ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут телекомунікацій

Кафедра системного аналізу

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: **«Алгоритмічне сортування команд для ігор в групі та плей-офф»**

Виконала: студентка 4 курсу,
групи САД–41 спеціальності
124 Системного аналізу_____
(шифр і назва спеціальності)

Бабаскін О.І._____
(прізвище та ініціали)

Керівник Гордієнко Т.Б._____
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут телекомунікацій

Кафедра Системного аналізу_____
Ступінь вищої освіти - «Бакалавр»_____
Спеціальність - 124 Системний аналіз_____

ЗАТВЕРДЖУЮ
Завідувач кафедри
Системного аналізу
_____Гордієнко Т.Б

“____” _____2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

_____Бабаскін Олексій Ігорович_____
(прізвище, ім‘я, по батькові)

1.Тема роботи:
Алгоритмічне сортування команд для ігор в групі та плей-офф
Керівник роботи____ Гордієнко Т.Б._____,
(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “ “_____2023 року №__.
2. Строк подання студентом роботи_____
3. Вхідні дані до роботи:
Література з питань оформлення силабусів;
Інструменти розробки додатків, мова програмування Go, JavaScript, бібліотеки, фреймворк Vue.js;
Науково-технічна література з питань, пов‘язаних з оформленням силабусів.

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

1. Провести аналіз процесів оформлення та роботи с силабусами;
2. Визначити структуру силабусу з метою подальшого автоматизованого формування силабусу;
3. Провести моделювання про процесів оформлення та роботи с силабусами
4. Спроектувати та розробити систему для автоматизації розрахунків даних силабусу;
5. Реалізувати інтерфейсні форми, орієнтовані на специфіку задачі, які забезпечують зручну взаємодію користувача.

Перелік графічного матеріалу

1_____
2._____
3._____
4._____
Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз створення силабусів	До 04.04	
2	Опис процесів для створення силабусів	До 04.04	
3	Прототипування інтерфейсу програми	04.04-25.04	
4	Реалізація системи	25.04-06.05	
5	Висновки+презентація	06.05-13.05	
6	Перевірка роботи на антиплагіат+Передзахист	14.05-01.06	
7	Захист роботи	02.06-21.06	
8	Випуск	30.06	

Студент _____ О.І. Бабаскін
(підпис) (прізвище та ініціали)

Керівник роботи _____ Т.Б. Гордієнко
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 65 с., 13 табл., 12 рис., 1 дод., 15 джерела.

Обсяг звіту: Звіт складається з 59 сторінок, включаючи 34 рисунки, таблиці, 8 додатків і посилається на 16 джерел.

Скорочення та умовні позначки

ІНФОРМАЦІЙНА СИСТЕМА, ФУТБОЛ, ФРЕЙМВОРК, HTML, CSS, JAVASCRIPT, JAVA, API, JSON, SPARK, POSTGRE

Опис тексту роботи:

Об'єкт дослідження: В роботі досліджується процес формування розкладу футбольних матчів.

Предмет дослідження: Розглядається розробка інформаційної системи, яка допомагає в оптимізації формування розкладу футбольних матчів в найпопулярніших лігах.

Ціль роботи: Розробка інформаційної системи, яка аналізує помилки букмекерських компаній при передбаченні результатів футбольних матчів у найпопулярніших лігах.

Методи дослідження: В роботі використовувалися такі методи, як аналіз даних, статистичні методи, комп'ютерне моделювання.

Результати та їх новизна: Реалізовано інформаційну систему, яка регулярно збирає дані про футбольні матчі і забезпечує оптимальне формування розкладу матчів. Результати дослідження вказують на помилки букмекерських компаній у передбаченні результатів у найпопулярніших лігах.

Рекомендації щодо використання результатів роботи: Розроблена інформаційна система може бути використана в діяльності букмекерських компаній для поліпшення точності передбачення результатів футбольних матчів.

Сфера застосування: Розроблена система може бути в розроблена система може бути використана в професійних футбольних лігах

Зміст

ВСТУП

Актуальність: футбол є однією з найпопулярніших ігор у світі, і змагання з його участю привертають велику увагу глядачів та учасників. Організація регулярних футбольних матчів вимагає значних зусиль та ресурсів. При цьому важливо забезпечити максимальну ефективність використання часу та ресурсів, щоб зробити гру більш доступною і привабливою для глядачів та учасників. У зв'язку з цим, автоматизація процесу формування розкладу футбольних матчів є актуальною темою для дослідження.

Ступінь вивченої проблеми: на сьогоднішній день, існує певна кількість програмного забезпечення, що дозволяють автоматизувати процес формування розкладу футбольних матчів. Однак, більшість з цих програм не враховують важливі фактори, такі як національні свята, важливі матчі в інших чемпіонатах, транспортні засоби, які використовуються для перевезення команд та багато іншого. Таким чином, дослідження та розробка нової інформаційної системи, яка враховує ці фактори та забезпечує оптимальний розклад, має високий ступінь актуальності та значимості.

Специфіка джерельної бази: для розробки інформаційної системи автоматизованого формування розкладу футбольних матчів будуть використовуватися різноманітні джерела інформації, такі як бази даних про команди, календарі національних свят, розклади інших чемпіонатів, інформація про транспорт та інше. Для збору та аналізу цієї інформації можуть використовуватися різні методи, такі як парсинг веб-сторінок, використання API та інші.

Об'єкт дослідження: об'єктом дослідження є процес формування розкладу футбольних матчів.

Предмет дослідження: предметом дослідження є розробка інформаційної системи автоматизованого формування розкладу футбольних матчів.

Мета і завдання роботи: метою даної роботи є розробка інформаційної системи, яка дозволить автоматизувати процес формування розкладу

футбольних матчів з урахуванням різних факторів та забезпечить оптимальний розклад для команд та глядачів.

Завданнями роботи є:

– аналіз існуючих систем автоматизації формування розкладу футбольних матчів;

– визначення критеріїв ефективності розкладу та врахування їх у розробці системи;

– розробка алгоритмів для формування розкладу з урахуванням різних факторів, таких як національні свята, важливі матчі в інших чемпіонатах, транспортні засоби, які використовуються для перевезення команд та інші;

– впровадження розробленої системи та тестування її ефективності.

Методика дослідження: для розробки інформаційної системи будуть використовуватися методи аналізу та проектування програмного забезпечення, а також методи обробки даних та аналізу результатів.

Наукова новизна роботи: науковою новизною даної роботи є розробка нової інформаційної системи автоматизованого формування розкладу футбольних матчів з урахуванням різних факторів та критеріїв ефективності розкладу.

Практична значущість результатів дослідження: результати даної роботи можуть бути корисними для організаторів футбольних змагань, які мають потребу в формуванні розкладу матчів, а також для футбольних клубів, які можуть отримати оптимальний розклад з урахуванням своїх потреб та інтересів. Крім того, розроблена система може бути використана для інших видів спорту, які також мають потребу в формуванні розкладу змагань.

Отже, розробка інформаційної системи автоматизованого формування розкладу футбольних матчів є актуальною темою, яка має наукову новизну та практичну значущість. Для її реалізації будуть використовуватися методи аналізу та проектування програмного забезпечення, а також методи обробки даних та аналізу результатів.

І ТЕОРЕТИЧНІ ОСНОВИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Основні поняття та визначення

Інформаційна система (ІС) - це сукупність взаємопов'язаних компонентів, які взаємодіють з метою збору, обробки, зберігання, передачі та використання інформації для досягнення певної мети.

Автоматизація - це процес перетворення ручних процедур та операцій в автоматичні, які здійснюються за допомогою програмного забезпечення.

Розклад - це документ, який містить інформацію про час та місце проведення футбольних матчів.

Об'єктом дослідження є процес формування розкладу футбольних матчів за допомогою інформаційної системи автоматизованого формування розкладу футбольних матчів.

Процес формування розкладу футбольних матчів є складним та вимагає значних зусиль та ресурсів. Для оптимального формування розкладу необхідно враховувати різноманітні фактори, такі як національні свята, важливі матчі в інших чемпіонатах, транспортні засоби, які використовуються для перевезення команд та інші.

Кількісні показники, які характеризують об'єкт дослідження, включають кількість команд, кількість матчів, кількість днів, на які складається розклад, кількість стадіонів.

Якісні показники, які характеризують об'єкт дослідження, включають якість розкладу футбольних матчів, ефективність використання часу та ресурсів при їх формуванні, а також відповідність розкладу потребам команд, глядачів та інших учасників.

Оцінювання якості розкладу можна провести за кількома параметрами, такими як кількість перенесених матчів, кількість конфліктів у розкладі, наявність днів відпочинку для команд та інші. Для ефективності використання ресурсів можна використовувати параметри, такі як кількість зайнятого часу для

формування розкладу, кількість зайнятих працівників та використання обладнання та програмного забезпечення.

Також важливо враховувати вимоги та потреби команд, які приймають участь у змаганнях, а також глядачів та інших учасників. Наприклад, розклад повинен враховувати важливі матчі в інших чемпіонатах, національні свята та інші фактори, які можуть впливати на розклад гри.

Для дослідження та оцінювання параметрів розкладу можна використовувати різні методики, наприклад, методи аналізу даних, статистичного моделювання, експертних оцінок та інших. Результати дослідження допоможуть враховувати всі фактори, які впливають на розклад, і забезпечити ефективну та оптимальну організацію футбольних матчів.

Якісні показники, які характеризують об'єкт дослідження, включають якість розкладу, тобто його точність та відповідність вимогам учасників та глядачів. Крім того, важливим показником є ефективність розкладу, тобто відповідність використання ресурсів та часу для проведення матчів. Інші показники можуть включати пропускну здатність стадіонів, наявність транспорту для команд, зручність для глядачів та інші.

Методика дослідження повинна включати ретельний аналіз поточної практики формування розкладів футбольних матчів та існуючих інформаційних систем для автоматизації цього процесу. Необхідно провести аналіз потреб учасників (команд, глядачів, організаторів), враховуючи їхні вимоги та обмеження. Також потрібно вивчити діючі законодавчі та нормативні акти, які впливають на формування розкладів футбольних матчів, а також вивчити досвід використання схожих інформаційних систем у суміжних галузях.

Окрім того, в рамках методики дослідження необхідно визначити ключові фактори, які впливають на формування розкладів футбольних матчів, такі як графік інших змагань, національні свята, кліматичні умови, наявність транспорту тощо. Потрібно також вивчити можливості застосування інформаційних технологій та алгоритмів для автоматизації процесу формування розкладів футбольних матчів.

1.2 Наявні системи передбачення переможців

Для аналізу можливих переможців було створено багато систем. Наприклад, вгадати переможців за допомогою штучного інтелекту на чемпіонат Євро-2016 намагалися такі компанії, як Google, Yahoo, Microsoft та інші.

Більшість з них вважала фаворитом збірну Німеччини. Ніхто з них не бачив фаворитом реального переможця – збірну Португалії. Розробленням методів та моделей займалися такі науковці: С. Добсон, А. Ротштейн, Дж. Годдард, С. Штовба. Стівен Добсон і Джон Годдард запропонували модель, де основним фактором вибрано кількість голів, забитих кожною командою при персональних зустрічах. Модель побудована близько на 30 сезонах даних.

Інший науковець, Ротштейн А. використовув алгоритм ставок методом нечіткої логіки на основі 12 чемпіонатів з футболу Фінляндії.

Алгоритм розрахунку ставок на футбол проводиться через нейрони і генетичне навчання. Штовба С. прогнозував розподіл місць у турнірній таблиці чемпіонату України з футболу на основі нечіткої логіки. Буурсма обрав набір функцій і використав ряд класифікацій алгоритмів, включаючи просту і логістичну регресії, байєсівську мережу, і дерево прийняття рішень, щоб передбачити результат футбольного матчу. Його передбачення мали три виходи (перемога команди господаря, нічия, перемога гостьової команди). Ймовірності цих трьох результатів були розраховані і результат із найбільшою ймовірністю був обраний [1].

Студенти Корнелльського університету створили систему на основі машинного навчання, котра декількома способами спрогнозувала переможця останнього Чемпіонату світу [2].

В їх результатах Франція мала шанси пройти до фіналу. Хорватія мала тільки шість відсотків потрапити до фіналу. Таким чином, ми бачимо, що на даний момент не існує системи, котра б могла безпомилково визначити результат матчу. А це означає, що як науковці, так і самі 7 букмекерські компанії роблять неточні передбачення.

Це дає можливість на основі власного аналізу зробити передбачення на ту чи іншу спортивну подію. Існують сервіси, котрі зберігають данні минулих матчів та данні букмекерів для них для подальшого аналізу з метою передбачення наступних результатів. Прикладом такого сервісу є oddsportal.com (див. рис. 1.1).

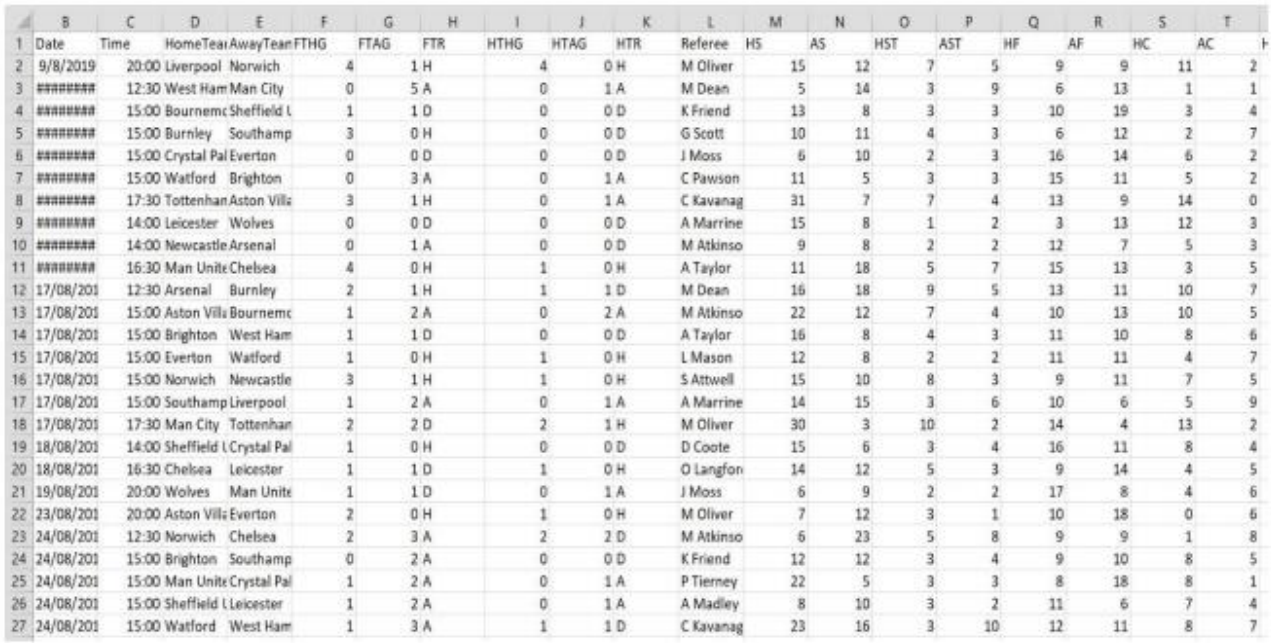
21/05, 14:00	**Liverpool** - Middlesbrough	**3:0**	1.13	9.49	21.41	9
14/05, 13:15	West Ham - **Liverpool**	**0:4**	5.83	4.08	1.63	9
07/05, 12:30	**Liverpool** - Southampton	**0:0**	1.58	4.21	6.16	9
01/05, 19:00	Watford - **Liverpool**	**0:1**	5.77	4.28	1.61	9
23/04, 15:30	**Liverpool** - Crystal Palace	**1:2**	1.51	4.43	7.15	9
16/04, 12:30	West Brom - **Liverpool**	**0:1**	4.57	3.69	1.85	9
08/04, 14:00	Stoke - **Liverpool**	**1:2**	3.43	3.32	2.27	9
05/04, 19:00	**Liverpool** - Bournemouth	**2:2**	1.43	4.76	8.14	9
01/04, 11:30	**Liverpool** - Everton	**3:1**	1.73	3.77	5.44	8
19/03, 15:30	Manchester City - **Liverpool**	**1:1**	1.95	3.80	3.94	8
12/03, 15:00	**Liverpool** - Burnley	**2:1**	1.27	6.10	12.55	8
04/03, 16:30	**Liverpool** - Arsenal	**3:1**	1.99	3.65	3.99	8
27/02, 19:00	Leicester - **Liverpool**	**3:1**	5.59	4.07	1.65	8
11/02, 16:30	**Liverpool** - Tottenham	**2:0**	2.23	3.41	3.46	8

Рисунок 1.1 Приклад даних з сайту oddsportal.com

Як бачимо, представлена історія матчів, їх результати та ставки. Але таке відображення даних не є зручним для аналізу результатів певної команди чи ліги в цілому. Воно не є наочним і складне для сприйняття.

Також немає можливості для підрахунку кількості помилок передбачень для певної команди протягом сезону, місяця тощо. По факту, це є просте відображення даних у вигляді звичайного списку, відсортоване за командами. Існують сайти, котрі пропонують придбати історичні дані про матчі та ставки на них. Інформація буде надана в файлах (наприклад .csv).

Також існують сайти з безкоштовними даними, наприклад football-data.co.uk/englandm.php. Цей ресурс надає дані у вигляді .csv фалу (див. рис. 1.2).



	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Date	Time	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	HS	AS	HST	AST	HF	AF	HC	AC
2	9/8/2019	20:00	Liverpool	Norwich	4	1	H	4	0	H	M Oliver	15	12	7	5	9	9	11	2
3	########	12:30	West Ham	Man City	0	5	A	0	1	A	M Dean	5	14	3	9	6	13	1	1
4	########	15:00	Bournemc	Sheffield U	1	1	D	0	0	D	K Friend	13	8	3	3	10	19	3	4
5	########	15:00	Burnley	Southamp	3	0	H	0	0	D	G Scott	10	11	4	3	6	12	2	7
6	########	15:00	Crystal Pal	Everton	0	0	D	0	0	D	J Moss	6	10	2	3	16	14	6	2
7	########	15:00	Watford	Brighton	0	3	A	0	1	A	C Pawson	11	5	3	3	15	11	5	2
8	########	17:30	Tottenhan	Aston Villa	3	1	H	0	1	A	C Kavanag	31	7	7	4	13	9	14	0
9	########	14:00	Leicester	Wolves	0	0	D	0	0	D	A Marrine	15	8	1	2	3	13	12	3
10	########	14:00	Newcastle	Arsenal	0	1	A	0	0	D	M Atkinso	9	8	2	2	12	7	5	3
11	########	16:30	Man Unite	Chelsea	4	0	H	1	0	H	A Taylor	11	18	5	7	15	13	3	5
12	17/08/201	12:30	Arsenal	Burnley	2	1	H	1	1	D	M Dean	16	18	9	5	13	11	10	7
13	17/08/201	15:00	Aston Villa	Bournemc	1	2	A	0	2	A	M Atkinso	22	12	7	4	10	13	10	5
14	17/08/201	15:00	Brighton	West Ham	1	1	D	0	0	D	A Taylor	16	8	4	3	11	10	8	6
15	17/08/201	15:00	Everton	Watford	1	0	H	1	0	H	L Mason	12	8	2	2	11	11	4	7
16	17/08/201	15:00	Norwich	Newcastle	3	1	H	1	0	H	S Attwell	15	10	8	3	9	11	7	5
17	17/08/201	15:00	Southamp	Liverpool	1	2	A	0	1	A	A Marrine	14	15	3	6	10	6	5	9
18	17/08/201	17:30	Man City	Tottenhan	2	2	D	2	1	H	M Oliver	30	3	10	2	14	4	13	2
19	18/08/201	14:00	Sheffield U	Crystal Pal	1	0	H	0	0	D	D Coote	15	6	3	4	16	11	8	4
20	18/08/201	16:30	Chelsea	Leicester	1	1	D	1	0	H	O Langfor	14	12	5	3	9	14	4	5
21	19/08/201	20:00	Wolves	Man Unite	1	1	D	0	1	A	J Moss	6	9	2	2	17	8	4	6
22	23/08/201	20:00	Aston Villa	Everton	2	0	H	1	0	H	M Oliver	7	12	3	1	10	18	0	6
23	24/08/201	12:30	Norwich	Chelsea	2	3	A	2	2	D	M Atkinso	6	23	5	8	9	9	1	8
24	24/08/201	15:00	Brighton	Southamp	0	2	A	0	0	D	K Friend	12	12	3	4	9	10	8	5
25	24/08/201	15:00	Man Unite	Crystal Pal	1	2	A	0	1	A	P Tierney	22	5	3	3	8	18	8	1
26	24/08/201	15:00	Sheffield U	Leicester	1	2	A	0	1	A	A Madley	8	10	3	2	11	6	7	4
27	24/08/201	15:00	Watford	West Ham	1	3	A	1	1	D	C Kavanag	23	16	3	10	12	11	8	7

Рисунок 1.2 Приклад даних з football-data.co.uk

Такі дані можуть бути зручними для подальшої роботи з ними, адже вони добре структуровані. Але проблема цього ресурсу в тому, що він надає інформацію тільки про команди Англії.

А також ці данні не будуть оновлюватися. Необхідно кожного разу знову завантажувати файл для подальшої роботи. Soccerstats пропонує багато інформації про всі аспекти гри. Цей сайт охоплює численні різні ліги з усього світу (див. рис. 1.3). Він дає можливість безпосередньо аналізувати результат спортивних подій. Але цей ресурс не має можливості аналізу передбачень букмекерів.

75.8% completed

2.72 goals per match

Table

		GP	W	D	L	GF	GA	GD	Pts	Form	PPG	last 8	CS	FTS
1	Liverpool	29	27	1	1	66	21	+45	82		2.83	2.63	41%	3%
2	Manchester City	28	18	3	7	68	31	+37	57		2.04	2.00	36%	11%
3	Leicester City	29	16	5	8	58	28	+30	53		1.83	1.00	34%	21%
4	Chelsea	29	14	6	9	51	39	+12	48		1.66	1.50	21%	21%
5	Manchester Utd	29	12	9	8	44	30	+14	45		1.55	1.75	28%	28%
6	Wolverhampton	29	10	13	6	41	34	+7	43		1.48	1.63	28%	17%
7	Sheffield Utd	28	11	10	7	30	25	+5	43		1.54	1.75	36%	25%
8	Tottenham	29	11	8	10	47	40	+7	41		1.41	1.38	14%	21%
9	Arsenal	28	9	13	6	40	36	+4	40		1.43	2.00	25%	18%
10	Burnley	29	11	6	12	34	40	-6	39		1.34	1.88	38%	31%
11	Crystal Palace	29	10	9	10	26	32	-6	39		1.34	1.38	31%	34%
12	Everton	29	10	7	12	37	46	-9	37		1.28	1.50	21%	24%
13	Newcastle Utd	29	9	8	12	25	41	-16	35		1.21	1.25	31%	38%

Su 17 May	Arsenal - Watford	pp.
Su 17 May	Burnley - Brighton	pp.
Su 17 May	Chelsea - Wolverhampton	pp.
Su 17 May	Crystal Palace - Tottenham	pp.
Su 17 May	Everton - Bournemouth	pp.
Su 17 May	Leicester City - Manchester Utd	pp.
Su 17 May	Manchester City - Norwich City	pp.
Su 17 May	Newcastle Utd - Liverpool	pp.
Su 17 May	Southampton - Sheffield Utd	pp.
Su 17 May	West Ham Utd - Aston Villa	pp.
Sa 9 May	Aston Villa - Arsenal	pp.

Next matches | All matches

Рисунок 1.3 Приклад даних з soccerstats.com

Сервіс whoscored також надає розширену статистику футбольних матчів, статистику команд та ліг (див. рис. 1.4). Але як і попередній ресурс, не містить даних про букмекерські компанії

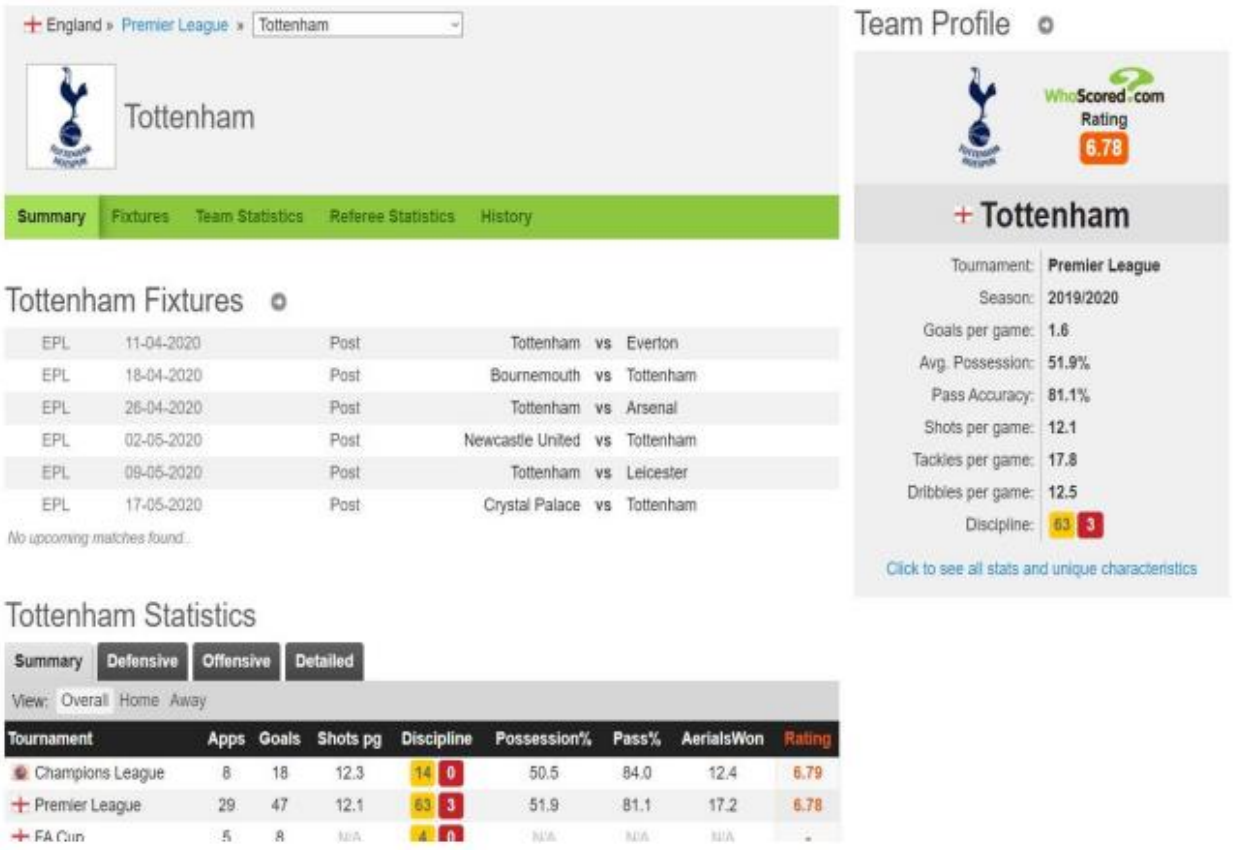


Рисунок 1.4 Приклад даних з ресурсу whoscored.com

Ресурс flashscore призначений для аналізу статистики окремих команд. Надає статистику про матчі, трансфери та поточний склад команд. Існує можливість перегляду передбачень букмекерів, але це реалізовано тільки як перегляд даних на один матч (див. рис. 1.5). Загальна статистика для команди чи ліги відсутня.

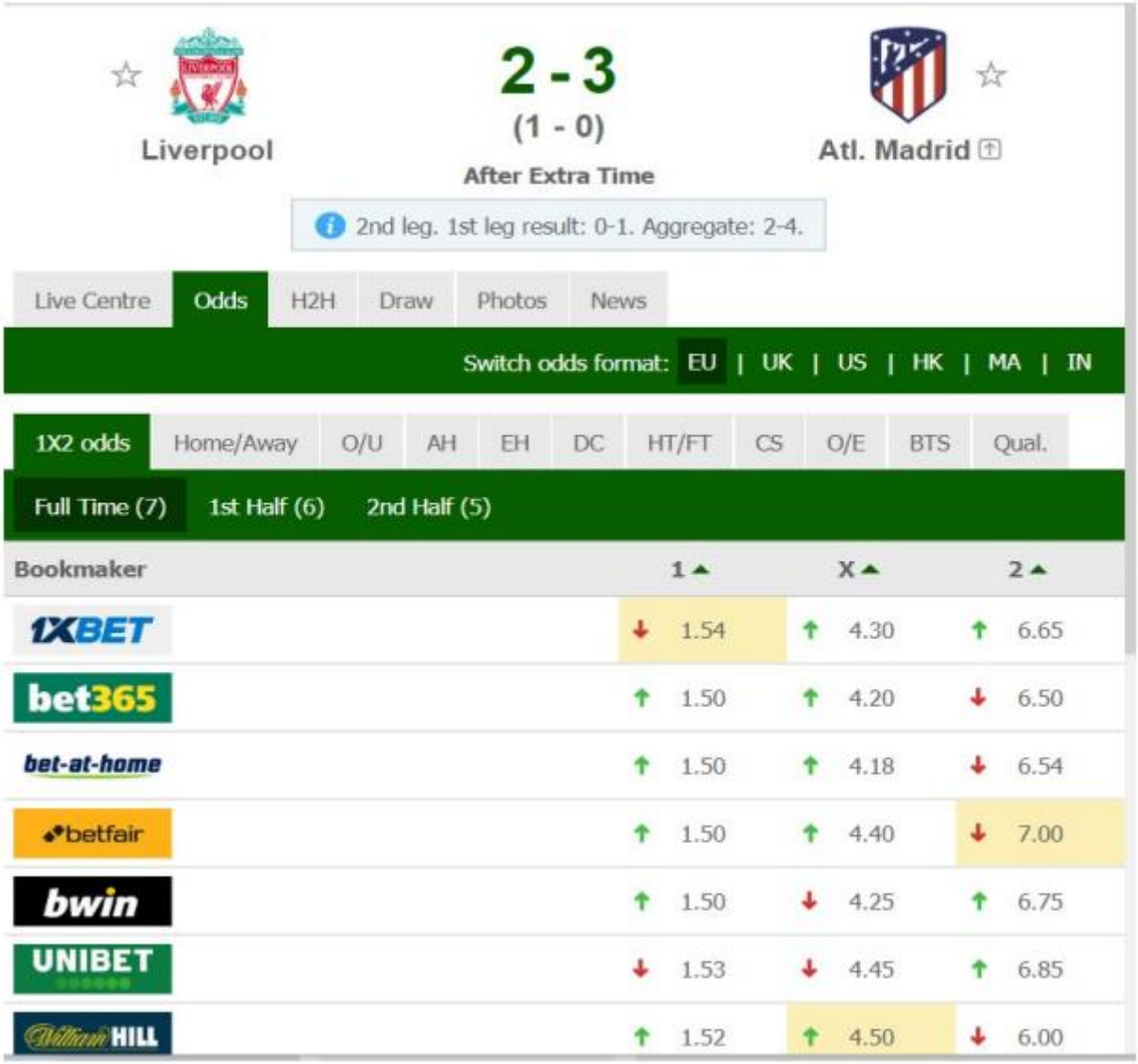


Рисунок 1.5 Приклад даних з ресурсу flashscore.com

Сайт ice2bet надає різну статистику для аналізу матчів та ліг. Серед переваг цього ресурсу можна відзначити можливість перегляду даних для ліг чи окремої команди та варіативність у сортуванні даних (див. рис. 1.6). Недоліками є незрозумілий інтерфейс без пояснень значення деяких типів сортування даних та відсутність в статистиці даних букмекерських компаній.

Рисунок 1.6 Приклад даних з сайту ice2bet.com

Також існують багато API для отримання передбачень на матчі та їх результати в реальному часі. Така система у сукупності з інструментом для візуалізації даних цілком може задовольнити вимоги нашої інформаційної системи. Але майже всі API мають обмежену кількість запитів за певний проміжок часу, що унеможливлює роботу з великою кількістю користувачів. Тому найдоцільніше буде розробити додаток, котрий регулярно буде збирати актуальні дані за певний проміжок часу і зберігати їх у базу даних інформаційної системи. REST (скор. англ. Representational State Transfer, «передача репрезентативного стану») – підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Був описаний і популяризований 2000 року Роєм Філдінгом, одним із творців протоколу HTTP. Прикладни́й програ́мний інтерфе́йс (інтерфейс програмування застосунків, інтерфейс прикладного програмування) (англ. Application Programming Interface, API) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек. 12 Сайт sportmonks надає зручний API для отримання даних про футбольні матчі та передбачення на них

ІІ РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

2.1. Аналіз і специфікація вимог до інформаційної підсистеми

Інформаційна система автоматизованого формування розкладу футбольних матчів повинна задовольняти певні вимоги, щоб бути ефективною та корисною для користувачів. Для того, щоб розробити інформаційну підсистему, необхідно провести аналіз вимог та специфікацію функціональних та нефункціональних вимог.

Функціональні вимоги:

– Можливість створення розкладу матчів для певної ліги або чемпіонату;

– Автоматичне визначення оптимального розташування та часу проведення матчів;

– Можливість додавання нових команд до ліги;

– Перегляд та редагування розкладу матчів;

– Можливість генерувати звіти про проведені матчі та результати;

– Повідомлення користувачів про зміни у розкладі.

Нефункціональні вимоги:

– Інформаційна підсистема повинна бути доступною для використання користувачами з різних пристроїв та операційних систем;

– Система повинна мати інтуїтивно зрозумілий та простий інтерфейс користувача;

– Інформаційна система повинна бути стійкою до помилок та відмов;

– Доступ до системи має бути обмеженим та забезпеченим авторизацією.

Для моделювання інформаційної підсистеми можна використовувати діаграму прецедентів, що дозволить проілюструвати основні взаємодії користувачів з системою та функціональні можливості системи. Також можна

використовувати UML-діаграму класів, щоб показати структуру системи та взаємозв'язки між класами.

Діаграма прецедентів дозволить проаналізувати взаємодії користувачів з системою та виділити основні прецеденти, які має реалізувати інформаційна підсистема. До основних прецедентів можуть відноситися: "Створення розкладу матчів", "Редагування розкладу матчів", "Перегляд розкладу матчів", "Додавання нової команди", "Генерація звітів про матчі", "Повідомлення користувачів про зміни у розкладі".

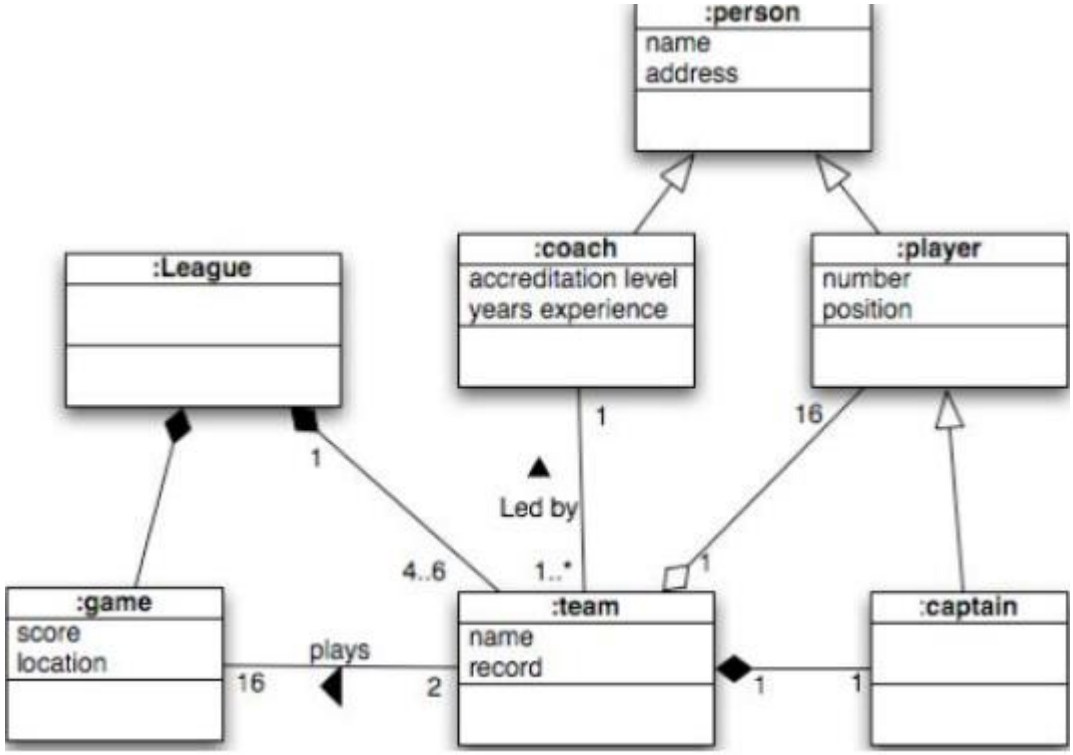


Рисунок 1.1 UML diagram

Аналіз та специфікація вимог до інформаційної підсистеми дозволять розробити систему, що відповідає потребам користувачів та вимогам функціональності системи. Після цього можна переходити до розробки технічного завдання, в якому будуть описані всі необхідні деталі системи, включаючи архітектуру, технології розробки, функціональні та нефункціональні вимоги, розклад робіт та інші важливі питання.

Після розробки технічного завдання можна переходити до проектування системи. В рамках цього етапу можна створити детальну діаграму класів, яка буде описувати всі класи та їх взаємозв'язки, а також діаграми послідовності та

діаграми активності, які будуть показувати послідовність та процес роботи системи.

Крім того, на етапі проектування слід визначити технології розробки, які будуть використовуватися для реалізації системи. Тут можна використовувати різні мови програмування та фреймворки, залежно від потреб системи та навичок розробників.

Після проектування можна переходити до реалізації системи, яка буде включати в себе написання коду, тестування та налагодження системи. Після цього можна впроваджувати систему та забезпечувати її підтримку та оновлення.

Отже, аналіз та специфікація вимог до інформаційної підсистеми є важливим етапом у розробці будь-якої програмної системи, який дозволяє зрозуміти потреби користувачів та розробити систему, що відповідає цим потребам. Після аналізу вимог слід переходити до розробки технічного завдання, проектування системи та реалізації.

2.2. Постановка та алгоритм розв’язання задачі

Дана задача полягає в автоматизованому формуванні розкладу футбольних матчів. Основне призначення задачі полягає в оптимізації процесу формування розкладу, забезпеченні правильної організації матчів та зменшенні витрат часу та ресурсів на цей процес. Розв’язання задачі за допомогою ЕОМ необхідне для покращення якості та ефективності формування розкладу.

Об’єктами управління задачі є футбольні клуби, стадіони, команди та їхні гравці, дати та часи проведення матчів.

Вихідна інформація для розв’язання задачі включає інформацію про футбольні клуби та їхніх гравців, інформацію про стадіони та їхню доступність, інформацію про графіки та періоди, в які необхідно проводити матчі, а також обмеження щодо кількості матчів, що можуть бути проведені однією командою.

Результатом розв’язання задачі є розклад проведення матчів для кожної команди. Розв’язання задачі повинно бути виконано з певною періодичністю, наприклад, на початку кожного футбольного сезону. Термін видачі вихідної

інформації повинен бути відповідним до потреб користувачів і залежить від конкретної ситуації.

Автоматизоване розв’язання задачі припиняється у випадках, коли немає можливості знайти оптимальний розклад або коли виникають непередбачувані обставини, що впливають на проведення матчів.

На підставі відомостей, що наведені в Розділі 1, можна припустити, що інформаційна система автоматизованого формування розкладу футбольних матчів може мати зв'язок з наступними задачами:

- Збір та обробка даних про команди та їх склади
- Вибір локації та стадіону для проведення матчу
- Вибір дати та часу проведення матчу
- Створення графіку тренувань для команд
- Формування списку арбітрів для проведення матчу
- Визначення результатів матчу та оновлення таблиці рейтингу команд

Інформаційна система автоматизованого формування розкладу футбольних матчів може використовувати дані з інших систем або підсистем для забезпечення повної та точної інформації про команди, стадіони, графік тренувань, арбітрів та інші параметри, які необхідні для ефективного формування розкладу.(рис.1.3)

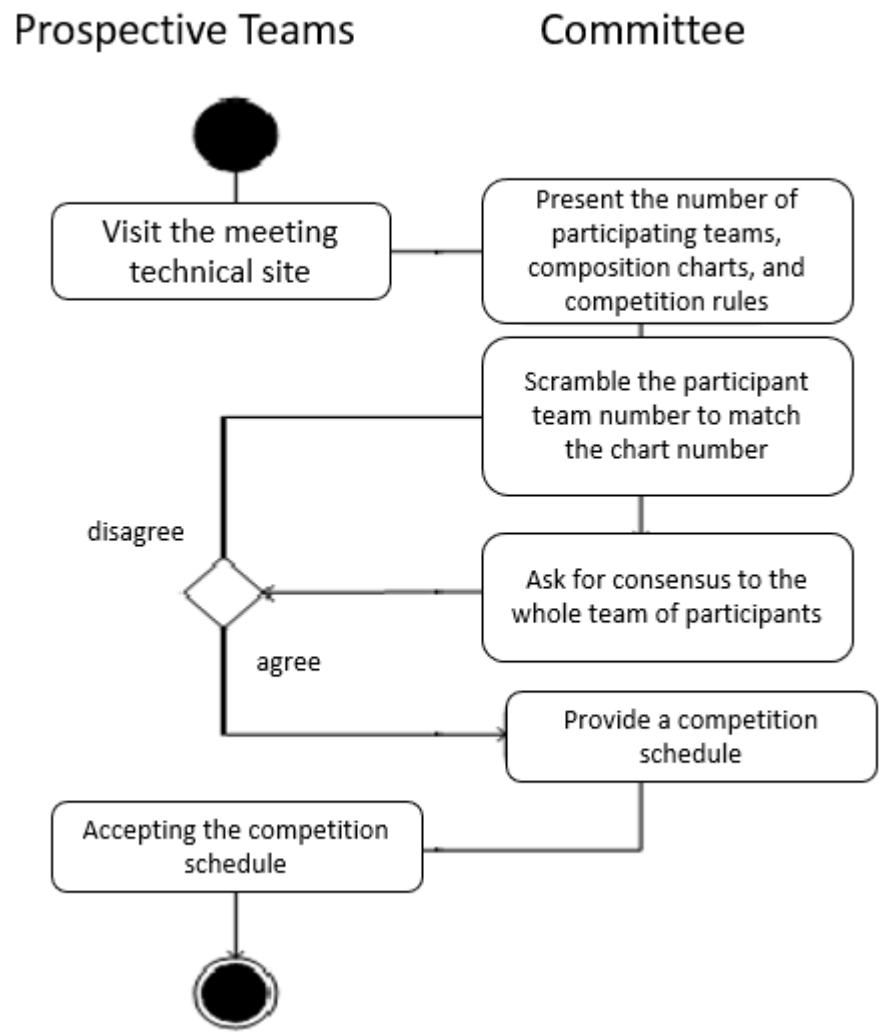


Рисунок 1.3 Інформаційна модель

Вихідна інформація**.** У тексті описати її призначення і використання, а потім дати перелік і опис вихідних повідомлень за формою, наведеною в табл.2.1; описують вихідні машинограми, відеограми та масиви, які формуються під час розв’язання задачі і зберігаються для розв’язання цієї або інших задач.

Таблиця 2.1

Перелік і опис вихідних повідомлень

з/п	Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувачі інформації
	2	3	4	5	6	7

Далі наводяться перелік і опис структурних одиниць вихідних повідомлень, які мають самостійне змістовне значення у вигляді тексту. При цьому необхідно вказати:

– повну назву структурної одиниці інформації (показника);

– ідентифікатор вихідного повідомлення, до складу якого входить відповідна структурна одиниця (показник);

– вимоги до точності та надійності обчислення показника (у разі потреби).

У додатку наводяться ескізи форм вихідних повідомлень, що формуються за автоматизованого розв’язання задачі.

Вхідна інформація**.** У тексті описують її призначення і способи одержання, а потім наводять перелік і опис вхідних повідомлень за формою табл. 2.2. До табл. 2.2 заносять вхідні документи, які будуть використані для формування оперативних файлів бази даних; файли, які потрапляють на вхід задачі чи комплексу з інших задач та їх комплексів, а також документи та файли, які мають довідковий характер і умовно можуть бути віднесені до постійної інформації.

Таблиця 2.2

Перелік і опис вхідних повідомлень

з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
	2	3	4	5	6

Далі наводяться перелік і опис структурних одиниць вхідних повідомлень, які мають самостійне змістовне значення у вигляді тексту. При цьому необхідно вказати:

- повну назву структурної одиниці інформації (показника);
- вимоги до точності числового значення показника (у разі потреби);
- джерела інформації (документ, відеокадр, база даних і т. ін.);
- ідентифікатор джерела інформації.

При описі алгоритму виокремлюють такі підрозділи: «Використовувана інформація»; «Результати розв’язання»; «Математичний опис»; «Алгоритм розв’язання задачі на ЕОМ».

Використовувана інформація**.** У тексті вказують призначення інформації, яку використовують. Потім наводять перелік масивів використовуваної інформації у формі табл 2. 3.

Таблиця2.3

Перелік масивів використовуваної інформації

Масив	Ідентифікатор	Максимальна кількість записів
1	2	3

У табл. 2.3 наводяться масиви, які сформовані з вхідних повідомлень і масивів, сформованих іншими алгоритмами для реалізації даного алгоритму.

Результати розв’язання. У тексті вказують призначення результатів, потім наводять перелік масивів результатної інформації за формою табл.2.4.

Таблиця 2.4

ПЕРЕЛІК МАСИВІВ РЕЗУЛЬТАТНОЇ ІНФОРМАЦІЇ

Масив	Ідентифікатор	Максимальна кількість записів
1	2	3

У табл. 4 наводяться масиви, які сформовані для видачі вихідних повідомлень (машинограм, відеограм та ін.), та масиви, що зберігаються для розв’язання даної або інших задач.

Математичний опис системи автоматизованого формування розкладу футбольних матчів включає наступні основні показники:

1. Кількість команд, які беруть участь в турнірі, позначена як N.
2. Кількість матчів між командами, позначена як M.
3. Кількість турів в турнірі, позначена як T.
4. Розклад матчів між командами, позначений як S.
5. Періодичність проведення турів, позначена як P.

Для розрахунку розкладу матчів можна використовувати наступну математичну модель:

1. Спочатку формується матриця відстаней між командами, яка показує відстань між кожною парою команд у турнірі. Позначимо цю матрицю як D.
2. За допомогою алгоритму класифікації, заснованого на методі K-середніх, команди розбиваються на кластери залежно від їхньої сили та рівня гри. Позначимо ці кластери як C.
3. На основі матриці відстаней та кластерів команд формується матриця суміжності між кластерами. Позначимо цю матрицю як A.

4. За допомогою алгоритму Дейкстри, шукається найкоротший шлях між кластерами в матриці A, який відповідає найбільш оптимальному розкладу матчів. Позначимо цей шлях як P.

5. Розклад матчів формується на основі шляху P та кількості турів T. Кожному туру ставиться у відповідність певний набір матчів, які відповідають шляху P та періодичності проведення турів P.

Математична модель системи автоматизованого формування розкладу футбольних матчів базується на наступних прийнятих допущеннях та оцінках відповідності розробленої моделі реальному процесу:

1. Кожна команда має свій унікальний ідентифікатор.

2. Кожний матч має свій унікальний ідентифікатор, дату та час початку, місце проведення, та дві команди-учасниці.

3. Кожна команда має свій рівень гри, який виражається числом від 0 до 1, де 1 - найвищий рівень гри.

4. Рівень гри команди залежить від її попередніх результатів, які впливають на її рейтинг.

5. Рейтинг команди можна визначити за допомогою математичної формули, яка враховує кількість та результати її попередніх матчів.

6. З метою рівномірного розподілу навантаження на команди, кожна команда повинна зіграти однакову кількість матчів.

7. Всі матчі повинні бути розподілені на певну кількість турів, кожен з яких має свою дату та час проведення.

8. На кожен тур може припадати довільна кількість матчів, проте кожна команда повинна грати тільки один матч на тур.

9. Перед кожним туром система автоматично формує оптимальний розклад матчів, використовуючи алгоритми оптимізації.

10. У разі необхідності, система може вносити зміни до розкладу матчів наступних турів, щоб уникнути конфліктів та забезпечити рівномірний розподіл навантаження на команди.

Математична модель системи автоматизованого формування розкладу футбольних матчів може бути викладена наступним чином:

1. Введення даних:

– Кількість команд, що беруть участь в змаганні (N);

– Кількість турів у змаганні (M);

– Кількість стадіонів, на яких можуть проводитись матчі (S);

– Календар днів, на які можуть бути заплановані матчі;

– Дані про кожну команду (назва, рейтинг, область, вік гравців тощо);

– Обмеження на кількість матчів, які може провести кожна команда.

2. Формування матриці суміжності:

– Кожен стовпець матриці відповідає турі змагання;

– Кожен рядок матриці відповідає команді змагання;

– Значення у комірках матриці вказують на те, чи зустрічаються ці дві команди у відповідному турі, і на якому стадіоні буде проводитись матч.

3. Формування розкладу матчів:

– Використовуючи матрицю суміжності та обмеження на кількість матчів, які може провести кожна команда, формується оптимальний розклад матчів для всіх турів;

– При формуванні розкладу враховуються також географічні та часові обмеження.

4. Розрахунок показників:

– Оцінюється ефективність розкладу (кількість днів, на які припадає по одному матчу, середня відстань між командами тощо);

– Розраховуються також інші показники, такі як загальна кількість матчів, що будуть зіграні, кількість матчів на кожному стадіоні, кількість матчів кожної команди тощо.

Математичні формули для розрахунку показників можуть бути наступними:

1. Формула для розрахунку ефективності розкладу: E = (1/NM) * ∑i=1 to N ∑j=1 to M (d_ij * x_ij)

де E - ефективність розкладу, N - кількість команд, M - кількість матчів, d_ij - відстань між командами i та j, x_ij - булева змінна, що вказує на те, чи зустрічаються команди i та j у матчі.

2. Формула для розрахунку максимальної кількості матчів для однієї команди: M_max = (N-1) * T

де M_max - максимальна кількість матчів для однієї команди, N - кількість команд, T - кількість турів.

3. Формула для розрахунку максимальної кількості матчів для двох команд між собою: M_max_ij = (T-1) + ceil((N-2)/2)

де M_max_ij - максимальна кількість матчів між командами i та j, T - кількість турів, ceil - округлення до більшого цілого числа.

4. Формула для розрахунку мінімальної кількості турів: T_min = ceil(2*N/M_max)

де T_min - мінімальна кількість турів.

5. Формула для розрахунку кількості матчів у турі: M = N/2

де M - кількість матчів у турі, N - кількість команд.

Ці формули були використані для розрахунку показників ефективності розкладу та планування кількості матчів та турів у системі автоматизованого формування розкладу футбольних матчів.

Алгоритм розв’язання задачі на ЕОМ для інформаційної системи автоматизованого формування розкладу футбольних матчів може бути наступним:

1. Введення вхідних даних:

– Кількість команд N.

– Кількість матчів M.

– Розклад забезпечує зіграння кожної команди по одному разу за M матчів.

– Матриця R розмірності NxM, де R(i,j) - це 1, якщо і-та команда грає у j-му матчі, і 0 в іншому випадку.

– Вага матчу, яка може бути від 0 до 1, де 1 означає, що матч має найбільшу вагу.

2. Обчислення вагового коефіцієнту матчу:

– Знаходження кількості матчів, яка відповідає вазі 1.

– Обчислення вагового коефіцієнту як відношення ваги матчу до кількості матчів з такою ж вагою.

3. Ініціалізація параметрів:

– Створення порожнього розкладу матчів.

– Початкове присвоєння мінімального значення ефективності розкладу E_min.

4. Генерація розкладів:

– Генерація всіх можливих розкладів матчів.

– Обчислення ефективності кожного з розкладів за допомогою формули (1).

– Порівняння отриманої ефективності з поточним мінімальним значенням E_min.

– Якщо ефективність поточного розкладу менше за мінімальне значення E_min, то присвоєння його значення E_min.

– Збереження поточного розкладу як оптимального, якщо його ефективність дорівнює мінімальному значенню E_min.

5. Виведення результатів:

– Виведення оптимального розкладу матчів.

– Виведення його ефективності.

Алгоритм розв’язання задачі на ЕОМ для інформаційної системи автоматизованого формування розкладу футбольних матчів може бути наступним:

1. Введення вихідних даних:

a. Кількість команд та їхні назви.

b. Кількість стадіонів та їхні назви.

c. Кількість днів, на які формується розклад.

d. Кількість матчів на день, що є допустимою нормою.

e. Матриця відстаней між стадіонами.

2. Формування списку матчів:

a. Створення матриці команд, де елемент (i, j) відповідає матчу між командами i та j.

b. Для кожної дати формується список матчів, які можуть бути проведені. c. Список матчів сортується за зменшенням середнього рангу команд.

3. Формування розкладу:

a. Для кожної дати формується список матчів, які можуть бути проведені та не конфліктують з іншими матчами.

b. Якщо не вдалося сформувати список матчів для дати, то вона пропускається.

c. Розклад записується у вигляді таблиці, де рядки відповідають датам, а стовпці - стадіонам.

4. Розрахунок показників:

a. Обчислення ефективності розкладу за формулою E = (1/NM) * ∑(Cij * Dij), де N - кількість днів, M - кількість стадіонів, Сій - коефіцієнт важливості матчу, Дій - відстань між стадіонами.

b. Обчислення кількості конфліктів у розкладі.

c. Обчислення середньої відстані, яку доводиться проїхати командам для проведення матчів.

5. Виведення результатів:

a. Розклад матчів.

b. Значення показників.

c. Графік ігор

6. Обробка помилок:

a. Перевірка правильності введених даних.

b. Обробка помилок під час розрахунків.

c. Виведення повідомлень про помилки та рекомендації щодо їх виправлення.

7. Збереження результатів:

a. Збереження розкладу матчів у відповідному форматі (наприклад, CSV, XML).

b. Збереження значень показників у відповідному форматі.

c. Збереження графіків, якщо вони створюються.

8. Вихід з програми.

Алгоритм у вигляді схеми:

Алгоритм розв’язання задачі можна подати у вигляді схеми за Державним стандартом 19.701—90. Нижче наведено загальний варіант схеми алгоритму:

1. Початок роботи.
2. Ввід даних про команди і обмежень.
3. Створення матриці суміжності.
4. Створення графу та обробка графу.
5. Розрахунок показників.
6. Виведення результатів.
7. Обробка помилок.
8. Збереження результатів.
9. Кінець роботи.

Інформаційна система автоматизованого формування розкладу футбольних матчів - це комп'ютерна програма, що розроблена з метою забезпечення зручного та ефективного процесу формування розкладу футбольних матчів. Система відповідає функціональним вимогам та постановці задачі, що були розроблені студентом у попередніх підрозділах.

Моделювання поведінки системи включає в себе діаграму прецедентів, діаграми послідовності та діаграми діяльності. Діаграма прецедентів документує функціональні вимоги у вигляді варіантів використання як типової взаємодії

системи з акторами. Склад та обсяг системи, що її подає діаграма прецедентів, відповідає функціональним вимогам до системи та постановці задачі.

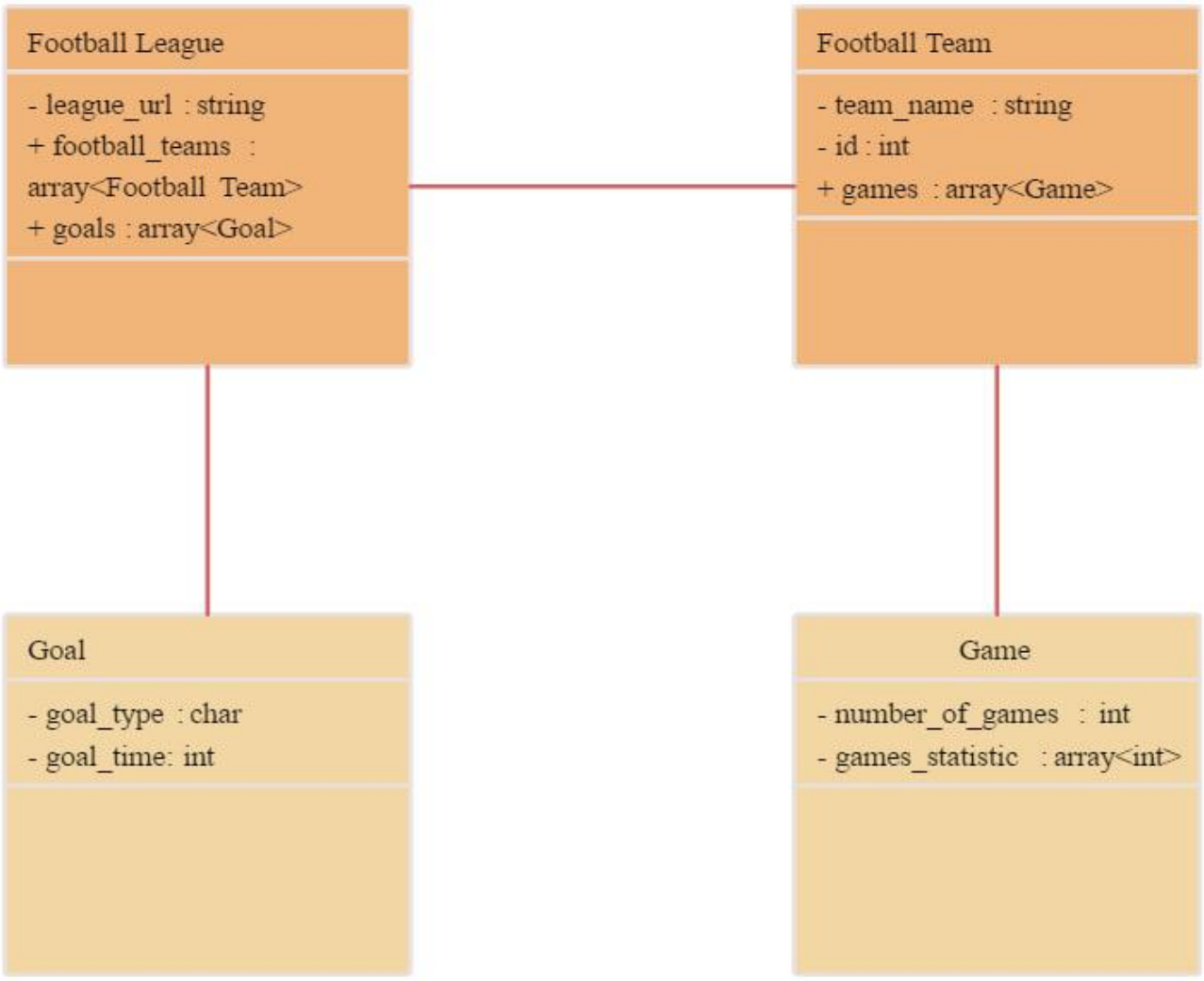
Діаграми послідовності розроблені для кожного прецеденту у відповідності до сценаріїв варіантів використання. Ці діаграми дозволяють виокремити всі об’єкти (компоненти) проектованої системи та показати послідовність їх взаємодії під час виконання варіанту використання.

Також, для систем з великою кількістю паралельних процесів рекомендується наводити опис поведінки у формі діаграм діяльності. Ці діаграми дозволяють показати послідовність дій та об'єктів, що беруть участь у процесі формування розкладу футбольних матчів.

Отже, за допомогою діаграм прецедентів, діаграм послідовності та діаграм діяльності система автоматизованого формування розкладу футбольних матчів забезпечує ефективний та зручний процес.

Інформаційна система автоматизованого формування розкладу футбольних матчів повинна мати чітку структуру класів, яка відображає її функціональні можливості та зв'язки між компонентами. Для цього рекомендується використовувати діаграми класів UML.

На діаграмах класів мають бути подані пакети класів та їх взаємозв'язки, підмножини класів з атрибутами та операціями класів, а також граничні класи та класи керування (рис. 1.4).



Риксунок 1.4 Діаграма касів

Класи-сутності, що відображають дані та поведінку з довгим життєвим циклом, є важливою складовою моделі структури системи. Граничні класи використовуються для моделювання інтерфейсів системи на високому рівні, а класи керування створюються для моделювання послідовної поведінки одного або кількох прецедентів та координації подій, що реалізують закладену в ці прецеденти поведінку.

Усі ці елементи моделі повинні бути чітко відображені на діаграмах класів, щоб забезпечити прозорість та зрозумілість структури системи. Це допоможе покращити якість проектування та зменшити ризик помилок під час розробки.

Інформаційна система автоматизованого формування розкладу футбольних матчів потребує ретельної перевірки розроблених проектних рішень

на етапі розподілу вимог за компонентами системи. Для цього рекомендовано застосовувати діаграму трасування та матриці взаємозв'язків.

Діаграма трасування є інструментом для візуалізації взаємозв'язків між вимогами та компонентами системи. На цій діаграмі повинні бути подані високорівневі вимоги, прецеденти та/або інші елементи моделі системи, а також відповідні зв'язки між ними. У разі великої кількості сутностей можна створити кілька діаграм за умов збереження логіки трасування.

Матриці взаємозв'язків доповнюють діаграму трасування та надають можливість більш детальної перевірки взаємозв'язків між вимогами та компонентами системи. У матрицях відображається, які вимоги пов'язані з якими компонентами системи, що дозволяє виявити можливі проблеми та помилки в проектних рішеннях.

Таким чином, застосування діаграми трасування та матриці взаємозв'язків є необхідним етапом верифікації розроблених проектних рішень для Інформаційної системи автоматизованого формування розкладу футбольних матчів.

ІІІ ПРОЕКТУВАННЯ КОМПОНЕНТІВ ПІДСИСТЕМИ

3.1. Інформаційне забезпечення

Інформаційна система автоматизованого формування розкладу футбольних матчів є комплексною програмно-апаратною системою, що забезпечує автоматизоване формування розкладу футбольних матчів. Вона має складну структуру і працює на базі спеціальної комп'ютерної системи управління базами даних.

Обрано СУБД, що відповідає вимогам до надійності і швидкодії, а також забезпечує можливість розширення та додавання нових функцій до системи.

Для забезпечення вірогідності та точності даних, в ІС автоматично використовуються методи контролю та верифікації даних.

Основні елементи системи включають в себе програмні модулі для формування розкладу матчів, інформаційні бази даних, інтерфейс користувача та засоби контролю даних.

Загальна схема ІЗ включає в себе збір та обробку даних про команди та стадіони, розподіл матчів на календар, формування графіка матчів, збереження та передачу результатів матчів. Усі ці елементи системи допомагають забезпечити ефективне та точне формування розкладу футбольних матчів.

Інформаційна система автоматизованого формування розкладу футбольних матчів потребує організації збору та передавання первинної інформації.

До переліку джерел та носіїв інформації можна віднести внутрішні/зовнішні підрозділи організацій та установ, Інтернет та базу даних. Кожне вхідне повідомлення має бути супроводжене вказівкою на джерело інформації, яке забезпечує своєчасне подання інформації на оброблення та формування вхідних документів у пам’яті ЕОМ.

Важливо вказати періодичність та спосіб надання інформації, а також її носії. Загальні вимоги до організації збору та передання інформації повинні бути чітко визначені, щоб забезпечити надійність та достовірність інформації.

Організація збору та передавання первинної інформації є важливою складовою ефективної роботи ІС автоматизованого формування розкладу футбольних матчів.

Інформаційна система автоматизованого формування розкладу футбольних матчів передбачає побудову системи класифікації та кодування. Для класифікації об’єктів будуть використовуватися різні системи, зокрема за країнами, лігами, командами тощо.

Для цього будуть використані відповідні класифікатори та коди, з назвами та структурою, які забезпечать уніфікований підхід до класифікації об’єктів.

Кожен код буде мати свій метод кодування, структуру та довжину, що забезпечить можливість ефективного використання системи класифікації та кодування в процесі автоматизованого формування розкладу футбольних матчів.

У додатках до системи наведені фрагменти та приклади класифікаторів, що полегшать розуміння та використання системи класифікації та кодування.

Інформаційна система автоматизованого формування розкладу футбольних матчів потребує розробки форм первинних документів, машинограм та відеокадрів.

При проектуванні форм документів необхідно враховувати їх склад, зміст та форму представлення, а також встановлювати вимоги до їх форматування. До складу первинних документів входять: акти, протоколи, звіти, розклади, плани та інші.

Кожен документ має своє найменування та ідентифікатор, визначений за допомогою системи класифікації та кодування. Передбачається, що документи будуть видачі користувачам в паперовій та електронній формі. Для цього необхідно встановити періодичність та терміни видачі документів.

Для відправлення документів електронною поштою будуть використовуватися спеціальні програми та захист від несанкціонованого доступу. Враховуючи специфіку інформації, були використані відеокадри та машинограми для передачі необхідної інформації.

Кожен запис у масивах містить такі поля: найменування (Name), ідентифікатор (ID), формат (Format), бізнес-правила (Business Rules), логічні чи семантичні зв'язки (Logical and Semantic Relationships). Крім того, у записі можуть бути первинний/вторинний ключ, умова на значення, обов'язкове поле і індексне поле.

Найменування поля вказує на те, яка інформація міститься в полі. Ідентифікатор поля використовується в програмі для доступу до відповідного поля запису. Формат поля визначає тип даних, що містяться в полі, наприклад, текст, числа, дата, час тощо.

Бізнес-правила вказують на обмеження і правила, які має відповідати значення поля. Логічні і семантичні зв'язки описують взаємозв'язки між різними полями запису, які можуть бути важливі при пошуку і фільтрації даних.

Первинний ключ - це унікальний ідентифікатор запису в масиві. Вторинний ключ може бути використаний для пошуку записів за іншими полями, які не є унікальними. Умова на значення встановлює обмеження на значення поля, наприклад, може бути встановлено максимальне або мінімальне значення.

Обов'язкове поле вказує на те, що поле має бути заповнене для кожного запису в масиві. Індексне поле використовується для прискорення пошуку записів у масиві за значенням поля.

Усі ці поля та їх значення є важливими для коректної роботи інформаційної системи автоматизованого формування розкладу футбольних матчів.

Коректна структура інформаційних масивів допоможе ефективно зберігати та швидко отримувати інформацію про розклад футбольних матчів.

Для розробки інформаційної системи автоматизованого формування розкладу футбольних матчів необхідно вибрати СКБД, що забезпечить ефективне зберігання та операції з даними. У зв'язку з цим, обрано реляційну СКБД - PostgreSQL. Причиною цього вибору є висока швидкодія, стабільність та надійність системи. Крім того, PostgreSQL забезпечує гарну масштабованість, що важливо для інформаційної системи, яка може зростати з часом.

Вимоги до системи включають підтримку транзакцій, забезпечення цілісності даних та високої продуктивності під час виконання запитів.

Обмеження включають необхідність підтримки багатьох користувачів та можливість резервного копіювання та відновлення даних. Усі ці вимоги та обмеження мають бути враховані на етапі проектування системи.

Інформаційна система автоматизованого формування розкладу футбольних матчів передбачає створення бази даних, яка буде зберігати всю необхідну інформацію про команди, гравців, місця проведення матчів, розклади і результати гри.

Для створення цієї бази даних було обрано пакет автоматизації проектування - AllFusion Erwin Data Modeler. Інфологічна модель бази даних, яка була запроектована, повинна бути нормалізованою. Крім того, у додатках до проекту необхідно надати короткий посібник, який допоможе користувачам створити інфологічну (логічну) модель бази даних з використанням CASE-засобів. Цей посібник має містити екранні форми, які проілюструють процес проектування бази даних.

Якщо проект передбачає розробку OLAP-системи, тоді необхідно додатково проектувати сховище даних. При проектуванні сховища, необхідно обґрунтувати вибір ROLAP моделі та описати структуру бази даних, яка буде джерелом даних (Data Sources) для сховища. Розглядаються дві ROLAP моделі: зірка та сніжинка. Обрана модель повинна відповідати вимогам проекту та забезпечувати оптимальну швидкість запитів до бази даних.

Після проектування сховища даних, необхідно переконатися, що дані, які будуть зберігатися у сховищі, будуть максимально корисні для потреб проекту.

Інформаційна система автоматизованого формування розкладу футбольних матчів має в своєму складі базу даних, що складається з інфологічної та даталогічної моделей. Для проектування інфологічної моделі використовуються CASE-засоби, зокрема пакет AllFusion Erwin Data Modeler. Проектована база даних має бути нормалізованою. Для користувачів системи

надається стисле керівництво з розробки інфологічної моделі, яке ілюструється екранними формами.

У даталогічній моделі бази даних представлені складові бази чи сховища даних в термінах мови опису даних (DDL), що містять фізичну модель бази даних та опис контрольного (тестового) прикладу основних таблиць. Діаграма бази даних наводиться в середовищі вибраної системи керування базами даних. Для завантаження таблиць реляційної бази чи сховища даних надаються форми первинних документів та контрольний (тестовий) приклад, який дозволяє перевірити правильність проектування бази даних.

Запити до бази даних формуються на мові SQL та наводяться в окремому параграфі. Всі ці елементи допомагають створити ефективну інформаційну систему автоматизованого формування розкладу футбольних матчів, яка дозволяє швидко та точно формувати розклад матчів, а також забезпечує зберігання необхідних даних для проведення аналітичних операцій та видачі звітів.

Інформаційна система автоматизованого формування розкладу футбольних матчів повинна бути розташована на сервері, який може бути установлений на об’єкті управління чи у зовнішньому дата-центрі.

Сервер повинен бути оснащений необхідними комплексами технічних засобів (КТЗ), які забезпечують функціонування системи. Зокрема, це можуть бути пристрої для збереження даних, процесори та інші пристрої, необхідні для роботи системи.

Крім того, система повинна мати доступ до мережі передачі даних для збереження та обміну інформацією про матчі. Оскільки система автоматизована, вона здатна працювати у режимі 24/7, що забезпечує постійну доступність та можливість отримання актуальної інформації про розклад матчів.

Для реалізації інформаційної системи автоматизованого формування розкладу футбольних матчів необхідно використовувати комплекс технічних засобів (КТЗ), що включає в себе ряд пристроїв та програмного забезпечення.

Для забезпечення швидкості та надійності роботи системи було обрано сервер з достатньою потужністю та оперативною пам'яттю.

Також у складі комплексу передбачено використання бази даних, яка забезпечить можливість швидкого та ефективного зберігання та обробки великої кількості інформації.

Для взаємодії з сервером та базою даних необхідно використовувати спеціалізоване програмне забезпечення, що дозволить розробити потрібний інтерфейс та забезпечити взаємодію з іншими комп'ютерами та пристроями.

Всі компоненти комплексу повинні працювати у відповідності з технічними вимогами та стандартами, що забезпечить надійність та ефективність роботи системи.

Автоматизоване робоче місце (АРМ) є основним інструментом для роботи з інформаційною системою автоматизованого формування розкладу футбольних матчів. АРМ складається з монітора, клавіатури, миші, принтера та спеціалізованого програмного забезпечення для роботи з системою.

Функціональні можливості АРМ включають в себе можливість внесення та редагування даних про футбольні матчі, побудову розкладу, вивід інформації про матчі на екран, друк розкладу та інші функції.

Кожен елемент АРМ має свою важливу роль у взаємодії з інформаційною системою, тому їх правильний вибір та налаштування гарантує ефективну роботу з системою формування розкладу футбольних матчів.

Схема мережі передачі даних є важливим елементом інформаційної системи автоматизованого формування розкладу футбольних матчів. У разі, якщо система передбачає мережевий режим взаємодії, необхідно розробити структуру мережі та вибрати оптимальний протокол передачі даних.

Відповідно до специфіки завдання, можна використовувати різні технології передачі даних, такі як Ethernet, Wi-Fi або Bluetooth. У системі також повинен бути визначений сервер для зберігання та обробки даних, а також допоміжні сервери для забезпечення резервного копіювання інформації та забезпечення безперебійної роботи мережі.

Підключення користувачів до мережі може здійснюватися як через провідну, так і бездротову мережу, залежно від потреб і можливостей. Окремою складовою частиною мережі є засоби захисту інформації від несанкціонованого доступу, а також засоби моніторингу та діагностики стану мережі та її компонентів.

Усі ці складові компоненти мережі повинні працювати у взаємодії та забезпечувати швидку та надійну передачу даних між компонентами системи.

3.2. Функціонал користувача

Основним функціоналом користувача є відображення візуалізованих даних, відсортованих за окремими критеріями. Для користувача реалізовано чотири вебсторінки. 34 Головна сторінка дає можливість обрати варіант сортування даних (див. рис. 3.2).

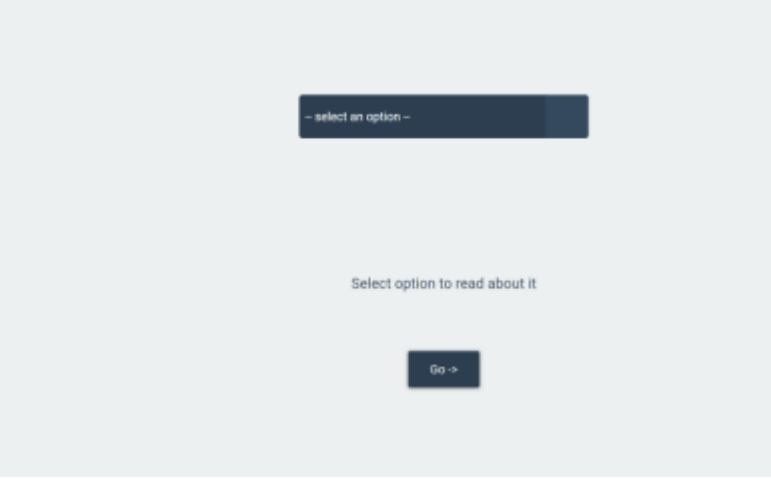


Рисунок 3.2 Головна сторінка

Необхідно обрати один з трьох варіантів сортування даних. При виборі з’явиться його опис (див. рис. 3.3).

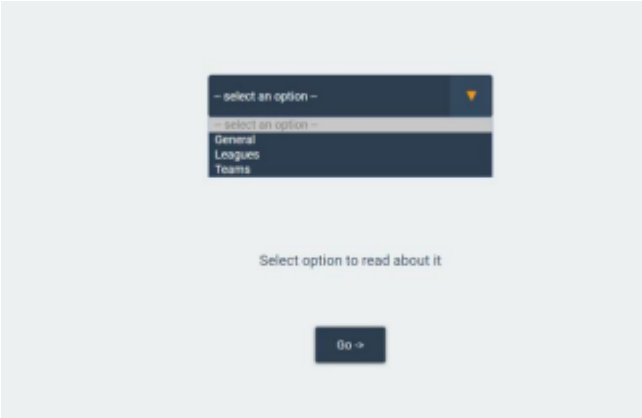


Рисунок 3.3 Варіанти сортування даних

При виборі другого варіанту з’являється додаткове поле зі списком ліг. Обравши третій варіант, спочатку з’являється додаткове поле зі списком ліг, а після вибору ліги з’являється ще одне поле зі списком команд цієї ліги (див. рис. 3.4).

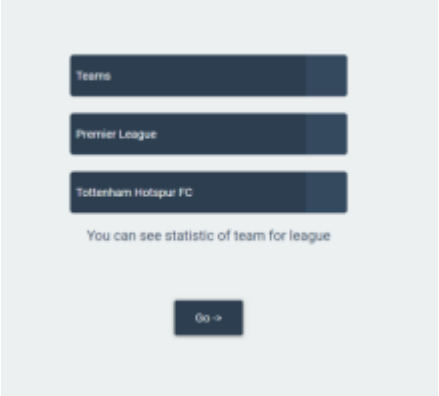


Рисунок 3.4 Сортування даних для окремої команди

Загальна діаграма показує кількість помилок для кожної ліги сумарно від усіх букмекерів у системі та кількість зіграних матчів (див. рис. 3.5). За замовчуванням, мінімальна різниця для похибки дорівнює одиниці. Для його зміни на кожній зі сторінок з даними є можливість налаштувати це значення. При цьому діаграма оновиться без перезавантаження сторінки.

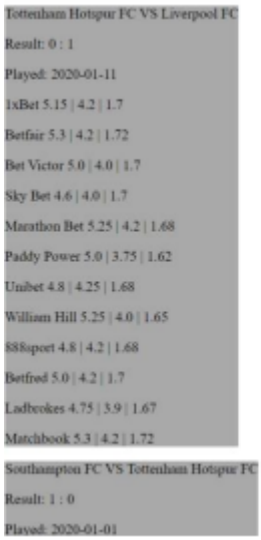


Рисунок 3.9 Інформація про матчі команди в конкретному місяці

У заголовку вказується назва команди і ліга, для якої відображається статистика. Це уточнюється, адже деякі команди мають матчі в двох лігах – своєї країни та Ліги Чемпіонів. При виборі певного значення (не важливо якого графіку), ми побачимо всі ігри відповідного місяця і передбачення всіх букмекерів на цей матч

3.3. Функціонал адміністратора

Основним функціоналом адміністратора є встановлення зв’язків між командами, отриманими з різних API. Найпростіший і найбільш надійний спосіб це робити – вручну. Проблема виникає через різні назви команд в залежності від API та навіть ліги, у якій вони грали матч. Так, команда FC Internazionale Milano може також мати назву FC Internazionale або Internazionale Milano. Також проблема може виникнути через наявність літер, відмінних від англійського алфавіту. Щоб потрапити на сторінку, необхідно ввести пароль адміністратора (див. рис. 3.10). У подальшому він буде переданий на сервер та порівняється з зашифрованим за допомогою ByCrypt паролем. 39 bcrypt – адаптивна криптографічна функція формування ключа, що використовується для безпечного зберігання паролів. Розробники: Нільс Провос і David Mazières.

Функція заснована на шифрі Blowfish, вперше представлена на USENIX у 1999 році [9].

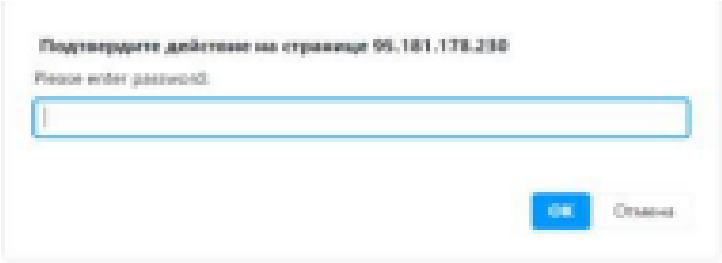


Рисунок 3.10 Запит на ведення пароля адміністратора

При невірному введені пароля сторінка не буде відображена. Для повторного введення пароля, якщо спроба була невдала, необхідно перезавантажити сторінку. Сторінка складається з трьох основних частин – таблиці, що містить вже відсортовані команди (див. рис. 3.11); таблиці, у якій відображається пара команд, яка буде додана до списку відсортованих; список всіх команд.

Рисунок 3.11 Таблиця з відсортованими командами

Таким чином, кожну пару команд можна видалити. Тоді в списку всіх команд вони не будуть позначатися як вже відсортовані. У таблиці всіх команд сірим кольором відображається команда, яка вже має відповідну пару з іншого API. Білим кольором – та, котра не має відповідника. Такі команди теж можуть бути, адже не на всі матчі букмекери робили передбачення (див. рис. 3.12).

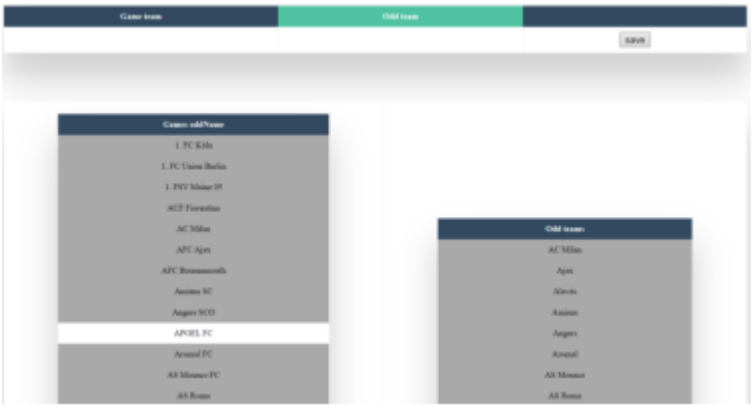


Рисунок 3.12 Таблиця з обраними командами та таблиці зі списками всіх команд

Обрана команда підсвічується кольором (див. рис. 3.13).

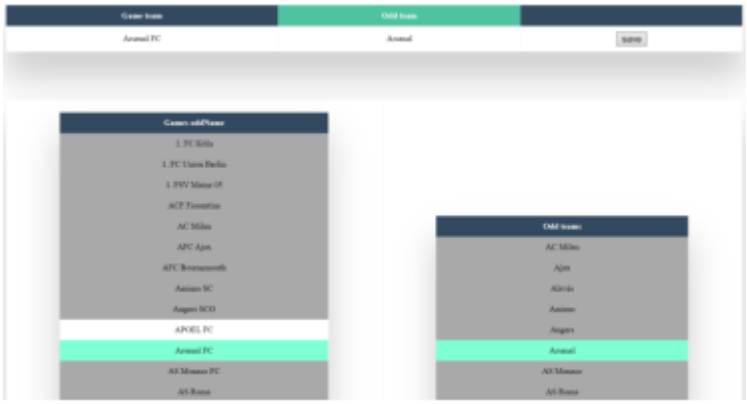


Рисунок 3.13 Обирання команд для додавання до списку відсортованих

При натисканні на команду вона додається до форми зберігання. Після натискання на кнопку «зберегти» пара буде додана до загального списку відсортованих команд.

ВИСНОВКИ

Після проведеного аналізу проблем дослідження виявлено, що наявні подібні додатки, які надають візуалізовані дані для користувачів з метою подальшого аналізу, такі як oddsportal.com, soccerstats.com, whoscored.com, flashscore.com, ice2bet.com. Також були розглянуті сервіси, які надають інформацію у форматі JSON без її візуалізації, зокрема sportmonks.com, allsportsapi.com, apifootball.com, football-data.org, the-odds-api.com. Були проаналізовані ці варіанти і визначено їх переваги та недоліки. За результатами дослідження виявлено, що наявні рішення не задовольняють вимоги нашої інформаційної системи через відсутність наочності або неможливість регулярного отримання даних.

Виходячи з аналізу проблем, було сформульовано наступні задачі дослідження. Потрібно регулярно отримувати дані з двох джерел - сервісу, який надає передбачення для матчів, та сервісу з результатами ігор (the-odds-api.com та football-data.org).

У результаті проектування бази даних були створені DFD-діаграма (для відображення потоків інформації в системі) та ERD-діаграма (для зображення основних сутностей системи та зв'язків між ними).

Для реалізації веб-додатку інформаційної системи була обрана мова програмування Java та шаблон проектування MVC. Під час розробки інформаційної системи було описано загальну структуру веб-додатку та надано UML-діаграму всіх класів. Для кращого розуміння архітектури додатку було детально описано структурно-значущі класи. Веб-додаток має три варіанти візуалізації.

Результатом розробки інформаційної системи є додаток, який здатен візуалізувати дані про футбольні матчі. Переглянути працюючий прототип можна за посиланням http://95.181.178.230. Отже, поставлені задачі були виконані. Працюючий прототип інформаційної системи свідчить про досягнення мети дослідження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бєлка Д.О. Моделі прогнозування футбольних матчів – URL: https://naub.oa.edu.ua/2019/моделі-прогнозування-футбольних-мат/

2. Groll A., Ley C., Schauberger G., Van Eetvelde H. – URL: https://arxiv.org/pdf/1806.03208v2.pdf

3. Freitag D. Documentation – URL: https://www.footballdata.org/documentation/api

4. The odds-api – URL: https://the-odds-api.com

5. Silberschatz A., Sudarshan S. Database system concepts. New York: McGrawHill., 2011.

6. Data Flow Diagram – URL: https://web.archive.org/web/20061023174119/http://www.infoarchgroup.com/qrdfd.htm

7. Spark – URL: http://sparkjava.com

8. Maven – URL: http://maven.apache.org/what-is-maven.html

9. Provos N., Mazières D., A Future-Adaptable Password Scheme. USENIX Annual Technical Conference. 81–92. 2012

10. Кей С. Java SE 8. Вводный курс. «Вильямс». 2014.

11. PL/pgSQL — процедурный язык SQL – URL: https://postgrespro.ru/docs/

12. jQuery.ajax() – URL: https://api.jquery.com/jquery.ajax/

13. HTML The language for building web pages – URL: https://www.w3schools.com

14. Chart.js Simple yet flexible JavaScript charting for designers & developers – URL: https://www.chartjs.org

15. Borovikov R. Top 10 JavaScript Charting Libraries for Every Data Visualization Need – URL: https://hackernoon.com/10-javascript-chartinglibraries-data-visualization-b77523d23372

ДОДАТКИ

```
import controllers.*;
import static spark.Spark.*;
public class Main {
 public static void main(String[] args){
 AdminController adminController = new AdminController();
 SelectorPageController getSelectorPage = new SelectorPageController();
 GeneralDiagramController generalDiagramController = new
GeneralDiagramController();
 ByTeamController byTeamController = new ByTeamController();
 ByLeagueController byLeagueController = new ByLeagueController();
 port(80);
 staticFileLocation("/public");
 get("/adminPage", adminController.getAdminPage);
 post("/resolveTeam", adminController.resolveTeam);
 post("/deleteResolvedTeam", adminController.deleteResolvedTeam);
 post("/checkPassword", adminController.checkPassword);
 get("/mainPage", getSelectorPage.getSelectorPage);
 get("/getLeagues", getSelectorPage.getLeagues);
 get("/getTeams", getSelectorPage.getTeams);
 get("/showGeneral", generalDiagramController.getGeneralDiagram);
 get("/showGeneralWithDiff",
generalDiagramController.getGeneralDiagramWithDiff);
 get("/showTeamStat", byTeamController.getGeneralDiagram);
 get("/showTeamStatWithDiff",
byTeamController.getGeneralDiagramWithDiff);
 get("/teamGamesIn", byTeamController.getGamesForMonth);
 get("/showLeagueStats", byLeagueController.getLeagueStats);
 get("/showLeagueStatsWithDiff",
byLeagueController.getLeagueStatsWithDiff);
```

```
 get("/", getSelectorPage.getSelectorPage);
 }
}
package controllers;
import dao.TeamsDao;
import spark.ModelAndView;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;
import utils.PasswordValidator;
import java.util.HashMap;
public class AdminController {
 private TeamsDao teamsDao= new TeamsDao();
 public final Route getAdminPage = ((request, response) -> {
 HashMap<String, Object> params = new HashMap<>();
 params.put("gameTeam", teamsDao.getGameTeams());
 params.put("oddTeam", teamsDao.getOddTeams());
 params.put("totalTeam", teamsDao.getTotalTeams());
 return new ThymeleafTemplateEngine().render(new
ModelAndView(params,"adminPage"));
 });
 public final Route checkPassword = ((request, response) -> {
 String password = request.queryParams("password");
 if(PasswordValidator.isPasswordsEqual(password)){
 request.session().attribute("auth", true);
 System.out.println("true");
 return true;
 }
 return "false";
 });
 public final Route resolveTeam = ((request, response) -> {
```

```
 if(!(Boolean) request.session().attribute("auth")) return "error";
 int gameId = Integer.parseInt(request.queryParams("gameId"));
 int oddId = Integer.parseInt(request.queryParams("oddId"));
 teamsDao.resolveTeams(gameId, oddId);
 response.status(200);
 return "ok";
 });
 public final Route deleteResolvedTeam = ((request, response) -> {
 if(!(Boolean) request.session().attribute("auth")) return "error";
 int gameId = Integer.parseInt(request.queryParams("gameId"));
 int oddId = Integer.parseInt(request.queryParams("oddId"));
 teamsDao.deleteResolvedTeam(gameId, oddId);
 response.status(200);
 return "ok";
 });
}
package controllers;
import dao.LeagueDao;
import model.Mistake;
import org.json.JSONArray;
import org.json.JSONObject;
import spark.ModelAndView;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;
import java.util.HashMap;
import java.util.List;
public class ByLeagueController {
 LeagueDao leagueDao = new LeagueDao();
 public final Route getLeagueStats = ((request, response) -> {
 int league_id = Integer.parseInt(request.queryParams("league_id"));
```

```
        List<Mistake> leagueStatsByTeam = leagueDao.getLeagueStatsByTeam(1,
league_id);
        HashMap<String, Object> model = new HashMap<>();
        JSONObject data = new JSONObject();
        JSONArray labels = new JSONArray();
        JSONArray mistakes = new JSONArray();
        for (Mistake teamMistake : leagueStatsByTeam) {
        labels.put(teamMistake.getName());
        mistakes.put(teamMistake.getMistakes());
        }
        data.put("labels", labels);
        JSONArray datasets = new JSONArray();
        JSONObject mistakesDataset = new JSONObject();
        mistakesDataset.put("label", "Mistakes");
        mistakesDataset.put("data", mistakes);
        JSONArray colors = new
       JSONArray().put("#FF6633").put("#FFB399").put("#FF33FF").put("#FFFF99
").put("#00B3E6").put
        (

        "#E6B333").put("#3366E6").put("#999966").put("#99FF99").put("#B34D4D")
.put(

        "#80B300").put("#809900").put("#E6B3B3").put("#6680B3").put("#66991A")
.put(

        "#FF99E6").put("#CCFF1A").put("#FF1A66").put("#E6331A").put("#33FFC
C").put(
```

"#66994D").put("#B366CC").put("#4D8000").put("#B33300").put("#CC80CC").put(

"#66664D").put("#991AFF").put("#E666FF").put("#4DB3FF").put("#1AB399").put(

"#E666B3").put("#33991A").put("#CC9999").put("#B3B31A").put("#00E680").put(

"#4D8066").put("#809980").put("#E6FF80").put("#1AFF33").put("#999933").put(

"#FF3380").put("#CCCC00").put("#66E64D").put("#4D80CC").put("#9900B3").put(

"#E64D66").put("#4DB380").put("#FF4D4D").put("#99E6E6").put("#6666FF");

```
mistakesDataset.put("backgroundColor", colors);
datasets.put(mistakesDataset);
data.put("datasets", datasets);
model.put("json", data.toString());
model.put("message", leagueDao.getLeagueById(league_id).getOdd_name());
return new ThymeleafTemplateEngine().render(new ModelAndView(model,
"leagueStats"));
});
public final Route getLeagueStatsWithDiff = ((request, response) -> {
int league_id = Integer.parseInt(request.queryParams("league_id"));
String diff = request.queryParams("diff");
List<Mistake> leagueStatsByTeam =
leagueDao.getLeagueStatsByTeam(Double.parseDouble(diff), league_id);
```

```
JSONArray mistakes = new JSONArray();
for (Mistake teamMistake : leagueStatsByTeam) {
mistakes.put(teamMistake.getMistakes());
}
System.out.println(mistakes.toString());
return mistakes.toString();
});
}
package controllers;
import dao.LeagueDao;
import dao.TeamsDao;
import model.*;
import org.json.JSONArray;
import org.json.JSONObject;
import spark.ModelAndView;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;
import java.util.HashMap;
import java.util.List;
public class ByTeamController {
TeamsDao teamsDao = new TeamsDao();
LeagueDao leagueDao = new LeagueDao();
public final Route getGeneralDiagram = ((request, response) -> {
int teamId = Integer.parseInt(request.queryParams("team_id"));
int league_id = Integer.parseInt(request.queryParams("league_id"));
List<Mistake> teamMistakes = teamsDao.teamMistakes(1.0, teamId,
league_id);
League league = leagueDao.getLeagueById(league_id);
Team team = teamsDao.getTeamById(teamId);
HashMap<String, Object> params = new HashMap<>();
```

```
params.put("json", generateDataJson(teamMistakes));
params.put("league", league.getOdd_name());
params.put("team", team.getName());
return new ThymeleafTemplateEngine().render(new ModelAndView(params,"teams"));
});
public final Route getGeneralDiagramWithDiff = ((request, response) -> {
String diff = request.queryParams("diff");
int teamId = Integer.parseInt(request.queryParams("team_id"));
int league_id = Integer.parseInt(request.queryParams("league_id"));
List<Mistake> teamMistakes = teamsDao.teamMistakes(Double.parseDouble(diff),
teamId, league_id);
JSONArray games = new JSONArray();
JSONArray mistakes = new JSONArray();
for(Mistake teamMistake : teamMistakes){
games.put(teamMistake.getGames());
mistakes.put(teamMistake.getMistakes());
}
return new JSONArray().put(games).put(mistakes).toString();
});
public final Route getGamesForMonth = ((request, response) -> {
String month = request.queryParams("mounth");
int teamId = Integer.parseInt(request.queryParams("team_id"));
int leagueId = Integer.parseInt(request.queryParams("league_id"));
HashMap<Game, List<Odd>> gamesForMonth = teamsDao.getGamesForMonth(teamId,
leagueId, month);
HashMap<String, Object> model = new HashMap<>();
model.put("games", gamesForMonth);
```

```
String team_name = teamsDao.getTeamById(teamId).getName();
model.put("team_name", team_name);
return new ThymeleafTemplateEngine().render(new
ModelAndView(model,"gamesForMonth"));
});
private String generateDataJson(List<Mistake> teamMistakes){
JSONObject data = new JSONObject();
JSONArray labels = new JSONArray();
JSONArray games = new JSONArray();
JSONArray mistakes = new JSONArray();
for(Mistake teamMistake : teamMistakes){
labels.put(teamMistake.getName());
games.put(teamMistake.getGames());
mistakes.put(teamMistake.getMistakes());
}
data.put("labels",labels);
JSONArray datasets = new JSONArray();
JSONObject gamesDataset= new JSONObject();
gamesDataset.put("label","Games");
gamesDataset.put("data", games);
JSONObject mistakesDataset= new JSONObject();
mistakesDataset.put("label", "Mistakes");
mistakesDataset.put("data", mistakes);
gamesDataset.put("borderColor", "rgb(0, 196, 255)");
gamesDataset.put("fill", false);
gamesDataset.put("lineTension", 0);
mistakesDataset.put("borderColor", "rgb(185, 83, 83)");
mistakesDataset.put("fill", false);
mistakesDataset.put("lineTension", 0);
datasets.put(gamesDataset);
```

```
 datasets.put(mistakesDataset);
 data.put("datasets",datasets);
 return data.toString();
 }
}
package controllers;
import dao.LeagueDao;
import model.Mistake;
import spark.ModelAndView;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;
import org.json.JSONArray;
import org.json.JSONObject;
import java.util.HashMap;
import java.util.List;
public class GeneralDiagramController {
 private LeagueDao leagueDao = new LeagueDao();
 public final Route getGeneralDiagram = ((request, response) -> {
 List<Mistake> leagueMistakeList = leagueDao.getLeagueMistakes(1);
 JSONObject root = new JSONObject();
 root.put("type", "bar");
 JSONObject data = new JSONObject();
 JSONArray labels = new JSONArray();
 JSONArray games = new JSONArray();
 JSONArray mistakes = new JSONArray();
 for(Mistake leagueMistake : leagueMistakeList){
 labels.put(leagueMistake.getName());
 games.put(leagueMistake.getGames());
 mistakes.put(leagueMistake.getMistakes());
 }
```

```
        data.put("labels",labels);
        JSONArray datasets = new JSONArray();
        JSONObject gamesDataset= new JSONObject();
        gamesDataset.put("label","Games");
        gamesDataset.put("data", games);
        JSONObject mistakesDataset= new JSONObject();
        mistakesDataset.put("label", "Mistakes");
        mistakesDataset.put("data", mistakes);
        gamesDataset.put("backgroundColor", "rgb(0, 196, 255)");
        mistakesDataset.put("backgroundColor", "rgb(185, 83, 83)");
        datasets.put(gamesDataset);
        datasets.put(mistakesDataset);
        data.put("datasets",datasets);
        root.put("data", data);
        JSONObject options = new JSONObject();
        JSONObject scales = new JSONObject();
        scales.put("xAxes", new JSONArray().put(new JSONObject().put("stacked",
"true")));
        scales.put("yAxes", new JSONArray().put(new JSONObject().put("stacked",
"true")));
        options.put("scales",scales);
        root.put("options", options);
       51
        HashMap<String, Object> params = new HashMap<>();
        params.put("json", data.toString());
        return new ThymeleafTemplateEngine().render(new
ModelAndView(params,"leagues"));
        });
        public final Route getGeneralDiagramWithDiff = ((request, response) -> {
        String diff = request.queryParams("diff");
```

```
List<Mistake> leagueMistakeList =
leagueDao.getLeagueMistakes(Double.parseDouble(diff));
JSONObject root = new JSONObject();
root.put("type", "bar");
JSONObject data = new JSONObject();
JSONArray labels = new JSONArray();
JSONArray games = new JSONArray();
JSONArray mistakes = new JSONArray();
for(Mistake leagueMistake : leagueMistakeList){
labels.put(leagueMistake.getName());
games.put(leagueMistake.getGames());
mistakes.put(leagueMistake.getMistakes());
}
data.put("labels",labels);
JSONArray datasets = new JSONArray();
JSONObject gamesDataset= new JSONObject();
gamesDataset.put("label","Games");
gamesDataset.put("data", games);
JSONObject mistakesDataset= new JSONObject();
mistakesDataset.put("label", "Mistakes");
mistakesDataset.put("data", mistakes);
gamesDataset.put("backgroundColor", "rgb(0, 196, 255)");
mistakesDataset.put("backgroundColor", "rgb(185, 83, 83)");
datasets.put(gamesDataset);
datasets.put(mistakesDataset);
data.put("datasets",datasets);
root.put("data", data);
JSONObject options = new JSONObject();
JSONObject scales = new JSONObject();
```

```
scales.put("xAxes", new JSONArray().put(new JSONObject().put("stacked", "true")));
scales.put("yAxes", new JSONArray().put(new JSONObject().put("stacked", "true")));
options.put("scales",scales);
root.put("options", options);
return new JSONArray().put(games).put(mistakes).toString();
});
}
package controllers;
import com.google.gson.Gson;
import dao.LeagueDao;
import dao.TeamsDao;
import spark.ModelAndView;
import spark.Route;
import spark.template.thymeleaf.ThymeleafTemplateEngine;
import java.util.HashMap;
public class SelectorPageController {
private LeagueDao leagueDao = new LeagueDao();
private TeamsDao teamDao = new TeamsDao();
private Gson gson = new Gson();
public final Route getSelectorPage = ((request, response) -> {
HashMap<String, Object> params = new HashMap<>();
return new ThymeleafTemplateEngine().render(new ModelAndView(params,"selector"));
});
public final Route getLeagues = ((request, response) -> {
return gson.toJson(leagueDao.getLeagues());
});
public final Route getTeams = ((request, response) -> {
```

```
 int league = Integer.parseInt(request.queryParams("league"));
 return gson.toJson(teamDao.getResolvedTeams(league));
 });
}
package dao;
import model.League;
import model.Mistake;
import utils.ConnectionManager;
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
public class LeagueDao {
 public List<Mistake> getLeagueMistakes(double diff) {
 String sql = "SELECT odd_name, (f).* FROM (SELECT odd_name,
get_mistakes_for_league(league_id, ?) AS f FROM leagues) sub order by 2;";
 try (Connection connection = ConnectionManager.getSimpleConnection();
 PreparedStatement statement = connection.prepareStatement(sql)) {
 statement.setBigDecimal(1, BigDecimal.valueOf(diff));
 ResultSet resultSet = statement.executeQuery();
 return pareLeagueMistakes(resultSet);
 } catch (SQLException e) {
 e.printStackTrace();
 }
 return null;
 }
 public List<Mistake> getLeagueStatsByTeam(double diff, int league) {
```

```
String sql ="select gt.name, sum(get_mistake_for_game(g.game_id, g.score_one,
g.score_two, ?)) m, 0 from games g\n" +
"join \n" +
"(select team_one as team_id from games where league_id =?\n" +
"union \n" +
"select team_two from games where league_id=?) t\n" +
"on (g.team_one=t.team_id or g.team_two=t.team_id)\n" +
"join game_teams gt on gt.team_id=t.team_id\n" +
"where t.team_id in (select game_team_id from odd_game_teams)\n\n" +
"group by gt.name order by m;";
try (Connection connection = ConnectionManager.getSimpleConnection();
PreparedStatement statement = connection.prepareStatement(sql)) {
statement.setBigDecimal(1, BigDecimal.valueOf(diff));
statement.setInt(2, league);
statement.setInt(3, league);
ResultSet resultSet = statement.executeQuery();
return pareLeagueMistakes(resultSet);
} catch (SQLException e) {

e.printStackTrace();
}
return null;
}
public List<League> getLeagues() {
String sql = "select * from leagues;";
try (Connection connection = ConnectionManager.getSimpleConnection();
PreparedStatement statement = connection.prepareStatement(sql)) {
ResultSet resultSet = statement.executeQuery();
return parseLeagues(resultSet);
```

```
        } catch (SQLException e) {
        e.printStackTrace();
        }
        return null;
        }
        public League getLeagueById(int id) {
        String sql = "select * from leagues where league_id=?;";
        try (Connection connection = ConnectionManager.getSimpleConnection();
        PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setInt(1, id);
        ResultSet resultSet = statement.executeQuery();
        return parseLeagues(resultSet).get(0);
        } catch (SQLException e) {
        e.printStackTrace();
        }
        return null;
        }
        private List<League> parseLeagues(ResultSet resultSet) throws SQLException
{
        List<League> leagues = new ArrayList<>();
        while (resultSet.next()) {
        League league = new League();
        league.setLeague_id(resultSet.getInt(1));

league.setOdd_name(League.generateCustmLeagueName(resultSet.getString(2)));
        league.setGame_name(resultSet.getString(3));
        leagues.add(league);
        }
        return leagues;
        }
```

```
    private List<Mistake> pareLeagueMistakes(ResultSet resultSet) throws
SQLException {
    List<Mistake> mistakes = new ArrayList<>();
    while (resultSet.next()) {
    Mistake mistake = new Mistake();
    String name = resultSet.getString(1);
    mistake.setName(League.generateCustmLeagueName(name));
    mistake.setMistakes(resultSet.getInt(2));
    mistake.setGames(resultSet.getInt(3));
    mistakes.add(mistake);
    }
    return mistakes;
    }}
   package dao;
   import model.*;
   import utils.ConnectionManager;
   import java.math.BigDecimal;
   import java.sql.Connection;
   import java.sql.PreparedStatement;
   import java.sql.ResultSet;
   import java.sql.SQLException;
   import java.util.ArrayList;
   import java.util.HashMap;
   import java.util.List;
   public class TeamsDao {
    public List<TotalTeam> getTotalTeams() {
    String sql = "select o.name, g.name, odd_team_id, game_team_id from
odd_game_teams
   ogt \n" +
    "join odd_teams o on ogt.odd_team_id = o.team_id\n" +
```

```
"join game_teams g on ogt.game_team_id = g.team_id;";
try (Connection connection = ConnectionManager.getSimpleConnection();
PreparedStatement statement = connection.prepareStatement(sql)) {
ResultSet resultSet = statement.executeQuery();
return parseTotalTeams(resultSet);
} catch (SQLException e) {
e.printStackTrace();
}
return null;
}
public void resolveTeams(int gameId, int oddId) {
String sql = "insert into odd_game_teams values(?,?)";
try (Connection connection = ConnectionManager.getSimpleConnection();
PreparedStatement statement = connection.prepareStatement(sql)) {
statement.setInt(1, oddId);
statement.setInt(2, gameId);
statement.executeUpdate();
} catch (SQLException e) {
e.printStackTrace();
}
}
public void deleteResolvedTeam(int gameId, int oddId) {
String sql = "delete from odd_game_teams where odd_team_id =? and game_team_id =
?";
try (Connection connection = ConnectionManager.getSimpleConnection();
PreparedStatement statement = connection.prepareStatement(sql)) {
statement.setInt(1, oddId);

statement.setInt(2, gameId);
```

```
statement.executeUpdate();
} catch (SQLException e) {
e.printStackTrace();
}
}
public List<Team> getGameTeams() {
String sql = "SELECT team_id, name, (select count(*) from odd_game_teams where
game_team_id=team_id) FROM game_teams order by name;";
return getSimpleTeams(sql);
}
public Team getTeamById(int id){
String sql = "select team_id, name, 2 from game_teams where team_id = ?";
try (Connection connection = ConnectionManager.getSimpleConnection();
PreparedStatement statement = connection.prepareStatement(sql)) {
statement.setInt(1, id);
ResultSet resultSet = statement.executeQuery();
return parseTeams(resultSet).get(0);
} catch (SQLException e) {
e.printStackTrace();
}
return null;
}
public List<Mistake> teamMistakes(double diff, int team, int league){
String sql = "select TO_CHAR(g.date, 'Month') m,
sum(get_mistake_for_game(g.game_id, g.score_one, g.score_two, ?)),
count(g.game_id),
EXTRACT(month from g.date) mt, EXTRACT(year from g.date) y from games
g where (team_one =
```

```
? or team_two = ?) and exists (select odd_id from
get_odd_for_game(g.game_id)) and
league_id = ? group by m, mt, y order by y, mt;";
try (Connection connection = ConnectionManager.getSimpleConnection();
PreparedStatement statement = connection.prepareStatement(sql)) {
statement.setBigDecimal(1, BigDecimal.valueOf(diff));
statement.setInt(2, team);
statement.setInt(3, team);
statement.setInt(4, league);
ResultSet resultSet = statement.executeQuery();
return pareMistakes(resultSet);
} catch (SQLException e) {
e.printStackTrace();
}
return null;
}
public List<Team> getOddTeams() {
String sql = "SELECT team_id, name, (select count(*) from odd_game_teams
where
odd_team_id=team_id) FROM odd_teams order by name;";
return getSimpleTeams(sql);
}
public List<Team> getResolvedTeams(int league) {
57
String sql = "select distinct gt.team_id, gt.name, 2 from games g join
game_teams
gt on (g.team_one=gt.team_id or g.team_two=gt.team_id) where g.league_id=?
and (select
odd_id from get_odd_for_game(g.game_id)) is not null order by gt.name;";
try (Connection connection = ConnectionManager.getSimpleConnection();
```

```
PreparedStatement statement = connection.prepareStatement(sql)) {
statement.setInt(1, league);
ResultSet resultSet = statement.executeQuery();
return parseTeams(resultSet);
} catch (SQLException e) {
e.printStackTrace();
}
return null;
}
public HashMap<Game, List<Odd>> getGamesForMonth(int teamId, int leagueId, String
month){
String sql = "select g.game_id, g.date, g.score_one, g.score_two, gt1.name,
gt2.name, ov.team_one, ov.team_two, ov.draft, bc.name, g.is_switched\n" +
"from (select game_id, date, score_one, score_two, team_one, team_two,
league_id, (f).* from (select gm.*, get_odd_for_game(gm.game_id) f from games gm) a) g
\n" +
"join game_teams gt1 on gt1.team_id = g.team_one\n" +
"join game_teams gt2 on gt2.team_id = g.team_two\n" +
"join odd_values ov on ov.odd_id = g.odd_id\n" +
"join bett_companies bc on bc.company_id = ov.company_id\n" +
"where (g.team_one = ? or g.team_two=?) and trim(TO_CHAR(g.date,
'Month'))=? and g.league_id=?;";
try (Connection connection = ConnectionManager.getSimpleConnection();
PreparedStatement statement = connection.prepareStatement(sql)) {
statement.setInt(1, teamId);
statement.setInt(2, teamId);
statement.setString(3, month);
statement.setInt(4, leagueId);
```

```
ResultSet resultSet = statement.executeQuery();
return parseOdds(resultSet);
} catch (SQLException e) {
e.printStackTrace();
}
return null;
}
private List<Team> getSimpleTeams(String sql) {
try (Connection connection = ConnectionManager.getSimpleConnection();
PreparedStatement statement = connection.prepareStatement(sql)) {
ResultSet resultSet = statement.executeQuery();
return parseTeams(resultSet);
} catch (SQLException e) {
e.printStackTrace();
}
return null;
}
private List<Team> parseTeams(ResultSet resultSet) throws SQLException {
List<Team> teams = new ArrayList<>();
while (resultSet.next()) {
Team team = new Team();
team.setTeamId(resultSet.getInt(1));
team.setName(resultSet.getString(2));
team.setResolved(resultSet.getInt(3)>=1);
teams.add(team);
}
return teams;
}
private List<TotalTeam> parseTotalTeams(ResultSet resultSet) throws
SQLException {
```

```
List<TotalTeam> teams = new ArrayList<TotalTeam>();
while (resultSet.next()) {
TotalTeam totalTeam = new TotalTeam();
totalTeam.setOddName(resultSet.getString(1));
totalTeam.setGameName(resultSet.getString(2));
totalTeam.setOddTeamId(resultSet.getInt(3));
totalTeam.setGameTeamId(resultSet.getInt(4));
teams.add(totalTeam);
}
return teams;
}
private List<Mistake> pareMistakes(ResultSet resultSet) throws SQLException
{
List<Mistake> mistakes = new ArrayList<>();
while (resultSet.next()) {
Mistake mistake = new Mistake();
mistake.setName(resultSet.getString(1));
mistake.setMistakes(resultSet.getInt(2));
mistake.setGames(resultSet.getInt(3));
mistakes.add(mistake);
}
return mistakes;
}
private HashMap<Game, List<Odd>> parseOdds(ResultSet resultSet) throws
SQLException {
HashMap<Game, List<Odd>> result = new HashMap<>();
while (resultSet.next()) {
Game game = new Game();
game.setGame_id(resultSet.getInt(1));
game.setDate(resultSet.getDate(2));
```

```
game.setScore_one(resultSet.getInt(3));
game.setScore_two(resultSet.getInt(4));
game.setTeam_one(resultSet.getString(5));
game.setTeam_two(resultSet.getString(6));
Odd odd = new Odd();
boolean isSwitched = resultSet.getBoolean(11);
if(!isSwitched) {
odd.setTeam_one(resultSet.getDouble(7));
odd.setTeam_two(resultSet.getDouble(8));
}
else {
odd.setTeam_one(resultSet.getDouble(8));
odd.setTeam_two(resultSet.getDouble(7));

}
odd.setDraft(resultSet.getDouble(9));
odd.setBett_company(resultSet.getString(10));
if(result.containsKey(game)) result.get(game).add(odd);
else {
List<Odd> odds = new ArrayList<>();
odds.add(odd);
result.put(game, odds);

}

}
return result;

}
}
```