КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

(факультет)

Інформаційних технологій

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

на тему: «Букмекерська платформа без маржинальності та з динамічним скрапінгом»

Павліха Роман Володимирович
(прізвище, ім’я та по батькові магістра повністю)

Київ 2023 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

(факультет)

Інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Тетяна ГОНЧАРЕНКО
«___» _____ 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

на тему: «Букмекерська платформа без маржинальності та з динамічним скрапінгом»

Виконав: студент 2–го курсу, групи КНм-22

Спеціальності: 122 «Комп'ютерні науки»

(шифр і назва напряму підготовки, спеціальності)

Магістрант Павліха Р.В.

(прізвище та ініціали)

Керівник к.т.н., доц.. Гончаренко Т.А.

(прізвище та ініціали)

Рецензент к.т.н., доц. Баліна О.І.

(прізвище та ініціали)

Київ, 2023 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій .
Кафедра: інформаційних технологій .
Освітній рівень: «магістр за ОПП» .
Спеціальність: 122 «Комп'ютерні науки» .

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Тетяна ГОНЧАРЕНКО
_____
«____» _____ 2023 р.

З А В Д А Н Н Я
ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

1. Тема роботи: Букмекерська платформа без маржинальності та з динамічним скрапінгом.
затверджена наказом ректора КНУБА № 1280/2 від «26» червня 2023р.
2. Керівник роботи: к.т.н., доцент Гончаренко Тетяна Андріївна.
3. Строк подання студентом роботи до захисту: 9 грудня 2023р.
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області.
Р.2. Методи та алгоритми для системи букмекерська платформа.
Р.3. Розробка програмного інструментарію для системи букмекерська платформа.
Р.4. Результати розробки.
5. Інформаційні слайди:
С.1. Титульний лист.
С.2. Вступ.
С.3. Аналіз предметної області.
С.4.Класифікація букмкерських платформ.
С.5. Мета та постановка задачі.
С.6.Чорний ящик.

С.7.Загальні технології розробки.
С.8-10.Технології розробки.
С.11.Діаграма компонентів.
С.12-13.Діаграма класів.
С.14.Діаграми послідовності.
С.15-16.Блок-схеми матчингу.
С.17-21.Результати розробки.
С.22.Висновок.

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р.1. Аналіз предметної області.	Вересень 2023
Р.2. Методи та алгоритми для системи букмекерська платформа.	Жовтень 2023
Р.3. Розробка програмного інструментарію для системи букмекерська платформа.	Листопад 2023
Р.4. Результати розробки.	Листопад 2023
Остаточне оформлення роботи	Грудень 2023
Направлення роботи на рецензування, перевірку на плагіат	Грудень 2023
Попередній захист роботи на кафедрі	Грудень 2023

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Розділ 1.			
Розділ 2.			
Розділ 3.			
Розділ 4.			
Розділ 5.			

8. Дата видачі завдання: «26» Червня 2023р.

Керівник _____ Гончаренко Т.А.
(підпис) (прізвище та ініціали)

Магістрант _____ Павліха Р.В.
(підпис) (прізвище та ініціали)

РЕЗЮМЕ

Київський національний університет будівництва і архітектури

Павліха Роман Володимирович

факультет автоматизації і інформаційних технологій,

група КНм-22

Тема атестаційної випускної роботи: «Букмекерська платформа без маржинальності та з динамічним скрапінгом»

освітній рівень: магістр,

спеціальність: 122 «Комп'ютерні науки»,

Науковий керівник: Гончаренко Тетяна Андріївна

Обсяг роботи. Атестаційна випускова робота магістра складається: розділів 4, стор. 112, рис. 41, завдання, анотація, вступу, висновків, списку використаних джерел та додатків.

Актуальність теми. В сучасному світі, на тлі стрімкого розвитку технологій та зростання впливу інтернет-технологій в усіх сферах життя, букмекерські платформи стають важливим елементом глобальної індустрії розваг та спорту. Проте, із зростанням конкуренції та зміною уявлень клієнтів, виникає необхідність в нових підходах до організації цього бізнесу. У цьому контексті, розробка букмекерської платформи без маржинальності та з використанням динамічного скрапінгу стає важливим завданням, оскільки вона спрямована на оптимізацію коефіцієнтів та поліпшення користувацького досвіду. Такий підхід відкриває можливості для покращення ефективності та конкурентоспроможності букмекерських платформ в сучасному ринковому середовищі.

У вступі атестаційної випускної роботи присвячено розробці букмекерської платформи, що інтегрує динамічний скрапінг та принцип

безмаржинальності, відзначаючи важливість технологічно продуманих рішень у висококонкурентному світі букмекерського бізнесу.

У першому розділі «Аналіз предметної області» проведено аналіз букмекерської індустрії, включаючи класифікацію платформ, вимоги гравців та ключових акторів. Розглянуто критерії класифікації, такі як тип доступу, регіональна доступність, види ставок та використані технології. Визначені вимоги гравців включають надійність, широкий вибір ставок, конкурентні коефіцієнти та інші. Зазначено мету дослідження та сформульовано завдання для розробки букмекерської платформи без маржинальності з динамічним скрапінгом.

У другому розділі «Методи та алгоритми для системи букмекерська платформа» розглянуто використання SequenceMatcher для ефективного матчингу даних, автоматизований підхід до визначення співпадінь ліг та команд на основі івентів, інтеграцію сучасних технологій веб-розробки (React, WebSocket) для створення зручного інтерфейсу у букмекерській платформі.

У третьому розділі «Розробка програмного інструментарію для системи букмекерська платформа» розглядається розробка програмного інструментарію для букмекерської платформи, включаючи архітектуру, взаємодію ключових компонентів через діаграми компонентів та класів, і конкретизацію взаємодії скраперів, агрегатора, back-end та front-end через діаграми послідовностей для забезпечення ефективного функціонування системи.

У четвертому розділі «Результати розробки» представлені успішна імплементація скраперів для збору різноманітних даних, ефективна розробка агрегатору, який об'єднує дані в цілісну базу, та позитивна оцінка зручності та функціональності інтерфейсу, що свідчить про вдалий процес створення надійної букмекерської платформи для ставок на спортивні події.

Ключові слова: букмекерська платформа, маржинальність букмекерів, скрапінг, динамічний скрапінг, ставки, автоматизація.

Якість оформлення проекту. Атестаційна випускна робота магістра оформлена у відповідності до діючих нормативних документів та методичних вказівок до виконання дипломних робіт для студентів спеціальності 122 «Комп'ютерні науки». Порушень та зауважень під час розробки та перевірки дипломної роботи не виявлено.

Загальний висновок стосовно роботи та присвоєння авторові освітньо-кваліфікаційного рівня «магістр». Робота виконана якісно та на високому рівні, студент продемонстрував достатній рівень теоретичної підготовки та сформованих практичних навичок в області сучасних інформаційних технологій. Заслуговує оцінки «відмінно».

Науковий керівник _____/ Тетяна ГОНЧАРЕНКО/

(підпис)

Посада, місце роботи: в.о. зав. каф. ІТ КНУБА, доцент, к.т.н.

«04» грудня 2023р.

АНОТАЦІЯ

Павліха Р.В. «Букмекерська платформа без маржинальності та з динамічним скрапінгом».

Атестаційна випускна робота магістра за спеціальністю: 122 «Комп’ютерні науки». – Київський національний університет будівництва та архітектури. – Київ, 2023 р.

Атестаційна робота магістра присвячена букмекерській платформі без маржинальності та з динамічним скрапінгом. Результатом розробки є букмекерська платформа, якій не необхідна велика кількість людей для підтримки її роботи, автоматизоване отримання та обробка даних від інших букмекерських платформ. Букмекерська платформа наведена в цій роботі є захищеною від людського фактору і є привабливою для користувачів.

Ключові слова: Букмекерська платформа, маржинальність букмекерів, скрапінг, динамічний скрапінг, ставки, автоматизація.

ANNOTATION

Pavlikha R.V. "Betting Platform without Margin and with Dynamic Scraping".

Master's Degree Certification Thesis in the specialty: 122 "Computer Sciences". – Kyiv National University of Construction and Architecture. – Kyiv, 2023.

The master's thesis is dedicated to a betting platform without margin and with dynamic scraping. The outcome of the development is a betting platform that does not require a large number of people to support its operation, automated retrieval, and processing of data from other betting platforms. The betting platform described in this work is protected from human error and is attractive to users. Keywords: Betting platform, bookmakers' margin, scraping, dynamic scraping, bets, automation.

РЕЦЕНЗІЯ
на атестаційну випускну роботу

Студента Павліхи Романа Володимировича
. Факультет автоматизації і інформаційних технологій .
спеціальності 122 «Комп'ютерні науки».
Тема роботи: Букмекерська платформа без маржинальності та з динамічним скрапінгом.
Обсяг роботи: атестаційна випускова робота магістра складається: розділів 4, стор. 112, рис. 41, завдання, анотація, вступу, висновків, списку використаних джерел та додатків.
Висновок про відповідність завданню: робота виконана у повній відповідності до завдання і у встановлений термін .
Актуальність обраної теми: обрана тема дослідження є актуальною оскільки сприяє впровадженню прогресивних технологій та вдосконаленню сервісів у сфері букмекерських послуг, реагуючи на зростаючі вимоги користувачів та забезпечуючи більш ефективне функціонування ринку онлайн-ставок.
Використання у роботі сучасних досягнень науки і техніки: розробка проекту базується на використанні сучасних інформаційних комп'ютерних технологій
Використання у роботі комп'ютерних технологій: Docker, GIT, Python, HTML, CSS, JS, React, MySQL, Redis, Draw.io.
Практичне значення роботи: впровадження сучасних інформаційних технологій має забезпечувати виконання ряду вимог, у тому числі наявність зручного і дружнього інтерфейсу, забезпечення безпеки за допомогою різних методів контролю та розмежування доступу до інформаційних ресурсів, підтримку розподіленої обробки інформації.
Якість оформлення роботи: випускна робота оформлений у відпо-відності до діючих нормативних документів та методичних вказівок для студентів спеціальності 122 «КН»
Зауваження та побажання: Зауважень не виявлено
Загальний висновок стосовно роботи та надання авторові освітнього ступеня "магістр": робота виконана на високому рівні, студент продемонстрував високий рівень теоретичної підготовки та сформованих практичних навичок в області сучасних інформаційних технологій. Заслуговує оцінки «відмінно».

Рецензент _____/ _____доц., к.т.н., Баліна О.І. ____/
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Посада, місце роботи: доцент кафедри інформаційних технологій проєктування та прикладної математики КНУБА
«04» грудня 2023р.

ЗМІСТ

Вступ

Атестаційна випускна робота присвячена розробці букмекерської платформи, яка об'єднує в собі дві ключові інновації - динамічний скрапінг та принцип безмаржинальності. У сучасному світі конкуренція в галузі букмекерського бізнесу висока, і важливо розробляти технологічно продумані та високоефективні платформи, щоб задовольняти зростаючі вимоги користувачів.

Метою дослідження є розробка, яка буде відповідати наступним вимогам:

1. Впровадження безмаржинального підходу: Система повинна дозволяти формування коефіцієнтів виключно на основі ставок користувачів, що виключає додавання маржі букмекера.

2. Динамічний скрапінг даних: Забезпечити ефективний та актуальний збір даних з різних джерел за допомогою скраперів та агрегатора.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Розробка скраперів: Створення програмних засобів для автоматизованого отримання даних з букмекерських сайтів.

2. Створення агрегатора: Розробка інструменту, який об'єднує та обробляє інформацію від різних джерел.

3. Реалізація безмаржинальності: Розробка алгоритмів формування коефіцієнтів на основі ставок користувачів.

4. Тестування та вдосконалення: Проведення тестів та аналіз результатів для впровадження подальших покращень.

Мета цього проекту - не просто створити ще одну букмекерську платформу, але визначити новий стандарт, де вплив користувачів на формування коефіцієнтів стає ключовим елементом. В цій роботі розглядаються технічні аспекти та переваги використання динамічного скрапінгу та безмаржинального підходу в букмекерстві.

1. Аналіз предметної області

1.1. Дослідження дефініцій «Букмекерська платформа», «Маржинальність», «Скрапінг».

1.1.1. Визначення букмекерської платформи та історія її виникнення

Букмекерська платформа – це система, що дозволяє користувачам робити ставки на різноманітні спортивні події, розважальні заходи чи інші події, на які можна робити прогнози. Букмекерська платформа може існувати як у фізичному середовищі (наприклад, букмекерські контори) так і в онлайні, де користувачі можуть робити ставки через веб-сайти або мобільні додатки.[1]

Ідея робити ставки на різні події існувала вже давно. У давніх цивілізаціях, таких як Рим або Давня Греція, люди робили ставки на гладіаторські бої чи спортивні змагання.

З появою Інтернету букмекерський бізнес зазнав революції. Онлайн-ставки стали надзвичайно популярними завдяки зручності та широкому вибору подій. Це призвело до створення великої кількості букмекерських платформ у мережі. [2]

Сучасні букмекерські платформи використовують новітні технології, такі як мобільні додатки, штучний інтелект для аналізу даних і динамічне ціноутворення. Вони пропонують ставки не лише на спортивні події, але й на політичні, культурні та інші події.

Букмекерська платформа перетворилася з невеликого кіоску на іподромі в глобальний бізнес, що використовує передові технології для надання послуг своїм користувачам.

1.1.2. Визначення маржинальності в букмекерських платформах

Маржинальність у сфері платформ контор є ключовим поняттям, яке відіграє важливу роль у функціонуванні та прибутковості цього бізнесу. Вона представляє собою відсоток, які букмекерська контора утримує з ринкових

ставок, щоб гарантувати собі прибуток незалежно від конкретного результату спортивного заходу або іншої події, на яку приймаються ставки.[3]

Для кращого розуміння розглянемо математичний аспект:

Нехай O_A та O_B – це коефіцієнти на дві протилежні події (наприклад, перемога команди А і перемога команди Б відповідно). Тоді маржинальність (М) можна обчислити за формулою:

$$M = 1 - (\frac{1}{O_A} + \frac{1}{O_B})$$

Якщо результат буде позитивним, це вказує на те, що букмекер утримує додатковий відсоток з ринкових ставок.

Приклад: За припущенням коефіцієнтів на команди А та Б дорівнюють 1.90. Маржинальність розраховується так:

$$M = 1 - (\frac{1}{1.9} + \frac{1}{1.9}) \approx 0.0526$$

Отже, маржинальність дорівнює приблизно 5.26%.

Маржинальність є основним джерелом доходу для букмекерських контор. Вона також дозволяє букмекерам покривати свої оперативні витрати, ризики, пов'язані з потенційними втратами, а також гарантувати прибуток для своєї діяльності. Важливо зазначити, що розмір маржі може змінюватися в залежності від конкретної букмекерської контори, виду спорту або конкретної події.

1.1.3. Визначення скрапінгу

Скрапінг є процесом автоматичного витягування даних з веб-ресурсів. Це може бути як безпосередній перегляд веб-сторінок та парсинг HTML-коду, так і використання HTTP-реквестів до backend серверу сайту, аналогічно до того, як це робить frontend. Спеціалізовані програми або скрипти, відомі як скрапери, реалізують ці методи для збору визначеної інформації.[4]

Динамічний скрапінг відноситься до процесу збору даних із веб-сторінок, які часто оновлюють свій контент в реальному часі. Це може бути виконано за допомогою вебсокетів або регулярних HTTP-реквестів до

сервера. Такий підхід особливо корисний для букмекерських платформ, де інформація може швидко змінюватися.

Використання скрапінгу для збору даних про спортивні події пропонує платформам численні переваги:

- Економія коштів: Автоматизація процесу збору даних через скрапінг дозволяє платформам значно економити кошти. По-перше, зменшується потреба в наймі додаткових співробітників для ручного введення даних. По-друге, це уникає витрат на комерційні сервіси, які надають спортивну інформацію за плату.
- Актуальність даних: Забезпечення користувачів найновішою інформацією.
- Швидка реакція на зміни: Можливість оперативно реагувати на зміни в розкладі подій або іншої важливої інформації.

1.2. **Класифікація букмекерських платформ.**

Букмекерська індустрія є різноманітною та динамічною, існує безліч платформ, які надають послуги ставок. Щоб зрозуміти специфіку кожної платформи та її місце на ринку, потрібно звернутися до класифікації. На рисунку 1.1. представлені основні критерії, за якими можна класифікувати букмекерські платформи.

Рис. 1.1. Основні критерії класифікації букмекерських платформ

1.2.1. За типом доступу

- Офлайн букмекери

Опис: Традиційні пункти прийому ставок.

Переваги: Особистий контакт із клієнтами, можливість надати відразу фізичні виплати. [5]

Недоліки: Обмежене географічне охоплення, вищі операційні витрати.

- Онлайн букмекери

Опис: Вебсайти та мобільні додатки.

Переваги: Широке географічне охоплення, зручність для користувача, низькі операційні витрати.

Недоліки: Залежність від технічної інфраструктури, потенційні проблеми із безпекою.

1.2.2. За регіональною доступністю

- Локальні букмекери

Опис: Оперують у конкретній країні або регіоні.

Переваги: Мають глибоке розуміння місцевого ринку та споживачів.

Недоліки: Обмежений ринок, залежність від місцевого законодавства.

- Міжнародні букмекери

Опис: Приймають ставки з різних країн.

Переваги: Ширший охоп ринку, різноманітність ставок.

Недоліки: Складність управління, потреба адаптації до різних законодавств.

1.2.3. За видами ставок

- Спортивні ставки

Опис: Ставки на традиційні спортивні події.

Переваги: Велика популярність, багатий вибір подій.

Недоліки: Сезонність деяких видів спорту.

- Кіберспорт

Опис: Ставки на комп'ютерні ігри.

Переваги: Швидко зростаючий ринок, молоде покоління гравців.

Недоліки: Нестабільність ринку, проблеми із регулюванням.

- Живі ставки

Опис: Робляться під час тривання спортивної події.

Переваги: Висока зацікавленість, можливість відстеження ходу події.

Недоліки: Технічні вимоги, необхідність швидкої реакції.

- Віртуальний спорт

Опис: Ставки на моделювані спортивні заходи.

Переваги: Незалежність від реальних подій, регулярність.

Недоліки: Нижчий рівень довіри порівняно з реальними спортивними подіями.

1.2.4. За використаними технологіями

- Традиційні платформи

Опис: Стандартні методи обробки ставок.

Переваги: Простота управління.

Недоліки: Обмежені можливості, низька конкурентоспроможність.

- Інноваційні платформи

Опис: Інтеграція сучасних технологій.

Переваги: Високий рівень автоматизації, персоналізація для користувача.

Недоліки: Вищі витрати на впровадження та підтримку.

1.2.5. За ліцензуванням

- Ліцензовані

Опис: Оперують з дозволу регуляторних органів.

Переваги: Юридична чистота, довіра з боку гравців.

Недоліки: Високі витрати на отримання та підтримку ліцензії, жорстке регулювання.

- Неліцензовані

Опис: Працюють без ліцензій.

Переваги: Відсутність додаткових витрат на ліцензування.

1.2.6. За методом отримання даних про спортивні події

- Ручний ввід

Опис: Дані про спортивні події вводяться вручну операторами або користувачами платформи.

Переваги: Можливість точного контролю над введеними даними, гнучкість у виборі подій.

Недоліки: Велика ймовірність людських помилок, витратний процес введення.

- Автоматична інтеграція

Опис: Платформи можуть інтегруватися з іншими базами даних або постачальниками даних для автоматичного отримання інформації про спортивні події.

Переваги: Висока швидкість оновлення, зниження ймовірності помилок.

Недоліки: Залежність від зовнішніх постачальників, потенційні проблеми зі сумісністю.

- Скрапінг

Опис: Автоматичний процес витягування даних з інших веб-сайтів або платформ.

Переваги: Можливість швидко збирати великі обсяги даних, гнучкість у виборі джерел інформації.

Недоліки: Правові ризики, оскільки скрапінг може порушувати авторські права або умови використання деяких сайтів; технічні виклики, пов'язані з частими змінами структури сайтів, з яких здійснюється скрапінг.

1.2.7. За методом розрахунку коефіцієнтів

- Традиційна маржинальність

Опис: Коефіцієнти розраховуються на основі ймовірності конкретних спортивних результатів, з урахуванням статистики, думки експертів, стану

команд/спортсменів тощо. Додаткова маржа додається для гарантії прибутку букмекеру.

Переваги: Більш збалансовані коефіцієнти, які відображають реальну ймовірність результату.

Недоліки: Потреба в експертній оцінці, можливість великих втрат у випадку невірних прогнозів.

- Ринкова маржинальність (динамічна)

Опис: Коефіцієнти розраховуються виключно на основі ставок користувачів, реагуючи на зміни в попиті та пропозиції на ринку ставок. Такий підхід схожий на той, який використовується на біржах ставок.

Переваги: Автоматична адаптація до ринкових умов, менший ризик для букмекера (оскільки ризик перекладається на гравців).

Недоліки: Можливість маніпуляції коефіцієнтами великими гравцями, менш передбачувані результати для окремих ставок.

1.3. **Ключові актори та їх ролі**

В області букмекерського бізнесу є кілька ключових учасників, які відіграють важливу роль у функціонуванні та розвитку цієї індустрії. Кожен з цих учасників відіграє свою роль у букмекерському екосистемі, і взаємодія між ними допомагає індустрії розвиватися та просуватися вперед.

1. Букмекери: Це компанії або особи, які приймають ставки від гравців на різні події. Вони встановлюють коефіцієнти, визначають правила ставок та виплачують виграші.

2. Гравці (паріори): Це особи, які роблять ставки через букмекерські платформи. Вони можуть робити ставки як для розваги, так і в професійних цілях.

3. Регулятори: Це організації або урядові структури, які регулюють діяльність букмекерських контор. Вони встановлюють правила та стандарти

для букмекерів, з метою захисту інтересів гравців та забезпечення справедливості гри.

4. Постачальники даних: Це компанії, які збирають і надають статистичну інформацію про спортивні та інші події. Букмекери використовують ці дані для встановлення коефіцієнтів та аналізу ринку ставок.

5. Афіліати: Це партнери, які приводять нових гравців на букмекерські платформи через рекламу, маркетингові кампанії або спеціалізовані веб-сайти. Зазвичай вони отримують комісійні від букмекерів за кожного нового гравця.

6. Розробники ПЗ: Це компанії або індивідуальні розробники, які створюють програмне забезпечення для букмекерських платформ, включаючи веб-сайти, мобільні додатки, системи управління ризиками тощо.

7. Експерти та аналітики: Це особи, які спеціалізуються на аналізі спортивних подій, ринків ставок та інших аспектів букмекерського бізнесу. Вони можуть працювати на букмекерські платформи, або надавати консультативні послуги.

1.4. Вимоги та очікування гравців до букмекерських платформ

При виборі букмекерської платформи гравці звертають увагу на ряд критеріїв, які визначають їхній досвід користування та взаємодію з платформою. Для створення конкурентоспроможної та привабливої для користувачів платформи, розробники та власники букмекерських платформ повинні враховувати наступні вимоги та очікування гравців:

1. Надійність та безпека:

- Гравці очікують, що їхні особисті та фінансові дані будуть захищені.
- Ліцензування та регулювання платформи в конкретній країні може служити показником її надійності.

2. Широкий вибір ставок:

- Різноманітність спортивних подій, на які можна робити ставки, а також доступ до різних видів ставок (прямі, тотал, гандікап тощо).

3. Конкурентні коефіцієнти:

- Гравці шукають платформи, які пропонують найкращі коефіцієнти, забезпечуючи максимальний можливий виграш.

4. Швидкість виплат:

- Гравці цінують платформи, які забезпечують швидке та прозоре зняття коштів.

5. Доступність та зручність інтерфейсу:

- Інтуїтивно зрозумілий та зручний для користувача дизайн сайту або додатку, а також адаптивність під різні пристрої (мобільні телефони, планшети тощо).

6. Підтримка користувачів:

- Наявність ефективної служби підтримки, яка може швидко допомогти у вирішенні питань або проблем.

7. Акції та бонуси:

- Бонуси за реєстрацію, поповнення рахунку, акції для постійних гравців — ці фактори можуть привернути увагу нових гравців та стимулювати постійних користувачів продовжувати гру на платформі.

8. Трансляції матчів:

- Можливість перегляду спортивних подій в прямому ефірі безпосередньо через платформу є великим плюсом для багатьох гравців.

9. Об'єктивність та чесність:

- Гравці очікують, що платформа працює чесно, не маніпулюючи коефіцієнтами або результатами.

1.5. **Мета та постановка задачі дослідження**

Метою розробки букмекерської платформи без маржинальності та з динамічним скрапінгом є забезпечення користувачам альтернативного та прозорого середовища для здійснення ставок на спортивні події.

Задача полягає у розробці букмекерської платформи, що використовує дані, отримані з різних джерел, для надання користувачам актуальної та консолідованої інформації для ставок, а також враховує ставки користувачів для формування коефіцієнтів.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Розробка скраперів:

- Вивчення основних джерел, які містять інформацію про букмекерські ставки.
- Розробка скраперів для кожного із визначених джерел.
- Оптимізація процесу збору даних таким чином, щоб забезпечити максимальну швидкість та точність.

2. Створення агрегатора:

- Розробка системи, що отримує дані від усіх розроблених скраперів.
- Впровадження алгоритмів для матчингу однакових даних з різних платформ.
- Забезпечення зберігання зібраної інформації у структурованому та зручному для аналізу форматі.

3. Формування динамічних коефіцієнтів:

- Розробка системи, яка враховує ставки користувачів для визначення коефіцієнтів виграшу.

4. Розробка букмекерської платформи:

- Створення користувацького інтерфейсу, що відображає консолідовану інформацію, отриману від агрегатора.

- Впровадження функціоналу для ставок користувачів на платформі.

Схема «чорного ящика» спрощує розуміння роботи системи, виокремлюючи вхідні та вихідні дані, а також основні компоненти процесу.[6]

Схеми «чорного ящика» представлено на рисунках 1.2. – 1.5.

Рис. 1.2. Чорний ящик до «Скрапер букмекерських платформ»

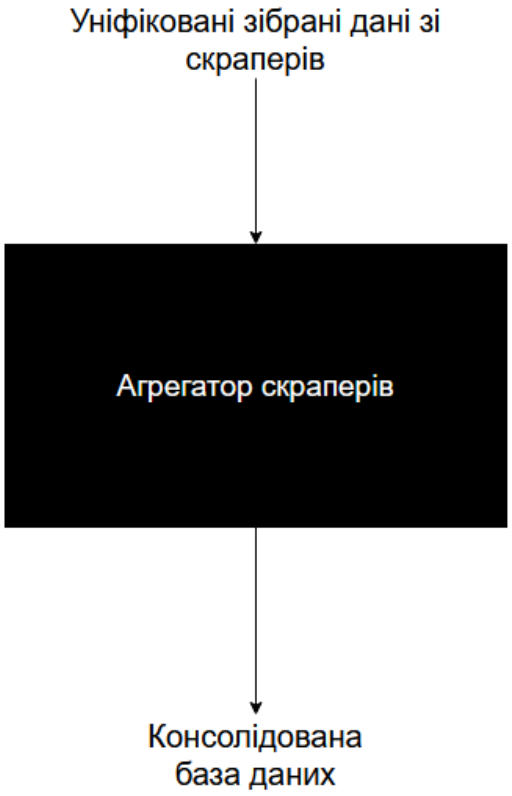


Рис. 1.3. Чорний ящик до «Агрегатор»

Рис. 1.4. Чорний ящик до «Формування динамічних коефіцієнтів»

Рис. 1.5. Чорний ящик до «Букмекерська платформа»

Висновок до розділу «Аналіз предметної області»

У цьому розділі був проведений аналіз предметної області, пов'язаної з букмекерською індустрією. Було розглянуто основні аспекти, які визначають функціонування та розвиток букмекерських платформ. Аналіз був спрямований на з'ясування класифікації букмекерських платформ, вимог та очікувань гравців, а також ключових акторів у цій індустрії.

Класифікація букмекерських платформ була представлена з урахуванням таких критеріїв, як тип доступу (онлайн та офлайн), регіональна доступність (локальні та міжнародні платформи), види ставок (спортивні, кіберспорт, живі та віртуальні ставки) та використані технології (традиційні та інноваційні платформи). Важливими були також критерії ліцензування та методи отримання даних про спортивні події.

Вимоги та очікування гравців до букмекерських платформ були визначені для створення конкурентоспроможної та привабливої платформи для користувачів. Вимоги включають надійність та безпеку, широкий вибір

ставок, конкурентні коефіцієнти, швидкість виплат, зручний інтерфейс, підтримку користувачів, акції та бонуси, трансляції матчів, об'єктивність та чесність.

Мета та постановка задачі дослідження були сформульовані з метою розробки букмекерської платформи без маржинальності та з динамічним скрапінгом для забезпечення користувачам альтернативного та прозорого середовища для здійснення ставок на спортивні події. Для досягнення цієї мети було визначено конкретні завдання, які включають розробку скраперів, створення агрегатора, формування динамічних коефіцієнтів та розробку букмекерської платформи.

Аналіз предметної області надає фундаментальне розуміння ключових аспектів букмекерської індустрії та визначає напрямки подальших досліджень та розробки букмекерської платформи без маржинальності.

2. Методи та алгоритми для системи букмекерська платформа

2.1. Загальні технології розробки

Docker є платформою для розробки, доставки та запуску додатків у контейнерах. Використання Docker дозволяє ізолювати додатки та їх залежності від операційної системи, що полегшує розгортання та забезпечує консистентність серед різних середовищ.[7]

Переваги Docker:

1. Ізоляція та Переносимість

Docker контейнери забезпечують ізольоване середовище для додатків, включаючи їхні залежності. Це робить їх переносимими між різними середовищами розробки, тестування та виробничими серверами.

2. Швидке Розгортання

Контейнери можна швидко створювати та запускати, забезпечуючи швидке розгортання та масштабування додатків.

3. Декларативне Середовище

Dockerfile, що використовується для конфігурації контейнера, дозволяє описувати середовище додатка декларативно. Це полегшує розгортання та управління залежностями.

Використання Docker в розробці:

1. Контейнеризація Сервісів

Кожна складова букмекерської платформи такі як скрапери, агрегатор, база даних та інші, може бути упакована в Docker контейнер, що дозволяє забезпечити ізоляцію та стабільність кожного сервісу.[8]

2. Розгортання на Локальному Середовищі

Розробка та тестування може відбуватися на локальному комп'ютері в ідентичному середовищі, яке використовується на виробничому сервері.

Логотип Docker представлено на рисунку 2.1.

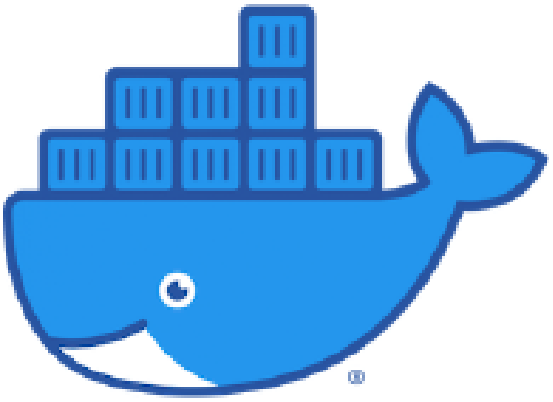


Рис. 2.1. Логотип Docker

Git – це розподілена система керування версіями, яка дозволяє зберігати історію змін у коді та спільно працювати над проектом в команді. Git надає засоби для відстеження та об'єднання змін, керування гілками та сприяє зручному спільному веденню кодової бази.[9]

Переваги використання Git:

1. Історія Змін:

Git забезпечує повну історію змін в коді, включаючи всі внесені зміни та коментарі до них. Це полегшує відстеження розвитку проекту та виявлення помилок.

2. Відгалуження та Злиття (Branching and Merging):

Git дозволяє створювати відгалуження для вирішення конкретних завдань, а потім об'єднувати їхні зміни в основну гілку. Це допомагає уникати конфліктів та забезпечує стабільність кодової бази.

3. Версіонування:

Завдяки Git, можна легко відновлювати попередні версії коду, якщо виникнуть проблеми або якщо потрібно повернутися до попереднього стану проекту.

Логотоп GIT представлено на рисунку 2.2.

Рис. 2.2. Логотип GIT

2.2. **Розробка скраперів**

2.2.1. Вибір технологій для розробки скраперів

Для реалізації задачі створення скраперів було обрано мову програмування Python. Обрання мови програмування Python для розробки методів та алгоритмів у системі букмекерської платформи є стратегічним рішенням, зумовленим кількома важливими факторами. Перш і найбільш вагомим аргументом у виборі Python є мій особистий досвід та кваліфікація як розробника на цій мові.[10]

Python відомий своєю простотою та зрозумілістю синтаксису, що полегшує розробку та підтримку коду. Крім того, величезна активна спільнота та багатофункціональні бібліотеки Python роблять його ідеальним вибором для широкого спектру завдань, включаючи розробку складних алгоритмів та систем обробки даних.[11]

Важливим фактором також є підтримка асинхронного програмування в мові Python.[12]

Логотип Python представлено на рисунку 2.3.

Рис. 2.3. Логотип Python

Використання aiohttp: Для асинхронних HTTP-запитів до веб-сайтів використовується бібліотека aiohttp.

Однією з ключових причин цього вибору є можливість бібліотеки надсилати асинхронні HTTP-запити. Це стає вирішальною перевагою у контексті оптимізації та прискорення процесу взаємодії із веб-серверами та збору даних.

Однією з основних причин використання асинхронних запитів є ефективне управління часом та ресурсами. Відмінною рисою aiohttp є можливість обробляти багато запитів паралельно без блокування виконання програми. Це особливо актуально в області букмекерських платформ, де інформація постійно змінюється і критично необхідно отримати найбільш актуальні дані.[13]

Використання aiohttp спрощує розробку алгоритмів, які вимагають паралельного виконання задач, а також забезпечує швидку відповідь від веб-серверів. Ця бібліотека володіє зручним інтерфейсом та високою продуктивністю, що робить її ідеальним інструментом для використання в букмекерських системах, де швидкість та точність грають ключову роль у наданні якісних послуг користувачам. Такий вибір сприяє покращенню продуктивності та надійності системи, що є пріоритетним завданням у сфері букмекерського бізнесу.[14]

Aiohttp активно підтримується та розривається з останніми трендами у світі асинхронного програмування. Це забезпечує не тільки сучасні можливості, але і гарантує актуальні патчі та оновлення безпеки.

Методи та алгоритми отримання даних:

aiohttp:

- Метод GET: Використовується для запиту контенту веб-сторінки.
- Метод POST: Для відправки даних на сервер, зокрема, для обходу авторизації чи відправки форм.
- Сесії: Дозволяють підтримувати стабільне з'єднання, зокрема для авторизованих запитів.
- Обробка винятків: Для виявлення та обробки помилок під час запиту.

Логотип aiohttp представлено на рисунку 2.4.

Рис. 2.4. Логотип aiohttp

Websockets: У випадках, коли веб-сайти використовують технологію websockets для передачі інформації, скрапери повинні використовувати відповідні інструменти для ефективного збору даних. Тут невід'ємною ролью виступає бібліотека websockets, яка дозволяє підключатися до вебсокетів та отримувати дані в режимі реального часу.

Багато сучасних веб-сайтів використовують технологію websockets для інтерактивного оновлення вмісту сторінки без необхідності повторних HTTP-запитів. Отже, для скрапінгу даних з таких ресурсів використання websockets стає ключовим елементом. Бібліотека websockets надає зручний та ефективний інтерфейс для підключення та взаємодії з веб-сокетами.[15]

Використання websockets у скрапері дозволяє отримувати актуальну інформацію миттєво після її зміни на веб-сайті. Це особливо важливо для систем букмекерських платформ, де швидкість реакції на зміни у подіях та коефіцієнтах може визначити успіх чи невдачу.

Такий підхід дозволяє ефективно та оперативно збирати дані, забезпечуючи коректну та актуальну інформацію для подальшого використання в системі букмекерської платформи. Загальною перевагою використання websockets є зменшення затримок при зборі даних та підвищення ефективності взаємодії з сучасними веб-ресурсами [16]

websockets:

- Асинхронне з'єднання: Встановлення двостороннього постійного з'єднання між клієнтом і сервером.
- Підписка на події: Дозволяє отримувати оновлення в реальному часі без необхідності постійних запитів до сервера.

2.2.2. Методи обходу обмежень на скрапінг

Деякі веб-сайти встановлюють заходи безпеки для уникнення скрапінгу, включаючи блокування IP-адрес, оскільки вони розуміють, що часті запити від одного IP можуть свідчити про автоматизовану діяльність, а не взаємодію реальної людини.[17]

Ротація User-Agents**:** Зміна ідентифікатора браузера (User-Agent) для кожного запиту може ускладнити виявлення автоматизованої діяльності. Використання різних User-Agents, які наслідують поведінку реальних браузерів, допомагає підтримувати анонімність та зменшує ризик блокування за несправжній або неправильний ідентифікації.

Ротація IP-адрес: Використовуючи різні мережеві точки доступу або сервіси VPN. Ротація дозволяє розподілити запити через різні адреси, унеможливлюючи виявлення та блокування конкретного адресного простору.

Використання проксі-серверів: допомагають розподілити навантаження та запити через різні IP-адреси. Це також служить як засіб уникнення блокування та забезпечення стабільної роботи скрапера

Затримка між запитами: Додавання випадкових затримок між запитами з одного IP-адреси може імітувати непередбачувану людську активність. Це може включати в себе випадкові інтервали затримки між запитами, враховуючи при цьому тип даних та специфіку веб-сайту.

Використання цих методів є необхідним у випадках, коли важливо уникати блокування IP-адрес та забезпечити ефективне та безперебійне скрапінг даних з веб-сайтів, які встановлюють подібні обмеження.[18]

Ці методи дозволяють уникнути блокування IP-адрес та забезпечити стабільну та неперервну роботу скрапера навіть у випадках високих обмежень безпеки, які встановлюють веб-сайти.

2.2.3. Уніфікація даних отриманих зі скраперів

Для забезпечення ефективного функціонування букмекерської платформи необхідно впроваджувати методи уніфікації даних, отриманих із різних джерел за допомогою скраперів. Цей процес включає в себе стандартизацію форматів, обробку відхилень та конвертацію інформації в єдиний структурований формат.[19]

Джерела часто представляють дані в різних форматах та структурах. Тому важливо забезпечити, щоб на виході зі скраперів дані були уніфіковані, що спрощує подальший аналіз та інтеграцію даних.

1. Стандартний формат даних: Визначення стандартого формату даних, який буде прийматися системою.

2. Трансформація даних: Після збору даних з конкретного джерела, необхідно застосувати алгоритми та методи трансформації, щоб перевести ці

дані в стандартний формат. Це може включати в себе відсіювання непотрібної інформації, перейменування полів, конвертацію типів даних та інше.

3. Валідація даних: Щоб забезпечити якість та достовірність даних, важливо валідувати їх перед зберіганням або передачею в наступні системи. Валідація може виявити непередбачені помилки, такі як неправильні формати дати, відсутність значень або неочікувані символи.

У вигляді алгоритмів, цей процес може включати в себе перевірку наявності та правильність ключових полів, нормалізацію одиниць вимірювання, а також розв'язання можливих конфліктів та неоднозначностей у даних. Застосування цих алгоритмів допомагає забезпечити консистентність та якість інформації, що є критичним для коректного функціонування букмекерської платформи.

2.2.4.Передача даних зі скраперів

Використання Redis для передачі даних: Після збору та обробки даних за допомогою скраперів, наступним кроком є передача цих даних до агрегатора. Для ефективної і швидкої передачі даних між скраперами та агрегатором було обрано Redis [20] - високопродуктивну інструментальну систему зберігання даних типу "ключ-значення".

Логотип Redis представлено на рисунку 2.5.

Рис. 2.5. Логотип Redis

Плюси Redis: [21]

1. Швидкість і асинхронність:
   - Redis працює у пам'яті, що забезпечує надзвичайно високу швидкість читання та запису даних. Це особливо важливо для скраперів, які повинні ефективно збирати та передавати великі обсяги даних. Асинхронність Redis дозволяє скраперам продовжувати роботу без очікування завершення передачі даних.
2. Простота реалізації:
   - Redis пропонує простий та ефективний інтерфейс для зберігання та отримання даних. Його легкість використання робить його ідеальним для використання у скраперах, де ефективність та швидкість грають ключову роль.
3. Розділення даних за скраперами:
   - Кожен скрапер може мати власний Redis, що дозволяє розділяти дані між різними букмекерськими платформами. Це унеможливлює конфлікти та забезпечує ізольованість даних.
4. Можливість кешування:
   - Redis надає можливість кешування даних. Це дозволяє зберігати тимчасові дані, що зменшує навантаження на інші частини системи та поліпшує загальну продуктивність.
5. Підтримка структурованих даних:
   - Redis не просто кешує рядки, але також підтримує роботу зі структурованими даними, такими як списки, хеші та множини. Це дозволяє зберігати дані у форматі, що найкраще підходить для конкретного типу інформації.

2.3. **Розробка агрегатору**

2.3.1. Вибір технологій для розробки агрегатору

Обрано мову програмування Python через його велику кількість бібліотек для роботи з мережевими запитами[22], обробки даних та реалізації алгоритмів матчингу. Python також володіє зручним синтаксисом та широкою спільнотою розробників.

Вибір системи управління базами даних (СУБД) є критичним етапом при розробці агрегатору для букмекерської платформи. У даному випадку для забезпечення надійності та ефективності було обрано MySQL.[23]

Стабільність. MySQL відомий своєю стабільністю та надійністю. Велика кількість компаній використовує MySQL для обробки великого обсягу даних, що свідчить про його високу стабільність.

Відмовостійкість. MySQL має механізми відмовостійкості, такі як реплікація та резервне копіювання, що дозволяють уникнути втрати даних у випадку аварій або відмов системи.

Швидкодія. MySQL відзначається високою швидкодією операцій з читання та запису даних, що є важливим аспектом для агрегатора, який повинен ефективно обробляти та зберігати великий обсяг інформації.

2.3.2. Підготовка даних до матчингу

На першому етапі агрегації даних необхідно забезпечити відсів невалідних символів у назвах, а також підготовити дані для подальшого матчингу.

1. Ввід даних: В даному етапі ми отримуємо назви команд та ліг із різних джерел, такі як букмекерські платформи чи інші джерела.

2. Перевірка символів: Виконуємо перевірку кожного символу у назві на належність до дозволених символів. Наприклад, у визначеній змінній ALLOWED_SYMBOLS міститься список дозволених символів (літери, цифри, пробіли тощо).

3. Вилучення заборонених символів: Будь-які символи, які не належать до списку дозволених символів, вилучаються з тексту.

4. Нижній регістр: Перетворення тексту в нижній регістр для забезпечення однорідності та уникнення помилок через регістр символів.

5. Вивід очищеного тексту: Результатом цього етапу є очищений текст назви команди або ліги, який містить лише дозволені символи та записаний в нижньому регістрі.

2.3.3. Алгоритм матчингу

У процесі розробки агрегатора для букмекерської платформи ключовим питанням стало визначення схожості даних з різних скраперів. Для цього було обрано метод SequenceMatcher з модуля difflib мови програмування Python.[24]

Основи SequenceMatcher: SequenceMatcher - це клас, який надає методи для порівняння пар рядків. Головний алгоритм, який використовується, базується на підрахунку співпадінь у послідовностях і визначенні ступеня схожості між двома рядками.

Математичний зміст SequenceMatcher метод ratio повертає дійсне число у діапазоні [0, 1], що представляє відношення подібності двох послідовностей. Воно обчислюється за формулою:

$$ratio = \frac{2*matches}{len(seq1)+len(seq2)}$$

де *matches* - це кількість символів, які співпадають в обох послідовностях, а seq1 та seq2 перша та друга послідовності.

Принцип роботи:

Основна ідея SequenceMatcher базується на алгоритмі "наївного" пошуку найбільшої спільної підпослідовності (LCS — Longest Common Subsequence)[25]. Для отримання високоефективності, difflib використовує техніку швидкого пошуку найбільших спільних підпослідовностей, використовуючи теорію множин для визначення неперетинаючихся підпослідовностей.[26]

За допомогою методу ratio(), SequenceMatcher обчислює ступінь подібності між двома рядками як відношення числа співпадінь до загального числа символів у обох рядках. Результат - це число в діапазоні від 0 (повне невідповідання) до 1 (повне співпадіння).

Застосування в проекті: У контексті нашого агрегатора, SequenceMatcher використовується для визначення схожості назв ліг, івентів та інших рядкових даних отриманих від різних скраперів. Якщо ступінь подібності перевищує певний поріг (наприклад, 0.8), система вважає, що рядки відповідають одне одному.

Переваги та обмеження: Використання SequenceMatcher має кілька переваг:

- Простота інтеграції завдяки вбудованому в Python модулю difflib.
- Висока точність при правильному налаштуванні порогового значення.

Однак існують і обмеження:

- Може виявитися менш ефективним при величезних об'ємах даних через витрати на обчислення.
- Може потребувати додаткової настройки для оптимальної роботи в специфічних умовах агрегатора.

Висновок: Використання SequenceMatcher виявилося ефективним рішенням для задачі матчингу даних в агрегаторі нашої букмекерської платформи. Даний підхід дозволив нам забезпечити високу якість визначення співпадінь між даними з різних джерел.

Матчинг ліг на основі івентів

У процесі агрегації даних із різних букмекерських платформ може виникнути виклик віднесення івентів до відповідних ліг. Однак, використовуючи алгоритмічний підхід, можливо автоматизувати процес визначення співпадінь між лігами на основі матчингу івентів.

Принцип роботи: Ідея полягає в тому, що якщо певний івент може бути зматчений між двома платформами, то й ліга, до якої цей івент належить,

також є однаковою на обох платформах. Тобто, при виявленні співпадінь у івентах, можна вважати, що відповідні ліги також співпадають.

Алгоритмічний підхід: Після того, як івенти було зматчено між двома платформами за допомогою SequenceMatcher, система перевіряє, до якої ліги кожен із цих івентів належить на своїй платформі. Якщо івенти належать до різних ліг на своїх платформах, система визначає ці ліги як співпадіння та з'єднує їх у своєму сховищі даних.

Переваги даного підходу:

- Автоматичне визначення співпадінь між лігами, що зменшує необхідність ручної інтервенції.
- Підвищення точності матчингу, оскільки івенти є більш конкретними і мають менше варіантів співпадінь порівняно з лігами.

Обмеження та уточнення: Хоча цей метод є досить ефективним для визначення співпадінь між лігами, важливо також враховувати можливість виникнення помилок у випадках, коли івенти можуть належати до різних ліг на різних платформах. Така ситуація може виникнути через особливості класифікації ліг на конкретних платформах.

Висновок: Матчинг ліг на основі івентів є прогресивним підходом у процесі агрегації даних з різних букмекерських платформ. Використання даного методу дозволяє автоматизувати процес матчингу та підвищити якість консолідованої бази даних.

Алгоритм матчингу команд на основі івентів та часових рамок

Вибірка івентів за часовими рамками

Перед тим як приступити до матчингу команд, важливо відсіяти івенти, які не підходять за часом проведення. Це може значно зменшити обсяг даних для аналізу.

- Визначте часове вікно для матчингу. Наприклад, +/- 15 хвилин від планового часу початку івенту.
- Виберіть івенти, які підходять під ці рамки.

Процес матчингу команд

Використовуючи івенти, які були відфільтровані на першому етапі, переходимо до безпосереднього матчингу команд.

- Для кожної команди з першого джерела:
  - Знайдіть відповідні команди з другого джерела, які знаходяться в межах визначеного часового вікна.
  - Застосуйте SequenceMatcher.ratio() для порівняння назв команд.
  - Виберіть команду з найвищим співвідношенням як найбільш імовірний кандидат для матчингу.

2.4. **Розрахунки коефіцієнтів на основі ставок користувачів (без маржинальності)**

Основною метою цього процесу є визначення коефіцієнтів виграшу для різних варіантів ставок на основі сум ставок користувачів. Даний підхід спрямований на підвищення привабливості букмекерської платформи для користувачів, дозволяючи їм отримувати більший прибуток.

2.4.1. Методика розрахунку

Методика розрахунку коефіцієнтів на основі ставок користувачів полягає в використанні загальної суми ставок на конкретний варіант результату події та визначенні коефіцієнта виграшу, який представляється на букмекерській платформі.

- K - коефіцієнт виграшу для конкретного користувача на конкретний результат.
- S (opposite) - сума ставок на інші варіанти результату (інших користувачів).
- S (result) - сума ставок на конкретний результат (включаючи ставку конкретного користувача).

Коефіцієнт виграшу може бути розрахований за наступною формулою:

$$K = \frac{S_{opposite}}{S_{result}} + 1$$

2.4.2. Заробіток з коефіцієнтів

Під час розробки букмекерської платформи без маржинальності, основною метою є створення ефективного та прибуткового сервісу для користувачів, які бажають здійснювати ставки на спортивні події. Однак з відсутністю маржинальності на коефіцієнтах, букмекерська компанія повинна знайти інші способи заробітку. Одним із таких способів є введення комісії з виграних ставок.

Відсоткова комісія обчислюється від суми виграшу користувача після розгляду ставки. Наприклад, при введенні 5% комісії, якщо гравець виграє 100 одиниць валюти, його виграш буде 95 одиниць, а решта 5 одиниць буде складати прибуток букмекерської компанії.

2.5. **Розробка букмекерської платформи**

Під час розробки букмекерської платформи, основними інструментами є React для фронтенду та HTML, CSS, JS (JavaScript) для створення користувацького інтерфейсу та логіки роботи платформи. У цьому розділі ми розглянемо методи та алгоритми, які можуть бути використані під час розробки букмекерської платформи.

2.5.1. Веб дизайн

Розробка Зовнішнього Вигляду: Для створення привабливого та зручного інтерфейсу використовується HTML та CSS. Правильна організація структури HTML-документів і використання CSS-стилів допомагають створити ефективний дизайн для користувачів. [27]

Адаптивний Дизайн: Розробка адаптивного дизайну за допомогою CSS та медіазапитів, що дозволяють адаптувати інтерфейс до різних пристроїв та розмірів екранів користувачів.[28]

2.5.2. Клієнтська логіка

Використання React: React є потужним інструментом для розробки клієнтської логіки. Він дозволяє створювати компоненти, які реагують на події користувачів та оновлюють інтерфейс на основі змін у даних. [29]

Маршрутизація: Використання бібліотеки React Router для реалізації маршрутизації, яка дозволяє переключатися між сторінками без перезавантаження сторінки. [30]

2.5.3. Взаємодія з сервером

Взаємодія з API: Використання AJAX-запитів та використання бібліотек, для взаємодії з сервером та отримання даних, необхідних для відображення на сторінці.[31]

Робота з WebSocket: Використання WebSocket-з'єднань для підтримки живих ставок та отримання оновлень даних в режимі реального часу.

Висновок до розділу «Методи та алгоритми для системи букмекерська платформа»

У цьому розділі розглянуто ключові аспекти матчингу даних та розробки букмекерської платформи. Використання SequenceMatcher виявилося ефективним для визначення схожості між даними з різних джерел, надаючи простий і точний метод. Автоматизований підхід до визначення співпадінь ліг на основі івентів спростовує процес і поліпшує консолідацію бази даних. Матчинг команд на основі івентів та часових рамок, сприймаючи розрахунки коефіцієнтів на основі ставок користувачів, відображає важливий етап у розробці букмекерської платформи. Застосування сучасних технологій веб-розробки, таких як React та WebSocket, допомагає створити зручний та функціональний інтерфейс для користувачів. В цілому, інтеграція цих елементів формує повноцінну платформу, яка поєднує в собі ефективний матчинг даних та високотехнологічну інфраструктуру букмекерського сервісу.

3. Розробка програмного інструментарію для системи букмекерська платформа

3.1. Діаграми компонентів

Діаграма компонентів є потужним інструментом для візуалізації структури та взаємодії компонентів системи. У контексті букмекерської платформи, діаграма компонентів дозволяє чітко визначити основні елементи системи, їх функції та взаємозв'язки.[32]

Діаграма компонентів представлена на рисунку 3.1.

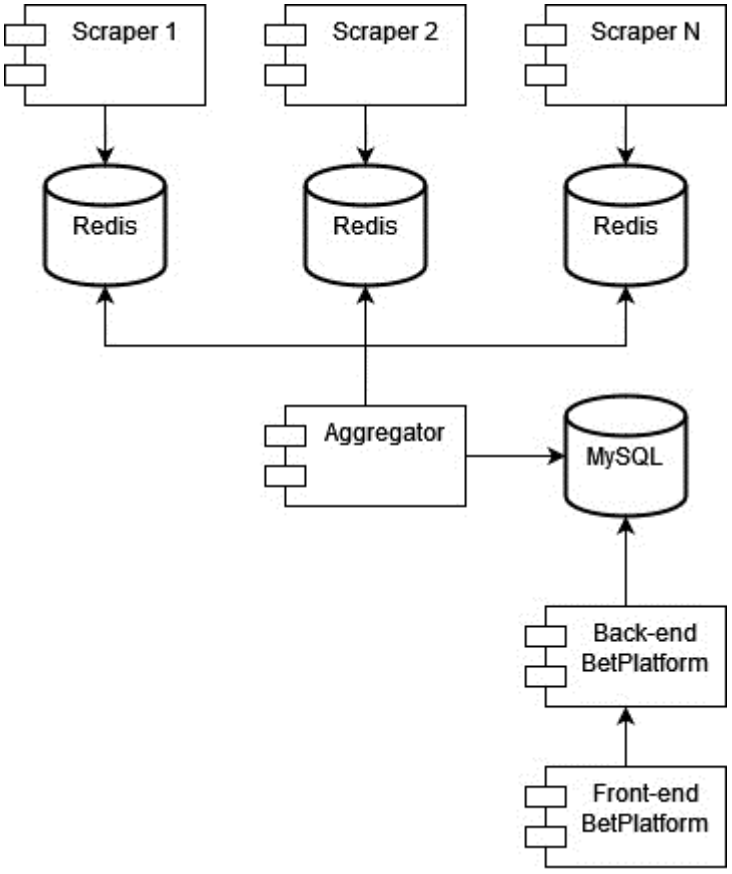


Рис. 3.1. Діаграма компонентів системи букмекерська платформа

Основні компоненти:

1. Scraper 1, Scraper 2, ..., Scraper N:
   - Кожен скрапер виконує роль взаємодії із зовнішніми букмекерськими платформами та зберігає отримані дані у відповідний Redis.

Кожен скрапер є незалежним компонентом, який обслуговує конкретну букмекерську платформу.

2. Redis:

- Для зберігання інформації використовується Redis. Кожен скрапер має свій власний Redis, що забезпечує розділення даних та запобігає конфліктам при записі.

3. Aggregator:

- Агрегатор відповідає за збір та об'єднання даних із різних Redis. Він використовує алгоритми матчингу для ідентифікації однакових подій та зберігає уніфіковані дані у MySQL-базі даних.

4. MySQL:

- База даних MySQL служить основним сховищем для уніфікованих даних. Вона забезпечує надійне зберігання та доступ до інформації для бекенду букмекерської платформи.

5. Back-end букмекерської платформи:

- Back-end відповідає за логіку бізнес-логіки, включаючи розрахунок коефіцієнтів, обробку ставок користувачів та взаємодію з базою даних. Він отримує дані з MySQL та надає їх для фронтенду через API.

6. Front-end букмекерської платформи:

- Front-end відповідає за відображення інформації для користувачів та надає зручний інтерфейс для роботи з букмекерською платформою. Він отримує дані від бекенду через API та відображає їх користувачам.

Взаємодія між компонентами:

- Кожен скрапер асинхронно взаємодіє з Redis для зберігання даних.
- Агрегатор періодично отримує дані із всіх Redis, використовуючи алгоритми матчингу для ідентифікації однакових подій.

- Оголошення нових даних агрегатором активує процес оновлення MySQL-бази даних.
- Back-end здійснює взаємодію з MySQL для отримання уніфікованих даних та розрахунку коефіцієнтів.
- Front-end використовує API для отримання актуальної інформації та відображення її для користувачів.

Завдання кожного компонента:

- Scraper: Забезпечити збір даних з букмекерських платформ та їх зберігання в Redis.
- Redis: Забезпечити швидкий та доступний доступ до даних для скраперів та агрегатора.
- Aggregator: Об'єднувати дані з різних скраперів, використовуючи алгоритми матчингу та зберігати у MySQL.
- MySQL: Забезпечити надійне та структуроване зберігання даних для бекенду.
- Back-end: Обробляти логіку бізнес-процесів, розраховувати коефіцієнти та взаємодіяти з MySQL.
- Front-end: Забезпечити зручний інтерфейс для користувачів та відображення актуальної інформації.

3.1.1. Компоненти агрегатора

Агрегатор, призначений для об'єднання даних від різних скраперів, є складною системою, що включає в себе кілька ключових компонентів:

Датагеттери: Ці компоненти служать для отримання уніфікованих даних від кожного скрапера. Їх головна задача - забезпечити стійке та швидке підключення до Redis, з якого вони отримують дані.

Форматтери: Після отримання даних, вони можуть потребувати подальшої трансформації. Форматтери адаптують дані до стандартного формату, що використовується в агрегаторі.

Валідатори: Ці компоненти перевіряють уніфіковані дані на наявність помилок, відсутність необхідних полів та інші можливі проблеми. Завдяки валідаторам забезпечується якість та консистентність даних.

Матчинг даних: Один з найважливіших компонентів агрегатора. Він використовує алгоритми порівняння, щоб визначити, чи відповідають одне одному різні записи з різних скраперів (наприклад, чи є "Ліга А" зі скрапера 1 тією ж самою "Лігою А" зі скрапера 2).Коли матчинг завершено, результати записуються в консолідовану базу даних.

Запис в базу даних: Після обробки, трансформації, валідації та матчингу даних, вони записуються в консолідовану базу даних. Це забезпечує їх централізоване зберігання та легкий доступ для подальшого використання.

Адміністративна панель: Для контролю, перевірки та ручного матчингу даних необхідно створити інтуїтивно зрозумілу адміністративну панель. Вона дозволить операторам системи:

- Переглядати зібрані та зматчені дані.
- Вручну матчити записи, які алгоритми не змогли автоматично визначити як ідентичні.
- Відстежувати ефективність роботи скраперів.
- Управляти процесом агрегації та матчингу.

3.2. **Діаграми класів**

Діаграма класів - це вид діаграми в мові моделювання UML (Unified Modeling Language), яка відображає структуру та взаємозв'язки класів у системі програмного забезпечення. Класи представляють собою шаблони для створення об'єктів, і діаграма класів визначає їх атрибути (властивості) та методи (операції), а також взаємозв'язки між ними.[33]

3.2.1. Діаграма класів агрегатора

Діаграма класів агрегатора включає низку ключових класів та їх взаємодію для обробки даних, отриманих від скраперів, та їх подальшого збереження.

Діаграму класів агрегатора представлено на рисунку 3.2.

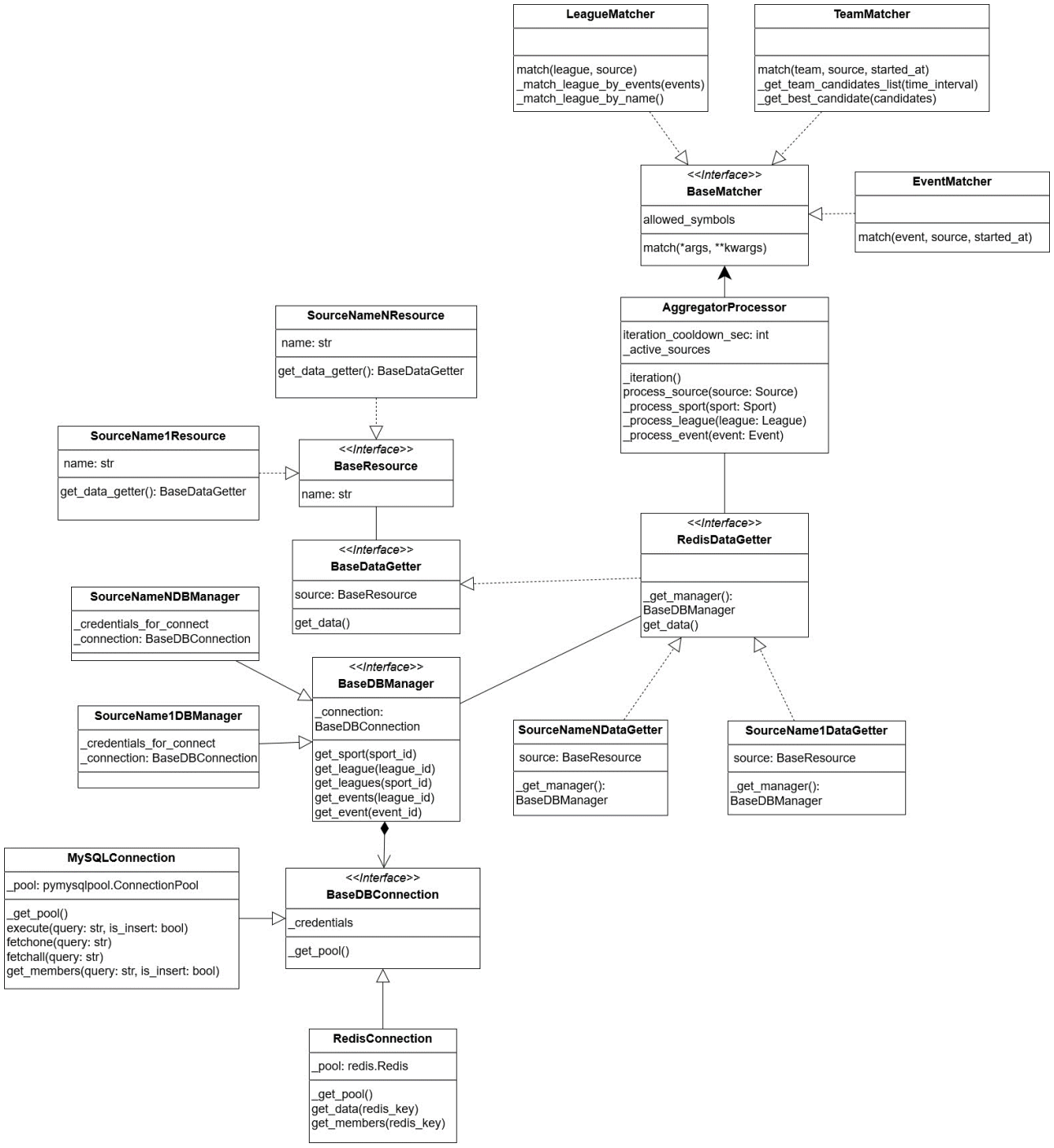


Рис. 3.2. Діаграма класів агрегатора

Нижче подано опис та взаємозв'язки цих класів:

1. LeagueMatcher

Методи:

- match(league, source): визначає відповідність ліги конкретному джерелу.
- _match_league_by_events(events): визначає відповідність ліги на основі подій.
- _match_league_by_name(): визначає відповідність ліги за назвою.

2. TeamMatcher

Методи:

- match(team, source, started_at): визначає відповідність команди конкретному джерелу.
- _get_team_candidates_list(time_interval): отримує список можливих кандидатів команди за певний часовий інтервал.
- _get_best_candidate(candidates): обирає найкращого кандидата зі списку.

3. EventMatcher

Методи:

- match(event, source, started_at): визначає відповідність події конкретному джерелу.

4. BaseMatcher (інтерфейс)

Параметри:

- allowed_symbols: дозволені символи.

Методи:

- match(*args, **kwargs): метод для визначення відповідності.

5. AggregatorProcessor

Параметри:

- iteration_cooldown_sec: часовий інтервал між ітераціями.
- _active_sources: активні джерела.

Методи:

- _iteration(): виконує ітерацію агрегатора.
- process_source(source): обробляє дані для конкретного джерела.
- _process_sport(sport): обробляє дані для конкретного виду спорту.
- _process_league(league): обробляє дані для конкретної ліги.
- _process_event(event): обробляє дані для конкретної події.

6. RedisDataGetter (інтерфейс)

Методи:

- _get_manager(): отримує менеджер Redis.

7. SourceNameNDataGetter

Параметри:

- source: джерело даних.

Методи:

- _get_manager(): отримує менеджер даних для конкретного джерела.

8. SourceName1DataGetter

Параметри:

- source: джерело даних.

Методи:

- _get_manager(): отримує менеджер даних для конкретного джерела.

9. BaseDataGetter (інтерфейс)

Параметри:

- source: джерело даних.

Методи:

- get_data(): отримує дані відповідно до вказаного джерела.

10. BaseResource (інтерфейс)

Параметри:

- name: ім'я ресурсу.

Методи:

- get_data_getter(): отримує об'єкт, який може отримувати дані.

11. SourceNameNDataGetter

Параметри:

- name: ім'я ресурсу.

Методи:

- get_data_getter(): отримує об'єкт, який може отримувати дані.

12. SourceName1DataGetter

Параметри:

- name: ім'я ресурсу.

Методи:

- get_data_getter(): отримує об'єкт, який може отримувати дані.

13. BaseDBManager (інтерфейс)

Параметри:

- _connection: з'єднання з базою даних.

Методи:

- get_sport(sport_id): отримує дані про вид спорту.
- get_league(league_id): отримує дані про лігу.
- get_leagues(sport_id): отримує дані про ліги для певного виду спорту.
- get_events(league_id): отримує дані про події для певної ліги.

14. SourceNameNDBManager

Параметри:

- _credentials_for_connect: дані для підключення.
- _connection: з'єднання з базою даних.

15. SourceName1DBManager

Параметри:

- _credentials_for_connect: дані для підключення.
- _connection: з'єднання з базою даних.

16. BaseDBConnection (інтерфейс)

Параметри:

- _credentials: дані для підключення.

Методи:

- _get_pool(): отримує пул підключень до бази даних.

17. RedisConnection

Параметри:

- _pool: пул підключень до Redis.

Методи:

- _get_pool(): отримує пул підключень до Redis.
- get_data(redis_key): отримує дані з Redis за ключем.
- get_members(redis_key): отримує список учасників (members) Redis за ключем.

18. MySQLConnection

Параметри:

- _pool: пул підключень до MySQL.

Методи:

- _get_pool(): отримує пул підключень до MySQL.
- execute(query, is_insert): виконує SQL-запит, повертає результат.
- fetchone(query): отримує один рядок з результатів SQL-запиту.
- fetchall(query): отримує всі рядки з результатів SQL-запиту.
- get_members(query, is_insert): отримує список учасників (members) за SQL-запитом.

3.2.2. Діаграма класів скрапера

Діаграма класів скрапера охоплює низку ключових класів, які взаємодіють між собою для ефективного отримання, перевірки, форматування та обробки даних з букмекерських ресурсів.

Діаграму класів скраперу представлено на рисунку 3.3.

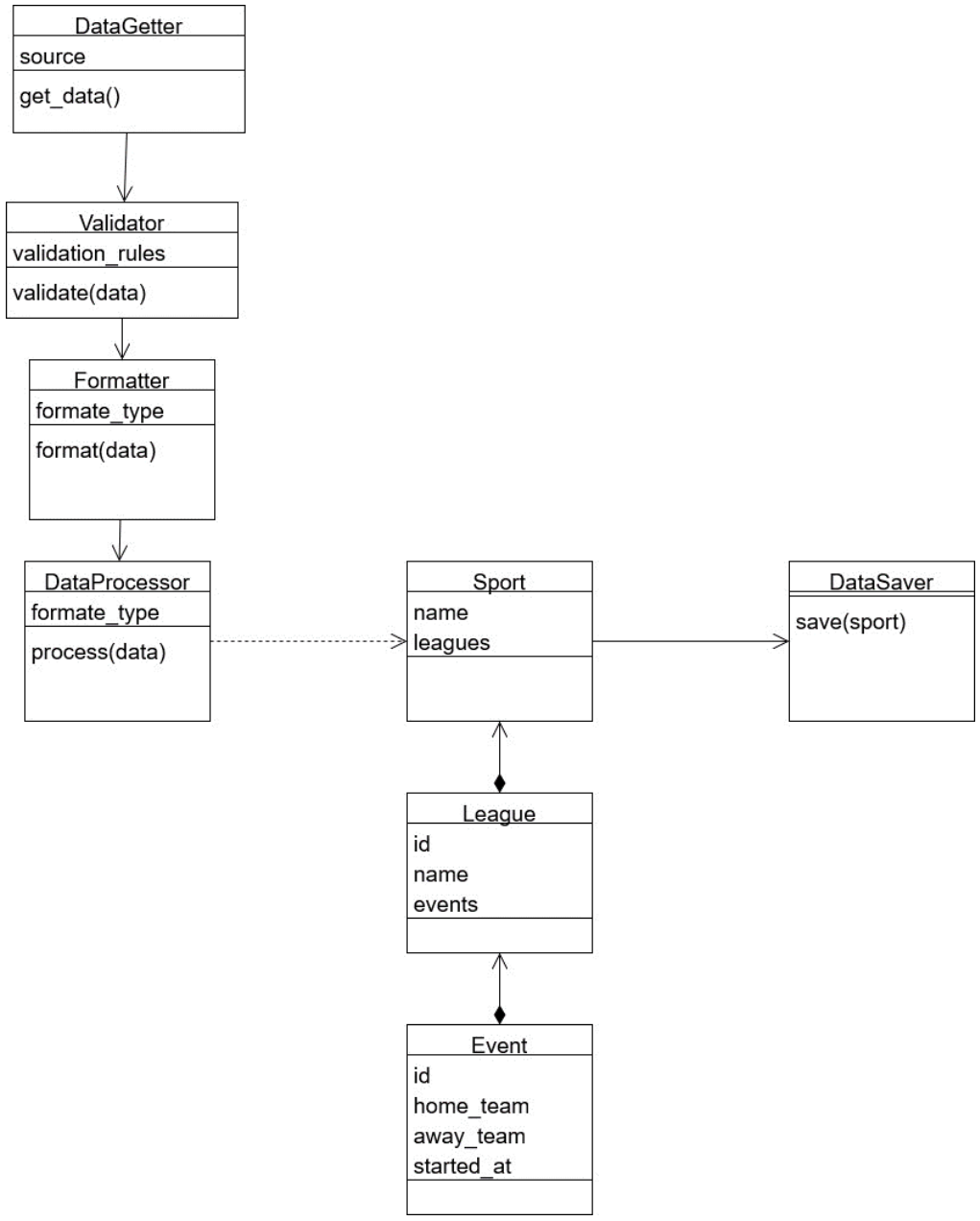


Рис. 3.3. Діаграма класів скрапера

Нижче подано опис та взаємозв'язки цих класів.

1. DataGetter:

Параметри:

- source: інформація про джерело даних, з якого отримуються дані.

Методи:

- get_data(): ініціює процес отримання даних з визначеного джерела.

2. Validator:

Параметри:

- validation_rules: правила валідації для перевірки даних.

Методи:

- validate(data): використовує встановлені правила для перевірки валідності даних.

3. Formatter:

Параметри:

- format_type: тип формату, який слід використовувати для даних.

Методи:

- format(data): використовує визначений тип формату для форматування даних.

4. DataProcessor:

Методи:

- process(): обробляє та підготовлює дані для подальшого використання.

5. Sport:

Параметри:

- name: ім'я виду спорту.
- leagues: список ліг, пов'язаних з цим видом спорту.

19. DataSaver:

Методи:

- save(): зберігає оброблені дані для подальшого використання.

20. League:

Параметри:

- id: унікальний ідентифікатор ліги.
- name: назва ліги.
- events: список подій, що відбуваються в цій лізі.

21. Event:

Параметри:

- id: унікальний ідентифікатор події.

- home_team: команда-господар події.
- away_team: команда-гість події.
- started_at: час початку події.

3.2.3. Діаграма класів Back-end букмекерської платформи

Діаграма класів back-end компоненту букмекерської платформи відображає структуру та взаємодію класів, що визначають основні функції та обробку даних на серверному рівні системи.

Діаграму класів Back-end букмекерської платформи представлено на рисунку 3.4.

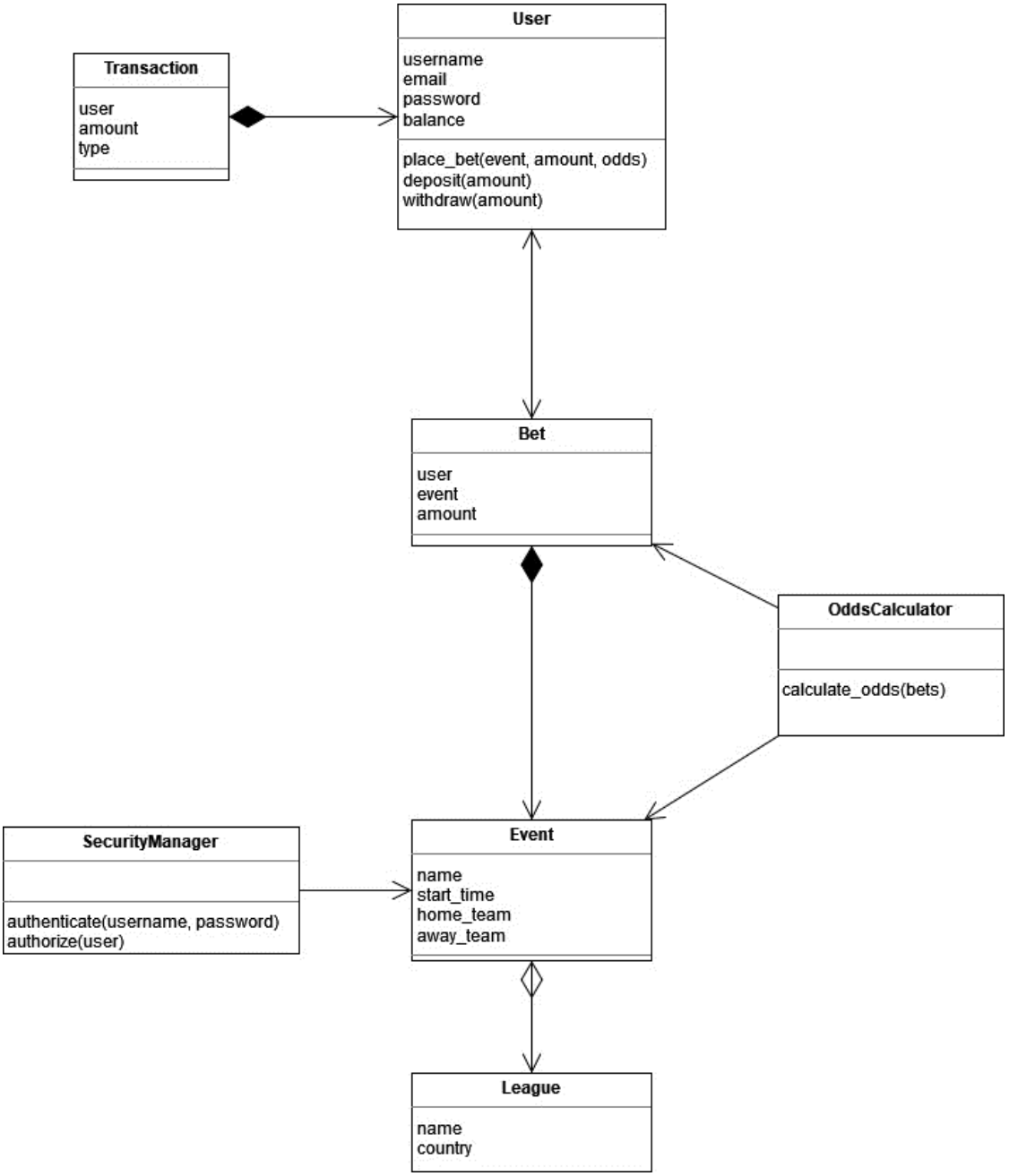


Рис. 3.4. Діаграма класів Back-end букмекерської платформи

Нижче подано опис та взаємозв'язки цих класів.

1. Клас "User" (Користувач):

Параметри:

- username: ім'я користувача
- email: електронна пошта користувача
- password: пароль користувача
- balance: баланс користувача

Методи:

- place_bet(event, amount, odds): розміщення ставки на певну подію
- deposit(amount): здійснення депозиту
- withdraw(amount): виведення грошей

2. Клас "Bet" (Ставка):

Параметри:

- user: посилання на користувача, який розмістив ставку
- event: посилання на подію, на яку робиться ставка
- amount: сума ставки
- odds: коефіцієнти для ставки

3. Клас "Event" (Подія):

Параметри:

- name: ім'я події
- start_time: час початку події
- teams: список команд, які беруть участь у події

4. Клас "League" (Ліга):

Параметри:

- name: ім'я ліги
- country: країна, до якої відноситься ліга

5. Клас "Transaction" (Транзакція):

Параметри:

- user: посилання на користувача, з яким пов'язана транзакція

- amount: сума транзакції
- type: тип транзакції (депозит, виведення)

6. Клас "OddsCalculator" (Калькулятор коефіцієнтів):

Методи:

- calculate_odds(bets): розрахунок коефіцієнтів на основі ставок

7. Клас "SecurityManager" (Менеджер безпеки):

Методи:

- authenticate(username, password): автентифікація користувача
- authorize(user, role): авторизація користувача з визначеними правами

Ця діаграма класів відображає ключові компоненти back-end букмекерської платформи та їх взаємодію, сприяючи розумінню структури системи та основних операцій, які вона виконує.

3.3. **Діаграми послідовності**

Діаграма послідовності є видом діаграми, який використовується для моделювання взаємодії між об'єктами в системі відносно часу. Вона дозволяє проектувальникам і розробникам відобразити порядок обміну повідомленнями між різними об'єктами або класами в рамках конкретного сценарію або функціональності.[34]

3.3.1. Діаграма послідовності отримання даних агрегатором від скраперів

Діаграма послідовності відображає процес отримання даних від різних скраперів та їх подальшу обробку із застосуванням AggregatorProcessor. У цьому процесі ключовими є AggregatorProcessor, SourceNameNResource, та SourceNameNDataGetter.

Діаграма послідовності отримання даних агрегатором від скраперів представлено на рисунку 3.5.

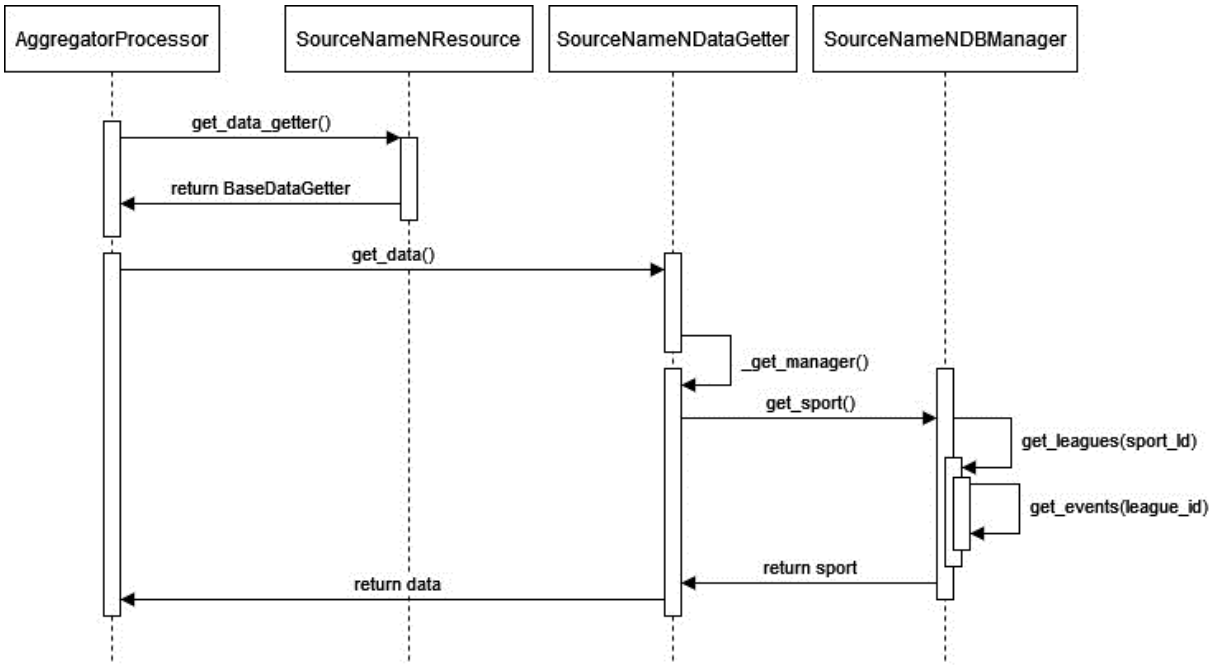


Рис. 3.5. Діаграма послідовності отримання даних агрегатором від скраперів

1. AggregatorProcessor:

Викликає метод get_data_getter() у SourceNameNResource.

Отримує повідомлення про отримання об'єкта типу BaseDataGetter через повернення BaseDataGetter.

2. SourceNameNResource:

Отримує повідомлення get_data_getter() від AggregatorProcessor.

Повертає об'єкт типу BaseDataGetter через повідомлення return BaseDataGetter.

3. AggregatorProcessor (продовження):

Кидає повідомлення get_data() у SourceNameNDataGetter.

Отримує повідомлення про отримання об'єкта типу data через повернення data.

4. SourceNameNDataGetter:

Отримує повідомлення get_data() від AggregatorProcessor.

Кидає повідомлення _get_manager() собі.

5. SourceNameNDataGetter (продовження):

Кидає повідомлення get_sport() у SourceNameNDBManager.

Отримує повідомлення про отримання об'єкта типу sport через повернення sport.

Кидає повідомлення get_leagues(sport_id) та get_events(league_id) у SourceNameNDBManager.

6. SourceNameNDBManager:

Отримує повідомлення get_sport() від SourceNameNDataGetter.

Кидає повідомлення get_leagues(sport_id) та get_events(league_id) собі.

7. SourceNameNDBManager (продовження):

Повертає повідомлення sport SourceNameNDataGetter через повернення sport.

Повертає повідомлення data SourceNameNDataGetter через повернення data.

8. SourceNameNDataGetter (продовження):

Повертає повідомлення data AggregatorProcessor через повернення data.

3.3.2. Діаграма послідовності матчингу агрегатора

У процесі матчингу в агрегаторі використовується діаграма послідовності для ілюстрації взаємодії між різними об'єктами.

Діаграму послідовності матчингу агрегатора показано на рисунку 3.6.

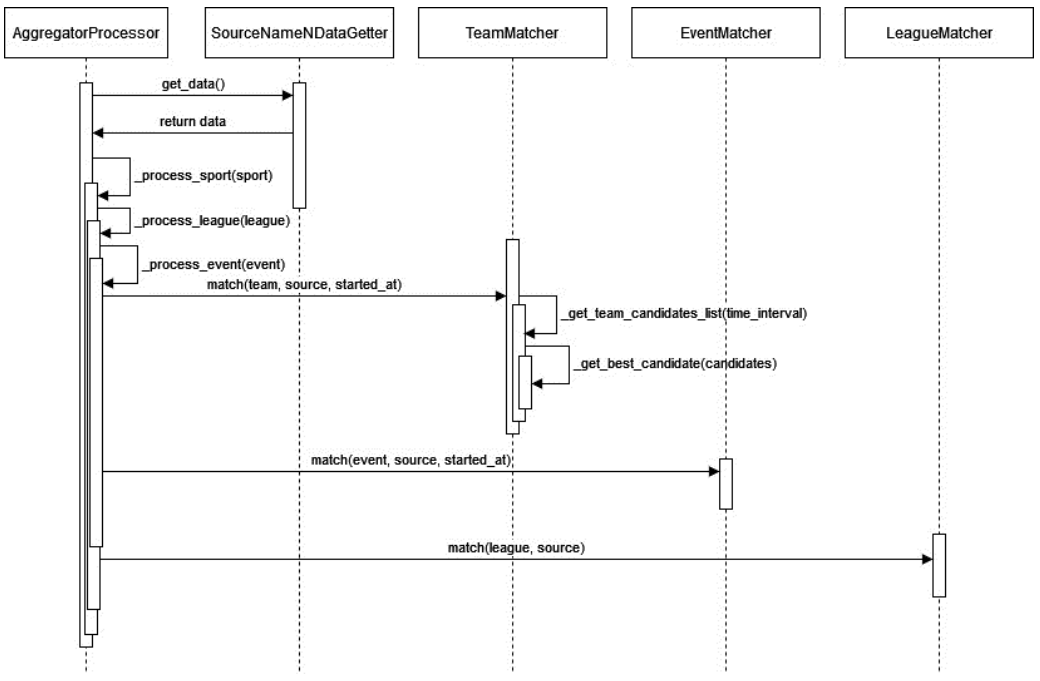


Рис. 3.6. Діаграма послідовності матчингу агрегатора

Нижче наведено деталізовану послідовність подій:

AggregatorProcessor кидає повідомлення get_data() в SourceNameNDataGetter:

AggregatorProcessor активує метод отримання даних від конкретного джерела (наприклад, SourceNameN).

SourceNameNDataGetter відповідає, повертаючи отримані дані.

AggregatorProcessor кидає повідомлення собі _process_sport(sport):

AggregatorProcessor викликає внутрішній метод обробки спорту (_process_sport), передаючи отримані дані про спорт.

AggregatorProcessor кидає повідомлення собі _process_league(league):

Після обробки спорту, AggregatorProcessor викликає внутрішній метод обробки ліги (_process_league), передаючи дані про лігу.

AggregatorProcessor кидає повідомлення собі _process_event(event):

Завершивши обробку ліги, AggregatorProcessor викликає метод обробки івенту (_process_event), передаючи дані про івент.

AggregatorProcessor кидає повідомлення match(team, source, started_at) в TeamMatcher:

AggregatorProcessor ініціює матчинг команди, передаючи інформацію про команду, її джерело та час початку.

AggregatorProcessor кидає повідомлення match(event, source, started_at) в EventMatcher:

Затемнюється матчинг івенту, передаючи інформацію про івент, його джерело та час початку.

AggregatorProcessor кидає повідомлення match(league, source) в LeagueMatcher:

Нарешті, AggregatorProcessor ініціює матчинг ліги, передаючи інформацію про лігу та її джерело.

3.3.3. Діаграма послідовності скраперів

У цьому розділі представлена діаграма послідовності для процесу скрапінгу даних, який виконується скраперами. Кожен з класів виконує свої функції в цьому процесі.

Діаграму послідовності скраперів показано на рисунку 3.7.

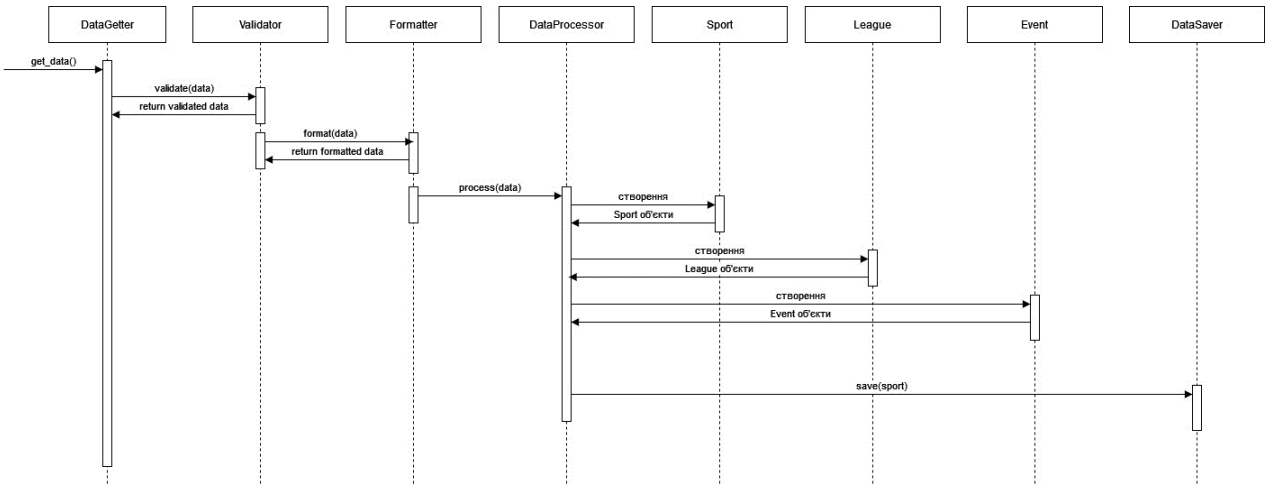


Рис. 3.7. Діаграма послідовності скраперів

1. DataGetter

Скрапер викликає DataGetter, передаючи інформацію про джерело даних.

DataGetter викликає свій метод get_data() для отримання даних з джерела.

2. Validator

Отримані дані передаються в Validator для валідації за встановленими правилами.

Validator викликає метод validate(data) для перевірки правильності даних.

3. Formatter

Після валідації дані подаються на форматування.

Formatter викликає свій метод format(data) для приведення даних до певного формату.

4. DataProcessor

Дані, які пройшли валідацію та форматування, передаються DataProcessor для обробки.

DataProcessor викликає свій метод process() для застосування обробки до даних.

5. Sport, League, Event

Опрацьовані дані розділяються на різні об'єкти, такі як Sport, League та Event, в залежності від структури даних.

Створюються об'єкти Sport, League та Event з відповідними атрибутами.

6. DataSaver

Створені об'єкти передаються DataSaver для збереження відформатованих та оброблених даних.

DataSaver викликає свій метод save() для збереження даних.

3.4. **Блок-схеми матчингу**

В рамках розробки алгоритму матчингу для обробки даних про команди з різних джерел, було створено блок-схеми для візуального представлення кроків алгоритму та його структури. Блок-схеми є графічними представленнями послідовності операцій та умов, які визначають логіку виконання алгоритму матчингу.[35]

Мета використання блок-схем:

Візуалізація алгоритму: Блок-схеми надають можливість візуалізувати логіку роботи алгоритму матчингу, допомагаючи зрозуміти послідовність операцій та умов.

Полегшення розуміння: Цей графічний метод допомагає розробникам та іншим учасникам проекту краще розуміти структуру та логіку алгоритму матчингу, зменшуючи можливість непорозумінь.

Аналіз та оптимізація: Блок-схеми дозволяють легко аналізувати алгоритм для виявлення можливих шляхів оптимізації та вдосконалення його продуктивності.

3.4.1. Блок-схема матчингу команд

Блок-схема "Матчинг команди" описує основні кроки та рішення, які виконуються під час процесу матчингу команди.

Блок-схема матчингу команд представлена на рисунку 3.8.

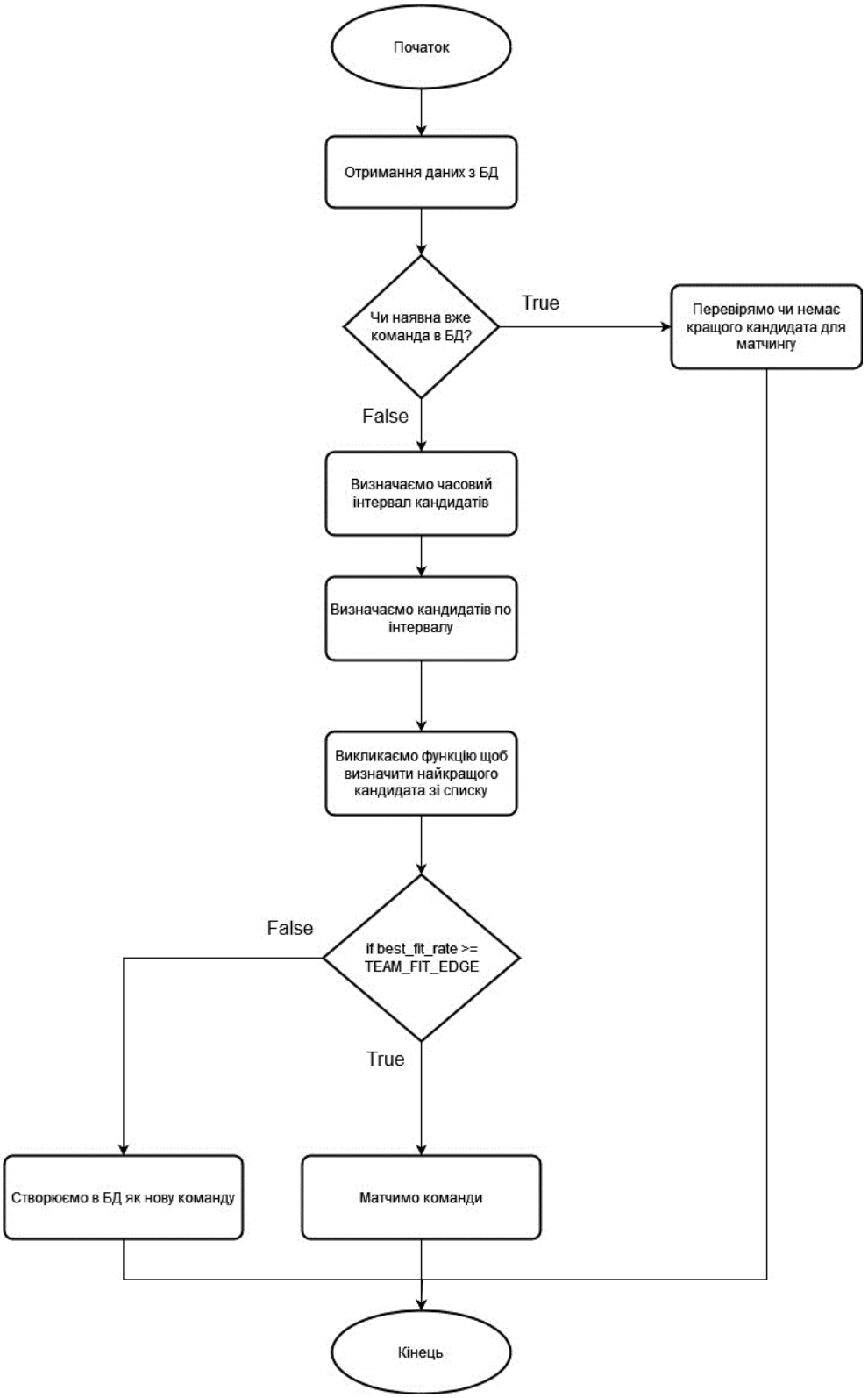


Рис. 3.8. Блок-схема матчингу команд

1. Початок:

Починаємо алгоритм матчингу команди.

2. Отримання даних з БД:

Здійснюємо операцію отримання даних про команду з бази даних.

3. Чи наявна вже команда в БД?

Якщо команда вже знаходиться в базі даних:

False: Переходимо до визначення часового інтервалу для кандидатів.

Визначаємо кандидатів по встановленому інтервалу.

Викликаємо функцію для визначення найкращого кандидата зі списку.

Якщо best_fit_rate >= TEAM_FIT_EDGE, переходимо до матчингу команд (кінець).

Якщо best_fit_rate < TEAM_FIT_EDGE, створюємо новий запис про команду в БД (кінець).

True: Перевіряємо, чи немає кращого кандидата для матчингу (кінець).

4. Матчинг команд:

Якщо у нас є кандидат, який відповідає заданому порогу матчингу (best_fit_rate >= TEAM_FIT_EDGE), здійснюємо матчинг команди.

5. Створення нової команди в БД:

Якщо виявлено, що немає кращого кандидата для матчингу (порог не досягнутий), створюємо новий запис про команду в БД.

6. Кінець:

Завершення алгоритму матчингу команди.

3.4.2. Блок-схема визначення найкращого кандидату зі списку команд

Блок-схема "Визначення найкращого кандидата" описує логіку та послідовність операцій, що відбуваються під час визначення найкращого кандидата зі списку.

Блок-схему визначення найкращого кандидату зі списку команд представлено на рисунку 3.9.

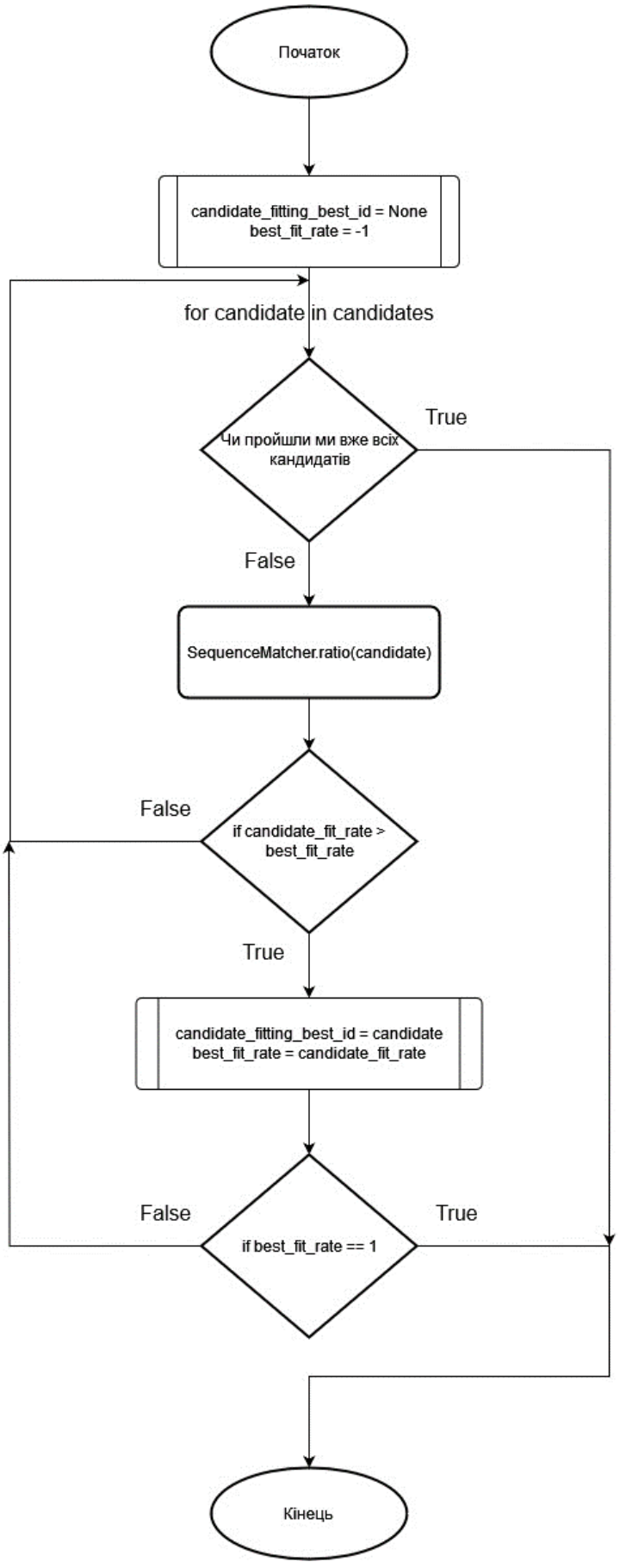


Рис. 3.9. Блок-схема визначення найкращого кандидату зі списку команд

1. Початок:

Ініціалізуємо змінні candidate_fitting_best_id та best_fit_rate на початкові значення.

2. candidate_fitting_best_id = None; best_fit_rate = -1:

Початкові значення для визначення найкращого кандидата.

3. for candidate in candidates:

Цикл, який проходить всіх кандидатів зі списку.

4. Чи пройшли ми вже всіх кандидатів:

False: Виконуємо порівняння кандидата з поточним кращим кандидатом.

Якщо candidate_fit_rate > best_fit_rate, оновлюємо значення best_fit_rate та candidate_fitting_best_id.

Переходимо до наступної ітерації циклу.

True: Завершення циклу, всі кандидати перевірені.

5. if best_fit_rate == 1:

False: Перевірка, чи є кандидат, для якого порівняння повністю відповідає (рівність 1).

Якщо best_fit_rate == 1, переходимо до наступної ітерації циклу.

True: Завершення циклу, так як знайдений кандидат, що повністю відповідає.

6. Кінець:

Завершення блок-схеми визначення найкращого кандидата.

3.4.3. Блок-схема матчингу івентів

Блок-схема матчингу івентів визначає логіку та послідовність операцій для визначення відповідності івента серед кандидатів.

Блок-схема матчингу івентів показана на рисунку 3.10.

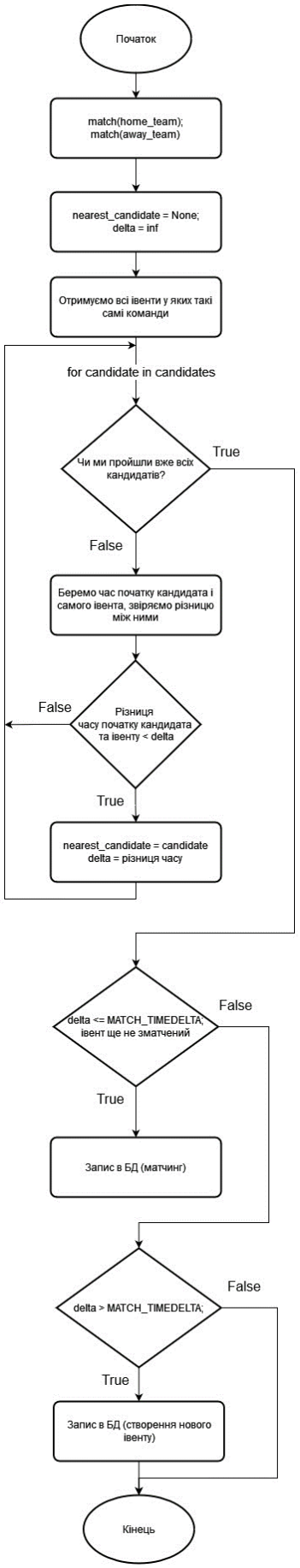


Рис. 3.10. Блок-схема матчингу івентів

1. Початок:

Запуск алгоритму матчингу для команди "home_team" та "away_team".

2. match(home_team); match(away_team):

Виклик функцій матчингу для команд "home_team" та "away_team".

3. nearest_candidate=None; delta=inf:

Ініціалізація змінних для визначення найближчого кандидата та різниці часу.

4. Отримуємо всі івенти у яких такі самі команди:

Отримання списку івентів, у яких зустрічаються обрані команди.

5. for candidate in candidates:

Цикл, що перебирає всіх кандидатів зі списку.

6. Чи ми дійшли вже всіх кандидатів?:

False: Виконуємо порівняння часу початку кандидата та івента.

Якщо різниця часу менше за поточну мінімальну, оновлюємо значення "nearest_candidate" та "delta".

Переходимо до наступної ітерації циклу.

True: Завершення циклу, всі кандидати перевірені.

7. delta<= MATCH_TIMEDELTA; івент ще не зматчений:

True: Запис в базу даних (матчинг івенту).

False: Перевірка, чи різниця часу менше або рівна заданому порогу.

9. delta<= MATCH_TIMEDELTA; івент ще не зматчений:

True: Запис в базу даних (створення нового івенту).

False: Завершення алгоритму матчингу.

3.4.4. Блок-схема матчингу ліг

Блок-схема матчингу ліг описує процес визначення відповідності ліги серед кандидатів.

Блок-схема матчингу ліг представлено на рисунку 3.11.

Рис. 3.11. Блок-схема матчингу ліг

1. Початок:

Запуск алгоритму матчингу для ліг.

2. Отримання даних з БД:

Звернення до бази даних для отримання інформації про ліги.

3. Чи наявна вже ліга в БД?:

True: Виконується матчинг івентів для ліги.

Матчинг івентів включає перевірку наявності івентів для цієї ліги та їхнє додавання у випадку необхідності.

Завершення алгоритму.

False: Виконується матчинг івентів, а також матчинг ліг на основі івентів.

Завершення алгоритму.

3.4.5. Блок-схема для матчингу ліг по івентам

Блок-схема матчингу ліг по зматченим івентам описує процес визначення відповідності ліги на основі івентів.

Блок-схему матчингу ліг по зматченим івентам представлено на рисунку 3.12.

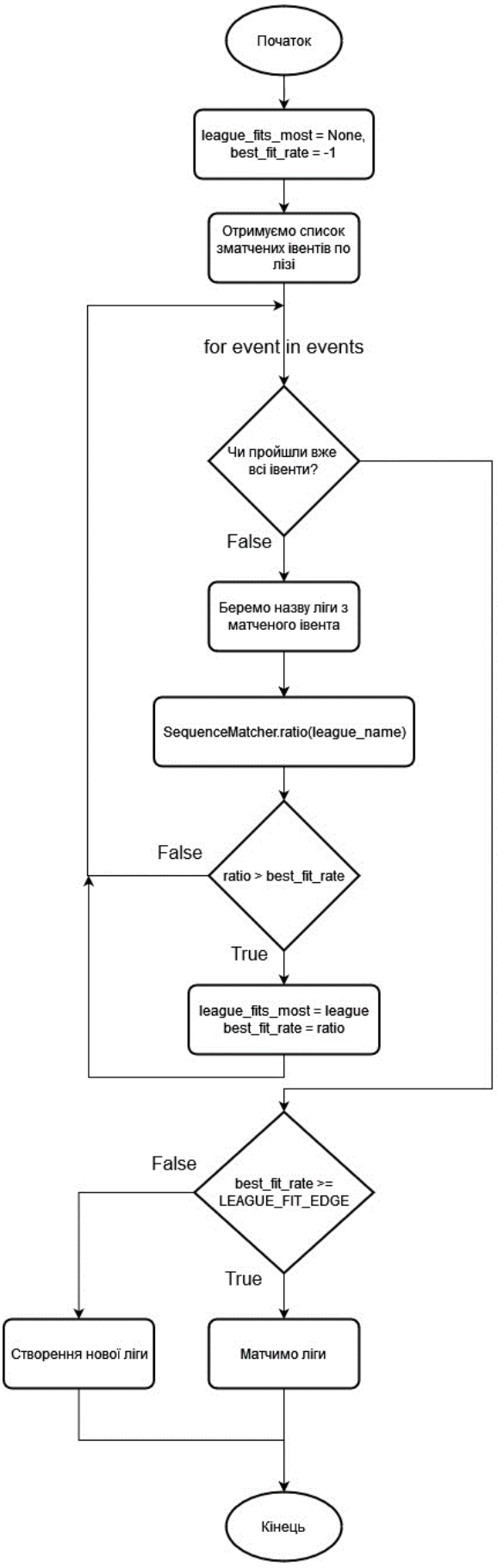


Рис. 3.12. Блок-схема матчингу ліг по зматченим івентам

1. Початок:

Запуск алгоритму матчингу ліг по зматченим івентам.

2. Ініціалізація змінних:

league_fits_most = None

best_fit_rate = -1

4. Отримання списку зматчених івентів по лізі:

Звернення до бази даних для отримання інформації про зматчені івенти в рамках ліг.

5. Цикл по івентах:

Чи пройшли вже всі івенти?

False:

Беремо назву ліги з матченого івента.

Застосовуємо порівняння назв ліг з допомогою SequenceMatcher.ratio().

Якщо ratio > best_fit_rate:

league_fits_most = league

best_fit_rate = ratio

Наступна ітерація циклу.

True:

Чи найкращий кандидат має відповідний рівень відповідності?

True: Матчимо ліги.

False: Створення нової ліги (запис в БД).

Висновок до розділу «Розробка програмного інструментарію для системи букмекерська платформа»

Розділ, присвячений розробці програмного інструментарію для системи букмекерської платформи, досліджує архітектуру та взаємодію ключових компонентів системи. Впровадження діаграм компонентів та класів надає уявлення про структуру та функціональність програмного забезпечення.

Діаграми компонентів надають важливий інструмент для візуалізації внутрішньої логіки системи. Вони визначають ключові елементи, такі як

скрапери, Redis, агрегатор, MySQL, back-end та front-end, і демонструють взаємозв'язки між ними.

На діаграмах послідовностей конкретизовано взаємодію скраперів із зовнішніми букмекерськими платформами, передачу даних до Redis, агрегацію даних агрегатором, а також взаємодію між back-end та front-end на рівні обробки ставок користувачів.

Діаграми класів глибоко розкривають внутрішню структуру агрегатора, скрапера та back-end, надаючи інформацію про класи та їх взаємодію. Це важливо для розуміння та підтримки розробленої системи.

У цьому розділі висвітлено ключові елементи, що спрямовані на ефективне функціонування системи букмекерської платформи.

4. Результати розробки

4.1. Результат розробки скраперів

Процес розробки скраперів для вилучення та обробки спортивних даних був успішно завершений. Основні досягнення та результати роботи включають:

- Ефективний Збір Даних:

Скрапери були розроблені для ефективного збору даних із різноманітних джерел, таких як букмекерські платформи та інші спортивні ресурси.

- Оптимізація Обробки Даних:

Реалізовані ефективні алгоритми обробки та аналізу отриманих даних, що дозволяють швидко та точно визначати спортивні ліги, команди та події.

- Використання Redis для Зберігання:

Для забезпечення швидкого та зручного доступу до даних, результати скрапінгу ефективно зберігаються та оновлюються у базі даних Redis.

- Модульність та Розширюваність:

Розроблені скрапери є модульними та легко розширюються, що дозволяє додавати нові джерела та покращувати функціональність без великих змін в коді.

- Інтеграція із Бекендом:

Забезпечено успішну інтеграцію скраперів із бекенд-системою, що дозволяє використовувати отримані дані у реальному часі.

Приклад даних в форматі JSON представлено на рисунку 4.1

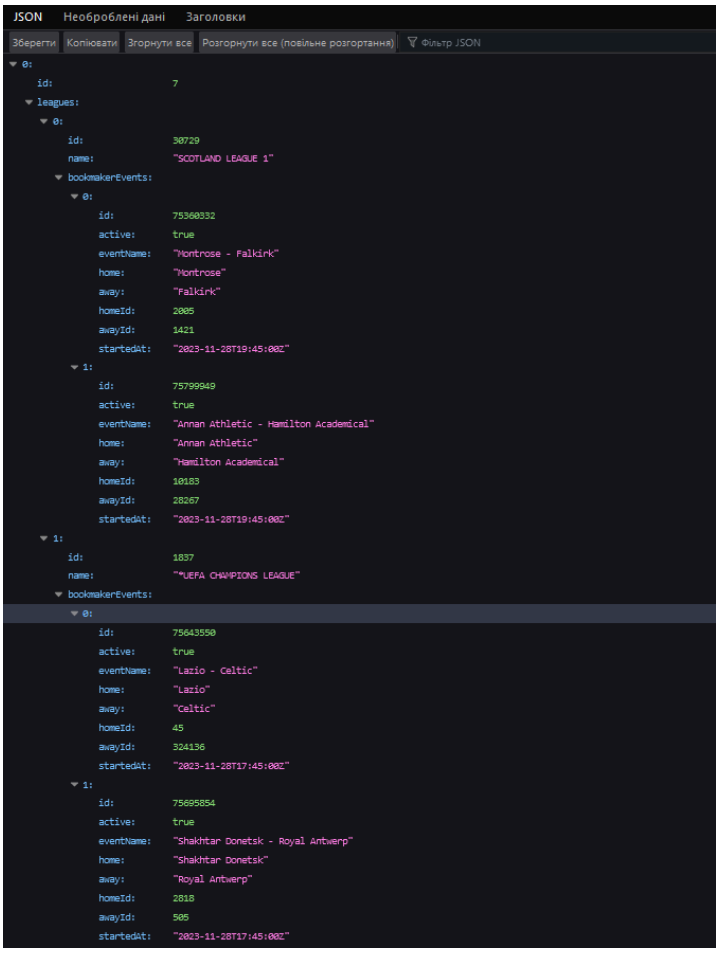


Рис. 4.1. JSON дані зі скраперу

На наведеному скріншоті можна побачити приклад JSON-структури даних, які ефективно зберігаються та оновлюються у Redis за допомогою скрапера. Ці дані відображають різні спортивні ліги та відповідні події для кожної ліги. Декілька ключових елементів цих даних:

- Ідентифікатор Ліги:

У кожному записі ліги маємо унікальний ідентифікатор, який дозволяє однозначно ідентифікувати конкретну лігу у системі.

- Назва та Ідентифікатори Подій:

Для кожної ліги містяться дані щодо назви ліги, а також перелік подій, які до неї відносяться. Кожна подія характеризується своєю назвою, командами, датою початку та ідентифікаторами команд.

- Статус та Деталі Події:

Для кожної події фіксується її статус (активна чи неактивна), назва, команди, їх ідентифікатори та час початку.

Ця структура даних дозволяє швидкий доступ до інформації про різні спортивні ліги та їхні події. Використання Redis для зберігання таких даних забезпечує швидкий доступ із можливістю динамічного оновлення, що є важливим для відстеження змін у світі спорту.

4.2. **Результат розробки агрегатору**

Під час розробки агрегатора для об'єднання та обробки спортивних даних отримано важливі результати:

- Успішна інтеграція скраперів:

Агрегатор успішно інтегрований з розробленими скраперами, забезпечуючи ефективний збір та оновлення спортивних даних.

- Ефективна обробка та матчинг даних:

Впроваджені алгоритми обробки та матчингу дозволяють агрегатору швидко та точно ідентифікувати ліги, команди та події.

- Інтерфейс адмін панелі для зручної взаємодії:

Розроблено зручний інтерфейс для взаємодії з агрегатором, що дозволяє адмінам легко аналізувати спортивні дані.

- Оптимізація Роботи З Базами Даних:

Використання Redis та MySQL сприяє ефективному зберіганню та доступу до оброблених та зматчених даних.

- Масштабованість та Модульність:

Агрегатор розроблено з урахуванням можливості масштабування та додавання нових джерел даних без значних змін у вихідному коді.

- Логування та Відстеження Помилок:

Налагоджено систему логування, що допомагає відслідковувати роботу агрегатора та швидко виявляти та усувати помилки.

- Інтеграція із Функціоналом Бекенду:

Успішно здійснено інтеграцію агрегатора із функціоналом бекенду, що дозволяє використовувати об'єднані дані в реальному часі.

4.2.1. Адміністративна панель агрегатора

Адміністративна панель агрегатора розроблена з метою полегшення та контролю за процесом матчингу та обробки спортивних даних. Основні можливості адмін панелі включають:

- Результати Матчингу:

Адміни можуть переглядати результати автоматичного матчингу для команд, ліг та івентів. Це дозволяє швидко перевірити, як алгоритми взаємодіють із зібраними даними.

- Підтвердження матчингу:

Забезпечена можливість ручного підтвердження матчингу, де адмін може вручну вказати відповідності між об'єктами даних. Це служить як додатковий рівень валідації та дозволяє уникнути можливих помилок.

- Матчинг об'єктів:

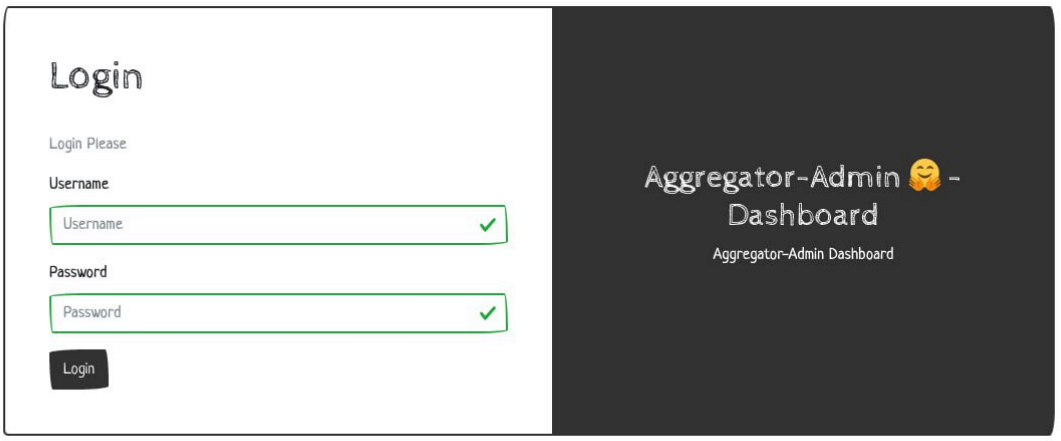
Додаткова функція дозволяє адмінам вручну зматчувати ліги та команди, що не були зматчені автоматично. Це особливо корисно для нових або незвичайних джерел даних.

- Логування та Аудит:

Система логування забезпечує адміністраторам можливість відстеження всіх змін та дій у панелі адміністратора для подальшого аудиту та виявлення можливих проблем.

Адміністративна панель агрегатора є потужним інструментом для забезпечення якості та точності обробки спортивних даних, а також для оперативного виправлення можливих невідповідностей чи помилок у процесі матчингу.

Огляд головної сторінки адміністративної панелі, де здійснюється авторизація користувача показано на рисунку 4.2



Login – Aggregator-Admin Dashboard

Рис. 4.2. Головна сторінка адміністративної панелі

Сторінка після авторизації показана на рисунку 4.3.

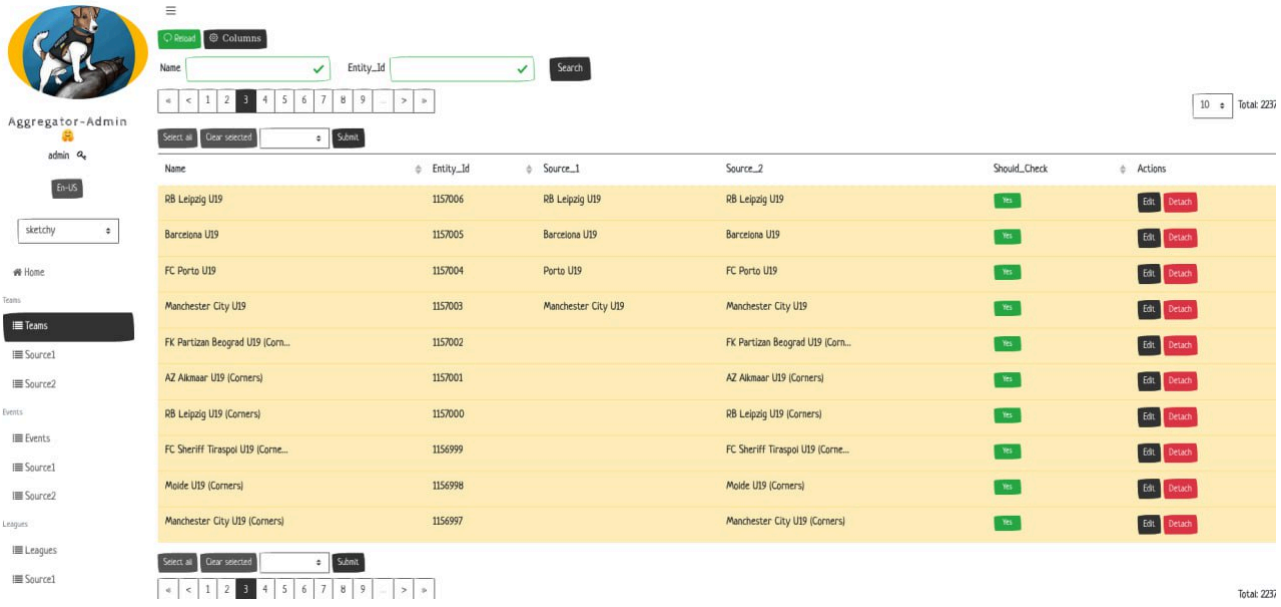


Рис. 4.3. Сторінка Events адміністративної панелі після авторизації

Навігаційне меню, розташоване ліворуч, виступає ключовим елементом цієї панелі, забезпечуючи швидкий доступ до основних функціональних частин системи.

Зматчені Команди (Teams):

У цьому розділі адміни можуть переглядати та управляти зматченими командами. Це важливий функціонал для моніторингу та контролю точності матчингу команд.

Не Зматчені Команди (Source 1 та Source 2):

Часто даних від різних джерел може бути неповним або вимагати ручного втручання. Розділи "Не Зматчені Команди від Source 1" та "Не Зматчені Команди від Source 2" дозволяють переглядати, матчити та змінювати команди, які ще не були зматчені автоматично.

Зматчені Івенти (Events):

Цей розділ дозволяє переглядати та керувати зматченими івентами. Адміністратори можуть визначити, як правильно вони були зіставлені та вживати заходи для виправлення помилок, якщо такі є.

Source 1 (Не зматчені івенти від Source 1): Тут вказані івенти від Source 1, які ще не зматчені та потребують обробки.

Source 2 (Не зматчені івенти від Source 2): Схожий розділ для івентів від Source 2, які також потребують уваги.

Leagues (Зматчені ліги): Розділ для перегляду та управління зматченими лігами.

Source 1 (Не зматчені ліги від Source 1): Цей пункт містить ліги від Source 1, які не були зматчені.

Source 2 (Не зматчені ліги від Source 2): Аналогічно для ліг від Source 2.

Функціональність

Пошук по назві та ID: Зручний інструмент для швидкого знаходження необхідних елементів.

Номери сторінок та фільтр по кількості елементів: Для ефективного управління великим обсягом інформації.

Зміна стилю front-end: Можливість персоналізації інтерфейсу за допомогою зміни стилів front-end зі списку доступних опцій.

На сторінці "Teams" адміністративної панелі розташована таблиця із зматченими командами, яка має наступні колонки:

Name (Назва): Відображає назву зматченої команди.

Entity_id (ID Команди): Показує унікальний ідентифікатор зматченої команди.

Source_1: Назва команди на ресурсі №1.

Source_2: Назва команди на ресурсі №2.

Should_check (Підтвердження): Логічне значення (boolean), що визначає, чи потрібне ручне підтвердження матчингу. Після автоматичного матчингу, якщо встановлено, адміністратор повинен ручно підтвердити матчинг на цій сторінці.

Actions (Дії): Колонка, що містить кнопки "Edit" та "Detach" для кожного зматченого запису.

Edit (Редагувати): Надає можливість змінити назву зматченої команди.

Detach (Розматчити): Дозволяє відмінити матчинг для відповідного ресурсу.

Сторінка не зматчених команд по ресурсу №1 показано на рисунку 4.4

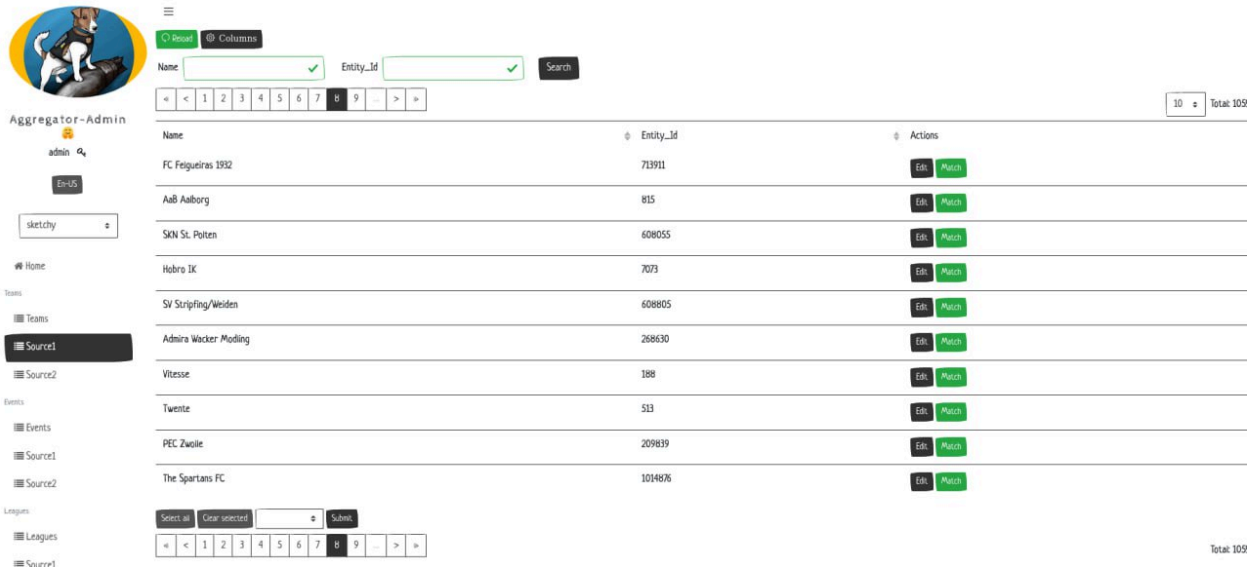


Рис 4.4. Сторінка не зматчених команд по ресурсу №1

На сторінці "Teams" для ресурсу "Source 1" розташована таблиця із зматченими командами, яка включає наступні колонки:

Name (Назва): Відображає назву зматченої команди.

Entity_id (ID Команди): Показує унікальний ідентифікатор зматченої команди.

Actions (Дії): Колонка, що містить дві кнопки для кожного запису:

Edit (Редагувати): Надає можливість змінити назву зматченої команди.

Match (Матчинг): Кнопка, яка запускає процес повторного матчингу для даної команди на ресурсі "Source 1".

Ця сторінка надає адміністратору специфічні інструменти для управління зматченими командами з ресурсу "Source 1". Кнопки "Edit" дозволяють внести зміни в існуючі дані, тоді як кнопка "Match" дозволяє знову провести матчинг для даної команди на ресурсі "Source 1". Це забезпечує більшу гнучкість у вирішенні можливих проблем чи оновлення даних на ресурсі.

Сторінка не зматчених команд по ресурсу №2 показана на рисунку 4.5

Рис. 4.5. Сторінка не зматчених команд по ресурсу №2

На сторінці "Не Зматчених Команд" агрегатора, адміністраторам доступна корисна функція - кнопка "Match". Ця кнопка відкриває сторінку матчингу конкретної команди. Цей інтерфейс розроблений для полегшення процесу ручного матчингу та забезпечення високого рівня точності у випадках, коли автоматичний матчинг не відбувся або потребує ручних корекцій.

На сторінці матчингу команд реалізований важливий функціонал:

Вивід кандидатів на матчинг:

Всі потенційні кандидати на матчинг команди відображаються на сторінці, надаючи адміністратору повний огляд доступних опцій.

Рейтинг співпадіння:

Кожен кандидат отримує рейтинг співпадіння, який обчислюється на серверному боці за допомогою SequenceMatcher. Цей рейтинг визначає, наскільки добре поточна команда відповідає кожному кандидату.

Сортування за рейтингом:

Список кандидатів відсортований за рейтингом співпадіння, щоб адміністратор міг швидко переглядати та пріоритизувати найбільш ймовірні варіанти.

Сторінка після натискання на кнопку "Match" показана на рисунку 4.6.

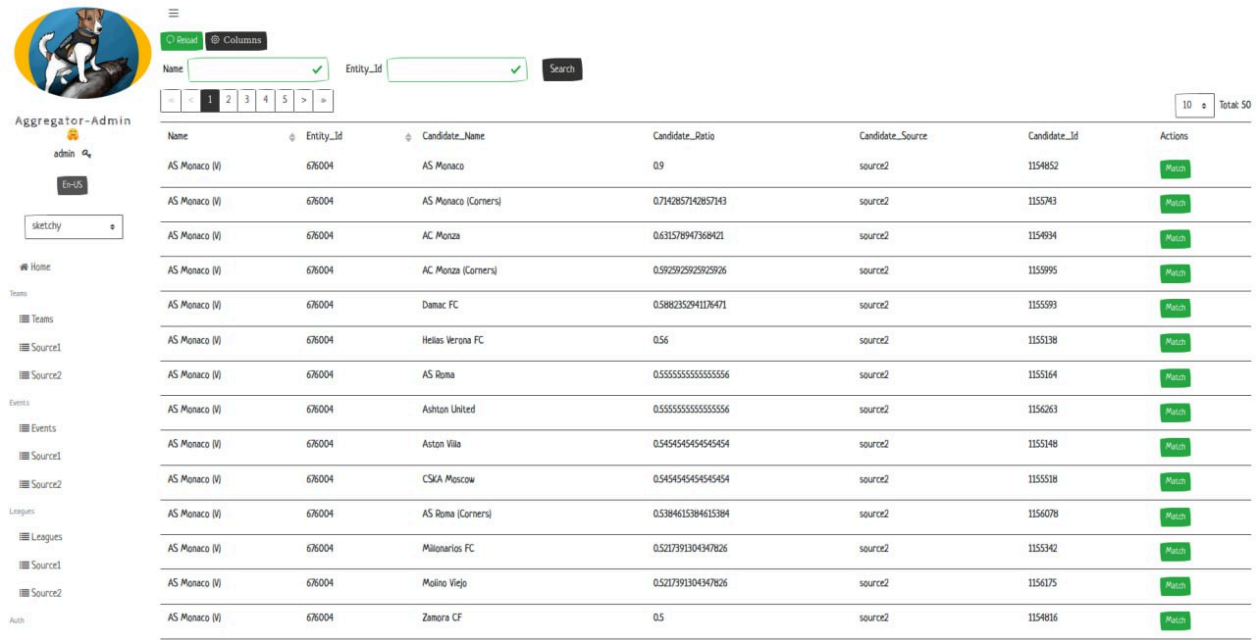


Рис. 4.6. Сторінка ручного матчингу команди

На сторінці матчингу конкретної команди реалізована можливість порівняння та вибору кандидата для матчингу. Сторінка включає наступні поля та елементи:

Name (Назва команди): Відображає назву команди, яку ви бажаєте зматчити.

Entity_id (ID Команди): Показує унікальний ідентифікатор команди, яку ви хочете зматчити.

Candidate_Name (Назва Кандидата): Відображає назву потенційної кандидатної команди.

Candidate_ratio (Оцінка Схожості): Представляє оцінку наскільки кандидат підходить для матчингу, в діапазоні від 0 до 1.

Candidate_Source (Джерело Кандидата): Вказує, з якого ресурсу (наприклад, "Source 1" чи "Source 2") отримана кандидатна команда.

Candidate_id (ID Кандидата): Показує унікальний ідентифікатор кандидата.

Actions (Дії): Колонка, що містить єдину кнопку "Match".

Match (Матчинг): Кнопка, яка ініціює процес матчингу між вибраною командою та кандидатом.

Сторінка ручного матчингу іншої команди показана на рисунку 4.7

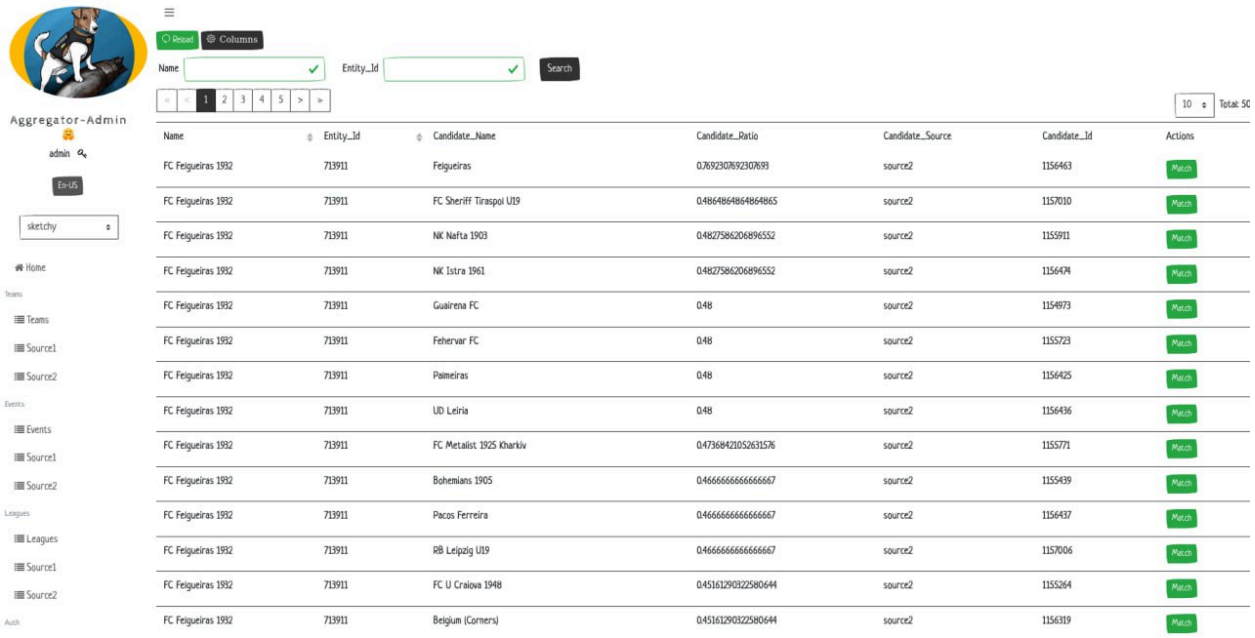


Рис. 4.7. Сторінка ручного матчингу іншої команди

У адміністративній панелі агрегатора надана унікальна можливість персоналізації вигляду інтерфейсу. Адміністраторам доступний функціонал, що дозволяє змінювати стилі front-end сторінок з використанням різноманітних опцій.

Ця можливість має кілька ключових особливостей:

Вибір стилю:

Адміністратор може обрати бажаний стиль в означеному списку доступних опцій. Це дозволяє адаптувати інтерфейс до особистих вподобань чи корпоративного стилю.

Застосування змін:

Після вибору стилю, зміни негайно застосовуються до відображення всіх сторінок адміністративної панелі. Це забезпечує миттєвий ефект та дозволяє адміністраторам оцінити вигляд інтерфейсу в реальному часі.

Різноманіття виглядів:

Оскільки доступний не лише один, а кілька стилів, адміністратори можуть експериментувати та обирати той, який вони вважають найбільш прийнятним для конкретного використання чи настрою.

Ця функція призначена для того, щоб адміністратори відчували себе комфортно та мали можливість адаптувати інтерфейс до своїх власних уподобань, що є важливим аспектом комфортної та продуктивної роботи з адміністративною панеллю.

Змінена новим стилем сторінка ручного матчингу команди показана на рисунку 4.8

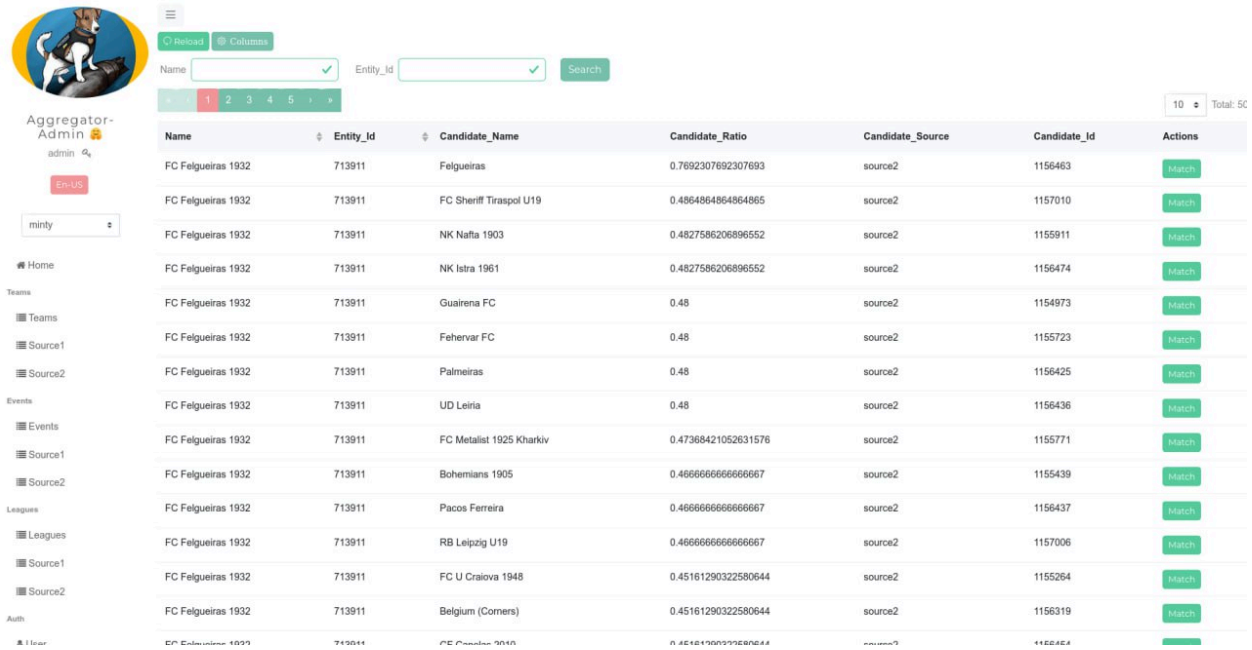


Рис. 4.8. Змінена сторінка з новим стилем

На сторінці зматчених команд та ліг адміністраторам надана можливість миттєво редагувати інформацію, а саме назви, щоб усунути будь-які неполадки чи доповнити дані. Для кожного об'єкта є окрема кнопка "Edit", яка відкриває інтерфейс редагування.

Ця функція має декілька ключових особливостей:

Індивідуальність для Кожного Об'єкта:

Кожній зматченій команді та лізі відповідає окрема кнопка "Edit". Це забезпечує індивідуальний підхід та спрощує знаходження конкретного об'єкта для редагування.

Зручний Інтерфейс Редагування:

Після натискання кнопки "Edit" відкривається інтерфейс редагування, де можна легко та швидко внести необхідні зміни. Це може включати зміну назви команди чи ліги.

Миттєве Оновлення:

Зміни, внесені через інтерфейс редагування, негайно застосовуються до відповідного об'єкта. Це дозволяє адміністраторам бачити результати своєї роботи в режимі реального часу.

Усунення Помилок та Доповнення Даних:

Адміністратори можуть вносити зміни, коли це необхідно, щоб усунути будь-які помилки у зматчених даних чи доповнити їх для отримання більш повної інформації.

Ця функція призначена для забезпечення максимальної гнучкості та швидкості в адмініструванні зматчених команд та ліг, надаючи зручний та швидкий спосіб редагування важливих даних.

Сторінка "Edit" зматченої команди показана на рисунку 4.9

Рис. 4.9. Сторінка "Edit" зматченої команди

Сторінка "Edit" не зматченої команди показана на рисунку 4.10

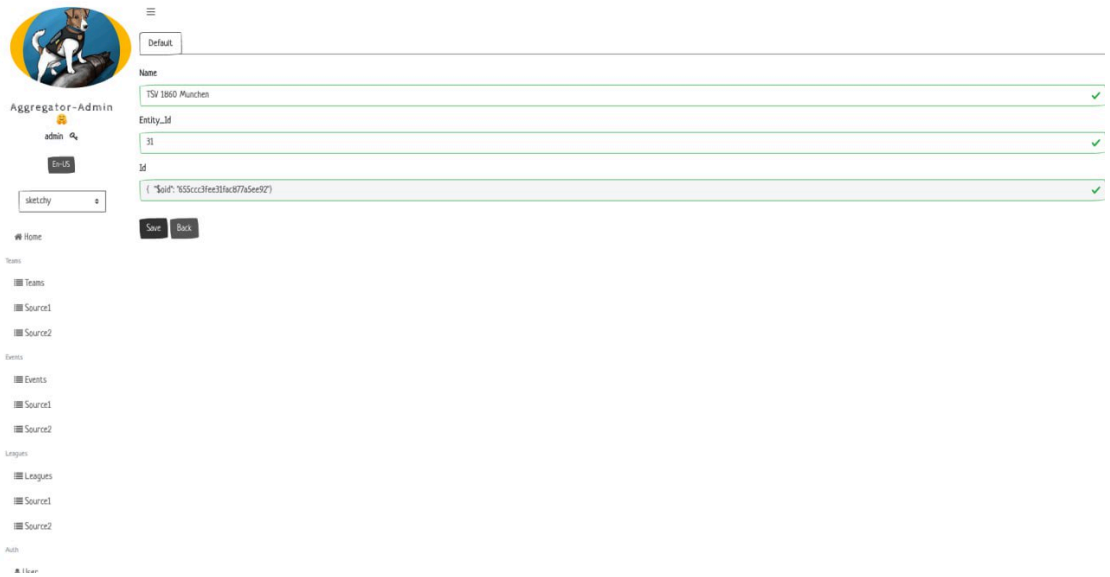


Рис. 4.10. Сторінка "Edit" не зматченої команди

Сторінка зматчених івентів показано на рисунку 4.11

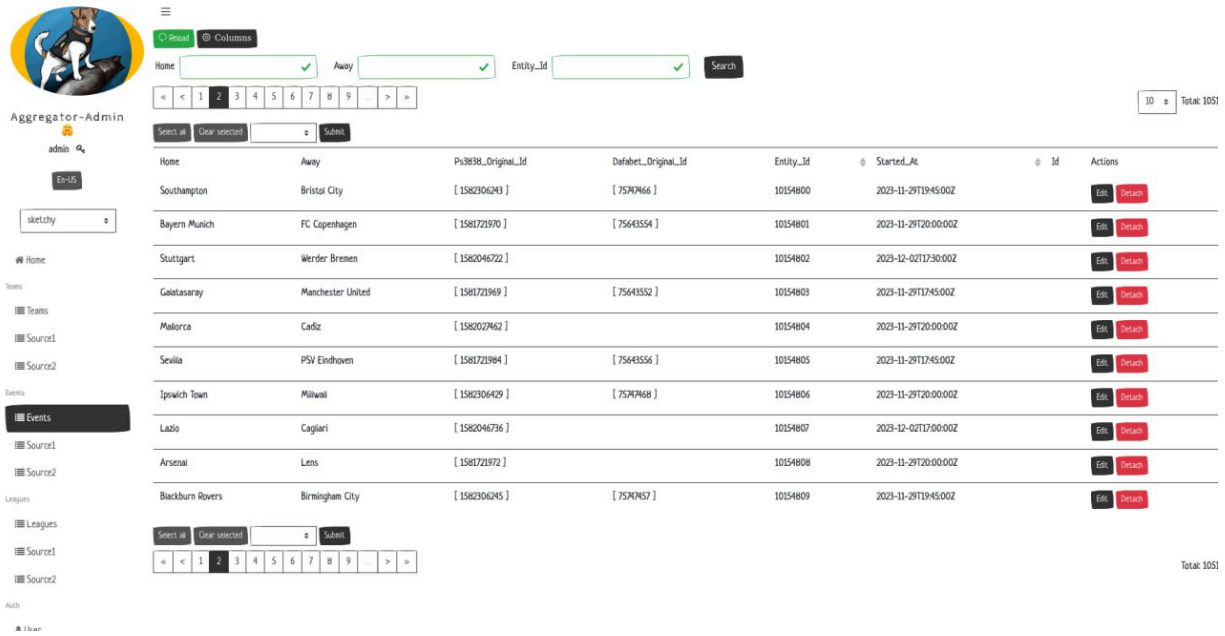


Рис. 4.11. Сторінка зматчених івентів

Сторінка Events (зматчені івенти).

Home (Назва Команди): Відображає назву команди "Home" в івенті.

Away (Назва Команди): Відображає назву команди "Away" в івенті.

Source1_Id (ID Івента на Ресурсі 1): Показує унікальний ідентифікатор івента на ресурсі 1.

Source2_Id (ID Івента на Ресурсі 2): Показує унікальний ідентифікатор івента на ресурсі 2.

Entity_id (ID Івента): Показує унікальний ідентифікатор івента.

Started_at (Час початку): Вказує час початку івента.

Actions (Дії): Колонка, що містить дві кнопки для кожного запису:

Edit (Редагувати): Надає можливість змінити дані івента.

Detach (Розматчити): Кнопка, яка дозволяє відмінити матчинг для обраного івента.

Сторінка не зматчених івентів показана на рисунку 4.12

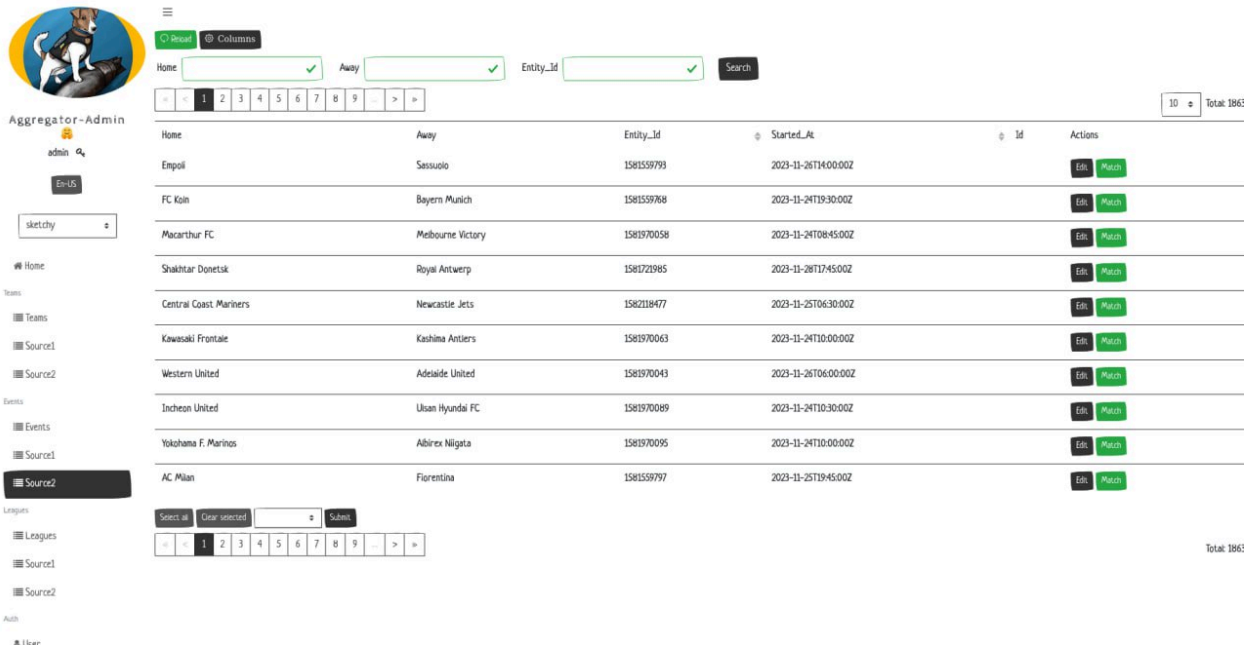


Рис. 4.12. Сторінка не зматчених івентів

Сторінка "Events" для ресурсу "Source 2"

На сторінці "Events" для ресурсу "Source 2" ви можете керувати івентами та проводити їхній матчинг. Сторінка включає наступні поля та елементи:

Home (Назва Команди): Відображає назву команди, що представляється як "Home" в івенті.

Away (Назва Команди): Відображає назву команди, що представляється як "Away" в івенті.

Entity_id (ID Івента): Показує унікальний ідентифікатор івента.

Started_at (Час початку): Вказує час початку івента, визначений у форматі UTC.

Actions (Дії): Колонка, що містить дві кнопки для кожного запису:

Edit (Редагувати): Надає можливість змінити дані івента.

Match (Матчинг): Кнопка, яка дозволяє розпочати процес матчингу для обраного івента з ресурсу "Source 2".

Сторінка після виконання пошуку по назві показана на рисунку 4.13

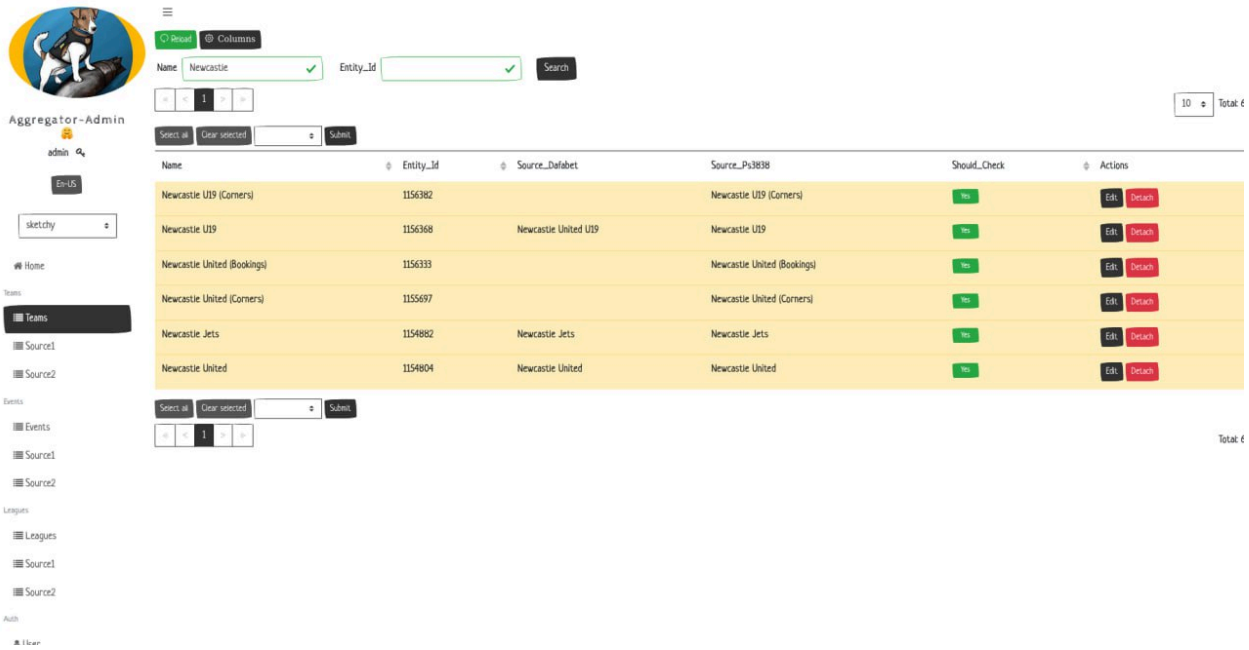


Рис. 4.13. Сторінка після виконання пошуку по назві

На сторінках зматчених об'єктів, таких як ліги, івенти та команди, система використовує візуальне підкреслення для позначення об'єктів, які потребують уваги адміністратора. Об'єкти, які автоматично зматчено, виділені жовтим кольором, сигналізуючи, що потрібно перевірити та підтвердити їхній матчинг.

Процес Підтвердження:

Відзначення Об'єктів:

Адміністратор може переглядати сторінку та визначати, які об'єкти потребують перевірки. Об'єкти, які очікують підтвердження, відзначаються жовтим кольором.

Простий Вибір Об'єктів:

Адміністратор може вибрати об'єкти для підтвердження, просто натисканням на відповідний рядок об'єкта. Це забезпечує легкий та інтуїтивно зрозумілий вибір.

Масова Перевірка:

Після вибору всіх об'єктів, які потребують підтвердження, адміністратор може використовувати опцію "Check in All", щоб швидко визначити всі обрані об'єкти.

Підтвердження та Застосування:

Після вибору об'єктів та використання опції "Check in All", адміністратор натискає кнопку "Submit", після чого система підтверджує обрані об'єкти та застосовує їхній статус.

Ефективне Управління:

Цей процес дозволяє адміністратору ефективно управляти підтвердженням матчингу об'єктів, роблячи процес максимально простим та швидким.

Цей інтуїтивно зрозумілий інструмент дозволяє адміністраторам легко та ефективно підтверджувати матчинг об'єктів, зменшуючи ймовірність помилок та оптимізуючи робочий процес.

Сторінка з демонстрацією виділення об’єктів для підтвердження матчингу показано на рисунках 4.14, 4.15

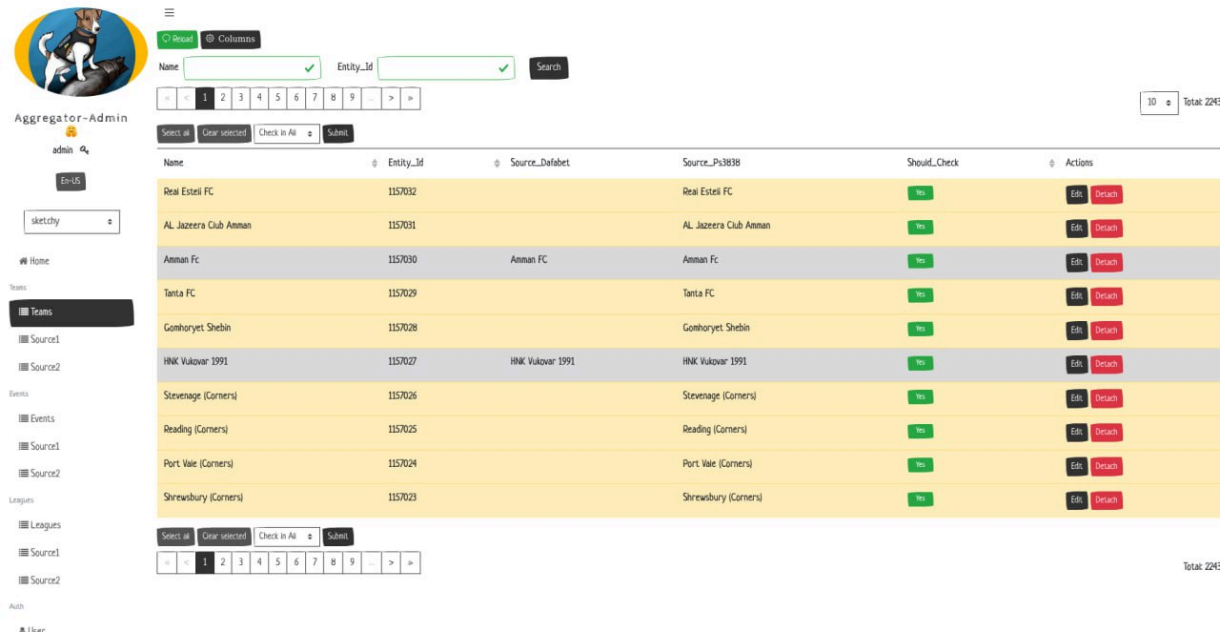


Рис. 4.14. Сторінка з демонстрацією виділення об’єктів для підтвердження матчингу

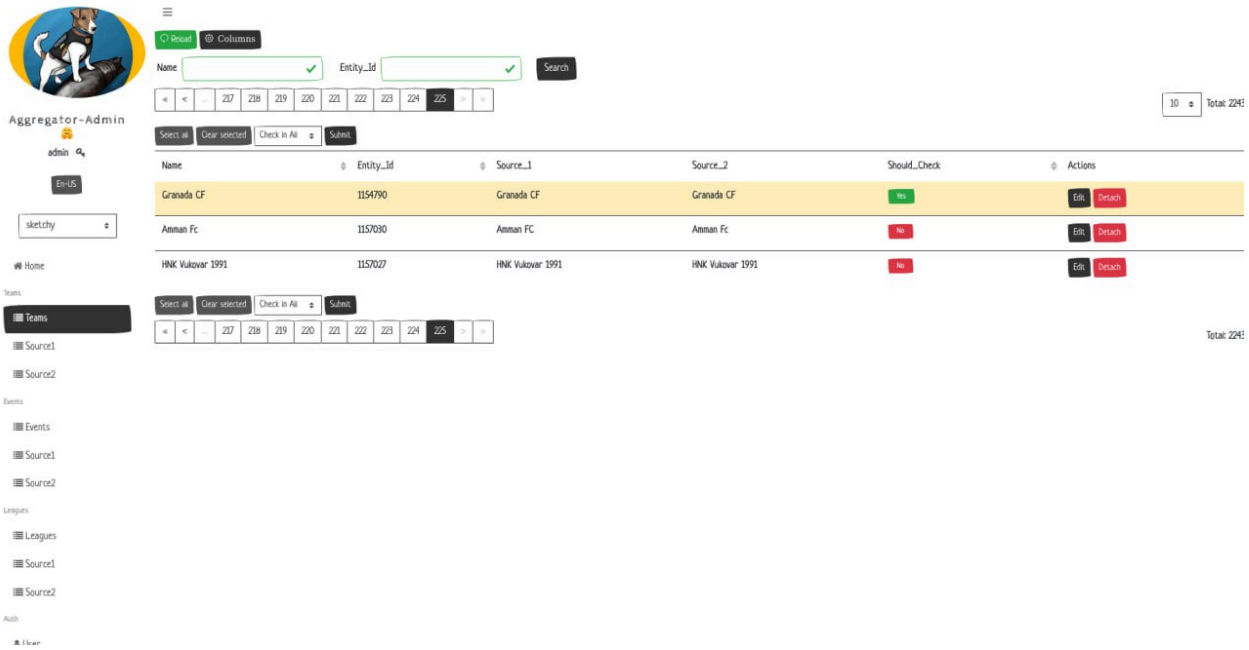


Рис. 4.15. Сторінка з підтвердженими командами

На сторінці зматчених об'єктів (команд, ліг, івентів) адміністраторам доступна кнопка "Detach" для кожного окремого об'єкта. Ця функціональність призначена для розматчування конкретного ресурсу та забезпечення гнучкості управління даними.

Ключові аспекти кнопки "Detach":

Розматчування зручне та індивідуальне:

Кожному зматченому об'єкту надано окрему кнопку "Detach". Це дозволяє адміністраторам розматчувати лише ті ресурси, які їм потрібно, зберігаючи інші незмінними.

Створення Сторінки Розматчування:

Після натискання на кнопку "Detach" відкривається сторінка розматчування, де адміністратор може визначити, які частини ресурсу слід розматчувати.

Можливість Розматчування Команд та Їхніх Івентів:

Розматчування команд також автоматично розматчує відповідні івенти, що містять ці команди. Це забезпечує логічну та послідовну роботу інструменту розматчування.

Можливість Визначення Обсягу Розматчування:

Адміністратор може визначити, наскільки широко або вузько слід розматчувати конкретний ресурс. Це робить процес розматчування дієвим та спрощеним.

Кнопка "Detach" створена для того, щоб надати адміністраторам повний контроль над розматчуванням, забезпечуючи простий та зрозумілий інструмент для цієї задачі.

Сторінка "Detach" показана на рисунку 4.16

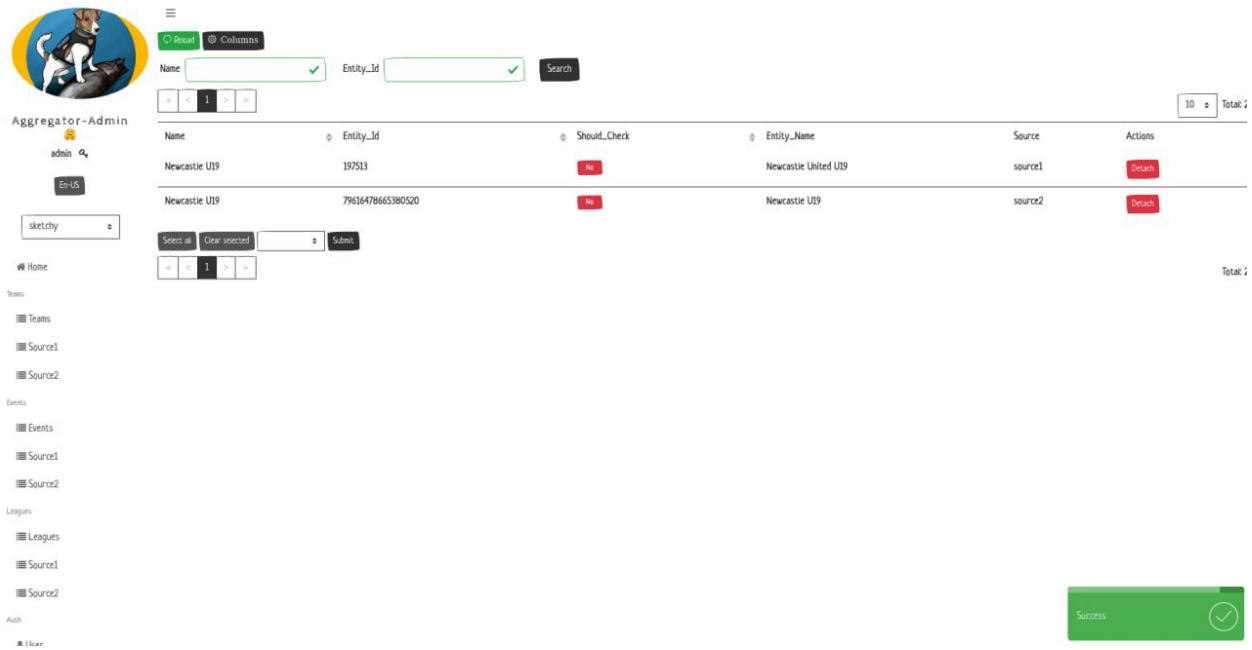


Рис. 4.16. Сторінка "Detach"

4.3. **Результат розробки сайту букмекерської платформи**

Головна сторінка букмекерської платформи. Ця сторінка становить центральну точку доступу до подій та івентів, визначаючи зручний інтерфейс для навігації та перегляду подій.

Основні Характеристики Головної Сторінки:

Перелік Ліг та Спортивних Подій:

По центру головної сторінки розташований перелік ліг, які визначають область спортивних подій. Це створює структурований та легко читабельний перелік доступних спортивних подій.

Список Івентів:

Кожна ліга супроводжується списком івентів, що включають дату та час їхнього початку. Це дозволяє користувачам оперативно вибирати події, які їх цікавлять.

Коефіцієнти на Маркеті 1X2:

Для кожного івента на головній сторінці вказані коефіцієнти на маркеті 1X2, що робить інформацію про ставки легко доступною та зрозумілою.

Навігаційний Список Спортів та Ліг:

Зліва від головного списку розташований навігаційний список спортів та ліг, дозволяючи користувачам швидко фільтрувати події за їхнім типом.

Сортування за Датою:

Зверху розміщені сортувальні кнопки "All", "1 Week", "Today", "Tomorrow", що дозволяють користувачам легко обирати область подій за певним періодом.

Головна сторінка букмекерської платформи показана на рисунку 4.17

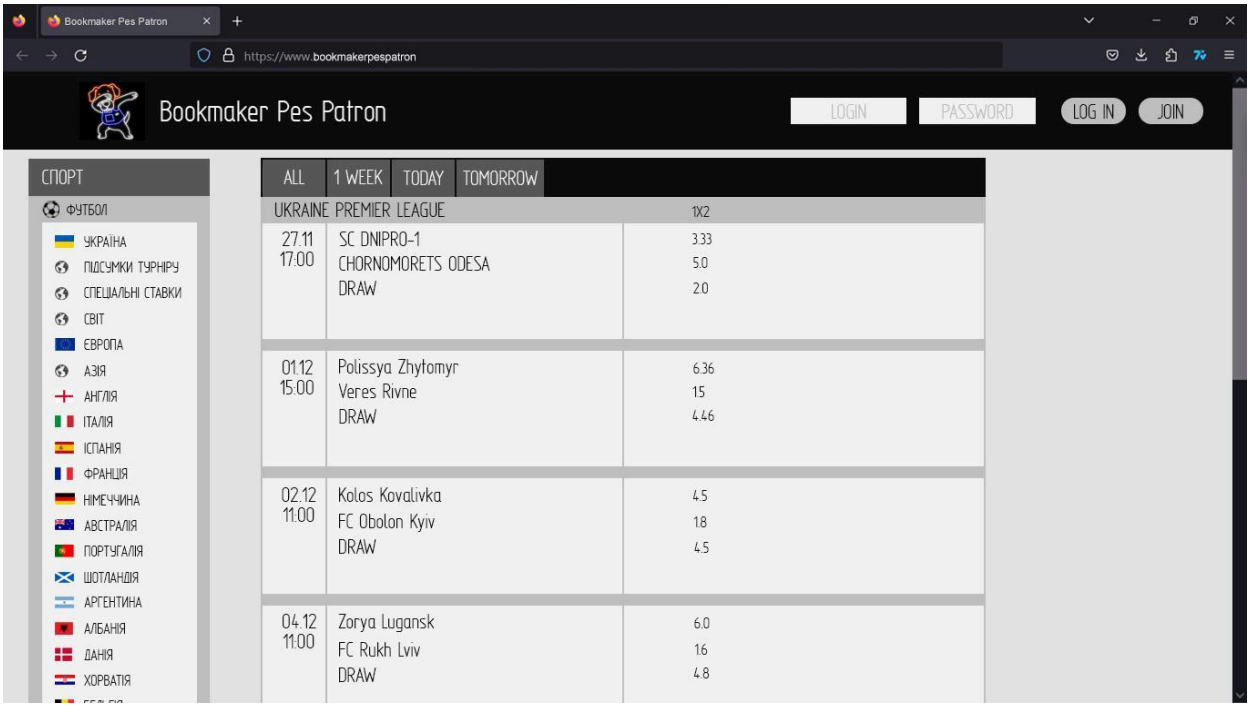


Рис. 4.17. Головна сторінка букмекерської платформи

Після натискання на конкретний івент з головної сторінки, користувач переноситься на сторінку івенту. Ця сторінка надає більш детальну та конкретну інформацію про спортивну подію, зокрема:

Інформація про Гру:

На сторінці івенту вказано деталі щодо того, хто грає проти кого і коли запланований початок гри.

Коефіцієнти на Маркеті 1X2:

Подробиці щодо коефіцієнтів на маркеті 1X2 представлені на сторінці івенту. Користувач може швидко переглядати і аналізувати коефіцієнти на перемогу команди 1, нічию чи перемогу команди 2.

Ставки Користувачів:

Додатково, для кожного варіанту (1, X, 2) користувач може переглядати загальну суму ставок, які були зроблені на цей варіант. Це надає додатковий рівень інформації та допомагає користувачам приймати обґрунтовані рішення.

Сторінка конкретного івенту показана на рисунку 4.18

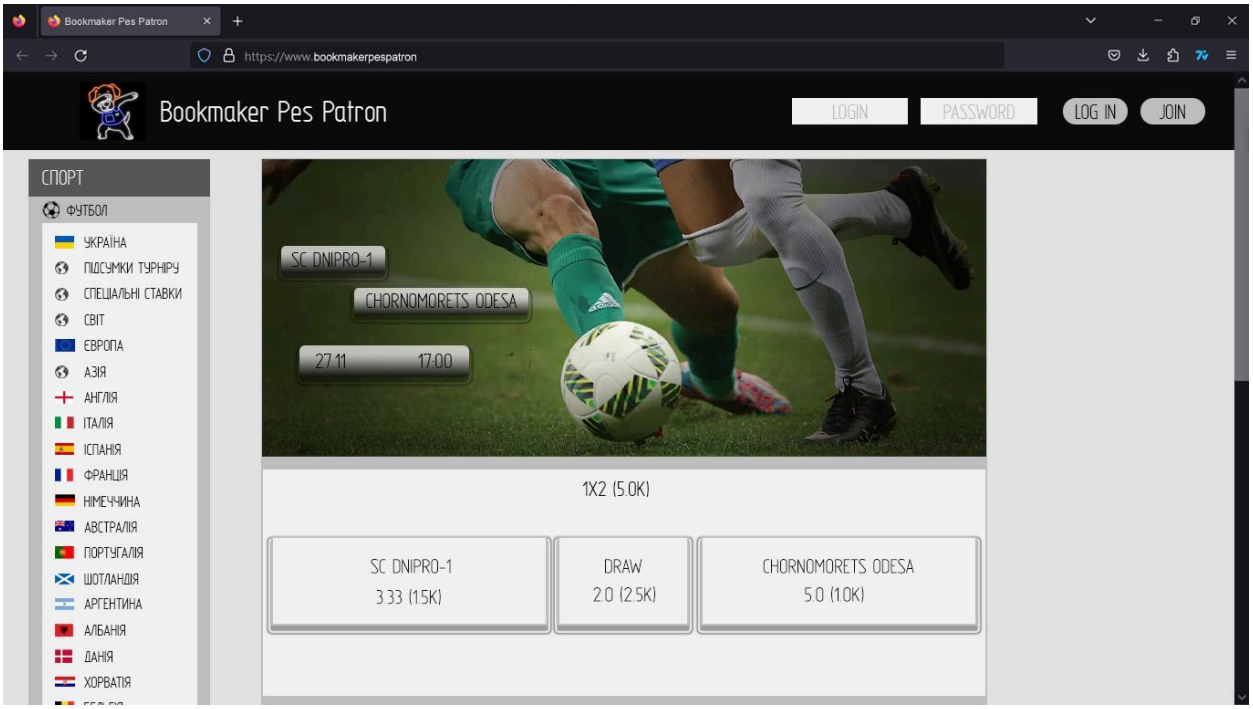


Рис. 4.18. Сторінка конкретного івенту

При розміщенні ставок на сторінці івенту, система забезпечує динамічні зміни у коефіцієнтах, що робить процес більш захоплюючим та інтерактивним. На цій сторінці важливою особливістю є:

Автоматичне Оновлення Коефіцієнтів:

При кожній новій ставці коефіцієнти автоматично оновлюються в реальному часі, враховуючи величину та напрямок ставки. Це дозволяє користувачам отримувати найсвіжішу інформацію про ринкові умови.

Візуальне Позначення Змін:

Зміни коефіцієнтів позначаються візуально стрілками, які вказують напрямок зміни. Наприклад, підняття коефіцієнту може бути відображено стрілкою вгору, а зниження - стрілкою вниз. Це важливо для того, щоб гравці могли швидко визначити, як змінилися шанси на подію.

Динамічність та Зручність:

Ця динамічність забезпечує гравцям зручність при виборі своїх ставок. Вони можуть швидко реагувати на зміни у коефіцієнтах та вчасно приймати рішення.

Сторінка івенту зі зміною коефіцієнтів показано на рисунку 4.19

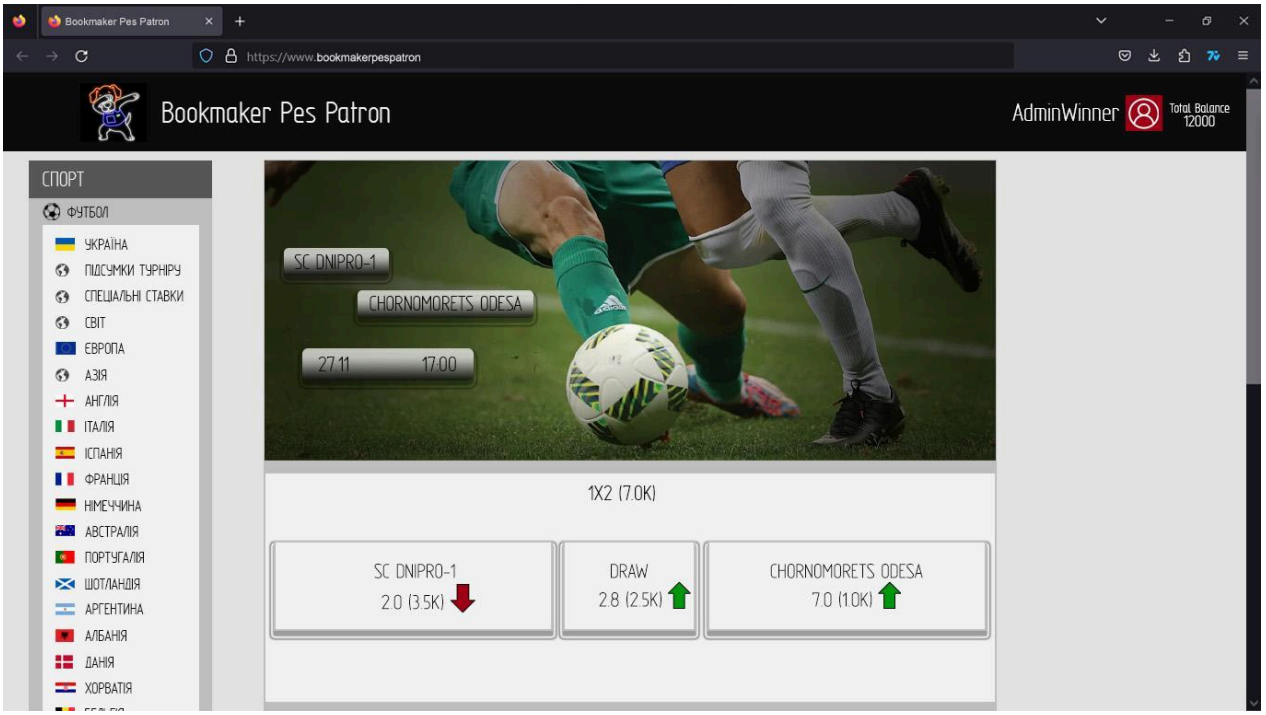


Рис. 4.19. Сторінка івенту зі зміною коефіцієнтів

Висновок до розділу «Результати розробки»

Результати розробки включають в себе успішну імплементацію скраперів для збору різноманітних даних з ресурсів, що надає можливість

отримати актуальні дані про ліги, івенти та команди. Ці скрапери виявилися ефективними та надійними джерелами інформації.

Паралельно успішно реалізовано розробку агрегатору, який об'єднує отримані дані в цілісну базу даних. Агрегатор відзначається ефективною обробкою та оновленням інформації, забезпечуючи користувачів актуальними та достовірними даними для використання на букмекерській платформі.

Описані функції головної сторінки та сторінки івенту підтверджують високу зручність інтерфейсу та динамічність системи, що робить платформу привабливою та функціональною для користувачів.

У цілому, результати розробки свідчать про вдалий процес створення букмекерської платформи, яка надійно взаємодіє зі скраперами та агрегатором, забезпечуючи користувачам можливість здійснювати ставки на спортивні події з високою ефективністю та точністю інформації.

ЗАГАЛЬНІ ВисновкИ

В даній АВР проведено аналіз предметної області, визначено вимоги та очікування гравців, розглянуто методи та алгоритми для розробки букмекерської платформи, детально описано програмний інструментарій та представлено результати розробки.

Розділ «Аналіз предметної області» надав фундаментальне розуміння букмекерської індустрії, визначивши ключові аспекти та напрямки подальших досліджень. Вимоги та очікування гравців були враховані при створенні концепції букмекерської платформи, яка відповідає їх потребам.

Розділ "Методи та алгоритми для системи букмекерської платформи" висвітлив ефективні методи матчингу даних та використання сучасних технологій веб-розробки. Інтеграція цих елементів утворює повноцінну платформу з ефективним матчингом та високотехнологічною інфраструктурою.

Розділ "Розробка програмного інструментарію для системи букмекерська платформа" представив детальний огляд архітектури та взаємодії компонентів системи, враховуючи ключові елементи для її ефективного функціонування.

Результати розробки свідчать про успішну імплементацію скраперів, ефективну роботу агрегатору та високу зручність інтерфейсу. Платформа надійно взаємодіє зі скраперами та агрегатором, забезпечуючи користувачам можливість здійснювати ставки на спортивні події з високою ефективністю та точністю інформації.

У цілому, дане дослідження є важливим кроком у розробці і вдосконаленні букмекерської платформи відповідно до сучасних стандартів та вимог гравців.

Список використаних джерел

1. Що таке букмекерська платформа? [Електронний ресурс]. – Режим доступу до ресурсу: https://elit.crimea.ua/
2. Історія букмекерських контор: від перших тоталізаторів до онлайн-беттингу [Електронний ресурс] – Режим доступу до ресурсу: https://pravda.if.ua/istoriya-bukmekerskyh-kontor-vid-pershyh-totalizatoriv-do-onlajn-bettyngu/
3. Маржа в букмекерстві: особливості роботи БК [Електронний ресурс] – Режим доступу до ресурсу: https://logincasino.ua/blog/marja-v-bukmekerstvi-osoblivosti-roboti-bk65105.html
4. What is web scraping? [Електронний ресурс] – Режим доступу до ресурсу: https://www.parsehub.com/blog/what-is-web-scraping/
5. Online vs. Offline Bookmakers [Електронний ресурс] – Режим доступу до ресурсу: https://thepointernewsonline.com/online-vs-offline-bookmakers-whats-more-reliable/
6. What is Black Box Model? [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/b/blackbox.asp
7. Docker [Електронний ресурс] – Режим доступу до ресурсу: https://docs.docker.com/get-started/overview/
8. Docker: простими словами про контейнеризацію Електронний ресурс] – Режим доступу до ресурсу: https://blog.iteducenter.ua/devops/docker-prostimi-slovami-pro-kontejnerizaciyu/
9. Scott Chason and Ben Straub. ProGIT – Version 2023-11-16 – С. 14-16
10. What is Python? [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/doc/essays/blurb/
11. Advantages Of Python Over Other Languages [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/advantages-of-python-over-other-languages

12. Asynchronous Programming in Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.mend.io/free-developer-tools/blog/asynchronous-programming-in-python-understanding-the-essentials/

13. Welcome to AIOHTTP [Електронний ресурс] – Режим доступу до ресурсу: https://docs.aiohttp.org/en/stable/

14. Asynchronous Web Scraping With Python & AIOHTTP [Електронний ресурс] – Режим доступу до ресурсу: https://oxylabs.io/blog/asynchronous-web-scraping-python-aiohttp

15. What is web socket and how it is different from the HTTP? [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/

16. Web Socket: The Fastest Way To Scrape Websites [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@lowweihong/scraping-in-another-dimension-7c6890a156da

17. What are the most effective techniques for avoiding web scraping blocks and bans? [Електронний ресурс] – Режим доступу до ресурсу: https://www.linkedin.com/advice/0/what-most-effective-techniques-avoiding-web-scraping-zn0pf

18. Tips For Web Scraping Without Getting Blocked/Blacklisted [Електронний ресурс] – Режим доступу до ресурсу: https://www.scraperapi.com/blog/10-tips-for-web-scraping/

19. What Is Data Unification and Why Does It Matter? [Електронний ресурс] – Режим доступу до ресурсу: https://www.webfx.com/blog/marketing/what-is-data-unification/

20. Introduction to Redis [Електронний ресурс] – Режим доступу до ресурсу: https://redis.io/docs/about/

21. Benefits of Redis [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/redis/

22. Python Libraries [Електронний ресурс] – Режим доступу до ресурсу: https://www.codingninjas.com/studio/library/python-libraries-and-their-features

23. What is MySQL? Everything You Need to Know [Електронний ресурс] – Режим доступу до ресурсу: https://ua.talend.com/resources/what-is-mysql/

24. SequenceMatcher in Python [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/sequencematcher-in-python-6b1e6f3915fc

25. Longest Common Subsequence (LCS) [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/longest-common-subsequence-dp-4/

26. What is SequenceMatcher() in Python? [Електронний ресурс] – Режим доступу до ресурсу: https://www.educative.io/answers/what-is-sequencematcher-in-python

27. HTML, CSS, and JavaScript: The Foundation of Front-End Development [Електронний ресурс] – Режим доступу до ресурсу: https://www.linkedin.com/pulse/html-css-javascript-foundation-front-end-development-waris-ahmed

28. Responsive design More [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design

29. What is React.js? Uses, Examples, & More [Електронний ресурс] – Режим доступу до ресурсу: https://blog.hubspot.com/website/react-js

30. Why Should You Use a Router in React.js? [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/reactjs-tutorial/routing-in-reactjs

31. Що таке AJAX та як вона працює? [Електронний ресурс] – Режим доступу до ресурсу: https://dzudzylo.com/javascript/shho-take-ajax-ta-yak-vona-pratsyuye.html

32. Повне розуміння діаграми компонентів UML за допомогою легкого методу [Електронний ресурс] – Режим доступу до ресурсу: https://www.mindonmap.com/uk/blog/uml-component-diagram/

33. What is Class Diagram? [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

34. Діаграма послідовності (Sequence Diagrams) [Електронний ресурс] – Режим доступу до ресурсу: https://www.maxzosim.com/sequence-diagrams/

35. Що таке блок-схеми та як ними користуватися? [Електронний ресурс] – Режим доступу до ресурсу: https://experience.dropbox.com/uk-ua/resources/flowcharts

Додаток А – інформаційні слайди

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури
Автоматизації і інформаційних технологій
Інформаційних технологій

Атестаційна випускна робота
на здобуття освітнього ступеня«магістр» на тему:
«Букмекерська платформа без маржинальності та з динамічним скрапінгом»

Виконав
студент групи КН-м22
Павліха Р.В.
Керівник
к.т.н., доц. Гончаренко Т.А.

Київ 2023р.

Вступ

2

Атестаційна випускна робота присвячена розробці букмекерської платформи, яка об'єднує в собі дві ключові інновації- динамічний скрапінг та принцип безмаржинальності. У сучасному світі конкуренція в галузі букмекерського бізнесу висока, і важливо розробляти технологічно продумані та високоефективні платформи, щоб задовольняти зростаючі вимоги користувачів.

Аналіз предметної області

3

Букмекерська платформа – це система, що дозволяє користувачам робити ставки на різноманітні спортивні події, розважальні заходи чи інші події, на які можна робити прогнози

Маржинальність у сфері платформ контор є ключовим поняттям, яке відіграє важливу роль у функціонуванні та прибутковості цього бізнесу. Вона представляє собою відсоток, які букмекерська контора утримує з ринкових ставок, щоб гарантувати собі прибуток незалежно від конкретного результату спортивного заходу або іншої події, на яку приймаються ставки

Скрапінг є процесом автоматичного витягування даних з веб-ресурсів. Це може бути як безпосередній перегляд веб-сторінок та парсинг HTML-коду, так і використання HTTP -реквестів до backend серверу сайту, аналогічно до того, як це робить frontend.

Класифікація букмекерських платформ

4

Мета та постановка задачі

5

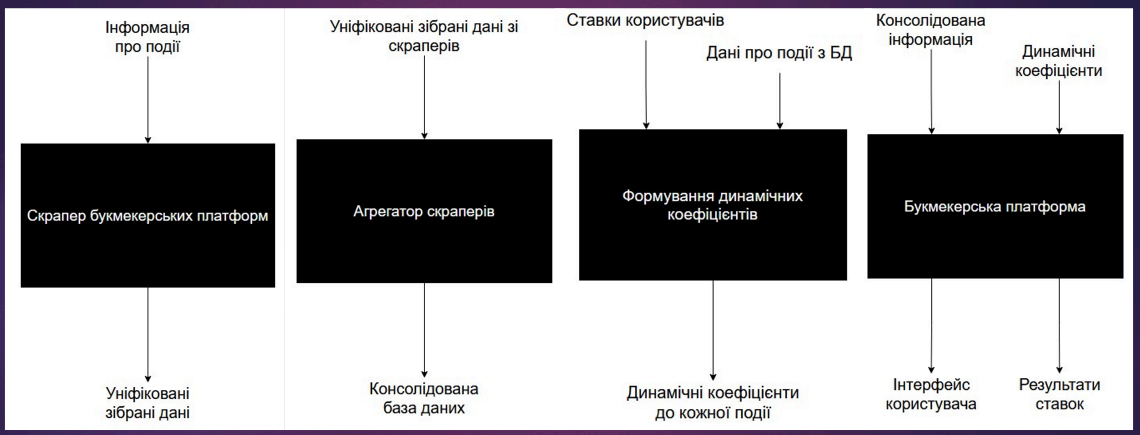
Метою розробки букмекерської платформи без маржинальності та з динамічним скрапінгом є забезпечення користувачам альтернативного та прозорого середовища для здійснення ставок на спортивні події.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Розробка скраперів
2. Створення агрегатора
3. Формування динамічних коефіцієнтів
4. Розробка букмекерської платформи

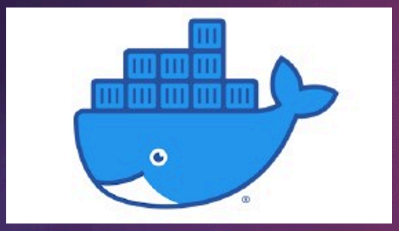
Чорний ящик

6



Загальні технології розробки

7



Технології розробки скраперів

8

Технології розробки агрегатору

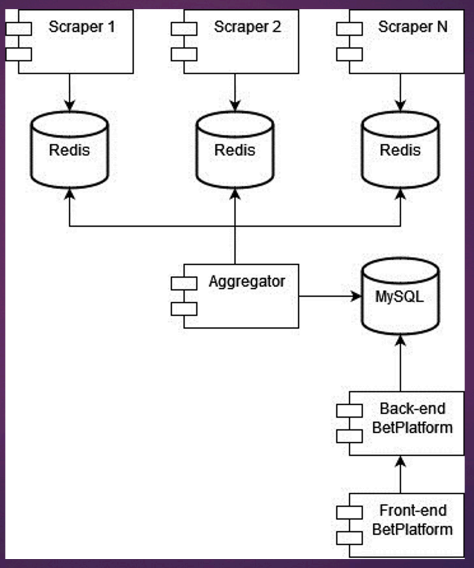
9

Технології розробки Букмекерської платформи

10

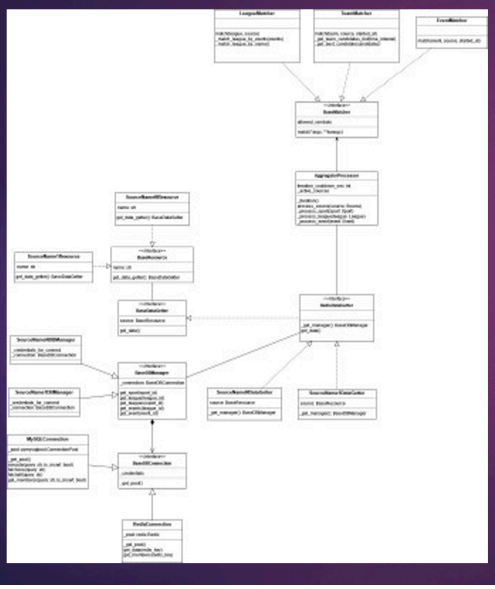
Діаграма компонентів

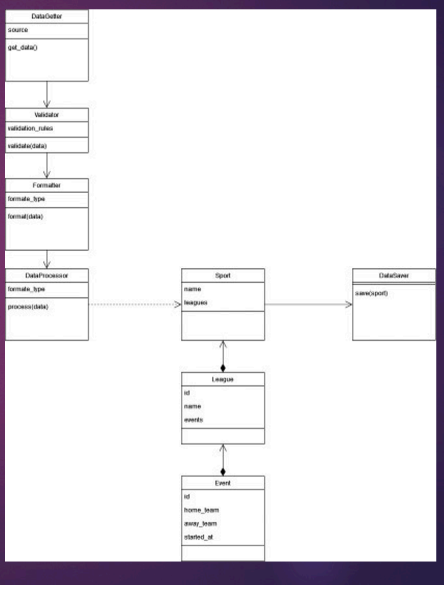
11



Діаграми класів

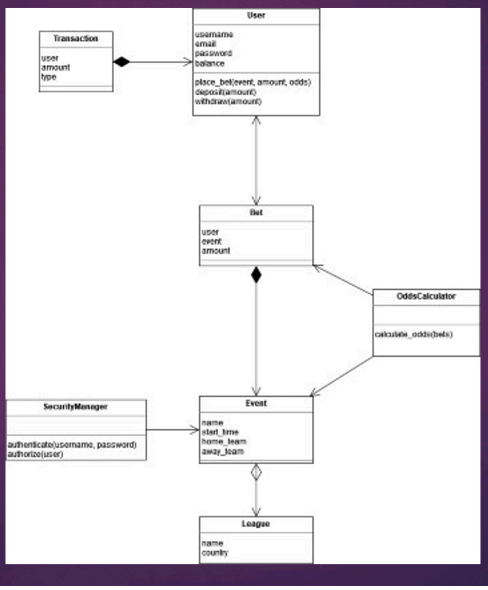
12





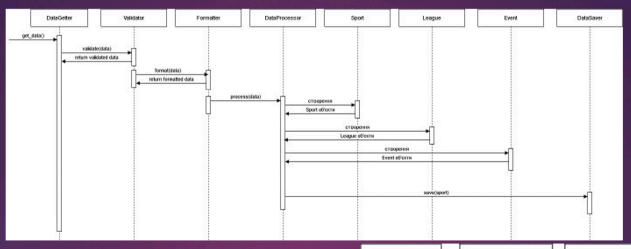
Діаграми класів

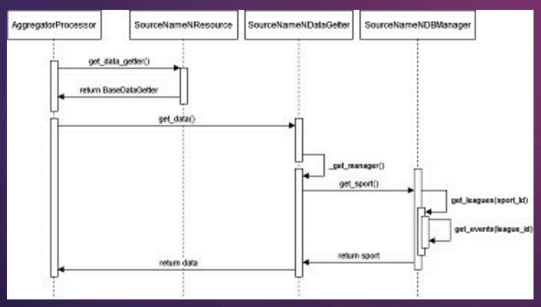
13

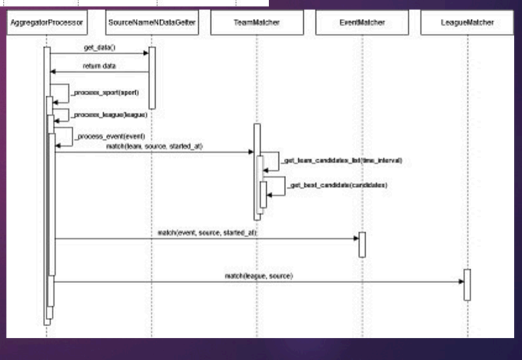


Діаграми послідовності

14

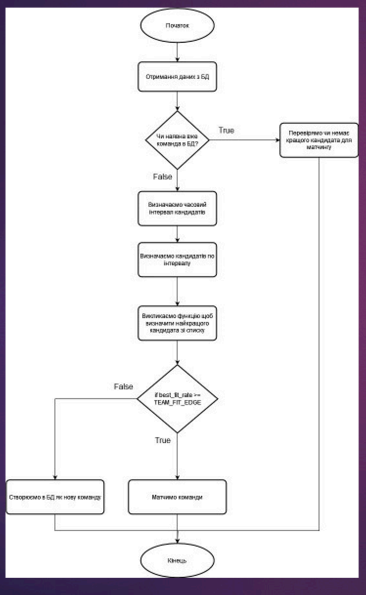


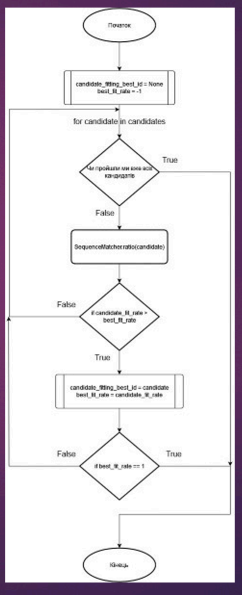




Блок-схеми матчингу

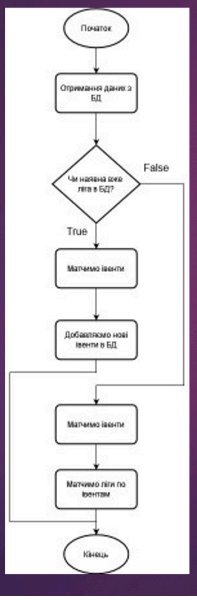
15

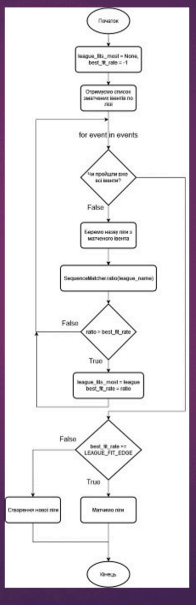




Блок-схеми матчингу

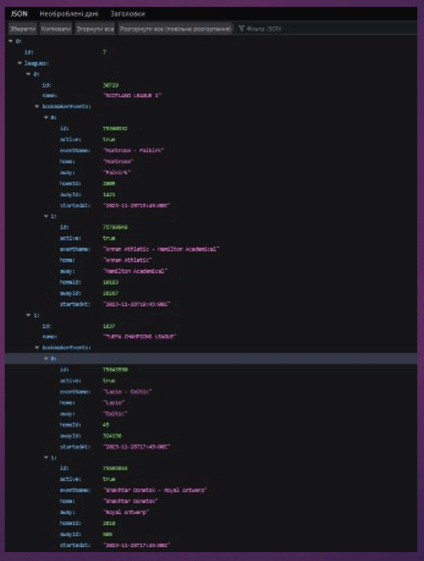
16





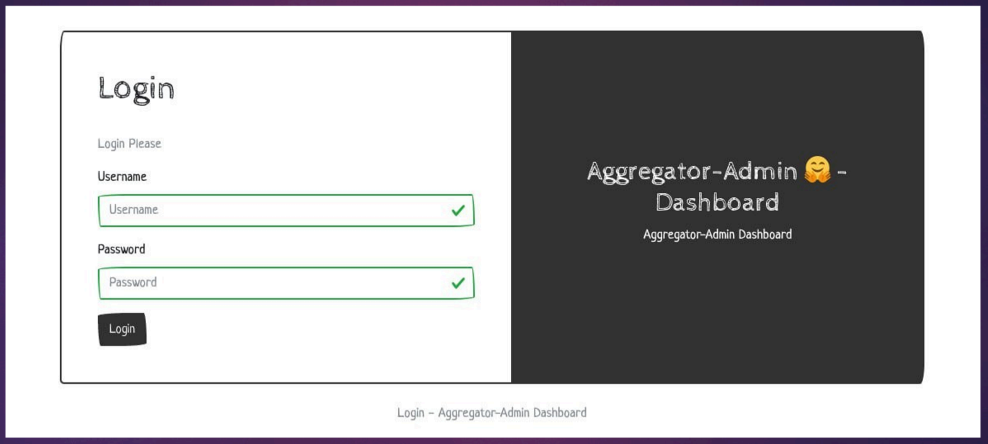
Результати розробки

17



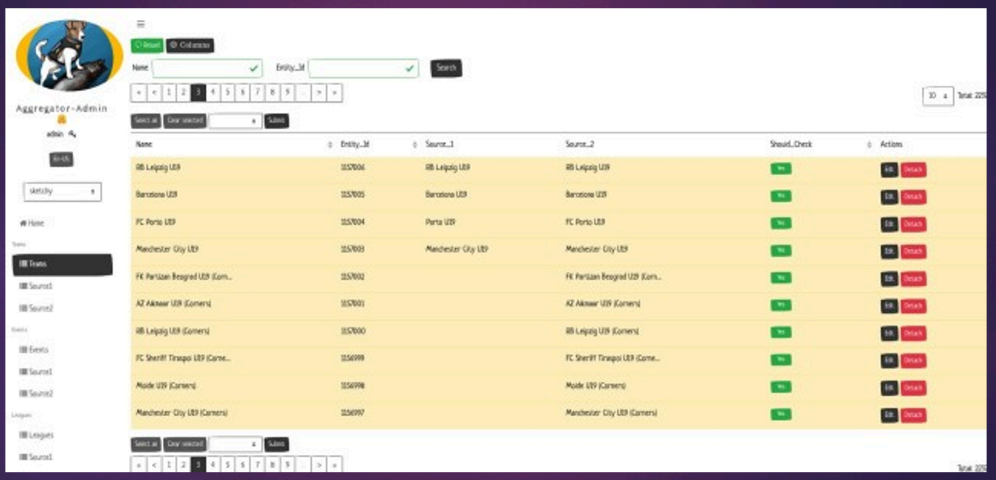
Результати розробки

18



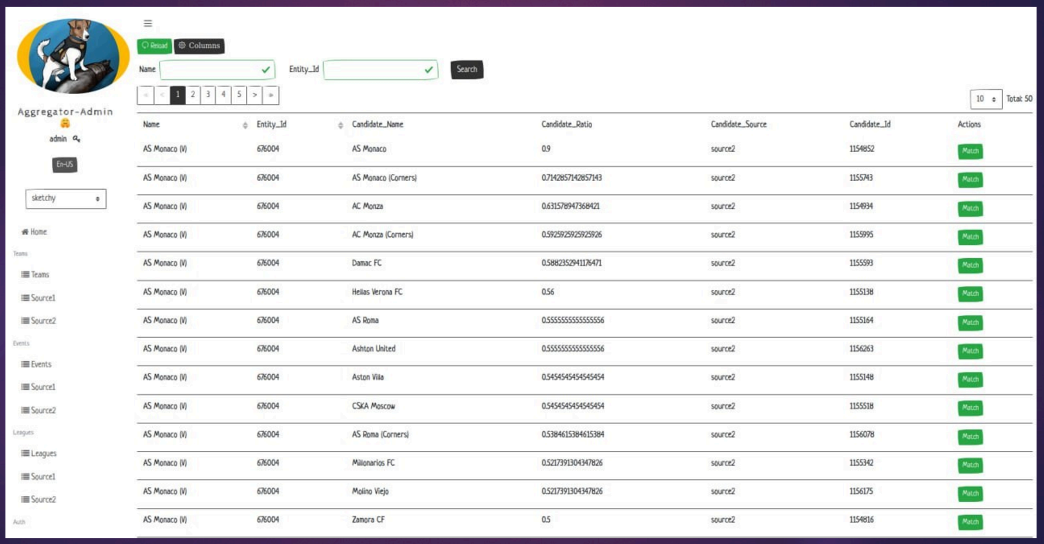
Результати розробки

19



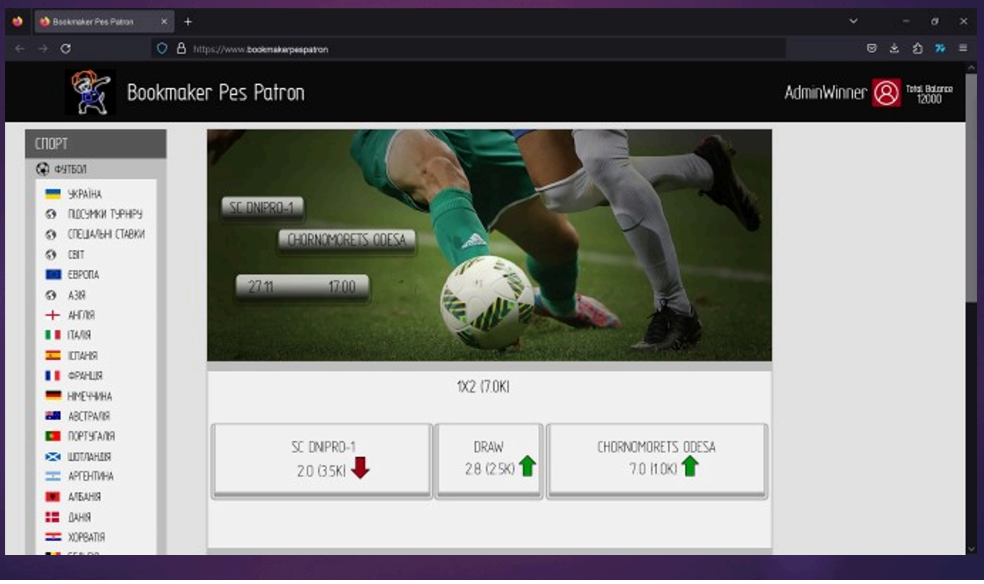
Результати розробки

20



Результати розробки

21



Висновок

22

В цій АВР проведено аналіз предметної області, визначено вимоги та очікування гравців, розглянуто методи та алгоритми для розробки букмекерської платформи, детально описано програмний інструментарій та представлено результати розробки.

Результати розробки свідчать про успішну імплементацію скраперів, ефективну роботу агрегатору та високу зручність інтерфейсу. Платформа надійно взаємодіє зі скраперами та агрегатором, забезпечуючи користувачам можливість здійснювати ставки на спортивні події з високою ефективністю та точністю інформації .

23

Дякую за увагу!