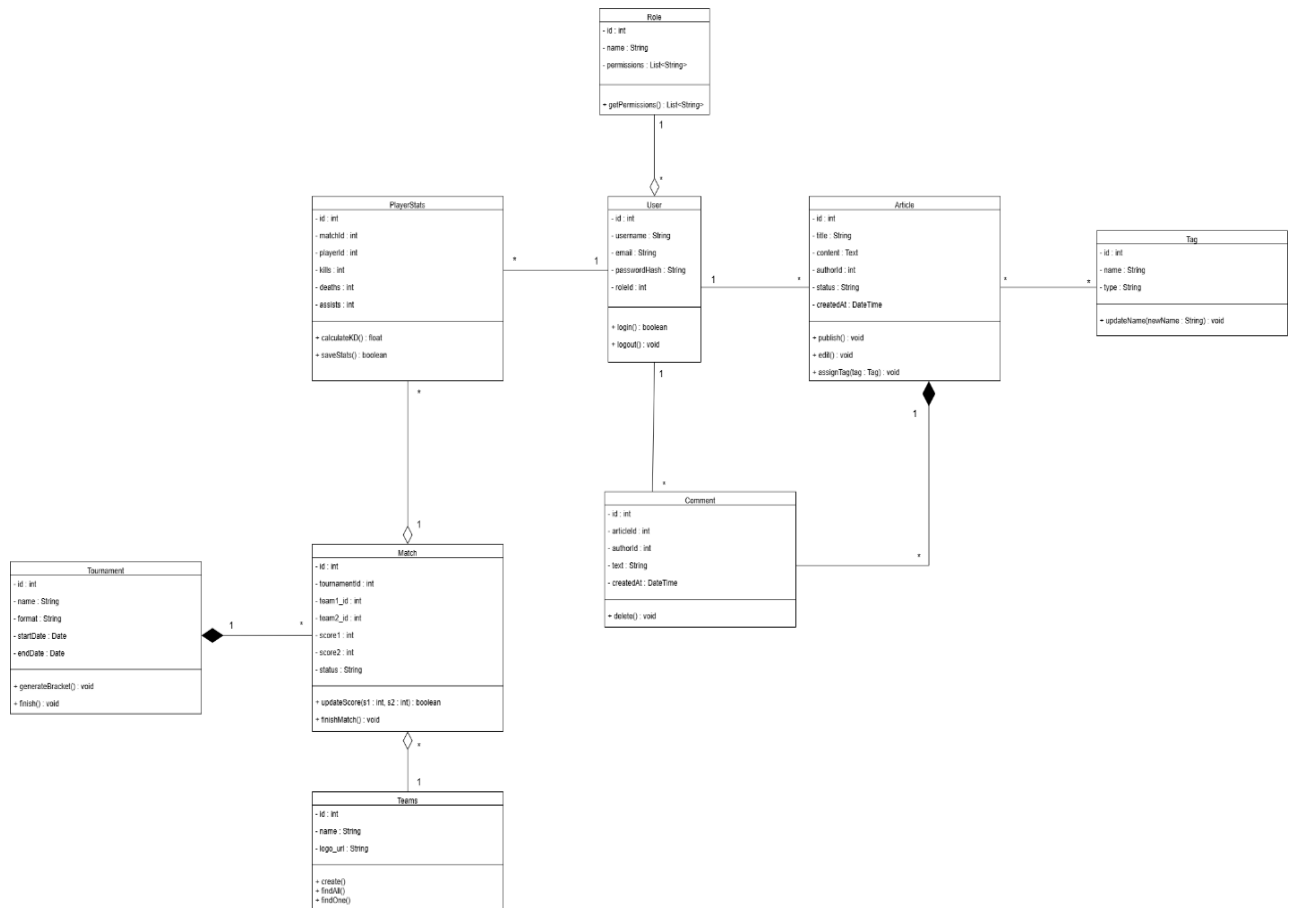


Рисунок 2.3 – Діаграма класів

Базовою сутністю домену управління користувачами є клас User, який інкапсулює ідентифікаційні атрибути (id, username, email, passwordHash) та основні методи авторизації (login(), logout()). Цей клас пов'язаний відношенням асоціації з сутністю Role, що регламентує рівень системних привілеїв (permissions) та визначає рольову модель доступу персоналу.

Архітектура домену управління контентом формується навколо класу Article (Стаття). Ця сутність акумулює такі властивості, як заголовок (title), текстове наповнення (content) та стан публікації (status), а також методи управління життєвим циклом матеріалу (publish(), edit(), assignTag()).

Важливим архітектурним рішенням є застосування відношення композиції між статтею та класом Comment: коментарі є структурними компонентами новини і підлягають каскадному видаленню у разі знищення батьківського об'єкта. Для забезпечення семантичної зв'язності інформації

впроваджено клас Tag, який взаємодіє зі статтею через асоціативний зв'язок типу «багато-до-багатьох».

Найбільшу структурну складність демонструє домен управління турнірними даними, ядром якого виступає клас Tournament. Він містить конфігураційні атрибути події (name, format, startDate, endDate) та алгоритмічні методи (generateBracket(), finish()). Дана сутність утворює строгий композиційний зв'язок формату «один-до-багатьох» із класом Match, оскільки ігрові поєдинки не можуть існувати поза контекстом конкретного змагання. Клас матчу зберігає поточні результати (score1, score2, status) та реалізує метод їх валідації (updateScore()). З метою ведення глибокої аналітики розроблено клас PlayerStats, що фіксує ключові ігрові метрики (kills, deaths, assists) і має відповідні зв'язки з матчем та профілем гравця.

Спроектована статична структура об'єктів створює надійне підґрунтя для реляційного моделювання, гарантує високий рівень нормалізації даних, мінімізує їх дублювання та забезпечує необхідну гнучкість для подальшого масштабування платформи.

2.2 Специфікація бази даних CyberCore CMS

Проектування структури бази даних виступає фундаментальним етапом розробки інформаційної системи CyberCore CMS. З огляду на жорсткі вимоги щодо надійності, гарантування цілісності інформації та необхідності виконання складних транзакційних операцій (зокрема, синхронного оновлення турнірних структур та індивідуальної статистики гравців), архітектурним рішенням для підсистеми зберігання обрано реляційну СУБД PostgreSQL. Зазначений вибір обґрунтовується високою продуктивністю ядра, ефективною підтримкою багаторівневих зв'язків між сутностями та повною відповідністю принципам транзакційності ACID. Реалізація таких характеристик є критично необхідною умовою для безпечного збереження

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		38

масивів кіберспортивних метрик та повного унеможливлення розсинхронізації даних на медіаплатформі.

Діаграма «сутність-зв'язок» (ER-діаграма) виступає базовим інструментом структурного моделювання, що візуалізує принципи взаємодії об'єктів у межах інформаційної системи. Такий підхід дає змогу формалізувати загальну архітектуру, чітко окреслюючи відношення між ключовими компонентами та набори їхніх атрибутів.

На етапі концептуального проєктування визначено базові сутності предметної області платформи CyberCore CMS та типи зв'язків між ними. Архітектуру моделі формують сутності: Користувач, Роль, Стаття, Коментар, Тег, Турнір, Матч, Команда та Статистика. Розроблена ER-діаграма (рисунок 2.4) відображає наскрізні інформаційні процеси системи. Відповідно до логіки моделі, сутність «Користувач» (журналіст) утворює зв'язок «один-до-багатьох» із сутністю «Стаття», яка, своєю чергою, може містити множину пов'язаних сутностей «Коментар» від читачів.

Взаємодію між об'єктами «Стаття» та «Тег» організовано за типом відношення «багато-до-багатьох», що забезпечує гнучке семантичне групування медіаматеріалів за дисциплінами чи організаціями. У межах спортивного домену сутність «Турнір» включає множину об'єктів «Матч», кожний з яких взаємодіє з сутністю «Команда» та генерує масив записів у сутності «Статистика гравців» (PlayerStats) для обчислення індивідуальних метрик кіберспортсменів. Функціональне розмежування прав доступу персоналу до адміністративного контуру реалізовано через інтеграцію сутності «Роль», що жорстко регламентує системні повноваження прив'язаних до неї облікових записів.

Відповідно до методології проєктування баз даних, логічна модель виступає закономірним етапом еволюції та деталізації концептуальної ER-діаграми. На цьому рівні інформаційна архітектура розширюється шляхом визначення конкретних типів даних для кожного атрибута,

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		39

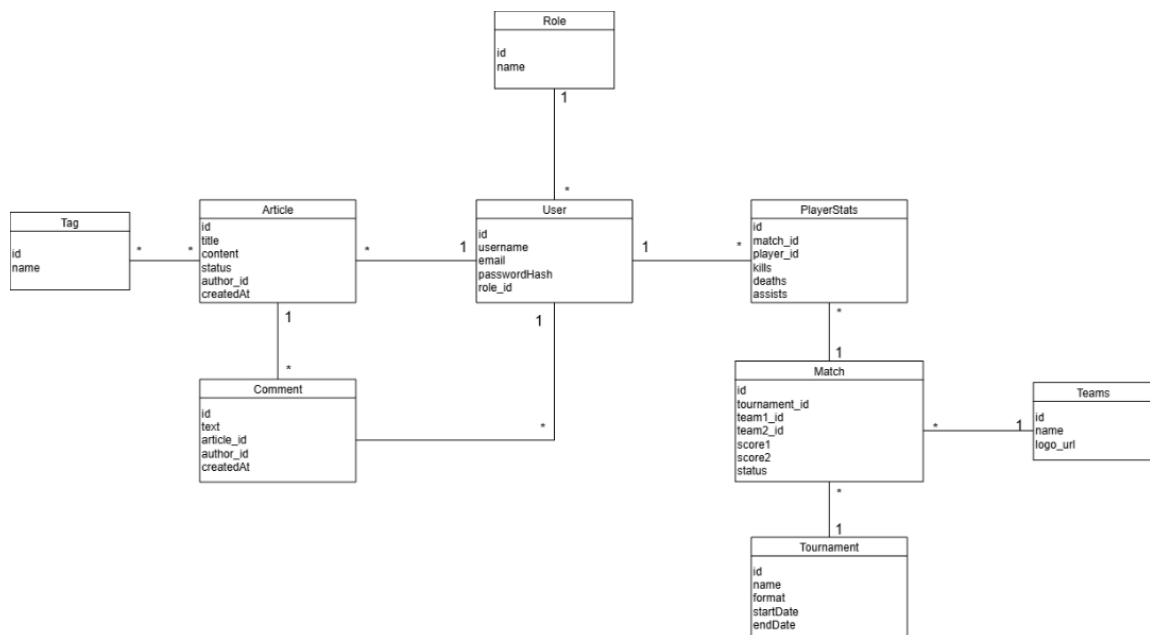


Рисунок 2.4 – Концептуальна модель даних

Розроблена логічна модель (рисунок 2.5) ілюструє безпосередню трансформацію абстрактних сутностей у реляційні таблиці. Ключовим архітектурним рішенням на даному етапі стала реалізація асоціативної (проміжної) таблиці article_tags. Оскільки обрана реляційна СУБД PostgreSQL не підтримує прямої реалізації зв'язків типу «багато-до-багатьох», зазначена таблиця виконує функцію зв'язуючої ланки.

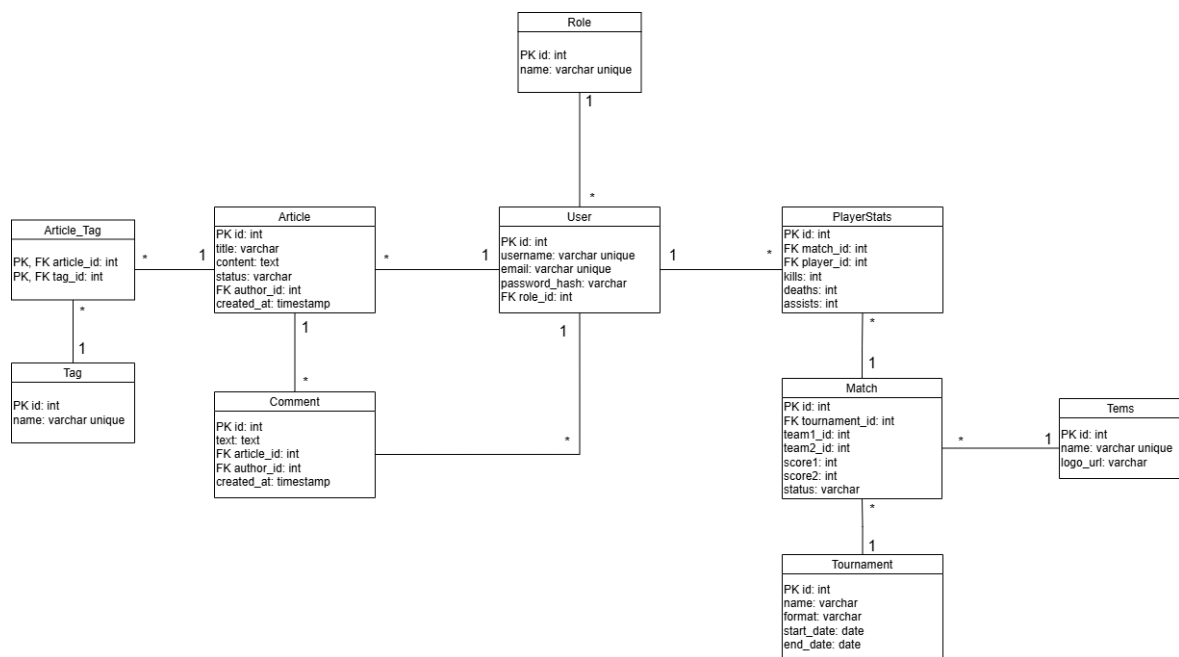


Рисунок 2.5 – Логічна модель

Для повноцінного розуміння архітектури збереження даних у системі CyberCore CMS нижче наведено детальний опис атрибутів кожної таблиці.

Таблиця roles (Ролі) – визначає рівні доступу та системні повноваження персоналу платформи

- id (int): Первинний ключ (РК), унікальний ідентифікатор ролі доступу;
- name (varchar): Унікальна назва ролі в системі (наприклад: Адміністратор, Журналіст, Менеджер турнірів), що має обмеження unique;

Таблиця users (Користувачі) – зберігає облікові та ідентифікаційні дані персоналу бек офісу та кінцевих читачів платформи:

- id (int): Первинний ключ (РК), унікальний ідентифікатор користувача;
- username (varchar): Унікальне ім'я користувача (нікнейм) для публічного відображення;
- email (varchar): Електронна пошта користувача, яка має обмеження унікальності та використовується як логін для авторизації;
- password_hash (varchar): Криптографічний хеш пароля для забезпечення безпечного зберігання облікових даних;
- role_id (int): Зовнішній ключ (FK), що посилається на відповідну роль у таблиці roles для розмежування привілеїв;

Таблиця articles (Статті) – центральна сутність для збереження новин, лонгвідів та медіаконтенту:

- id (int): Первинний ключ (РК), унікальний ідентифікатор матеріалу;
- title (varchar): Заголовок статті чи оперативної новини;
- content (text): Повний текстовий вміст публікації з підтримкою HTML верстки;
- status (varchar): Поточний стан життєвого циклу матеріалу (чернетка, на модерації, опубліковано);
- author_id (int): Зовнішній ключ (FK), який посилається на ідентифікатор автора з таблиці users;

- created_at (timestamp): Дата та час створення або виводу матеріалу в публічну стрічку;

Таблиця comments (Коментарі) – призначена для збереження інтерактивних текстових повідомлень читачів під статтями:

- id (int): Первинний ключ (РК), унікальний ідентифікатор коментаря;
- text (text): Текстовий вміст повідомлення користувача;
- article_id (int): Зовнішній ключ (FK), який прив'язує коментар до конкретної публікації в таблиці articles;
- author_id (int): Зовнішній ключ (FK), посилення на унікальний профіль автора повідомлення з таблиці users;
- created_at (timestamp): Штамп часу для фіксації моменту публікації коментаря читачем;

Таблиця tags (Теги) – довідник семантичних маркерів для категоризації та побудови зв'язків між матеріалами:

- id (int): Первинний ключ (РК), унікальний ідентифікатор тегу;
- name (varchar): Унікальна назва семантичного маркера;

Таблиця article_tags (Зв'язки статей та тегів) – асоціативна проміжна таблиця для реалізації реляційного зв'язку «багато до багатьох»:

- article_id (int): Частина складеного первинного ключа та зовнішній ключ (FK), що посиляється на відповідну публікацію в таблиці articles;
- tag_id (int): Частина складеного первинного ключа та зовнішній ключ (FK), що посиляється на відповідну категорію в таблиці tags;

Обмеження: Складений унікальний індекс на пару полів article_id та tag_id для гарантії цілісності даних та запобігання дублюванню однакових маркерів в межах однієї статті.

Таблиця tournaments (Турніри) – містить глобальну інформацію про кіберспортивні змагання та ліги:

- id (int): Первинний ключ (РК), унікальний ідентифікатор турнірної події;
- name (varchar): Офіційна назва змагання;
- format (varchar): Формат побудови сітки;

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

- start_date (date): Дата початку проведення турнірних ігор;
- end_date (date): Дата фінального поєдинку та закриття події;

Таблиця matches (Матчі) – зберігає дані про окремі ігрові зустрічі (серії карт) в межах змагань:

- id (int): Первинний ключ (PK), унікальний ідентифікатор конкретної гри;
- tournament_id (int): Зовнішній ключ (FK), посилання на батьківську сутність з таблиці tournaments;
- team1_id та team2_id (INT) – зовнішні ключі, що посилаються на ідентифікатори команд із таблиці teams;
- score1 (int): Кількість очок (виграних карт), набраних першим колективом;
- score2 (int): Кількість очок (виграних карт), набраних другим колективом;
- status (varchar): Поточний спортивний статус зустрічі (заплановано, live, завершено);

Таблиця teams (Команди) – містить інформацію про команду що виступає на турнірі :

- id (INT) – унікальний ідентифікатор команди (первинний ключ);
- name (VARCHAR) – офіційна назва кіберспортивної команди;
- logo_url (VARCHAR) – посилання на графічне зображення команди;

Таблиця player_stats (Статистика гравців) – містить детальні цифрові показники кіберспортсменів за підсумками кожної окремої гри:

- id (int): Первинний ключ (PK), унікальний ідентифікатор рядка статистичних даних;
- match_id (int): Зовнішній ключ (FK), що пов'язує показники з конкретним поєдинком у таблиці matches;
- player_id (int): Зовнішній ключ (FK), посилання на унікальний обліковий запис гравця з таблиці users;
- kills (int): Кількість знищених супротивників за гру;
- deaths (int): Кількість смертей кіберспортсмена в межах матчу;

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

– assists (int): Кількість результативних допомог своїй команді;

Наведена структура атрибутів дозволяє системі CyberCore CMS ефективно маніпулювати складними масивами інформації, забезпечуючи при цьому чітке розмежування прав доступу редакції, оперативне каскадне оновлення турнірних сіток та високу швидкість обробки аналітичних даних завдяки правильно налаштованим індексам, обмеженням унікальності та зовнішнім ключам.

Фізична модель бази даних інформаційної системи CyberCore CMS виступає безпосередньою програмною реалізацією спроектованої логічної структури в середовищі реляційної СУБД PostgreSQL. На цьому етапі розробки абстрактні сутності повністю трансформовано в реальні реляційні таблиці із суворо визначеними типами даних, обмеженнями та системою зовнішніх ключів. На рисунку 2.6 проілюстровано остаточну побудову фізичної структури бази даних у середовищі СУБД.

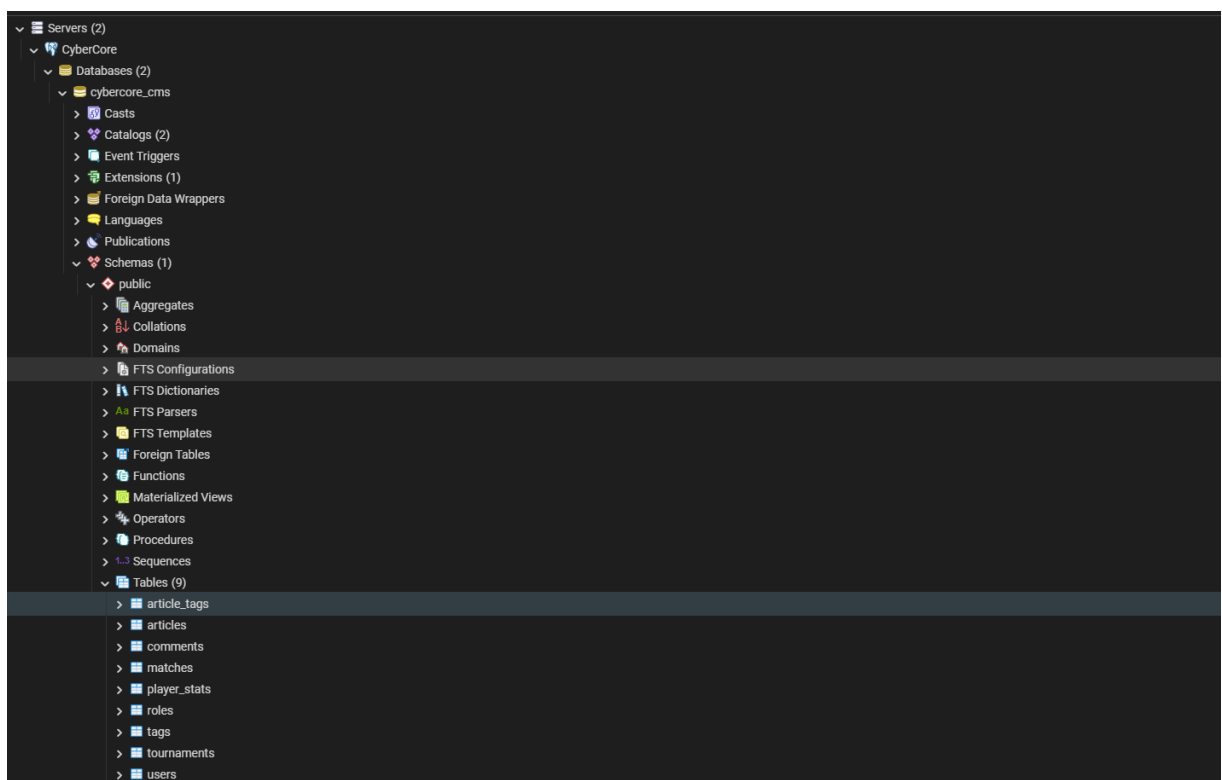


Рисунок 2.6 – Результат створення таблиць

2.3 Архітектурна специфікація CyberCore CMS

Відповідно до методології архітектурного моделювання С4, діаграма контексту виступає першим і найвищим рівнем абстракції, що слугує концептуальною відправною точкою для візуалізації програмного забезпечення. Її головне призначення полягає у демонстрації «загальної картини» шляхом розташування досліджуваної системи в центрі та відображення її оточення – користувачів і зовнішніх сервісів, з якими здійснюється взаємодія [8]. Для формалізації високорівневої структури кіберспортивної медіаплатформи CyberCore CMS та її зв'язків із зовнішнім середовищем розроблено контекстну модель (рисунок 2.7). Цей рівень моделювання дає змогу чітко окреслити межі системи, ідентифікувати ключових акторів та визначити інтеграційні точки з технічними ресурсами. На відміну від глибших рівнів декомпозиції, контекстна діаграма свідомо абстрагується від деталей внутрішньої технологічної реалізації, концентруючи увагу на глобальних інформаційних потоках та базових протоколах взаємодії на рівні програмного комплексу в цілому [8].

Центральним вузлом наведеної моделі виступає безпосередньо програмна система CyberCore CMS – комплексний вебзастосунок для агрегації кіберспортивного контенту та адміністрування турнірних сіток. Згідно із затвердженою рольовою специфікацією, система обслуговує чотири категорії користувачів. Менеджер турнірів відповідає за ініціалізацію змагань та фіксацію результатів матчів (зокрема індивідуальної KDA-статистики). Журналіст забезпечує платформу текстовими матеріалами та оперативними новинами. Адміністратор здійснює глобальну модерацію контенту й управління обліковими записами персоналу. Кінцевий споживач інформаційного продукту – Читач (вболівальник) – наділений правами перегляду публікацій, моніторингу турнірних таблиць та коментування. Комунікація всіх перелічених суб'єктів із вебзастосунком відбувається за захищеним протоколом HTTPS через стандартні клієнтські браузерери.

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

Враховуючи необхідність обробки значних масивів динамічної інформації, на діаграмі виокремлено критичні інфраструктурні елементи, що виступають як зовнішні системи (External Systems). Першою з них є система керування реляційними базами даних PostgreSQL, яка гарантує надійність, транзакційність та цілісність збереження профілів, турнірних структур і статистичних показників. Другим надважливим зовнішнім компонентом є кіберспортивні програмні інтерфейси (External Esports APIs). Платформа інтегрується з ними для отримання верифікованих первинних даних про поточні ігрові події та трансфери, що суттєво мінімізує обсяги ручного введення інформації.

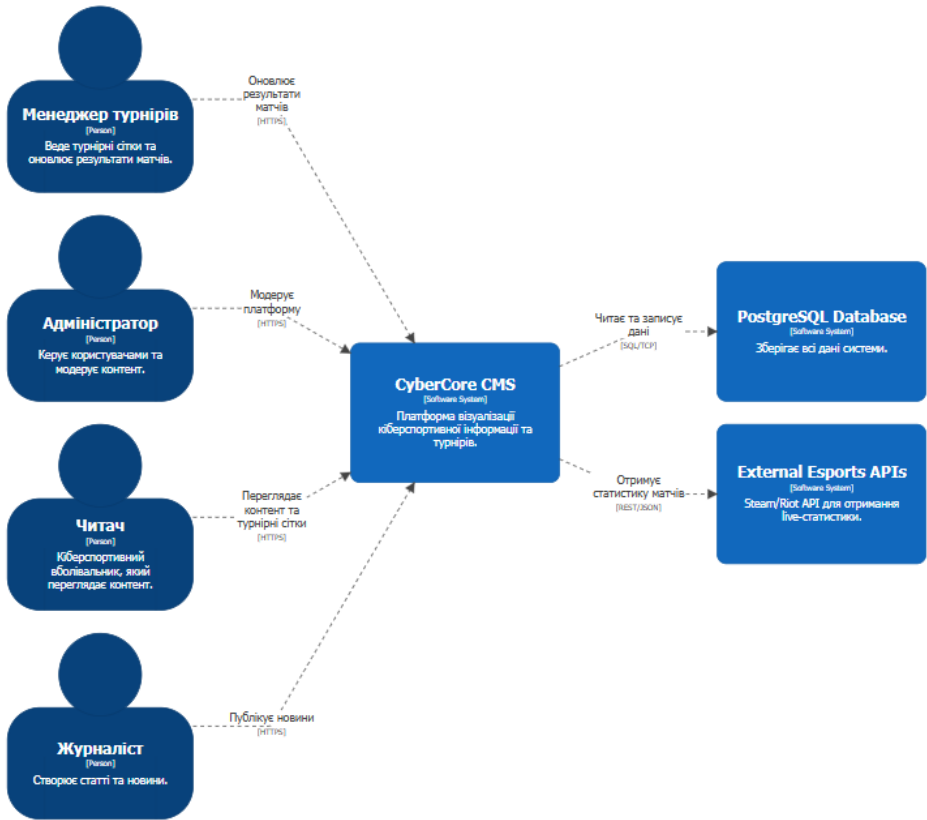


Рисунок 2.7 – C4 контекстна діаграма

Загалом побудована контекстна діаграма повністю корелює з раніше розробленими моделями прецедентів (Use Case) та потоків даних (DFD). Вона наочно підтверджує, що CyberCore CMS функціонує як цілісний інтеграційний

хаб, який консолідує роботу редакційного бек-офісу із зовнішніми джерелами даних для безперебійного постачання найактуальнішого контенту кінцевим користувачам платформи.

Згідно з принципами методології С4, після формування загального контексту процес архітектурного моделювання переходить до рівня контейнерів. На цьому етапі розкривається високорівнева організація програмної архітектури: монолітна система декомпозується на окремі розгортані блоки із чітким визначенням ключових технологічних рішень та механізмів їхньої взаємодії.

З метою деталізації внутрішньої структури CyberCore CMS та обґрунтування оптимального технологічного стека розроблено діаграму контейнерів (рисунок 2.8). Зважаючи на специфіку кіберспортивної медіаплатформи, що передбачає обробку значної кількості паралельних запитів, асинхронний перерахунок статистичних метрик та потенційну необхідність live-оновлення результатів, для реалізації бекенду обрано серверний фреймворк NestJS (екосистема Node.js) із суворою типізацією на TypeScript. Таке рішення гарантує високу продуктивність завдяки подієво-орієнтованій архітектурі (Event-driven) та механізмам неблокуючого введення-виведення (Non-blocking I/O), що значно ефективніше для високонавантажених динамічних систем порівняно з класичними синхронними підходами.

Архітектуру CyberCore CMS логічно розділено на три базові контейнери:

- 1) Single Page Application (React SPA). Клієнтський рівень застосунку, скомпільований у форматі статичних файлів і призначений для виконання безпосередньо у браузері користувача. Контейнер імплементує інтерфейси для всіх категорій акторів: від WYSIWYG-редактора Журналіста та конструктора турнірних сіток Менеджера турнірів до публічного адаптивного порталу для вболівальників. Обмін даними із серверною частиною здійснюється асинхронно через протоколи REST API (HTTP/JSON).

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		47

- 2) Backend API (Node.js / NestJS). Серверний рівень, відповідальний за реалізацію бізнес-логіки. Цей контейнер обробляє вхідні запити від клієнтського SPA, управляє процесами автентифікації на базі JWT-токенів, забезпечує рольову валідацію доступу (RBAC), адмініструє життєвий цикл медіаматеріалів та ініціює виконання фонових процесів (каскадне оновлення турнірних таблиць і перерахунок KDA-статистики).
- 3) Database (PostgreSQL). Рівень постійного зберігання даних, що містить повну реляційну структуру системи. Механізми транзакційності СУБД PostgreSQL гарантують абсолютну цілісність каскадних змін у таблицях матчів та статистики, унеможливлюючи ризик розсинхронізації інформації. Взаємодія між Backend API та базою даних забезпечується на рівні протоколу TCP із застосуванням інструментарію Prisma ORM.

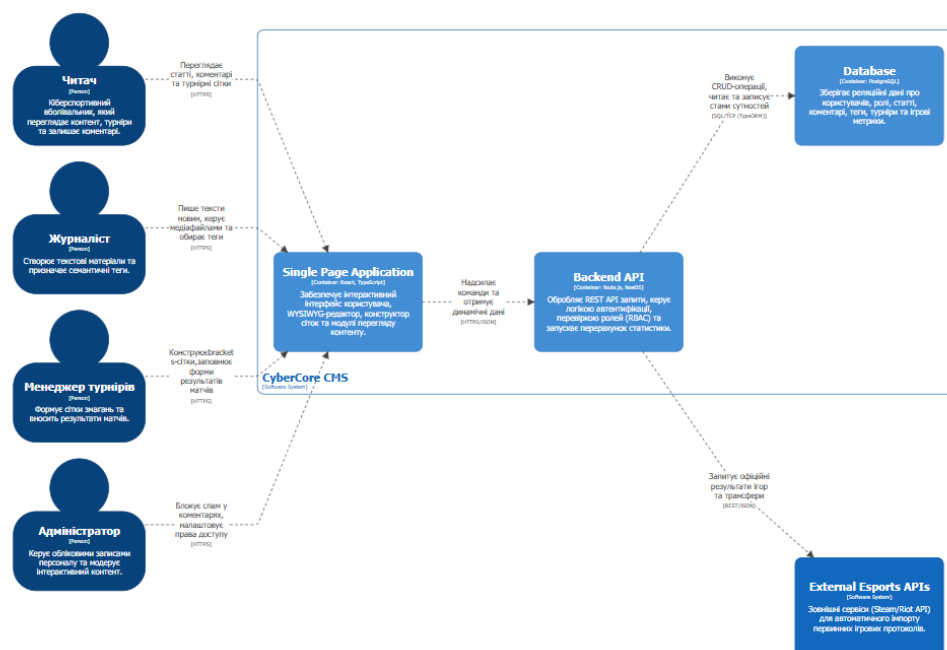


Рисунок 2.8 – C4 діаграма контейнерів

Спроектowana трирівнева декомпозиція дозволяє надійно ізолювати шар представлення від бізнес-логіки та рівня зберігання даних. Це забезпечує

можливість гнучкого горизонтального масштабування кожного окремого компонента, що є критично важливим при пікових навантаженнях під час трансляції масштабних кіберспортивних змагань.

Відповідно до сучасних інженерних практик, висвітлених у рекомендаціях Amazon Web Services (AWS) [9], застосування формату Architectural Decision Records (ADR) є галузевим стандартом для документування процесу еволюції програмної архітектури. Ця методологія передбачає фіксацію не лише самого факту технологічного вибору, а й детального контексту ухвалення рішення, наявних обмежень, розглянутих альтернативних варіантів та потенційних наслідків [9].

Керуючись зазначеним підходом, невіддільним етапом проєктування кіберспортивної системи CyberCore CMS стало ухвалення та належне документування ключових архітектурних рішень у форматі ADR. У межах проєкту було розроблено три базові документи: вибір системи керування базами даних (ADR-001), вибір загального архітектурного стилю вебзастосунку (ADR-002), а також обґрунтування технологічного стека бекенду з відмовою від класичних синхронних фреймворків (ADR-003).

Повні тексти зазначених документів ADR із розгорнутим аналізом альтернатив та очікуваних наслідків наведено в додатку А до пояснювальної записки.

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ CYBERCORE CMS

3.1. Організація середовища розробки та базове налаштування технологічного стека

Розробка інформаційної системи CyberCore CMS розпочалася з етапу конфігурування базового робочого середовища та ініціалізації програмних проєктів. Як основну платформу для виконання серверного коду обрано середовище Node.js, що забезпечує асинхронну обробку подій. Ініціалізація бекенд-частини здійснювалася за допомогою інтерфейсу командного рядка фреймворку NestJS (Nest CLI). Це дозволило автоматично згенерувати стандартизовану архітектуру директорій, яка включає контролери, сервіси та модулі, забезпечуючи суворе дотримання принципів інверсії управління та впровадження залежностей. Завдяки цьому серверна частина отримала модульну структуру, готову до подальшого масштабування.

Наступним кроком стала інтеграція реляційної системи керування базами даних PostgreSQL. Для забезпечення ефективної взаємодії між серверною бізнес-логікою та рівнем збереження даних було розгорнуто інструментарій об'єктно-реляційного відображення Prisma ORM [10]. Процес конфігурування включав встановлен пакетів менеджером залежностей та ініціалізацію базового файлу конфігурації бази даних (рис. 3.1)

```
model player_stats {
  id          Int      @id @default(autoincrement())
  match_id    Int?
  player_id   Int?
  kills       Int?     @default(0)
  deaths      Int?     @default(0)
  assists     Int?     @default(0)
  matches     matches? @relation(fields: [match_id], references: [id], onDelete: Cascade, onUpdate: NoAction)
  users       users?   @relation(fields: [player_id], references: [id], onDelete: Cascade, onUpdate: NoAction)

  @@index([match_id], map: "idx_player_stats_match_id")
  @@index([player_id], map: "idx_player_stats_player_id")
}
```

Рисунок 3.1 – Фрагмент конфігураційного файлу Prisma з описом моделей даних

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		50

Розгортання клієнтської частини системи реалізовано на базі бібліотеки React із використанням сучасного інструментарію збірки проєктів, що гарантує швидкий процес розробки завдяки механізму гарячої заміни модулів (HMR) [11]. Архітектура фронтенд-додатка побудована за строгим компонентним підходом, де користувацький інтерфейс розділено на незалежні перевикористовувані блоки. Для забезпечення узгодженості та мінімізації помилок під час розробки, як на стороні клієнта, так і на сервері, в обох проєктах налаштовано правила компіляції коду мовою TypeScript [12]. Структура каталогів розробленої системи відображає розподіл на клієнтський та серверний застосунки (рис. 3.2).

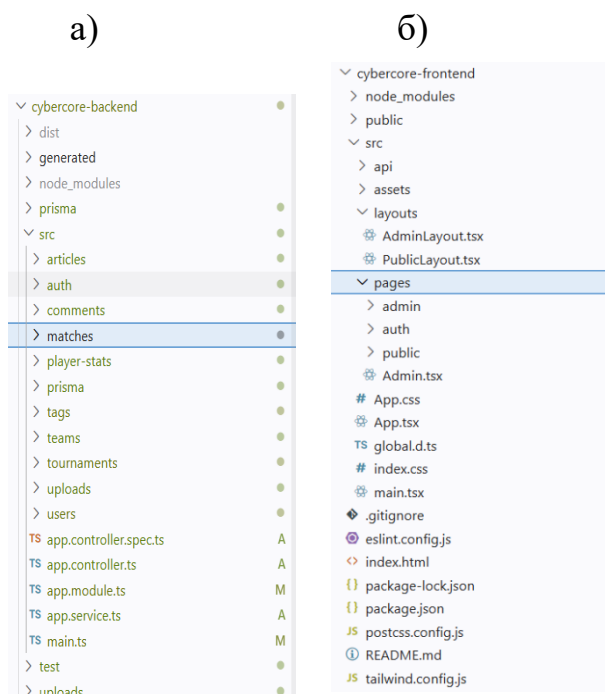


Рисунок 3.2 – Дерево каталогів: а) серверний застосунок; б) клієнтський застосунок

Такий комплексний підхід до організації робочого простору дозволив створити надійний програмний фундамент для переходу до безпосереднього написання бізнес-логіки та проєктування користувацьких інтерфейсів кіберспортивної медіаплатформи.

3.2 Програмна реалізація бізнес-логіки серверної частини

Програмна реалізація бізнес-логіки серверної частини інформаційної системи управління контентом онлайн-платформи у сфері кіберспорту базується на використанні фреймворку NestJS. Архітектура бекенду побудована за суворим модульним принципом, де кожна предметна область (користувачі, контент, турніри) виділена в окремий незалежний модуль, що містить власні контролери для обробки HTTP-запитів та сервіси для виконання бізнес-правил. Такий підхід забезпечує високу зв'язність всередині модулів та низьке зачеплення між ними.

Для забезпечення інформаційної безпеки та розмежування прав доступу реалізовано комплексну підсистему автентифікації та авторизації на основі JSON Web Tokens (JWT). Процес ідентифікації користувача відбувається шляхом перевірки криптографічного хешу пароля, після чого сервер генерує унікальний підписаний токен, який клієнтський застосунок використовує для подальших запитів (рис. 3.3). Валідація доступу до захищених маршрутів (ендпоінтів) реалізована за допомогою механізму Guards (Охоронці) у NestJS. Було розроблено спеціалізований RolesGuard, який перехоплює запит до його потрапляння в контролер, декодує JWT-токен та звіряє роль користувача із метаданими, необхідними для виконання конкретної операції.



```
1 import { Injectable, CanActivate, ExecutionContext, ForbiddenException } from '@nestjs/common';
2 import { Reflector } from '@nestjs/core';
3 @Injectable()
4 export class RolesGuard implements CanActivate {
5   constructor(private reflector: Reflector) {}
6   canActivate(context: ExecutionContext): boolean {
7     const requiredRoles = this.reflector.getAllAndOverride<number[]>('roles', [
8       context.getHandler(),
9       context.getClass(),
10    ]);
11    if (!requiredRoles) {
12      return true;
13    }
14    const { user } = context.switchToHttp().getRequest();
15    if (!user || !requiredRoles.includes(user.role_id)) {
16      throw new ForbiddenException('У вас недостатньо прав (не та роль) для виконання цієї дії');
17    }
18    return true;
19  }
20 }
```

Рисунок 3.3 – Програмна реалізація механізму рольової перевірки доступу

Модуль управління контентом відповідає за обробку операцій створення, редагування та публікації медіаматеріалів. Його програмна реалізація тісно пов'язана з використанням об'єктно-реляційного відображення Prisma ORM. У сервісі статей (ArticleService) реалізовано методи для асинхронного збереження чернеток та їх переведення у статус опублікованих. Особливістю даного модуля є програмна обробка зв'язків типу «багато-до-багатьох» між статтями та семантичними тегами (рис. 3.4). Під час створення публікації сервіс перевіряє наявність вказаних тегів у базі даних, створює нові за необхідності та автоматично формує записи в асоціативній проміжній таблиці article_tags в межах єдиної транзакції бази даних, що гарантує цілісність контенту.

```
@Injectable()
export class ArticlesService {
  constructor(private prisma: PrismaService) {}

  async create(createArticleDto: any) {
    const { title, content, image_url, author_id, tags } = createArticleDto;
    const dataToCreate: any = {
      title,
      content,
      image_url,
      author_id,
    };
    if (tags && Array.isArray(tags) && tags.length > 0) {
      dataToCreate.article_tags = {
        create: tags.map(({ tagName: string }) => ({
          tags: {
            connectOrCreate: {
              where: { name: tagName },
              create: { name: tagName }
            }
          }
        })))
      };
    }
    return this.prisma.articles.create({
      data: dataToCreate
    });
  }
}
```

Рисунок 3.4 – Реалізація методу створення медіапублікації з прив'язкою семантичних тегів

Для унеможливлення розсинхронізації даних (наприклад, коли рахунок матчу оновився, а індивідуальна статистика гравців не збереглася через програмний збій), усі операції запису інкапсульовано у транзакції Prisma

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		53

(prisma.\$transaction). Це забезпечує виконання принципів ACID: якщо хоча б один етап оновлення турнірної сітки або перерахунку статистики зазнає невдачі, система виконує автоматичний відкат (rollback) усіх змін до початкового стану (рис. 3.5). Крім того, застосування неблокуючого введення-виведення середовища Node.js дозволяє ефективно обробляти такі масивні транзакційні запити у фоновому режимі, не блокуючи основний потік виконання вебсервера і забезпечуючи високу швидкість відгуку для кінцевих користувачів.

```

async updateMatchResults(id: number, updateData: any) {
  const { score1, score2, status, playerStats } = updateData;
  return this.prisma.$transaction(async (prisma) => {
    const match = await prisma.matches.update({
      where: { id },
      data: { score1, score2, status },
    });
    if (playerStats && Array.isArray(playerStats)) {
      await prisma.player_stats.deleteMany({
        where: { match_id: id }
      });

      const statsData = playerStats.map((stat: any) => ({
        match_id: id,
        player_id: stat.player_id,
        kills: stat.kills,
        deaths: stat.deaths,
        assists: stat.assists,
      }));

      await prisma.player_stats.createMany({
        data: statsData
      });
    }
  });
  return match;
}

```

Рисунок 3.5 – Транзакційне оновлення результатів матчу та KDA-статистики гравців

Таким чином, розроблена серверна частина CyberCore CMS повністю задовольняє вимоги до продуктивності, безпеки та надійності обробки кіберспортивних даних, утворюючи стійкий програмний фундамент для клієнтського застосунку.

3.3 Розробка інтерфейсу користувача та клієнтської частини

Проектування та розробка клієнтської частини інформаційної системи CyberCore CMS реалізована за архітектурою Single Page Application (SPA). Базовим інструментом розробки обрано бібліотеку React у поєднанні зі строгими правилами типізації мови TypeScript. Застосування компонентного підходу дозволило декомпонувати складні користувацькі інтерфейси на дрібні, незалежні та перевикористовувані модулі (наприклад, картки новин, навігаційні панелі, поля пошуку), що суттєво спростило процес тестування та подальшого масштабування платформи.

Навігація між сторінками застосунку здійснюється на стороні клієнта за допомогою бібліотеки react-router-dom без перезавантаження сторінки у браузері. Для забезпечення безпеки користувацьких даних та розмежування зон відповідальності персоналу реалізовано механізм захищених маршрутів. Цей механізм перевіряє наявність валідного JWT-токена у локальному сховищі браузера та звіряє роль авторизованого користувача з правами доступу, необхідними для перегляду конкретної сторінки.

Важливим елементом клієнтської частини є публічний модуль агрегації та відображення медіаконтенту (стрічка новин). Інтерфейс розроблено у темній колірній схемі, що повністю відповідає сучасним UI/UX трендам кіберспортивних платформ (рис. 3.6). Для ефективної навігації великими масивами інформації реалізовано інтерактивні інструменти пошуку за текстовим входженням та фільтрації публікацій за семантичними тегами. Дані асинхронно завантажуються з бекенду і рендеряться за допомогою CSS Grid у вигляді адаптивної сітки карток. Кожна картка містить графічну обкладинку, заголовок, динамічний перелік тегів та службові метадані.

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		55

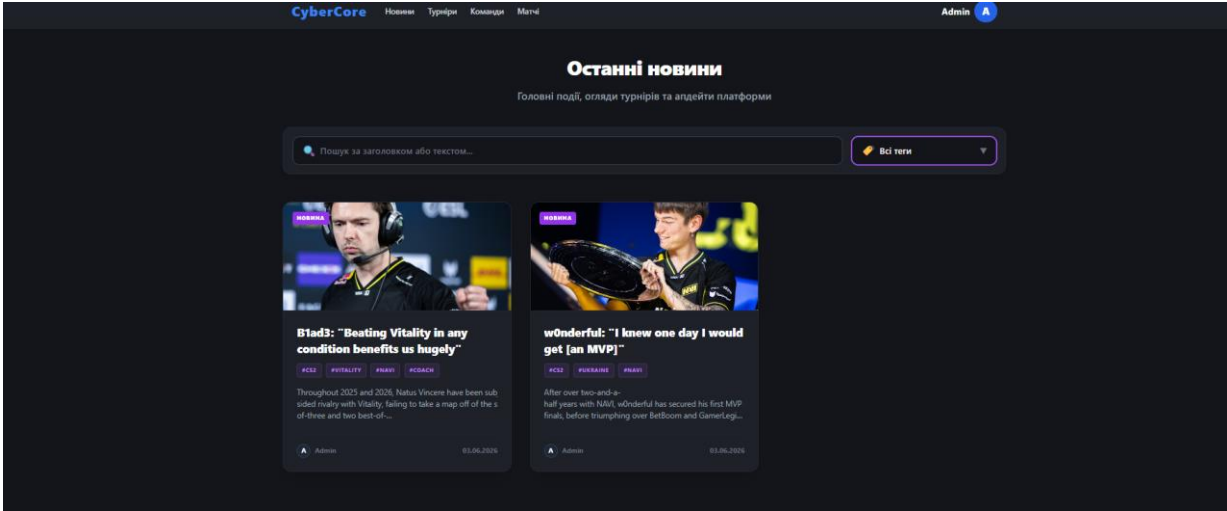


Рисунок 3.6 – Інтерфейс стрічки новин платформи CyberCore із модулем фільтрації

Для модуля управління контентом (робоче місце Журналіста) інтегровано спеціалізований текстовий WYSIWYG-редактор. Він надає редакційному персоналу інструменти для форматування тексту, інтеграції медіафайлів та структурування лонгвідів без необхідності написання HTML-коду. Інтерфейс створення публікації містить інтерактивний модуль для вибору семантичних тегів, який підтягує існуючі категорії з бази даних через API та дозволяє призначати їх кліком миші (рис. 3.7).

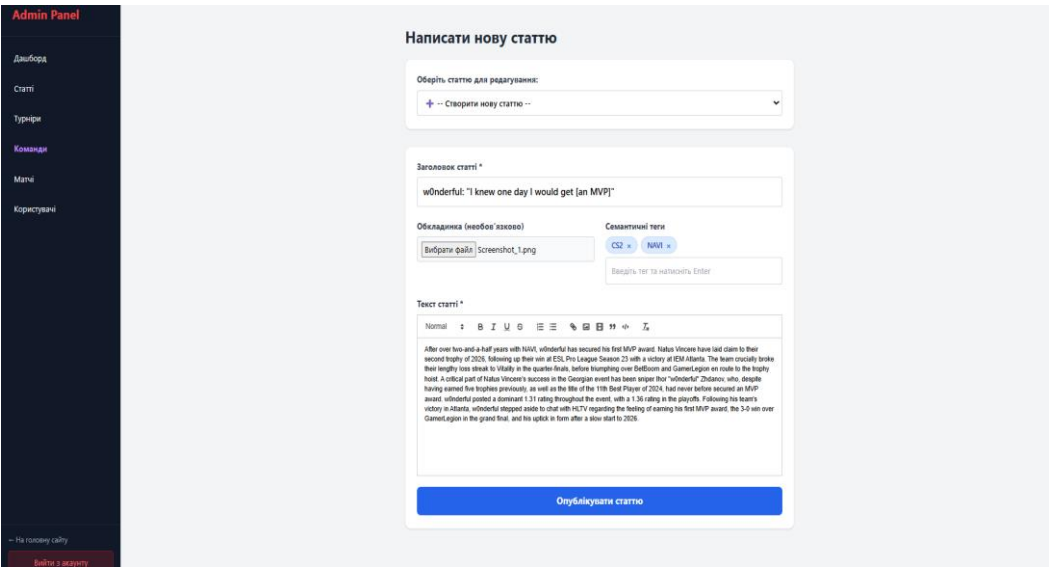


Рисунок 3.7 – Інтерфейс створення кіберспортивної публікації та призначення семантичних тегів

Робоче місце Менеджера турнірів реалізовано у вигляді комплексного дашборду. Центральним елементом цього модуля є візуальний конструктор турнірних сіток та панель управління змаганнями. Зокрема, під час ініціалізації нового турніру застосовано складну логіку обробки форм, де формат турніру динамічно генерується шляхом комбінації різних стадій (рис 3.8). Інтерфейс дозволяє взаємодіяти з кожним окремим матчем через модальні вікна. При натисканні на картку поєдинку система відкриває форму для внесення підсумкового рахунку та індивідуальної статистики гравців. Асинхронна відправка цих даних на сервер за допомогою HTTP-клієнта axios [13] запускає транзакційні процеси, після чого клієнтська частина миттєво оновлює візуальне відображення турнірної таблиці завдяки механізмам реактивності React.

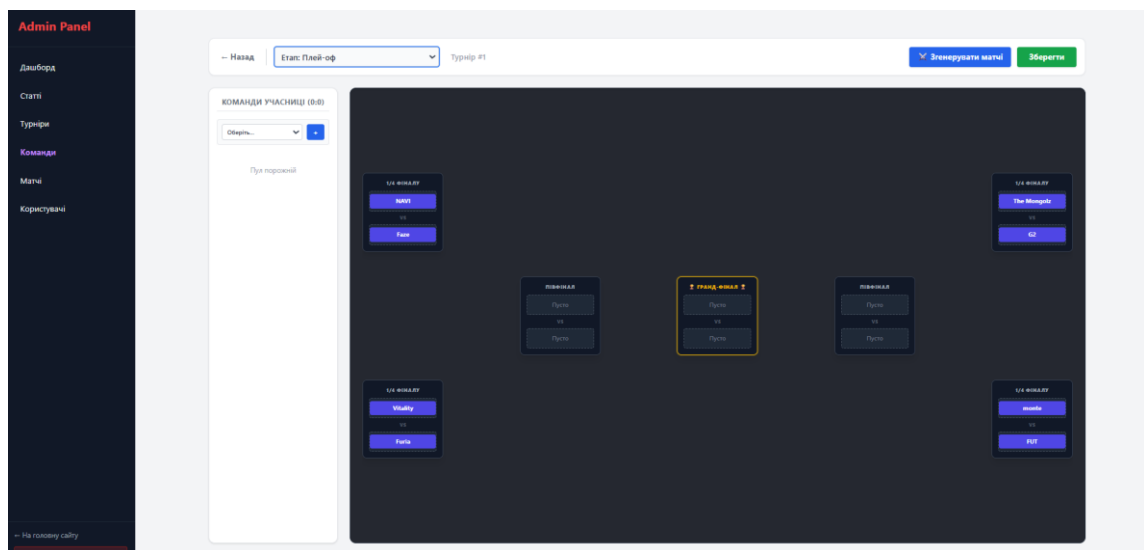


Рисунок 3.8 – Візуальне відображення турнірної сітки та інтерфейс управління подією

Для забезпечення чистоти архітектури клієнтської частини вся логіка взаємодії з серверним API була винесена в окремі сервісні модулі. Такий підхід інкапсулює налаштування HTTP-запитів і відокремлює шар отримання даних від шару їх візуального представлення. Детальні програмні реалізації базового налаштування HTTP-клієнта, приклади React-компонентів стрічки новин та панелі управління турнірами винесено до Додатку Б.

3.4 Забезпечення інформаційної безпеки та тестування системи

Під час розробки інформаційної системи CyberCore CMS особливу увагу було приділено питанням інформаційної безпеки, оскільки платформа працює з персональними даними користувачів та забезпечує публікацію медіаконтенту. Захист серверної частини базується на вбудованих механізмах фреймворку NestJS. Для запобігання атакам типу XSS (міжсайтовий скриптинг) та підробці HTTP-заголовків було налаштовано модуль Helmet. Захист від несанкціонованих міждоменних запитів реалізовано шляхом суворого конфігурування політики CORS, яка дозволяє приймати запити виключно з домену клієнтського застосунку.

Безпека збереження облікових даних персоналу забезпечується використанням криптографічної бібліотеки bcrypt. Усі паролі хешуються перед записом у базу даних, що унеможливлює їх компрометацію навіть у випадку витоку бази даних..

Окремим рівнем захисту є валідація та санітизація вхідних даних. Для цього в системі застосовано паттерн DTO (Data Transfer Object) у поєднанні з бібліотекою class-validator [14]. Кожен запит від клієнта проходить автоматичну перевірку на відповідність типам даних, наявність обов'язкових полів та відсутність шкідливого коду ще до моменту потрапляння в контролер бізнес-логіки (рис. 3.9).

```
import { ApiProperty } from '@nestjs/swagger';
import { IsDateString, IsNotEmpty, IsString } from 'class-validator';

export class CreateTournamentDto {
  @ApiProperty({ description: 'Назва турніру', example: 'IEM Cologne' })
  @IsString()
  @IsNotEmpty()
  name: string;

  @ApiProperty({ description: 'формат проведення', example: 'Single Elimination' })
  @IsString()
  @IsNotEmpty()
  format: string;

  @ApiProperty({ description: 'Дата початку турніру', example: '2026-08-15' })
  @IsDateString({}, { message: 'Дата початку має бути у форматі ISO (наприклад, YYYY-MM-DD)' })
  @IsNotEmpty()
  start_date: string;

  @ApiProperty({ description: 'Дата завершення турніру', example: '2026-08-25' })
  @IsDateString({}, { message: 'Дата завершення має бути у форматі ISO' })
  @IsNotEmpty()
  end_date: string;
}
```

Рисунок 3.9 – Програмна реалізація DTO-моделі для валідації вхідних даних

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		58

Після завершення стадії розробки систему було піддано комплексному тестуванню. Функціональне тестування серверного API проводилося за допомогою інструменту Swagger UI [15]. Під час тестування перевірялася коректність обробки HTTP-запитів, формування відповідних статус-кодів та робота механізмів авторизації. Наприклад, спроба звернутися до захищеного маршруту без валідного JWT-токена коректно відхиляється системою з помилкою 401 Unauthorized (рис. 3.10).



Рисунок 3.10 – Результати функціонального тестування REST API

Тестування клієнтської частини (Frontend) включало перевірку адаптивності інтерфейсу на різних роздільних здатностях екрана, тестування навігації та перевірку коректності роботи захищених маршрутів (Protected Routes). Окремо перевірялася реактивність системи: при додаванні результатів матчу в адміністративній панелі, стрічка подій та візуальна сітка турніру на клієнтській стороні миттєво оновлювалися завдяки синхронізації станів у React. Фрагменти коду, що відповідають за забезпечення валідації даних на сервері, винесено до Додатку В.

Успішні результати тестування підтвердили, що розроблена система працює стабільно, має високий рівень захисту від базових векторів кібератак та повністю відповідає заявленим функціональним вимогам.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання – виконано проєктування та розробку інформаційної системи управління контентом онлайн-платформи у сфері кіберспорту (CyberCore CMS). Створений програмний продукт дозволяє автоматизувати роботу редакційного персоналу та менеджерів змагань, утворюючи єдиний інформаційний простір для взаємодії зі спільнотою вболівальників.

За підсумками виконання роботи зроблено такі основні висновки:

- 1) На етапі системного та бізнес-аналізу досліджено предметну область кіберспортивних медіа. Використання методологій BPMN 2.0 та IDEF3 дозволило чітко розмежувати зони відповідальності між автоматизованими сервісами та персоналом. Побудова карт шляху користувача (UJM) для акторів «Журналіст» та «Менеджер турнірів» виявила критичні проблеми (людський фактор при введенні масивів статистики, ризик втрати незбереженого тексту), що лягло в основу архітектурних вимог до нової системи.
- 2) У процесі інформаційного проєктування розроблено надійну архітектуру за методологією C4. Виконано концептуальне, логічне (ER-діаграми) та фізичне проєктування бази даних. Вибір реляційної СКБД PostgreSQL повністю виправдав себе, забезпечивши підтримку складних зв'язків між сутностями (статтями, тегами, матчами, ігровою статистикою) та гарантуючи відповідність принципам ACID. Усі ключові рішення були успішно задокументовані за стандартом ADR.
- 3) Програмну реалізацію серверної частини здійснено за допомогою фреймворку NestJS на базі середовища Node.js із суворою типізацією TypeScript. Завдяки подієво-орієнтованій архітектурі та впровадженню Prisma ORM вдалося реалізувати складні транзакційні алгоритми – зокрема, механізм синхронного каскадного оновлення турнірних сіток та перерахунку індивідуальної KDA-статистики гравців без ризику розсинхронізації баз даних.

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		60

- 4) Розробка клієнтської частини за моделлю Single Page Application на базі React забезпечила створення адаптивного та інтерактивного інтерфейсу користувача. Реалізовано публічну стрічку новин з динамічною фільтрацією за тегами, модулі адміністрування з вбудованим WYSIWYG-редактором для журналістів, а також комплексний конструктор змагань для гнучкого налаштування стадій турнірів.
- 5) Впровадження підсистеми кібербезпеки на базі JWT-токенів та рольової моделі доступу забезпечило надійне розмежування прав (Адміністратор, Менеджер, Журналіст, Читач). Завдяки використанню DTO-моделей та інструментарію санітизації даних система захищена від поширених вразливостей.
- 6) Функціональне тестування підтвердило стабільність роботи системи, високу швидкість обробки клієнтських запитів та коректність виконання закладеної бізнес-логіки. Мета кваліфікаційної роботи досягнута в повному обсязі. Створена інформаційна система CyberCore CMS є масштабованим, безпечним та готовим до використання програмним рішенням, що може бути ефективно впроваджене в діяльність кіберспортивних організацій та профільних медіаресурсів.

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		61

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. HLTV. URL: <https://www.hltv.org/> (дата звернення: 10.05.2026).
2. ESL Pro Tour – The Ultimate CS Competition. *ESL Pro Tour*. URL: <https://pro.eslgaming.com/tour/cs/> (дата звернення: 10.05.2026).
3. Liquipedia Hub Wiki. *Liquipedia*. URL: https://liquipedia.net/hub/Main_Page (дата звернення: 10.05.2026).
4. Unwin D. IDEF0 – a neglected but very useful enterprise architecture tool. *LinkedIn*. URL: <https://www.linkedin.com/pulse/idef0-neglected-very-useful-enterprise-architecture-tool-unwin-maicd/> (дата звернення: 12.05.2026).
5. GeeksforGeeks. Use Case Diagram – Unified Modeling Language (UML) – GeeksforGeeks. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/system-design/use-case-diagram/> (дата звернення: 14.05.2026).
6. GeeksforGeeks. What is DFD (Data Flow Diagram) – GeeksforGeeks. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/software-engineering/what-is-dfddata-flow-diagram/> (дата звернення: 16.05.2026).
7. Rigorous Semantics for BPMN 2. 0 Process Diagrams / F. Kossak та ін. Springer, 2016. 235 с.
8. C4 model. URL: <https://c4model.com/> (дата звернення: 17.05.2026).
9. ADR process. URL: <https://docs.aws.amazon.com/prescriptive-guidance/latest/architectural-decision-records/adr-process.html> (дата звернення: 20.05.2026).
10. Prisma: Next-generation Node.js and TypeScript ORM for Databases [Електронний ресурс]. – Режим доступу: <https://www.prisma.io/docs>
11. React – A JavaScript library for building user interfaces [Електронний ресурс]. – Режим доступу: <https://react.dev/>
12. TypeScript: Typed JavaScript at Any Scale [Електронний ресурс]. – Режим доступу: <https://www.typescriptlang.org/docs/>

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		62

13. Axios: Promise based HTTP client for the browser and node.js [Електронний ресурс]. – Режим доступу: <https://axios-http.com/docs/intro>
14. NestJS: A progressive Node.js framework [Електронний ресурс]. – Режим доступу: <https://docs.nestjs.com/>
15. OpenAPI Specification [Електронний ресурс] / Swagger. – Режим доступу: <https://swagger.io/specification/>

					КРБ.ІСТ-16.00.00.000 ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТОК А

ADR ДОКУМЕНТИ

ADR-001: Вибір системи керування базами даних (СУБД)

Статус: Ухвалено

Контекст:

Проект CyberCore CMS розробляється як онлайн-платформа для управління кіберспортивним медіаконтентом та турнірними даними. Головною бізнес-метою є прискорення роботи редакції на 40%. Специфіка системи передбачає зберігання складної реляційної структури: кожен турнір містить десятки матчів, кожен матч генерує індивідуальну статистику для 10 гравців (K/D/A), а статті пов'язані з множиною тегів та коментарів. Крім того, під час проведення великих кіберспортивних турнірів (Majors) база даних повинна витримувати високе транзакційне навантаження при одночасному оновленні рахунків та публікації новин. Потрібне сховище, здатне гарантувати цілісність зв'язків та забезпечувати каскадні оновлення.

Рішення:

У якості основної системи керування базами даних (СУБД) обрано об'єктно-реляційну базу даних PostgreSQL.

Обґрунтування:

- **Гарантія цілісності даних (ACID):** У системі реалізовано суворий зв'язок між турнірами, матчами та статистикою. Використання реляційної моделі з жорсткими зовнішніми ключами (Foreign Keys) і каскадними обмеженнями (ON DELETE CASCADE) гарантує, що при видаленні матчу не залишиться "осиротілої" статистики гравців. Транзакційність унеможливорює розсинхронізацію турнірної сітки при паралельному введенні результатів кількома менеджерами.
- **Аналітичні можливості:** PostgreSQL дозволяє ефективно виконувати складні агрегаційні запити (наприклад, вирахувати середній Win-rate або K/D гравця за весь сезон), що є критично важливим для кіберспортивної аналітики.
- **Робота з JSONB:** PostgreSQL чудово підтримує тип даних JSONB, що залишає можливість у майбутньому зберігати гнучкі неструктуровані дані (наприклад, сирі логи з ігрових API) без зміни основної схеми бази.

Відхилені альтернативи:

- **MongoDB (NoSQL):** Відхилено. Документно-орієнтовані бази гірше справляються з підтримкою суворих реляційних зв'язків (наприклад, багаторівнева залежність Турнір-Матч-Статистика). Перенесення відповідальності за транзакційність на рівень коду бекенду призвело б до ризику пошкодження спортивних метрик.

- **MySQL / MariaDB:** Відхилено. Попри високу швидкість, інструментарій PostgreSQL для роботи зі складними агрегаціями, віконними функціями та JSONB історично є більш продуктивним для аналітичних спортивних систем.

Наслідки:

- **Позитивні:** Система отримала абсолютну стійкість до втрати та розсинхронізації спортивних даних, сувору валідацію зв'язків на рівні БД.
- **Негативні/Ризики:** Жорстка схема вимагає створення нових міграцій БД при кожній зміні формату турнірів або додаванні нових ігрових показників.

ADR-002: Вибір загального архітектурного стилю додатка

Статус: Ухвалено

Контекст:

Система CyberCore CMS має два принципово різних вектори використання: закритий бек-офіс для персоналу (складні форми, конструктор турнірних сіток, WYSIWYG-редактор) та публічна частина для читачів. Для забезпечення високої швидкості роботи менеджерів турнірів, рендеринг інтерактивних сіток (brackets) має відбуватися миттєво. Водночас прогнозується хвилеподібне навантаження: у період відсутності турнірів активність мінімальна, а під час фіналів (live-матчів) навантаження на систему може зростати в десятки разів.

Рішення: Вибрано архітектурний стиль «Розподілений модульний моноліт (Decoupled Client-Server)». Клієнтська частина бек-офісу реалізується як відокремлений Single Page Application (React SPA), а серверна — як ізольований RESTful API моноліт.

Обґрунтування:

- **Висока інтерактивність бек-офісу:** Використання SPA переносить обчислювально важкі завдання (перетягування команд у сітці, редагування статей) на сторону браузера. Це забезпечує миттєвий зворотний зв'язок без перезавантаження сторінки.
- **Незалежне масштабування:** Оскільки клієнт (статичні файли SPA) і сервер (API) розділені, під час пікових навантажень можна масштабувати лише бекенд-сервери (через балансувальник), а фронтенд віддавати через глобальні CDN (Content Delivery Network) з нульовим навантаженням на сервери бази даних.
- **Чіткий поділ відповідальності:** Фронтенд-розробники можуть фокусуватися виключно на UI/UX реактивних дашбордів, а бекенд-інженери — на оптимізації API та агрегації спортивної статистики.

Відхилені альтернативи:

- **Мікросервісна архітектура (Microservices):** Відхилено. Розділення системи на мікросервіси (сервіс статей, сервіс турнірів, сервіс користувачів) створило б

величезний інфраструктурний оверхед (Kubernetes, Service Mesh) для відносно невеликої команди. Моноліт з модульною структурою є оптимальним рішенням.

- **Server-Side Rendering (SSR) для бек-офісу:** Відхилено. Класичні багатосторінкові додатки (де HTML генерується на сервері) є занадто повільними для реалізації інтерактивних конструкторів турнірних сіток (Brackets) та динамічного введення рахунків.

Наслідки:

- **Позитивні:** Висока швидкість реакції інтерфейсу (UX), що прямо впливає на досягнення бізнес-мети (прискорення роботи редакції).
- **Негативні/Ризики:** Розробникам необхідно підтримувати два окремих репозиторії (фронтенд і бекенд), що ускладнює налаштування CI/CD пайплайнів порівняно зі звичайним монолітом.

ADR-003: Вибір технологічного стека для реалізації клієнт-серверної взаємодії

Статус: Ухвалено

Контекст:

Згідно з обраною архітектурою, серверна частина (Backend API) має обробляти REST-запити від React SPA. Особливістю кіберспортивної платформи є необхідність обробки асинхронних подій: отримання live-результатів від зовнішніх API, миттєве фонове збереження чернеток статей та каскадний перерахунок метрик (K/D/A) після завершення матчу, який не повинен блокувати роботу менеджера турнірів. Потрібен технологічний стек, який найкраще справляється з паралельним введенням-виведенням (I/O) та забезпечує сувору типізацію складних спортивних даних.

Рішення: Для реалізації Backend API обрано екосистему Node.js з використанням фреймворку NestJS та мови TypeScript.

Обґрунтування:

- **Асинхронність та неблокуючий I/O:** Node.js базується на подієво-орієнтованій моделі (Event Loop). Це дозволяє бекенду ефективно обробляти сотні одночасних мережових запитів (збереження даних, запити до Riot/Steam API) без виділення окремого потоку оперативної пам'яті на кожен запит. Це ідеально підходить для live-оновлень спортивних подій.
- **Суворая типізація (TypeScript):** Кіберспортивні дані мають складну структуру. Використання TypeScript на бекенді дозволяє описувати чіткі інтерфейси (Interfaces) для матчів та статистики, що нівелює до 40% потенційних помилок (runtime errors) ще на етапі написання коду.

- **Архітектурна модульність:** Фреймворк NestJS змушує розробників застосовувати паттерни Dependency Injection та чітко ділити код на контролери (Controllers), сервіси (Services) та модулі. Це забезпечує високу якість коду на рівні Enterprise-стандартів.

Відхилені альтернативи:

- **Ruby on Rails:** Відхилено. Незважаючи на високу швидкість створення MVP, Rails є синхронним фреймворком (по одному потоку на запит), що робить його менш ефективним при масовій асинхронній роботі з мережею (наприклад, постійне опитування зовнішніх ігрових API). Крім того, динамічна типізація Ruby збільшує ризик помилок при роботі зі складною статистикою матчів порівняно з TypeScript.
- **PHP / Laravel:** Відхилено. Аналогічно до Rails, класичний PHP є синхронним. Крім того, використання Node.js дозволяє розробникам писати як фронтенд (React), так і бекенд на одній мові (JavaScript/TypeScript), що дозволяє команді бути крос-функціональною (Full-stack).

Наслідки:

- **Позитивні:** Бекенд оптимізований для високих мережевих навантажень. Команда використовує єдину мову (TypeScript) на всьому проєкті, що дозволяє перевикористовувати інтерфейси між фронтендом та бекендом.
- **Негативні/Ризики:** Фреймворк NestJS має значно вищий поріг входження та більш стрімку криву навчання (learning curve) порівняно з простішими інструментами (на кшталт Express.js або Ruby on Rails), що може сповільнити онбординг нових розробників у перші тижні роботи.

ДОДАТОК Б

Налаштування HTTP-клієнта для взаємодії з API

```
import axios from 'axios';

const api = axios.create({
  baseURL: 'http://localhost:3000/api/v1',
});

api.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem('token');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);

export default api;
```

Фрагмент React-компонента відображення публічної стрічки новин (Articles)

```
import { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import api from '../api/axios';

export default function Articles() {
  const [articles, setArticles] = useState<any[]>([]);
  const [loading, setLoading] = useState(true);

  const [searchTerm, setSearchTerm] = useState("");
  const [selectedTag, setSelectedTag] = useState("");

  useEffect(() => {
    const fetchArticles = async () => {
      try {
```

```

    const response = await api.get('/articles');
    setArticles(Array.isArray(response.data) ? response.data : []);
  } catch (error) {
    console.error('Помилка завантаження новин', error);
  } finally {
    setLoading(false);
  }
};

fetchArticles();
}, []);

if (loading) return <div className="min-h-screen bg-[#14151a] p-10 text-center font-bold text-gray-500">Завантаження новин...</div>;

const allTags = Array.from(
  new Set(
    articles.flatMap(article =>
      article.article_tags?.map((at: any) => at.tags?.name) || []
    ).filter(Boolean)
  )
).sort();

const filteredArticles = articles.filter(article => {
  const matchesSearch =
    article.title.toLowerCase().includes(searchTerm.toLowerCase()) ||
    (article.content && article.content.toLowerCase().includes(searchTerm.toLowerCase()));

  const matchesTag = selectedTag
    ? article.article_tags?.some((at: any) => at.tags?.name === selectedTag)
    : true;

  return matchesSearch && matchesTag;
});

return (
  <div className="min-h-screen bg-[#14151a] py-12 px-4 sm:px-6 lg:px-8">
    <div className="max-w-7xl mx-auto">
      <div className="text-center mb-10">
        <h1 className="text-4xl font-black text-white tracking-tight mb-4">Останні новини</h1>
        <p className="text-lg text-gray-400 font-medium">Головні події, огляди турнірів та апдейти платформи</p>
      </div>

```

```

<div className="flex flex-col md:flex-row gap-4 mb-12 bg-[#1e2128] p-4 rounded-2xl border border-gray-800 shadow-lg">
  <div className="flex-1 relative">
    <span className="absolute left-4 top-1/2 -translate-y-1/2 text-gray-500 text-xl">🔍</span>
    <input
      type="text"
      placeholder="Пошук за заголовком або текстом..."
      value={searchTerm}
      onChange={(e) => setSearchTerm(e.target.value)}
      className="w-full bg-[#14151a] border border-gray-700 text-white p-3 pl-12 rounded-xl focus:outline-none focus:border-purple-500 font-medium"
    />
  </div>
  <div className="md:w-64 relative">
    <span className="absolute left-4 top-1/2 -translate-y-1/2 text-gray-500 text-xl">🏷️</span>
    <select
      value={selectedTag}
      onChange={(e) => setSelectedTag(e.target.value)}
      className="w-full bg-[#14151a] border border-gray-700 text-white p-3 pl-12 pr-10 rounded-xl focus:outline-none focus:border-purple-500 font-medium cursor-pointer appearance-none"
    >
      <option value="">Всі теги</option>
      {allTags.map(tag => (
        <option key={tag as string} value={tag as string}>{tag as string}</option>
      ))}
    </select>
    <div className="absolute right-4 top-1/2 -translate-y-1/2 pointer-events-none text-gray-500">▼</div>
  </div>
</div>

{filteredArticles.length === 0 ? (
  <div className="text-center text-gray-500 py-16 font-medium border border-gray-800 rounded-2xl bg-[#1e2128]">
    <div className="text-5xl mb-4 opacity-50">📄</div>
    За вашим запитом нічого не знайдено.
  </div>
) : (
  <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">
    {filteredArticles.map((article) => (
      <Link

```

```

key={article.id}
to={` /articles/${article.id}`}

className="bg-[#1e2128] rounded-2xl overflow-hidden shadow-xl border border-gray-800 transition-all
hover:border-purple-500 flex flex-col group"
>

<div className="h-48 bg-[#1a1c23] relative overflow-hidden border-b border-gray-800">
  {article.image_url || article.image ? (
    <img
      src={article.image_url || article.image}
      alt={article.title}
      className="w-full h-full object-cover group-hover:scale-105 transition-transform"
    />
  ) : (
    <div className="w-full h-full bg-gradient-to-br from-gray-800 to-gray-900 flex items-center justify-
center">
      <span className="text-6xl opacity-20">🔒</span>
    </div>
  )}

  <div className="absolute top-4 left-4 bg-purple-600 text-white text-[10px] font-black px-2 py-1 rounded
uppercase shadow-lg">
    Новина
  </div>
</div>

<div className="p-6 flex flex-col flex-1">
  <h3 className="text-xl font-black text-white mb-3 line-clamp-2 group-hover:text-purple-400 transition-
colors">
    {article.title}
  </h3>

  {article.article_tags && article.article_tags.length > 0 && (
    <div className="flex flex-wrap gap-2 mb-4">
      {article.article_tags.map((at: any) => (
        <span key={at.tags?.id} className="text-[10px] font-bold uppercase bg-purple-500/10 text-purple-400
border border-purple-500/20 px-2 py-1 rounded">
          #{at.tags?.name}
        </span>
      ))}
    </div>
  )}
}

```

```

<div
  className="text-gray-400 text-sm mb-6 line-clamp-3 flex-1 overflow-hidden"
  dangerouslySetInnerHTML={{ __html: article.content || 'Детальніше про цю подію читайте всередині...'
}}
/>

<div className="flex items-center justify-between text-xs font-bold text-gray-500 pt-4 border-t border-gray-
800/50 mt-auto">
  <span className="flex items-center gap-2">
    <div className="w-6 h-6 rounded-full bg-gray-800 border border-gray-700 flex items-center justify-center
text-white shadow-inner">
      {article.users?.username?.charAt(0).toUpperCase() || 'A'}
    </div>
    {article.users?.username || 'Адміністрація'}
  </span>
  <span>
    {article.created_at ? new Date(article.created_at).toLocaleDateString('uk-UA') : 'Сьогодні'}
  </span>
</div>
</div>
</Link>
  )))
</div>
  ))
</div>
</div>
);
}

```

React-компонент панелі управління турнірами

```

import { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import api from '../api/axios';

interface Tournament {
  id: number;
  name: string;
  format: string;
  start_date: string;
  end_date: string;
}

export default function ManageTournaments() {

```

```

const [tournaments, setTournaments] = useState<Tournament[]>([]);
const [loading, setLoading] = useState(true);
const [message, setMessage] = useState({ text: "", type: "" });
const [showForm, setShowForm] = useState(false);

const [name, setName] = useState("");
const [groupFormat, setGroupFormat] = useState('Швейцарська система (Swiss)');
const [playoffFormat, setPlayoffFormat] = useState('Single Elimination');
const [startDate, setStartDate] = useState("");
const [endDate, setEndDate] = useState("");
const fetchTournaments = async () => {
  try {
    const response = await api.get('/tournaments');
    if (Array.isArray(response.data)) {
      setTournaments(response.data);
    } else {
      setTournaments([]);
    }
  } catch (error) {
    console.error('Помилка завантаження турнірів', error);
    setTournaments([]);
  } finally {
    setLoading(false);
  }
};

useEffect(() => {
  fetchTournaments();
}, []);

const handleCreate = async (e: React.FormEvent) => {
  e.preventDefault();
  setMessage({ text: "", type: "" });
  try {
    const combinedFormat = groupFormat === 'Без групової стадії'
      ? playoffFormat
      : `${groupFormat} → ${playoffFormat}`;

    await api.post('/tournaments', {
      name,
      format: combinedFormat,
    });
  } catch (error) {
    console.error('Помилка створення турніру', error);
    setMessage({ text: "Не вдалося створити турнір", type: "error" });
  }
};

```

```

    start_date: new Date(startDate).toISOString(),
    end_date: new Date(endDate).toISOString(),
  });

  setMessage({ text: 'Турнір успішно створено!', type: 'success' });
  setName("");
  setGroupFormat('Швейцарська система (Swiss)');
  setPlayoffFormat('Single Elimination');
  setStartDate("");
  setEndDate("");
  setShowForm(false);
  fetchTournaments();
} catch (error: any) {
  console.error(error);

  const serverError = error.response?.data?.message || 'Помилка при створенні турніру.';
  setMessage({ text: `Помилка: ${serverError}`, type: 'error' });
}
};

const handleDelete = async (id: number) => {
  if (!window.confirm('Ви впевнені, що хочете видалити цей турнір?')) return;
  try {
    await api.delete(`/tournaments/${id}`);
    setTournaments(tournaments.filter(t => t.id !== id));
    setMessage({ text: 'Турнір видалено!', type: 'success' });
  } catch (error) {
    setMessage({ text: 'Помилка при видаленні турніру.', type: 'error' });
  }
};

return (
  <div className="p-8 max-w-6xl mx-auto">

    <div className="flex justify-between items-center mb-6">
      <h1 className="text-3xl font-bold text-gray-800">Управління турнірами</h1>
      <button
        onClick={() => setShowForm(!showForm)}
        className={`px-6 py-2 rounded font-bold text-white transition shadow-sm ${
          showForm ? 'bg-gray-500 hover:bg-gray-600' : 'bg-green-600 hover:bg-green-700'
        }`}
      >

```

```

        {showForm ? 'Скасувати' : '+ Створити турнір'}
    </button>
</div>

{message.text && (
    <div className={`mb-6 p-4 rounded-lg font-bold text-center ${message.type === 'error' ? 'bg-red-100 text-red-700'
: 'bg-green-100 text-green-700'}`}>
        {message.text}
    </div>
)}
{showForm && (
    <form onSubmit={handleCreate} className="bg-white p-6 rounded-xl shadow border border-gray-100 mb-8
animate-fade-in">
        <h2 className="text-xl font-bold mb-4 border-b pb-2">Новий турнір</h2>

        <div className="mb-4">
            <label className="block text-gray-700 font-bold mb-2">Назва турніру *</label>
            <input
                type="text" required
                value={name} onChange={(e) => setName(e.target.value)}
                className="w-full p-2 border border-gray-300 rounded focus:border-blue-500 focus:outline-none"
                placeholder="CyberCore Major 2026"
            />
        </div>

        <div className="grid grid-cols-1 md:grid-cols-2 gap-4 mb-4 bg-gray-50 p-4 rounded border border-gray-200">
            <div>
                <label className="block text-gray-700 font-bold mb-2 text-sm">Групова стадія (Group Stage)</label>
                <select
                    value={groupFormat} onChange={(e) => setGroupFormat(e.target.value)}
                    className="w-full p-2 border border-gray-300 rounded focus:border-blue-500 focus:outline-none bg-white
cursor-pointer"
                >
                    <option value="Швейцарська система (Swiss)">Швейцарська система (Swiss)</option>
                    <option value="Кругова система (Round Robin)">Кругова система (Round Robin)</option>
                    <option value="GSL (Подвійне вибування)">GSL (Подвійне вибування в групах)</option>
                    <option value="Без групової стадії">Без групової стадії (Одразу плей-оф)</option>
                </select>
            </div>
            <div>
                <label className="block text-gray-700 font-bold mb-2 text-sm">Стадія Плей-оф (Playoffs)</label>

```

```

        <select
            value={playoffFormat} onChange={(e) => setPlayoffFormat(e.target.value)}
            className="w-full p-2 border border-gray-300 rounded focus:border-blue-500 focus:outline-none bg-white
cursor-pointer"
        >
            <option value="Single Elimination">Single Elimination (на ви́літ)</option>
            <option value="Double Elimination">Double Elimination (з нижньою сіткою)</option>
        </select>
    </div>
</div>

```

```

<div className="grid grid-cols-1 md:grid-cols-2 gap-4 mb-6">

```

```

    <div>

```

```

        <label className="block text-gray-700 font-bold mb-2">Дата початку *</label>

```

```

        <input

```

```

            type="date" required

```

```

            value={startDate} onChange={(e) => setStartDate(e.target.value)}

```

```

            className="w-full p-2 border border-gray-300 rounded focus:border-blue-500 focus:outline-none"

```

```

        />

```

```

    </div>

```

```

    <div>

```

```

        <label className="block text-gray-700 font-bold mb-2">Дата завершення *</label>

```

```

        <input

```

```

            type="date" required

```

```

            value={endDate} onChange={(e) => setEndDate(e.target.value)}

```

```

            className="w-full p-2 border border-gray-300 rounded focus:border-blue-500 focus:outline-none"

```

```

        />

```

```

    </div>

```

```

</div>

```

```

        <button type="submit" className="bg-blue-600 text-white px-6 py-2 rounded font-bold hover:bg-blue-700
transition">

```

```

            Зберегти турнір

```

```

        </button>

```

```

    </form>

```

```

  })

```

```

<div className="bg-white rounded-xl shadow overflow-hidden">

```

```

    <table className="w-full text-left border-collapse">

```

```

        <thead>

```

```

<tr className="bg-gray-900 text-white">
  <th className="p-4 border-b border-gray-700">ID</th>
  <th className="p-4 border-b border-gray-700">Назва</th>
  <th className="p-4 border-b border-gray-700">Формат турніру</th>
  <th className="p-4 border-b border-gray-700">Терміни проведення</th>
  <th className="p-4 border-b border-gray-700">Дії</th>
</tr>
</thead>
<tbody>
  {loading ? (
    <tr>
      <td colSpan={5} className="p-4 text-center text-gray-500">Завантаження турнірів...</td>
    </tr>
  ) : tournaments.length === 0 ? (
    <tr>
      <td colSpan={5} className="p-4 text-center text-gray-500 py-8">Турнірів поки немає. Натисніть "Створити турнір" вище!</td>
    </tr>
  ) : (
    tournaments.map((tournament) => (
      <tr key={tournament.id} className="hover:bg-gray-50 transition border-b border-gray-100">
        <td className="p-4 text-gray-600 font-bold">#{tournament.id}</td>
        <td className="p-4 font-bold text-gray-800">{tournament.name}</td>
        <td className="p-4">
          { /* Виводимо склеєний формат красивим бейджиком */ }
          <span className="bg-indigo-50 text-indigo-700 px-2 py-1 rounded border border-indigo-100 text-sm font-semibold">
            {tournament.format}
          </span>
        </td>
        <td className="p-4 font-semibold text-gray-700 text-sm">
          {new Date(tournament.start_date).toLocaleDateString('uk-UA')} - {new Date(tournament.end_date).toLocaleDateString('uk-UA')}
        </td>
        <td className="p-4 flex gap-2">
          <Link
            to={`\admin/tournaments/${tournament.id}/bracket`}
            className="bg-purple-100 text-purple-700 px-3 py-1 rounded hover:bg-purple-600 hover:text-white transition font-bold text-sm flex items-center">
            >

```

 Керувати сіткою

```
</Link>

<button
  onClick={() => handleDelete(tournament.id)}
  className="bg-red-50 text-red-600 px-3 py-1 rounded hover:bg-red-600 hover:text-white transition font-
bold text-sm"
  >
  Видалити
</button>
</td>
</tr>
))
)}
</tbody>
</table>
</div>
</div>
);
}
```

ДОДАТОК В

Реалізація DTO-класу для валідації створення матчу

```
import { ApiProperty } from '@nestjs/swagger';
import { IsInt, IsNotEmpty, IsOptional, IsString, IsIn } from 'class-validator';

export class CreateMatchDto {

  @ApiProperty({ description: 'ID турніру', example: 1 })
  @IsInt({ message: 'tournament_id має бути цілим числом' })
  @IsNotEmpty({ message: 'tournament_id є обов'язковим полем' })
  tournament_id: number;

  @ApiProperty({ description: 'ID першої команди', example: 1 })
  @IsInt({ message: 'team1_id має бути цілим числом' })
  @IsNotEmpty()
  team1_id: number;

  @ApiProperty({ description: 'ID другої команди', example: 2 })
  @IsInt({ message: 'team2_id має бути цілим числом' })
  @IsNotEmpty()
  team2_id: number;

  @ApiProperty({ description: 'Статус матчу', example: 'scheduled', required: false })
  @IsString()
  @IsOptional()
  @IsIn(['scheduled', 'live', 'completed'], { message: 'Статус має бути одним із: scheduled, live, completed' })
  status?: string;

  @ApiProperty({ description: 'Рахунок команди 1', example: 2, required: false })
  @IsInt()
  @IsOptional()
  score1?: number;

  @ApiProperty({ description: 'Рахунок команди 2', example: 0, required: false })
  @IsInt()
  @IsOptional()
  score2?: number;
```

```
}
```

Конфігурація глобального конвеєра валідації (ValidationPipe)

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { DocumentBuilder, SwaggerModule } from '@nestjs/swagger';
import { ValidationPipe } from '@nestjs/common';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  app.setGlobalPrefix('api/v1');

  app.useGlobalPipes(
    new ValidationPipe({
      whitelist: true,
      forbidNonWhitelisted: true,
      transform: true,
    }),
  );

  const config = new DocumentBuilder()
    .setTitle('CyberCore CMS API')
    .setDescription('REST API специфікація для кіберспортивної платформи')
    .setVersion('1.0')
    .addBearerAuth()
    .build();

  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('docs', app, document);

  app.enableCors();

  await app.listen(3000);
}

bootstrap();
```

Бібліографічна довідка

Тема роботи: Проектування та розробка інформаційної системи управління контентом онлайн-платформи у сфері кіберспорту

Обсяг бакалаврської роботи 63 сторінки.

Перелік графічного матеріалу:

1. Діаграма класів
2. C4 контекстна діаграма

Дата закінчення БР

Студент

Роман СЕМЕНЮК