МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

ПРОЕКТУВАННЯ ТА РОЗРОБКА МІКРОСЕРВІСУ ДЛЯ СИМУЛЯЦІЇ ТА АНАЛІЗУ МАТЧІВ

Кваліфікаційна робота
на здобуття ступеня вищої освіти «бакалавр»

Виконав: студент 4 курсу 431 групи
Спеціальності: 122. Комп'ютерні науки
Освітньо-професійної програми: програмування
Прасько Артем Віталійович
Керівник: кандидат педагогічних наук Вінник Максим Олександрович
Рецензент: доц. Кафедри алгебри, геометрії та мат. аналізу
Таточенко Володимир Іванович

Херсон – Івано-Франківськ – 2023

ЗМІСТ

ВСТУП:

Актуальність теми: Сучасні технології дозволяють створювати веб-симуляції на основі заданих факторів, будь то симуляції поведіники природних явищ, або технічних. Однак існуючі рішення не задовольняють потреби бізнесів, які хочуть симулювати кіберспортивні матчі, та на основі цього робити спостереження та висновки.

***Об'єкт дослідження*:** мікросервісна веб-платформа Match Simulation.

***Предмет дослідження*:** технології створення мікросервісів.

***Мета і завдання дослідження*:** метою дослідження є розробка програмного забезпечення, яке дозволить користувачам блокчейн системи зручно та ефективно розраховувати потенційний дохід від ставок. Завдання дослідження:

1) Аналіз бекенд архітектур, та пошук оптимальних рішень при їх побудові
2) Виявлення потреб та можливостей для побудови проектування та розробки програмного забезпечення
3) Розробка технічної архітектури та програмного забезпечення.
4) Дослідження функцій та можливостей мікросервісу Match Simulation, створеного з використанням PHP Laravel і MySQL і пропонуючого API інтерфейс.

РОЗДІЛ 1
ОСНОВНІ ПОЛОЖЕННЯ ПРО СЕРВІС, МІКРОСЕРВІСИ ТА ВИКОРИСТАНІ ТЕХНОЛОГІЇ

1.1 Основні положення про створений сервіс

Описані функції та можливості веб-платформи Match Simulation розроблені для бізнесів, які хочуть симулювати кіберспортивні матчі та ділитися своїми результатами з партнерами. Клієнти можуть створювати, керувати та відстежувати свої моделювання, що дозволяє їм отримувати значну перевагу над конкурентами.

Мікросервіс Match Simulation є ідеальним рішенням для бізнесів, що працюють у галузі кіберспорту та хочуть збільшити ефективність своєї діяльності. У наш час, коли кіберспорт є однією з найбільш швидко зростаючих галузей розваг, цей сервіс дозволить клієнтам привернути до себе більше уваги та досягти успіху на ринку.

1.2 Огляд мікросервісної архітектури та її переваги

Мікросервісна архітектура передбачає поділ програмного забезпечення на невеликі сервіси, кожен з яких відповідає за виконання конкретної функції або завдання. Кожен сервіс може функціонувати незалежно від інших сервісів, бути масштабованим та підтримуватися окремо.

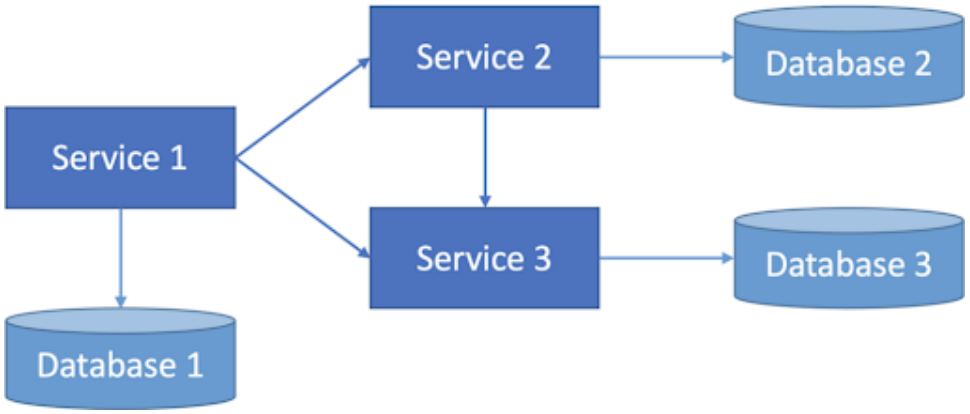

Рисунок 1.1 - Вид мікросервісної архітектури

Мікросервіси можуть бути написані на різних мовах програмування та використовувати різні технології, що дає можливість використовувати ті технології, які найкраще підходять для виконання потреб проекту.

При проектуванні архітектури проекту використовують підхід до делегування на сервіси. Цей підхід полягає у тому, щоб весь бізнес-логіку розділити на окремі сервіси, кожен з яких відповідає за свої функції. При цьому, кожен сервіс може бути реалізований в окремому мікросервісі, що дозволяє забезпечити більшу гнучкість та масштабованість системи.

Моделі використовуються для представлення даних у системі. Кожен сервіс має свою власну модель даних, яка відповідає за конкретний аспект бізнес-логіки, що реалізується сервісом. Наприклад, у випадку електронної комерції, можуть бути моделі для замовлень, товарів, користувачів тощо.

Роути визначають маршрутизацію запитів у системі. Кожен сервіс має свої власні роути, які відповідають за приймання запитів та передачу їх до відповідного сервісу.

Репозиторії використовуються для забезпечення доступу до даних. Кожен сервіс має свій власний репозиторій, який забезпечує доступ до моделей даних, що використовуються сервісом.

Рівень до якого звертається кожен сервіс залежить від рівня складності бізнес-логіки, що реалізується сервісом. Наприклад, простіші сервіси можуть звертатися до бази даних напряму, тоді як складніші сервіси можуть звертатися до інших сервісів або навіть до сторонніх API.

Таблиця 1

Порівняльна характеристика монолітної та мікросервісної архітектур

Особливості	**Моноліт**	**Мікросервіси**
Архітектура	Всі компоненти додатку розташовані в одному великому контейнері	Кожен сервіс має свій власний контейнер, що дозволяє їх масштабувати та підтримувати окремо
Масштабованість	Складно масштабувати окремі компоненти, оскільки вони залежать один від одного	Легко масштабувати окремі сервіси, що дає можливість масштабувати додаток в цілому
Розгортання	Весь додаток розгортається на одному сервері	Кожен сервіс може бути розгорнутий на окремому сервері
Розвиток	Складно розвивати окремі компоненти, оскільки вони залежать один від одного	Легко розвивати окремі сервіси, оскільки вони не залежать один від одного
Помилки	Помилки в одному компоненті можуть впливати на весь додаток	Помилки в одному сервісі не впливають на інші сервіси

1.3 Мова програмування PHP та її застосування

PHP є однією з найпопулярніших мов програмування у світі веб-розробки. За допомогою PHP можна створювати динамічні веб-сайти та веб-додатки, які можуть взаємодіяти з базами даних та іншими сервісами. Основні переваги PHP полягають у широкій підтримці веб-серверами, великій кількості готових бібліотек та фреймворків, а також у зручному синтаксисі.

Недоліками PHP можуть бути деякі обмеження в якості коду, відсутність строгого типізації, а також складність у підтримці старих версій мови. Однак, з огляду на популярність та доступність PHP, було вирішено використовувати цю мову програмування для реалізації Match Simulation.

Окрім цього, PHP має вбудовану підтримку для взаємодії з веб-сервером, що забезпечує зручний та швидкий доступ до веб-додатків. Також PHP забезпечує широкі можливості для роботи з СУБД та роботи з веб-сервісами.

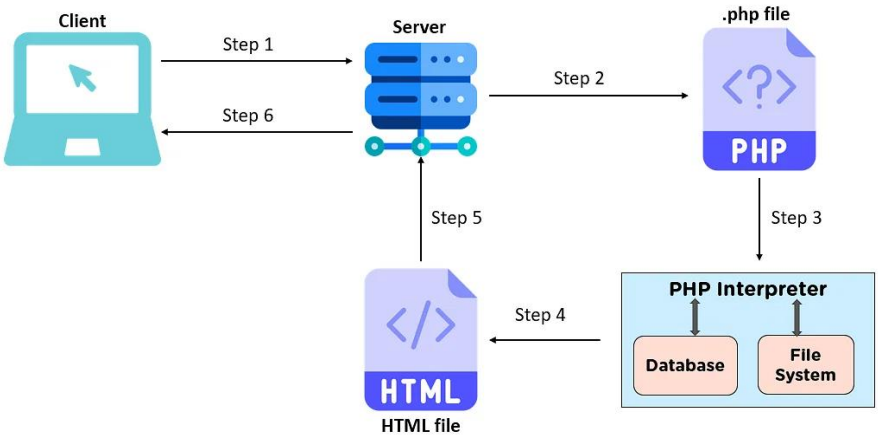

Рисунок 1.2 - Стандартна обробка PHP файлів інтерпретатором

1.4 Фреймворк Laravel

Laravel є одним з найпопулярніших фреймворків для веб-розробки на сьогоднішній день. Використання Laravel для розробки Match Simulation було обумовлено його багатофункціональністю та гнучкістю. За допомогою Laravel можна ефективно використовувати мікросервісну архітектуру, а також миттєво реагувати на зміни, що є важливою перевагою для нашого проекту.

Laravel надає широкий спектр можливостей для розробки веб-додатків, включаючи зручний вбудований механізм маршрутизації, використання ORM (Object-Relational Mapping) для взаємодії з базою даних, роботу з шаблонами та велику кількість розширень. Крім того, Laravel має велику спільноту розробників, що дозволяє отримувати швидку підтримку та відповіді на запитання.

Використання Laravel для розробки Match Simulation дозволило зменшити час та зусилля, потрібні для розробки веб-додатку. Крім того, він надав можливість розгортати додаток швидко та з легкістю, що дозволило зосередитись на основній функціональності додатку.

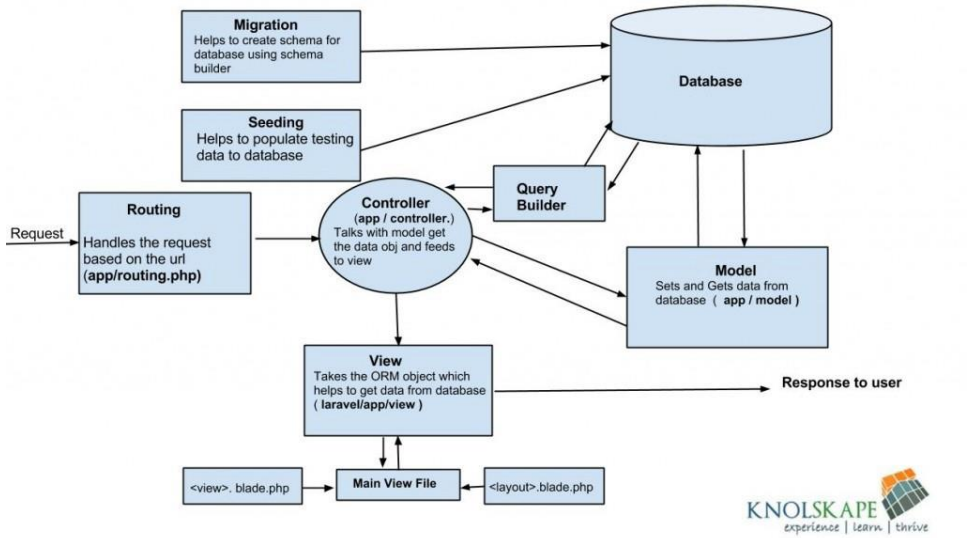

Рисунок 1.3 - Архітектура Laravel

Таблиця 2

Переваги Laravel над іншими backend фрейморками

Фреймворк	**Переваги Laravel**
Symfony	Менший розмір коду, більш проста конфігурація, широкі можливості розширення, вбудований роутинг та шаблонізатор
Ruby on Rails	Швидка розробка завдяки вбудованим інструментам та конвенціям, лаконічний синтаксис, хороша документація
Django	Багато вбудованих функцій, таких як адміністративний інтерфейс, захист від CSRF, ORM та багато іншого, зручна структура проекту
Express.js	Дуже швидкий та легкий, багато різноманітних модулів та пакетів для побудови різних додатків

1.5 MySQL

MySQL є однією з найпопулярніших відкритих СУБД, яка забезпечує зберігання та організацію даних. Її використання в Match Simulation було обумовлено кількома факторами, такими як:

Відкритість та безкоштовність: MySQL є відкритим програмним забезпеченням, що дає змогу використовувати його безкоштовно, що є важливим фактором для стартапу з обмеженим бюджетом.

Надійність та швидкість: MySQL має високу надійність та швидкість роботи, що дозволяє забезпечувати швидкий доступ до даних та їх зберігання.

Сумісність з Laravel: Laravel має вбудовану підтримку для MySQL, що дозволяє легко та ефективно взаємодіяти з базою даних.

Таблиця 3

Порівняльна таблиця Mysql та інших БД

Функціональність / База даних	**MySQL**	**PostgreSQL**	**Oracle**	**MS SQL Server**
Вільна та відкрита для використання	Так	Так	Ні	Ні
Підтримка зовнішніх ключів	Так	Так	Так	Так
Підтримка транзакцій	Так	Так	Так	Так
Повнотекстовий пошук	Так	Так	Так	Так
Реплікація даних	Так	Так	Так	Так
Підтримка розподіленої БД	Так	Так	Так	Так
Підтримка ACID	Так	Так	Так	Так

Підтримка процедур та тригерів	Так	Так	Так	Так
Підтримка резервного копіювання	Так	Так	Так	Так
Підтримка підзапитів	Так	Так	Так	Так

1.6 GuzzleHTTP

GuzzleHttp - це бібліотека, що дозволяє здійснювати HTTP запити в PHP за допомогою простого та зрозумілого API. Вона є однією з найпопулярніших бібліотек для роботи з API-інтерфейсами у світі PHP.

GuzzleHttp надає користувачам зручний інтерфейс для взаємодії з різними API. Вона підтримує HTTP/1.1 і HTTP/2, автоматичну обробку Cookies, має можливість підключення до проксі-серверів і т.д.

Для реалізації Match Simulation було вирішено використовувати GuzzleHttp для роботи з API-інтерфейсом. Бібліотека була вибрана з-за своєї простоти та зручності використання, а також через те, що вона дозволяє легко та швидко здійснювати запити до зовнішніх сервісів.

До переваг GuzzleHttp можна віднести:

- Простий інтерфейс для створення рядків запитів, POST-запитів, потокового великого завантаження, потокового великого завантаження, використання HTTP-кукі, завантаження JSON-даних і т.д...
- Може надсилати як синхронні, так і асинхронні запити, використовуючи один і той же інтерфейс.

- Використовує інтерфейси PSR-7 для запитів, відповідей і потоків. Це дозволяє використовувати інші PSR-7-сумісні бібліотеки разом з Guzzle.
- Абстрагується від базового HTTP-транспорту, дозволяючи вам писати код, незалежний від середовища і транспорту, тобто без жорсткої залежності від cURL, PHP-потоків, сокетів або неблокуючих циклів обробки подій.

```php
public function create_request(string $endpoint_path, string $request_type, array $body = null): JsonResponse
{
    $headers = ['x-api-key' => env('X_API_KEY')];
    $simulation_url = env('SIMULATION_URL') . "/simulation/$endpoint_path";
    $client = new Guzzle();
    try {
        $response = $client->request($request_type, $simulation_url, [
            'headers' => $headers,
            'json' => $body,
        ]);
        $response = json_decode($response->getBody(), true);
        return $this->jsonResponse($response, Response::HTTP_OK);
    } catch (\GuzzleHttp\Exception\RequestException $ex) {
        if ($ex->getResponse()) {
            $response = $ex->getResponse()->getBody()->getContents();
            return $this->jsonResponse(json_decode($response, true), Response::HTTP_BAD_REQUEST);
        }
        return $this->jsonResponse($ex->getMessage(), Response::HTTP_INTERNAL_SERVER_ERROR);
    }
}
```

Рисунок 1.4 - Приклад запиту через Guzzle

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

2.1 Архітектурне проектування

2.1.1 Міграції

Для проектування бази даних використовується ORM Eloquent.

За допомогою цієї орм давайте створемо файли міграції, які після запуску процесу через CLI створять потрібні таблиці та поля з відношеннями та з ключами за потреби.

Міграції — це як контроль версій вашої бази даних, що дозволяє вашій команді визначати та надавати спільний доступ до визначення схеми бази даних програми.

Фасад Laravel Schema забезпечує підтримку бази даних для створення та маніпулювання таблицями в усіх підтримуваних Laravel системах баз даних. Як правило, міграції використовуватимуть цей фасад для створення та зміни таблиць і стовпців бази даних.

Для їх створення через CLI (інтерфейс командної строки) використаємо команду:

```
php artisan make:migration create_table_simulation_webhook
```

Тепер ми маємо схему міграцій потрібну нам для комунікації з БД. Запустимо команду `php artisan migrate`

←T→	id	name	identifier	created_at	updated_at	key	url	secret
	1	213213	1	NULL	NULL	111	http://webhook.site/1f30d1d6-d479-40dd-9e1a-cb35fc...	"super_secret_secret"
	2	ad	3	NULL	NULL	12	http://webhook.site/3da62e3c-820e-49b5-a558-39b4c7...	"second_secret"
	3	partner100		NULL	NULL	1212	http://localhost:8000/simulation/bets/webhook	"third"
	4	"fourth"		NULL	NULL	key111	http://localhost:8000/simulation/bets/webhook	"secret_number_4"

Рис 2.2 - таблиця в БД після міграції та заповнення

2.1.2 Проектування архітектури симуляції матчів

Симуляція гри не має під собою жодного випадкового чинника, та повністю залежить від рівня команди в рейтингу, карти в грі, сили гравців, тощо.

Побудуємо архітектуру:

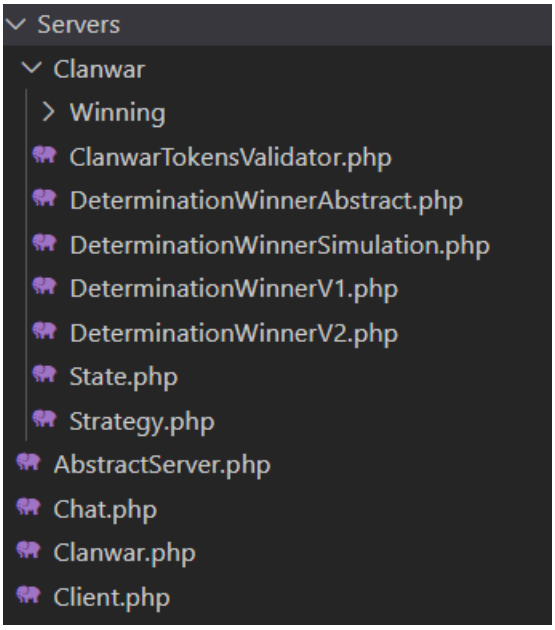

Рис 2.3 - Архітектура симуляції матчів

Як видно на рисунку 2.3, в папці Servers->Clanwar є файли, в кожному зберігається бізнес логіка за принципом SPR. Фактори визначення переможця зберігаються в Servers->Clanwar->Winning (Рис 2.4)

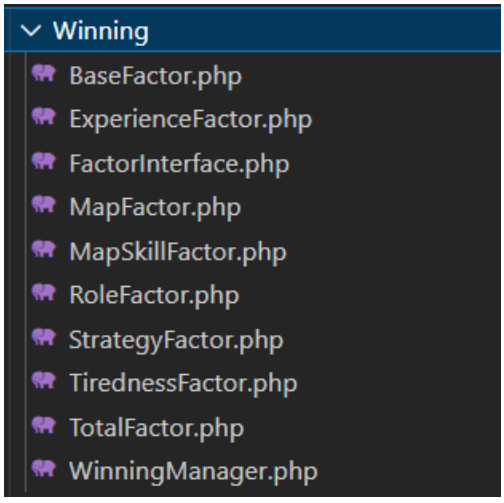

Рис 2.4 - Архітектура факторів на перемогу команди

2.2 Реалізація системи

2.2.1 Створення проекту

Для створення нового проекту на Laravel можна скористатися менеджером пакетів Composer. Це найбільш популярний інструмент для управління залежностями та створення проектів на PHP.

Перш за все, потрібно встановити Composer, якщо він ще не встановлений на комп'ютері. Після цього можна створити новий проект командою:

```
composer create-project --prefer-dist
laravel/laravel match-simulation
```

Ця команда створить новий проект на Laravel з назвою " match-simulation". Після того, як проект буде створений, можна налаштувати з'єднання з базою даних у файлі .env:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=database-name

DB_USERNAME=database-username
DB_PASSWORD=database-password
```

В основному проекті використовується база даних MySQL, тому для нового проекту було використано ту саму базу даних, щоб уникнути додаткового ускладнення архітектури.

2.2.2 Мідлвар ендпоінти

В новому проекті була створена група мідлвар ендпоінтів:

- GET /simulation/lines - повертає усі доступні матчі (lines)
- GET /simulation/lines/{id} - повертає інформацію про конкретний матч (line)
- GET /simulation/lines/{id}/predictions - повертає усі прогнози на певний матч (predictions)
- POST /simulation/lines/{id}/predictions - створює новий прогноз на матч (prediction)
- GET /simulation/predictions/{id} - повертає інформацію про конкретний прогноз (prediction)
- GET /simulation/predictions - повертає усі доступні прогнози на матчі (predictions)
- POST /simulation/predictions/{id}/cancel - відміняє прогноз на матч (prediction)

Ця група ендпоінтів використовує мідлвар з назвою "simulation", який відповідає за перевірку, чи дозволено користувачу здійснювати запити на ендпоінти. Префікс "/simulation" вказує, що всі ендпоінти в цій групі належать до "симуляційного" режиму роботи.

2.2.3 Мідлвар та його роль у зв’язку мікросервісів

Мідлвар - це проміжний шар між запитом та відповіддю на нього. Він дозволяє модифікувати запити та відповіді, перевіряти права доступу та виконувати різноманітні операції з ними.

У нашому випадку мідлвар потрібен для того, щоб перенаправити запити з основного сервісу на мікросервіс, де знаходяться необхідні ендпоінти для проведення симуляцій та отримання прогнозів. Мідлвар також додає до запиту заголовок з ключем API для автентифікації запиту на мікросервісі. Код мідлвару доданий в додатку 2.

2.2.3 - Парсер

Основна логіка парсера полягає в тому, що він завантажує вміст сторінки рейтингу команд з сайту hltv.org і проходить по рядках цього вмісту, витягуючи потрібну інформацію. Це потрібно на етапі 2.2.3, для синхронизації команд з сайту, та в БД.

Зокрема, парсер знаходить позицію команди в рейтингу, кількість набраних командою очок, список гравців команди та країну, з якої ця команда походить.

Ця інформація зберігається в змінній $teams у вигляді асоціативного масиву, де ключами є позиції команд у рейтингу, а значеннями є асоціативні масиви з назвою команди, кількістю очок, списком гравців та країною походження.

Код містить декілька методів для витягування різних даних з рядків сторінки, таких як назва команди, кількість очок, список гравців та країна походження. Для витягування деяких даних використовуються регулярні вирази.

Також в коді є використання функції sleep(), яка забезпечує затримку на 20 секунд перед завантаженням сторінки команди з її додатковою інформацією з сайту hltv.org. Це зроблено з метою

запобігання блокування IP-адреси, якщо запити виконуються занадто швидко. Код парсеру доданий в додатку 2.

2.2.4 - Крон синхронизації

Створена команда синхронизує дані про рейтинг кіберспортивних команд з базою даних основного проекту, та містить декілька етапів:

- Ініціалізація крон-задачі через crontab.
- Виконання парсера.
- Оновлення бази даних.

Загалом, ця команда дозволить автоматизувати процес збору та оновлення даних про команди та їх рейтинги, що є важливим етапом для забезпечення актуальності інформації для подальшої симуляції.

2.2.5 - Логіка симуляція матчів

Основна логіка використовує моделі Laravel, такі як Team та Match для отримання даних про команди та збереження матчів в базі даних.

Крім того, вона використовує фабрику режимів ModeFactory, яка надає різні режими гри для вибору з них під час генерації матчів.

Кожен матч генерується з випадково обраним режимом гри, двома випадково обраними командами, коєфіцієнтами команд, які визначаються на основі їх рейтингів та вибраного режиму гри, кількістю раундів і запланованим часом початку матчу.

2.2.5 - Процес симуляції

В методі симуляції визначається, на якій зараз карті знаходяться команди та які рахунки вони мають. Якщо матч знаходиться на другій половині карти, то запускається метод calculateSecondPart(), який розраховує рахунок другої половини карти.

Потім визначаються очки за симуляційні раунди, які отримані в результаті симуляції гри. Для кожної команди визначається загальна кількість очок. Якщо одна з команд забрала більше 16 очок, то її очки обмежуються до 16.

Далі випадковим чином визначається, чи будуть обміняні очки між командами. Якщо отриманий випадковий коефіцієнт менше від коефіцієнту з симуляції, то точки між командами будуть обмінюватися. Так утворюється коєфіцієнт випадковості.

Потім зберігаються результати кожного раунду, визначається переможець на карті та зберігається статистика гравців у базі даних.

Логіка процесу гри буде прикріплена у додатку 2.

2.2.6 - Сокети

Розглянемо детальніше, як працюють сокети в контексті симуляцій та процесу гри.

Сокети - це механізм, який дозволяє різним процесам та комп'ютерам обмінюватися даними через мережу. У контексті симуляцій, сокети можуть бути використані для створення мережевого з'єднання між сервером та клієнтами. Сервер може виконувати симуляцію, а клієнти у реальному часі дивитися на процеси матчу.

Процес гри, зазвичай, складається з декількох етапів. Першим етапом є ініціалізація гри, коли встановлюються параметри гри, ініціалізуються об'єкти та інші необхідні дії. Після цього, гра переходить до етапу головного циклу. У головному циклі гра отримує введення від гравців, оновлює свій стан та відображує новий стан гри на екрані. Після цього, цикл повторюється до закінчення гри.

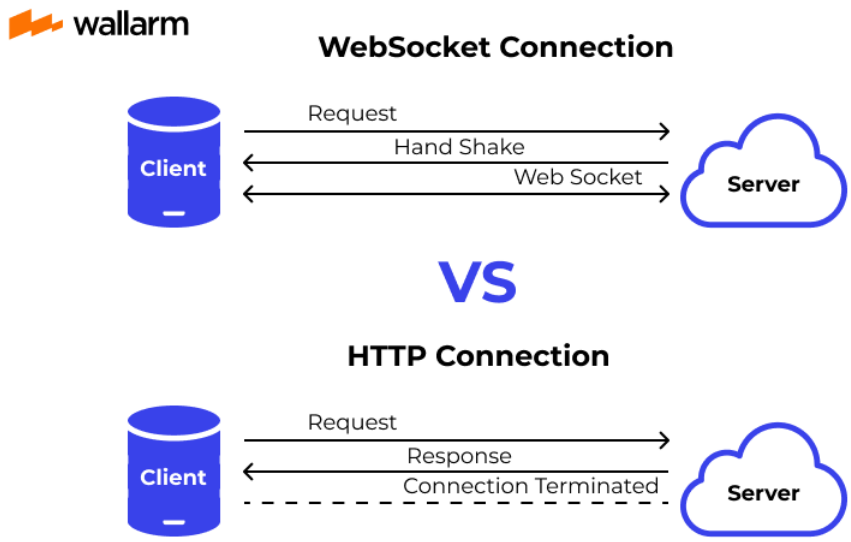

Рис 2.4 - Приклад серверного з’єднання через сокети

2.2.7 - Безпека та API ключі для клієнтів

Ключ API - це унікальний ідентифікатор, який дозволяє партнеру отримати доступ до мікросервісу. Ключ API може бути випадковим рядком, який генерується мікросервісом і передається партнеру або ж може бути визначений партнером самостійно і переданий мікросервісу для автентифікації.

Для того, щоб забезпечити безпеку, ключ API повинен бути зашифрований та збережений в безпечному місці. У даному випадку, ключ API зберігається в файлі конфігурації та хешується в файлі .env.

Крім того, ключ API зберігається в базі даних, щоб забезпечити перевірку на кожен запит. Під час кожного запиту мікросервіс перевіряє ключ API у запиті з ключем у базі даних, щоб переконатися, що партнер, який надсилає запит, авторизований.

Таблиця 4

Таблиця, яка зберігає данні про ключі партнерів

Поле	**Тип**	**Опис**
id	integer	Унікальний ідентифікатор партнера
name	string	Назва партнера
identifier	string	Ідентифікатор партнера
key	string	Ключ API для взаємодії з партнером
created_at	timestamp	Дата та час створення запису у базі даних
updated_at	timestamp	Дата та час останньої модифікації запису

Отже, автентифікація ключа API - це важливий аспект забезпечення безпеки системи, який дозволяє забезпечити доступ до мікросервісу лише авторизованим партнерам.

2.2.8 - Механізм вебхуків

Вебхук - це спеціальний механізм, що дозволяє підписаним на нього користувачам отримувати автоматичні сповіщення про певні події, що відбуваються в системі. У даному випадку, після закінчення матчу, система відправляє вебхук з результатами матчу усім партнерам, які підписані на розсилку вебхуків.

У розробленій системі кіберспортивних матчів було вирішено використовувати вебхуки для сповіщення партнерів про результати матчів. Для забезпечення безпеки передачі даних, було введено шифрування вебхуків за допомогою алгоритму SHA-256.

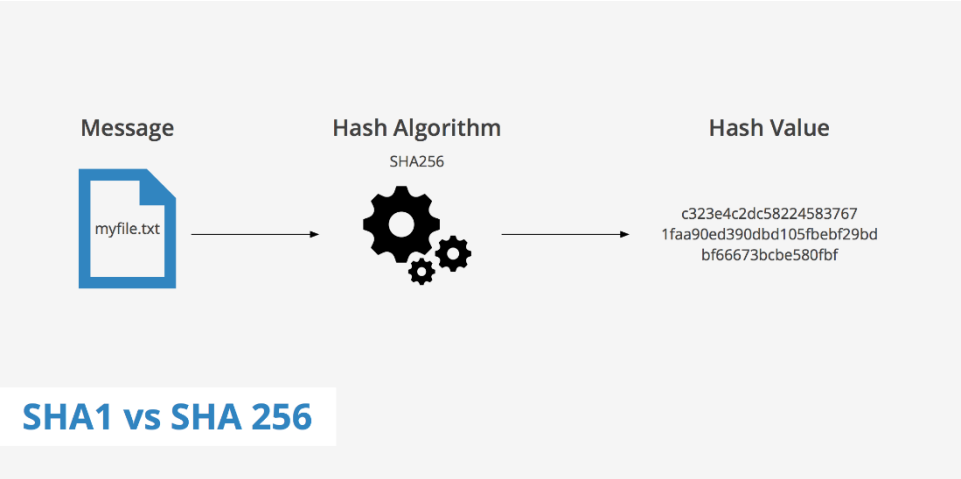

Рис.2.5 - Приклад SHA-256 шифрування

Для кожного партнера було згенеровано унікальний secret key, який використовувався для створення підпису в заголовку запиту x-api-sign. Цей підпис містив значення хеш-функції від об'єднання мітки часу (timestamp) та тіла вебхуку у форматі JSON. Підпис було відправлено

разом з вебхуком в заголовку запиту x-api-sign для перевірки автентичності та цілісності вебхуку на боці отримувача.

Це дозволило зменшити ризик можливих атак на передачу даних та забезпечити високу ступінь безпеки та надійності системи кіберспортивних матчів. Код логіки вебхуків прикріплено в додатку 2.

Таблиця 4

Таблиця вебхуків та їх статусів

Поле	**Тип даних**	**Опис**
id	integer	Унікальний ідентифікатор вебхуку
route	string	URL-адреса, за якою доступний вебхук
signature	string	Шифрований підпис,
timestamp	integer	Час відправлення вебхуку Unix-часу
status	string	Статус вебхуку

2.2.9 - Прийом та обробка вебхуків

Для клієнтів була розроблена та інтегрована система прийому та обробки вебхуків. Ця система отримує вебхуки від кіберспортивної платформи і обробляє їх у відповідності до встановлених правил. Після

отримання вебхуку, система розшифровує його за допомогою алгоритму SHA-256 та перевіряє його підпис, щоб забезпечити правильність відправника та цілісність даних.

Після перевірки підпису, система зберігає дані вебхуку в базі даних та обробляє їх згідно з правилами, встановленими для кожного клієнта. Обробка може включати оновлення даних в системі клієнта, сповіщення користувачів та інші дії. Така система дозволяє клієнтам миттєво отримувати дані про матчі та інші події на кіберспортивній платформі та дозволяє автоматизувати багато процесів, що значно покращує ефективність та точність роботи.

2.2.10 - Результати матчів

Таблиця 5

Таблиця респонсу партнеру

Поле	Опис
id	Унікальний ідентифікатор прогнозу
line_id	ID матчу, на який був зроблений прогноз
partner_id	ID партнера, що зробив прогноз
user_id	ID користувача, що зробив прогноз
team_id	ID команди, на яку був зроблений прогноз
win	Переможець прогнозу (так/ні)
status	Статус прогнозу

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено бекенд-систему для обробки кіберспортивних матчів. Для зберігання даних була використана реляційна база даних MySQL. Модулі були розділені на мікросервіси, які можна запускати окремо і взаємодіяти між собою за допомогою REST API.

Результатом роботи є працююча система з обробки кіберспортивних матчів, яка може обробляти велику кількість запитів і забезпечує високу надійність і масштабованість. Використання мікросервісної архітектури дозволяє швидко впроваджувати нові функції і зміни, а також підтримувати і розширювати систему без впливу на роботу інших модулів.

Обрані технології були доцільними через їхню відомість та популярність в галузі розробки, а також завдяки широкій підтримці та наявності документації. Використання мікросервісної архітектури дозволило зменшити ризик залежності між модулями та забезпечити більш гнучке керування ресурсами сервера.

Були виконані наступні завдання:

Розроблено функціонал для:

- Обробки даних матчів;
- Обробки вебхуків;
- Безпеки та шифрування;
- Роботи з партнерами;

Використано технології:

- Мікросервісів
- PHP
- Laravel
- MySQL
- Git/Gitlab

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація PHP. [Електронний ресурс] - URL: https://www.php.net/docs.php
2. Створення проекту з нуля. [Електронний ресурс] URL: https://laravel.com/docs/10.x#your-first-laravel-project
3. Laravel routing. [Електронний ресурс] URL: https://laravel.com/docs/10.x/routing
4. Laravel models. [Електронний ресурс] URL: https://laravel.com/docs/10.x/eloquent#generating-model-classes
5. MySQL документація. [Електронний ресурс] URL: https://dev.mysql.com/doc/
6. Laravel Query raw building. [Електронний ресурс] URL: https://blog.quickadminpanel.com/5-ways-to-use-raw-database-queries-in-laravel/
7. Як будувати мікросервіси. [Електронний ресурс] URL: https://www.atlassian.com/microservices/microservices-architecture/building-microservices
8. Побудуй перший мікросервіс. [Електронний ресурс] URL: https://medium.com/swlh/building-your-first-microservice-80c90af74d9b
9. Посібник по налаштуванню crontab. [Електронний ресурс] URL: https://crontab.guru/
10. Пояснення роботи SHA-256. [Електронний ресурс] URL: https://sectigostore.com/blog/sha-256-algorithm-explained-by-a-cyber-security-consultant/
11. Порівняння мікросервісів та монолітів. [Електронний ресурс] URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith
12. Побудува highload PHP додатків. [Електронний ресурс] URL: https://evnedev.com/blog/development/10-core-points-of-the-architecture-of-a-high-load-php-project/

13. Клієнт серверна архітектура. [Електронний ресурс] URL: https://medium.com/@IvanZmerzlyi/%D0%BA%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0-%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0-%D1%82%D0%B0-%D1%80%D0%BE%D0%BB%D1%96-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D1%96%D0%B2-9893d8048229
14. Робота з АПІ ключами. [Електронний ресурс] URL:https://swagger.io/docs/specification/authentication/api-keys/
15. Що таке вебхуки. [Електронний ресурс] URL: https://zapier.com/blog/what-are-webhooks/
16. Імплементація шифрування в вебхуки. [Електронний ресурс] URL: https://hookdeck.com/webhooks/guides/how-to-implement-sha256-webhook-signature-verification
17. Посібник по роботі з АПІ. [Електронний ресурс] URL: https://technologyadvice.com/blog/information-technology/how-to-use-an-api/
18. Робота з асинхронністю в laravel. [Електронний ресурс] URL: https://laravelactions.com/2.x/dispatch-jobs.html
19. Як приймати вебхуки в laravel. [Електронний ресурс] URL: https://medium.com/@onadeji.sam/how-to-recieve-webhook-in-laravel-c6fc40dfce88
20. Посібник по розробкі парсеру. [Електронний ресурс] URL: https://laravel.com/api/8.x/Illuminate/Console/Parser.html

ДОДАТКИ

Додаток а

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Прасько Артем Віталійович, учасник освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

· вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;

· принципів та правил академічної доброчесності;

· нульової толерантності до академічного плагіату;

· моральних норм та правил етичної поведінки;

· толерантного ставлення до інших;

· дотримуватися високого рівня культури спілкування;

– надавати згоду на:

· безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;

· оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;

· використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов’язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв’язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі у будь-якій діяльності, пов’язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов’язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров’ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й проти мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

12.09.2019 р.		Прасько Артем
(дата)	(підпис)	(ім’я, прізвище)

Додаток б

Клас перевірки Апі ключів

```
namespace App\Simulation;

use GuzzleHttp\Client;
use GuzzleHttp\Exception\RequestException;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Response;

class SimulationApi
{
    /**
     * @var string
     */
    private $baseUrl;

    /**
     * @var string
     */
    private $apiKey;

    /**
     * @var Client
     */
    private $httpClient;

    /**
     * SimulationApi constructor.
     *
```

```
     * @param string $baseUrl
     * @param string $apiKey
     * @param Client $httpClient
     */
    public function __construct(string $baseUrl, string $apiKey, Client $httpClient)
    {
        $this->baseUrl = $baseUrl;
        $this->apiKey = $apiKey;
        $this->httpClient = $httpClient;
    }

    /**
     * @param string $endpointPath
     * @param string $requestType
     * @param array|null $body
     *
     * @return JsonResponse
     */
    public function createRequest(string $endpointPath, string $requestType, array $body = null): JsonResponse
    {
        $headers = ['x-api-key' => $this->apiKey];
        $url = $this->baseUrl . "/simulation/$endpointPath";

        try {
            $response = $this->httpClient->request($requestType, $url, [
                'headers' => $headers,
                'json' => $body,
            ]);
            $data = json_decode($response->getBody(), true);
```

```
            return $this->jsonResponse($data,
Response::HTTP_OK);
        } catch (RequestException $ex) {
            if ($ex->getResponse()) {
                $response = $ex->getResponse()-
>getBody()->getContents();
                return $this-
>jsonResponse(json_decode($response, true),
Response::HTTP_BAD_REQUEST);
            }
            return $this->jsonResponse($ex->getMessage(),
Response::HTTP_INTERNAL_SERVER_ERROR);
        }
    }

    /**
     * @param array $data
     * @param int $status
     *
     * @return JsonResponse
     */
    private function jsonResponse(array $data, int
$status): JsonResponse
    {
        return new JsonResponse($data, $status);
    }
}
```

Клас парсеру

```
namespace App\Services\HLTV;

use Illuminate\Notifications\Notifiable;
```

```
use App\Models\Country;


class Parser
{
    use Notifiable;


    public $teams = [];
    private $position;


    public function
routeNotificationForSlack($notification)
    {
        return env('SLACK_NOTIFICATIONS');
    }


    public function __construct()
    {
        $file =
@file_get_contents("https://www.hltv.org/ranking/teams");
        if ($file) {
            $this->processFile($file);
        }
    }


    private function processFile($content)
    {
        $lines = explode(chr(10), $content);
        foreach ($lines as $line) {
            $position = $this->getTeamPosition($line);
            if ($position) {
                $this->position = $position;
                continue;
            }
```

```
            $points = $this->getTeamPoints($line);
            if ($points) {
                $this->teams[$this->position]['points'] =
trim($points);
                continue;
            }

            $players = $this->getTeamPlayers($line);
            if ($players) {
                $this->teams[$this-
>position]['players'][] = $players;
                continue;
            }

            $country = $this->getTeamCountry($line);
            if ($country) {
                $this->teams[$this->position]['country']
= $country;
            }
        }
    }

    private function getTeamPosition($line)
    {
        if (substr_count($line, "<div class=\"ranking-
header\"><span class=\"position\">#")) {
            $position = explode("<", explode("<div
class=\"ranking-header\"><span class=\"position\">#",
$line)[1])[0];
            $name = explode("\" src", explode("alt=\"",
$line)[1])[0];
            $this->teams[$position]['team'] = $name;
            return $position;
        }
        return null;
    }
```

```
    private function getTeamPoints($line)
    {
        if (substr_count($line, "<div class=\"teamLine
sectionTeamPlayers")) {
            $points = explode("points)",
explode("class=\"points\">(", $line)[1])[0];
            return $points;
        }
        return null;
    }

    private function getTeamPlayers($line)
    {
        if (substr_count($line, "<td class=\"player-
holder\">")) {
            $player = explode("\" src", explode("<img
alt=\"", $line)[1])[0];
            return $player;
        }
        return null;
    }

    private function getTeamCountry($line)
    {
        if (substr_count($line, "<div class=\"more")) {
            $moreLink = explode("more)",
explode("class=\"moreLink\">(", $line)[0])[0];
            preg_match('#<a href="(.+?)"#', $moreLink,
$teamUrl);
            $teamUrl = $teamUrl[1];
            sleep(20);
            $teamPage =
file_get_contents("https://www.hltv
private function Country($val)
{
```

```
    if (substr_count($val, "<div class=\"more")) {
        $moreLink = explode("more)",
explode("class=\"moreLink\">(", $val)[0])[0];
        preg_match('#<a href="(.+?)"#', $moreLink,
$team_url);
        $team_url = $team_url[1];
        sleep(20);
        $team_page =
file_get_contents("https://www.hltv.org" . $team_url);
        preg_match('#<div class="team-country text-
ellipsis">(.+?)</div>#', $team_page, $response);
        preg_match('#title="(.+?)">#', $response[0],
$country);
        $country = $country[1];
        $country_id = Country::where('name', $country)-
>first();
        $this->teams[$this->position]['country'] =
$country_id ? trim($country_id->id) : 247;
        return true;
    }
    return false;
}
```

Клас кронтаб для створення матчів

```
<?php

namespace App\Console\Commands\Simulation;

use Illuminate\Console\Command;
use App\Models\Team;
use App\Models\Match;
use App\Models\Modes\ModeFactory;

class SimulateMatches extends Command
```

```
{
    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'simulate:matches';


    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Makes 12 random simulated
matches from top30 one time at hour';


    /**
     * Create a new command instance.
     *
     * @return void
     */
    public function __construct()
    {
        parent::__construct();
    }


    /**
     * Execute the console command.
     *
     * @return mixed
     */
    public function handle()
```

```
    {


        $modes = [
            1 => [
                ModeFactory::ALL_RANDOM,
                ModeFactory::BAN_BAN_BAN_RANDOM_1,
                ModeFactory::BAN_BAN_RANDOM_1,
                ModeFactory::BAN_RANDOM_1
            ],
            3 => [
                ModeFactory::ALL_RANDOM,
                ModeFactory::BAN_PICK_RANDOM_3,
                ModeFactory::PICK_RANDOM_3,
                ModeFactory::BAN_PICK_BAN_RANDOM_3,
                ModeFactory::BAN_BAN_RANDOM_3,
                ModeFactory::BAN_BAN_PICK_RANDOM_3
            ],
            5 => [
                ModeFactory::ALL_RANDOM,
                ModeFactory::PICK_BAN_PICK_RANDOM_5,
                ModeFactory::BAN_PICK_PICK_RANDOM_5,
                ModeFactory::PICK_PICK_BAN_RANDOM_5,
                ModeFactory::PICK_PICK_RANDOM_5
            ]
        ];


        $teams = Team::Join('rating', 'team.id', '=',
'rating.team_id')
            ->select('team_id', 'number')-
>where('rating.game_id', '=', 0)
            ->orderBy('rating.number')->limit(30)->get();
        $team_id_list = array();
```

```
        foreach ($teams as $team) {
            array_push($team_id_list, [$team->team_id,
$team->number]);
        }


        $timeIterator = 0;
        function getCoef($one, $two)
        {
            $diff = abs($one - $two);


            $firstStep = 50 + $diff * 1.3;
            $secondStep = 59.1 + ($diff - 7) * 1.45;
            $thirdStep = 76.5 + ($diff - 19) * 1.6;


            if ($diff < 8) {
                return $firstStep;
            }
            if ($diff < 20) {
                return $secondStep;
            } else {
                return $thirdStep;
            }
        }
        for ($i = 1; $i <= 12; $i++) {


            $rounds =  array_rand($modes);
            $mode = array_rand($modes[$rounds]);
            $random_team = array_rand($team_id_list, 2);
```

```
            $first_team = $team_id_list[$random_team[0]];
            $first_id = $first_team[0];
            $first_number = $first_team[1];


            $second_team =
$team_id_list[$random_team[1]];
            $second_id = $second_team[0];
            $second_number = $second_team[1];


            $teams_coef = getCoef($first_number,
$second_number);
            $coef1 = 1 / (($teams_coef + 4) / 100);
            $coef2 = 1 / ((100 -$teams_coef + 4) / 100);


            Match::create([


                'game_id' => 1,
                'team_id' => $first_id,
                'enemy_team_id' => $second_id,
                'team_coef' => $coef1,
                'enemy_team_coef' => $coef2,
                'mode' => $mode,
                'status' => 1,
                'tournament_id' => 1,
                'rounds_count' => $rounds,
                'scheduled_to' => date("Y-m-d H:i:s",
strtotime("+" . $timeIterator .  "minutes")),


            ]);
            $timeIterator += 5;
        }
```

```
    }
}
```

Клас відправки вебхуків

```
<?php


namespace App\Jobs;


use App\Models\Match;
use App\Models\Simulation\Bet;
use Illuminate\Bus\Queueable;
use Illuminate\Queue\SerializesModels;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use App\Models\User;
use App\Models\Deposit;
use App\Jobs\SendWebhook;
class CalculateSimulations implements
ShouldQueue
{
    use Dispatchable, InteractsWithQueue,
Queueable, SerializesModels;


    protected $line_id;
    /**
     * Create a new job instance.
```

```
     *
     * @return void
     */
    public function __construct($line_id)
    {
        $this->line_id = $line_id;
    }


    /**
     * Execute the job.
     *
     * @return void
     */
    public function handle()
    {
        $winner_team = Match::where('id', $this-
>line_id)->first();


        Prediction::where('line_id', $this-
>line_id)
            ->where('status',
Prediction::STATUS_PREDICTION_CREATE)
            ->update(['status' =>
Prediction::STATUS_ PREDICTION _LOSS]);


        Prediction::where('line_id', $this-
>line_id)
```

```
            ->where('status',
Prediction::STATUS_ PREDICTION_LOSS)
            ->where('team_id', $winner_team-
>winner)
            ->update(['status' =>
Prediction::STATUS_ PREDICTION_WIN]);


        $predictions=
Prediction::where('line_id', $this->line_id)
            ->whereBetween('status', [

Prediction::STATUS_PREDICTION_WIN,

Prediction::STATUS_PREDICTION_LOSS
            ])
            ->get();


        foreach ($predictions as $ prediction) {
            $status = $ predictions->status ==
Prediction::STATUS_PREDICTION_WIN  ? $
predictions ->win_status: 0;
            User::where('id', $prediction-
>user_id)->increment('balance',
$received_amount);


            $user = User::select('balance')-
>where('id', $bet->user_id)->first();


            if ($received_amount) {
```

```
                Deposit::create([
                    'user_id' => $prediction-
>user_id,
                    'amount' =>
$received_amount,
                    'balance' => $user->balance,
                    'comment' => 'win
prediction’
                ]);
            }


        }
        dispatch(new SendWebhook($this-
>line_id));
    }
}
```