МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБСКРЕЙПІНГ ДЛЯ ПОШУКУ ДОГОВІРНИХ СПОРТИВНИХ МАТЧІВ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 401.22010101

Виконав студент 4-го курсу, групи 401
_____ М. В. Богінський
«17» червня 2024 р.

Керівник: канд. техн. наук, доцент
_____А. П. Бойко
«17» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Богінському Микиті Віталійовичу

1. Тема кваліфікаційної роботи «Вебскрейпінг для пошуку договірних спортивних матчів».

Керівник роботи Бойко Анжела Петрівна, канд. техн. наук, доц.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «17» червня 2024 р.

3. Вхідні (початкові) дані до роботи: експертні оцінки можливостей систем для пошуку договірних матчів.

Очікуваний результат: розроблений застосунок для пошуку договірних матчів за допомогою технології вебскрейпінг.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– загальна існуючих систем пошуку та попередження підозрілої активності у спортивних матчах;

– розробка архітектури системи та визначення необхідних компонентів;

– розробка алгоритмів пошуку та попередження підозрілої активності у спортивних матчах на основі аналізу даних, отримав від вебскрейпінгу;

– розробка програмного забезпечення для керування системою та збору даних;

– вибір та інтеграція необхідних програмних рішень;

– розробка інтерфейсу користувача для взаємодії з системою та відображення даних;

– тестування створеного застосунку.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Природне світло у виробничому приміщенні»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __канд. техн. наук, доц. Бойко А. П._____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Богінський М. В._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_14_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебскрейпінг для пошуку договірних спортивних матчів

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7	Виконання КРБ: аналіз існуючих систем для виявлення договірних матчів, огляд технологій, розробка програмного забезпечення	13.05.2024	22.06.2024	Виконано
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2024	26.06.2024	Виконано

Розробив студент __Богінський М. В._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ канд. техн. наук, доц. Бойко А. П.____ _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_29_» _____01_____ 2024 р.

АНОТАЦІЯ
кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили
Богінського Микити Віталійовича
Тема: «Вебскрейпінг для пошуку договірних спортивних матчів»

Актуальність даної роботи зумовлена зростанням кількості договірних спортивних матчів та необхідністю розробки ефективних методів їхнього виявлення. Використання технології вебскрейпінгу дозволяє автоматизувати процес збору та аналізу даних про спортивні події, що є важливим для своєчасного виявлення підозрілої активності та запобігання маніпуляціям результатами змагань.

Об'єкт роботи – процес скрепінгу договірних спортивних матчів.

Предмет роботи – технології розробки вебскрейпінгу договірних спортивних матчів.

Метою кваліфікаційної роботи є розробка та вдосконалення автоматизованої системи для пошуку договірних спортивних матчів на основі технології вебскрейпінгу. Одним із завдань було дослідження ефективності різних методів вебскрейпінгу для виявлення підозрілої активності у спортивних матчах.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків та додатків.

У першому розділі розглядається аналіз предметної сфери та постановка задачі для вебскрейпінгу договірних спортивних матчів.

У другому розділі досліджено методи, технології та інструменти вебскрейпінгу договірних спортивних матчів.

У третьому розділі описано створення та налаштування програмного забезпечення для вебскрейпінгу, використовуючи сучасні бібліотеки та технології. Апробація розробленої системи відбувалася на відкритих тестових наборах. Проведена оптимізація алгоритмів для отримання максимальної точності та ефективності системи.

У четвертому розділі наведено результати оптимізації системи, інформаційні діаграми, інструкцію користувача та графіки виявлених договірних матчів. У результаті розроблено систему для пошуку договірних спортивних матчів з використанням технології вебскрейпінгу, що демонструє високу ефективність та точність у виявленні підозрілої активності.

Кваліфікаційна робота містить 57 сторінок, 12 рисунків, 0 таблиць, 27 джерел та 1 додаток.

Ключові слова: вебскрейпінг, спортивні матчі, договірні матчі, автоматизація, аналіз даних.

ABSTRACT

for bachelor's qualification work of a student of group 401 of Petro Mohyla Black Sea National University

Mykyta Bohinskyi

Topic: "Web Scraping for Detecting Fixed Sports Matches"

The relevance of this work is due to the increasing number of fixed sports matches and the need to develop effective methods for detecting them. The use of web scraping technology allows automating the process of collecting and analyzing data on sports events, which is crucial for timely detection of suspicious activities and preventing match-fixing.

The object of the work is the process of scraping fixed sports matches.

The subject of the work is the technologies for developing web scraping for fixed sports matches.

The aim of the qualification work is to develop and improve an automated system for detecting fixed sports matches based on web scraping technology. One of the tasks was to study the effectiveness of various web scraping methods for detecting suspicious activities in sports matches.

The explanatory note consists of an introduction, four chapters, conclusions, and appendices.

The first chapter considers the analysis of the subject area and the formulation of the task for web scraping of fixed sports matches. The second chapter examines the methods, technologies, and tools of web scraping for fixed sports matches. The third chapter describes the creation and configuration of the software for web scraping, using modern libraries and technologies. The developed system was tested on open test sets. Additionally, algorithm optimization was carried out to achieve maximum accuracy and efficiency. The fourth chapter presents the results of system optimization, information diagrams, user instructions, and charts of detected fixed matches.

As a result, a system for detecting fixed sports matches using web scraping

technology was developed, demonstrating high efficiency and accuracy in identifying suspicious activities.

The qualification work contains 57 pages, 12 pictures, 0 tables, 27 references, and 1 appendice.

Key words: web scraping, sports matches, fixed matches, automation, data analysis.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 3
ВСТУП 4
1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ ТА ПОСТАНОВКА ЗАДАЧІ ДЛЯ ВЕБСКРЕЙПІНГ ПОШУКУ ДОГОВІРНИХ СПОРТИВНИХ МАТЧІВ 6
1.1 Основні терміни та поняття, пов‘язані з вебскрейпінг пошуком договірних матчів 6
1.2 Огляд сучасного стану веб скрейпинг в контексті пошуку договірних спортивних матчів 20
1.3 Постановка задачі для веб скрейпингу договірних спортивних матчів 21
Висновки до розділу 1 22
2 МЕТОДИ, ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ ВЕБСКРЕЙПІНГУ ДОГОВІРНИХ СПОРТИВНИХ МАТЧІВ 24
2.1 Огляд методів вебскрейпінгу та їх ефективність у виявленні договірних матчів 24
2.2 Вибір технологій для реалізації веб скрейпінгу 25
2.3 Опис використаних інструментів для веб скрейпингу договірних спортивних матчів 26
2.4 Огляд сучасних сервісів та API для отримання спортивних даних 29
Висновки до розділу 2 31
3 РЕАЛІЗАЦІЯ ОБРАНИХ ТЕХНОЛОГІЙ ТА СЕРВІСІВ ДЛЯ ПОСТАВЛЕНОЇ ЗАДАЧІ СТВОРЕННЯ ВЕБСКРЕЙПЕРА 32
3.1 Реалізація технічної частини вебскрейперу 32
3.2 Реалізація технічної частини вебскрейперу 37
Висновки до розділу 3 44
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА КЕРІВНИЦТВО КОРИСТУВАЧА 45
4.1 Деталі процесу збору даних 45
4.2 Візуалізація результатів 46
Висновки до розділу 4 49
ВИСНОВКИ 50
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 52
ДОДАТОК А Лістинг коду 55

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	База даних
ETL	–	Extract, transform, load (витяг, перетворення та завантаження)

ВСТУП

Пошук договірних матчів стає надзвичайно важливим у сучасному спортивному світі через їх серйозний вплив на довіру до чесності та справедливості змагань. З ростом популярності спортивних заходів і збільшенням кількості турнірів та змагань, знаходження вигідних та безпечних контрактів стає ключовим аспектом для спортивних організацій та спортсменів.

Є багато способів виявити нечесність команди, наприклад, стежити за новинами та форумами, соціальними мережами гравців, тренерів та інших осіб, які можуть бути причетні до договірних матчів. Робити приватні розслідування, як це роблять великі змагання, якщо є підозра у нечесності.

Але всі вони потребують великих людських ресурсів. У цьому може допомогти сучасні технології, які допоможуть автоматично збирати дані та допомагати ефективному аналізу, які не тільки у декількох випадках розслідування, а взагалі стежити за станом сучасного спортивного світу і вчасно реагувати на підозрілу активність.

Мета даної роботи полягає у розробці та вдосконаленні автоматизованої системи пошуку договірних матчів.

Об’єкт дослідження: процес скрейпингу договірних спортивних матчів.

Предмет дослідження: технології розробки вебскрейпингу договірних спортивних матчів.

Основні завдання:

– аналіз існуючих систем пошуку та попередження підозрілої активності у спортивних матчах;

– розроблення архітектури системи та визначення необхідних компонентів та модулів;

– розроблення алгоритмів пошуку та попередження підозрілої активності у спортивних матчах на основі аналізу даних, отриманих від веб-скрапінгу;

– розроблення програмного забезпечення для керування системою та збору даних;

– вибір та інтеграція необхідних програмних рішень;

– розроблення інтерфейсу користувача для взаємодії з системою та відображення даних;

– тестування створеного застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ ТА ПОСТАНОВКА ЗАДАЧІ ДЛЯ ВЕБСКРЕЙПІНГ ПОШУКУ ДОГОВІРНИХ СПОРТИВНИХ МАТЧІВ

1.1 Основні терміни та поняття, пов‘язані з вебскрейпінг пошуком договірних матчів

1.1.1 Вебскрейпінг

Вебскрейпінг [24] це процес автоматичного збору даних з веб-сайтів за допомогою програм або скриптів [2]. Цей метод широко використовується для автоматизованого отримання інформації з Інтернету без прямої взаємодії з користувачем. Вебскрейпінг дозволяє отримувати доступ до різноманітної інформації з веб-сайтів, такої як ціни на товари, новини, дані про продукти або послуги, оцінки, відгуки тощо.

Основна ідея вебскрейпінгу полягає у програмному аналізі HTML-коду веб-сторінок і вилученні корисної інформації з визначених ресурсів. Для цього використовуються спеціальні інструменти або бібліотеки, які дозволяють здійснювати HTTP-запити, отримувати веб-сторінки, парсити їх HTML-структуру і видобувати необхідні дані.

Вебскрейпінг застосовується в багатьох галузях, включаючи електронну комерцію (для моніторингу цін, аналізу конкурентів), дослідження ринків, аналітику даних, наукові дослідження (для збору статистики, текстового аналізу) та багато іншого. Проте важливо пам'ятати, що вебскрейпінг повинен відбуватися в межах закону та з дотриманням політики веб-сайту щодо збору даних.

1.1.2 Договірні матчі

Договірні матчі (або "фіксовані матчі") є серйозним порушенням етичних принципів та спортивних стандартів [11], коли учасники спортивних змагань домовляються про результати заздалегідь з метою отримання матеріальної вигоди або

інших переваг. Ця практика підриває основи чесної та справедливої конкуренції в спорті та порушує довіру вболівальників та спортивних організацій.

Основні аспекти договірних матчів включають:

– узгодження результатів: учасники спортивних змагань домовляються про результати перед початком гри. Наприклад, команди можуть домовитись про те, хто переможе або про точний рахунок гри;

– мотивація: головною мотивацією за участь у договірних матчах є фінансова вигода. Основні ставки можуть стосуватися спортивних ставок, зокрема у букмекерських конторах, де результат гри вже відомий учасникам;

– порушення інтегритету спорту: договірні матчі порушують основні принципи спорту, такі як справедливість, чесність та конкурентоспроможність. Це ставить під загрозу інтегритет спортивних змагань і веде до знехтування з боку глядачів та учасників;

– правові наслідки: у багатьох країнах договірні матчі вважаються злочином і можуть мати серйозні правові наслідки для учасників і організаторів. Це може включати штрафи, дискваліфікацію, судові позови або навіть кримінальне переслідування;

– боротьба з явищем: спортивні організації та правоохоронні органи активно борються з договірними матчами. Вони використовують різні методи, включаючи моніторинг, розслідування, впровадження строгих правил та санкцій для запобігання цьому явищу.

Узгоджені результати спортивних змагань підривають довіру у спортивному світі та впливають на цінності, на яких ґрунтується спорт. Запобігання договірним матчам є важливим завданням для збереження інтегритету та справедливості у спорті.

1.1.3 HTML (HyperText Markup Language)

Це стандартна мова розмітки, що використовується для створення структури

веб-сторінок. HTML визначає складові елементи веб-сторінки, такі як заголовки, абзаци, списки, таблиці, зображення, посилання та інші елементи, які спільно створюють вміст сторінки.

Основні характеристики HTML:

– теги: HTML базується на використанні тегів, які визначають різні елементи сторінки. Теги мають вигляд <назва_тегу>, існують парні теги, наприклад <p> для абзаців, та одиночні теги, наприклад <img> для вставки зображень;

– структура документа: HTML-документ має структуру, що складається з різних секцій. Основні секції включають <head>, де розміщуються метадані (наприклад, заголовок сторінки і підключення до CSS або JavaScript), та <body>, де розміщується вміст сторінки, видимий для користувачів;

– елементи сторінки: HTML має багато різних елементів для відображення тексту, медіа-файлів, посилань та іншого. Деякі основні елементи включають:

  – <p> для абзаців тексту;
  – <h1>, <h2>, <h3>, ... <h6> для заголовків різного рівня;
  – та <ol> для ненумерованого та нумерованого списків відповідно;
  – <img> для вставки зображень;
  – <a> для створення посилань;
  – <table> для створення таблиць;
  – <form> для створення форм для відправки даних на сервер;

– атрибути: теги можуть мати атрибути, які дозволяють додатково налаштовувати поведінку або вигляд елементів. Наприклад, атрибут src використовується для вказання шляху до зображення у тегу <img>, а атрибут href використовується для вказання URL-адреси посилання у тегу <a>;

– семантика: HTML використовується для створення семантичної структури сторінок, що сприяє кращому розумінню вмісту для пошукових систем та адаптивності для асистентів та апаратних пристроїв. Семантичні елементи HTML, такі

як <header>, <footer>, <nav>, <article>, <section>, допомагають уявити структуру сторінки та зробити її більш доступною для інтерпретації.

HTML є основою веб-розробки і є важливою складовою будь-якого веб-сайту. Він використовується разом з CSS для створення структури і вигляду сторінок, а також з JavaScript для динамічної інтерактивності. Розуміння HTML є важливим для всіх, хто цікавиться веб-розробкою та створенням вмісту для Інтернету.

1.1.4 CSS (Cascading Style Sheets)

Це мова стилів, яка використовується для опису зовнішнього вигляду і форматування веб-сторінок. Головна мета CSS полягає в розділенні представлення (стилю) веб-сторінок від їх вмісту (структури), що дозволяє створювати стильові правила для керування виглядом і відображенням елементів на сторінці.

Основні аспекти CSS:

– селектори: CSS використовує селектори для вибору елементів на сторінці, до яких будуть застосовуватися стилі. Наприклад, селектор h1 застосовує стилі до всіх заголовків першого рівня (<h1>);

– властивості і значення: кожна стилізована властивість має значення, яке вказує, як буде виглядати відповідний елемент. Наприклад, властивість color визначає колір тексту, а font-size - розмір шрифту;

– каскадування і спадковість: концепція "каскадування" означає, що стилі можуть спадати від батьківських елементів до дочірніх елементів у веб-дереві. Це дозволяє ефективно керувати стилями і застосовувати загальні стилі до багатьох елементів одночасно;

– боксова модель: CSS визначає боксову модель, яка описує, як веб-елемент відображається у вигляді прямокутного блоку. Боксова модель включає властивості, такі як width, height, padding, border і margin, що визначають розмір, відступи та рамку елемента;

– різноманітність елементів стилізації: CSS дозволяє стилізувати різноманітні елементи веб-сторінок, такі як текст, фон, межі, відступи, розміри шрифтів, кольори, розташування елементів та багато іншого. Це дозволяє створювати привабливі та функціональні веб-інтерфейси;

– медіа-запити: CSS підтримує медіа-запити, які дозволяють застосовувати різні стилі залежно від параметрів пристрою, таких як ширина вікна, роздільна здатність екрану або тип пристрою. Це дозволяє створювати адаптивний дизайн, який виглядає добре на різних пристроях;

– перетворення і анімації: CSS дозволяє застосовувати перетворення (такі як обертання, масштабування, зсув) і створювати анімації для створення динамічних інтерфейсів та ефектів.

CSS є потужним інструментом для веб-розробників, що дозволяє створювати стильовані інтерфейси з великою гнучкістю і контролем. Розуміння CSS є важливим для досягнення сучасних стандартів дизайну та розробки веб-сторінок.

1.1.5 XPath

Це мова запитів для навігації та вибору елементів у структурованих документах, зокрема в XML-документах. XPath надає можливість точно вибирати певні частини документа за допомогою шляхів або виразів, що дозволяє ефективно виконувати скрейпінг веб-сторінок та аналізувати дані.

Основні концепції XPath:

– шляхи до елементів: XPath використовує шляхи для навігації по структурі документа і вибору потрібних елементів. Наприклад, шлях //div[@class='content'] вибирає всі елементи з класом content у всьому документі.

– ширази вибору: XPath дозволяє використовувати різні вирази вибору для визначення, які елементи вибирати. Наприклад:

– //p - вибирає всі елементи <p> у документі;

– //div[@id='main'] - вибирає всі елементи з ідентифікатором main;

– //a[@href='https://example.com'] - вибирає всі посилання <a> з атрибутом href, що містить https://example.com.

– вирази фільтрації: XPath дозволяє застосовувати умови для фільтрації вибору елементів. Наприклад:

– //div[@class='article'] - вибирає всі елементи з класом article;

– //p[@id='intro'][@class='important'] - вибирає всі елементи <p> з ідентифікатором intro і класом important;

– відносний шлях: XPath також підтримує відносні шляхи, що починаються з певного вузла. Наприклад:

– .//span - вибирає всі елементи в межах поточного контексту;

– вирази вибору тексту і атрибутів: XPath дозволяє вибирати текстовий вміст елементів або значення їх атрибутів. Наприклад:

– //h1/text() - вибирає текстовий вміст усіх елементів <h1>;

– //img/@src - вибирає значення атрибута src усіх елементів <img>.

– функції: XPath містить різноманітні вбудовані функції для обробки та маніпулювання даними. Наприклад, функція contains() використовується для перевірки наявності певного тексту у вмісті елемента.

XPath є потужним інструментом для скрейпінгу веб-сторінок, особливо коли структура документа є складною або потрібно вибирати конкретні елементи за певними критеріями. Він дозволяє точно визначати та отримувати потрібні дані для подальшого аналізу чи обробки.

1.1.6 Краулер або скрапер (також відомий як "бот" або "спайдер")

Це програмне забезпечення, призначене для автоматичного перегляду веб-сторінок з метою отримання інформації з Інтернету. Основна функція краулера полягає в автоматизованому зборі даних з веб-сайтів для подальшого аналізу, обробки

або зберігання.

Основні характеристики краулера:

– автоматизація перегляду: краулери автоматично переглядають веб-сторінки за допомогою програмної логіки, що імітує поведінку користувача в Інтернеті. Вони можуть переходити по гіперпосиланнях, аналізувати структуру сторінок і витягувати корисну інформацію;

– збір даних: краулери здатні збирати різноманітні дані з веб-сторінок, такі як текст, зображення, посилання, метадані та інше. Ці дані можуть використовуватись для аналізу ринків, моніторингу конкурентів, агрегації контенту або для створення індексів для пошукових систем;

– обробка і аналіз даних: після збору даних краулер може виконувати обробку і аналіз інформації згідно з встановленими правилами і логікою. Наприклад, він може витягувати статистику, розпізнавати певні шаблони або класифікувати дані за категоріями;

– масштабованість: краулери здатні працювати у масштабі, переглядаючи тисячі або навіть мільйони веб-сторінок за короткий проміжок часу. Вони можуть оптимізувати час перегляду і обробки, використовуючи паралельні процеси та інші техніки;

– використання в різних галузях: краулери широко використовуються у різних галузях, таких як електронна комерція (для збору цін та описів товарів), дослідження ринків (для аналізу трендів), медіа і змістовий маркетинг (для агрегації контенту), фінанси (для моніторингу фінансових даних) та інші галузі.

1.1.7 Сесія

Це механізм, який дозволяє зберігати дані про стан взаємодії між веб-сервером і клієнтом протягом певного часу. Сесії використовуються для зберігання інформації про стан певного користувача під час його візиту на веб-сайті. Основні аспекти сесій

включають:

– унікальність ідентифікатора: кожна сесія має унікальний ідентифікатор (session ID), який використовується для ідентифікації конкретної сесії між клієнтом і сервером. Цей ідентифікатор зазвичай передається через куки (cookies) або URL-параметри;

– зберігання стану: сесії дозволяють зберігати дані про стан взаємодії між клієнтом і сервером. Ці дані можуть включати інформацію про авторизацію користувача, обрані налаштування, діючий контекст роботи з сайтом тощо;

– час життя сесії: сесії мають обмежений час життя, який визначається налаштуваннями сервера. Часто це час між початком і завершенням сеансу користувача на веб-сайті. Після закінчення терміну дії сесії інформація про стан втрачається;

– захист і безпека: сесії використовуються для забезпечення безпеки взаємодії між клієнтом і сервером. Наприклад, ідентифікатор сесії може бути шифрованим для запобігання підмені ідентифікатора або несанкціонованого доступу до даних сесії;

– приклад застосування: сесії широко використовуються для зберігання стану аутентифікації користувача під час його перебування на веб-сайті. Наприклад, після успішного входу на сайт ідентифікатор сесії зберігається на стороні клієнта, що дозволяє серверу відслідковувати стан авторизації під час взаємодії з користувачем;

– типи сесій: існують різні типи сесій, такі як сесії на основі сеансів (session-based sessions) або сесії на основі стану (stateful sessions), які використовуються для зберігання стану між запитами клієнта до сервера.

1.1.8 Проксі-сервер (proxy server)

Це проміжний сервер, що діє як посередник між клієнтом і веб-сервером. Основна функція проксі-сервера полягає в пересиланні запитів і відповідей між користувачем і веб-сервером, забезпечуючи при цьому додатковий рівень контролю

та фільтрації трафіку. Основні аспекти проксі-сервера включають:

– пересилання запитів: проксі-сервер перенаправляє запити користувача до веб-сервера і отримує відповіді від сервера, виступаючи при цьому як посередник. Це дозволяє зберігати анонімність і приховувати реальні IP-адреси клієнтів від сервера;

– кешування контенту: проксі-сервер може зберігати кешовані копії запитаного контенту, таких як веб-сторінки, зображення або відео. Це дозволяє прискорити доступ до популярного контенту і зменшити трафік мережі;

– фільтрація трафіку: проксі-сервер може використовуватись для фільтрації трафіку і блокування небажаних веб-сайтів або контенту. Він може обмежувати доступ до певних ресурсів відповідно до налаштуваних правил і обмежень;

– підвищення безпеки: проксі-сервер допомагає підвищити безпеку мережі, ховаючи реальні IP-адреси користувачів від зовнішніх джерел і захищаючи мережу від атак типу DDoS або SQL-ін'єкцій;

– перепрограмування трафіку: деякі проксі-сервери можуть змінювати та перепрограмовувати запити і відповіді між клієнтом і сервером. Це дозволяє встановлювати правила маршрутизації, обробляти запити до сервера і оптимізувати трафік;

– внутрішні мережеві проксі: проксі-сервери також використовуються у внутрішніх мережах організацій для контролю доступу до Інтернету, моніторингу використання мережі та захисту корпоративної інформації.

1.1.9 Application Programming Interface (API)

Це набір правил, протоколів і інструментів, які дозволяють різним програмним компонентам взаємодіяти між собою. API визначає способи, за допомогою яких програми можуть обмінюватися даними і функціями, надаючи іншим програмам доступ до своїх можливостей.

Основні аспекти API включають:

– визначення функціональності: API визначає, які операції і функції можуть використовувати інші програми для спілкування з даним програмним компонентом чи сервісом. Це включає доступ до певних функцій, операцій з даними, а також обмін повідомленнями і даними;

– стандартизація інтерфейсу: API встановлює стандарти і правила для взаємодії між різними програмними компонентами. Це дозволяє розробникам створювати програми, які легко інтегруються з існуючими сервісами і компонентами;

– управління доступом: API може контролювати доступ до певних функцій чи даних, використовуючи механізми автентифікації і авторизації. Це забезпечує безпеку і конфіденційність даних під час взаємодії між програмами;

– формати обміну даними: API визначає формати обміну даними, такі як JSON, XML або інші стандарти, які використовуються для передачі інформації між програмними компонентами;

– документація та приклади використання: API надає документацію, описуючи доступні операції, їх параметри і приклади використання. Це допомагає розробникам розуміти, як користуватися API і інтегрувати його в свої програми.

1.1.10 ETL (Extract, Transform, Load)

Це процес обробки даних, що включає вилучення (extract), трансформацію (transform) і завантаження (load) даних з одного джерела в інше, зазвичай з великим обсягом даних. Цей процес є ключовим у сферах обробки даних, бізнес-аналітики, бізнес-інтелекту, інтеграції даних і Data Warehousing.

Основні етапи ETL [21]:

– вилучення (Extract): вилучення включає отримання даних з одного чи декількох джерел даних. Джерелами можуть бути бази даних, текстові файли, веб-сервери, API, спеціалізовані додатки тощо. Під час вилучення дані зазвичай копіюються або експортуються з джерела до проміжного сховища для подальшої

обробки;

– трансформація (Transform): трансформація включає обробку та перетворення даних згідно з вимогами бізнесу або системи. Цей етап може включати фільтрацію, очищення, перетворення типів даних, об'єднання даних з різних джерел, агрегацію, уніфікацію формату даних тощо. Метою трансформації є підготовка даних для оптимального аналізу, звітування або завантаження до цільової системи;

– завантаження (Load): завантаження включає вставку або завантаження оброблених і трансформованих даних до цільової бази даних, дата-складу чи системи, яка використовує ці дані для аналізу, звітування або іншої обробки. Цільовою системою може бути Data Warehouse, Data Mart, операційна база даних або інше сховище даних.

Основні переваги ETL включають [23]:

– оптимізація обробки даних: ETL дозволяє автоматизувати процеси обробки даних, що допомагає ефективно переміщувати та аналізувати великі обсяги інформації;

– стандартизація і якість даних: етап трансформації дозволяє стандартизувати дані і покращує їх якість перед завантаженням;

– інтеграція даних: ETL дозволяє інтегрувати дані з різних джерел для створення комплексних систем обробки даних;

– підтримка бізнес-аналітики: чисті та оброблені дані, що завантажуються після ETL, використовуються для бізнес-аналізу, звітування та прийняття рішень.

ETL є ключовим компонентом для інтеграції даних в організаціях, оскільки він дозволяє ефективно керувати і аналізувати великі обсяги інформації з різних джерел для підтримки бізнес-процесів і прийняття стратегічних рішень.

1.1.11 База даних (Database)

Це структурована колекція даних, організована для ефективного зберігання,

управління та доступу до інформації. Бази даних використовуються для зберігання великих обсягів даних у структурованому форматі, що дозволяє здійснювати швидкий доступ до інформації і забезпечувати її цілісність та безпеку. Основні аспекти баз даних включають:

– структура даних: база даних визначає структуру даних, включаючи типи даних, відносини між даними, обмеження цілісності та інші характеристики. Дані організовані у вигляді таблиць (реляційні бази даних), де кожна таблиця має стовпці з певними типами даних;

– мова запитів: для взаємодії з базою даних використовується мова запитів, яка дозволяє створювати, змінювати, видаляти та опитувати дані. Найпоширенішою мовою запитів є SQL (Structured Query Language);

– управління даними: система управління базами даних (СУБД) відповідає за управління базою даних. СУБД забезпечує доступ до даних, виконує запити, забезпечує цілісність даних, контролює доступ користувачів та інші функції;

– цілісність даних: база даних дотримується правил цілісності, які гарантують коректність та унікальність даних. Ці правила включають обмеження цілісності даних, унікальність ключів, зовнішні ключі тощо;

– безпека даних: база даних забезпечує механізми безпеки для захисту конфіденційності, цілісності та доступності даних. Це включає автентифікацію користувачів, авторизацію доступу, шифрування даних та інші заходи безпеки;

– швидкий доступ до даних: оптимізована структура бази даних дозволяє швидкий доступ до даних за допомогою запитів, індексації та інших методів оптимізації.

Бази даних використовуються у багатьох галузях інформаційних технологій, включаючи веб-розробку, бізнес-аналітику, системи управління відносинами з клієнтами (CRM), системи управління виробництвом (ERP), системи управління контентом (CMS) та інші. Вони є основою для зберігання, організації та аналізу даних,

необхідних для прийняття рішень і підтримки бізнес-процесів.

1.1.12 Куки (Cookies)

Це невеликі фрагменти даних, які веб-сайти відправляють і зберігають на комп'ютері користувача через веб-браузер. Вони використовуються для зберігання інформації про користувача та їхню взаємодію з веб-сайтом. Куки є важливим механізмом для збереження стану сесії, налаштувань користувача, історії перегляду та інших даних, які дозволяють забезпечити персоналізований досвід користувача на веб-сайті.

Основні характеристики куків:

– зберігання інформації: куки дозволяють веб-сайтам зберігати певні дані на комп'ютері користувача, наприклад, ідентифікатор сесії, налаштування сайту, історію перегляду, згоди на використання файлів cookie тощо;

– стан сесії: куки використовуються для збереження стану сесії користувача. Наприклад, після входу на сайт користувача, куки можуть зберігати ідентифікатор сесії, щоб сервер знав, що це той самий користувач під час перегляду сторінок;

– персоналізація контенту: куки використовуються для збереження налаштувань та персоналізації контенту на веб-сайті відповідно до вподобань користувача;

– відстеження активності: куки дозволяють веб-сайтам відстежувати активність користувача, наприклад, їхні дії на сайті, інтереси та звички перегляду;

– аналіз і маркетинг: куки використовуються для збирання даних про користувачів для аналізу відвідуваності веб-сайту та рекламних кампаній.

Хоча куки є корисним інструментом для покращення користувацького досвіду, вони також можуть породжувати проблеми з приватністю і безпекою даних. Наприклад, деякі користувачі можуть бути обурені відстеженням їхньої активності, а куки також можуть бути використані для створення спеціальної технології,

спрямованої на рекламу або персоналізовані рекламні повідомлення.

У браузерах зазвичай є можливість керувати тим, як вони обробляють файли cookie, наприклад, встановлювати обмеження на їхнє збереження або видаляти старі файли cookie. Це дозволяє користувачам контролювати свою приватність та безпеку під час використання веб-сайтів, які використовують файли cookie для збереження даних.

1.1.13 HTTP (Hypertext Transfer Protocol)

Це протокол передачі даних, що використовується для комунікації між веб-серверами і клієнтами (наприклад, веб-браузерами). Він визначає правила обміну інформацією між веб-клієнтом і веб-сервером, включаючи передачу гіпертекстових документів, зображень, відео та інших ресурсів через Інтернет.

Основні характеристики HTTP:

– запит-відповідь: Протокол HTTP базується на моделі клієнт-сервер, де клієнт (наприклад, веб-браузер) надсилає запит до сервера, а сервер відправляє відповідь з необхідними даними (наприклад, веб-сторінкою або ресурсом);

– методи запитів: HTTP визначає різні методи запитів, такі як GET, POST, PUT, DELETE, PATCH та інші. Кожен метод має свою специфікацію та використовується для виконання певних дій з веб-ресурсами. Наприклад:

– GET: запит на отримання вмісту ресурсу;

– POST: відправка даних для обробки на сервері (наприклад, відправка форми);

– PUT: оновлення вмісту ресурсу;

– DELETE: видалення ресурсу;

– PATCH: часткове оновлення ресурсу;

– заголовки: HTTP включає заголовки, що містять метадані про запит або відповідь. Заголовки можуть містити інформацію про тип контенту, кодування, розмір

файлу, кешування тощо;

– сесії і куки: HTTP є безстічним протоколом, що означає, що кожен запит обробляється незалежно від попередніх запитів. Для збереження стану сесії і забезпечення послідовності дій використовуються куки і інші механізми (наприклад, сеанси);

– безпека: HTTP може бути забезпечений за допомогою HTTPS (HTTP over SSL/TLS), що забезпечує шифрування даних між веб-клієнтом і веб-сервером для забезпечення приватності та безпеки даних;

– кешування: HTTP підтримує механізми кешування, що дозволяють зберігати копії ресурсів на клієнтському боці або посередницьких серверах для покращення швидкодії та ефективності.

HTTP є основним протоколом для взаємодії між веб-клієнтами і веб-серверами, і він використовується у більшості веб-додатків і сервісів. Існують різні версії протоколу HTTP, такі як HTTP/1.1, HTTP/2 та HTTP/3, кожна з яких має свої характеристики і вдосконалення для оптимізації передачі даних через мережу Інтернет.

1.2 Огляд сучасного стану веб скрейпинг в контексті пошуку договірних спортивних матчів

Для вивчення питання вебскрейпінгу в контексті пошуку договірних спортивних матчів, варто звернутися до існуючих ресурсів, які надають інструменти для отримання даних з веб-сайтів букмекерів [3]. Ось деякі з цих ресурсів:

– gingeleski/odds-portal-scraper [4];

– Seb943/scrapeOP [5];

– S1M0N38/soccerapi [6];

– cvidan/bet365-scraper [7].

Ці ресурси представляють інструменти, які дозволяють отримувати дані про

ставки та результати спортивних подій. Однак, варто відзначити, що більшість з них мають певні обмеження та недоліки:

– деякі з цих скрейперів вже досить старі і можуть не відповідати сучасним вимогам. Наприклад, код може бути неуніверсальним або вимагати значних модифікаій для роботи з сучасними веб-сайтами;

– більшість існуючих рішень спеціалізуються на певних видах спорту або конкретних букмекерських сайтах [12]. Це може ускладнити їх використання для аналізу широкого спектру спортивних подій або для виявлення договірних матчів у різних спортивних дисциплінах;

– деякі скрейпери можуть мати обмежені можливості аналізу даних для виявлення незвичайних змін у коефіцієнтах або результатів, що можуть свідчити про договірні дії.

Враховуючи ці обмеження, дослідження нових підходів до вебскрейпінгу для аналізу договірних спортивних матчів може стати актуальним напрямком розвитку. Такі підходи можуть включати розробку більш універсальних інструментів, які забезпечать ефективний збір та аналіз даних для виявлення потенційних договірних матчів у різних спортивних галузях.

1.3 Постановка задачі для веб скрейпингу договірних спортивних матчів

Для розв'язання завдання з вебскрейпінгу договірних спортивних матчів необхідно визначити конкретні вимоги до розробки програмного забезпечення з наступними характеристиками:

– універсальність в межах однієї платформи. Оскільки кожен веб-сайт букмекера має власну структуру даних та методи їх отримання, програма повинна бути здатна збирати усі дані з що стосуються пошуку договірних спортивних матчів. Важливо захоплювати максимально можливу кількість даних, які потім можна фільтрувати та обробляти;

– частота отримання даних: оскільки пошук договірних матчів базується на різких змінах коефіцієнтів на результати подій, програма повинна регулярно запитувати дані, навіть з інтервалом близько до 30 секунд, для вчасного виявлення змін;

– модульність: система повинна бути розділена на окремі модулі, кожен з яких відповідає за збір даних з певного джерела. Це дозволить легко розширювати підтримку нових джерел даних або вносити зміни у вже існуючі модулі без змін у загальній структурі програми [13];

– уніфікація: отримані дані повинні бути перетворені в однаковий формат для подальшого аналізу. Це дозволить забезпечити однаковість обробки і аналізу даних з різних джерел;

– візуалізація: оскільки веб-скрейпер використовується як вспоміжний інструмент для ідентифікації договірних матчів, зібрані дані повинні бути зручно візуалізовані. Графіки, діаграми або інші форми візуалізації дозволять оперативно виділяти потенційно підозрілі зміни в коефіцієнтах або результатів матчів.

Ці вимоги дозволять створити ефективний інструмент для збору та аналізу даних про спортивні події з метою виявлення можливих договірних дій. Реалізація такого інструменту вимагатиме комплексного підходу до програмування з урахуванням специфіки роботи з веб-сайтами букмекерів та потреб аналізу даних в режимі реального часу.

Висновки до розділу 1

У цьому розділі проведено детальний аналіз предметної сфери, пов'язаної з вебскрейпінгом договірних спортивних матчів. Було визначено основні терміни та поняття, такі як вебскрейпінг та договірні матчі [10]. Вебскрейпінг визначається як процес автоматичного збору даних з веб-сайтів за допомогою програм або скриптів. Цей метод дозволяє отримувати різноманітну інформацію з веб-сайтів без прямої

взаємодії з користувачем, аналізуючи HTML-код веб-сторінок та вилучаючи корисну інформацію з визначених ресурсів.

Також було проведено огляд сучасного стану вебскрейпінгу в контексті пошуку договірних спортивних матчів, включаючи розгляд існуючих інструментів для отримання даних з веб-сайтів букмекерів. Однак більшість цих інструментів мають певні обмеження та недоліки, такі як застарілий код або потреба у значних модифікаціях для роботи з сучасними веб-сайтами.

Постановка задачі для вебскрейпінгу договірних спортивних матчів включає визначення вимог до розробки програмного забезпечення, яке повинно бути універсальним, модульним, забезпечувати уніфікацію даних та їх візуалізацію [8]. Також важливо, щоб програма регулярно запитувала дані з інтервалом до 30 секунд для вчасного виявлення змін у коефіцієнтах на результати подій.

2 МЕТОДИ, ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ ВЕБСКРЕЙПІНГУ ДОГОВІРНИХ СПОРТИВНИХ МАТЧІВ

2.1 Огляд методів вебскрейпінгу та їх ефективність у виявленні договірних матчів

Один із методів вебскрейпінгу, що досліджується, полягає у використанні автоматизованих скриптів для періодичного отримання даних з веб-сайтів букмекерів [9]. Ці скрипти можуть аналізувати зміни у коефіцієнтах ставок на різні спортивні події та спостерігати за несподіваними змінами, що можуть бути ознакою договірних дій. Наприклад, раптове падіння коефіцієнта на певну команду без очевидної причини може викликати підозру.

Інший підхід до вебскрейпінгу полягає у зборі та аналізі великої кількості даних про ставки з різних джерел, включаючи соціальні медіа та форуми спортивних спільнот. Цей метод дозволяє виявляти патерни або образці в ставках, які можуть свідчити про незвичайну активність або домовленості між учасниками.

Крім того, аналіз тексту та контенту ставок на спорт є ще одним ефективним методом. Під час цього аналізу можна використовувати природну мову оброблення для виявлення ключових слів або фраз, що можуть вказувати на договірні дії. Наприклад, певні комбінації слів у коментарях або описах ставок можуть бути підозрілими.

Додатково, застосування методів машинного навчання [14] для аналізу змін у коефіцієнтах ставок відкриває нові можливості у виявленні договірних дій. Моделі машинного навчання можуть навчитися розпізнавати аномалії у змінах коефіцієнтів, які можуть вказувати на недоречну діяльність.

Серед усіх цих підходів, аналіз змін у коефіцієнтах ставок вважається найбільш ефективним для виявлення договірних матчів. Цей підхід дозволяє систематично виявляти відхилення у коефіцієнтах, які можуть бути результатом маніпуляцій із

спортивними подіями.

У підсумку, вебскрейпінг є важливим інструментом для спортивних аналітиків у боротьбі з договірними діями. Розглянуті методи дозволяють систематично аналізувати дані про ставки на спорт і виявляти патерни, що можуть свідчити про недоречну діяльність у світі спорту.

2.2 Вибір технологій для реалізації веб скрейпінгу

При виборі технологій для реалізації веб скрейпінгу важливо враховувати як ефективність аналізу змін у коефіцієнтах ставок, так і затратність та доступність інструментів. Одним з найбільш ефективних методів аналізу є спостереження за змінами в коефіцієнтах, оскільки це дозволяє виявляти можливі договірні дії або недоречну активність на ринку ставок.

Python [22] є однією з найбільш популярних мов програмування для веб скрейпінгу через свою простоту, багатий екосистему та велику кількість відкритих бібліотек. Він є мовою з відкритим вихідним кодом, що сприяє швидкому розвитку та спільнотному підтримці. Python використовуватиметься для отримання даних з веб-сайтів та їх подальшої обробки.

Python має ряд переваг, що робить його ідеальним вибором для веб скрейпінгу. Його простий синтаксис дозволяє швидко розробляти скрипти, а багатий вибір бібліотек, таких як **requests** [27], **BeautifulSoup** [26] і **Scrapy**, спрощує отримання та обробку даних зі сторінок веб-сайтів. Бібліотека **requests** дозволяє здійснювати HTTP-запити для отримання сторінок, а **BeautifulSoup** допомагає парсити HTML-структури, щоб отримати потрібну інформацію. **Scrapy**, з іншого боку, забезпечує розширені можливості для створення веб-скраперів зі складнішою логікою.

SQL (Structured Query Language) є потужним інструментом для роботи з базами даних і підходить для зберігання та управління отриманими даними після веб скрейпінгу. Використання SQL бази даних для зберігання дозволяє зберігати великі

обсяги даних у структурованому форматі, що спрощує подальший аналіз і звітність.

SQL пропонує широкий набір можливостей для роботи з даними. Запити SQL дозволяють швидко і ефективно витягувати, фільтрувати та обробляти інформацію з бази даних. Крім того, SQL бази даних, такі як PostgreSQL, MySQL або SQLite, мають масштабність, стабільність та підтримку транзакцій, що робить їх ідеальними для зберігання важливих даних зі скрапінгу.

Обраний підхід до використання Python для скрапінгу та SQL для зберігання даних відображає оптимальне поєднання ефективності та доступності для вирішення завдань аналізу змін у коефіцієнтах ставок. Цей підхід забезпечує швидкий розвиток інструментів для збору даних з веб-сайтів та ефективне зберігання цих даних для подальшого аналізу і виявлення договірних дій.

2.3 Опис використаних інструментів для веб скрейпингу договірних спортивних матчів

PyCharm [18] є інтегрованим середовищем розробки (IDE) для Python, розробленою компанією JetBrains. Це потужний інструмент, який надає розширені можливості для написання, тестування і налагодження Python-коду. Основні риси PyCharm включають:

– редактор коду з підсвічуванням синтаксису: PyCharm надає зручний редактор коду з автоматичним підсвічуванням синтаксису Python, що спрощує написання коду і виявлення помилок;

– автоматичне доповнення коду: інтелектуальне автодоповнення PyCharm допомагає швидко завершувати код, показуючи доступні методи, змінні та атрибути;

– управління проектами: PyCharm дозволяє легко створювати, керувати та організовувати проекти Python;

– інтегрована підтримка віртуальних середовищ: PyCharm спрощує створення та використання віртуальних середовищ для ізоляції залежностей проекту;

– вбудований дебагер: інтегрований дебагер PyCharm дозволяє відстежувати виконання програми крок за кроком для виявлення та усунення помилок;

– підтримка версійного контролю: PyCharm інтегрується з системами контролю версій, такими як Git, що дозволяє ефективно керувати кодом у спільній роботі.

Бібліотека requests [17] є однією з найпопулярніших бібліотек для здійснення HTTP-запитів у Python. Вона дозволяє легко взаємодіяти з веб-сайтами, отримувати сторінки, відправляти дані і отримувати відповіді. Деякі особливості та функціональність requests включають:

– GET і POST запити: requests дозволяє виконувати різноманітні HTTP-запити, такі як GET і POST, для отримання та відправлення даних;

– параметри запитів: можливість передавати параметри запиту для налаштування отримання відповідей від сервера;

– обробка cookies: requests підтримує роботу з cookies, що дозволяє зберігати та використовувати інформацію про сесію;

– обробка заголовків: можливість додавати та налаштовувати HTTP-заголовки для взаємодії з сервером;

– обробка відповідей: requests дозволяє отримувати та обробляти відповіді сервера, що може бути HTML-сторінкою або JSON-даними;

– сесії та перенаправлення: requests підтримує управління сесіями і перенаправленнями, що дозволяє зберігати стан між запитами.

Бібліотека requests є потужним інструментом для веб скрейпінгу, оскільки спрощує взаємодію з веб-сайтами та дозволяє ефективно отримувати та оброблювати дані. Враховуючи особливості кожного сайту, важливо правильно налаштовувати параметри запитів, включаючи cookies, заголовки та формат даних, щоб отримати необхідну інформацію з сайту для подальшого аналізу.

SQLAlchemy [16] - це бібліотека для роботи з базами даних у Python, яка надає

зручний і гнучкий інтерфейс для взаємодії з реляційними базами даних. Вона дозволяє створювати, використовувати і керувати SQL-запитами, працювати з об'єктно-реляційним відображенням даних (ORM) та здійснювати транзакційну обробку даних. Нижче розглянемо детальніше функції і особливості SQLAlchemy:

– ORM (об'єктно-реляційне відображення): однією з ключових функцій SQLAlchemy є ORM, яке дозволяє використовувати об'єктно-орієнтовану парадигму для роботи з базою даних. За допомогою ORM можна використовувати класи Python для представлення таблиць у базі даних, атрибути класів - це стовпці таблиць, а об'єкти цих класів - це рядки даних у таблицях. ORM спрощує взаємодію з базою даних і дозволяє працювати з даними на більш високому рівні абстракції;

– сесії і транзакції: SQLAlchemy надає механізми для роботи з транзакціями в базі даних. Вона дозволяє створювати сесії, які представляють собою контексти роботи з базою даних, і здійснювати в них транзакційну обробку даних. Сесії SQLAlchemy дозволяють здійснювати операції збереження, оновлення та видалення даних в базі даних з безпекою транзакцій;

– SQL-вирази і запити: SQLAlchemy дозволяє конструювати складні SQL-запити за допомогою об'єктів Python. Вона надає API для побудови SQL-виразів, умов, об'єднань та фільтрів даних. Це дозволяє здійснювати потужні операції вибірки та маніпуляції даними у базі даних;

– підтримка різних баз даних: SQLAlchemy підтримує різні системи управління базами даних (СУБД), такі як PostgreSQL, MySQL, SQLite, Oracle та інші. Вона абстрагує деталі взаємодії з конкретною СУБД і дозволяє писати універсальний код для роботи з різними базами даних;

– міграції баз даних: SQLAlchemy має вбудовану підтримку міграцій баз даних за допомогою інструменту Alembic. Це дозволяє автоматизувати процес зміни структури бази даних і керувати версіонуванням схеми;

– інтеграція з іншими інструментами: SQLAlchemy може легко інтегруватися

з іншими бібліотеками і фреймворками Python для створення повноцінних веб-додатків з доступом до баз даних.

Загалом, SQLAlchemy [25] є потужним інструментом для роботи з базами даних у Python, який дозволяє створювати ефективні та надійні програми для роботи з даними. Вона комбінує в собі зручний ORM, механізми транзакцій, підтримку різних СУБД і багато інших функцій, що роблять розробку базоданихних додатків в Python простіше і ефективніше.

2.4 Огляд сучасних сервісів та API для отримання спортивних даних

Отримання спортивних даних через сучасні сервіси та API є ключовою складовою для багатьох аспектів аналізу в спортивній аналітиці, зокрема для виявлення договірних матчів у ставках. На сьогоднішній день існує багато сервісів, які надають дані про спортивні події та ставки, проте доступ до відкритих API з документацією обмежений і, в більшості випадків, платний.

Сервіси для ставок, такі як Betfair, William Hill, 1xBet, Pinnacle і багато інших, надають доступ до даних про спортивні події, коефіцієнти та результати матчів. Однак, для отримання доступу до їх API [20] зазвичай потрібно мати комерційну угоду і платити відповідний обсяг коштів за використання сервісу. Це ставить певні обмеження для дослідників та аналітиків, які прагнуть використовувати ці дані для власних досліджень та аналізу.

Щодо відкритих API з документацією, їх кількість досить обмежена. Деякі з них, які надають дані про спортивні події, мають обмежений безкоштовний тариф або вимагають підписки для доступу до повного функціоналу. Це може ускладнювати роботу з даними для аналітичних цілей, особливо у випадку великих обсягів даних або потреби в регулярних оновленнях.

Один із підходів до отримання спортивних даних - використання спеціалізованих платних сервісів, таких як Thunderpick, який надає доступ до

широкого спектру спортивних подій та ставок з можливістю отримання даних через їх API [15]. Thunderpick пропонує зручний і добре задокументований API, що дозволяє отримувати дані про різні види спорту, коефіцієнти ставок та іншу важливу інформацію для подальшого аналізу і обробки.

Отже, для аналізу договірних матчів та спортивної активності планується використовувати сервіс Thunderpick, який надає доступ до необхідних спортивних даних через їх зручне та надійне API.

Висновок: Для аналізу договірних матчів та спортивних подій в контексті ставок і спортивної аналітики, вебскрейпінг виступає як ключовий інструмент для збору та аналізу даних з веб-сайтів букмекерів та інших джерел [3]. Розглянуті методи, технології та інструменти вебскрейпінгу дозволяють ефективно виявляти можливі договірні дії через аналіз змін у коефіцієнтах ставок, тексту ставок, а також використання методів машинного навчання для розпізнавання аномалій.

PyCharm, як інтегроване середовище розробки (IDE), забезпечує зручність у написанні, тестуванні і налагодженні Python-коду, що дозволяє ефективно розробляти скрипти для вебскрейпінгу. Бібліотека requests дозволяє легко здійснювати HTTP-запити для отримання та обробки даних зі сторінок веб-сайтів, враховуючи специфіку кожного сайту, таку як обробка кукісів, заголовків і форматів даних. SQLAlchemy стає потужним інструментом для роботи з базами даних у Python, спрощуючи збереження та управління отриманими даними.

У контексті отримання спортивних даних через сервіси та API, є обмеження щодо доступності відкритих API з документацією, більшість з яких є платними. Спеціалізовані платні сервіси, такі як Thunderpick, можуть стати важливим джерелом даних для аналізу спортивних подій і ставок. Thunderpick надає зручний та добре задокументований API для отримання даної інформації, що дозволяє ефективно використовувати її для аналітичних цілей.

У підсумку, для аналізу договірних матчів та спортивної активності на сервісі

Thunderpick буде використовуватися комбінація вебскрейпінгу з використанням PyCharm, бібліотеки requests для отримання даних, SQLAlchemy для збереження і управління ними, а також API самого сервісу Thunderpick для отримання актуальних даних про спортивні події та ставки.

Висновки до розділу 2

Цей розділ присвячено огляду методів вебскрейпінгу та їх ефективності у виявленні договірних матчів. Було розглянуто різні технології, що використовуються для реалізації вебскрейпінгу, та обрано найбільш відповідні для поставленої задачі. Також описано інструменти, які використовуються для вебскрейпінгу договірних спортивних матчів, включаючи їх функціональні можливості та переваги.

Серед технологій, які розглядаються, можна виділити використання протоколів HTTP для взаємодії з веб-серверами, а також бібліотек та фреймворків, що дозволяють здійснювати HTTP-запити, парсити HTML-структуру веб-сторінок та видобувати необхідні дані. Огляд сучасних сервісів та API для отримання спортивних даних показав, що існують різні рішення для автоматизованого збору інформації про ставки та результати спортивних подій, але кожне з них має свої особливості та обмеження.

Вибір конкретних технологій та інструментів базувався на їх здатності забезпечити регулярний збір даних, модульність системи, уніфікацію отриманих даних та зручну візуалізацію для аналізу. Це дозволить створити ефективну систему для виявлення договірних спортивних матчів на основі даних, отриманих від вебскрейпінгу.

3 РЕАЛІЗАЦІЯ ОБРАНИХ ТЕХНОЛОГІЙ ТА СЕРВІСІВ ДЛЯ ПОСТАВЛЕНОЇ ЗАДАЧІ СТВОРЕННЯ ВЕБСКРЕЙПЕРА

3.1 Реалізація технічної частини вебскрейперу

Одним з ефективних підходів до реалізації вебскрейперу є використання принципу ETL (Extract, Transform, Load). Цей підхід дозволяє організувати процес збирання, обробки та зберігання даних у структурованій та логічній послідовності, що полегшує його реалізацію та подальше використання. У даній роботі розглянуто основні етапи створення вебскрейперу за принципом ETL, а також його переваги та практичне значення.

Для реалізації ETL проекту був обраний Python з кількох важливих причин:

– багатий набір бібліотек: Python має широкий спектр бібліотек і фреймворків, таких як BeautifulSoup, Scrapy, Pandas та SQLAlchemy, які значно спрощують процес витягнення, трансформації та завантаження даних. Це дозволяє швидко розробляти функціональність без необхідності писати все з нуля;

– легкість у навчанні та використанні: Python відомий своєю простотою та читабельністю, що робить його ідеальним вибором для розробників різного рівня. Це особливо важливо для підтримки та розвитку проекту у майбутньому;

– широка підтримка спільноти: велика та активна спільнота Python забезпечує доступ до багатьох ресурсів, документації та прикладів коду. Це полегшує вирішення проблем та обмін досвідом між розробниками;

– гнучкість та масштабованість: Python дозволяє легко інтегруватися з іншими мовами та системами, що забезпечує гнучкість у розробці ETL проекту. Крім того, існують інструменти та бібліотеки, які підтримують паралельні обчислення та роботу з великими обсягами даних;

– крос-платформеність: Python працює на різних операційних системах, що робить його універсальним інструментом для розробки та розгортання ETL рішень на

будь-якій інфраструктурі;

– інтеграція з базами даних: Python має потужні інструменти для роботи з різними базами даних, такими як SQLAlchemy для ORM (Object-Relational Mapping) та бібліотеки для взаємодії з SQL і NoSQL базами даних, що робить його зручним для реалізації складних ETL процесів.

З огляду на всі ці переваги, Python є оптимальним вибором для реалізації ETL проекту, забезпечуючи ефективність, надійність та простоту в розробці та підтримці.

3.1.1 Реалізація стукрутри проекту

Оскільки сервіс буде взаємодіяти з базами даних дуже часто, важливо застосувати систематичний підхід та найкращі практики, які використовуються при створенні backend застосунків. Це забезпечить ефективність роботи, легкість у підтримці та масштабуванні проекту.

У кореневій директорії проекту (рис. 3.1) розташовані дві основні папки: env та app.

– папка env: тут зберігаються всі залежності та бібліотеки, необхідні для роботи програми. Це дозволяє чітко відокремити середовище виконання проекту від його логіки, що полегшує встановлення та оновлення залежностей;

– папка app: у цій папці зберігається вся логіка проекту. Вона включає модулі для взаємодії з базами даних, обробки даних, а також інші компоненти, необхідні для функціонування вебскрейперу. Така організація структури дозволяє легко орієнтуватися у проекті, а також розширювати його функціональність.

Цей підхід не лише сприяє систематизації та організації коду, але й забезпечує легкість у майбутніх поліпшеннях та масштабуванні проекту. Наприклад, для контролю версій бази даних можна легко підключити інструмент Alembic. Alembic дозволяє відстежувати зміни в схемі бази даних, автоматизувати процеси міграції та забезпечує зручний спосіб керування змінами у структурі даних.

Таким чином, застосування кращих практик для організації структури проекту гарантує надійність, гнучкість та простоту в підтримці вебскрейперу, що є критично важливим для його успішної експлуатації та розвитку.

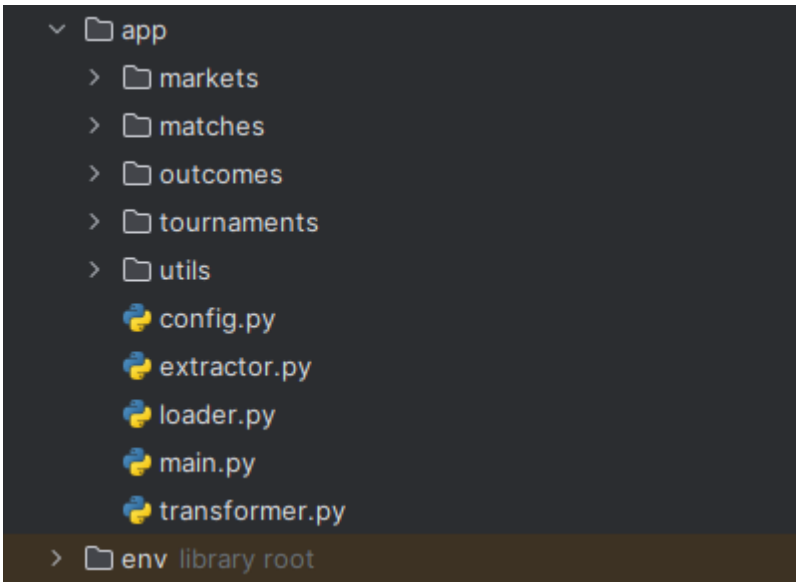

Рисунок 3.1 – Перелік директорій додатку

3.1.1.1 Папка venv

У папці «venv» в Python-проекті зберігається віртуальне середовище, яке містить усі залежності та бібліотеки, необхідні для роботи цього проекту. Це ізолює їх від глобальних налаштувань системи, забезпечуючи незалежність і стабільність середовища розробки.

3.1.1.2 Структура папки app

У папці app розташовані підпапки, що відповідають основним компонентам програми. Кожна з цих підпапок містить схеми для SQLAlchemy та файли Data Access Object (DAO), які відповідають таблицям у базі даних. Папки tournaments, matches, markets та outcomes містять схеми та DAO для відповідних таблиць у базі даних.

Основні компоненти папки app:

– папка dao: файл base.py. В даному файлі зберігається база, від якої успадковуються інші DAO. Якщо потрібно, інші DAO можуть успадковувати цю базу, в іншому випадку використовуються методи, задані за замовчуванням. Це робиться для забезпечення однорідного і гнучкого підходу до роботи з базою даних;

– папки tournaments, matches, markets, outcomes: схеми (schemas): визначають структуру бази даних, включаючи таблиці та їхні поля, а також задають взаємозв'язки між таблицями; DAO (Data Access Object): шаблон проектування, який абстрагує доступ до даних, надаючи стандартні методи для створення, читання, оновлення та видалення записів;

– папка utils: містить додаткові файли, які допомагають у роботі програми. Ці файли можуть включати різні утиліти, хелпери та інші допоміжні модулі;

– файл config.py: у цьому файлі витягуються ключі з .env файлу і перетворюються у посилання на базу даних, готове до використання. Це класичне застосування такої методики, яке також дозволяє при майбутній інтеграції або розширенні програми додати інші значення;

– файл database.py: у цьому файлі налаштовується асинхронний двигун бази даних і фабрика сесій для роботи з базою даних за допомогою SQLAlchemy. Також визначається базовий клас для всіх моделей, що використовуються в програмі;

– файл extractor.py: в ETL-процесі модуль extractor відповідає за отримання повного об'єму даних на момент запуску функції. Повертає дані у вигляді словника;

– файл transformer.py: в ETL-процесі модуль transformer приймає словник сирих даних, очищає від зайвих даних, та перетворює їх у універсальний формат для завантаження в базу даних;

– файл loader.py: в ETL-процесі модуль loader відповідає за завантаження даних у базу даних за певною логікою;

– файл main.py: в файлі main.py всі модулі - extractor, transformer, loader - є частинами централізованого процесу витягування, трансформації та завантаження

даних (ETL). Після запуску програми користувач визначає, як часто збирати дані з веб-сайту.

Програма викликає модулі extractor, transformer та loader, передаючи дані по порядку відповідним частинам ETL-процесу.

Після завантаження даних в базу даних вони можуть бути проаналізовані.

Такий підхід до організації структури проекту забезпечує надійність, гнучкість та простоту в підтримці ETL-системи, що є критично важливим для її успішної експлуатації та розвитку.

3.1.2 Реалізація схем БД для визначення структури даних

База даних - це систематизована колекція даних, яка зберігається та організовується в електронному вигляді на комп'ютері або сервері. Схеми баз даних визначають структуру цих даних, включаючи таблиці та їх взаємозв'язки. Вони визначають структуру кожної таблиці, включаючи поля та типи даних, а також зв'язки між таблицями, надаючи каркас для організації та доступу до даних.

Опис таблиць наведено нижче:

а) таблиця "Турніри" (Tournaments):
- id (ціле число, первинний ключ): унікальний ідентифікатор турніру;
- name (текстове поле, обов'язкове): назва турніру;
- sport (текстове поле, обов'язкове): вид спорту турніру;
- status (текстове поле, обов'язкове): статус турніру;
- bookmaker (текстове поле, обов'язкове): назва букмекерської контори, яка організує турнір;

б) таблиця "Матчі" (Matches):
- id (ціле число, первинний ключ): унікальний ідентифікатор матчу;
- tournament_id (ціле число, зовнішній ключ): зв'язок з таблицею "Турніри";
- home (текстове поле, обов'язкове): назва команди-господаря;

- away (текстове поле, обов'язкове): назва гостьової команди;
- time_start (текстове поле, обов'язкове): час початку матчу;
- status (текстове поле, обов'язкове): статус матчу;

в) таблиця "Маркети" (Markets):

- id (ціле число, первинний ключ): унікальний ідентифікатор маркету;
- match_id (ціле число, обов'язкове): зв'язок з таблицею "Матчі";
- name (текстове поле, обов'язкове): назва маркету;

г) таблиця "Результати" (Outcomes):

- id (ціле число, первинний ключ): унікальний ідентифікатор результату;
- market_id (ціле число, обов'язкове): зв'язок з таблицею "Маркети";
- name (текстове поле, обов'язкове): назва результату;
- odds (дробове число, обов'язкове): коефіцієнт;
- time_stamp (дата та час, обов'язкове): момент часу запису результату;
- status (текстове поле, обов'язкове): статус результату.

Ці схеми визначають структуру бази даних для зберігання інформації про турніри, матчі, маркети та результати. Кожна таблиця має свій унікальний набір полів, які визначають дані, що зберігаються в цих таблицях.

3.2 Реалізація технічної частини вебскрейперу

Користувач буде запускати лише головний файл (рис. 3.2), і подальші кроки розберемо детальніше, що робить кожна функція.

```python
import time
import asyncio
from extractor import get_all_matches_info as extract
from transformer import transform
from loader import load


async def scrap(sleep_time):  1 usage

    while True:
        time_start = time.time()
        try:
            result = await extract()
            # print(len(result))
            transformed_data = await transform(result)
            # print(len(transformed_data))
            # print(transformed_data)
            load(transformed_data)
        except Exception as e:
            print(e)
        time_end = time.time()
        print(time_end - time_start)

        await asyncio.sleep(sleep_time)


sleep_time_q = input("Enter sleep time: ")

loop = asyncio.get_event_loop()
loop.run_until_complete(scrap(sleep_time_q))
```

Рисунок 3.2 – Код файлу main.py

Цей код виконує скрапінг даних з веб-сайту періодично з вказаним інтервалом.

Опис коду:

а) імпорт бібліотек та функцій:
- time та asyncio - для керування часом та асинхронних операцій відповідно;
- extract - функція для отримання інформації про всі матчі з джерела даних;
- transform - функція для перетворення отриманих даних;
- load - функція для завантаження трансформованих даних у базу даних;

б) асинхронна функція scrap:
- цикл while True виконується постійно;
- визначається час початку операції скрапінгу;
- виконується спроба отримання даних через функцію extract();

– отримані дані піддаються трансформації за допомогою функції transform();

– трансформовані дані завантажуються в базу даних за допомогою функції load();

– у разі виникнення помилки виводиться відповідне повідомлення;

– визначається час завершення операції скрапінгу;

– виводиться час виконання операції;

– засинає на вказаний користувачем час;

в) введення користувачем інтервалу часу:

– користувач вводить інтервал часу між кожним запуском скрапера;

г) запуск асинхронної функції scrap:

– створюється асинхронний цикл;

– викликається функція scrap з введеним користувачем інтервалом часу.

Цей код реалізує автоматизований процес веб-скрапінгу та завантаження даних у базу даних з заданим інтервалом часу. Користувач запускає лише головний файл програми, а детальна логіка кожної функції буде розглянута пізніше.

3.2.1 Файл extractor.py

Розглянемо файл transformer.py (рис. 3.3).

```python
import requests
import asyncio
import aiohttp
from app.utils.settings import url_api, json_allMatch, json_match, headers
import copy


# scrap line, no detail info
def get_all_matches():  1 usage
    matches = []
    json = copy.deepcopy(json_allMatch)
    page_info = {'hasNextPage': True}

    while page_info['hasNextPage']:
        response = requests.post(url=url_api, json=json, headers=headers).json()
        edges = response.get('data', {}).get('allMatch', {}).get('edges', [])
        matches.extend(edges)
        page_info = response.get('data', {}).get('allMatch', {}).get('pageInfo', {})
        if page_info.get('hasNextPage'):
            json['variables']['after'] = page_info['endCursor']

    return matches


# get detail info for specific match
async def get_match_info(session, bookmaker_id):  1 usage
    json = json_match.copy()
    json['variables']['matchId'] = bookmaker_id

    async with session.post(url=url_api, json=json, headers=headers) as resp:
        return await resp.json()


async def get_all_matches_info():  1 usage
    id_list = [data.get("node", {}).get("id", "") for data in get_all_matches()]
    async with aiohttp.ClientSession() as session:
        tasks = [get_match_info(session, bookmaker_id) for bookmaker_id in id_list]
        return await asyncio.gather(*tasks)
```

Рисунок 3.3 – Код файлу extractor.py

Детальний опис коду:

а) імпорт бібліотек та налаштувань:

– requests, asyncio, aiohttp - для роботи з HTTP-запитами та асинхронними операціями;

– url_api, json_allMatch, json_match, headers - налаштування для API запитів, такі як URL-адреса, JSON-структури, та заголовки запиту;

б) функція get_all_matches:

– отримує неповні дані про матчі через виклик API;

– повертає список з отриманими матчами;

в) асинхронна функція get_match_info:

– отримує детальну інформацію про конкретний матч за його ідентифікатором через асинхронний запит;

– використовується бібліотека aiohttp для асинхронних HTTP-запитів;

– повертає результат у форматі JSON;

г) асинхронна функція get_all_matches_info:

– отримує списки ідентифікаторів матчів, використовуючи функцію get_all_matches;

– створює асинхронний сеанс aiohttp.ClientSession() для виконання багато асинхронних запитів;

– для кожного ідентифікатора матчу створюється задача асинхронного запиту до get_match_info;

– використовується asyncio.gather для паралельного виконання всіх запитів та отримання результатів;

– повертає список детальної інформації про кожен матч.

Цей код потрібен для отримання повних даних про матчі, оскільки серверна частина веб-сайту обмежує обсяг отримуваної інформації. Функція get_all_matches отримує неповні дані, а функція get_match_info викликається для кожного матчу окремо, отримуючи повні дані для кожного з них.

3.2.2 Файл transformer.py

Розглянемо файл transformer.py (рис. 3.4).

```python
from app.utils.MatchBetBoom import MatchBetBoom


def transform_to_obj(match_raw):  1 usage
    match_raw = match_raw['data']['match']
    return MatchBetBoom(
            match_id=match_raw['id'],
            tournament_name=match_raw['tournament']['name'],
            sport_name=match_raw['tournament']['sport']['name'],
            state=match_raw['state'],
            time_start=match_raw['datePlannedStart'],
            home=match_raw['teams'][0]['team']['name'],
            away=match_raw['teams'][1]['team']['name'],
            markets=match_raw['marketGroups']+match_raw['mainMarketGroups']
        ).to_dict()


async def transform(data):  2 usages
    return [transform_to_obj(match_raw) for match_raw in data]
```

Рисунок 3.4 – Код файлу transformer.py

Цей код виконує трансформацію сирих даних про матчі у внутрішнє об'єктне представлення для подальшої обробки та аналізу:

а) функція transform_to_obj:

- отримує сиру інформацію про конкретний матч;
- виконує трансформацію цих сирі дані у внутрішнє об'єктне представлення за допомогою класу MatchBetBoom;
- повертає словник, що містить дані про матч у вигляді об'єкта MatchBetBoom;

б) асинхронна функція transform:

- отримує дані у формі списку сирих даних про матчі;
- застосовує трансформацію до кожного матчу вхідних даних, викликаючи функцію transform_to_obj;
- повертає список об'єктів MatchBetBoom, що містять трансформовані дані про кожен матч.

3.2.3 Файл loader.py

Цей код (ДОДАТОК А) виконує наступні дії:

а) створює список унікальних турнірів із наданих даних, перевіряючи їх унікальність за назвою, спортом і книгарнею. Якщо такий турнір відсутній, він додається до бази даних через DAO (Data Access Object) для турнірів;

б) знаходить id турнірів, які вже є у базі даних;

в) оновлює дані про матчі, замінюючи назву турніру на його id;

г) додає дані про матчі до бази даних через dao для матчів;

д) знаходить існуючі матчі за допомогою їхніх основних характеристик (назва команди та id турніру);

е) додає дані про ринки (markets) до бази даних через dao для ринків, приєднуючи їх до відповідних матчів;

ж) знаходить існуючі ринки за допомогою id матчу та назви ринку;

з) запускає асинхронні операції обробки результатів (outcomes) для кожного матчу;

и) додає дані про результати (outcomes) до бази даних через dao для результатів, приєднуючи їх до відповідних ринків.

Цей код використовується для обробки та завантаження даних про турніри, матчі, ринки та результати з наданих даних у базу даних. Кожен крок обробки даних відповідає певному сегменту процесу обробки даних та використовує DAO для доступу до бази даних.

Висновки до розділу 3

У процесі аналізу та обробки наданих даних про матчі з веб-сайту за допомогою API було створено програмне забезпечення, яке автоматизує процес завантаження, трансформації та завантаження цих даних до бази даних (рис. 3.5).

	id [PK] integer	tournament_id integer	home text	away text	time_start text	status text
1	1	581	Virtus.pro	9 Pandas	2024-06-14T07:30:00Z	STARTED
2	2	582	FearX	Gen.G	2024-06-14T08:05:00Z	STARTED
3	3	583	3DMAX	Passion UA	2024-06-14T08:05:00Z	STARTED
4	4	584	Royal Never Give Up	LGD Gaming	2024-06-14T09:11:00Z	STARTED
5	5	585	Detroit Pistons (Steel)	Atlanta Hawks (Brent)	2024-06-14T09:12:00Z	STARTED
6	6	586	Manchester Utd (Mia)	Chelsea (July)	2024-06-14T09:12:00Z	STARTED
7	7	586	Everton (Kelly)	Manchester City (Tracy)	2024-06-14T09:14:00Z	STARTED
8	8	587	Torino (Hurricane)	Bergamo Calcio (PinkElephant)	2024-06-14T09:16:00Z	STARTED
9	9	587	Inter (Niskanen15)	Fiorentina (Junior)	2024-06-14T09:18:00Z	STARTED
10	10	588	Aurora Gaming	Team Falcons	2024-06-14T09:25:00Z	NOT_STARTED
11	11	589	TOR	Infinite Gaming	2024-06-14T09:25:00Z	NOT_STARTED
12	12	586	Arsenal (Brenda)	Manchester Utd (Mia)	2024-06-14T09:26:00Z	NOT_STARTED
13	13	586	Everton (Kelly)	Manchester City (Tracy)	2024-06-14T09:28:00Z	NOT_STARTED

Рисунок 3.5 – Результат у базі даних після запуску програми

Під час першого етапу, дані були отримані через виклик API та оброблені для подальшого використання. Враховуючи особливості даних, які надходять з серверної частини веб-сайту, була виконана обробка та трансформація сирого потоку даних у структуровану форму. Це дозволило створити унікальний список турнірів та здійснити додавання цих турнірів до бази даних разом з даними про матчі, ринки та результати.

Кожен етап обробки даних був чітко розподілений на відповідні функції, що спростило розуміння та підтримку програми. Використання асинхронності та паралельного виконання задач підвищило ефективність обробки даних.

Загальний висновок полягає в тому, що створена програма дозволяє ефективно та автоматично збирати, трансформувати та завантажувати дані про матчі в базу даних, що полегшує подальший аналіз та використання цих даних у спортивних аналітичних додатках або ігрових платформах.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА КЕРІВНИЦТВО КОРИСТУВАЧА

У цьому розділі буде детально описано процес використання програми, розробленої для пошуку та попередження підозрілої активності у спортивних матчах. Розглянуто, як користувачі можуть ефективно використовувати різні функції програми для збору і аналізу даних.

4.1 Деталі процесу збору даних

Запуск програми:

– запустіть файл main.py;

– вкажіть інтервал часу в секундах (рис. 4.1), через який програма буде збирати дані.

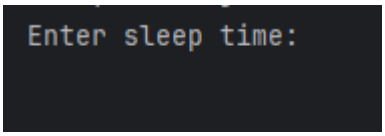

Рисунок 4.1 – Вказування часу

Робота програми:

– після налаштування інтервалу часу програму можна залишити працювати в автономному режимі;

– при кожному зборі даних програма буде фіксувати результат, включаючи час виконання та можливі помилки на будь-якому етапі (рис. 4.2).

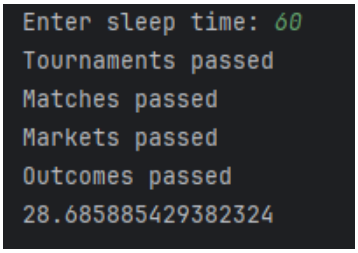

Рисунок 4.2 – Демонстрація результат збору даних, включаючи час виконання та виниклі помилки

4.2 Візуалізація результатів

У цьому пункті описується процес візуалізація результатів у вигляді графіків (рис. 4.3).

```python
import asyncio
from app.outcomes.dao import OutcomeDAO
from app.tournaments.dao import TournamentsDAO
from app.matches.dao import MatchDAO
from app.markets.dao import MarketDAO


async def start():  1 usage
    tournaments = await TournamentsDAO.find_all()
    for tournament in tournaments:
        print(f"[{tournament['id']}] | {tournament['name']} | {tournament['sport']}")
    select_tournament_id = int(input("Select tournament: "))
    matches = await MatchDAO.find_all(tournament_id=select_tournament_id)
    for match in matches:
        print(f"[{match['id']}] | {match['home']} vs {match['away']} | {match['time_start']}")
    select_match_id = int(input("Select match: "))
    markets = await MarketDAO.find_all(match_id=select_match_id)
    for market in markets:
        print(f"[{market['id']}] | {market['name']}")
    select_market_id = int(input("Select market: "))
    outcomes = await OutcomeDAO.find_all(market_id=select_market_id)
    visualize(outcomes)

asyncio.run(start())
```

Рисунок 4.3 – Код програми для візуалізації

Спершу програма запитує у користувача, який турнір він хоче аналізувати. Користувач обирає турнір зі списку, який виводиться на екран (рис. 4.4).

```
[952] | GLL 2024 Summer | League of Legends
[953] | Ultraliga 2024 Summer | League of Legends
[954] | VCT 2024: EMEA Stage 2 | Valorant
[955] | LLA 2024 Closing | League of Legends
[956] | 2024 Global StarCraft II League Season 2: Code S | Starcraft 2
[957] | RLCS 2024 - Major 2: London | Rocket League
[958] | VCT 2024: Americas Stage 2 | Valorant
Select tournament:
```

Рисунок 4.4 – Запит турніру

Після вибору турніру програма запитує у користувача, який матч він хоче аналізувати з обраного турніру. Користувач обирає матч зі списку, який виводиться на екран (рис. 4.5).

```
[956] | 2024 Global StarCraft II League Season 2: Code S | Starcraft 2
[957] | RLCS 2024 - Major 2: London | Rocket League
[958] | VCT 2024: Americas Stage 2 | Valorant
Select tournament: 700
[394] | Team Falcons vs M80 | 2024-06-14T18:00:00Z
[395] | Team Vitality vs PWR | 2024-06-14T18:00:00Z
[396] | Gen.G Mobil1 Racing vs GRIDSERVE Resolve | 2024-06-14T18:00:00Z
[397] | Gentle Mates Alpine vs Gaimin Gladiators | 2024-06-14T18:00:00Z
[398] | FURIA Esports vs jobless | 2024-06-14T18:00:00Z
[401] | Team BDS vs Twisted Minds | 2024-06-14T18:45:00Z
[402] | Team Secret vs Magnifico | 2024-06-14T18:45:00Z
[403] | Karmine Corp vs Chiefs ESC | 2024-06-14T18:45:00Z
Select match:
```

Рисунок 4.5 – Запит матчу

Після вибору матчу програма запитує у користувача, який маркет (ринок ставок) він хоче аналізувати. Користувач обирає маркет зі списку, який виводиться на екран (рис. 4.6).

```
[403] | Karmine Corp vs Chiefs ESC | 2024-06-14T18:45:00Z
Select match: 401
[1900] | Winner
[1901] | Winner Map 1
[1902] | Winner Map 2
[1903] | Winner Map 3
[1904] | Winner Map 4
[1905] | Winner Map 5
[1906] | Winner
Select market:
```

Рисунок 4.6 – Запит маркету

Програма отримує результати (outcomes) для обраного маркету та виводить їх на екран. Після цього вона будує графіки на основі отриманих даних (рис. 4.7).

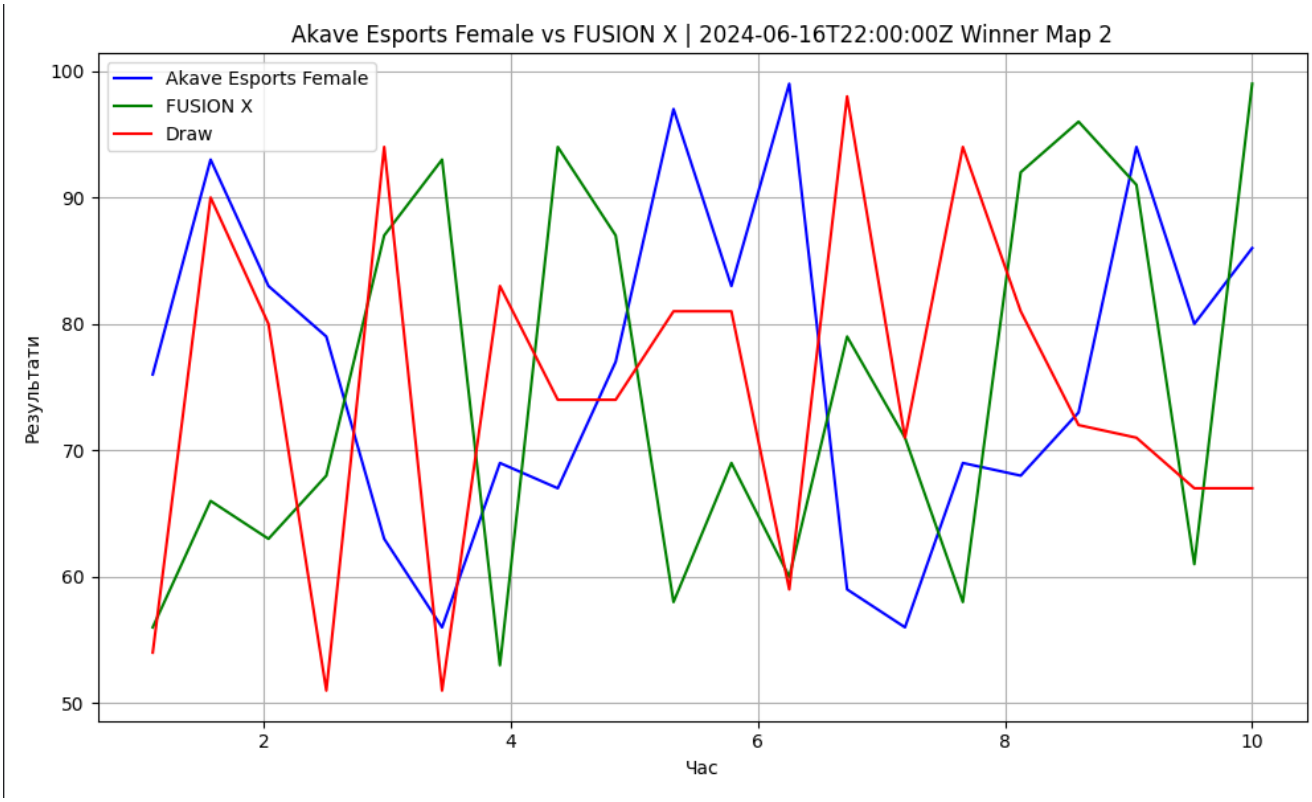

Рисунок 4.7 – Приклад візуалізації

Висновки до розділу 4

У даному розділі було розглянуто процес запуску розробленої програми для пошуку договірних спортивних матчів. Описано необхідні етапи для успішного старту програми.

Також розглянули візуалізацію результатів збору та аналізу даних про спортивні матчі. Після запуску програма збирає дані з визначених веб-джерел за допомогою технологій вебскрейпінгу. Отримані дані обробляються та аналізуються для виявлення підозрілих активностей, які можуть свідчити про договірні матчі. Результати цього аналізу відображаються у вигляді графіків та діаграм, що дозволяє користувачам швидко зрозуміти тенденції та аномалії у спортивних подіях.

Таким чином, даний розділ надав повний огляд процесу запуску та функціонування програми, а також способів візуалізації даних, що дозволяє ефективно виявляти договірні матчі.

ВИСНОВКИ

У процесі дослідження було розглянуто та розроблено автоматизовану систему для виявлення договірних спортивних матчів за допомогою вебскрейпінгу. Головною метою роботи було створення інструменту, який дозволяє ефективно збирати, аналізувати та візуалізувати дані з різних джерел для виявлення потенційно підозрілих активностей у спортивних змаганнях.

Було проведено огляд сучасного стану вебскрейпінгу та існуючих систем для пошуку договірних матчів. Виявлено, що більшість наявних рішень мають обмеження у вигляді застарілого коду та специфічних для певних видів спорту чи букмекерських сайтів інструментів. Визначено основні поняття та терміни, пов'язані з вебскрейпінгом та договірними матчами, що заклало основу для подальшої розробки системи.

Система була спроектована на основі модульного підходу, що забезпечило її гнучкість та масштабованість. Кожен модуль відповідає за окремий аспект збору та обробки даних. Реалізовано ETL-процес (Extract, Transform, Load), що дозволяє організувати збирання, обробку та зберігання даних у структурованій послідовності.

Використано мову програмування Python, завдяки її простоті та багатому набору бібліотек для вебскрейпінгу, таких як BeautifulSoup, Scrapy та SQLAlchemy. Для зберігання та управління даними використано SQL, що забезпечило швидкий доступ до інформації та ефективне управління базами даних.

Було реалізовано функціональні можливості веб-скрейпера, включаючи збирання даних з різних джерел, їх обробку та уніфікацію, а також візуалізацію результатів для зручного аналізу. Проведено тестування створеного застосунку, що підтвердило його ефективність у виявленні підозрілих змін у коефіцієнтах ставок та результатах матчів.

Розроблена система може бути використана спортивними організаціями, аналітиками та дослідниками для виявлення договірних матчів, що сприятиме

підвищенню чесності та справедливості у спортивних змаганнях. Використання автоматизованих інструментів дозволить значно зменшити витрати на людські ресурси та підвищити ефективність аналізу даних.

Таким чином, виконана робота є важливим внеском у сферу спортивної аналітики та забезпечення чесності спортивних змагань, відкриваючи нові можливості для виявлення договірних матчів за допомогою сучасних технологій вебскрейпінгу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Вебскрейпінг та аналіз даних з Python: вебсайт. URL: https://www.dataquest.io/blog/web-scraping-tutorial-python/ (дата звернення: 30.04.2024).

2. Вебскрейпінг для початківців: вебсайт. URL: https://realpython.com/web-scraping-with-python/ (дата звернення: 28.04.2024).

3. Вебскрейпінг з Scrapy: вебсайт. URL: https://scrapy.org/ (дата звернення: 09.05.2024).

4. GitHub - gingeleski/odds-portal-scraper: репозиторій. URL: https://github.com/gingeleski/odds-portal-scraper (дата звернення: 01.06.2024).

5. GitHub - Seb943/scrapeOP: репозиторій. URL: https://github.com/Seb943/scrapeOP (дата звернення: 03.06.2024).

6. GitHub - S1M0N38/soccerapi: репозиторій. URL: https://github.com/S1M0N38/soccerapi (дата звернення: 06.06.2024).

7. GitHub - cvidan/bet365-scraper: репозиторій. URL: https://github.com/cvidan/bet365-scraper (дата звернення: 10.06.2024).

8. Web Scraping Best Practices: вебсайт. URL: https://www.kdnuggets.com/2018/11/web-scraping-best-practices.html (дата звернення: 05.05.2024).

9. Beautiful Soup Documentation: вебсайт. URL: https://beautiful-soup-4.readthedocs.io/en/latest/ (дата звернення: 10.05.2024).

10. Data Science з Python: вебсайт. URL: https://jakevdp.github.io/PythonDataScienceHandbook/ (дата звернення: 07.05.2024).

11. Match-fixing: вебсайт. URL: https://www.interpol.int/en/Crimes/Corruption/Match-fixing (дата звернення: 25.05.2024).

12. Sports Betting Web Scraping Using Python: вебсайт. URL:

https://medium.com/analytics-vidhya/sports-betting-web-scraping-using-python-44faeff28d48 (дата звернення: 28.05.2024).

13. 5 Web Scraping Projects to Level Up Your Skills: вебсайт. URL: https://towardsdatascience.com/5-web-scraping-projects-to-level-up-your-skills-95c5dd486edb (дата звернення: 31.05.2024).

14. Machine Learning for Analyzing Betting Odds: вебсайт. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167923620302456 (дата звернення: 05.06.2024).

15. Thunderpick API Documentation: вебсайт. URL: https://thunderpick.com/api (дата звернення: 15.06.2024).

16. Основи SQLAlchemy: вебсайт. URL: https://docs.sqlalchemy.org/en/14/tutorial/ (дата звернення: 29.04.2024).

17. Requests: бібліотека HTTP для Python: вебсайт. URL: https://docs.python-requests.org/en/latest/ (дата звернення: 02.05.2024).

18. Керівництво користувача PyCharm: вебсайт. URL: https://www.jetbrains.com/pycharm/documentation/ (дата звернення: 03.05.2024).

19. Використання API для отримання спортивних даних: вебсайт. URL: https://www.sportsdata.io/developers/api-documentation (дата звернення: 04.05.2024).

20. Кращі практики роботи з API: вебсайт. URL: https://developers.google.com/api-design (дата звернення: 05.05.2024).

21. Інтеграція даних за допомогою ETL: вебсайт. URL: https://www.informatica.com/services-and-training/glossary-of-terms/etl-definition.html (дата звернення: 06.05.2024).

22. Програмування на Python: офіційна документація: вебсайт. URL: https://www.python.org/doc/ (дата звернення: 08.05.2024).

23. Основи ETL для обробки даних: вебсайт. URL: https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-etl/ (дата звернення: 10.05.2024).

24. Автоматизований збір даних за допомогою Python: вебсайт. URL: https://www.packtpub.com/product/automated-data-collection-with-python/9781789533329 (дата звернення: 11.05.2024).

25. Основи SQLAlchemy для роботи з базами даних: вебсайт. URL: https://auth0.com/blog/sqlalchemy-orm-tutorial-for-python-developers/ (дата звернення: 14.05.2024).

26. Збір даних з використанням Python та BeautifulSoup: вебсайт. URL: https://medium.com/python-pandemonium/web-scraping-with-python-beautiful-soup-cheat-sheet-6d198d1d3c5e (дата звернення: 15.05.2024).

27. Requests для початківців: вебсайт. URL: https://realpython.com/python-requests/ (дата звернення: 16.05.2024).

ДОДАТОК А

Лістинг коду

```
import json
import asyncio
from datetime import datetime
from markets.dao import MarketDAO
from outcomes.dao import OutcomeDAO
from tournaments.dao import TournamentsDAO
from matches.dao import MatchDAO

async def add_data(data, bookmaker):

    tournaments_unique_new = []
    for match in data:
        tournament = {"name": match['tournament']['name'], "sport":
match["sport"]['name'], "status": "SomeStatus", "bookmaker": "BetBoom"}
        if tournament not in tournaments_unique_new:
            tournaments_unique_new.append(tournament)
    if tournaments_unique_new:
        await TournamentsDAO.add(tournaments_unique_new)

    result = await
TournamentsDAO.find_by_name_and_bookmaker([tournament['name'] for tournament
in tournaments_unique_new], bookmaker)
    tournaments_unique = {data['name']: data['id'] for data in result}

    print("Tournaments passed")

    for match in data:
        match['tournament'] = {
            "name": match['tournament']['name'],
            "id": tournaments_unique[match['tournament']['name']]}

    matches_to_add = []
    for match in data:
        match_data = {
            "tournament_id": match['tournament']['id'],
            "home": match['home'],
            "away": match['away'],
            "time_start": match['time_start'],
            'status': match["state"]
        }
        matches_to_add.append(match_data)
    if matches_to_add:
        await MatchDAO.add(matches_to_add)
```

```
    matches_to_search = [{"home": match['home'], "away": match["away"],
"tournament_id": match['tournament']['id']} for
                         match in data]

    existed_matches = await MatchDAO.find_by_fields(matches_to_search)
    existed_matches_dict = {(existed_match['home'], existed_match['away'],
existed_match['tournament_id']): existed_match['id'] for existed_match in
existed_matches}

    for match in data:
        match['db_id'] = existed_matches_dict[(match['home'], match['away'],
match['tournament']['id'])]

    print("Matches passed")

    markets_to_add = []

    for match in data:
        db_id = match['db_id']
        for market in match['markets']:
            market_data = {
                "match_id": db_id,
                "name": market['name']
            }
            markets_to_add.append(market_data)
    if markets_to_add:
        await MarketDAO.add(markets_to_add)

    new_markets = await MarketDAO.find_by_fields(markets_to_add)
    new_markets_dict = {(data['match_id'], data['name']): data['id'] for data
in new_markets}
    for match in data:
        for market in match['markets']:
            market['db_id'] = new_markets_dict[(match['db_id'],
market['name'])]

    print("Markets passed")

    tasks = [process_outcomes(match) for match in data]
    await asyncio.gather(*tasks)

    outcomes_to_add = []

    for match in data:
        for market in match['markets']:
```

```
        for outcome in market['outcomes']:
            data = {
                "market_id": market['db_id'],
                "name": outcome['name'],
                "odds": outcome["odds"],
                "time_stamp": datetime.now(),
                "status": match["state"]
            }
            outcomes_to_add.append(data)
if outcomes_to_add:
    await OutcomeDAO.add(outcomes_to_add)

print("Outcomes passed")
```