ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Проєктування та розробка веб-застосунку для відстеження кіберспортивних турнірів та ігрових новин»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Технології цифрового розвитку»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Андрій АНЧЕВ
(підпис)

Виконав: здобувач(ка) вищої освіти групи ТЦР-43
Андрій АНЧЕВ

Керівник: Вадим ЧИТУЛЯН
Старший викладач

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Технологій цифрового розвитку

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Технології цифрового розвитку»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Технологій цифрового розвитку
_____ Вікторія ЖЕБКА
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Анчеву Андрію Олексійовиу

1. Тема кваліфікаційної роботи: «Проєктування та розробка веб-застосунку для відстеження кіберспортивних турнірів та ігрових нови», керівник кваліфікаційної роботи Старший викладач Вадим ЧИТУЛЯН, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року №51.

2. Строк подання кваліфікаційної роботи «18» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: інформаційні ресурси та програмні компоненти системи розпізнавання зображень, а також модулі обробки та відображення даних у середовищі програмування Python і JavaScript, механізми взаємодії з API кіберспортивних платформ для отримання інформації про матчі, турніри та новини.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області та постановка задачі створення веб-застосунку.
2. Огляд існуючих платформ і технологій збору та обробки даних
3. Розробка, програмна реалізація, тестування та оцінювання ефективності роботи веб-застосунку для відстеження турнірів і новин у сфері кіберспорту.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Алгоритм роботи застосунку.
6. Діаграма класів.
7. Екранні форми.
8. Апробація результатів дослідження

6. Дата видачі завдання «20» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та огляд сучасних веб-технологій і платформ для відстеження кіберспортивних турнірів та ігрових новин	20.02.2026-15.03.2026	виконано
2	Вивчення технічної документації API кіберспортивних сервісів, веб-фреймворків та мов програмування Python і JavaScript для розробки веб-застосунку	16.03.2026 - 07.04.2026	виконано
3	Розробка архітектури програмної системи: модулі збору даних, обробки інформації, зберігання та взаємодії з користувацьким інтерфейсом	08.04.2026 - 21.04.2026	виконано
4	Програмна реалізація веб-застосунку із використанням сучасних фреймворків та бібліотек для відображення турнірів і новин кіберспорту	22.04.2026 - 06.05.2026	виконано
5	Реалізація та налаштування механізмів обробки даних, тестування системи на різних сценаріях використання	06.05.2026 - 13.05.2026	виконано
6	Підготовка демонстраційних матеріалів та попередній захист роботи	14.05.2026 - 17.05.2026	виконано
7	Попередній захист роботи	18.05.2026	виконано

Здобувач(ка) вищої освіти _____ Андрій АНЧЕВ
(підпис)

Керівник кваліфікаційної роботи _____ Вадим ЧИТУЛЯН
(підпис

РЕФЕРАТ

Текстова частина бакалаврської роботи: 61 сторінки, 26 рисунків, 3 таблиці, 29 джерел.

Мета роботи – розробка веб-застосунку Esports Tracker, який забезпечує зручний доступ до інформації про кіберспортивні матчі, турніри та ігрові новини, а також надає можливість персоналізованого відстеження подій користувачами.

Об'єкт дослідження – інформаційна система моніторингу кіберспортивних подій та новин ігрової індустрії.

Предмет дослідження – методи проєктування та розробки веб-застосунків для відстеження кіберспортивних матчів, турнірів і новин з використанням сучасних веб-технологій та зовнішніх API.

Короткий зміст роботи: у роботі використано аналіз літературних і технічних джерел для дослідження сучасних підходів до розробки вебзастосунків, систем моніторингу кіберспортивних подій та інтеграції зовнішніх API. Застосовано методи системного аналізу та проєктування, що дали змогу змоделювати структуру вебзастосунку, визначити архітектуру клієнтської та серверної частин, а також спроєктувати взаємодію між ними. У процесі розробки використано методи конфігурації та програмної реалізації, зокрема для створення серверної частини на платформі Node.js із використанням Express і бази даних MongoDB, а також для реалізації клієнтського інтерфейсу засобами HTML, CSS і JavaScript.

Сферою використання застосунку є інформаційний супровід користувачів у процесі відстеження кіберспортивних подій та взаємодії з персоналізованим функціоналом вебплатформи.

КЛЮЧОВІ СЛОВА: КІБЕРСПОРТ, ВЕБ-ЗАСТОСУНОК, ESPORTS TRACKER, NODE.JS, EXPRESS, MONGODB, PANDA SCORE API, MATCH TRACKING, ТУРНІРИ, ІГРОВІ НОВИНИ, WEB-РОЗРОБКА

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасних умовах стрімкого розвитку цифрових технологій та індустрії кіберспорту спостерігається значне зростання кількості турнірів, команд, гравців і інформаційних ресурсів, присвячених ігровим подіям. Кіберспорт перетворився на глобальне явище, яке об'єднує мільйони користувачів по всьому світу та формує нові підходи до споживання інформації й організації дозвілля. Водночас велика кількість розрізнених джерел даних ускладнює оперативний доступ до актуальної інформації про матчі, результати, новини та аналітику.

Існуючі веб-ресурси та платформи часто не забезпечують достатнього рівня зручності, персоналізації та інтеграції даних, що створює потребу у розробці ефективних інформаційних систем для централізованого моніторингу кіберспортивних подій. Особливо актуальним є створення веб-застосунків, які поєднують функції збору, обробки та візуалізації інформації в реальному часі з можливістю адаптації під індивідуальні потреби користувачів.

Актуальність даного дослідження зумовлена необхідністю розробки сучасного веб-застосунку для відстеження кіберспортивних турнірів та ігрових новин, який забезпечить зручний доступ до структурованої інформації, підвищить ефективність її сприйняття та сприятиме розвитку цифрових сервісів у сфері кіберспорту.

Мета роботи – розробка веб-застосунку Esports Tracker, який забезпечує зручний доступ до інформації про кіберспортивні матчі, турніри та ігрові новини, а також надає можливість персоналізованого відстеження подій користувачами.

Для досягнення мети, у бакалаврській роботі успішно виконано наступні завдання:

1. Провести аналіз предметної області кіберспорту та визначити основні вимоги до систем відстеження кіберспортивних турнірів і новин.
2. Дослідити існуючі веб-платформи та інформаційні сервіси у сфері кіберспорту, визначити їх переваги та недоліки.

3. Проаналізувати сучасні технології та інструменти веб-розробки (фреймворки, бібліотеки, API) для створення інтерактивних інформаційних систем.

4. Розробити архітектуру веб-застосунку для збору, обробки та відображення інформації про кіберспортивні події.

5. Реалізувати програмний прототип веб-застосунку із використанням сучасних технологій клієнтської та серверної розробки.

6. Забезпечити інтеграцію із зовнішніми джерелами даних (API кіберспортивних платформ) для отримання актуальної інформації про турніри та новини.

7. Провести тестування функціональних можливостей веб-застосунку та оцінити ефективність його роботи.

Об'єкт дослідження – інформаційна система моніторингу кіберспортивних подій та новин ігрової індустрії.

Предмет дослідження – методи проєктування та розробки веб-застосунків для відстеження кіберспортивних матчів, турнірів і новин з використанням сучасних веб-технологій та зовнішніх API.

Розробити рекомендації щодо подальшого вдосконалення системи та розширення її функціональних можливостей.

Методи дослідження:

1. Аналіз літературних та технічних джерел – дослідження сучасних підходів до створення веб-застосунків, систем моніторингу кіберспортивних подій та використання зовнішніх API.

2. Методи системного аналізу та проєктування – моделювання структури веб-застосунку, визначення архітектури клієнтської та серверної частин.

3. Методи конфігурації та програмної реалізації – розробка серверної частини на платформі Node.js із використанням Express та бази даних MongoDB, реалізація клієнтського інтерфейсу за допомогою HTML, CSS та JavaScript.

4. Експериментальні методи – тестування функціонування веб-застосунку, перевірка коректності роботи API-запитів, авторизації користувачів та системи персональних улюблених подій.

Методи дослідження. Під час написання бакалаврської кваліфікаційної роботи були використані методи теоретичного дослідження для збору, обробки та відображення інформації про кіберспортивні турніри і новини. Для забезпечення зручної взаємодії користувача із системою застосовано веб-інтерфейс, створений із використанням сучасних бібліотек та технологій, що забезпечує наочне відображення даних і можливість керування процесом перегляду інформації.

Наукова новизна одержаних результатів. Реалізована система моніторингу кіберспортивних подій дозволяє отримувати та обробляти інформацію у режимі реального часу, відображати результати матчів, розклад турнірів та актуальні новини у зручному для користувача форматі. Це забезпечує можливість адаптації веб-застосунку до різних умов використання та розширення його функціональних можливостей.

Практична значущість одержаних результатів. Результати роботи можуть бути використані у навчальному процесі для демонстрації принципів розробки веб-застосунків та інформаційних систем, у дослідницьких проєктах, пов’язаних із аналізом даних, а також у практичних задачах створення сучасних цифрових сервісів. Розроблений веб-застосунок демонструє перспективність використання веб-технологій як ефективного інструменту для автоматизації процесів збору, аналізу та представлення інформації у сфері кіберспорту.

Апробацію результатів кваліфікаційної роботи здійснено шляхом публікації тез доповідей у матеріалах Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація» (Збірник тез. – К.: ДУІКТ, 2026):

1. Анчев А. О. Розробка та оптимізація веб-платформи для агрегації даних про кіберспортивні турніри та ігрові події // Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація». Збірник тез. – К.: ДУІКТ, 2026.(подано до друку)

2. Анчев А. О. Інформаційна система підтримки користувачів у кіберспортивному середовищі з функціями моніторингу турнірів та новин // Всеукраїнська науково-технічна конференція «Технології цифрового розвитку:

програмні системи та децентралізація». Збірник тез. – К.: ДУІКТ, 2026.(подано до друку)

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-СИСТЕМ ДЛЯ МОНІТОРИНГУ КІБЕРСПОРТИВНИХ ПОДІЙ

1.1 Розвиток кіберспорту та цифрових платформ для відстеження турнірів

У сучасному цифровому середовищі кіберспорт набуває стрімкого розвитку та поступово трансформується з нішевого виду розваг у повноцінну індустрію з багатомільйонною аудиторією. Зростання кількості кіберспортивних дисциплін, турнірів та професійних команд сприяє активному формуванню спеціалізованих інформаційних платформ, призначених для відстеження результатів матчів, розкладу подій і новин ігрової індустрії. Разом із цим підвищується потреба користувачів у швидкому доступі до актуальної та структурованої інформації, що обумовлює необхідність дослідження розвитку кіберспорту та аналізу сучасних цифрових платформ для моніторингу кіберспортивних подій.

Кіберспорт (англ. esports) — це вид змагальної діяльності, що базується на використанні комп'ютерних ігор, у якому окремі гравці або команди змагаються між собою відповідно до встановлених правил. На відміну від традиційних відеоігор, кіберспорт передбачає професійний підхід, наявність тренувального процесу, участь у турнірах різного рівня та отримання фінансових винагород. Сучасний кіберспорт є складовою цифрової економіки та включає не лише гравців, а й організаторів турнірів, спонсорів, стрімінгові платформи та широку аудиторію глядачів.

Розвиток кіберспорту розпочався ще у 1970–1980-х роках із появою перших ігрових турнірів, однак найбільшого поширення він набув у 1990-х роках разом із розвитком персональних комп'ютерів та локальних мереж. Саме в цей період з'явилися перші популярні дисципліни, такі як Quake, StarCraft та Counter-Strike, які стали основою для формування кіберспортивних спільнот. Подальший розвиток інтернет-технологій у 2000-х роках сприяв переходу від локальних змагань до міжнародних турнірів, що значно розширило аудиторію кіберспорту [1]. У 2010-х

роках кіберспорт досяг нового рівня завдяки розвитку стрімінгових платформ, таких як Twitch і YouTube, що дозволило мільйонам користувачів у реальному часі спостерігати за матчами. Сьогодні кіберспорт є глобальною індустрією з великими призовими фондами, професійними лігами та постійним зростанням популярності серед користувачів у всьому світі.

Сучасний кіберспорт представлений широким спектром ігрових дисциплін, серед яких особливої популярності набули такі проєкти, як Counter-Strike 2 (CS2) / Counter-Strike: Global Offensive (CS:GO), Dota 2, League of Legends (LoL) та Valorant. Кожна з цих дисциплін має власну специфіку, ігрову механіку, аудиторію та систему проведення змагань, що формує унікальну екосистему кіберспорту [2].

Counter-Strike 2 (CS2), що є продовженням легендарної серії Counter-Strike, а також її попередня версія CS:GO, належать до жанру командних шутерів від першої особи (FPS) [3]. У цих іграх дві команди — терористи та контртерористи — змагаються між собою у виконанні або запобіганні виконанню певних завдань, таких як встановлення або знешкодження вибухового пристрою. CS:GO протягом багатьох років залишалася однією з найпопулярніших кіберспортивних дисциплін у світі, завдяки простим правилам, динамічному геймплею та високому рівню змагальності. З виходом CS2 гра отримала оновлений рушій, покращену графіку та вдосконалені механіки, що ще більше підвищило її популярність серед гравців і глядачів. Турніри з Counter-Strike, зокрема серії Major, збирають мільйони глядачів та мають значні призові фонди (рис. 1.1).

Рис. 1.1 Афіша Counter-Strike 2

Dota 2 є однією з найвідоміших ігор жанру MOBA (Multiplayer Online Battle Arena), у якій дві команди по п’ять гравців змагаються за знищення головної будівлі супротивника (рис. 1.2). Особливістю Dota 2 є глибока стратегічна складова, велика кількість персонажів (героїв) з унікальними здібностями та необхідність командної взаємодії [4]. Гра відома своїм найбільшим у світі кіберспорту турніром — The International, призовий фонд якого формується частково за рахунок спільноти гравців і часто перевищує десятки мільйонів доларів. Це робить Dota 2 однією з найбільш фінансово привабливих дисциплін у кіберспорті.

Рис. 1.2 Афіша Dota 2

League of Legends (LoL) також належить до жанру MOBA і є прямим конкурентом Dota 2. Гра вирізняється більш динамічним геймплеєм, доступністю для нових гравців та добре розвиненою екосистемою професійних ліг. Riot Games активно розвиває кіберспортивну складову LoL, організовуючи регіональні чемпіонати та світовий турнір World Championship, який щороку збирає величезну аудиторію (рис. 1.3). Особливістю LoL є високий рівень організації турнірів та стабільність професійної сцени, що робить її однією з найвпливовіших дисциплін у кіберспорті.

Рис. 1.3 League of Legends

Valorant — це відносно нова, але вже надзвичайно популярна кіберспортивна дисципліна, яка поєднує елементи тактичного шутера та гри з унікальними персонажами (агентами), кожен з яких має власні здібності. Гра була розроблена компанією Riot Games і швидко здобула популярність завдяки поєднанню точності стрільби, командної взаємодії та використання спеціальних умінь. Valorant активно розвиває свою кіберспортивну сцену через міжнародні турніри серії Valorant Champions Tour, що сприяє залученню нових гравців і глядачів (рис. 1.4).

Рис. 1.4 Valorant

Зазначені ігри формують основу кіберспорту, забезпечуючи різноманітність ігрових форматів та залучаючи широку аудиторію користувачів. Їх популярність

сприяє розвитку інформаційних платформ, які дозволяють ефективно відстежувати результати матчів, турніри та новини, що є важливим аспектом у контексті даного дослідження.

Однією з характеристик кіберспорту є його масштабність, яка проявляється у значній кількості міжнародних турнірів, високих призових фондах та багатомільйонній аудиторії глядачів. Щорічно проводяться сотні кіберспортивних змагань різного рівня — від локальних онлайн-турнірів до глобальних чемпіонатів, що об'єднують найсильніші команди світу. Найбільш престижні турніри, такі як The International з Dota 2, CS Major з Counter-Strike або World Championship з League of Legends, характеризуються високим рівнем організації та значними фінансовими винагородами. Призові фонди окремих змагань можуть сягати десятків мільйонів доларів, що свідчить про економічну значущість кіберспорту як індустрії [4].

Показником розвитку кіберспорту є стрімке зростання аудиторії. Кількість глядачів кіберспортивних подій постійно збільшується, включаючи як активних фанатів, так і випадкових користувачів, які цікавляться ігровою індустрією. За оцінками аналітичних компаній, глобальна аудиторія кіберспорту налічує сотні мільйонів людей, що робить його одним із найпопулярніших напрямів цифрових розваг. Така популярність стимулює розвиток нових інформаційних сервісів, спрямованих на оперативне інформування користувачів про події кіберспорту.

Завдяки глобальній мережі стало можливим формування міжнародної кіберспортивної спільноти та організація турнірів без географічних обмежень. Інтернет також виступає основою для функціонування різноманітних веб-застосунків і платформ, які надають користувачам доступ до статистики, новин і трансляцій.

Стрімінгових платформи, таких як Twitch і YouTube, забезпечують можливість трансляції матчів у режимі реального часу, взаємодії з аудиторією через чат, а також збереження відеоконтенту для подальшого перегляду. Стрімінгові сервіси сприяють популяризації кіберспорту, розширенню його аудиторії та залученню нових користувачів. Крім того, вони виконують роль інформаційних центрів, де користувачі можуть отримувати актуальні новини,

аналітику та огляди матчів.

Незважаючи на стрімкий розвиток кіберспорту та зростання кількості інформаційних ресурсів, суттєвою проблемою залишається фрагментованість даних. Інформація про кіберспортивні турніри, результати матчів, новини та статистику часто розміщується на різних платформах і сервісах, що ускладнює її пошук та аналіз для користувачів. Для отримання повної картини подій користувач змушений звертатися до декількох джерел одночасно, що знижує зручність використання інформаційних ресурсів і потребує додаткових витрат часу. Відсутність єдиного централізованого рішення, яке б поєднувало всі необхідні функції в одному інтерфейсі, обумовлює необхідність розробки сучасних веб-застосунків для агрегування, обробки та зручного представлення інформації про кіберспортивні події.

1.2 Аналіз сучасних веб-технологій для створення інформаційних систем

Веб-застосунок — це програмна система, що функціонує в середовищі веб-браузера та забезпечує взаємодію користувача з інформаційними ресурсами через мережу Інтернет. На відміну від традиційних настільних програм, веб-застосунки не потребують встановлення на пристрій користувача, оскільки їх основна логіка розміщується на сервері, а доступ до функціоналу здійснюється через веб-інтерфейс. Це забезпечує універсальність використання незалежно від операційної системи чи типу пристрою.

Сучасні веб-застосунки характеризуються високим рівнем інтерактивності, швидкістю обробки даних та можливістю роботи в режимі реального часу. Вони дозволяють користувачам виконувати широкий спектр дій, таких як перегляд інформації, взаємодія з контентом, авторизація, обмін даними та персоналізація сервісів. Завдяки використанню сучасних веб-технологій, веб-застосунки можуть забезпечувати функціональність, наближену до повноцінних програмних систем.

Особливістю веб-застосунків є їх клієнт-серверна архітектура, у якій обробка даних розподіляється між клієнтською частиною (frontend) та серверною частиною

(backend). Такий підхід дозволяє ефективно організувати взаємодію між користувачем і системою, забезпечити масштабованість та гнучкість розробки. Саме завдяки цим характеристикам веб-застосунки широко використовуються у різних сферах, зокрема для створення інформаційних систем, платформ моніторингу та онлайн-сервісів.

У контексті даної роботи веб-застосунок розглядається як основний інструмент для реалізації системи відстеження кіберспортивних турнірів та ігрових новин, що забезпечує доступ до актуальної інформації, її обробку та зручне відображення для кінцевого користувача. Функціонування веб-застосунків базується на використанні комплексу технологій, які умовно поділяються на клієнтську та серверну частини, а також системи зберігання даних і засоби взаємодії із зовнішніми сервісами.

Клієнтська частина (frontend) відповідає за відображення інформації та взаємодію з користувачем. Основними технологіями frontend-розробки є HTML, CSS та JavaScript. HTML (HyperText Markup Language) використовується для створення структури веб-сторінки та визначення її елементів. CSS (Cascading Style Sheets) забезпечує оформлення та візуальне представлення інтерфейсу, дозволяючи створювати зручний і привабливий дизайн. JavaScript є мовою програмування, що забезпечує динамічну поведінку сторінок, обробку подій, виконання запитів до серверу та оновлення даних без перезавантаження сторінки [5]. Сукупність цих технологій дозволяє створювати інтерактивні та адаптивні веб-інтерфейси.

Серверна частина (backend) виконує обробку запитів, взаємодію з базою даних та реалізацію бізнес-логіки застосунку. У даній роботі для реалізації серверної частини використовується середовище Node.js, яке дозволяє виконувати JavaScript на стороні сервера та забезпечує високу продуктивність при обробці великої кількості запитів. Для побудови веб-серверу застосовується фреймворк Express, який спрощує створення маршрутів (routes), обробку HTTP-запитів та організацію структури серверного застосунку. Використання Node.js у поєднанні з Express дозволяє ефективно реалізувати серверну логіку та забезпечити швидку взаємодію з клієнтською частиною.

База даних, використовується для зберігання інформації про користувачів, налаштування та інші дані системи. У даній роботі використовується MongoDB — нереляційна (NoSQL) база даних, яка зберігає дані у вигляді документів у форматі JSON-подібних структур [6]. Такий підхід забезпечує гнучкість структури даних, високу швидкість роботи та зручність інтеграції з JavaScript-середовищем. MongoDB добре підходить для веб-застосунків, що працюють із динамічними даними та потребують масштабованості.

API (Application Programming Interface) — інтерфейси програмування застосунків, які дозволяють різним системам взаємодіяти між собою. API забезпечують отримання даних із зовнішніх сервісів, зокрема інформації про кіберспортивні матчі, турніри, команди та новини. У контексті даного дослідження використання API дозволяє автоматизувати процес збору актуальної інформації та інтегрувати її у веб-застосунок. Це значно спрощує розробку та забезпечує доступ до актуальних даних у режимі реального часу.

Для реалізації веб-застосунку, призначеного для відстеження кіберспортивних турнірів та ігрових новин, використовується сукупність сучасних технологій, кожна з яких виконує окрему функцію у загальній архітектурі системи. Розподіл технологій за компонентами дозволяє чітко організувати процес обробки даних, забезпечити ефективну взаємодію між клієнтською та серверною частинами, а також підвищити масштабованість і надійність застосунку. Основні технології, що використовуються у розробці веб-застосунку, наведено у таблиці 1.1.

Таблиця 1.1

Основні технології веб-застосунку

Компонент системи	Технологія	Роль у системі
Клієнтський рівень (Frontend)	HTML5, CSS3, JavaScript (ES6+)	Формування структури інтерфейсу, стилізація, динамічна взаємодія з користувачем

Продовження таблиці 1.1

Компонент системи	Технологія	Роль у системі
Серверний рівень (Backend)	Node.js, Express.js	Реалізація REST API, обробка HTTP-запитів, бізнес-логіка, маршрутизація
Рівень зберігання даних	MongoDB (NoSQL), Mongoose ODM	Зберігання профілів користувачів, обраних подій та службових даних
Інтеграційний рівень (зовнішні дані)	PandaScore API	Отримання актуальних даних про матчі, турніри, команди та статуси подій

Як видно з таблиці 1.1, для розробки веб-застосунку використовується комплекс взаємопов’язаних технологій, які забезпечують повноцінне функціонування системи. Клієнтська частина, реалізована за допомогою HTML, CSS та JavaScript, відповідає за відображення інформації та взаємодію з користувачем. Серверна частина, побудована на основі Node.js та Express, забезпечує обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних. Використання MongoDB як нереляційної бази даних дозволяє ефективно працювати з динамічними структурами даних і забезпечує гнучкість зберігання інформації. Інтеграція із зовнішнім API PandaScore дає змогу отримувати актуальні дані про кіберспортивні події в режимі реального часу.

Поєднання зазначених технологій формує єдину екосистему, яка дозволяє створити ефективний, масштабований та зручний у використанні веб-застосунок для відстеження кіберспортивних турнірів і новин.

1.3 Огляд існуючих платформ для моніторингу кіберспортивних матчів та новин

У кіберспортивному середовищі існує значна кількість онлайн-платформ, які надають інформацію про турніри, матчі, команди та новини ігрової індустрії. Найбільш відомими серед них є HLTV, Liquipedia та PandaScore, які забезпечують доступ до актуальних даних та активно використовуються гравцями, аналітиками та фанатами кіберспорту.

Платформа HLTV є однією з найпопулярніших ресурсів, присвячених дисципліні Counter-Strike. Вона надає детальну інформацію про матчі, включаючи результати, статистику гравців, рейтинги команд, а також новини та аналітичні матеріали. HLTV відома своєю глибокою статистикою та широкими можливостями для аналізу ігор, що робить її важливим інструментом для професійної сцени (рис. 1.5).

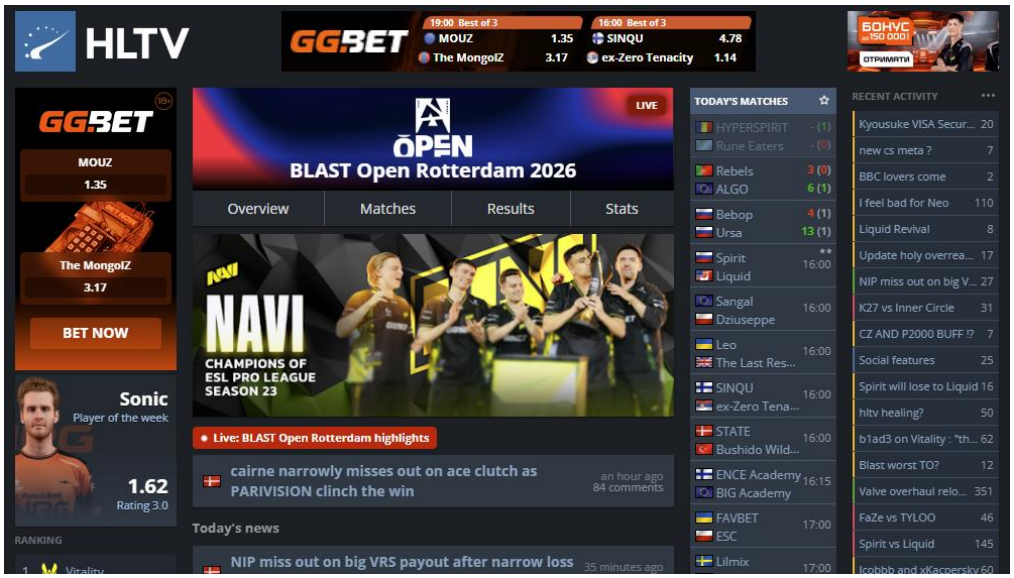


Рис. 1.5 Платформа HLTV

Liquipedia являє собою багатофункціональну енциклопедію кіберспорту, яка охоплює широкий спектр дисциплін, таких як Dota 2, League of Legends, Counter-Strike та інші (рис. 1.6). Платформа побудована у форматі wiki, що дозволяє користувачам редагувати та доповнювати інформацію. Liquipedia містить дані про турніри, команди, гравців, історію матчів і структуру змагань, що робить її

універсальним джерелом інформації [6].

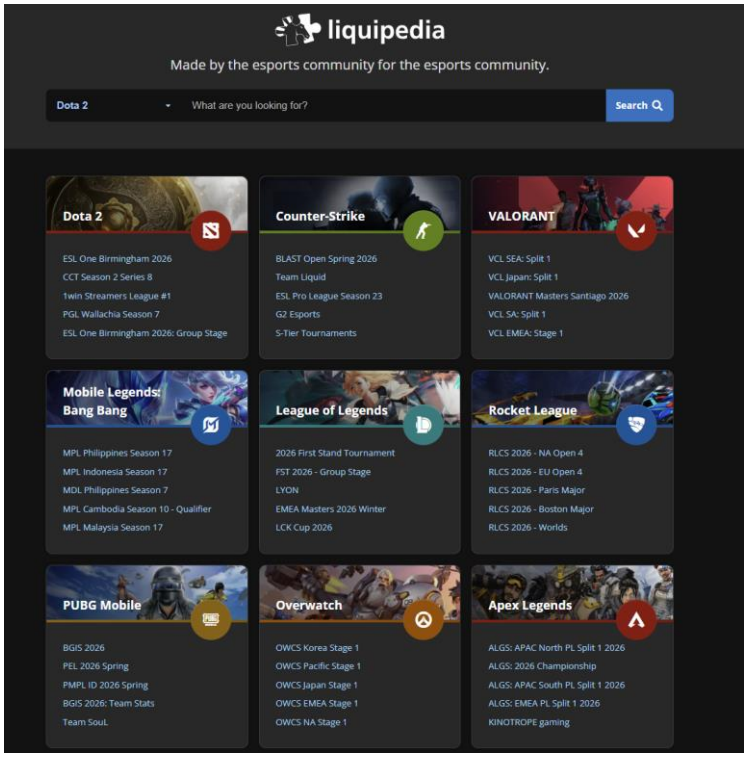


Рис. 1.6 Платформа Liquipedia

PandaScore є платформою та API-сервісом, який надає структуровані дані про кіберспортивні події. Основною перевагою PandaScore є можливість інтеграції з веб-застосунками, що дозволяє автоматично отримувати інформацію про матчі, турніри, результати та статистику [7]. Це робить її особливо корисною для розробників інформаційних систем і сервісів у сфері кіберспорту.

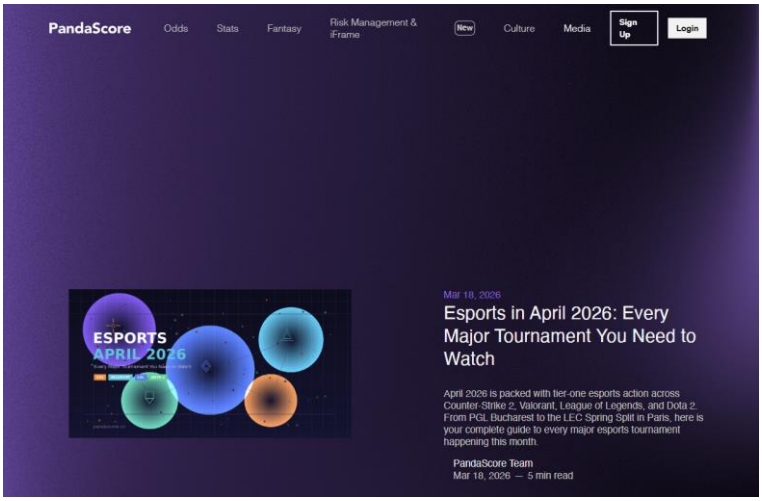


Рис. 1.7 Платформа PandaScore

Зазначені платформи виконують важливі функції, надають доступ до інформації про матчі, результати змагань, статистичні показники гравців і команд, а також публікують новини та аналітичні матеріали. Вони дозволяють користувачам відстежувати розвиток кіберспортивних подій і отримувати актуальні дані в режимі реального часу.

Незважаючи на значну функціональність, існуючі платформи мають низку недоліків. Однією з основних проблем є відсутність достатнього рівня персоналізації, що обмежує можливості користувачів адаптувати інформацію відповідно до власних потреб. Крім того, деякі сервіси мають складний або перевантажений інтерфейс, що ускладнює навігацію та знижує зручність використання. Також важливою проблемою є відсутність єдиного рішення, яке б об'єднувало всі необхідні функції — від перегляду матчів і статистики до новин — в одному зручному інтерфейсі. Для більш детального аналізу можливостей існуючих платформ та визначення їхніх переваг і недоліків доцільно провести їх порівняння за ключовими критеріями. У табл. 1.2 наведено зіставлення функціональних можливостей популярних кіберспортивних сервісів та розробленого веб-застосунку Esports Tracker.

Таблиця 1.2

Порівняння існуючих платформ та розробленого веб-застосунку Esports Tracker

Критерій	HLTV	Liquipedia	PandaScore	Esports Tracker (розроблений)
Відображення матчів	Так	Так	Так (через API)	Так
Турніри	Так	Так	Так	Так
Статистика	Так (детальна)	Так	Так	Так
Новини	Так	Так	Ні	Ні
Підтримка різних дисциплін	Обмежено	Так	Так	Так
Персоналізація	Ні	Ні	Ні	Так
Інтуїтивний інтерфейс	Середній	Складний	Відсутній UI	Так
Єдина система (all-in-one)	Ні	Ні	Ні	Так
Інтеграція API	Ні	Ні	Ні	Так
Можливість розширення	Обмежена	Середня	Висока	Висока

Як видно з таблиці 1.2, існуючі платформи забезпечують базовий функціонал для відстеження кіберспортивних подій, однак мають певні обмеження. Зокрема, HLTV орієнтована переважно на одну дисципліну, Liquipedia має складну структуру навігації, а PandaScore не надає власного користувацького інтерфейсу, оскільки є API-сервісом. Усі розглянуті платформи не забезпечують достатнього рівня персоналізації та не об’єднують усі необхідні функції в одному середовищі [8].

На відміну від них, розроблений веб-застосунок Esports Tracker поєднує функціональність відображення матчів, турнірів, статистики та новин із можливістю персоналізації та зручним інтерфейсом. Це дозволяє створити єдину інформаційну систему, орієнтовану на потреби користувача та здатну забезпечити більш ефективний доступ до кіберспортивної інформації.

Аналіз існуючих платформ свідчить про наявність потреби у створенні веб-застосунку, який би поєднував у собі переваги наявних сервісів, усуваючи їх недоліки та забезпечуючи зручний, інтуїтивно зрозумілий і персоналізований доступ до інформації про кіберспортивні події.

Висновок до розділу 1

Розвиток кіберспорту підтверджує його перехід у масштабну цифрову індустрію з високою динамікою турнірів, зростанням аудиторії та посиленням ролі онлайн-сервісів. Популярність дисциплін на кшталт Counter-Strike, Dota 2, League of Legends і Valorant, а також вплив стрімінгових платформ сформували сталий попит на системи, що дають швидкий доступ до розкладу, результатів, статистики й новин. Водночас виокремлено ключову проблему предметної області: інформація розпорошена між різними ресурсами, через що користувачеві складно отримувати цілісну картину подій в одному місці.

Аналіз технологічної бази показав доцільність клієнт-серверного підходу для реалізації інформаційної вебсистеми моніторингу. Поєднання HTML/CSS/JavaScript на фронтенді, Node.js та Express на бекенді, MongoDB для

зберігання даних і зовнішнього API (PandaScore) для наповнення актуальною інформацією забезпечує функціональність, масштабованість і гнучкість рішення. Обраний стек є логічним для задачі оперативного відстеження кіберспортивних подій, оскільки підтримує інтерактивний інтерфейс, швидку обробку запитів і роботу з динамічними даними.

Огляд існуючих платформ (HLTV, Liquipedia, PandaScore) засвідчив, що кожна з них має сильні сторони, але жодна не закриває комплексно всі потреби користувача в одному інтерфейсі. Виявлені обмеження стосуються передусім персоналізації, зручності навігації та формату “all-in-one”, де об’єднані матчі, турніри, статистика й новини. Це обґрунтовує необхідність розробки Esports Tracker як єдиної інтегрованої системи, що поєднує переваги наявних підходів і усуває їх практичні недоліки.

2 ПРОЄКТУВАННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ВІДСТЕЖЕННЯ КІБЕРСПОРТИВНИХ ТУРНІРІВ

2.1 Аналіз функціональних вимог до веб-застосунку Esports Tracker

На етапі проєктування веб-застосунку важливим завданням є визначення його функціональних вимог, оскільки саме вони формують основу майбутньої структури системи, її модулів та логіки взаємодії з користувачем. Для веб-застосунку Esports Tracker, призначеного для відстеження кіберспортивних турнірів та ігрових новин, функціональні вимоги визначають перелік основних можливостей, які система повинна забезпечувати для задоволення потреб користувачів.

Головною метою застосунку є надання користувачеві зручного інструменту для отримання актуальної інформації про кіберспортивні події в єдиному інформаційному середовищі. Це передбачає реалізацію функцій перегляду матчів, турнірів, ігрових новин, отримання детальної інформації про події, а також можливість персоналізації взаємодії із системою. У зв’язку з цим функціональні вимоги повинні враховувати як базові інформаційні потреби користувача, так і додаткові можливості, що підвищують зручність використання веб-застосунку.

До основних функціональних вимог веб-застосунку Esports Tracker доцільно віднести можливість перегляду списку актуальних кіберспортивних матчів із зазначенням команд, часу проведення, статусу події та результатів. Окремою вимогою є забезпечення доступу до інформації про турніри, включаючи їх назву, формат, дату проведення та перелік учасників. Не менш важливою є функція відображення новин ігрової індустрії, що дозволяє користувачам оперативно отримувати актуальні відомості про події у світі кіберспорту.

Крім базових інформаційних можливостей, система повинна підтримувати функції авторизації та автентифікації користувачів, що дозволить реалізувати персоналізовану взаємодію з веб-застосунком. Зокрема, авторизований користувач

повинен мати можливість зберігати обрані матчі або турніри до списку улюблених, щоб швидко отримувати доступ до подій, які його цікавлять. Такий підхід підвищує практичну цінність веб-застосунку та покращує користувацький досвід.

Додатково до функціональних вимог варто віднести можливість пошуку та фільтрації інформації за певними критеріями, наприклад за назвою турніру, дисципліною або статусом матчу. Це дозволяє скоротити час пошуку необхідних даних та зробити роботу із системою більш ефективною. Таким чином, аналіз функціональних вимог є необхідним етапом, який дозволяє визначити ключові можливості веб-застосунку та сформувати основу для його подальшого проєктування.

Наведені у таблиці 2.1 функціональні вимоги відображають базовий набір можливостей, необхідних для повноцінного функціонування веб-застосунку Esports Tracker. Їх реалізація дозволить забезпечити зручний доступ до інформації про кіберспортивні події, покращити навігацію в системі та створити умови для персоналізованого використання застосунку. Сукупність визначених функцій формує логічну основу майбутньої архітектури системи та впливає на вибір програмних засобів для її реалізації.

Таблиця 2.1

Основні функціональні вимоги веб-застосунку Esports Tracker

№	Функціональна вимога	Характеристика	Кількісний критерій (приймання)
1	Перегляд матчів	Відображення списку актуальних та майбутніх матчів	На сторінці виводиться не менше 20 подій; час завантаження списку до 2 с
2	Перегляд турнірів	Надання інформації про турніри, дати та учасників	Виведення щонайменше 10 турнірів; сортування за датою початку виконується коректно у 100% перевірок
3	Перегляд новин	Відображення новин кіберспортивної та ігрової тематики	На сторінці відображається не менше 15 новин; оновлення стрічки не рідше 1 разу на 30 хв

Продовження таблиці 2.1

4	Авторизація користувача	Реєстрація, вхід та ідентифікація користувача	Успішна реєстрація/вхід до 2 с; 100% захищених маршрутів доступні лише з валідним токеном
5	Додавання в улюблене	Збереження обраних подій для швидкого доступу	Операція додавання/видалення виконується до 1 с; зміни відображаються без перезавантаження сторінки
6	Пошук інформації	Пошук матчів, турнірів або новин за ключовими словами	Час відповіді пошуку до 1 с; релевантні результати для точного запиту - не менше 90%
7	Фільтрація даних	Відбір інформації за дисципліною, статусом або датою	Застосування фільтра до 1 с; коректність фільтрації - не менше 95% тестових сценаріїв
8	Оновлення даних	Отримання актуальної інформації із зовнішніх API	Період оновлення - кожні 30 с (для live); доступність API-даних у системі - не менше 95% часу

Визначення типів користувачів, дозволяє сформувати відповідні функціональні можливості та забезпечити різні рівні доступу до інформації залежно від потреб користувача.

У межах даного веб-застосунку доцільно виділити два основних типи користувачів:

Неавторизований користувач — це користувач, який не пройшов процедуру реєстрації або входу в систему. Такий користувач має доступ до базового функціоналу застосунку, зокрема перегляду списку матчів, турнірів та новин кіберспорту. Він може ознайомлюватися з інформацією про події, однак не має можливості зберігати дані або персоналізувати інтерфейс.

Авторизований користувач — це користувач, який пройшов процедуру реєстрації та входу в систему. Окрім базового функціоналу, він отримує розширені можливості взаємодії із застосунком. Зокрема, авторизований користувач може

додавати матчі або турніри до списку улюблених, отримувати швидкий доступ до обраних подій, а також користуватися персоналізованими функціями системи.

Виділення зазначених типів користувачів дозволяє забезпечити гнучкість функціоналу веб-застосунку та підвищити зручність його використання залежно від рівня взаємодії користувача із системою.

Отримані результати дозволяють сформувати цілісне уявлення про структуру та логіку роботи веб-застосунку, що є необхідним етапом перед розробкою його архітектури. На основі визначених вимог у наступному підрозділі буде здійснено проєктування архітектури системи та визначення взаємодії її основних компонентів.

2.2 . Архітектура веб-системи та структура програмних модулів

Архітектура розробленого веб-застосунку «Esports Tracker» побудована за клієнт-серверним принципом, що є стандартним підходом для сучасних веб-систем. Така архітектура передбачає розподіл функціональності між клієнтською частиною (frontend) та серверною частиною (backend), які взаємодіють між собою через API. Це забезпечує гнучкість, масштабованість та зручність супроводу програмного забезпечення [15, 16].

Клієнтська частина реалізована з використанням HTML, CSS та JavaScript і відповідає за відображення інтерфейсу користувача та обробку його дій. Вона забезпечує динамічне оновлення контенту без перезавантаження сторінки, використовуючи асинхронні запити до серверу. Основними функціональними компонентами клієнтської частини є модулі відображення матчів і турнірів, система вкладок, модальні вікна для перегляду деталей подій, а також модуль авторизації користувача.

Серверна частина реалізована на платформі Node.js із використанням фреймворку Express і відповідає за обробку запитів клієнта, взаємодію з базою даних та реалізацію бізнес-логіки застосунку. Сервер надає REST API для отримання інформації про матчі та турніри, а також для обробки операцій

авторизації та керування улюбленими подіями. Обмін даними між клієнтом і сервером здійснюється у форматі JSON, що забезпечує простоту інтеграції та обробки інформації.

Структура програмних модулів системи організована таким чином, щоб забезпечити розділення відповідальностей та зручність подальшої модифікації. На клієнтській стороні виділено модулі для роботи з API, відображення даних (рендеринг матчів і турнірів), управління станом інтерфейсу (вкладки, модальні вікна), а також модуль авторизації. Серверна частина містить окремі маршрути (routes) для обробки різних типів запитів, контролери для реалізації логіки обробки даних та моделі для взаємодії з базою даних.

Важливим елементом архітектури є використання токен-орієнтованої системи авторизації, яка дозволяє забезпечити захищений доступ до ресурсів системи. Після успішного входу користувача сервер генерує токен, який зберігається на клієнтській стороні та використовується для подальших запитів до API. Це дозволяє реалізувати персоналізовані функції, такі як збереження улюблених подій [17].

Обрана архітектура веб-системи забезпечує ефективну взаємодію між компонентами, високу продуктивність, масштабованість та зручність розширення функціоналу. Чітка модульна структура програмного забезпечення дозволяє легко вносити зміни та підтримувати систему в актуальному стані.

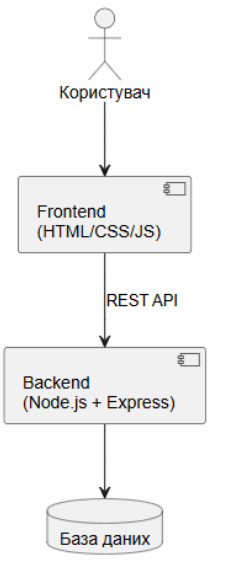


Рис. 2.1 Архітектура веб-системи «Esports Tracker»

На рисунку 2.1 подано загальну архітектуру веб-системи «Esports Tracker», яка побудована за клієнт-серверним принципом. Взаємодія користувача із системою здійснюється через браузер, у якому відображається клієнтська частина застосунку. Саме frontend відповідає за побудову інтерфейсу, відображення матчів і турнірів, перемикання вкладок, роботу модальних вікон, а також за забезпечення взаємодії користувача з функціями авторизації та персональних улюблених подій.

Клієнтська частина надсилає HTTP-запити до серверної частини через REST API. Сервер, реалізований на Node.js та Express, обробляє отримані запити, виконує перевірку даних, реалізує бізнес-логіку застосунку та формує відповіді у форматі JSON. Окремо серверна частина забезпечує механізм авторизації користувачів і перевірку токенів доступу для захищених операцій [17-20].

Зберігання та отримання даних виконується через базу даних або зовнішні джерела даних, де міститься інформація про користувачів, матчі, турніри та список обраних подій. Така архітектура забезпечує розподіл відповідальностей між компонентами системи, підвищує зручність супроводу та дозволяє легко розширювати функціональність застосунку в майбутньому.

Структура папок проєкту веб-застосунку «Esports Tracker»

Нижче наведено приклад логічної структури папок проєкту:

Для зручності розробки та супроводу програмного продукту структура проєкту була організована за модульним принципом. Це дозволило розділити логіку роботи системи на окремі функціональні блоки та спростити подальше розширення застосунку.

Кожен модуль відповідає за окрему частину функціоналу веб-застосунку, зокрема обробку даних користувачів, управління турнірами, командами, матчами та статистикою. Такий підхід забезпечує кращу читабельність коду, зменшує залежність між компонентами системи та полегшує процес тестування й налагодження окремих частин проєкту.

Окремо були виділені директорії для зберігання статичних ресурсів, конфігураційних файлів, API-запитів та компонентів інтерфейсу користувача. Це

дозволило впорядкувати структуру застосунку відповідно до сучасних практик веб-розробки та спростити командну роботу над проєктом у процесі подальшої підтримки та масштабування системи.

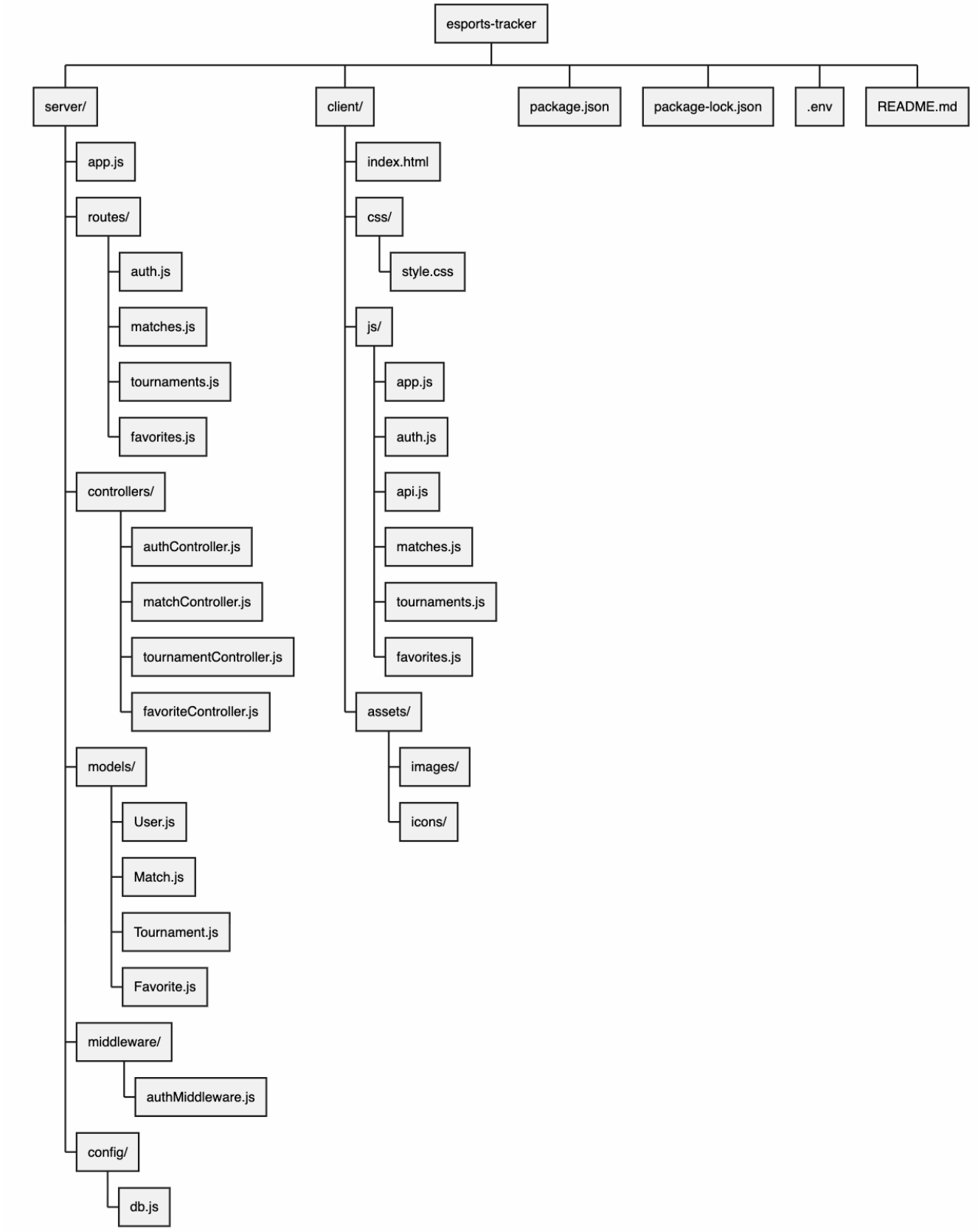


Рис. 2.2 Дерево структури проєкту esports-tracker (server/client та основні модулі)

Структура проєкту веб-застосунку організована за модульним принципом, що дозволяє чітко розмежувати функціональність клієнтської та серверної частин. У кореневій папці проєкту розміщено основні конфігураційні файли, зокрема package.json, який містить перелік залежностей та сценарії запуску застосунку, файл .env для збереження змінних середовища, а також README.md з описом проєкту [21].

Папка server містить серверну частину веб-системи. Файл app.js є головною точкою входу, де виконується ініціалізація сервера, підключення маршрутів і налаштування middleware. У папці routes зосереджено маршрути API, які відповідають за окремі функціональні напрями: авторизацію, отримання даних про матчі, турніри та роботу з улюбленими подіями. Папка controllers містить модулі бізнес-логіки, які обробляють запити та формують відповіді. У папці models розміщено моделі даних, що описують структуру сутностей системи, а middleware містить службові компоненти, наприклад перевірку токенів авторизації. Папка config використовується для налаштування підключення до бази даних.

Папка client містить клієнтську частину застосунку. Файл index.html є основною HTML-сторінкою системи, яка формує базову структуру інтерфейсу. У каталозі css розташовані файли стилів, що визначають зовнішній вигляд елементів сторінки. Каталог js об’єднує JavaScript-модулі, кожен із яких виконує окрему функцію: робота з API, авторизація, відображення матчів, турнірів та списку улюблених. Папка assets використовується для зберігання графічних ресурсів, зокрема зображень та іконок.

2.3 Проєктування бази даних та взаємодії із зовнішнім API PandaScore

Одним із ключових етапів розробки веб-додатку «Esports Tracker» є проєктування структури зберігання даних та організація взаємодії із зовнішнім джерелом інформації. Оскільки застосунок орієнтований на відображення актуальних кіберспортивних подій, матчів і турнірів, важливим завданням стало

забезпечення стабільного отримання, обробки та збереження необхідних даних. Для цього було спроєктовано базу даних застосунку та обрано зовнішній API PandaScore як джерело актуальної інформації про кіберспортивні події.

Під час проєктування бази даних було враховано особливості предметної області та функціональні потреби системи. У структурі даних необхідно було передбачити збереження інформації про користувачів, матчі, турніри, а також персональні списки улюблених подій. База даних повинна забезпечувати цілісність інформації, уникнення дублювання записів та можливість швидкого доступу до потрібних даних під час роботи застосунку.

Основною сутністю системи є користувач, для якого зберігаються облікові дані, необхідні для автентифікації та персоналізації функціоналу. До таких даних належать унікальний ідентифікатор, ім’я користувача, електронна пошта, пароль у зашифрованому вигляді, а також службова інформація, пов’язана зі створенням облікового запису. Саме ця сутність є основою для реалізації авторизації та прив’язки персональних даних до конкретного користувача [22-24].

Наступними важливими сутностями є матчі та турніри**.** Для сутності матчу доцільно зберігати ідентифікатор події, назву гри, інформацію про учасників, статус матчу, дату і час початку, а також додаткові атрибути, отримані із зовнішнього API. Для турнірів зберігаються назва, дисципліна, час проведення, список команд або учасників, а також інша доступна інформація, яка використовується у клієнтському інтерфейсі. Наявність окремих сутностей для матчів і турнірів дозволяє чітко організувати структуру даних та спростити подальшу обробку.

Для реалізації персоналізованої функції улюблених подій передбачено окрему сутність favorites, яка пов’язує користувача з конкретною подією. У такій таблиці або колекції зберігаються ідентифікатор користувача, тип події, ідентифікатор матчу або турніру, а також дата додавання до обраного. Такий підхід дозволяє користувачу формувати власний список важливих подій та швидко отримувати до них доступ.

Логічно структура даних може бути представлена у вигляді зв’язків між основними сутностями: один користувач може мати багато записів у списку улюблених, а кожен запис улюбленого пов’язаний з конкретною подією. Таким чином реалізується зв’язок типу «один-до-багатьох» між користувачем та улюбленими подіями. Подібна модель є достатньо простою, зрозумілою та ефективною для поставлених завдань.

Важливу роль у роботі системи відіграє інтеграція із зовнішнім API PandaScore. Саме цей сервіс використовується для отримання актуальної інформації про кіберспортивні матчі, турніри, команди та інші пов’язані дані. Використання зовнішнього API дозволяє не наповнювати базу даних вручну, а отримувати актуальний контент автоматично, що особливо важливо для динамічної предметної області кіберспорту.

Взаємодія із PandaScore реалізується на серверній стороні застосунку. Сервер формує HTTP-запити до зовнішнього API, передає необхідні параметри фільтрації та автентифікаційний ключ, після чого отримує відповідь у форматі JSON. Отримані дані проходять етап обробки: із загальної відповіді виділяються лише ті поля, які необхідні для роботи застосунку, після чого інформація або передається безпосередньо клієнтській частині, або за потреби зберігається у локальній базі даних.

Такий підхід має кілька переваг. По-перше, сервер виступає проміжною ланкою між клієнтом і зовнішнім сервісом, що дозволяє приховати службові ключі доступу та підвищити безпеку системи. По-друге, серверна обробка даних дозволяє уніфікувати формат відповіді для клієнтської частини, спрощуючи побудову інтерфейсу. По-третє, у разі необхідності можна реалізувати кешування або часткове локальне збереження даних, що зменшить навантаження на зовнішній API та пришвидшить роботу застосунку.

Під час проєктування взаємодії із зовнішнім API також важливо враховувати можливі помилки доступу, відсутність окремих полів у відповіді, перевищення ліміту запитів або тимчасову недоступність сервісу. Саме тому у серверній частині доцільно передбачити обробку виняткових ситуацій, перевірку валідності

отриманих даних та формування коректних повідомлень для клієнтського інтерфейсу.

Отже, проєктування бази даних та механізму взаємодії із зовнішнім API PandaScore є важливою складовою побудови веб-додатку «Esports Tracker». Раціонально організована структура даних забезпечує зручне зберігання інформації про користувачів та персональні події, а інтеграція із зовнішнім API дозволяє підтримувати актуальність контенту та підвищує функціональність системи. У сукупності це створює надійну основу для подальшої реалізації серверної та клієнтської частин застосунку.

Для наочного подання структури основних сутностей системи та зв’язків між ними доцільно використати схему бази даних веб-додатку [25].

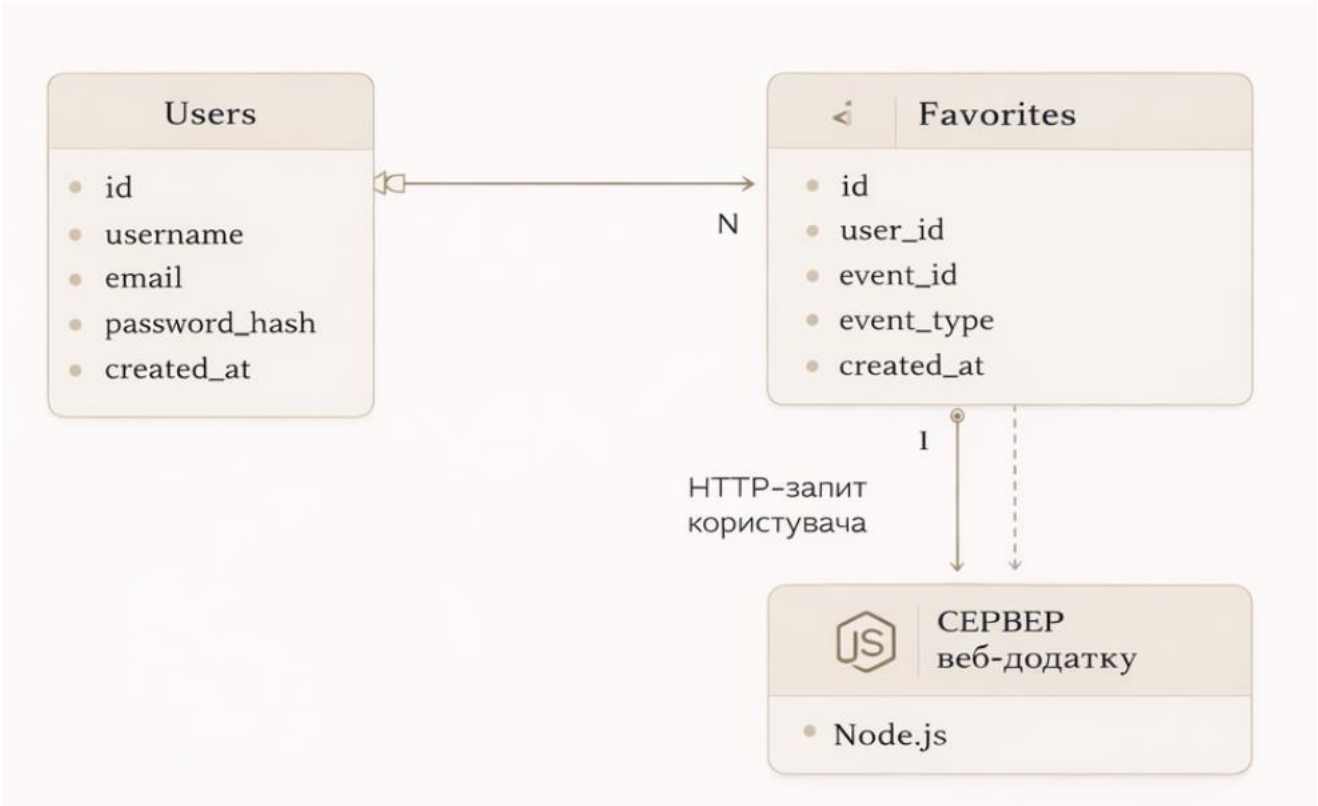


Рис. 2.3 Структура бази даних веб-додатку «Esports Tracker»

- Users
  - id
  - username
  - email

    - password_hash
    - created_at
- Favorites
    - id
    - user_id
    - event_id
    - event_type
    - created_at
- Matches
    - id
    - panda_id
    - name
    - game
    - status
    - begin_at
- Tournaments
    - id
    - panda_id
    - name
    - game
    - begin_at
    - end_at

Зв’язки:

- Users 1 : N Favorites
- Favorites пов’язується з Matches або Tournaments через event_id

Окремо доцільно подати схему інформаційної взаємодії між серверною частиною застосунку та зовнішнім API PandaScore (рис. 2.3), оскільки саме цей механізм забезпечує отримання, обробку та подальше відображення актуальної інформації про кіберспортивні події. На схемі доцільно відобразити послідовність обміну даними: формування сервером HTTP-запитів до API, отримання відповідей у форматі JSON, їх аналіз, фільтрацію та адаптацію до внутрішньої структури вебзастосунку. Такий підхід дає змогу чітко простежити, яким чином зовнішні дані про матчі, турніри, команди та результати інтегруються у систему й стають доступними для подальшого використання на клієнтській стороні.

Подання такої схеми також є важливим з погляду розуміння логіки функціонування серверної частини, адже вона виконує роль проміжної ланки між зовнішнім джерелом даних і користувацьким інтерфейсом. Сервер не лише

надсилає запити до PandaScore, а й забезпечує перевірку коректності отриманих даних, обробку можливих помилок, оптимізацію структури відповіді та передачу підготовленої інформації клієнтському застосунку у зручному для відображення вигляді. (рис. 2.3).

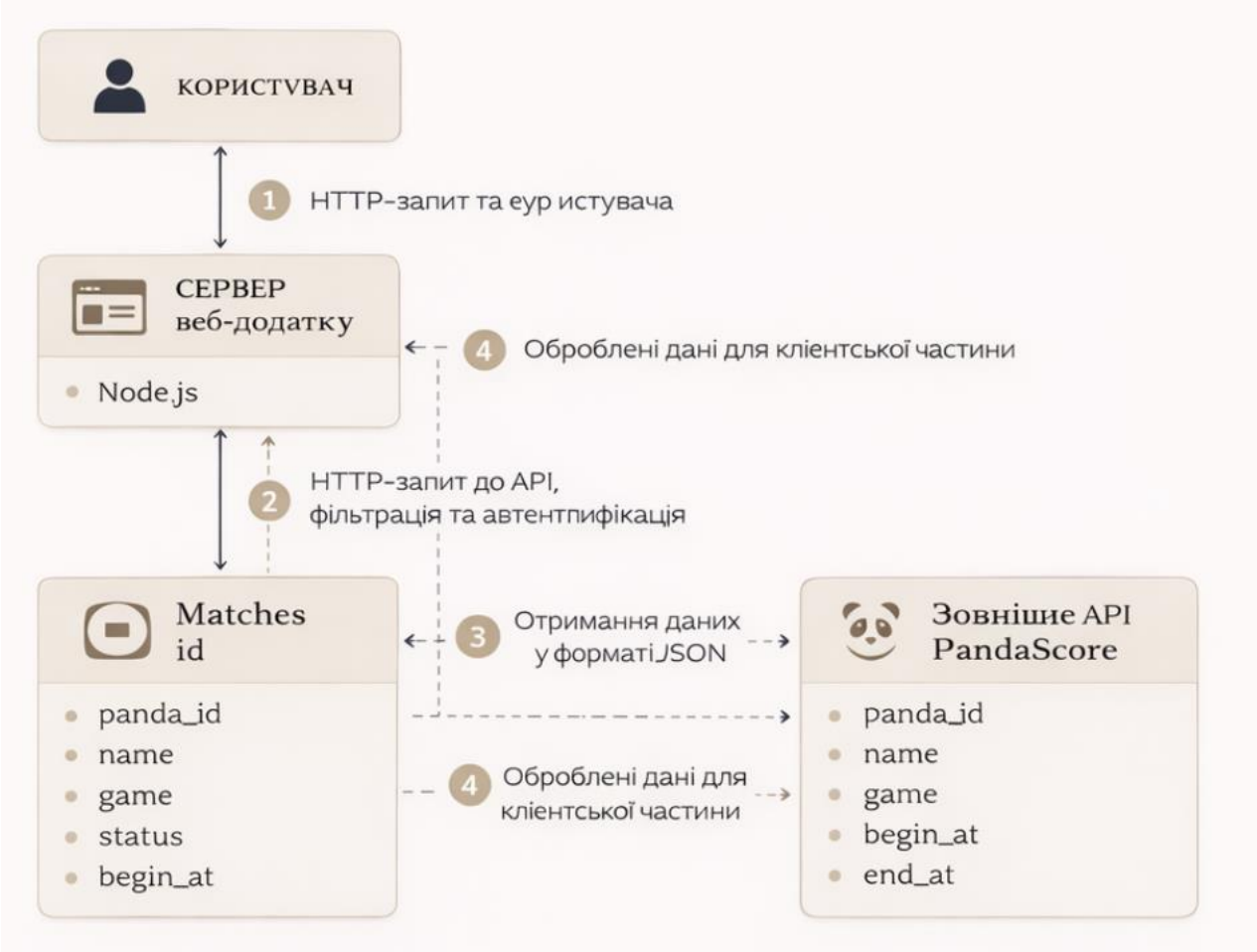


Рис. 2.4 Схема взаємодії веб-додатку «Esports Tracker» із зовнішнім API PandaScore

На схемі доцільно відобразити послідовність обміну даними між основними компонентами системи. Користувач ініціює запит через веб-інтерфейс, клієнтська частина передає його до серверного API, після чого сервер формує звернення до PandaScore. Зовнішній сервіс повертає структуровані дані про матчі або турніри, які обробляються сервером і передаються клієнту у зручному для відображення форматі. Такий підхід дозволяє централізувати обробку даних та захистити службові параметри доступу до зовнішнього сервісу [25].

Висновок до розділу 2

У розділі сформовано цілісну основу проєктування веб-застосунку Esports Tracker через систематизацію функціональних вимог, орієнтованих на реальні сценарії використання. Визначено ключові можливості системи: перегляд матчів, турнірів і новин, авторизацію, додавання подій до улюблених, пошук, фільтрацію та оновлення даних із зовнішніх джерел. Такий набір функцій забезпечує не лише базову інформативність, а й персоналізовану взаємодію користувача із сервісом.

Обґрунтовано клієнт-серверну архітектуру та модульну структуру програмних компонентів, що забезпечує розподіл відповідальностей між frontend і backend, спрощує супровід та подальше масштабування системи. Визначена організація модулів (routes, controllers, models, middleware, клієнтські JS-модулі) формує зрозумілу логіку обробки запитів і стабільну взаємодію між інтерфейсом, бізнес-логікою та джерелами даних. Реалізація токен-орієнтованої авторизації додатково підвищує захищеність доступу до персоналізованих функцій.

Спроєктовано структуру бази даних і механізм інтеграції із зовнішнім API PandaScore, що дозволяє підтримувати актуальність контенту та ефективно керувати даними користувачів і подій. Виокремлені сутності Users, Favorites, Matches, Tournaments і визначені зв’язки між ними забезпечують цілісність та гнучкість моделі даних. У підсумку розділ формує завершену проєктну базу для практичної реалізації системи, її надійної роботи та подальшого функціонального розширення.

3 РОЗРОБКА ВЕБ-ДОДАТКУ «ESPORTS TRACKER»

3.1 Реалізація серверної частини застосунку на Node.js та Express

Веб-додаток Esports Tracker було розроблено як повнофункціональну платформу для моніторингу кіберспортивних матчів і турнірів у реальному часі. Основною метою проєкту є надання користувачам зручного та персоналізованого інтерфейсу для відстеження подій у чотирьох ключових дисциплінах: Counter-Strike (csgo), Dota 2, League of Legends та Valorant. Додаток складається з трьох основних функціональних блоків, реалізованих у вигляді вкладок: розділ матчів (з розподілом на live та upcoming), розділ турнірів та розділ улюблених подій. Останній блок працює з обов'язковою авторизацією, завдяки чому обрані матчі та турніри зберігаються персонально для кожного зареєстрованого користувача в базі даних.

Технологічна основа проєкту побудована на класичному стеку MEAN з мінімальними зовнішніми залежностями. На серверній стороні використовується середовище виконання Node.js у поєднанні з фреймворком Express версії 4.19.2. Для роботи з базою даних обрано MongoDB разом з ODM-бібліотекою Mongoose 8.7.0. Авторизація реалізована через JSON Web Tokens (JWT) та хешування паролів за допомогою bcryptjs. Для виконання зовнішніх HTTP-запитів застосовується бібліотека Axios 1.13.6. Додаткові пакети, такі як cors та dotenv, забезпечують коректну роботу з CORS-політикою та безпечне зберігання конфігураційних змінних. Фронтенд-частина виконана виключно на чистому JavaScript, HTML5 та CSS3 без використання будь-яких JavaScript-фреймворків, що дозволяє зберегти мінімалізм і швидкість завантаження [27].

Архітектура проєкту чітко розділена на серверну та клієнтську частини. Усі серверні компоненти розташовані в папці server, тоді як статичні файли фронтенду (index.html, script.js та style.css) розміщені в view/assets. Головний файл app.js виконує роль точки входу: він підключає змінні середовища, налаштовує Express-

сервер, активує CORS і парсинг JSON, підключає статичну роздачу файлів, встановлює з'єднання з MongoDB через MONGO_URI та реєструє чотири групи маршрутів під префіксом /api. Головний маршрут / повертає index.html, що робить додаток доступним одразу після запуску.

Модель користувача, описана в models/user.js, є центральним елементом системи персоналізації. Крім стандартних полів username, email та password (з унікальними обмеженнями), схема містить вбудований масив favorites. Кожен елемент масиву має два поля: type (з enum-обмеженням «match» або «tournament») та id типу Mixed. Використання Mixed дозволяє зберігати як числові, так і рядкові ідентифікатори з PandaScore без додаткових перетворень. Такий підхід значно спрощує логіку збереження та пошуку улюблених елементів, оскільки вони зберігаються безпосередньо в документі користувача, а не в окремій колекції.

Механізм авторизації реалізований у middleware/auth.js і застосовується до всіх захищених маршрутів. Middleware перевіряє наявність заголовка Authorization типу Bearer, витягує токен, верифікує його за допомогою JWT_SECRET і записує ідентифікатор користувача в req.userId. У випадку відсутності або невалідності токена повертається статус 401. Контролер авторизації (auth.controller.js) забезпечує два основні методи: register та login. При реєстрації пароль хешується з коефіцієнтом 10, створюється новий документ користувача, генерується JWT-токен з терміном дії 7 днів і повертається клієнту разом з username. Процес логіну аналогічний, з додатковою перевіркою відповідності пароля.

Найскладнішим і найважливішим компонентом є контролер esports.controller.js, який відповідає за всю взаємодію з зовнішнім API PandaScore. Базовий URL визначено як https://api.pandascore.co, а авторизація здійснюється через Bearer-токен з змінної PANDA_API_KEY. Для узгодження назв ігор створено допоміжну функцію getGamePath, яка перетворює внутрішні ідентифікатори (counter_strike, dota2, league_of_legends, valorant) у формати, очікувані PandaScore (csgo, dota2, lol, valorant).

Особливої уваги заслуговує функція нормалізації normalizeMatch. Вона вирішує типову проблему PandaScore, коли поле name може бути відсутнім або не

містити «vs». Функція завжди гарантує наявність двох об’єктів у масиві opponents, генерує коректне display_name та додає поле videogame. Завдяки цьому фронтенд отримує однорідні дані незалежно від особливостей конкретного матчу. Основні методи контролера дозволяють отримувати всі матчі (live + upcoming з сортуванням за часом початку), окремо live-матчі, матч за ідентифікатором, список турнірів (з сортуванням за begin_at) та конкретний турнір. Усі запити виконуються асинхронно через Axios з обробкою помилок.

Система улюблених реалізується двома контролерами (esports.controller.js та favorites.controller.js) для гнучкості маршрутизації. Метод toggleFavorite приймає об’єкт {type, id}, перевіряє наявність елемента в масиві favorites (використовуючи == для коректного порівняння number/string), додає або видаляє його та зберігає документ. Метод getFavorites повертає поточний список улюблених для авторизованого користувача. Така реалізація забезпечує повну прив’язку до акаунту та миттєве оновлення після кожного перемикання [27].

Маршрутизація побудована за принципом розділення відповідальності: окремі файли для auth, matches, tournaments та favorites. Усі захищені маршрути (favorites та потенційно детальні перегляди) проходять через middleware protect. Фронтенд-логіка (script.js) виконує авторизацію, зберігає токен у localStorage, здійснює fetch-запити до відповідних ендпоінтів з передачею Bearer-токена, обробляє вкладки та динамічно оновлює розділ улюблених після кожної дії toggle.

Для запуску проєкту достатньо створити файл .env з трьома ключовими змінними (MONGO_URI, JWT_SECRET, PANDA_API_KEY), виконати npm install та запустити команду npm run dev (через nodemon) або npm start. Сервер автоматично підключається до бази даних і стає доступним за адресою http://localhost:3000.

Підсумовуючи, Esports Tracker демонструє ефективне поєднання сучасних практик backend-розробки з мінімалістичним фронтендом. Завдяки нормалізації даних, вбудованій системі favorites та надійній JWT-авторизації додаток забезпечує стабільну роботу, персоналізацію та легке масштабування. Реалізований підхід дозволяє легко додавати нові ігри, реал-тайм оновлення через WebSocket або

кешування відповідей PandaScore у майбутніх ітераціях. Проєкт повністю відповідає вимогам сучасної веб-розробки та може слугувати основою для більш складних кіберспортивних платформ.

На рисунку 3.1 наведено компонентну діаграму реалізованого програмного рішення. Діаграма відображає взаємодію між клієнтською частиною, серверними модулями, базою даних та зовнішнім API PandaScore.

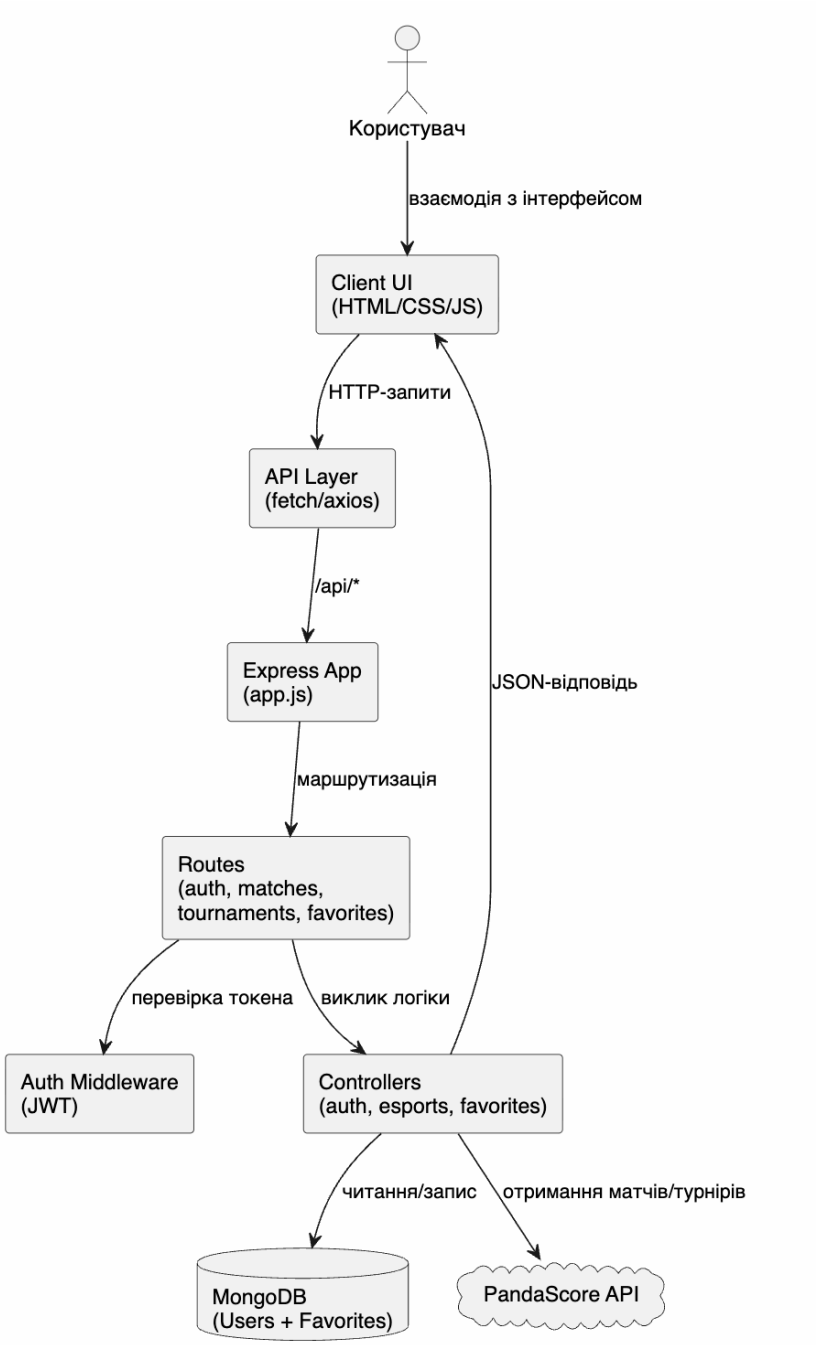


Рис. 3.1 Компонентна діаграма реалізованої системи Esports Tracker

Діаграма демонструє модульну клієнт-серверну архітектуру, у якій сервер виступає центральним вузлом обробки, авторизації та інтеграції із зовнішнім джерелом даних.

На рисунку 3.2 показано послідовність дій під час входу користувача в систему та додавання/видалення подій зі списку улюблених.

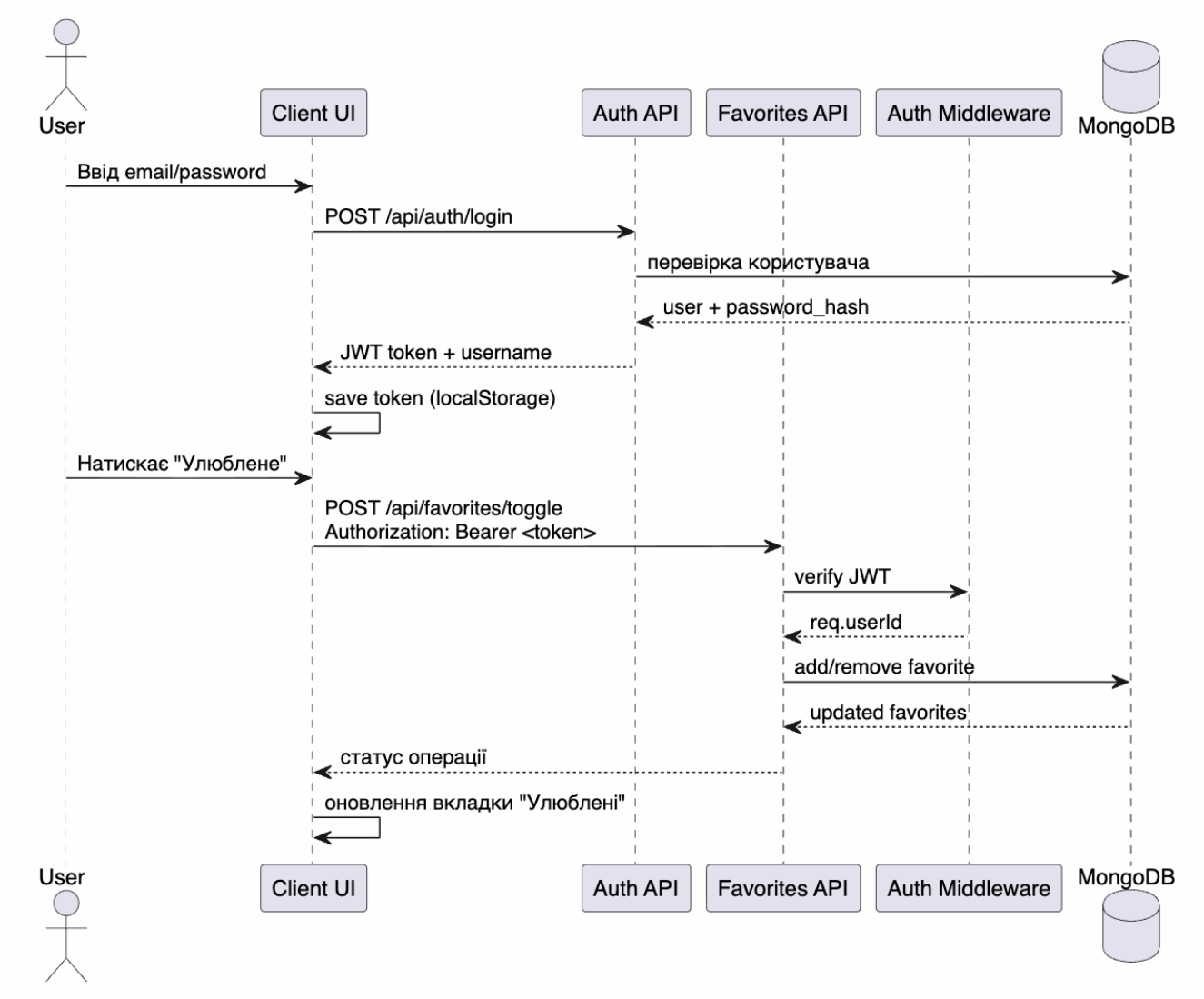


Рис. 3.2 Діаграма послідовності авторизації та керування улюбленими подіями

Діаграма підтверджує коректну реалізацію захищеного доступу до персоналізованого функціоналу через JWT та прив’язку улюблених подій до конкретного користувача.

На рисунку 3.3 представлено сценарій отримання та нормалізації кіберспортивних подій із зовнішнього API з подальшим відображенням у клієнтському інтерфейсі.

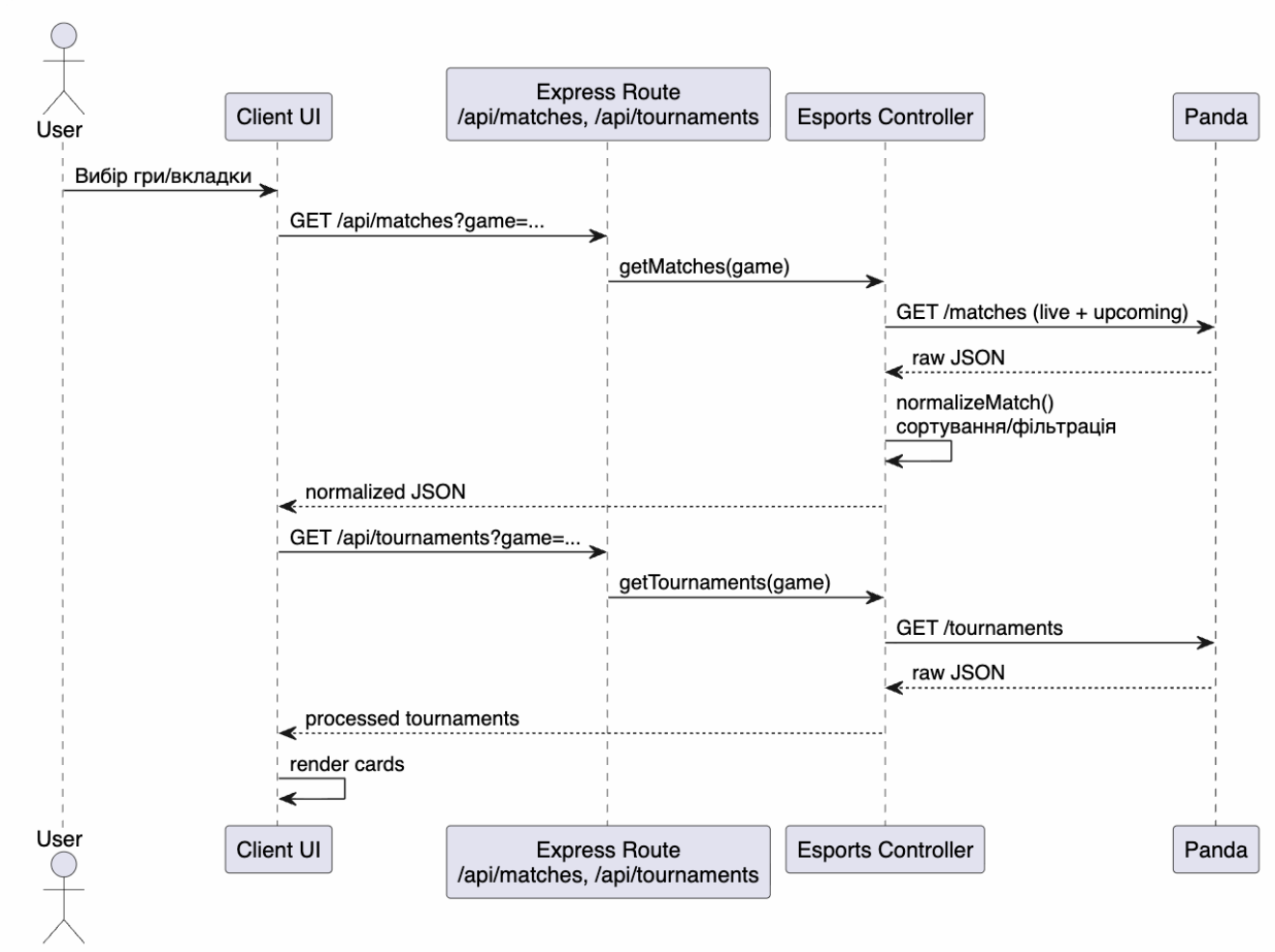


Рис. 3.3 Діаграма послідовності завантаження матчів і турнірів через PandaScore API

Діаграма показує, що серверна частина не лише проксує запити, а й виконує обробку та уніфікацію даних перед передачею на frontend.

На рисунку 3.4 наведено activity-діаграму логіки клієнтського застосунку: ініціалізація, перемикання вкладок, взаємодія з модальним вікном і автооновлення даних.

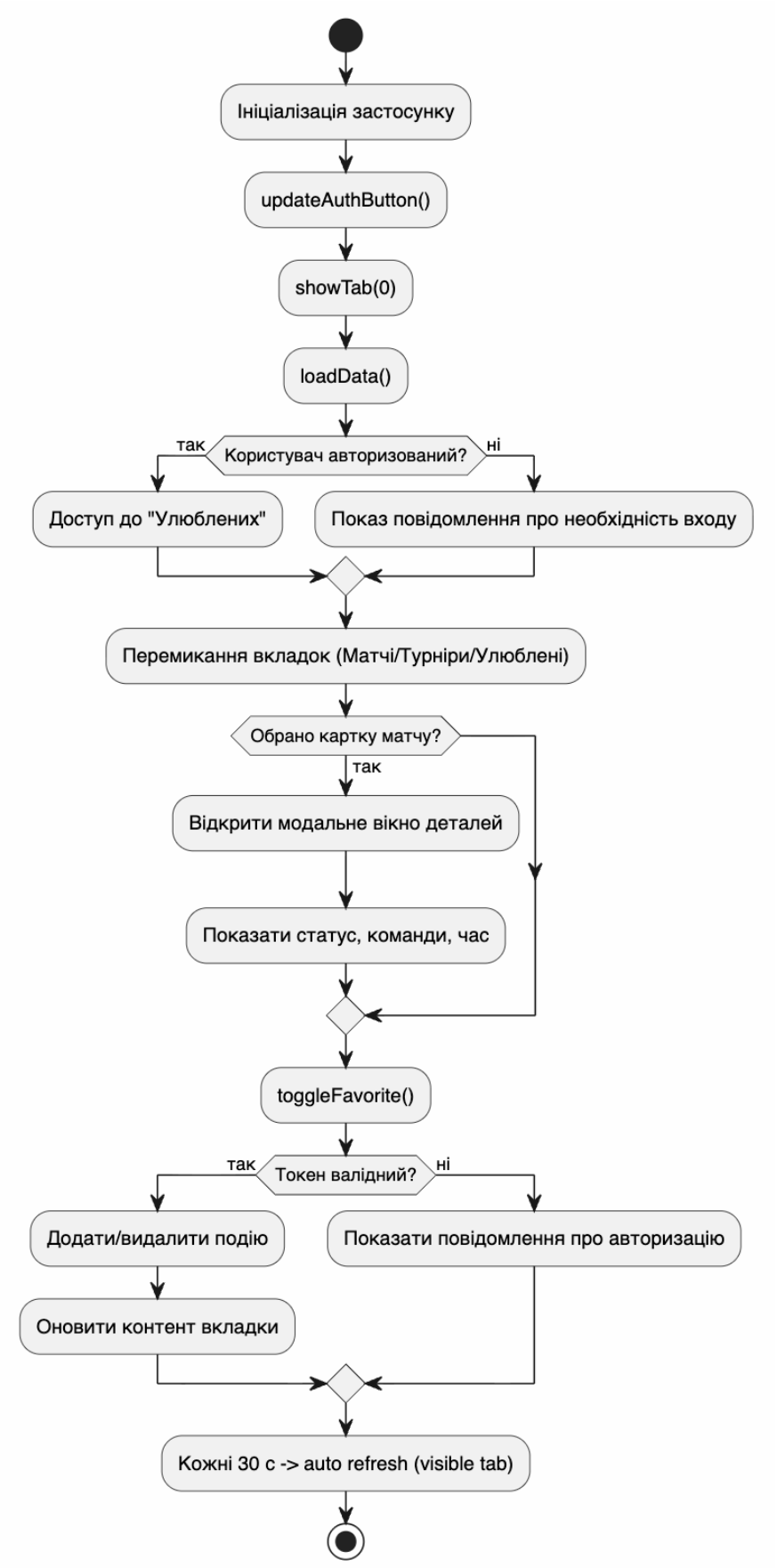


Рис. 3.4 Activity-діаграма роботи клієнтського інтерфейсу

Activity-діаграма відображає повний клієнтський цикл роботи системи та демонструє динамічний характер оновлення даних без перезавантаження сторінки.

3.2 Реалізація клієнтського інтерфейсу та системи відображення матчів і турнірів

Клієнтський інтерфейс веб-застосунку є елементом взаємодії користувача із системою, оскільки саме він забезпечує відображення інформації та доступ до функціональних можливостей застосунку. У межах розробки веб-застосунку Esports Tracker клієнтська частина реалізована з використанням сучасних веб-технологій, зокрема HTML, CSS та JavaScript, що дозволяє створити інтерактивний, зручний та адаптивний інтерфейс.

Основною метою реалізації клієнтського інтерфейсу є забезпечення наочного відображення інформації про кіберспортивні матчі, турніри та новини, а також створення інтуїтивно зрозумілої навігації для користувача. Інтерфейс побудовано таким чином, щоб користувач міг швидко отримувати доступ до необхідних даних, переглядати актуальні події та взаємодіяти із системою без зайвих ускладнень.

На рис. 3.5 представлено головний інтерфейс веб-застосунку Esports Tracker, який призначений для відображення актуальних кіберспортивних матчів у режимі реального часу. Верхня частина сторінки містить навігаційне меню, що забезпечує швидкий доступ до основних розділів системи, зокрема «Матчі», «Турніри», «Улюблені», а також функцій сповіщень і авторизації користувача. Це дозволяє користувачеві легко перемикатися між різними функціональними модулями застосунку.

Під навігаційною панеллю реалізовано блок вибору ігрової дисципліни, який представлений у вигляді інтерактивних елементів (карток) для таких популярних ігор, як Counter-Strike 2, Dota 2, League of Legends та Valorant. Активна дисципліна візуально виділяється, що дозволяє користувачу швидко зрозуміти, для якої гри відображається інформація.

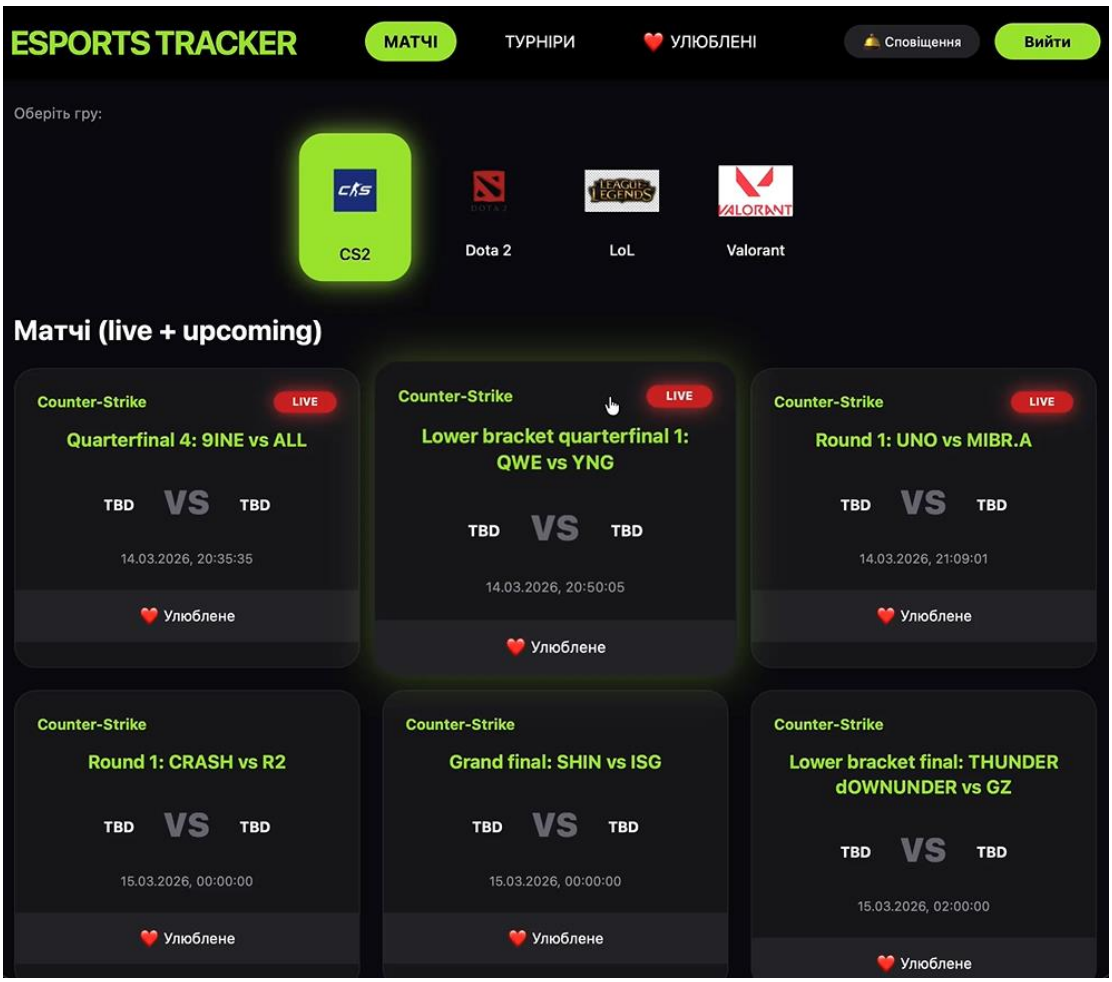


Рис. 3.5 Головна сторінка веб-застосунку Esports Tracker (відображення матчів)

Основну частину сторінки займає блок «Матчі (live + upcoming)», у якому відображаються як поточні (live), так і майбутні матчі. Інформація представлена у вигляді окремих карток, кожна з яких містить назву турніру або стадії, назви команд, статус матчу (наприклад, LIVE), а також дату і час проведення. Такий підхід забезпечує зручне сприйняття інформації та дозволяє користувачу швидко орієнтуватися серед великої кількості подій.

Кожна картка матчу також містить можливість додавання події до списку улюблених, що реалізовано у вигляді відповідного елемента інтерфейсу. Це забезпечує персоналізацію використання застосунку та дозволяє користувачеві зберігати цікаві для нього матчі [26-28].

Загалом інтерфейс сторінки виконано у сучасному стилі з акцентом на контрастність і візуальне виділення ключових елементів, що підвищує зручність використання та забезпечує ефективне відображення інформації про

кіберспортивні події.

На рисунку 3.6 зображено елемент клієнтського інтерфейсу веб-застосунку Esports Tracker, який відповідає за відображення детальної інформації про обраний матч. Даний елемент реалізовано у вигляді модального вікна, що відкривається поверх основного інтерфейсу при взаємодії користувача з карткою матчу.

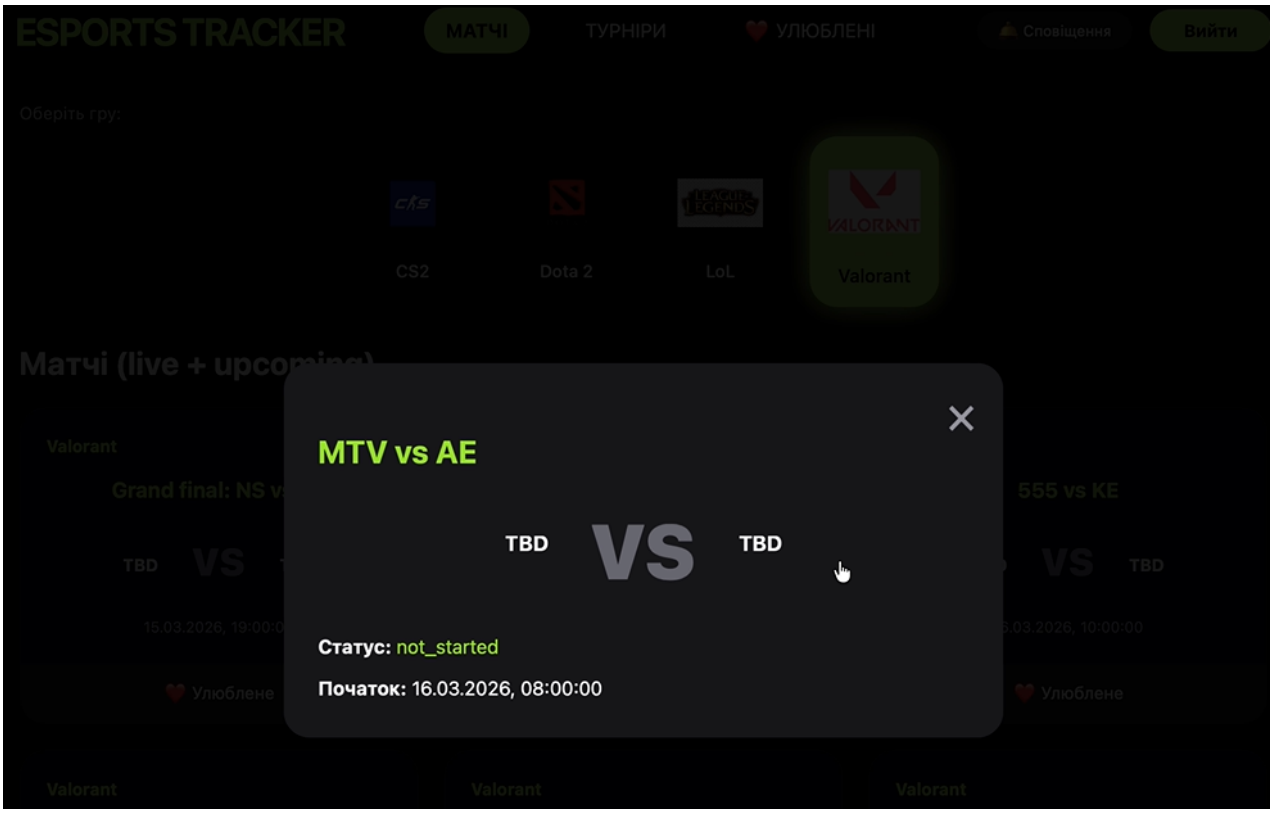


Рис. 3.6 Вікно детальної інформації про матч

У центральній частині вікна відображається назва матчу, що включає назви команд, між якими відбувається протистояння. Нижче розміщено блок із візуальним позначенням «VS», який підкреслює формат змагання. Також у вікні представлено поточний статус матчу, який дозволяє визначити етап проведення події (наприклад, «not_started»), а також інформацію про дату і час початку.

Модальне вікно має інтуїтивно зрозумілий інтерфейс і забезпечує швидкий доступ до додаткової інформації без переходу на інші сторінки. Для закриття вікна передбачено відповідний елемент керування у вигляді кнопки «×», розташованої у верхньому правому куті.

Реалізація підходу дозволяє підвищити зручність використання веб-застосунку, забезпечуючи оперативний перегляд детальної інформації про матчі без порушення основного сценарію взаємодії користувача із системою.

На рис. 3.7 представлено інтерфейс розділу «Улюблені» веб-застосунку Esports Tracker, який призначений для відображення збережених користувачем матчів і турнірів. Даний розділ є елементом персоналізації системи та забезпечує швидкий доступ до обраних подій.

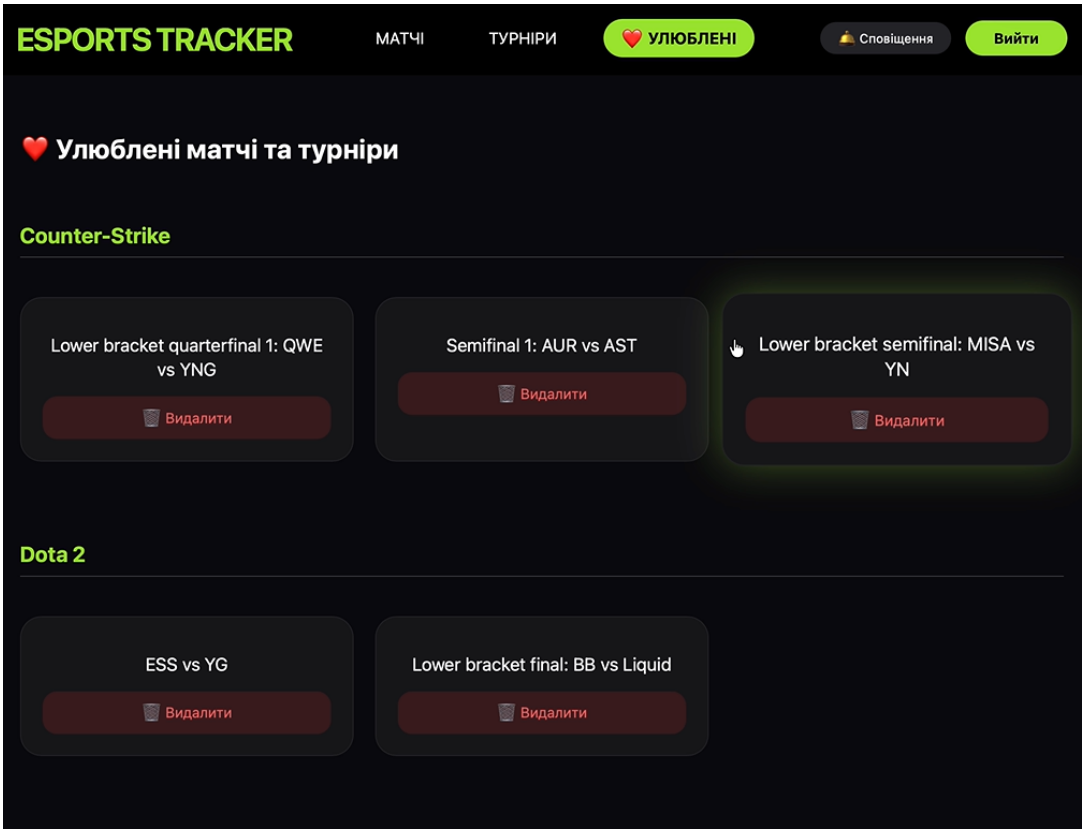


Рис. 3.7 Інтерфейс розділу «Улюблені матчі та турніри»

Інтерфейс структуровано за ігровими дисциплінами, зокрема Counter-Strike та Dota 2, що дозволяє логічно групувати інформацію та спрощує навігацію для користувача. У межах кожної дисципліни відображаються картки матчів, які були додані до списку улюблених.

Кожна картка містить назву матчу або стадії турніру, а також інтерактивний елемент керування для видалення події зі списку улюблених. Функція видалення реалізована у вигляді кнопки «Видалити», що дозволяє користувачеві швидко

керувати своїм списком обраних подій.

Реалізація даного розділу забезпечує підвищення зручності використання веб-застосунку, оскільки дозволяє користувачеві зберігати важливі для нього матчі та оперативно отримувати доступ до них без необхідності повторного пошуку. Такий підхід сприяє покращенню користувацького досвіду та підвищує ефективність взаємодії із системою.

На рис. 3.8 представлено інтерфейс розділу «Улюблені матчі та турніри» у випадку, коли користувач не пройшов авторизацію в системі. Даний стан інтерфейсу реалізує механізм обмеження доступу до персоналізованих функцій веб-застосунку.

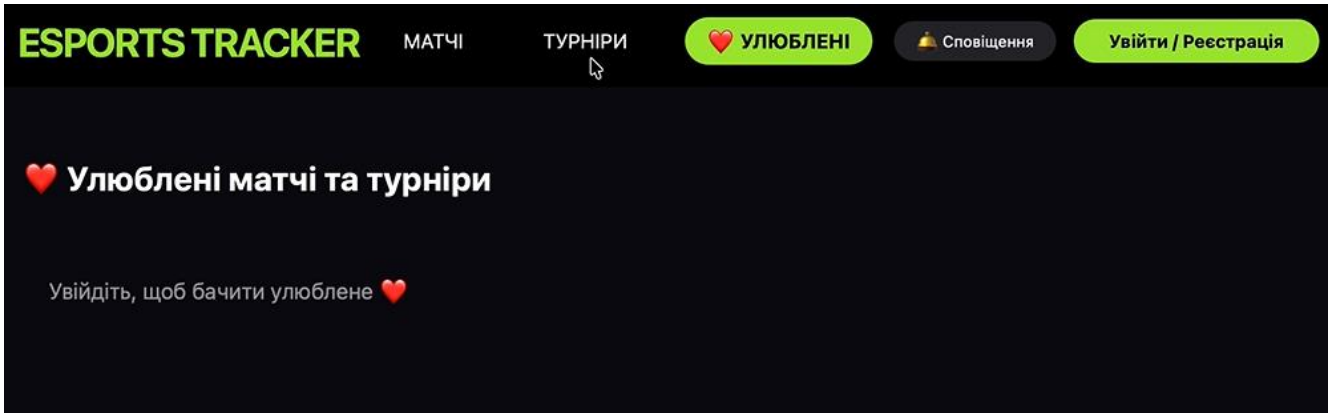


Рис. 3.8 Відображення розділу «Улюблені» для неавторизованого користувача

У центральній частині сторінки відображається інформаційне повідомлення із закликом до виконання авторизації, що інформує користувача про необхідність входу в систему для перегляду збережених подій. Такий підхід дозволяє забезпечити зрозумілу комунікацію з користувачем та підвищує зручність взаємодії із застосунком [28].

У верхній частині інтерфейсу розміщено кнопку «Увійти / Реєстрація», яка забезпечує швидкий доступ до функцій авторизації. Це дозволяє користувачу безпосередньо перейти до процесу входу в систему та отримати доступ до розширеного функціоналу, включаючи перегляд і керування улюбленими матчами та турнірами.

Реалізація такого підходу дозволяє чітко розмежувати права доступу між авторизованими та неавторизованими користувачами, що сприяє підвищенню безпеки системи та забезпечує коректну роботу механізмів персоналізації.

У системі передбачено розмежування доступу до функціоналу залежно від статусу користувача, що дозволяє реалізувати механізми персоналізації та підвищити безпеку роботи веб-застосунку. Використання сучасних веб-технологій забезпечило адаптивність інтерфейсу та можливість його подальшого розширення.

Отримані результати реалізації клієнтського інтерфейсу створюють основу для повноцінного функціонування веб-застосунку та забезпечують ефективне відображення інформації про кіберспортивні події. У наступному підрозділі буде розглянуто реалізацію інших компонентів системи та їх взаємодію.

3.3 Реалізація авторизації користувачів та системи персональних улюблених подій

У рамках розроблюваного веб-застосунку особлива увага приділяється не лише реалізації базового механізму входу в систему, але й створенню функціоналу персональних улюблених подій. Даний функціонал дозволяє користувачам відмічати цікаві для них події, формуючи індивідуальний список для подальшого перегляду та відстеження. Це значно підвищує зручність користування системою та сприяє покращенню користувацького досвіду.

У даному підрозділі буде розглянуто процес реалізації системи авторизації користувачів, а також механізм додавання, збереження та відображення улюблених подій. Особливу увагу буде приділено структурі коду, взаємодії клієнтської та серверної частин, а також використанню сучасних підходів до організації безпечної роботи з обліковими записами.

На рисунку 3.9 представлено фрагмент клієнтського JavaScript-коду, який відповідає за керування станом авторизації користувача та взаємодію з інтерфейсом вкладок у веб-застосунку.

```
let token = localStorage.getItem('token') || null;
let isLoginMode = true;

function updateAuthButton() {
    const btn = document.getElementById('auth-btn');
    if (token) {
        btn.textContent = 'Вийти';
        btn.onclick = logout;
    } else {
        btn.textContent = 'Увійти / Реєстрація';
        btn.onclick = showLoginModal;
    }
}

// === ПЕРЕМИКАННЯ ВКЛАДОК (додано) ===
function showTab(tab) {
    document.querySelectorAll('.tab-content').forEach(el => el.classList.add('hidden'));
    document.getElementById(`tab-content-${tab}`).classList.remove('hidden');

    document.querySelectorAll('.tab-btn').forEach((btn, i) => {
        btn.classList.toggle('active', i === tab);
    });

    if (tab === 0 || tab === 1) loadData();
    if (tab === 2) loadFavorites();
}
```

Рис. 3.9 Реалізація перевірки авторизації користувача та перемикання вкладок інтерфейсу

У першій частині коду здійснюється отримання токена авторизації з локального сховища браузера (localStorage). Якщо токен присутній, це означає, що користувач вже авторизований у системі. Функція updateAuthButton() змінює текст кнопки та її поведінку: у випадку авторизованого користувача відображається кнопка «Вийти», яка викликає функцію виходу (logout), а для неавторизованого – кнопка «Увійти / Реєстрація», що відкриває форму входу [28, 29].

Друга частина коду реалізує механізм перемикання вкладок інтерфейсу за допомогою функції showTab(tab). При виклику функції всі вкладки приховуються шляхом додавання CSS-класу hidden, після чого відображається лише обрана вкладка. Одночасно змінюється активний стан кнопок вкладок через додавання або видалення класу active.

Крім того, залежно від вибраної вкладки, викликаються відповідні функції

завантаження даних: loadData() для основних розділів та loadFavorites() для відображення списку улюблених подій користувача. Це забезпечує динамічне оновлення контенту без перезавантаження сторінки та підвищує зручність взаємодії з системою.

На рисунку 3.10 представлено фрагмент JavaScript-коду, який реалізує логіку роботи модального вікна для перегляду детальної інформації про обрану подію (матч).

```
// === ЗАКРИТТЯ МОДАЛОК ===
function closeModal(modalId) {
    const modal = document.getElementById(modalId);
    if (modal) modal.classList.add('hidden');
    document.body.style.overflow = 'visible';
}

function showMatchDetail(id) {
    apiCall(`/api/matches/${id}`).then(data => {
        if (!data) return;
        document.getElementById('modal-content').innerHTML = `
            <h2 class="text-3xl font-bold mb-4 text-lime-400">${data.name || 'Деталі матчу'}</h2>
            <div class="flex gap-8 justify-center my-8">
                <div class="text-center">
                    <img src="${data.opponents?.[0]?.image_url || ''}" class="w-24 h-24 rounded-2xl bg-zinc-800 mx-auto" onerror="this.style.display='none'">
                    <p class="mt-3 font-bold">${data.opponents?.[0]?.name || '?'}</p>
                </div>
                <div class="text-6xl text-zinc-500 self-center font-black">VS</div>
                <div class="text-center">
                    <img src="${data.opponents?.[1]?.image_url || ''}" class="w-24 h-24 rounded-2xl bg-zinc-800 mx-auto" onerror="this.style.display='none'">
                    <p class="mt-3 font-bold">${data.opponents?.[1]?.name || '?'}</p>
                </div>
            </div>
            <div class="space-y-3 text-lg">
                <p><strong>Статус:</strong> <span class="${data.status === 'running' ? 'text-red-500' : 'text-lime-400'}">${data.status}</span></p>
                <p><strong>Початок:</strong> ${new Date(data.begin_at).toLocaleString('uk-UA')}</p>
            </div>
        `;
        document.getElementById('match-modal').classList.remove('hidden');
        document.body.style.overflow = 'hidden';
    });
}
```

Рис. 3.10 Реалізація відображення деталей матчу у модальному вікні

У першій частині коду описано функцію closeModal(modalId), яка відповідає за закриття модального вікна. Вона знаходить відповідний елемент за ідентифікатором, додає йому CSS-клас hidden, що приховує його з інтерфейсу, а також відновлює прокрутку сторінки шляхом зміни властивості overflow у тілі документа.

Основна логіка реалізована у функції showMatchDetail(id), яка виконує асинхронний запит до серверного API (/api/matches/${id}) для отримання інформації про конкретний матч. Після отримання даних формується HTML-розмітка, яка динамічно вставляється у вміст модального вікна.

У модальному вікні відображаються назва події, інформація про учасників (команди або гравців), їх зображення, а також статус матчу та час його початку. Для підвищення надійності реалізації використовується перевірка на наявність даних (optional chaining), що дозволяє уникнути помилок при відсутності окремих полів.

Після формування контенту модальне вікно відображається шляхом видалення класу hidden, а прокрутка основної сторінки блокується (overflow: hidden), що забезпечує зручний перегляд інформації користувачем.

На рисунку 3.11 представлено фрагмент JavaScript-коду, який відповідає за взаємодію користувача з функціоналом улюблених подій, а також за обробку додаткових дій інтерфейсу, зокрема закриття модальних вікон.

```
// Закриття по ESC
document.addEventListener('keydown', e => {
    if (e.key === 'Escape') {
        if (!document.getElementById('match-modal').classList.contains('hidden')) closeModal('match-modal');
        if (!document.getElementById('auth-modal').classList.contains('hidden')) closeModal('auth-modal');
    }
});

// === УЛЮБЛЕНЕ ===
async function toggleFavorite(type, id) {
    if (!token) {
        alert('Увійдіть, щоб додавати в улюблене ❤');
        return;
    }
    const result = await apiCall('/api/favorites/toggle', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ type, id })
    });

    if (result?.message) {
        const msg = result.message === 'Додано' ? '✅ Додано в улюблене!' : '❌ Видалено';
        const notif = document.createElement('div');
        notif.className = 'fixed bottom-8 left-1/2 -translate-x-1/2 bg-lime-400 text-black px-8 py-4 rounded-3xl font-bold shadow-2xl z-[200]';
        notif.textContent = msg;
        document.body.appendChild(notif);
        setTimeout(() => notif.remove(), 2000);
    }

    if (!document.getElementById('tab-content-2').classList.contains('hidden')) loadFavorites();
    else loadData();
}
```

Рис. 3.11 Реалізація додавання подій до списку улюблених та обробка взаємодії користувача

У верхній частині коду реалізовано обробник події натискання клавіші (keydown). При натисканні клавіші Escape виконується перевірка стану модальних вікон, і якщо вони відкриті, викликається функція closeModal(), що забезпечує їх закриття. Це підвищує зручність користування інтерфейсом.

Основна логіка додавання подій до улюблених реалізована у функції toggleFavorite(type, id). Перед виконанням операції перевіряється наявність токена авторизації. Якщо користувач не авторизований, виводиться повідомлення з вимогою увійти в систему, що забезпечує контроль доступу до персоналізованих функцій. У випадку авторизованого користувача здійснюється асинхронний POST-запит до серверного API (/api/favorites/toggle) з передачею параметрів події.

У відповідь система отримує результат виконання операції, на основі якого формується повідомлення про успішне додавання або видалення події зі списку улюблених. Для інформування користувача використовується динамічно створене повідомлення (notification), яке додається до DOM-структури сторінки та автоматично зникає через певний проміжок часу [29].

Такий підхід покращує користувацький досвід без необхідності перезавантаження сторінки. Після виконання операції відбувається оновлення контенту: якщо активна вкладка улюблених, викликається функція loadFavorites(), інакше — loadData(). Це забезпечує актуальність відображених даних у системі.

На рисунку 3.12а та 3.12б представлено фрагмент JavaScript-коду, який реалізує функціонал авторизації та реєстрації користувачів у веб-застосунку, а також керування відповідним модальним вікном.

```
// === АВТОРИЗАЦІЯ ===
function showLoginModal() {
    isLoginMode = true;
    document.getElementById('modal-title').textContent = 'Увійти';
    document.getElementById('auth-action-btn').textContent = 'Увійти';
    document.getElementById('switch-text').textContent = 'Немає акаунту? Зареєструйся';
    document.getElementById('auth-modal').classList.remove('hidden');
    document.body.style.overflow = 'hidden';
}

function toggleAuthMode() {
    isLoginMode = !isLoginMode;
    const title = document.getElementById('modal-title');
    const btn = document.getElementById('auth-action-btn');
    const switchText = document.getElementById('switch-text');

    if (isLoginMode) {
        title.textContent = 'Увійти'; btn.textContent = 'Увійти';
        switchText.textContent = 'Немає акаунту? Зареєструйся';
    } else {
        title.textContent = 'Реєстрація'; btn.textContent = 'Зареєструватися';
        switchText.textContent = 'Вже маєте акаунт? Увійти';
    }
}
```

Рис. 3.12а Реалізація процесу авторизації та реєстрації користувача

```
async function handleAuth() {
    const email = document.getElementById('email').value.trim();
    const password = document.getElementById('password').value.trim();
    if (!email || !password) return alert('Заповніть усі поля');

    const url = isLoginMode ? '/api/auth/login' : '/api/auth/register';
    const body = { email, password, username: email.split('@')[0] };

    try {
        const res = await fetch(url, {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(body)
        });
        const data = await res.json();

        if (res.ok && data.token) {
            token = data.token;
            localStorage.setItem('token', token);
            closeModal('auth-modal');
            updateAuthButton();
            loadData();
            const greet = document.createElement('div');
            greet.className = 'fixed top-8 right-8 bg-lime-400 text-black px-6 py-3 rounded-3xl font-bold';
            greet.textContent = isLoginMode ? '👋 Ласкаво просимо!' : '🎉 Реєстрація успішна!';
            document.body.appendChild(greet);
            setTimeout(() => greet.remove(), 2500);
        } else {
            alert(data.error || 'Помилка');
        }
```

Рис. 3.12б Реалізація процесу авторизації та реєстрації користувача

На рисунку 3.13 У першій частині коду описано функцію apiCall(endpoint, options), яка використовується для виконання HTTP-запитів до серверної частини. Якщо користувач авторизований, до заголовків запиту автоматично додається токен у форматі Bearer, що забезпечує захищений доступ до API. У разі отримання відповіді зі статусом 401 (неавторизований доступ) викликається функція logout(), що підвищує безпеку системи.

Функція loadData() відповідає за завантаження даних відповідно до обраного користувачем фільтра (наприклад, гри). Вона ініціює запити до API для отримання списків матчів та турнірів, після чого передає отримані дані у відповідні функції відображення [29].

Функція renderMatches(matches) виконує динамічне формування HTML-розмітки для відображення матчів у вигляді сітки. У випадку відсутності даних користувачу виводиться відповідне повідомлення. Для кожного матчу створюється окремий блок, що містить інформацію про гру, статус (наприклад, активний матч), учасників, їх зображення та час початку.

```
// === API ===
async function apiCall(endpoint, options = {}) {
    if (token) {
        options.headers = options.headers || {};
        options.headers.Authorization = `Bearer ${token}`;
    }
    const res = await fetch(endpoint, options);
    if (res.status === 401) { logout(); return null; }
    return res.json();
}

// === ЗАВАНТАЖЕННЯ ДАНИХ ===
async function loadData() {
    const game = document.getElementById('game-filter').value;
    document.getElementById('matches-grid').innerHTML = '<p class="text-center col-span-3 text-zinc-400">Завантаження матчів...
    document.getElementById('tournaments-grid').innerHTML = '<p class="text-center col-span-3 text-zinc-400">Завантаження турні

    const matches = await apiCall(`/api/matches?game=${game}`);
    renderMatches(matches || []);

    const tournaments = await apiCall(`/api/tournaments?game=${game}`);
    renderTournaments(tournaments || []);
}

function renderMatches(matches) {
    const grid = document.getElementById('matches-grid');
    if (!matches.length) {
        grid.innerHTML = '<p class="text-center col-span-3 text-zinc-400">Немає матчів для цієї гри</p>';
        return;
    }
    grid.innerHTML = matches.map(m => `
        <div onclick="showMatchDetail(${m.id})"
             class="group bg-zinc-900 rounded-3xl overflow-hidden hover:scale-105 transition-all cursor-pointer border border-z
            <div class="p-6">
                <div class="flex justify-between items-center">
                    <span class="text-lime-400 font-bold">${m.video_game?.name || 'Esports'}</span>
                    ${m.status === 'running' ? `<span class="live px-5 py-1 bg-red-600 rounded-full text-xs font-bold animate-p
                </div>

                <div class="flex justify-center items-center gap-8 my-8">
                    <div class="text-center">
                        <img src="${m.opponents?.[0]?.image_url || ''}" class="w-16 h-16 mx-auto rounded-full bg-zinc-800" oner
                        <div class="font-bold mt-2">${m.opponents?.[0]?.name || '?'}</div>
                    </div>
                    <div class="text-4xl text-zinc-500 font-black">VS</div>
                    <div class="text-center">
                        <img src="${m.opponents?.[1]?.image_url || ''}" class="w-16 h-16 mx-auto rounded-full bg-zinc-800" oner
                        <div class="font-bold mt-2">${m.opponents?.[1]?.name || '?'}</div>
                    </div>
                </div>

                <div class="text-center text-zinc-400 text-sm">${new Date(m.begin_at).toLocaleString('uk-UA')}</div>
            </div>
            <button onclick="event.stopImmediatePropagation(); toggleFavorite('match', ${m.id});"
                    class="w-full py-4 bg-zinc-800 hover:bg-lime-400 hover:text-black transition-colors text-sm font-medium">
```

Рис. 3.13 Реалізація API-викликів та відображення списку матчів у веб-застосунку

Крім того, кожен елемент списку є інтерактивним: при натисканні відкривається детальна інформація про матч, а також передбачена можливість додавання події до списку улюблених через відповідну кнопку. Це забезпечує зручну навігацію та взаємодію користувача з контентом без перезавантаження сторінки.

На рисунку 3.14 представлено фрагмент JavaScript-коду, який відповідає за початкову ініціалізацію веб-застосунку та реалізацію механізму періодичного оновлення даних.

```
// === INIT ===
updateAuthButton();
loadData();
showTab(0);

// Live оновлення
setInterval(() => { if (document.visibilityState === 'visible') loadData(); }, 30000);
```

Рис. 3.14 Ініціалізація застосунку та механізм автоматичного оновлення даних

У першій частині коду викликаються основні функції ініціалізації: updateAuthButton() — для оновлення стану кнопки авторизації відповідно до наявності токена, loadData() — для первинного завантаження даних із сервера, та showTab(0) — для відображення початкової вкладки інтерфейсу. Це забезпечує коректний старт роботи застосунку після його відкриття користувачем.

У другій частині реалізовано механізм автоматичного оновлення даних (live-оновлення) за допомогою функції setInterval(). Кожні 30 секунд перевіряється, чи активна вкладка браузера (document.visibilityState === 'visible'). У разі активного стану викликається функція loadData(), що дозволяє отримувати актуальну інформацію без необхідності ручного оновлення сторінки.

Висновок до розділу 3

У розділі 3 виконано практичну реалізацію веб-застосунку Esports Tracker та підтверджено працездатність обраної архітектури на рівні серверної і клієнтської частин. Серверний модуль на базі Node.js та Express забезпечує стабільну обробку REST-запитів, інтеграцію з PandaScore API, нормалізацію вхідних даних і уніфіковане повернення результатів у форматі JSON. Реалізовані маршрути для

матчів, турнірів, авторизації та улюблених подій формують цілісну backend-логіку, придатну до подальшого масштабування.

Клієнтський інтерфейс, побудований на HTML/CSS/JavaScript, забезпечує інтуїтивну взаємодію з даними: відображення live/upcoming матчів, перегляд турнірів, модальні вікна деталей подій, роботу вкладок і динамічне оновлення контенту без перезавантаження сторінки. Продемонстровано реалізацію механізмів персоналізації, зокрема додавання/видалення обраних подій, а також коректне розмежування доступу для авторизованих і неавторизованих користувачів, що позитивно впливає на зручність і безпеку використання системи.

Реалізація авторизації на основі JWT і хешування паролів через bcryptjs забезпечує захищений доступ до персональних функцій, а модель зберігання favorites у профілі користувача спрощує керування обраними подіями. У підсумку розроблений застосунок відповідає поставленим вимогам: забезпечує актуальний моніторинг кіберспортивних подій, персоналізований користувацький сценарій та технологічну основу для подальшого розвитку, зокрема впровадження кешування, WebSocket-оновлень і розширення переліку дисциплін.

ВИСНОВКИ

Проаналізовано теоретичні основи розробки веб-систем для моніторингу кіберспортивних подій, зокрема досліджено розвиток кіберспорту як сучасного цифрового явища та роль веб-платформ у відстеженні турнірів і матчів. Визначено ключові тенденції розвитку галузі, що характеризуються високою динамічністю, необхідністю оперативного доступу до даних та зростанням попиту на інтерактивні інформаційні сервіси.

Виокремлено сучасні веб-технології, що використовуються для створення інформаційних систем, та обґрунтовано доцільність застосування клієнт-серверної архітектури для реалізації подібних застосунків. Проведений аналіз існуючих платформ для моніторингу кіберспортивних подій дозволив визначити їх переваги та недоліки, зокрема обмежені можливості персоналізації та недостатню гнучкість інтерфейсу.

Обґрунтовано функціональні вимоги до веб-застосунку «Esports Tracker», визначено його основні можливості та сценарії використання. Спроєктовано архітектуру системи, яка базується на розподілі функцій між клієнтською та серверною частинами, що забезпечує ефективну обробку даних, масштабованість та зручність супроводу програмного забезпечення.

Розроблено структуру бази даних, яка дозволяє зберігати інформацію про користувачів, матчі, турніри та персональні улюблені події. Організовано взаємодію із зовнішнім API PandaScore, що забезпечує отримання актуальних даних про кіберспортивні події. Продемонстровано доцільність використання серверної обробки даних для підвищення безпеки та оптимізації роботи застосунку.

Реалізовано веб-додаток «Esports Tracker» із використанням сучасних технологій, зокрема Node.js та Express для серверної частини та JavaScript для клієнтського інтерфейсу. Забезпечено динамічне відображення даних, реалізовано систему авторизації користувачів, механізм додавання подій до списку улюблених, а також інтерактивну взаємодію без перезавантаження сторінки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Riot Games. League of Legends Esports Overview [Електронний ресурс]. – Режим доступу: https://lolesports.com/ (дата звернення: 15.03.2026).
2. Valve Corporation. Dota 2 Official Website [Електронний ресурс]. – Режим доступу: https://www.dota2.com/ (дата звернення: 15.03.2026).
3. Valve Corporation. Counter-Strike 2 Official Website [Електронний ресурс]. – Режим доступу: https://www.counter-strike.net/ (дата звернення: 15.03.2026).
4. Riot Games. Valorant Official Website [Електронний ресурс]. – Режим доступу: https://playvalorant.com/ (дата звернення: 15.03.2026).
5. HLTV.org. Counter-Strike News & Statistics [Електронний ресурс]. – Режим доступу: https://www.hltv.org/ (дата звернення: 15.03.2026).
6. Liquipedia. Esports Wiki Platform [Електронний ресурс]. – Режим доступу: https://liquipedia.net/ (дата звернення: 15.03.2026).
7. PandaScore. Esports API Documentation [Електронний ресурс]. – Режим доступу: https://developers.pandascore.co/ (дата звернення: 15.03.2026).
8. Mozilla Foundation. MDN Web Docs – Web Technologies [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/ (дата звернення: 15.03.2026).
9. OpenJS Foundation. Node.js Official Documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs (дата звернення: 15.03.2026).
10. Express.js. Fast, Unopinionated, Minimalist Web Framework [Електронний ресурс]. – Режим доступу: https://expressjs.com/ (дата звернення: 15.03.2026).
11. MongoDB Inc. MongoDB Documentation [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs/ (дата звернення: 15.03.2026).
12. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures [Електронний ресурс]. – Режим доступу:

https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (дата звернення: 15.03.2026).

13. Newman S. Building Microservices: Designing Fine-Grained Systems. – Sebastopol : O'Reilly Media, 2021. – 280 p.

14. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. – Boston : Addison-Wesley, 2003. – 560 p.

15. Postman Inc. API Platform Documentation [Електронний ресурс]. – Режим доступу: https://learning.postman.com/docs/ (дата звернення: 15.03.2026).

16. Axios. Promise based HTTP client for the browser and Node.js [Електронний ресурс]. – Режим доступу: https://axios-http.com/docs/intro (дата звернення: 15.03.2026).

17. JSON.org. Introducing JSON [Електронний ресурс]. – Режим доступу: https://www.json.org/json-en.html (дата звернення: 15.03.2026).

18. OWASP Foundation. OWASP Top Ten Web Application Security Risks [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-top-ten/ (дата звернення: 15.03.2026).

19. JWT.io. JSON Web Tokens Introduction [Електронний ресурс]. – Режим доступу: https://jwt.io/introduction (дата звернення: 15.03.2026).

20. NPM Inc. npm Documentation [Електронний ресурс]. – Режим доступу: https://docs.npmjs.com/ (дата звернення: 15.03.2026).

21. WHATWG. HTML Living Standard [Електронний ресурс]. – Режим доступу: https://html.spec.whatwg.org/ (дата звернення: 15.03.2026).

22. W3C. Cascading Style Sheets (CSS) Specification [Електронний ресурс]. – Режим доступу: https://www.w3.org/Style/CSS/ (дата звернення: 15.03.2026).

23. ECMA International. ECMAScript® Language Specification [Електронний ресурс]. – Режим доступу: https://tc39.es/ecma262/ (дата звернення: 15.03.2026).

24. Google Developers. Web Fundamentals [Електронний ресурс]. – Режим доступу: https://developers.google.com/web (дата звернення: 15.03.2026).

25. Docker Inc. Docker Documentation [Електронний ресурс]. – Режим доступу: https://docs.docker.com/ (дата звернення: 15.03.2026).

26. Nginx Inc. NGINX Documentation [Електронний ресурс]. – Режим доступу: https://nginx.org/en/docs/ (дата звернення: 15.03.2026).

27. Heroku. Node.js Application Deployment Guide [Електронний ресурс]. – Режим доступу: https://devcenter.heroku.com/categories/nodejs-support (дата звернення: 15.03.2026).

28. Microsoft. REST API Design Guidelines [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design (дата звернення: 15.03.2026).

29. Fowler M. Patterns of Enterprise Application Architecture. – Boston : Addison-Wesley, 2002. – 533 p.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНОКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА
«Проєктування та розробка вебзастосункуз використанням Python для відстеженнякіберспортивнихтурнірів та ігрових новин»

Виконав студент: групиТЦР-43 Андрій АНЧЕВ

Керівник роботи: старший викладач кафедриВадим ЧИТУЛЯН

Київ – 2026

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: розробка вебзастосунку Esports Tracker, який забезпечує зручний, швидкий і ефективний доступ до інформації про кіберспортивні матчі, турніри та ігрові новини, а також надає можливість персоналізованого відстеження подій для підвищення якості й оперативності отримання актуального контенту користувачами.

Об’єкт дослідження: інформаційна система моніторингу кіберспортивних подій та новин ігрової індустрії .

Предметдослідження: методи проєктування та розробки веб-застосунків для відстеження кіберспортивних матчів, турнірів і новин з використанням сучасних веб-технологій та зовнішніх API.

ПОРІВНЯННЯ ІСНУЮЧИХ ПЛАТФОРМ ТА РОЗРОБЛЕНОГО ВЕБ-ЗАСТОСУНКУ ESPORTS TRACKER

Аналіз трьох провідних кіберспортивних сервісів виявив спільну проблему: жодна платформа не закриває всі потреби користувача в єдиному інтерфейсі.

Критерій	HLTV	Liquipedia	PandaScore	Esports Tracker (розроблений)
Відображення матчів	Так	Так	Так (через API)	Так
Турніри	Так	Так	Так	Так
Статистика	Так (детальна)	Так	Так	Так
Новини	Так	Так	Ні	Ні
Підтримка різних дисциплін	Обмежено	Так	Так	Так
Персоналізація	Ні	Ні	Ні	Так
Інтуїтивний інтерфейс	Середній	Складний	Відсутній UI	Так
Єдина система (all-in-one)	Ні	Ні	Ні	Так
Інтеграція API	Ні	Ні	Ні	Так
Можливість розширення	Обмежена	Середня	Висока	Висока

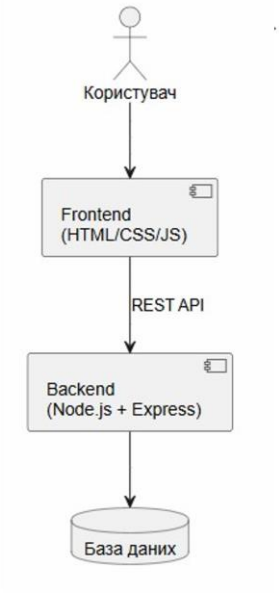
ОСНОВНІ ФУНКЦІОНАЛЬНІ ВИМОГИ ВЕБ-ЗАСТОСУНКУ ESPORTS TRACKER

№	Функціональна вимога	Характеристика	Кількісний критерій (приймання)
1	Перегляд матчів	Відображення списку актуальних та майбутніх матчів	На сторінці виводиться не менше 20 подій; час завантаження списку до 2 с
2	Перегляд турнірів	Надання інформації про турніри, дати та учасників	Виведення щонайменше 10 турнірів; сортування за датою початку виконується коректно у 100% перевірок
3	Перегляд новин	Відображення новин кіберспортивної та ігрової тематики	На сторінці відображається не менше 15 новин; оновлення стрічки не рідше 1 разу на 30 хв
4	Авторизація користувача	Реєстрація, вхід та ідентифікація користувача	Успішна реєстрація/вхід до 2 с; 100% захищених маршрутів доступні лише з валідним токеном
5	Додавання в улюблене	Збереження обраних подій для швидкого доступу	Операція додавання/видалення виконується до 1 с; зміни відображаються без перезавантаження сторінки
6	Пошук інформації	Пошук матчів, турнірів або новин за ключовими словами	Час відповіді пошуку до 1 с; релевантні результати для точного запиту - не менше 90%
7	Фільтрація даних	Відбір інформації за дисципліною, статусом або датою	Застосування фільтра до 1 с; коректність фільтрації - не менше 95% тестових сценаріїв
8	Оновлення даних	Отримання актуальної інформації із зовнішніх API	Період оновлення - кожні 30 с (для live); доступність API-даних у системі - не менше 95% часу

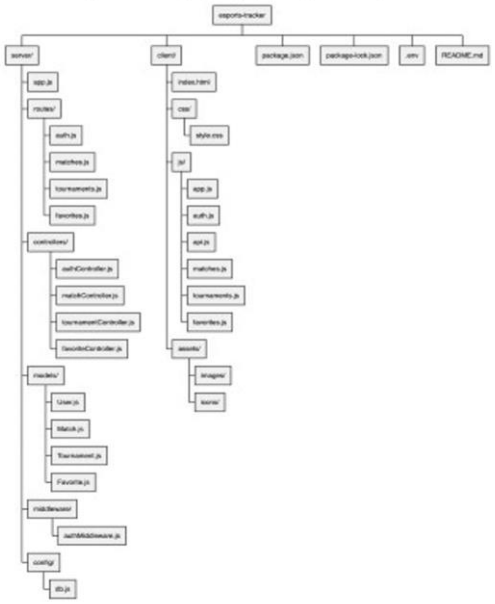
ОСНОВНІ НЕФУНКЦІОНАЛЬНІ ВИМОГИ ВЕБ-ЗАСТОСУНКУ ESPORTS TRACKER

№	Нефункціональна вимога	Характеристика	Кількісний критерій (приймання)
1	Продуктивність	Час завантаження головної сторінки	до 2 сек при стабільному інтернеті
2	Швидкодія API	Час відповіді бекенду	до 500 мс для 95% запитів
3	Доступність сервісу	Uptime	не менше 99.5% на місяць
4	Безпека	Захист даних користувача	HTTPS,хешування паролів, JWT сесії
5	Юзабіліті	Зручність інтерфейсу	ключові дії виконуються за 2–3 кліки
6	Кросплатформність	Підтримка браузерів/пристроїв	Chrome, Firefox, Edge + коректна мобільна версія
7	Надійність	Відновлення після помилок	система не втрачає дані, коректно обробляє збої
8	Підтримуваність	Простота оновлення	модульна архітектура, документований API

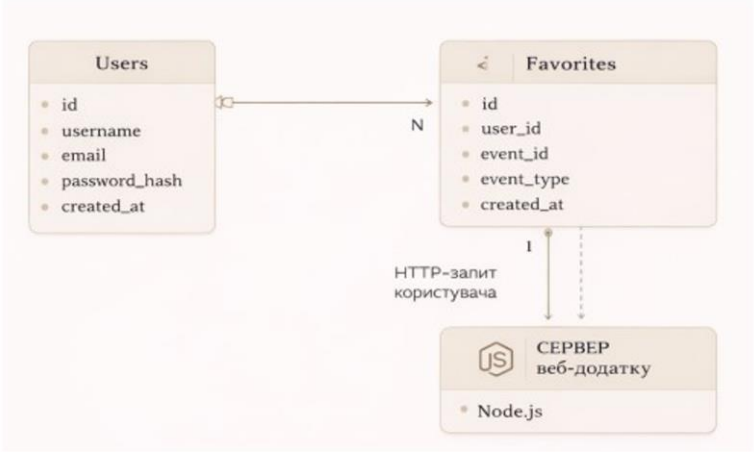
Архітектура веб-системи «Esports



Дерево структури проєкту

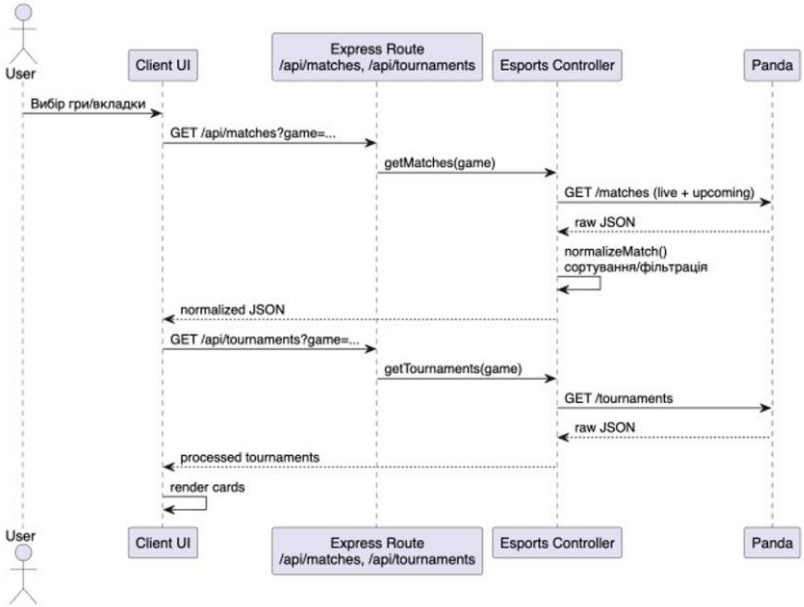


СТРУКТУРА БАЗИ ДАНИХ ВЕБ -ДОДАТКУ «ESPORTS TRACKER»



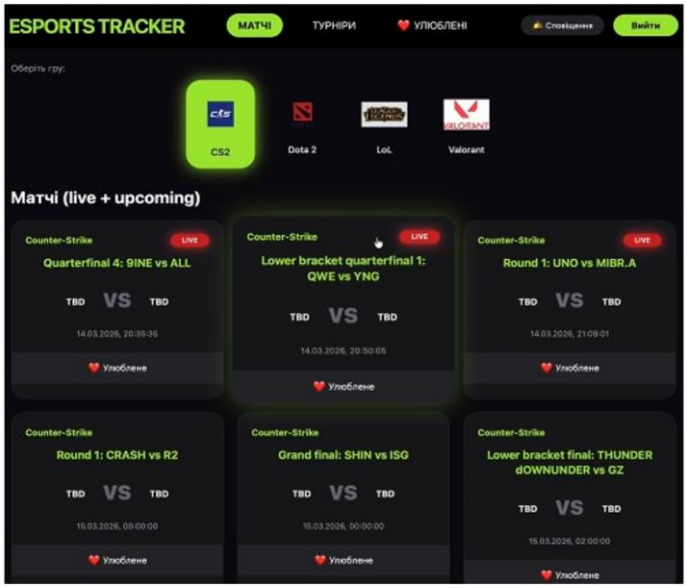
7

Діаграма послідовності завантаження матчів і турнірів



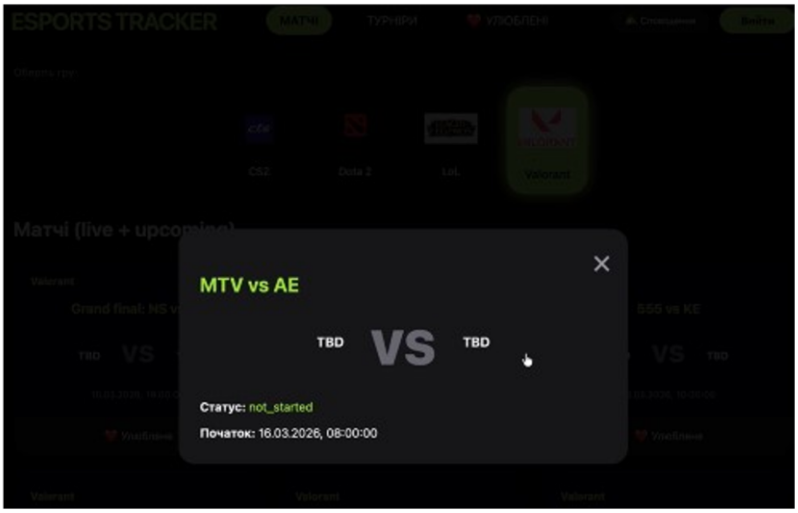
8

ГОЛОВНА СТОРІНКА ВЕБ-ЗАСТОСУНКУ ESPORTS TRACKER



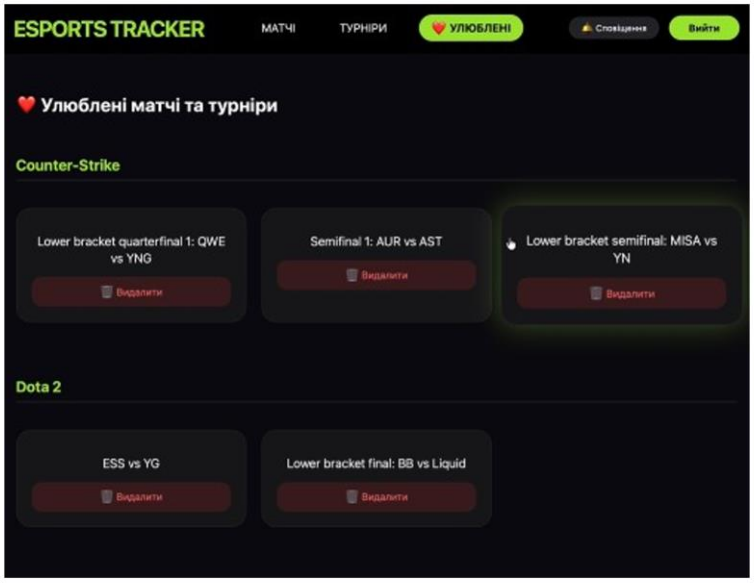
9

ВІКНО ДЕТАЛЬНОЇ ІНФОРМАЦІЇ ПРО МАТЧ

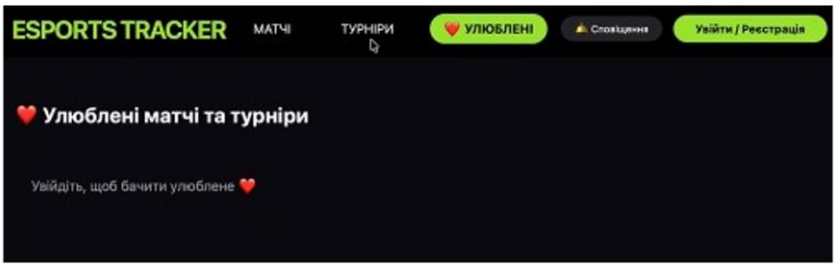


10

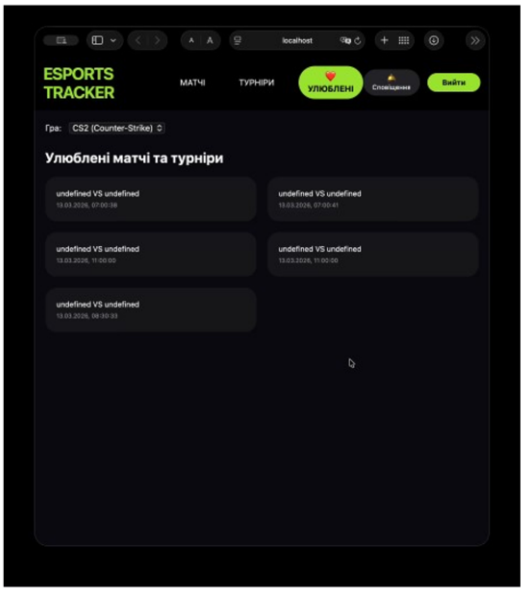
ІНТЕРФЕЙС РОЗДІЛУ «УЛЮБЛЕНІ МАТЧІ ТА ТУРНІРИ»



ВІДОБРАЖЕННЯ РОЗДІЛУ «УЛЮБЛЕНІ» ДЛЯ НЕАВТОРИЗОВАНОГО КОРИСТУВАЧА



ВІДЕО ВЕБ-ЗАСТОСУНКУ ESPORTS TRACKER



ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Анчев А.О. Інформаційна система підтримки користувачів у кіберспортивному середовищі з функціями моніторингу турнірів та новин. Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація », 2025 р., Київ, Державний університет інфраструктури та технологій . Збірник тез. К.: ДУІТ, 2025. Подано до друку.

2. Анчев А.О. Розробка та оптимізація веб-платформи для агрегації даних про кіберспортивні турніри та ігрові події. Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація », 2025 р., Київ, Державний університет інфраструктури та технологій . Збірник тез. К.: ДУІТ, 2025. Подано до друку.

ВИСНОВКИ

1. Проаналізовано теоретичні основи веб-систем моніторингу кіберспортивних подій та визначено тенденції розвитку галузі
2. Обґрунтовано використання клієнт -серверної архітектури та сучасних веб-технологій
3. Проведено аналіз існуючих платформ і виявлено їх недоліки (обмежена персоналізація , гнучкість)
4. Сформовано функціональні вимоги до системи «Esports Tracker»
5. Спроєктовано архітектуру системи та структуру бази даних
6. Реалізовано інтеграцію з API PandaScore для отримання актуальних даних
7. Розроблено веб-застосунок із використанням Node.js, Express та JavaScript
8. Реалізовано авторизацію , динамічний інтерфейс та систему улюблених подій

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б. ЛІСТИНГ

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Esports Tracker • CS2 • Dota 2 • LoL • Valorant</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <link rel="stylesheet" href="style.css">
</head>
<body class="bg-zinc-950 text-white min-h-screen">
<nav class="bg-black p-5 flex justify-between items-center sticky top-0 z-50">
  <h1 class="text-4xl font-bold text-lime-400 tracking-tighter">ESPORTS TRACKER</h1>
  <div class="flex gap-8 text-lg">
    <button onclick="showTab(0)" id="tab0" class="tab-btn active">МАТЧІ</button>
    <button onclick="showTab(1)" id="tab1" class="tab-btn">ТУРНІРИ</button>
    <button onclick="showTab(2)" id="tab2" class="tab-btn">♡ УЛЮБЛЕНІ</button>
  </div>
  <div class="flex items-center gap-4">
    <button onclick="requestNotificationPermission()" class="px-5 py-2 bg-zinc-800 hover:bg-lime-400 hover:text-black rounded-full text-
sm">🛎 Сповіщення</button>
    <button onclick="showLoginModal()" id="auth-btn" class="px-8 py-2 bg-lime-400 text-black font-bold rounded-full">Увійти /
Реєстрація</button>
  </div>
</nav>

<div class="max-w-7xl mx-auto p-6">
  <!-- ВИБІР ГРИ ІКОНКАМИ -->
  <div class="mb-10">
    <p id="game-label" class="text-zinc-400 mb-4">Оберіть гру:</p>
    <div id="game-switcher" class="flex gap-8 justify-center flex-wrap"></div>
  </div>

  <!-- ТАБ 0: Матчі -->
  <div id="tab-content-0" class="tab-content">
    <h2 class="text-3xl mb-6 font-bold">Матчі (live + upcoming)</h2>
    <div id="matches-grid" class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6"></div>
  </div>

  <!-- ТАБ 1: Турніри -->
  <div id="tab-content-1" class="tab-content hidden">
    <h2 class="text-3xl mb-6 font-bold">Турніри</h2>
    <div id="tournaments-grid" class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6"></div>
  </div>

  <!-- ТАБ 2: Улюблені -->
  <div id="tab-content-2" class="tab-content hidden">
    <h2 class="text-3xl mb-6 font-bold">♡ Улюблені матчі та турніри</h2>
    <div id="favorites-grid" class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6"></div>
  </div>
</div>
```

```
<!-- МОДАЛКА ДЕТАЛЕЙ -->
<div id="match-modal" onclick="if(event.target.id==='match-modal') closeModal('match-modal')"
    class="hidden fixed inset-0 bg-black/90 flex items-center justify-center z-[100]">
  <div class="bg-zinc-900 rounded-3xl p-8 max-w-2xl w-full mx-4 relative">
    <button onclick="closeModal('match-modal')" class="absolute top-6 right-6 text-5xl leading-none text-zinc-400 hover:text-lime-400">×</button>
    <div id="modal-content" class="mt-8"></div>
  </div>
</div>

<!-- МОДАЛКА АВТОРИЗАЦІЇ (повна) -->
<div id="auth-modal" class="hidden fixed inset-0 bg-black/90 flex items-center justify-center z-[100]">
  <div class="bg-zinc-900 rounded-3xl p-10 w-full max-w-md">
    <h3 id="modal-title" class="text-2xl mb-6 text-center">Увійти</h3>
    <input id="email" type="email" placeholder="Email" class="w-full p-4 bg-zinc-800 rounded-2xl mb-4 text-lg">
    <input id="password" type="password" placeholder="Пароль" class="w-full p-4 bg-zinc-800 rounded-2xl mb-6 text-lg">
    <button onclick="handleAuth()" id="auth-action-btn" class="w-full py-4 bg-lime-400 text-black font-bold text-lg rounded-2xl">Увійти</button>
    <p onclick="toggleAuthMode()" id="switch-text" class="text-center mt-6 text-lime-400 cursor-pointer">Немає акаунту? Зареєструйся</p>
  </div>
</div>
<script src="script.js"></script>
</body>
</html>
```