

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Вебсервіс з проведення турнірних змагань»

Виконав студент групи ІПЗп-22-1
МУРАВИЦЬКИЙ Владислав Юрійович

Керівник роботи:
ТОЛСТОЙ Ігор Анатолійович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«16» липня 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **МУРАВИЦЬКИЙ Владислав Юрійович**

Керівник роботи: **ТОЛСТОЙ Ігор Анатолійович**

Тема роботи: **«Вебсервіс з проведення турнірних змагань»**,

затверджена Наказом закладу вищої освіти від **«16» липня 2026 р., №396/с**

Вихідні дані для роботи: об'єктом дослідження є процес організації, проведення та суддівства турнірних змагань з єдиноборств. Предмет дослідження – моделі, методи, алгоритми та програмні засоби автоматизації проведення турнірних змагань у вигляді вебсервісу.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Толстой І.А.	16.07.2026р.	16.07.2026р.
2	Толстой І.А.	16.07.2026р.	16.07.2026р.
3	Толстой І.А.	16.07.2026р.	16.07.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу та пояснювальної записки. Пояснювальна записка до випускної кваліфікаційної роботи містить текстову частину, викладену на 118 сторінках друкованого тексту. Список використаних джерел містить 48 найменувань. У роботі наведено 43 рисунки та 34 таблиці. Робота містить 7 додатків (А, Б, В, Г, Д, Е, Ж). Загальний обсяг роботи – 143 сторінки.

Метою роботи є розроблення вебзастосунку для управління повним циклом турнірних змагань з єдиноборств. Система вирішує проблеми дублювання даних, зводить до мінімуму людський фактор під час підрахунку результатів та забезпечує програмну підтримку суддівства в реальному часі на основі абстрактного рушія правил.

Використано об'єктно-орієнтований підхід до аналізу та проектування системи (з UML-моделюванням варіантів використання й класів), проектування реляційної бази даних з нормалізацією, модель машини станів для суддівства поєдинків, розширюваний рушій правил оцінювання (патерн «Стратегія»), що дозволяє додавати нові дисципліни без зміни ядра, а також алгоритми жеребкування методами теорії графів.

Спроектовано й реалізовано вебсервіс із підтримкою кількох дисциплін, автоматичною генерацією сіток, централізованим серверним суддівством із трансляцією стану в реальному часі, рольовим розмежуванням доступу та онлайн-оплатою внесків. Систему розгорнуто у хмарі з конвеєром CI/CD; коректність підтверджено понад 500 автоматизованими тестами з покриттям коду 91,6 %.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ, ВЕБСЕРВІС, ЄДИНОБОРСТВО, СУДДІВСТВО, ТУРНІРНА СІТКА.

					ІПЗ.КР.Б – 121 – 26 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Вебсервіс для проведення турнірних змагань Пояснювальна записка	Літ.	Арк.	Аркуші	
Розроб.		Муравицький В. Ю.							
Перевір.							3	143	
Керівник		Толстой І. А.				Житомирська політехніка, група ІПЗп-22-1			
Рецензент		Вакалюк Т. А.							
Зав. каф.		Вакалюк Т. А.							

ABSTRACT

The bachelor's qualification work consists of a software package and an explanatory note. The explanatory note contains a text section comprising 118 pages of printed text. The list of references contains 48 entries. The work includes 43 illustrations and 34 tables. It also contains 7 appendices. The total length of the work is 143 pages.

The objective of this thesis is to develop a web application for managing the full cycle of martial arts tournaments. The system solves the problem of data duplication, minimizes the human factor during the calculation of results, and provides real-time software support for refereeing based on an abstract rules engine.

An object-oriented approach was used for the analysis and design of the system (including UML modeling of use cases and classes), the design of a normalized relational database, a state machine model for match judging, an extensible scoring rules engine (the “Strategy” pattern), which allows new disciplines to be added without changing the core, as well as draw algorithms based on graph theory.

A web service was designed and implemented with support for multiple disciplines, automatic bracket generation, centralized server-based judging with real-time status updates, role-based access control, and online payment of fees. The system has been deployed in the cloud with a CI/CD pipeline; its correctness has been verified by over 500 automated tests with 91.6% code coverage.

KEYWORDS: AUTOMATION, WEB SERVICE, MARTIAL ARTS, REFEREEING, TOURNAMENT BRACKET.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

ЗМІСТ	5
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЧНИХ РІШЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ ТУРНІРНИХ ЗМАГАНЬ	10
1.1. Постановка задачі та аналіз бізнес-процесів організації змагань.....	10
1.2. Аналіз аналогів програмного продукту	13
1.3. Вибір архітектури веборієнтованої системи управління турнірами	18
1.4. Обґрунтування вибору технологічного стеку.....	21
1.5. Обмеження розробленої системи	27
Висновки до першого розділу	31
РОЗДІЛ 2. ПРОЄКТУВАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОЇ СТРУКТУРИ ТА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ	33
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	33
2.2. Проєктування бази даних системи	44
2.3. Проєктування математичного забезпечення та алгоритмів роботи системи	51
2.4. Реалізація програмного коду модулів серверної системи.....	62
2.5. Компонентна будова та реалізація клієнтського додатку	67
2.6. Аналіз загроз та забезпечення інформаційної безпеки системи	71
Висновки до другого розділу	76
РОЗДІЛ 3. ІНТЕРФЕЙС, РОЗГОРТАННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ	78
3.1. Порядок встановлення, налаштування та розгортання системи.....	78
3.2. Структура та основні елементи інтерфейсу.....	85
3.3. Тестування роботи програмного комплексу.....	98
3.4. Дослідження продуктивності системи	104
Висновки до третього розділу	108
ВИСНОВКИ	110
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	114
ДОДАТКИ	119

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ACID – вимоги до надійних транзакцій: атомарність, узгодженість, ізолюваність, стійкість (Atomicity, Consistency, Isolation, Durability)

API – програмний інтерфейс застосунку (Application Programming Interface)

ASGI – асинхронний шлюзовий інтерфейс сервера (Asynchronous Server Gateway Interface)

CI/CD – безперервна інтеграція та доставлення (Continuous Integration / Continuous Delivery)

CSRF – міжсайтова підробка запиту (Cross-Site Request Forgery)

CTE – узагальнений табличний вираз (Common Table Expression)

E2E – наскрізне тестування (End-to-End)

ER – модель «сутність–зв'язок» (Entity–Relationship)

JSON – текстовий формат обміну даними (JavaScript Object Notation); JSONB – його двійкове подання

ORM – об'єктно-реляційне відображення (Object-Relational Mapping)

RBAC – рольове розмежування доступу (Role-Based Access Control)

REST – передавання репрезентативного стану (Representational State Transfer)

SPA – односторінковий застосунок (Single Page Application)

SQL – мова структурованих запитів (Structured Query Language)

TLS – протокол захисту транспортного рівня (Transport Layer Security)

UML – уніфікована мова моделювання (Unified Modeling Language)

WKF – Всесвітня федерація карате (World Karate Federation)

БД – база даних

ООП – об'єктно-орієнтоване програмування

ПЗ – програмне забезпечення

СКБД – система керування базами даних

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Організація масштабних турнірів з контактних єдиноборств вимагає одночасної та безпомилкової обробки даних сотень учасників у стислі терміни, в умовах кількох татамі, паралельної роботи суддівських бригад і послідовних етапів – реєстрації, зважування, жеребкування та фіксації результатів поєдинків. Актуальність теми зумовлена тим, що традиційні підходи до організації таких змагань не відповідають цим вимогам.

Паперовий документообіг є основним джерелом втрат і затримок інформації: бланки зважування та протоколи поєдинків заповнюються вручну на кожному етапі, що вимагає повторного перенесення тих самих даних учасника (ПБ, вагова категорія, клуб) у кілька документів і створює умови для описок, втрати бланків та розбіжностей між паперовими носіями на різних майданчиках.

Використання розрізнених електронних таблиць лише частково усуває ці недоліки. Таблиці, які незалежно ведуть оператори реєстрації, зважування та суддівства, не синхронізуються між собою в реальному часі, не мають автоматичної перевірки коректності введених даних (наприклад, відповідності учасника заявленій ваговій категорії) і за одночасного редагування кількох користувачами призводять до конфліктів версій і втрати останніх змін.

Наслідком цього є дублювання інформації – дані про одного й того самого учасника вводяться незалежно кілька разів на різних етапах, що збільшує ймовірність розбіжностей між джерелами. Ручне формування турнірних сіток на сотні учасників є комбінаторно складною задачею: виконання її оператором вручну спричиняє критичні помилки – пропуски учасників, порушення принципу розведення представників одного клубу, некоректний розподіл за кваліфікацією. Зі зростанням кількості учасників такі помилки трапляються частіше, а затримка синхронізації відомостей між операторами на різних майданчиках безпосередньо призводить до затримок в оновленні результатів.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Ситуація ускладнюється специфікою контактних видів спорту. Учасників жорстко типізують за віком, вагою, статтю та кваліфікацією, правила оцінювання різняться навіть у межах одного виду спорту, а кілька суддівських бригад одночасно фіксують бали на різних татамі. Наявні на ринку рішення переважно є закритими комерційними продуктами. Їхня внутрішня логіка недоступна для модифікації, а адаптація під регламенти конкретних федерацій обмежена. Через це організатори підлаштовують власні процеси під можливості програми, а не навпаки. Окремою проблемою є слабе відокремлення загальної управлінської логіки від правил конкретної дисципліни, що ускладнює розширення системи новими видами спорту. Це обґрунтовує доцільність розроблення відкритого масштабованого вебсервісу з абстрактним рушієм правил та універсальними алгоритмами генерації турнірних сіток різних форматів.

Метою роботи є розроблення вебсервісу для управління, проведення та суддівства турнірних змагань з єдиноборств, що реалізує програмну генерацію турнірних сіток різних форматів і суддівство в реальному часі на основі абстрактного рушія правил. Для досягнення поставленої мети сформульовано такі завдання: проаналізувати проблеми предметної області та недоліки наявних програмних рішень; визначити архітектуру й узагальнену структуру системи для роботи в умовах нестабільного мережевого з'єднання; обґрунтувати вибір технологічного стеку та реляційної моделі баз даних для гарантування узгодженості результатів під час паралельного суддівства; спроектувати алгоритми генерації турнірних сіток різних форматів; розробити абстрактний рушій правил для підтримки кількох видів єдиноборств без зміни ядра системи; виконати програмну реалізацію компонентів вебсервісу та перевірити їхню коректність і продуктивність тестуванням.

Об'єктом дослідження є процеси управління, проведення та суддівства турнірних змагань з єдиноборств, зокрема етапи реєстрації учасників, жеребкування, зважування та фіксації результатів поєдинків, у межах яких за традиційних

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

(паперових та таблично-орієнтованих) підходів виникають дублювання даних і операторські помилки.

Предметом дослідження є методи, алгоритми та програмні засоби побудови веборієнтованих систем управління турнірами, що забезпечують усунення дублювання даних, мінімізацію операторських помилок та обробку інформації в реальному часі під час проведення масштабних змагань.

Для розв'язання поставлених задач у роботі поєднано кілька методів. Порівняльний аналіз застосовано для зіставлення наявних програмних аналогів і технологій реалізації. Математичне моделювання на основі теорії графів і комбінаторики використано для розроблення алгоритмів генерації турнірних сіток. Емпіричні методи, зокрема функціональне, інтеграційне та навантажувальне тестування, застосовано для перевірки коректності й продуктивності програмного комплексу.

Практичне значення одержаних результатів полягає у створенні інструментарію для спортивних клубів і федерацій, що забезпечує цифрову реєстрацію учасників, жеребкування та ведення протоколів у реальному часі. Завдяки API-центричній архітектурі й абстрактному рушієві правил систему можна адаптувати під регламенти різних дисциплін без зміни ядра. Програмний комплекс придатний до впровадження під час змагань із карате та інших контактних видів спорту й масштабується відповідно до кількості учасників і майданчиків.

Структура та обсяг проекту. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел із 48 найменувань (5 сторінок), має 7 додатків (25 сторінок), 34 таблиці та 43 ілюстрації. Повний обсяг роботи становить 143 сторінки, з них 118 сторінок основного тексту.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЧНИХ РІШЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ ТУРНІРНИХ ЗМАГАНЬ

1.1. Постановка задачі та аналіз бізнес-процесів організації змагань

Програмне забезпечення для турнірів з контактних єдиноборств (карате, дзюдо, тхеквондо) вимагає глибокого розуміння життєвого циклу таких заходів. Організація змагань є безперервним процесом, який структурно поділяється на кілька логічних етапів. Підготовчий етап охоплює електронну реєстрацію клубів, подання індивідуальних заявок та налаштування вагових і вікових категорій. Далі настає етап жеребкування, під час якого алгоритми генерують турнірні сітки та розводять бійців з одного клубу по різних гілках турнірного дерева. Безпосередньо в день турніру система переходить в активну фазу. Майданчики (татамі) працюють паралельно, судді фіксують результати поєдинків у реальному часі, а переможці просуваються сіткою до фіналу. Цей цикл повторюється до повного завершення змагань (рис. 1.1).

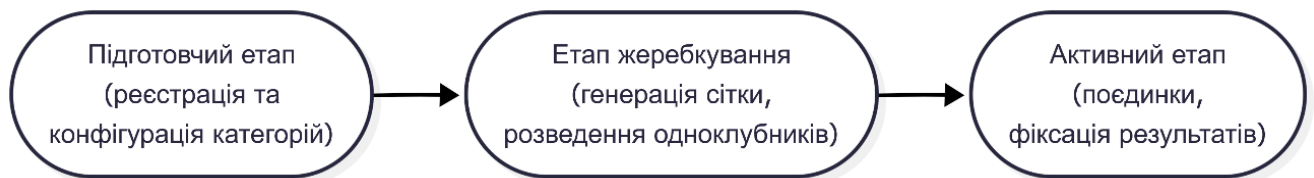


Рис. 1.1 – Схема бізнес-процесу проведення турніру

Програмне забезпечення для таких подій має працювати в умовах високого конкурентного навантаження. Оскільки суддівські оцінки надходять паралельно з кількох терміналів навколо татамі, виникає ризик станів гонитви (race conditions). Щоб не втратити жодного бала під час одночасного натискання кнопок, система повинна гарантувати транзакційну ізоляцію. Практично це означає застосування механізмів оптимістичного або песимістичного блокування записів на рівні бази даних для дотримання вимог ACID [1].

Першим алгоритмічно навантаженим процесом є конфігурація змагань. Формування категорій вимагає врахування віку, ваги, статі та рівня кваліфікації [2].

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

Організатори створюють конкретні дивізіони, моделюючи їх як класичні реляційні таблиці в PostgreSQL. Кожна категорія отримує чітко визначені стовпці: мінімальний та максимальний вік (min_age, max_age), межі ваги (min_weight, max_weight), а також дозволена стать (allowed_gender). Замість зберігання абстрактних напівструктурованих документів система спирається на строгу типізацію та декларативні обмеження цілісності бази даних. Під час електронної реєстрації алгоритм фільтрує базу учасників через стандартні SQL-запити з умовами BETWEEN для числових діапазонів, що дозволяє швидко розподілити спортсменів і гарантує відсутність порожніх перетинів без використання поліморфних структур чи складних нереляційних індексів [3].

Для обробки результатів поєдинків обрано архітектурний підхід Domain Model [4] разом із патерном Strategy [5]. Основна ідея полягає у відокремленні ядра системи від правил конкретного виду спорту (Open/Closed Principle) [6]. Наприклад, під час вільних поєдинків («куміте») система працює як кінцевий автомат. Кожна оцінка змінює стан матчу аж до термінального (зокрема, дострокової перемоги при відриві у 8 очок) [7]. Ситуація з дисципліною «ката» (демонстрацією техніки ведення бою без реального супротивника) інша. За відсутності електронної системи WKF вимагає мануального суддівства прапорцями. У такому разі архітектура адаптується: рушій працює як реєстратор, приймає результати з єдиного термінала оператора та фіксує підсумкове мажоритарне рішення однією транзакцією [8].

Побудова сіток математично спирається на теорію графів [9]. Звичайна олімпійська система (Single Elimination) представляється збалансованим бінарним деревом [10]. Кількість учасників рідко є ідеальним ступенем двійки, тому алгоритм доповнює дерево порожніми вузлами (Byes) для просування рейтингових спортсменів. Якщо формат передбачає вибування після двох поразок (Double Elimination), структура ускладнюється до спрямованого ациклічного графа (DAG). Тут результати основної сітки постійно змінюють склад втішної, що вимагає рекурсивного контролю цілісності. Для малих категорій зазвичай застосовують

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

кругову схему у вигляді повного графа. Для масових подій з жорстким таймінгом підходить швейцарська система. Тур у ній формується шляхом пошуку паросполучення мінімальної ваги з урахуванням поточного рейтингу учасників [11]. Незалежно від обраного формату, після генерації сітки алгоритм виконує розведення одноклубників (club seeding), щоб представники однієї команди не зустрілися між собою на ранніх етапах турніру.

У день змагань критичною стає комунікаційна інфраструктура. Майданчики працюють паралельно, а мережеве з'єднання на аренах часто буває нестабільним. Відповідно до теореми CAP [12], доцільно розгортати систему як повністю автономний локальний вузол (self-hosted) у межах мережі арени [13]. Турнірний сервер працює незалежно від магістрального інтернет-провайдера. Замість побудови складних механізмів відкладеної синхронізації з хмарою та розв'язання конфліктів злиття даних, архітектура покладається на єдину точку істини безпосередньо на місці проведення змагань. Усі операції з базою даних, генерація сіток та фіксація балів відбуваються локально, що гарантує максимальну швидкість відгуку та абсолютну доступність системи для суддівських терміналів протягом усього активного етапу. Для миттєвого оновлення локальних табло замість традиційного HTTP-polling використовується повнодуплексний протокол WebSocket [14]. Це усуває накладні витрати на постійне відкриття TCP-з'єднань [15].

Перелік вимог до розроблюваного програмного забезпечення наведено в таблиці 1.1.

Таблиця 1.1

Вимоги до програмного комплексу управління турнірами

Тип вимоги		Опис вимоги		
Функціональні вимоги				
ФВ-1		Автоматична генерація вагово-вікових категорій із реляційною фільтрацією розріджених даних для запобігання пустим перетинам.		

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Тип вимоги	Опис вимоги
Функціональні вимоги	
ФВ-2	Побудова турнірних сіток (бінарні дерева, DAG, кругова схема) з алгоритмічним розведенням одноклубників на ранніх етапах.
ФВ-3	Транзакційно узгоджена агрегація суддівських балів у режимі реального часу через підключений модуль правил (Strategy pattern).
ФВ-4	Автоматичне просування переможців графом турніру зі збереженням цілісності зв'язків батьківських та дочірніх вузлів.
Нефункціональні вимоги	
НФВ-1	Підтримка до 500 одночасних підключень із часом відгуку сервера (Server Response Time) не більше 200 мс у 95-му процентилі.
НФВ-2	Забезпечення наскрізної затримки (end-to-end latency) оновлення табло менше 200 мс у локальній мережі арени через протокол WebSocket.
НФВ-3	Можливість повністю автономного розгортання системи в локальній мережі арени без необхідності синхронізації з хмарними сервісами (Edge Computing, теорема CAP).
НФВ-4	Відповідність Open/Closed Principle. Додавання нових правил оцінювання не вимагає перекомпіляції ядра системи.

1.2. Аналіз аналогів програмного продукту

Розробка архітектури вимагає аналізу наявних платформ. Ринок спеціалізованого програмного забезпечення поділений між кількома монополістами. Ці системи оцінено за трьома інженерними метриками. Першою є здатність масштабувати бізнес-логіку. Другою виступає наявність відкритого програмного інтерфейсу. Третьою вимогою визначено підтримку автономного розгортання в умовах нестабільної мережі.

Платформа MyUventex [16] підтримує генерацію розгалуженої системи дивізіонів (рис. 1.2) та має фінансовий модуль для прийому стартових внесків.

Головним її недоліком для нашої задачі є виключно хмарна (SaaS) модель. Це створює єдину точку відмови (Single Point of Failure): зникнення магістрального інтернету на арені повністю зупиняє турнір. Спортивні змагання часто проходять у легкоатлетичних манежах із металевими перекриттями або в підвальних приміщеннях. Під час скупчення тисяч глядачів пропускна здатність мобільних мереж критично деградує. Відсутність локального сервера унеможливорює роботу суддівських терміналів, перетворюючи архітектурну залежність від хмари на неприпустимий ризик для організаторів.

Відкриті REST-інтерфейси [17] у системі відсутні, тому підключити спеціалізоване апаратне забезпечення чи мобільний клієнт для суддів неможливо. Робота в ізольованому контурі архітектурно не передбачена [18].

Рис. 1.2 – Екран вибору дивізіонів платформи MyUventex [16]

Архітектура серверної частини Smoothcomp [19] має інше обмеження. Правила оцінювання поєдинків та алгоритми рейтингування жорстко закодовані в монолітне ядро [20]. Зміна ваги балів у нетиповій естафеті вимагає прямого втручання розробників у вихідний код. Регламенти федерацій єдиноборств регулярно оновлюються. Місцеві турніри часто застосовують адаптовані правила для дітей, змінюючи поріг дострокової перемоги або тривалість поєдинку. Монолітна архітектура Smoothcomp змушує організаторів підлаштовувати свої регламенти під

наявні в системі алгоритми, оскільки самостійно додати нову логіку нарахування балів неможливо. Відсутність абстрактного рушія правил (Ruleset Engine) порушує принцип відкритості/закритості [6]. При цьому клієнтську частину платформи реалізовано якісно: клієнтська частина використовує повнодуплексні канали (WebSocket) [14] для миттєвого оновлення сіток у браузері (рис. 1.3).

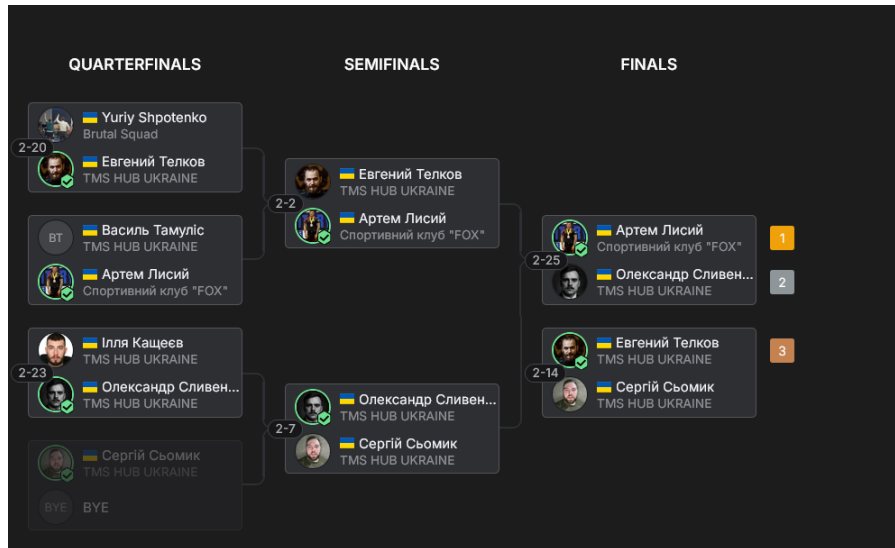


Рис. 1.3 – Турнірна сітка Smoothcomp [19]

Офіційний стандарт Всесвітньої федерації карате, система Sportdata [21], орієнтована на автономну роботу в локальній мережі (LAN). Вона не залежить від стабільності зовнішнього інтернет-з'єднання. Натомість її стек технологій спирається на парадигму «товстого клієнта» (thick client). Організатори розгортають локальний сервер і встановлюють десктопну програму на кожен суддівський пристрій (рис. 1.4). Спілкування між вузлами відбувається через HTTP-polling. Використання десктопної програми вимагає від організаторів закупівлі або оренди повноцінних ноутбуків для кожного татамі. Відкритий вебстек дозволив би використовувати значно дешевші планшети на базі Android або iOS. Крім того, синхронне опитування сервера десятками клієнтів одночасно перевантажує локальну мережу арени, що часто будується на тимчасових Wi-Fi маршрутизаторах. Це генерує значно більший мережевий трафік порівняно з постійними з'єднаннями [22]. Протоколи обміну даними закриті, що перешкоджає розробці легких багатоплатформних клієнтів.

CATEGORY	TEAM	SEX	ENTRIES	DRAW MODE	POOLS	#MATCHES	Status	AGE FROM	AGE TOO OLD	ENTRY FEE EUR	DRAW MODE OPT...	LIMIT (CLUB)	LIMIT (CATEGORY)	Entries Stat...	Short name
Female Kata	f	f	17	Elimination (KO-Mode)	auto	22	Finished	16	100	0.0 -		0	0	A	
Female Kumite -50 Kg	f	f	8	Elimination (KO-Mode)	auto	9	Finished	18	100	0.0 -		0	0		
Female Kumite -55 Kg	f	f	12	Elimination (KO-Mode)	auto	15	Finished	18	100	0.0 -		0	0		
Female Kumite -61 Kg	f	f	14	Elimination (KO-Mode)	auto	17	Finished	18	100	0.0 -		0	0		
Female Kumite -68 Kg	f	f	12	Elimination (KO-Mode)	auto	15	Finished	18	100	0.0 -		0	0		
Female Kumite 68+ ...	f	f	11	Elimination (KO-Mode)	auto	14	Finished	18	100	0.0 -		0	0		
Female Team Kata	t	f	8	Elimination (KO-Mode)	auto	9	Finished	16	100	0.0 -		0	0		
Female Team Kumite	t	f	14	Elimination (KO-Mode)	auto	17	Finished	18	100	0.0 -		0	0		
Male Kata	m	m	21	Elimination (KO-Mode)	auto	26	Finished	18	100	0.0 -		0	0		
Male Kumite -60 Kg	m	m	20	Elimination (KO-Mode)	auto	25	Finished	18	100	0.0 -		0	0		
Male Kumite -67 Kg	m	m	21	Elimination (KO-Mode)	auto	26	Finished	18	100	0.0 -		0	0		
Male Kumite -75 Kg	m	m	21	Elimination (KO-Mode)	auto	26	Finished	18	100	0.0 -		0	0		
Male Kumite -84 Kg	m	m	20	Elimination (KO-Mode)	auto	25	Finished	18	100	0.0 -		0	0		
Male Kumite 84+ Kg	m	m	12	Elimination (KO-Mode)	auto	15	Finished	18	100	0.0 -		0	0		
Male Team Kata	t	m	13	Elimination (KO-Mode)	auto	16	Finished	16	100	0.0 -		0	0		
Male Team Kumite	t	m	14	Elimination (KO-Mode)	auto	17	Finished	18	100	0.0 -		0	0		

Рис. 1.4 – Модуль керування категоріями Sportdata SET[21]

Жоден із розглянутих аналогів не поєднує сучасний багатоплатформний вебстек із можливістю ізольованої роботи. Власна розробка базується на підході Deployment Agnosticism (незалежність від середовища розгортання) [13], що усуває жорстку прив'язку архітектури до зовнішньої інфраструктури. Ядро системи разом з усіма необхідними системними залежностями пакується в OCI-сумісні контейнери (Docker). Цей технічний крок забезпечує повну відтворюваність середовища виконання та дозволяє запускати ідентичний образ як у глобальній хмарі для онлайн-реєстрації учасників, так і на локальному сервері арені (Edge Computing) для автономного проведення боїв без звернень до зовнішньої мережі [12]. Порівняння архітектурних, алгоритмічних та комерційних можливостей систем наведено в таблиці 1.2.

Таблиця 1.2

Порівняльний аналіз систем управління турнірами

Технічний / Функціональний показник	MyUventex	Smoothcomp	Sportdata	Власна розробка
Архітектура та комунікації				
Робота через багатоплатформний браузер (без інсталяції)	+	+	-	+
Відкритий REST API для підключення сторонніх клієнтів	-	-	-	+

Продовження таблиці 1.2

Технічний / Функціональний показник	MyUventex	Smoothcomp	Sportdata	Власна розробка
Підтримка повнодуплексного зв'язку (WebSocket)	+	+	-	+
Моделі розгортання				
Глобальне хмарне розгортання (SaaS)	+	+	-	+
Автономне локальне розгортання (Self-hosted)	-	-	+	+
Алгоритмічна гнучкість				
Автоматичне розведення одноклубників (Club Seeding)	+	+	+	+
Підключення власних правил оцінювання (Ruleset Engine)	-	-	-	+
Динамічна зміна графа сітки під час змагань	-	-	-	+
Комерційна зрілість екосистеми				
Інтегрований фінансовий модуль (прийом платежів)	+	+	+	+
Глобальна база профілів та рейтингів спортсменів	+	+	+	-
Офіційна сертифікація міжнародними федераціями	+	+	+	-

Нова платформа логічно поступається комерційним гігантам за обсягом історичних даних (рейтинги) та відсутністю сертифікацій. Базові ж потреби монетизації покриває інтеграція з API еквайрингу Monobank [23] для автоматизованого прийому внесків. Архітектурно система пропонує вищу гнучкість: комбінація контейнеризації, REST API [24] та патерну Strategy [5] дозволяє розгорнути турнір із власними правилами на будь-якому обладнанні, незалежно від наявності доступу до інтернету.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

1.3. Вибір архітектури веборієнтованої системи управління турнірами

Проектування системи вимагає вирішення конфлікту між високим навантаженням на запис та масовим читанням даних. Навантаження на запис генерують суддівські бригади, які фіксують результативні дії, порушення та зміни часу кожні кілька секунд на кожному активному татамі. Водночас масове читання формується сотнями глядачів, тренерів та спортсменів, які безперервно перевіряють оновлення турнірних сіток та публічних табло зі своїх мобільних пристроїв. У сучасній інженерній практиці для таких задач розглядають класичний шаруватий моноліт, розподілену мікросервісну архітектуру та модульний моноліт.

Класичний шаруватий моноліт об'єднує всю логіку в одній кодовій базі з поділом на шари представлення, бізнес-правил і доступу до даних. Відсутність жорстких меж між функціональними областями зазвичай призводить до високої зв'язності коду. З часом така система деградує до архітектурного антипатерну "велика грудка бруду" [25]. Якщо організаторам потрібно додати нову спортивну дисципліну зі специфічними правилами підрахунку очок, це вимагатиме каскадних змін по всій кодовій базі. Така залежність порушує принцип відкритості та закритості [6], що робить безпечне розширення платформи неможливим.

Розподілена мікросервісна архітектура [26] ізолює домени фізично. У контексті автономного турнірного сервера такий підхід є технічно надлишковим і створює нездоланні перешкоди під час розгортання. Спортивні змагання найчастіше обслуговуються одним локальним комп'ютером середньої продуктивності. Запуск повноцінного кластера мікросервісів із балансувальниками навантаження, зовнішніми брокерами повідомлень та системами розподіленого логування на такій машині призведе до вичерпання оперативної пам'яті ще до початку боїв. Крім того, інсталяція розподіленої системи в умовах відсутності магістрального інтернету вимагає високої кваліфікації технічного персоналу. Це унеможлиблює швидку підготовку арени силами звичайного організаційного комітету.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Серверну частину платформи реалізовано за архітектурою модульного моноліту [20]. Система розгортається як єдиний процес операційної системи. Код жорстко поділено на ізольовані предметні застосунки екосистеми Django. Вони розділяють логіку ідентифікації користувачів, управління життєвим циклом турнірів, обробки правил суддівства та алгоритмічної генерації сіток. Усі модулі працюють у межах єдиного процесу та використовують спільну реляційну схему бази даних. Замість штучної ізоляції через суворі програмні інтерфейси система допускає прямі зв'язки між моделями різних доменів через механізм зовнішніх ключів. Такий підхід залишає кодову базу добре структурованою за бізнес-функціями і водночас уникає накладних витрат на серіалізацію даних під час кросмодульних запитів. Спільним компонентом виступає шар об'єктно-реляційного відображення. Він інкапсулює запити до єдиної бази даних PostgreSQL і гарантує виконання каскадних оновлень турнірної сітки в межах однієї локальної транзакції [2].

Зворотний проксі-сервер Nginx розділяє трафік. Він направляє запити прикладного програмного інтерфейсу [17], довготривалі з'єднання WebSocket та запити за статикою до відповідних внутрішніх служб. Перед Nginx розгорнуто сервер Caddy для автоматичного отримання сертифікатів та завершення (термінації) захищених з'єднань. Наявність сертифікатів дозволяє використовувати захищені протоколи HTTPS та WSS навіть у локальному середовищі, що є обов'язковою вимогою сучасних веббраузерів для коректної роботи Service Workers та безпечного зберігання сесійних файлів cookie. Адміністративні операції використовують сесійну автентифікацію зі збереженням контексту підключення в базі даних. Захист від надмірної кількості запитів реалізовано на рівні самого застосунку через відповідні класи обмеження фреймворку Django REST Framework, що дозволяє гнучко налаштовувати квоти для захисту серверної частини від перевантажень. Синхронне опитування клієнтів не застосовується через ризик вичерпання пулу з'єднань бази даних надлишковим трафіком [22]. Замість нього працюють повнодуплексні з'єднання за протоколом WebSocket [14].

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Щоб розвантажити реляційну базу PostgreSQL від масового читання, архітектуру доповнено брокером Redis, який працює в оперативній пам'яті. Без цього брокера кожне оновлення таблиці вимагало б виконання важких реляційних запитів для агрегації поточного стану поєдинку, що швидко вичерпало б ресурси сервера під час одночасної роботи кількох татамі. Коли суддівський модуль фіксує зміну рахунку, він записує результат у базу даних і паралельно публікує подію в Redis за патерном видавця та підписника. Маршрутизація повідомлень відбувається через механізм ізольованих каналів. Кожному татамі призначається унікальний ідентифікатор топика. Клієнтський застосунок публічного табло підписується виключно на канал свого майданчика і отримує лише релевантні оновлення. Цей механізм повністю ізолює транзакційне ядро від читацького навантаження і розглядається як типовий патерн інтеграції в сучасній архітектурі Python [29, 30].

На основі наведеного аналізу побудовано загальну структурну схему вебсервісу. Кількісні показники обраних рішень наведено у розділі, присвяченому тестуванню системи.

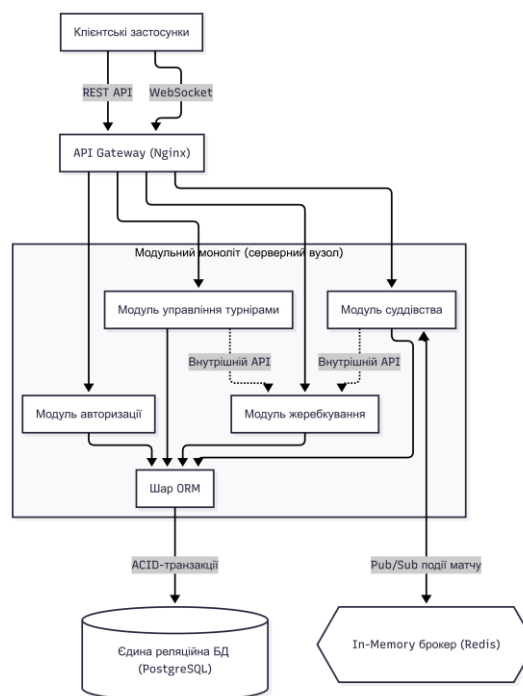


Рис. 1.5 – Структурна схема архітектури вебсервісу управління турнірними змаганнями

1.4. Обґрунтування вибору технологічного стеку

Технологічний стек платформи базується на трьох архітектурних обмеженнях: необхідності обробки складної реляційної логіки (генерація рекурсивних турнірних графів), підтримці двонапрямленої синхронізації даних у реальному часі та жорстких строках на реалізацію інфраструктурного коду.

Основною мовою серверної частини обрано Python 3.12. Вибір зумовлений стабільністю цієї версії для промислової експлуатації та повною сумісністю з усіма необхідними асинхронними бібліотеками [31]. Середовище виконання упаковується в офіційний мінімалістичний образ python:3.12-slim для розгортання через Docker. Система свідомо не використовує експериментальні функції на кшталт відключення глобального блокування інтерпретатора (GIL) чи JIT-компіляції з новіших версій, оскільки поточна архітектура ефективно вирішує проблему конкурентності за рахунок асинхронного вводу-виводу (ASGI) та рознесення навантаження між незалежними робочими процесами. Завдяки цьому досягається стабільна та передбачувана поведінка пам'яті під час розрахунку багатовимірних турнірних сіток на кількох ядрах процесора. Альтернативні екосистеми мають інші обмеження для цієї задачі. Архітектура Node.js спирається на однопотоковий цикл подій (Event Loop). Згідно з дослідженнями продуктивності [32], виконання CPU-інтенсивних алгоритмів на графах у такому середовищі блокує головний потік, що призводить до деградації часу відгуку для всіх активних мережових з'єднань. Мова Go забезпечує високу швидкодію завдяки нативній компіляції, проте відсутність комплексних фреймворків перекладає написання базового інфраструктурного каркаса безпосередньо на розробника.

Для побудови вебсервісу застосовано фреймворк Django 5.2 [33]. Мікрофреймворки (мінімалістичні фреймворки без вбудованих компонентів), такі як FastAPI, вимагають ручної інтеграції компонентів (SQLAlchemy, Alembic, адміністративної панелі), що збільшує обсяг шаблонного коду. Django надає вбудовану систему міграцій та генератор адміністративного інтерфейсу. Для

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

забезпечення сумісності та надійної роботи реляційної моделі всі таблиці бази даних, включно з таблицями зв'язків для турнірних матчів та реєстрацій спортсменів, використовують стандартні сурогатні ідентифікатори на основі типу BigAutoField. Такий підхід гарантує безпроблемну інтеграцію з усіма сторонніми пакетами екосистеми Django, спрощує побудову складних запитів через ORM та дозволяє уникнути проблем, які часто виникають при використанні складених первинних ключів.

Зіставлення екосистем за моделлю конкурентності, інструментарієм та поведінкою під навантаженням наведено в таблиці 1.3.

Таблиця 1.3

Порівняння серверних технологій

Критерій	Python 3.13 + Django 5.2	Node.js 22 + Express	Go 1.22 + Gin
Модель конкурентності	Асинхронність через ASGI та робочі процеси	Однопоточковий цикл подій (важкі задачі у Worker Threads)	легковагові потоки виконання (goroutines)
Швидкість розробки	Висока (вбудовані ORM, міграції, адміністративна панель)	Середня (збирається зі сторонніх пакетів)	Нижча (потребує багато ручного коду)
Обчислювально складні задачі	Стабільна робота завдяки асинхронним оптимізаціям	Ризик блокування циклу подій (Event Loop)	Висока завдяки компіляції в нативний код
Архітектурна структура	Задана фреймворком (шаблон MVT)	Залежить від архітектурних рішень розробника	Без єдиного стандарту структури

Дані турніру утворюють щільну мережу багатовимірних зв'язків: один спортсмен одночасно фігурує в заявці клубу, категорії, поточній парі та рейтингу. У документо-орієнтованих базах даних (наприклад, MongoDB) моделювання таких структур вимагає вибору між денормалізацією та нормалізацією. Денормалізація (вкладення документів) неминуче призводить до дублювання: зміна статусу

		Муравецький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

спортсмена вимагатиме пошуку і перезапису всіх вкладених копій, що створює ризик аномалій оновлення (Update Anomalies) [1]. З іншого боку, використання нормалізованих посилань (references) змушує обходити граф через агрегаційний конвеєр \$graphLookup. Оскільки обробка кожної стадії конвеєра відбувається в оперативній пам'яті з жорстким лімітом у 100 МБ, глибокий обхід турнірного дерева призведе до помилки перевищення ліміту. Згідно з документацією системи, для стадії \$graphLookup параметр allowDiskUse ігнорується, тому база не зможе скинути тимчасові дані на диск і перерве виконання запиту [34]. Отже, використання документо-орієнтованих NoSQL рішень є технічно недоцільним для надійної обробки рекурсивних турнірних графів.

Зазначені недоліки зумовили вибір реляційної бази даних PostgreSQL 18. Саме вона бере на себе відповідальність за несуперечливість даних. Турнірна сітка є графом, глибина якого динамічно залежить від кількості учасників. Рушій PostgreSQL підтримує рекурсивні узагальнені табличні вирази (WITH RECURSIVE) [35], дозволяючи обходити такі структури на рівні ядра бази даних із використанням оптимізованих індексів, уникаючи багатоетапних конвеєрів агрегації. Для зберігання змінних конфігурацій локальних турнірів система використовує нативний тип JSONB з підтримкою GIN-індексів. Таке поєднання реляційної схеми для основних сутностей та JSONB для гнучких параметрів повністю закриває потребу в поліморфному зберіганні. Це дозволяє тримати всю логіку в межах єдиної транзакційної системи і не розгортати додаткову NoSQL базу [36].

Таблиця 1.4

Порівняння систем керування базами даних

Функціональна характеристика	PostgreSQL 18	MySQL 9.0	MongoDB 7.0
Парадигма зберігання	Об'єктно-реляційна	Реляційна	Документо-орієнтована (NoSQL)

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Продовження таблиці 1.4

Функціональна характеристика	PostgreSQL 18	MySQL 9.0	MongoDB 7.0
Відповідність вимогам ACID	Повна, сувора консистентність	Повна (у рушії InnoDB)	Часткова (через механізм сесій та WiredTiger)
Обробка ієрархічних даних	Підтримка рекурсивних CTE (WITH RECURSIVE)	Рекурсивні CTE (з версії 8.0)	Відсутня (обхід виконується через агрегаційний конвеєр \$graphLookup)
Робота з JSON	Нативний тип JSONB із GIN-індексами	Тип JSON без глибокої оптимізації	Основний формат зберігання (BSON)

Стандартний інтерфейс WSGI працює за моделлю «один потік на один запит». Він блокує системні потоки під час очікування мережевого вводу-виводу або відповідей від бази даних. Через це WSGI не здатен ефективно утримувати тисячі постійно відкритих WebSocket-з'єднань, оскільки це швидко вичерпало б доступний пул потоків операційної системи. Цю задачу покладено на специфікацію ASGI та пакет Django Channels [29]. Замість блокування ASGI використовує цикл подій, де один асинхронний робочий процес перемикається між множиною з'єднань одночасно і споживає значно менше оперативної пам'яті. Сховище Redis виконує роль брокера повідомлень (Channel Layer). З'єднання групуються за ідентифікаторами татамі, що дозволяє Redis розсилати події конкретного поєдинку виключно зацікавленим клієнтам без навантаження загальної мережі.

Для уніфікації середовищ локальної розробки та промислової експлуатації застосовано контейнеризацію Docker (рис. 1.6). Nginx виконує роль зворотного проксі-сервера та маршрутизує вхідний трафік на основі HTTP-заголовків. Синхронні HTTP-запити направляються на класичний WSGI-сервер Gunicorn. Коли клієнт ініціює перехід на WebSocket через заголовок Upgrade, Nginx перехоплює цей запит і

перенаправляє його на спеціалізований ASGI-сервер Daphne. Цей поділ ізолює синхронний і асинхронний трафік у незалежних пулах процесів. Завдяки такій конфігурації, навіть якщо група обчислювально важких запитів тимчасово заблокує робочі процеси Gunicorn, це не вплине на швидкість доставлення повідомлень у реальному часі через Daphne.

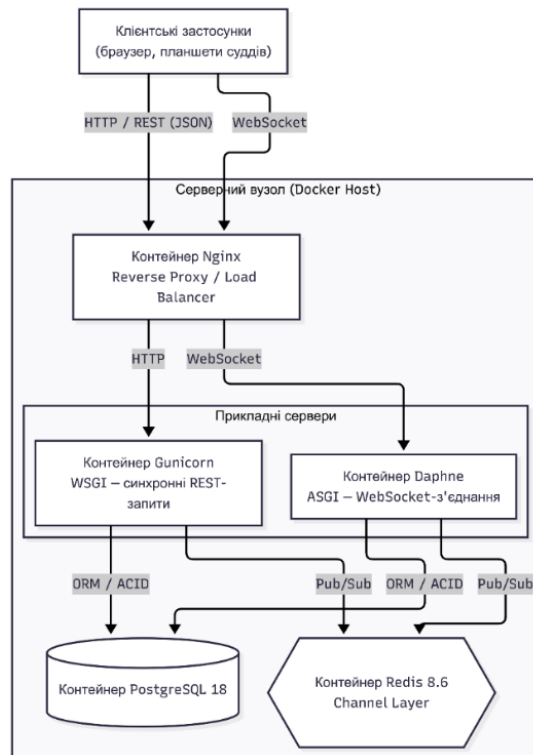


Рис. 1.6 – Схема розгортання програмного комплексу

Реалізація клієнтської частини (панелі судді) вимагає мінімізації сприйманої затримки в умовах нестабільного з'єднання на спортивних аренах [37], оскільки суддівська бригада фіксує результативні дії у високому темпі. Архітектурна залежність Angular від бібліотеки RxJS суттєво підвищує поріг входження, а Vue 3 потребує інтеграції сторонніх менеджерів для обробки складних оптимістичних оновлень. Уникнути цих інфраструктурних ускладнень дозволила бібліотека React 19. Її вбудовані хуки (hooks) `useActionState` та `useOptimistic` [38] реалізують патерн Optimistic UI без додаткових залежностей. Вони відображають нараховані бали на екрані планшета одразу після натискання, до отримання підтвердження від сервера. Інтерфейс при цьому не блокується і дозволяє судді вводити наступні оцінки, а в разі

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

мережевої помилки React автоматично відкочує стан до попереднього коректного значення. Збірку застосунку виконує Vite 8 зі збирачем Rolldown [39], що додатково зменшує розмір завантажених клієнтом файлів завдяки ефективному видаленню невикористаного коду (tree shaking).

Порівняння клієнтських технологій наведено в таблиці 1.5.

Таблиця 1.5

Порівняння клієнтських технологій

Характеристика	React 19 + Vite 8	Vue 3 + Vite 8	Angular 17 + Webpack
Архітектурний підхід	Компонентна бібліотека	Прогресивний фреймворк	Монолітний фреймворк
Оновлення DOM	Віртуальний DOM (Fiber)	Віртуальний DOM (Проху-реактивність)	Інкрементальний DOM
Керування асинхронним станом	Вбудовані хуки (hooks) useActionState, useOptimistic	Composition API	RxJS (висока складність)

Рекомендовані апаратні вимоги до серверної частини наведено у таблиці 1.6.

Таблиця 1.6

Рекомендовані вимоги до апаратного забезпечення серверної частини

Компонент інфраструктури	Мінімальні вимоги (тестове середовище)	Рекомендовані вимоги (промислове середовище)	Обґрунтування навантаження
Процесор (CPU)	2 ядра (x86_64 / ARM64)	4+ ядра	Робота інтерпретатора Python 3.12 та робочих процесів ASGI
Оперативна пам'ять (RAM)	4 ГБ	8 ГБ	Резервація пам'яті під PostgreSQL, Redis та Django

Компонент інфраструктури	Мінімальні вимоги (тестове середовище)	Рекомендовані вимоги (промислове середовище)	Обґрунтування навантаження
Дискова підсистема	20 ГБ SSD	50+ ГБ NVMe SSD	Забезпечення IOPS для операцій реляційної бази даних
Мережевий інтерфейс	100 Мбіт/с	1 Гбіт/с	Утримання конкурентних WebSocket-з'єднань

Обраний стек переводить архітектурні обмеження у конкретні технологічні рішення. База даних PostgreSQL гарантує цілісність рекурсивних даних, зв'язка ASGI з Redis забезпечує повнодуплексний обмін повідомленнями, а екосистема Django знімає потребу в написанні інфраструктурного коду. Експериментальні дані щодо швидкодії цієї конфігурації під навантаженням наведено у третьому розділі. Обраний стек технологій безпосередньо впливає з поставлених завдань та потреб предметної області. База даних PostgreSQL вирішує проблему узгодженості результатів під час паралельного суддівства. WebSocket-з'єднання усувають затримки в оновленні табло. Контейнеризація задовольняє вимогу щодо автономної роботи системи в локальній мережі спортивної арени без доступу до магістрального інтернету.

1.5. Обмеження розробленої системи

Функціональні обмеження визначають межі застосовності платформи. Система підтримує п'ять алгоритмів жеребкування. До них належать олімпійська сітка з вибуванням після першої поразки, кругова система, подвійне вибування, олімпійська сітка з репешажем (втішними поєдинками) та швейцарська система. Формати з груповим етапом і наступним етапом плей-оф не реалізовано. Відсутність змішаних форматів зумовлена комбінаторною складністю автоматичного перенесення результатів груп у вузли дерева вибування. Такий перехід вимагає розв'язання

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

нічийних ситуацій за додатковими показниками та динамічної перебудови графа в межах однієї транзакції бази даних, що виходить за межі поточного алгоритмічного ядра. Побудова графа вимагає наявності щонайменше двох учасників у категорії. Максимальна кількість учасників в одній сітці алгоритмічно обмежена 128 спортсменами. Серверна частина генерує сітки такого розміру за частки мілісекунди, тому це обмеження продиктовано виключно вимогами до продуктивності клієнтського застосунку під час візуалізації дерева поєдинків у браузері. Бінарне дерево на 128 учасників вимагає одночасної побудови понад двох сотень вузлів-поєдинків та відповідної кількості векторних з'єднувальних ліній. Відображення масивніших структур перевантажує об'єктну модель документа (DOM). Це призводить до критичного падіння частоти кадрів і робить навігацію по сітці неможливою на слабких мобільних пристроях.

Формат проведення призначається автоматично або фіксується організатором вручну. Наприклад, кругова система застосовується для категорій до п'яти спортсменів включно, а олімпійська призначається для шести і більше. Предметна область платформи звужена до контактних єдиноборств із бальною або прапорцевою системою оцінювання. Сюди належать карате, дзюдо за правилами Міжнародної федерації дзюдо та тхеквондо за правилами Всесвітнього тхеквондо. Дисципліни з вимірюванням метричних показників або фіксацією перетину фінішу не підтримуються через принципову різницю в логіці підбиття підсумків. Абстрактний рушій правил побудовано на моделі дискретних подій та автоматі станів. Цей автомат реагує виключно на фіксовані команди судді й не має архітектурних засобів для обробки високочастотних поточкових даних від апаратних датчиків сили удару чи фотофінішу. Платформа також не підтримує жорсткого прив'язування поєдинків до конкретного часу розкладу. Замість цього використовується динамічна черга, де наступний поєдинок викликається на вільне татамі. Таке рішення ухвалено через специфіку бойових мистецтв. Травми, медичні тайм-аути та процедури розгляду

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

протестів роблять точну тривалість поєдинку непередбачуваною. Жорсткий розклад неминуче призвів би до каскадних затримок усього турніру.

Технічні обмеження обумовлені обраним архітектурним підходом та інфраструктурою. Клієнтський застосунок використовує сучасні вебінтерфейси без підключення компонентів зворотної сумісності (поліфілів). Через це коректне відображення інтерфейсу на застарілих браузерях не гарантується. Панель оператора татамі вимагає використання пристроїв із діагоналлю екрана від восьми дюймів. Розміщення всіх елементів суддівського управління на екранах смартфонів порушує ергономіку та збільшує ризик хибних натискань. Система примусово блокує відкриття робочого місця судді на пристроях із малим розширенням, оскільки недостатня площа сенсорних кнопок гарантовано призведе до помилок нарахування балів під час динамічного бою.

Система не має режиму офлайн-роботи для клієнтських пристроїв. У разі втрати зв'язку з локальним сервером планшет судді не кешує результативні дії локально, оскільки сервером єдиної істини виступає виключно база даних. Локальне накопичення балів на клієнті порушило б синхронність публічного табло. Без постійного підтвердження від бази даних локальний таймер на планшеті втратить синхронізацію з іншими спостерігачами. На стороні сервера жорстка прив'язка до версій інфраструктури PostgreSQL 18 та Python 3.12 зумовлена специфікою кодової бази і використанням рекурсивних виразів. Оскільки кожна дія судді дописується в таблицю подій за моделлю Event Sourcing, фіксація бала використовує транзакційні блокування на рівні рядків. Якщо на арені працює велика кількість майданчиків, інтенсивність дискових операцій стрімко зростає. Серіалізація відповідей та запис у журнал випереджального запису (WAL) створюють постійне навантаження на накопичувач. Використання магнітних жорстких дисків замість твердотільних накопичувачів призведе до утворення черги транзакцій і затримає відгук програмного інтерфейсу понад допустимі норми.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

Для узгодження таймерів поєдинків система спирається виключно на системний час сервера. Будь-який дрейф часу між сервером та планшетами суддів призведе до похибок у тривалості раундів. При горизонтальному масштабуванні серверного шару на кілька вузлів це вимагає обов'язкового налаштування протоколу мережевого часу (NTP) для запобігання розсинхронізації клієнтських табло. Якщо адміністратор вручну змінить системний час на сервері під час активного поєдинку, серверний розрахунок таймера видасть непередбачуваний результат.

Відновлення втрачених з'єднань WebSocket відбувається за алгоритмом експоненційної затримки. Робота у бездротових мережах аргентини супроводжується регулярними втратами пакетів. Алгоритм перепідключення створює ризик масового одночасного запиту. Якщо головний маршрутизатор перезавантажиться, всі суддівські термінали спробують перепідключитися в один момент. Це пікове навантаження на підсистему автентифікації здатне викликати тимчасову відмову в обслуговуванні. До моменту відновлення постійного з'єднання дії судді передаються на сервер через резервні запити за протоколом HTTP. Цей механізм дозволяє продовжувати поєдинок навіть за умов нестабільного двостороннього каналу, проте швидкість оновлення табло в такому режимі деградує.

Організаційні та правові обмеження залишають низку процесів поза межами програмного управління. Допуск спортсмена до змагань можливий лише після ручного внесення результатів фізичного зважування відповідальною особою секретаріату. Процедури перевірки ліцензій, медичних довідок та страхових полісів не цифровізовані і покладаються виключно на організаційний комітет турніру. Програма функціонує лише як система реєстрації підсумкового факту допуску і не зберігає юридичні копії документів учасників. Права тренера або судді в системі надаються адміністратором тільки після позасистемної верифікації відповідних документів. Платформа використовує підхід нульової довіри до нових облікових записів. Користувач зберігає статус звичайного глядача до моменту ручного

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

підтвердження його ролі організатором, що захищає персональні дані неповнолітніх спортсменів від несанкціонованого доступу.

Інтеграція платіжного модуля реалізована виключно з програмним інтерфейсом українського банку Monobank. Система виконує криптографічну перевірку цифрового підпису кожного платіжного сповіщення за алгоритмом ECDSA. Цей механізм жорстко прив'язаний до специфікації конкретного банку. Це обмежує можливість програмного прийому стартових внесків у міжнародному форматі та вимагає від закордонних учасників використання альтернативних ручних методів оплати з подальшим підтвердженням у системі. Відсутність інтеграції зі шлюзами на кшталт PayPal або Stripe змушує секретаріат турніру приймати іноземні внески готівкою і самостійно зводити фінансовий баланс в адміністративній панелі платформи.

Висновки до першого розділу

Аналіз предметної області та бізнес-процесів організації змагань виявив ключові джерела обчислювальної складності. До них належать реляційна фільтрація даних під час формування категорій, операції з турнірними графами та ризик станів гонитви при паралельній фіксації суддівських оцінок. Зіставлення комерційних платформ MyUventex, Smoothcomp та Sportdata підтвердило відсутність на ринку рішення, яке одночасно забезпечує автономну роботу в локальній мережі, пропонує легкі багатоплатформні вебінтерфейси та дозволяє змінювати логіку суддівства. Задля вирішення цих проблем спроектовано архітектуру системи за патерном модульного моноліту. Такий підхід безпосередньо зумовлений вимогою автономного розгортання платформи на локальному обладнанні спортивних арен, де підтримка повноцінної мікросервісної інфраструктури є технічно неможливою через апаратні обмеження. Обґрунтовано вибір бази даних PostgreSQL як основи транзакційного ядра. На відміну від документо-орієнтованих систем на кшталт MongoDB, які обмежують обхід графів лімітами оперативної пам'яті під час агрегацій, реляційний рушій дозволяє обробляти багаторівневі турнірні сітки за допомогою оптимізованих рекурсивних виразів. Для

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

трансляції перебігу змагань спроектовано комунікаційний шар на базі асинхронного інтерфейсу ASGI, брокера Redis та протоколу WebSocket. Обрана комбінація миттєво оновлює публічні табло і повністю ізолює базу даних від масових запитів на читання. Окреслено функціональні, технічні та організаційні обмеження системи, які встановлюють межі програмного управління турнірами перед етапом проектування структур даних.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

РОЗДІЛ 2. ПРОЄКТУВАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОЇ СТРУКТУРИ ТА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Модель варіантів використання (Use Case) пов'язує неформальні вимоги, сформульовані за результатами аналізу бізнес-процесів, із об'єктно-орієнтованою декомпозицією системи [42]. У підрозділі уточнено вимоги до системи, побудовано рольову модель доступу, описано прецеденти в тому вигляді, в якому їх реалізовано, та подано розподіл класів предметної області за модулями.

Функціональні вимоги, узагальнені у підрозділі 1.1, на етапі проєктування структуровано за функціональними областями, що стають підставою для модульної декомпозиції. До цих областей належать управління обліковими записами та організаційною структурою (реєстрація з підтвердженням пошти, автентифікація, відновлення доступу, керування ролями через запити на верифікацію); ведення довідників спортсменів і команд із масовим імпортом та фіксацією динаміки ваги; конфігурування турнірів, категорій і вагово-вікових дивізіонів; реєстрація учасників, зважування, контроль присутності та фінансові розрахунки; автоматична генерація турнірних сіток усіх форматів; суддівство поєдинків за підключеним набором правил із підбиттям підсумків. Окрему наскрізну область становить трансляція стану змагань у режимі реального часу.

Нефункціональні вимоги задають якісні характеристики системи. Першою є транзакційна цілісність: операція, що змінює стан турнірного дерева, виконується атомарно, оскільки часткове застосування змін порушило б узгодженість сітки. Другою вимогою є низька латентність комунікаційного шару, від якої залежить швидкість відображення нарахованих балів.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Решта вимог (відмовостійкість, масштабованість, рольове розмежування доступу та розширюваність) обґрунтована у першому розділі. Латентність і час відгуку конкретизовано проєктними цільовими показниками за класами операцій, диференційованими за каналом взаємодії (синхронний REST або асинхронний WebSocket), у таблиці 2.1.

Таблиця 2.1

Проєктні вимоги до часу відгуку системи за класами операцій

Клас операції	Канал взаємодії	Цільовий час відгуку (проєктна вимога)	Обґрунтування вимоги
Нарахування або скасування балу суддею	WebSocket	до 200 мс	Збереження ефекту миттєвого відгуку; на боці клієнта посилюється оптимістичним оновленням інтерфейсу
Оновлення публічного табло після фіксації події	WebSocket	до 300 мс	Синхронність зображення на табло з діями суддівської бригади біля татамі
Фіксація результату та просування переможця сіткою	REST (атомарна транзакція)	до 500 мс	Узгоджене оновлення кількох сутностей у межах однієї ACID-транзакції
Завантаження турнірної сітки категорії	REST	до 1 с	Вибірка деревоподібної структури з рекурсивним зв'язком за один SQL-запит без багаторазових звернень із коду
Автоматична генерація сітки (до 32 учасників)	REST	до 2 с	Обчислювально складна операція з розведенням одноклубників, що виконується одноразово
Автентифікація та завантаження профілю	REST	до 1 с	Початкова взаємодія користувача із системою

Наведені значення є проєктними цільовими показниками, а не результатами вимірювань. Їх фактичну перевірку за результатами навантажувального тестування подано в розділі 3. Розподіл за каналами підпорядкований вимозі: операції, чутливі до

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

затримки (нарахування балів, оновлення табло), винесено в асинхронний канал, а операції зі суворою вимогою цілісності виконуються в транзакційному контурі реляційної бази.

Рольову модель розмежування доступу побудовано з огляду на те, що дійові особи системи різняться за функціями та способом набуття повноважень; її засновано на моделі рольового розмежування доступу (Role-Based Access Control, RBAC) [43]. Вона поєднує два механізми: глобальну роль, що зберігається в обліковому записі й діє в межах усієї системи, та контекстні (турнір-залежні) ролі, які надаються лише в межах конкретного змагання. Поведінку ролей уточнюють спеціальні атрибути облікового запису. Такий поділ розмежовує базові категорії користувачів і дозволяє делегувати повноваження в межах окремого турніру.

Визначено шість глобальних ролей.

Глядач є базовою роллю кожного нового запису. Він має доступ до публічної частини (перелік турнірів, сітки, рейтинги, табло реального часу, значною мірою без автентифікації), керує власним профілем і може подати запит на підвищення ролі.

Тренер веде власний контингент: створює та редагує картки спортсменів клубу, формує команди, виконує масовий імпорт, веде журнал ваги та реєструє підопічних у категоріях. Видимість обмежена його контингентом, а щодо вже поданих заявок він може лише знімати їх зі змагань, що захищає узгодженість стартових списків.

Суддя обслуговує поєдинки на закріпленому за ним татамі: нараховує та скасовує бали, керує таймером, фіксує першість результативної дії, визначає переможця, подає рішення колегії у формальних комплексах і зберігає підсумки. Ігрові операції на чужому майданчику відхиляються.

Організатор виступає власником турнірів. Він конфігурує категорії (зокрема через інтелектуальний розбір назв дивізіонів), керує життєвим циклом змагання, запускає генерацію сіток і розподіл по татамі, формує бригади, переглядає фінансовий звіт та ініціює виведення коштів, переважно для власних турнірів.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Персонал обслуговує секретаріат (зважування, присутність, групове оновлення заявок) і нарівні з організатором та адміністратором розглядає запити на ролі.

Адміністратор має найширші повноваження: керує клубами й користувачами, встановлює кредитні ліміти, контролює борги та підтверджує виведення коштів, отримуючи безумовний дозвіл у більшості перевірок доступу.

Базові ролі уточнюють атрибути та контекстні відношення. Атрибут президента клубу розширює фінансові права тренера на весь клуб; атрибут головного адміністратора (суперкористувача) відповідає привілейованому запису, що створюється при розгортанні й поєднує адміністративну роль із повним доступом до службової панелі. Контекстні ролі визначаються відношенням до турніру: головний суддя турніру отримує суддівсько-технічні повноваження, близькі до організаторських, і обслуговує будь-який його майданчик. Головний секретар турніру отримує вищі секретарські права для координації реєстрації та зважування. Приналежність до персоналу турніру надає базові секретарські права, а до суддівської колегії надає право судді працювати на майданчиках змагання. Так одна особа може мати скромну глобальну роль, але розширені повноваження в межах окремого турніру.

Перехід від глядача до привілейованої ролі відбувається лише через верифікацію: користувач подає запит із підтвердними документами, який адміністратор схвалює чи відхиляє з коментарем. У разі схвалення роль змінюється автоматично (для тренера разом із прив'язкою до клубу), а до того запис зберігає роль глядача, що унеможливорює самочинне привласнення повноважень. Рольову модель узагальнено в таблиці 2.2.

Таблиця 2.2

Рольова модель системи та розмежування повноважень

Роль або атрибут	Тип	Спосіб набуття	Ключові повноваження
Глядач	Глобальна роль	За замовчуванням при реєстрації	Перегляд турнірів, сіток, рейтингів і табло; керування власним профілем

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

Продовження таблиці 2.2.

Роль або атрибут	Тип	Спосіб набуття	Ключові повноваження
Тренер	Глобальна роль	Верифікація запиту	Ведення власних спортсменів і команд, реєстрація на категорії, оплата та зняття власних заявок
Суддя	Глобальна роль	Верифікація запиту	Суддівство поєдинків на закріпленому татамі, керування таймером, фіксація результатів
Організатор	Глобальна роль	Верифікація запиту	Створення та конфігурування турнірів, керування їх станом, жеребкування, фінанси
Персонал	Глобальна роль	Призначення	Зважування, контроль присутності, групові операції секретаріату, розгляд запитів на ролі
Адміністратор	Глобальна роль	Призначення	Безумовний доступ до прикладних операцій, керування клубами, користувачами та фінансами платформи
Президент клубу	Атрибут тренера	Призначення	Оплата реєстрацій за всіх членів клубу, перегляд заявок у межах клубу
Головний адміністратор	Атрибут адміністратора	Створення при розгортанні	Повний доступ до прикладної логіки та службової панелі керування
Головний суддя турніру	Контекстна роль	Призначення організатором	Технічне адміністрування конкретного турніру, суддівство на будь-якому його татамі
Головний секретар турніру	Контекстна роль	Призначена організатором	Повний контроль над процесами зважування, реєстрації та управління заявками турніру
Персонал турніру	Контекстна роль	Призначення організатором	Секретарські операції в межах конкретного турніру

Варіанти використання системи розглянуто в розрізі функціональних областей, кожна з яких об'єднує логічно пов'язані сценарії з характерним складом акторів; нижче їх описано так, як реалізовано у програмному інтерфейсі. Узагальнену модель подано в Додатку Б.

Область управління обліковими записами доступна найширшому колу акторів. Незареєстрований відвідувач реєструється та підтверджує пошту одноразовим кодом (з можливістю повторного надсилання). Автентифікація реалізована парою «пошта-пароль» та через зовнішнього постачальника ідентифікації з використанням стандартних серверних сесій фреймворку (SessionAuthentication) [33]. Втрачений доступ відновлюється двокроковим скиданням пароля за кодом. Автентифікований користувач редагує профіль, змінює пароль і подає запит на привілейовану роль; розгляд таких запитів є окремим прецедентом для адміністратора, персоналу та організатора. Самостійним сценарієм є верифікація учасницького бейджа за підписаним токеном, що підтверджує автентичність реєстрації за QR-кодом без входу.

Область ведення довідників обслуговується тренером: він створює, редагує та вилучає картки спортсменів (з автоматичним обмеженням власним контингентом), виконує масовий імпорт із CSV або Excel із розпізнаванням колонок і валідацією рядків, веде журнал ваги (що оновлює поточну вагу) та формує команди клубу для командних дисциплін.

Область управління турнірами та категоріями доступна організаторові й, у технічній частині, головному судді. Організатор створює турнір і керує його життєвим циклом (відкриття реєстрації, активна фаза, завершення), кожен перехід перевіряється на передумови. Категорії конфігуруються поодиночці або груповим імпортом з інтелектуальним розбором назв (стать, вік, вага, набір правил, формат). Окремі прецеденти формують перегляд статистики й рейтингів, а також фінансовий звіт та оплату комісії, доступні лише власникам турніру.

Область реєстрації та фінансів об'єднує тренера, головного секретаря турніру, персонал і організатора. Реєстрації спортсменів і команд створюються та

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

вилучаються, але видалення оплаченої заявки заборонене. Натомість її знімають зі змагань, що за відповідної політики ініціює супутнє повернення коштів (відношення розширення). Головний секретар або персонал підтверджує зважування з контролем вагових меж, відмічає присутність і виконує групове оновлення заявок. Фінансові прецеденти реалізують повний цикл онлайн-оплати: користувач формує рахунок, який транслюється до платіжного шлюзу, а підтвердження надходить через зворотний виклик (webhook) або шляхом ручної синхронізації; керівник клубу оплачує внески за весь клуб. Організатор формує запит на виведення коштів завершеного турніру, підтвердження якого є прерогативою адміністратора.

Область жеребкування доступна організаторові та головному судді: генерація сітки категорії в будь-якому форматі, видалення сітки, генерація чергового туру швейцарської системи, пакетна генерація для всіх категорій, автоматичний розподіл категорій по татамі за балансуванням навантаження та закріплення поєдинків категорії за татамі.

Область суддівства охоплює прецеденти реального часу (суддя, організатор, головний суддя турніру): нарахування та скасування балів, узагальнена подія набору правил, встановлення Senshu, призначення переможця, нічия, скидання поєдинку; керування таймером (запуск, призупинення, відновлення, скидання, тривалість, додавання часу) і перемикання його показу на табло. Для формальних комплексів реалізовано рішення колегії та вибір її складу; для командних дисциплін передбачено призначення спортсменів на бої та створення вирішального поєдинку. Підбиття підсумків оформлено прецедентами розрахунку, фіксації результатів із ручним перевизначенням місць та скасування фіксації.

Область керування татамі й табло пов'язує суддівський і глядацький контури. Організатор або головний суддя створює й конфігурує татамі, закріплений суддя призначає та звільняє поточний поєдинок, обирає категорію для табло й отримує знімок стану майданчика для відновлення з'єднання; глядацький контур замикає

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

спостереження за публічним табло татамі в реальному часі. Цілісну діаграму варіантів використання подано в Додатку Б.

Клас поєдинку пов'язаний сам із собою двома рефлексивними асоціаціями (перехід переможця до наступного поєдинку та, у форматах із втішними зустрічами, переможеного до нижньої сітки), що перетворює сукупність поєдинків на спрямований граф. Поєдинок зберігає стан напівструктурованим документом і є агрегатом упорядкованого потоку незмінних подій. Логіку оцінювання відокремлено від доменних класів: правила нарахування балів і визначення переможця винесено в ієрархію класів-стратегій рушія правил (патерн «Стратегія»), що взаємодіє з поєдинком через сервісний рівень [5]. Детальну діаграму класів предметної області подано в Додатку В.

Ієрархія рушія правил реалізує сформульовану в першому розділі вимогу розширюваності. Абстрактний базовий клас визначає спільний інтерфейс набору правил (формування початкового стану, опрацювання узагальненої події, визначення переможця командної зустрічі), від якого успадковуються два проміжні класи за способом суддівства: для безперервного нарахування балів (з операціями застосування результативної дії та перевірки дострокового завершення) і для визначення переможця голосуванням колегії (з агрегацією прапорців). Конкретні дисципліни успадковуються від відповідного проміжного класу та реалізують лише власні параметри й алгоритми, тож підключення нової дисципліни зводиться до створення класу-нащадка та його реєстрації, не зачіпаючи ядра системи.

Логіку інкапсульовано у методах доменних класів («товсті моделі»), що зберігає цілісність даних незалежно від кінцевої точки (ендпоінту) API, з якого відбувається виклик [4]. Окремо опрацьовано рекурсивне просування учасників сіткою та автоматичну зміну статусів змагань за розкладом.

Основні методи доменних класів та їхнє алгоритмічне призначення наведено в таблиці 2.3.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

Призначення ключових методів доменних класів системи

Клас	Метод	Призначення та алгоритм роботи
User	get_full_name()	Повертає конкатеноване повне ім'я користувача. Враховує наявність або відсутність по батькові (patronymic), що потрібно для офіційних протоколів та звітів.
RoleRequest	save()	Перевизначений метод збереження моделі. Містить тригерну логіку: якщо запит переводиться персоналом у статус APPROVED, метод автоматично оновлює поле role у пов'язаного об'єкта User.
Athlete	calculate_current_age(ref_date)	Динамічно обчислює повну кількість років спортсмена на передану дату ref_date (зазвичай це дата старту турніру). Використовується категоріями для перевірки допуску.
Tournament	open_registration()	Переводить змагання зі стану DRAFT у REGISTRATION.
Tournament	complete_tournament()	Завершує змагання. Звільняє персонал та суддів (відв'язує від татамі), розраховує борг комісії платформи (platform_fee_amount) за офлайн-оплатами, ініціює масове повернення (refund) через MonobankService для заявок, що були зняті.
Tournament	auto_transition_statuses()	Метод класу (викликається за розкладом). Аналізує системний час та дати турнірів; автоматично переводить турніри зі статусу «Draft» у «Registration» і далі в «Active», якщо організатор не перевищив credit_limit.
Category	validate_athlete_eligibility()	Перевіряє спортсмена за трьома вимірами: стать (зокрема MIXED), вік (через calculate_current_age()) та відповідність базової ваги обмеженням min_weight / max_weight.
Registration	confirm_weight_in(weight)	Фіксує фактичну вагу. Здійснює перевірку ваги проти обмежень категорії, переводить заявку в статус CONFIRMED та створює запис аудиту в моделі AthleteWeightLog.

Клас	Метод	Призначення та алгоритм роботи
Match	set_winner(reg, method)	Виконується у транзакції. Встановлює переможця поєдинку, після чого: 1) якщо це командний бій – оновлює рахунок батьківського матчу; 2) інакше викликає advance_participant(); 3) перевіряє умови генерації репешажу або гранд-фіналу (Bracket Reset).
Match	advance_participant()	Переносить переможця до next_match.reg_first або reg_second. Одночасно аналізує loser_next_match (для нижньої сітки) і переносить туди переможенного. Викликає handle_auto_walkover().
Match	handle_auto_walkover()	Виявляє ситуації неявки (WITHDRAWN). Якщо суперник був попередньо знятий з турніру, метод автоматично присуджує технічну перемогу (Walkover) активному спортсмену.
BaseRuleSet	determine_team_winner()	Обчислює переможця командної зустрічі, підраховуючи виграні індивідуальні поєдинки в межах team_bouts. За нічиєї порівнює сумарні технічні бали або викликає Extra Bout.

Описану об'єктно-орієнтовану структуру розподілено за модулями: вона реалізується в архітектурі модульного моноліту, обґрунтованій у підрозділі 1.3 [20]. Кожна функціональна область відповідає окремому модулю (застосунку фреймворку) з визначеною зоною відповідальності, що взаємодіє з рештою через визначені програмні інтерфейси. Модуль облікових записів реалізує ідентифікацію та рольову модель; модуль довідників містить картки спортсменів, команди та їх приналежність до клубів; модуль турнірів зосереджує бізнес-логіку турнірів, категорій, реєстрацій та їх життєвого циклу; модуль жеребкування містить алгоритмічне ядро генерації сіток на чистих функціях; модуль поєдинків поєднує доменні класи поєдинку та події з координаційним сервісом, що пов'язує суддівські дії з рушієм правил і журналом подій; модуль наборів правил реалізує ієрархію стратегій оцінювання з реєстром; модуль татамі керує майданчиками та їх зв'язком із поєдинками й табло; модуль фінансів обслуговує рахунки, транзакції та виведення коштів з інтеграцією платіжного шлюзу; наскрізний службовий модуль містить спільні утиліти, зокрема

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

трансляцію подій реального часу та засоби посторінкового виведення. Відповідність основних класів модулям узагальнено в таблиці 2.4.

Таблиця 2.4

Розподіл основних класів за модулями системи

Модуль системи	Зона відповідальності	Основні класи та компоненти
Облікові записи	Ідентифікація, рольова модель, організаційна структура	User, Club, RoleRequest, EmailConfirmationCode, PasswordResetCode
Довідник спортсменів	Ведення спортсменів і команд	Athlete, Team, AthleteWeightLog
Турніри	Турніри, категорії, реєстрації, життєвий цикл	Tournament, Category, Registration
Жеребкування	Алгоритми генерації сіток усіх форматів	BracketGenerator, RepechageService, функції утиліт
Поєдинки	Перебіг поєдинків, журнал подій, координація суддівства	Match, MatchEvent, MatchService
Набори правил	Ієрархія стратегій оцінювання та реєстр	BaseRuleSet, PointsRuleSet, FlagsRuleSet, конкретні набори правил, реєстр
Татамі	Майданчики, поточні поєдинки, публічні табло	Tatami, TatamiService
Фінанси	Рахунки, транзакції, виведення коштів, платіжний шлюз	PaymentInvoice, Transaction, PayoutRequest, MonobankService
Службовий модуль	Наскрізні утиліти, трансляція подій реального часу	Функції трансляції, засоби посторінкового виведення, обробники WebSocket

Сформована модель варіантів використання, рольова модель розмежування доступу та об'єктно-орієнтована структура предметної області утворюють концептуальний каркас системи, на якому ґрунтуються подальші проєктні рішення. Наступний підрозділ конкретизує цей каркас на рівні зберігання даних, подаючи проєктування структури реляційної бази даних та відповідну діаграму «сутність-зв'язок».

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

2.2. Проектування бази даних системи

Зберігання стану платформи реалізовано на базі об'єктно-реляційної СКБД PostgreSQL. Більшість сутностей (турніри, категорії, заявки, поєдинки, фінансові операції) має чітку структуру та зберігається у реляційних таблицях із декларативними обмеженнями. Окремі фрагменти зі змінною формою, яка залежить від підключеного набору правил (стан поєдинку, конфігурація цін, навантаження подій), зберігаються як документи JSON.

Систему керування базами даних обрано на етапі аналізу засобів реалізації (підрозділ 1.4) – це об'єктно-реляційна СКБД PostgreSQL. У проєктних рішеннях цього підрозділу задіяно дві її можливості. Перша – апарат декларативних обмежень цілісності: зовнішні ключі з різними правилами реагування на видалення, часткові унікальні та перевірочні обмеження. Він дозволяє перенести частину правил предметної області на рівень схеми, де їх дотримання забезпечує сама СКБД, не покладаючись на коректність прикладного коду. Друга – нативний тип JSONB з індексуванням, що дає гібридне зберігання, за якого слабоструктуровані фрагменти співіснують із реляційними стовпцями в одному записі.

Логічну схему сформовано відображенням моделей об'єктно-реляційного відображення (ORM) фреймворку Django на таблиці реляційної бази. Кожному класу-моделі відповідає таблиця, кожному атрибуту – стовпець відповідного типу, кожному відношенню асоціації – зовнішній ключ або, для зв'язків «багато до багатьох», сполучна таблиця. Схему нормалізовано до третьої нормальної форми: усі неключові атрибути функціонально залежать від первинного ключа, а транзитивні залежності усунуто винесенням довідкових сутностей (клуб, користувач) в окремі таблиці [1, 36]. Первинними ключами в усіх таблицях є сурогатні цілочислові ідентифікатори, що спрощує побудову зовнішніх ключів і не залежить від змін природних ідентифікаторів. Концептуальну модель у нотації «сутність–зв'язок» з усіма основними таблицями та відношеннями подано в Додатку Г. Нижче структуру таблиць розглянуто за функціональними підсистемами.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

Ядром предметної моделі є ієрархія «турнір → категорія → заявка», що відтворює організацію змагань: турнір складається з вагово-вікових категорій, у кожен з яких подають заявки. Корінною сутністю ієрархії є таблиця tournament, яка зберігає організаційні відомості про змагання та повний набір фінансових налаштувань (таблиця Ж.1, Додаток Ж). Окрім наведених стовпців, турнір пов'язаний із користувачами двома відношеннями «багато до багатьох» – складом персоналу та суддівською колегією, – кожне з яких реалізовано сполучною таблицею з парою зовнішніх ключів. Стан турніру змінюється не довільно, а через методи життєвого циклу (таблиця 2.3); завершення турніру є складеною операцією, що звільняє персонал і суддів, нараховує комісію платформи за офлайн-заявками та ініціює відкладене групове повернення коштів за знятими онлайн-оплаченими заявками.

Кожен турнір поділяється на категорії, що задають межі дивізіону та параметри проведення поєдинків (таблиця Ж.2, Додаток Ж). Поле ruleset_key з'єднує категорію з механізмом наборів правил з попереднього підрозділу: збережений рядковий ключ використовує реєстр для добору стратегії оцінювання під час суддівства. Несуперечливість вікового та вагового діапазонів перевіряє метод очищення перед збереженням, а відповідність конкретного спортсмена критеріям категорії – окремий метод валідації (таблиця 2.3).

Участь спортсмена або команди в категорії фіксує таблиця заявок registration. Вона пов'язує довідник спортсменів із турнірною структурою і зберігає стан проходження учасником адміністративних та фінансових процедур (таблиця Ж.3, Додаток Ж). Заявка стосується або спортсмена, або команди. Цю взаємовиключність контролює метод очищення залежно від ознаки командної категорії. Повторну реєстрацію того самого учасника в одній категорії унеможливлюють часткові унікальні обмеження (розглянуто далі). Перехід заявки в стан підтвердження виконує метод зважування, який викликає персонал або головний секретар турніру. Цей метод звіряє зафіксовану вагу з межами категорії та створює запис у журналі зважувань

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

спортсмена. Переведення заявки в стан знятої каскадно ініціює автоматичне присудження технічних перемог суперникам у запланованих поєдинках.

Профіль спортсмена відокремлено від облікового запису користувача: адміністративні дії від імені спортсмена виконує його тренер, а сам спортсмен здебільшого не користується платформою. Довідкову підсистему утворюють три таблиці – профіль athlete, журнал контрольних зважувань athlete_weight_log та команда team для групових дисциплін (таблиця Ж.4, Додаток Ж). Журнал зважувань ведеться автоматично: створення профілю породжує початковий запис із базовою вагою, а кожне офіційне зважування на турнірі додає черговий запис із посиланням на джерело заміру, тож історія ваги лишається повною та хронологічно впорядкованою. Поле приміток для системних записів заповнюється фіксованими шаблонами, а для записів, доданих тренером вручну, – довільним текстом. Склад команди задає відношення «багато до багатьох» між таблицями команди та спортсмена через сполучну таблицю, що дозволяє одному спортсменові входити до кількох команд. Поточний вік не зберігається стовпцем, а обчислюється на опорну дату методом з таблиці 2.3, що уникає зберігання похідних даних та їх застарівання.

Змагальну підсистему утворюють таблиці поєдинку match, журналу подій match_event та майданчика tatami. Таблиця match поєднує три ролі: вузол турнірного дерева, носій поточного рахунку й таймера та точку прив'язки до фізичного майданчика (таблиця Ж.5, Додаток Ж). Рекурсивні зовнішні ключі next_match_id та loser_next_match_id зберігають топологію сітки на рівні бази; їх докладно розглянуто далі в частині про моделювання дерева. Фіксація результату виконується в межах атомарної транзакції, після чого переможець переноситься в наступний поєдинок за вказівником next_match_id (таблиця 2.3).

Кожну суддівську дію – нарахування балу, попередження, рішення прапорами, операції з таймером – фіксує окремий запис таблиці match_event, що реалізує модель реєстрації подій [6] (таблиця Ж.6, Додаток Ж). Пару стовпців match_id та sequence захищено унікальним обмеженням: воно гарантує лінійне впорядкування подій

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

кожного поєдинку та забезпечує ідемпотентність, бо повторне надходження події з уже зайнятим номером база відхиляє, унеможливаючи подвійне зарахування за нестабільного зв'язку.

Фізичні майданчики описує таблиця *tatami*, що пов'язує турнір, призначеного суддю, поточний поєдинок та апаратне табло (таблиця Ж.7, Додаток Ж). Пару стовпців турніру та номера майданчика захищено унікальним обмеженням, що унеможливорює два майданчики з однаковим номером у межах турніру.

Фінансову підсистему утворюють три таблиці: рахунок на оплату *billing_payment_invoice*, що агрегує суму до сплати та інтегрується з платіжним шлюзом; транзакція *billing_transaction*, що фіксує кожен рух коштів, зокрема повернення; та запит на виведення коштів *billing_payout_request*, через який організатор отримує зароблені внески (таблиця Ж.8, Додаток Ж). Один рахунок може охоплювати кілька заявок – наприклад, коли керівник клубу сплачує внески за весь контингент, – що відображено відношенням «багато до багатьох» між рахунком і заявкою. Цілочислова сума транзакції з можливим від'ємним значенням описує єдиним типом запису і оплату, і повернення, а унікальність ідентифікатора (референсу) транзакції захищає від повторного проведення тієї самої операції.

Підсистему ідентифікації та організаційної структури утворюють таблиця облікових записів *auth_user* з рольовою моделлю доступу, довідник клубів *club*, таблиця запитів на верифікацію ролі *role_request* та допоміжні таблиці одноразових кодів підтвердження пошти й скидання пароля (таблиця Ж.9, Додаток Ж). Схвалення запиту на верифікацію автоматично змінює глобальну роль користувача на запитану, а для ролі тренера встановлює його приналежність до вказаного клубу; цю логіку інкапсульовано в методі збереження запиту (таблиця 2.3), що унеможливорює зміну ролі поза процедурою верифікації.

Турнірна сітка є деревом, у якому переможці поєдинків нижчого рівня просуваються до поєдинків вищого рівня аж до фіналу. Для зберігання такої структури обрано список суміжності – класичне подання ієрархій у реляційних базах

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

[44]: кожен запис поєдинку містить рекурсивний зовнішній ключ next_match_id на поєдинок, до якого переходить його переможець. Корінь дерева – фінал – має next_match_id зі значенням NULL; формат подвійного вибування додатково використовує ключ loser_next_match_id для просування переможеного до нижньої сітки, що перетворює дерево на орієнтований граф. Зв'язки нижньої сітки встановлюються під час генерації сітки. Такий підхід тримає всю топологію в одній таблиці, без окремих таблиць рівнів чи раундів. Логіку просування показано на рисунку 2.1 на прикладі сітки на вісім учасників.

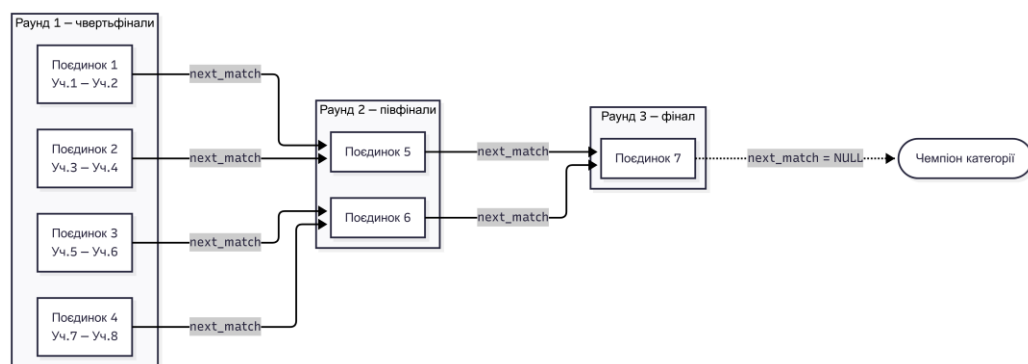


Рис. 2.1 – Рекурсивне моделювання турнірної сітки на вісім учасників за допомогою зовнішнього ключа next_match з посиланням на ту саму таблицю

Сітку категорії завантажують одним плоским запитом за ідентифікатором категорії, а групування поєдинків за раундами та відновлення дерева виконують у кодї застосунку. Таке рішення уникає лавини дрібних запитів, характерної для почергового обходу дерева зверненнями до бази, і узгоджується з проєктним показником часу завантаження сітки (таблиця 2.1; вимірювання наведено в розділі 3). Список суміжності припускає обхід і засобами рекурсивного запиту СКБД, проте в поточній реалізації відновлення дерева покладено на застосунок.

Частина даних предметної області має змінну форму, що залежить від контексту і не лягає на наперед задану реляційну схему. Опис такої інформації фіксованим набором стовпців дав би або розріджену таблицю з порожніми полями, або узагальнену структуру «сутність–атрибут–значення» з її відомими вадами. Тому для змінних фрагментів застосовано гібридне зберігання: реляційні стовпці – для даних з

усталеною структурою, за якими виконують вибірки та з'єднання, а тип JSONB – для фрагментів, форму яких визначає підключений набір правил або стан зовнішньої взаємодії. Підхід застосовано в чотирьох місцях схеми. Стовець match_state таблиці поєдинку зберігає проміжний стан суддівства, склад якого різниться для дисциплін: оцінювання за очками, підрахунок прапорів у формальних комплексах і послідовність боїв командної зустрічі мають різну структуру, зведення якої до єдиного набору стовпців було б штучним. Стовпці ruleset_prices та ruleset_team_prices таблиці турніру зберігають відображення «ключ набору правил → ціна», склад ключів якого заздалегідь невідомий і залежить від набору дисциплін змагання. Стовець payload таблиці подій містить навантаження події, структуру якого визначає її тип. Перевага JSONB перед текстовим зберіганням – можливість індексувати документи й виконувати запити за їхнім вмістом засобами СКБД, що тримає слабоструктуровані дані доступними для аналізу без втрати гнучкості форми.

Суддівство поєдинку реалізовано за моделлю реєстрації подій (Event Sourcing) [6], за якої джерелом істини є не поточний знімок стану, а впорядкована послідовність незмінних подій, що до нього призвели. Кожна суддівська дія дописується окремим записом до таблиці match_event і не змінюється та не вилучається; строге впорядкування подій поєдинку гарантує унікальне обмеження на пару «поєдинок – порядковий номер». Поточний рахунок у стовпцях таблиці поєдинку та у фрагменті match_state є матеріалізованою проєкцією журналу, призначеною для швидкого читання без щоразового відтворення стану з усього журналу. Така організація дає кілька практичних наслідків. Журнал подій утворює повний аудиторський слід поєдинку із зазначенням автора кожної дії, що потрібно для розгляду спірних ситуацій. Відтворення стану послідовним застосуванням подій дозволяє реалізувати скасування останньої дії як проєкцію журналу без останнього запису. Ідемпотентність, забезпечена унікальністю порядкового номера, захищає від подвійного зарахування за нестабільного зв'язку. Послідовність подій слугує джерелом для трансляції змін стану всім підключеним спостерігачам у реальному

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

часі, що відповідає проєктному показнику латентності комунікаційного шару (таблиця 2.1; вимірювання наведено в розділі 3).

Частину правил предметної області перенесено на рівень схеми, де їх дотримання забезпечує сама СКБД. Цілісність посилань підтримують зовнішні ключі з правилами реагування на видалення, дібраними за семантикою зв'язку: довідкові записи, втрата яких порушила б історію, захищено від видалення, поки на них є посилання; підпорядковані записи, що не мають сенсу поза батьківським, вилучаються разом із ним; некритичні посилання при втраті пов'язаного запису обнуляються, завдяки чому видалення майданчика чи судді не знищує історію проведених поєдинків. Унікальність бізнес-сутностей (пошта користувача, назва клубу, ідентифікатор транзакції, номер майданчика в турнірі, порядковий номер події в поєдинку) забезпечують унікальні обмеження. Семантичні правила, що не виражаються простими обмеженнями (узгодженість віку, ваги, дат, типу учасника категорії), перевіряють методи очищення на рівні застосунку. Продуктивність найчастіших вибірок – за позицією в сітці, станом поєдинку, приналежністю спортсмена, станом заявки – забезпечують індекси на основі В-дерева за відповідними стовпцями. Точний перелік обмежень та індексів кожної таблиці наведено в її структурі в Додатку Ж.

Часткові унікальні обмеження таблиці заявок мають особливість: оскільки заявка стосується або спортсмена, або команди, звичайне унікальне обмеження було б непридатним через порожні значення одного з ключів. Тому застосовано два часткові унікальні індекси, що активуються лише для непорожнього відповідного ключа і забезпечують унікальність окремо для індивідуальних та командних заявок. Структурну коректність самого турнірного графа – відсутність циклів і єдиний корінь – забезпечують не обмеження бази, а детермінований алгоритм генерації сітки та перевірка зайнятості слотів під час просування учасника, що відхиляє запис у вже заповнений поєдинок. Тобто посиальну цілісність гарантує схема, а коректність топології дерева – прикладний рівень.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

Динаміку руху даних між сховищами під час життєвого циклу турніру (реєстрація, зважування, жеребкування, суддівство) наведено на діаграмі потоків даних у Додатку Д.

Спроектована структура бази даних забезпечує зберігання стану всіх підсистем платформи з підтриманням посилювальної цілісності на рівні схеми та з гнучкістю для слабоструктурованих даних. Наступний підрозділ переходить від статичної структури зберігання до алгоритмів, що оперують цими даними, – насамперед до автоматичної генерації турнірних сіток та оброблення суддівських подій у реальному часі.

2.3. Проектування математичного забезпечення та алгоритмів роботи системи

Алгоритми генерації турнірних сіток реалізовано як чисті функції, ізольовані від ORM. Вони оперують спрощеним представленням учасника (ідентифікатор, найменування, клуб, посівний номер), а порожній прохід (BYE) позначається сигнальним об'єктом. Сервісний рівень перетворює підтверджені заявки на структури-учасники, викликає відповідну функцію і зберігає отриману топологію турніру у вигляді записів поєдинків у межах однієї транзакції.

Таблиця 2.5

Підтримувані формати турнірних сіток та алгоритми їх генерації

Формат	Умова застосування	Базовий алгоритм
Олімпійська система (single elimination)	Великі індивідуальні категорії	Доповнення до 2^k , стандартний посів, побудова дерева вибування
Олімпійська система з репешажем	Категорії з боротьбою за бронзу	Та сама сітка вибування з добудовою втішних гілок
Подвійне вибування (double elimination)	Другий шанс після першої поразки	Верхня і нижня сітки, гранд-фінал
Коловий формат (round robin)	Малі категорії	Метод кола

Формат	Умова застосування	Базовий алгоритм
Швейцарська система (swiss)	Велика кількість учасників за обмежений час	Парування з мінімізацією різниці результатів

Для командних категорій формат визначається автоматично під час генерації: за наявності щонайменше шести команд застосовується олімпійська система, інакше – коловий формат.

Сітка з вибуванням є повним двійковим деревом [10], що вимагає кількості початкових слотів, рівної ступеню двійки. Для довільної кількості підтверджених заявок $n \geq 2$ розмір сітки визначається за формулою (2.1):

$$N = 2^{\lceil \log_2 n \rceil}, \quad (2.1)$$

де N – розмір сітки (кількість початкових слотів); n – кількість підтверджених заявок.

Обчислення реалізовано цілочисельним зсувом без операцій із рухомою комою, що виключає похибки округлення логарифма. Кількість порожніх проходів, якими сітку доповнено до повного дерева, визначається за формулою (2.2):

$$B = N - n, \quad (2.2)$$

де B – кількість порожніх проходів (BYE); N і n – розмір сітки та кількість заявок відповідно. Число раундів сітки визначається за формулою (2.3):

$$R = \lceil \log_2 N \rceil = \log_2 N, \quad (2.3)$$

де R – кількість раундів сітки з вибуванням.

Розподіл учасників за слотами здійснюється стандартним алгоритмом посіву, що гарантує зустріч найсильніших учасників якомога пізніше. Послідовність позицій будується рекурсивно: для мінімальної сітки $S(1) = (1,2)$; кожне подвоєння розміру до $N_{k+1} = 2^{k+1}$ розгортає кожну позицію попереднього рівня у пару за формулою (2.4):

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

$$S_{2i-1}^{(k+1)} = S_i^{(k)}, \quad S_{2i}^{(k+1)} = N_{k+1} + 1 - S_i^{(k)}, \quad (2.4)$$

де $S_i^{(k)}$ – i -й елемент послідовності посіву для сітки розміру $N_k = 2^k$; N_{k+1} – розмір сітки на наступному рівні рекурсії.

Для восьми слотів послідовність (1, 8, 4, 5, 2, 7, 3, 6); сусідні пари утворюють поєдинки першого раунду. Учасники впорядковуються за посівним номером (за його відсутності – за ідентифікатором), розставляються по відповідних позиціях, решта слотів заповнюється порожніми проходами.

Топологію поєдинків формує окрема функція: слоти першого раунду обходяться парами, наступні раунди добудовуються до кореня. Посилання next_match кожного поєдинку вказує на батьківський за правилом $\lfloor (\text{match_order} - 1) / 2 \rfloor$; слот, у пару до якого потрапив порожній прохід, породжує поєдинок з автоматичним переможцем. Узагальнену схему алгоритму показано на рисунку 2.2.

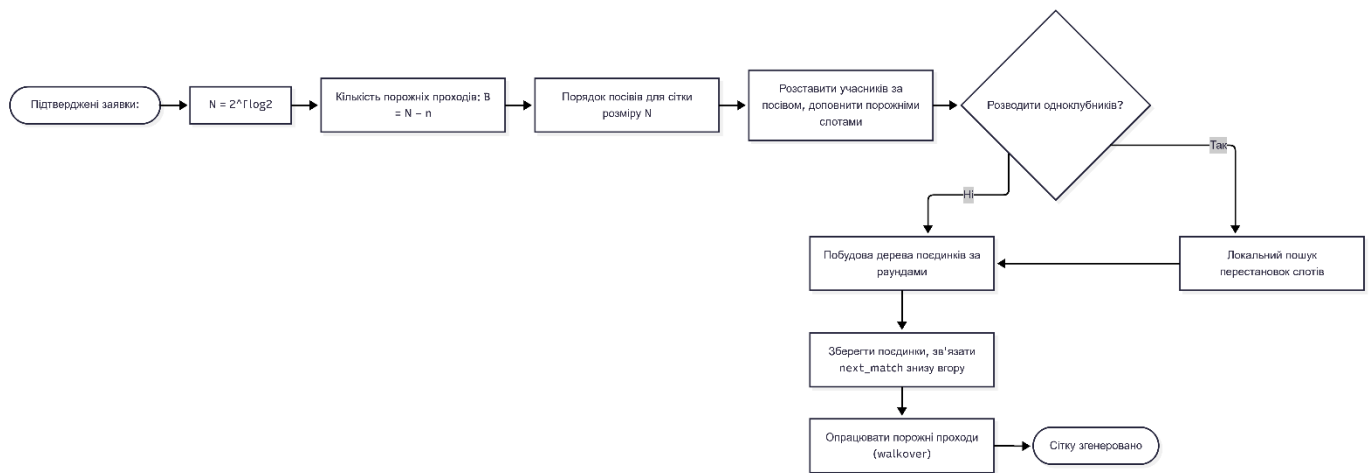


Рис. 2.2 – Узагальнена схема алгоритму генерації сітки з вибуванням

Стандартний посів оперує лише спортивним рейтингом і не враховує клубної приналежності учасників, що може зводити вихованців одного клубу вже в першому раунді. Для пом'якшення цього застосовано евристику розведення одноклубників (Club Seeding) [9]. Кількість конфліктів у розкладі слотів s рахується за формулою (2.5):

$$C(s) = |\{j: s_{2j} \neq \emptyset, s_{2j+1} \neq \emptyset, \text{club}(s_{2j}) = \text{club}(s_{2j+1})\}|, \quad (2.5)$$

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

де s – розклад слотів першого раунду; s_{2j} і s_{2j+1} – учасники j -ї пари; $club(\cdot)$ – клуб учасника; символ $\emptyset$ позначає порожній прохід.

Алгоритм виконує стохастичний локальний пошук: на кожній з не більш ніж 100 ітерацій знаходить першу конфліктну пару і обмінює її другого учасника з випадково обраним слотом, приймаючи перестановку лише при зменшенні $C(s)$. Пошук завершується достроково при нульовій кількості конфліктів. Оскільки безконфліктне розведення за наявності численних представників одного клубу не завжди досяжне, алгоритм повертає найкращу знайдену конфігурацію без гарантії нульового результату. Схему евристики наведено на рисунку 2.3.

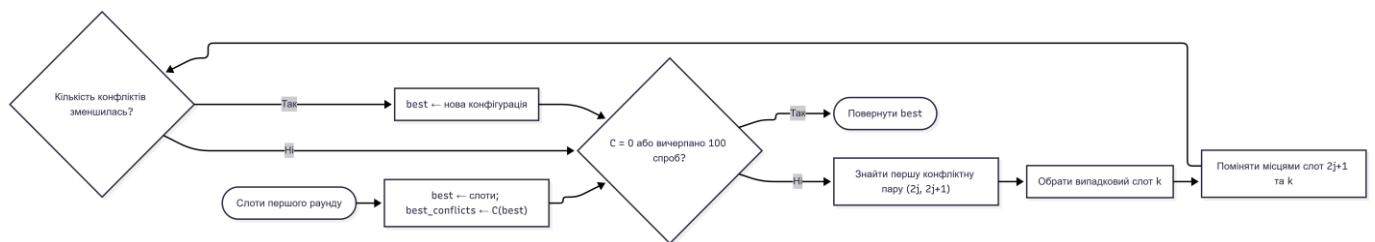


Рис. 2.3 – Евристика розведення одноклубників у першому раунді

У коловому форматі кожен учасник зустрічається з кожним; розклад будується за класичним методом кола [9]. За непарної кількості учасників до списку додається фіктивний учасник – суперництво з ним означає вільний тур. Перший елемент списку фіксується, решта циклічно обертається; у кожному турі формуються симетричні пари. Загальна кількість турів визначається за формулою (2.6),

$$T = n - 1, \quad (2.6)$$

а сумарна кількість поєдинків – за формулою (2.7):

$$\binom{n}{2} = \frac{n(n-1)}{2}, \quad (2.7)$$

де T – кількість турів колового турніру; n – кількість його учасників.

Підсумкові місця в коловому форматі визначаються за набраними очками (три за перемогу, одне за нічию), а за їх рівності – за результатом особистої зустрічі та різницею набраних і пропущених балів.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Швейцарська система поєднує обмежену кількість турів із зведенням суперників із близькими результатами, спираючись на визнані міжнародні стандарти парування [11]. Перший тур формується детерміновано: учасники впорядковуються за посівом, після чого перша половина списку зводиться з другою. За непарної кількості найслабший учасник отримує технічну перемогу у вільному турі. Гранична кількість турів визначається за формулою (2.8):

$$T_{\max} = \lceil \log_2 n \rceil, \quad (2.8)$$

де $T_{\max}$ – гранична (максимальна) кількість турів швейцарської системи; n – кількість учасників.

Кожен наступний тур мінімізує сумарну розбіжність результатів суперників за умови, що жодна пара не повторюється. Задачу сформульовано як пошук досконалого парування M мінімальної вартості за формулою (2.9):

$$\min_M \sum_{(a,b) \in M} (\text{score}(a) - \text{score}(b))^2, \quad b \notin \text{Opp}(a) \quad \forall (a,b) \in M, \quad (2.9)$$

де M – шукане досконале парування туру; $\text{score}(a)$ – поточна кількість очок учасника a ; $\text{Opp}(a)$ – множина суперників, з якими учасник a вже провів поєдинки.

Розв'язок знаходиться методом гілок і меж [45] з поверненням: перший учасник без пари зводиться по черзі з допустимими кандидатами, відсортованими за зростанням різниці очок. Гілка відсікається при досягненні поточного мінімуму штрафу; загальна кількість ітерацій обмежена 10 000 для гарантованого завершення. За неможливості сформувати тур без повторень алгоритм повертає порожній результат, і система сигналізує про це. Хід пошуку показано на рисунку 2.4.

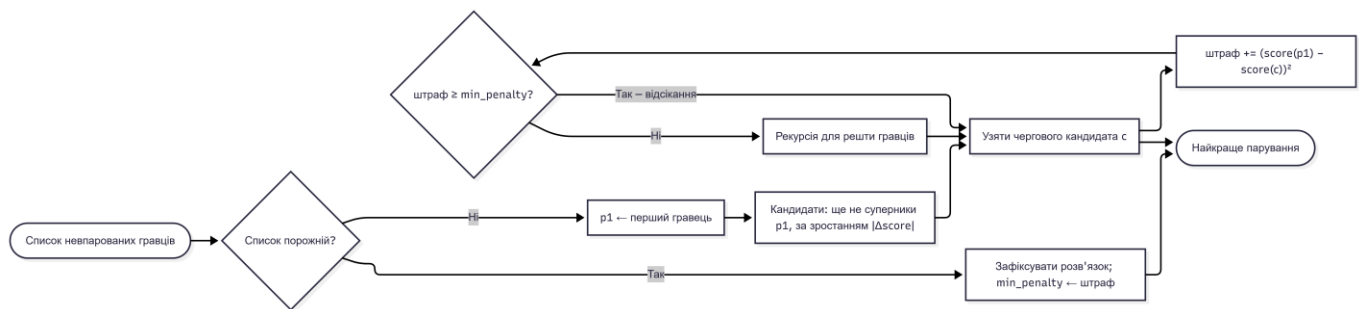


Рис. 2.4 – Пошук оптимального парування швейцарського туру методом гілок і меж

Підсумкове ранжування у швейцарській системі враховує не лише власні очки, а й силу суперників за коефіцієнтом Бухгольца, що визначається за формулою (2.10):

$$\text{Buchholz}(p) = \sum_{o \in \text{Opp}(p)} \text{points}(o), \quad (2.10)$$

де p – поточний учасник; $\text{Opp}(p)$ – множина всіх суперників, з якими учасник p уже провів поєдинки; $\text{points}(o)$ – кількість очок, набраних суперником o .

Учасники впорядковуються за лексикографічним ключем (очки, коефіцієнт Бухгольца, різниця балів, набрані бали, стабільний випадковий код для розв'язання абсолютних збігів), після чого додатковий прохід коригує порядок сусідів з однаковою кількістю очок за результатом їхньої особистої зустрічі.

Платформа підтримує два механізми надання учасникам другого шансу після поразки. Сітка репешажу є надбудовою над олімпійською системою: втішні гілки добудовуються окремою службою після визначення фіналістів. Подвійне вибування – самостійний формат із повноцінними верхньою і нижньою сітками. Для розрізнення раундів обох механізмів у межах однієї таблиці поєдинків застосовано діапазонне кодування `round_index` (таблиця 2.6).

Таблиця 2.6

Діапазонне кодування індексу раунду для складених форматів

Діапазон <code>round_index</code>	Призначення
1 ... k	Раунди верхньої сітки (і звичайної олімпійської системи)

Діапазон round_index	Призначення
100 + lvl	Рівні нижньої сітки подвійного вибування
200	Гранд-фінал
201	Вирішальний поєдинок після скидання сітки (bracket reset)
300 + j	Поєдинки втішних гілок (репешаж)

У подвійному вибуванні учасник вибуває лише після другої поразки. Верхня сітка будується за правилами олімпійської; кожен переможений у ній переходить до нижньої, де поразка вже остаточна, а кількість поєдинків на lvl -му рівні нижньої сітки визначається за формулою (2.11):

$$M_{lvl} = \frac{N}{2^{\lfloor \frac{lvl+3}{2} \rfloor}}, \quad lvl = 1, \dots, 2k - 2, \quad (2.11)$$

де M_{lvl} – кількість поєдинків на lvl -му рівні нижньої сітки; $N = 2^k$ – розмір сітки; k – кількість раундів верхньої сітки.

Представник нижньої сітки до гранд-фіналу має одну поразку; якщо він перемагає у гранд-фіналі, проводиться додатковий вирішальний поєдинок (bracket reset). У спрощеному варіанті гранд-фінал є єдиним і остаточним.

Втішна сітка (репешаж) формується для кожного фіналіста окремо: пул збирає всіх суперників, переможених цим фіналістом на шляху до фіналу (крім самого фінального поєдинку). При $p \geq 2$ переможених будується ланцюг втішних поєдинків з індексом раунду $300 + j$; переможець пулу здобуває бронзову нагороду. При $p = 1$ учасник отримує третє місце без додаткового поєдинку. Оскільки фіналістів двоє, передбачені дві бронзові нагороди – по одній на кожен пул.

Підсумкове ранжування учасників обчислює єдина служба, що диспетчеризує розрахунок залежно від формату сітки. Базова статистика накопичує для кожного учасника кількість перемог, нічий і поразок, набрані та пропущені бали; очки

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

нараховуються за стандартним принципом: 3 за перемогу, 1 за нічию. В олімпійській системі переможець фіналу займає перше місце, суперник – друге; два переможці півфіналів або обидва отримують третє місце, або розрізняються лексикографічним ключем для розмежування третього і п'ятого місць. У коловому форматі впорядкування здійснюється за очками через особисту зустріч і різницю балів; у швейцарській – за Бухгольцем; у форматах із репешажем і подвійним вибуванням – за переможцями відповідних гілок.

Ядро суддівства побудовано як узагальнений механізм за патерном «Стратегія» [5] у поєднанні з реєстром і моделлю машини станів [6]. Кожен набір правил є окремою стратегією, зареєстрованою під унікальним ключем; категорія посилається на нього, ядро отримує реалізацію через реєстр. Спільний інтерфейс визначає початковий стан поєдинку і функцію переходу: вона приймає поточний стан, тип події та корисне навантаження й повертає оновлений стан, ознаку завершення, переможця і спосіб перемоги. Набори правил поділяються на оцінювання за очками і голосуванням; у межах першої групи співіснують три парадигми (таблиця 2.7).

Таблиця 2.7

Парадигми оцінювання поєдинків у ядрі суддівства

Парадигма	Спосіб оцінювання	Умова завершення	Приклади дисциплін
Накопичення очок із порогом переваги	Бали за технічні дії, окремий облік попереджень	Поріг переваги в рахунку, ліміт попереджень, вичерпання часу або рішення суддів	Поєдинки («куміте» WKF), дзюдо
Голосування прапорами	Голоси непарної бригади суддів	Більшість прапорів	Дисципліна «ката» (демонстрація техніки)
Багатораундовий бій	Бали й порушення в межах кожного раунду	Дві виграні раунди або ліміт порушень	Тхеквондо

За першою парадигмою – накопичення очок із порогом переваги – кожна результативна дія додає фіксовану кількість очків, попередження обліковуються окремо. Поєдинок завершується достроково при досягненні порогу переваги або граничної кількості попереджень; інакше переможець визначає рахунок після завершення часу або рішення суддів. Параметризація порогом переваги, лімітом попереджень і тривалістю дозволяє одній моделі обслуговувати дисципліни з різними числовими налаштуваннями. Відповідний автомат показано на рисунку 2.5.

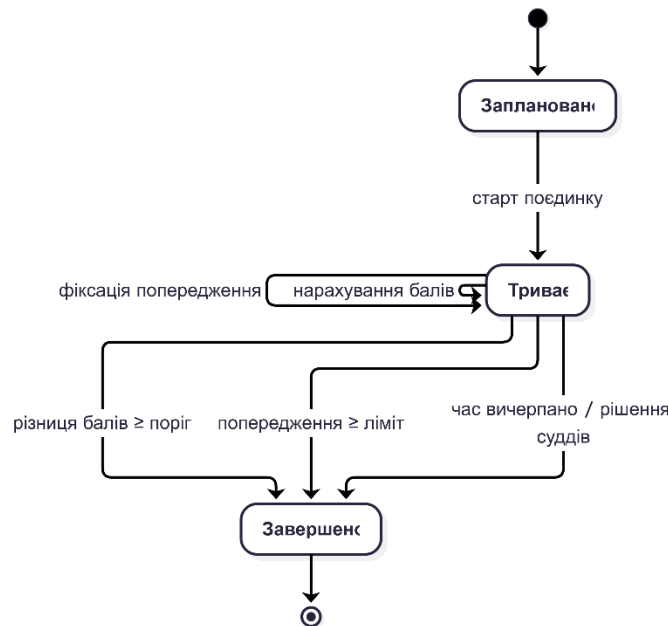


Рис. 2.5 – Машина станів поєдинку з накопиченням очок і порогом переваги

За другою парадигмою – голосування прапорами – бригада з непарної кількості суддів (3 або 5) одночасно голосує за одного з учасників. Переможець визначається за формулою (2.12):

$$\text{winner} = \begin{cases} \text{aka}, & f_{\text{aka}} > f_{\text{ao}}, \\ \text{ao}, & f_{\text{aka}} < f_{\text{ao}}, \end{cases} \quad f_{\text{aka}} + f_{\text{ao}} = J, \quad (2.12)$$

де f_{aka} , f_{ao} – кількість прапорів (голосів), поданих за учасників aka та ao відповідно; J – кількість суддів у бригаді. Непарність бригади виключає нічию.

Третя парадигма – багатораундовий бій – застосовується, зокрема, у тхеквондо: поєдинок складається з кількох раундів за принципом «до двох виграних». Стан

відстежує номер поточного раунду, рахунок, кількість порушень та кількість виграних раундів кожного бійця. Раунд може завершитися достроково при критичній перевазі в рахунку або граничній кількості порушень у його межах; накопичення критичних порушень за весь поєдинок призводить до миттєвої поразки незалежно від рахунку. Відповідний автомат показано на рисунку 2.6.

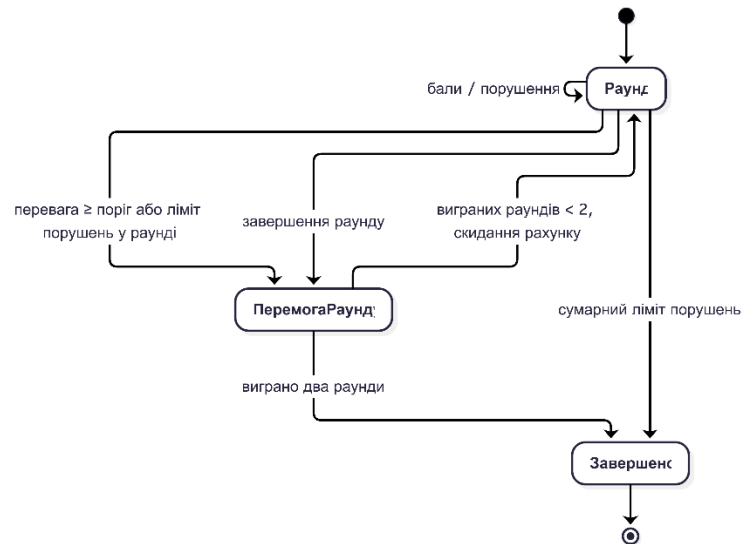


Рис. 2.6 – Машина станів багатораундового поєдинку

Усі набори правил з оцінюванням за очками підтримують командний формат: зустріч двох команд з s бійців складається з послідовних індивідуальних боїв між їхніми представниками. Переможця командної зустрічі визначає більшість виграних регулярних боїв: за розміру команди в s бійців необхідна кількість перемог визначається за формулою (2.13):

$$W = \left\lfloor \frac{s}{2} \right\rfloor + 1, \quad (2.13)$$

де W – необхідна для перемоги в зустрічі кількість виграних боїв; s – розмір команди (кількість бійців).

Якщо команда набирає W перемог, то виграє зустріч достроково. За рівної кількості перемог після всіх боїв переможця визначають за сумою технічних балів, а при повній рівності – за результатом додаткового вирішального бою.

Усі зміни стану поєдинку проходять через єдину координувальну службу, що пов'язує ядро суддівства, журнал подій і денормалізовану проєкцію стану. Після суддівської дії служба викликає функцію переходу відповідного набору правил, дописує до журналу незмінний запис події з порядковим номером і корисним навантаженням у форматі JSONB та оновлює поточний рахунок, статус і переможця поєдинку – все в межах однієї транзакції. Трансляцію змін спостерігачам відкладено до фіксації транзакції, що виключає розсилання стану, який може бути згодом скасовано. Часову послідовність виконання цих операцій та архітектуру транзакційного бар'єра наведено на рисунку 2.7.

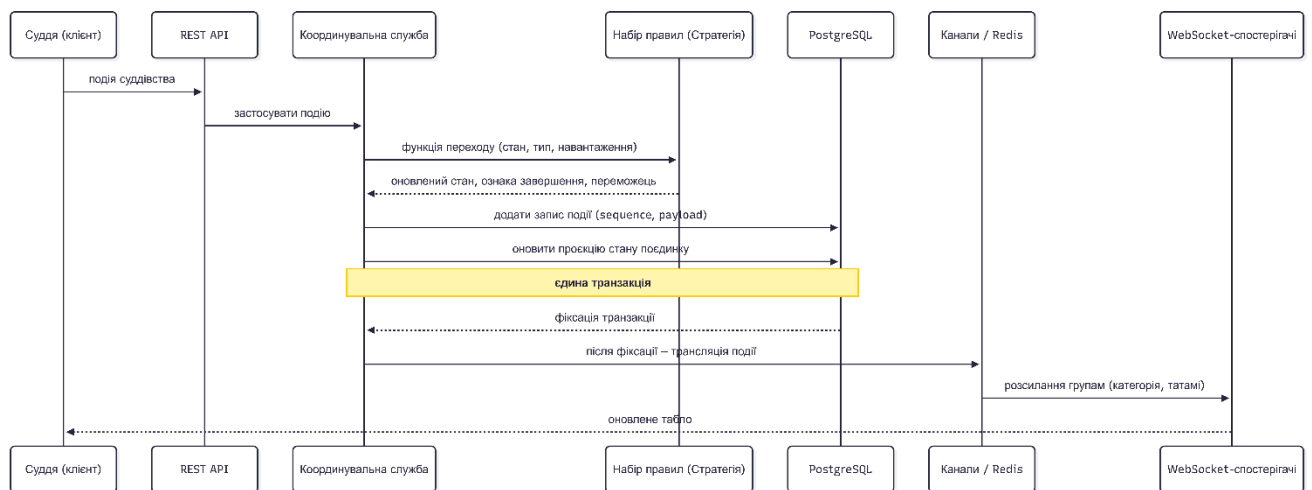


Рис. 2.7 – Послідовність оброблення суддівської дії та трансляції стану

Після успішної фіксації транзакції служба трансляції розсилає оновлений стан через рівень асинхронних каналів [29] за моделлю Pub/Sub [5, 46]. Кожна подія одночасно надходить до двох груп підписників – категорії та майданчика (татамі), – тому оновлення досягають і сторінки результатів, і табло одночасно. Клієнти отримують зміни через постійне WebSocket-з'єднання без повторного опитування сервера.

Загальну схему міжмодульної взаємодії та потоків даних у режимі реального часу зображено на рисунку 2.8.

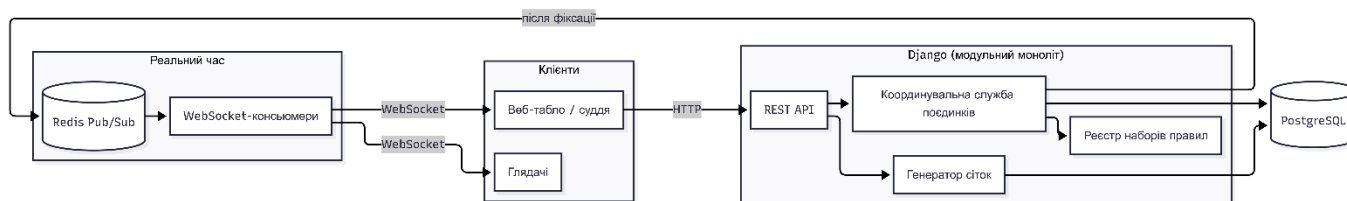


Рис. 2.8 – Взаємодія модулів системи в режимі реального часу

2.4. Реалізація програмного коду модулів серверної системи

Серверна частина системи реалізована мовою Python 3.12 [31] із застосуванням фреймворку Django 5.2 [33] та розширення Django REST Framework [24]. Підтримка режиму реального часу забезпечується бібліотекою Django Channels [29]. Повні лістинги ключових процедур винесено в Додаток Е.

Для розгортання та стабільної роботи серверної частини платформи сформовано базові вимоги до інфраструктури. Оскільки архітектура передбачає підтримку постійних WebSocket-з'єднань, кешування в оперативній пам'яті через Redis та інтенсивне обслуговування транзакційної бази даних PostgreSQL, ресурсна база має відповідати навантаженню. Зведені вимоги до апаратного забезпечення наведено в таблиці 2.8.

Таблиця 2.8

Вимоги до апаратного забезпечення серверного середовища

Характеристика	Мінімальні вимоги (тестове середовище)	Рекомендовані вимоги (production)
Процесор (vCPU)	2 ядра	4 ядра
Оперативна пам'ять	4 ГБ	8 ГБ
Накопичувач	SSD з 20 ГБ вільного простору	50+ ГБ NVMe SSD
Операційна система	Linux (Ubuntu 22.04 або новіша)	Linux (Ubuntu 22.04 або новіша)

Серверну систему організовано за архітектурним стилем модульного моноліту [20, 26, 28]. Код поділено на вісім предметних застосунків (apps) екосистеми Django

та один інфраструктурний модуль утиліт. Кожен застосунок відповідає за визначену зону предметної області і містить власні моделі, компоненти перетворення даних (серіалізатори), контролери та сервісний рівень. Модулі працюють у межах єдиного процесу та використовують спільну реляційну схему бази даних. Це дозволяє встановлювати прямі зв'язки між моделями різних доменів через механізм зовнішніх ключів (ForeignKey) і виконувати каскадні оновлення в межах однієї локальної транзакції. Зони відповідальності застосунків узагальнено в таблиці 2.9.

Таблиця 2.9

Застосунки серверної системи та їхні зони відповідальності

Застосунок	Зона відповідальності	Ключові одиниці
accounts	Користувачі, автентифікація, ролі, клуби	User, Club, рольові дозволи
athletes	Картки спортсменів, журнал зважувань	Athlete, AthleteWeightLog
tournaments	Турніри, категорії, заявки, авто-переходи статусів	Tournament, Category, Registration
rulesets	Абстраговане ядро суддівства різних дисциплін	BaseRuleSet, реєстр правил
brackets	Генерація сіток і жеребкування	BracketGenerator, чисті функції-утиліти
matches	Поєдинки, журнал подій, таймер, координатор стану	Match, MatchEvent, MatchService
tatamis	Татамі, черга та призначення поєдинків	Tatami
billing	Рахунки, операції повернення коштів, комісія платформи, інтеграція оплат	Transaction, MonobankService
common	Трансляція подій, спільні інфраструктурні утиліти	broadcast_*

Залежності між застосунками суворо спрямовані від змагальної логіки до ядра предметної області. Генерація сіток і поєдинки спираються на турніри й правила, татамі та білінг – на турніри. Ядро правил ізольоване від інших предметних застосунків, що сприяє його вільному повторному використанню в майбутньому. Граф основних залежностей між модулями наведено на рисунку 2.9.

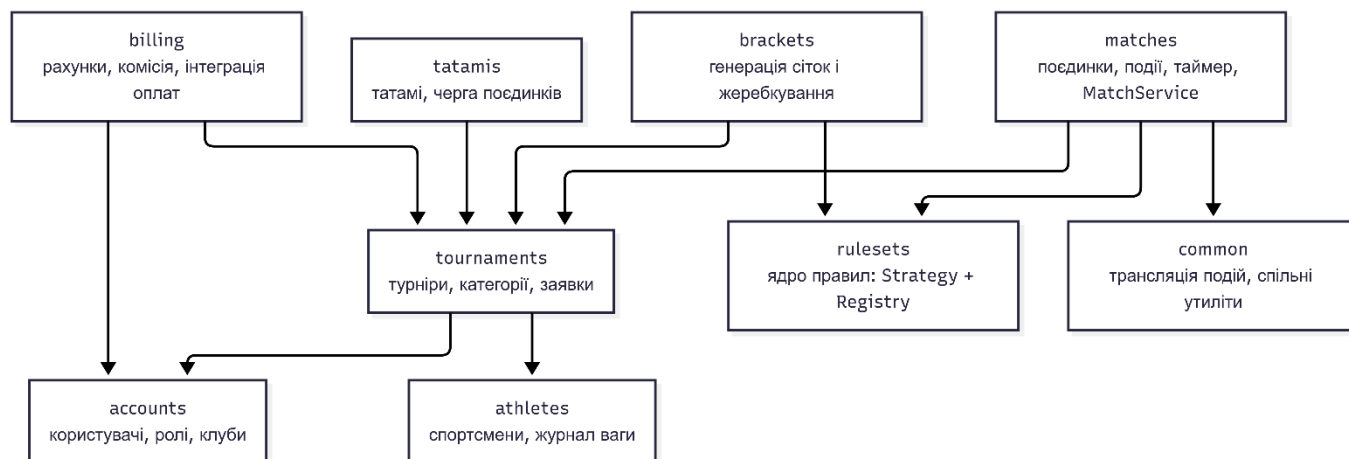


Рис. 2.9 – Структура застосунків модульного моноліту та залежності між ними

Підтримку різних дисциплін єдиноборств у ядрі правил реалізовано поєднанням патернів «Стратегія» [5] та «Реєстр». Кожна дисципліна описується окремим класом-стратегією, що інкапсулює правила нарахування балів, методи перемоги й умови завершення. Координатор поєдинку натомість оперує виключно абстрактним інтерфейсом. Реєстрація стратегій виконується декларативно: спеціальний декоратор класу під час імпорту додає правила до глобального словника за унікальним ключем. Отримання стратегії інкапсульовано у фабричній функції. Такий підхід підтримує принцип відкритості/закритості (додавання нової дисципліни зводиться до створення нового класу без модифікації наявного коду). Серіалізація правил також є динамічною: система обходить зареєстровані класи і формує для клієнта опис доступних дій, що дозволяє автоматично будувати інтерфейс оператора табло (лістинг E.1).

Основним компонентом серверної логіки виступає координатор стану поєдинку (клас MatchService), що керує його життєвим циклом. Координатор делегує застосування правил відповідній стратегії. Під час нарахування балів він відновлює поточний стан, передає подію стратегії, перевіряє умову завершення і зберігає результат. Послідовність викликів під час нарахування балів наведено на рисунку 2.10.

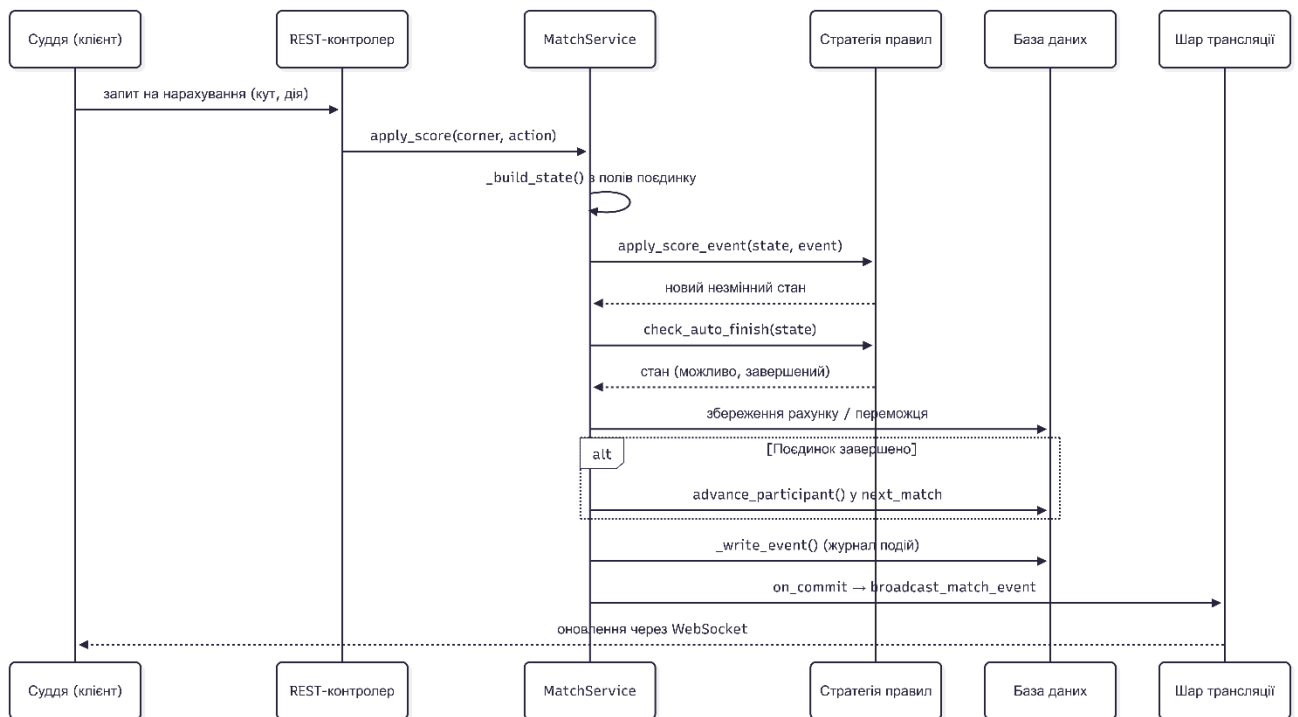


Рис. 2.10 – Послідовність викликів усередині координатора стану під час нарахування балів

Усі дії над поєдинком фіксуються окремим записом у журналі подій із послідовним порядковим номером. Унікальність пари «поєдинок – номер» жорстко підтримується на рівні бази даних. Це не лише дозволяє точно відтворювати перебіг поєдинку, але й дає змогу реалізувати скасування останньої дії шляхом повторного програвання подій (за принципом event sourcing).

Для уникнення маніпуляцій з ігровим часом таймер поєдинку валідується виключно сервером. Під час паузи розраховується фактичний приріст часу від моменту запуску. Якщо розбіжність із надісланим клієнтським значенням перевищує допустиму мережеву затримку, система приймає серверний розрахунок. Стан таймера транслюється всім спостерігачам із серверною позначкою часу для синхронізації відліку.

Фіксація результату поєдинку та просування переможця змінює стан одразу кількох записів, тому виконується атомарно в межах транзакції. Відповідний метод спочатку перевіряє коректність учасників, фіксує переможця, після чого просуває

його до наступного слоту в турнірній сітці (лістинг Е.4). Для формату подвійного вибування цей же метод спрямовує переможеного до відповідного рівня нижньої сітки. Розсилання оновлень спостерігачам відкладається до моменту успішної фіксації транзакції, що запобігає трансляції незбереженого чи помилкового стану.

Доставлення оновлень користувачам у реальному часі реалізовано через шар трансляції з використанням Django Channels [29] та брокера Redis. Повідомлення надсилаються паралельно у дві групи: потокової категорії та конкретного татамі (лістинг Е.5). Завдяки цьому досягається синхронне оновлення сторінки загальних результатів і локального суддівського табло.

Розмежування доступу побудовано на основі рольової моделі (RBAC) [43]. Організатор керує виключно власними турнірами, судді оперують призначеними татамі, а тренери керують заявками своїх спортсменів. Усі вхідні дані проходять сувору валідацію компонентами серіалізації Django REST Framework, що мінімізує ризики ін'єкцій. Передача даних здійснюється захищеним протоколом HTTPS/WSS, а для запобігання підробці міжсайтових запитів застосовуються вбудовані механізми захисту від CSRF.

Фінансові операції інтегровано з платіжним шлюзом Monobank [23]. Сервісний шар створює рахунки на оплату, обробляє повернення коштів і перевіряє статуси транзакцій. Достовірність сповіщень (зворотних викликів) від банку підтверджується криптографічною перевіркою підпису за алгоритмом ECDSA [47], що унеможливорює імітацію успішної оплати зловмисниками.

Відповідність винесених фрагментів коду лістингам додатка Е узагальнено в таблиці 2.10.

Таблиця 2.10

Відповідність винесених фрагментів програмного коду лістингам додатка Е

Фрагмент коду (модуль)	Призначення	Лістинг
Реєстр і фабрика правил (rulesets/registry.py)	Патерн «Стратегія» з реєстром та динамічна серіалізація правил	Е.1

Фрагмент коду (модуль)	Призначення	Лістинг
Оптимальне швейцарське парування (brackets/utils.py)	Парування з мінімізацією різниці результатів методом гілок і меж	Е.2
Генерація сітки подвійного вибування (brackets/services.py)	Побудова верхньої й нижньої сіток та гранд-фіналу	Е.3
Фіксація переможця і просування (matches/models.py)	Транзакційне завершення поєдинку та оновлення дерева сітки	Е.4
Шар трансляції подій (common/broadcast.py)	Віялове розсилання оновлень у групи category і tatami	Е.5

2.5. Компонентна будова та реалізація клієнтського додатку

Клієнтська частина системи реалізована як вебзастосунок з односторінковою архітектурою (Single Page Application, SPA) з використанням бібліотеки React 19 [38] та інструментарію Vite 8 [39]. Оскільки оператор татамі, тренер і глядач працюють із динамічним станом поєдинку, який змінюється кілька разів на секунду, така архітектура дозволяє оновлювати лише змінені фрагменти інтерфейсу. Це усуває потребу в повному перезавантаженні сторінки та підвищує загальну швидкість відгуку платформи.

Архітектура клієнтського застосунку побудована за багаторівневим принципом, що відповідає концепції відокремлення подання від логіки роботи з даними. Найвищий рівень утворюють сторінки (маршрути), наприклад: перелік турнірів, панель оператора чи інформаційне табло. Вони компонуються з повторно використовуваних елементів подання. Доступ до серверних даних та управління станом реального часу інкапсульовано в користувацьких хуках (custom hooks), які абстрагують деталі мережевої взаємодії. Найнижчий рівень становлять інфраструктурні модулі: клієнт API, утиліти безпеки та сховище глобального стану. Розподіл клієнтського коду за цими рівнями узагальнено в таблиці 2.11.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

Таблиця 2.11

Рівні організації клієнтського застосунку та їхня відповідальність

Рівень	Призначення	Складові
Сторінки (pages)	Прив'язані до маршрутів екрани застосунку	TournamentListPage, OperatorPanelPage, ScoreboardPage та інші
Компоненти подання (components)	Повторно використовувані елементи інтерфейсу	bracket, operator, scoreboard, tournament, layout, ui
Користувацькі хуки (custom hooks)	Інкапсуляція доступу до даних і стану реального часу	useWebSocket, useTatamiSocket, useTournamentSocket, useTimer
Інфраструктура (lib, store)	Мережева взаємодія, безпека, глобальний стан	api (axios), csrf, authStore (zustand)

Взаємодія між виділеними архітектурними шарами має сувору односпрямованість. Сторінки та візуальні компоненти не виконують прямих запитів до сервера, а звертаються до інфраструктурних модулів виключно через рівень користувацьких хуків. Це запобігає жорсткому зв'язуванню інтерфейсу з мережевою логікою та значно полегшує незалежне тестування складових. Загальний граф залежностей між рівнями клієнтського застосунку наведено на рисунку 2.11.

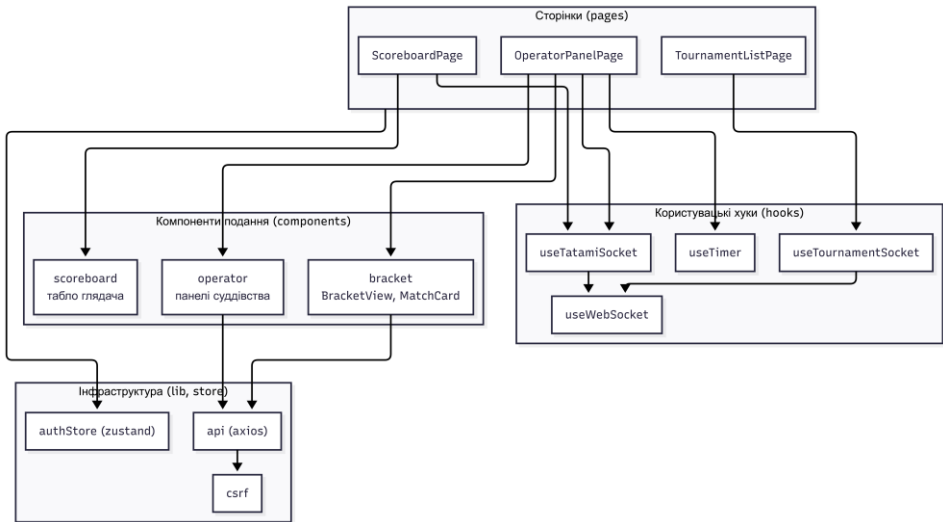


Рис. 2.11 – Компонентна структура клієнтського застосунку та залежності між рівнями

Навігацію застосунку реалізовано декларативно засобами бібліотеки React Router. Маршрути поділено на три групи: публічні (без загального макета), публічні зі спільним макетом та захищені. Контроль доступу забезпечує обгортковий компонент, який перевіряє стан автентифікації та наявність необхідної ролі [43] перед відображенням вмісту. Важливо зазначити, що клієнтська перевірка прав доступу виконує функцію покращення користувацького досвіду (уникнення запитів, які гарантовано завершаться помилкою), тоді як остаточне розмежування прав та гарантування безпеки забезпечується виключно серверною частиною.

Стан автентифікації керується за допомогою бібліотеки Zustand. Використовуються сеансові ідентифікатори, що передаються у захищених HttpOnly файлах cookie, що захищає їх від доступу з боку клієнтських скриптів та мінімізує ризики XSS-атак. Для захисту від міжсайтової підробки запитів (CSRF) реалізовано перехоплювач (interceptor), який додає спеціальний токен до заголовків модифікуючих запитів відповідно до сучасних методів протидії вебуразливостям [48].

Складність організації робочого місця судді зумовлена розмаїттям парадигм оцінювання в різних дисциплінах єдиноборств. Цю задачу вирішено перенесенням патерну «Стратегія» [5] на рівень клієнтських компонентів. Сторінка панелі оператора діє як компонент-диспетчер: вона утримує загальний стан поточного поєдинку, таймера та черги боїв, а вибір конкретного візуального компонента подання здійснює на основі правил категорії (таблиця 2.12).

Таблиця 2.12

Відповідність парадигм суддівства спеціалізованим панелям оператора

Ознака поєдинку	Панель оператора	Парадигма суддівства
Командна зустріч	Панель командної зустрічі	Послідовність індивідуальних боїв із підрахунком перемог
Ключ правил «Тхеквондо»	TaekwondoOperatorPanel	Реєстрація технічних дій із захисним спорядженням

Ознака поєдинку	Панель оператора	Парадигма суддівства
Ключ правил «джудо»	JudoOperatorPanel	Фіксація вищих оцінок («іппону», «вадза-арі») та покарань
Режим суддівства «за очками»	KumiteWKFOperatorPanel	Накопичення очок і попереджень («куміте»)
Режим суддівства «за балами»	KataOperatorPanel	Усереднення суддівських оцінок («ката»)

Такий підхід зберігає систему відкритою до розширення: для додавання нової спортивної дисципліни достатньо розробити окремий візуальний компонент і додати умову в диспетчер, не змінюючи решту клієнтського коду. Усі спеціалізовані панелі дотримуються спільного контракту взаємодії, отримуючи дані таймера та передаючи події через зворотні виклики (callbacks).

Двосторонній зв'язок із сервером реалізовано через протокол WebSocket [14]. Модуль (хук) керування з'єднанням автоматично відновлює розриви зв'язку за алгоритмом експоненційного відступу (Exponential Backoff) [15], поступово збільшуючи інтервал від однієї до тридцяти секунд. Це дозволяє уникнути лавиноподібного перевантаження сервера у випадку тимчасових мережеских збоїв. Клієнтський годинник синхронізується із серверним часом шляхом систематичного обчислення мережевої затримки: виміри з аномальними відхиленнями відкидаються, а результати експоненційно усереднюються для згладжування коливань.

Відлік ігрового часу відображається незалежно від частоти кадрів браузеру завдяки оптимізованому циклу оновлення. Для забезпечення миттєвої реакції інтерфейсу застосовано підхід оптимістичного оновлення (Optimistic UI). Завдяки вбудованим можливостям React 19 [38] результати дій судді одразу відображаються на екрані без очікування підтвердження від сервера, а згодом непомітно узгоджуються із серверним станом, що надходить через з'єднання за протоколом WebSocket. Поєднання оптимістичного відтворення з наступною синхронізацією приховує

мережеві затримки та забезпечує плавність роботи застосунку, зберігаючи при цьому авторитетність серверних даних.

Відображення турнірної сітки реалізовано компонентом, який автоматично адаптує візуалізацію до формату змагань. Формати з вибуванням подаються як набір раундів, а у випадку подвійного вибування (Double Elimination) поєдинки розподіляються на вкладки верхньої сітки, нижньої сітки та гранд-фіналу. Коловий формат і швейцарська система відображаються у вигляді підсумкової таблиці з додатковим розкладом турів. Інформаційне табло працює як окремий оптимізований маршрут без елементів навігації, що підписується на події татамі та синхронно відтворює стан, керований оператором.

Спроектовані підсистеми клієнтського застосунку забезпечують розділення логіки та подання, поліморфну обробку правил суддівства за допомогою патерну «Стратегія» і стабільну роботу в реальному часі через WebSocket.

2.6. Аналіз загроз та забезпечення інформаційної безпеки системи

Інформаційна безпека системи базується на захисті фінансових транзакцій, персональних даних учасників та результатів змагань від несанкціонованого доступу й модифікації. Захист реалізовано на рівнях клієнтського застосунку, прикладного інтерфейсу (API), бази даних та комунікаційних каналів.

Захист доменних даних забезпечує розмежування повноважень на основі ролей (RBAC) [43]. Архітектура системи розділяє відповідальність між клієнтським та серверним шарами. Клієнтська частина (React + Vite) виконує функції інтерфейсного відображення та покращення користувацького досвіду (UX), приховуючи або відображаючи елементи керування на основі збереженої ролі в стані автентифікації. Захист маршрутів на рівні клієнта за допомогою компонентів-обгорток запобігає випадковому переходу глядача до адміністративних панелей. Серверна частина (Django REST Framework) є єдиним авторитетним джерелом перевірки прав доступу. Усі запити до API проходять через декларативні класи дозволів (PermissionClasses).

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

Навіть у разі навмисної модифікації коду клієнтського застосунку зломисник не має змоги здійснити несанкціоновану дію, оскільки кожна кінцева точка API перевіряє не лише глобальну роль користувача (IsAuthenticated, IsOrganizer), а й контекстне право власності на об'єкт (наприклад, перевірка, чи є організатор власником турніру або чи закріплено суддю за відповідним татамі).

Захист сесійних даних браузера від міжсайтового скриптингу (XSS) реалізовано через збереження ідентифікатора сесії в захищених файлах cookie. Використання параметра SESSION_COOKIE_HTTPONLY = True забороняє доступ до файлів cookie за допомогою сценаріїв мовою JavaScript (зокрема через об'єкт document.cookie), що нейтралізує загрозу прямого викрадення активних сесій шкідливим кодом. Додатковий прапорець SESSION_COOKIE_SECURE = True гарантує передачу файлів cookie виключно шифрованими каналами під керуванням TLS.

Вбудований механізм перевірки токенів Django захищає від атак типу CSRF. Для всіх запитів, що змінюють стан системи (POST, PUT, PATCH, DELETE), клієнтська частина зчитує CSRF-токен із заголовка відповіді та передає його в HTTP-заголовку X-CSRFToken. Сервер відхиляє будь-які модифікаційні запити, якщо токени не збігаються. Для обмеження транскордонної передачі сесій додатково встановлено параметр SESSION_COOKIE_SAMESITE = "Lax", який дозволяє надсилати файли cookie лише в межах первинного контексту сайту, блокуючи автоматичне прикріплення сесії під час переходів із зовнішніх джерел.

На рівні бази даних головною загрозою є можливість виконання SQL-ін'єкцій через текстові поля введення. Запобігання цьому вектору забезпечується використанням об'єктно-реляційного відображення Django ORM [33]. Запити до бази даних формуються через конструктор запитів (QuerySet API), який автоматично застосовує параметризацію та екранування вхідних значень на рівні драйвера СУБД PostgreSQL. Ручні SQL-запити обмежено двома службовими діагностичними зверненнями (перевірка з'єднання та стан активності СУБД), жоден з яких не використовує конкатенацію користувацьких даних.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Захист прикладного інтерфейсу від передачі некоректних даних та атак «відмова в обслуговуванні» (DoS) починається з валідації. Усі вхідні дані проходять перевірку за допомогою вбудованих засобів серіалізації Django REST Framework [24]. Кожне поле перевіряється на відповідність типу, довжину, діапазон значень та бізнес-правила. Для захисту від автоматизованого перевантаження запитами (атак повного перебору паролів або генерації надлишкового трафіку) на рівні конфігурації фреймворку [24] налаштовано обмеження частоти запитів (Throttling). Клас AnonRateThrottle обмежує неавтентифікованих користувачів лімітом у 100 запитів на хвилину, а UserRateThrottle надає авторизованим учасникам 500 запитів. Для критичних операцій (автентифікація, відновлення паролів) клас ScopedRateThrottle встановлює жорсткий ліміт у 5 запитів на хвилину.

Захист каналів реального часу (Django Channels) від перехоплення та модифікації трафіку забезпечується трьома рівнями. Першим рівнем є застосування протоколу WSS (WebSocket Secure), за яким з'єднання встановлюється поверх TLS. Це забезпечує надійне шифрування даних на шляху між браузером та сервером і запобігає атакам типу «людина посередині» (MITM). Другий рівень полягає в обов'язковій автентифікації під час встановлення з'єднання (handshake), коли маршрутизатор ASGI за допомогою обгортки AuthMiddlewareStack [29] зчитує сесійні файли cookie клієнта та підтверджує особу користувача на сервері. Третім елементом захисту є суворий контроль напрямку передачі даних. WebSocket-канал функціонує виключно для трансляції (downstream) інформації від сервера до під'єднаних клієнтів. Клієнтські застосунки не мають права передавати керуючі команди або змінювати стан поєдинків через WebSocket-з'єднання. Усі дії суддів (нарахування балів, запуск таймера) надсилаються як стандартні REST-запити, що проходять перевірку прав доступу на рівні контролерів, завдяки чому нейтралізуються ризики ін'єкції команд через кадри WebSocket.

Інтеграція платіжної системи Monobank [23] створює ризики шахрайства через підміну зворотних викликів (повідомлень про успішну оплату). Для запобігання

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

зарахуванню фіктивних оплат без реального списання коштів розроблено механізм криптографічної перевірки кожного вхідного повідомлення від банку. Послідовність виконання криптографічної валідації зображено на рисунку 2.12. Усі сповіщення від Monobank супроводжуються цифровим підписом у заголовку X-Sign. Процедура перевірки складається з отримання публічного ключа банку (який кешується у Redis для захисту від надлишкових запитів), декодування підпису з формату Base64 та безпосередньої верифікації підпису тіла запиту. Для перевірки застосовується алгоритм електронного цифрового підпису на еліптичних кривих ECDSA в поєднанні з хеш-функцією SHA-256 (через бібліотеку cryptography). Математично це підтверджує, що повідомлення не було модифіковано в мережі й відправлено саме платіжним шлюзом Monobank. У разі невідповідності підпису транзакція автоматично відхиляється.

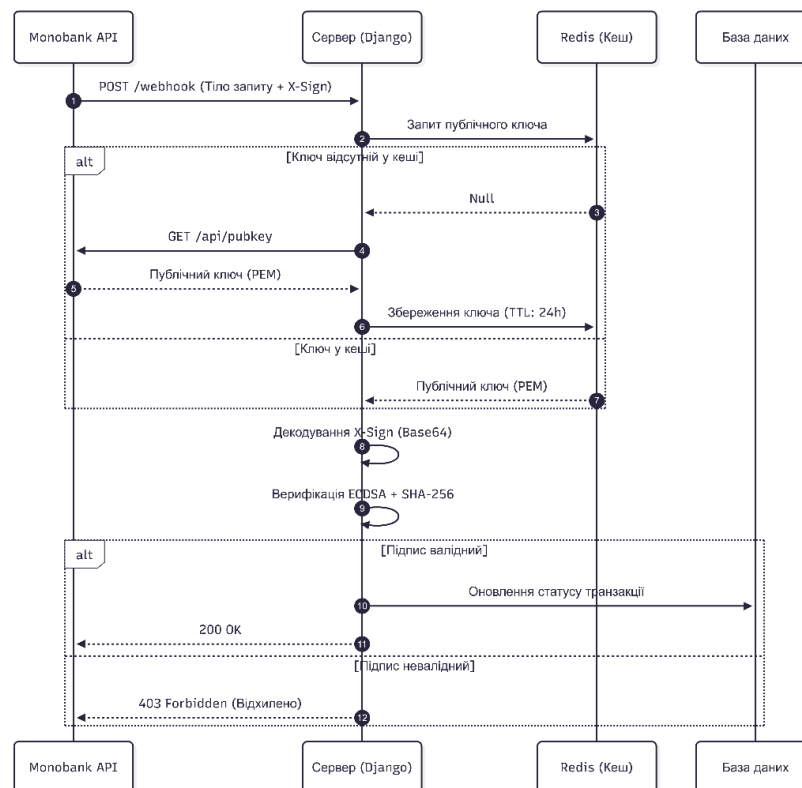


Рис. 2.12 – Послідовність криптографічної перевірки платіжних сповіщень

Вектори загроз та застосовані методи їх мінімізації систематизовано в таблиці 2.13.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

Матриця загроз інформаційній безпеці системи та методи їх мінімізації

Вектор загрози	Можливі наслідки	Метод мінімізації загрози
Підміна ідентифікатора користувача або привілеїв	Несанкціонована зміна ролі; отримання доступу до керування сторонніми турнірами.	Перевірка повноважень на рівні Django REST Framework (PermissionClasses); авторизація з контролем власності на об'єкти.
Маніпуляція даними змагань або результатами	Фальсифікація оцінок; зміна результатів реєстрацій чи турнірних сіток.	Ведення незмінного журналу подій поєдинку (Event Sourcing); обмеження доступу до панелі татами виключно для закріплених суддів.
Викрадення сесійних даних користувача	Перехоплення ідентифікатора сесії для компрометації облікового запису.	Зберігання сесій у файлах cookie із прапорцями HttpOnly, SameSite=Lax та Secure; передача трафіку виключно через HTTPS.
Міжсайтове підроблення запитів (CSRF)	Виконання небажаних дій від імені автентифікованого користувача при переході на сторонній ресурс.	Перевірка CSRF-токена (X-CSRFToken в заголовках) для всіх методів, що змінюють стан даних.
Впровадження стороннього коду (XSS)	Викрадення конфіденційних даних; виконання шкідливих сценаріїв у браузері.	Автоматичне екранування виводів у React; блокування доступу JavaScript до сесійних файлів cookie.
SQL-ін'єкції та атаки на бази даних	Несанкціоноване читання чи видалення даних; порушення працездатності СУБД.	Використання Django ORM із вбудованою параметризацією та екрануванням усіх запитів до БД.
Атаки типу «відмова в обслуговуванні» (DoS)	Перевантаження сервера запитами; уповільнення роботи платформи під час змагань.	Налаштування лімітів частоти запитів (Throttling) на рівні API Django REST Framework.

Вектор загрози	Можливі наслідки	Метод мінімізації загрози
Перехоплення WebSocket-трафіку або ін'єкція	Підслуховування перебігу поєдинків; спроби передачі фальсифікованих суддівських команд.	Використання шифрованого протоколу WSS; односпрямований (broadcast) характер з'єднання з блокуванням вхідних команд.
Фальсифікація платіжних сповіщень	Зарахування фіктивних оплат внесків без фактичного перерахування коштів.	Валідація цифрового підпису Monobank (X-Sign) за допомогою алгоритму ECDSA та публічного ключа.

Висновки до другого розділу

Рольова модель доступу розробленої системи базується на принципі RBAC і містить шість глобальних та три контекстні ролі. Набуття привілейованих прав вимагає обов'язкової верифікації адміністратором. Прецеденти використання згруповано у сім функціональних областей, а доменні класи розподілено між дев'ятьма додатками модульної архітектури. Схему реляційної бази даних нормалізовано до третьої нормальної форми. Зберігання слабоструктурованих конфігурацій реалізовано за допомогою нативного типу JSONB. Процес суддівства поєдинків працює за моделлю Event Sourcing. Цей підхід створює повний аудиторський слід кожної дії та гарантує ідемпотентність операцій в умовах нестабільного мережевого зв'язку на спортивній арені. Важливу частину правил предметної області перенесено безпосередньо на рівень реляційної схеми, тому їх дотримання гарантує база даних незалежно від прикладного коду застосунку. Алгоритмічне ядро здатне генерувати п'ять форматів турнірної сітки. Усі математичні алгоритми написані як чисті функції без залежності від об'єктно-реляційного відображення для забезпечення ізольованого тестування. Розширюваність правил суддівства забезпечує архітектурний патерн стратегії спільно з глобальним реєстром. Завдяки цьому підключення нової спортивної дисципліни відбувається без

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

модифікації наявного ядра системи. Узгодженість даних у режимі реального часу підтримує транзакційний бар'єр, який відкладає трансляцію оновлень до успішної фіксації транзакції, у поєднанні з оптимістичним оновленням інтерфейсу клієнта. Інформаційну безпеку побудовано на кількох незалежних рівнях. Комплекс заходів охоплює перевірку власності на об'єкти, захист сеансових даних у браузері, обмеження частоти запитів до програмного інтерфейсу, використання суворо односпрямованого каналу WebSocket та криптографічну перевірку платіжних сповіщень банку за алгоритмом ECDSA для запобігання фальсифікації транзакцій.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

РОЗДІЛ 3. ІНТЕРФЕЙС, РОЗГОРТАННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ

3.1. Порядок встановлення, налаштування та розгортання системи

Розгортання системи реалізовано як відтворюваний процес. Код описує середовище виконання, залежності та конфігурацію, що дозволяє розгорнути стек на будь-якому сумісному вузлі. Усі компоненти запаковано в контейнери Docker, їхня взаємодія задана декларативно, а доставлення нових версій на сервер виконує автоматизований конвеєр.

Програмний комплекс розгортається як сукупність ізольованих контейнерів, об'єднаних засобами Docker Compose. Контейнеризація фіксує для кожної служби заздалегідь визначений набір залежностей, тому розгортання на робочій станції розробника і на хмарному сервері дає однакове середовище й усуває розбіжності конфігурацій. Середовищем дослідної експлуатації обрано віртуальний сервер платформи Microsoft Azure (Ubuntu Server 24.04 LTS, машина серії B із підтримкою сплесків навантаження), що відповідає вимогам до апаратного забезпечення з підрозділу 2.4. Вибір Azure зумовлено безкоштовним освітнім доступом для студентів, безкоштовним повним доменним іменем виду <мітка>.<region>.cloudapp.azure.com для публічної IP-адреси, яке знімає потребу купувати окремий домен під TLS-сертифікат, та сплесковою ціною моделлю серії B, що відповідає нерівномірному навантаженню турнірної системи з піками в дні змагань.

Стек промислового середовища складається з п'яти служб. Зовнішнім периметром є контейнер caddy. Це зворотний проксі, який приймає трафік на портах 80 і 443, автоматично отримує та поновлює TLS-сертифікати від центру сертифікації Let's Encrypt і термінує захищене з'єднання. За проксі розташовано контейнер frontend з вебсервером nginx. Він віддає статичні файли застосунку React і виконує внутрішню маршрутизацію. Звернення до прикладного інтерфейсу (/api/), WebSocket-каналу (/ws/), службової панелі (/admin/) та файлів (/static/, /media/) nginx ретранслює на

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

застосунковий контейнер, а решту запитів обслуговує поверненням головного документа застосунку. Ядром системи є контейнер backend із застосунком Django під керуванням асинхронного сервера Daphne за протоколом ASGI. Це дозволяє одночасно обслуговувати синхронні HTTP-запити й довготривалі WebSocket-з'єднання. Стан системи зберігають дві службові бази. Контейнер db із PostgreSQL 16 є основним сховищем предметних даних, а контейнер redis із Redis 7 виконує роль шару каналів для трансляції подій реального часу.

Дані, які мають пережити перезапуск контейнерів, зберігаються в іменованих томах Docker. Том pgdata зберігає файли бази PostgreSQL, том media містить завантажені користувачами файли, а томи caddy_data і caddy_config зберігають отримані TLS-сертифікати та конфігурацію проксі. Це запобігає повторному зверненню до центру сертифікації після кожного перезапуску.

Контейнери взаємодіють у внутрішній мережі Docker за іменами служб, тому назовні відкрито лише порти проксі (80 і 443), а порт застосункового сервера (8000) та порти баз даних залишаються недоступними з-поза меж сервера, що звужує поверхню атаки. Схему розгортання зі складом контейнерів, напрямками трафіку та прив'язкою томів наведено на рис. 3.1.

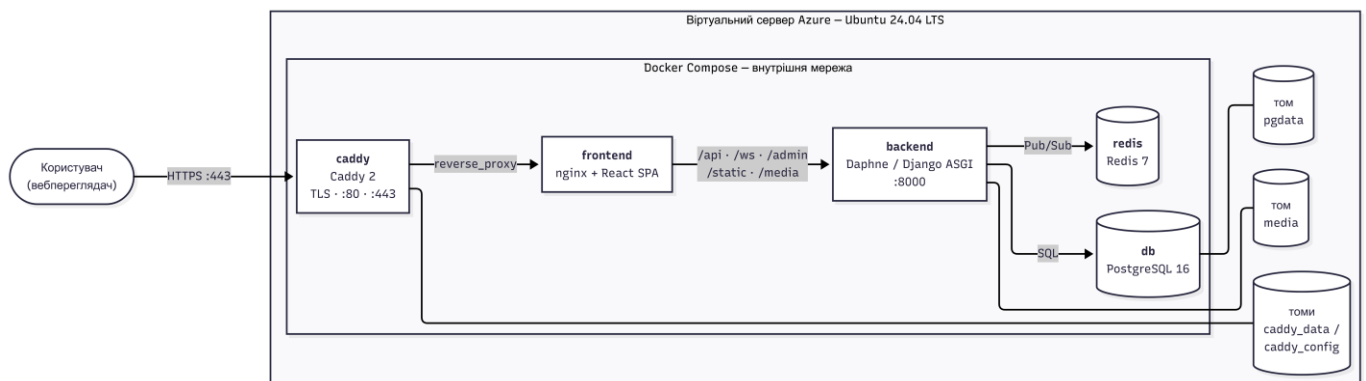


Рис. 3.1 – Схема розгортання програмного комплексу

Доступ до системи ззовні здійснюється за повним доменним іменем виду <мітка>.<region>.cloudapp.azure.com, яке безкоштовно надає Azure для публічної IP-адреси сервера. Це ім'я слугує одночасно адресою застосунку для користувачів і доменом, на який Caddy отримує TLS-сертифікат. Для розробки та локального

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

тестування передбачено окремий опис стека (docker-compose.yml), у якому образи збираються з вихідного коду, тоді як опис для промислової експлуатації (compose.prod.yml) завантажує вже зібрані образи з реєстру контейнерів, що розділяє контури розробки та промислової експлуатації.

Образи обох частин системи будуються за технологією багатоетапної збірки (multi-stage build): процес поділено на ізольовані стадії, а у фінальний образ потрапляє лише потрібне для виконання, без інструментів збирання та проміжних артефактів. Для клієнтської частини перша стадія на образі Node встановлює залежності та збирає production-версію застосунку збирачем Vite, а друга на полегшеному образі nginx отримує лише скомпільований статичний пакет; вихідний код, середовище Node та каталог залежностей у фінальний образ не переносяться, завдяки чому розмір образу клієнтської частини становить 77 МБ. Для серверної частини перша стадія встановлює Python-залежності, а друга на образі python:3.12-slim отримує тільки готові пакети та код застосунку, без кеша й засобів збирання, причому збирання статичних файлів виконується під час складання образу; розмір образу серверної частини становить 441 МБ. Обидва образи оснащено вбудованою перевіркою працездатності (HEALTHCHECK), що періодично звертається до проби /healthz/, тому стан кожного контейнера відстежується автоматично на рівні середовища виконання.

Конфігурування реалізовано винесенням усіх параметрів, що залежать від середовища, у змінні оточення, які застосунок зчитує під час запуску. Підхід відповідає методології 12-факторного застосунку (Twelve-Factor App) й дає дві переваги: той самий образ контейнера без повторного збирання працює і в розробці, і в промисловій експлуатації, а чутливі відомості (секретний ключ, паролі, токени) не потрапляють у вихідний код і репозиторій, а зберігаються лише у файлі оточення на сервері. На робочому сервері параметри задаються у файлі .env, який Docker Compose використовує і для підстановки значень в опис стека, і як джерело змінних оточення застосункового контейнера. Перелік основних параметрів конфігурування наведено в таблиці 3.1.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

Основні параметри конфігурування середовища (.env)

Параметр	Призначення	Приклад / типове значення
DJANGO_SECRET_KEY	Криптографічний ключ для підпису сесій, CSRF-токенів і службових даних	випадковий рядок 50+ символів
DJANGO_DEBUG	Режим налагодження; у промисловій експлуатації вимкнено	False
DJANGO_ALLOWED_HOSTS	Перелік доменних імен, від яких застосунок приймає запити	<мітка>.<region>.cloudapp.azure.com
DB_NAME, DB_USER, DB_PASSWORD	Реквізити підключення до бази PostgreSQL	tournament_db / tournament_user / <пароль>
DB_HOST, DB_PORT	Адреса служби бази у внутрішній мережі Compose	db / 5432
REDIS_HOST, REDIS_PORT	Адреса служби Redis для шару каналів	redis / 6379
USE_REDIS_CHANNEL_LAYERS	Увімкнення трансляції подій реального часу через Redis	True
DOMAIN	Домен, на який Caddy отримує TLS-сертифікат	<мітка>.<region>.cloudapp.azure.com
ACME_EMAIL	Контактна адреса для центру сертифікації Let's Encrypt	<пошта адміністратора>
IMAGE_BACKEND, IMAGE_FRONTEND	Координати образів у реєстрі GHCR	ghcr.io/<власник>/tournament-backend:latest
MONOBANK MOCK PAYMENTS	Режим імітації платежів без реальних списань	True
MONOBANK_USE_SANDBOX, MONOBANK_TOKEN	Параметри інтеграції з платіжним сервісом	True / <порожньо>
AUTH_THROTTLE_RATE	Обмеження частоти спроб автентифікації	5/minute

Окремий клас параметрів впливає на безпеку з'єднання й задається не у файлі оточення, а у самому описі стека або визначається автоматично. У промисловому режимі застосунок встановлює захищені атрибути для сесійних і CSRF-файлів cookie і довіряє заголовку X-Forwarded-Proto, який проксі передає у внутрішню мережу, що дозволяє Django розпізнавати початкове захищене з'єднання попри незашифрований внутрішній канал між проксі та вебсервером. Віддачу статичних файлів службової панелі у режимі з вимкненим налагодженням забезпечує проміжний шар WhiteNoise, інтегрований в образ застосунку, тому стиль адміністративного інтерфейсу зберігається без окремого сховища статички.

Послідовність розгортання на підготовленому сервері є детермінованою. У робочому каталозі розміщуються три файли: опис стека `compose.prod.yml`, конфігурація проксі `Caddyfile` та заповнений файл оточення `.env`. Після цього розгортання зводиться до завантаження образів із реєстру контейнерів, підняття стека у фоновому режимі та застосування міграцій бази даних.

Порядок запуску служб керується оголошеними залежностями зі станами готовності. Бази `db` і `redis` оснащено перевітками працездатності: для PostgreSQL це команда `pg_isready`, для Redis – `ping`. Контейнер `backend` оголошено залежним від обох баз з умовою `service_healthy`, тому Docker Compose не запускає його, доки бази не повідомлять про готовність приймати з'єднання. Контейнер `frontend` залежить від готовності застосункового сервера, а проксі `caddy` – від готовності вебсервера. Ланцюгова залежність задає строгий порядок підняття стека (`db`, `redis`, `backend`, `frontend`, `caddy`) і усуває стан гонитви, за якого застосунок міг би звернутися до ще не ініціалізованої бази.

Коректність розгортання підтверджується на двох рівнях. На інфраструктурному перевіряється стан контейнерів командою `docker compose -f compose.prod.yml ps`, яка має показати всі п'ять служб у стані «Up», а служби з перевітками працездатності – у стані «healthy». На застосунковому рівні

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

працездатність перевіряється зверненням до службових кінцевих точок ззовні, через публічний домен і захищене з'єднання.

Службові кінцеві точки реалізовано в модулі `apps/common/health.py`. Точка `/healthz/` є пробою життєздатності (`liveness`). Вона повертає HTTP-код 200 без звернення до зовнішніх залежностей і підтверджує, що застосунковий процес запущено. Точка `/readyz/` є пробою готовності (`readiness`). Вона послідовно перевіряє доступність бази (запитом `SELECT 1`) та шару каналів Redis (службовим повідомленням) і повертає код 200 виключно за справності обох залежностей, інакше повертається код 503 із переліком несправних компонентів. Поділ відповідає практиці оркестрування контейнерів: проба життєздатності виявляє завислий процес, що потребує перезапуску, а проба готовності визначає момент, коли застосунок можна вводити під навантаження; саме на неї спирається перевірка працездатності застосункового контейнера в описі стека. Успішне проходження перевірок свідчить, що проксі отримав TLS-сертифікат, контейнери стартували у правильному порядку, а застосунок установив з'єднання зі своїми залежностями. Результат перевірки наведено на рис. 3.2.

```
last login: Sun Jun 28 02:13:50 2026 from 37.115.67.140
gemn1@DiplomaVM:~$ cd ~/app
docker compose -f compose.prod.yml ps
curl -sI https://tournaments-app.spaincentral.cloudapp.azure.com/ | head -n 1
curl -s https://tournaments-app.spaincentral.cloudapp.azure.com/healthz/ && echo

curl -s https://tournaments-app.spaincentral.cloudapp.azure.com/readyz/ && echo
NAME                IMAGE                                COMMAND
SERVICE            CREATED                            STATUS          PORTS
app-backend-1       ghcr.io/gemn1/tournament-backend:latest  "daphne -b 0.0.0.0
..." backend      About an hour ago  Up About an hour (healthy)  8000/tcp
app-caddy-1         caddy:2-alpine                      "caddy run --config
..." caddy        About an hour ago  Up About an hour           0.0.0.0:80->80
/tcp, [::]:80->80/tcp, 0.0.0.0:443->443/tcp, [::]:443->443/tcp, 443/udp, 2019/tc
p
app-db-1            postgres:16-alpine                  "docker-entrypoint.
s..." db          About an hour ago  Up About an hour (healthy)  5432/tcp
app-frontend-1     ghcr.io/gemn1/tournament-frontend:latest  "/docker-entrypoint
..." frontend    About an hour ago  Up About an hour (healthy)  80/tcp
app-redis-1         redis:7-alpine                      "docker-entrypoint.
s..." redis       About an hour ago  Up About an hour (healthy)  6379/tcp
HTTP/2 200
{"status": "ok"}
{"status": "ok"}
gemn1@DiplomaVM:~/app$
```

Рис. 3.2 – Перевірка стану контейнерів та службових проб розгорнутої системи

Управління версіями вихідного коду здійснюється системою Git. Основним репозиторієм є репозиторій на GitHub, де виконуються конвеєри інтеграції та доставлення; додатково налаштоване автоматичне дублювання коду до репозиторію

університетської платформи GitLab (git.ztu.edu.ua) при кожному оновленні основної гілки. Розробку організовано за моделлю функціональних гілок: стабільний стан проекту зберігається в гілці main, а кожна нова можливість або виправлення розробляється в окремій гілці feature/*. Інтеграція змін відбувається через запити на злиття (Pull Request), які запускають конвеєр перевірок: зміни потрапляють до main лише після успішного проходження всіх етапів контролю якості. Така організація захищає стабільну гілку від неперевіраних змін і забезпечує простежуваність історії розробки.

Доправлення коду від репозиторію до робочого сервера автоматизовано засобами GitHub Actions, на якій побудовано чотири конвеєри, що покривають шлях від внесення зміни до її розгортання. Поділ на окремі конвеєри дозволяє запускати кожну групу перевірок за власною подією-тригером. Основним є конвеєр безперервної інтеграції (CI), що спрацьовує на кожне внесення змін та на кожен запит на злиття. Він складається з етапів контролю якості: статичний аналіз і перевірка стилю коду (ruff для серверної частини, eslint із компілятором TypeScript для клієнтської), сканування залежностей обох частин на відомі вразливості засобами rip-audit та npm audit, запуск тестів серверної частини під керуванням pytest із вимірюванням покриття та відправленням звіту до Codecov, складання клієнтського застосунку збирачем Vite, контрольне збирання образів контейнерів, а також наскрізне тестування інтерфейсу засобами Playwright на повністю піднятому стеку. Окремий конвеєр виконує статичний аналіз якості та безпеки коду на SonarCloud і спрацьовує на зміни в основній гілці та запити на злиття. Ще один конвеєр виконує реплікацію репозиторію до університетського GitLab. Завершує ланцюг конвеєр доставлення (CD), який запускається лише після успішного завершення інтеграційного конвеєра в основній гілці: він збирає остаточні образи, публікує їх у реєстрі GitHub Container Registry (GHCR), копіює актуальні файли розгортання на сервер захищеним каналом і віддалено оновлює стек – завантажує нові образи, перезапускає служби й застосовує міграції. Структуру конвеєра наведено на рис. 3.3.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

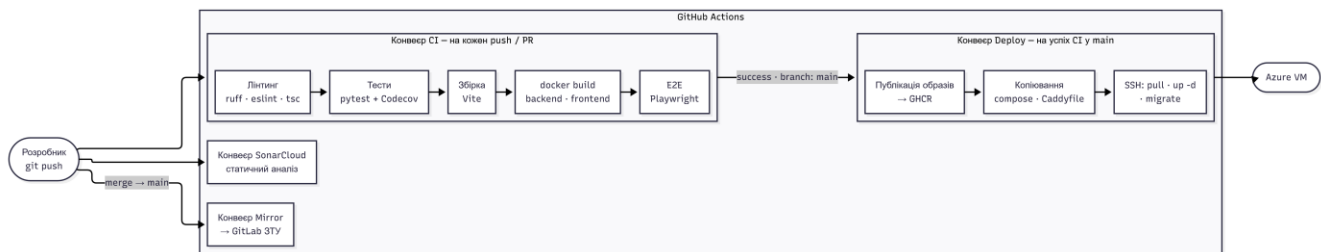


Рис. 3.3 – Структура конвеєра безперервної інтеграції та доставлення

Конвеєр реалізує принцип «зібрати один раз – розгорнути будь-де»: образи збираються виключно в середовищі GitHub Actions і завантажуються на сервер готовими, тому на сервері відсутні вихідний код та інструменти збирання – він лише завантажує образи з реєстру та запускає їх. Це зменшує вимоги до ресурсів сервера, скорочує час розгортання та гарантує ідентичність протестованого й розгорнутого артефактів. Прив'язка доставлення до успішного завершення інтеграції допускає на сервер лише код, що пройшов увесь контроль якості.

3.2. Структура та основні елементи інтерфейсу

Клієнтську частину реалізовано за моделлю односторінкового застосунку (SPA). Переходи між екранами виконуються на боці браузера без повного перезавантаження сторінки. З погляду користувача система поділяється на дві зони доступу. Публічна зона доступна без автентифікації й охоплює перегляд переліку турнірів, їхніх сіток, глобальних рейтингів спортсменів та інформаційних табло змагань у режимі реального часу. Захищена зона вимагає входу й відкриває користувачеві лише ті робочі місця, що відповідають його ролі. Склад головного меню формується динамічно залежно від ролі автентифікованого користувача. Інтерфейс адаптовано до різної ширини екрана, починаючи від настільних робочих станцій секретаріату і закінчуючи планшетами суддівських бригад біля татамі.

Каркас застосунку утворює сталий макет із верхньої панелі навігації, основної області вмісту та нижнього колонтитула. Верхня панель закріплена у верхній частині вікна й містить ліворуч логотип системи, що є посиланням на головну сторінку, по центру розміщено головне меню, а праворуч знаходиться елемент керування

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

обліковим записом. Склад головного меню залежить від стану автентифікації та ролі користувача. Пункти «Турніри» й «Рейтинги» доступні всім відвідувачам, пункт «Атлети» з'являється після входу, а спеціалізовані пункти «Панель тренера», «Панель секретаря» та «Мій татамі» відображаються лише авторизованим для цих розділів ролям. Неавтентифікованому відвідувачу праворуч показано кнопку «Увійти», а автентифікованому користувачеві надається спадне меню облікового запису з ім'ям, адресою електронної пошти, позначкою поточної ролі та переходами до профілю, робочого місця й виходу із системи. На вузьких екранах горизонтальне меню згортається у мобільне, що розкривається кнопкою-гамбургером, чим забезпечується робота із застосунком на планшетах і смартфонах. Структуру головної сторінки разом із панеллю навігації наведено на рис. 3.4.

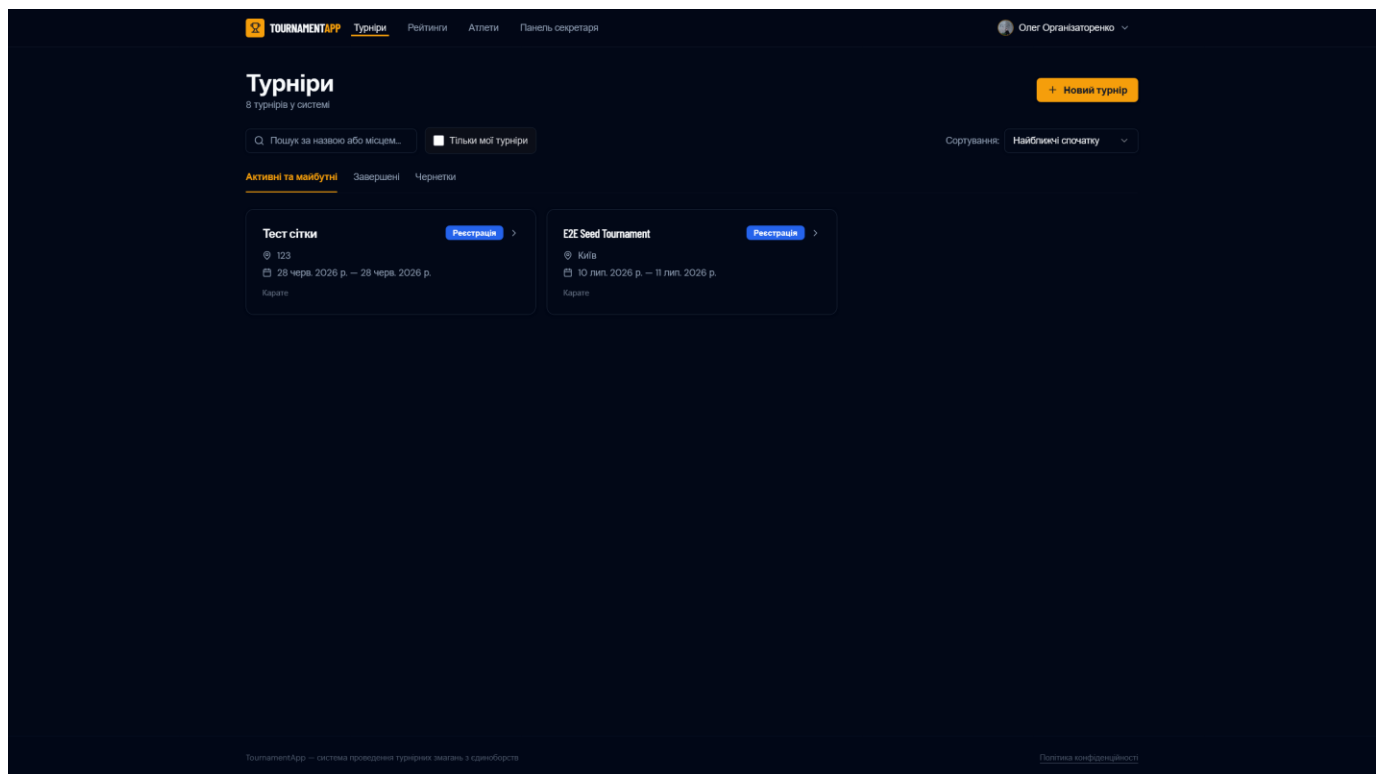


Рис. 3.4 – Головна сторінка системи та головне меню навігації

Доступ до захищених можливостей системи розпочинається з реєстрації облікового запису. Форма реєстрації збирає основні відомості про користувача та супроводжується підтвердженням адреси електронної пошти. На вказану адресу надсилається лист із посиланням-токеном, перехід за яким активує обліковий запис.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

Вхід реалізовано двома способами: парою «електронна пошта і пароль» та через зовнішнього постачальника ідентифікації Google, що позбавляє користувача потреби запам'ятовувати окремий пароль. На випадок втрати доступу передбачено відновлення пароля за посиланням, надісланим на пошту. Сторінку входу з обома способами автентифікації наведено на рис. 3.5.

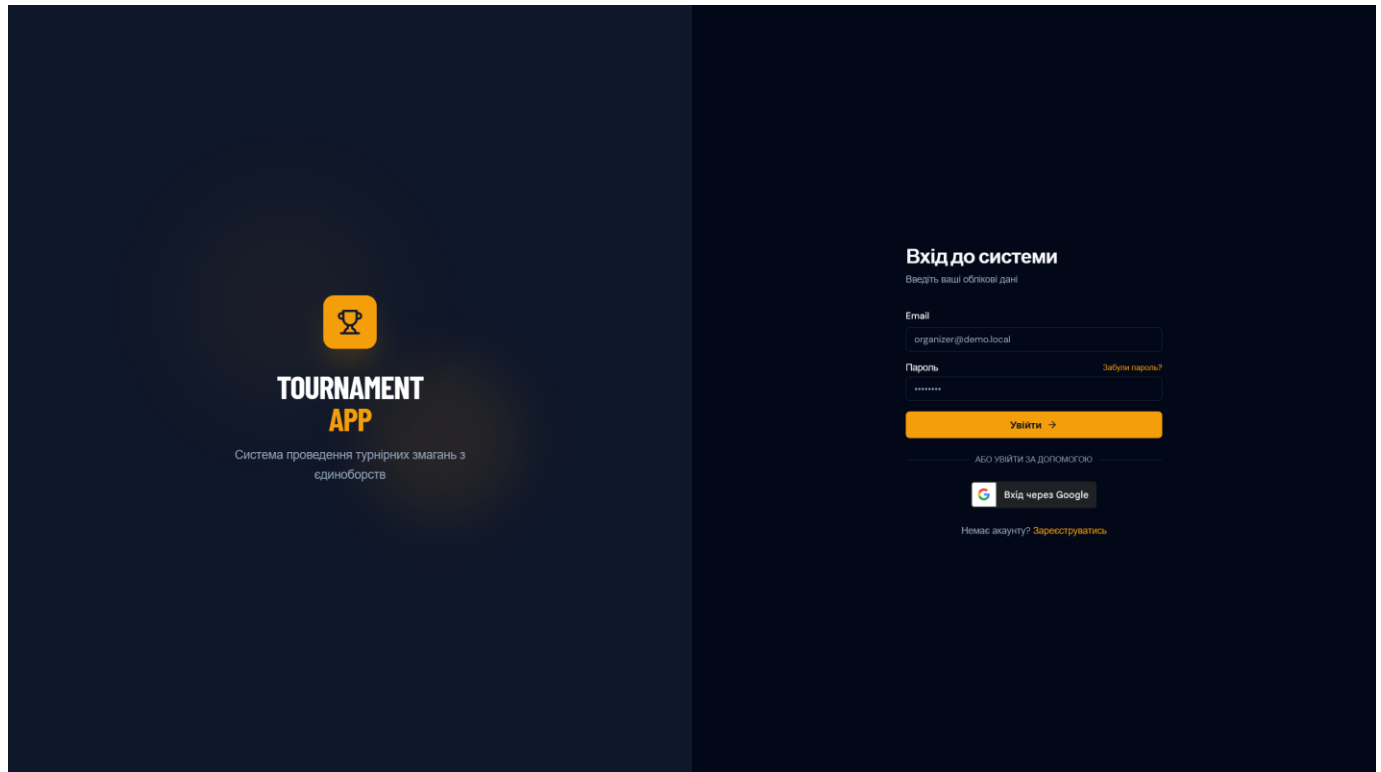


Рис. 3.5 – Сторінка входу до системи

Робоче місце тренера зосереджує засоби ведення власного контингенту спортсменів і взаємодії з турнірами. Його організовано як набір функціональних розділів, доступних із верхньої панелі кабінету: реєстр клубу, реєстрація на турнір, рахунки та оплата, активні заявки, рейтинг залу та публічне табло (Live Scoreboard). У заголовку кабінету відображаються відомості про тренера та його клуб, що задає контекст подальших дій і узгоджується з обмеженням видимості даних власним контингентом.

Основним розділом кабінету є реєстр спортсменів клубу, який подає підопічних набором карток із можливістю перемикання на компактне табличне подання та з підсумковим лічильником загальної кількості атлетів. Кожна картка містить

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

фотографію або ініціали спортсмена, його прізвище, ім'я та по батькові, кваліфікаційний рівень (розряд, ступінь кю чи дан, колір пояса), а також поточну вагу й вік. На картці згруповано швидкі дії над записом – перегляд динаміки ваги, перелік заявок спортсмена, формування перепустки (бейджа), редагування та видалення. Для роботи з численним складом клубу передбачено пошук за прізвищем і впорядкування записів за різними критеріями, а також групові операції: масовий імпорт анкетних даних із файлів CSV або Excel, експорт усіх бейджів одним пакетом та додавання нового спортсмена. Така організація дозволяє тренеру вести весь контингент в одному місці. Реєстр спортсменів у робочому місці тренера наведено на рис. 3.6.

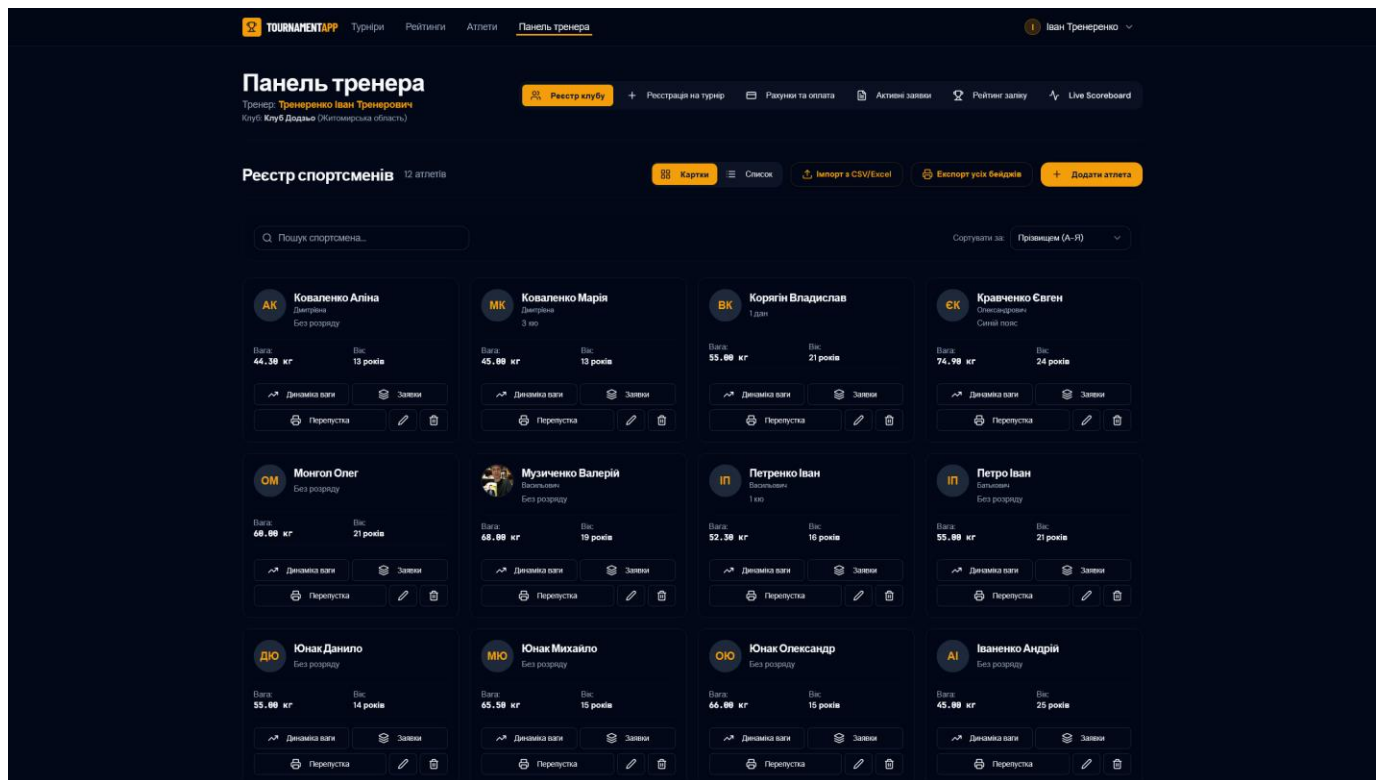


Рис. 3.6 – Робоче місце тренера: реєстр спортсменів клубу

Окремий сценарій робочого місця тренера включає оплату стартових внесків за участь спортсменів у змаганнях. Після подання заявок система формує рахунок і за командою тренера ініціює платіж через інтеграцію з сервісом Monobank за моделлю еквайрингу. Користувача перенаправляє на захищену сторінку платіжного шлюзу, де відображено суму до сплати, призначення платежу із зазначенням спортсменів та два способи оплати (сканування QR-коду застосунком monobank або ручне введення

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

реквізитів картки). Платежі виконуються без реального списання коштів у режимі тестового середовища (sandbox) зі справжньою сторінкою еквайрингу Monobank або в режимі повної імітації для автономної розробки. Сторінку оплати стартового внеску наведено на рис. 3.7.

Після підтвердження оплати застосунок синхронізує стан рахунку, і той набуває статусу «Сплачено». Деталі рахунку доступні тренерові в розділі «Рахунки та оплата» й містять унікальний ідентифікатор рахунку, суму платежу, дату створення, призначення (стартові внески чи комісія платформи) та повний перелік заявлених спортсменів із категоріями; цей перелік за потреби можна скопіювати або вивантажити окремим файлом для звітності. Деталі сплаченого рахунку наведено на рис. 3.8.

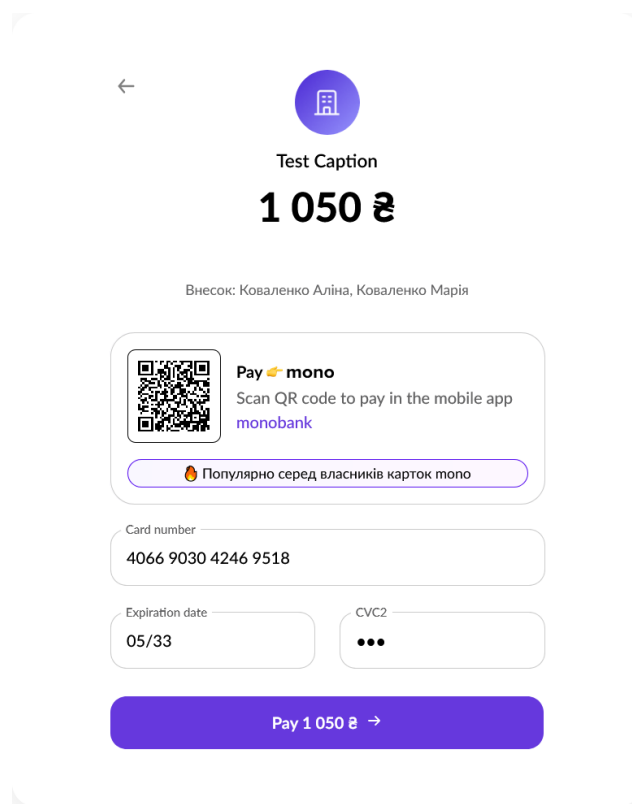


Рис. 3.7 – Оплата реєстраційного внеску (інтеграція з Monobank)

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

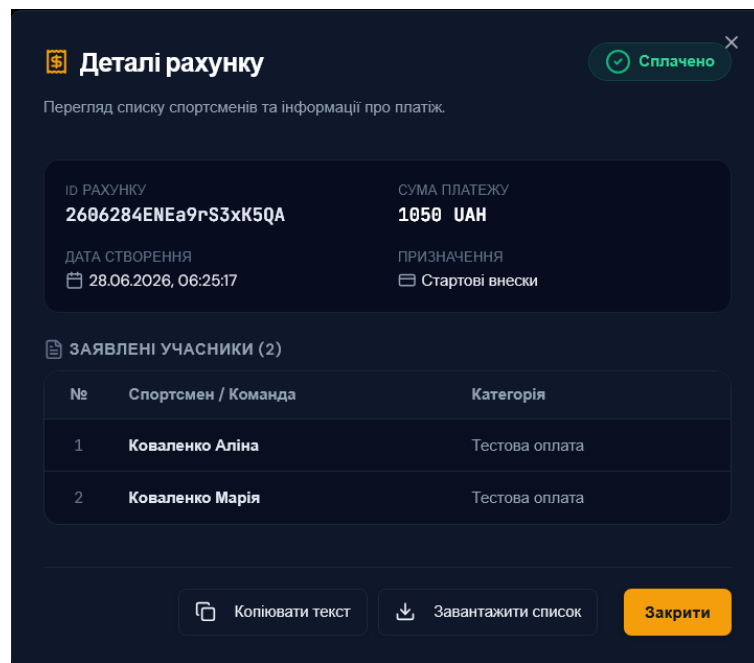


Рис. 3.8 – Результат успішної оплати

Робоче місце організатора охоплює повний життєвий цикл змагання – від його створення до підбиття фінансових підсумків. У заголовку сторінки турніру відображаються його назва, поточний стан (реєстрація, проведення, завершення), порядковий номер, дати проведення, вид єдиноборства та ознака відкритості реєстрації. Звідси організатор керує життєвим циклом змагання: відкриває та закриває реєстрацію, розпочинає і завершує турнір, відкриває публічне табло татамі, переходить до керування татамі й автоматичного розподілу категорій по них, запускає пакетну генерацію всіх турнірних сіток, а також редагує або вилучає турнір. Зміст сторінки структуровано вкладками «Інформація», «Категорії», «Учасники» та «Результати», що розмежовують конфігураційні, реєстраційні та підсумкові відомості.

Основною є вкладка категорій, на якій організатор формує вагово-вікові дивізіони; створення категорій пришвидшено інтелектуальним розбором їх текстових назв і додатково підтримано масовим імпортом. Кожна категорія подається карткою, що відображає її назву, запланований час і номер закріпленого татамі, стан реєстрації, основні параметри (стать, вагові та вікові обмеження), обраний формат турнірної

сітки (олімпійська система на вибування, подвійне вибування, колова чи швейцарська система – відповідно до алгоритмів з підрозділу 2.3), кількість зареєстрованих учасників, розмір стартового внеску та ознаку того, чи згенеровано для неї сітку. Передбачено пошук і впорядкування категорій, їх групування за татамі, перемикання між картковим і списковим поданням та пакетні операції друку й вивантаження всіх сіток. Зокрема, інтерфейс картки категорії дозволяє організатору миттєво оцінити рівень заповненості дивізіону та ухвалити рішення про необхідність його об'єднання з іншими групами в разі недобору учасників. Для зручності навігації масивами даних передбачено візуальне кодування: статуси реєстрації та наявність згенерованої сітки позначаються відповідними індикаторами, що пришвидшує аудит турніру. Екран керування турніром і його категоріями наведено на рис. 3.9.

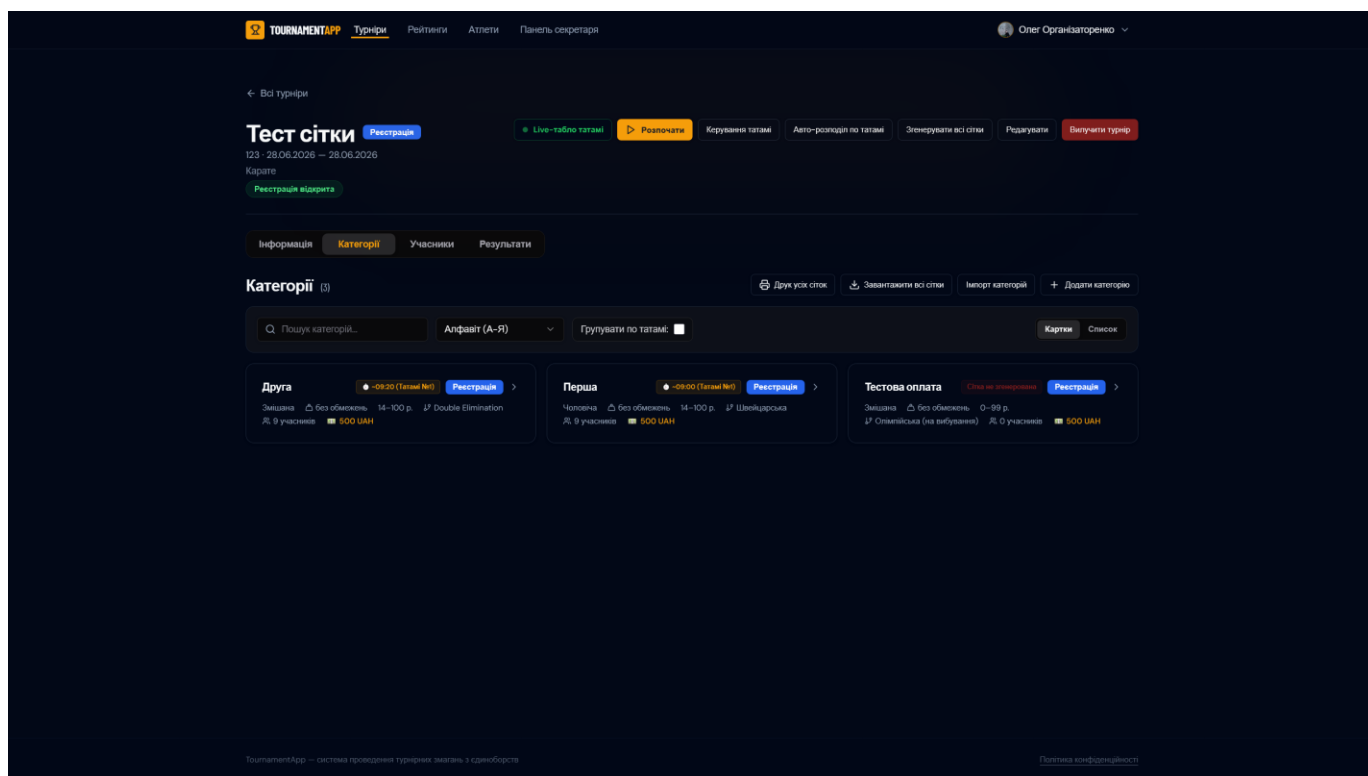


Рис. 3.9 – Робоче місце організатора: керування турніром і категоріями

Адміністративно-фінансовий контур робочого місця організатора подано окремим поданням, що зводить статистику змагання та фінансовий звіт: кількість учасників, стан оплат, нарахування платформі та доступну до виведення суму. Близькими за призначенням є екран керування татамі, на якому майданчики турніру

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		91

закріплюються за суддями, та робоче місце секретаріату, що обслуговує зважування, контроль присутності й групове оновлення стану заявок у день змагань. Фінансовий дашборд (панель огляду) розділено на логічні блоки: у верхній частині виведено ключові агреговані метрики (загальну суму очікуваних надходжень, фактично сплачені кошти через інтегрований шлюз та готівкові розрахунки), а нижче розташовано блок взаєморозрахунків із платформою. Така деталізація забезпечує фінансову прозорість турніру та дозволяє організатору контролювати рух коштів без залучення сторонніх облікових систем. Фінансово-статистичне подання турніру наведено на рис. 3.10.

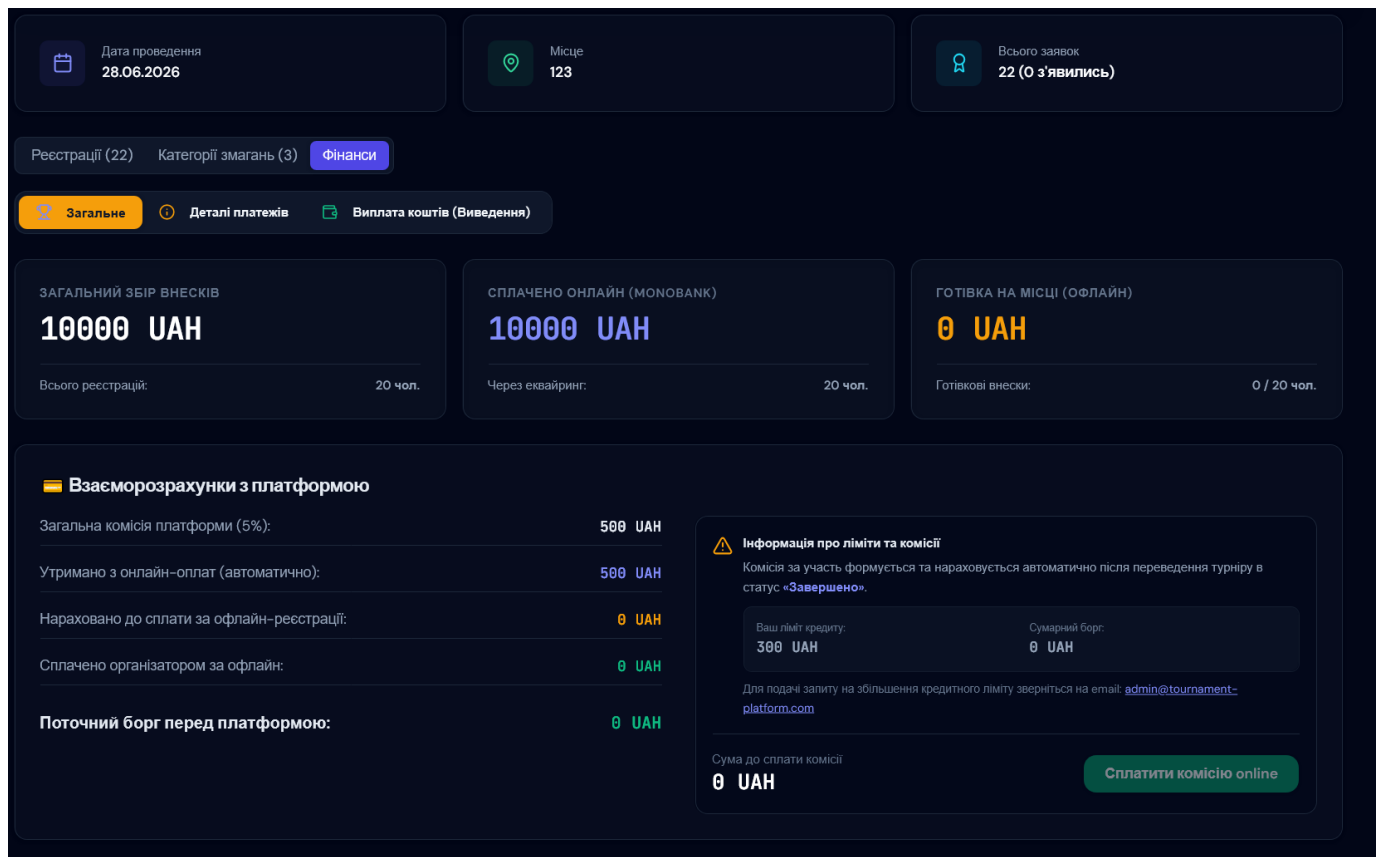


Рис. 3.10 – Робоче місце організатора: статистика та фінансовий звіт турніру

Суддівство поєдинків здійснюється через панель оператора татамі. Це найвимогливіший до латентності екран системи, розрахований на роботу пальцями з планшета біля майданчика, з великими елементами керування. Відповідно до клієнтської реалізації патерну «Стратегія» з підрозділу 2.5, єдина сторінка оператора

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

визначає підключений до категорії набір правил і відображає спеціалізовану панель тієї дисципліни, у якій проводиться поєдинок. Спільними для всіх дисциплін є таймер поєдинку із серверним джерелом істини та керуванням часом, засоби нарахування та скасування результативних дій, виправлення помилкових нарахувань і перезапуск поєдинку, оголошення переможця та збереження підсумків. Склад балів, штрафів і допоміжних показників визначається конкретними правилами. Кожна дія негайно відображається на публічному табло через канал реального часу. Інтерфейс цієї панелі не є жорстко закодованим. Він генерується динамічно на основі JSON-конфігурації правил (Ruleset), отриманої від сервера. Натискання кнопок оцінок використовує React-хук useOptimistic для миттєвого оновлення локального стану (Optimistic UI) ще до підтвердження транзакції базою даних. Панель оператора для дисципліни карате (поєдинків «куміте») з таймером і нарахуванням балів наведено на рис. 3.11.

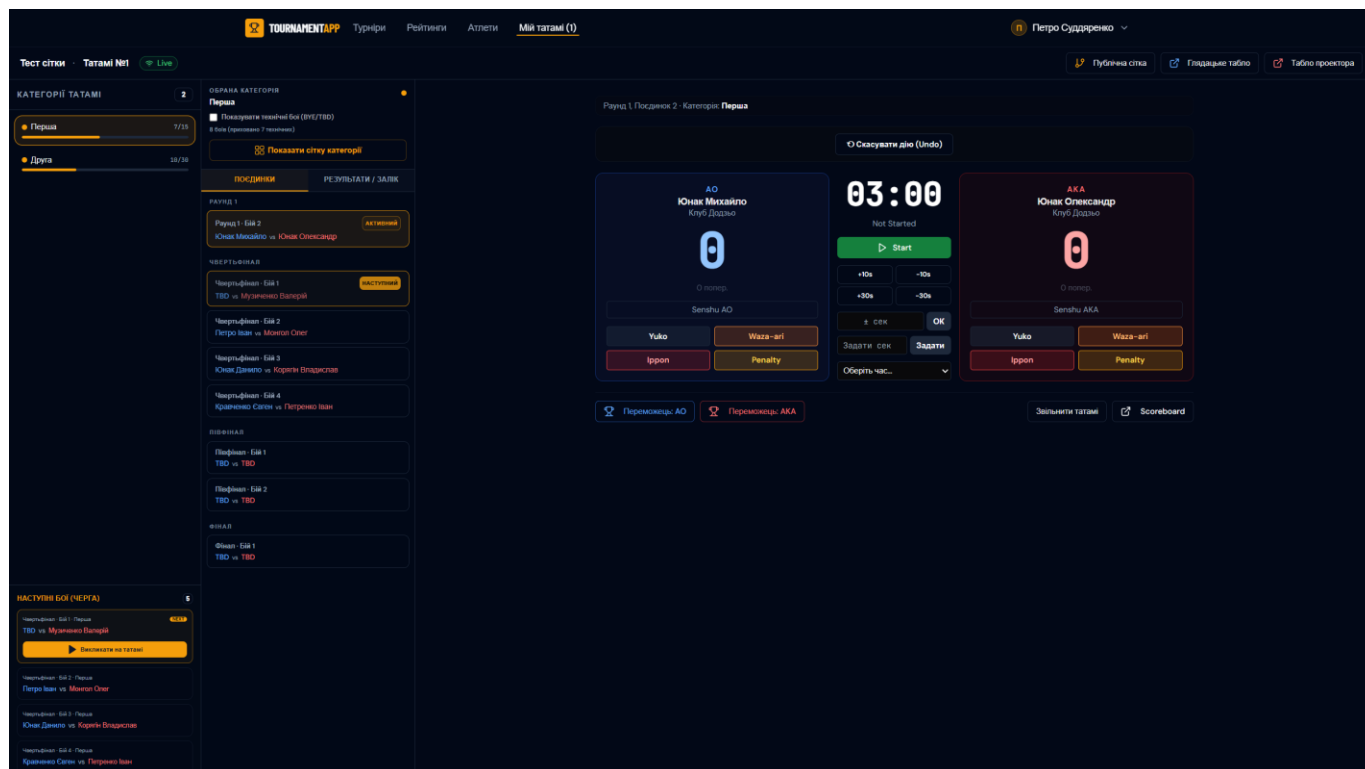


Рис. 3.11 – Панель оператора: поєдинки карате

Для дисципліни тхеквондо панель оператора відображає характерну для неї структуру нарахування балів і поділ поєдинку на раунди, що демонструє адаптивність єдиного інтерфейсу оператора до різних систем суддівства. Тут клієнтський

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

застосунок підлаштовується під логіку багатораундової машини станів. Замість єдиного лічильника компонент відображає агреговані результати виграних раундів, а кнопки прив'язані до специфічних дій (удари в корпус, голову та дії з розворотом). Усі події передаються на сервер через REST API з відповідними ідентифікаторами корисного навантаження (payload) для валідації. Відповідний екран наведено на рис. 3.12.

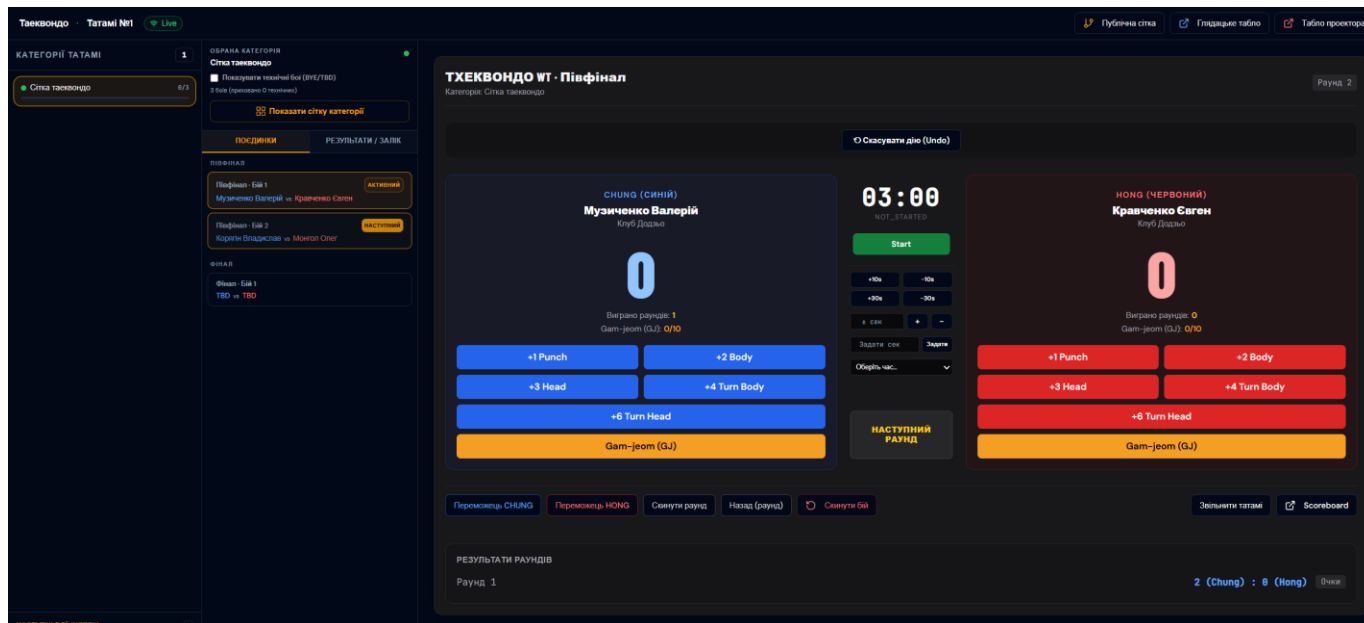


Рис. 3.12 – Панель оператора: поєдинки тхеквондо

Окремим випадком є обслуговування командних зустрічей та дисципліни дзюдо з її специфічною системою оцінок (чиста перемога «іппон», бал «вадза-арі») і фіксацією покарань. Для дзюдо інтерфейс приховує лічильники дрібних балів і залишає лише критичні оцінки та систему покарань («шідо»), накопичення яких автоматично ініціює дострокове завершення. У командному форматі над панеллю індивідуального бою додатково рендериться компонент-диспетчер. Він відстежує загальний рахунок перемог у зустрічі та керує чергою виходу спортсменів (суб-боями) згідно з рекурсивним зв'язком `parent_team_match_id` бази даних. Панель оператора для такого сценарію наведено на рис. 3.13 та 3.14.

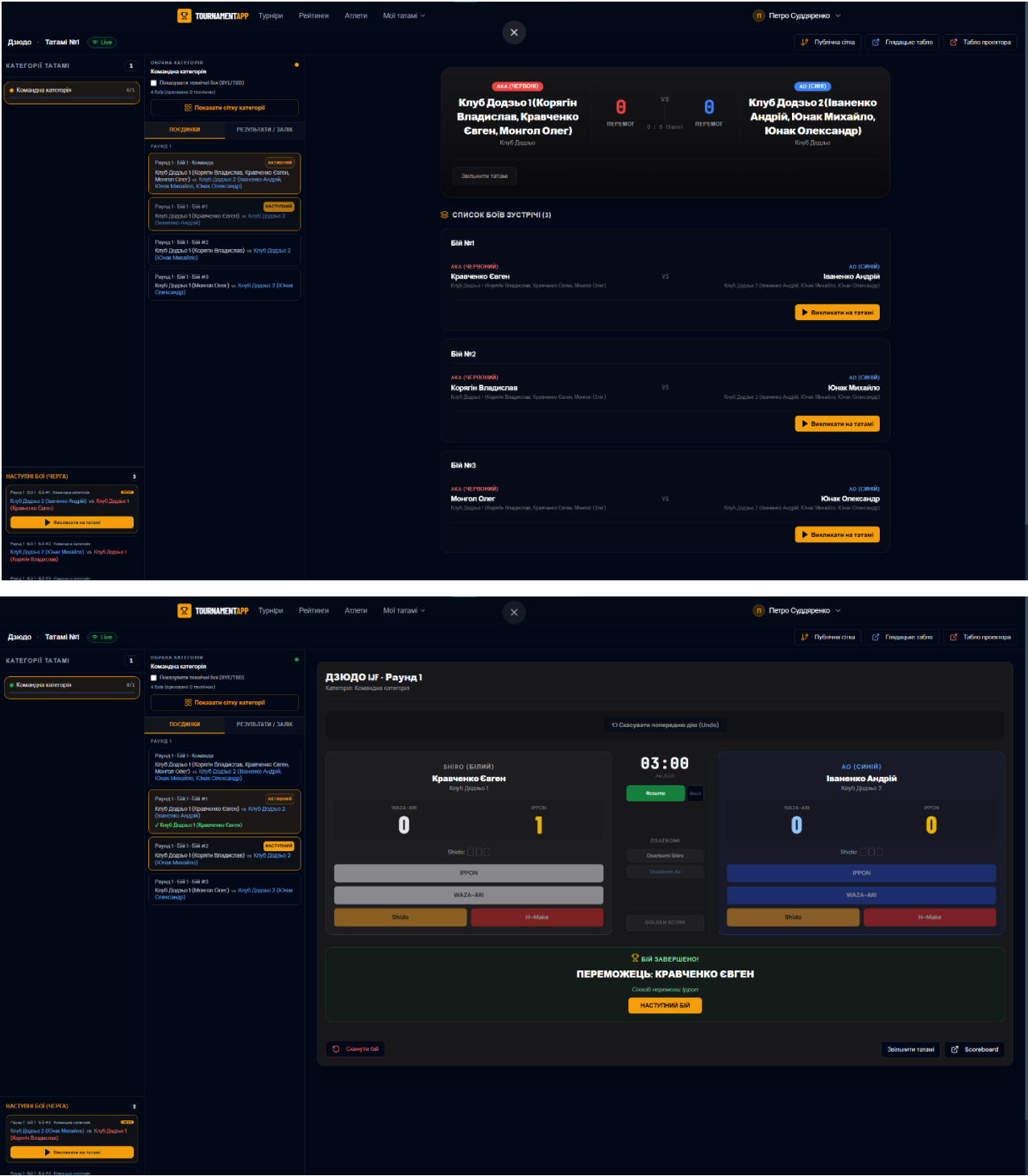


Рис. 3.13 – 3.14 – Панель оператора: командна зустріч (дзюдо)

Результати жеребкування й перебіг змагання відображаються у вигляді інтерактивної турнірної сітки. Компонент візуалізації підтримує всі передбачені формати: пряме та подвійне вибування, а також коловий формат, який подається турнірною таблицею. Кожен поєдинок зображується картою з учасниками та

рахунком, а сітка допускає горизонтальне прокручування для змагань із великою кількістю раундів. Рендеринг дерева вибування виконується на клієнті шляхом обходу списку суміжності (next_match_id), отриманого від сервера. Для плавного масштабування великих сіток застосовуються оптимізовані SVG-елементи. Обчислення координат вузлів кешується, що запобігає зайвим перемальовуванням DOM-дерева під час оновлення рахунку в окремих матчах. Турнірну сітку категорії наведено на рис. 3.15.

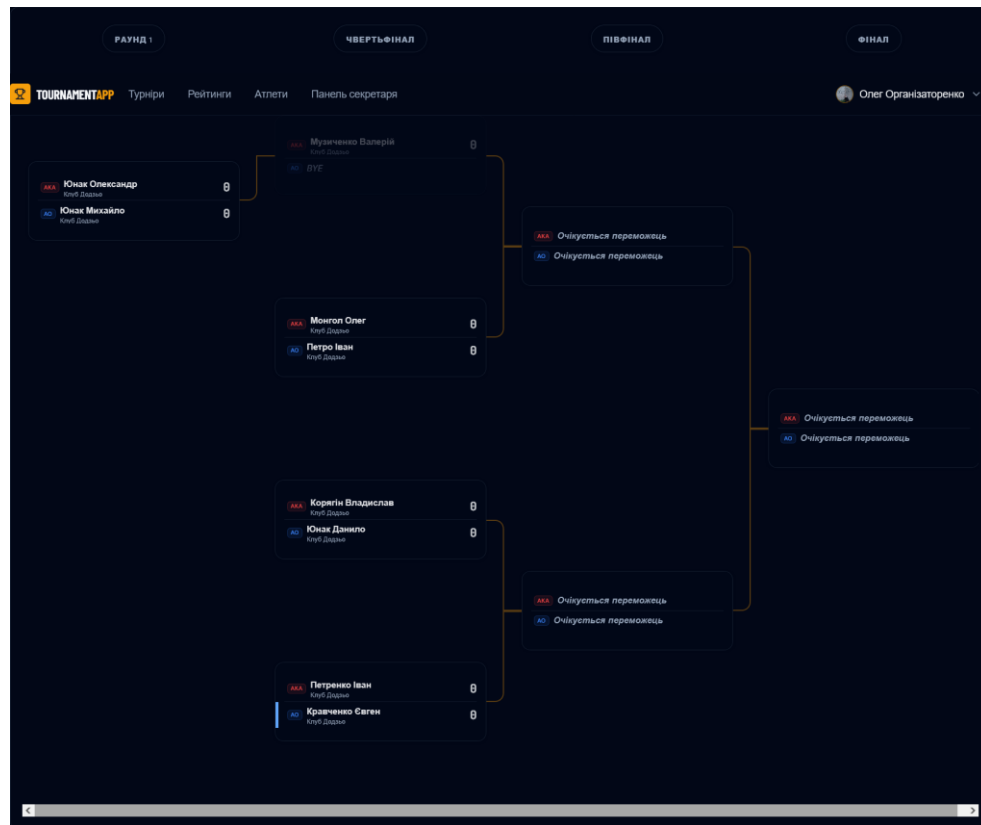


Рис. 3.15 – Візуалізація турнірної сітки категорії

Для трансляції перебігу поєдинків глядачам і учасникам передбачено окреме інформаційне табло, яке відкривається за власною адресою без елементів навігації застосунку та призначене для виведення на великий екран у залі. Табло відображає поточний стан поєдинку на обраному татамі – учасників, рахунок, час і службові показники – й оновлюється в режимі реального часу через WebSocket-з'єднання синхронно з діями суддівської бригади. Інформаційне табло татамі наведено на рис. 3.16.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		96

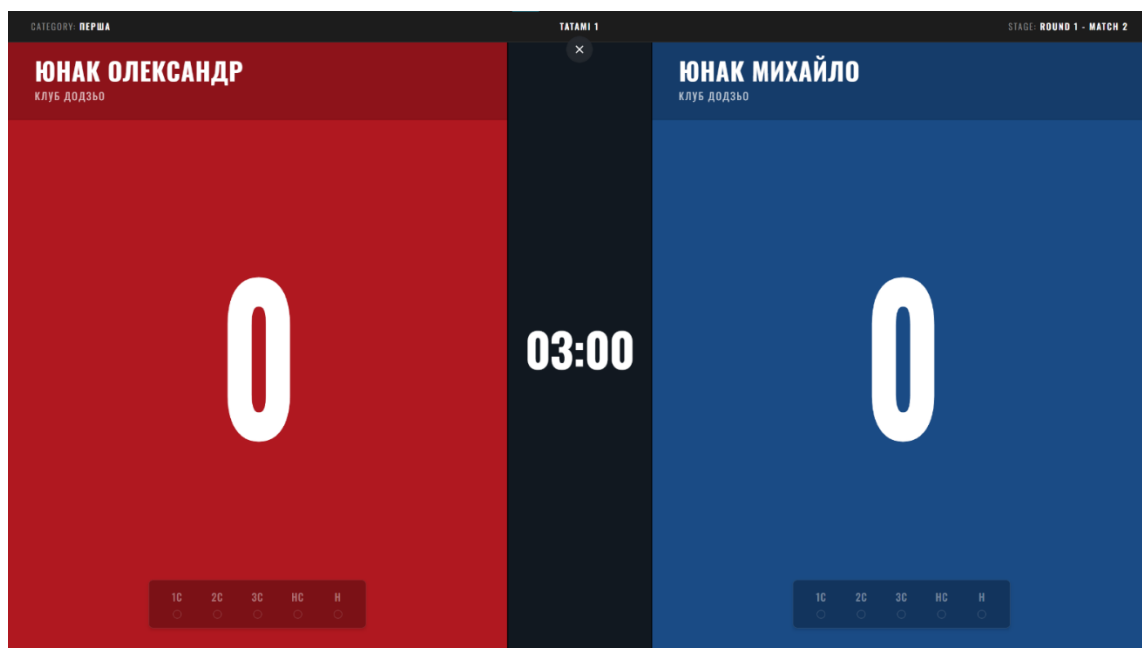


Рис. 3.16 – Інформаційне табло татамі

Адміністрування системи на рівні даних здійснюється через вбудовану панель керування Django, оформлену темою `django-unfold`, і доступне привілейованому обліковому запису головного адміністратора. Ліву навігаційну область згруповано за функціональними областями, що відповідають модульній структурі системи: «Управління доступом» (користувачі, заявки на ролі, клуби, бани), «Спортсмени» (профілі, команди, історія зважувань), «Турніри» (турніри, категорії, реєстрації), «Проведення» (татамі, матчі, події матчів) та «Фінанси та білінг» (рахунки, транзакції, виплати). Для кожної сутності доступні перегляд, пошук, фільтрування, створення та масове редагування записів, що робить панель інструментом прямого керування даними на доповнення до прикладних робочих місць.

Окрім засобів керування записами, панель містить власну головну сторінку «Панель управління TournamentSystem», яка подає метрики та стан платформи в режимі реального часу й виконує роль операційного монітора. У верхньому ряді відображено експлуатаційні показники з переходами до відповідних розділів: кількість активних турнірів, кількість запитів на верифікацію ролей, що очікують розгляду, та кількість активних поєдинків. Окремий блок показує навантаження й ресурси сервера – використання процесора та оперативної пам'яті, а також кількість

		Муравйовський В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		97

активних з'єднань із базою PostgreSQL. Блок загального статусу підтверджує, що сервер ASGI (Daphne), черга повідомлень Redis і база PostgreSQL запущені в середовищі Docker, а WebSocket-з'єднання з табло та клієнтами працюють у фоновому режимі, і наводить агреговані підсумки – загальну кількість користувачів і турнірів. Адміністративну панель із головною інформаційною сторінкою наведено на рис. 3.17.

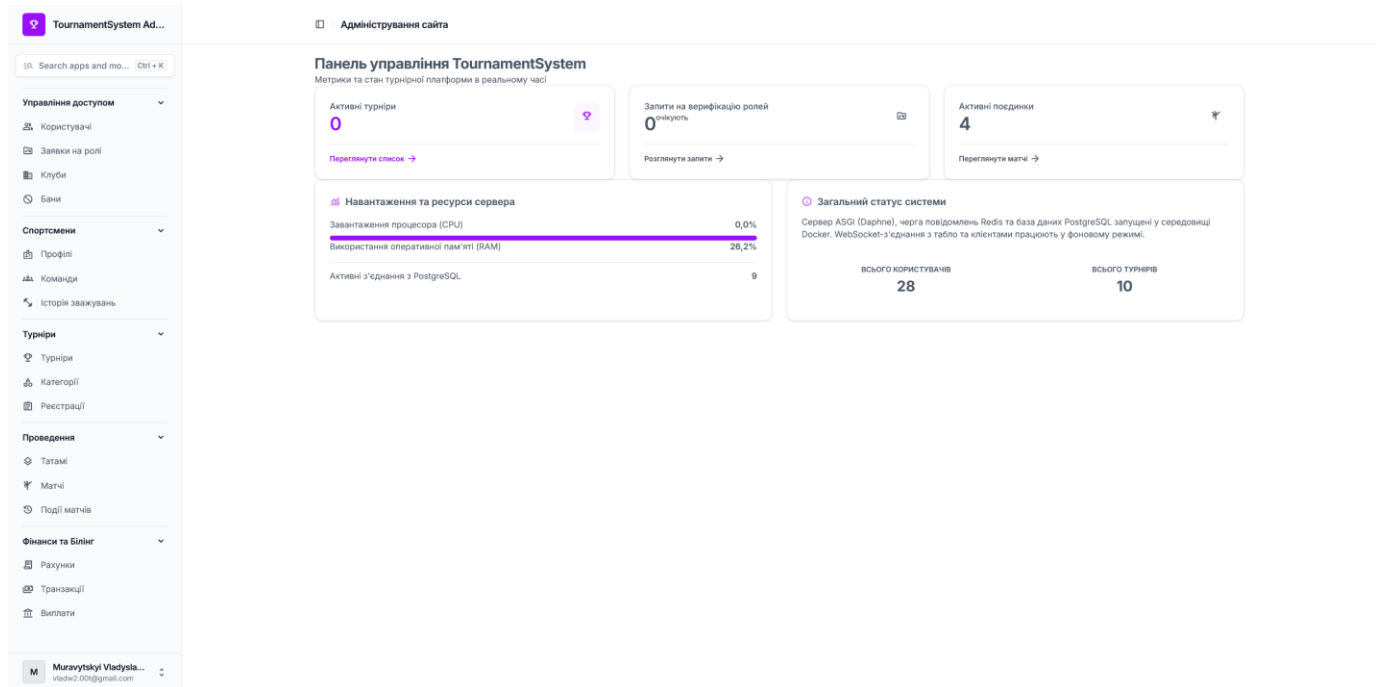


Рис. 3.17 – Адміністративна панель системи (django-unfold)

3.3. Тестування роботи програмного комплексу

Тестування підтверджує відповідність розробленого програмного комплексу проектним вимогам. Для перевірки складної предметної логіки (насамперед алгоритмів формування турнірних сіток і підбиття підсумків) створено набір автоматизованих тестів. Вони виконуються під час кожного внесення змін у складі конвеєра безперервної інтеграції.

Тестування базується на принципі піраміди: велика кількість швидких низькорівневих перевірок переважає над повільними високорівневими. Основу піраміди утворюють модульні (unit) тести, які ізольовано перевіряють окремі функції, моделі та бізнес-правила. Наступний рівень містить інтеграційні тести для перевірки

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

взаємодії модулів між собою та з базою даних (генерація сіток, переходи станів, прикладний інтерфейс). Вершиною є наскрізні (E2E) тести, які відтворюють дії користувача в реальному браузері. Серверна частина тестується фреймворком pytest, а клієнтські та наскрізні сценарії перевіряються засобами Playwright. Загалом система містить 405 серверних тестів і 105 наскрізних сценаріїв. Інструмент coverage вимірює покриття коду, після чого агрегований звіт публікується в сервісах Codecov і SonarCloud. Загальне покриття серверного коду тестами становить 91,6 %. Розподіл тестів за рівнями піраміди наведено на рис. 3.18.

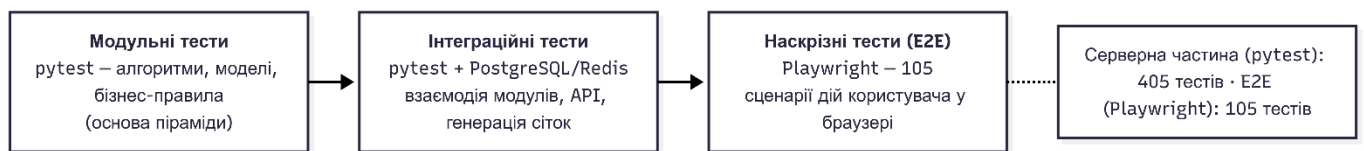


Рис. 3.18 – Розподіл автоматизованих тестів за рівнями

Розподіл серверних і наскрізних тестів за підсистемами, що відповідають модульній структурі системи, наведено в таблиці 3.2. Найбільше тестів припадає на підсистеми зі складнішою логікою – управління турнірами, проведення поєдинків, рушій правил суддівства та генерацію сіток.

Таблиця 3.2

Розподіл автоматизованих тестів за підсистемами

Підсистема (модуль)	Призначення	Тип тестів	Кількість
Облікові записи та доступ (accounts)	автентифікація, рольова модель, верифікація	модульні, інтеграційні	30
Спортсмени та команди (athletes)	картки спортсменів, команди, зважування	модульні, інтеграційні	22
Турніри, категорії, реєстрації (tournaments)	життєвий цикл змагання, дивізіони, підсумки	модульні, інтеграційні	86
Генерація турнірних сіток (brackets)	алгоритми сіток усіх форматів	модульні, інтеграційні	31
Поєдинки та події (matches)	хід поєдинку, фіксація переможців, журнал подій	модульні, інтеграційні	79

Продовження таблиці 3.2

Підсистема (модуль)	Призначення	Тип тестів	Кількість
Двигун правил суддівства (rulesets)	стратегії дисциплін, бали, штрафи	модульні	60
Татамі та диспетчеризація (tatamis)	черги поєдинків, призначення, таймер	модульні, інтеграційні	59
Фінанси та білінг (billing)	рахунки, платежі, виплати	модульні, інтеграційні	32
Спільні компоненти (common)	службові проби стану, трансляція подій	модульні	6
Серверна частина разом		pytest	405
Наскрізнi сценарії інтерфейсу	автентифікація, кабінети, оператор, сітка, табло	Playwright (E2E)	105
Усього			510

Алгоритми формування турнірних сіток і підбиття підсумків проходять верифікацію на наборах вхідних даних із заздалегідь відомим і аналітично обчисленим результатом. Це дозволяє однозначно встановити коректність математичної реалізації. Узагальнені результати наведено в таблиці 3.3.

Таблиця 3.3

Верифікація реалізованих алгоритмів

Алгоритм (підрозділ 2.3)	Вхідні дані	Очікуваний результат	Статус
Обчислення розміру сітки	5 / 17 учасників	сітка на 8 / 32 слоти	Пройдено
Кількість автоматичних перемог (BYE)	5 / 9 учасників	3 / 7 BYE	Пройдено
Стандартний розподіл посівів	сітка на 8 слотів	послідовність [1, 8, 4, 5, 2, 7, 3, 6]	Пройдено

Продовження таблиці 3.3

Алгоритм (підрозділ 2.3)	Вхідні дані	Очікуваний результат	Статус
Розведення одноклубників	4 уч., двоє з одного клубу поряд	відсутність одноклубників у першому раунді	Пройдено
Олімпійська система (на вибування)	8 учасників	7 матчів (4+2+1), один фінал	Пройдено
Олімпійська система з BYE	5 учасників	8 слотів, 7 матчів, 3 walkover	Пройдено
Колова система	6 учасників	15 матчів, кожна пара рівно один раз	Пройдено
Швейцарська система, перший тур	6 учасників	3 пари за принципом «сильніший–слабший»	Пройдено
Тай-брейки швейцарської системи	рівні очки в усіх учасників	впорядкування за Бухгольцем, різницею балів та особистою зустріччю	Пройдено
Подвійне вибування (короткий гранд-фінал)	переможець нижньої сітки виграє гранд-фінал	без створення «скидання сітки»	Пройдено
Втішна сітка (repechage)	8 учасників	місця 1, 2, 3, 3 (дві бронзові нагороди)	Пройдено

Три найскладніші алгоритми варто розглянути докладніше. Перевірку розведення одноклубників побудовано на сценарії, у якому до сітки з чотирьох учасників потрапляють два спортсмени одного клубу на суміжних позиціях, тобто з гарантованою зустріччю вже в першому раунді. Після застосування алгоритму перерозподілу тест підтверджує, що в жодній парі першого раунду не залишилося представників одного клубу. Окремим тестом перевірено поведінку у виродженому випадку, коли всі учасники належать до одного клубу: алгоритм не зациклюється, а завершується, повертаючи повний склад без спроби неможливого розведення.

Підбиття підсумків у швейцарській системі з додатковими показниками перевірено тестом, що відтворює змагання шести учасників, сконструйоване так, що всі набирають однакову кількість очок та однаковий коефіцієнт Бухгольца. За таких умов перевіряється послідовне застосування додаткових критеріїв упорядкування (різниці набраних і пропущених балів, а далі результату особистої зустрічі). Оскільки один з учасників у безпосередньому поєдинку переміг іншого з рівними показниками, він має посісти вище місце незалежно від інших коефіцієнтів. Тест підтверджує, що підсумкове впорядкування відповідає очікуваному, чим засвідчується правильність реалізації багаторівневого механізму тай-брейків.

Перевірку втішних сіток (perchage) виконано для характерного для карате формату з двома бронзовими нагородами. Тест відтворює змагання восьми учасників, автоматичну генерацію втішних поєдинків для тих, хто програв майбутнім фіналістам, та підсумковий розподіл місць. Очікуваним і підтвердженим результатом є присвоєння місць 1, 2 та двох третіх місць, що відповідає правилам нагородження дисципліни.

Наскрізний (E2E) тест підтверджує працездатність системи як єдиного цілого та відтворює повний життєвий цикл змагання в браузері. Розроблений сценарій «золотого шляху» проводить турнір через усі ключові стадії. Спочатку від імені організатора створюється турнір із відкритою реєстрацією та категорія. Далі від імені тренера реєструються спортсмени, рахунок позначається сплаченим і генерується турнірна сітка. Після цього сценарій тестує взаємодію через інтерфейс. На сторінці турніру організатор натискає кнопку запуску, тест перевіряє появу підтвердження про початок змагання і додає татамі. На завершальному етапі сценарій відвідує три екрани й перевіряє їхній стан. Згенерована сітка відображає раунди, інформаційна панель дня показує нове татамі як вільне з посиланням на панель оператора, а публічне табло татамі переходить у стан очікування поєдинку. Цей тест гарантує узгоджену взаємодію підсистем генерації сіток, інформаційної панелі дня та інформаційного табло.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		102

Усі описані тести виконуються автоматично в конвеєрі безперервної інтеграції під час кожного внесення змін до репозиторію, а їх успішне проходження є обов'язковою умовою для злиття змін в основну гілку та подальшого розгортання. Успішне завершення всіх етапів конвеєра (статичного аналізу коду (лінтингу), модульних та інтеграційних тестів, наскрізного тестування й контрольного збирання образів) наведено на рис. 3.19.

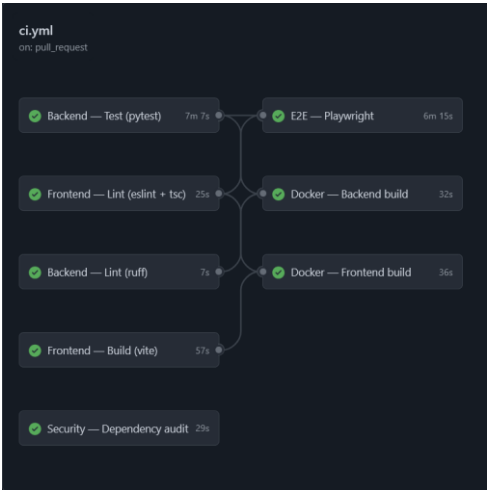


Рис. 3.19 – Успішне проходження конвеєра безперервної інтеграції (GitHub Actions)

Окрім факту проходження тестів, якість коду контролюється кількісно. Звіт сервісу SonarCloud підтверджує проходження порогу якості (Quality Gate), наводить досягнутий рівень покриття коду тестами та показники надійності, безпеки й придатності до супроводу. Підсумковий звіт про якість і покриття коду наведено на рис. 3.20.

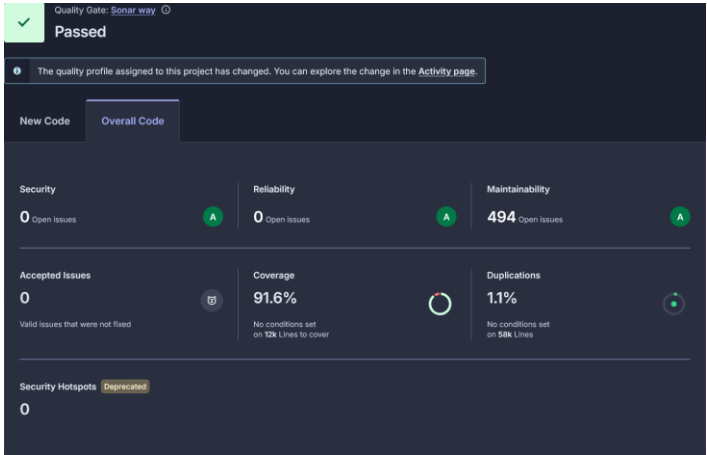


Рис. 3.20 – Звіт про якість і покриття коду (SonarCloud)

Проходження понад 500 автоматизованих тестів, верифікація ключових алгоритмів та позитивний висновок системи статичного аналізу підтверджують коректність реалізованої предметної логіки програмного комплексу.

3.4. Дослідження продуктивності системи

Експериментальна перевірка проєктних показників часу відгуку (з таблиці 2.1) спирається на автоматизовані заміри. Програма вимірювань охоплює чотири блоки: алгоритмічний час генерації турнірних сіток, час відгуку REST-ендпоінтів, латентність доставлення подій через WebSocket та поведінку системи під зростаючим навантаженням.

Вимірювання проводилися на комп'ютері з процесором AMD Ryzen 5 5600H (12 потоків) та 16 ГБ оперативної пам'яті під керуванням Windows 10. Серверна частина працювала у середовищі Docker Compose з вимкненим режимом налагодження (DJANGO_DEBUG=False). Застосунок обслуговував ASGI-сервер Uvicorn із чотирма робочими процесами. База даних PostgreSQL 16 та брокер Redis 7 функціонували в суміжних контейнерах на тому самому сервері, а запити надсилалися на localhost без використання зворотного проксі. Ця конфігурація відтворює параметри промислового режиму, проте працює на єдиному фізичному вузлі без горизонтального масштабування. Тому отримані результати навантаження є базовою (нижньою) оцінкою порівняно з повноцінним розгортанням. Слід зазначити, що робочий контейнер серверної частини запускає сервер Daphne без зовнішнього супервізора процесів і працює як єдиний процес, оскільки Daphne, на відміну від Uvicorn, не підтримує запуск кількох робочих процесів прапорцем; отже, наведені нижче показники пропускної здатності, отримані на чотирьох процесах Uvicorn, відображають потенціал багатопотокової конфігурації, а не поточну ємність промислового розгортання.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		104

Час генерації сіток замірявся локально на чистих функціях модуля жеребкування без звернень до бази даних. Кожен формат і розмір сітки пройшов серію повторів для збору медіани та 95-го процентиля. Результати наведено в таблиці 3.4.

Таблиця 3.4

Алгоритмічний час генерації турнірних сіток (медіана / p95, мс)

Формат	8 уч.	16 уч.	32 уч.	64 уч.	128 уч.
Олімпійська система	0,008	0,013	0,026	0,056	0,143
Подвійне вибування	0,019	0,030	0,051	0,098	0,219
Колова система	0,004	0,013	0,045	0,164	0,852
Швейцарська (перший тур)	0,005	0,008	0,016	0,039	0,107
Швейцарське парування (повне)	0,056	0,082	11,9	14,9	36,5
Розведення одноклубників (найгірший випадок)	0,198	0,270	0,427	0,729	1,35

Для олімпійської системи, подвійного вибування та першого туру швейцарської системи час зростає лінійно з розміром сітки. Навіть для 128 учасників він не перевищує частки мілісекунди. Час повного швейцарського парування залежить не лише від кількості учасників, а й від щільності обмежень у вхідних даних. Оскільки парування використовує метод гілок і меж із поверненням, за умови щільного графа зіграних пар та наявності клубних конфліктів алгоритм перебирає тисячі варіантів перед знаходженням допустимого рішення або досягненням ліміту ітерацій. Цей фактор пояснює розкид часу між суміжними розмірами сітки. При цьому найбільший зафіксований час (37 мс на 128 учасників) залишається на два порядки меншим за проєктну вимогу. Евристика розведення одноклубників у найгіршому випадку додає до розрахунку не більше 1,4 мс.

Час відгуку REST-ендпоінтів перевірявся у двох режимах: ізольовано та під фоновим навантаженням. Зіставлення з проєктними показниками таблиці 2.1 наведено в таблиці 3.5.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		105

Час відгуку REST-ендпоінтів проти проєктних показників (p95)

Операція	Проєктний показник	Ізольовано	Статус
Автентифікація та завантаження профілю	до 1000 мс	68 мс	відповідає
Завантаження турнірної сітки категорії	до 1000 мс	953 мс	відповідає
Автоматична генерація сітки	до 2000 мс	1099 мс	відповідає
Фіксація результату та просування переможця	до 500 мс	681 мс	перевищено

Завантаження сітки наблизилося до межі проєктного показника (953 мс при цілі 1000 мс). Початкові заміри демонстрували понад 1200 мс через проблему надлишкових запитів до бази даних під час серіалізації (N+1 query problem). Для сітки на 128 учасників компонент перетворення даних генерував понад тисячу окремих SQL-запитів через незавантажені зв'язки. Застосування оптимізованої вибірки пов'язаних об'єктів через select_related та використання кешу попередньої вибірки зменшили час відгуку до 953 мс.

Фіксація результату з просуванням переможця перевищує проєктний показник (681 мс проти цільових 500 мс). Операція виконується однією атомарною транзакцією, яка поєднує чотири складові: оновлення поточного поєдинку, просування переможця в наступний слот сітки, запис події до журналу та обробку технічних перемог за порожніми проходами. Згідно з результатами профілювання, сама транзакція займає близько 150-180 мс. Фіксація WAL-журналу PostgreSQL через міст віртуалізації Docker додає 80-120 мс. Публікація двох подій у Redis у межах того самого запиту коштує ще 120-150 мс. Серіалізація відповіді з вкладеними даними спортсменів, клубів і тренерів додає 150-180 мс. Перевищення показника зосереджено в накладних витратах середовища (фіксація журналу через віртуалізований диск, синхронна публікація подій), а не в доменній логіці. Напрямою подальшої оптимізації є винесення публікації подій у фонові задачі та спрощення процесу формування (серіалізації) відповіді.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		106

Латентність доставлення подій через WebSocket обчислюється як проміжок часу від надсилання суддівської дії до отримання трансляції підписником табло. За результатами вимірювань медіана становить 119 мс, а 95-й процентиль дорівнює 184 мс. Це відповідає проєктним показникам для нарахування бала (до 200 мс) та оновлення табло (до 300 мс). Обидва показники збігаються, оскільки нарахування бала й оновлення табло є однією подією. Після фіксації зміни сервер публікує єдиний пакет у групу категорії, на яку підписані суддівське і публічне табло, тому клієнти отримують дані одночасно.

Поведінка системи під навантаженням перевірялася зростальним профілем зі змішаним складом клієнтів. Сімдесят відсотків клієнтів надсилали HTTP-запити на читання (списки турнірів, категорій, завантаження сітки) із паузою 0,5-1,5 с між запитами, а тридцять відсотків утримували постійне WebSocket-з'єднання з табло. Результати наведено в таблиці 3.6.

Таблиця 3.6

Поведінка системи під зростальним навантаженням

Одночасні користувачі	Пропускна здатність, запит/с	Частка помилок HTTP	Медіана HTTP, мс	p95 HTTP, мс	Активні WS-з'єднання
10	4,8	0 %	142	1698	3
50	7,8	30 %	1063	1885	15

За десяти одночасних користувачів система обслуговує запити без помилок, а WebSocket-з'єднання тримаються стабільно. За п'ятдесяти користувачів з'являється 30 % помилок. Усі вони спричинені перевищенням часу очікування на стороні клієнта (встановлений поріг – 5 секунд). Це відбувається через повну утилізацію чотирьох робочих процесів Unicorn паралельним завантаженням важких структур сіток (до 300 КБ для 128 учасників), внаслідок чого решта запитів накопичується в черзі. WebSocket-з'єднання при цьому не дають помилок на жодному рівні навантаження. Зафіксована межа стосується конкретної конфігурації локального

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		107

тестування (один сервер, чотири робочі процеси, середовище без зворотного проксі й кешування). Горизонтальне масштабування застосункового шару під спільними базою та брокером передбачено архітектурою модульного моноліту, що дозволить підвищити цю межу в промисловому середовищі. Заміри підтверджують відповідність алгоритмічного ядра генерації сіток і доставлення подій проєктним показникам. Фіксація результату перевищує цільові 500 мс через накладні витрати тестового середовища, для чого визначено відповідні архітектурні оптимізації. Навантажувальний тест окреслив межу поточної конфігурації та підтвердив стабільність WebSocket-шару.

Висновки до третього розділу

Розгортання системи реалізовано як відтворюваний процес. Усі компоненти запаковано в контейнери Docker з використанням багатоетапної збірки. Конфігурація зчитується зі змінних оточення. Доставлення нових версій виконує конвеєр безперервної інтеграції на основі GitHub Actions. Платформа успішно працює на віртуальному сервері Azure. Службові проби життєздатності та готовності підтверджують коректність розгортання. Залежності між контейнерами задають строгий порядок підняття компонентів.

Інтерфейс системи розподілено на робочі місця для різних категорій користувачів. Єдиний клієнтський застосунок адаптує доступний набір функцій відповідно до ролі авторизованого користувача. Панель оператора динамічно підлаштовується під дисципліну поточного поєдинку завдяки реалізації патерну стратегії на боці клієнта.

Якість програмного комплексу підтверджує набір із 510 автоматизованих тестів. Ця кількість об'єднує 405 серверних і 105 наскрізних перевірок. Загальне покриття серверного коду тестами становить 91,6 відсотка. Алгоритми генерації сіток і підбиття підсумків успішно проходять перевірку на наборах вхідних даних з аналітично обчисленим результатом. Це охоплює логіку розведення одноклубників, обчислення

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		108

додаткових показників швейцарської системи та побудову формату з двома бронзовими нагородами. Наскрізний сценарій відтворює повний цикл турніру через усі стадії в браузері. Усі тести інтегровані в конвеєр доставлення, а статичний аналіз коду в сервісі SonarCloud підтверджує подолання порогу якості.

Експериментальні заміри засвідчують відповідність продуктивності цільовим проектним показникам. Алгоритмічне ядро генерації сіток і модуль доставлення подій через протокол WebSocket укладаються в установлені вимоги зі значним запасом. Генерація сітки на 128 учасників виконується за частки мілісекунди для більшості форматів. Латентність трансляції подій становить 184 мілісекунди при допустимій нормі до 300 мілісекунд. Усунення надлишкових запитів до бази даних знизило час завантаження сітки категорії до 953 мілісекунд при ліміті в одну секунду. Фіксація результату з просуванням переможця становить 681 мілісекунду і незначно перевищує цільовий показник у 500 мілісекунд. Згідно з результатами профілювання це перевищення зосереджено в накладних витратах тестового середовища під час запису журналу бази через віртуалізований диск та при синхронній публікації подій, а не в самій доменній логіці. Отримані дані визначають напрям подальшої архітектурної оптимізації. Навантажувальний тест окреслив межу поточної конфігурації. Зафіксовано стабільну роботу до 10 одночасних користувачів без помилок та випадки перевищення часу очікування при 50 активних користувачах. Водночас тест підтвердив абсолютну стабільність комунікаційного шару реального часу на всіх рівнях навантаження. Зібрані метрики створюють кількісну основу для оцінювання системи та визначають пріоритети доопрацювання продукту перед промисловим використанням.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		109

ВИСНОВКИ

Мету кваліфікаційної роботи, а саме розроблення вебсервісу для управління, проведення та суддівства турнірних змагань з єдиноборств, досягнуто повністю. Реалізовано працездатний прототип, який охоплює весь життєвий цикл змагання від цифрової реєстрації учасників і жеребкування до проведення поєдинків та підбиття фінансових і спортивних підсумків, здійснює програмну генерацію турнірних сіток п'яти форматів і суддівство в реальному часі на основі абстрактного рушія правил з підтримкою трьох дисциплін. У межах дослідження послідовно виконано всі шість поставлених завдань.

Аналіз предметної області виявив три ключові джерела обчислювальної складності: реляційну фільтрацію даних під час формування категорій, операції з турнірними графами та ризик станів гонитви при паралельній фіксації суддівських оцінок. Порівняння комерційних платформ MyUventex, Smoothcomp та Sportdata показало, що жодна з них одночасно не забезпечує автономну роботу в локальній мережі, легкий багатоплатформний вебінтерфейс і можливість змінювати логіку суддівства.

Для роботи в умовах нестабільного мережевого з'єднання обрано архітектуру модульного моноліту, оскільки повноцінна мікросервісна інфраструктура технічно неможлива на типовому обладнанні спортивних арен. Доменні класи розподілено між дев'ятьма предметними додатками, що дозволяє запускати систему автономно на локальному сервері без доступу до магістрального інтернету. Для трансляції перебігу змагань спроектовано комунікаційний шар на основі асинхронного інтерфейсу ASGI, брокера Redis та протоколу WebSocket, який миттєво оновлює публічні табло та ізолює базу даних від масових запитів на читання.

Вибір бази даних обґрунтовано зіставленням з документо-орієнтованими системами на кшталт MongoDB: на відміну від них, реляційний рушій PostgreSQL дозволяє обробляти багаторівневі турнірні сітки за допомогою оптимізованих

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		110

рекурсивних запитів, не обмежених лімітами оперативної пам'яті під час агрегацій. Схему бази даних нормалізовано до третьої нормальної форми, а зберігання слабоструктурованих конфігурацій реалізовано через нативний тип JSONB; частину правил предметної області перенесено безпосередньо на рівень реляційної схеми, тому їх дотримання забезпечує сама база даних незалежно від прикладного коду. Узгодженість суддівства в реальному часі забезпечено моделлю Event Sourcing, яка створює повний аудиторський слід кожної дії та гарантує ідемпотентність операцій, а також транзакційним бар'єром, що відкладає трансляцію оновлень до успішної фіксації транзакції, у поєднанні з оптимістичним оновленням інтерфейсу на клієнті.

Алгоритмічне ядро для генерації п'яти форматів турнірної сітки реалізовано у вигляді чистих функцій, не залежних від об'єктно-реляційного відображення, що забезпечило їх ізольоване тестування. Алгоритми виконують розведення одноклубників та обчислення багаторівневих додаткових показників швейцарської системи за допомогою рекурсивних запитів бази даних.

Розроблено абстрактний рушій правил на основі архітектурного патерну стратегії у поєднанні з глобальним реєстром дисциплін, що повністю ізолює логіку нарахування балів від управлінського ядра системи. Завдяки цьому підключення нової спортивної дисципліни не потребує модифікації наявного ядра, що підтверджено реалізацією підтримки трьох дисциплін: карате за регламентами Всесвітньої федерації карате, дзюдо за стандартами Міжнародної федерації дзюдо та тхеквондо за регламентами Всесвітнього тхеквондо. На клієнті цей самий принцип втілено в панелі оператора, яка динамічно підлаштовує набір доступних дій під дисципліну поточного поєдинку.

Серверну частину реалізовано на основі Django REST Framework та Django Channels, а клієнтський застосунок побудовано на бібліотеці React із єдиним інтерфейсом, який адаптує набір доступних функцій до ролі користувача. Рольову модель доступу побудовано за принципом RBAC із шістьма глобальними та трьома контекстними ролями; набуття привілейованих прав вимагає обов'язкової верифікації

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		111

адміністратором. Захист системи додатково охоплює перевірку власності на об'єкти, захист сеансових даних, обмеження частоти запитів до програмного інтерфейсу, суворо односпрямований канал WebSocket та криптографічну перевірку платіжних сповіщень банку за алгоритмом ECDSA. Розгортання організовано як відтворюваний процес: компоненти запаковано в контейнери Docker з багатоетапною збіркою, конфігурація зчитується зі змінних оточення, а доставлення нових версій виконує конвеєр безперервної інтеграції на основі GitHub Actions на віртуальному сервері Azure.

Коректність підтверджено набором із 510 автоматизованих тестів (405 серверних і 105 наскрізних), що дали загальне покриття серверного коду на рівні 91,6 відсотка; статичний аналіз у сервісі SonarCloud підтвердив подолання встановленого порогу якості. Алгоритми жеребкування та підбиття підсумків верифіковано на наборах даних з аналітично обчисленим результатом, а наскрізний сценарій відтворює повний цикл турніру через усі стадії в браузері. Експериментальні заміри підтвердили відповідність продуктивності проєктним вимогам: генерація сітки на 128 учасників виконується за частки мілісекунди для більшості форматів, латентність трансляції подій через WebSocket становить 184 мілісекунди за допустимої норми 300 мілісекунд, а усунення надлишкових запитів до бази даних знизило час завантаження сітки категорії до 953 мілісекунд при ліміті в одну секунду. Фіксація результату з просуванням переможця становить 681 мілісекунду і незначно перевищує цільовий показник у 500 мілісекунд; за даними профілювання це перевищення зосереджено в накладних витратах тестового середовища: записі журналу бази через віртуалізований диск та синхронній публікації подій, а не в доменній логіці. Навантажувальний тест зафіксував стабільну роботу без помилок до 10 одночасних користувачів і випадки перевищення ліміту очікування при 50, тоді як комунікаційний шар реального часу лишався повністю стабільним на всіх рівнях навантаження; отримані метрики визначають пріоритети доопрацювання системи перед промисловим використанням.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		112

Практична цінність отриманих результатів полягає в тому, що розроблений програмний комплекс доведено до рівня працездатного прототипу, придатного до дослідної експлуатації на реальних змаганнях із карате та потенційно інших контактних видах спорту. Він замінює паперовий документообіг і розрізнені електронні таблиці єдиним джерелом даних про учасника, а автоматизація жеребкування й суддівства знижує ризик операторських помилок, притаманний ручним процесам. Архітектура застосунку дозволяє надалі підвищувати межу одночасного навантаження через кешування відповідей на читання та горизонтальне масштабування застосункового шару, відповідно до зростання кількості учасників і майданчиків.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		113

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. Sebastopol : O'Reilly Media, 2017. 616 p. ISBN 978-1449373320.
2. Date C. J. An Introduction to Database Systems, 8th Edition / C. J. Date. – Pearson, 2003. – 1008 p. ISBN: 978-0321197849.
3. Li Y., Manoharan S. A performance comparison of SQL and NoSQL databases. 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), Victoria, BC, Canada, 2013. P. 15–19. DOI: 10.1109/PACRIM.2013.6625441.
4. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler. – Boston : Addison-Wesley, 2002. – 560 p. ISBN: 978-0321127426.
5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Reading : Addison-Wesley, 1994. 416 p. ISBN 0-201-63361-2.
6. Vernon V. Implementing Domain-Driven Design. Upper Saddle River : Addison-Wesley, 2013. 656 p. ISBN 978-0321834577.
7. World Karate Federation. WKF Kumite Competition Rules 2026 // Офіційний портал WKF. 2026. Режим доступу: https://www.wkf.net/files/pdf/documents/WKF%202026%20Kumite%20Competition%20Rules%20MASTER%20COPY_V11.pdf (дата звернення: 16.07.2026).
8. World Karate Federation. WKF Kata Competition Rules 2026 // Офіційний портал WKF. 2026. Режим доступу: https://www.wkf.net/files/pdf/documents/WKF%20Kata%20Competition%20Rules%2026%20MASTER%20COPY_V2.pdf (дата звернення: 16.07.2026).
9. Kendall G. Scheduling in sports: An annotated bibliography / G. Kendall, S. Knust, C. Ribeiro, D. Rubin // Computers & Operations Research. – 2010. – Vol. 37, No. 1. – P. 1-19. DOI: 10.1016/j.cor.2009.05.013

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		114

10. Cormen T. H. Introduction to Algorithms, Fourth Edition / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. – MIT Press, 2022. – 1312 p. ISBN: 978-0262046305.
11. Csató L. Ranking by pairwise comparisons for Swiss-system tournaments / L. Csató // Central European Journal of Operations Research. – 2013. – Vol. 21. – P. 783-803. DOI: 10.1007/s10100-012-0261-8
12. Gilbert S., Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News, 2002. Vol. 33, № 2. P. 51–59. DOI: 10.1145/564585.564601.
13. Pahl C. Containers and clusters for edge cloud architectures – a technology review / C. Pahl, B. Lee // 2015 3rd International Conference on Future Internet of Things and Cloud. – IEEE, 2015. – P. 379-386. DOI: 10.1109/FiCloud.2015.35
14. MDN Web Docs. The WebSocket API (WebSockets). URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (дата звернення: 16.07.2026).
15. Tanenbaum A. S., Wetherall D. J. Computer Networks. 5th ed. Boston : Pearson, 2010. 960 p. ISBN 978-0132126953.
16. Платформа MyUventex. Офіційний сайт системи управління змаганнями. 2026. URL: <https://uventex.com/> (дата звернення: 16.07.2026).
17. RESTful web API design // Microsoft Azure Architecture Center. 2024. URL: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design> (дата звернення: 16.07.2026).
18. Varghese B., Wang N., Barbhuiya S., Kilpatrick P., Nikolopoulos D. S. Challenges and opportunities in edge computing. 2016 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 2016. P. 20-26. DOI: 10.1109/SmartCloud.2016.18.
19. Платформа Smoothcomp. Офіційний сайт турнірної системи. 2026. URL: <https://smoothcomp.com/> (дата звернення: 16.07.2026).

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		115

20. Al-Qora'n L., Al-Said Ahmad A. Modular Monolith Architecture in Cloud Environments: A Systematic Literature Review. Future Internet. 2025. Vol. 17, № 11. Article 496. DOI: 10.3390/fi17110496.
21. Платформа Sportdata. Офіційний сайт технологічної платформи. 2026. URL: <https://www.sportdata.org/> (дата звернення: 16.07.2026).
22. Pimentel V., Nickerson B. G. Communicating and displaying real-time data with WebSocket. IEEE Internet Computing, 2012. Vol. 16, № 4. P. 45-53. DOI: 10.1109/MIC.2012.64.
23. API документація: Еквайринг // Monobank. 2026. URL: <https://monobank.ua/api-docs/acquiring> (дата звернення: 16.07.2026).
24. Django REST framework Collaborative. Django REST framework 3.15 Documentation. 2024. URL: <https://www.django-rest-framework.org/> (дата звернення: 16.07.2026).
25. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. Sebastopol : O'Reilly Media, 2020. 432 p. ISBN 978-1492043454.
26. Kalske M., Mäkitalo N., Mikkonen T. Challenges When Moving from Monolith to Microservice Architecture. Current Trends in Web Engineering : ICWE 2017 International Workshops / eds. I. Garrigós, M. Wimmer. Cham : Springer, 2018. (Lecture Notes in Computer Science ; vol. 10544). P. 32–47. DOI: 10.1007/978-3-319-74433-9_3.
27. Richardson C. Microservices Patterns: With examples in Java. Shelter Island : Manning Publications, 2018. 520 p. ISBN 978-1617294549.
28. Blinowski G., Ojdowska A., Przybyłek A. Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. IEEE Access. 2022. Vol. 10. P. 20357–20374. DOI: 10.1109/ACCESS.2022.3152803.
29. Channels Documentation. Introduction to Channels. URL: <https://channels.readthedocs.io/en/latest/introduction.html> (дата звернення: 16.07.2026).

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		116

30. Percival H., Gregory B. Architecture Patterns with Python: Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven Microservices. Sebastopol : O'Reilly Media, 2020. 306 p. ISBN 978-1492052203.
31. Python Software Foundation. What's New In Python 3.12. URL: <https://docs.python.org/3/whatsnew/3.12.html> (дата звернення: 16.07.2026).
32. Casciaro M., Mammino L. Node.js Design Patterns: Design and implement production-grade Node.js applications using proven patterns and techniques. 3rd ed. Birmingham : Packt Publishing, 2020. 664 p. ISBN 978-1839214110.
33. Django Software Foundation. Django 5.2 release notes. URL: <https://docs.djangoproject.com/en/5.2/releases/5.2/> (дата звернення: 16.07.2026).
34. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide. 3rd ed. Sebastopol : O'Reilly Media, 2019. 514 p. ISBN 978-1491954461.
35. PostgreSQL Global Development Group. WITH Queries (Common Table Expressions). PostgreSQL 18 Documentation. URL: <https://www.postgresql.org/docs/current/queries-with.html> (дата звернення: 16.07.2026).
36. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York : McGraw-Hill Education, 2019. 1376 p. ISBN 978-0078022159.
37. Seow S. C. Designing and Engineering Time: The Psychology of Time Perception in Software. Boston : Addison-Wesley, 2008. 224 p. ISBN 978-0321509185.
38. React v19.0. React Documentation. URL: <https://react.dev/blog/2024/12/05/react-19> (дата звернення: 16.07.2026).
39. Vite. Getting Started. URL: <https://vite.dev/guide/> (дата звернення: 16.07.2026).
40. International Judo Federation. IJF Sport and Organisation Rules (SOR), version 30.01.2025 // Офіційний портал IJF. 2025. URL: <https://sor.ijf.org/> (дата звернення: 16.07.2026).
41. World Taekwondo. WT Competition Rules & Interpretation (September 30, 2024), Kyorugi // Офіційний портал World Taekwondo, розділ Rules and Regulations.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		117

URL: <https://www.worldtaekwondo.org/inside/documents/rar/list> (дата звернення: 16.07.2026).

42. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd ed. Upper Saddle River : Pearson, 2004. 736 p. ISBN 978-0131489066.

43. Sandhu R. S., Coyne E. J., Feinstein H. L., Youman C. E. Role-Based Access Control Models. IEEE Computer. 1996. Vol. 29, № 2. P. 38–47. DOI: 10.1109/2.485845.

44. Celko J. Joe Celko's Trees and Hierarchies in SQL for Smarties. 2nd ed. Burlington : Morgan Kaufmann (Elsevier), 2012. 286 p. ISBN 978-0123877338.

45. Morrison D. R., Jacobson S. H., Sauppe J. J., Sewell E. C. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization. 2016. Vol. 19. P. 79–102. DOI: 10.1016/j.disopt.2016.01.005.

46. Nast M., Raddatz H., Rother B., Golatowski F., Timmermann D. A Survey and Comparison of Publish/Subscribe Protocols for the Industrial Internet of Things (IIoT). Proceedings of the 12th International Conference on the Internet of Things (IoT '22). ACM, 2022. P. 193–200. DOI: 10.1145/3567445.3571107.

47. Elliptic curve cryptography (ECDSA) // Cryptography 42.0.0 Documentation. 2024. URL: <https://cryptography.io/en/latest/hazmat/primitives/asymmetric/ec/> (дата звернення: 16.07.2026).

48. Likaj X., Khodayari S., Pellegrino G. Where We Stand (or Fall): An Analysis of CSRF Defenses in Web Frameworks. Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '21). San Sebastian, Spain, 2021. P. 370–385. DOI: 10.1145/3471621.3471846.

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		118

ДОДАТКИ

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		119

Деталізована схема бізнес-процесу проведення турніру

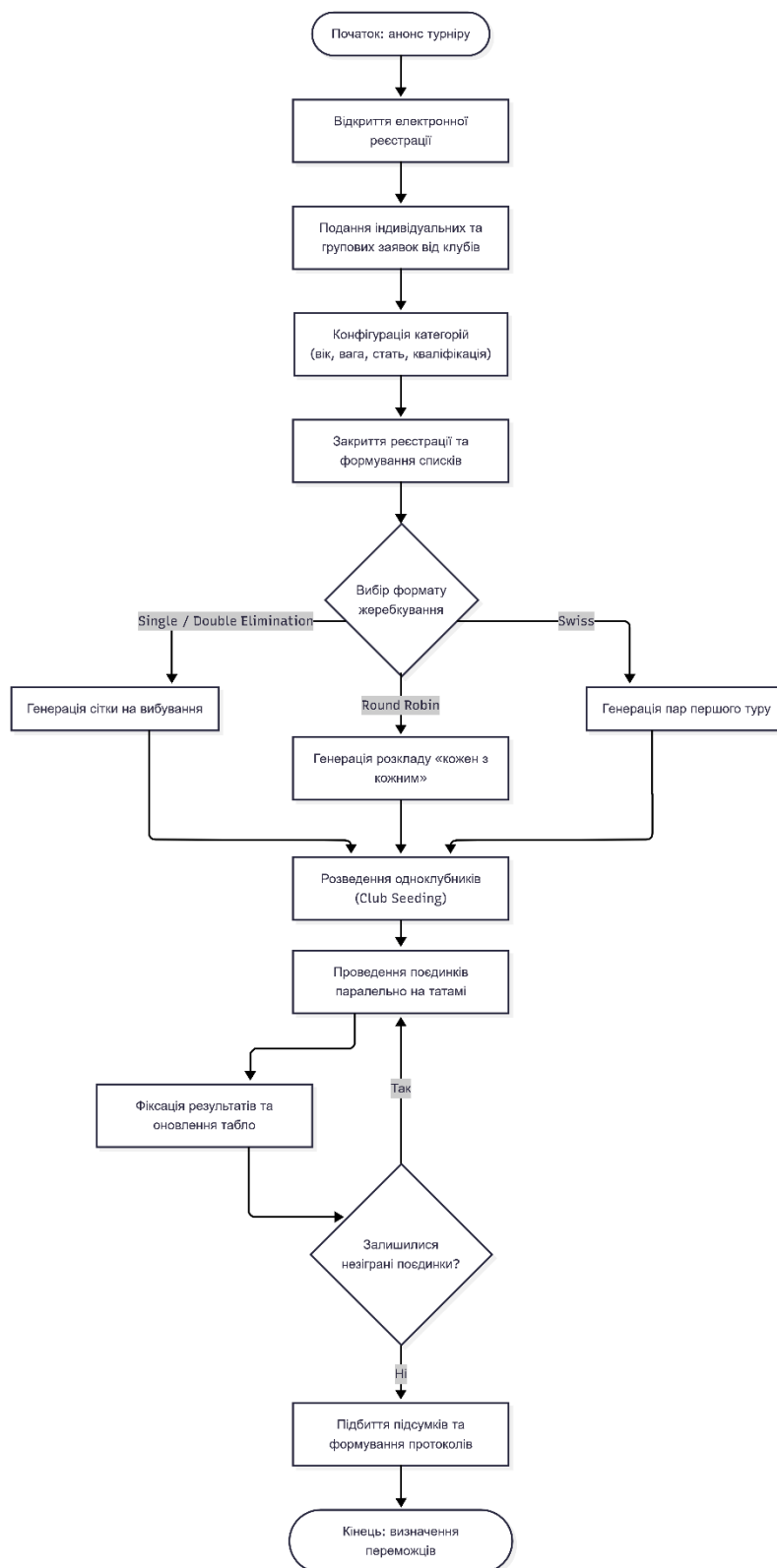


Рис. А.1 – Бізнес-процес проведення турніру

		Муравицький В. Ю.		
		Толстой І. А.		
Змн.	Арк.	№ докум.	Підпис	Дата

Модель варіантів використання системи

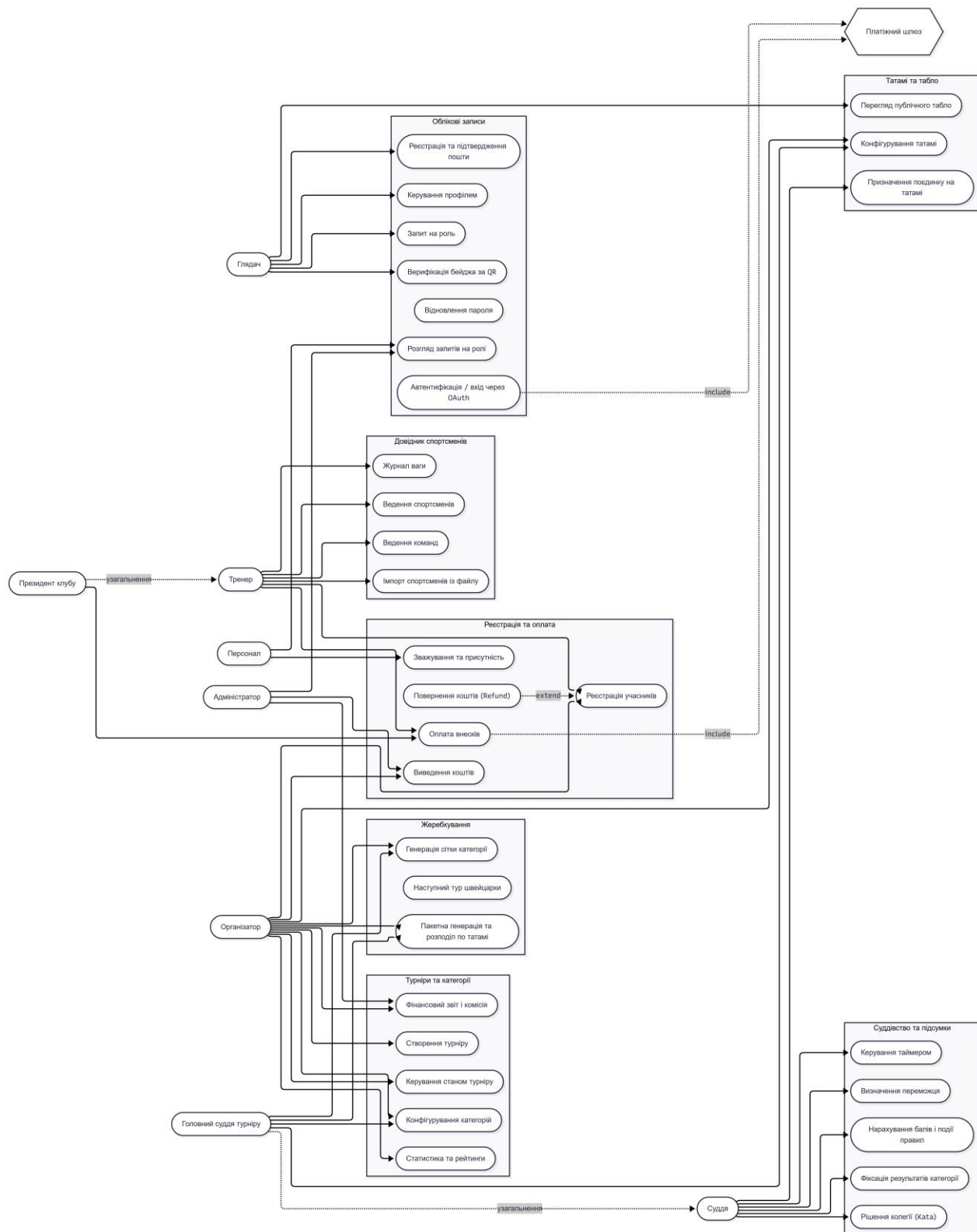


Рис. Б.1 – Діаграма варіантів використання системи

Об'єктно-орієнтована структура системи

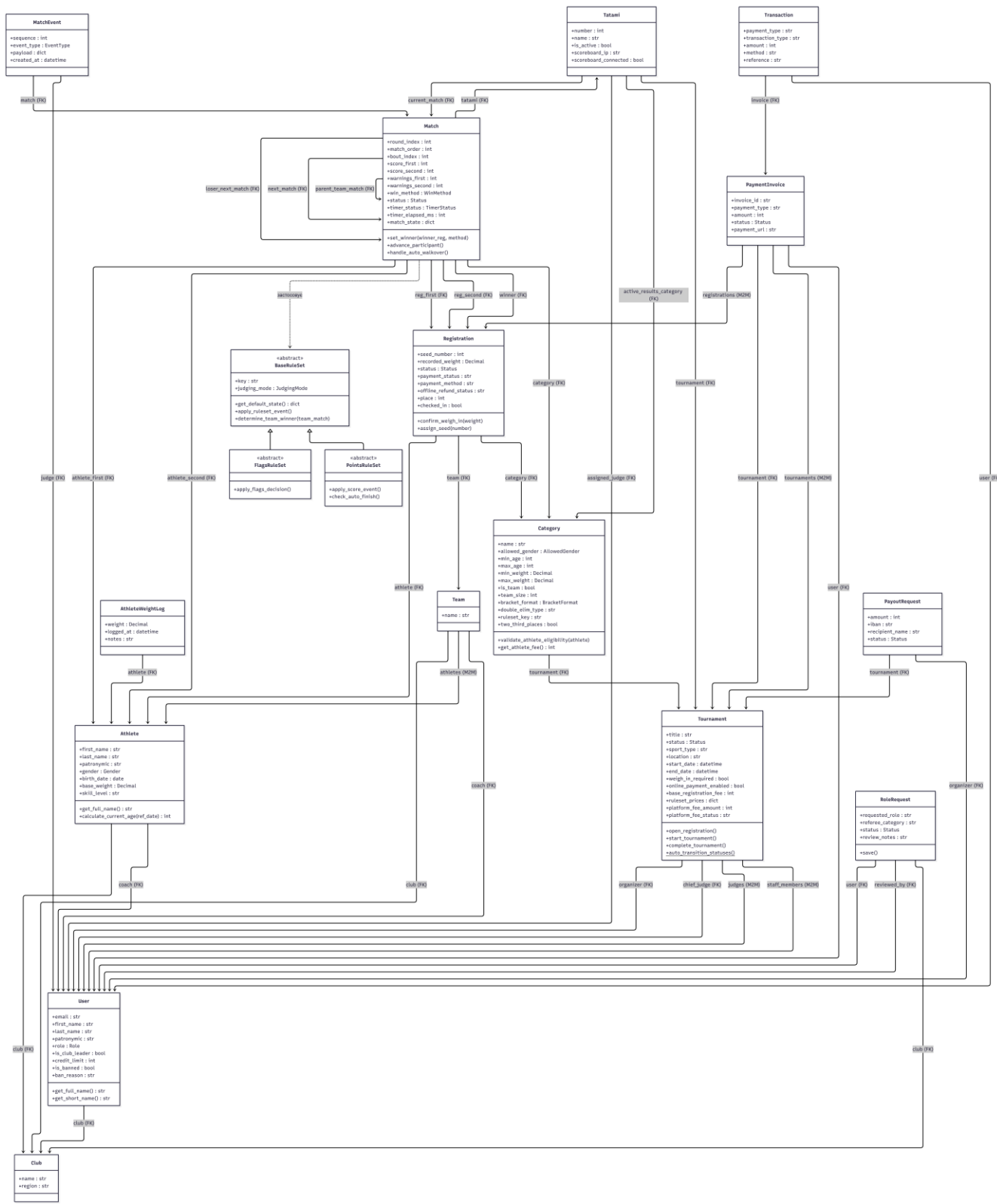


Рис. В.1 – Діаграма класів предметної області

Структура системи бази даних

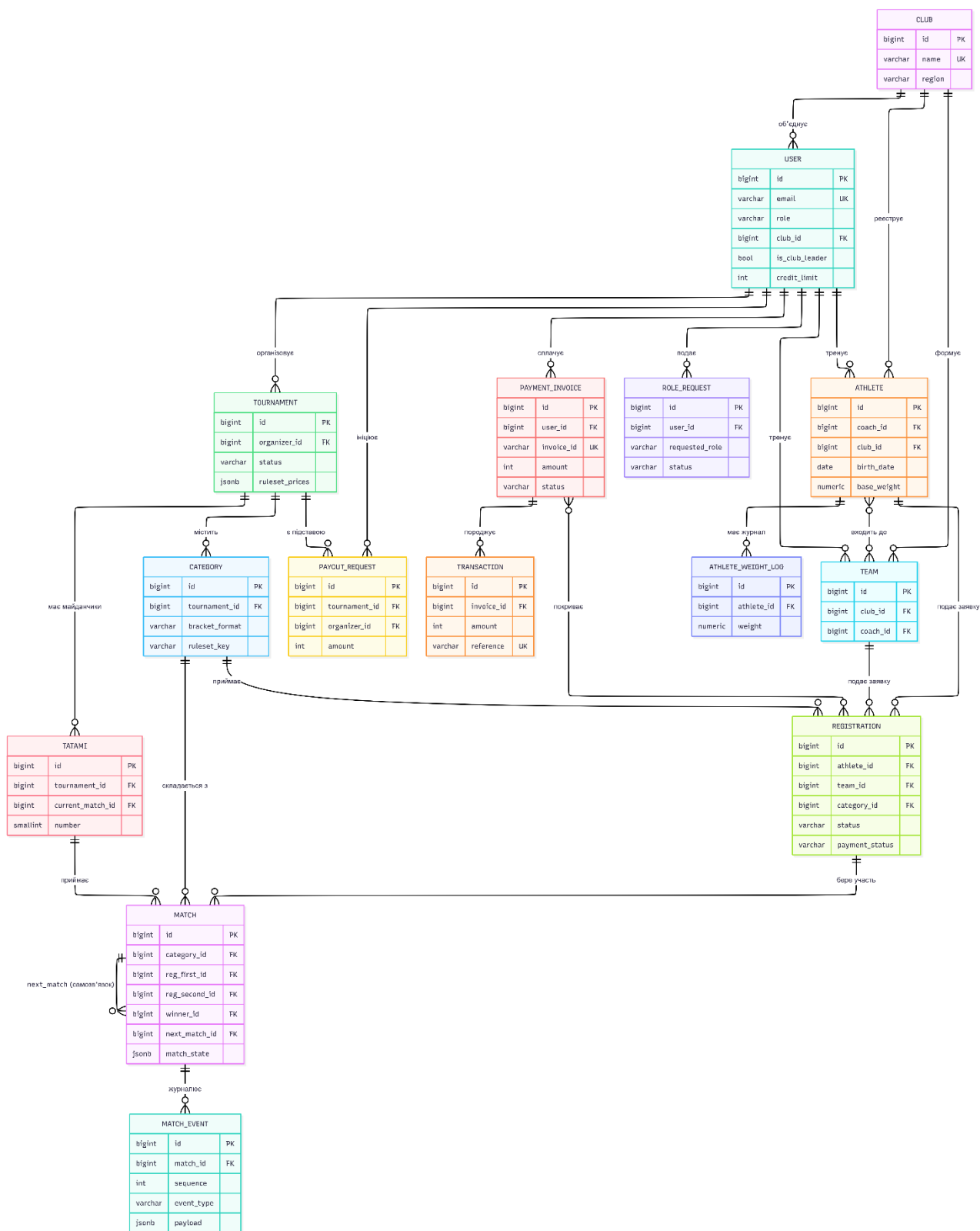


Рис. Г.1 – Діаграма «сутність–зв'язок» бази даних

Потоки даних у системі

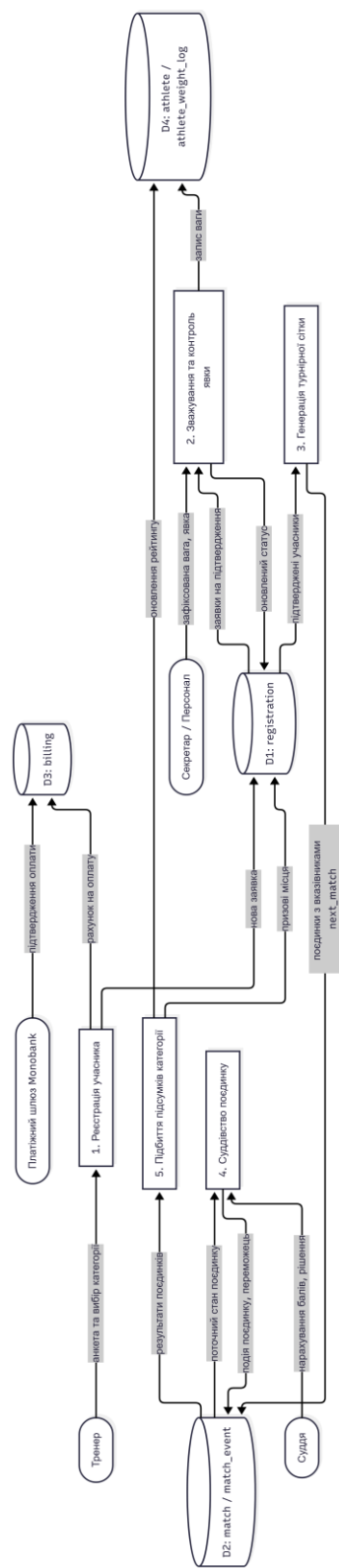


Рис. Д.1 – Діаграма потоків даних під час проведення змагання

Фрагменти програмного коду серверних модулів

Лістинг Е.1 – Реєстр і фабрика правил суддівства (rulesets/registry.py):

```

from apps.rulesets.base import BaseRuleSet
_REGISTRY: dict[str, type[BaseRuleSet]] = {}
def register_ruleset(cls: type[BaseRuleSet]) -> type[BaseRuleSet]:
    _registry[cls.key] = cls
    return cls
def get_ruleset(key: str) -> BaseRuleSet:
    try:
        return _registry[key]
    except KeyError as exc:
        raise KeyError(f"Unknown ruleset: '{key}'. Available: {list(_registry)}") from exc
def list_rulesets() -> list[dict]:
    from apps.rulesets.base import PointsRuleSet
    result = []
    for cls in _registry.values():
        instance = cls()
        entry: dict = {
            "key": cls.key,
            "name": cls.name,
            "sport_type": cls.sport_type,
            "judging_mode": cls.judging_mode,
            "default_duration_seconds": instance.get_default_duration_seconds()
            if isinstance(instance, PointsRuleSet)
            else None,
            "win_methods": [{"key": w.key, "label": w.label} for w in
instance.get_win_methods()],
        }
        if isinstance(instance, PointsRuleSet):
            entry["score_actions"] = [
                {
                    "key": a.key,
                    "label": a.label,
                    "points": a.points,
                    "is_warning": a.is_warning,
                }
                for a in instance.get_score_actions()
            ]
        else:
            entry["score_actions"] = []
        result.append(entry)
    return result

```

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		125

Лістинг Е.2 – Оптимальне швейцарське парування методом гілок і меж (brackets/utils.py):

```
def pair_swiss_optimal(players: list, past_opponents: dict[int, set[int]]) ->
list[tuple] | None:
    """
    Знаходить глобально оптимальне швейцарське парування для списку учасників.
    Мінімізує суму квадратів різниці результатів між парами суперників методом
    гілок і меж, забороняючи повторні зустрічі вже зіграних суперників.
    """
    best_matching = []
    min_penalty = float("inf")
    iterations = 0
    max_iterations = 10000
    def backtrack(unpaired, current_matching, current_penalty):
        nonlocal best_matching, min_penalty, iterations

        iterations += 1
        if iterations > max_iterations:
            return

        if current_penalty >= min_penalty:
            return

        if not unpaired:
            best_matching = list(current_matching)
            min_penalty = current_penalty
            return

        p1 = unpaired[0]
        # Кандидати, що ще не грали з p1
        candidates = [c for c in unpaired[1:] if c.id not in past_opponents.get(p1.id,
set())]
        # Сортуюмо за різницею результатів – кращі пари знаходяться раніше, відсікання
агресивніше
        candidates.sort(key=lambda c: abs(p1.score - c.score))

        for c in candidates:
            penalty = (p1.score - c.score) ** 2
            next_unpaired = [p for p in unpaired if p.id != p1.id and p.id != c.id]
            backtrack(next_unpaired, current_matching + [(p1, c)], current_penalty +
penalty)

    backtrack(players, [], 0)

    if min_penalty == float("inf"):
        return None
    return best_matching
```

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		126

Лістинг Е.3 – Генерація сітки подвійного вибування (brackets/services.py):

```
@transaction.atomic
def generate_double_elimination(self) -> list[Match]:
    """Створює повне дерево double-elimination матчів."""
    self._validate_preconditions()
    registrations = self._get_confirmed_registrations()
    self._assign_seeds_if_missing(registrations)
    duration_ms = self._get_match_duration_ms()

    participants = [self._to_participant(r) for r in registrations]
    reg_by_id = {r.id: r for r in registrations}

    slots = utils.build_single_elimination_slots(participants, avoid_club_conflicts=True)
    bracket_size = utils.next_power_of_two(len(participants))
    import math

    k = int(math.ceil(math.log2(bracket_size)))

    # 1. Гранд-фінал (round_index = 200, match_order = 1)
    gf_match = Match.objects.create(
        category=self.category,
        round_index=200,
        match_order=1,
        status=Match.Status.SCHEDULED,
        timer_duration_ms=duration_ms,
    )

    losers_matches = {}

    # 2. Нижня сітка (Losers Bracket) - будується знизу вгору
    for lvl in range(2 * k - 2, 0, -1):
        M = bracket_size // (2 ** ((lvl + 3) // 2))
        for m in range(1, M + 1):
            if lvl == 2 * k - 2:
                next_match = gf_match
            elif lvl % 2 == 1:
                next_match = losers_matches[(lvl + 1, m)]
            else:
                next_match = losers_matches[(lvl + 1, (m + 1) // 2)]

            match_obj = Match.objects.create(
                category=self.category,
                round_index=100 + lvl,
                match_order=m,
                next_match=next_match,
                status=Match.Status.SCHEDULED,
                timer_duration_ms=duration_ms,
            )
            losers_matches[(lvl, m)] = match_obj
```

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		127

```

# 3. Верхня сітка (Winners Bracket) - будується знизу вгору
winners_matches = {}
for r in range(k, 0, -1):
    M = bracket_size // (2**r)
    for m in range(1, M + 1):
        if r == k:
            next_match = gf_match
        else:
            next_match = winners_matches[(r + 1, (m + 1) // 2)]

        if r == 1:
            loser_next_match = losers_matches[(1, (m + 1) // 2)]
        else:
            loser_next_match = losers_matches[(2 * r - 2, M - m + 1)]

        reg_first = None
        reg_second = None
        if r == 1:
            slot_first = slots[2 * m - 2]
            slot_second = slots[2 * m - 1]
            if slot_first:
                reg_first = reg_by_id.get(slot_first.id)
            if slot_second:
                reg_second = reg_by_id.get(slot_second.id)

        match_obj = Match.objects.create(
            category=self.category,
            reg_first=reg_first,
            reg_second=reg_second,
            round_index=r,
            match_order=m,
            next_match=next_match,
            loser_next_match=loser_next_match,
            status=Match.Status.SCHEDULED,
            timer_duration_ms=duration_ms,
        )
        winners_matches[(r, m)] = match_obj

# 4. Опрацювання всіх порожніх проходів
self._process_byes_for_category(self.category)

return list(
    Match.objects.filter(category=self.category).order_by("round_index",
"match_order")
)

```

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		128

Лістинг Е.4 – Транзакційна фіксація переможця та просування учасника (matches/models.py):

```
@transaction.atomic
def set_winner(self, winner_registration, method=WinMethod.DECISION):
    """Фіксує переможця, завершує матч і просуває його у next_match."""
    if winner_registration not in (self.reg_first, self.reg_second):
        raise ValidationError("Переможцем може бути лише учасник цього поєдинку")

    self.winner = winner_registration
    self.win_method = method
    self.status = self.Status.COMPLETED
    from django.utils import timezone

    self.completed_at = timezone.now()

    update_fields = ["winner", "win_method", "status", "completed_at"]
    if self.category.is_team and not self.parent_team_match:
        update_fields.extend(["score_first", "score_second"])
    self.save(update_fields=update_fields)

# Якщо це суб-бій командного матчу:
if self.parent_team_match:
    self._handle_parent_team_match_update()
else:
    self.advance_participant()

# 1. Репешаж: добудова втішних боїв після завершення півфіналів
if self.category.bracket_format == "single_repechage":
    if self.next_match and self.next_match.next_match_id is None:
        final_match = self.next_match
        semis = Match.objects.filter(category=self.category,
next_match=final_match)
        if all(semi.status == Match.Status.COMPLETED for semi in semis):
            from apps.brackets.services import RepechageService

            RepechageService.generate_for_category(self.category)

# 2. Гранд-фінал Double Elimination: створення Bracket Reset за потреби
if self.category.bracket_format == "double_elimination" and self.round_index
== 200:
    if self.category.double_elim_type == "full":
        wb_finalist = None
        source_matches = Match.objects.filter(
            category=self.category, next_match=self,
        ).select_related("reg_first", "reg_second")
        for src in source_matches:
            if src.loser_next_match_id is not None:
                wb_finalist = src.winner
                break

        lb_winner_won = (wb_finalist is not None) and (self.winner !=
wb_finalist)

        br_already_exists = Match.objects.filter(
            category=self.category, round_index=201
        ).exists()
```

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		129

```

if lb_winner_won and not br_already_exists:
    super_final = Match.objects.create(
        category=self.category,
        reg_first=wb_finalist,
        reg_second=self.winner,
        round_index=201,
        match_order=1,
        is_bracket_reset=True,
        tatami=self.tatami,
        timer_duration_ms=self.timer_duration_ms,
        status=self.Status.SCHEDULED,
    )
    self.redirect_to_match_id = super_final.id
    # ... оновлення татамі й трансляція ...

# 3. Обробка порожніх проходів
from apps.brackets.services import BracketGenerator

generator = BracketGenerator(self.category)
generator._process_byes_for_category(self.category)

from apps.common.broadcast import broadcast_match_update

transaction.on_commit(lambda: broadcast_match_update(self))

def advance_participant(self):
    """Переносить переможця в наступний матч дерева."""
    if self.next_match and self.winner:
        nxt = self.next_match
        nxt.refresh_from_db()
        if nxt.reg_first_id == self.winner_id or nxt.reg_second_id == self.winner_id:
            if nxt.reg_first_id is not None and nxt.reg_second_id is not None:
                raise ValidationError(f"Наступний матч {nxt.id} вже заповнено обома
учасниками")
            elif nxt.reg_first is None:
                nxt.reg_first = self.winner
                nxt.save(update_fields=["reg_first"])
            elif nxt.reg_second is None:
                nxt.reg_second = self.winner
                nxt.save(update_fields=["reg_second"])
            else:
                raise ValidationError(f"Наступний матч {nxt.id} вже заповнено обома
учасниками")

        from apps.common.broadcast import broadcast_match_update

        transaction.on_commit(lambda: broadcast_match_update(nxt))

# Автоматичне вирішення неявки суперника, якщо він знятий з турніру
nxt.refresh_from_db()
nxt.handle_auto_walkover()

```

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		130

```

# Просування того, хто програв (для Double Elimination)
if self.loser_next_match:
    loser = self.reg_second if self.winner == self.reg_first else self.reg_first
    if loser:
        nxt_loser = self.loser_next_match
        nxt_loser.refresh_from_db()
        if nxt_loser.reg_first is None:
            nxt_loser.reg_first = loser
            nxt_loser.save(update_fields=["reg_first"])
        elif nxt_loser.reg_second is None:
            nxt_loser.reg_second = loser
            nxt_loser.save(update_fields=["reg_second"])

    from apps.common.broadcast import broadcast_match_update

    transaction.on_commit(lambda: broadcast_match_update(nxt_loser))
    nxt_loser.refresh_from_db()
    nxt_loser.handle_auto_walkover()

```

Лістинг Е.5 – Шар трансляції подій у режимі реального часу
(common/broadcast.py):

```

import time
from asgiref.sync import async_to_sync
from channels.layers import get_channel_layer
MATCH_EVENT_TYPE = "match.event"
def tatami_group(match) -> str | None:
    if not match.tatami_id:
        return None
    t = match.tatami
    return f"tatami{t.tournamentid}{t.number}"
def broadcast_match_event(match, event) -> None:
    """Віялове розсилання в групу категорії та групу татамі (за наявності)."""
    from apps.matches.serializers import MatchSerializer
    channel_layer = get_channel_layer()
    payload = {
        "type": MATCH_EVENT_TYPE,
        "match_id": match.id,
        "event": {
            "sequence": event.sequence,
            "event_type": event.event_type,
            "payload": event.payload,
        },
        "match": MatchSerializer(match).data,
    }
    async_to_sync(channel_layer.group_send)(f"category_{match.category_id}", payload)
    group = _tatami_group(match)
    if group:
        async_to_sync(channel_layer.group_send)(group, payload)
def broadcast_timer_state(match) -> None:
    group = _tatami_group(match)
    if not group:
        return

```

```

channel_layer = get_channel_layer()
async_to_sync(channel_layer.group_send)(
    group,
    {
        "type": "timer.state",
        "match_id": match.id,
        "server_ts_ms": int(time.time() 1000),
        "state": {
            "status": match.timer_status,
            "started_at_ms": (
                int(match.timer_started_at.timestamp() 1000)
                if match.timer_started_at
                else None
            ),
            "elapsed_ms": match.timer_elapsed_ms,
            "duration_ms": match.timer_duration_ms,
        },
    },
)

```

		Муравицький В. Ю.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Толстой І. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		132

Структура таблиці tournament (турнір)

Поле	Тип даних	Призначення та обмеження
id	BIGINT	Сурогатний первинний ключ
organizer_id	BIGINT	Зовнішній ключ на користувача-власника; ON DELETE RESTRICT – видалення зайнятого організатора заборонено
title	VARCHAR(255)	Назва турніру
description	TEXT	Текстовий опис
sport_type	VARCHAR(100)	Вид спорту (карате, дзюдо, тхеквондо тощо)
location	VARCHAR(255)	Місце проведення
start_date, end_date	TIMESTAMPTZ	Межі проведення змагання
registration_start, registration_end	TIMESTAMPTZ, NULL	Вікно прийому заявок
completed_at	TIMESTAMPTZ, NULL	Момент завершення турніру
status	VARCHAR(20)	Стан життєвого циклу: draft, registration, active, completed
weigh_in_required	BOOLEAN	Чи передбачено офіційне зважування
online_payment_enabled	BOOLEAN	Доступність онлайн-оплати внесків
payment_details	TEXT	Реквізити для офлайн-оплати
base_registration_fee	INTEGER, CHECK ≥ 0	Базовий внесок за участь (за замовчуванням 500)
ruleset_prices	JSONB	Відображення «ключ набору правил → ціна» для індивідуальних категорій
base_team_registration_fee	INTEGER, NULL	Базовий внесок за одного учасника командної категорії
ruleset_team_prices	JSONB	Ціни командних категорій за наборами правил
commission_payer	VARCHAR(20)	Платник комісії платформи: buyer або organizer

Продовження таблиці Ж.1

Поле	Тип даних	Призначення та обмеження
refund_policy	VARCHAR(20)	Політика повернення внесків: refundable або non_refundable
platform_fee_status	VARCHAR(20)	Стан оплати комісії платформи: paid або unpaid
platform_fee_amount	INTEGER, CHECK ≥ 0	Нарахована комісія за офлайн-заявки
chief_judge_id	BIGINT, NULL	Зовнішній ключ на головного суддю; ON DELETE SET NULL
use_check_in	BOOLEAN	Чи ведеться відмітка фактичної явки учасників
created_at	TIMESTAMPTZ	Момент створення запису

Таблиця Ж.2

Структура таблиці category (категорія)

Поле	Тип даних	Призначення та обмеження
id	BIGINT	Сурогатний первинний ключ
tournament_id	BIGINT	Зовнішній ключ на турнір; ON DELETE CASCADE
name	VARCHAR(100)	Назва категорії
allowed_gender	VARCHAR(10)	Допустима стать: male, female, mixed
min_age, max_age	SMALLINT, CHECK ≥ 0	Віковий діапазон дивізіону
min_weight, max_weight	NUMERIC(5,2), NULL	Ваговий діапазон (NULL – категорія без вагових обмежень)
allowed_skill_level	VARCHAR(50)	Допустимий рівень майстерності
is_team	BOOLEAN	Ознака командної категорії
team_size	SMALLINT, CHECK ≥ 0	Кількість бійців у команді (за замовчуванням 3)
registration_fee	INTEGER, NULL	Спеціальний внесок, що перевизначає базову вартість турніру

Продовження таблиці Ж.2

Поле	Тип даних	Призначення та обмеження
bracket_format	VARCHAR(50)	Формат сітки: single_elimination, round_robin, single_repechage, double_elimination, swiss
double_elim_type	VARCHAR(10)	Підтип подвійного вибування: full або short
ruleset_key	VARCHAR(50)	Ключ підключеного набору правил (за замовчуванням karate_wkf)
match_duration_seconds	INTEGER, NULL	Тривалість поєдинку, що перевизначає значення з набору правил
schedule_order	INTEGER, CHECK ≥ 0	Порядок категорії у розкладі
two_third_places	BOOLEAN	Чи присуджуються два третіх місця
judges_count	SMALLINT, NULL	Кількість суддів для дисципліни «ката»

Таблиця Ж.3

Структура таблиці registration (заявка на участь)

Поле	Тип даних	Призначення та обмеження
id	BIGINT	Сурогатний первинний ключ
athlete_id	BIGINT, NULL	Зовнішній ключ на спортсмена (індивідуальна заявка); ON DELETE CASCADE
team_id	BIGINT, NULL	Зовнішній ключ на команду (командна заявка); ON DELETE SET NULL
category_id	BIGINT	Зовнішній ключ на категорію; ON DELETE CASCADE
seed_number	INTEGER, NULL	Номер посіву, встановлюваний алгоритмом жеребкування
recorded_weight	NUMERIC(5,2), NULL	Вага, зафіксована під час офіційного зважування
status	VARCHAR(20)	Стан заявки: pending, confirmed, rejected, withdrawn
payment_status	VARCHAR(20)	Стан оплати: unpaid, paid
payment_method	VARCHAR(20)	Спосіб оплати: online, offline
offline_refund_status	VARCHAR(20)	Стан повернення готівки: none, pending, confirmed

Продовження таблиці Ж.3

Поле	Тип даних	Призначення та обмеження
place	SMALLINT, NULL	Призове місце, отримане за підсумками категорії
checked_in	BOOLEAN	Підтвердження фактичної явки на місце проведення
created_at	TIMESTAMP TZ	Момент подання заявки

Таблиця Ж.4

Структура таблиць довідника спортсменів

Таблиця	Поле	Тип даних	Призначення та обмеження
athlete	id	BIGINT	Сурогатний первинний ключ
athlete	coach_id	BIGINT	Зовнішній ключ на тренера; ON DELETE CASCADE
athlete	club_id	BIGINT	Зовнішній ключ на клуб; ON DELETE RESTRICT
athlete	first_name, last_name	VARCHAR(100)	Ім'я та прізвище
athlete	patronymic	VARCHAR(100)	По-батькові
athlete	gender	VARCHAR(10)	Стать: male, female
athlete	birth_date	DATE	Дата народження (підстава для обчислення віку)
athlete	base_weight	NUMERIC(5,2)	Базова вага для попередньої перевірки відповідності категорії
athlete	skill_level	VARCHAR(50)	Рівень майстерності (пояс, дан)
athlete	photo	VARCHAR	Шлях до файлу фотографії
athlete_weight_log	id	BIGINT	Сурогатний первинний ключ
athlete_weight_log	athlete_id	BIGINT	Зовнішній ключ на спортсмена; ON DELETE CASCADE
athlete_weight_log	weight	NUMERIC(5,2)	Зафіксована вага
athlete_weight_log	logged_at	TIMESTAMP TZ	Момент фіксації
athlete_weight_log	notes	VARCHAR(255)	Примітка (джерело заміру)
team	id	BIGINT	Сурогатний первинний ключ

Муравицький В. Ю.

Толстой І. А.

Змн.

Арк.

№ докум.

Підпис

Дата

ІПЗ.КР.Б – 121 – 26 – ПЗ

Арк.

136

Продовження таблиці Ж.4

Таблиця	Поле	Тип даних	Призначення та обмеження
team	name	VARCHAR(150)	Назва команди
team	club_id	BIGINT	Зовнішній ключ на клуб; ON DELETE CASCADE
team	coach_id	BIGINT	Зовнішній ключ на тренера; ON DELETE CASCADE

Таблиця Ж.5

Структура таблиці match (поєдинок)

Поле	Тип даних	Призначення та обмеження
id	BIGINT	Сурогатний первинний ключ
category_id	BIGINT	Зовнішній ключ на категорію; ON DELETE CASCADE
parent_team_match_id	BIGINT, NULL	Рекурсивний зовнішній ключ на командну зустріч для суб-боїв; ON DELETE CASCADE
athlete_first_id, athlete_second_id	BIGINT, NULL	Зовнішні ключі на атлетів (Ака, Ао) для командних боїв; ON DELETE SET NULL
bout_index	SMALLINT, NULL	Порядковий номер бою в командній зустрічі
reg_first_id, reg_second_id	BIGINT, NULL	Зовнішні ключі на заявки-учасники; ON DELETE SET NULL
round_index	SMALLINT	Номер раунду (1 – перший, далі 1/8, 1/4 тощо)
match_order	SMALLINT	Порядковий номер поєдинку в раунді
tatami_id	BIGINT, NULL	Зовнішній ключ на майданчик; ON DELETE SET NULL
score_first, score_second	INTEGER	Поточний рахунок учасників
warnings_first, warnings_second	INTEGER	Накопичені попередження
winner_id	BIGINT, NULL	Зовнішній ключ на заявку-переможця; ON DELETE SET NULL

Продовження таблиці Ж.5

Поле	Тип даних	Призначення та обмеження
win_method	VARCHAR(100)	Спосіб перемоги (системні ідентифікатори): decision, ippon, wazaari, hantei, hansoku, walkover тощо
match_duration	INTEGER, NULL	Фактична тривалість поєдинку (с)
next_match_id	BIGINT, NULL	Рекурсивний зовнішній ключ для побудови дерева сітки; NULL позначає фінал; ON DELETE SET NULL
loser_next_match_id	BIGINT, NULL	Рекурсивний зовнішній ключ на наступний поєдинок (список суміжності); NULL позначає фінал
senshu	VARCHAR(4)	Першість результативної дії: none, aka, ao
flags_aka, flags_ao	SMALLINT, NULL	Кількість прапорів суддів у дисципліні формальних комплексів
judges_count	SMALLINT, NULL	Кількість суддів колегії («ката»)
status	VARCHAR(20)	Стан: scheduled, ongoing, completed
started_at, completed_at	TIMESTAMP TZ, NULL	Моменти початку та завершення
timer_status	VARCHAR(15)	Стан таймера: not_started, running, paused, finished
timer_started_at	TIMESTAMP TZ, NULL	Момент останнього запуску таймера
timer_elapsed_ms, timer_duration_ms	INTEGER	Накопичений та повний час поєдинку (мс)
show_timer	BOOLEAN	Чи відображати таймер на табло
is_bracket_reset	BOOLEAN	Ознака супер-фіналу (Bracket Reset) у подвійному вибуванні
match_state	JSONB	Слабоструктурований стан поєдинку, специфічний для набору правил

Структура таблиці match_event (подія поєдинку)

Поле	Тип даних	Призначення та обмеження
id	BIGINT	Сурогатний первинний ключ
match_id	BIGINT	Зовнішній ключ на поєдинок; ON DELETE CASCADE
sequence	INTEGER, CHECK ≥ 0	Порядковий номер події в межах поєдинку
event_type	VARCHAR(20)	Тип події: score, warning, senshu, flags_decision, start, finish, операції з таймером тощо
payload	JSONB	Корисне навантаження події (деталі дії)
judge_id	BIGINT, NULL	Зовнішній ключ на суддю-автора; ON DELETE SET NULL
created_at	TIMESTAMPTZ	Момент реєстрації події

Структура таблиці tatami (змагальний майданчик)

Поле	Тип даних	Призначення та обмеження
id	BIGINT	Сурогатний первинний ключ
tournament_id	BIGINT	Зовнішній ключ на турнір; ON DELETE CASCADE
assigned_judge_id	BIGINT, NULL	Зовнішній ключ на закріпленого суддю; ON DELETE SET NULL
number	SMALLINT, CHECK ≥ 0	Порядковий номер майданчика в межах турніру
name	VARCHAR(50)	Необов'язкова назва майданчика
current_match_id	BIGINT, NULL	Зовнішній ключ на поточний поєдинок; ON DELETE SET NULL
active_results_category_id	BIGINT, NULL	Категорія, результати якої виводяться на табло; ON DELETE SET NULL
is_active	BOOLEAN	Ознака задіяності майданчика
scoreboard_ip	INET, NULL	IP-адреса апаратного табло
scoreboard_connected	BOOLEAN	Стан підключення табло

Структура таблиць фінансової підсистеми

Таблиця	Поле	Тип даних	Призначення та обмеження
billing_payment_invoice	id	BIGINT	Сурогатний первинний ключ
billing_payment_invoice	user_id	BIGINT	Зовнішній ключ на платника; ON DELETE CASCADE
billing_payment_invoice	invoice_id	VARCHAR(100), UNIQUE, NULL	Ідентифікатор інвойсу у Monobank
billing_payment_invoice	payment_type	VARCHAR(20)	Тип оплати: registrations, platform_fee
billing_payment_invoice	amount	INTEGER, CHECK ≥ 0	Сума до сплати (UAH)
billing_payment_invoice	status	VARCHAR(20)	Стан: pending, paid, expired, failed
billing_payment_invoice	tournament_id	BIGINT, NULL	Зовнішній ключ на турнір; ON DELETE SET NULL
billing_payment_invoice	payment_url	VARCHAR(500)	Посилання на сторінку оплати
billing_payment_invoice	created_at, updated_at	TIMESTAMPTZ	Моменти створення та оновлення
billing_transaction	id	BIGINT	Сурогатний первинний ключ
billing_transaction	invoice_id	BIGINT, NULL	Зовнішній ключ на рахунок; ON DELETE SET NULL
billing_transaction	user_id	BIGINT	Зовнішній ключ на платника; ON DELETE CASCADE
billing_transaction	payment_type	VARCHAR(20)	Тип оплати
billing_transaction	transaction_type	VARCHAR(20)	Тип транзакції: payment, refund
billing_transaction	amount	INTEGER	Сума (UAH); від'ємна при поверненні
billing_transaction	method	VARCHAR(20)	Спосіб: online, offline

Продовження таблиці Ж.8

Таблиця	Поле	Тип даних	Призначення та обмеження
billing_transaction	reference	VARCHAR(100), UNIQUE	Унікальний ідентифікатор транзакції
billing_transaction	monobank_receipt_id	VARCHAR(100)	Ідентифікатор квитанції Monobank
billing_transaction	receipt_url	VARCHAR(500)	Посилання на офіційну квитанцію
billing_transaction	requires_refund	BOOLEAN	Ознака конфлікту оплати (подвійний платіж)
billing_transaction	created_at	TIMESTAMPTZ	Момент транзакції
billing_payout_request	id	BIGINT	Сурогатний первинний ключ
billing_payout_request	tournament_id	BIGINT	Зовнішній ключ на турнір; ON DELETE CASCADE
billing_payout_request	organizer_id	BIGINT	Зовнішній ключ на організатора; ON DELETE CASCADE
billing_payout_request	amount	INTEGER, CHECK ≥ 0	Сума виплати (UAH)
billing_payout_request	iban	VARCHAR(34)	IBAN отримувача
billing_payout_request	recipient_name, recipient_code, purpose	VARCHAR	Реквізити та призначення платежу
billing_payout_request	status	VARCHAR(20)	Стан: pending, completed, rejected
billing_payout_request	created_at, updated_at	TIMESTAMPTZ	Моменти створення та оновлення

Структура таблиць підсистеми ідентифікації

Таблиця	Поле	Тип даних	Призначення та обмеження
auth_user	id	BIGINT	Сурогатний первинний ключ
auth_user	email	VARCHAR(254), UNIQUE	Адреса пошти – основний ідентифікатор для входу
auth_user	password	VARCHAR(128)	Хеш пароля
auth_user	first_name, last_name, patronymic	VARCHAR(150)	Прізвище, ім'я, по-батькові
auth_user	role	VARCHAR(50)	Глобальна роль: organizer, coach, judge, admin, spectator, staff
auth_user	club_id	BIGINT, NULL	Зовнішній ключ на клуб; ON DELETE SET NULL
auth_user	phone, gender, birth_date, skill_level, referee_category	VARCHAR / DATE	Анкетні відомості та кваліфікація
auth_user	email_verified	BOOLEAN	Чи підтверджено адресу пошти
auth_user	name_locked	BOOLEAN	Заборона самостійної зміни ПІБ
auth_user	is_club_leader	BOOLEAN	Атрибут керівника клубу (розширені фінансові права)
auth_user	credit_limit	INTEGER	Кредитний ліміт комісійної заборгованості (за замовчуванням 1000)
auth_user	is_banned, ban_reason	BOOLEAN, TEXT	Стан блокування та його причина
auth_user	is_active, is_staff	BOOLEAN	Технічні прапори активності та доступу до службової панелі
auth_user	date_joined	TIMESTAMP TZ	Момент реєстрації
club	id	BIGINT	Сурогатний первинний ключ

Муравицький В. Ю.

Толстой І. А.

Арк.

ІПЗ.КР.Б – 121 – 26 – ПЗ

142

Змн.

Арк.

№ докум.

Підпис

Дата

Продовження таблиці Ж.9

Таблиця	Поле	Тип даних	Призначення та обмеження
club	name	VARCHAR(200), UNIQUE	Назва клубу або федерації
club	region	VARCHAR(150)	Регіон з переліку областей України
role_request	id	BIGINT	Сурогатний первинний ключ
role_request	user_id	BIGINT	Зовнішній ключ на заявника; ON DELETE CASCADE
role_request	requested_role	VARCHAR(50)	Бажана роль
role_request	club_id	BIGINT, NULL	Клуб для ролі тренера; ON DELETE SET NULL
role_request	referee_category	VARCHAR(100)	Суддівська категорія
role_request	status	VARCHAR(20)	Стан: pending, approved, rejected
role_request	details	TEXT	Додаткові відомості про кваліфікацію
role_request	document, photo_with_id	VARCHAR	Шляхи до підтвердних файлів
role_request	reviewed_by_id	BIGINT, NULL	Зовнішній ключ на рецензента; ON DELETE SET NULL
role_request	created_at, reviewed_at	TIMESTAMPTZ	Моменти подання та розгляду
email_confirmation_code / password_reset_code	user_id	BIGINT, UNIQUE	Зовнішній ключ «один до одного» на користувача; ON DELETE CASCADE
email_confirmation_code / password_reset_code	code	VARCHAR(6)	Одноразовий код
email_confirmation_code / password_reset_code	created_at, expires_at	TIMESTAMPTZ	Момент створення та термін дії коду