Ожго Ю. А., здобувач,
Дацюк Д. В., асистент
Державний університет «Житомирська політехніка»

ROUNDROBINTOURNAMENT АЛГОРИТМ ДЛЯ ГЕНЕРАЦІЇ РОЗКЛАДУ СПОРТИВНОЇ ЛІГИ

Розробка програми для генерації розкладу спортивних подій є складним завданням, яке передбачає багато аспектів та вимагає уважної підготовки. Визначення формату та структури події є ключовим етапом у цьому процесі. Алгоритм для генерації розкладу спортивних подій відіграє значну роль у забезпеченні оптимального розкладу проведення турніру чи змагань. Потрібно зважати на різноманітні чинники, адже розробка цього алгоритму вимагає уважного врахування різноманітних факторів, таких як кількість учасників, розподіл команд, часові обмеження та особливості конкретного виду спорту.

Темою проекту є адміністрування любительських футбольних ліг і власне однією із задач проекту є генерація розкладу для ліги. Враховуючи що за спортивний сезон одна команда повинна зіграти з кожною командою дивізіону, було вирішено використати RoundRobinTournament алгоритм для генерації розкладу.

RoundRobinTournament (Круговий турнір) – це алгоритм, за яким кожен учасник зустрічається з усіма іншими учасниками по черзі. В одному круговому розкладі учасник грає з іншим учасником лише один раз [1]. Ця система вважається найбільш справедливою, але водночас вимагає найбільшого числа ігор для розподілу місць, порівняно з іншими турнірними системами.

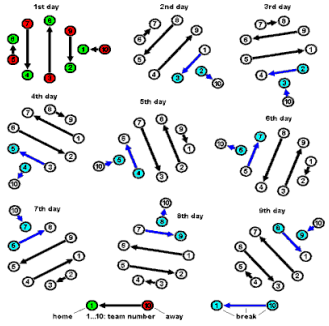

Рис. 1. Візуалізація алгоритму на прикладі 10 учасників

Якщо спростити опис алгоритму, то для його реалізації нам потрібно розрахувати кількість раундів, після цього потрібно опрацювати кожен з раундів і згенерувати матчі між командами. Кількість раундів напряму залежить від кількості команд; якщо кількість команд парна, то кількість раундів розраховується за формулою - $(n-1)$, де n це кількість команд. Якщо кількість команд непарна, то кількість раундів відповідатиме кількості команд. Кількість матчів, що відбудуться за раунд, можна розрахуватиза такими формулами:

- $(\frac{n}{2})(n-1)$, якщо n парне
- $\left(\frac{n-1}{2}\right)n$, якщо nнепарне.

Перейдемо власне до реалізації алгоритму. Так як генерація розкладу це тільки частина проекту і сам проект являє собою набір різних компонентів(мобільний додаток, API, web-портал для адміністрування), то реалізація алгоритму була виконана на стороні API. Для написання APIчастини було використано Node.jstехнологію, для спрощення процесу розробки був використаний Nest.jsфреймворк.

Nest.js – це фреймворк для створення ефективних, масштабованих серверних програм Node.js [2]. Перевагою фреймворку є те, що Nest.js повністю підтримує Typescripti базується на DependencyInjection (DI) підході. Головна ідея DIполягає в тому, щоб відокремити процес створення та управління залежностями від основної логіки програми.

API компонент побудовано за підходом SOA. SOA(Service-OrientedArchitecture) – це підхід до розробки програмного забезпечення, при якому функціональність програми поділяється на незалежні служби, що можуть взаємодіяти між собою [3]. Основні принципи SOA:

- незалежність служб;
- стандартизовані інтерфейси;
- відкритий доступ до служб.

Слідуючи підходу SOA, для генерації розкладу було створено окремий модуль Schedule. Schedule модуль об`єднує в собі такі компоненти:

- schedule.module – основний файл модулю, який об`єднує в собі всі потрібні компоненти і створює необхідні залежності;
- schedule.controller – відповідає за маршрутизацію, і власне в цьому компоненті створено маршрути, за якими мобільний додаток зможе отримати розклад;
- schedule.service – відповідає за бізнес логіку, власне в цьому компоненті реалізовано генерацію розкладу.

В schedule.service створено метод, який реалізує Round Robin Tournament. В загальному реалізація цього алгоритму виглядає як подвійний цикл, перший цикл відповідає за кількість раундів, вкладений цикл відповідає за генерацію пар команд та зсув команд у списку для наступного раунду. Якщо сильно спростити, то загальна структура коду виглядатиме приблизно так:

```
    function generateRoundRobin(teams) {
const numTeams = teams.length;
const rounds = numTeams % 2 === 0 ? numTeams - 1 : numTeams;
const factor = numTeams % 2 === 0 ? numTeams / 2 : (numTeams - 1) / 2;
const schedule = [];
for (let round = 0; round <rounds; round++) {
const roundMatches = [];
for (let i = 0; i <factor; i++) {
const match = [teams[i], teams[numTeams - 1 - i]];
roundMatches.push(match);
      }
schedule.push(roundMatches);
// Зсувкомандуспискудлянаступногораунду
teams.splice(1, 0, teams.pop());
   }
return schedule;
}
```

Отже, після аналізу алгоритмів для генерації розкладу, основним алгоритмом було вибрано Round Robin Tournament, так як цей алгоритм задовольняє всі необхідні критерії для футбольних спортивних ліг. Було розроблено APIчастину проекту яка, в тому числі, і реалізує алгоритм генерації. Використання додатку дозволяє значно спростити адміністрування спортивної ліги, особливо коли це стосується формування розкладу на сезон.

Список використаних джерел

1. Optimizing Game Scheduling With Round-Robin Algorithms. URL: https://cactusware.com/blog/round-robin-scheduling-algorithms#:~:text=It’s%20a%20scheduling%20algorithm%20where,all%20the%20matches%20are%20completed
2. Nest.js. URL: https://docs.nestjs.com/
3. What is SOA (Service-Oriented Architecture)? URL: https://aws.amazon.com/what-is/service-oriented-architecture/