Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Інтернет-платформа для ведення статистики настільних ігор
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗППІПЗ.190155.19.07.ПЗ

Виконав студент IV курсу група ІПЗ-17-1 _____ І. В. Нетреба
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ Н. І. Праворська
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Г. І. Бедратюк
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

1 червня 2022 р.

Хмельницький 2022

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Бедратюк Г. І. ст. викладач		
Антиплагіат	Гурман І. В. доцент кафедри ІПЗ		

7. Дата видачі завдання « 05 » лютого 2021 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних тем ДП	01.12– 30.12.2021	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2022	
3 Проектування програмного забезпечення	01.02 – 28.02 2022	
4 Програмна реалізація	01.03 – 10.04.2022	
5 Тестування програмного забезпечення	11.04 – 30.04.2022	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2022	
7 Попередній захист ДП	травень 2022 (згідно графіка)	
8 Перевірка ДП на плагіат, нормоконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2022	
9 Підготовка до захисту та захист ДП	з 01.06.2022	

Студент _____ І. В. Нетреба
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Н. І. Праворська
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту: Інтернет-платформа для ведення статистики настільних ігор.

Автор проекту: Нетреба Ігор Вікторович.

Керівник проекту: Праворська Наталія Іванівна.

Пояснювальна записка: 61с., 22 рис., 5 табл., 3дод., 8 джерел.

Графічна частина: 16 слайдів.

ІНТЕРНЕТ ПЛАТФОРМА, СТАТИСТИКА, НАСТІЛЬНІ ІГРИ, ВЕБ-ДОДАТОК, PHP.

Метою проекту є розробка інтернет-платформи для ведення статистики настільних ігор.

В ході розробки дипломного проекту було розглянуто існуючі рішення, складена їх характеристика, проведено дослідження предметної області з виділенням нефункціональних та функціональних вимог до розроблювальної платформи. Було обрано архітектуру веб-додатку, описано структуру та моделі бази даних, спроектовано серверну частину та інтерфейс користувача, проведено аналіз та вибір технологій і методів реалізації системи. Здійснено програмну реалізацію, розробка: бази даних, серверної частини, керівництва користувача і проведено тестування платформи.

1.06.2022
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4		Пояснювальна записка			
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні	16		
			слайди			

Змн.	Арк.	№ докум.	Підпис	Дата	ЗППІПЗ.190155.19.07.ПЗ		
Виконав		Нетреба І.В.			Інтернет-платформа для ведення статистики настільних ігор	Літ.	Арк. 1 / Аркушів 1
Керівник		Праворська Н.І.			Відомість документів		
Н. контр.		Бедратюк Г. І.				ХНУ, ІПЗ-17-1	
Зав. каф.		Бедратюк Л.П.					

ЗМІСТ

ВСТУП

Гра – є важливий і, безумовно, корисний елемент у житті кожної дитини та у житті кожної дорослої. Просто з віком людина стає трохи раціональнішою, і щоб переконатися в корисності, їй потрібно побачити конкретні факти, наведемо їх.

У дитинстві діти завжди зайняті різноманітними іграми. Вони щось вигадують для себе, та своїх друзів, з головою занурюються у явні світи і щиро насолоджуються цим процесом. В цей час дитина ставиться до гри як до серйозного заняття і дивується чому дорослі не хочуть до неї приєднатися. З плином часом це змінюються. З'являється загальна уставка що гра непотрібна, легковажна, від чого можна відмовитись. Ці думки переносяться і на настільні ігри.

По-перше настільні ігри покращують роботу мозку. Це несподівано, але це дійсно так. Під час гри гравець дізнається багато нового - починаючи від правил та умов, закінчуючи цікавими фактами, які обговорюються з іншими гравцями. Нова інформація – це корисне живлення для розуму. Крім того, настільні ігри скорочують можливість виникнення хвороби Альцгеймера і допомагають уникнути раннього недоумства.

По-друге, гра стимулює зони мозку, які відповідальні за думки та спогади людини, а значить, завдяки іграм тренеруються навички у вирішенні проблем та прийнятті складних рішень. Іншими словами, настільні ігри роблять людський мозок сильнішим - йому простіше впоратися з будь-якими викликами. Чи то складнощі на роботі, чи повсякденні труднощі – ігрові підходи допоможуть з ними впоратися.

По-третє ігри веселять та розслабляють. Сміх – це приємний побічний ефект настільних ігор та один із факторів, чому людям взагалі подобається грати. Коли людина добре проводить час, скорочуємо виникнення стресу, що є актуальним в теперішній політичній обстановці та через пандемію . А як показало

дослідження RealNetwork, 53% людей грають у настільні ігри, щоб позбутися стресу.

Пояснити це досить просто: людина зосереджується на грі, а робочі моменти та рутинні проблеми залишаються десь осторонь. А коли після жвавої гри особа повертається у звичні турботи, вони вже не виглядають такими втомливими та непідйомними.

І останнє і найголовніше – ігри зближують людей та зміцнюють відносини. Вся суть настільних ігор – взаємодія з оточуючими. Не має значення, чи грає людина зі старими друзями, родичами, колегами, чи розігрує партію з людьми, яких бачe вперше в житті, — настільні ігри допоможуть весело провести час і краще пізнати інших людей.

Деякі ігри поставлять гравця серед учасників, інші можуть розкрити підприємницькі нахили або приміряти цікавий образ. Людина переживає нетиповий, іноді екстремальний досвід і це відбувається разом з іншими користувачами. Особа знає про небезпеки і сором'язливості, і розкриваємось один перед одним.

Актуальність теми полягає в тому що гравцеві потрібна особиста статистика інформація для аналізу своїх партій в тій чи іншій ігри, в першу чергу це стосується тих гравців які приймають участь в турнірах і моніторити свої досягнення (наприклад, за допомоги деяких сайтів чи додатків) на основі яких можна робити висновки для поліпшення своїх особистих показників, що в свою чергу приведе до поліпшення загальної статистики. Проблему можна вирішити шляхом створення веб-ресурсу чи мобільного додатка, яка дає можливість реалізації проекта.

Метою проекту є створення інтернет-платформи для ведення статистики настільних ігор шляхом створення веб-ресурсу.

Завдання які потрібно реалізувати:

– дослідити предметну область, з особливостями її реалізації і функціоналу

– дослідити фреймворк Laravel який буде слугувати базою для створення додатку;

– зробити аналіз існуючих рішень на ринку, визначити їх переваги та недоліки і мінімальний функціонал;

– розробити систему, яка буде задовольняти вимоги користувача;

– провести тестування системи.

Результатом дипломного проекту є функціонуюча платформа, яка надає можливість користувачу вести особисту статистику настільних ігор.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Статистичні дані – інформація, отримана на підставі проведених статистичних спостережень, що опрацьована і подана у формалізованому вигляді відповідно до загальноприйнятих принципів та методології. Статистичні дані, що є результатом зведення та угруповання первинних даних, за умови забезпечення їх знеособленості являють собою зведену знеособлену статистичну інформацію (дані).

Виникає питання, як саме будуть зберігатись дані для подальшого використання. У випадку з мобільними девайсами дані можна зберігати децентралізовано, тобто користувач має можливість використовувати та оновлювати інформацію в локальній репліці, і при підключені до мережі Інтернет оновити їх у центральному сховищі даних. У випадку з веб-додатком усі дані будуть зберігатись у базі даних, тому що ці бази дуже добре і детально описані, виникає необхідність обрати будь-яку мову програмування і працювати через загальний і зручний інтерфейс обробки даних. Таким чином, можна стандартизовано обробляти дані не хвилюючись, що вони будуть оброблені якось по іншому.

Перше, що слід враховувати при проектуванні бази даних це те, що ігри класифікуються по жанровій приналежності на декілька категорій, але для нас важливі тільки чотири. Перша категорія, це ігри європейської школи – ігри в яких не стикаються інтереси гравців безпосередньо, гравці не можуть бути виключені з гри, а роль удачі хоч і є, але досить мала в таких іграх переможцем стає той, хто набрав найбільшу кількість переможних балів. Друга категорія, це ігри американської школи – на відміну від європейський ігор, в таких іграх конфлікт інтересів підкреслюється, як правило, в іграх велика роль випадковості, в таких

іграх переможцем стає або той, хто першим виконав ціль, або набрав найбільшу кількість очок. Третя категорія, це кооперативні ігри, в них гравці об’єднано грають проти гри і в ній перемагають або програють одразу всі. Четверта і остання категорія це командні ігри, в яких йде розподіл на команди з певною кількістю гравців в кожній і перемогу отримає вся команда не залежно від результату кожного гравця в окремому випадку.

Головна проблема статистичних додатків для настільних ігор – те, що вони ідеально підходять для мобільних додатків адже кожен завжди зможе зберегти в локальному сховищі дані про партію, незалежно від під’єднання до Інтернету і ергономічність, користувач повинен отримувати потрібну йому інформацію швидко і ефективно без лишніх рухів.

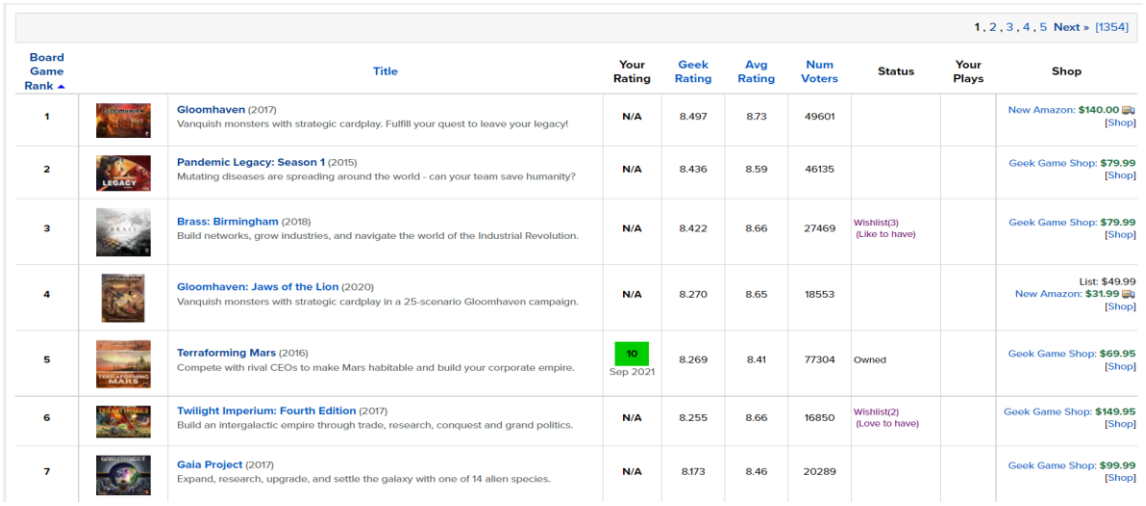
Щоб уникнути цих недоліків, потрібно використати платформу, яка дозволить створити максимально зручний користувацький інтерфейс і в разі чого оновлювати його, щоб звершити кількість лишніх дій потрібно налаштувати додаток так, щоб головна інформація завжди була доступна з будь-якого місця сайту і максимально скоротити кількість рутинної роботи надаючи при цьому максимальну кількість різноманітної інформації і не тільки.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Розглянемо існуючі варіанти платформ для ведення статистики, які найбільш поширені серед користувачів, щоб визначити усі недоліки та переваги того чи іншого ресурсу, які слід враховувати під час розробки та визначені функціональний вимог якими повинна володіти платформа.

BoardGameGeek – он-лайн форум, який існує з 2000 року для любителів настільних ігор і база даних ігор, яка містить різні огляди, відео та зображення для понад 139 000 різних настільних ігор, сюди входять як ігри європейського стиля, так і американського, та карткові ігри.

Перший мінус, який нас зустрічає одразу після реєстрації і переходу до списку ігор– застарілий дизайн, який зображено на рисунку 1.1.



1 , 2 , 3 , 4 , 5 Next » [1354]

Board Game Rank	Title	Your Rating	Geek Rating	Avg Rating	Num Voters	Status	Your Plays	Shop
1	**Gloomhaven** (2017) Vanquish monsters with strategic cardplay. Fulfill your quest to leave your legacy!	N/A	8.497	8.73	49601			New Amazon: $140.00 [Shop]
2	**Pandemic Legacy: Season 1** (2015) Mutating diseases are spreading around the world - can your team save humanity?	N/A	8.436	8.59	46135			Geek Game Shop: $79.99 [Shop]
3	**Brass: Birmingham** (2018) Build networks, grow industries, and navigate the world of the Industrial Revolution.	N/A	8.422	8.66	27469	Wishlist(3) (Like to have)		Geek Game Shop: $79.99 [Shop]
4	**Gloomhaven: Jaws of the Lion** (2020) Vanquish monsters with strategic cardplay in a 25-scenario Gloomhaven campaign.	N/A	8.270	8.65	18553			List: $49.99 New Amazon: $31.99 [Shop]
5	**Terraforming Mars** (2016) Compete with rival CEOs to make Mars habitable and build your corporate empire.	10 Sep 2021	8.269	8.41	77304	Owned		Geek Game Shop: $69.95 [Shop]
6	**Twilight Imperium: Fourth Edition** (2017) Build an intergalactic empire through trade, research, conquest and grand politics.	N/A	8.255	8.66	16850	Wishlist(2) (Love to have)		Geek Game Shop: $149.95 [Shop]
7	**Gaia Project** (2017) Expand, research, upgrade, and settle the galaxy with one of 14 alien species.	N/A	8.173	8.46	20289			Geek Game Shop: $99.99 [Shop]

Рисунок 1.1 – Сторінка з топ 100 іграми

Перш ніж переглянути статистику потрібно додати гру до колекції. Для цього користувач може знайти її через форму пошуку і в списку схожих запитів, або ж безпосередньо на сторінці гри шляхом натиску на кнопку «Add to Collection». В процесі додання є можливість обрати необхідні статуси для доданої гри і при необхідності ввести коментар, який буде відображено для інших користувачів. Також наявна функція налаштування інформації про гру і розширеної приватної інформації (вартість купівлі, якість та інші).

Після чого, можна додати партію шляхом обирання необхідної гри і внесення усіх необхідних параметрів як зображено на рисунку 1.2

Але єдина статистика яка наявна користувачу це перегляд партії в ту чи іншу гру за місяць і перегляд оцінений ігор які наявні та які не наявні в колекції їх оцінка на форумі через виставлення оцінки за шкалою від 1 до 10, їх рейтингу на сайті, статусу користувача кількості ігор і коментарієм яка зображено на рисунку 1.3.

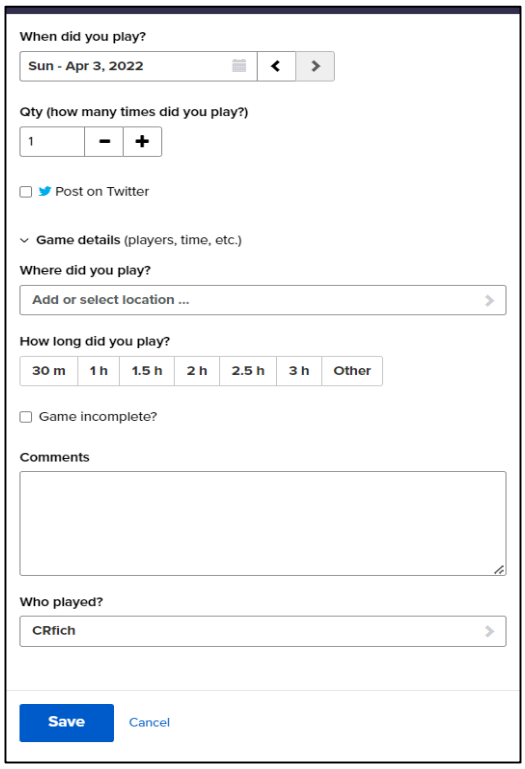


Рисунок 1.2 – Форма додання партії

Наразі, це єдина інтернет платформа, в якій є можливість вести хоч якусь статистику, але існують мобільні додатки для статистики настільних ігор, які ми також розглянемо для більш детального дослідження наявних аналогів.

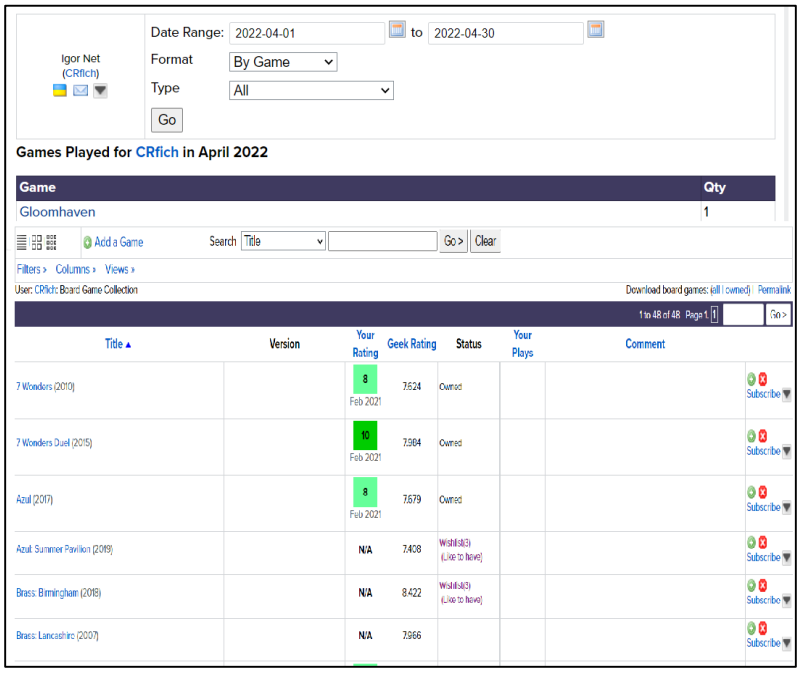


Рисунок 1.3 – Колекція користувача і зіграні партії

Додаток My BGG, випущений в 2019 році, в нього більше 10 тис. завантажень і оцінка в 4.2 з 5. Платформу відрізняє простий дизайн і доволі мілкий шрифт який не дуже зручно сприймати на око, який зображено на рисунку 1.4. Але, щодо функціоналу, тут дійсно широкий асортимент, крім стандартних рішень, які доступні на BoardGameGeek, а це: відвідати форум, подивитись коментарі чи залишити власний, доступ до пошуку ігор і додання їх в свою колекцію. Також є можливість переглянути розширену статистику, наприклад де зіграно партій, з ким їх зіграно, в якій день тижня, також додаток дозволяє переглянути відсоток перемог. Статистика тут більш детальна ніж на сайті, але тут присутня інформація від сайту, де можна переглянути його розділи, але інколи це зроблено не дуже красиво, наприклад: при переході на статтю відкривається сторінка сайту в додатку.

.

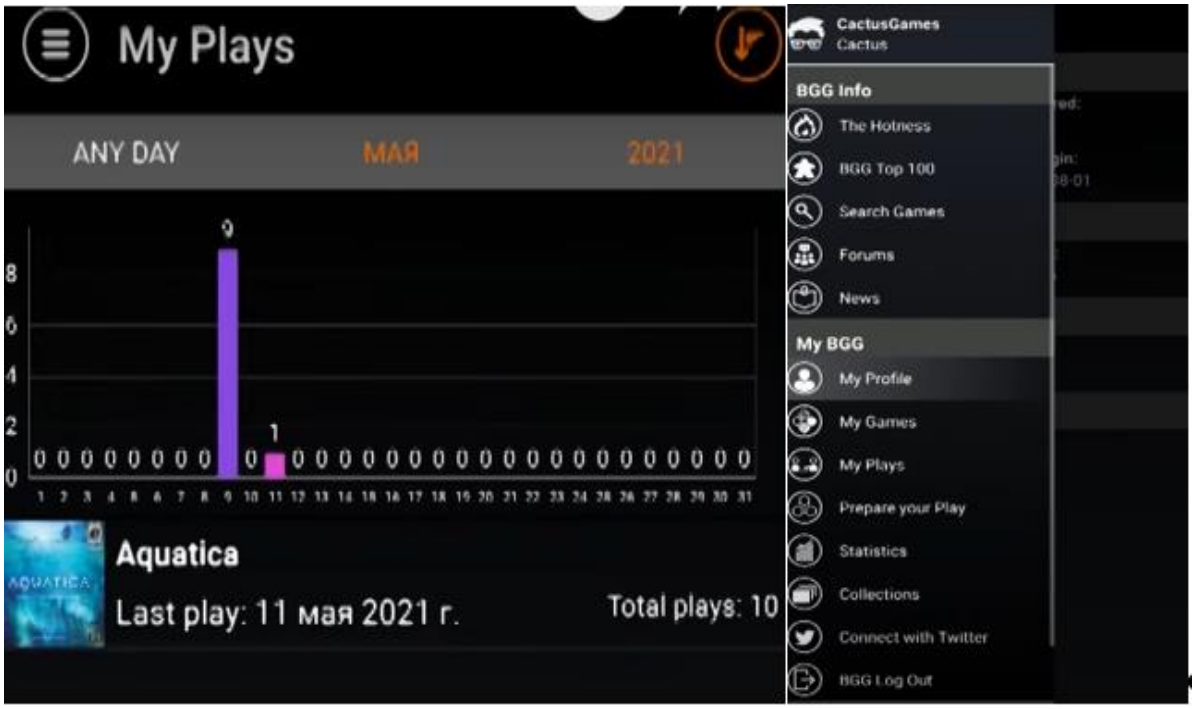


Рисунок 1.4 – Навігаційне меню та головна сторінка

Наступний і найкращий серед мобільних аналогів – це додаток BGStats. Перше, що потрібно сказати – це те, що додаток не є безкоштовний. Випущений

в 2017 році має оцінку в 4.8 з 5 і більше ніж 15 тисяч завантажень. Додаток відразу зустрічає простим і строгим дизайном з приємним зеленим кольором як зображено на рисунку 1.5. Дизайн додатку на нормальному рівні, але якщо говорити про графіки і гістограми вони привертають увагу своєю наочністю.

Рисунок 1.5 – Сторінки додатку BGStats

Функціонал являється високою перевагою його додатку, тут уся базова статистика і зверху, ще пласт можливостей. Є змога переглянути статистику по зіграним партіям в різних варіаціях: по кожній грі, по дням тижня, по кількості гравців, можна переглянути загальний відсоток перемог, середню кількість очків. На перший погляд статистика нічим не відрізняється від попередніх аналогів, але насправді вона тут набагато гнучкіша і візуалізована краще. Додаток дозволяє сортувати дані по різних критеріях, дає змогу не просто переглянути статистику за останій місяць, рік чи весь час, а й обрати будь-який проміжок часу, можна зробити експорт, імпорт даних, а також зробити облачну синхронізацію, яка дозволяє ділитись з іншими гравцями. За додаткову плату можна придбати: різні завдання, які гравець може поставити сам собі, а додаток буде спостерігає за їх виконанням; можна створювати власні фільтри, ці фільтри дозволяють об’єднувати партії в різні групи і по ним робити окрему статистику; можна

отримати, ще глибшу статистику, яка дасть досить багато нового. Наприклад: гравець хоче переглянути в грі статистику по очкам і глибока статистика показує максимальну кількість очків за весь час, мінімальну кількість очків, максимальну кількість очків при програші, середню кількість очків, і хто досягнув цих результатів.

Отже, після аналізу існуючих рішень, можна визначити наступні функції, якими повинен володіти майбутній додаток, а саме можливість:

– додавати ігри в колекцію;

– додавати партії;

– створювати місця і гравців;

– переглядати історії партій, загальну та окрему по певних критеріях;

– мати приємний дизайн і надавати широкий асортимент для користування;

– ділитись партіями з іншими гравцями.

1.3 Аналіз вимог до програмного забезпечення та розробка технічного завдання

Після аналізу предметної області та аналогів на ринку, потрібно визначитись з вимогами до веб-додатку, та описати їх. Найкращий спосіб для цього – використання мови UML. UML (англ. Unified Modeling Language) – уніфікована мова об’єктно-орієнтованого моделювання, може застосовуватись на усіх етапах життєвого циклу розробки додатків. Для цього використовуються різні діаграми, які надають можливість представити систему у такому вигляді, щоб її можна було легко перевести в програмний код. Щоб визначити вимоги, потрібно створити діаграмму варіантів використання.

Суть діаграмми варіантів використання полягає в наступному: система, яка проектується, подається у вигляді множини сутностей або акторів, які взаємодіють з системою за допомогою так званих варіантів використання. При

цьому актором називається будь яка сутність, що взаємодіє з системою зовні, а варіанти використання описують сервіси, яка система надає акторові.

Спершу, потрібно описати усіх можливих акторів, котрі будуть задіяні в системі. Усі користувачі (актори) представлені у таблиці 1.1.

Таблиця 1.1 – Опис акторів розроблюваного ПЗ

Актор	Короткий опис
Неавторизований користувач	Немає ніякий прав і не може здійснювати операції в додатку, крім авторизації і реєстрації з подальшою аторизацією.
Авторизований користувач	Може виконувати операції з колекцією, гравцями, місцями, партіями, а також може створювати рейтинги ігор, ділитися партіями з іншими зареєстованими користувачами.
Супер-адміністратор	Має усі права, крім того може надавати права іншим користувача.
Адміністратор	Має права редактора і є можливість редагувати інформацію про ігри, теми для статтей, ігрових жанрів і переглядати та оцінювати ідеї користувачів.
Редактор	Отримує можливість перевірки та редагування коментарів користувачів.

Після переліку акторів, їх короткого опису та їх можливостей, потрібно описати варіанти використанням з вказанням найменуванням та актора, до якого вони відносяться. В таблиці 1.2 наведені варіанти використання акторів розроблюваного програмного забезпечення.

Таблиця 1.2 – Опис варіантів використання розроблюваного ПЗ

Актор	Найменування	Опис
Неавторизований користувач	Вхід	Користувач вводить пошту та пароль для авторизації.
Авторизований користувач	Реєстрація	Користувач може зареєструватись в додатку.
	Створити суб’єктивний рейтинг ігор	Можливість обрати позицію гри в рейтингу без оцінок.
	Створити об’єктивний рейтинг ігор	Можливість оцінити гру по критерієм з подальшим відображенням списку посортований по середньому балу.
	Редагування акаунта	Можливість зміни налаштувань користувача.
	Додання ідей	Можливість додати ідею функції для сайта.
	CRUD операції для завдань	Можливість додати\редагувати чи видалити завдання.
	CRUD операції для цілей	Можливість додати\редагувати чи видалити ціль для завдання.
	CRUD операції для гравців	Можливість додати\редагувати чи видалити гравців.
	CRUD операції для місць проведення	Можливість додати\редагувати чи видалити місця, де грались парії.
	CRUD операції для ігор в колекції	Можливість додати\редагувати чи видалити гру в колекції.

Кінець таблиці 1.2

	CRUD операції для ігрових партій	Можливість додати\редагувати чи видалити партію.
Авторизований користувач	Комбінування ігор	Можливість об’єднати статистику ігор
	Зміна налаштувань гри	Можливість вказування середнього часу гри, стандартного місця проведення.
	Редагування ролей	Можливість редагувати ролі, які приймали участь у партії.
	Редагування плейліста	Можливість створити плейліст з пісень для окремої гри
Супер-адміністратор	Надання ролей користувачам	Можливість супер-адміністратору надати роль користувачу для виконання певних операцій.
Супер-адміністратор\ адміністратор	Редагування статусу ідей користувачів	Можливість змінити статус ідеї на: в розробці, відхилено чи прийнято до уваги.
	Редагування параметрів гри	Можливість змінити інформацію про гру
	Редагування тем статей	Можливість додати\видалити чи редагувати тему для статей
	Редагування ігрових жанрів	Можливість додати\редагувати чи видалити ігровий жанр
Супер-адміністратор\ адміністратор\редактор	Редагування коментарів	Можливість редагування чи видалення коментарів

Після того, як визначено актори та варіанти використання, можна побудувати діаграмму використання, як зображено на рисунку 1.6 і рисунку 1.7.

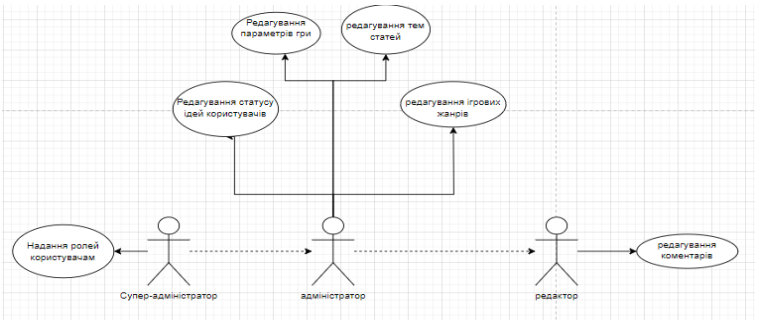


Рисунок 1.6 – Діаграмма використання адміністрації та редактора

На основі аналізу вимог було розроблено технічне завдання, яке подано у додатку А.

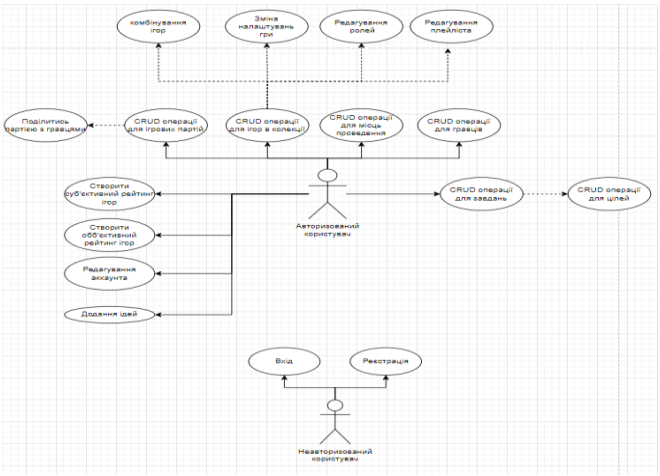


Рисунок 1.7 – Діграмма використання авторизованого та неавторизованого користувача

В результаті проведеної роботи в розділі було проаналізовано особливості розробки статистичних сайтів для ведення статистики ігор. Також були розглянуті наявні програмні забезпечення з виділенням проблем, та переваг кожної, в результаті чого було визначенно функціональні вимоги до розроблюваного програмного забезпечення.

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕНЯ

2.1 Аналіз та вибір архітектури систем

Оскільки майбутнє програмне забезпечення являє у сутності своїй сайт з інтерактивом, то при розробці було прийнято рішення використовувати клієнт-серверну архітектуру, на якій побудовані усі сайти і інтернет-сервіси. Також її використовуються і десктоп-додатки, які передають дані через інтернет.

Клієнт – локальний комп’ютер на стороні віртуального користувача, який виконує відправку запита на сервер для того, щоб надати дані чи для виконання певної групи дій.

Сервер – досить потужний комп’ютер або ж спеціальне системне обладнання, яке призначене для того, щоб отримувати запити від клієнта і обробляти їх виконуючи певну групу дій, також сервер слугує для надання доступу до системних ресурсів.

Особливостями такої моделі є те, що один сервер може слугувати для обробки запитів одразу від декількох клієнтів одночасно, якщо одночасно приходить декілька запитів, то такі запити стають в певну чергу і сервер виконує їх по черзі. Запити також можуть мати пріоритет виконання, в такому випадку запити з більш високим пріоритетом будуть виконуватись в першу чергу.

Архітектура клієнт-сервер слугує тільки для формулювання принципів віртуального обміну повідомлень між локальними комп’ютерами, а усі правила взаємодії знаходяться всередині протоколу.

Мережевий протокол – це особливий набір правил, виходячи з якого виконується точна взаємодія між комп’ютерами всередині віртуальної мережі, існують наступні мережеві протоколи:

– TCP – вид протоколу, який є сполучною ланкою для встановлення якісного з'єднання між двома пристроями, передачі даних та верифікації їх отримання;

– IP – протокол, до якого входить коректність доставки повідомлень за обраною адресою. У ньому інформація ділиться на пакети, які можуть постачатися по-різному;

– MAC – вид протоколу, виходячи з якого відбувається процес верифікації мережевих обладнань. Всі пристрої, які підключені до мережі Інтернет, містять свою оригінальну MAC-адресу;

– ICMP – протокол, який відповідає за обмін даними, але не використовується для передачі інформації;

– UDP – протокол, керуючий передачею даних, але дані проходять верифікацію при отриманні. Цей протокол працює швидше, ніж протокол TCP;

– HTTP – протокол передачі інформації (гіпертексту), з урахуванням якого функціонують всі сьогоднішні сайти. У його можливості входить процес запитування необхідних даних у віртуально віддаленої системи (файли, веб-сторінки та інше).

Саме на HTTP і буде працювати розроблюваний проект. Методи, які використовує протокол:

– GET –для отримання інформації;

– POST –для відправлення інформації;

– PUT –для оновлення інформації;

– PATCH –для часткового оновлення інформації;

– DELETE –для видалення інформації;

– OPTIONS –для опису параметрів з’єднання з обраною інформацією на сервері.

В нашому випадку концепція побудови клієнт-серверної системи буде полягати в принципі «Слабкий клієнт – продуктивний сервер». У такій моделі весь процес обробки інформації перенесений на потужність серверу, а у користувача права доступу досить обмежені. Клієнт взаємодіє з сервером створюючи і відправляючи запит, сервер приймає вхідні дані, обробляє і відправляє вже оброблені дані, після чого клієнту виводяться дані на екран.

В результаті нюанси взаємодії системи клієнт-сервер дозволяють користувачам розподіляти функціонал і обчислювальні навантаження між підключеними клієнтськими веб-продуктами і серверними додатками при різноманітних процесах тестування (від тестування бази даних до замірів загальної продуктивності системи).

2.2 Опис структури та моделі бази даних

Найкращий варіант для зберігання даних і найбезпечніший – використання бази даних. База даних – це інтегрована сукупність структурованих і взаємозалежних даних, організована за певними правилами, які передбачають загальні принципи опису, зберігання і обробки даних.

Реляційна база даних – це тип бази даних, що зберігає інформацію в електронних таблицях і здійснює пошук даних в одній таблиці на підставі ключових полів іншої таблиці.

Реляційні бази даних мають низку переваг і недоліків. До переваг відносяться:

– простота і доступність для розуміння користувачем;

– повна незалежність даних;

– для організації запитів і написання прикладного ПЗ немає необхідності знати конкретну організацію БД у зовнішній пам'яті;

– суворі правила проектування.

Недоліки реляційної бази даних:

– далеко не завжди предметна область може бути представлена у вигляді «таблиць»;

– в результаті логічного проектування з'являється множина «таблиць»;

– БД займає відносно багато зовнішньої пам'яті;

– відносно низька швидкість доступу до даних.

Порівнявши переваги та недоліки реляційної бази даних було прийнято рішення використовувати вільну реляційну систему управління базами даних MySQL.

Основуючись на діаграмі використання, яка була створена раніше, можна виокремити дані, які потрібно буде зберігати, а саме: користувачів, теми обговорень, завдання, цілі, місця, гравців, налаштування для партії, саму партію, рейтинг по оцінкам, рейтинг по бажанню, ідеї користувачів, жанри ігрові, ролі користувачів, пости та коментарі до них та інші, перелік таблиць з описом і списком атрибутів описаний нижче.

User – колекція користувачів системи які надали дані для реєстрації, містить наступні атрибути:

- id – унікальний ідентифікатор користувача;
- name – ім’я користувача;
- email – електрона пошта користувача;
- password – пароль від акаунта;
- created_at – дата створення акаунта користувача;
- updated_at – дата оновлення акаунта користувача.

Topics – теми для обговорень, містить наступні атрибути:

- id – унікальний ідентифікатор теми;
- name – назва теми;
- created_at – дата створення теми;
- updated_at – дата оновлення теми.

Tasks – колекція завдань користувача, містить наступні атрибути:

- id – унікальний ідентифікатор завдання;
- name – назва завдання;
- challenge_id – сутність цілі;
- finished – статус виконання;
- created_at – дата створення завдання;
- updated_at – дата оновлення завдання.

Challenges – колекція цілей створені користувачем для конкретного завдання, містить наступні атрибути:

– id – унікальний ідентифікатор цілі;

– name – назва цілі;

– tasks_id – сутність завдання;

– finished – статус виконання цілі;

– user_id – сутність користувача;

– created_at – дата створення цілі;

– updated_at – дата створення цілі.

Genres – колекція жанрів настільних ігор, містить наступні атрибути:

– id – унікальний ідентифікатор жанру;

– name – назва жанру;

– created_at – дата створення жанру;

– updated_at – дата оновлення жанру.

Roles – колекція ролей користувачів, містить наступні атрибути:

– id – унікальний ідентифікатор ролі;

– name – назва ролі;

– created_at – дата створення;

– updated_at – дата оновлення.

Ratings – колекція оцінок користувача для гри по певним характеристикам, містить наступні атрибути:

– id – унікальний ідентифікатор рейтингу;

– game_id – сутність гри;

– user_id – сутність користувача;

– impression – загальні враження від гри;

– atmospheric – атмосферність гри;

– replay – рівень реграбельності;

– complexity – рівень складності гри;

– quality – якість компонентів гри;

– interaction – рівень взаємодії між гравцями;

– entertainment – рівень розважальності гри;

– created_at – дата створення рейтингу;

– updated_at – дата оновлення рейтингу.

Posts – колекція містить інформацію про створені користувача, пости під відповідними темами до гри, містить наступні атрибути:

– id – унікальний ідентифікатор поста;

– title – назва поста;

– user_id – сутність користувача;

– topic_id – сутність теми;

– game_id – сутність гри;

– created_at – дата створення поста;

– updated_at – дата оновлення поста.

Players – колекція гравців, з якими проводяться ігрові партії, містить наступні атрибути:

– id – унікальний ідентифікатор користувача;

– user_id – сутність користувача;

– friend_id – сутність користувача, зареєстрованого на сайті;

– name – ім’я гравця;

– image – зображення гравця;

– created_at – дата створення гравця;

– updated_at – дата оновлення гравця.

Places – колекція місць проведення партій користувача, містить наступні атрибути:

– id – унікальний ідентифікатор місця;

– user_id – сутність користувача;

– name – назва місця;

– default – статус місця по замовчуванню;

– created_at – дата створення місця;

– updated_at – дата оновлення місця.

Match_Settings – колекція налаштувань до партії, містить наступні атрибути:

– id – унікальний ідентифікатор налаштування партії;

– user_id – сутність користувача;

– game_id – сутність гри;

– place_id – сутність місця;

– comment – коментар до партії;

– date – дата зіграної партії;

– rounds – кількість раундів;

– timer – таймер;

– include – статус, включити в статистику партію чи ні;

– created_at – дата створення налаштувань партії;

– updated_at – дата оновлення налаштувань партії.

Matches – колекція партій зіграних користувачем, містить наступні атрибути:

– id – унікальний ідентифікатор партії;

– match_settings_id – сутність налаштувань партії;

– player_id – сутність гравця;

– victory – статус перемоги;

– role – роль гравця;

– startPlayer – перший гравець;

– newPlayer – новий гравець;

– startPosition – стартова позиція;

– type – тип гри;

– score – переможні бали;

– circumstances – назва перешкоди;

– team – назва команди;

– created_at – дата створення партії;

– updated_at – дата оновлення партії.

Comments – колекція коментарів до поста, містить наступні атрибути:

– id – унікальний ідентифікатор коментаря;

– post_id – сутність поста;

– user_id – сутність користувача;

– text – текст коментаря;

– parent_id – сутність батьківського коментаря;

– created_at – дата створення коментаря;

– updated_at – дата оновлення коментаря.

User_tops – колекція рейтингу ігор, впорядкованих по бажанню користувача до поста, містить наступні атрибути:

– id – унікальний ідентифікатор рейтингу;

– game_id – сутність гри;

– user_id – сутність користувача;

– created_at – дата створення рейтингу;

– updated_at – дата оновлення рейтингу.

User_ideas – колекція ідей користувача для поліпшення сайту, містить наступні атрибути:

– id – унікальний ідентифікатор ідеї;

– user_id – сутність користувача;

– text – текст ідеї;

– status – статус ідеї;

– created_at – дата створення рейтингу;

– updated_at – дата оновлення рейтингу.

Combined_games – колекція комбінованих ігор для злиття статистики, містить наступні атрибути:

– id – унікальний ідентифікатор комбінації;

– user_id – сутність користувача;

– game_first – сутність гри для комбінації;

– game_second – сутність гри, з якою відбувається комбінація;

– created_at – дата створення комбінації;

– updated_at – дата оновлення комбінації.

Game_playlists – колекція пісень користувача для конкретної гри:

– id – унікальний ідентифікатор ідеї;

– user_id – сутність користувача;

– game_id – сутність гри;

– url – посилання на трек;

– name – назва треку;

– created_at – дата створення пісні;

– updated_at – дата оновлення пісні.

Отже, після як було зроблено перелік усіх необхідних таблиць для мінімального функціонування системи з подальшою можливістю для модифікації, можна буде переходити для відтворення фізичної моделі бази даних.

2.3 Проектування серверної частини веб-додатку

Оскільки спроектована Інтернет-платформа буде виконувати майже усі, функції без неї візуальна частина не матиме ніякого функціоналу, отже серверна частина є однією з найважливіших складових частин. Вона повинна пов’язувати між собою роботу клієнтської частини сайту та базу даних.

Сервер отримує запит від клієнта і залежно від того, що потрібно виконати, змінює чи оновлює дані в базі даних, отримує дані та обробляє їх, відправляючи відповідь для відображення у користувача. Для керування такими запитами буде використовуватись програмний інтерфейс REST API.

REST API (Representational state transfer) – це прикладний програмний інтерфейс, який використовує HTTP-запити для отримання, розміщення і видалення даних. Базується на виконані шести вимог, а саме:

– клієнт-серверна модель – розподілення на клієнт-сервер дає можливість розвиватись частинам однієї системи незалежно одна від одної;

– відсутність стану – при дотримані цієї вимоги, клієнтська та серверна частина зможуть «розуміти» один одного без опори на отримання одна від одної даних;

– кешування – виконання кешування відповідей з боку серверної частини дає можливість виключити ймовірність надходження невірних, або вже застарілих відомостей клієнтам;

– одноманітність інтерфейсу – використання єдиного формату взаємодії, що описує особливості надсилання запитів та відповідей, робить інтерфейс уніфікованим;

– шари абстракції – кожен компонент потрапляє в якийсь шар і спілкується лише з компонентами в шарі під ним або в шарі над ним;

– запитування коду – єдине необов'язкове обмеження. Якщо йому слідувати, кожен клієнт зможе збільшити функціональність, відправивши запит на сервер для наступного завантаження коду.

Щоб отримати інформацію з сервера використовується POST та GET запити. Відмінність їх у тому, що GET використовується для отримання різної інформації, яка потрібна користувачу, а POST використовується для збереження даних. Ці формати визначені завдяки стандартному протоколу HTTP.

2.4 Проектування інтерфейсу користувача

Загалом будь-яка інтернет-платформа використовує прийняті шаблони структури, які складаються з декількох частин та є досить актуальними для усіх сайтів.

В цілому веб-додаток може складатись з п’яти частин:

– шапка (header) – це верхня частина сайту, яка як правило включає в себе назву компанії, логотип, панель навігації, основні контактні дані. Представляє

собою один з основних елементів оформлення веб-платформи, який впливає на його привабливість у Мережі та зручність користування сайтом загалом. Зрозуміло, що хеддер також важливий і для юзабіліті, оскільки надає відвідувачам ключову інформацію про ресурс. Основні елементи шапки повинні відповідати на запитання користувачу: що являє собою компанія, що вона пропонує, як з нею зв'язатися, які є акції, гарантії та інше;

– навігація (nav) – для зручності переміщення сторінками необхідна продумана навігація по сайту. Це система засобів, що допомагають шукати потрібну інформацію. Завдання розробника – зробити так, щоб відвідувач дістався сторінки за 2-3 кліки, але це можливо не завжди;

– головна частина (main) – основна частина сайту, яка містять усю необхідну інформацію і займає найбільше місця на сторінці;

– бокова панель (sidebar) – це закріплена бічна панель ресурсу, область навігації або допоміжної інформації, що графічно відокремлена від основної області контенту. Бічна панель необхідна для того, щоб допомагати відвідувачам пересуватися сайтом, знаходити певний контент або скористатися будь-яким функціоналом. У сайдбарі можуть розташовуватися: навігаційне меню, інформаційні блоки, функціональні елементи, оголошення з рекламою, пропозиції товарів та послуг, додаткові віджети;

– підвал сайту (footer) – це наскрізний структурний елемент, розташований у нижній частині сторінки. Використовується, переважно як блок додаткової інформації. У підвалі сайту, зазвичай, вказується інформація, яку потрібно легко знайти на будь-якій сторінці, тому там може частково дублюватися зміст шапки. Конкретний набір даних залежить від типу сайту, тематики, дизайну, змісту інших блоків і т. д. Типові приклади сторінки контенту, що розміщується внизу: контакти, навігація, правова інформація.

Можна обійтись без усіх складових частин крім основної, але це буде сильно впливати на роботу сайту.

2.5 Створення макету веб-додатку та дизайн

Макет веб-додатку – це його реалістичний прототип, створений в графічній програмі на кшталт Photoshop. Він виглядає, як готовий сайт: зі всіма блоками тексту і зображеннями, кнопками, фоновими малюнками і іншими елементами. Єдина відмінність в тому, що в макеті немає функціонального наповнення.

Макет сайту – результат роботи дизайнера по завданню замовника. Далі макет переходить до програміста та верстальщика, вони перетворюють графічний файл в справжній сайт.

В ідеалі, в створені макету мають приймати участь багато спеціалістів, серед них: копірайтер, дизайнер користувацьких інтерфейсів, програміст бекенда та маркетолог.

Перше, що потрібно зробити – підібрати кольори не більше п’яти: пара для шрифту, пара базових кольорів для фону і один для акцентування уваги.

На рисунку 2.1 зображено чотири основних темних кольори, які будуть використовуватись для фону та блоків, палітра для кращого і зручнішого підбору кольорів для акцентування уваги, серед яких відбираються три основних.

Після чого було створені направляючі для сайдбару і навігаційного меню під які будуть підлаштовуватись об’єкти на сторінці.

Потім було створено базові елементи – блоки інформативні: статистика за період, блок з останньою партією, інформацією про колекцію, статистика по дням тижня і кількості гравців та інша статистика разом з елементами сайдбару і навігаційного меню.

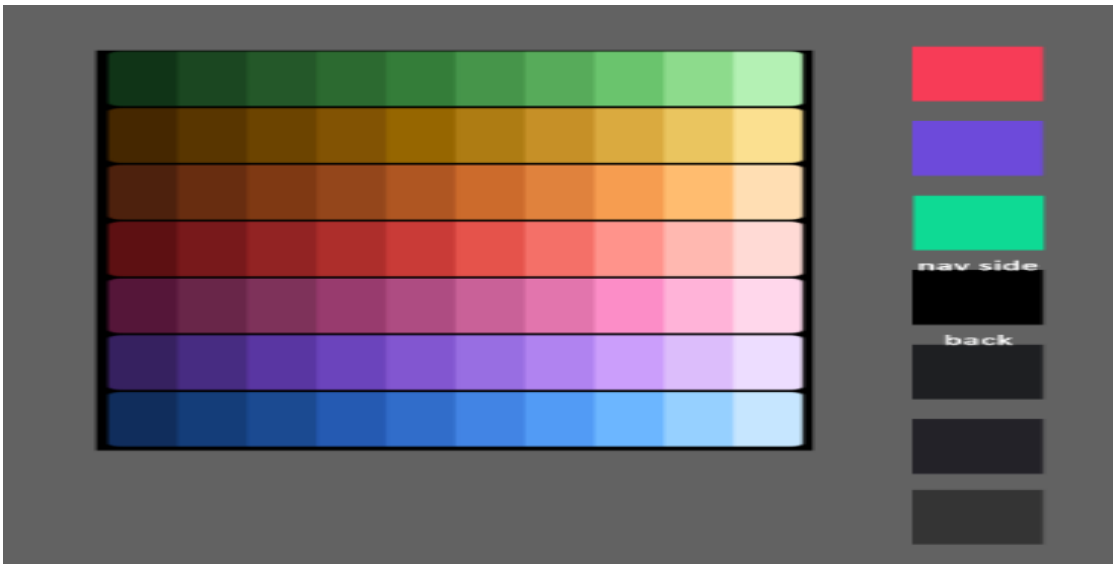


Рисунок 2.1 – Палітра кольорів і інші кольори

В решті, почалась детально пророблятись кожна з областей. Готовий варіант прототипу зображений на рисунку 2.2.

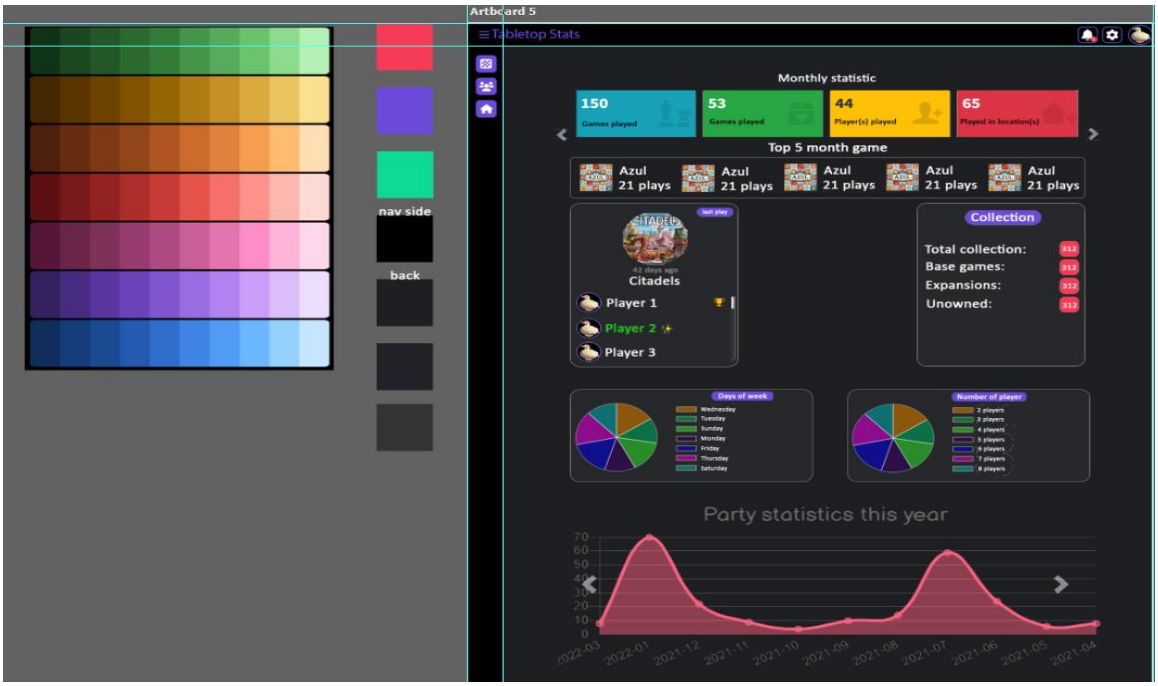


Рисунок 2.2 – Результат створення прототипу головної сторінки

Після того як макет був розроблений, можна починати розробляти його реальний варіант з можливими змінами, як зображено на рисунку 2.3.

Рисунок 2.3 – Головна сторінка сайту

На прикладі головної сторінки можна переглянути готовий результат, який містить усю потрібну інформацію. Нижче на рисунку 2.4 приведено результат з відкритим сайдбаром.

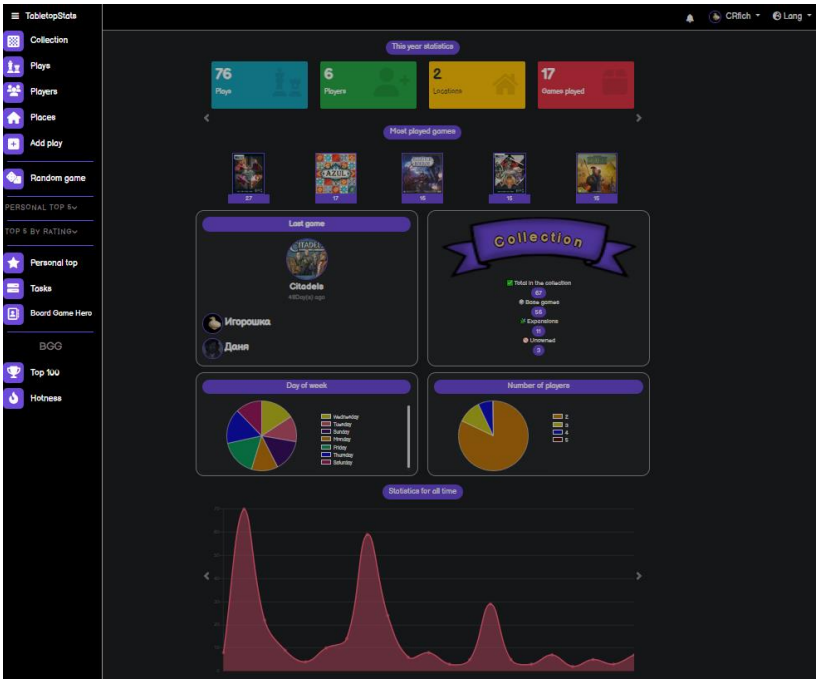


Рисунок 2.4 – Головна сторінка з відкритим сайдбаром

Елементи сайдбару було впорядковано за частотою використання і розділено на блок, в закритому ж вигляді відображають тільки самі потрібні елементи.

2.6 Аналіз та вибір технологій і методів реалізації системи

Основу інтернет-платформи буде складати фреймворк Laravel – безкоштовний веб-фреймворк із відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC, фреймворк має достатню кількість переваг, серед них:

– велика спільнота підтримує фреймворк – завдяки відкритому вихідному коду та високій популярності даного фреймворку. Замовнику дуже легко знайти компанію, що спеціалізується на фреймворку Laravel, а виконавцю надається велика бібліотека написаних програм для Laravel;

– MVC – структура Model-View-Controller дозволяє ізолювати один від одного компоненти для виконання різноманітних завдань;

– ORM – у Laravel використовується Eloquent ORM, яка спрощує роботу з базою даних;

– шаблонізатор – в Laravel використовується легковажний і високопродуктивний (завдяки кешуванню) шаблонізатор Blade, за допомогою якого можна легко стандартизувати та використовувати шаблон;

– аутентифікація та інтеграція з сервісами – в Laravel, завдяки пакету Socialite, з самого початку є можливість авторизації користувача через різні сервіси, а також доступна різноманітність драйверів для роботи з email та розсилкою SMS повідомлень;

– PHPUnit-тести – ручне тестування підходить переважно для невеликих проектів. При масштабних проектах, автоматичне тестування економить значну кількість часу, що, відповідно, економить бюджет. Також у Laravel можна виконувати модульне та функціональне тестування;

– модульність – Laravel надає вбудовані бібліотеки та модулі, які допомагають покращити веб-програму. Кожен модуль інтегрований із менеджером залежностей Composer, що спрощує оновлення;

– безпека – Laravel пропонує ряд вбудованих функцій безпеки: захист від SQL-ін'єкцій, захист від підробки міжсайтових запитів (CSRF), захист від XSS атак та ін.;

– продуктивність – завдяки кешуванню, оптимізації фронтенду та правильному поділу коду на компоненти, вдається забезпечувати справді швидкий доступ до даних;

– мультимовність – Laravel чудово підходить для мультимовного проекту. У цьому PHP-фреймворку не важко налаштувати багатомовність для вашого сайту.

– помилки та винятки – обробка помилок та виключень доступна «з коробки» для будь-якого нового проекту на Laravel. Крім того, Laravel інтегрований з бібліотекою журналів Monolog, яка забезпечує підтримку багатьох потужних обробників журналів.

– система міграцій БД – спрощує розгортання та оновлення веб-програми, позбавляючи розробника від помилок та конфліктів, особливо якщо над проектом працює команда розробників.

Також буде використовуватись SCSS – "діалект" мови SASS. SASS – це мова схожа на HAML (дуже лаконічний шаблонізатор), але призначена для спрощення створення CSS-коду. Простіше кажучи, SASS – це така мова, код якої спеціальною ruby-програмою транслюється у звичайний CSS код. Синтаксис цієї мови дуже гнучкий, він враховує безліч дрібниць, які так бажані в CSS. Більше того, в ньому є навіть логіка (@if, @each), математика (можна складати числа, рядки, так і кольори).

Також буде використовуватись шаблонізатор Blade – це простий, але потужний двигун шаблонів, що входить до складу Laravel. На відміну від деяких шаблонизаторів PHP, Blade не обмежує розробника у використанні звичайного "сирого" коду PHP у шаблонах. Насправді всі шаблони Blade компілюються у звичайний PHP-код і кешуються доти, доки не будуть змінені, що означає, що Blade додає фактично нульове навантаження вашому додатку.

JasvaScript та jQuery – розроблені спеціально для того, щоб створювати інтерактивні сайти. Такі сайти реагують на ваші дії, бібліотека jQuery ж допомагає легко отримувати доступ до будь-якого елемента DOM, звертатися до атрибутів та вмісту елементів DOM, маніпулювати ними.

PHP – мова програмування на якій буде написаний додаток. Головна сфера застосування PHP – написання скриптів, що працюють на стороні сервера; таким чином, PHP здатний виконувати все те, що виконує будь-яка інша програма CGI, наприклад, обробляти дані форм, генерувати динамічні сторінки або надсилати та приймати cookies

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

Після того, як була спроектована база даних у другому розділі, можна приступити до розробки її в проекту. Для цього будуть використовуватись міграції.

Міграції – це щось подібне до системи контролю версій для бази даних. Вони дозволяють команді розробників змінювати структуру бази даних, водночас залишаючись у курсі змін інших учасників. Зазвичай міграції йдуть пліч-о-пліч з конструюванням структур для більш простого поводження з архітектурою нашої бази даних. Міграції бази даних вирішують проблему додавання стовпця у локальну базу даних.

Фреймворк Laravel, який використовувався при розробці має зручний спосіб для створення і взаємодії з міграціями. Щоб додати міграцію потрібно виконати команду php artisan make:migration {назва міграції}, міграція буде одразу ж переміщена в директорію database/migrations. Назва файла кожної міграції також буде містити часову мітку, щоб фреймворк міг визначити порядок використання міграцій.

Також можна використовувати параметри --table і --create для вказування імені таблиці і того факту, що міграція буде створювати нову таблицю. Ці параметри просто завчасно створюють вказану таблицю в створюваному файлі міграції.

Будь-який клас міграції містить два методи: up і down. Перший метод використовується для створення нових таблиць чи оновлення старих, другий же скасовує операції, які виконуються в методі up, нижче буде приведений приклад міграції:

```php
public function up()
{
    Schema::create('match_settings', function (Blueprint $table) {
        $table->id();
        $table->foreignId("user_id")->constrained()->onDelete("cascade")->onUpdate("cascade");
        $table->foreignId("game_id")->constrained()->onDelete("cascade")->onUpdate("cascade");
        $table->foreignId("place_id")->constrained()->onDelete("cascade")->onUpdate("cascade");
        $table->text("comment",20000)->nullable();
        $table->date("date");
        $table->integer("rounds")->nullable();
        $table->time("timer")->nullable();
        $table->boolean("include")->nullable();
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('match_settings');
}
```

Рисунок 3.1 – Фрагмент коду міграції

В обох методах можна використовувати будівельник структур Laravel для зручного створення і змін таблиць. Таким чином, за допомогою міграції буде створено структуру бази даних, яка описана в другому розділі.

3.2 Розробка серверної частини Інтернет-платформи

У попередніх розділах було вказано, що в основі проекту лежить фреймворк Laravel, спершу його потрібно завантажити. Структура проекту наступна:

– Database – директорія містить файли міграції бази даних, моделі фабрик для штучного створення даних наших таблиць і моделі для «посіву» штучних елементів створених в фабриках для тестування системи.

– Public – містить index.php файл, який є відправною точкою усіх запитів нашого додатку. Також є місцем збереженням наших зображень, файлів CSS і JavaScript.

– Resources – директорія, яка відповідає за збереження наших відображень, некомпільованих файлів CSS і JavaScript, також містить файли локалізацій.

– Routes – містить усі визначення маршрутів для нашої програми, за замовчуванням включено кілька файлів маршрутів: web.php, api.php, console.php і channels.php.

– Tests – директорія, яка містить наші тести.

– Vendor – містить усі залежності композеру.

– Bootstrap – містить файл, який завантажує фреймворк, також містить каталог кешу, який містить файли, згенеровані фреймворком для оптимізації продуктивності, такі як файли кешу маршрутів і служб.

– App – містить основний код нашої програми.

Спершу буде приведений код контролеру який відповідає за сторінку з електронним постером і містить два методи, перший відповідає за завантаження сторінки з інформацією про виконані завдання, другий за зміну статусу завдання:

```
class HeroController extends Controller
{
    public function BoardGameHero()
    {
        $skills_list = BGH::where('user_id', Auth::id())->pluck('skills')-
>first();
        return view("functions.boardgamehero",['skills_list'
=>$skills_list]);
    }
    public function BGHUpdate(Request $request)
    {
        $BGH_model = new BGH();
        $BGH_model->updateBGH($request);
        return response()->json(['data'=>BGH::where('user_id',
Auth::id())->pluck('skills')->first()]);
    }
}
```

Нижче наведено фрагмент коду, який відповідає за парсинг сторінки BoardGameGeek для пошуку ігор, щоб додати в колекцію, який в якості параметру отримує текст і по цьому текст здійснює пошук усіх відповідностей:

```
   public function BGGFind($game_name)
    {
        $games = [];
        $name = str_replace(" ","%20", $game_name);
        $url =
"https://boardgamegeek.com/search/boardgame/page/$page?q=$name";
        $client = new Client();
```

```
        $crawler = $client->request('GET', $url);
        $crawler->filter('.primary')->each(function($node) use
(&$game_name, &$games){
                $game =
                [
                    "name" => $node->text(),
                    "year" => $node->ancestors()->filter('.dull')->text(),
                    "url" => $node->attr('href'),
                    "image" => $node->ancestors()->ancestors()-
>ancestors()->filter('img')->getNode(0)->getAttribute('src'),
                ];
                array_push($games, $game);
        });
        return $games;
    }
```

Останній фрагмент коду, який буде приведено, відповідає за збір статистики для усіх ролей, які були вказані в процесі додання партій:

```
    public function RoleStatistic($match_sett)
    {
        $role_statistic = [];
        $plays = Matches::whereIn('match_settings_id',$match_sett)-
>where('role','!=', null)->select('role','match_settings_id','victory')-
>distinct('role')->get()->toArray();


        for ($i=0; $i < count($plays); $i++) {
                $list = explode(',',$plays[$i]['role']);

                for ($j=0; $j < count($list); $j++) {

            $check = in array( $list[$j] , array column($role statistic,
'role'));

            if($check){
                  $searched_index = array_search($list[$j] ,
array_column($role_statistic, 'role'));
                  $plays[$i]['victory'] === 1 ?
$role_statistic[$searched_index]["wins"] ++:
$role_statistic[$searched_index]["loses"]++ ;


            }
            else{
              array_push(
                  $role_statistic,
                  [
                      "role"=>$list[$j],
                      "wins"=>$plays[$i]['victory'] === 0 ? 0 : 1,
                      "loses"=> $plays[$i]['victory'] === 1 ? 0 : 1,
                      "percent" => 0,
                ]);
                }
            }
        }
        for ($i=0; $i < count($role_statistic); $i++) {
```

```
            $role_statistic[$i]['percent'] = $role_statistic[$i]["wins"]
=== 0 ? 0 : floor($role_statistic[$i]["wins"] /
($role_statistic[$i]["wins"]+$role_statistic[$i]["loses"]) * 100);
        }

        array_multisort(array_column($role_statistic, 'percent'),
SORT_DESC, $role_statistic);
        return $role_statistic;
    }
```

3.3 Керівництво користувача

Для роботи з платформою потрібно перейти на сайт, після чого потрібно провести авторизуватись чи зареєструватись на сайті для подальшого використання. Після того, як користувач увійшов в акаунт, його зустріне головна сторінка, яка надає статистику за різний період (місяць, рік, весь час), інформацію про останню зіграну партію, статус колекції, дві кругові діаграми з статистикою по кількості гравців, статистику по дням тижня, та діаграму по кількості партій за останній рій і весь час.

Рисунок 3.2 – Головна сторінка сайту

Також, в будь який момент користувач може змінити мову, після чого користувач є змога відкрити бокове меню і перейти на сторінку колекції ігор.

Користувач переходить на вкладку «Пошук на BGG» та вводить назву як зображено на рисунку 3.3 та рисунку 3.4.

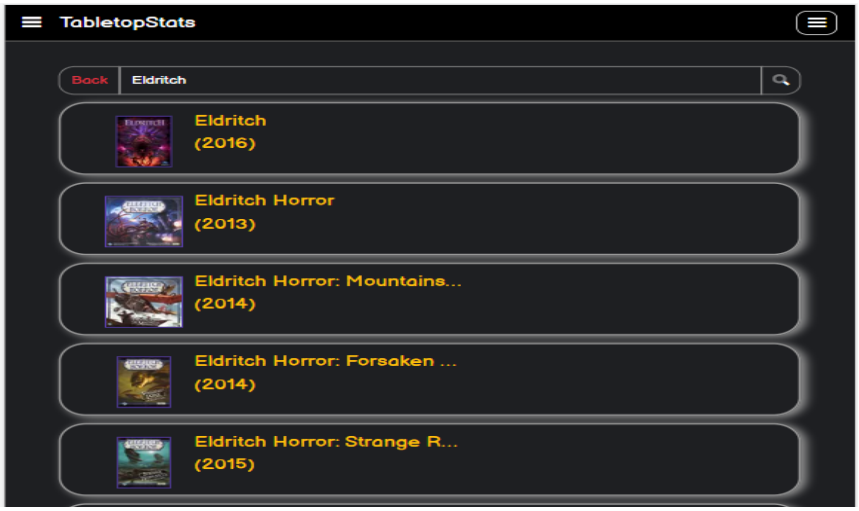


Рисунок 3.3 – Знайдені ігри

Після того, як відбувся пошук ігор чи доповнень по назві, необхідно натиснути на потрібну опцію і обрати тип гри, чи є в наявності гра і чи це базова гра або доповнення.

Рисунок 3.4 – Форма додання в колекції.

Після додання гри в колекцію можна перейти на сторінку гри, де є змога додати нову партію, але перед тим потрібно додати гравців та місця де проводяться партії.

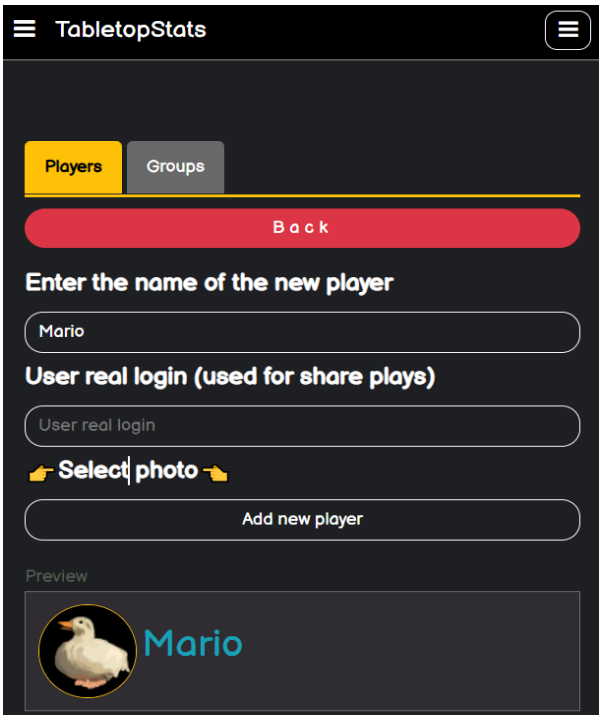


Рисунок 3.5 – Додання нового гравця

Тепер з наявними місцями і гравцями можна додати партію, для цього потрібно перейти на форму, обрати місце проведення партії, та запустити таймер, щоб визначити скільки тривала партія, обрати день гри, та вибрати гравців, і в залежності від типу гри заповнити потрібні поля, та обрати переможця.

Game	Plays	Wins	Loses	All plays
7 Wonders	2	2	0	100%
Dungeons and Dragons: The Legend of Drizzt Board Game	2	2	0	100%
Inis	1	1	0	100%
Lanterns: The Harvest Festival	1	1	0	100%
Neuroshima Hex! 3.0	1	1	0	100%
Red 7	2	2	0	100%
Star Wars: Rebellion	3	3	0	100%
Terraforming Mars	2	2	0	100%
The Quest for El Dorado	1	1	0	100%
Through the Ages: A New Story of Civilization	1	1	0	100%
Azul: Stained Glass of Sintra	4	3	1	75%
Nations	4	3	1	75%
Azul	22	15	7	68%
Flick 'em Up!: Dead of Winter	3	2	1	66%
Room 25	3	2	1	66%
Wingspan	6	4	2	66%
7 Wonders Duel	17	11	6	64%
Karuba	5	3	2	60%
Kingdomino	10	6	4	60%
Unmatched: Robin Hood vs. Bigfoot	10	6	4	60%
Hanamikoji	14	8	6	57%
Carcassonne Big Box	13	7	6	53%
Sagrada	15	8	7	53%
Star Realms: Frontiers	15	8	7	53%
Unmatched: Buffy the Vampire Slayer	26	14	12	53%
Patchwork	8	4	4	50%
Unmatched: Cobble and Fog	10	5	5	50%

Рисунок 3.6 – Приклад статистики по зіграним іграм

Крім того користувач має змогу внести додаткову інформацію про гравця в момент проведення гри: роль персонажа, початкове місце, чи вперше грав в гру і чи був першим гравцем.

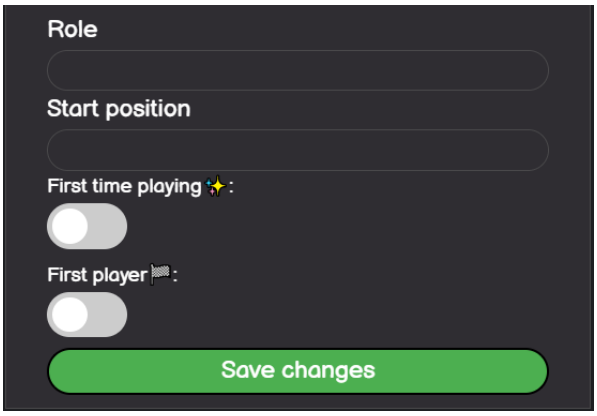


Рисунок 3.7 – Додаткова інформація про гравця

Крім базового функціоналу існує наявний функціонал для підвищення відвідування сайту. Прикладом такого функціоналу є електрона версія скретч-постеру «Board Game Hero», який містить завдання, що потребують стерання монеткою в фізичній версії. В електронній ж версії можна просто натиснути на потрібне завдання після виконання і відповідні характеристики одразу зміняться. Зображення постеру приведено нижче на рисунку 3.8.

Рисунок 3.8 – Приклад додаткової функції

Наявний функціонал буде змінюватись за вимогою користувачів і поліпшуватись для збільшення популярності та для зручного використання.

3.4 Технічні характеристики Інтернет-платформи

Оскільки програмний продукт представляє собою сайт, користувачу необхідний браузер Google Chrome або Opera і підключення до мережі Інтернет. Можна виділити наступні технічні характеристики для використання :

- процесор – 233 MHz;
- оперативна пам'ять – 1024 Мб;
- відеоадаптер і монітор;
- пристрої для взаємодії з додатком – мишка чи тачпад та клавіатура;
- оперативна система.

Для підтримки сайту необхідний сервер з наступними мінімальними характеристиками:

- дисковий простір – 1 Гб;
- передача даних – 5 Гб/місяць;
- FTP–акаунти – 2 шт.;
- пропускна спроможність – 1 Гб/сек.

3.5 Завантаження Інтернет-платформи на хостинг

В процесі розробки було обрано українську фірму, яка займається хостингом zzz.com.ua, і надає як безкоштовні так і платні пакети послуг. Серед них можна вибрати те, що найкраще підходить користувачу,уснує наявна можливість зареєструвати власний домен. Для першої роботи з хостингом було обрано безкоштовний варіант, який характеризується наступними параметрами:

- дисковий простір 5 Гб;
- дисковий простір для пошти 1 Гб;

– Email-акаунти 1шт;

– передача даних 5 Гб;

– кількість файлів до 60000;

– максимальна кількість запитів до бази даних в годину 15000;

– максимальні кількість змін в базі даних за годину 3000;

– пропускна спроможність 1Гб/сек.

В процесі роботи з пробним періодом сайт надає широкий асортимент для використання, щоб зрозуміти чи підходить саме цей хостинг користувачу.

Під час роботи з хостингом в першу чергу потрібно створити домен вказавши ім’я домену, яке буде перевірено на наявність, також є можливість зареєструвати безкоштовний домен, після чого потрібно створити базу даних.

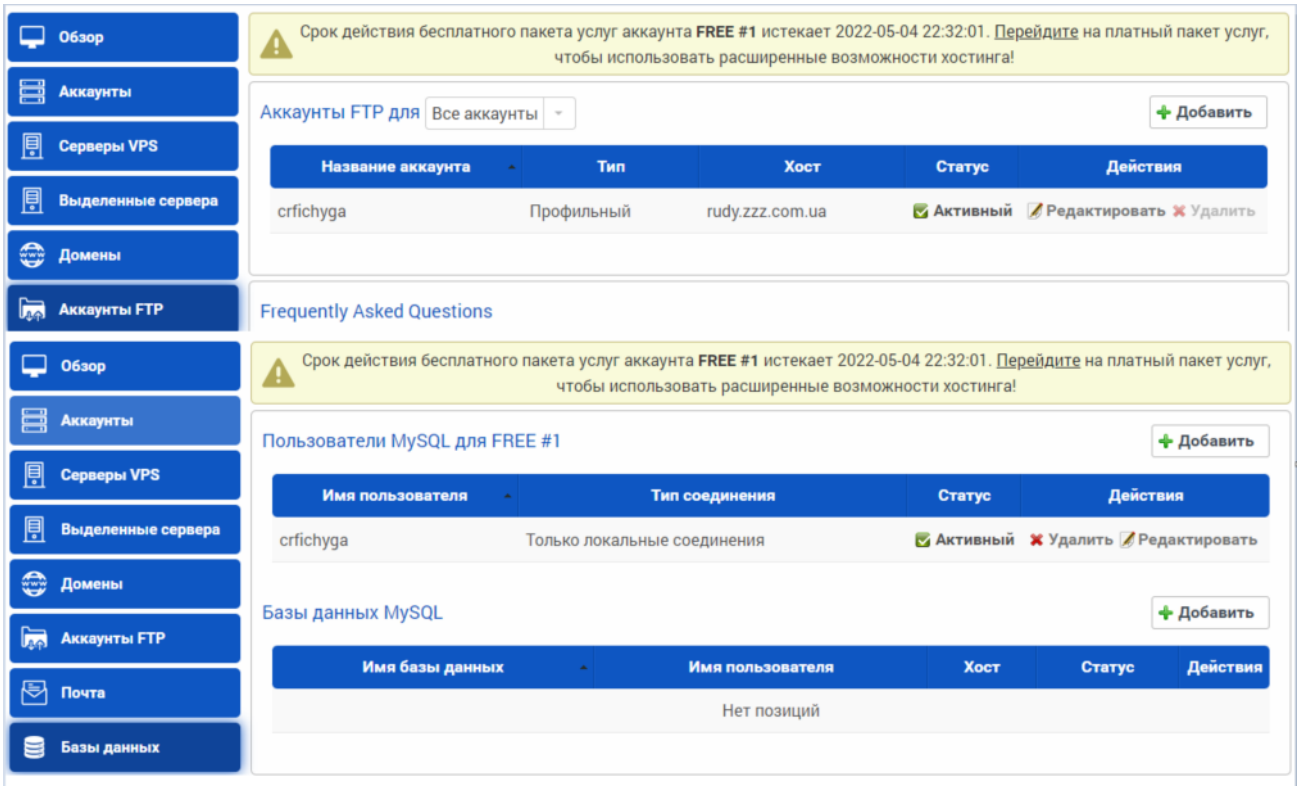


Рисунок 3.10 – Створена база даних і акаунт FTP

Після створення домену і бази даних потрібно завантажити усі файли на FTP-клієнт, можна завантажувати файли окремо, або архівом з розширенням .rar який автоматично розархівірується у папку.

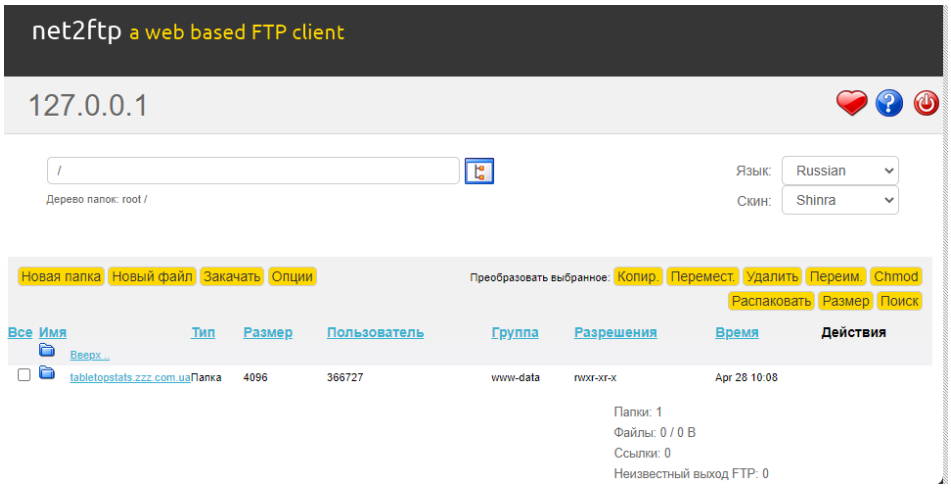


Рисунок 3.10 – FTP-клієнт

Даний розділ представляє опис розробленої інтернет-платформи для ведення статистики настільних ігор, розроблено базу даних за допомогою міграцій і було описано керівництво користувача і відбулось завантаження на хостинг.

4. ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

4.1 Вибір та обґрунтування методів тестування веб-додатку

Створення сайту – складний процес, в якому кожен учасник в команді виконує конкретні дії. Найважливішу роль в створені веб-ресурсу займає кінцевий етап розробки – тестування веб-додатка. Ця процедура найважливіша тому що, від якості тестування залежить життя проекта і його подальший супровід та розвиток. Багато розробників не приділяють достатньо уваги саме цьому етапу, в результаті чого виникають помилки, які можуть призвести до великих затрат часу, сил і грошей. Якщо сайт буде працювати не коректно, то це буде викликати дискомфорт у користувачів під час його використання і може призвести до зменшення кількості останніх, та репутації сайту.

Основні цілі тестування веб-ресурсу – перевірка чи виконує ресурс усі пред'явлені функціональні вимоги згідно з технічним завданням. Спеціаліст-тестувальник повинен створити штучні умови, які можуть виникнути в процесі роботи сайту. Якщо в процесі тестувальник знаходить помилку то передає звіт про знайдені помилки проектному менеджеру, який в свою чергу повинен усунути їх. Після чого сайт знову і знову тестують, поки сайт не буде ідеальним.

Існує досить багато способів та методів перевіряти та тестувати веб-ресурси для того, щоб досягти коректного функціонування сайта. При цьому спеціалісти створюють план дій.

В рамках дипломного проекту будуть задіяні такі етапи тестування як:

– функціональне тестування;
– тестування на зручність використання;
– перевірка на безпеку;
– UI Testing або ж тестування інтерфейсу.

Проведемо огляд кожного з етапів тестування і його призначення.

Функціональне тестування – перевірка, яка висвітлює неправильну роботу саме функціоналу додатка, сюди можна віднести:

– перевірка усіх головних функцій веб-ресурсу на коректність;

– коректність внутрішніх посилань;

– перевірки форм користувача, наприклад: додання коментаря, та інші;

– перевірка сторінок і полів для авторизації та реєстрації;

– перевірка додання, видалення та редагування даних користувачів.

Функціональне тестування має декілька видів: негативне тестування, тести еквівалентності, інтуїтивне тестування, дослідницьке тестування.

Негативне тестування – це тестування на збій. Проводиться для забезпечення стабільності системи суть полягає в тому, щоб перевірити чи користувач буде отримувати якісь помилки і де вони можуть виникнути.

Тести еквівалентності – це група тестів, які повинні в результаті давати однаковий результат позитивний чи негативний.

Дослідницьке тестування – це інтуїтивне тестування, котре розуміє під собою одночасне проектування і виконання тестів.

Інтуїтивне тестування – це тестування без якої-небудь підготовки, наприклад перевірка форм, чи є перевірка правильності вводу контактних даних.

Наступним етапом є юзабіліті тестування, або ж тестування на зручність використання, який призначений для перевірки рівня зручності веб-ресурсу для користувача – на скільки легко знайти потрібну інформацію, тобто комфортність при використані. Серед цілей такого тестування:

– зрозуміти чи зручна навігація;

– подивитись зі сторони користувача на сайт і яке враження створюється при використані;

– визначити, чи зрозумілий і зручний сайт;

– визначити займе на сайті.

Перше що перевіряється – всі сторінки, поля, форми, кнопки і чи зрозумілі вони саме для користувача, чи є доступ до головної сторінки зі всіх інших і навпаки. На другому етапі перевіряється контент на наявність граматичних помилок чи досить інформативний той чи інший контент. Третім і останнім

кроком є зручність використання, тобто на скільки веб-додаток, а саме його структура, зрозуміла і чи є зайві компоненті на сайту.

Третім етапом тестування є UI тестування, тобто тестування користувацького інтерфейсу. Його не слід плутати з юзабіліті тестування. Цілями цього етапу є:

– перевірка на відповідність усім стандартам графічних інтерфейсів;
– тестування з різними розширеннями екрану;
– кросбраузерність – сумісність з іншими браузера;
– тестування інтерфейсу на різних девайсах;
– тестування локалізації: точність переводу, довжина назв та інше.

І останній але не менш важливий тестування безпеки сайту – перевірка на вразливість сайту до різних атак.

4.2 Модульне тестування

В першу чергу потрібно скласти тестові сценарії, за допомогою яких ми зможемо протестувати функціональні можливості додатку. Тестові сценарії модульних тестів будуть наведені нижче в таблиці 4.1.

Таблиця 4.1 – Сценарії модульних тестів

Ідентифікатор	Модуль	Очікуваний результат
U-T-1	Реєстрація користувача	Створений новий запис в таблиці
U-T-2	Авторизація користувача	Перехід на головну сторінку

Продовження таблиці 4.1

1	2	3
U-T-3	Видалення користувача	Обліковий запис успішно видалено
U-T-4	Перевірка ролі	Користувач має відповідну роль
U-T-5	Перевірка логіну	Логін не існує
U-T-6	Перевірка пошти	Пошта не існує
U-T-7	Перевірка прав доступу	Перехід на сторінку адміністрації

Фрагмент коду який демонструє перевірку авторизації та реєстрації нового користувача:

```
public function test_authorise()
    {
        $this->post('/login',[
            'email' => 'fichyga@ukr.net',
            'password' => '1234567890'
        ]);

        $this->get('/');
        $this->assertTrue(true);
    }
    public function test_registration()
    {

        $this->post('/register',[
            'name' => 'TEEST',
            'email' => 'fichyga_test@ukr.net',
            'password' => '1234567890',
            'password_confirmation' => '1234567890'
        ]);

        $this->get('/');
        $this->assertTrue(true);
    }
```

Наступним кроком є системне тестування, в якому максимально будуть протестовані усі модулі системи. Системне тестування відноситься до методів тестування чорного ящика, що у свою чергу, не вимагає знань о внутрішній роботі системи.

Таблиця 4.2 Тестові сценарії системних тестів

Ідентифікатор	Модуль	Вихідні дані	Очікуваний результат
S-T-1	Додання в колекцію гри	1. Натиснути на кнопку «Пошук на BGG»; 2. Ввести в поле пошуку назву; 3. Натиснути на кнопку пошуку; 4. Обрати потрібну гру; 5. Обрати налаштування; 6. Натиснути кнопку «додати».	1. Відображення форми пошуку; 2. Відображається список схожих результатів; 3. Відкривається форма додання в колекцію; 4.Оновлення сторінки з відображенням гри.
S-T-2	Видалення з колекції	1. Натиснути на кнопку видалення; 2. Підтвердити видалення.	1. Відображення форми підтвердження; 2. Оновлення сторінки без відображення гри
S-T-3	Додання гравця	1. Натиснути на кнопку додання; 2. Ввести необхідну інформацію; 3. Натиснути кнопку збереження.	1. Відображається форма додання гравця; 2.Інформація оновлена і оновлення сторінки з відображенням нового гравця

Продовження тиблиці 4.2

S-T-4	Видалення гравця	1. Натиснути на кнопку видалення; 2. Підтвердити видалення.	1. Відображення форми підтвердження; 2. Оновлення сторінки без відображення гравця
S-T-5	Додання місця	1. Натиснути на кнопку додання; 2. Ввести необхідну інформацію; 3. Натиснути кнопку збереження.	1. Відображається форма додання місця; 2.Інформація оновлена і оновлення сторінки з відображенням нового місця
S-T-6	Видалення місця	1. Натиснути на кнопку видалення; 2. Підтвердити видалення.	1. Відображення форми підтвердження; 2. Оновлення сторінки без відображення місця
S-T-7	Додання партії	1. Перейти на сторінку гри; 2. Натиснути на кнопку додання партії; 3. Обрати гравців; 4. Обрати налаштування 5. Натиснути кнопку додання.	1. Відображення сторінки гри; 2. Відображення форми додання партії; 3.У формі з’явились обрані гравці; 4. Зміна налаштувань; 5. Оновлення сторінки.

4.3 Аналіз результатів тестування Інтернет-платформи

В результаті проведеного модульного тестування були отримані результати, які представлені на рисунку 4.1. Після виконання та перевірки, можна зробити висновок, що система працює вірно, а усі заявлені функції працюють.

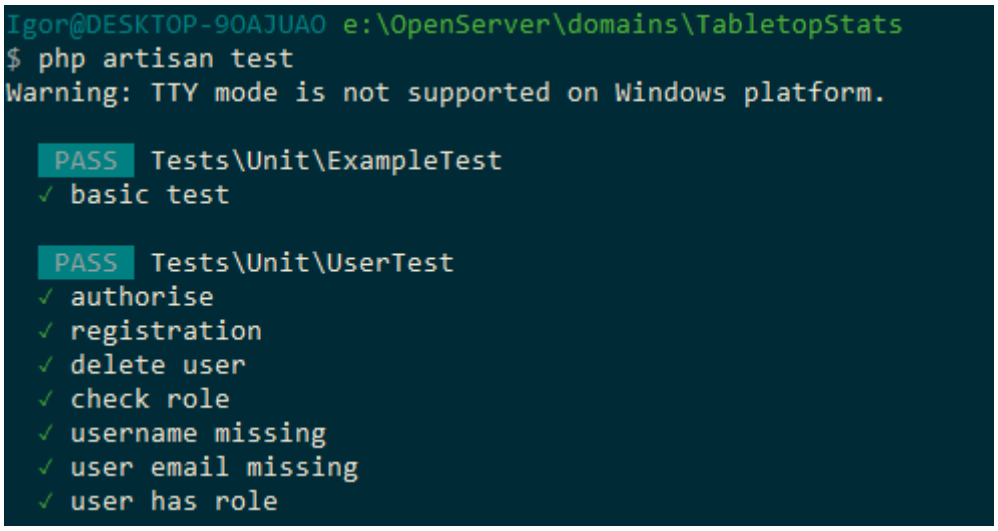


Рисунок 4.1 – результати тестування

Отже, в даному розділі було розглянуті наявні можливості для тестування програмного продукту, в результаті було проведено системне та модульне тестування для перевірки працездатності системи, усі тести пройдені успішно, а система готова до роботи.

ВИСНОВКИ

Темою дипломного проектування є Інтернет-платформа для ведення статистики настільних ігор. У сучасному світі йде заміна бумажної документації на електрону все швидше, до якого завжди можна отримати доступ і з будь-якої точки світу.

Прикладом такої заміни є Інтернет-платформа для статистики настільних ігор. Після аналізу ринку і наявного програмного забезпечення було вирішено, що наявність веб-додатків досить мала і конкуренції поки що не так багато, тому платформа розроблюється у вигляді сайту. Також в процесі аналізу рішень було складено ряд вимог, якими повинна володіти система та необхідний функціонал, щоб задовольнити потреби користувача.

В основі створення лежала мета розробити веб-додаток, який дозволить кожному бажаючому вести власну статистику ігор без зайвих витрат коштів і для економії часу. Сайт було розроблено в процесі виконання дипломного проекту. Тема і сайт є досить актуальною тому, що в сучасному світі популяризація настільних ігор йде в гору і кожен з часом приходить до того, що хочеться десь зберігати інформацію про партії, а веб-додатки не володіють усім необхідним для цього, мобільні ж платформи, які можуть скласти конкуренцію не є безкоштовними.

В результаті реалізації дипломного проекту було створено макет сайту для того, щоб зрозуміти, яким повинен бути зовнішній вигляд майбутньої Інтернет-платформи. Після реалізації запланованого функціоналу було розроблено додатковий функціонал і проведено тестування, які довели, що програмний продукт є зручним та придатним до використання.

Отже, результатом виконання дипломного проекту є Інтернет-платформа для ведення статистики настільних ігор, яка може використовуватись усіма бажаючими, з наявністю переваг та унікальних функцій, яких немає в аналогах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Уніфікована мова програмування UML [Електронний ресурс] / Портал знань. – Режим доступу до ресурсу: http://www.znannya.org/?view=uml
2. PHP Documentation. [Електронний ресурс] – Режим доступ до ресурсу: https://www.php.net/manual/ru/index.php
3. Laravel Documentation. [Електронний ресурс] – Режим доступ до ресурсу: https://laravel.com/docs/9.x
4. ЯК ЗРОБИТИ МАКЕТ САЙТУ, ЯКЩО ВИ НОВИЧОК:. [Електронний ресурс] / Koloro – Режим доступ до ресурсу: https://koloro.ua/ua/blog/dizain/kak-sdelat-maket-sayta-crazy-ideas.html
5. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert C. Martin – Publisher(s): Pearson, 2017. – 432 p
6. Bootstrap v5.1 Documentation. [Електронний ресурс] – Режим доступ до ресурсу: https://getbootstrap.com/docs/5.1/getting-started/introduction/
7. Sass Documentation. [Електронний ресурс] – Режим доступ до ресурсу: https://sass-lang.com/documentation
8. Laravel Documentation. [Електронний ресурс] – Режим доступ до ресурсу: https://laravel.com/docs/9.x

ДОДАТОК А

(Обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробка Інтернет-платформи для ведення статистики настільних ігор.

1 Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект», затверджене завідувачем кафедри інженерії програмного забезпечення.

Найменування розробки: Інтернет-платформа для ведення статистики настільних ігор.

2 Призначення розробки

Інтернет-платформа призначення для збереження інформації про зіграні партії з відображенням статистики по даним.

Користувачами сайту є люди, які займаються професійно чи на рівні любителя настільними іграми і хочуть зберігати інформацію про ці ігри.

Користувач може додати настільну гру, гравців, місця проведення ігор для подальшого додання партій у різних варіантах. На основі збережених даних користувач буде отримувати різноманітну інформацію.

Користувач може створювати власні завдання та використовувати готовий варіант у вигляді електронного постеру.

Користувач може створювати власний рейтинг за двома ознаками: по оцінкам, або за власним вподобанням.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Інтернет платформа повинна володіти наступними функціями:

Створення нового акаунту користувача, якщо такий ще не зареєстрований, в іншому випадку наявність можливості входу на сайт. Після переходу на головну сторінку користувачу потрібно відобразити усі базові функції для роботи з сайтом, помістивши їх в навігаційне та бокове меню.

Після чого користувач отримає змогу додавати та редагувати особисту інформацію чи інформацію для подальшого використання.

Після додання гравців користувач повинен отримати змогу створювати групи різних гравців, щоб зменшити кількість монотонної роботи.

На сторінці гри користувач повинен мати можливість змінювати налаштування до яких відноситься: середній час гри, щоб враховувати час, якщо таймер не був запущений; або користувач не виставив власноруч тривалість партії; місце проведення партії за замовчуванням, щоб кожен раз не обирати. Можливість створення власної медіатеки пісень для подальшого використання, можливість оцінювання гри за певними категоріями, що потрібно для відображення ігор в рейтингу по виставленим оцінкам.

В момент додавання партії, користувач повинен заповнювати усі обов’язкові поля: гравців; дату проведення; не обов’язкові поля, заповнення яких буде впливати на статистику. Серед неї: інформацію про роль гравця, визначення переможця, чи починав гравець першим і чи перша це його гра, місце гравця на початку партії, кількість очок гравця чи перепону, якщо гравці грають проти гри разом.

3.2 Вимоги до надійності

Розроблюване ПЗ повинно мати:

– можливість самовідновлення після збоїв, таких як відключення електроживлення тощо;

– підтримання захисту від несанкціонованого доступу;

– інформація, яка повинна зберігатись у шифрованому вигляді;

– валідація даних на стороні сервера і на стороні клієнта.

3.3 Вимоги до складу та параметрів технічних засобів

Оскільки програмний продукт представляє собою сайт, користувачу необхідний браузер Google Chrome або Opera і підключення до мережі Інтернет. Можна виділити наступні технічні вимоги для використання: процесор – 233 MHz; оперативна пам'ять – 1024 Мб; відеоадаптер і монітор; пристрої для взаємодії з додатком – мишка чи тачпад та клавіатура; оперативна система.

Для підтримки сайту необхідний сервер з наступними мінімальними характеристиками: дисковий простір – 1 Гб; передача даних – 5 Гб/місяць; FTP–акаунти – 2 шт.; пропускна спроможність – 1 Гб/сек.;

3.4 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати на різних браузерах.

3.5 Вимоги до транспортування та зберігання

Програма, та її документація, поставляються у цифровому вигляді. Умови експлуатації програмного забезпечення збігаються з умовами експлуатації серверу, на якому буде розміщене програмне забезпечення.

3.6 Спеціальні вимоги

Програма повинна мати дружній інтерфейс, розрахований на користувача середньої кваліфікації (з точки зору комп'ютерної грамотності). Мова програмування визначається вибором виконавця.

4. Вимоги до програмної документації

В ході розробки програми повинні бути підготовлені: текст програми, опис програми, технічне завдання, короткий посібник користувачу.

5. Стадії та етапи розробки

Стадії та етапи розробки програмного забезпечення для реалізації Інтернет-платформи для ведення статистики настільних ігор подані у таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.22 – 31.01.22	Обґрунтування необхідності розробки програми	Характеристика та опис ПЗ; підстави для розробки і призначення ПЗ; функціональні вимоги до розроблюваної системи; порядок контролю і приймання ПЗ.
Ескізний проект 01.02.22 – 14.02.22	Розробка ескізного проекту	Визначення структури вхідних і вихідних даних; попередній вибір технологій; визначення потрібних алгоритмів.

Кінець таблиці А.1

Технічний проект 15.02.22 – 28.02.22	Розробка технічного проекту	Затвердження структури вхідних і вихідних даних; розробка алгоритмів; розробка структури програми; вибір технологій.
Робочий проект 01.03.22 – 10.04.22	Розробка програмного забезпечення	Реалізація програмного забезпечення; виправлення помилок; тестування.
Розробка програмної документації 11.04.22 – 20.04.22	Розробка документації для програмного забезпечення	Розробка документації користувача, інструкції по запуску та зупинці ПЗ
Тестування системи 21.04.22 – 30.04.22	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Здача проекту	Здача програми замовнику	Передача вихідного коду та документації замовнику

6. Порядок контролю та приймання

Контроль і приймання розробки здійснюються кінцевими користувачами та на етапі тестування, потрібно зробити системне та модульне тестування.

ДОДАТОК Б

(Обов’язковий)

КОД ПРОГРАМИ

```
class AddPlays
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    public $info;
    public $friend_id;

    public function __construct($info,$friend_id)
    {
        $this->info = $info;
        $this->friend_id = $friend_id;
    }

}
class BGH extends Model
{
    use HasFactory;

    public function updateBGH($request)
    {
        $BGH = BGH::where('user_id', Auth::id())->first();

        if($BGH === null){
            BGH::create([
                'user_id' => Auth::id(),
                'skills' => $request->skill,
            ]);
        }
        else{
            if(strpos($BGH->skills, $request->skill) === false){
                $BGH->skills = $BGH->skills . $request->skill . "," ;
                $BGH->save();
            }
            else{
                $removedSkill = str_replace($request->skill . "," , '', $BGH-
>skills);
                $BGH->skills = $removedSkill;
                $BGH->save();
            }
        }
    }

}
class UserTop extends Model
{
    use HasFactory;

    protected $fillable = ['game_id','user_id'];

    public function Game()
    {
       return $this->belongsTo(Game::class);
    }

    public function saveSubjectivelyTop($game_id)
    {
        $tag = UserTop::firstOrCreate(['game_id'=>$game_id,'user_id' => Auth::id()]);

    }

    public function getRattedGames()
    {
```

```
        $user_top =  UserTop::where('user_id',Auth::id())->get();
        $games = [];
        for ($i=0; $i < count($user_top); $i++) {
            array_push($games,[
                $user_top[$i]->game->name,
                $user_top[$i]->game->image,
                $user_top[$i]->game->id
            ]);
        }
        return $games;
    }

    public function removeSubjectivelyTop($game_id)
    {
        UserTop::where('user_id',Auth::id())->where('game_id',$game_id)->first()-
>delete();
    }

    public function changeSubjectivelyTop($old_game_id, $new_game_id)
    {
        $first_game =  UserTop::where('user_id',Auth::id())-
>where('game_id',$old_game_id)->first();
        $check_second_game = UserTop::where('user_id',Auth::id())-
>where('game_id',$new_game_id)->first();
        if($first_game !== null && $check_second_game !== null){
                $first_game->game_id = $new_game_id;
                $first_game->save();
                $check_second_game->game_id = $old_game_id;
                $check_second_game->save();
            }
        else{
                $first_game->game_id = $new_game_id;
                $first_game->save();
        }
    }
}
class AdminController extends Controller
{

    public function homePage()
    {
        return view('admin.home',[

        ]);
    }

    public function Roles()
    {
        $users = User::all();
        $roles = Role::all();
        return view('admin.super-admin',[
            'users'=>$users,
            'roles'=>$roles
        ]);
    }

    public function ChangeRoles(Request $request, $id)
    {
        $user = User::where('id',$id)->first();
        $roles = Role::all();
        foreach ($roles as $role) {
            $user->removeRole($role);
        }
        $user->assignRole($request->input('role'));
```

```
        return back();
    }


    public function GlobalSettings($id)
    {
        $game = Game::where('id',$id)->first();
        $genres = Genre::all();
        $topics = Topic::all();

        return view('admin.game-settings',[
           'game'=>$game,
           'genres'=>$genres,
           'topics'=>$topics
        ]);
    }

    public function GameList()
    {
        $games = Game::paginate(50);

        return view('admin.games',[
            'games'=>$games
        ]);
    }


    public function PostPage($id)
    {
       $post = Post::where('id',$id)->first();
       return view('admin.post',[
           'post'=>$post
       ]);
    }
}
class ChallengeController extends Controller
{

    public function UserChallenges()
    {
        $challenges = Challenge::where('user_id',Auth::id())->where('finished',null)-
>get();
        $icons = ChallengeIcon::all();
        return view("functions.challenges",['challenges'=>$challenges,
'icons'=>$icons]);
    }

    public function AddChallenges(Request $request)
    {
        $challenge = new Challenge();
        $challenge->createchallenge($request);
        return back();
    }

    public function DeleteChallenge(Request $request)
    {
        Challenge::where('id',$request->id)->delete();
        return response()->json([]);
    }

    public function FinishChallenge(Request $request)
    {
        $challenge = new Challenge();
        $challenge->finishChallenge($request);
```

```
        return response()->json([]);
    }

    public function AddTaskToChallenge(Request $request)
    {
        $task = new Task();
        $task->createTask($request);
        return response()->json(['task'=>$task]);
    }

    public function DeleteTask(Request $request)
    {
        Task::where('id',$request->id)->delete();
        return response()->json([]);

    }

    public function FinishTask(Request $request)
    {
        $task = new Task();
        $task->finishTask($request);
        return response()->json([]);
    }


}
class CollectionController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $collection_list = new UserCollection();
        return view('functions.collection',['collection_list'=>$collection_list-
>getCollectionList()]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
         $game = new Game();
         $saved_id = $game->AddNewGame($request);
         $new_game = new UserCollection();
         $new_game->AddToCollection($saved_id,$request);
         return back();
    }
```

```
    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        UserCollection::where("game_id",$id)->where("user_id",Auth::id())->first()-
>delete();
        return  redirect()->action([CollectionController::class, 'index']);
    }

    public function GetGameBGG(Request $request)
    {
        $find_game = new Game();
        return response()->json(['games_list'=>$find_game->BGGFind($request->name)]);
    }
}

class CommentController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
```

```
    //
}

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    //
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $comment = new Comment();
    $comment->saveComment($request);
    return back();
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    //
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    $comment = new Comment();
    $comment->changeComment($request, $id);
    return back();
}

/**
```

```
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        Comment::where('id',$id)->first()->delete();
        return back();
    }

    public function subComment(Request $request, $post_id,$comment_id)
    {
        $comment = new Comment();
        $comment->saveSubComment($request, $post_id, $comment_id);
        return back();
    }
}
class GenreController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $genres = Genre::all();
        $games = Game::paginate(20);
        return view('admin.genres',[
            'genres'=>$genres,
            'games'=>$games
        ]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {

        $genre = new Genre();
        $genre->AddGenre($request);
        return back();
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
```

```
     */
    public function show($id)
    {
        //
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        $genre = new Genre();
        $genre->RenameGenre($request,$id);
        return back();
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $genre = new Genre();
        $genre->RemoveGenre($id);
        return back();
    }

    public function AddGameGenre(Request $request,$id)
    {
        $genre = new Genre();
        $genre->AddGenreToGame($request,$id);
        return back();
    }
}
class GroupController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
```

```
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $group = new Group();
        $group->addGroup($request);
        return back();
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        $group = Group::where('id',$id)->first();
        $group->name = $request->name;
        $group->save();
        return back();
    }


    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
```

```
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $group = new Group();
        $group->deleteGroup($id);
        return back();
    }

    public function GetGroupPlayers(Request $request)
    {
        if($request->id === "0"){
            $group = new Group;
            return response()->json(['players'=>$group->getLastPlayersGroup()]);
        }
        $players = PlayerGroup::where('group_id',$request->id)->pluck('player_id');
        return response()->json(['players'=>$players]);
    }

    public function ChangeGroupPlayers(Request $request)
    {
        $players = new PlayerGroup();
        return response()->json(['players'=>$players->ChangeGroupPlayers($request)]);
    }


}
class Game extends Model
{
    use HasFactory;

    public function checkGenre($game_id,$genre_id)
    {
      return GameGenre::where('game_id',$game_id)->where('genre_id',$genre_id)-
>select('game_id')->first();
    }

    public function BGGFind($game_name)
    {
        $games = [];
        $name = str_replace(" ","%20", $game_name);
        $url = "https://boardgamegeek.com/search/boardgame/page/$page?q=$name";
        $client = new Client();
        $crawler = $client->request('GET', $url);
        $crawler->filter('.primary')->each(function($node) use (&$game_name,
&$games){
                $game =
                [
                    "name" => $node->text(),
                    "year" => $node->ancestors()->filter('.dull')->text(),
                    "url" => $node->attr('href'),
                    "image" => $node->ancestors()->ancestors()->ancestors()-
>filter('img')->getNode(0)->getAttribute('src'),
                ];
                array_push($games, $game);
        });
        return $games;
    }


    public function AddNewGame($request)
    {
            $game = Game::where('name',$request->name)->first();
               if($game === null)
```

```
            {
                $game = new Game();
                $game->name = str_replace("&amp;","and",$request->name);
                $game->image = $request->image;
                $game->year      = $request->year;
                $game->url       = $request->url;
                $game->save();
            }
        return $game->id;
    }


    public function GetAllStats($id)
    {
        $combined_games = CombinedGames::where('game_first',$id)-
>pluck('game_second')->toArray();
        array_push($combined_games,intval($id));

        $match_sett = MatchSetting::where('user_id',Auth::id())-
>whereIn('game_id',$combined_games)->pluck('id');
        $game_type = UserCollection::whereIn('id',$combined_games)->pluck('type')-
>first();
        $default_stats = [];
        $coop_stats = [];
        $team_stats = [];
        $game_info = $this->GameInfo($combined_games , $match_sett);
        $point_type = UserCollection::whereIn('id',$combined_games)-
>pluck('points_directed')->first();
        $role_statistic = $this->RoleStatistic($match_sett,$game_type);

        $mouth_statistic=DB::table('match_settings')
            ->where('date', '>=', Carbon::now()->subMonth(6)->toDateTimeString())
            ->whereIn('game_id',$combined_games)
            ->where('user_id',Auth::id())
            ->select(DB::raw('count(id) as plays'),DB::raw("DATE_FORMAT(date, '%Y-
%m') as dates"))
            ->groupBy('dates')
            ->orderBy('dates','desc')
            ->get();

       $players_statistics =$this->PlayersInfo($match_sett);

       $plays = MatchSetting::where('user_id',Auth::id())-
>whereIn('game_id',$combined_games)->count();
       $coop = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","coop")->count();
       $team = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","team")->count();
       $default = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","default")->count();
       $check_point =  Matches::whereIn('match_settings_id',$match_sett)-
>where('score','!=', null)->count() == 0 ? false : true ;
       $players_stats = [];

       $statistic_score = [];
       $wins = $this->GetWinsCount($match_sett);
       $loses = $this->GetLosesCount($match_sett);
       $user_players =  Player::where('user_id',Auth::id())->pluck('id');
       $statistic_score =
       [
           "plays" => $plays,
           "percent" => $wins === 0 ? 0 : floor($wins / ($coop+$team+$default) *
100),
           "wins" =>  $wins,
```

```
            "loses" => $loses,
        ];

         if($coop !== 0)
         {

             $circumstances = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","coop")->distinct('circumstances')->pluck('circumstances');
             $circumstances_stats = [];


             for ($i=0; $i < count($circumstances); $i++) {
                 $name =  $circumstances[$i];
                 $players_wins = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","coop")->distinct('match_settings_id')-
>where('circumstances',$circumstances[$i])->where('victory',true)->count();
                 $players_plays = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","coop")->distinct('match_settings_id')-
>where('circumstances',$circumstances[$i])->count();
                 $players_loses = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","coop")->distinct('match_settings_id')-
>where('circumstances',$circumstances[$i])->where('victory',false)->count();


                  if($players_plays >= 1)
                  {
                     $player_info =
                     [
                         "circumstances"=>  $name,
                         "loses" => $players_loses,
                         "wins" => $players_wins,
                         "plays" => $players_plays,
                         "percent" => $players_wins === 0 ? 0 : floor($players_wins /
$players_plays * 100),

                     ];
                     array_push($circumstances_stats,$player_info);
                 }
             }
             array_multisort(array_column($circumstances_stats, 'percent'), SORT_DESC,
$circumstances_stats);
             $coop_stats = [$circumstances_stats];

         }
         if($team !== 0)
         {

         }
     if($default !== 0)
     {

             $best = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","default")->max('score');
             $average = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","default")->avg('score');
             $lowest = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","default")->min('score');
             $lowest_win = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","default")->where('victory',true)->min('score');
             $lowest_lose = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","default")->where('victory',false)->max('score');

             $default_score =
             [
```

```
                "best" => $best,
                "average" => round($average,1),
                "lowest" =>  $lowest,
                "lowest_win" => $lowest_win,
                "highest_lose" => $lowest_lose,

            ];
            $average = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","default")->avg('score');


            $role_vs_role_statistic = $this-
>RoleVsRoleStatistic($match_sett,$game_type);
            $player_vs_player_statistic = $this-
>PlayerVsPlayerStatistic($match_sett,$game_type);
            $default_stats = [
            $default_score,
            $role_vs_role_statistic,
            $player_vs_player_statistic];

      }

    for ($i=0; $i < count($user_players); $i++) {
        $player_id = Player::where('id',$user_players[$i])->first()->id;
        $name = Player::where('id',$user_players[$i])->first()->name;
        $players_wins = $this->GetPlayerWins($match_sett,$user_players[$i]);
        $players_plays = $this->GetPlayerPlays($match_sett,$user_players[$i]);
        $players_score = $this->GetPlayerScore($match_sett,$user_players[$i]);
        $players_best = $this->GetPlayerBest($match_sett,$user_players[$i]);
        $players_lowest = $this->GetPlayerLowest($match_sett,$user_players[$i]);

        if($players_plays >= 1)
        {
            $player_info =
            [
                "id"=>$player_id,
                "name"=>  $name,
                "wins" => $players_wins,
                "percent" => $players_wins === 0 ? 0 : floor($players_wins /
$players_plays * 100),
                "plays" => $players_plays,
                "average" => round($players_score / $players_plays) ,
                "best" => $players_best,
                "lowest" => $players_lowest,
            ];
            array_push($players_stats,$player_info);
        }
    }

    array_multisort(array_column($players_stats, 'percent'), SORT_DESC,
$players_stats);

    $player_role_statistic = $this->PlayerRoleStatistic($match_sett,$game_type);

        return [
            $game_info,
            $point_type,
            $players_statistics,
            $role_statistic,
            $mouth_statistic,
            $coop_stats,
            $team_stats,
            $default_stats,
            $check_point,
```

```
            $statistic_score,
            $players_stats,
            $player_role_statistic
        ];
    }

    public function PlayersInfo($match_sett)
    {
        $players_array = [];
        for ($i=0; $i < count($match_sett); $i++) {
            $players_count = Matches::where('match_settings_id',$match_sett[$i])-
>count();
            if($players_count !== 0)  array_push($players_array,$players_count);
        }
        return array_count_values($players_array);
    }

    public function GameInfo($id,$match_sett)
    {
        $last_play = MatchSetting::whereIn('game_id',$id)->max('date');
        $first_play = MatchSetting::whereIn('game_id',$id)->min('date');
        $place_count = MatchSetting::whereIn('game_id',$id)->distinct('place_id')-
>pluck('place_id')->count();

        $players_count =  Matches::whereIn('match_settings_id',$match_sett)-
>distinct('player_id')->pluck('player_id')->count();


        $game_info =
        [
            "last_play" => $last_play,
            "first_play" => $first_play,
            "place_count" => $place_count,
            "players_count" => $players_count,
        ];

        return $game_info;
    }

    public function getHomeInfo()
    {
        $game_info = [];

        $base_count = UserCollection::where("user_id",Auth::id())-
>where("available",1)->where("base",1)->count();
        $expansion_count = UserCollection::where("user_id",Auth::id())-
>where("available",1)->where("base",0)->count();
        $unowned_count = UserCollection::where("user_id",Auth::id())-
>where("available",0)->count();

        $dates = MatchSetting::where('user_id',Auth::id())->where('include',1)-
>pluck('date');
        $day = array(__('days.monday') , __('days.tuesday') , __('days.wednesday') ,
__('days.thursday') , __('days.friday') , __('days.saturday') , __('days.sunday') );
        $days = [];
        foreach($dates as $date)
        {
            array_push($days,$day[date("w",strtotime($date))]);
        }

      $last_play = MatchSetting::where('user_id',Auth::id())->where('include',1)-
>orderBy('date','desc')->orderBy('created_at','desc')->first();
      if($last_play !== null){$last_play->date = date_diff(new DateTime(), new
DateTime($last_play->date))->days;
```

```
      if($last_play->date  == 0)$last_play->date = __('days.today') ;
      else if($last_play->date  == 1) $last_play->date = __('days.yesterday');
        else $last_play->date .=  __('home.last_play_date');}
        $all_time_stats=DB::table('match_settings')
            ->where('user_id',Auth::id())
            ->select(DB::raw('count(id) as plays'),DB::raw("DATE_FORMAT(date, '%Y-
%m') as dates"))
            ->groupBy('dates')
            ->orderBy('dates','desc')
            ->get()->toArray();

        $year_stats=DB::table('match_settings')
            ->whereBetween('date', [
                Carbon::now()->startOfYear(),
                Carbon::now()->endOfYear(),
            ])
            ->where('user_id',Auth::id())
            ->select(DB::raw('count(id) as plays'),DB::raw("DATE_FORMAT(date, '%Y-
%m') as dates"))
            ->groupBy('dates')
            ->orderBy('dates','desc')
            ->get()->toArray();



        array_push($game_info,$last_play);
        array_push($game_info,$this->getMonthStats());
        array_push($game_info,$this->getYearStats());
        array_push($game_info,$this->getAllTimeStats());
        array_push($game_info,$expansion_count);
        array_push($game_info,$base_count);
        array_push($game_info,$unowned_count);
        array_push($game_info,$all_time_stats);
        array_push($game_info,$year_stats);
        array_push($game_info,array_count_values($days));
        array_push($game_info,$this->AllPlayersInfo());
        return $game_info;
    }
    public function AllPlayersInfo()
    {
        $players_array = [];
        $match_sett = MatchSetting::where('user_id',Auth::id())->where('include',1)-
>pluck('id');

        for ($i=0; $i < count($match_sett); $i++) {
            $def = Matches::where('match_settings_id',$match_sett[$i])->count();
            array_push($players_array,$def);
        }
        $players_array = array_filter($players_array);
        return array_count_values($players_array);
    }
    public function getMonthStats()
    {
        $last_mouth_plays_stats = [];
        $plays = MatchSetting::whereBetween('date', [
                Carbon::now()->startOfMonth(),
                Carbon::now()->endOfMonth(),
            ])->where('include',true)->where('user_id',Auth::id());
        $mouth_plays_statistic = $plays->count();
        $mouth_game_statistic = $plays->distinct("game_id")->count();
        $mouth_place_statistic = $plays->distinct("place_id")->count();

        $settings_last_mouth = $plays->pluck("id");
```

```
        $month = Matches::whereIn('match_settings_id',$settings_last_mouth)-
>distinct('player_id')->pluck("player_id")->toArray();
        $players_mouth_count  = count(array_unique($month));

        $mouth_game_plays =  MatchSetting::select('game_id', DB::raw('COUNT(game_id)
as count'))->groupBy('game_id')->where('user_id',Auth::id())->where('include',true)-
>where('date', '>=', Carbon::now()->subMonth(1)->toDateTimeString())-
>orderBy('count', 'desc')->limit(5)->get();

        array_push($last_mouth_plays_stats,$mouth_plays_statistic);
        array_push($last_mouth_plays_stats,$players_mouth_count);
        array_push($last_mouth_plays_stats,$mouth_place_statistic);
        array_push($last_mouth_plays_stats,$mouth_game_statistic);
        array_push($last_mouth_plays_stats,$mouth_game_plays);

        return  $last_mouth_plays_stats;
    }
    public function getYearStats()
    {
        $last_mouth_plays_stats = [];
        $plays = MatchSetting::whereBetween('date', [
            Carbon::now()->startOfYear(),
            Carbon::now()->endOfYear(),
        ])->where('user_id',Auth::id())->where('include',true);

        $mouth_plays_statistic = $plays->count();
        $mouth_game_statistic = $plays->distinct("game_id")->count();
        $mouth_place_statistic = $plays->distinct("place_id")->count();

        $settings_last_mouth = $plays->pluck("id");
        $year = Matches::whereIn('match_settings_id',$settings_last_mouth)-
>distinct('player_id')->pluck("player_id")->toArray();
        $players_mouth_count  = count(array_unique($year));

        $mouth_game_plays =  MatchSetting::select('game_id', DB::raw('COUNT(game_id)
as count'))->groupBy('game_id')->where('include',1)->where('date', '>=',
Carbon::now()->subMonth(12)->toDateTimeString())->orderBy('count', 'desc')->limit(5)-
>get();


        array_push($last_mouth_plays_stats,$mouth_plays_statistic);
        array_push($last_mouth_plays_stats,$players_mouth_count);
        array_push($last_mouth_plays_stats,$mouth_place_statistic);
        array_push($last_mouth_plays_stats,$mouth_game_statistic);
        array_push($last_mouth_plays_stats,$mouth_game_plays);

        return  $last_mouth_plays_stats;
    }
    public function getAllTimeStats()
    {
        $last_mouth_plays_stats = [];

        $plays = MatchSetting::where('user_id',Auth::id())->where('include',true);

        $mouth_plays_statistic = $plays->count();
        $mouth_game_statistic = $plays->distinct("game_id")->count();
        $mouth_place_statistic = $plays->distinct("place_id")->count();

        $settings_last_mouth = $plays->pluck("id");
        $allTime = Matches::whereIn('match_settings_id',$settings_last_mouth)-
>distinct('player_id')->pluck("player_id")->toArray();
        $players_mouth_count  = count(array_unique($allTime));
```

```
        $mouth_game_plays =  MatchSetting::select('game_id', DB::raw('COUNT(game_id)
as count'))->where('include',1)->groupBy('game_id')->orderBy('count', 'desc')-
>limit(5)->get();


        array_push($last_mouth_plays_stats,$mouth_plays_statistic);
        array_push($last_mouth_plays_stats,$players_mouth_count);
        array_push($last_mouth_plays_stats,$mouth_place_statistic);
        array_push($last_mouth_plays_stats,$mouth_game_statistic);
        array_push($last_mouth_plays_stats,$mouth_game_plays);

        return  $last_mouth_plays_stats;
    }
    public function RoleStatistic($match_sett)
    {
        $role_statistic = [];
        $plays = Matches::whereIn('match_settings_id',$match_sett)-
>where('role','!=', null)->select('role','match_settings_id','victory')-
>distinct('role')->get()->toArray();


        for ($i=0; $i < count($plays); $i++) {
                $list = explode(',',$plays[$i]['role']);

                for ($j=0; $j < count($list); $j++) {

            $check = in_array( $list[$j] , array_column($role_statistic, 'role'));

            if($check){
                  $searched_index = array_search($list[$j] ,
array_column($role_statistic, 'role'));
                  $plays[$i]['victory'] === 1 ?
$role_statistic[$searched_index]["wins"] ++:
$role_statistic[$searched_index]["loses"]++ ;


            }
            else{
              array_push(
                  $role_statistic,
                  [
                      "role"=>$list[$j],
                      "wins"=>$plays[$i]['victory'] === 0 ? 0 : 1,
                      "loses"=> $plays[$i]['victory'] === 1 ? 0 : 1,
                      "percent" => 0,
                ]);
                }
            }
        }
        for ($i=0; $i < count($role_statistic); $i++) {
            $role_statistic[$i]['percent'] = $role_statistic[$i]["wins"] === 0 ? 0 :
floor($role_statistic[$i]["wins"] /
($role_statistic[$i]["wins"]+$role_statistic[$i]["loses"]) * 100);
        }

        array_multisort(array_column($role_statistic, 'percent'), SORT_DESC,
$role_statistic);
        return $role_statistic;
    }
    public function RoleVsRoleStatistic($match_sett,$type)
    {

        $role_statistic = [];
        $statistic = [];
```

```
        $role = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","default")->where('role','!=', null)->distinct('role')->pluck('role');

        for ($i=0; $i < count($role); $i++) {
        $role_battles = Matches::where('role',$role[$i])->where("type","default")-
>distinct('match_settings_id')->pluck('match_settings_id');

        $opponent = Matches::whereIn('match_settings_id',$role_battles)
            ->where("type","default")
            ->where('role', '!=' , $role[$i])
            ->select('role','victory')
            ->get('role','victory')
            ->toArray();

            foreach ($opponent as $result) {

                $result_of_battle = [
                    "role" => $role[$i],
                    "opponent" => $result['role'],
                    "wins" =>  $result['victory'] == 1 ? 0 : 1,
                    "count" => 1,
                    "percent" => 0,
                ];

                array_push($role_statistic,$result_of_battle);

            }

        }

        for ($i=0; $i < count($role_statistic); $i++) {
         for ($j=count($role_statistic)-1; $j > $i; $j--) {

            if($role_statistic[$i]['role'] == $role_statistic[$j]['role'] &&
             $role_statistic[$i]['opponent'] == $role_statistic[$j]['opponent'])
             {
                $role_statistic[$i]['wins'] += $role_statistic[$j]['wins'];
                $role_statistic[$i]['count'] += $role_statistic[$j]['count'];
                array_splice($role_statistic,$j,1);
             }
         }
        }

        for ($i=0; $i < count($role_statistic); $i++) {
            $role_statistic[$i]['percent'] = $role_statistic[$i]["wins"] === 0 ? 0 :
floor($role_statistic[$i]["wins"] / ($role_statistic[$i]["count"]) * 100);
        }
         return $role_statistic;
    }
    public function PlayerRoleStatistic($match_sett,$type)
    {

        $plays = Matches::whereIn('match_settings_id',$match_sett)
                ->orderBy('player_id')
                ->groupBy('role', 'player_id')
                ->select('role','victory',
                'player_id'
                ,DB::raw('COUNT(*) as plays_count'))
                ->selectRaw('COUNT(case when victory = 1 then 0 else null end) as
wins_count')
                ->get()->toArray();
        return $plays;
    }
    public function PlayerVsPlayerStatistic($match_sett,$type)
```

```
    {

        $player_statistic = [];
        $player = Matches::whereIn('match_settings_id',$match_sett)-
>where("type","default")->distinct('player_id')->pluck('player_id');
        for ($i=0; $i < count($player); $i++) {
        $player_battles = Matches::where('player_id',$player[$i])-
>where("type","default")->whereIn('match_settings_id',$match_sett)-
>distinct('match_settings_id')->pluck('match_settings_id');

        $opponent = Matches::whereIn('match_settings_id',$player_battles)
            ->where("type","default")
            ->select('player_id','victory')
            ->get()
            ->toArray();

            foreach ($opponent as $result) {

                if($player[$i] !== $result['player_id'])
                {
                    $result_of_battle = [
                    "player" =>  Player::where('id',$player[$i])->value('name'),
                    "opponent" => Player::where('id',$result['player_id'])-
>value('name'),
                    "wins" =>  $result['victory'] == 1 ? 0 : 1,
                    "count" => 1,
                    "percent" => 0,
                ];

                array_push($player_statistic,$result_of_battle);

                }
            }

        }

        for ($i=0; $i < count($player_statistic); $i++) {
         for ($j=count($player_statistic)-1; $j > $i; $j--) {

            if($player_statistic[$i]['player'] == $player_statistic[$j]['player'] &&
             $player_statistic[$i]['opponent'] == $player_statistic[$j]['opponent'])
             {
                $player_statistic[$i]['wins'] += $player_statistic[$j]['wins'];
                $player_statistic[$i]['count'] += $player_statistic[$j]['count'];
                array_splice($player_statistic,$j,1);
             }
         }
        }

        for ($i=0; $i < count($player_statistic); $i++) {
            $player_statistic[$i]['percent'] = $player_statistic[$i]["wins"] === 0 ?
0 : floor($player_statistic[$i]["wins"] / ($player_statistic[$i]["count"]) * 100);
        }
         return $player_statistic;
    }
    public function GetPlayerWins($match_sett , $id)
    {

        return Matches::whereIn('match_settings_id',$match_sett)-
>where('player_id',$id)->where('victory',true)->count();
    }
    public function GetPlayerPlays($match_sett , $id)
    {
```

```
        return Matches::whereIn('match_settings_id',$match_sett)-
>where('player_id',$id)->count();
    }
    public function GetPlayerScore($match_sett , $id)
    {
        return Matches::whereIn('match_settings_id',$match_sett)-
>where('player_id',$id)->sum("score");
    }

    public function GetPlayerBest($match_sett , $id)
    {
     return Matches::whereIn('match_settings_id',$match_sett)-
>where('player_id',$id)->max("score");
    }
    public function GetPlayerLowest($match_sett , $id)
    {
     return Matches::whereIn('match_settings_id',$match_sett)-
>where('player_id',$id)->min("score");
    }
    public function GetWinsCount($match_sett)
    {
        return Matches::whereIn('match_settings_id',$match_sett)-
>distinct('match_settings_id')->where('victory',true)->count();
    }
    public function GetLosesCount($match_sett)
    {
        return Matches::whereIn('match_settings_id',$match_sett)-
>distinct('match_settings_id')->where('victory',false)->count();
    }



    public function SuccessMatchTo($friend_match_id)
    {


        $friend_match = MatchSetting::where('id',$friend_match_id)->first();

        $check_collection = UserCollection::where("user_id",Auth::id())-
>where('game_id',$friend_match->game_id)->first();
        $friend_settings = UserCollection::where("user_id",$friend_match->user_id)-
>where('game_id',$friend_match->game_id)->first();
        if($check_collection === null)
        {

            $collection = new UserCollection();
            $collection->user_id = Auth::id();
            $collection->game_id = $friend_settings->game_id;
            $collection->type = $friend_settings->type;
            $collection->points_directed = $friend_settings->points_directed;
            $collection->base  = $friend_settings->base;
            $collection->available = 0;
            $collection->save();
        }

        $place_avaible = Place::where('user_id',Auth::id())-
>where('name',$friend_match->place->name . '('. $friend_match->user->name . ')')-
>first();

        if($place_avaible === null)
        {
            $place = new Place();
            $place->user_id = Auth::id();
```

```
            $place->name =  $friend_match->place->name . '('. $friend_match->user-
>name . ')';
            $place->save();
        }
        else{
            $place =  $place_avaible->id;
        }

        $settings = new MatchSetting();
        $settings->user_id = Auth::id();
        $settings->game_id = $friend_match->game_id;
        $settings->place_id =  $friend_match->place->id;
        $settings->comment = $friend_match->comment;
        $settings->date = $friend_match->date;
        $settings->rounds = $friend_match->rounds;
        $settings->timer = $friend_match->timer;
        $settings->include = $friend_match->include;
        $settings->save();

            foreach($friend_match->matches as $play)
            {

                $new_play = new Matches();
                $new_play->match_settings_id = $settings->id;

                $player_f = Player::where('user_id',Auth::id())
                ->where('name',$play->player->name)
                ->where('friend_id',$play->player->friend_id)
                ->where('image','friend_friend.jpg')
                ->first();

                if(Player::where('user_id',Auth::id())->where('friend_id',$play-
>player->friend_id)->first() !== null)
                {
                    $new_play->player_id = Player::where('user_id',Auth::id())-
>where('friend_id',$play->player->friend_id)->first()->id;
                }
                else if($player_f !== null)
                {
                    $new_play->player_id = $player_f->id;
                }
                 else{
                    $player = new Player();
                    $player->user_id = Auth::id();
                    $player->friend_id = $play->player->friend_id;
                    $player->name = $play->player->name;
                    $player->image = 'friend_friend.jpg';
                    $player->save();
                    $new_play->player_id = $player->id;

                 }
                $new_play->victory = $play->victory;
                $new_play->role = $play->role;
                $new_play->StartPlayer = $play->StartPlayer;
                $new_play->NewPlayer = $play->NewPlayer;
                $new_play->StartPosition = $play->StartPosition;
                $new_play->type = $play->type;
                $new_play->score = $play->score;
                $new_play->circumstances = $play->circumstances;
                $new_play->team = $play->team;
                $new_play->save();

            }
```

```
    }

    public function ShareMatch($match_id)
    {

        $match = MatchSetting::where('id',$match_id)->first();
        $player_list = [];
        $check_notif = Auth::user()
            ->notifications
            ->where('type','App\Notifications\NewPlaysNotification')
            ->where('data.match',$match_id)
            ->first();
        if(count($match->matches) !== 0) $player_list = $match->matches-
>pluck('player_id');
        foreach($player_list as $player){
            $friend = Player::where('friend_id','!=',Auth::id())-
>where('id',$player)->first();
            if($friend &&  $friend->friend_id != Auth::id() && $check_notif){
                event(new AddPlays($match,$friend->friend_id));
            }
        }

    }

    public function GetRoles($id)
    {
        $combined_games = CombinedGames::where('game_first',$id)-
>pluck('game_second')->toArray();
        array_push($combined_games,intval($id));

        $matches = MatchSetting::where('user_id',Auth::id())-
>whereIn('game_id',$combined_games)->get();
        $roles = [];
        foreach ($matches as $match) {
            if(count($match->matches)) {
                foreach ($match->matches as $match) {
                    if($match['role'])array_push($roles,strtok($match['role'],','));
                }
            }
        }

        return array_unique($roles);
    }

    public function RemoveRoleFromMatches($id, $role)
    {
        $combined_games = CombinedGames::where('game_first',$id)-
>pluck('game_second')->toArray();
        array_push($combined_games,intval($id));

        $matches_set = MatchSetting::where('user_id',Auth::id())-
>whereIn('game_id',$combined_games)->get();
        foreach ($matches_set as $match) {
            if(count($match->matches)) {
                foreach ($match->matches as $match) {
                    if($match->role == $role){
                        $match->role = NULL;
                        $match->save();
                    }
                }
            }
        }
    }
```

```
    public function RenameRoleFromMatches($id, $role,$new_role)
    {
        $combined_games = CombinedGames::where('game_first',$id)-
>pluck('game_second')->toArray();
        array_push($combined_games,intval($id));

        $matches_sett = MatchSetting::where('user_id',Auth::id())-
>whereIn('game_id',$combined_games)->get();
        foreach ($matches_sett as $match) {
            if(count($match->matches)) {
                foreach ($match->matches as $match) {
                    if($match->role == $role){
                        $match->role = $new_role;
                        $match->save();
                    }
                }
            }
        }
    }
}

class GamePlaylist extends Model
{
    use HasFactory;

    public function AddToPlaylist($request)
    {
      $music = new GamePlaylist();
      $music->user_id = Auth::id();
      $music->game_id = $request->game_id;
      $music->name = $request->video_name;
      $music->url = $request->url;
      $music->save();
      return $music->id;
    }

    public function RemoveFromPlaylist($request)
    {
        GamePlaylist::where('id',$request->id)->first()->delete();
    }
}

class MatchSetting extends Model
{
    use HasFactory;

    public function Game()
    {
        return $this->belongsTo(Game::class);
    }
    public function Place()
    {
        return $this->belongsTo(Place::class);
    }
    public function User()
    {
        return $this->belongsTo(User::class);
    }
    public function Matches()
    {
        return $this->hasMany(Matches::class,'match_settings_id');
    }

    public function GetAllPlays()
```

```
    {

        $all_plays = MatchSetting::where("user_id",Auth::id())->orderBy('date',
'desc')->get()->groupBy("date");
        return $all_plays;
    }
    public function GetGamePlays($id)
    {
        $combined_games = CombinedGames::where('game_first',$id)-
>pluck('game_second')->toArray();
        array_push($combined_games,intval($id));
        $combined_games;

        $all_plays = MatchSetting::where("user_id",Auth::id())->orderBy('date',
'desc')->whereIn("game_id",$combined_games)->get()->groupBy("date","desc");
        return $all_plays;
    }
    public function GetPlayerPlays($id)
    {
        $match = Matches::where("player_id",$id)->pluck("match_settings_id")-
>toArray();
        $all_plays = MatchSetting::whereIn("id",$match)->orderBy('date', 'desc')-
>get()->groupBy("date");
        return $all_plays;
    }


    public function GetTop100()
    {
        $url = "https://boardgamegeek.com/browse/boardgame";
        $games = [];

        $client = new Client();

        $crawler = $client->request('GET', $url);

        $crawler->filter('.primary')->each(function($node) use (&$games){
        $rating = $node->ancestors()->filter('.collection_bggrating')-
>each(function($node){
                 return $node->text();
        });
                $game =
                [
                    "name" => $node->text(),
                    "year" => $node->ancestors()->filter('.dull')->text(),
                    "url" => $node->attr('href'),
                    "image" => $node->ancestors()->ancestors()->ancestors()-
>filter('img')->getNode(0)->getAttribute('src'),
                    "description" => $node->ancestors()->filter('.smallefont')-
>text(),
                    "geek_rating" => $rating[0],
                    "avg_rating" => $rating[1],
                    "num_voters" =>  $rating[2],
                ];

                array_push($games, $game);

        });

        return $games;
    }

    public function GetHotness()
    {
```

```
        $url = "https://www.boardgameoracle.com/list/bgg-hotness";
        $games = [];

        $client = new Client();

        $crawler = $client->request('GET', $url);

        $crawler->filter('.MuiPaper-elevation1')->each(function($node) use (&$games){
                $game =
                [
                    "name" => $node->filter(".MuiTypography-root")->text(),
                    "year" => $node->filter(".MuiTypography-colorTextSecondary")-
>text(),
                    "image" => $node->filter(".MuiCardMedia-root")->attr("style"),
                ];

                array_push($games, $game);

        });

        return $games;
    }
}

class Place extends Model
{
    use HasFactory;

    public function addPlace($request)
    {
        $check_place = Place::where("name",$request->input("name"))-
>where("user_id",Auth::id())->first();

        if($check_place === null)
        {
            $place = new Place();
            $place->name = $request->input("name");
            $place->user_id = Auth::id();
            $place->save();
        }
    }

    public function renamePlace($request, $id)
    {
        $place = Place::where("id", $id)->first();
        $place->name = $request->name;
        $place->save();
    }
}

class Player extends Model
{
    use HasFactory;

    public function User(){
        return $this->belongsTo(User::class, 'friend_id');
    }

    public function checkPlaysCount(){
        return Matches::where('player_id',$this->id)->count();
    }

    public function getWinRate()
    {
```

```
        $stats = $this->getMainStats($this->id);
        return floor($stats["wins"] /  $stats["plays"] * 100) ;
    }

    public function getLosesRate()
    {
        return 100 - $this->getWinRate();
    }

    public function AddPlayer($request){
        $player = new Player();
        $player->user_id = Auth::id();
        $player->name = $request->name;
        $player->friend_id = User::where('name',$request->friend)->pluck('id')-
>first();

        if($request->image !== null)
        {
            $player->image =  Crypt::encryptString($request->image-
>getClientOriginalName()).".". $request->image->extension();
        }
        if($player->save())
        {
            if($request->image !== null) $request->image-
>move(public_path('avatars'),  $player->image );
        }
    }

    public function UpdatePlayer($request, $id){
        $player = Player::where("id",$id)->first();
        $player->name = $request->input("name");
        $player->friend_id = User::where('name',$request->friend)->pluck('id')-
>first();

        if($request->image !== null)
        {
            $player->image = $request->image->getClientOriginalName();
        }
        else  $player->image = $player->image;
        if($player->save())
        {
            if($request->image !== null) $request->image-
>move(public_path('avatars'), $player->image);
        }
    }

    public function getPlayerStats($id){
        $player = Player::find($id);
        $mainStats = $this->getMainStats($id);
        $gamesStats = $this->getGamesStats($id);
        $statsByLocation = $this->getSatsOnLocation($id);
        $playersCount = $this->opponentPlaysCount($id);


        return [
            "player" => $player,
            "mainStats" => $mainStats,
            "gamesStats" => $gamesStats,
            "statsByLocation" => $statsByLocation,
            "playersCount" => $playersCount,
        ];

    }
```

```
    public function locationStats($loc){
        $locations_array = [];
        for ($i=0; $i < count($loc); $i++) {
            $players_count = MatchSetting::where('id',$loc[$i]->match_settings_id)-
>first()->place->name;
            array_push($locations_array,$players_count);
        }
        return $locations_array;
    }

    public function playersStats($list){
        $players_array = [];
        for ($i=0; $i < count($list); $i++) {
            array_push($players_array,$list[$i]->player_id);
        }
        return $players_array;
    }

    public function deletePlayer($request, $id){
        $check = Player::where("id",$id)->where("user_id",Auth::id())->first();
        // $deleted_player_plays =  Matches::where('player_id',$request->player)-
>pluck('match_settings_id')->toArray();
        // $player_for_swap =
Matches::whereIn('match_settings_id',$deleted_player_plays)-
>where('player_id','!=',$request->player)->pluck('match_settings_id');
        // dd($deleted_player_plays,$check);
        if($check){
            // if($request->has('player')){
            //     Matches::where('player_id',$id)->where('player_id','!=',$request-
>has('player'))->update(array('player_id',$request->has('player')));
            // }
            $player = Player::where('user_id',Auth::id())->where("id",$id)->first();
            $image = "avatars/".$player->image;
            if(File::exists($image) && $player->image !== 'friend_friend.jpg' &&
$player->image !== 'default-player.png') {
                File::delete($image);
            }
            $player->delete();
        }
    }

    public function statsOnGames($PlaysGame, $game_statistic_info)
    {
        for ($i=0; $i < count($PlaysGame) ; $i++) {
            $game =
            [
                "name" => MatchSetting::where("id",$PlaysGame[$i]-
>match_settings_id)->first()->game->name,
                "win" => $PlaysGame[$i]->victory == 1 ? 1 : 0,
                "lose" => $PlaysGame[$i]->victory == 0 ? 1 : 0,
                "percent" => 0
             ];
             if(array_search($game["name"], array_column($game_statistic_info,
'name')) === false)
             {
              array_push($game_statistic_info,$game);
             }
             else{
                 $game_statistic_info[array_search($game["name"],
array_column($game_statistic_info, 'name'))]["win"] += $game["win"];
                 $game_statistic_info[array_search($game["name"],
array_column($game_statistic_info, 'name'))]["lose"] += $game["lose"];
             }
         }
```

```
         return $game_statistic_info;
    }

    public function getMainStats($id){

        $plays_count = Matches::where('player_id',$id)->count();
        $wins_count  = Matches::where('player_id',$id)->where('victory',true)-
>count();
        $loses_count = Matches::where('player_id',$id)->where('victory',false)-
>count();


        return [
                "plays" => $plays_count,
                "wins" => $wins_count,
                "loses" => $loses_count,
                "percent" => $wins_count === 0 ? 0 : floor($wins_count / $plays_count
* 100),
            ];
    }

    public function getGamesStats($id){
        $game_statistic_info = [];

        $game_statistic_info = $this->statsOnGames(Matches::where("player_id",$id)-
>get(),$game_statistic_info);

        for ($i=0; $i < count($game_statistic_info); $i++) {
            $game_statistic_info[$i]['percent'] = $game_statistic_info[$i]["win"] ===
0 ? 0 : floor($game_statistic_info[$i]["win"] /
($game_statistic_info[$i]["win"]+$game_statistic_info[$i]["lose"]) * 100);
        }

         array_multisort(array_column($game_statistic_info, 'percent'), SORT_DESC,
$game_statistic_info);

         return $game_statistic_info;
    }



    public function opponentPlaysCount($id){
        $matches = Matches::where("player_id",$id)->pluck("match_settings_id")-
>toArray();

        $match_settings = MatchSetting::whereIn("id",$matches)->get();
        $allPlays = [];

        for ($i=0; $i < count($matches); $i++) {
            array_push($allPlays, $match_settings[$i]->matches-
>where("player_id","!=",$id)->pluck("player_id")->toArray());
        }

        $count = array_count_values(array_merge(...$allPlays));
        foreach($count as $key => $val){
            $newkey = Player::where("id",$key)->first()->name;
            $count[$newkey] = $count[$key];
            unset($count[$key]);
        }

        return $count;
    }
    public function getSatsOnLocation($id){
        $matches = Matches::where("player_id",$id)->get();
```

```
        return array_count_values($this->locationStats($matches));
    }
}

//locale variables
let locale_0_plays = "null";
let locale_confirm = "null";
let locale_remove_expansion_annotation = "null";
let locale_remove_play_annotation = "null";
let locale_yes_remove = "null";
let locale_cancel = "null";

//--------

$(document).ready(function () {
    $("#base-game-filter").on("change", function (event) {
        event.preventDefault();
        document.getElementById("no-played-games-filter").checked = false;
        if (this.checked) $(".game").removeClass("hide");
        else $(".game").addClass("hide");
    });

    $("#expansions-filter").on("change", function (event) {
        event.preventDefault();
        document.getElementById("no-played-games-filter").checked = false;
        if (this.checked) $(".expansion").removeClass("hide");
        else $(".expansion").addClass("hide");
    });

    $("#no-played-games-filter").on("change", function (event) {
        event.preventDefault();
        if (this.checked) {
            document.getElementById("base-game-filter").checked = false;
            document.getElementById("expansions-filter").checked = false;
            let games = document.getElementsByClassName("game");
            for (let index = 0; index < games.length; index++) {
                if (
                    games[index].querySelector(".last-play").innerHTML !==
                    locale_0_plays
                )
                    games[index].classList.add("hide");
                else games[index].classList.remove("hide");
            }
            $(".expansion").addClass("hide");
        } else {
            $(".game").addClass("hide");
        }
    });

    $(".dropdown-input").click(function (event) {
        event.stopPropagation();
    });

    $("#bgg-close-add").click(function (event) {
        event.preventDefault();
        document.getElementById("collection-form-block").classList.add("hide");
        document.getElementById("search-block").classList.remove("hide");

    });

    $("#bgg-close-block").click(function (event) {
        event.preventDefault();
        document.getElementById("bgg-search-block").classList.add("hide");
        document.getElementById("collection-game-list").classList.remove("hide");
```

```
});

$("#bgg-open-block").click(function (event) {
    event.preventDefault();
    document.getElementById("bgg-search-block").classList.remove("hide");
    document.getElementById("collection-game-list").classList.add("hide");
});

$("#bgg-search-form").on("submit", function (e) {
    e.preventDefault();

    let searched_name = $("#bgg-search").val();
    let token = $("input[name='_token']", "#bgg-search-form").val();

    $.ajax({
        url: "/BGGSearch",
        type: "POST",
        data: {
            _token: token,
            name: searched_name,
        },
        success: function (data) {
            SearchFill(data.games_list);
        },
        error: function (data) {
            console.log("Oops something going wrong ^_^");
        },
    });
});
$(".delete-match-icon").on("click", function (e) {
    e.preventDefault();
    Swal.fire({
        title: locale_confirm,
        text: locale_remove_play_annotation,
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#3085d6",
        cancelButtonColor: "#d33",
        confirmButtonText: locale_yes_remove,
        cancelButtonText: locale_cancel,
    }).then((result) => {
        if (result.isConfirmed) {
            window.location = $(this).attr("href");
        }
        if (result.cancel) {
            return false;
        }
    });
});
$(".remove-expansion").on("click", function (e) {
    e.preventDefault();
    let deleteUrl = $(this).attr("href");
    Swal.fire({
        title: locale_confirm,
        text: locale_remove_expansion_annotation,
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#3085d6",
        cancelButtonColor: "#d33",
        confirmButtonText: locale_yes_remove,
        cancelButtonText: locale_cancel,
    }).then((result) => {
        if (result.isConfirmed) {
            let form = document.createElement("form");
```

```
            form.setAttribute("method", "post");
            form.setAttribute("action", deleteUrl);

            let csrfField = document.createElement("input");
            csrfField.setAttribute("type", "hidden");
            csrfField.setAttribute("name", "_token");
            csrfField.setAttribute(
                "value",
                $('meta[name="csrf-token"]').attr("content")
            );
            form.appendChild(csrfField);

            let methodField = document.createElement("input");
            methodField.setAttribute("type", "hidden");
            methodField.setAttribute("name", "_method");
            methodField.setAttribute("value", "DELETE");
            form.appendChild(methodField);

            document.body.appendChild(form);
            form.submit();
        }
        if (result.cancel) {
            return false;
        }
    });
});



$(".game").click(function (event) {
    this.getElementsByClassName("link")[0].click();
});

$("input[name='base']").change(function () {
    if ($(this).is(":checked") === false) {
        document.getElementById("points-type").disabled = true;
        document.getElementById("points-type").checked = false;
        document.getElementById("no-points-type").disabled = true;
        document.getElementById("no-points-type").checked = false;
        $("input[name='type']").each(function () {
            this.checked = false;
            this.disabled = true;
        });
    } else {
        document.getElementById("points-type").disabled = false;
        document.getElementById("points-type").checked = false;
        document.getElementById("no-points-type").disabled = false;
        document.getElementById("no-points-type").checked = true;
        document.getElementById("default-type").checked = true;
        $("input[name='type']").each(function () {
            this.disabled = false;
        });
    }
});
$("input[name='type']").change(function () {
    if ($(this).val() == "coop" || $(this).val() == "team") {
        document.getElementById("points-type").disabled = true;
        document.getElementById("points-type").checked = false;
        document.getElementById("no-points-type").disabled = false;
        document.getElementById("no-points-type").checked = true;
    }
    else{
        document.getElementById("points-type").disabled = false;
        document.getElementById("points-type").checked = false;
```

```
            document.getElementById("no-points-type").checked = true;
         document.getElementById("no-points-type").disabled = false;
        }
    });

    function SearchFill(games_list) {
        document.getElementById("searched-list").innerHTML = "";

        for (let count = 0; count <= games_list.length; count++) {
            let block_list_item = document.createElement("div");
            block_list_item.classList.add("collection-game-block");
            block_list_item.addEventListener("click", function (event) {
                ShowAddGame(this);
            });
            let img_block = document.createElement('div');
            img_block.classList.add('collection-image-block');

            let img = document.createElement("img");
            img.classList.add("collection-game-image");
            img.setAttribute("src", "" + games_list[count].image);

            img_block.appendChild(img);

            let info_block = document.createElement('div');
            info_block.classList.add("game-info");

            let block_item_name = document.createElement("span");
            block_item_name.classList.add("game-name");
            block_item_name.innerHTML += games_list[count].name;

            let block_item_year = document.createElement("p");
            block_item_year.classList.add("game-year");
            block_item_year.innerHTML += games_list[count].year;

            info_block.appendChild(block_item_name);
            info_block.appendChild(block_item_year);

            let block_item_url = document.createElement("div");
            block_item_url.classList.add("list-game-url");
            block_item_url.classList.add("hide");
            block_item_url.innerHTML += games_list[count].url;

            block_list_item.appendChild(img_block);
            block_list_item.appendChild(info_block);
            block_list_item.appendChild(block_item_url);
            document.getElementById("searched-list").appendChild(block_list_item);
        }
    }

    $(document).on("click", ".boardgame-list-item", function () {
        let searched_img = $("img", this).attr("src");
        let searched_name = $(".list-game-name", this).text();
        let searched_year = $(".list-game-year", this).text();
        let searched_url = $(".url_link", this).attr("href");

        let img = document.getElementById("toggle-img").setAttribute("src",
searched_img);
        $("#toggle-name").text(searched_name);
        $("#toggle-year").text(searched_year);
        $("#toggle-url").attr("href", searched_url);
    });

    $("#back-to-search-but").click(function (event) {
        event.preventDefault();
```

```
        document.getElementById("add-block").classList.toggle("show-add-block");
    });

    $("#collection-search").on("keyup", function () {
        let list = document.getElementsByClassName("game-name");

        for (var i = 0; i < list.length; i++) {
            if (list[i].innerHTML.indexOf($(this).val()) !== -1) {

list[i].parentElement.parentElement.parentElement.classList.remove("hide");
            } else {

list[i].parentElement.parentElement.parentElement.classList.add("hide");
            }
        }
    });

    function ShowAddGame(block) {
        document.getElementById("collection-form-block").classList.remove("hide");
        document.getElementById("search-block").classList.add("hide");
        document.getElementById("bgg-searched-
image").setAttribute("src",block.getElementsByClassName("collection-game-
image")[0].getAttribute("src"));
        document.getElementById("colleciton-form-url").value =
block.getElementsByClassName("list-game-url")[0].innerHTML;
        document.getElementById("colleciton-form-img").value =
block.getElementsByClassName("collection-game-image")[0].getAttribute("src");
        document.getElementById("bgg-searched-name").value =
block.getElementsByClassName("game-name")[0].innerHTML.replace('amp;','');
        document.getElementById("bgg-searched-name").style.width =
document.getElementById("bgg-searched-name").value.length + "ch";
        document.getElementById("bgg-searched-year").value =
block.getElementsByClassName("game-year")[0].innerHTML;
        document.getElementById("bgg-searched-url").innerHTML
=block.getElementsByClassName("list-game-url")[0].innerHTML;
        document.getElementById("bgg-add-block").classList.remove("hide");

    }
});
window.Setlocale = function (locale_data) {
    locale_ = locale_data[0];
    locale_confirm = locale_data[1];
    locale_remove_expansion_annotation = locale_data[2];
    locale_remove_play_annotation = locale_data[3];
    locale_yes_remove = locale_data[4];
    locale_cancel = locale_data[5];
};
$(document).ready(function() {



    $('.remove-task').on("click",function(event){
        let task = this;
        RemoveTask(task);

    });

    $('.remove-challenge').on("click",function(event){
       let challenge = this;
       RemoveChallenge(challenge);
    });

    $('.task-complete').on('change',function (e) {
```

```
        FinishTask(this);
    });

    $('.add-task-form').on('submit',function (e) {
        e.preventDefault();
        let task_list = this.previousElementSibling;
        let token = $('meta[name="csrf-token"]').attr('content');
        $.ajax({
            url: "/AddTask",
            type: "POST",
            data: {
                "_token": token,
                challenge:this.children[1].dataset.challenge,
                name:this.children[1].value
                },
                success:function(data){
                        AddTask(data.task,task_list);
                        FillProgressBar();
                  },
            error: function(data){
            }
        });
    })
    FillProgressBar();

    $('.challenge-icon').on('click',function(){
        document.getElementById('open-icons-library').style.backgroundImage =
"url('/challenge-icons/" + this.src.split('http://tabletopstats/challenge-icons/')[1]
+ "')";
        document.getElementById('challenge-icon').value = this.dataset.id;
    });

      $('#open-icons-library').on('click',function() {
            $('.challenge').addClass('hide');
            $('#icons-library').removeClass('hide');
      });
});


function AddTask(task,list) {

    let block = document.createElement('div');

    let input = document.createElement('input');
    input.setAttribute("type",'checkbox');
    input.setAttribute('data-id',task['id']);
    input.classList.add('task-complete');
    input.onchange = function(event) { FinishTask(this); };

    let span = document.createElement('span');
    span.innerHTML = task['name'];

    let button = document.createElement('button');
    button.setAttribute('data-id',task['id']);
    button.classList.add('remove-task');
    button.innerHTML = '✕';
    button.onclick = function(event) { RemoveTask(this); };

    block.appendChild(input);
    block.appendChild(span);
    block.appendChild(button);


    list.appendChild(block);
```

```
}

function RemoveChallenge(challenge) {
    let token = $('meta[name="csrf-token"]').attr('content');
    $.ajax({
        url: "/DeleteChallenge",
        type: "POST",
        data: {
            "_token": token,
            id:challenge.dataset.id
            },
            success:function(data){
                challenge.parentElement.remove();
              },
        error: function(data){

        }
    });
}

function RemoveTask(task) {
    let token = $('meta[name="csrf-token"]').attr('content');

    $.ajax({
        url: "/DeleteTask",
        type: "POST",
        data: {
            "_token": token,
            id:task.dataset.id
            },
            success:function(data){
                task.parentElement.remove();FillProgressBar();
              },
        error: function(data){

        }
    });
}

function FinishTask(element) {
    let token = $('meta[name="csrf-token"]').attr('content');
    $.ajax({
        url: "/FinishTask",
        type: "POST",
        data: {
            "_token": token,
            id:element.dataset.id
            },
            success:function(data){
                MoveProgressBar(element);
              },
        error: function(data){

        }
    });
}

function MoveProgressBar(element) {
    let parentElement = element.parentElement.parentElement.parentElement;
    let progressBar =
element.parentElement.parentElement.previousElementSibling.children[0];
    let list = element.parentElement.parentElement;
    let all_tasks = $(list).find('.task-complete').length;
    let completed_task =$(list).find('.task-complete:checked').length;
```

```
    let percent = completed_task / all_tasks ;
    progressBar.style.width =  percent * 100 + "%";
    progressBar.style.background =  getColor(percent);



    if(percent === 1 && all_tasks > 0)
    {
        let complete_but = document.createElement('button');
        complete_but.classList.add('complete-challenge');
        complete_but.innerHTML = 'complete таск';
        complete_but.setAttribute('value',element.dataset.id);
        complete_but.onclick = function (e) { CompleteChallenge(parentElement);}

        if (!$(parentElement).find('.complete-
challenge').length)parentElement.appendChild(complete_but);
    }
    if(percent !== 1)
    {
        $(parentElement).find('.complete-challenge').remove();
    }
}

function FillProgressBar() {
    $('.task-complete').each(function (){
        MoveProgressBar(this);
    });
    $('.challenge').each(function (){
       let count = $(this).find('.task-complete').length;
       if(count === 0){
        $(this).find('.progress-bar').css('background', 'none');
        $(this).find('.progress-bar').css('width', '0%');
       }

    });
}

function CompleteChallenge(element) {

    let token = $('meta[name="csrf-token"]').attr('content');
    $.ajax({
        url: "/FinishChallenge",
        type: "POST",
        data: {
            "_token": token,
            id:$(element).find('.task-name').data("challenge")
            },
            success:function(data){
                element.remove();
              },
        error: function(data){

        }
    });
}

let selected_team = null;


$(document).ready(function () {

    // Open player block

    $('#open-players').on('click', function(){
```

```
        let save_form = document.getElementById('save-match-form');
        if(save_form)save_form.classList.add('hide');
        let update_form = document.getElementById('update-match-form');
        if(update_form)update_form.classList.add('hide');
        if(document.getElementById('forum-block'))document.getElementById('forum-
block').classList.add('hide');
        document.getElementById('chose-players-block').classList.remove('hide');
    });

    // Close player block

    $('#close-players-list').on('click', function(){
        let save_form = document.getElementById('save-match-form');
        if(save_form)save_form.classList.remove('hide');
        let update_form = document.getElementById('update-match-form');
        if(update_form)update_form.classList.remove('hide');
        if(document.getElementById('forum-block'))document.getElementById('forum-
block').classList.remove('hide');
        document.getElementById('chose-players-block').classList.add('hide');
    });

    // Select group player

    function SelectPlayers(players_list) {
        $("input[type=checkbox][name=chosen_players]:checked").each(function () {
            this.click();
        });
        $("input[type=checkbox][name=chosen_players]").each(function () {
            for (let j = 0; j < players_list.length; j++)
                if (this.dataset.id == players_list[j]) this.click();
        });
    }

    // Player select

    $(".players_checkbox").change(function (event) {
        if(this.checked === false){
            document.getElementById("player-" + this.dataset.id + "-block").remove();
            return false;
        }
        CreatePlayer(this);
    });

    // Create player default block

 window.CreatePlayer = function CreatePlayer(element,updatePlayer = null,team =
selected_team) {
        let match_type = $("input[type=radio][name=type]:checked").data("type");
        let block = match_type !== "team" ? document.getElementById("players-list") :
document.getElementById("team-list-" + team);
            let single = document.createElement("div");
            single.classList.add("single-player-block");
            single.id = "player-" + element.dataset.id + "-block";

            if(match_type === "team"){
                let player_exist = document.getElementById("player-" +
element.dataset.id + "-block");

                if(player_exist){
                    let player = player_exist.cloneNode(true);
                    player.lastChild.addEventListener("click", function () {
                        openUserInfo(element.dataset.id);
                    });
                    block.appendChild(player);
```

```
                    player_exist.remove();
                    return false;
                }
            }
            if (match_type !== "coop" && match_type !== "team"){

                let check_winner_icon  = document.createElement("label");
                check_winner_icon.classList.add("check-winner-icon");
                check_winner_icon.classList.add("no-winner");
                check_winner_icon.setAttribute('for','check-winner-'+
element.dataset.id);
                check_winner_icon.innerHTML = "♛";
                check_winner_icon.dataset.player = element.dataset.id;
                check_winner_icon.onclick = function (event) {
                    ChangeWinner(this);
                };
                if(updatePlayer !== null && updatePlayer['victory'])
check_winner_icon.classList.remove('no-winner');

                single.appendChild(check_winner_icon);
            }

            let check_winner = document.createElement("input");
            check_winner.setAttribute('type', 'checkbox');
            check_winner.setAttribute('name', 'winners[]');
            check_winner.setAttribute('id','check-winner-'+ element.dataset.id);
            check_winner.value = element.dataset.id;
            check_winner.classList.add("check-winner");
            check_winner.classList.add("hide");
            if(updatePlayer !== null && updatePlayer['victory']) check_winner.checked
= true;

            let game_player_name  = document.createElement("label");
            game_player_name.classList.add('player-name-label');
            game_player_name.innerHTML = element.dataset.name;

            let game_player_id = document.createElement("input");
            game_player_id.setAttribute("type","hidden");
            game_player_id.setAttribute("readOnly","true");
            game_player_id.setAttribute("name","player[]");
            game_player_id.classList.add("player-name");
            game_player_id.value += element.dataset.id;

            let i = document.createElement("i");
            i.classList.add("fa-id-card");
            i.classList.add("fa");
            i.setAttribute("aria-hidden", "true");
            i.addEventListener("click", function () {
                openUserInfo(element.dataset.id);
            });

            let player_role = document.createElement("input");
            player_role.setAttribute("type","hidden");
            player_role.setAttribute("readOnly","true");
            player_role.setAttribute('name','roles[]');
            player_role.id = "player-role-" + element.dataset.id;
            if(updatePlayer !== null) player_role.value = updatePlayer['role'];

            let player_position = document.createElement("input");
            player_position.setAttribute("type","hidden");
            player_position.setAttribute("readOnly","true");
            player_position.setAttribute('name','positions[]');
            player_position.id = "player-position-" + element.dataset.id;
```

```
            if(updatePlayer !== null) player_position.value =
updatePlayer['StartPosition'];

            let player_first_player = document.createElement("input");
            player_first_player.setAttribute("type","hidden");
            player_first_player.setAttribute('name','firstPlayer[]');
            player_first_player.classList.add('hide');
            player_first_player.id = "first-player-" + element.dataset.id;
            player_first_player.value = false;
            if(updatePlayer && updatePlayer['StartPlayer']) player_first_player.value
= true;

            let player_first_time = document.createElement("input");
            player_first_time.setAttribute("type","hidden");
            player_first_time.setAttribute('name','firstTime[]');
            player_first_time.classList.add('hide');
            player_first_time.id = "first-time-" + element.dataset.id;
            player_first_time.value = false;
            if(updatePlayer && updatePlayer['NewPlayer']) player_first_time.value =
true;

            single.appendChild(game_player_name);
            single.appendChild(game_player_id);
            single.appendChild(check_winner);
            single.appendChild(player_role);
            single.appendChild(player_position);
            single.appendChild(player_first_player);
            single.appendChild(player_first_time);

            if(match_type === "team"){
                let team_name  = document.createElement("input");
                team_name.setAttribute("type","hidden");
                team_name.setAttribute('name','team[]');
                team_name.classList.add('hide');
                team_name.id = "player-"+element.dataset.id+"-time";
                team_name.value = document.getElementById(team + '-team-name').value;

                single.appendChild(team_name);
            }



            single.appendChild(i);

            if ($("input[type=radio][name=type]:checked").data("type")=== "points") {
                let input = document.createElement("input");
                input.setAttribute("type", "text");
                input.setAttribute("name", "points[]");
                input.setAttribute("placeholder", "locale_points");
                input.classList.add("game-player-scores");
                input.classList.add("score-input");
                input.addEventListener("keyup", CheckWinner);
                input.onkeypress = function (e) {
                    e = e || window.event;
                    var charCode =
                        typeof e.which == "number" ? e.which : e.keyCode;

                    if (!charCode || charCode == 8) {
                        return;
                    }
                    var typedChar = String.fromCharCode(charCode);
                    if (/\d/.test(typedChar)) {
                        return;
                    }
```

```
                    if (typedChar == "-" && this.value == "") {
                        return;
                    }
                    return false;
                };
                if(updatePlayer) input.value = updatePlayer['score'];
                single.appendChild(input);
            }

            block.appendChild(single);

            if($("input[type=radio][name=type]:checked").data("type") ===
"points")CheckWinner();

        if (
            document.getElementsByClassName("single-player-block").length !== 0
        ) {
            $("#definition-with-animation").removeAttr("disabled");
            $("#definition-without-animation").removeAttr("disabled");
        } else {
            $("#definition-with-animation").attr("disabled", "disabled");
            $("#definition-without-animation").attr("disabled", "disabled");
        }
    }

    // Get groups players

    $(".get-group-players").on("click", function (e) {
        let group_id = this.dataset.id;

        let token = $('meta[name="csrf-token"]').attr("content");
        $.ajax({
            url: "/get-group-players",
            type: "POST",
            data: {
                _token: token,
                id: group_id,
            },
            success: function (data) {
                SelectPlayers(data.players);
            },
            error: function (data) {},
        });
    });

    // Save user info

    $("#save-info").on("click", function () {
        event.preventDefault();

        document.getElementById("player-role-" + this.dataset.id).value =
document.getElementById("player-role").value;
        document.getElementById("player-position-" + this.dataset.id).value =
document.getElementById("player-start-pos").value;
        document.getElementById("first-player-" + this.dataset.id).value =
document.getElementById("start_player").checked;
        document.getElementById("first-time-" + this.dataset.id).value =
document.getElementById("first_game").checked;

        document.getElementById("info-close").click();
    });

    // Close user info
```

```
    $("#info-close").click(function (event) {

        document.getElementById('add-default-match-block').classList.remove('hide');
        document.getElementById('player-info').classList.add('hide');
    });

});

    // Check winner

    window.CheckWinner = function CheckWinner() {
        var selected = document.getElementsByClassName("game-player-scores");
        var max = selected[0].value;
        for (let index = 0; index < selected.length; index++) {
            if (selected[index].value !== "-")
                max = Math.max(max, parseInt(selected[index].value));
        }
        for (let index = 0; index < selected.length; index++) {
            if (selected[index].value == max){
                    selected[index].parentElement.children[1].checked = true;
                    selected[index].parentElement.children[0].classList.remove("no-
winner")
                    $(selected[index]).siblings(".check-
winner").prop('checked',true);
                }
            else{
                selected[index].parentElement.children[1].checked = false;
                selected[index].parentElement.children[0].classList.add("no-winner");
                $(selected[index]).siblings(".check-winner").prop('checked',false);
            }
        }
    }

    // Choose default place

    window.chooseDefaultPlace = function chooseDefaultPlace(game_settings){
        if(game_settings["points_directed"] ===
"points")$("input[type=radio][name='type'][data-type='points']").click();
        if(game_settings["points_directed"] === "no-
points")$("input[type=radio][name='type'][data-type='no-points']").click();
        if(game_settings["type"] === "coop")$("input[type=radio][name='type'][data-
type='coop']").click();
        if(game_settings["type"] === "team")$("input[type=radio][name='type'][data-
type='team']").click();
    }

    // Open block to select player for team

    window.AddTeamPlayer = function AddTeamPlayer(selected) {
        selected_team = selected; // selected team id
        $(".players_list input:checked").each(function () {
            $("input:checkbox").prop("checked", false);
        });

        selectTeamPlayer();
        let save_form = document.getElementById('save-match-form');
        if(save_form)save_form.classList.add('hide');
        let update_form = document.getElementById('update-match-form');
        if(update_form)update_form.classList.add('hide');

        if(document.getElementById('forum-block'))document.getElementById('forum-
block').classList.add('hide');
        document.getElementById('chose-players-block').classList.remove('hide');
    }
```

```
    // Select already selected team player

    function selectTeamPlayer(){
        let player = document.getElementById('team-list-' + selected_team);
        for (let i = 0; i < (player.children).length; i++) {
            $('.players_checkbox[data-id = ' + player.children[i].children[1].value +
']').prop('checked', true);
        }
    }

    // Fill user info block

    function openUserInfo(id) {

        document.getElementById("player-role").value = "";
        document.getElementById("player-start-pos").value = "";
        $("#player-info input:checked").each(function () {
            $("input:checkbox").prop("checked", false);
        });

        document.getElementById("player-role").value =
document.getElementById('player-role-'+id).value;
        document.getElementById("player-start-pos").value =
document.getElementById('player-position-'+id).value;
        if (document.getElementById('first-player-'+id).value === "true")
$("#start_player").prop("checked", true);
        if (document.getElementById('first-time-'+id).value === "true")
$("#first_game").prop("checked", true);

        document.getElementById('save-info').dataset.id = id;
        document.getElementById('add-default-match-block').classList.add('hide');
        document.getElementById('player-info').classList.remove('hide');

    }
const { add } = require("lodash");

let lastType = null;
let collection = [];
let colors = [
    "rgb(255, 99, 132,0.8)",
    "rgb(201, 20, 111,0.8)",
    "rgb(1, 200, 200,0.5)",
    "rgb(1, 200, 111,0.5)",
    "rgb(255, 255, 25,0.5)",
    "rgb(255, 0, 255,0.5)",
    "rgb(5, 5, 255,0.5)",
    "rgb(55, 255, 55,0.5)",
    "rgb(255, 150, 1,0.5)",
    "rgb(100, 1, 1,0.5)",
    "rgb(66, 6, 120,0.5)",
];

$(document).ready(function () {

    // Choose place for default play

    $("input[type=radio][name=chosen_place]").change(function () {
        document.getElementById("location-name").innerHTML = this.value;
        document.getElementById("chosen_place").value = this.dataset.id;
    });

    // Open location block
```

```
    $("#location-name").click(function (e) {
        document.getElementById('places-block').classList.remove('hide');
        document.getElementById('add-default-match-block').classList.add('hide');
    });

     // Close location block

     $("#place-close-add").click(function (e) {
        document.getElementById('places-block').classList.add('hide');
        document.getElementById('add-default-match-block').classList.remove('hide');
    });

    // Increase rounds count

    $("#round-increase").on("click", function () {
        let rounds = document.getElementById("rounds-counter");
        if (rounds.value === "") {
            rounds.value = 0;
        } else {
            document.getElementById("rounds-counter").value++;
        }
    });

    // Decrease rounds count

    $("#round-decrease").on("click", function () {
        let rounds = document.getElementById("rounds-counter");
        if (rounds.value === "") {
            rounds.value = 0;
        }
        else {
            if (rounds.value > 0) {
                document.getElementById("rounds-counter").value--;
                if (rounds.value == 0)
                    document.getElementById("rounds-counter").value = "";
            }
        }

    });

    // First player definition with animation

    $("#definition-with-animation").on("click", function (e) {
        let numSegments = document.getElementsByClassName('single-player-
block').length;
        let segments = [];
        for (let index = 0; index < numSegments; index++) {
            segments.push({
                fillStyle: colors[index],
                text: $('.player-name-label')[index].innerHTML,
                id: $('input[name="player[]"]')[index].value,
            });
        }


        document.getElementById('wheel-block').classList.remove('hide');
        let theWheel = new Winwheel({
            outerRadius: 212,
            centerX: 217,
            centerY: 219,
            textFontSize: 28,
            numSegments: numSegments,
            segments: segments,
            animation: {
```

```
                type: "spinToStop",
                duration: 5,
                spins: 8,
                callbackFinished: checkFirstPlayer,
            },
        });

        theWheel.startAnimation();
    });

     // First player definition without animation

     $("#definition-without-animation").on("click", function (e) {
         let inputs = $('input[name="firstPlayer[]"]');
         inputs.attr('value','false');
         let random = Math.floor(Math.random() * inputs.length);
         inputs[random].value = 'true';
         alert($('.player-name-label')[random].innerHTML + " first ПЛАЙЯР");
    });

    // Select first player

    function checkFirstPlayer(indicatedSegment) {
        let inputs = $('input[name="firstPlayer[]"]');
        inputs.attr('value','false');
        alert(indicatedSegment.text + " first ПЛАЙЯР");
        $(`#first-player-${indicatedSegment.id}`).attr('value','true');
    }

    // Set default place

    window.setDefaultPlace = function (place_id) {
        $("input[type=radio][name=chosen_place][data-id='"+place_id+"']").click();
    };

    // Change game type

    $("input[type=radio][name=type]").change(function () {
        let count = document.getElementsByClassName("single-player-block");
        if ($(this).data("type") === "no-points") {

            $(".players_checkbox").attr("name", "chosen_players");
            $(".score-input").remove();
            if (lastType == "team") {
                CreatePlayerBlock();
                $(".players_checkbox").prop("checked", false);
            }
            if (document.getElementsByClassName("save-match")[0] != null)
                document.getElementsByClassName("save-match")[0].id ="save-default-
no-points";
            if (document.getElementsByClassName("update-match")[0] != null)
                document.getElementsByClassName("update-match")[0].id ="save-default-
no-points";
            $("#coop-player-block").remove();
            for (let index = 0; index < count.length; index++) {
                if ($(count[index]).find('.check-winner').length === 0) {

                    let check_winner_icon  = document.createElement("label");
                    check_winner_icon.classList.add("check-winner-icon");
                    check_winner_icon.setAttribute('for','check-winner-'+
count[index].childNodes[1].value);
                    check_winner_icon.innerHTML = "♚";
                    check_winner_icon.classList.add("no-winner");
                    check_winner_icon.onclick = function (event) {
```

```
                        ChangeWinner(this);
                    };

                    let check_winner = document.createElement("input");
                    check_winner.setAttribute('type', 'checkbox');
                    check_winner.setAttribute('name', 'winners[]');
                    check_winner.setAttribute('id','check-winner-'+
count[index].childNodes[1].value);
                    check_winner.value = count[index].childNodes[1].value;
                    check_winner.classList.add("check-winner");
                    check_winner.classList.add("hide");

                    count[index].prepend(check_winner_icon);
                    count[index].prepend(check_winner);
                }
            }
        }
        if ($(this).data("type") === "points") {

            $(".players_checkbox").attr("name", "chosen_players");
            if (lastType == "team") {
                CreatePlayerBlock();
                $(".players_checkbox").prop("checked", false);
            }
            for (let index = 0; index < count.length; index++) {
                let input = document.createElement("input");
                input.id = "checkbox-" + index;
                input.setAttribute("type", "text");
                input.setAttribute("name", "points[]");
                input.setAttribute("placeholder", "locale_points");//FIXME:
                input.classList.add("game-player-scores");
                input.classList.add("score-input");
                input.addEventListener("keyup", function(){CheckWinner()});
                input.onkeypress = function (e) {
                    e = e || window.event;
                    var charCode =
                        typeof e.which == "number" ? e.which : e.keyCode;

                    if (!charCode || charCode == 8) {
                        return;
                    }
                    var typedChar = String.fromCharCode(charCode);
                    if (/\d/.test(typedChar)) {
                        return;
                    }
                    if (typedChar == "-" && this.value == "") {
                        return;
                    }
                    return false;
                };
                count[index].appendChild(input);

                if ($(count[index]).find('.check-winner').length === 0) {
                    let check_winner_icon  = document.createElement("label");
                    check_winner_icon.classList.add("check-winner-icon");
                    check_winner_icon.setAttribute('for','check-winner-'+
count[index].childNodes[1].value);
                    check_winner_icon.innerHTML = "♛";
                    check_winner_icon.classList.add("no-winner");
                    check_winner_icon.onclick = function (event) {
                        ChangeWinner(this);
                    };

                    let check_winner = document.createElement("input");
```

```
                    check_winner.setAttribute('type', 'checkbox');
                    check_winner.setAttribute('name', 'winners[]');
                    check_winner.setAttribute('id','check-winner-'+
count[index].childNodes[1].value);
                    check_winner.value = count[index].childNodes[1].value;
                    check_winner.classList.add("check-winner");
                    check_winner.classList.add("hide");

                    count[index].prepend(check_winner_icon);
                    count[index].prepend(check_winner);
                }
            }



            $("#coop-player-block").remove();
        }
        if ($(this).data("type") === "coop") {
            ChangeToCoop(count);
        }
        if ($(this).data("type") === "team") {
            $("#coop-player-block").remove();
            ChangeToTeam();
        }
        lastType = $(this).val();
    });

    $("#collection-select").on('change',function (e) {
        let selected_game = this.value;
        var result = collection.filter(obj => {
            return obj.game_id == selected_game
        })
        document.getElementById('selected-game-image').setAttribute('src',
result[0].game.image);

      if( result[0].type === "coop" ||  result[0].type === "team"){
        $("input[type=radio][name=type][value='" + result[0].type + "']").click();
      }
      if( result[0].type === "default")
      {
        $("input[type=radio][name=type][value=default][data-type='" +
result[0].points_directed + "']").click();
      }
    });

});

window.ChangeToTeam = function ChangeToTeam(){

        $(".players_checkbox").prop("ckecked", false);

        if (document.getElementsByClassName("save-match")[0] != null)
            document.getElementsByClassName("save-match")[0].id =
                "save-team";
        if (document.getElementsByClassName("update-match")[0] != null)
            document.getElementsByClassName("update-match")[0].id =
                "save-team";

            document.getElementById("open-players").remove();
        document.getElementById("players-list").innerHTML = "";
        $(".players_checkbox").attr("name", "chosen_team_players");
        let block = document.getElementById("players-list");
        block.innerHTML = null;
        let team_list_description = document.createElement("span");
```

```
        team_list_description.innerHTML = team_list;
        team_list_description.classList.add("match-descriptions");

        let team_list = document.createElement("div");
        team_list.id = "team-list";

        let team_add = document.createElement("div");
        team_add.id = "add-team";
        team_add.addEventListener("click", function(){
            AddTeam();
         });
        team_add.innerHTML = "locale_add_team";//FIXME:

        block.appendChild(team_list_description);
        block.appendChild(team_list);
        block.appendChild(team_add);
    }

    window.ChangeToCoop = function ChangeToCoop(){

        $(".players_checkbox").attr("name", "chosen_players");
        if (lastType == "team") {
            CreatePlayerBlock();
            $(".players_checkbox").prop("checked", false);
        }
        $("#coop-player-block").remove();
        $(".score-input").remove();
        $(".check-winner").remove();
        $(".check-winner-icon").remove();

        if (document.getElementsByClassName("save-match")[0] != null)
            document.getElementsByClassName("save-match")[0].id =
                "save-coop";
        if (document.getElementsByClassName("update-match")[0] != null)
            document.getElementsByClassName("update-match")[0].id =
                "save-coop";

        let block = document.getElementById("player-block");
        let coop_block = document.createElement("div");
        coop_block.id = "coop-player-block";

        let circumstances_description = document.createElement("span");
        circumstances_description.innerHTML = "locale_circumstances";//FIXME:
        circumstances_description.classList.add("match-descriptions");

        let circumstances_block = document.createElement("div");
        circumstances_block.classList.add("circumstances_block");

        let circumstances = document.createElement("input");
        circumstances.id = "circumstances";

circumstances.setAttribute("placeholder","locale_circumstances_name");//FIXME:
        circumstances.setAttribute("type", "text");
        circumstances.setAttribute('name','circumstances');
        circumstances_block.appendChild(circumstances);

        let result_description = document.createElement("span");
        result_description.innerHTML = "locale_result"; //FIXME:
        result_description.classList.add("match-descriptions");

        let form_toggle = document.createElement("div");
        form_toggle.classList.add("form_toggle");

        let toggle_item_1 = document.createElement("div");
```

```
    toggle_item_1.classList.add("form_toggle-item");
    toggle_item_1.classList.add("item-1");

    let lose = document.createElement("input");
    lose.id = "fid-1";
    lose.setAttribute("type", "radio");
    lose.setAttribute("name", "victory");
    lose.setAttribute("value", "false");
    lose.setAttribute("checked", "checked");

    let lose_label = document.createElement("label");
    lose_label.id = "fid-1";
    lose_label.setAttribute("for", "fid-1");
    lose_label.innerHTML = "locale_lose";//FIXME:

    let toggle_item_2 = document.createElement("div");
    toggle_item_2.classList.add("form_toggle-item");
    toggle_item_2.classList.add("item-2");

    let victory = document.createElement("input");
    victory.id = "fid-2";
    victory.setAttribute("type", "radio");
    victory.setAttribute("name", "victory");
    victory.setAttribute("value", "true");

    let victory_label = document.createElement("label");
    victory_label.id = "fid-2";
    victory_label.setAttribute("for", "fid-2");
    victory_label.innerHTML = "locale_win";//FIXME:

    form_toggle.appendChild(toggle_item_1);
    toggle_item_1.appendChild(lose);
    toggle_item_1.appendChild(lose_label);
    form_toggle.appendChild(toggle_item_2);
    toggle_item_2.appendChild(victory);
    toggle_item_2.appendChild(victory_label);
    coop_block.appendChild(circumstances_description);
    coop_block.appendChild(circumstances_block);
    coop_block.appendChild(result_description);
    coop_block.appendChild(form_toggle);

    block.prepend(coop_block);
}

function CreatePlayerBlock() {
    document.getElementById("add-team").remove();
    document.getElementById("team-list").remove();

    let block = document.getElementById("player-block");

    let players_block = document.getElementById("players-list");
    players_block.innerHTML = null;

    let player_list_desc = document.createElement("span");
    player_list_desc.innerHTML = "locale_player_list";//FIXME:
    player_list_desc.classList.add("match-descriptions");

    let player_block = document.createElement("div");
    player_block.id = "player-default-list";

    let add_player = document.createElement("input");
    add_player.className = "btn btn-success";
    add_player.id = "open-players";
    add_player.setAttribute('value','adddddddddddd');
```

```
        add_player.setAttribute('type','button');
        add_player.onclick = function (event) {
            document.getElementById('save-match-form').classList.add('hide');
            document.getElementById('forum-block').classList.add('hide');
            document.getElementById('chose-players-block').classList.remove('hide');
        };
        add_player.innerHTML = "locale_add_player";//FIXME:

        players_block.appendChild(player_list_desc);
        players_block.appendChild(player_block);
        players_block.appendChild(add_player);

        block.prepend(players_block);
        if (lastType === "coop") {
            $(".players_checkbox").attr("name", "chosen_players");
            if (lastType == "team") {
                CreatePlayerBlock();
                $(".players_checkbox").prop("checked", false);
            }
            $("#coop-player-block").remove();
            $(".score-input").remove();
            $(".check-winner").remove();
            document.getElementsByClassName("save-form")[0].id = "save-coop-form";
            if (document.getElementsByClassName("save-match")[0] != null)
                document.getElementsByClassName("save-match")[0].id = "save-coop";
            if (document.getElementsByClassName("update-match")[0] != null)
                document.getElementsByClassName("update-match")[0].id = "save-coop";

            let block = document.getElementById("no-team");
            let coop_block = document.createElement("div");
            coop_block.id = "coop-player-block";

            let circumstances_description = document.createElement("span");
            circumstances_description.innerHTML = "locale_circumstances";//FIXME:
            circumstances_description.classList.add("match-descriptions");

            let circumstances_block = document.createElement("div");
            circumstances_block.classList.add("circumstances_block");

            let circumstances = document.createElement("input");
            circumstances.id = "circumstances";
            circumstances.setAttribute("placeholder", "locale_circumstances_name");
//FIXME:
            circumstances.setAttribute("type", "text");
            circumstances_block.appendChild(circumstances);

            let result_description = document.createElement("span");
            result_description.innerHTML = "locale_result";//FIXME:
            result_description.classList.add("match-descriptions");

            let form_toggle = document.createElement("div");
            form_toggle.classList.add("form_toggle");

            let toggle_item_1 = document.createElement("div");
            toggle_item_1.classList.add("form_toggle-item");
            toggle_item_1.classList.add("item-1");

            let lose = document.createElement("input");
            lose.id = "fid-1";
            lose.setAttribute("type", "radio");
            lose.setAttribute("name", "victory");
            lose.setAttribute("value", "false");
            lose.setAttribute("checked", "checked");
```

```
            let lose_label = document.createElement("label");
            lose_label.id = "fid-1";
            lose_label.setAttribute("for", "fid-1");
            lose_label.innerHTML = "locale_lose";//FIXME:

            let toggle_item_2 = document.createElement("div");
            toggle_item_2.classList.add("form_toggle-item");
            toggle_item_2.classList.add("item-2");

            let victory = document.createElement("input");
            victory.id = "fid-2";
            victory.setAttribute("type", "radio");
            victory.setAttribute("name", "victory");
            victory.setAttribute("value", "true");

            let victory_label = document.createElement("label");
            victory_label.id = "fid-2";
            victory_label.setAttribute("for", "fid-2");
            victory_label.innerHTML = "locale_win";//FIXME:

            form_toggle.appendChild(toggle_item_1);
            toggle_item_1.appendChild(lose);
            toggle_item_1.appendChild(lose_label);
            form_toggle.appendChild(toggle_item_2);
            toggle_item_2.appendChild(victory);
            toggle_item_2.appendChild(victory_label);
            coop_block.appendChild(circumstances_description);
            coop_block.appendChild(circumstances_block);
            coop_block.appendChild(result_description);
            coop_block.appendChild(form_toggle);

            players_block.prepend(coop_block);
            block.prepend(players_block);
        }
    }

    window.GetCollection = function GetCollection(collection_list){
        collection = collection_list;
    }

let player_id = null;
let group_id = null;

//locale variables
let locale_confirm = "null";
let locale_remove_player_annotation = "null";
let locale_yes_player_game = "null";
let locale_cancel = "null";
let locale_points_plays_count = "null";
let locale_points_wins_count = "null";
let locale_points_loses_count = "null";
let locale_points_percent = "null";
//--------
let players = [];

$(document).ready(function () {
    $(".tablinks").on("click", function (e) {
        content = document.getElementsByClassName("content");
        otherTabs = document.getElementsByClassName("tablinks");
        for (let index = 0; index < content.length; index++) {
            content[index].classList.add("hide");
        }
        for (let index = 0; index < otherTabs.length; index++) {
            otherTabs[index].classList.remove("activeTab");
```

```
    }
    this.classList.add("activeTab");
    document.getElementsByClassName(this.id)[0].classList.remove("hide");
});

$('.open-settings').on('click', function(){
    let parent_element = this.parentElement.parentElement;
    parent_element.children[0].classList.toggle('hide');
    parent_element.children[1].classList.toggle('hide');
    $(parent_element).find(".play-settings").toggleClass('hide')
    parent_element.classList.toggle('padding');
});

$(".group-name").on("click", function (e) {
    SingleBlockSelect(this.dataset.id);
});

$(".remove-group").on("click", function (e) {
    e.preventDefault();
    let deleteUrl = $(this).attr("href");
    Swal.fire({
        title: "remove группе?",
        text: "Ю шуре?",
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#3085d6",
        cancelButtonColor: "#d33",
        confirmButtonText: "ЕС",
        cancelButtonText: "КАНСЕЛ",
    }).then((result) => {
        if (result.isConfirmed) {
            let form = document.createElement("form");
            form.setAttribute("method", "post");
            form.setAttribute("action", deleteUrl);

            let csrfField = document.createElement("input");
            csrfField.setAttribute("type", "hidden");
            csrfField.setAttribute("name", "_token");
            csrfField.setAttribute(
                "value",
                $('meta[name="csrf-token"]').attr("content")
            );
            form.appendChild(csrfField);

            let methodField = document.createElement("input");
            methodField.setAttribute("type", "hidden");
            methodField.setAttribute("name", "_method");
            methodField.setAttribute("value", "DELETE");
            form.appendChild(methodField);

            document.body.appendChild(form);
            form.submit();
        }
        if (result.cancel) {
            return false;
        }
    });
});


$(".rename-open-block").on("click", function (e) {
    e.preventDefault();
    let parent = this.parentElement;
    $('.group-single-block').removeClass('hide');
```

```
    if (document.getElementById("rename-block") != null) {
        document.getElementById(
            "rename-block"
        ).previousElementSibling.style.display = "block";
        document.getElementById("rename-block").remove();
    }

    parent.classList.add("hide");

    let rename_block = document.createElement("div");
    rename_block.id = "rename-block";

    let cancel = document.createElement("button");
    cancel.innerHTML = lang.get('players.back');
    cancel.onclick = function (e) {
        document.getElementById("rename-block").remove();
    };

    let form = document.createElement("form");
    form.setAttribute("method", "post");
    form.setAttribute("action", $(this).attr("href"));

    let csrfField = document.createElement("input");
    csrfField.setAttribute("type", "hidden");
    csrfField.setAttribute("name", "_token");
    csrfField.setAttribute(
        "value",
        $('meta[name="csrf-token"]').attr("content")
    );
    form.appendChild(csrfField);

    let methodField = document.createElement("input");
    methodField.setAttribute("type", "hidden");
    methodField.setAttribute("name", "_method");
    methodField.setAttribute("value", "PATCH");
    form.appendChild(methodField);

    let input = document.createElement("input");
    input.setAttribute("type", "text");
    input.setAttribute("name", "name");
    input.setAttribute("value", this.dataset.group);
    form.appendChild(input);

    let submit = document.createElement("button");
    submit.setAttribute("type", "submit");
    submit.innerHTML = lang.get('players.update');
    form.appendChild(submit);
    form.appendChild(cancel);

    rename_block.appendChild(form);


    parent.parentElement.appendChild(rename_block);
});

$(".add-to-group").on("click", function (e) {
    let token = $('meta[name="csrf-token"]').attr("content");

    if (group_id != null) {
        $.ajax({
            url: "/change-group-players",
            type: "POST",
            data: {
                _token: token,
```

```
                player_id: this.value,
                group_id: group_id,
            },
            success: function (data) {
                GetPlayersGroup(data.players);
            },
            error: function (data) {},
        });
    }
});

$(".delete-player").on("click", function (e) {
    e.preventDefault();

    let getPlayers = players;
    getPlayers.forEach(object => {
        delete object['friend_id'];
        delete object['default'];
        delete object['created_at'];
        delete object['image'];
        delete object['updated_at'];
        delete object['user'];
        delete object['user_id'];
    });

    var options = {};
    $.map(getPlayers,
        function(o) {
            options[o.id] = o.name;
        });

    let deleteUrl = $(this).attr("href");
    Swal.fire({
        title: locale_confirm,
        // input: 'select',
        // inputOptions: options,
        // inputPlaceholder: 'Чузе player ту гет олле гаймес',
        text: locale_remove_player_annotation,
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#3085d6",
        cancelButtonColor: "#d33",
        confirmButtonText: locale_yes_player_game,
        cancelButtonText: locale_cancel,
    }).then((result) => {
        if (result.isConfirmed) {
            let form = document.createElement("form");
            form.setAttribute("method", "post");
            form.setAttribute("action", deleteUrl);

            let csrfField = document.createElement("input");
            csrfField.setAttribute("type", "hidden");
            csrfField.setAttribute("name", "_token");
            csrfField.setAttribute(
                "value",
                $('meta[name="csrf-token"]').attr("content")
            );
            form.appendChild(csrfField);

            let methodField = document.createElement("input");
            methodField.setAttribute("type", "hidden");
            methodField.setAttribute("name", "_method");
            methodField.setAttribute("value", "DELETE");
            form.appendChild(methodField);
```

```
                // if(result.value){
                //     let playerField = document.createElement("input");
                //     playerField.setAttribute("type", "hidden");
                //     playerField.setAttribute("name", "player");
                //     playerField.value = result.value;
                //     form.appendChild(playerField);
                // }


                document.body.appendChild(form);
                form.submit();
            }
            if (result.cancel) {
                return false;
            }
        });
    });

    $("#new-player-input").on("keyup", function () {
        document.getElementById("player-name-preview").innerText =
            $(this).val();
    });

    $("#file").on("change", function () {
        if (this.files && this.files[0]) {
            var reader = new FileReader();
            reader.onload = function (e) {
                $("#player-image-preview").attr("src", e.target.result);
            };
            reader.readAsDataURL(this.files[0]);
            document
                .getElementById("player-image-preview")
                .classList.remove("hide");
        }
    });

    $("#player-search").on("keyup", function () {
        let list = document.getElementsByClassName("player-name");
        for (var i = 0; i < list.length; i++) {
            if (list[i].innerHTML.indexOf($(this).val()) !== -1) {

list[i].parentElement.parentElement.parentElement.classList.remove("hide");
            } else {

list[i].parentElement.parentElement.parentElement.classList.add("hide");
            }
        }
    });
    $("#get-all-player-place").on("click", function (e) {
        document
            .getElementsByClassName("all-game-plays")[0]
            .classList.remove("hide");
        document.getElementsByClassName("fresh-info")[0].classList.add("hide");
    });
    $('[data-toggle="tooltip"]').tooltip();
    $("#close-all-player-place").on("click", function (e) {
        document
            .getElementsByClassName("all-game-plays")[0]
            .classList.add("hide");
        document
            .getElementsByClassName("fresh-info")[0]
            .classList.remove("hide");
    });
```

```
    $("#add-player-show").on("click", function () {
        document.getElementsByClassName("add-player-
block")[0].classList.remove("hide");
        $('.players-list').addClass('hide');
    });

    $(".rename-player-show").on("click", function () {
        document.getElementsByClassName("rename-player-
block")[0].classList.remove("hide");
        $('.players-list').addClass('hide');
        document.getElementById("rename-player-id").value = this.dataset.id;
        document.getElementById("rename-user-name").value = this.dataset.user;
        document.getElementById("rename-player-input").value = this.dataset.username;
        document.getElementById("player-update-form").setAttribute(
                "action",
                "http://tabletopstats/players/" + this.dataset.id
            );
    });

    $("#back-to-player").on("click", function () {
        document.getElementsByClassName("add-player-block")[0].classList.add("hide");
        $('.players-list').removeClass('hide');
    });

    $("#back-to-player-rename").on("click", function () {
        document
            .getElementsByClassName("rename-player-block")[0]
            .classList.add("hide");
        $('.players-list').removeClass('hide');
    });

    $("#player-form").on("submit", function () {
        if (document.getElementById("add-player-input").value === "")
            return false;
    });
});
window.Setlocale = function (locale_data) {
    locale_confirm = locale_data[0];
    locale_remove_player_annotation = locale_data[1];
    locale_yes_player_game = locale_data[2];
    locale_cancel = locale_data[3];
    locale_points_plays_count = locale_data[4];
    locale_points_wins_count = locale_data[5];
    locale_points_loses_count = locale_data[6];
    locale_points_percent = locale_data[7];
};

function GetPlayersGroup(players) {
    let all_players = document.getElementsByClassName("player-no-added");

    $(all_players).each(function () {
        this.classList.remove("player-added");
    });

    for (let i = 0; i < all_players.length; i++) {
        for (let j = 0; j < players.length; j++) {
            if (all_players[i].value == players[j])
                all_players[i].classList.add("player-added");
        }
    }
}


function SingleBlockSelect(id) {
```

```
    group_id = id;
    let token = $('meta[name="csrf-token"]').attr("content");
    $.ajax({
        url: "/get-group-players",
        type: "POST",
        data: {
            _token: token,
            id: id,
        },
        success: function (data) {
            GetPlayersGroup(data.players);
            $('.add-to-group').prop( "disabled", false );
        },
        error: function (data) {},
    });
}

window.chart = function (playersCount, mainStats, statsByLocation) {
    let dates = [];
    let plays_c = [];

    let colors = [
        "rgb(255, 99, 132,0.5)",
        "rgb(201, 20, 111,0.5)",
        "rgb(1, 200, 200,0.5)",
        "rgb(1, 200, 111,0.5)",
        "rgb(255, 255, 25,0.5)",
        "rgb(255, 0, 255,0.5)",
        "rgb(5, 5, 255,0.5)",
        "rgb(55, 255, 55,0.5)",
        "rgb(255, 150, 1,0.5)",
        "rgb(100, 1, 1,0.5)",
        "rgb(66, 6, 120,0.5)",
    ];
    shuffle(colors);
    var ctx5 = document.getElementById("days-stats-chart").getContext("2d");

    var myChart5 = new Chart(ctx5, {
        type: "pie",
        data: {
            labels: Object.keys(playersCount),
            datasets: [
                {
                    backgroundColor: colors.slice(0, playersCount.length),
                    borderColor: "rgb(255, 255, 255,0.5)",
                    data: Object.values(playersCount),
                },
            ],
        },
        options: {
            legend: {
                display: true,
                position: "bottom",
            },
            tooltips: {
                enabled: true,
            },
            scales: {
                xAxes: [
                    {
                        gridLines: {
                            display: false,
                        },
                        ticks: {
```

```
                        display: false,
                    },
                },
            ],
            yAxes: [
                {
                    gridLines: {
                        color: "rgba(77, 77, 77,0)",
                    },
                    ticks: {
                        display: false,
                    },
                },
            ],
        },
    },
});

var ctx3 = document.getElementById("scores-statistics-chart");
var myChart3 = new Chart(ctx3, {
    type: "bar",
    data: {
        labels: [
            locale_points_plays_count,
            locale_points_wins_count,
            locale_points_loses_count,
            locale_points_percent,
        ],
        datasets: [
            {
                data: [
                    mainStats["plays"],
                    mainStats["wins"],
                    mainStats["loses"],
                    mainStats["percent"],
                ],
                fillColor: [
                    "rgba(220,220,220,0.5)",
                    "navy",
                    "red",
                    "orange",
                ],
                backgroundColor: [
                    "rgba(255, 99, 132,0.2)",
                    "rgba(255, 159, 64, 0.2)",
                    "rgba(255, 205, 86, 0.2)",
                    "rgba(75, 192, 192, 0.2)",
                    "rgba(54, 162, 235, 0.2)",
                    "rgba(153, 102, 255, 0.2)",
                ],

                borderColor: [
                    "rgba(255, 99, 132)",
                    "rgba(255, 159, 64)",
                    "rgba(255, 205, 86)",
                    "rgba(75, 192, 192)",
                    "rgba(54, 162, 235)",
                    "rgba(153, 102, 255)",
                ],
                borderWidth: 1,
                datalabels: {
                    color: "white",
                    anchor: "end",
                    align: "bottom",
```

```
                    font: {
                        weight: "bold",
                        size: 20,
                    },
                },
            },
        ],
    },
    plugins: [ChartDataLabels],
    options: {
        legend: {
            display: false,
        },

        tooltips: {
            titleFontSize: 17,
            bodyFontSize: 20,
        },
        scales: {
            y: {
                ticks: { display: false },
                beginAtZero: true,
            },
            x: {
                ticks: {
                    display: false,
                },
            },
            yAxes: [
                {
                    ticks: {
                        display: false,
                        beginAtZero: true,
                    },
                    gridLines: {
                        display: false,
                    },
                },
            ],
            xAxes: [
                {
                    ticks: {
                        display: false,
                    },
                    gridLines: {
                        display: false,
                    },
                },
            ],
        },
    },
});

shuffle(colors);
var ctx6 = document.getElementById("players-number-chart").getContext("2d");
var myChart6 = new Chart(ctx6, {
    type: "pie",
    data: {
        labels: Object.keys(statsByLocation),
        datasets: [
            {
                backgroundColor: colors.slice(0, statsByLocation.length),
                borderColor: "rgb(255, 255, 255,0.5)",
                data: Object.values(statsByLocation),
```

```
                },
            ],
        },
        options: {
            legend: {
                display: true,
                position: "bottom",
            },
            tooltips: {
                enabled: true,
            },
            scales: {
                xAxes: [
                    {
                        gridLines: {
                            display: false,
                        },
                        ticks: {
                            display: false,
                        },
                    },
                ],
                yAxes: [
                    {
                        gridLines: {
                            color: "rgba(77, 77, 77,0)",
                        },
                        ticks: {
                            display: false,
                        },
                    },
                ],
            },
        },
    });
};

function shuffle(array) {
    array.sort(() => Math.random() - 0.5);
}
window.FillDefaultPlayersStats = function (gamesStats) {
    let table = document.getElementById("points-table-stats");

    for (let index = 0; index < gamesStats.length; index++) {
        let tr = document.createElement("tr");
        let plays_count = 0;
        plays_count = gamesStats[index]["win"] + gamesStats[index]["lose"];
        let name = document.createElement("td");
        name.innerHTML = gamesStats[index]["name"];
        let wins = document.createElement("td");
        wins.innerHTML = gamesStats[index]["win"];
        let loses = document.createElement("td");
        loses.innerHTML = gamesStats[index]["lose"];
        let plays = document.createElement("td");
        plays.innerHTML = plays_count;
        let percent = document.createElement("td");
        percent.innerHTML = gamesStats[index]["percent"] + "%";

        tr.style.color = getColor(gamesStats[index]["percent"] / 100);

        tr.appendChild(name);
        tr.appendChild(plays);
        tr.appendChild(wins);
        tr.appendChild(loses);
```

```
        tr.appendChild(percent);
        table.appendChild(tr);
    }
};

window.getPlayers = function getPlayers(players_list){
    players = players_list;
};
```

ДОДАТОК В

(Обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Тема дипломного проекту: Інтернет-платформа для ведення статистики настільних ігор

Розробив:
Студент групи ІПЗс-19-1
Нетреба І.В.
Керівник роботи:
К. П. Н., Доцент Праворська Н.І.

Мета та завдання дослідження

- Метою проекту є проектування та створення інтернет-платформи для ведення статистики настільних ігор шляхом створення веб-ресурсу, яка б вирішувала проблеми ефективного ведення особистої статистики.

Навіщо вести статистику?

- Актуальність теми полягає в тому що немає великої кількості аналогів, гравцеві потрібна особиста статистика інформація для аналізу своїх партій в тій чи іншій ігри, в першу чергу це стосується тих гравців які приймають участь в турнірах і моніторити свої досягнення (наприклад, за допомоги деяких сайтів чи додатків) на основі яких можна робити висновки для поліпшення своїх особистих показників, що в свою чергу приведе до поліпшення загальної статистики. Проблему можна вирішити шляхом створення веб-ресурсу чи мобільного додатка, яка дає можливість реалізації проєкта.

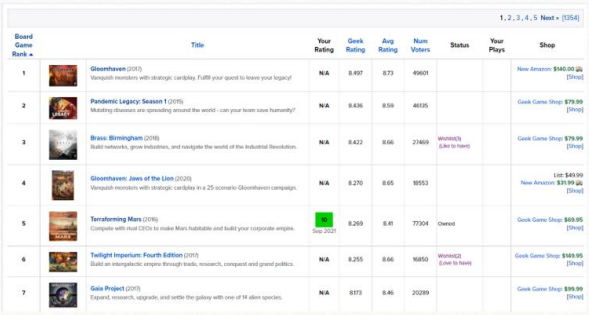
Завдання дослідження

Завдання які потрібно реалізувати:

- дослідити предметну область, з особливостями її реалізації і функціоналу
- дослідити фреймворк Laravel який буде слугувати базою для створення додатку;
- зробити аналіз існуючих рішень на ринку, визначити їх переваги та недоліки і мінімальний функціонал;
- розробити систему, яка буде задовольняти вимоги користувача;
- провести тестування системи.
- надати максимальну кількість різноманітної статистики з малої кількості вхідних данних
- зменшити до мінімуму кількість рутинної роботи на сайті
- Зробити додаткові функції які б збільшили відвіданість платформи

Наявне програмне забезпечення

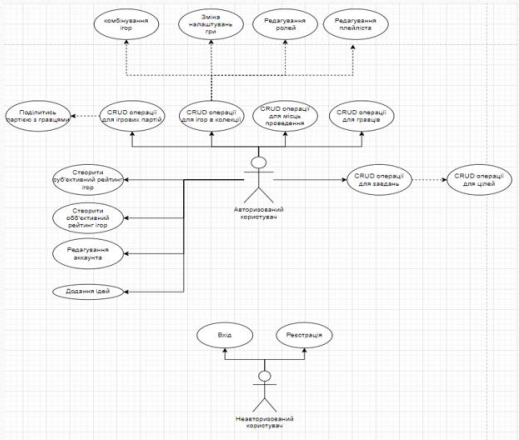
BoardGameGeek - онлайн форум який існує з 2000 року для любителів настільних ігор і база даний ігор, яка містить різні огляди, відео та зображення для понад 139 000 різних настільних ігор, сюди входять як ігри європейського стиля, так і американського, та карткові ігри



- Наступний і найкращий серед мобільних аналогів це додаток BGStats, перше що потрібно сказати це те, що додаток не безкоштовний. Випущений в 2017 році має оцінку в 4.8 з 5 і більше ніж 15 тисяч завантажень.

Аналіз вимог до програмного забезпечення

- Діаграмма взаємодії за допомогою UML
- Авторизований користувач- може виконувати операції з колекцією, гравцями, місцями, партіями, а також може створювати рейтинги ігор, ділитися партіями з іншими зареєстованими користувачами.
- Неавторизований користувач - немає ніякий прав і не може здійснювати операції в додатку крім авторизації і реєстрації з подальшою аторизацією.

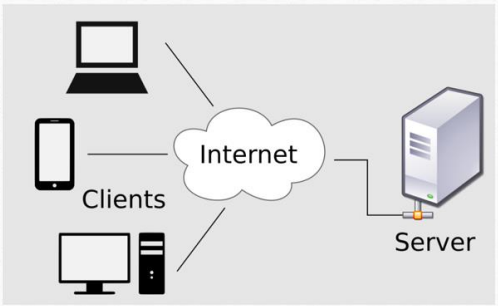


Аналіз та вибір архітектури систем

Оскільки майбутне програмне забезпечення являє у сутності своїй сайт з інтерактивом, то рішення по вибору архітектури не заставляє себе довго чекати і обирається клієнт-серверна архітектура на якій побудовані усі сайти і інтернет-сервіси. Також її використовуються і десктоп-додатки, які передають дані через інтернет.

Складові:

- Клієнт
- Сервер



Опис структури та моделі бази даних

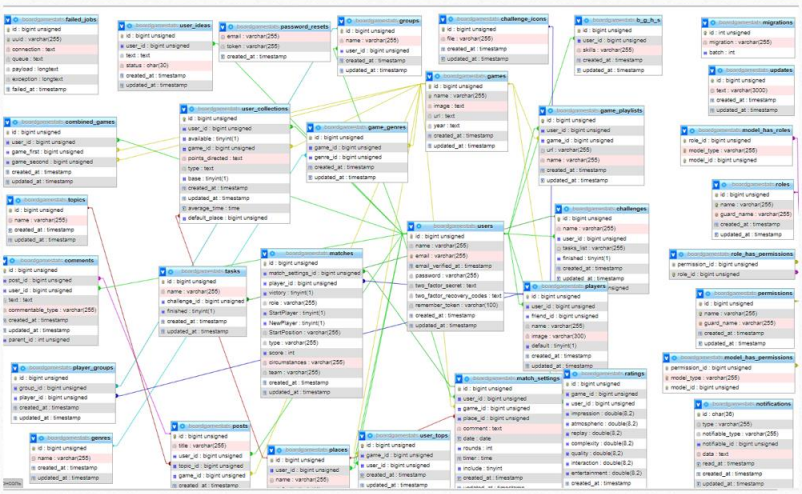
База даних – інтегрована сукупність структурованих і взаємозалежних даних, організована за певними правилами, які передбачають загальні принципи опису, зберігання і обробки даних.

Переваги:

- простота і доступність для розуміння користувачем;
- повна незалежність даних;
- для організації запитів і написання прикладного ПЗ немає необхідності знати конкретну організацію БД у зовнішній пам'яті;
- суворі правила проектування.

Недоліки:

- далеко не завжди предметна область може бути представлена у вигляді «таблиць»;
- в результаті логічного проектування з'являється множина «таблиць»;
- БД займає відносно багато зовнішньої пам'яті;
- відносно низька швидкість доступу до даних.



Створення макету веб-додатку та дизайн

Макет веб-додатку – це його реалістичний прототип, створений в графічній програмі на кшталт Photoshop

Аналіз та вибір технологій і методів реалізації системи

- Основу інтернет-платформи буде складати фреймворк Laravel – безкоштовний веб-фреймворк із відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC, фреймворк має достатню кількість переваг.
- Також буде використовуватись SCSS – "діалект" мови SASS. SASS це мова схожа на HAML (дуже лаконічний шаблонізатор), але призначена для спрощення створення CSS-коду.
- JasvaScript та jQuery
- PHP

Розробка бази даних

Міграції – щось подібне до системи контролю версій для нашої бази даних. Вони дозволяють команді розробників змінювати структуру бази даних, водночас залишаючись у курсі змін інших учасників.

```php
public function up()
{
    Schema::create('match_settings', function (Blueprint $table) {
        $table->id();
        $table->foreignId("user_id")->constrained()->onDelete("cascade")->onUpdate("cascade");
        $table->foreignId("game_id")->constrained()->onDelete("cascade")->onUpdate("cascade");
        $table->foreignId("place_id")->constrained()->onDelete("cascade")->onUpdate("cascade");
        $table->text("comment",20000)->nullable();
        $table->date("date");
        $table->integer("rounds")->nullable();
        $table->time("time")->nullable();
        $table->boolean("include")->nullable();
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('match_settings');
}
```

Розробка серверної частини Інтернет-платформи

Структура проекту наступна:

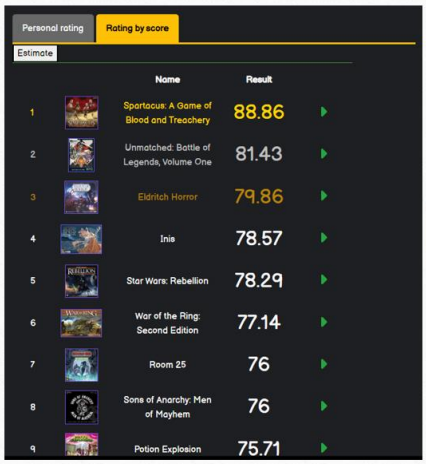
- Database
- Public
- Resources
- Routes
- Tests
- Vendor
- Bootstrap
- App

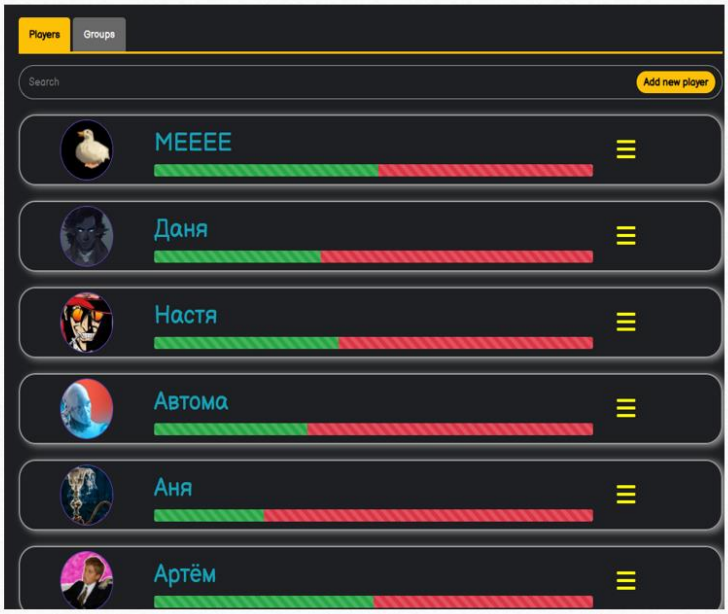
Контроллер відповідальний за особистий рейтинг користувача

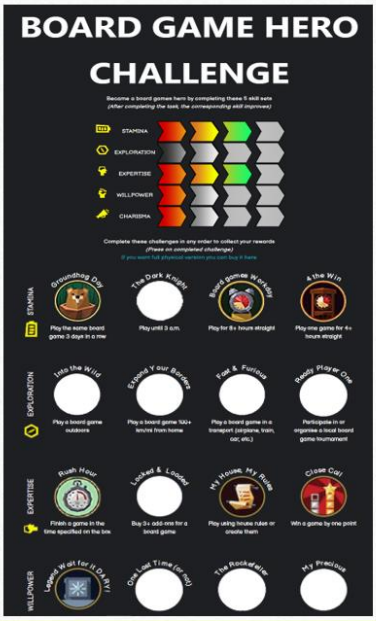
```php
public function changeRatting(Request $request, $id)
{

    $rating = new Rating();
    $rating->saveRating($request, $id);
    if($request->ajax()){
        return response()->json(['games'=>$rating->getRatedGame()]);
    }
    return back();
}

public function getUserTop()
{
    $rating = new Rating();
    $games = $rating->getRatedGame();
    $user_all_games = UserCollection::where("user_id",Auth::id())
        ->where('base',1)
        ->pluck("game_id");
    $all_games = Game::whereIn("id",$user_all_games)->orderBy("name", "asc")->get();
    $user_subjectively_games = UserTop::where("user_id",Auth::id())->get();
    return view('functions.user-top',['games'=>$games,'user_games'=>$all_games,'user_top'=>$user_subjectively_games]);
}
```







Висновок

Отже, результатом виконання дипломного проекту є Інтернет-платформа для ведення статистики настільних ігор , яка поступово розроблялась в процесі виконання індивідуального завдання спершу був проведений аналізу ринку і наявного програмного забезпечення було вирішено що аналогів серед веб-додатків досить мала і конкуренції поки що не так багато, тому і платформа розроблюється у вигляді сайту.

Також в процесі аналізу рішень було складено ряд вимог, якими повинна володіти система та необхідний функціонал, щоб задовільнити потреби користувача.

В основі створення лежала мета розробити веб-додаток, який дозволить кожному бажаючому вести власну статистику ігор без зайвих витрат коштів і для економії часу, може використовуватись для задоволення усіх бажаючих, з наявністю переваг та унікальних функцій, яких немає в аналогах.

Tue May 24 18:02:32 EEST 2022, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 17.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

ID: 103897 Назва: Інтернет-платформа для ведення статистики настільних ігор Додано в БД: 2022-05-24 Автора: І. В. Нетреба Керівники: Н. І. Праворська Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	60313	543	15572 (26%)	152 (28%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
103263	Назва: Зіт з переддипломної практики Додано в БД: 2022-05-04 Автора: І.В. Нетреба Керівники: Праворська Н. І. Консультанти: Опоненти:	10264 (17.0%)	92 (17.0%)

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1011323559

Дата перевірки:
24.05.2022 18:55:17 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
24.05.2022 18:55:33 EEST

ID користувача:
100005589

Назва документа: **Диплом_Нетреба_ІПЗс-19-1(без кода)**

Кількість сторінок: **68** Кількість слів: **10069** Кількість символів: **79871** Розмір файлу: **2.39 MB** ID файлу: **1011209964**

11.9%
Схожість

Найбільша схожість: **8.29%** з джерелом з Бібліотеки (ID файлу: **1008215667**)

2.89% Джерела з Інтернету | 106 ..... Сторінка 70

10.2% Джерела з Бібліотеки | 111 ..... Сторінка 71

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 4

ВІДГУК

керівника дипломної проекту

на здобуття ступеня бакалавра

студента групи ІПЗс-19-1 Нетреби І.В.

на тему: «Інтернет-платформа для ведення статистики настільних ігор»

Представлений дипломний проект присвячений розробці інтернет-платформи для ведення статистики настільних ігор. В існуючих на сьогоднішній день аналогах виникає потреба в розширені різноманітної статистики, адаптації до нових ринків, зручному і сучасному дизайні. Тому вони, забезпечують лише базові потреби користувача. З огляду на це, дипломний проект характеризується актуальністю і своєчасністю.

У представленій роботі було розглянуто існуючі рішення, складена їх характеристика, проведено дослідження предметної області з виділенням нефункціональних та функціональних вимог до розроблювальної платформи. Було обрано архітектуру веб-додатку, описано структуру та моделі бази даних, спроектовано серверну частину та інтерфейс користувача, проведено аналіз та вибір технологій і методів реалізації системи. Здійснено програмну реалізацію, розробка: бази даних, серверної частини, керівництва користувача і проведено тестування платформи.

Позитивними рисами дипломного проекту є системність, структурованість та послідовність викладення матеріалу. Робота повною мірою розкрита, відповідає поставленій цілі та всім вимогам, що були поставлені перед студентом. Під час виконання дипломного проекту студент Нетреба І.В. проявив себе грамотним, кваліфікованим спеціалістом здатним приймати самостійно складні технічні рішення.

Дипломна робота виконана на високому науково-технічному рівні та заслуговує на оцінку «відмінно», а Нетреба І. В. – присвоєння кваліфікації «бакалавр» з інженерії програмного забезпечення.

Керівник: к.п.н., доц.  Н. І. Праворська

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Нетреби І.В.
Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-19-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

22.05.2022
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Інтернет-платформа для ведення статистики настільних ігор»
Автор: Нетреба Ігор Вікторович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Праворська Наталія Іванівна, кандидат педагогічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання), у структурі змісту, назвах розділів/підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 11,9% і адресується до 217 джерел, що, з урахуванням наведених обгрунтувань, відповідає характеру теми і свідчить на користь дипломного проекту.

Керівник _____ Н. І. Праворська

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс-19-1
Нетреби І. В
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему дипломного проекту освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: «Інтернет-платформа для ведення статистики настільних ігор.»

(керівник дипломного проекту – Праворська Наталія Іванівна)
Прізвище, ім'я, по батькові

1.03.2022
Дата

Підпис студента

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Бакалавр»

Дипломник: Нетреба Ігор Вікторович

Тема: Інтернет платформа для ведення статистики настільних ігор

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень _____; кількість сторінок записки 61

1.Короткий зміст пояснювальної записки та прийнятих рішень У дипломному проекті було досліджено і проаналізовано предметну область, усі функціональні та нефункціональні вимоги. Був проведений аналіз існуючих програм на ринку, розглянуто їх переваги і недоліки, та доведено актуальність розробки нового програмного забезпечення. Було розглянуто інструменти для реалізації дипломного проекту, в результаті чого створено програмне забезпечення. Також було проведено тестування програми, за результатами якого доведено, що розроблене програмне забезпечення працює коректно та готове до експуатації.

2. Висновок про відповідність проекту поставленому завданню Дипломний проект виконаний відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, визначено мету та завдання дипломноого проектування. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведено аналіз архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. В четвертому розділі було виконано модульне тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу програми.

4. Позитивні сторони проекту Тематика дипломного проекту є актуальною, оскільки інші аналоги не є достатньо розвинутими та не мають достатньої кількості функціональних можливостей. Також було застосовано унікальні функціональні можливості, яких не має в аналогах

5. Негативні сторони проекту У даній роботі потрібно доповнити інтернет платформу ще більшою інформацією про ігри та оновити вхідне меню з демонстрацією функціональності сайту.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконано відповідно до теми дипломного проекту та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про дипломний проект в цілому Дипломний проект заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики дипломного проекту. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

9. Оцінка дипломного проекту Дипломний проект виконаний у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «відмінно».

РЕЦЕНЗЕНТ Говорущенко Тетяна Олександрівна, доктор технічних наук, професор, зав. кафедри комп'ютерної інженерії та інформаційних систем (КІ)

"1" червня 2022 р. _____
(підпис)