

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ НАДАННЯ ТЕНІСНОЇ
СТАТИСТИКИ ОНЛАЙН

SOFTWARE FOR TENNIS STATISTICS ONLINE ACCESS

Виконав: студент(ка) 4 курсу, групи КНТ-219сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Нурієв В.В.

(прізвище та ініціали)

Керівник Степаненко О.О.

(прізвище та ініціали)

Рецензент Поляков М.О.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Нурієва Віталія Валентиновича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення надання тенісної статистики онлайн

Software for Tennis Statistics Online Access

керівник проєкту (роботи) Степаненко Олександр Олексійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Степаненко О.О., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання « 4 » квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

(підпис) Нурієв В.В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Степаненко О.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 90 с., 25 рис., 1 табл., 3 дод., 14 джерел.

ТЕНІС, СТАТИСТИКА, ІНДИВІДУАЛЬНІ ПОКАЗНИКИ ВИСТУПУ, ВЕБЗАСТОСУНОК, PYTHON, DJANGO.

Об'єкт роботи – процес відслідковування тенісної статистики. Предмет роботи – програмне забезпечення надання тенісної статистики онлайн. Мета роботи – розробити програмне забезпечення надання тенісної статистики онлайн для забезпечення можливостей індивідуального відслідковування.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мови Python, JavaScript, HTML, CSS, вебфреймворк Django, система керування базами даних MySQL.

Результати. Розробку програмного забезпечення надання тенісної статистики онлайн виконано за допомогою вебфреймворку Django з визначенням структури даних, вимог до програми, розробленням структури і алгоритму роботи програми на основі цих вимог, реалізацією відповідних одиниць. Програма дозволяє переглядати результати змагань, показники виступу тенісистів у поєдинках, за весь турнір, всі сезони турніру, весь сезон, всю кар'єру, переглядати порівняні показники виступу пар тенісистів, створювати власні списки тенісистів, турнірів, пар тенісистів для доступу.

Висновки. Розроблене програмне забезпечення дозволяє підвищити зручність роботи зі статистикою виступу тенісистів на основі індивідуального відслідковування, що дозволяє привертати увагу широкого числа користувачів.

Галузь використання. Програмне забезпечення може використовуватися в якості основи для роботи тенісного порталу.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 90 pages, 25 figures, 1 table, 3 appendixes, 14 sources.

TENNIS, STATISTICS, INDIVIDUAL PERFORMANCE INDICATORS, WEB APPLICATION, PYTHON, DJANGO.

The object of the work is the process of tracking tennis statistics. Subject of the work is software for providing tennis statistics online. The aim of the work is to develop software for providing tennis statistics online to ensure individual tracking capabilities.

Materials, methods and technical means: object-oriented programming, Python languages, JavaScript, HTML, CSS, Django web framework, MySQL database management system.

Results. The development of software for providing tennis statistics online was performed using the Django web framework with the definition of data structure, program requirements, development of the structure and algorithm of the program based on these requirements, the implementation of appropriate units. The program allows users to view the results of competitions, performance of tennis players in matches, during the whole tournament, during all seasons of the tournament, during all season, during all career, view comparable performance of tennis pairs, create own lists of tennis players, tournaments, tennis pairs.

Conclusions. The developed software allows to increase convenience of work with statistics of performance of tennis players on the basis of individual tracking, allowing to attract attention of a wide number of users.

Field of use. The software can be used as a basis for the operation of the tennis portal.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Основні поняття тенісної статистики	10
1.2 Огляд програмних аналогів.....	14
1.3 Висновки за розділом 1	18
2 Розробка архітектури програми.....	19
2.1 Вибір засобів проєктування і розробки	19
2.2 Визначення функціональних вимог до програмного забезпечення	20
2.3 Проєктування бази даних	24
2.4 Визначення структури програми.....	26
2.5 Висновки за розділом 2	27
3 Розробка програми	29
3.1 Алгоритм функціонування програми.....	29
3.2 Опис реалізованих одиниць вебзастосунку.....	31
3.3 Висновки за розділом 3	47
4 Експлуатація та тестування програми	48
4.1 Опис програми.....	48
4.2 Експлуатація програми.....	48
4.3 Тестування програми.....	54
4.4 Висновки за розділом 4	54
Висновки	55
Перелік джерел посилання	56
Додаток А Технічне завдання	58
Додаток Б Текст програми	62
Додаток В Слайди презентації.....	84

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ATP – Association of Tennis Professionals;

WTA – Women's Tennis Association.

ВСТУП

Теніс – один з дуже популярних видів спорту, щодо розвитку і стану якого можна зазначити наступне:

- приблизно 87 мільйонів людей грають у теніс у всьому світі, що складає 1,17% населення світу;
- теніс є одним із найпопулярніших видів спорту на телебаченні і широко визнаний одним із найпопулярніших індивідуальних, тобто некомандних видів спорту в світі;
- аналізуючи дані Google Trends щодо тенісу в порівнянні з іншими видами спорту на глобальному рівні, виявляється, що теніс випереджає багато видів спорту з точки зору інтересу пошуку: великі командні види спорту, такі як футбол і баскетбол, є більш популярними, але теніс привертає більший загальний інтерес, ніж інші індивідуальні види спорту, такі як бокс, плавання та їзда на велосипеді [1].

Окрім того потрібно відзначити, що протягом останнього періоду відбувається зростання тенісу, при чому це відбувається зокрема на фоні вируючої пандемії:

- пандемія спричинила зростання популярності тенісу, а якби не пандемія, то багато дослідників доходять до висновку, що теніс був би там, де він був у 2019 чи 2018 роках;
- зростання в критичних умовах пояснюється тим, що теніс дозволяє перебувати на безпечній соціальній дистанції, а тому він придатний для заповнення деяких необхідних фізичних, розумових і соціальних порожнеч, тож можна сказати, що пандемія допомогла висвітлити переваги цього виду спорту;
- протягом останніх років у тенісних клубах відчувається нестача тенісних м'ячів, а продажі ракеток початкового рівня виросли на 40%, до того ж заповнюваність тенісних майданчиків фактично збільшується [2].

Зважаючи на таку активність, користувачі потребують доступу до статистики, результатів тенісних змагань онлайн. І цей процес викликає значний інтерес. Сучасні тенденції роботи з інформацією полягають в її персоналізації. І попередній пошук вказує на те, що як раз з персоналізацією статичних даних є певні прогалини. Саме на владнання цих прогалин і направлена дана робота. Метою роботи було визначено розробку програмного забезпечення надання тенісної статистики онлайн для забезпечення можливостей індивідуального відслідковування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття тенісної статистики

Для того щоб визначити основні статистичні показники в тенісі, спочатку потрібно дати визначення основним термінам у даній галузі:

- ейс – м'яч, який подається настільки добре, що суперник не може торкнутися його своєю ракеткою;
- рівність – однакова кількість очок у гравців;
- бекхенд – удар, який використовується для повернення м'ячів, що влучили у ліву частину майданчика праворукого гравця (або в праву сторону ліворукого);
- базова лінія – задня лінія майданчика, яка проходить паралельно сітці та перпендикулярно бічній лінії;
- дьос – кількість очок 40-40, тобто рівний рахунок, і кожна сторона вже виграла щонайменше три очки;
- подвійна помилка – невдача обох спроб виконати подачу, при цьому подаючий втрачає очко;
- дропшот – м'яко вдарений м'яч з великою кількістю обертів, який потрапляє біля сітки після перетину.
- помилка – подача м'яча, яка не потрапляє у відповідну зону для подачі;
- форхенд – удар, який використовується для повернення м'ячів, що влучили в праву сторону майданчика праворукого гравця (або в ліву сторону ліворукого);
- гейм – частина сету, яка завершується, коли один гравець або сторона або виграє чотири очки і випереджає свого суперника щонайменше на два очки, або сторона виграє два очки поспіль після дьосу;
- поєдинок – загальне змагання, переможець якого зазвичай визначається найкращими двома з трьох сетів;
- аут – м'яч, що приземлився за межі лінії майданчика;

- оверхед – удар ракеткою над головою, подібний до руху зверху, під час гри;
- очко – найменша одиниця оцінки;
- сет – підрахункова одиниця, яка присуджується гравцеві або команді, яка виграла: 6 або більше геймів і має перевагу в два гейми або 6 геймів і гру на тайбрейку;
- слайс – удар, який надає м'ячу зворотнє обертання шляхом удару по м'ячу рухами від високого до низького;
- смеш – сильний удар зверху;
- тайбрейк – система в традиційному тенісі, яка використовується для визначення переможця сету, коли рахунок дорівнює 6-6 [3].

Для роботи з тенісною статистикою спочатку потрібно визначити основні показники виступу тенісистів, адже саме на їх основі накопичується статистика за окремими поєдинкам, а в результаті сумарно за виступ на турнірі, в сезоні, за кар'єру, а також створюються відповідні рейтинги за кожним конкретним показником, порівнюються уміння тенісистів між собою.

До основних показників, які формують статистику виступу тенісистів у поєдинку, входять:

- ейси (подачі на виліт): зазвичай даний показник характеризується відсотковим співвідношенням кількості ейсів (подач, які не були прийняті суперником, тобто до яких суперник взагалі не доторкнувся, і які дозволили заробити очки) до загальної кількості подач, хоча може окремо визначатися і абсолютним значенням;
- виграні подачі: зазвичай визначаються як відсоток виграних подач (подача, якої суперник торкнувся ракеткою, але не повернута на майданчик) від загальної кількості подач, хоча може окремо визначатися і абсолютним значенням;
- відсоток перших подач у грі – відсоток перших подач, які не були відправлені в аут і за якими не слідувала повторна (друга) подача;

– відсоток виграних перших подач, які були у грі, – відсоток перших подач, які дозволили набрати очки тенісисту, серед всіх перших подач, які не були відправлені в аут;

– відсоток других подач у грі – відсоток других подач, які не були відправлені в аут і залишились у грі;

– відсоток виграних других подач, які були у грі, – відсоток других подач, які дозволили набрати очки тенісисту, серед всіх других подач, які не були відправлені в аут;

– подвійні помилки – відсоток подвійних помилок від загальної кількості других подач;

– пікова швидкість подачі – максимальна швидкість подачі для всього матчу (окремо можуть розглядатися перша і друга подачі, загалом подачі, а також середня швидкість всіх подач);

– виграні прийоми – відсоток прийомів, які завершилися виграними очками, серед всіх виконаних прийомів подач суперника;

– невимушені помилки на прийомі – відсоток невимушених помилок у виконаних прийомах подач;

– виграні прийоми першої подачі – відсоток очок, виграних під час прийому першої подачі суперника;

– виграні прийоми другої подачі – відсоток очок, виграних під час прийому другої подачі суперника;

– виграні нетпойнти – відсоток очок, виграних у сітки;

– виграні нетпойнти в загальній кількості виграних очок – відсоток виграних у сітки очок від загальної кількості виграних очок під час матчу;

– брейкпойнти на прийомі – кількість брейкпойнтів, отриманих під час подачі суперника;

– виграні брейкпойнти – відсоток виграних брейкпойнтів з подач суперника;

– збережені брейкпойнти – відсоток брейкпойнтів, отриманих під час гри на власну подачу;

- перемоги у розіграші – відсоток виграних розіграшів відносно всієї кількості розіграшів;
- вимушені помилки в розіграші – відсоток вимушених помилок у всіх програних розіграшах;
- невимушені помилки в розіграші – відсоток невимушених помилок у всіх в програмних розіграшах;
- коефіцієнт домінування – співвідношення відсотку очок, виграних на подачах супротивника, до відсотку очок, програних на власних подачах;
- загальна відстань, подолана в матчі, – відстань, яку подолав у матчі окремий гравець;
- відстань, яку подолав тенісист за сет, – середня відстань, яку подолав у кожній партії окремий гравець;
- відстань, яку подолав тенісист за отримане очко, – середня відстань, яку подолав окремий гравець на кожне отримане очко [4]-[6].

Однак перелік показників не обмежується лише тільки переліченими. До них також можна віднести зокрема наступні:

- співвідношення ейсів до подвійних помилок;
- співвідношення швидкості першої подачі до швидкості другої подачі;
- співвідношення максимальної швидкості подачі до середньої швидкості;
- кількість брейкпойнтів;
- середня кількість брейкпойнтів на поєдинок;
- сумарна кількість розіграних очок;
- кількість виграних очок;
- відсоток виграних очок;
- кількість очок за поєдинок;
- кількість очок за сет;
- кількість зіграних тайбрейків;
- кількість виграних тайбрейків;

- відсоток виграних тайбрейків серед всіх зіграних тайбрейків;
- кількість тайбрейків на один поєдинок;
- зіграна кількість сетів;
- виграна кількість сетів;
- відсоток виграних сетів;
- середня кількість сетів на один поєдинок;
- кількість зіграних поєдинків;
- кількість виграних поєдинків;
- відсоток виграних поєдинків.

Частина цих показників більше характеризує окремий поєдинок, але може узагальнюватися до усереднених значень за всю кар'єру тенісиста, сезон виступу або конкретний турнір, частина показників використовуються тільки як узагальнені. Наприклад, кількість апсетів (протистоянь з тенісистом, який в рейтингу займав вище місце на момент поєдинку), відсоток виграних апсетів, кількість зворотних апсетів (програних протистоянь з тенісистами, які займали нижчі позиції в рейтингу).

Окрім безпосередніх окремих дій оцінюються також відсоток виграних сетів серед всіх сетів, у яких брав участь тенісист, відсоток виграних поєдинків серед всіх поєдинків, у яких брав участь тенісист.

Також аналогічні показники вводяться для оцінки дій суперника. Це зокрема відсоток ейсів, які було виконано суперником, відсоток подвійних помилок суперника.

Основними світовими рейтинговими системами тенісистів на даний момент є: Association of Tennis Professionals (ATP), Women's Tennis Association (WTA) [7]-[8].

1.2 Огляд програмних аналогів

Оскільки, як було проаналізовано раніше, теніс є одним з найпопулярніших видів спорту, то звичайно, що існує достатньо багато

вебсайтів, які надають можливість роботи з тенісною статистикою. У даному підрозділі вони будуть розглянуті в якості програмних аналогів і проаналізовані щодо тих можливостей, які вони надають користувачу.

Першим програмним аналогом є вебсайт Ultimate Tennis Statistics [9] (рис. 1.1).

Ultimate Tennis Statistics Updated: 04-04-2022

Welcome to **Ultimate Tennis Statistics**, the ultimate men's tennis statistics destination for the die-hard tennis fans! Be it latest or historical results, in-match statistics, records or all-time greats comparisons, Elo ratings or tournament forecasts, you will find it here. More details on the [About](#) page.

Latest Forecasts Powered by *Tennis Crystal Ball*

Tournament	Surface	Draw	1-st Favorite	Forecast
Houston	Clay	KO 28	Reilly Opelka 24.7%	Forecast
Marrakech	Clay	KO 32	Felix Auger Aliassime 16.0%	Forecast

Latest Results

Date	Tournament	Surface	Draw	Winner
23-03-2022	Miami Masters	Hard	KO 96	Carlos Alcaraz Garfia
10-03-2022	Indian Wells Masters	Hard	KO 96	Taylor Harry Fritz
21-02-2022	Acapulco	Hard	KO 32	Rafael Nadal

Top 10 04-04-2022 ATP

R.	Name	Pts.
1	Djokovic	8420
2	Medvedev	8410
3	Zverev	7195
4	Nadal	7115
5	Tsitsipas	5980
6	Berrettini	4945
7	Ruud	4380
8	Rublev	4375
9	A. Aliassime	3625
10	Norrie	3440

[Live Scores](#) via [Enetscores](#)

Рисунок 1.1 – Вебсайт Ultimate Tennis Statistics

Ultimate Tennis Statistics надає можливість отримувати інформацію про:

- результати останніх поєдинків;
- світові рейтинги ATP і WTA, а також рейтинги за покриттям, на якому грають тенісисти;
- статистику протистоянь тенісистів, що включає узагальнену статистику, статистику за сезонами, за поєдинками, порівняну динаміку зміни місць у рейтингу, статистику за всіма показниками в порівнянні;
- результати конкретних турнірів;
- сумарну статистику результатів за турніром;

- статистику тенісистів за показниками;
- дозволяє виконувати пошук тенісистів.

До недоліків даного ресурсу потрібно віднести відсутність можливостей індивідуального відслідковування. Тобто кожен раз, заходячи на даний вебсайт, користувач буде вимушений заново шукати потрібну йому статистику. Це не на стільки проблемно, коли вона стосується найближчого поєдинку, але якщо користувачу потрібно знайти необхідного йому тенісиста, його загальну статистику, статистику за певний поєдинок, порівняти між собою декількох тенісистів, то незручність одразу стає помітною.

Другим аналогом є вебсайт Tennis Explorer (рис. 1.2). Даний сайт більше присвячений накопиченню результатів, а не значень показників як характеристик окремих поєдинків або накопичених показників виступів загалом [10].

The screenshot displays the Tennis Explorer website interface. The header features the site's logo, navigation links for 'TENNISEXPLORER', 'LIVE SCORES', and 'COMPETITION', along with a login/register section. The main content area is dedicated to the profile of Novak Djokovic, including his photo, personal details (Country: Serbia, Height/Weight: 187 cm / 80 kg, Age: 34), and current rankings. A 'Next match' section indicates no upcoming matches. Below this is a 'Tournaments results' section with a search bar. The 'Player's record' section shows a table of wins/losses for singles, doubles, and mixed doubles across different surfaces (Clay, Hard, Indoors, Grass, Not set) for the years 2022 and 2021. On the right side, there are sections for 'TENNIS TWEETS' featuring tweets from WTA and Caroline Wozniacki, and 'THIS WEEK'S TOURNAMENTS' listing various tournaments with their prize money.

Player's record

	W/L - singles	W/L - doubles	W/L - mixed doubles				
	Year	Summary	Clay	Hard	Indoors	Grass	Not set
2022		2/1	-	2/1	-	-	-
2021		56/7	18/3	20/3	7/1	7/0	4/0

THIS WEEK'S TOURNAMENTS

Main tournaments	Prize money
Houston	\$ 10
Marrakech	\$ 7
Charleston	\$ 12
Bogotá WTA	\$ 8

Lower level tournaments

Tournament	Prize money
Mexico City challenger	\$ 8
Salinas challenger	\$ 7
Murcia challenger	\$ 8
Oeiras 2 challenger	\$ 6

Рисунок 1.2 – Вебсайт Tennis Explorer

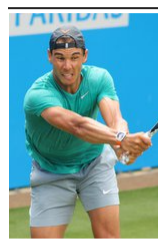
Надається можливість переглядати рейтинги тенісистів за заданим роком, отримувати інформацію про виступ потрібного тенісиста, включаючи результати його поєдинків, виграні титули, його травми.

Даний вебсайт має можливості реєстрації і авторизації, проте дані засоби можуть використовуватися для підтримки роботи з форумом, щоб поспілкуватися з іншими користувачами, а через те, що статистика виступу тенісистів за показниками на сайті загалом відсутня, то і індивідуальне відслідковування також не може бути використане.

На противагу другому аналогу і відповідній групі аналогів, які можна сюди долучити, потрібно розглянути вебсайт, повністю присвячений статистиці, – вебсайт Tennis Abstract (рис. 1.3) [11].

tennisabstract.com

Player Search



Rafael Nadal [ESP]
[@RafaelNadal](#)
 Date of birth: 03-Jun-1986
 Plays: Left (two-handed backhand)
 Current rank: 4
 Peak rank: 1 (18-Aug-2008)
 Doubles peak: 26 (08-Aug-2005)
 Profiles: [ATP](#) | [ITF](#) | [DC](#) | [Wiki](#)
[Titles/Finals](#)
 Photo: [si.robi](#)

[Singles Results](#) [Head-to-Heads](#) [Event Records](#)

TOTALS	Match	Tiebreak	Ace%	1stIn	1st%	2nd%	Hld%	SPW	Brk%	RPW	TPW	DR
Last 52	40-5 (89%)	10-4 (71%)	4.9%	64.4%	72.5%	53.8%	83.8%	65.8%	33.5%	42.7%	54.2%	1.25
Hard	21-2 (91%)	8-2 (80%)	6.1%	65.0%	74.0%	54.0%	86.5%	67.3%	29.0%	40.9%	54.0%	1.25
Clay	19-3 (86%)	2-2 (50%)	3.5%	63.7%	70.7%	52.8%	80.7%	64.2%	38.8%	44.7%	54.5%	1.25
Grass	0-0 (-)	0-0 (-)	-	-	-	-	-	-	-	-	-	-
Grand Slams	12-1 (92%)	2-2 (50%)	5.5%	63.2%	75.9%	53.1%	86.3%	67.5%	34.3%	42.2%	54.7%	1.30
vs Top 10	7-3 (70%)	0-3 (0%)	2.8%	66.0%	65.2%	52.0%	77.9%	61.0%	27.8%	40.3%	50.8%	1.03
vs Righties	33-5 (87%)	8-4 (67%)	4.7%	64.2%	72.0%	54.2%	83.5%	65.6%	33.3%	42.6%	54.0%	1.24
vs Lefties	7-0 (100%)	2-0 (100%)	6.4%	65.4%	74.8%	51.9%	85.3%	66.9%	34.4%	43.2%	55.0%	1.31
Best of 3	28-4 (88%)	8-2 (80%)	4.6%	65.1%	70.6%	54.3%	82.3%	64.9%	33.0%	43.0%	53.9%	1.22
Best of 5	12-1 (92%)	2-2 (50%)	5.5%	63.2%	75.9%	53.1%	86.3%	67.5%	34.3%	42.2%	54.7%	1.30

[show yearly totals](#) [hide splits](#)

Matches (40-5) > Last 52

[Reverse Loss Scores](#) [Stats: Serve](#) [Return](#) [Raw](#)

▲ Head-to-Head	Date	Tournament	Surface	Rd	Rk	vRk	Score	More	DR	A%	DF%	1stIn	1st%	2nd%	BPSvd	Time
▲ Exclude Opp	7-Mar-2022	Indian Wells Masters	Hard	F	4	20	(20) Taylor Fritz [USA] d. (4) Nadal	(ch)	0.85	2.6%	3.8%	66.7%	55.8%	53.8%	5/9	2:06
▲ Time Span	7-Mar-2022	Indian Wells Masters	Hard	SF	4	19	(4) Nadal d. (19) Carlos Alcaraz [ESP]	(ch)	1.03	1.9%	3.7%	66.7%	62.5%	44.4%	9/14	3:12
▲ Surface	7-Mar-2022	Indian Wells Masters	Hard	QF	4	132	(4) Nadal d. (WC) Nick Kyrgios [AUS]	(ch)	1.07	9.8%	6.9%	65.7%	77.6%	48.6%	3/5	2:46
▲ Level	7-Mar-2022	Indian Wells Masters	Hard	R16	4	17	(4) Nadal d. (17) Reilly Opelka [USA]		1.81	8.2%	4.1%	69.9%	84.3%	72.7%	3/4	2:11
▲ Event	7-Mar-2022	Indian Wells Masters	Hard	R32	4	29	(4) Nadal d. (27) Daniel Evans [GBR]		1.33	6.1%	7.6%	60.6%	75.0%	65.4%	3/4	1:43
▲ Round	7-Mar-2022	Indian Wells Masters	Hard	R64	4	38	(4) Nadal d. Sebastian Korda [USA]		1.05	5.1%	8.2%	62.2%	60.7%	51.4%	7/11	2:29

Рисунок 1.3 – Вебсайт Tennis Abstract

Даний вебсайт також не має засобів індивідуального доступу, але він має достатньо широку кількість про статистику виступів тенісистів, включаючи результати, значення окремих показників у кожному окремому поєдинку, що тенісист провів. Окрім того можна виконувати пошук тенісистів, переглядати рейтинги, статистику за сезон тенісиста, історію позицій в рейтингу на кінець сезону, статистику за турнірами, дані про

протистояння з суперниками, з якими тенісист проводив поєдинки найчастіше.

1.3 Висновки за розділом 1

Перший розділ направлений на аналіз предметної області, в межах якого було виділено основні поняття і показники, які стосуються статистики індивідуальних виступів тенісистів, проведено огляд програмних аналогів, включаючи вебсайти Ultimate Tennis Statistics, Tennis Explorer, Tennis Abstract. Проведений аналіз продемонстрував, що частина цих сайтів направлена на доступ безпосередньо до широкої кількості показників, якими характеризується виступ тенісиста в поєдинку, а інша частина – на перегляд результатів тенісних поєдинків і спілкування поціновувачів. Проте наявні програмні аналоги не дозволяють індивідуальний доступ до статистики, що актуально в умовах великої кількості показників, великої кількості турнірів, великої кількості тенісистів. Звичайно, що вболівальників зазвичай цікавлять результати і статистика окремих тенісистів, а також загалом результати турнірів. Відповідно зручним способом роботи було б обрання потрібних тенісистів, турнірів зі всього набору, щоб під час доступу до вебсайту починати роботу саме з обраних позицій. Тому в роботі важливо розробити програмне забезпечення, яке дозволить вирішити дану проблему.

Завданнями роботи для сукупного досягнення мети є:

- аналіз програмних аналогів;
- визначення засобів проєктування і розробки програми;
- проєктування програми;
- реалізація і налагодження програми.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір засобів проєктування і розробки

Оскільки створення даного вебзастосунку має певну специфіку – передбачає роботу з великими наборами даних, що визначають значення показників виступу тенісистів, при цьому ці набори потрібно постійно обробляти, щоб накопичувати значення за певні періоди, то логічним вибором було обрання для розробки основної мови, яка на даний момент використовується для оброблення даних зокрема в вебзастосунках, – Python.

У подальшому виникло питання вибору конкретного вебфреймворку серед Flask [12] і Django [13], щодо якого за сукупністю показників (табл. 2.1) було обрано Django.

Таблиця 2.1 – Вибір фреймворку розробки вебзастосунку

Критерій	Flask	Django
Зручна робота з даними	Так	Так
Зручна робота над формуванням вебсторінок в динаміці	Так	Так
Адміністративна панель за замовчуванням	Ні	Так
Інструменти безпеки	Гірші	Кращі
Об'єктно-реляційне відображення за замовчуванням	Ні	Так

2.2 Визначення функціональних вимог до програмного забезпечення

Програмне забезпечення надання тенісної статистики онлайн повинно надавати можливість роботи з цілим набором функцій, які мають використовуватися для зручної роботи:

- пошук тенісистів;
- перегляд тенісистів за алфавітом;
- перегляд списку турнірів;
- перегляд доступних сезонів турніру;
- внесення даних тенісистів;
- редагування даних тенісистів;
- автоматичний розрахунок показників виступу тенісиста за сезон турніру;
- автоматичний розрахунок показників виступу тенісиста за турнір;
- автоматичний розрахунок показників виступу тенісиста за сезон;
- автоматичний розрахунок показників виступу тенісиста за всю кар'єру;
- внесення даних турнірів;
- редагування даних турнірів;
- внесення сезонів турнірів;
- внесення поєдинків турнірів;
- перегляд поєдинків турнірів;
- внесення подій поєдинку;
- внесення рейтингів тенісистів;
- відображення динаміки зміни виступу тенісиста за обраним параметром;
- відображення порівняльної динаміки зміни виступу двох тенісистів за обраним параметром;
- перегляд результатів турніру;

- перегляд порівняльної статистики обраних тенісистів;
- перегляд історії матчів між обраними тенісистами;
- перегляд рейтингів тенісистів за обраною датою;
- перегляд усіх поєдинків тенісиста;
- перегляд поєдинків тенісиста за обраним турніром;
- перегляд поєдинків тенісиста за обраним сезоном;
- перегляд поєдинків тенісиста за сезоном турніру;
- перегляд детальної статистики за обраним поєдинком за загальними і індивідуальними показниками з розділенням за групами;
- перегляд детальних результатів обраного поєдинку;
- відображення списків тенісистів, упорядкованих за значеннями індивідуальних показників;
- перегляд інформації про тенісиста;
- перегляд статистики виступу тенісиста проти інших тенісистів;
- перегляд історії місць тенісиста у світових рейтингах;
- авторизація користувачів;
- реєстрація користувачів;
- створення списків відслідковування тенісистів користувачем;
- редагування списків відслідковування тенісистів користувачем;
- створення списків відслідковування результатів турнірів користувачем;
- редагування списків відслідковування результатів турнірів користувачем;
- створення списків відслідковування пар протистоянь користувачем;
- редагування списків відслідковування пар протистоянь користувачем.

У відповідності з даними функціональними можливостями визначено сценарії роботи неавторизованого користувача (рис. 2.1):

- пошук тенісистів;
- перегляд поточних поєдинків;

- перегляд турнірів;
- порівняння тенісистів, включаючи перегляд порівняльної статистики обраних тенісистів та перегляд історії матчів між обраними тенісистами;
- перегляд інформації про тенісиста, включаючи перегляд поєдинків тенісиста, перегляд показників виступу тенісиста;
- перегляд рейтингів тенісистів;
- реєстрація користувачів.



Рисунок 2.1 – Діаграма прецедентів неавторизованого користувача

Сценарії роботи авторизованого користувача (рис. 2.2):

- керування списками тенісистів;
- керування списками турнірів;
- керування списками пар протистоянь;

- пошук тенісистів;
- перегляд поточних поєдинків;
- перегляд турнірів;
- порівняння тенісистів;
- перегляд інформації про тенісиста;
- перегляд рейтингів тенісистів;
- авторизація користувача.



Рисунок 2.2 – Діаграма прецедентів авторизованого користувача

Сценарії роботи менеджера (рис. 2.3):

- пошук тенісистів, що як альтернативний сценарій включає перегляд тенісистів за алфавітом;
- керування турнірами;

- керування даними тенісистів;
- внесення подій поєдинку, що включає розрахунок показників виступу тенісиста;
- керування рейтингом тенісистів;
- авторизація менеджера.

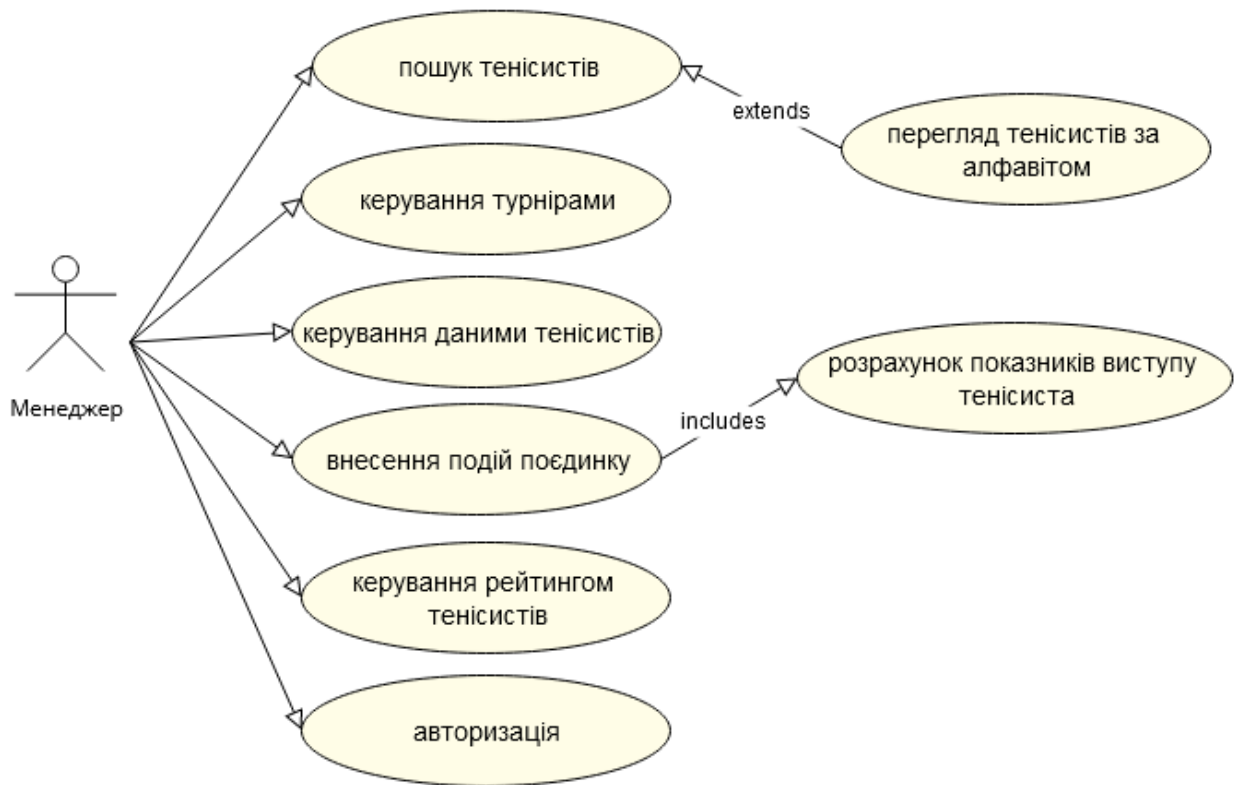


Рисунок 2.3 – Діаграма прецедентів менеджера

2.3 Проєктування бази даних

Базу даних було створено на основі системи керування базами даних MySQL.

За результатами вивчення предметної області, виконаного в розділі 1, було сформовано список основних сутностей предметної області зі створенням відповідних таблиць бази даних:

- tennis_player – тенісист;
- tennis_country – країна, національність якої має тенісист;

- tennis_tournament – турнір, в якому беруть участь тенісисти;
- tennis_season – сезон;
- tennis_match – тенісний поєдинок, який відбувається в межах певного турніру;
- tennis_match_set – сет тенісного поєдинку;
- tennis_match_game – гейм сету тенісного поєдинку;
- tennis_player_match_statistics – показники виступу тенісиста в поєдинку;
- tennis_tournament_season – сезон турніру, тобто проведення турніру в певний сезон (рік);
- tennis_rating_system – рейтингова система;
- tennis_rating – положення тенісиста в конкретному рейтингу;
- tennis_overall_statistics – статистика виступу тенісиста, тобто узагальнення його показників за деякий період;
- tennis_tournament_season_overall_statistics – узагальнена статистика виступу тенісиста за сезон турніру (тобто фактично конкретне проведення турніру);
- tennis_tournament_overall_statistics – узагальнена статистика виступу тенісиста за всі сезони турніру;
- tennis_season_overall_statistics – узагальнена статистика виступу тенісиста за весь сезон;
- tennis_user_players – список тенісистів, які хоче переглядати користувач;
- tennis_user_tournaments – список турнірів, які хоче переглядати користувач;
- tennis_user_headtoheads – список пар протистоянь тенісистів, які хоче переглядати користувач.

Схема створеної бази даних представлена на рис. 2.4.

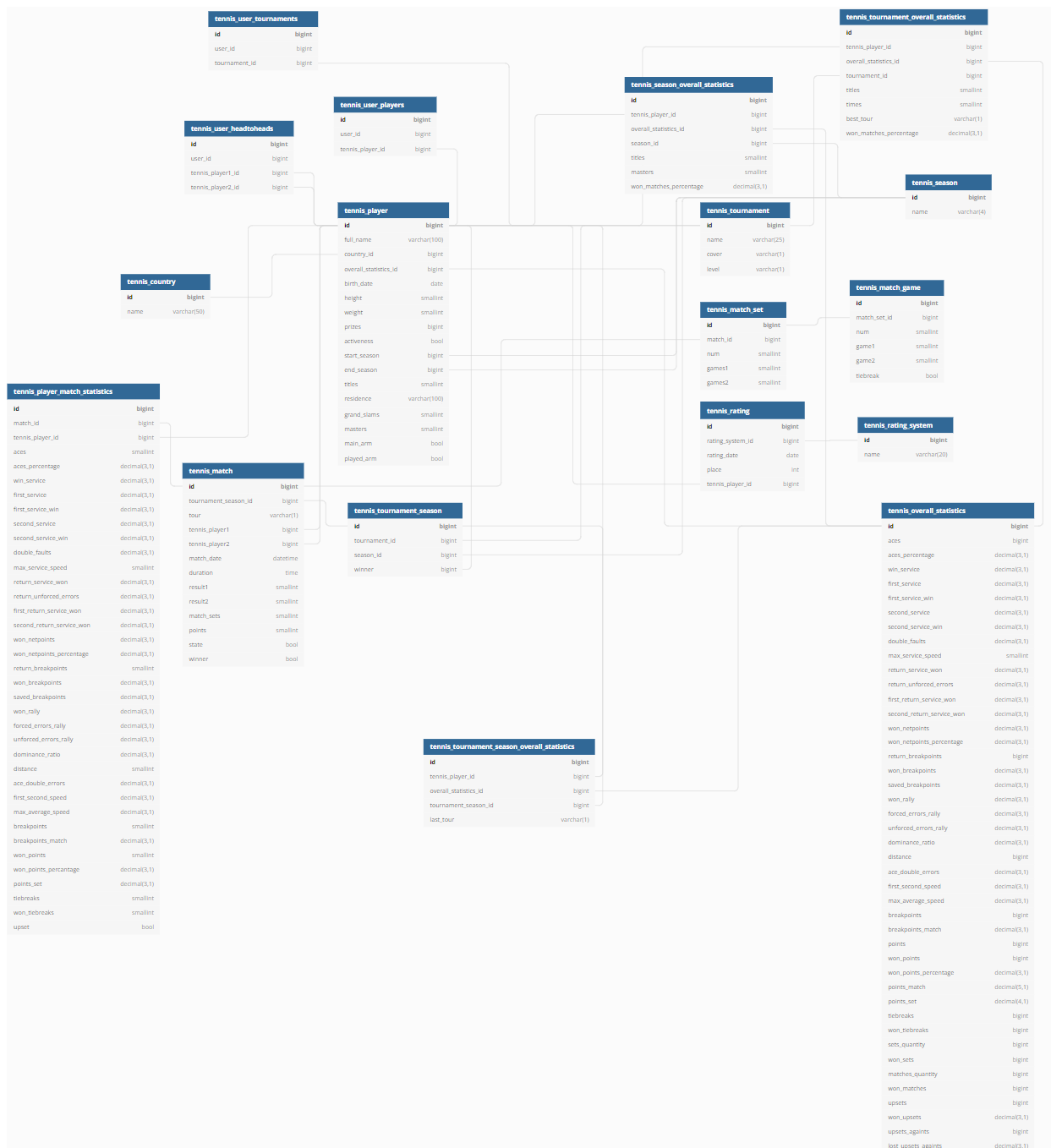


Рисунок 2.4 – Схема бази даних

2.4 Визначення структури програми

Структуру програми (рис. 2.5) побудовано на основі структури, яка передбачається застосуванням вебфреймворку Django.

Окрім безпосередньо самого стандартного застосунку Django було створено окремо налаштування застосунку та статичні файли, до яких зокрема належать скрипти Javascript. Окрім базової роботи вони також

забезпечують зокрема роботу з даними на користувацькій стороні. Це пояснюється тим, що не завжди потрібно відображати всі дані користувачу, хоча для скорочення часу роботи вони завантажуються в повній мірі, а вже в кожен поточний момент користувач обирає потрібну йому порцію.



Рисунок 2.5 – Структура програми

До самої структури застосунку входять моделі, представлення, менеджер URL для керування представленнями, які мають відображатися, шаблони вебсторінок та тести, які використовуються для перевірки коректності роботи застосунку.

2.5 Висновки за розділом 2

Виконано вибір фреймворку Django для розробки вебзастосунку після порівняння з фреймворком Flask як найбільш популярних фреймворків

розробки вебзастосунків мовою Python. Виділено основні сценарії роботи авторизованого і неавторизованого користувачів, менеджера з програмою. Побудовано схему бази даних. Визначено структуру програми.

3 РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програми

Функціональність, яка надається програмою для авторизованого і неавторизованого користувачів, пов'язана в основному з відтворенням певної інформації. У той же час взаємодія менеджера з програмою не на стільки стандартна, оскільки передбачає також додаткові програмні дії і обчислення. Тому в даному підрозділі описано детально саме логіку роботи менеджера з програмою для визначення алгоритму функціонування програми (рис. 3.1).

Під час роботи з програмою менеджер має доступ до основних її функцій. Вони передбачають внесення даних про тенісиста, його пошук, створення турніру, а також вибір турніру з наданого списку. Після того, як турнір обрано, менеджер може або відредагувати його дані, або створити сезон турніру, або обрати один зі створених сезонів. У останньому випадку менеджеру надається можливість створити поєдинок. Одразу зі створенням поєдинку реалізується автоматичне створення сету та гейму. Після цього менеджер отримує можливість вносити події гейму. Кожна подія обробляється і внесені параметри призводять до оновлення статистики тенісиста в поєдинку.

Після кожного такого оновлення реалізується перевірка завершення гейму. Якщо гейм не завершився, то надається можливість знову внести подію. Якщо гейм завершився, то перевіряється, чи завершено сет. Якщо сет не завершено, то створюється новий гейм. Якщо сет завершено, то перевіряється, чи завершено поєдинок. Якщо поєдинок не завершено, то створюється новий сет. Якщо поєдинок завершено, то автоматично реалізується оновлення статистики тенісиста в сезоні турніру, оновлення статистики тенісиста в турнірі загалом та оновлення статистики тенісиста за кар'єру.

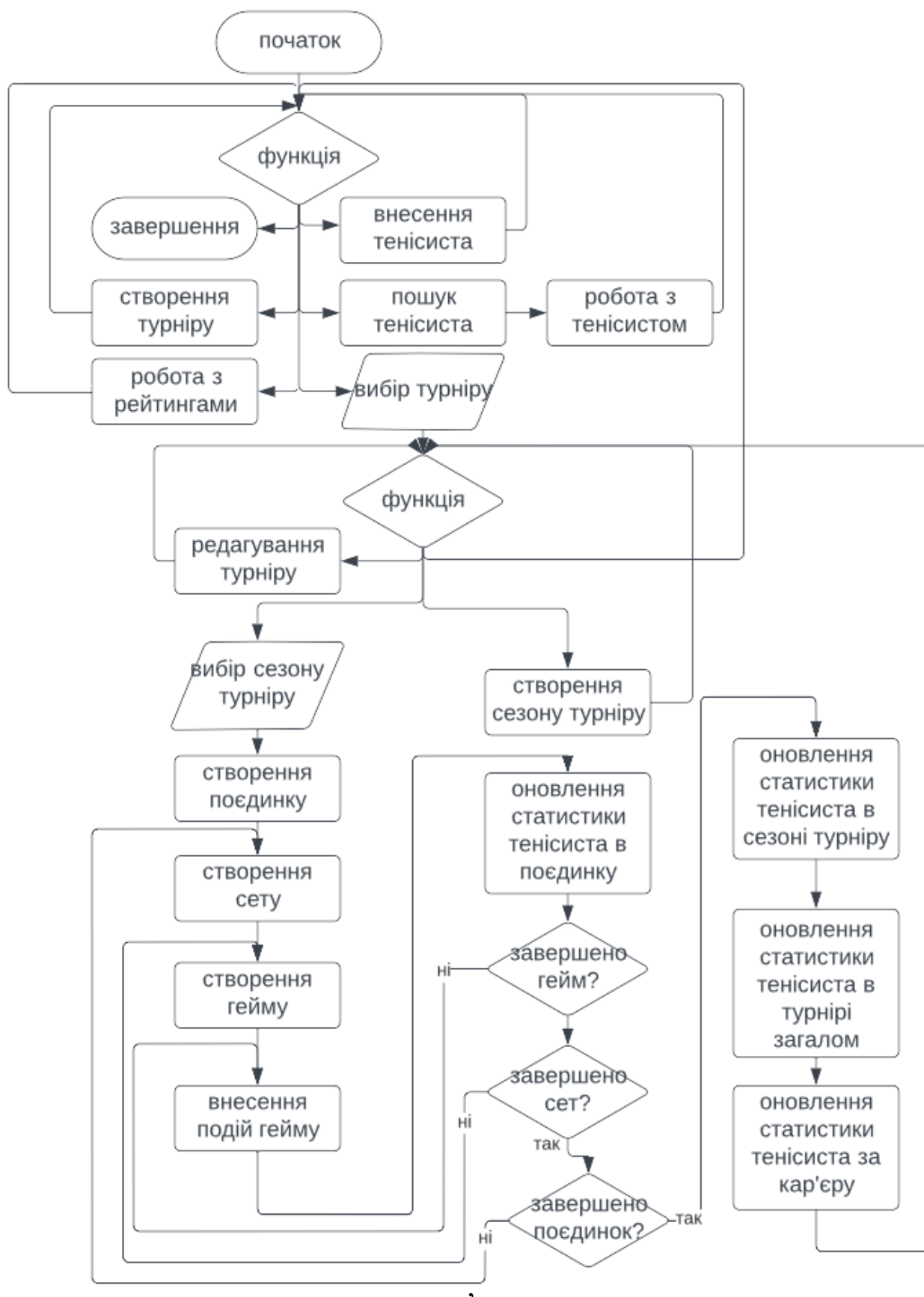


Рисунок 3.1 – Схема функціонування програми для менеджера

3.2 Опис реалізованих одиниць вебзастосунку

Модель тенісиста організована класом Player за допомогою наступних атрибутів класу:

- full_name – повне ім'я тенісиста, що є рядком символів, створеним як об'єкт класу CharField;
- country_id – країна походження тенісиста, що є зовнішнім ключем, який пов'язаний з моделлю Country і є об'єктом класу ForeignKey;
- overall_statistics_id – узагальнена статистика за всю кар'єру виступу тенісиста, визначає зв'язок типу один-до-одного з моделлю Overall_statistics, створюючи об'єкт класу OneToOneField;
- birth_date – дата народження тенісиста, яка є об'єктом класу DateField;
- height – висота тенісиста, яка є об'єктом класу SmallIntegerField;
- weight – вага тенісиста, яка є об'єктом класу SmallIntegerField;
- prizes – сумарна кількість призових, виграна тенісистом за всю його кар'єру, яка є об'єктом класу BigIntegerField;
- activeness – активність (визначає, чи проводить на даний момент тенісист поєдинки, чи він вже завершив кар'єру), яка є об'єктом класу BooleanField;
- start_season – сезон початку кар'єри тенісиста, що є зовнішнім ключем, який пов'язаний з моделлю Season і є об'єктом класу ForeignKey;
- end_season – сезон завершення кар'єри тенісиста, що є зовнішнім ключем, який пов'язаний з моделлю Season і є об'єктом класу ForeignKey;
- titles – кількість титулів, яку за всю кар'єру виграв тенісист, що є об'єктом класу SmallIntegerField;
- residence – інформація про поточне проживання тенісиста (місто, країна, де він проживає на даний момент), яка є об'єктом класу CharField;
- grand_slams – сумарна кількість перемог тенісиста в турнірах Великого шолому, яка є об'єктом класу SmallIntegerField;

- `masters` – сумарна кількість перемог тенісиста в турнірах класу `Мастерз`, яка є об'єктом класу `SmallIntegerField`;

- `main_arm` – основна рука (ліва або права), якою грає тенісист, є об'єктом класу `BooleanField`;

- `played_arm` – параметр, що визначає, чи грає тенісист обома руками, чи використовує тільки одну руку, є об'єктом класу `BooleanField`.

Модель країни організована класом `Country` за допомогою атрибута класу `name`, що визначає назву країни і представлений максимально 50-символьним рядком класу `CharField`.

Модель турніру організована класом `Tournament` за допомогою наступних атрибутів класу:

- `name` – назва турніру, яка є об'єктом класу `CharField`;

- `cover` – покриття майданчика, на якому відбувається турнір, що є об'єктом класу `CharField` і передбачає вибір варіантів;

- `level` – клас (рівень) турніру, що є об'єктом класу `CharField` і передбачає вибір варіантів.

Вибір варіантів покриття включає:

- трава (значення «G»);

- тверде покриття (хард, значення «H»);

- ґрунт (значення «C»);

- коврове (значення «R»).

Вибір варіантів рівня турніру включає:

- великий шолом (значення «A»);

- мастерз (хард, значення «B»);

- олімпіада (значення «C»);

- рівень АТР 250 (значення «D»);

- рівень АТР 500 (значення «E»).

Модель сезону організована класом `Season` за допомогою атрибута класу `name`, що визначає назву сезону (може бути просто рік) і представлений максимально 4-символьним рядком класу `CharField`.

Модель сезону турніру організована класом `Tournament_season` за допомогою наступних атрибутів класу:

- `tournament_id` – турнір, що є зовнішнім ключем, який пов'язаний з моделлю `Tournament`, є об'єктом класу `ForeignKey`;
- `season_id` – сезон, що є зовнішнім ключем, який пов'язаний з моделлю `Season`, є об'єктом класу `ForeignKey`;
- `winner` – переможець даного сезону турніру, що є зовнішнім ключем, який пов'язаний з моделлю `Player`, є об'єктом класу `ForeignKey`, використовується для пришвидшення доступу до інформації про переможця.

Модель тенісного поєдинку організована класом `Match` за допомогою наступних атрибутів класу:

- `tournament_season_id` – сезон турніру, в якому відбувається поєдинок, є зовнішнім ключем, який пов'язаний з моделлю `Tournament_season`, є об'єктом класу `ForeignKey`;
- `tennis_player1` – перший тенісист, що бере участь у поєдинку, є зовнішнім ключем, який пов'язаний з моделлю `Player`, є об'єктом класу `ForeignKey`;
- `tennis_player2` – другий тенісист, що бере участь у поєдинку, є зовнішнім ключем, який пов'язаний з моделлю `Player`, є об'єктом класу `ForeignKey`;
- `match_date` – дата проведення поєдинку, яка є об'єктом класу `DateTimeField`;
- `tour` – тур турніру, в якому проведено даний поєдинок, є об'єктом класу `CharField`, що забезпечує вибір варіанту;
- `duration` – тривалість поєдинку, яка є об'єктом класу `TimeField`;
- `result1` – кількість виграних сетів першим тенісистом, яка є об'єктом класу `SmallIntegerField`;
- `result2` – кількість виграних сетів другим тенісистом, яка є об'єктом класу `SmallIntegerField`;

- `match_sets` – кількість сетів у поєдинку, яка є об'єктом класу `SmallIntegerField`;
- `points` – кількість очок, яку було розіграно в поєдинку, є об'єктом класу `SmallIntegerField`;
- `state` – стан, в якому знаходиться поєдинок (завершений або ні), є об'єктом класу `BooleanField`;
- `winner` – переможець поєдинку (перший або другий тенісист), є об'єктом класу `BooleanField`: використовується для пришвидшення доступу до інформації про переможця без необхідності додаткових дій.

Варіанти турів турніру включають:

- фінал (значення «A»);
- поєдинок за третє місце (значення «B»);
- півфінал (значення «C»);
- чвертьфінал (значення «D»);
- одна восьма фіналу (значення «E»);
- одна шістнадцята фіналу (значення «F»);
- одна тридцять друга фіналу (значення «G»);
- одна шістдесят четверта фіналу (значення «H»);
- одна сто двадцять восьма фіналу (значення «I»);
- кваліфікація (значення «J»).

Функція `additionalMatch` визначає додаткові дії, які мають виконуватися під час створення нового поєдинку. В такому випадку одразу створюється новий сет та новий гейм, які дозволяють в подальшому менеджеру вносити дані за ними:

```
m = Match_set.objects.create(match_id=instance.id)
m.save()
Match_game.objects.create(match_set=m.id)
```

Після цього також створюються записи про статистику виступу першого тенісиста в поєдинку та другого. Також визначається, хто з тенісистів знаходиться вище в рейтингу на поточний момент і в залежності від цього визначається, чи це є апсет для першого тенісиста, чи для другого:

```
if tenp.place < tenp2.place:
    p1.unset = True
    p2.unset = False
elif tenp.place > tenp2.place:
    p2.unset = True
    p1.unset = False
```

Після цього також перевіряється, чи створено за кожним з тенісистів, що беруть участь в поєдинку, записи про статистику їх виступу в сезоні турніру, турнірі загалом та сезоні загалом.

Після цього реалізовано підключення даних функції як постобробника процесу ініціалізації об'єктів класу поєдинка:

```
post_init.connect(additionalMatch, Match)
```

Модель сету тенісного поєдинку організована класом `Match_set` за допомогою наступних атрибутів класу:

- `match_id` – поєдинок, в якому відбувається сет, є зовнішнім ключем, який пов'язаний з моделлю `Match`, є об'єктом класу `ForeignKey`;
- `num` – порядковий номер сету, є об'єктом класу `SmallIntegerField`;
- `games1` – кількість геймів у сеті, виграна першим тенісистом, є об'єктом класу `SmallIntegerField`;
- `games2` – кількість геймів у сеті, виграна другим тенісистом, є об'єктом класу `SmallIntegerField`.

Модель гейму тенісного поєдинку організована класом `Match_game` за допомогою наступних атрибутів класу:

- `match_set` – сет, в якому відбувається гейм, є зовнішнім ключем, який пов'язаний з моделлю `Match_set`, є об'єктом класу `ForeignKey`;
- `num` – порядковий номер гейму, є об'єктом класу `SmallIntegerField`;
- `game1` – кількість очок, набрана в геймі першим тенісистом, є об'єктом класу `SmallIntegerField`;
- `game2` – кількість очок, набрана в геймі другим тенісистом, є об'єктом класу `SmallIntegerField`;
- `tiebreak` – позначка, що визначає, чи даний гейм є тайбрейком чи ні, є об'єктом класу `BooleanField`.

Метод `save(self, *args, **kwargs)` забезпечує підтримку реалізації основних процесів взаємодії між різними статистичними складовими у відповідності з логікою, описаною в попередньому підрозділі.

Зокрема під час реалізації даного методу спочатку визначаються індивідуальні показники виступу тенісистів у поєдинку. Після цього перевіряється рахунок гейму. Це пов'язано з тим, що поточне розігране очко може бути останнім в геймі, а до того ж гейм може бути останнім в сеті. Для виявлення цієї ситуації спочатку перевіряється, чи даний гейм є тайбрейком, чи ні. Визначається різна процедура для тайбрейка та звичайного гейму, оскільки вони мають різні системи очок.

Якщо розіграні очки виявилися останніми в геймі, то додатково перевіряється, чи цей гейм не є останнім в сеті. Якщо ні, то тільки створюється новий гейм. Якщо це останній розіграш в сеті, то також перевіряється, чи не завершено гру. Якщо ні, то створюється новий сет і гейм.

При завершенні розіграшу тайбрейку оновлюється статистика тайбрейків тенісиста. Аналогічно оновлюються також і інші подібні статистичні показники. Це зокрема кількість виграних очок тенісистами, відсоток виграних очок тощо.

У випадку завершення гри також фіксується переможець, оновлюється статистика гравця щодо кількості виграних титулів загалом і певної категорії, якщо це був останній поєдинок на турнірі, оновиться інформація про переможця турніру:

```

if m.tour == 'A':
    pl = Player.objects.get(id=m.tennis_player2.id)
    pl.titles = pl.titles + 1
    if m.tournament_season_id.tournament_id.level == 'A':
        pl.grand_slams = pl.grand_slams + 1
    elif m.tournament_season_id.tournament_id.level == 'B':
        pl.masters = pl.masters + 1
    ts = Tournament_season.objects.get(id=m.tournament_season_id)
    ts.winner = pl

```

Модель статистики виступу тенісиста в окремому поєдинку організована класом `Player_match_statistics` за допомогою наступних атрибутів класу:

- `match_id` – поєдинок, в якому відбувається сет, є зовнішнім ключем, який пов'язаний з моделлю `Match`, є об'єктом класу `ForeignKey`;
- `tennis_player_id` – поєдинок, в якому відбувається сет, є зовнішнім ключем, який пов'язаний з моделлю `Player`, є об'єктом класу `ForeignKey`;
- `aces` – кількість ейсів, виконана тенісистом, є об'єктом класу `SmallIntegerField`;
- `aces_percentage` – відсоток ейсів, є об'єктом класу `DecimalField`;
- `win_service` – виграні подачі тенісиста, є об'єктом класу `DecimalField`;
- `first_service` – відсоток перших подач у поєдинку, виконаних тенісистом, є об'єктом класу `DecimalField`;

- first_service_win – відсоток виграних перших подач тенісистом у поєдинку, є об'єктом класу DecimalField;
- second_service – відсоток других подач тенісиста у поєдинку, є об'єктом класу DecimalField;
- second_service_win – відсоток виграних других подач тенісистом, є об'єктом класу DecimalField;
- double_faults – подвійні помилки, є об'єктом класу DecimalField;
- max_service_speed – пікова швидкість подачі, є об'єктом класу SmallIntegerField;
- return_service_won – виграні прийоми, є об'єктом класу DecimalField;
- return_unforced_errors – невимушені помилки на прийомі, є об'єктом класу DecimalField;
- first_return_service_won – виграні прийоми першої подачі, є об'єктом класу DecimalField;
- second_return_service_won – виграні прийоми другої подачі, є об'єктом класу DecimalField;
- won_netpoints – виграні нетпойнти, є об'єктом класу DecimalField;
- won_netpoints_percentage – виграні нетпойнти в загальній кількості виграних очок, є об'єктом класу DecimalField;
- return_breakpoints – брейкпойнти на прийомі, є об'єктом класу SmallIntegerField;
- won_breakpoints – виграні брейкпойнти, є об'єктом класу DecimalField;
- saved_breakpoints – збережені брейкпойнти, є об'єктом класу DecimalField;
- won_rally – перемоги у розігравші, є об'єктом класу DecimalField;
- forced_errors_rally – вимушені помилки в розігравші, є об'єктом класу DecimalField;

- `unforced_errors_rally` – невимушені помилки в розіграві, є об'єктом класу `DecimalField`;
- `dominance_ratio` – коефіцієнт домінування, є об'єктом класу `DecimalField`;
- `distance` – загальна відстань, подолана тенісистом у поєдинку, є об'єктом класу `SmallIntegerField`;
- `ace_double_errors` – співвідношення ейсів до подвійних помилок, є об'єктом класу `DecimalField`;
- `first_second_speed` – співвідношення швидкості першої подачі до швидкості другої подачі, є об'єктом класу `DecimalField`;
- `max_average_speed` – співвідношення максимальної швидкості подачі до середньої швидкості, є об'єктом класу `DecimalField`;
- `breakpoints` – кількість брейкпойнтів, є об'єктом класу `SmallIntegerField`;
- `breakpoints_match` – середня кількість брейкпойнтів на поєдинок, є об'єктом класу `DecimalField`;
- `won_points` – кількість виграних очок, є об'єктом класу `SmallIntegerField`;
- `won_points_percantage` – відсоток виграних очок, є об'єктом класу `DecimalField`;
- `points_set` – кількість очок, розіграних за сет, є об'єктом класу `DecimalField`;
- `tiebreaks` – кількість зіграних тайбрейків, є об'єктом класу `SmallIntegerField`;
- `won_tiebreaks` – кількість виграних тайбрейків, є об'єктом класу `SmallIntegerField`;
- `upset` – показник, який визначає, чи є даний поєдинок для даного тенісиста апсетом, є об'єктом класу `BooleanField`.

Метод `save(self, *args, **kwargs)` реалізує об'єднання статистики в залежності від змін статистики виступу тенісиста в поєдинку. Для цього

перевіряється стан поєдинку: якщо він завершений, то оновлюється статистика виступу тенісиста в даному сезоні турніру. Оновлення відбувається шляхом переобчислення всіх показників (значення в основному усереднюються або сумуються).

Після цього відбувається накопичення статистики за всі сезони виступу тенісиста в даному турнірі, що є основою обчислення статистики виступу тенісиста в турнірі загалом за всі сезони.

Далі відбувається накопичення статистики за весь сезон виступу тенісиста в різних турнірах на основі виділення всіх виступів тенісиста за даний сезон в турнірах. На останньому кроці накопичується статистика всіх виступів тенісиста за всю кар'єру.

Модель рейтингової системи організована класом `Rating_system` за допомогою атрибута класу `name`, що визначає назву країни і представлений максимально 20-символьним рядком класу `CharField`.

Модель місця тенісиста в рейтингу організована класом `Rating` за допомогою наступних атрибутів класу:

- `rating_system_id` – рейтингова система, за якою визначено місце, є зовнішнім ключем, який пов'язаний з моделлю `Rating_system`, є об'єктом класу `ForeignKey`;
- `tennis_player_id` – тенісист, є зовнішнім ключем, який пов'язаний з моделлю `Player`, є об'єктом класу `ForeignKey`;
- `place` – місце, яке в рейтингу займає тенісист, є об'єктом класу `IntegerField`;
- `rating_date` – дата, на яку визначений даний рейтинг, є об'єктом класу `DateField`.

Модель узагальненої статистики організована класом `Overall_statistics` за допомогою наступних атрибутів класу:

- `aces` – кількість ейсів, виконана тенісистом, є об'єктом класу `BigIntegerField`;
- `aces_percentage` – відсоток ейсів, є об'єктом класу `DecimalField`;

- win_service – виграні подачі тенісиста, є об'єктом класу DecimalField;
- first_service – відсоток перших подач, виконаних тенісистом, є об'єктом класу DecimalField;
- first_service_win – відсоток виграних перших подач тенісистом, є об'єктом класу DecimalField;
- second_service – відсоток других подач тенісиста, є об'єктом класу DecimalField;
- second_service_win – відсоток виграних других подач тенісистом, є об'єктом класу DecimalField;
- double_faults – подвійні помилки, є об'єктом класу DecimalField;
- max_service_speed – пікова швидкість подачі, є об'єктом класу SmallIntegerField;
- return_service_won – виграні прийоми, є об'єктом класу DecimalField;
- return_unforced_errors – невимушені помилки на прийомі, є об'єктом класу DecimalField;
- first_return_service_won – виграні прийоми першої подачі, є об'єктом класу DecimalField;
- second_return_service_won – виграні прийоми другої подачі, є об'єктом класу DecimalField;
- won_netpoints – виграні нетпойнти, є об'єктом класу DecimalField;
- won_netpoints_percentage – виграні нетпойнти в загальній кількості виграних очок, є об'єктом класу DecimalField;
- return_breakpoints – брейкпойнти на прийомі, є об'єктом класу BigIntegerField;
- won_breakpoints – виграні брейкпойнти, є об'єктом класу DecimalField;
- saved_breakpoints – збережені брейкпойнти, є об'єктом класу DecimalField;

- won_rally – перемоги у розігравші, є об'єктом класу DecimalField;
- forced_errors_rally – вимушені помилки в розігравші, є об'єктом класу DecimalField;
- unforced_errors_rally – невимушені помилки в розігравші, є об'єктом класу DecimalField;
- dominance_ratio – коефіцієнт домінування, є об'єктом класу DecimalField;
- distance – загальна відстань, подолана тенісистом, є об'єктом класу BigIntegerField;
- ace_double_errors – співвідношення ейсів до подвійних помилок, є об'єктом класу DecimalField;
- first_second_speed – співвідношення швидкості першої подачі до швидкості другої подачі, є об'єктом класу DecimalField;
- max_average_speed – співвідношення максимальної швидкості подачі до середньої швидкості, є об'єктом класу DecimalField;
- breakpoints – кількість брейкпойнтів, є об'єктом класу BigIntegerField;
- breakpoints_match – середня кількість брейкпойнтів на поєдинок, є об'єктом класу DecimalField;
- won_points – кількість виграних очок, є об'єктом класу BigIntegerField;
- won_points_percantage – відсоток виграних очок, є об'єктом класу DecimalField;
- points_set – кількість очок, розіграних у середньому за сет, є об'єктом класу DecimalField;
- tiebreaks – кількість зіграних тайбрейків, є об'єктом класу BigIntegerField;
- won_tiebreaks – кількість виграних тайбрейків, є об'єктом класу BigIntegerField;

- points – сумарна кількість очок, в розіграші якої брав участь тенісист, є об'єктом класу BigIntegerField;
- points_match – кількість очок за поєдинок, є об'єктом класу DecimalField;
- sets_quantity – кількість сетів, проведених тенісистом, є об'єктом класу BigIntegerField;
- won_sets – кількість виграних тенісистом сетів, є об'єктом класу BigIntegerField;
- matches_quantity – кількість проведених тенісистом поєдинків, є об'єктом класу BigIntegerField;
- won_matches – відсоток виграних тенісистом поєдинків, є об'єктом класу BigIntegerField;
- upsets – кількість проведених тенісистом апсетів, є об'єктом класу BigIntegerField;
- won_upsets – відсоток виграних тенісистом апсетів, є об'єктом класу DecimalField;
- upsets_againts – кількість проведених тенісистом поєдинків, які були апсетами для суперника, є об'єктом класу BigIntegerField;
- lost_upsets_againts – відсоток програних тенісистом апсетів суперника, є об'єктом класу DecimalField.

Модель узагальненої статистики виступу тенісиста в сезоні турніру організована класом Tournament_season_overall_statistics за допомогою наступних атрибутів класу:

- tennis_player_id – тенісист, є зовнішнім ключем, який пов'язаний з моделлю Player, є об'єктом класу ForeignKey;
- overall_statistics_id – поєдинок, в якому відбувається сет, є зовнішнім ключем, який пов'язаний з моделлю є об'єктом класу ForeignKey Overall_statistics;
- tournament_season_id – сезон турніру, є зовнішнім ключем, який пов'язаний з моделлю Tournament_season, є об'єктом класу ForeignKey;

– `last_tour` – останній тур, в якому брав участь тенісист на турнірі даного сезону, є об'єктом класу `CharField`, значення якого можна обирати з переліченого вище набору варіантів.

Модель узагальненої статистики виступу тенісиста в турнірі за всі його сезони організована класом `Tournament_overall_statistics` за допомогою наступних атрибутів класу:

– `tennis_player_id` – тенісист, є зовнішнім ключем, який пов'язаний з моделлю `Player`, є об'єктом класу `ForeignKey`;

– `overall_statistics_id` – запис про узагальнену статистику, є зовнішнім ключем, який пов'язаний з моделлю `Overall_statistics`, є об'єктом класу `ForeignKey`;

– `tournament_id` – турнір, статистика до якого належить, є зовнішнім ключем, який пов'язаний з моделлю `Tournament`, є об'єктом класу `ForeignKey`;

– `titles` – кількість титулів, яку в даному турнірі здобув тенісист, є об'єктом класу `SmallIntegerField`;

– `times` – кількість сезонів, в якій в даному турнірі брав участь тенісист, є об'єктом класу `SmallIntegerField`;

– `best_tour` – найвищий тур, до якого за всі сезони доходив тенісист у даному турнірі, є об'єктом класу `CharField`;

– `won_matches_percentage` – відсоток виграних поєдинків тенісистом у всіх сезонах даного турніру, є об'єктом класу `DecimalField`.

Модель узагальненої статистики виступу тенісиста в усьому сезоні організована класом `Season_overall_statistics` за допомогою наступних атрибутів класу:

– `tennis_player_id` – тенісист, є зовнішнім ключем, який пов'язаний з моделлю `Player`, є об'єктом класу `ForeignKey`;

– `overall_statistics_id` – запис про узагальнену статистику, є зовнішнім ключем, який пов'язаний з моделлю `Overall_statistics`, є об'єктом класу `ForeignKey`;

- season_id – сезон, до якого належить статистика, є зовнішнім ключем, який пов'язаний з моделлю Season, є об'єктом класу ForeignKey;
- titles – кількість титулів, виграних тенісистом за даний сезон, є об'єктом класу SmallIntegerField;
- masters – кількість турнірів рівня мастерз, у яких переміг даний тенісист, є об'єктом класу SmallIntegerField;
- won_matches_percentage – відсоток виграних поєдинків у сезоні тенісистом, є об'єктом класу DecimalField.

Модель списку тенісистів користувача організована класом User_players за допомогою наступних атрибутів класу:

- user_id – користувач, якому належить список, є зовнішнім ключем, який пов'язаний з моделлю User, є об'єктом класу ForeignKey;
- tennis_player_id – тенісист, включений в список, є зовнішнім ключем, який пов'язаний з моделлю Player, є об'єктом класу ForeignKey.

Модель списку турнірів користувача організована класом User_tournaments за допомогою наступних атрибутів класу:

- user_id – користувач, якому належить список, є зовнішнім ключем, який пов'язаний з моделлю User, є об'єктом класу ForeignKey;
- tournament_id – турнір, включений до списку тенісиста, є зовнішнім ключем, який пов'язаний з моделлю Tournament, є об'єктом класу ForeignKey.

Модель списку пар протистоянь тенісистів, виділених користувачем, організована класом User_headtoheads за допомогою наступних атрибутів класу:

- user_id – користувач, якому належить список, є зовнішнім ключем, який пов'язаний з моделлю User, є об'єктом класу ForeignKey;
- tennis_player1_id – перший тенісист, включений в список, є зовнішнім ключем, який пов'язаний з моделлю Player, є об'єктом класу ForeignKey;

– `tennis_player2_id` – другий тенісист, включений в список, є зовнішнім ключем, який пов'язаний з моделлю `Player`, є об'єктом класу `ForeignKey`.

Представлення у свою чергу забезпечують взаємодію між моделями та шаблонами. У відповідності з даною логікою було розроблено набір класів і відповідних їм шаблонів для реалізації функціональних можливостей, які були визначені в вимогах до програми.

Підтримка роботи з інформацією про тенісистів реалізована через представлення `PlayerSearchView`, що забезпечує наповнення списку тенісистів за пошуковим запитом користувача. Отримані дані використовуються в роботі шаблону `search_tennis_player.html`. Він, як і інші шаблони, створені за допомогою шаблону [14]. Базовою моделлю є клас `Player`. Сортювання знайдених результатів відбувається за кількістю титулів, тобто найтитулованіших тенісистів буде знайти простіше ніж невідомих, якщо було введено частину імені, яка є достатньо популярною серед тенісистів.

Представлення `PlayerSeasonView`, `PlayerTournamentSeasonView` забезпечують перегляд статистики про виступ тенісиста в заданому сезоні та в заданому сезоні заданого турніру відповідно.

Представлення `PlayerRatingView` виконує формування списку найуспішніших 1000 тенісистів за заданим показником, значення якого прочитується через GET-запит:

```
spar = self.request.GET.get('criterion')
```

Представлення `PlayerCreate` забезпечує підтримку створення запису про нового тенісиста. Аналогічно забезпечується внесення нових поєдинків.

Представлення `PlayerUpdate` забезпечує підтримку оновлення інформації про тенісиста. Оновлення інформації про поєдинки, як було описано вище, реалізується автоматично на основі роботи користувача з

програмою через представлення `OverallStatisticsUpdate`, що надає можливість оновлення даних за допомогою внесення відповідних подій поєдинку.

3.3 Висновки за розділом 3

Описано основну послідовність, що надається для реалізації роботи над внесенням статистики і відповідно є основою функціонування програми для підтримки дій користувача. Описано основні реалізовані одиниці програми з детальним описом структури класів.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Опис програми

Робота зі статистичними показниками виступу спортсменів в різних видах спорту є важливою, оскільки на основі цієї інформації можна виконувати більш докладний аналіз, виконувати прогнози. Теніс є одним з найпопулярніших видів спорту, а тому такий інструмент для даного виду спорту важливий. Подібні програмні аналоги існують, проте вони не мають засобів індивідуальної роботи, що є важливим, оскільки статистика виступу за кожним поєдинком зазвичай представлена величезною кількістю показників, а якщо її ще помножити на кількість тенісистів, поєдинків, турнірів, сезонів, то серед всіх цих даних легко заплутатися. Тому потрібний зручний відповідний інструментарій. Саме на реалізацію даної функціональності і направлено роботу даної програми з забезпеченням доступу до статистики виступу тенісистів.

Для підтримки роботи програми необхідно, щоб апаратна частина включала процесор (4 ядра, не менше 2,1 ГГц), оперативну пам'ять (від 4 Гб), жорсткий диск з наявним вільним простором (від 500 Мб).

4.2 Експлуатація програми

Для експлуатації програми потрібно звернутися за URL-адресою, після чого відкриється вебсторінка, яка визначає роботу користувача без авторизації. Дана сторінка має набір пунктів меню, що включають:

- результати: можна переглянути результати поєдинків, відсортовані за датою, коли цей поєдинок відбувся (натискаючи на результат, можна перейти до вебсторінки перегляду детальної інформації, що включає зокрема статистику виступу кожного тенісиста в цьому поєдинку);

- турніри: можна обрати потрібний турнір зі списку, після цього відкриється сторінка з результатами останнього сезону турніру і списком

поєдинків, звідки можна обрати будь-який з інших сезонів, що відбулися раніше, з випадаючого списку;

- тенісисти: базується на виконанні пошуку, за результатами якого можна обрати потрібного тенісиста і переглянути інформацію по ньому (рис. 4.1);

- пари: порівняльна інформація по парам тенісистів:

- рейтинги: можна переглянути рейтинги тенісистів на окремі сторінці, для керування вибором конкретного рейтингу призначений випадаючий список в самому меню, останнім пунктом якого є показники (у такому випадку активується список вже на самій сторінці, звідки можна задати конкретний показник виступу тенісиста, щоб відсортувати тенісистів за узагальненим значенням даного показника).

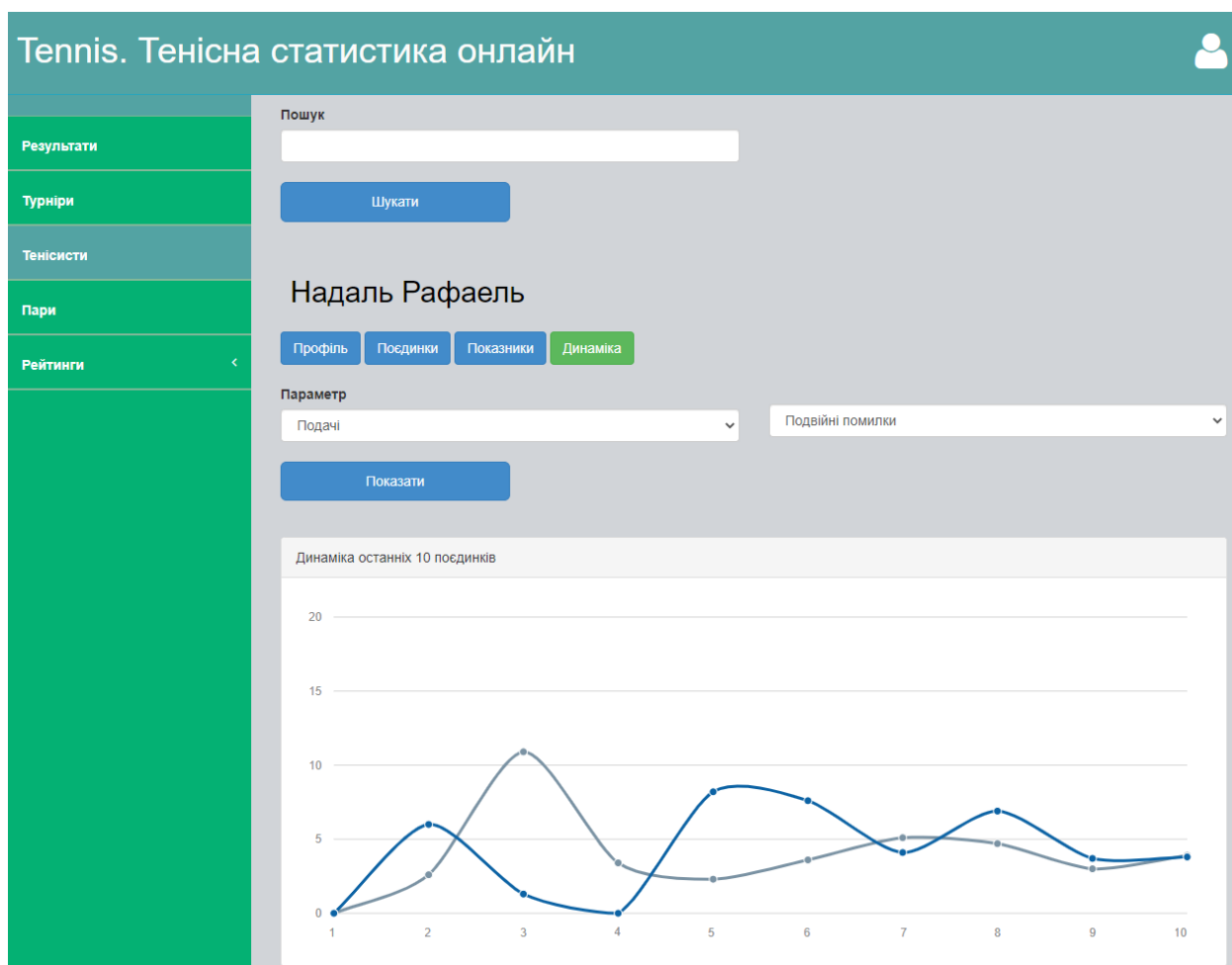


Рисунок 4.1 – Динаміка зміни показників тенісиста

За результатами пошуку тенісистів при натисканні на відповідне ім'я відкривається вебсторінка з полем для пошуку тенісистів та інформацією за обраним тенісистом. Ця інформація включає кнопки для вибору варіанту відображення. Кнопка профіль дозволяє переглянути інформацію про тенісиста. Кнопка поєдинки дозволяє переглянути список поєдинків тенісиста, відсортованих за датою, натискаючи на кожен з яких можна перейти до повної інформації за поєдинком. Кнопка показники дозволяє відтворити таблицю зі всіма сумарними значеннями всіх показників виступу тенісиста. Кнопка динаміка дозволяє відтворити графік з результатами 10 останніх поєдинків, при цьому з першого випадаючого списку можна задати групу параметрів, а з другого – конкретний показник. На графіку темнішим кольором відображається лінія самого тенісиста, а світлішим – його суперників. При наведенні на кожну точку виводиться інформація про те, кому яке значення відповідає.

Робота з парами тенісистів дозволяє спочатку виконати пошук потрібних тенісистів. Для цього надано два списки, які дозволяють вводити ім'я, а склад списку при цьому одразу коригується під введення. Після вибору потрібних тенісистів необхідно натиснути на кнопку “Обрати”, тоді нижче на сторінці з'явиться відповідна парна інформація (рис. 4.2). Вона включає узагальнену інформацію про кількість і результати парних поєдинків цих тенісистів, а також можливість вибору критерію порівняння з випадаючого списку.

У випадку вибору показників відображається порівняння за сумарними значеннями всіх статистичних показників виступу цих тенісистів, у випадку вибору поєдинків відображається список поєдинків, у випадку вибору динаміки відображається графік і додатковий список для вибору параметра, за яким будується графік у протистояннях між тенісистами.

У правому верхньому куті знаходиться піктограма, при натисканні на яку випадає список з можливістю авторизації (як у даному випадку), реєстрації або виходу (якщо користувач вже авторизувався).

Tennis. Тенісна статистика онлайн

Результати
Турніри
Тенісисти
Пари
Рейтинги

Пари тенісистів призначені для порівняння виступу тенісистів

Введіть прізвище першого тенісиста: Федерер Роджер

Введіть прізвище другого тенісиста: Надаль Рафаель

Обрати

40 поєдинків між суперниками

40 % поєдинків виграв Федерер Роджер

14 поєдинків турнірів Великого Шолому

19 поєдинків на турнірах серії Мастерз

Порівняння тенісистів

Федерер Роджер	Показник	Надаль Рафаель
11452	Ейси	3867
10.0	Відсоток ейсів	4.3
69.7	Виграні подачі	67.5
62.1	Перші подачі, %	68.1
77.4	Виграні перші подачі, %	72.2
84.1	Другі подачі, %	86.8
57.0	Виграні другі подачі, %	57.3

Рисунок 4.2 – Відображення пар протистоянь

У випадку авторизованого звичайного користувача його список в меню включає додатковий пункт списки, що містить випадаюче меню, серед пунктів якого тенісисти, турніри та пари тенісистів.

У випадку вибору списку тенісистів відображається набір кнопок, що включають:

- поєдинки (рис. 4.3), де відображається список поєдинків тенісистів, що входять до переліку користувача, відсортованих за датою, коли вони відбувались;
- показники, де відображається список показників виступу зі значеннями в таблиці для всіх тенісистів зі списку;
- динаміка, де відображається графік, аналогічний розглянутому вище, але для тенісистів зі списку;
- список, де міститься набір тенісистів, які було раніше додано: для кожного є можливість вилучити, а також поле пошуку для додавання нового тенісиста.

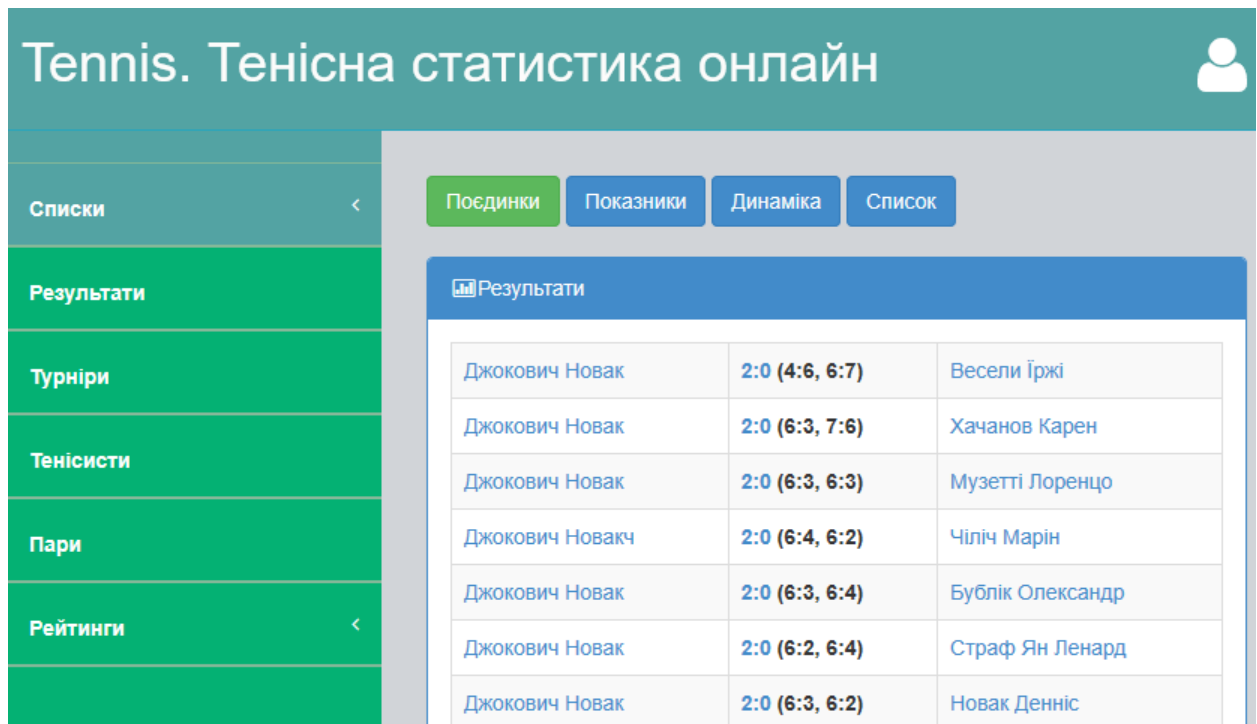


Рисунок 4.3 – Результати власного списку тенісистів

Якщо авторизований користувач є менеджером, то його меню включає наступні пункти:

- турніри, звідки можна додавати нові турніри, редагувати інформацію, відкривати потрібний, звідки додавати, редагувати сезони, а далі вже відкривати сторінку з потрібним сезоном турніру, звідки вносити поєдинки;

- тенісисти, звідки можна аналогічно вносити, редагувати інформацію про тенісистів (спочатку відображається пошук тенісиста з можливістю переходу до алфавітного відображення);

- рейтинги, звідки можна вносити положення тенісистів за рейтинговою системою певною, а також переглядати вже внесені (АТР, WTA).

У випадку відкриття сторінки незавершеного поєдинку менеджером (рис. 4.4) надається можливість введення загальної тривалості поєдинку, списку тестів, який формується автоматично за принципом сет – гейм – подія. Події, які вже було внесено за останнім розіграшом очок,

відображаються внизу таблиці. Таку подію можна вилучити, якщо інформація була внесена помилково. За кожним тенісистом можна ввести подолану ним відстань (вимагається ввести обов’язково тільки за введення останніх результатів поєдинку). Відображається пікова швидкість подачі, при цьому внизу сторінки при кожній подачі вводиться її швидкість при збереженні події. При цьому основний результат події повинен бути обраний один зі списку. Після вибору пункту внизу з додаткового набору можна також позначити, чи була виконана в процесі помилка тенісистом. Після введення всіх даних потрібно натиснути на кнопку “Зберегти”.

Турніри

Тенісисти

Рейтинги

Тенніс. Тенісна статистика онлайн

Джокович Новак - Пауль Томмі, 14.05.2022

Barcelona Open 1/16 фінала

Тривалість

Зберегти

1 сет	6:2	
2 сет	2:0	
1 гейм	40:15	
2 гейм	40:0	
3 гейм	15:0	
Подія	Аут	Вилучити

Джокович Новак

Відстань

Пікова швидкість подачі

180 км/год

☒ Подача

Результат події

☐ Ейс
☐ Аут
☐ Успішна подача
☐ Успішний розіграш
☐ Прогреш на розіграші
☐ Подвійна помилка
☐ Виграний нетпойнт
☐ Програний нетпойнт

Додатково

☐ Вимушена помилка при розіграші
☐ Невимушена помилка при розіграші
☐ Невимушена помилка на прийомі

Швидкість подачі

Зберегти

Пауль Томмі

Відстань

Пікова швидкість подачі

171 км/год

☐ Подача

Рисунок 4.4 – Внесення подій поєдинку менеджером

Рахунок, сети розраховуються автоматично. Також автоматично визначається той, хто виконує подачу. Якщо подачу все-таки виконує інший тенісист, то потрібно обрати відповідну позицію під прізвищем відповідного тенісиста.

Якщо немає необхідності вносити результати саме динамічно онлайн, то значення можна ввести через адміністративну панель (додати до адреси сайту admin), звідки можна ввести одразу загальні значення показників статистики поєдинку для кожного тенісиста.

4.3 Тестування програми

Тестування програми було здійснено за декількома напрямками.

Перший напрямок включав перевірку реалізації функціональних вимог програмою. Для цього спочатку було розроблено набір тестів, що зокрема включали додавання поєдинків, внесення даних тенісистів, турнірів, які перевірялись під час написання самої програми. Після завершення роботи було окремо перевірено повністю весь список функціональних вимог, підтвердивши працездатність.

Другий напрямок включав роботу з вебзастосунком через різні браузери. Перевірка включала браузери Chrome (зокрема версія 100.0.4896.88), Mozilla Firefox (зокрема версія 99), Microsoft Edge (зокрема версія 44.18362.1.0). Результати підтвердили, що програма є працездатною у всіх протестованих варіантах.

4.4 Висновки за розділом 4

Програма працює в режимах неавторизованого користувача, авторизованого користувача, менеджера, кожен з яких було описано. Виконано успішне тестування програми.

ВИСНОВКИ

Виконано аналіз предметної області, в межах якого було виділено основні поняття і показники, які стосуються статистики індивідуальних виступів тенісистів, проведено огляд програмних аналогів, включаючи вебсайти Ultimate Tennis Statistics, Tennis Explorer, Tennis Abstract. Враховуючи в них відсутність можливості швидкого доступу до потрібних тенісистів, саме дана можливість було визначено в якості особливості розробленої програми.

Розробку програмного забезпечення надання тенісної статистики онлайн виконано за допомогою вебфреймворку Django з визначенням структури даних на основі використання бази даних, вимог до програми, розробленням структури програми і алгоритму роботи програми на основі цих вимог, реалізацією відповідних одиниць вебзастосунку. Для розробки використано мови програмування Python, JavaScript, мову розмітки HTML, мову стилю сторінок CSS, вебфреймворк Django, систему керування базами даних MySQL.

Програма дозволяє переглядати результати тенісних змагань, окремих поєдинків тенісистів, показники виступу тенісистів у окремих поєдинках, а також накопичену статистику за весь турнір, за всі сезони турніру, за весь сезон, а також за всю кар'єру, переглядати порівняні показники виступу пар тенісистів, історію їх протистоянь, виконувати пошук тенісистів, створювати власні списки тенісистів, турнірів, пар тенісистів для надання швидкого доступу.

Розроблене програмне забезпечення дозволяє підвищити зручність роботи зі статистикою виступу тенісистів на основі індивідуального відслідковування. Таке рішення у свою чергу дозволяє привертати увагу користувачів і потенційно заробляти за допомогою реклами на основі створеної бази користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Czermak, C. Tennis Popularity Statistics 2021 [Electronic resource] / C. Czermak. – Access mode : <https://tenniscreative.com/tennis-popularity-statistics/>.
2. Sprouse, M. Why Tennis Continues to Grow (And Will It Last) [Electronic resource] / M. Sprouse. – Access mode : <https://www.tennisct.com/why-tennis-continues-to-grow>.
3. Tennis Terms and Words To Know [Electronic resource]. – Access mode : <https://www.usta.com/en/home/improve/tips-and-instruction/national/tennis-terms-definitions.html>.
4. Effect of a Seeding System on Competitive Performance of Elite Players During Major Tennis Tournaments [Electronic resource] / Yixiong Cui, Yue Zhao, Haoyang Liu, Miguel-Angel Gomez, Ran Wei, Yuanlong Liu. – Access mode : <https://www.frontiersin.org/articles/10.3389/fpsyg.2020.01294/full>.
5. Matchplay characteristics and performance indicators of male junior and entry professional tennis players [Electronic resource] / Ales Filipcic, Bojan Leskosek, Miguel Crespo, Tjasa Filipcic. – Access mode : <https://journals.sagepub.com/doi/full/10.1177/1747954120988002>.
6. Joint Performance Indicators In Tennis [Text] / M. Khalili, M. Osaileh, A. Kuliabin, O. Abdelfattah // Dirasat : Educational Sciences. – 2020. – Vol. 47, Issue 3. – Pp. 489-508.
7. Rankings. Singles. ATP Tour. Tennis [Electronic resource]. – Access mode : <https://www.atptour.com/en/rankings/singles>.
8. The Official Home of the Women's Tennis Association [Electronic resource]. – Access mode : <https://www.wtatennis.com/>
9. Ultimate Tennis Statistics [Electronic resource]. – Access mode : <https://www.ultimatetennisstatistics.com/>.
10. Tennis Explorer [Electronic resource] : Tennis Stats, WTA & ATP Tennis Betting. – Access mode : <https://www.tennisexplorer.com/>.

11. Tennis Abstract [Electronic resource] : ATP and WTA Match Results, Splits, and Analysis. – Access mode : <http://www.tennisabstract.com/>.
12. Welcome to Flask – Flask Documentation (2.1.x) [Electronic resource]. – Access mode : <https://flask.palletsprojects.com/en/2.1.x/>.
13. The web framework for perfectionists with deadlines. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/>.
14. SIMINTA Free Website Template [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page197/siminta>.

ДОДАТОК А
Технічне завдання

Вступ

Робота зі статистичними показниками виступу спортсменів в різних видах спорту є важливою, оскільки на основі цієї інформації можна виконувати більш докладний аналіз, виконувати прогнози. Теніс є одним з найпопулярніших видів спорту, а тому такий інструмент для даного виду спорту важливий. Подібні програмні аналоги існують, проте вони не мають засобів індивідуальної роботи, що є важливим, оскільки статистика виступу за кожним поєдинком зазвичай представлена величезною кількістю показників, то серед всіх цих даних легко заплутатися. Тому потрібний зручний відповідний інструментарій.

А.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується на основі завдання, виданого за темою «Програмне забезпечення надання тенісної статистики онлайн» у відповідності з наказом № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма має використовуватися для отримання доступу до результатів і статистичних показників виступу тенісистів онлайн з виділенням інформації про тих тенісистів, які цікавлять користувача.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Набір функціональних вимог, які має забезпечувати програма, включає:

- пошук тенісистів;
- перегляд турнірів та їх сезонів;
- перегляд тенісистів за алфавітом;
- внесення та редагування даних тенісистів;
- автоматичний розрахунок показників виступу тенісиста за сезон турніру, турнір, сезон та за всю кар'єру;
- внесення та редагування даних турнірів;
- внесення та перегляд поєдинків турнірів;
- внесення подій поєдинку;
- внесення рейтингів тенісистів;
- внесення сезонів турнірів;
- відображення динаміки зміни виступу тенісиста за обраним параметром;
- відображення порівняльної динаміки зміни виступу двох тенісистів за обраним параметром;
- перегляд результатів турніру;
- перегляд порівняльної статистики обраних тенісистів;
- перегляд історії матчів між обраними тенісистами;
- перегляд рейтингів тенісистів за обраною датою;
- перегляд усіх поєдинків тенісиста;
- перегляд поєдинків тенісиста за обраним турніром, за сезоном, за сезоном турніру;
- перегляд детальної статистики за обраним поєдинком за загальними і індивідуальними показниками з розділенням за групами;
- перегляд детальних результатів обраного поєдинку;
- відображення списків тенісистів, упорядкованих за значеннями індивідуальних показників;
- перегляд інформації про тенісиста;
- перегляд статистики виступу тенісиста проти інших тенісистів;
- перегляд історії місць тенісиста у світових рейтингах;

- авторизація і реєстрація користувачів;
- створення і редагування списків відслідковування тенісистів користувачем;
- створення і редагування списків відслідковування результатів турнірів користувачем;
- створення і редагування списків відслідковування пар протистоянь користувачем.

А.3.2 Умови експлуатації

Умовами експлуатації є виділення менеджера або менеджерів для реалізації внесення даних в систему. Менеджер може також виконувати роль адміністратора, але краще виділити для цього окремо системного адміністратора, який також буде керувати базою даних.

А.3.3 Вимоги до складу та параметрів технічних засобів

Для підтримки роботи програми необхідно, щоб апаратна частина включала процесор (4 ядра, не менше 2,1 ГГц), оперативну пам'ять (від 4 Гб), жорсткий диск з наявним вільним простором (від 500 Мб).

А.4 Порядок контролю та приймання

Дипломна кваліфікаційна робота має виконуватися у відповідності з визначеним календарним планом згідно встановлених завдань, їх строків і результатів виконання.

ДОДАТОК Б
Текст програми

Б.1 Текст файла models.py

```

from django.db import models
from django.contrib.auth.models import User
from django.urls import reverse
from django.utils import timezone
from django.db.models.signals import post_init
from django.conf import settings
import datetime

TOURS = (
    ('A', 'Фінал'),
    ('B', 'За 3 місце'),
    ('C', 'Півфінал'),
    ('D', 'Чвертьфінал'),
    ('E', '1/8'),
    ('F', '1/16'),
    ('G', '1/32'),
    ('H', '1/64'),
    ('I', '1/128'),
    ('J', 'Кваліфікація')
)

COVERS = (
    ('G', 'Трава'),
    ('H', 'Хард'),
    ('C', 'Грунт'),
    ('R', 'Коврове')
)

LEVELS = (
    ('A', 'Великий шолом'),
    ('B', 'Мастерз'),
    ('C', 'Олімпіада'),
    ('D', 'АТР 250'),
    ('E', 'АТР 500')
)

class Player (models.Model):
    full_name = models.CharField(max_length=100)
    country_id = models.ForeignKey(Country, related_name='info',
on_delete=models.CASCADE)
    overall_statistics_id = models.OneToOneField(Overall_statistics,
on_delete=models.CASCADE, blank=True)
    birth_date = models.DateField(blank=True)
    height = models.SmallIntegerField(blank=True)
    weight = models.SmallIntegerField(blank=True)
    prizes = models.BigIntegerField(default = 0)
    activeness = models.BooleanField(default = True)
    start_season = models.ForeignKey(Season, on_delete=models.CASCADE,
blank=True)
    end_season = models.ForeignKey(Season, on_delete=models.CASCADE,
blank=True)
    titles = models.SmallIntegerField(default = 0)
    residence = models.CharField(max_length=100)
    grand_slams = models.SmallIntegerField(default = 0)
    masters = models.SmallIntegerField(default = 0)
    main_arm = models.BooleanField(default = True)
    played_arm = models.BooleanField(default = False)

```

```

    def __str__(self):
        return self.full_name

    def get_absolute_url(self):
        return reverse('player-detail', kwargs={'pk': self.pk})

class Country (models.Model):
    name = models.CharField(max_length=50)

    def get_absolute_url(self):
        return reverse('season-detail', kwargs={'pk': self.pk})

class Tournament (models.Model):
    name = models.CharField(max_length=25)
    cover = models.CharField(max_length=1, choices= COVERS)
    level = models.CharField(max_length=1, choices= LEVELS)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('tournament-detail', kwargs={'pk': self.pk})

class Season (models.Model):
    name = models.CharField(max_length=4)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('season-detail', kwargs={'pk': self.pk})

class Tournament_season (models.Model):
    tournament_id = models.ForeignKey(Tournament, related_name='info',
on_delete=models.CASCADE)
    season_id = models.ForeignKey(Season, related_name='infom',
on_delete=models.CASCADE)
    winner = models.ForeignKey(Player, related_name='infow',
on_delete=models.CASCADE)

class Match (models.Model):
    tournament_season_id = models.ForeignKey(Tournament_season,
related_name='info', on_delete=models.CASCADE)
    tennis_player1 = models.ForeignKey(Player, related_name='infop1',
on_delete=models.CASCADE)
    tennis_player2 = models.ForeignKey(Player, related_name='infop2',
on_delete=models.CASCADE)
    match_date = models.DateTimeField(default=timezone.now)
    tour = models.CharField(max_length=1, choices=TOURS)
    duration = models.TimeField(blank = True)
    result1 = models.SmallIntegerField(blank = True)
    result2 = models.SmallIntegerField(blank = True)
    match_sets = models.SmallIntegerField(blank = True)
    points = models.SmallIntegerField(blank = True)
    state = models.BooleanField(default = False)
    winner = models.BooleanField(blank = True)

    def __str__(self):
        return self.tennis_player1.full_name + ' - ' +
self.tennis_player2.full_name

    def get_absolute_url(self):

```

```

        return reverse('match-detail', kwargs={'pk': self.pk})

    def additionalMatch(instance, *args, **kwargs):
        m = Match_set.objects.create(match_id=instance.id)
        m.save()
        Match_game.objects.create(match_set=m.id)
        p1 = Player_match_statistics.objects.create(match_id=m.id,
        tennis_player_id=instance.tennis_player1)
        p2 = Player_match_statistics.objects.create(match_id=m.id,
        tennis_player_id=instance.tennis_player2)
        tenp = Rating.objects.filter(tennis_player_id =
        instance.tennis_player1).order_by('-rating_date')[0]
        tenp2 = Rating.objects.filter(tennis_player_id =
        instance.tennis_player2).order_by('-rating_date')[0]
        p1.save()
        p2.save()
        if tenp.place < tenp2.place:
            p1.unset = True
            p2.unset = False
        elif tenp.place > tenp2.place:
            p2.unset = True
            p1.unset = False
        p1.save()
        p2.save()
        try:
            go =
            Tournament_season_overall_statistics.objects.get(tennis_player_id=instance.te
            nnis_player1, tournament_season_id=instance.tournament_season_id)
        except Tournament_season_overall_statistics.DoesNotExist:
            o = Overall_statistics.create()
            o.save()

        Tournament_season_overall_statistics.create(tennis_player_id=instance.tennis_p
        layer1, tournament_season_id=instance.tournament_season_id, overall_statistics_id =
        o)

        try:
            go =
            Tournament_season_overall_statistics.objects.get(tennis_player_id=instance.te
            nnis_player2, tournament_season_id=instance.tournament_season_id)
        except Tournament_season_overall_statistics.DoesNotExist:
            o = Overall_statistics.create()
            o.save()

        Tournament_season_overall_statistics.create(tennis_player_id=instance.tennis_p
        layer2, tournament_season_id=instance.tournament_season_id, overall_statistics_id =
        o)

        try:
            go =
            Tournament_overall_statistics.objects.get(tennis_player_id=instance.tennis_pl
            ayer1, tournament_id=instance.tournament_season_id.tournament_id)
        except Tournament_overall_statistics.DoesNotExist:
            o = Overall_statistics.create()
            o.save()

        Tournament_season_overall_statistics.create(tennis_player_id=instance.tennis_p
        layer1, tournament_id=instance.tournament_season_id.tournament_id,
        overall_statistics_id = o)

        try:
            go =
            Tournament_overall_statistics.objects.get(tennis_player_id=instance.tennis_pl
            ayer2, tournament_id=instance.tournament_season_id.tournament_id)
        except Tournament_overall_statistics.DoesNotExist:

```

```

        o = Overall_statistics.create()
        o.save()

    Tournament_season_overall_statistics.create(tennis_player_id=instance.tennis_p
layer2, tournament_id=instance.tournament_season_id.tournament_id,
overall_statistics_id = o)
    try:
        go =
    Season_overall_statistics.objects.get(tennis_player_id=instance.tennis_player
1, season_id=instance.tournament_season_id.season_id)
    except Season_overall_statistics.DoesNotExist:
        o = Overall_statistics.create()
        o.save()

    Season_overall_statistics.create(tennis_player_id=instance.tennis_player1,
season_id=instance.tournament_season_id.season_id, overall_statistics_id = o)
    try:
        go =
    Season_overall_statistics.objects.get(tennis_player_id=instance.tennis_player
2, season_id=instance.tournament_season_id.season_id)
    except Season_overall_statistics.DoesNotExist:
        o = Overall_statistics.create()
        o.save()

    Season_overall_statistics.create(tennis_player_id=instance.tennis_player2,
season_id=instance.tournament_season_id.season_id, overall_statistics_id = o)

    post_init.connect(additionalMatch, Match)

    class Match_set (models.Model):
        match_id = models.ForeignKey(Match, related_name='info',
on_delete=models.CASCADE)
        num = models.SmallIntegerField(default = 1)
        games1 = models.SmallIntegerField(default = 0)
        games2 = models.SmallIntegerField(default = 0)

        def __str__(self):
            return str(self.games1) + ':' + str(self.games2)

    class Match_game (models.Model):
        match_set = models.ForeignKey(Match_set, related_name='info',
on_delete=models.CASCADE)
        num = models.SmallIntegerField(default = 1)
        game1 = models.SmallIntegerField(blank = True)
        game2 = models.SmallIntegerField(blank = True)
        tiebreak = models.BooleanField(default = False)

        def __str__(self):
            return str(self.games1) + ':' + str(self.games2)

        def save(self, *args, **kwargs):
            m = Match.objects.get(id=matchset.match_id.id)
            m.points = m.points + 1
            pms1 = Player_match_statistics.objects.get(match_id = m,
tennis_player_id = self.match_set.match_id.tennis_player1)
            pms2 = Player_match_statistics.objects.get(match_id = m,
tennis_player_id = self.match_set.match_id.tennis_player2)
            pms1.points_set = m.points / (m.result1 + m.result2)
            pms2.points_set = m.points / (m.result1 + m.result2)
            if not self.tiebreak and self.game1 == 60:
                matchset = Match_set.objects.get(id=self.match_set)
                matchset.games1 = matchset.games1 + 1

```

```

        matchset.save()
        if (matchset.games1 - matchset.games2) == 2 and
matchset.games1 >= 6:
            m.result1 = m.result1 + 1
            if not (((m.result1 + m.result2 == 3) and (m.result1
== 2 or m.result2 == 2)) or ((m.result1 + m.result2 == 5)
and (m.result1 == 3 or m.result2 == 3))):
                m =
Match_set.objects.create(self.match_set.match_id.id)
            Match_game.objects.create(self.match_set=m)
        elif not self.tiebreak and self.game2 == 60:
            matchset = Match_set.objects.get(id=self.match_set)
            matchset.games2 = matchset.games2 + 1
            matchset.save()
            if (matchset.games2 - matchset.games1) == 2 and
matchset.games2 >= 6:
                m.result2 = m.result2 + 1
                if not (((m.result1 + m.result2 == 3) and (m.result1
== 2 or m.result2 == 2)) or ((m.result1 + m.result2 == 5)
and (m.result1 == 3 or m.result2 == 3))):
                    m =
Match_set.objects.create(self.match_set.match_id.id)
                Match_game.objects.create(self.match_set=m)
        elif self.tiebreak and self.game1 >= 7 and (self.game1 -
self.game2) == 2:
            pms1.won_tiebreaks = pms1.won_tiebreaks + 1
            pms1.tiebreaks = pms1.tiebreaks + 1
            pms2.tiebreaks = pms2.tiebreaks + 1
            matchset = Match_set.objects.get(id=self.match_set)
            matchset.games1 = matchset.games1 + 1
            matchset.save()
            m.result1 = m.result1 + 1
            if not (((m.result1 + m.result2 == 3) and (m.result1 == 2
or m.result2 == 2)) or ((m.result1 + m.result2 == 5) and
(m.result1 == 3 or m.result2 == 3))):
                m =
Match_set.objects.create(self.match_set.match_id.id)
            Match_game.objects.create(self.match_set=m)
        elif self.tiebreak and self.game2 >= 7 and (self.game2 -
self.game1) == 2:
            pms2.won_tiebreaks = pms2.won_tiebreaks + 1
            pms1.tiebreaks = pms1.tiebreaks + 1
            pms2.tiebreaks = pms2.tiebreaks + 1
            matchset = Match_set.objects.get(id=self.match_set)
            matchset.games2 = matchset.games2 + 1
            matchset.save()
            m.result2 = m.result2 + 1
            if not (((m.result1 + m.result2 == 3) and (m.result1 == 2
or m.result2 == 2)) or ((m.result1 + m.result2 == 5) and
(m.result1 == 3 or m.result2 == 3))):
                m =
Match_set.objects.create(self.match_set.match_id.id)
            Match_game.objects.create(self.match_set=m)
        q1 = 0
        for on in mm:
            if not on.tiebreak and on.game1 == 15:
                q1 = q1 + 1
            elif not on.tiebreak and on.game1 == 30:
                q1 = q1 + 2
            elif not on.tiebreak and on.game1 == 40:
                q1 = q1 + 3
            elif not on.tiebreak and on.game1 == 50:

```

```

        q1 = q1 + 4
    elif not on.tiebreak and on.game1 == 60:
        q1 = q1 + 5
    elif on.tiebreak:
        q1 = q1 + on.game1
    pms1.won_points = q1
    pms2.won_points = m.points - q1
    pms1.won_points_percantage = pms1.won_points / m.points
    pms2.won_points_percantage = pms2.won_points / m.points
    if (m.result1 + m.result2 == 3) and (m.result1 == 2 or m.result2
== 2):
        m.state = True
        mm = Match_game.objects.filter(match_set__match_id =
self.match_set.match_id)
        if m.result1 == 2:
            m.winner = False
        else:
            m.winner = True
        if m.tour == 'A':
            pl = Player.objects.get(id=m.tennis_player2.id)
            pl.titles = pl.titles + 1
            if m.tournament_season_id.tournament_id.level ==
'A':
                pl.grand_slams = pl.grand_slams + 1
            elif m.tournament_season_id.tournament_id.level ==
'B':
                pl.masters = pl.masters + 1
            ts =
Tournament_season.objects.get(id=m.tournament_season_id)
            ts.winner = pl
        elif (m.result1 + m.result2 == 5) and (m.result1 == 3 or
m.result2 == 3):
            m.state = True
            if m.result1 == 3:
                m.winner = False
            else:
                m.winner = True
            if m.tour == 'A':
                pl = Player.objects.get(id=m.tennis_player2.id)
                pl.titles = pl.titles + 1
                if m.tournament_season_id.tournament_id.level ==
'A':
                    pl.grand_slams = pl.grand_slams + 1
                elif m.tournament_season_id.tournament_id.level ==
'B':
                    pl.masters = pl.masters + 1
                ts =
Tournament_season.objects.get(id=m.tournament_season_id)
                ts.winner = pl
            m.save()
            pms1.save()
            pms2.save()
            super(Match_game, self).save(*args, **kwargs)

class Player_match_statistics (models.Model):
    match_id = models.ForeignKey(Match, related_name='infot',
on_delete=models.CASCADE)
    tennis_player_id = models.ForeignKey(Player, related_name='infot',
on_delete=models.CASCADE)
    aces = models.SmallIntegerField(default = 0, blank = True)
    aces_percentage = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)

```

```

        win_service = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        first_service = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        first_service_win = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        second_service = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        second_service_win = models.DecimalField(max_digits = 3, decimal_places
= 1, default = 0, blank = True)
        double_faults = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        max_service_speed = models.SmallIntegerField(default = 0, blank = True)
        return_service_won = models.DecimalField(max_digits = 3, decimal_places
= 1, default = 0, blank = True)
        return_unforced_errors = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        first_return_service_won = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        second_return_service_won = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        won_netpoints = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        won_netpoints_percentage = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        return_breakpoints = models.SmallIntegerField(default = 0, blank = True)
        won_breakpoints = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        saved_breakpoints = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        won_rally = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        forced_errors_rally = models.DecimalField(max_digits = 3, decimal_places
= 1, default = 0, blank = True)
        unforced_errors_rally = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        dominance_ratio = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        distance = models.SmallIntegerField(default = 0, blank = True)
        ace_double_errors = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        first_second_speed = models.DecimalField(max_digits = 3, decimal_places
= 1, default = 0, blank = True)
        max_average_speed = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        breakpoints = models.SmallIntegerField(default = 0, blank = True)
        breakpoints_match = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        won_points = models.SmallIntegerField(default = 0, blank = True)
        won_points_percentage = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        points_set = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        tiebreaks = models.SmallIntegerField(default = 0, blank = True)
        won_tiebreaks = models.SmallIntegerField(default = 0, blank = True)
        upset = models.BooleanField(default = False)

    def save(self, *args, **kwargs):
        if self.match_id.state:
            ts =
Tournament_season_overall_statistics.objects.get(tennis_player_id=self.tennis_player_
id, tournament_season_id=self.match_id.tournament_season_id)

```

```

        data = ts.overall_statistics_id
        sts = Player_match_statistics.objects.filter(tennis_player_id=
self.tennis_player_id,
match_id__tournament_season_id=self.match_id.tournament_season_id)
        data.aces = sum(sts.values_list('aces'))
        data.aces_percentage = sum(sts.values_list('aces_percentage')) /
len(sts)
        data.win_service = sum(sts.values_list('win_service')) / len(sts)
        data.first_service = sum(sts.values_list('first_service')) /
len(sts)
        data.first_service_win =
sum(sts.values_list('first_service_win')) / len(sts)
        data.second_service = sum(sts.values_list('second_service')) /
len(sts)
        data.second_service_win =
sum(sts.values_list('second_service_win')) / len(sts)
        data.double_faults = sum(sts.values_list('double_faults')) /
len(sts)
        data.max_service_speed =
max(sts.values_list('max_service_speed'))
        data.return_service_won =
sum(sts.values_list('return_service_won')) / len(sts)
        data.return_unforced_errors =
sum(sts.values_list('return_unforced_errors')) / len(sts)
        data.first_return_service_won =
sum(sts.values_list('first_return_service_won')) / len(sts)
        data.second_return_service_won =
sum(sts.values_list('second_return_service_won')) / len(sts)
        data.won_netpoints = sum(sts.values_list('won_netpoints')) /
len(sts)
        data.won_netpoints_percentage =
sum(sts.values_list('won_netpoints_percentage')) / len(sts)
        data.return_breakpoints =
sum(sts.values_list('return_breakpoints'))
        data.won_breakpoints = sum(sts.values_list('won_breakpoints')) /
len(sts)
        data.saved_breakpoints =
sum(sts.values_list('saved_breakpoints')) / len(sts)
        data.won_rally = sum(sts.values_list('won_rally')) / len(sts)
        data.forced_errors_rally =
sum(sts.values_list('forced_errors_rally')) / len(sts)
        data.unforced_errors_rally =
sum(sts.values_list('unforced_errors_rally')) / len(sts)
        data.dominance_ratio = sum(sts.values_list('dominance_ratio')) /
len(sts)
        data.distance = sum(sts.values_list('distance'))
        data.ace_double_errors =
sum(sts.values_list('ace_double_errors')) / len(sts)
        data.first_second_speed =
sum(sts.values_list('first_second_speed')) / len(sts)
        data.max_average_speed =
max(sts.values_list('max_average_speed'))
        data.breakpoints = sum(sts.values_list('breakpoints'))
        data.breakpoints_match =
sum(sts.values_list('breakpoints_match')) / len(sts)
        data.won_points = sum(sts.values_list('won_points'))
        data.won_points_percentage =
sum(sts.values_list('won_points_percentage')) / len(sts)
        data.points_set = sum(sts.values_list('points_set')) / len(sts)
        data.tiebreaks = sum(sts.values_list('tiebreaks'))
        data.won_tiebreaks = sum(sts.values_list('won_tiebreaks'))

```

```

        data.points = sum(Match.objects.filter(tournament_season_id =
self.match_id.tournament_season_id, tennis_player1 = self.tennis_player_id |
tennis_player2 = self.tennis_player_id).values_list('points'))
        data.matches_quantity =
sum(Match.objects.filter(tournament_season_id = self.match_id.tournament_season_id,
tennis_player1 = self.tennis_player_id | tennis_player2 = self.tennis_player_id))
        r1 = Match.objects.filter(tournament_season_id =
self.match_id.tournament_season_id, tennis_player1 = self.tennis_player_id)
        r2 = Match.objects.filter(tournament_season_id =
self.match_id.tournament_season_id, tennis_player2 = self.tennis_player_id)
        data.won_matches = len([x for x in r1 if x.result1 > x.result2])
+ len([x for x in r2 if x.result2 > x.result1])
        data.points_match = data.won_points / data.matches_quantity
        data.sets_quantity =
sum(Match.objects.filter(tournament_season_id = self.match_id.tournament_season_id,
tennis_player1 = self.tennis_player_id | tennis_player2 =
self.tennis_player_id).values_list('result1')) +
sum(Match.objects.filter(tournament_season_id = self.match_id.tournament_season_id,
tennis_player1 = self.tennis_player_id | tennis_player2 =
self.tennis_player_id).values_list('result2'))
        s1 = Match_set.objects.filter(match_id__tournament_season_id=
self.match_id.tournament_season_id, tennis_player1= self.tennis_player_id)
        s2 = Match_set.objects.filter(match_id__tournament_season_id=
self.match_id.tournament_season_id, tennis_player2= self.tennis_player_id)
        data.won_sets = len([x for x in s1 if x.games1 > x.games2]) +
len([x for x in s2 if x.games2 > x.games1])
        data.upsets = len([x for x in r1 if x.upset]) + len([x for x in
r2 if not x.upset])
        data.won_upsets = len([x for x in r1 if x.upset and x.result1 >
x.result2]) + len([x for x in r2 if not x.upset and x.result2 > x.result1])
        data.upsets_againts = len([x for x in r1 if not x.upset]) +
len([x for x in r2 if x.upset])
        data.lost_upsets_againts = len([x for x in r1 if not x.upset and
x.result1 > x.result2]) + len([x for x in r2 if x.upset and x.result2 > x.result1])
        data.save()
        to =
Tournament_overall_statistics.objects.get(tennis_player_id=self.tennis_player_id,
tournament_id=self.match_id.tournament_season_id.tournament_id)
        data = to.overall_statistics_id
        sts =
Tournament_season_overall_statistics.objects.filter(tennis_player_id=
self.tennis_player_id, tournament_season_id__tournament_id
=self.match_id.tournament_season_id.tournament_id)
        data.aces = sum(sts.values_list('aces'))
        data.aces_percentage = sum(sts.values_list('aces_percentage')) /
len(sts)
        data.win_service = sum(sts.values_list('win_service')) / len(sts)
        data.first_service = sum(sts.values_list('first_service')) /
len(sts)
        data.first_service_win =
sum(sts.values_list('first_service_win')) / len(sts)
        data.second_service = sum(sts.values_list('second_service')) /
len(sts)
        data.second_service_win =
sum(sts.values_list('second_service_win')) / len(sts)
        data.double_faults = sum(sts.values_list('double_faults')) /
len(sts)
        data.max_service_speed =
max(sts.values_list('max_service_speed'))
        data.return_service_won =
sum(sts.values_list('return_service_won')) / len(sts)

```

```

        data.return_unforced_errors =
sum(sts.values_list('return_unforced_errors')) / len(sts)
        data.first_return_service_won =
sum(sts.values_list('first_return_service_won')) / len(sts)
        data.second_return_service_won =
sum(sts.values_list('second_return_service_won')) / len(sts)
        data.won_netpoints = sum(sts.values_list('won_netpoints')) /
len(sts)
        data.won_netpoints_percentage =
sum(sts.values_list('won_netpoints_percentage')) / len(sts)
        data.return_breakpoints =
sum(sts.values_list('return_breakpoints'))
        data.won_breakpoints = sum(sts.values_list('won_breakpoints')) /
len(sts)
        data.saved_breakpoints =
sum(sts.values_list('saved_breakpoints')) / len(sts)
        data.won_rally = sum(sts.values_list('won_rally')) / len(sts)
        data.forced_errors_rally =
sum(sts.values_list('forced_errors_rally')) / len(sts)
        data.unforced_errors_rally =
sum(sts.values_list('unforced_errors_rally')) / len(sts)
        data.dominance_ratio = sum(sts.values_list('dominance_ratio')) /
len(sts)
        data.distance = sum(sts.values_list('distance'))
        data.ace_double_errors =
sum(sts.values_list('ace_double_errors')) / len(sts)
        data.first_second_speed =
sum(sts.values_list('first_second_speed')) / len(sts)
        data.max_average_speed =
max(sts.values_list('max_average_speed'))
        data.breakpoints = sum(sts.values_list('breakpoints'))
        data.breakpoints_match =
sum(sts.values_list('breakpoints_match')) / len(sts)
        data.won_points = sum(sts.values_list('won_points'))
        data.won_points_percentage =
sum(sts.values_list('won_points_percentage')) / len(sts)
        data.points_set = sum(sts.values_list('points_set')) / len(sts)
        data.tiebreaks = sum(sts.values_list('tiebreaks'))
        data.won_tiebreaks = sum(sts.values_list('won_tiebreaks'))
        data.points = sum(sts.values_list('points'))
        data.matches_quantity = sum(sts.values_list('matches_quantity'))
        data.won_matches = sum(sts.values_list('won_matches'))
        data.points_match = sum(sts.values_list('points_match')) /
len(sts)
        data.sets_quantity = sum(sts.values_list('sets_quantity'))
        data.won_sets = sum(sts.values_list('won_sets'))
        data.upsets = sum(sts.values_list('upsets'))
        data.won_upsets = sum(sts.values_list('won_upsets'))
        data.upsets_againts = sum(sts.values_list('upsets_againts'))
        data.lost_upsets_againts =
sum(sts.values_list('lost_upsets_againts'))
        data.save()
        so =
Season_overall_statistics.objects.get(tennis_player_id=self.tennis_player_id,
season_id=self.match_id.tournament_season_id.season_id)
        data = so.overall_statistics_id
        sts =
Tournament_season_overall_statistics.objects.filter(tennis_player_id=
self.tennis_player_id, tournament_season_id__season_id
=self.match_id.tournament_season_id.season_id)
        data.aces = sum(sts.values_list('aces'))

```

```

data.aces_percentage = sum(sts.values_list('aces_percentage')) /
len(sts)
data.win_service = sum(sts.values_list('win_service')) / len(sts)
data.first_service = sum(sts.values_list('first_service')) /
len(sts)
data.first_service_win =
sum(sts.values_list('first_service_win')) / len(sts)
data.second_service = sum(sts.values_list('second_service')) /
len(sts)
data.second_service_win =
sum(sts.values_list('second_service_win')) / len(sts)
data.double_faults = sum(sts.values_list('double_faults')) /
len(sts)
data.max_service_speed =
max(sts.values_list('max_service_speed'))
data.return_service_won =
sum(sts.values_list('return_service_won')) / len(sts)
data.return_unforced_errors =
sum(sts.values_list('return_unforced_errors')) / len(sts)
data.first_return_service_won =
sum(sts.values_list('first_return_service_won')) / len(sts)
data.second_return_service_won =
sum(sts.values_list('second_return_service_won')) / len(sts)
data.won_netpoints = sum(sts.values_list('won_netpoints')) /
len(sts)
data.won_netpoints_percentage =
sum(sts.values_list('won_netpoints_percentage')) / len(sts)
data.return_breakpoints =
sum(sts.values_list('return_breakpoints'))
data.won_breakpoints = sum(sts.values_list('won_breakpoints')) /
len(sts)
data.saved_breakpoints =
sum(sts.values_list('saved_breakpoints')) / len(sts)
data.won_rally = sum(sts.values_list('won_rally')) / len(sts)
data.forced_errors_rally =
sum(sts.values_list('forced_errors_rally')) / len(sts)
data.unforced_errors_rally =
sum(sts.values_list('unforced_errors_rally')) / len(sts)
data.dominance_ratio = sum(sts.values_list('dominance_ratio')) /
len(sts)
data.distance = sum(sts.values_list('distance'))
data.ace_double_errors =
sum(sts.values_list('ace_double_errors')) / len(sts)
data.first_second_speed =
sum(sts.values_list('first_second_speed')) / len(sts)
data.max_average_speed =
max(sts.values_list('max_average_speed'))
data.breakpoints = sum(sts.values_list('breakpoints'))
data.breakpoints_match =
sum(sts.values_list('breakpoints_match')) / len(sts)
data.won_points = sum(sts.values_list('won_points'))
data.won_points_percentage =
sum(sts.values_list('won_points_percentage')) / len(sts)
data.points_set = sum(sts.values_list('points_set')) / len(sts)
data.tiebreaks = sum(sts.values_list('tiebreaks'))
data.won_tiebreaks = sum(sts.values_list('won_tiebreaks'))
data.points = sum(sts.values_list('points'))
data.matches_quantity = sum(sts.values_list('matches_quantity'))
data.won_matches = sum(sts.values_list('won_matches'))
data.points_match = sum(sts.values_list('points_match')) /
len(sts)
data.sets_quantity = sum(sts.values_list('sets_quantity'))

```

```

        data.won_sets = sum(sts.values_list('won_sets'))
        data.upsets = sum(sts.values_list('upsets'))
        data.won_upsets = sum(sts.values_list('won_upsets'))
        data.upsets_againts = sum(sts.values_list('upsets_againts'))
        data.lost_upsets_againts =
sum(sts.values_list('lost_upsets_againts'))
        data.save()
        ps = Player.objects.get(id=self.tennis_player_id)
        data = ps.overall_statistics_id
        sts =
Tournament_season_overall_statistics.objects.filter(tennis_player_id=
self.tennis_player_id)
        data.aces = sum(sts.values_list('aces'))
        data.aces_percentage = sum(sts.values_list('aces_percentage')) /
len(sts)
        data.win_service = sum(sts.values_list('win_service')) / len(sts)
        data.first_service = sum(sts.values_list('first_service')) /
len(sts)
        data.first_service_win =
sum(sts.values_list('first_service_win')) / len(sts)
        data.second_service = sum(sts.values_list('second_service')) /
len(sts)
        data.second_service_win =
sum(sts.values_list('second_service_win')) / len(sts)
        data.double_faults = sum(sts.values_list('double_faults')) /
len(sts)
        data.max_service_speed =
max(sts.values_list('max_service_speed'))
        data.return_service_won =
sum(sts.values_list('return_service_won')) / len(sts)
        data.return_unforced_errors =
sum(sts.values_list('return_unforced_errors')) / len(sts)
        data.first_return_service_won =
sum(sts.values_list('first_return_service_won')) / len(sts)
        data.second_return_service_won =
sum(sts.values_list('second_return_service_won')) / len(sts)
        data.won_netpoints = sum(sts.values_list('won_netpoints')) /
len(sts)
        data.won_netpoints_percentage =
sum(sts.values_list('won_netpoints_percentage')) / len(sts)
        data.return_breakpoints =
sum(sts.values_list('return_breakpoints'))
        data.won_breakpoints = sum(sts.values_list('won_breakpoints')) /
len(sts)
        data.saved_breakpoints =
sum(sts.values_list('saved_breakpoints')) / len(sts)
        data.won_rally = sum(sts.values_list('won_rally')) / len(sts)
        data.forced_errors_rally =
sum(sts.values_list('forced_errors_rally')) / len(sts)
        data.unforced_errors_rally =
sum(sts.values_list('unforced_errors_rally')) / len(sts)
        data.dominance_ratio = sum(sts.values_list('dominance_ratio')) /
len(sts)
        data.distance = sum(sts.values_list('distance'))
        data.ace_double_errors =
sum(sts.values_list('ace_double_errors')) / len(sts)
        data.first_second_speed =
sum(sts.values_list('first_second_speed')) / len(sts)
        data.max_average_speed =
max(sts.values_list('max_average_speed'))
        data.breakpoints = sum(sts.values_list('breakpoints'))

```

```

        data.breakpoints_match =
sum(sts.values_list('breakpoints_match')) / len(sts)
        data.won_points = sum(sts.values_list('won_points'))
        data.won_points_percentage =
sum(sts.values_list('won_points_percentage')) / len(sts)
        data.points_set = sum(sts.values_list('points_set')) / len(sts)
        data.tiebreaks = sum(sts.values_list('tiebreaks'))
        data.won_tiebreaks = sum(sts.values_list('won_tiebreaks'))
        data.points = sum(sts.values_list('points'))
        data.matches_quantity = sum(sts.values_list('matches_quantity'))
        data.won_matches = sum(sts.values_list('won_matches'))
        data.points_match = sum(sts.values_list('points_match')) /
len(sts)

        data.sets_quantity = sum(sts.values_list('sets_quantity'))
        data.won_sets = sum(sts.values_list('won_sets'))
        data.upsets = sum(sts.values_list('upsets'))
        data.won_upsets = sum(sts.values_list('won_upsets'))
        data.upsets_againts = sum(sts.values_list('upsets_againts'))
        data.lost_upsets_againts =
sum(sts.values_list('lost_upsets_againts'))
        data.save()
        m = Match.objects.get(id=matchset.match_id.id)
        m.points = m.points + 1
        super(Match_game, self).save(*args, **kwargs)

class Rating_system (models.Model):
    name = models.CharField(max_length=20)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('rating-system', kwargs={'pk': self.pk})

class Rating (models.Model):
    rating_system_id = models.ForeignKey(Rating_system, related_name='info',
on_delete=models.CASCADE)
    tennis_player_id = models.ForeignKey(Player, related_name='infor',
on_delete=models.CASCADE)
    place = models.IntegerField()
    rating_date = models.DateField(default=datetime.date.today, blank=True)

class Overall_statistics (models.Model):
    aces = models.BigIntegerField(default = 0, blank = True)
    aces_percentage = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
    win_service = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
    first_service = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
    first_service_win = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
    second_service = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
    second_service_win = models.DecimalField(max_digits = 3, decimal_places
= 1, default = 0, blank = True)
    double_faults = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
    max_service_speed = models.SmallIntegerField(default = 0, blank = True)
    return_service_won = models.DecimalField(max_digits = 3, decimal_places
= 1, default = 0, blank = True)

```

```

        return_unforced_errors = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        first_return_service_won = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        second_return_service_won = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        won_netpoints = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        won_netpoints_percentage = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        return_breakpoints = models.BigIntegerField(default = 0, blank = True)
        won_breakpoints = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        saved_breakpoints = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        won_rally = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        forced_errors_rally = models.DecimalField(max_digits = 3, decimal_places
= 1, default = 0, blank = True)
        unforced_errors_rally = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        dominance_ratio = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        distance = models.BigIntegerField(default = 0, blank = True)
        ace_double_errors = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        first_second_speed = models.DecimalField(max_digits = 3, decimal_places
= 1, default = 0, blank = True)
        max_average_speed = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        breakpoints = models.BigIntegerField(default = 0, blank = True)
        breakpoints_match = models.DecimalField(max_digits = 3, decimal_places =
1, default = 0, blank = True)
        won_points = models.BigIntegerField(default = 0, blank = True)
        won_points_percentange = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)
        points_set = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        tiebreaks = models.BigIntegerField(default = 0, blank = True)
        won_tiebreaks = models.BigIntegerField(default = 0, blank = True)
        points = models.BigIntegerField(default = 0)
        points_match = models.DecimalField(max_digits = 5, decimal_places = 1,
default = 0, blank = True)
        sets_quantity = models.BigIntegerField(default = 0)
        won_sets = models.BigIntegerField(default = 0)
        matches_quantity = models.BigIntegerField(default = 0)
        won_matches = models.BigIntegerField(default = 0)
        upsets = models.BigIntegerField(default = 0)
        won_upsets = models.DecimalField(max_digits = 3, decimal_places = 1,
default = 0, blank = True)
        upsets_againts = models.BigIntegerField(default = 0)
        lost_upsets_againts = models.DecimalField(max_digits = 3, decimal_places
= 1, default = 0, blank = True)

class Tournament_season_overall_statistics (models.Model):
    tennis_player_id = models.ForeignKey(Player, related_name='infot',
on_delete=models.CASCADE)
    overall_statistics_id = models.ForeignKey(Overall_statistics,
related_name='infott', on_delete=models.CASCADE)
    tournament_season_id = models.ForeignKey(Tournament_season,
related_name='infott', on_delete=models.CASCADE)
    last_tour = models.CharField(max_length=1, choices= TOURS)

```

```

class Tournament_overall_statistics (models.Model):
    tennis_player_id = models.ForeignKey(Player, related_name='infot',
on_delete=models.CASCADE)
    overall_statistics_id = models.ForeignKey(Overall_statistics,
related_name='infott', on_delete=models.CASCADE)
    tournament_id = models.ForeignKey(Tournament, related_name='infott',
on_delete=models.CASCADE)
    titles = models.SmallIntegerField(default = 0)
    times = models.SmallIntegerField(default = 0)
    best_tour = models.CharField(max_length=1, choices= TOURS)
    won_matches_percentage = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)

class Season_overall_statistics (models.Model):
    tennis_player_id = models.ForeignKey(Player, related_name='infot',
on_delete=models.CASCADE)
    overall_statistics_id = models.ForeignKey(Overall_statistics,
related_name='infop', on_delete=models.CASCADE)
    season_id = models.ForeignKey(Season, related_name='infou',
on_delete=models.CASCADE)
    titles = models.SmallIntegerField(default = 0)
    masters = models.SmallIntegerField(default = 0)
    won_matches_percentage = models.DecimalField(max_digits = 3,
decimal_places = 1, default = 0, blank = True)

class User_players (models.Model):
    user_id = models.ForeignKey(User, related_name='infop',
on_delete=models.CASCADE)
    tennis_player_id = models.ForeignKey(Player, related_name='infou',
on_delete=models.CASCADE)

class User_tournaments (models.Model):
    user_id = models.ForeignKey(User, related_name='infot',
on_delete=models.CASCADE)
    tournament_id = models.ForeignKey(Tournament, related_name='infou',
on_delete=models.CASCADE)

class User_headtoheads (models.Model):
    user_id = models.ForeignKey(User, related_name='infop',
on_delete=models.CASCADE)
    tennis_player1_id = models.ForeignKey(Player, related_name='infoh',
on_delete=models.CASCADE)
    tennis_player2_id = models.ForeignKey(Player, related_name='infoh',
on_delete=models.CASCADE)

```

Б.2 Текст файла /views/tennis_player.py

```

from django.views.generic import ListView, UpdateView
from django.views.generic.edit import CreateView
from django.views.generic.detail import DetailView
from .models import Player, Season_overall_statistics,
Tournament_overall_statistics, Tournament_season_overall_statistics
from django.urls import reverse_lazy

class PlayerSearchView(ListView):
    template_name = 'search_tennis_player.html'
    model = Player
    context_object_name = 'players'
    paginate_by = 100

```

```

    def get_queryset(self):
        whom = self.request.GET.get('tp')
        return
Player.objects.filter(full_name__contains=whom).order_by('-titles')

    def get_context_data(self, **kwargs):
        context = super(PlayerSearchView,
self).get_context_data(**kwargs)
        whom = self.request.GET.get('tp')
        context['whom'] = whom
        return context

class PlayersView(ListView):
    template_name = 'tennis_players.html'
    model = Player
    context_object_name = 'players'
    ordering = ['full_name']

class PlayerView (DetailView):
    template_name = 'tennis_player.html'
    model = Player

class PlayerSeasonView (ListView):
    template_name = 'player_seasons.html'
    model = Season_overall_statistics

    def get_queryset(self):
        whom = self.request.GET.get('tp')
        return
Season_overall_statistics.objects.filter(id=whom).order_by('-
season_id__name').values_list('season_id')

    def get_context_data(self, **kwargs):
        context = super(PlayerSeasonsView,
self).get_context_data(**kwargs)
        setseason = self.request.GET.get('season')
        context['season'] = setseason
        whom = self.request.GET.get('tp')
        context['statistics'] =
Season_overall_statistics.objects.get(season_id__id=setseason, tennis_player_id=whom)
        return context

class PlayerTournamentSeasonView (ListView):
    template_name = 'player_tournaments.html'
    model = Tournament_season_overall_statistics

    def get_queryset(self):
        whom = self.request.GET.get('tp')
        return
Tournament_season_overall_statistics.objects.filter(tennis_player_id__id=whom).order_
by('tournament_season_id__tournament_id__level',
'tournament_season_id__tournament_id__name',
'tournament_season_id__season_id__name').values_list('tournament_season_id')

    def get_context_data(self, **kwargs):
        context = super(PlayerSeasonsView,
self).get_context_data(**kwargs)
        whom = self.request.GET.get('tp')
        setseason = self.request.GET.get('season')
        settournam = self.request.GET.get('tournam')
        context['season'] = setseason
        context['tournament'] = settournam

```

```

        whom = self.request.GET.get('tp')
        tourseason =
Tournament_season.objects.get(tournament_id__id=setseason, season_id__id=settournam)
        context['statistics'] =
Tournament_season_overall_statistics.objects.get(tournament_season_id=tourseason,
tennis_player_id__id=whom)
        return context

class PlayerRatingView (ListView):
    template_name = 'player_rating.html'
    model = Player
    context_object_name = 'players'
    paginate_by = 100

    def get_queryset(self):
        spar = self.request.GET.get('criterion')
        if spar == 'aces':
            return Player.objects.all().order_by('-
overall_statistics_id__aces')[ :1000]
        elif spar == 'aces_percentage':
            return Player.objects.all().order_by('-
overall_statistics_id__aces_percentage')[ :1000]
        elif spar == 'win_service':
            return Player.objects.all().order_by('-
overall_statistics_id__win_service')[ :1000]
        elif spar == 'first_service':
            return Player.objects.all().order_by('-
overall_statistics_id__first_service')[ :1000]
        elif spar == 'first_service_win':
            return Player.objects.all().order_by('-
overall_statistics_id__first_service_win')[ :1000]
        elif spar == 'second_service':
            return Player.objects.all().order_by('-
overall_statistics_id__second_service')[ :1000]
        elif spar == 'second_service_win':
            return Player.objects.all().order_by('-
overall_statistics_id__second_service_win')[ :1000]
        elif spar == 'double_faults':
            return Player.objects.all().order_by('-
overall_statistics_id__double_faults')[ :1000]
        elif spar == 'max_service_speed':
            return Player.objects.all().order_by('-
overall_statistics_id__max_service_speed')[ :1000]
        elif spar == 'return_service_won':
            return Player.objects.all().order_by('-
overall_statistics_id__return_service_won')[ :1000]
        elif spar == 'return_unforced_errors':
            return Player.objects.all().order_by('-
overall_statistics_id__return_unforced_errors')[ :1000]
        elif spar == 'first_return_service_won':
            return Player.objects.all().order_by('-
overall_statistics_id__first_return_service_won')[ :1000]
        elif spar == 'second_return_service_won':
            return Player.objects.all().order_by('-
overall_statistics_id__second_return_service_won')[ :1000]
        elif spar == 'won_netpoints':
            return Player.objects.all().order_by('-
overall_statistics_id__won_netpoints')[ :1000]
        elif spar == 'won_netpoints_percentage':
            return Player.objects.all().order_by('-
overall_statistics_id__won_netpoints_percentage')[ :1000]
        elif spar == 'return_breakpoints':

```

```

        return Player.objects.all().order_by('-
overall_statistics_id_return_breakpoints')[:1000]
    elif spar == 'won_breakpoints':
        return Player.objects.all().order_by('-
overall_statistics_id_won_breakpoints')[:1000]
    elif spar == 'saved_breakpoints':
        return Player.objects.all().order_by('-
overall_statistics_id_saved_breakpoints')[:1000]
    elif spar == 'won_rally':
        return Player.objects.all().order_by('-
overall_statistics_id_won_rally')[:1000]
    elif spar == 'forced_errors_rally':
        return Player.objects.all().order_by('-
overall_statistics_id_forced_errors_rally')[:1000]
    elif spar == 'unforced_errors_rally':
        return Player.objects.all().order_by('-
overall_statistics_id_unforced_errors_rally')[:1000]
    elif spar == 'dominance_ratio':
        return Player.objects.all().order_by('-
overall_statistics_id_dominance_ratio')[:1000]
    elif spar == 'distance':
        return Player.objects.all().order_by('-
overall_statistics_id_distance')[:1000]
    elif spar == 'ace_double_errors':
        return Player.objects.all().order_by('-
overall_statistics_id_ace_double_errors')[:1000]
    elif spar == 'first_second_speed':
        return Player.objects.all().order_by('-
overall_statistics_id_first_second_speed')[:1000]
    elif spar == 'max_average_speed':
        return Player.objects.all().order_by('-
overall_statistics_id_max_average_speed')[:1000]
    elif spar == 'breakpoints':
        return Player.objects.all().order_by('-
overall_statistics_id_breakpoints')[:1000]
    elif spar == 'breakpoints_match':
        return Player.objects.all().order_by('-
overall_statistics_id_breakpoints_match')[:1000]
    elif spar == 'won_points':
        return Player.objects.all().order_by('-
overall_statistics_id_won_points')[:1000]
    elif spar == 'won_points_percantage':
        return Player.objects.all().order_by('-
overall_statistics_id_won_points_percantage')[:1000]
    elif spar == 'points_set':
        return Player.objects.all().order_by('-
overall_statistics_id_points_set')[:1000]
    elif spar == 'tiebreaks':
        return Player.objects.all().order_by('-
overall_statistics_id_tiebreaks')[:1000]
    elif spar == 'won_tiebreaks':
        return Player.objects.all().order_by('-
overall_statistics_id_won_tiebreaks')[:1000]
    elif spar == 'points':
        return Player.objects.all().order_by('-
overall_statistics_id_points')[:1000]
    elif spar == 'points_match':
        return Player.objects.all().order_by('-
overall_statistics_id_points_match')[:1000]
    elif spar == 'sets_quantity':
        return Player.objects.all().order_by('-
overall_statistics_id_sets_quantity')[:1000]

```

```

        elif spar == 'won_sets':
            return Player.objects.all().order_by('-
overall_statistics_id__won_sets')[:1000]
        elif spar == 'matches_quantity':
            return Player.objects.all().order_by('-
overall_statistics_id__matches_quantity')[:1000]
        elif spar == 'won_matches':
            return Player.objects.all().order_by('-
overall_statistics_id__won_matches')[:1000]
        elif spar == 'upsets':
            return Player.objects.all().order_by('-
overall_statistics_id__upsets')[:1000]
        elif spar == 'won_upsets':
            return Player.objects.all().order_by('-
overall_statistics_id__won_upsets')[:1000]
        elif spar == 'upsets_againts':
            return Player.objects.all().order_by('-
overall_statistics_id__upsets_againts')[:1000]
        elif spar == 'lost_upsets_againts':
            return Player.objects.all().order_by('-
overall_statistics_id__lost_upsets_againts')[:1000]

    def get_context_data(self, **kwargs):
        context = super(PlayerRatingView,
self).get_context_data(**kwargs)
        spar = self.request.GET.get('criterion')
        context['search'] = spar
        return context

class PlayerCreate(CreateView):
    model = Player
    fields = ['full_name', 'country_id', 'birth_date', 'height', 'weight',
'prizes', 'activeness', 'start_season', 'end_season', 'titles', 'residence',
'grand_slams', 'masters', 'main_arm', 'played_arm', 'overall_statistics_id']
    template_name = 'update_player.html'

    def get_success_url(self):
        return reverse_lazy('player-detail', kwargs={'pk':
self.object.pk})

class PlayerUpdate(UpdateView):
    model = Player
    fields = ['full_name', 'country_id', 'birth_date', 'height', 'weight',
'prizes', 'activeness', 'start_season', 'end_season', 'titles', 'residence',
'grand_slams', 'masters', 'main_arm', 'played_arm']
    template_name = 'update_player.html'

    def get_success_url(self):
        return reverse_lazy('player-detail', kwargs={'pk':
self.object.pk})

class OverallStatisticsUpdate(UpdateView):
    model = Overall_statistics
    fields = ['aces', 'aces_percentage', 'win_service', 'first_service',
'first_service_win', 'second_service', 'second_service_win', 'double_faults',
'max_service_speed', 'return_service_won', 'return_unforced_errors',
'first_return_service_won', 'second_return_service_won', 'won_netpoints',
'won_netpoints_percentage', 'return_breakpoints', 'won_breakpoints',
'saved_breakpoints', 'won_rally', 'forced_errors_rally',
'unforced_errors_rally', 'dominance_ratio', 'distance', 'ace_double_errors',
'first_second_speed', 'max_average_speed', 'breakpoints', 'breakpoints_match',
'won_points', 'won_points_percantage', 'points_set', 'tiebreaks',

```

```

'won_tiebreaks', 'points', 'points_match', 'sets_quantity', 'won_sets',
'matches_quantity', 'won_matches', 'upsets', 'won_upsets', 'upsets_againts',
'lost_upsets_againts']
    template_name = 'update_player.html'

    def get_success_url(self):
        num = self.object.pk
        pl = Player.objects.get(overall_statistics_id__id=num)
        return reverse_lazy('player-detail', kwargs={'pk': pl.id})

class HeadToHeaderView(ListView):
    template_name = 'head_to_head.html'
    model = Player
    context_object_name = 'players'

    def get_queryset(self):
        whom1 = self.request.GET.get('tp1')
        whom2 = self.request.GET.get('tp2')
        return Player.objects.all().filter(id=whom1 | id=whom2)

    def get_context_data(self, **kwargs):
        context = super(HeadToHeaderView, self).get_context_data(**kwargs)
        whom1 = self.request.GET.get('tp1')
        whom2 = self.request.GET.get('tp2')
        context['whom1'] = whom1
        context['whom2'] = whom2
        matches = Match.objects.filter(tennis_player1 = whom1 |
tennis_player2 = whom1, tennis_player1 = whom2 | tennis_player2 = whom1)
        context['matches'] = matches
        matches = Match.objects.filter(tennis_player1 = whom1 | tennis_player2 =
whom1, tennis_player1 = whom2 | tennis_player2 = whom1)
        context['matches_count'] = matches.count()
        context['grand'] =
matches.filter(tournament_season_id__tournament_id__level='A')
        context['masters'] =
matches.filter(tournament_season_id__tournament_id__level='B')
        total = matches.filter(tennis_player1 = whom1, winner =
False).count
        total = total + matches.filter(tennis_player2 = whom1, winner =
True).count
        context['masters'] = total
        return context

class HeadToHeadSeasonView(ListView):
    template_name = 'head_to_head_season.html'
    model = Season_overall_statistics
    context_object_name = 'statistics'

    def get_queryset(self):
        whom1 = self.request.GET.get('tp1')
        whom2 = self.request.GET.get('tp2')
        setseason = self.request.GET.get('season')
        data =
Season_overall_statistics.objects.all().filter(tennis_player_id__id=whom1 |
tennis_player_id__id=whom2)
        if setseason:
            return data.filter(season_id__id=setseason)
        else:
            return data

    def get_context_data(self, **kwargs):
        context = super(HeadToHeaderView, self).get_context_data(**kwargs)

```

```

        whom1 = self.request.GET.get('tp1')
        whom2 = self.request.GET.get('tp2')
        setseason = self.request.GET.get('season')
        context['whom1'] = whom1
        context['whom2'] = whom2
        context['setseason'] = setseason
        return context

class HeadToHeadTournamentView(ListView):
    template_name = 'head_to_head_tournament.html'
    model = Tournament_overall_statistics
    context_object_name = 'statistics'

    def get_queryset(self):
        whom1 = self.request.GET.get('tp1')
        whom2 = self.request.GET.get('tp2')
        settournament = self.request.GET.get('tournament')
        data =
Tournament_overall_statistics.objects.all().filter(tennis_player_id__id=whom1 |
tennis_player_id__id=whom2)
        if settournament:
            return data.filter(tournament_id__id=setttournament)
        else:
            return data

    def get_context_data(self, **kwargs):
        context = super(HeadToHeadView, self).get_context_data(**kwargs)
        whom1 = self.request.GET.get('tp1')
        whom2 = self.request.GET.get('tp2')
        setseason = self.request.GET.get('season')
        context['whom1'] = whom1
        context['whom2'] = whom2
        context['setseason'] = setseason
        return context

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

Дипломна кваліфікаційна робота бакалавра

**Програмне забезпечення надання тенісної статистики
онлайн
Software for Tennis Statistics Online Access**

Виконав

студент групи КНТ-219сп

В.В. Нурієв

Керівник

О.О. Степаненко

2022

Рисунок В.1 – Слайд № 1

Об'єкт, предмет, мета, завдання роботи

Об'єкт роботи – процес відслідковування тенісної статистики.

Предмет роботи – програмне забезпечення надання тенісної статистики онлайн.

Мета роботи – розробити програмне забезпечення надання тенісної статистики онлайн для забезпечення можливостей індивідуального відслідковування.

Завдання роботи:

- ✓ аналіз програмних аналогів;
- ✓ визначення функціональних вимог до програми надання тенісної статистики онлайн;
- ✓ проектування програми;
- ✓ реалізація і налагодження програми.

2

Рисунок В.2 – Слайд № 2

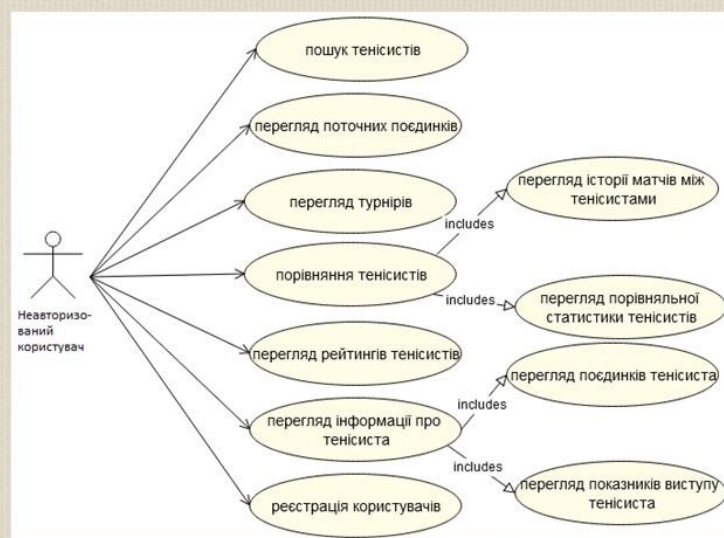
Вибір фреймворку розробки вебзастосунку

Критерій	Flask	Django
Зручна робота з даними	Так	Так
Зручна робота над формуванням вебсторінок в динаміці	Так	Так
Адміністративна панель за замовчуванням	Ні	Так
Інструменти безпеки	Гірші	Кращі
Об'єктно-реляційне відображення за замовчуванням	Ні	Так

3

Рисунок В.3 – Слайд № 3

Діаграма прецедентів



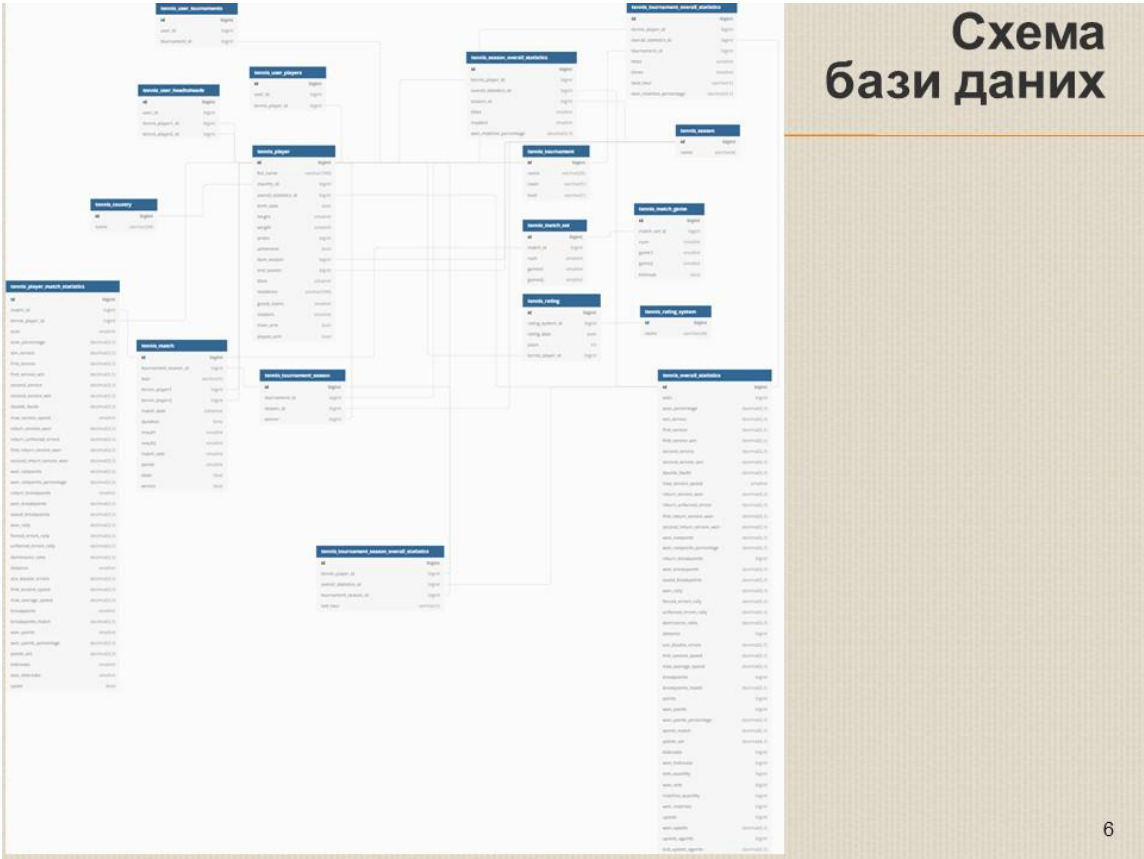
4

Рисунок В.4 – Слайд № 4



5

Рисунок В.5 – Слайд № 5



6

Рисунок В.6 – Слайд № 6

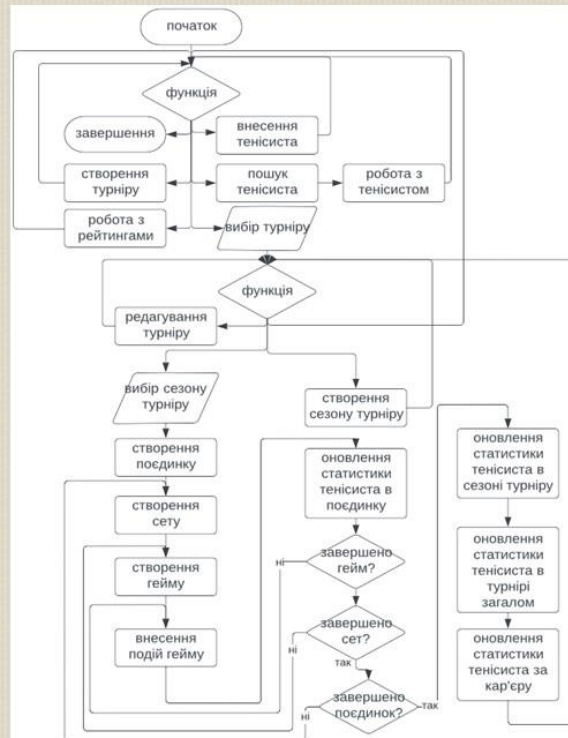
Структура програми



7

Рисунок В.7 – Слайд № 7

Схема функціонування програми для менеджера



8

Рисунок В.8 – Слайд № 8

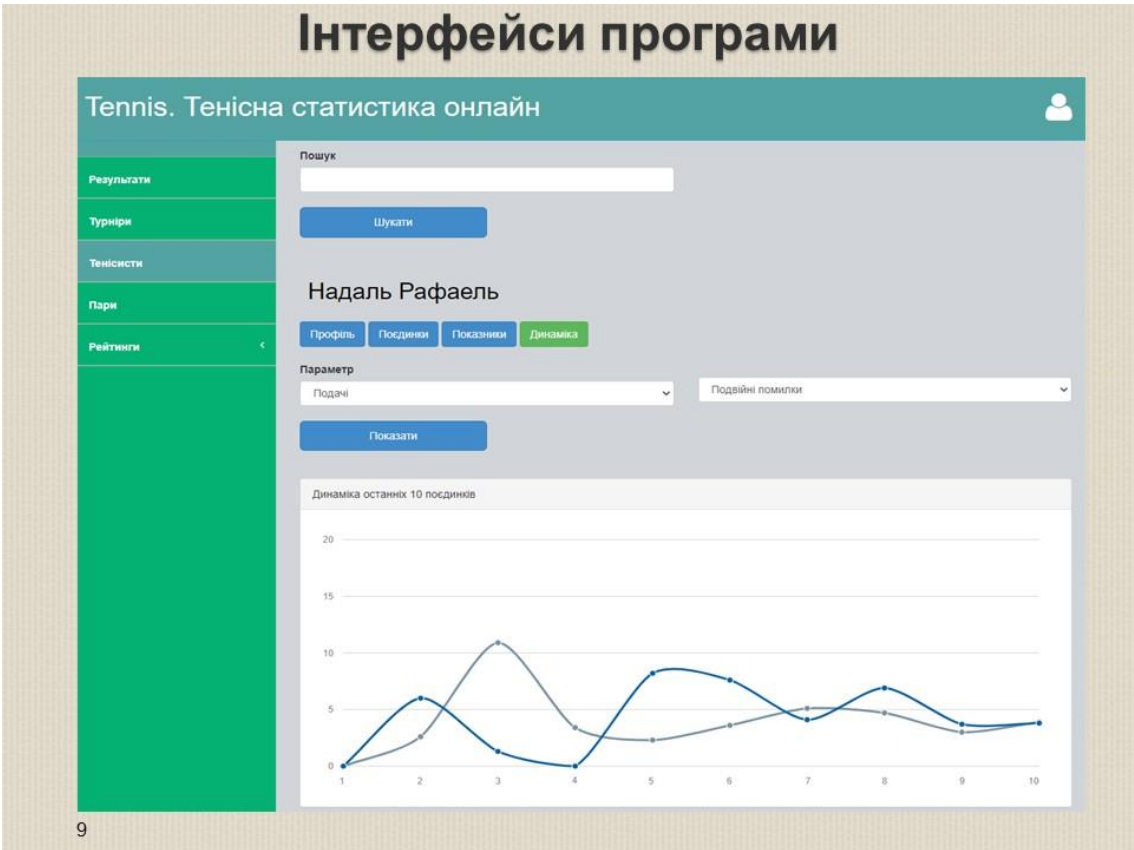


Рисунок В.9 – Слайд № 9

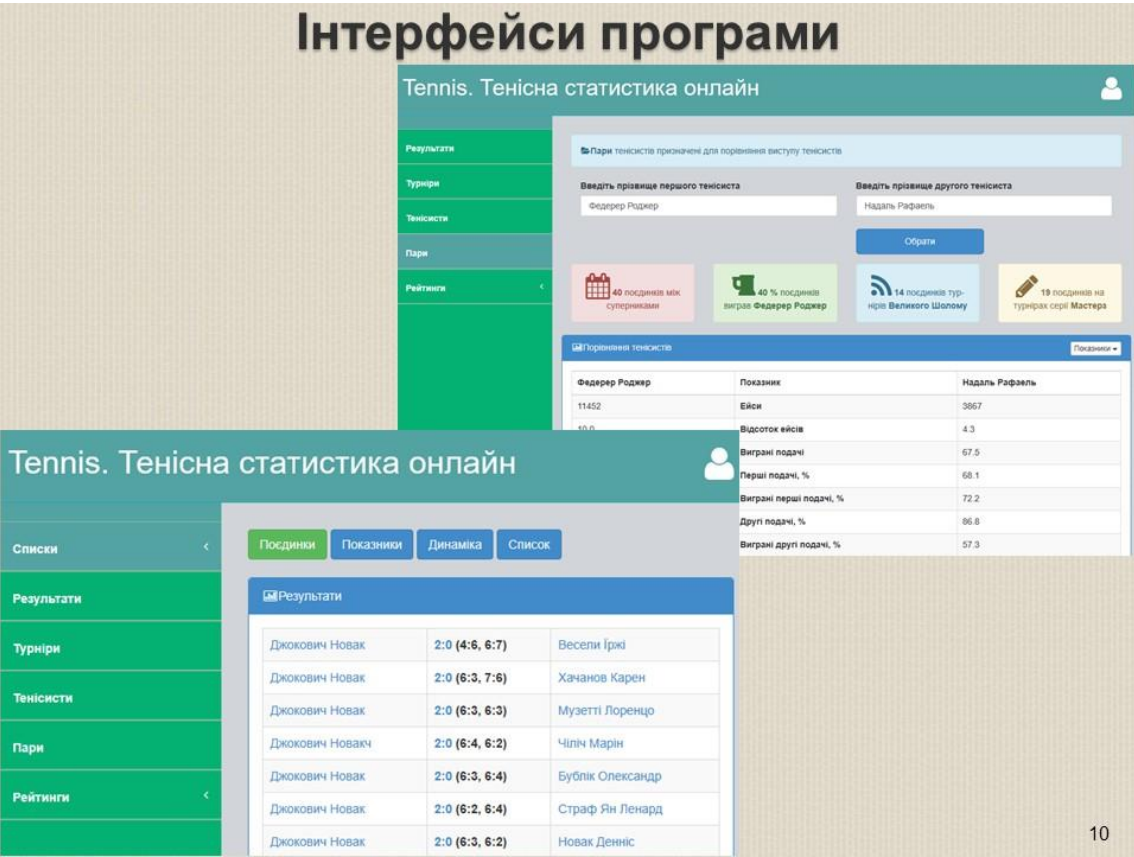


Рисунок В.10 – Слайд № 10

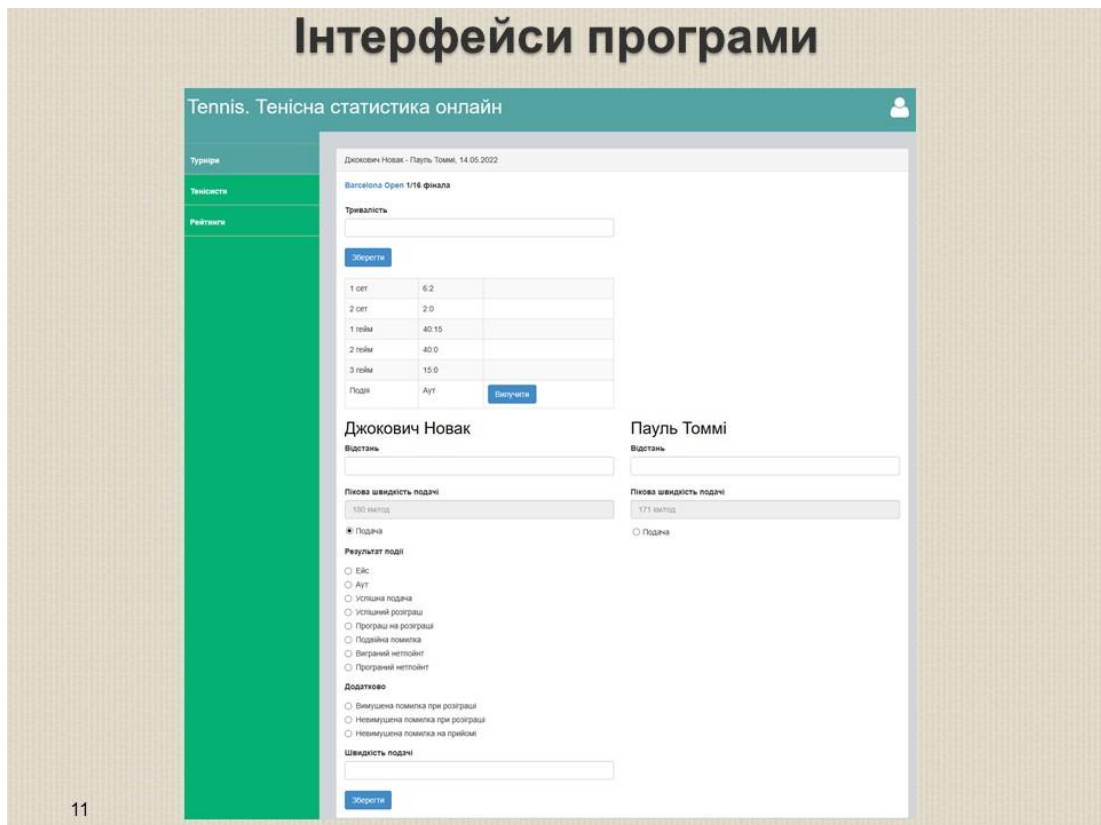


Рисунок В.11 – Слайд № 11

Висновки

Розробку програмного забезпечення надання тенісної статистики онлайн виконано за допомогою вебфреймворку Django з визначенням структури даних на основі використання бази даних, вимог до програми, розробленням структури програми і алгоритму роботи програми на основі цих вимог, реалізацією відповідних одиниць вебзастосунку. Для розробки використано мови програмування Python, JavaScript, мову розмітки HTML, мову стилю сторінок CSS, вебфреймворк Django, систему керування базами даних MySQL.

Програма дозволяє переглядати результати тенісних змагань, окремих поєдинків тенісистів, показники виступу тенісистів у окремих поєдинках, а також накопичену статистику за весь турнір, за всі сезони турніру, за весь сезон, а також за всю кар'єру, переглядати порівняні показники виступу пар тенісистів, історію їх протистоянь, виконувати пошук тенісистів, створювати власні списки тенісистів, турнірів, пар тенісистів для надання швидкого доступу.

12

Рисунок В.12 – Слайд № 12