Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Космических и информационных технологий

институт

Вычислительной техники

кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« ___» _____ 2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника

Код и наименование направления

Информационная система клуба настольного тенниса

тема

Руководитель	_____	доцент, канд. физ.-мат. наук	К. В. Коршун
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		М. К. Пройдаков
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	доцент, канд. физ.-мат. наук	К. В. Коршун
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2022

РЕФЕРАТ

Выпускная квалификационная работа по теме «Информационная система клуба настольного тенниса» содержит 52 страницы текстового документа, 17 иллюстраций, 3 приложения, 7 использованных источников.

НАСТОЛЬНЫЙ ТЕННИС, ИГРОК, МАТЧ, ТУРНИР, VUE.JS, LARAVEL, MYSQL, API.

Целью выпускной квалификационной работы является разработка информационной системы клуба настольного тенниса.

Основные задачи:

1. анализ задания на ВКР;
2. выбор инструментов разработки;
3. проектирование и разработка системы.

В ходе работы был определен необходимый функционал системы, проведен анализ существующих решений и анализ подходов разработки. Были выбраны инструменты разработки и спроектирована информационная система.

Результатом работы является разработанная информационная система, отвечающая всем выдвинутым функциональным требованиям.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный рынок приложений переполнен разнообразным набором программ, огромным количеством приложений для самых разных нужд, однако среди всего этого разнообразия не всегда удается найти необходимое.

На сегодняшний день имеется несколько решений для ведения клубов по настольному теннису, однако, среди них нет бесспорного фаворита. Хотелось бы иметь приложение, не перегруженное лишними, не нужными функциями и имеющее функциональный дизайн. Основной функцией данного приложения должна быть возможность вести записи о прошедших мачтах и турнирах. Также, было бы очень удобно в любой момент времени иметь возможность просмотреть результаты прошедших матчей и турниров, видеть список всех игроков клуба и отслеживать их статистику и текущий рейтинг.

Целью текущей бакалаврской работы является разработка информационной системы клуба настольного тенниса.

Задачами текущей бакалаврской работы являются:

- анализ задания на ВКР;
- выбор инструментов разработки;
- проектирование и разработка системы.

1 Анализ предметной области

1.1 Постановка задачи

Необходимо разработать информационную систему клуба настольного тенниса, имеющую следующий функционал:

- обеспечение осуществления записи результатов матчей;
- обеспечение осуществления записи результатов турниров;
- предоставление возможности просмотра в открытом доступе результатов матчей;
- предоставление возможности просмотра в открытом доступе результатов турниров;
- осуществление возможности создания профилей игроков;
- осуществление возможности редактирования профилей игроков
- предоставление возможности просмотра в открытом доступе актуальной информации об игроках;
- осуществление подсчета статистики и рейтинга игроков.

1.2 Анализ существующих решений

Существует несколько продуктов, в той или иной степени имеющих необходимый функционал. Ниже рассмотрим некоторые из найденных мной в интернете.

«Турнирная сетка»

«Турнирная сетка» является приложением на android устройства, размещено оно в Google Play. В приложении вбиваются имена участников турнира, строится турнирная сетка, затем пользователь заносит итоги матчей. На этом функционал приложения заканчивается. Отсутствует рейтинг игроков, невозможно провести обычный матч, только турниры. Нет никакой возможности

посмотреть итоги прошлых турниров у всех, кроме того, кто вел записи на своем устройстве.

Кроме узкого функционала стоит отметить также непрезентабельный и нефункциональный дизайн – разметка элементов крайне неудобная, цветовая гамма не сочетается, а главное, совершенно непонятно, как пользоваться этим приложением. Все эти недостатки также отмечают и пользователи в отзывах, что продемонстрированно ниже на рисунке 1.

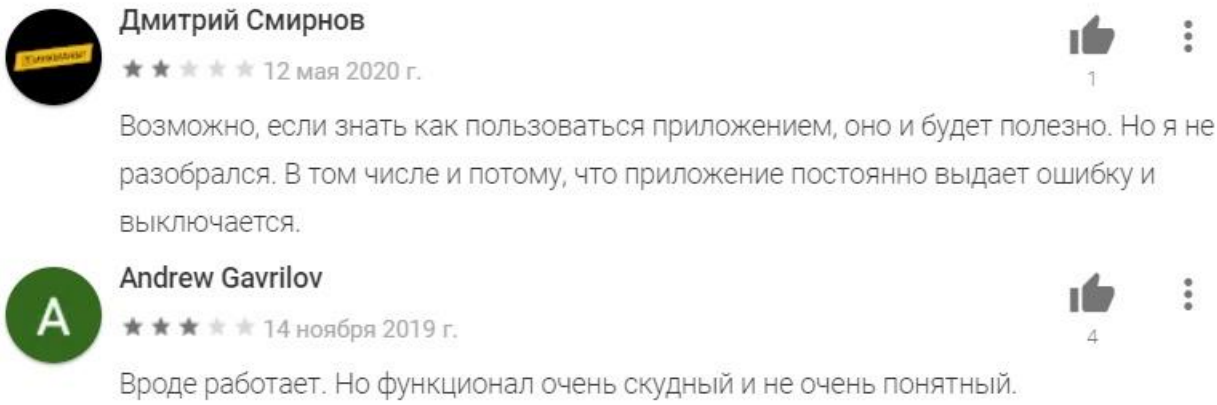

Рисунок 1 – Пользовательские отзывы на «Турнирную сетку»

«Клубный рейтинг»

«Клубный рейтинг» – это программа для организаторов турниров, или для тех, кто занимается расчетом рейтинга по настольному теннису. Подходит для проведения больших и не очень турниров, в целом, справляется со своими обязанностями. Есть дополнительные функции, например, справочник, графики рейтинга, можно также проводить не только классические турниры на выбывание, но и круговые турниры. Однако, для установки и работы программы требуется персональный компьютер на базе операционной системы Windows. Кроме того, «Клубный рейтинг» не работает без программы «1С Предприятие». Главным минусом данной системы является необходимость наличия компьютера для записи данных, что является очень неудобным условием для организаторов и судей, полностью при отсутствии ПК, а также нет возможности

остальным игрокам посмотреть всю информацию с любого другого устройства, кроме того, на котором установлен «Клубный рейтинг» и ведутся записи. Интерфейс начальной страницы программы представлен ниже на рисунке 2.

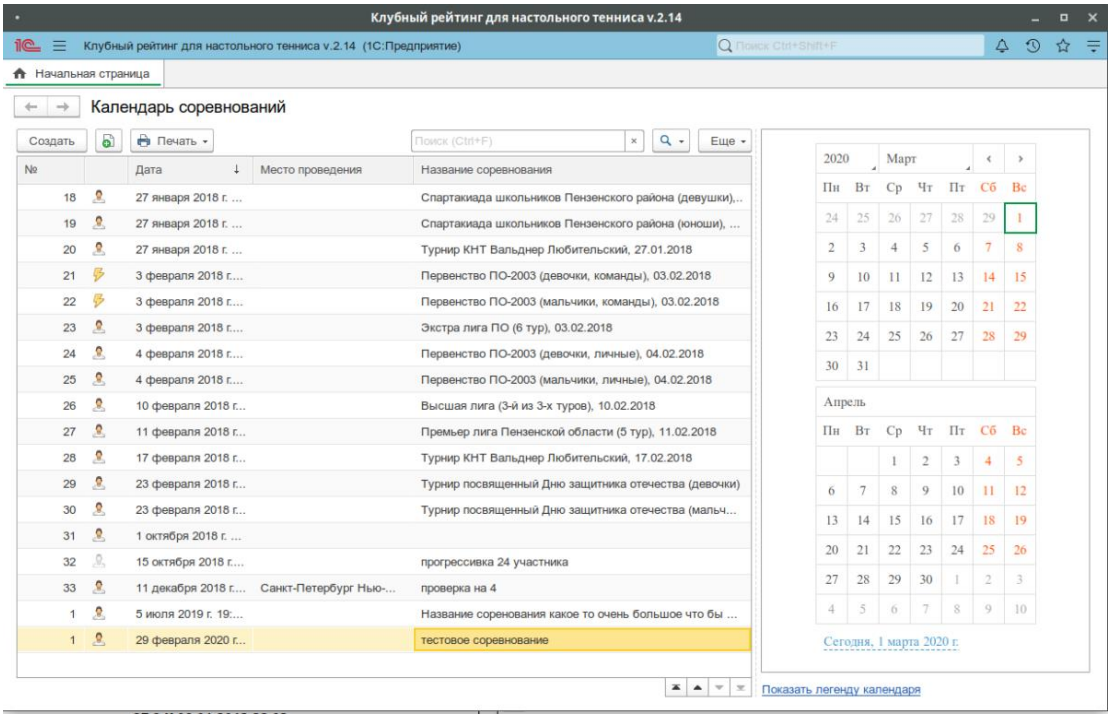

Рисунок 2 – Интерфейс начальной страницы «Клубного рейтинга»

«TTW»

Это огромный портал настольного тенниса, для федераций настольного тенниса, клубов, турниров и соревнований любых масштабов - от самых маленьких до самых больших. В системе имеются клубы не только России, но также и Беларуси, Украины, Казахстана. Главный плюсом и отличием от других систем считается рейтинг. При расчете рейтинга используется оригинальная формула Федерации настольного тенниса России (ФТНР). Также при расчёте TTW-Рейтинга учитывается коэффициент турнира (КТ), который определяется от среднего рейтинга участников данного турнира. Официальные соревнования, проводимые под эгидой ФНТР, также имеют свои коэффициенты соревнований.

Таким образом, TTW-Рейтинг является единственным из известных любительских рейтингов, чья концепция максимально соответствует концепции

официального рейтинга ФНТР. TTW-рейтинг полностью отвечает задачам спортсменов (профессионалов и любителей) и их тренеров не только иметь некую неизменную навсегда шкалу, по которой можно лишь производить посев, но, что особенно важно, TTW-рейтинг даёт возможность игрокам достигать рейтинговых значений, действительно соответствующих их возрастающему классу и мастерству[1].

Система имеет большое количество разной информации:

- учёт рейтинга (есть графики);
- список турниров (название турнира, дата, город проведения, количество игроков и имя победителя);
- список игроков (имя, фамилия, отчество, фото, город, количество турниров, количество игр, выигрыши-поражения, текущий рейтинг, позиция в рейтинге контакты, дата последнего занесенного в систему матча).

Сайт также позволяет подключится к системе рейтинга и вести учет клуба и его игроков, но для этого нужно подать заявку на регистрацию клуба. Необходимая для регистрации информации представлена ниже на рисунке 3.

В итоге, платформа хорошо подходит для клубов, профессиональных и полупрофессиональных матчей, но совсем не пригодна для небольших дружеских компаний и любительских игр.

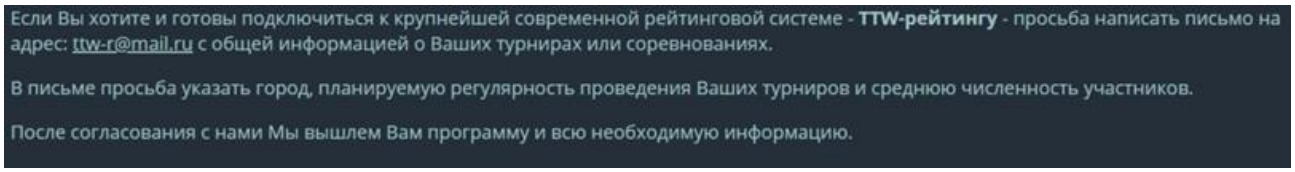

Рисунок 3 – Подключение к TWW-рейтингу

«Rankedin»

Можно создать клуб и вести статистику. Однако, чтобы учитывать рейтинг игроков они должны зарегистрироваться и вступить в клуб самостоятельно, что не подходит для случайных, спонтанных матчей и турниров.

«Tivitapp»

Каждый игрок должен иметь под рукой (под столом) телефон и записывать счет. Приложение предоставляет удобный интерфейс, есть рейтинг игроков. Отлично для компании примерно в 8 человек. Из минусов то, что необходимо обоим участникам игры зайти в приложение на своих устройствах и иметь интернет подключение, также для ведения счета нужно какое-нибудь приспособление для крепления под столом, либо убрать телефон в какое-то другое место и подбегать к нему для регистрации изменения счета.

«Pingpongapp»

Отлично подходит для больших клубов. Приложение имеет большое количество функций. Есть как мобильное, так и web-приложение, что очень удобно, поскольку не обязательно скачивать приложение. Однако, не имеет рейтинговой системы и возможности проведения турниров.

Итоги анализа существующих решений

Подводя итог, все выше перечисленные приложения имеют как свои минусы, так и плюсы. К сожалению, не одна из систем не может полностью выполнить поставленные задачи. Нужно реализовывать свою информационную систему клуба настольного тенниса.

1.3 Анализ подходов разработки

Для создания любого приложения нужно определить какого типа оно будет. На сегодняшний день существует три основных типа приложений:

- десктопное приложение;
- мобильное приложение;
- веб-приложение.

Десктопное приложение

Из-за отсутствия возможности и среды разработки для Mac OS устройств, а в случае Linux устройств, из-за малого количества устройств на этой системе и

малом опыте разработки программного обеспечения под эту операционную систему, разработка велась бы исключительно под операционную систему Windows.

Для реализации десктопного приложения можно взять язык программирования C# и конечно же .NET Framework, поскольку этот язык был разработан специально работы с этим фреймворком. Язык C# является довольно-таки «мощным» и бурно развивающийся на данный момент, имеет большое количество библиотек и шаблонов. В качестве базы данных (БД) подошел бы и SQLite и MySQL и PostgreSQL. Для работы с этими базами данных существует Entity Framework, который позволяет использовать язык запросов LINQ, а не сложные конструкции SQL.

Однако, как было упомянуто выше в анализе существующих предложений, а именно при рассмотрении «Клубного Рейтинга», десктопное приложение имеет большой недостаток, а именно, оно привязывает пользователей к использованию компьютера, что очень неудобно для игроков и организаторов. Разработка десктопного приложения не подходит.

Мобильное приложение

При создании мобильного приложения разработка будет вестись исключительно для Android устройств, из-за отсутствия возможности и среды разработки для iOS устройств.

Разработка будет вестись на языке Kotlin, поскольку этот язык является приоритетным в разработке на ОС Android.

Для хранения данных будет использована база данных SQlite.

Веб-приложение

Существует два основных подхода для создания веб-приложения Single Page Application (SPA) и Multiple Page Application (MPA). Целесообразнее было бы выбрать SPA, по следующим причинам:

– скорость загрузки SPA гораздо выше, чем у обычных сайтов. Это один из ключевых факторов удобства для пользователя;

– из-за своей скорости и «легкости» SPA являются более отзывчивыми, чем MPA.

При таком подходе SPA должно «общаться» с бэкендом через API. Это означает управление состоянием авторизации через API, а также управление состоянием на стороне клиента.

Существует три основных способа рендеринга web-страниц:

– Client-Side Rendering;
– Server-Side Rendering;
– Static-Site Generation.

Выбор пал на Client-Side Rendering

Рендеринг на клиенте с использованием Single Page Application (SPA) – это один из популярных подходов с появлением js-фреймворков. При генерации на клиенте сервер отправляет клиенту JS файлы и статичный HTML. Затем клиент отправляет запросы API, чтобы получить ресурсы. После получения приложение начинает рендеринг.

Преимущества рендеринга на клиенте:

– приложения легко разрабатывать и размещать на хостинге. Нет необходимости в сервере для приложений, рендерящихся на стороне клиента. Можно просто разместить свое приложение на любой CDN или файловый хостинг. Есть множество бесплатных способов размещения;
– нет необходимости полной перезагрузки страницы. Пользователи могут переходить по страницам без серии обращений к серверу. Из-за этого создаётся ощущение высокой скорости работы, почти как у нативного приложения.

Недостатки рендеринга на клиенте:

– приложения с рендерингом на клиенте часто сложно продвигать в поисковиках. Хотя Google заявляют, что индексируют сайты, рендерящиеся с помощью JS, они ранжируют эти сайты очень низко.

Если ваш сайт загружается слишком долго, он может быть проиндексирован как пустая страница;

- на медленных ноутбуках и мобильных устройствах рендеринг на клиенте может замедлить загрузку на несколько секунд. Пользователь может покинуть страницу, не дождавшись окончания загрузки;
- приложения с рендерингом на стороне клиента выполняют дополнительный запрос к серверу с API для рендеринга. Соответственно, ваш сайт будет загружаться дольше, чем аналогичное статичное приложение или приложение с рендерингом на стороне сервера[2].

Используя Client-Side Rendering и запросы к API, появляется возможность реализовать Progressive Web Application (PWA). PWA – это веб-приложение, которое может быть установлено на устройство. Оно может работать оффлайн, когда нет подключения к интернету, используя данные, закэшированные во время последней работы с приложением.

Операционные системы, поддерживающие PWA[3]:

- Android;
- iOS;
- iPadOS;
- macOS;
- Linux;
- Windows;
- Chrome OS.

Браузеры поддерживающие, PWA[3]:

- Chrome;
- Safari;
- Firefox (до версии Firefox 85);
- Edge;
- Samsung Internet;

– Opera;
– Brave;
– Baidu;
– Яндекс.Браузер.

Для более высокой скорости работы и удобства написания кода стоит использовать фреймворки, как на бэкенде так и на фронтенде. Laravel (PHP) - для бэкенда и Vue.js (JavaScript) – для фронтенда. Для хранения данных подошла бы база данных MySQL.

Итог анализа подходов разработки

Рассматривая все вышеперечисленные способы реализации, остановимся на веб-приложении, а конкретнее на PWA. Самым важным достоинством данной реализации является отсутствие привязки к какой-либо определенной платформе.

2 Проектирование системы

2.1 Инструменты разработки

Visual Studio Code

Visual Studio Code (VS Code) – это редактор исходного кода, разработанный компанией Microsoft. Легкий быстрый и мощный. Предоставляет возможность разработки во множестве сфер: десктопные приложения, веб-разработка, мобильная разработка.

Плюсы VS Code:

- поддерживает множество языков, включая JavaScript с фреймворком Vue.js и PHP с фреймворком Laravel, а также HTML и CSS;
- имеет удобную систему расширений, встроенную прямо в приложение. Имеется огромное количество расширений для разных задач;
- встроенный отладчик для веб-разработки (для других проектов можно установить расширения);
- интегрированный терминал;
- IntelliSense – это вспомогательная функция автодополнения кода, которая дописывает названия функций, методов, зарезервированных переменных, исходя из первых напечатанных. VS Code предоставляет IntelliSense для JavaScript, Vue.js, CSS, HTML, PHP, Laravel.

Vue.js

Клиентскую часть приложения будем разрабатывать, использую фреймворк Vue.js. Главным преимуществом Vue.js, по сравнению с другими фреймворками, такими как ReactJS и Angular, является легкость его освоения. Для создания веб-приложений на данном фреймворке разработчику нужно знать только основы HTML, CSS и JavaScript. В изучении также помогает подробнейшая официальная документация, в том числе и на русском языке, а также наличие большого комьюнити.

Vue.js предоставляет удобную и гибкую работу с объектной моделью документа (DOM). Копирование исходной модели DOM и использование ее виртуальной модели для обновления элементов вместо обновления всей модели повышает производительность веб-приложения и производит рендер страницы быстрее. Упрощается обновление данных и связных компонентов. [3]

Благодаря разделению кода при разработке на компоненты повышается его читаемость, появляется больше возможностей для модульного тестирования, а также огромным плюсом является возможность повторно использовать компоненты.

Laravel

Laravel является мощным инструментом разработки веб-приложений. Легок в освоении и удобен в разработке. Имеет подробную документацию на английском и русском языках, а также обширное комьюнити. Отлично подходит для реализации API.

Другими преимуществами являются:

- Eloquent ORM – облегчает работу с базами данных. Позволяет получать, вставлять, обновлять и удалять записи бд;
- Unit-тесты – laravel позволяет тестировать приложение с помощью встроенного PHPUnit;
- Разделение кода (паттерн Model-View-Controller) – написание более читаемого кода, разделяет фронтенд и бэкенд разработку;
- Система миграций баз данных – инструмент для создания, обновления и удаления баз данных.

MySQL

MySQL – одна из самых востребованных систем управления базами данных (СУБД) в веб-разработке. Относится к реляционным СУБД. Имеет высокую производительность и обеспечивает высокий уровень безопасности.

В частности, следует отметить такие положительные факторы как:

– используется в связке со множеством языков программирования, хорошо работает с интерфейсами API (отлично подходит для проектов, написанных с помощью фреймворка Laravel);
– легко устанавливается, а также имеет большое количество плагинов для более удобной работы;
– поддерживает разные типы строковых и численных данных, форматов дат и т.д.;
– поддержка со стороны хостингов;
– является масштабируемой. Хорошо подходит для работы как с большим, так и с малым количеством данных.

Open Server

Для тестирования серверной части нужно использовать локальный сервер, однако, мы не будем вручную устанавливать сервер Apache, интерпретатор PHP, базу данных MySQL и заниматься их настройкой, вместо этого будем использовать Open Server.

Open Server – портативная программная среда, используемая для тестирования сайтов прямо на компьютере, даже в условиях отсутствия интернет соединения. Плюсами данной платформы являются:

– простая установка;
– отсутствие необходимости сложной настройки сервера;
– быстрое начало работы;
– удобная работа с сайтами;
– автономность, независимость от определенного компьютера;

С помощью Open Server очень просто можно развернуть свой сервер, нет необходимости в знании Apache и настраивании сервера с помощью написания конфигурационных файлов – все работает «из коробки». Управляется сервер из оболочки и позволяет автоматически создавать виртуальные хосты.

PhpMyAdmin

Для администрирования базы данных будем использовать веб-приложение PhpMyAdmin. Оно позволяет работать с базой данных MySQL через браузер. Данное веб-приложение имеет следующие полезные функции:

- создание и удаление БД;
- создание, копирование, удаление, переименование, изменение таблиц;
- удаление, правка, добавление полей;
- выполнение SQL-запросов;
- управление привилегиями и пользователями MySQL;
- осуществление поиска в БД или в её разделах.

Postman

Postman – это один из самых популярных инструментов для тестирования API. С помощью Postman можно отправлять запросы серверным частям приложений и получать ответ от них. С его помощью удобно тестировать бэкенд.

Основными преимущества данного инструмента:

- позволяет начать работу без настраивания, нет необходимости в знании языков программирования;
- поддерживает разные API (REST, SOAP, GraphQL);
- запускается на любых ОС;
- можно проводить тесты как в ручном, так и в автоматизированном режиме;
- имеется возможность сохранения запросов в папках и коллекциях;
- имеется возможность изменения параметров запросов.

2.2 Архитектура системы

REST API

Разрабатываемая информационная система придерживается определенного способа взаимодействия клиента, сервера и баз данных – REST

API. Representational State Transfer (REST) в переводе – это передача состояния представления. Технология позволяет получать и модифицировать данные и состояния удаленных приложений, передавая HTTP-вызовы через интернет или любую другую сеть[4].

Ниже, на рисунке 4 представлена схема взаимодействия API.

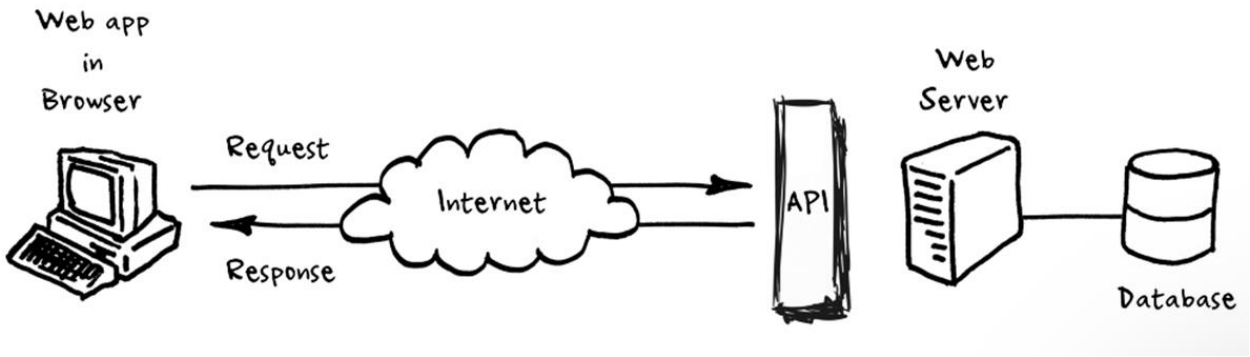

Рисунок 4 – Схема взаимодействия API

Здесь Web app in Browser – условно та часть проекта, написанная на Vue.js, иначе говоря – это фронтенд.

Следом идет API, являющийся на самом деле частью проекта, который написан на Laravel, то есть, частью серверного приложения, реализующего доступ к данным серверного приложения клиентскому приложению по определенному URL.

Последней частью является Database – это сформированная для проекта база данных MySQL.

Пользователь в web app in Browser вбивает в адресную строку URL-адрес https://clubtt.ru/rating, ожидая перейти на сайт и увидеть список всех игроков и их рейтинг. Происходит переход на сервер clubtt.ru и запрос ресурса под названием /rating. Чтобы запрашиваемый ресурс, выполнял нужные действия, используют разные способы обращения к нему. REST API основывается на протоколе передачи гипертекста Hypertext Transfer Protocol (HTTP). Это стандартный протокол в интернете, созданный для передачи гипертекста. В REST API есть четыре классических HTTP-метода:

– GET – чтение ресурса;
– POST – создание нового ресурса;
– PUT (PATCH) – обновление (редактирование) ресурса;
– DELETE – удаление ресурса.

В каждом HTTP-запросе есть заголовок, за которым следует описание объекта на сервере – это и есть его состояние.

Таким образом серверная часть приложения понимает, что от него хотят и либо сама выполняет требуемое, либо отправляет запрос базе данных. База данных отправляет данные (в нашем случае список всех игроков) серверной части. Серверная часть приложения передает список игроков в JSON формате клиентской части. Клиентская часть приложения, основываясь на полученные данные, «отрисовывает» страницу. Пользователь видит страницу со списком всех игроков.

MVC

Laravel позволяет легко писать веб-приложения придерживаясь паттерна Model View Controller (MVC). Эта архитектура делит все компоненты проекта на три вида[5]:

– Model (модель) предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность;
– View (представление) показывает пользователю эти данные и изменяется, если меняется модель;
– Controller (контроллер) направляет данные от пользователя к системе и наоборот.

Схема взаимодействие компонентов MVC паттерна в Laravel представлена ниже на рисунке 5.

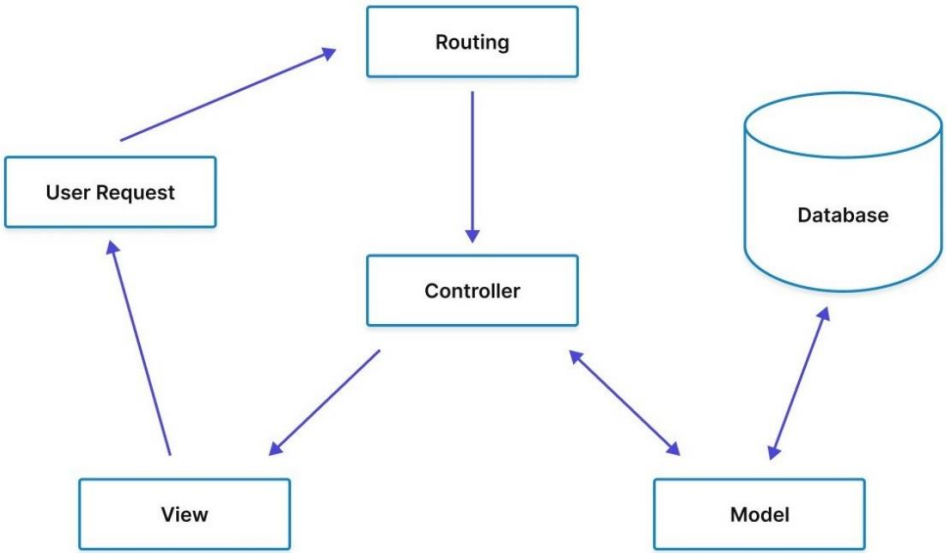

Рисунок 5 – Схема взаимодействия компонентов MVC паттерна в Laravel

Взяв прошлый пример с запросом клиента по адресу https://clubtt.ru/rating, более подробно разберем паттерн MVC, и что происходит в серверном приложении в это время.

Запрос ресурса под названием /rating обрабатывается в файле маршрутов (Routing) и вызывает нужный контроллер и его метод. Этот метод с помощью модели обращается к базе данных – модель позволяет запрашивать данные из таблиц баз данных – и получает список всех игроков. Контроллер передает полученные данные представлению (View).

2.3 Определение функциональных требований

На основе необходимого функционала системы, описанного в пункте 1.3 были сформированы функциональные требования информационной системы и поделены на модули:

а) Работа с пользователями:
   1) авторизация пользователя;
   2) выход из аккаунта пользователя.

б) Работа с матчами:
   1) отображение истории матчей;

2) создание нового матча.

в) Работа с турнирами:

1) отображение истории турниров;

2) отображение матчей, проведенных в рамках турнира

3) создание нового турнира.

г) Работа с игроками:

1) отображение всех игроков;

2) создание нового игрока;

3) редактирование игрока.

Работа с пользователями

Для работы приложения не обязательна аутентификация пользователя, просматривать список игроков, историю матчей и историю турниров может и не аутентифицированный пользователь. Для создания матча, создания турнира, создания игрока и его редактирования нужна аутентификация. Для этого пользователю нужно необходимо предусмотреть систему авторизации пользователей.

Система авторизации должна использовать следующие данные, с помощью которых осуществляется вход в систему:

а) логин;

б) пароль.

Работа с матчами

Информационная система должна предоставлять возможность создания матчей для авторизированных пользователей и просмотра истории матчей для всех пользователей.

а) Каждый матч содержит следующие данные:

1) ФИО первого игрока;

2) ФИО второго игрока;

3) счет первого игрока;

4) счет второго игрока;

5) рейтинг, который был у первого игрока до этого матча;

6) рейтинг, который был у второго игрока до этого матча;
7) дата проведения.

б) Для создания матча используются следующие данные введенные пользователем:
1) ФИО первого игрока;
2) ФИО второго игрока;
3) счет первого игрока;
4) счет второго игрока.

Работа с турнирами

Информационная система должна предоставлять возможность создания турниров для авторизированных пользователей и просмотра истории турниров для всех пользователей.

а) Каждый турнир содержит следующие данные:
1) количество участников;
2) дата проведения.

б) Для создания турнира пользователь должен выбрать из списка всех участвующих в этом турнире игроков.

При нажатии на турнир, должен появляться список всех матчей, проведенных в рамках этого турнира.

Работа с игроками

Информационная система должна предоставлять возможность добавления новых игроков, редактирования данных о существующих и возможность просмотра списка игроков, а также их статистики и актуального рейтинга.

а) Каждый игрок содержит следующие данные:
1) фамилия;
2) имя;
3) отчество;
4) количество побед;
5) количество всех проведенных матчей;
6) рейтинг.

б) Для добавления игрока используются следующие данные, введенные пользователем:
   1) фамилия;
   2) имя;
   3) отчество.

в) Редактировать у игрока можно следующие данные:
   1) фамилия;
   2) имя;
   3) отчество.

2.4 Структура приложения

Представления

Представления являются клиентской частью веб-приложения. Пользователь взаимодействует только с ними, добавляя или изменяя базу данных через данные, передаваемые через представления контроллеру.

Рассматривая требуемы функционал можно сделать вывод, что необходимы следующие представления:

а) Представление аутентификации:
   1) Login.vue – представление для авторизации пользователя.

б) Представление матчей:
   1) History.vue – представление для отображения истории всех прошедших матчей.
   2) CreateDuel.vue – представление для создания матча, представляющее из себя страницу с двумя полями и списком игроков.

в) Представление турниров:
   1) History.vue – представление для отображения истории всех прошедших турниров. При нажатии на турнир открывается страница со списком всех матчей, проведенных в рамках этого турнира.

2) CreateTournament.vue – представление для создания турнира, представляющее из себя список всех игроков, которых нужно выбрать для участия в турнире.

г) Представление игроков:

1) Raiting.vue – представление для отображения всех игроков в порядке невозрастания рейтинга.
2) Menu.vue – представление, через которое должен предоставляться выбор для создания нового игрока или редактирования информации уже существующего.

Контроллеры

При разработке структуры приложения был сделан вывод, что в системе должны использоваться следующие контроллеры, реализующие функции, которые обеспечивают наличие требуемого функционала приложения:

а) AuthController.php – контроллер, осуществляющий работу с пользователем: отвечает за авторизацию пользователей;

б) DuelController.php – контроллер, осуществляющий работу с матчами: передает их представлению и создает новые;

в) TournamentController.php – контроллер, осуществляющий работу с турнирами: передает их представлению и создает новые;

г) PlayerController.php – контроллер, осуществляющий работу с игроками: передает их представлению, создает новые, и редактирует данные занесенных в базу данных игроков.

Модели

Шаблон проектирования MVC включает в себя использование моделей, которые представляют собой данные. Таким образом, шаблон проектирования будет использовать следующие модели, которые будут содержать нижеперечисленные поля:

а) User – является моделью представления данных о пользователях системы. Содержит следующие поля:

1) уникальный идентификатор пользователя;

2) уникальный логин пользователя;
3) пароль пользователя.

б) Duel – является моделью представления данных о матчах. Содержит следующие поля:
1) уникальный идентификатор матча;
2) идентификатор первого игрока;
3) идентификатор второго игрока;
4) счет первого игрока;
5) счет второго игрока;
6) рейтинг первого игрока до матча;
7) рейтинг второго игрока после матча;
8) идентификатор турнира;
9) номер матча в порядке проведения внутри турнира;
10) дата и время проведения матча.

в) Tournament – является моделью турнира:
1) уникальный идентификатор турнира;
2) тип турнира;
3) количество участников турнира;
4) дата и время проведения турнира.

г) Player – является моделью представления данных о пользователях системы, содержит следующие поля:
1) уникальный идентификатор игрока;
2) имя игрока;
3) фамилия игрока;
4) отчество игрока;
5) количество побед игрока;
6) количество всех проведенных матчей;
7) текущий рейтинг игрока.

3 Разработка системы

3.1 Клиентская часть

Для разработки клиентской части приложения было решено использовать фрэймворк Vue.js. Чтобы установить его в проект воспользуемся рекомендованным официальной документацией способом – менеджером пакетов NPM. Данный пакетный менеджер по умолчанию устанавливается вместе с программной платформой Node.js. Начнем с его установки.

На официальном сайте программной платформы Node.js – https://nodejs.org – необходимо скачать актуальную рекомендованную версию. Установка происходит стандартно. После установки необходимо убедится, что все было установлено правильно. В командной строке Windows пропишем две команды:

а) `node -v`;

б) `npm -v`.

Если все было установлено верно, первая команда выведет версию Node.js, которая была установлена на компьютер, а вторая команда выведет версию NPM (а также актуальную на данный момент). Результат введения команд изображен ниже на рисунке 6.

Рисунок 6 – Проверка версий Node.js и NPM

Теперь приступим к установке Vue.js. Создадим папку, в которой будет находится наш проект, откроем ее с помощью Visual Studio Code. В терминале, предоставленном VS code пропишем следующую команду:

```
npm install vue@next
```

Эта команда установит последнюю стабильную версию Vue.js в папку проекта. Запустим локально проект с помощью следующей команды:

```
npm run serve
```

Для сборки проекта используется команда:

```
npm run build
```

Vue предоставляет официальный CLI для быстрого создания каркаса одностраничных приложений. Предлагаемые шаблоны содержат всё необходимое для организации современной фронтенд-разработки. За несколько минут можно получить работающую конфигурацию с горячей перезагрузкой модулей, линтингом кода при сохранении и настроенной конфигурацией production-сборки[6]. Для его установки пропишем в терминале следующую команду:

```
npm install -g @vue/cli
```

Данная команда установит в проект Vue CLI, который является стандартным инструментарием для разработки на Vue.js, предоставляющим множество других возможностей, помимо быстрого создания каркаса одностраничных приложений. С его помощью установим плагин PWA следующей командой:

```
vue add pwa
```

На этом подготовка проекта завершена. Можно приступать к написанию кода.

Страницы–представления

а) App.vue – главная страница:

Включает в себя следующий метод:

mounted() – инициирует проверку аутентификации пользователя и получение списка игроков, матчей, турниров.

б) Login.vue – страница аутентификации:

Включает в себя следующие методы:

1) guestAuth() – вход в качестве гостя, перенаправляет на главную страницу;
2) adminAuth() – авторизовывает пользователя.

в) Home.vue – домашняя страница:

Включает в себя следующие компоненты:

1) MainHeader.vue – элемент вверху, содержащий в себе название «TT Club» и кнопку перехода в меню;
2) MainFooter.vue – элемент внизу, содержащий кнопки «Рейтинг», «+», «Матчи».
3) PopupCreate.vue – всплывающий элемент, появляющийся при нажатии на кнопку «+»;

г) Rating.vue – страница со списком всех игроков отсортированным в порядке невозрастания рейтинга:

1) Включает в себя следующий компонент:

   UserCard – компонент содержащий в себе ФИО игрока и рейтинг.

2) Включает в себя вычисляемое значение:

   players - возвращает массив объектов, содержащих информацию об игроках.

д) History.vue – страница, отображающая историю всех матчей:

1) Включает в себя следующие компоненты:
   1) MatchCard – компонент, содержащий в себе информацию о прошедшем матче;
   2) TournamentCard – компонент, содержащий в себе дату и количество участников, прошедшего турнира;
2) Включает в себя следующие вычисляемые значение:
   1) historyCards() – возвращает массив дуэлей или турниров в зависимости от выбранной категории;

2) typeCard() – возвращает название компонента, зависимости от выбранной категории;
3) nextPage() – возвращает ссылку на следующую страницу пагинации.

3) Включает в себя следующий метод:

loadNextPages() – при нажатии на кнопку «Показать ещё» загружает следующую страницу пагинации.

е) Menu.vue – страница предоставляющая возможность создания нового игрока и редактирования информации уже существующих, в также предоставляющий возможность выхода из аккаунта и перехода на страницу аутентификации:

1) Включает в себя следующие вычисляемые значения:
   1) isAuth() – возвращает статус авторизации пользователя.
   2) players() – возвращает количество игроков, которые будут отрисованы;
2) Включает в себя следующие методы:
   1) logout() – инициация выхода из аккаунта и переход на страницу Login.vue;
   2) login() – инициация авторизации и переход на страницу Login.vue.

ж) CreateDuel.vue – страница создания одиночных матчей:

1) Включает в себя следующий компонент:

   DuelScore – счет игроков.

2) Включает в себя следующие вычисляемые значения:
   1) players() – возвращает всех игроков клуба;
   2) accessToSaveDuel() – проверка на отсутствие ничьи.
3) Включает в себя следующие методы:
   1) choosePlayer() – добавляет игрока, как участника матча;
   2) chooseInput() – выбрать строку для добавления игрока, ка участника;
   3) nextStage() – переход к записи счета;

4) saveDuel() – инициирует запрос отправки результатов матча на сервер.

з) CreateTournament.vue – страница создания турниров:

1) Включает в себя следующий компонент:

TournamentGrid – компонент, представляющий из себя этапы турнира.

2) Включает в себя следующие вычисляемые значения:

1) participantsSelected() – проверяет выбран ли игрок.
2) players() – возвращает всех игроков клуба;
3) selectedPlayer() – возвращает выбранных для участия в турнире игроков;
4) accessToCreate() – проверка на количество игроков, которых должно быть больше нуля и кратно восьми.

3) Включает в себя следующий метод:

1) togglePlayer() – переключение игроков между не выбранным и выбранным;
2) createTournament() – переход к заполнению сетки турнира.

3.2 Серверная часть

Для разработки серверной части приложения развернем локальный сервер с помощью Open Server. На официальном сайте https://ospanel.io/ необходимо скачать актуальную версию установочного пакета программы. Установка происходи стандартно, при выборе компонентов, которые нужно установить выбираем компактную установку. Заходим в Open Server Panel, идем в настройки и выбираем необходимые модули:

- HTTP: Apache_2.4-PHP_8.0-8.1;
- PHP: PHP_8.0;
- MySQL/MariaDB: MySQL-8.0-Win10.

Теперь нужно создать проект. Для этого запускаем сервер, нажимая на кнопку «запустить» – красный флажок сменится зеленым, информирую об успешно запущенном сервере. Откроем консоль и перейдем в директорию, в которой будет хранится наш проект с помощью следующей команды:

```
cd <директория>
```

Воспользуемся встроенным в Open Server пакетным менеджером Composer для создания проекта. Пропишем следующую команду:

```
composer create-project laravel/laravel <название проекта>
```

Данная команда создала в директории laravel-проект с нашим названием. Теперь можно приступать к написанию кода.

Контроллеры

При разработке структуры приложения был сделан вывод, что в системе должны использоваться следующие контроллеры, реализующие функции, которые обеспечивают наличие требуемого функционала приложения.

Все контроллеры и их методы:

а) AuthController.php – контроллер, осуществляющий обработку маршрутов отвечающих за аутентификацию пользователей.

Методы:

1) login()

Описание: осуществляет авторизацию.

Возвращаемые данные: в случае успеха – JSON-объект, содержащий токен доступа, тип токена, время жизни токена; в случае неудачи – JSON-объект, содержащий сообщение «Unauthorized».

2) logout()

Описание: осуществляет выход из аккаунта.

Возвращаемые данные: JSON-объект, содержащий сообщение «Successfully logged out».

б) DuelController.php – контроллер, осуществляющий работу с матчами.

Методы:

1) showAllDuels()

Описание: запрашивает всю историю не турнирных матчей из базы данных и передает её клиенту.

Возвращаемые данные: JSON-объект со списком всех игроков.

2) showPlayerDuelsInfo(Player $player)

Описание: запрашивает все матчи, в которых участвовал определенный игрок, из базы данных и передает их клиенту.

Параметры: Player $player – id игрока.

Возвращаемые данные: JSON-объект со списком всех матчей, в которых участвовал игрок.

3) showTournamentDuelsInfo(Tournament $tournament)

Описание: запрашивает все матчи определенного турнира из базы данных и передает их клиенту.

Параметры: Tournament $tournament – принимает id турнира.

Возвращаемые данные: JSON-объект со списком всех матчей, прошедших в рамках данного турнира.

4) createDuel(DuelRequest $request, $id_tournament = NULL)

Описание: создает матч, заносит его в базу данных и обновляет информацию о участвовавших в этом матче игроках.

Параметры: DuelRequest $request – JSON-объект с полученными от клиента данными матча; $id_tournament = NULL – id турнира.

в) TournamentController.php – контроллер, осуществляющий работу с турнирами.

Методы:

1) showAllTournaments()

Описание: запрашивает всю историю турниров из базы данных и передает её клиенту.

Возвращаемые данные: JSON-объект со списком всех турниров.

2) createTournament(TournamentRequest $request)

Описание: создает турнир, заносит его в базу данных, создает все матчи, проведенные в рамках этого турнира и заносит их в базу

данных, обновляет информацию о участвовавших в этом матче игроках.

Параметры: TournamentRequest $request – JSON-объект с полученными от клиента данными турнира.

г) PlayerController.php – контроллер, осуществляющий работу с игроками.

Методы:

1) showAllPlayers()

Описание: запрашивает список всех игроков из базы данных и передает его клиенту.

Возвращаемые данные: JSON-объект со списком всех игроков.

2) createPlayer(PlayerRequest $request)

Описание: создает игрока и заносит его в базу данных.

Параметры: PlayerRequest $request – JSON-объект с полученными от клиента данными об игроке.

3) editPlayer(PlayerRequest $request, Player $player)

Описание: редактирует информацию об игроке.

Параметры: PlayerRequest $request – JSON-объект с полученными от клиента данными об игроке; Player $player – id игрока.

4) updatePlayers($id_winner, $id_looser)

Описание: обновляет рейтинг, количество побед и общее количество проведенных игр пары игроков.

Параметры: $id_winner – id выигравшего игрока; $id_looser – id проигравшего игрока.

Модели

Laravel содержит ORM-библиотеку Eloquent, предоставляющую способ работы с базой данных, который часто удобнее обычного построителя запросов. При использовании Eloquent каждая таблица БД имеет соответствующую «Модель», которая используется для взаимодействия с этой таблицей. Помимо получения записей из таблицы БД, модели Eloquent также позволяют вставлять,

обновлять и удалять записи из таблицы[7]. Структура базы данных изображена на рисунке 7.

Users:

1) id – идентификатор пользователя, первичный ключ. Имеет тип bigint;
2) login – имя/логин для идентификации. Имеет тип char;
3) password – пароль пользователя. Имеет тип char;
4) created_at – время создания. Имеет тип timestamp;
5) updated_at – время последнего изменения. Имеет тип timestamp.

Players:

1) id – идентификатор пользователя, первичный ключ. Имеет тип bigint;
2) name – имя игрока. Имеет тип char;
3) surname – фамилия игрока. Имеет тип char;
4) patronymic – отчество игрока. Имеет тип char. Может принимать значение равное NULL;
5) victories – количество побед игрока. Имеет тип smallint;
6) all_games – количество поражений игрока. Имеет тип smallint;
7) rating – текущий рейтинг игрока. Имеет тип double. Имеет дефолтное значение равное 100;
8) created_at – время создания. Имеет тип timestamp;
9) updated_at – время последнего изменения. Имеет тип timestamp.

Duels:

1) id – идентификатор матча, первичный ключ. Имеет тип bigint;
2) id_first – идентификатор первого игрока. Является внешним ключом. Имеет тип bigint;
3) id_second – идентификатор второго игрока. Является внешним ключом. Имеет тип bigint;
4) score_first – счет первого игрока. Имеет тип tinyint;
5) score_second – счет второго игрока. Имеет тип tinyint;
6) rating_first – рейтинг первого игрока до матча. Имеет тип double;
7) rating_second – рейтинг второго игрока до матча. Имеет тип double;

8) id_tournament – идентификатор турнира. Если матч был не турнирный, то принимает значение NULL. Является внешним ключом. Имеет тип bigint;
9) index_duel – номер матча в порядке проведения турнира. Если матч был не турнирный, то принимает значение NULL. Имеет тип tinyint;
10) created_at – время создания. Имеет тип timestamp;
11) updated_at – время последнего изменения. Имеет тип timestamp.

Tournaments:

1) id – идентификатор турнира, первичный ключ. Имеет тип bigint;
2) number_participants – количество участников турнира. Имеет тип tinyint;
3) created_at – время создания. Имеет тип timestamp;
4) updated_at – время последнего изменения. Имеет тип timestamp.

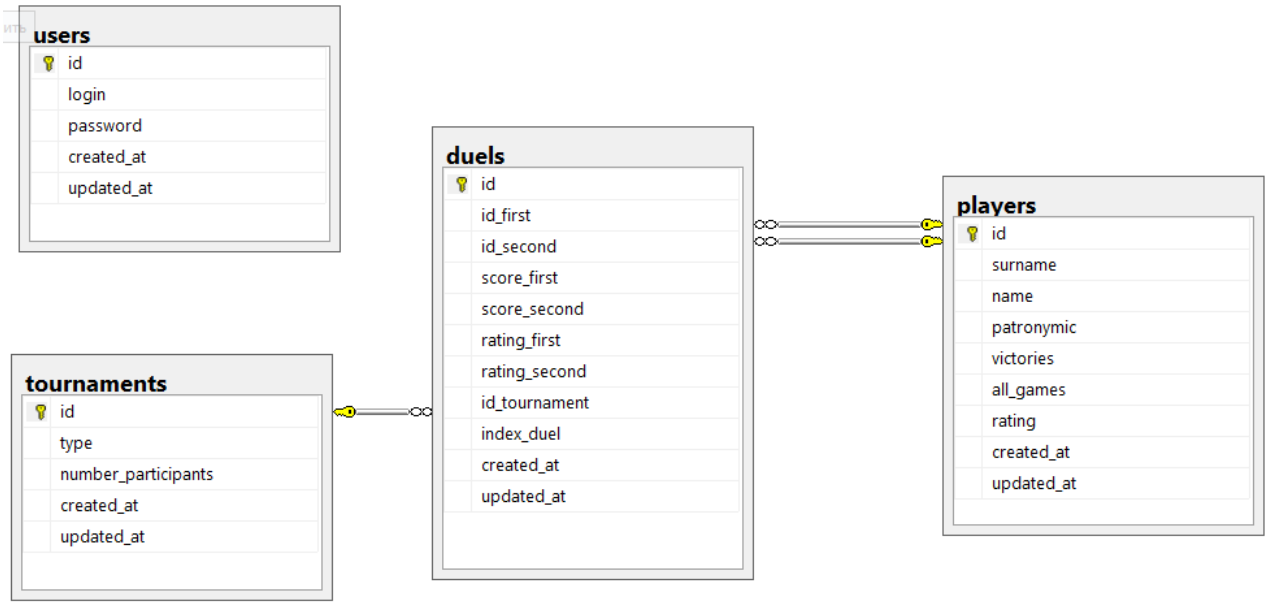

Рисунок 7 – Структура базы данных

3.3 API

API имеет 10 методов – ровно столько, сколько было определено функциональных требований в подпункте 2.3. Чтобы обратиться к методу API нужно отправить запрос на адрес:

https://api.clubtt.ru/api

Для всех методов существует свой уникальный относительный путь, по которому нужно обращаться, необходимо также указывать тип запроса (GET, POST, PATCH). Некоторые методы требуют отправки JSON-объекта с данными в теле запроса, а также отправки JWT-токена в заголовке запроса. В случае успеха в ответ приходит JSON-объект.

Рассмотрим все имеющиеся методы API.

а) Работа с пользователями:

1) Авторизация пользователя:

Относительный путь: /auth/login;

Метод: POST;

Отправляемые данные:

body:

```
{
   "login": "test",
   "password": "3223"
}
```

Получаемые данные:

```
{
   "access_token": "<JWT-токен>",
   "token_type": "bearer",
   "expires_in": 2592000
}
```

2) Выход из аккаунта пользователя:

Относительный путь: /auth/logout;

Метод: POST;

Отправляемые данные:

headers:

```
{
   Autorization: "<тип токена> <JWT-токен>"
```

}

Получаемые данные:

{

"message": "Successfully logged out"

}

б) Работа с матчами:

1) Вернуть историю не турнирных матчей:

Относительный путь: /duels;

Метод: GET;

Получаемые данные: JSON-объект с полем data, содержащим в себе массив 15 объектов (последними не турнирными матчами, занесенными в базу данных) и полем links, содержащим в себе объект с ссылками на первую, последнюю, предыдущую и следующую страницы.

2) Создание нового матча:

Относительный путь: /create/duel;

Метод: POST;

Отправляемые данные:

body:

{

"id_first": 1,

"id_second": 4,

"score_first": 3,

"score_second": 0,

"index_duel": null

}

headers:

{

Autorization: "<тип токена> <JWT-токен>"

}

в) Работа с турнирами:

1) Вернуть историю турниров:

Относительный путь: /tournaments;

Метод: GET;

Получаемые данные: JSON-объект с полем data, содержащим в себе массив 15 объектов (последними турнирами, занесенными в базу данных) и полем links, содержащим в себе объект с ссылками на первую, последнюю, предыдущую и следующую страницы.

2) Вернуть историю матчей, проведенных в рамках турнира:

Относительный путь: /tournaments/{tournament}/duelsinfo;

Метод: GET;

Получаемые данные: JSON-объект с полем data, содержащим в себе массив матчей, проведенных в рамках данного турнира, занесенными в базу данных.

3) Создание нового турнира:

Относительный путь: /create/tournament;

Метод: POST;

Отправляемые данные:

body: JSON-объект с полями: number_participants, содержащим в себе количество игроков; type, содержащим в себе тип турнира; duels, содержащим в себе массив объектов всех прошедших в рамках турнира матчей.

headers:

{

Autorization: "<тип токена> <JWT-токен>"

}

г) Работа с игроками:

1) Вернуть всех игроков:

Относительный путь: /players;

Метод: GET;

Получаемые данные: JSON-объект с полем data, содержащим в себе массив всех игроков.

2) Создать нового игрока:

Относительный путь: /create/player;

Метод: POST;

Отправляемые данные:

body:

```
{
   "surname": "Иванов",
   "name": "Иван",
   "patronymic": "Иванович"
}
```

headers:

```
{
   Autorization: "<тип токена> <JWT-токен>"
}
```

3) Редактировать данные игрока:

Относительный путь: /edit/player;

Метод: PATCH;

Отправляемые данные:

body:

```
{
   "surname": "Васильев",
   "name": "Василий",
   "patronymic": "Васильевич"
}
```

headers:

```
{
   Autorization: "<тип токена> <JWT-токен>"
}
```

3.4 Демонстрация реализованной системы

Демонстрация проводилась на мобильном Android-устройстве через PWA.

Для демонстрации системы обратимся к подпункту 2.3 и на основе функциональных требований рассмотрим приложение.

а) Работа с пользователями:

1) авторизация пользователя представлена на рисунке 8;

Рисунок 8 – авторизация пользователя

2) выход из аккаунта пользователя представлен на рисунке 9.

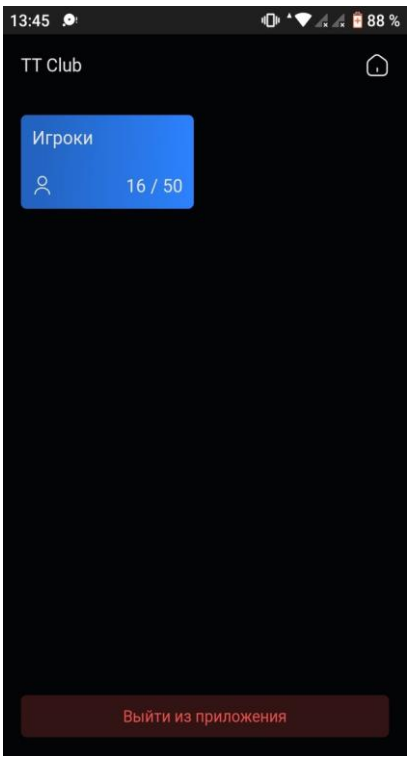

Рисунок 9 – выход из аккаунта пользователя

б) Работа с матчами:

1) Отображение истории не турнирных матчей изображено на рисунке 10;

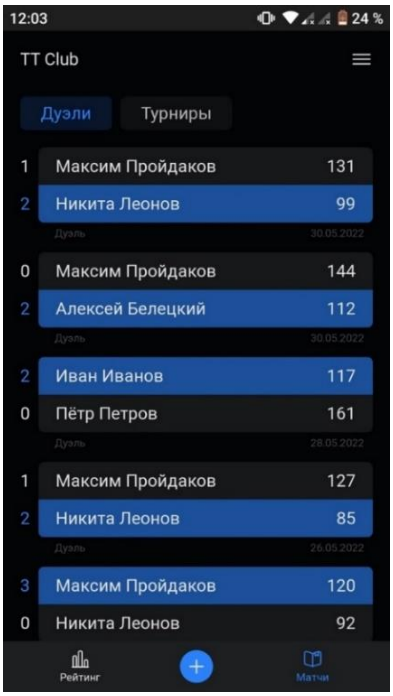

Рисунок 10 – история не турнирных матчей

2) Создание нового не турнирного матча изображено на рисунке 11.

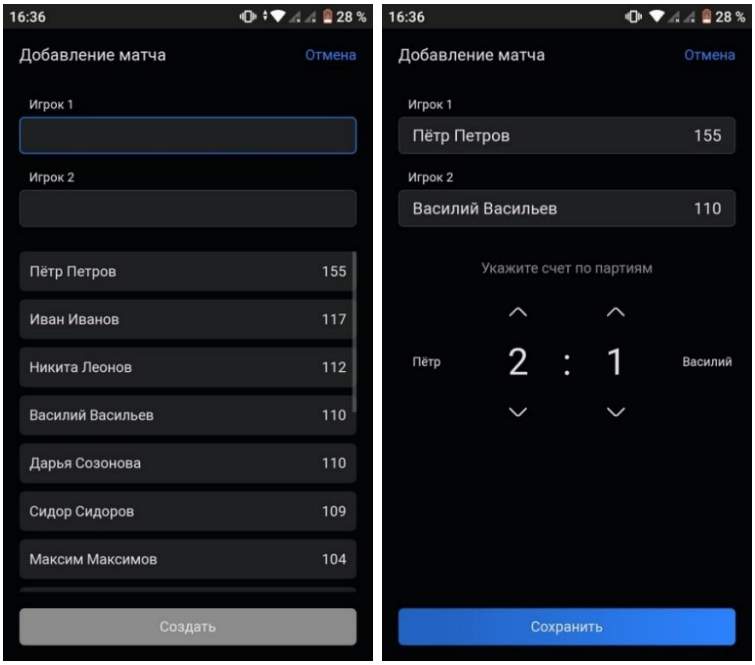

Рисунок 11 – создание не турнирного матча

в) Работа с турнирами:

1) Отображение истории турниров изображено на рисунке 12;

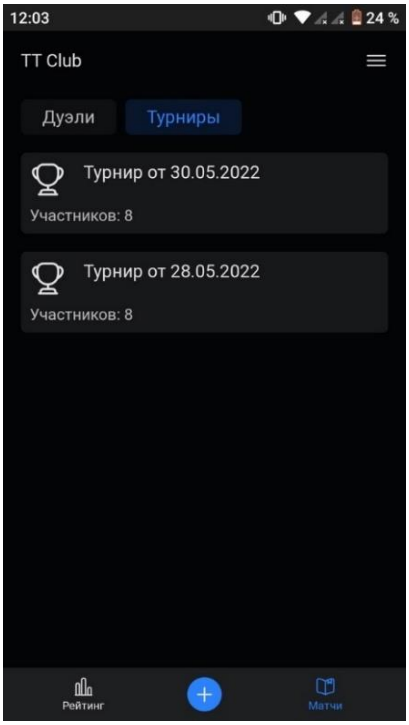

Рисунок 12 – история турниров

2) Отображение матчей, проведенных в рамках турнира, изображено ниже на рисунке 13;

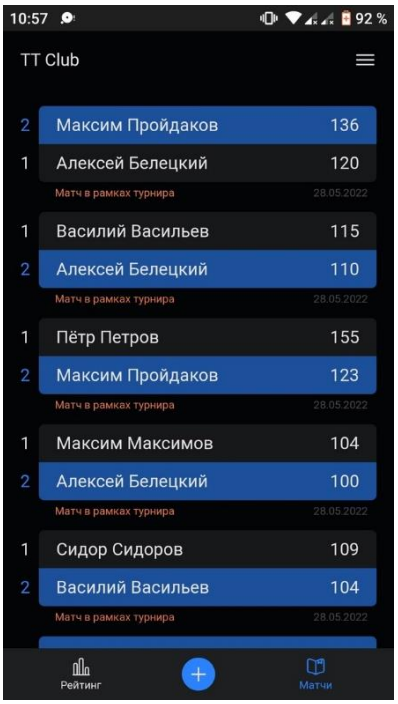

Рисунок 13 – история турнирных матчей

3) Создание нового турнира изображено ниже на рисунках 14.

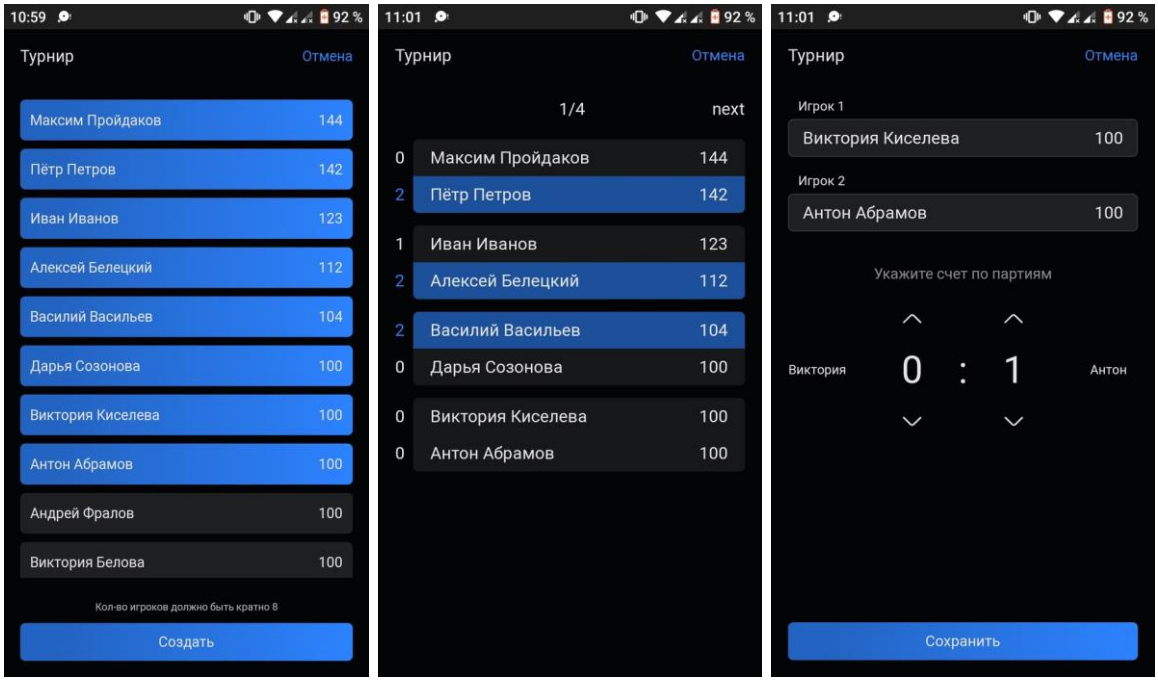

Рисунок 14 – создание турнира

г) Работа с игроками:

1) Список всех игроков изображен ниже на рисунке 15;

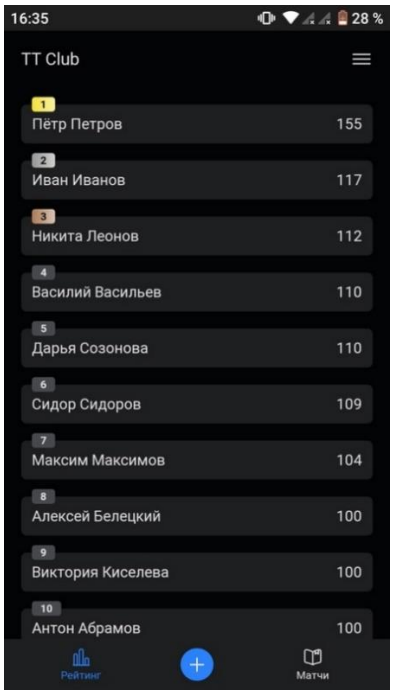

Рисунок 15 – список всех игроков

2) Создание нового игрока изображено ниже на рисунке 16;

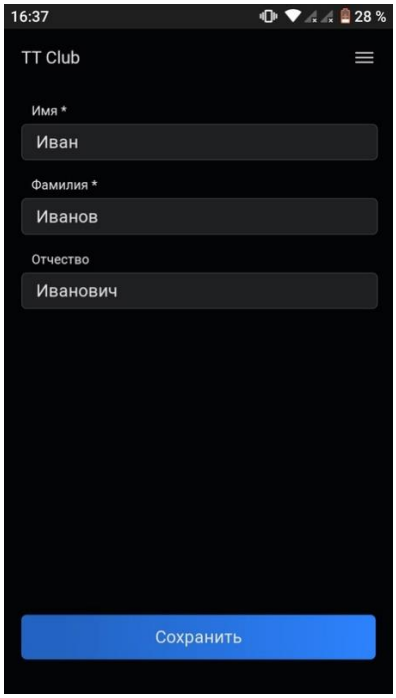

Рисунок 16 – создание нового игрока

3) Редактирование данных игрока изображено ниже на рисунке 17.

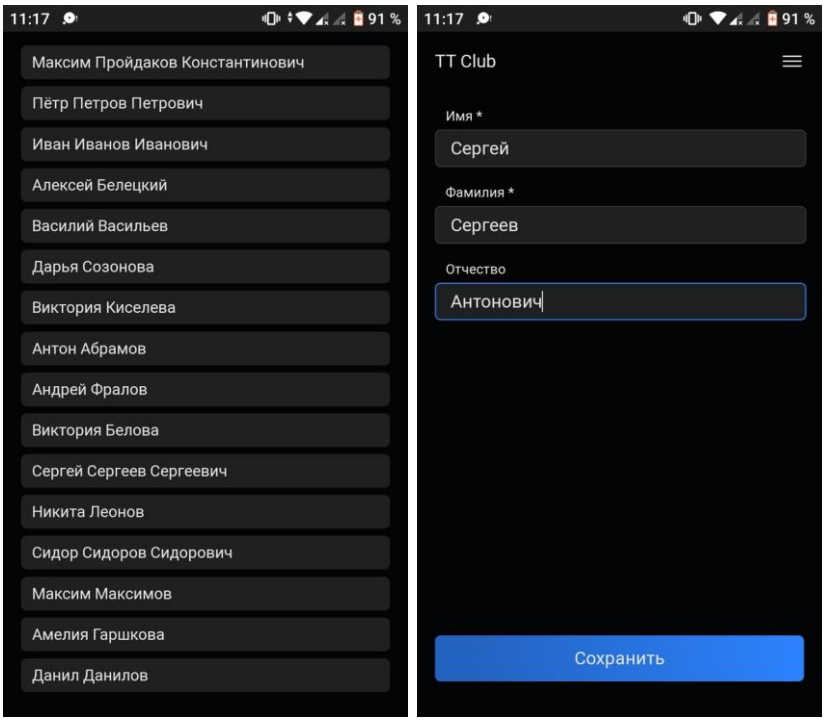

Рисунок 17 – редактирование игрока

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была разработана информационная система клуба настольного тенниса. Поставленная цель была достигнута путем создания веб-приложения, написанного с помощью фреймворков Vue.js и Laravel, а также СУБД MySQL.

В ходе работы были выполнены следующие задачи:

- проанализировано задание на ВКР;
- выбраны инструменты разработки;
- спроектирована и разработана система.

Разработанное приложение имеет следующий функционал:

а) Работа с пользователями:
   1) авторизация пользователя;
   2) выход из аккаунта пользователя.

б) Работа с матчами:
   1) отображение истории матчей;
   2) создание нового матч.

в) Работа с турнирами:
   1) отображение турниров;
   2) отображение матчей, проведенных в рамках турнира;
   3) создание нового турнира.

г) Работа с игроками:
   1) отображение всех игроков;
   2) создание нового игрока;
   3) редактирование игрока.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Глобальный рейтинг TTW [Электронный ресурс] : TTW-рейтинг – Режим доступа: http://r.ttw.ru/ttw-rating/

2. Хабрхабр [Электронный ресурс] : Рендеринг на клиенте, на сервере и генерация статических сайтов – Режим доступа: https://habr.com/ru/post/526828/

3. Разработка сайтов, интернет-магазинов и web-сервисов Сибирикс [Электронный ресурс] : Прогрессивные веб-приложения 2020 Часть 1 – Режим доступа: https://blog.sibirix.ru/PWA-2020-vol-1/

4. Mail.ru Cloud Solutions [Электронный ресурс] : Введение в Rest API: что это простыми словами – Режим доступа: https://mcs.mail.ru/blog/vvedenie-v-rest-api#:~:text=Representational%20State%20Transfer%20(REST)%20%D0%B2,%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%20%D0%B8%D0%BB%D0%B8%20%D0%BB%D1%8E%D0%B1%D1%83%D1%8E%20%D0%B4%D1%80%D1%83%D0%B3%D1%83%D1%8E%20%D1%81%D0%B5%D1%82%D1%8C

5. SkillFactory.Блог [Электронный ресурс] : laravel – Режим доступа: https://blog.skillfactory.ru/glossary/laravel/

6. Vue.js [Электронный ресурс] : Установка — Vue.js – Режим доступа: https://ru.vuejs.org/v2/guide/installation.html

7. Laravel Framework Russian Community [Электронный ресурс] : Eloquent - Начало работы – Режим доступа: https://laravel.su/docs/8.x/eloquent

ПРИЛОЖЕНИЕ А

Код файла PlayerController.php

```
<?php
namespace App\Http\Controllers;
use App\Models\Player;
use App\Http\Requests\Player\PlayerRequest;
use App\Http\Resources\Player\PlayerResource;

class PlayerController extends Controller
{
    public function showAllPlayers()
    {
        return PlayerResource::collection(Player::orderByDesc('rating')->get());
    }
    public function createPlayer(PlayerRequest $request)
    {
        if(Player::orderByDesc('id')->first()->id==50) {
            return response()->json('Достигнуто максимальное количество игрков',
500);
        }
        $player = $request->validated();

        Player::create([
            'surname' => $player['surname'],
            'name' => $player['name'],
            'patronymic' => $player['patronymic']
        ]);
    }
    public function editPlayer(PlayerRequest $request, Player $player)
    {
        $player->update($request->validated());
    }


    public function updatePlayers($id_winner, $id_looser){


        $winner = Player::find($id_winner);
        $looser = Player::find($id_looser);


        $delta_rating = (100 - ($winner['rating'] - $looser['rating']))/10;
        $winner['rating'] += $delta_rating;
        $looser['rating'] -= $delta_rating;


        if($looser['rating'] < 1){
            $looser['rating'] = 1;
        }
```

```
            $winner['victories'] += 1;
            $winner['all_games'] += 1;
            $looser['all_games'] += 1;

            Player::find($id_winner)->update([
                'victories' => $winner['victories'],
                'all_games' => $winner['all_games'],
                'rating' => $winner['rating'],
            ]);

            Player::find($id_looser)->update([
                'all_games' => $looser['all_games'],
                'rating' => $looser['rating'],
            ]);
        }
    }
```

ПРИЛОЖЕНИЕ Б

Код файла TournamentController.php

```
<?php

namespace App\Http\Controllers;
use App\Models\Tournament;
use App\Http\Requests\Tournament\TournamentRequest;
use App\Http\Requests\Duel\DuelRequest;
use App\Http\Resources\Tournament\TournamentResource;
use App\Http\Controllers\DuelController;

class TournamentController extends Controller
{
    public function showAllTournaments(){
        return TournamentResource::collection(Tournament::orderByDesc('id')-
>cursorPaginate());
    }

    public function createTournament(TournamentRequest $request){
        $id_tournament = Tournament::create([
            'type' => $request->validated('type'),
            'number_participants' => $request->validated('number_participants')
        ])->id;

        $duelController = new DuelController();
        foreach ($request->validated('duels') as $duel){
            $duelRequest = new DuelRequest();
            $duelRequest->replace($duel);
            $duelController->createDuel($duelRequest, $id_tournament);
        }
    }
}
```

ПРИЛОЖЕНИЕ В

Код файла DuelController.php

```
<?php

namespace App\Http\Controllers;
use App\Models\Player;
use App\Models\Duel;
use App\Models\Tournament;
use App\Http\Requests\Duel\DuelRequest;
use App\Http\Resources\Duel\DuelResource;

class DuelController extends Controller
{
    public function showAllDuels(){
        return DuelResource::collection(Duel::orderByDesc('id')-
>cursorPaginate());
    }

    public function showPlayerDuelsInfo(Player $player){
        return DuelResource::collection(Duel::where('id_first', $player->id)-
>orwhere('id_second', $player->id)->orderByDesc('id')->cursorPaginate());
    }

    public function showTournamentDuelsInfo(Tournament $tournament){
        return DuelResource::collection(Duel::where('id_tournament',
$tournament->id)->orderByDesc('id')->get());
    }

    public function createDuel(DuelRequest $request, $id_tournament = NULL)
    {
        $duel = $request->all();

        $id_first = $duel['id_first'];
        $id_second = $duel['id_second'];

        $rating_first = Player::find($id_first)->rating;
        $rating_second = Player::find($id_second)->rating;

        $playerController = new PlayerController();
```

```
        if ($duel['score_first'] > $duel['score_second']){
            $playerController->updatePlayers($id_first, $id_second);
        } elseif($duel['score_first'] < $duel['score_second']) {
            $playerController->updatePlayers($id_second, $id_first);
        } else {
            return response()->json('Ничья недоступна', 400);
        }

        Duel::create([
            'id_first' => $id_first,
            'id_second' => $id_second,
            'score_first' => $duel['score_first'],
            'score_second' => $duel['score_second'],
            'rating_first' => $rating_first,
            'rating_second' => $rating_second,
            'id_tournament' => $id_tournament,
            'index_duel' => $duel['index_duel']
        ]);
    }
}
```

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительной техники
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«10» 06 2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
Код и наименование направления

Информационная система клуба настольного тенниса
тема

Руководитель	Курш, 17.06.22 подпись, дата	доцент, канд. физ.-мат. наук должность, ученая степень	К. В. Коршун инициалы, фамилия
Выпускник	17.06.22 подпись, дата		М. К. Пройдаков инициалы, фамилия
Нормоконтролер	Курш, 17.06.22 подпись, дата	доцент, канд. физ.-мат. наук должность, ученая степень	К. В. Коршун инициалы, фамилия

Красноярск 2022