МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-приложения для записи на спортивные секции

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-374.ВКР

Научный руководитель,
профессор кафедры СП, д.г.н.,
к.ф.-м.н.
_____ С.М. Абдуллаев

Автор работы,
студент группы КЭ-404
_____ А.Ю. Шевелев

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-404
Шевелеву Андрею Юрьевичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)

Разработка веб-приложения для записи на спортивные секции.

2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.

3. **Исходные данные к работе**

3.1. Попова-Коварцева Д.А. Основы программирования баз данных: учеб. пособие / Д.А. Попова-Коварцева, Е.В. Сопченко // Самара: Изд-во Самарского университета, 2019. – 112 с.

3.2. ASP.NET. [Электронный ресурс] URL: https://dotnet.microsoft.com/en-us/apps/aspnet (дата обращения: 10.02.2024 г.).

3.3. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. // СПб.: Питер, 2021. – 352 с.

3.4. Карпович Е.Е. Методы тестирования и отладки программного обеспечения: учебник. // Москва: МИСИС, 2020. – 136 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Провести анализ предметной области.

4.2. Спроектировать и реализовать серверную часть веб-приложения.

4.3. Спроектировать и реализовать клиентскую часть веб-приложения.

4.4. Провести тестирование реализованного веб-приложения.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
профессор кафедры СП, д.г.н., к.ф.-м.н. С.М. Абдуллаев

Задание принял к исполнению А.Ю. Шевелев

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В современном мире спорт является неотъемлемой частью жизни человека. Особое значение имеют вопросы укрепления и поддержания физического и духовного здоровья. В связи с этим, каждый человек так или иначе занимается спортивной активностью. Для того, чтобы достичь хороших результатов, необходимо специальное оборудование, а также знания правильной техники выполнения упражнений, в таком случае люди обращаются в специализированные спортивные залы. Помимо индивидуальных занятий имеют большую популярность групповые виды спорта, например, футбол, хоккей и др. Об этом говорит активное проведение различных соревнований.

В случае обращения в спортивную организацию, сотрудникам организации необходимо вести учет посещений, составлять расписание занятий. Также возможны ситуации, при которых необходимо корректировать расписание, при этом нужно уведомить всех клиентов. Без автоматизации указанных процессов возникают неудобства.

Таким образом, разработка веб-приложения позволит упростить работу спортивной организации, а именно:

1) автоматизировать формирование расписания для каждого клиента;
2) уведомлять клиентов о смене расписания;
3) упростить запись клиентов на спортивные занятия.

Для правильного функционирования приложение должно состоять из серверной и клиентской частей.

Постановка задачи

Целью выпускной квалификационной работы является разработка клиент-серверного веб-приложения для записи на спортивные секции.

Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области;
2) спроектировать и реализовать серверную часть веб-приложения;
3) спроектировать и реализовать клиентскую часть веб-приложения;
4) провести тестирование реализованного веб-приложения.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 39 страниц, объем списка литературы – 16 источников.

В первой главе проводится анализ предметной области, обзор аналогичных решений и средств разработки, необходимых для реализации приложения.

Вторая глава посвящена проектированию системы. В этой главе определяются функциональные и нефункциональные требования к разрабатываемому приложению, представлена диаграмма вариантов использования и схема базы данных.

В третьей главе описывается архитектура разрабатываемого приложения и процесс разработки.

В четвертой главе приведены результаты тестирования разработанного приложения.

В заключении приведены результаты выполненной работы и описано дальнейшее развитие проекта.

В приложении приведены результаты функционального тестирования.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Средства разработки

ASP.NET [1] является технологией для создания веб-приложений на платформе .NET. Преимущества данной технологии:

1) разрабатываемые приложения могут быть развернуты на всех основных операционных системах;
2) поддержка работы с большинством распространенных СУБД;
3) технология является расширяемой, поскольку состоит из набора независимых компонентов.

Angular [2] является платформой для разработки масштабируемых веб-приложений. Особенностями данной платформы являются:

1) расширяемость приложений;
2) набор библиотек, которые охватывают широкий спектр функций;
3) взаимодействие клиент-сервер.

Обзор аналогичных приложений

В качестве примеров существующих аналогичных решений, которые предоставляют запись на секции, были выбраны три сайта, описание которых приведено ниже.

Веб-сайт «Карта спорта»

«Карта спорта» [3] содержит большую базу секций и краткую информацию о них, а также сведения о тренерах, отличительной особенностью можно выделить то, что сайт позволяет не только записаться на занятия в зале, но и полностью арендовать весь зал. Все секции разделены по категориям, что позволяет их удобно фильтровать. В качестве недостатка можно отметить то, что нельзя записаться непосредственно на сайте – необходимо связываться с организаторами по телефону. На рисунке 1 изображена главная страница сайта «Карта спорта». Данный сайт позиционирует себя самой большой базой спортивных учреждений. На главной странице расположены тематические изображения и блок, содержащий недавно добавленные секции.

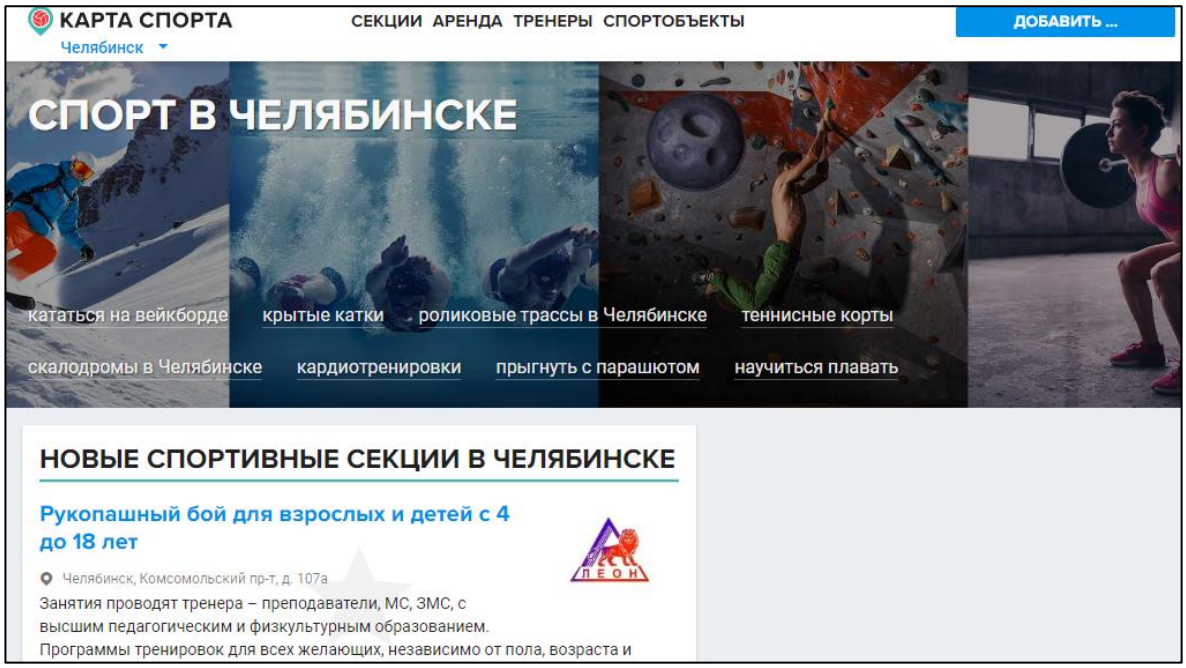

Рисунок 1 – Веб-сайт «Карта спорта»

Веб-сайт «После уроков»

Сайт «После уроков» [4] включает в себя большое количество занятий, не только спортивных, но и музыкальных, художественных и многих других. Аналогично предыдущему сайту, присутствует разделение занятий по категориям. Каждое занятие имеет отдельную страницу с подробной информацией. На сайте можно отправить заявку на запись, перед этим необходимо пройти авторизацию. Главная страница сайта изображена на рисунке 2.

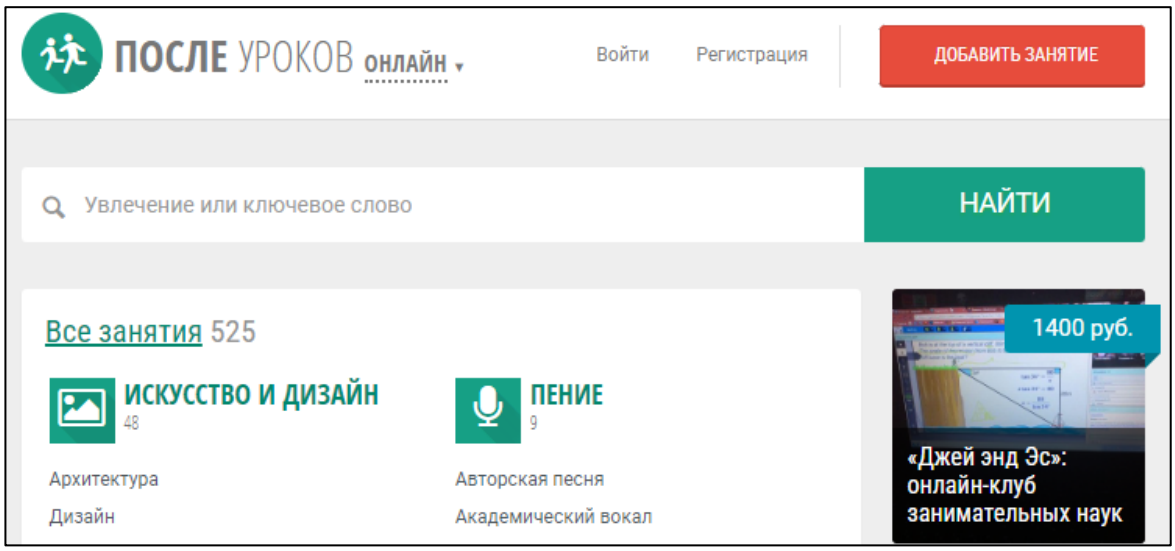

Рисунок 2 – Веб-сайт «После уроков»

Веб-сайт «Шилопоп»

Сайт [5] предлагает множество занятий из разных областей. Страницы со сведениями о секции являются наиболее информативными, в сравнении с предыдущими аналогами. Здесь можно посмотреть место проведения на карте, прочитать отзывы других клиентов. Также, как и на предыдущих сайтах, есть возможность добавить свою секцию. Аналогично первому сайту – для записи на секцию необходимо связаться с организатором по телефону. Главная страница сайта представлена на рисунке 3.

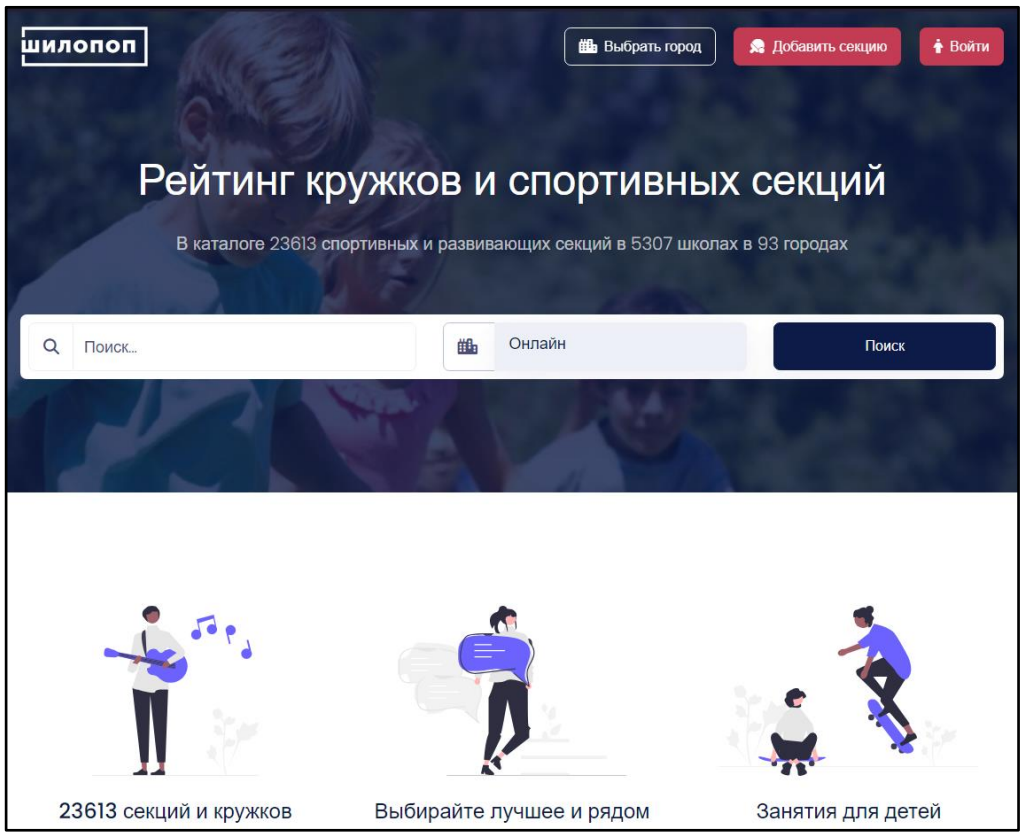

Рисунок 3 – Веб-сайт «Шилопоп»

Вывод по обзору аналогичных решений

Во всех приведенных аналогичных приложениях нет возможности записаться на занятия в один клик. Для записи необходимо отправить заявку и ожидать подтверждения, или связаться с организатором по телефону. Таким образом, в разрабатываемом приложении отличительной особенностью должна являться возможность записи на веб-сайте, не связываясь с организатором.

2. ПРОЕКТИРОВАНИЕ

2.1. Анализ требований

Перед началом разработки необходимо определить требования, которым должно соответствовать разрабатываемое приложение. Эти требования можно разделить на две группы – функциональные и нефункциональные.

Функциональные требования определяют, каким должно быть поведение продукта в тех или иных условиях.

Функциональные требования, которым должно соответствовать разрабатываемое приложение, приведены ниже.

1. Пользователь должен иметь возможность регистрации и авторизации.
2. Пользователь должен иметь возможность просматривать список секций.
3. Пользователь должен иметь возможность записаться на секцию.
4. Пользователь должен иметь возможность отменить запись на секцию.
5. Пользователь должен иметь возможность записаться на индивидуальное занятие.
6. Пользователь должен иметь возможность отменить запись на индивидуальное занятие.
7. Пользователь должен иметь возможность просматривать расписание занятий.
8. Тренер должен иметь возможность добавить секцию.
9. Тренер должен иметь возможность добавить индивидуальное занятие.
10. Тренер должен иметь возможность добавлять расписание занятий.

Нефункциональные требования – требования, определяющие свойства, которые система должна демонстрировать, или ограничения, которые

она должна соблюдать, не относящиеся к поведению системы. Нефункциональные требования приведены ниже.

1. Сервер должен быть написан на платформе ASP.NET.
2. Данные должны храниться в базе данных.
3. В качестве СУБД необходимо использовать PostgreSQL [6].
4. Клиент должен быть написан на платформе Angular.
5. Отображение расписания должно выполняться с помощью библиотеки FullCalendar [7].

2.2. Диаграмма вариантов использования

На основании сформулированных функциональных и нефункциональных требований составлена диаграмма вариантов использования. Диаграмма приведена на рисунке 4.

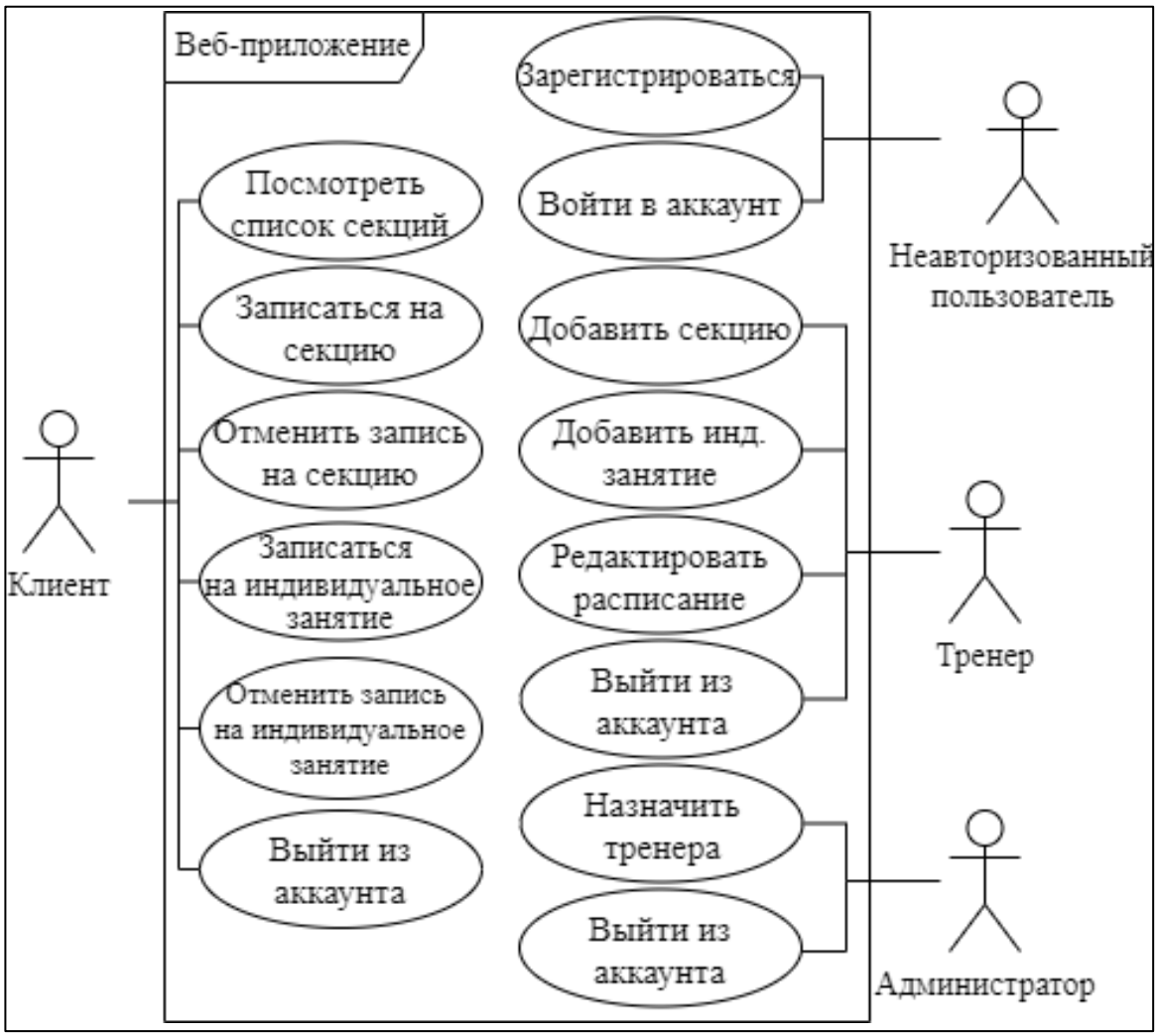

Рисунок 4 – Диаграмма вариантов использования

В разрабатываемой системе выделено 4 основных актера:

1) «Клиент» – пользователь, который выполнил вход в аккаунт клиента;

2) «Тренер» – пользователь, который выполнил вход в аккаунт тренера;

3) «Неавторизованный пользователь» – любой посетитель сайта, который не выполнил вход в аккаунт;

4) «Администратор» – владелец сайта или его доверенное лицо.

Ниже приведено описание основных вариантов использования.

Неавторизованный пользователь

Действия, которые может выполнять неавторизованный пользователь.

1. Зарегистрироваться – пользователь выполняет регистрацию и становится клиентом.

2. Войти в аккаунт – пользователь выполняет вход в аккаунт.

Клиент

Действия, которые может выполнять клиент.

1. Посмотреть список секций – пользователь может ознакомиться со всеми доступными секциями.

2. Записаться на секцию – пользователь может записаться на занятия выбранной секции.

3. Отменить запись на секцию – пользователь может отменить ранее выполненную запись на секцию.

4. Записаться на индивидуальное занятие – пользователь может записаться на индивидуальное занятие с тренером.

5. Отменить запись на индивидуальное занятие – пользователь может отменить ранее выполненную запись на индивидуальное занятие.

6. Выйти из аккаунта – клиент может закончить работу с сайтом и выйти из аккаунта.

Тренер

Действия, которые может выполнять тренер.

1. Добавить секцию – тренер может добавить новую секцию, при этом необходимо будет заполнить некоторую информацию о секции, например время проведения, вид спорта и т.д.

2. Добавить инд. занятие – тренер может добавить индивидуальное занятие.

3. Редактировать расписание – тренер может добавлять или удалять расписание занятий.

4. Выйти из аккаунта – тренер может закончить работу с сайтом и выйти из аккаунта.

Администратор

Поскольку тренер имеет больше ответственности, в отличие от обычного пользователя, который является клиентом, то необходимо обеспечить контроль создания тренерских аккаунтов. Для этой цели выделен актер «Администратор», имеющий следующие действия:

1) назначить тренера – администратор может выбрать клиента из списка и сделать его тренером;

2) выйти из аккаунта – администратор завершает работу с системой и выходит из аккаунта.

2.3. Проектирование базы данных

Поскольку есть необходимость хранить достаточно большой объем данных, то необходимо разработать базу данных, причем данные должны быть актуальными для всех пользователей, следовательно база данных должна находиться на сервере. Для того, чтобы работать с базой данных необходимо использовать СУБД. Выбор пал на PostgreSQL. Далее приведены достоинства данной СУБД.

1. СУБД находится в открытом доступе и распространяется бесплатно.

2. Можно установить на все основные операционные системы.
3. Позволяет хранить большие объемы данных.
4. Производительность и широкий функционал.

При разработке будет применяться подход Code First [8], суть которого заключается в том, что сначала пишется код, описываются определенные сущности, затем на основании сущностей автоматически генерируется база данных. Взаимодействие с базой данных будет выполняться посредством технологии Entity Framework [9]. На рисунке 5 приведена ER-диаграмма базы данных.

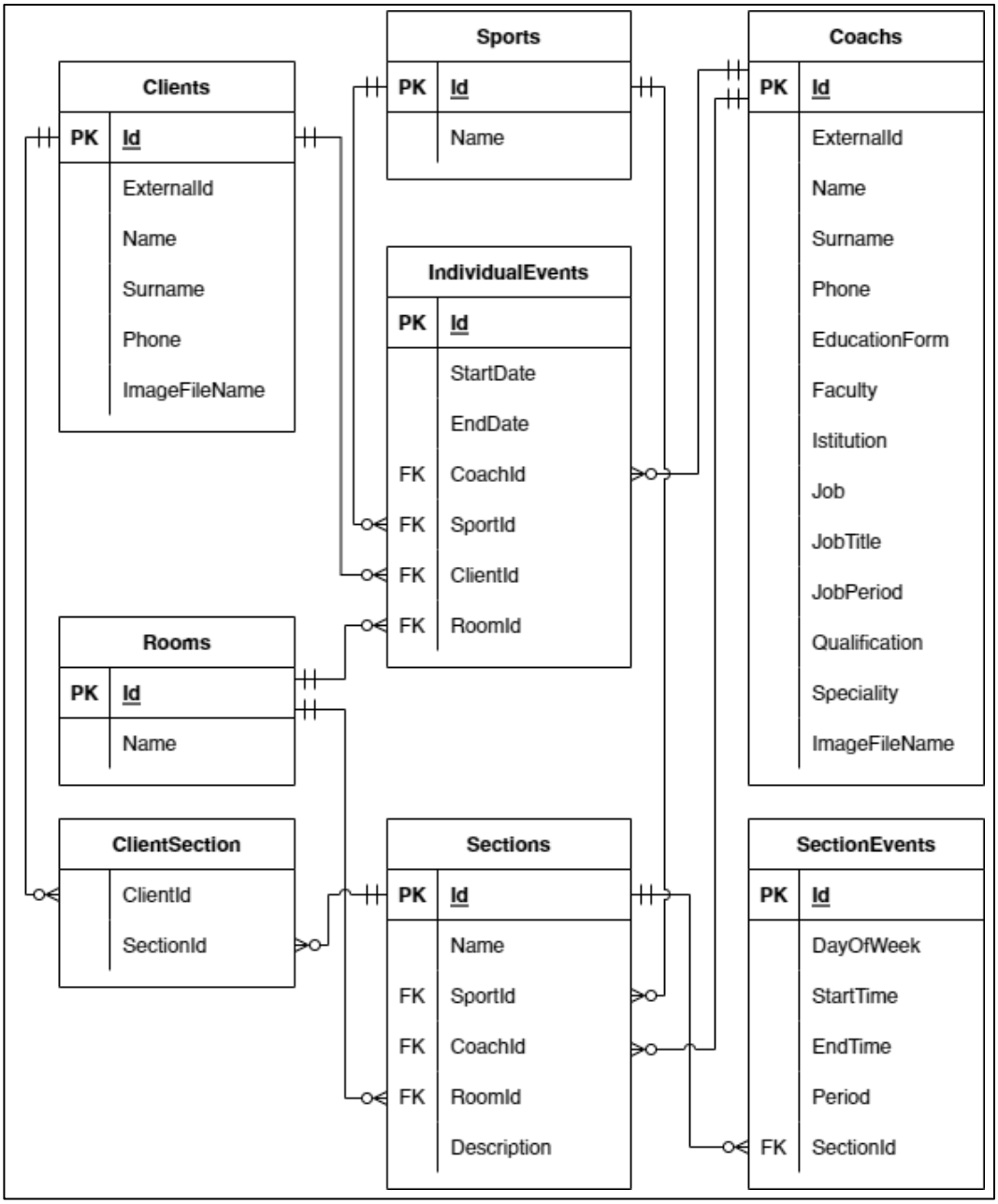

Рисунок 5 – ER-диаграмма базы данных

Ниже представлено описание таблиц базы данных.

1. В таблице `Coachs` хранится информация о тренерах. Таблица включает фамилию и имя тренера, контактный телефон, информацию об образовании и карьере.

2. Таблица `Clients` хранит информацию о клиентах. Включает фамилию и имя клиента и телефон для связи.

3. Таблица `Rooms` хранит предопределенные сведения о помещениях или площадках, в которых могут проводиться занятия. За секцией может быть закреплено только одно помещение, при этом в одном помещении могут проводить несколько разных секций. Связь один ко многим.

4. Таблица `Sports` хранит предопределенные сведения о видах спорта. Аналогично предыдущему пункту, секция может относиться только к одному виду спорта, но определенный вид спорта может быть закреплен за несколькими секциями, а значит связь так же один ко многим.

5. Таблица `Sections` хранит сведения о секциях. Включает название секции, вид спорта, зал, в котором проводятся занятия и идентификатор тренера, который проводит занятия. Секция может проводиться только одним тренером, но один тренер может вести несколько секций, следовательно определяется связь один ко многим. В одной секции может быть несколько клиентов и один клиент может состоять в нескольких секциях, отсюда связь между клиентом и секцией – многие ко многим.

6. Таблица `ClientSection` является связующей, поскольку между клиентом и секцией связь многие ко многим.

7. Таблица `SectionEvents` хранит сведения о занятиях секций, время, день, период проведения.

8. Таблица `IndividualEvents` хранит сведения об индивидуальных занятиях. На конкретное занятие может быть записан только один клиент, при этом клиент может записаться на несколько разных занятий, наблюдается связь один ко многим.

2.4. Архитектура разрабатываемой системы

Разрабатываемое веб-приложение имеет клиент-серверную архитектуру. На рисунке 6 приведена диаграмма компонентов разрабатываемого приложения.

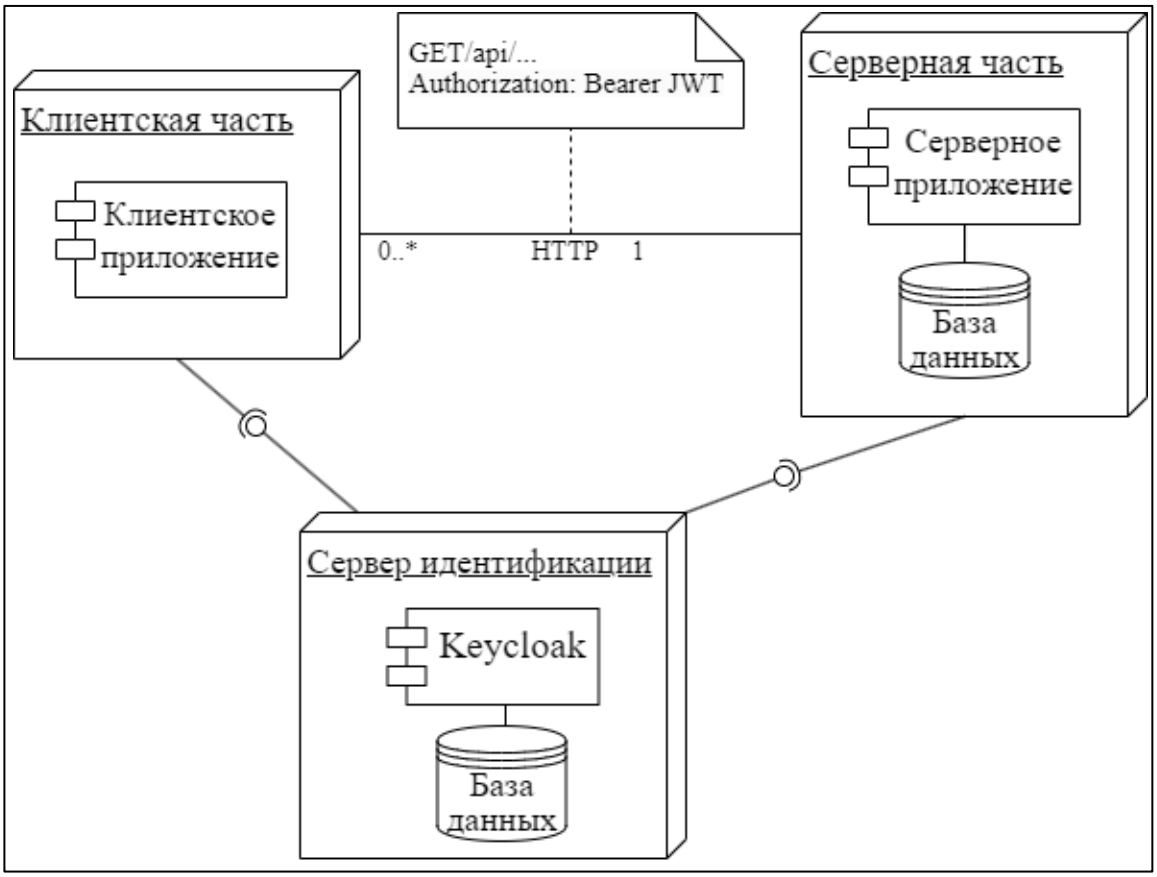

Рисунок 6 – Диаграмма компонентов

Ниже приведено описание основных компонентов, которые включает в себя клиент-серверная архитектура.

1. Клиент – устройство или приложение, которое отправляет серверу запрос на выполнение операции.

2. Сервер – компьютер, который принимает запросы от клиентов и предоставляет запрашиваемую информацию или выполняет необходимую операцию.

3. Протокол обмена данными – набор правил, которые определяет каким образом клиенты и серверы обмениваются данными.

4. Сервер идентификации – компонент, обеспечивающий авторизацию и аутентификацию клиентов, необходим для предотвращения несанкционированного доступа к системе.

Серверная часть приложения и сервер идентификации используют хранилище данных, которые могут находиться как на одном, так и на двух отдельных серверах. Исходя из вышесказанного, можно сделать вывод, что архитектура имеет многоуровневый тип.

Архитектура серверной части

Архитектура серверной части основана на принципах чистой архитектуры [10]. Ключевым принципом такой архитектуры является разделение разрабатываемой системы на уровни. Круги, представленные на рисунке 7 представляют уровни системы. Главным правилом является то, что зависимости должны быть направлены внутрь. Ни один из внутренних уровней не знает о существовании внешних, и данные, которые были объявлены во внешних уровнях, не должны использоваться во внутренних.

Рисунок 7 – Чистая архитектура

Следует отметить, что диаграмма, приведенная на рисунке 7 отображает идею лишь схематически и на самом деле ограничения по числу уровней не существует.

Архитектура клиентской части

Клиентская часть спроектирована на основании компонентного подхода. Компонент является частью приложения и содержит собственную логику. Суть компонентного подхода состоит в следующем:

1) для каждого компонента генерируется собственный HTML-шаблон, в котором описывается дизайн страницы.

2) для каждого компонента генерируется файл стилей (`css`, `scss`);

3) логика компонента описывается в `TypeScript` файле.

Также для взаимодействия с серверной частью рекомендуется реализовать специальные сервисы. Сервис необходим для предоставления данных компонентам и должен быть узконаправленным. На рисунке 8 приведена архитектура клиентской части.

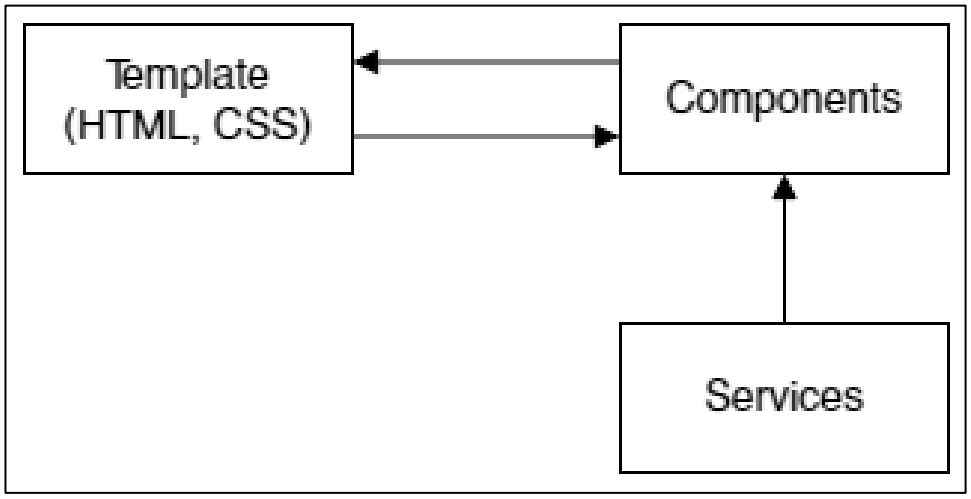

Рисунок 8 – Архитектура клиентского приложения

3. РЕАЛИЗАЦИЯ

3.1. Настройка базы данных

Как было сказано ранее, при разработке приложения будет применяться подход Code First. Подход Code First упрощает процесс разработки и обеспечивает более высокий уровень абстракции при работе с базой данных.

Для описания сущностей была реализована базовая сущность, приведенная в листинге 1, от которой наследуются все остальные. Базовая сущность содержит только поле `Id`, которое определяет идентификатор сущности. В качестве параметра `T` указывается тип данных, который будет иметь поле `Id`.

Листинг 1 – Класс `BaseEntity`

```
public abstract class BaseEntity<T>
{
    public T Id { get; set; } = default!;
}
```

В листинге 2 приведен пример сущности, которая наследуется от базовой. В данном случае идентификатор будет иметь тип `Guid`. Также сущность имеет навигационное поле `Section`, необходимое для описания связей.

Листинг 2 – Сущность `Sport`

```
public class Room : BaseEntity<Guid>
{
    public string Name { get; set; } = string.Empty;
    public List<Section>? Section { get; set; }
}
```

При подходе Code First таблицы и сущности сопоставляются с помощью правил, определенных в Entity Framework, но иногда необходимо переопределить логику этих правил. В листинге 3 приведен пример настройки связи многие ко многим с использованием Fluent API [11].

Листинг 3 – Настройка связей между сущностями

```
modelBuilder.Entity<Section>()
    .HasMany(s => s.Client)
    .WithMany(c => c.Section)
    .UsingEntity(j => j.ToTable("ClientSection"));
```

3.2. Реализация серверной части

Для доступа к разрабатываемой системе необходимо выполнять аутентификацию пользователей. Также необходимо разграничивать пользователей на клиентов и тренеров. В качестве инструмента для идентификации пользователей был выбран Keycloak [12]. Keycloak – это продукт с открытым исходным кодом. Основные возможности данного инструмента:

1) регистрация и аутентификация пользователей;
2) выдача JWT-токена для проверки подлинности аккаунта;
3) обеспечение единой точки входа.

Помимо этого, Keycloak поддерживает настройку клиентов – приложений, которые могут использовать Keycloak для аутентификации, а также областей, в которых определяются пользователи, их роли, группы и др.

После того, как пользователь прошел аутентификацию, он получает доступ к функционалу приложения.

При разработке серверной части использовался подход REST API [13]. Клиентская часть взаимодействует с серверной посредством HTTP запросов. Ниже представлен пример создания секции посредством метода POST.

В первую очередь необходимо определить шаблон объекта передачи данных (DTO). В листинге 4 приведен пример шаблона передачи данных для метода создания секции.

Листинг 4 – Класс `CreateSectionModel`

```
public class CreateSectionModel
{
    public required IFormFile Image { get; set; }
    public required string Name { get; set; }
    public required string Description { get; set; }
    public required Guid SportId { get; set; }
    public required Guid RoomId { get; set; }
}
```

Из листинга 4 видно, что для создания новой секции необходимо задать обязательные поля `Name` – название секции, `Description` – описание секции, `SportId` и `RoomId` – уникальные идентификаторы вида спорта и зала

соответственно, а также поле Image – тематическое изображение, которое будет отображаться при просмотре информации о секции.

Для выполнения бизнес-логики используется библиотека MediatR, поэтому после создания шаблона передачи данных необходимо создать класс, реализующий интерфейс IRequest. Данный класс не содержит логики и представляет из себя контейнер данных, необходимых для выполнения заданных операций. В разрабатываемом приложении эти классы обозначены как команды. В листинге 5 приведено объявление команды для создания новой секции.

Листинг 5 – Класс CreateSectionCommand

```
public class CreateSectionCommand
: IRequest<CreatedOrUpdatedEntityViewModel<Guid>>
{
    [FromForm]
    public required CreateSectionModel Body { get; set; }
}
```

Из листинга 5 видно, что CreateSectionCommand наследуется от интерфейса IRequest<TResponse>, где TResponse – возвращаемый тип данных. В качестве возвращаемого типа данных указана модель представления данных CreatedOrUpdatedEntityViewModel<Guid>, данная модель используется в качестве результата выполнения операции создания или обновления данных сущности. Эта модель содержит только идентификатор сущности, которая была создана или обновлена.

Далее необходимо определить обработчик запроса, который будет выполнять заданную операцию. Обработчик должен реализовывать интерфейс IRequestHandler<TCommand, TResponse>, где TCommand – команда, описывающая объект запроса, TResponse – тип возвращаемых данных, не является обязательным. В листинге 6 приведен пример кода обработчика запроса для создания новой секции.

Листинг 6 – Обработчик SectionCommandsHandler

```
public async Task<CreatedOrUpdatedEntityViewModel<Guid>>
Handle(CreateSectionCommand request, CancellationToken cancellationToken)
{  var coach = await coachService
   .GetCoachAync(contextAccessor.IdentityUserId, false, cancellationToken);
```

```
    var sectionWithSameName = await dbContext.Sections
        .Where(x => x.Name == request.Body.Name)
        .SingleOrDefaultAsync(cancellationToken);
    if (sectionWithSameName != null)
    {
        throw new BusinessLogicException(
            $"Секция с названием \"{sectionWithSameName.Name}\" уже
            существует!");
    }
    var sport = await sportService.GetSportAsync(request.Body.SportId,
cancellationToken);
    var room = await roomService.GetRoomAsync(request.Body.RoomId,
cancellationToken);
    var sectionToCreate = sectionMapper.MapToEntity((request.Body,
coach.Id));
    var createdSection = await dbContext.AddAsync(sectionToCreate,
cancellationToken);
    if (request.Body.Image != null)
    {
        createdSection.ImageFileName = imageService
       .SaveSectionImage(request.Body.Image, null);
    }
    await dbContext.SaveChangesAsync(cancellationToken);
    return new CreatedOrUpdatedEntityViewModel(createdSection.Entity.Id);
}
```

Далее при обращении к контроллеру, можно выполнить созданный запрос, пример демонстрируется в листинге 7.

Листинг 7 – Выполнение запроса

```
[HttpPost]
public async Task<CreatedOrUpdatedEntityViewModel<Guid>>
CreateSection([FromForm] CreateSectionCommand command, CancellationToken
cancellationToken)
{
    return await sender.Send(command, cancellationToken);
}
```

3.3. Реализация клиентской части

Клиентским приложением является веб-сайт – приложение на платформе Angular. Разрабатываемый веб-сайт является многостраничным, поэтому необходимо установить маршруты для каждого компонента – страницы сайта. В листинге 8 приведен пример настройки маршрутизации.

Листинг 8 – Установка маршрутов

```
export const routes: Routes = [
  { path: '', component: BaseComponent},
  { path: 'sections/page/:page', component: SectionsComponent, canActivate:
         [AuthGuard]},
  { path: 'add-event', component: AddEventComponent, canActivate:
         [AuthGuard], data: {roles: ["Coach"]} },
  { path: 'schedule', component: ScheduleComponent, canActivate:
         [AuthGuard] } ];
```

Реализация главной страницы

Главная страница, приведенная на рисунке 9, является по большей части информационной страницей. Ее цель – заинтересовать потенциального клиента, путем демонстрирования важных особенностей. На главной странице разрабатываемого приложения расположен баннер, содержащий название, слоган, а также кнопку регистрации. Кнопка регистрации будет отображена только для неавторизованных пользователей, после входа в аккаунт отображение кнопки будет отключено. В верхней части сайта реализовано навигационное меню, а также расположено фото профиля пользователя с выпадающим меню, из которого можно перейти в настройки профиля или же выйти из аккаунта. Навигационное меню сайта будет закреплено на всех остальных страницах.

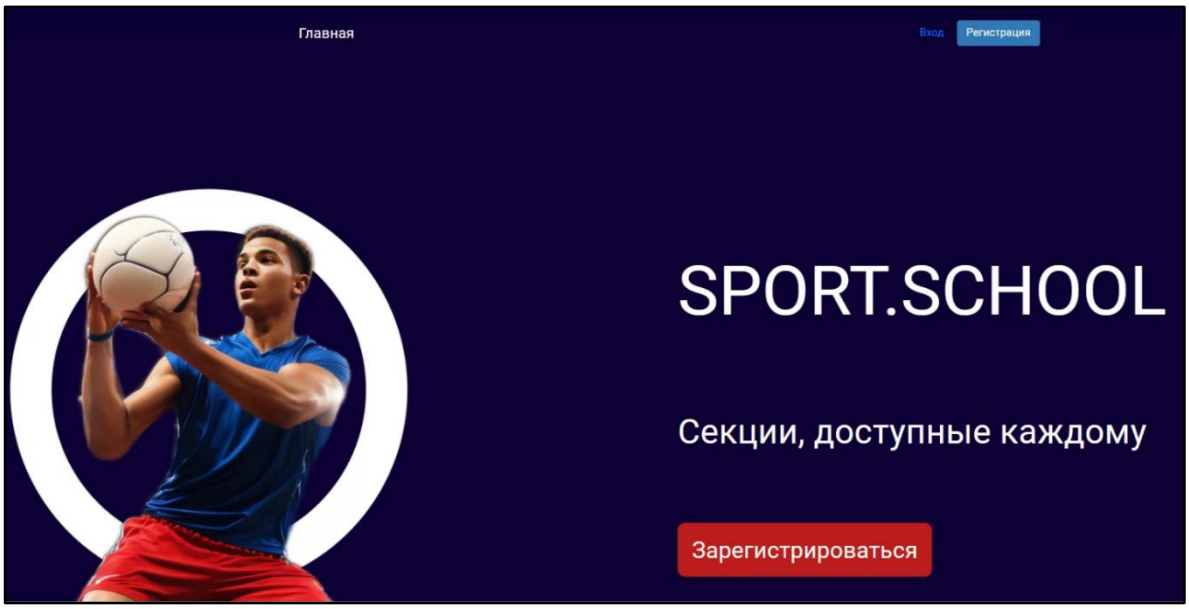

Рисунок 9 – Дизайн главной страницы

Следует отметить то, что если пользователь не выполнил вход в аккаунт, то в навигационном меню будут скрыты ссылки, ведущие на другие страницы сайта.

Ниже расположен блок с краткой информацией, способной замотивировать клиента записаться на занятие. Здесь отображены основные достоинства, которые должны заинтересовать клиента.

В самом низу страницы расположен элемент «карусель» с изображениями тренеров.

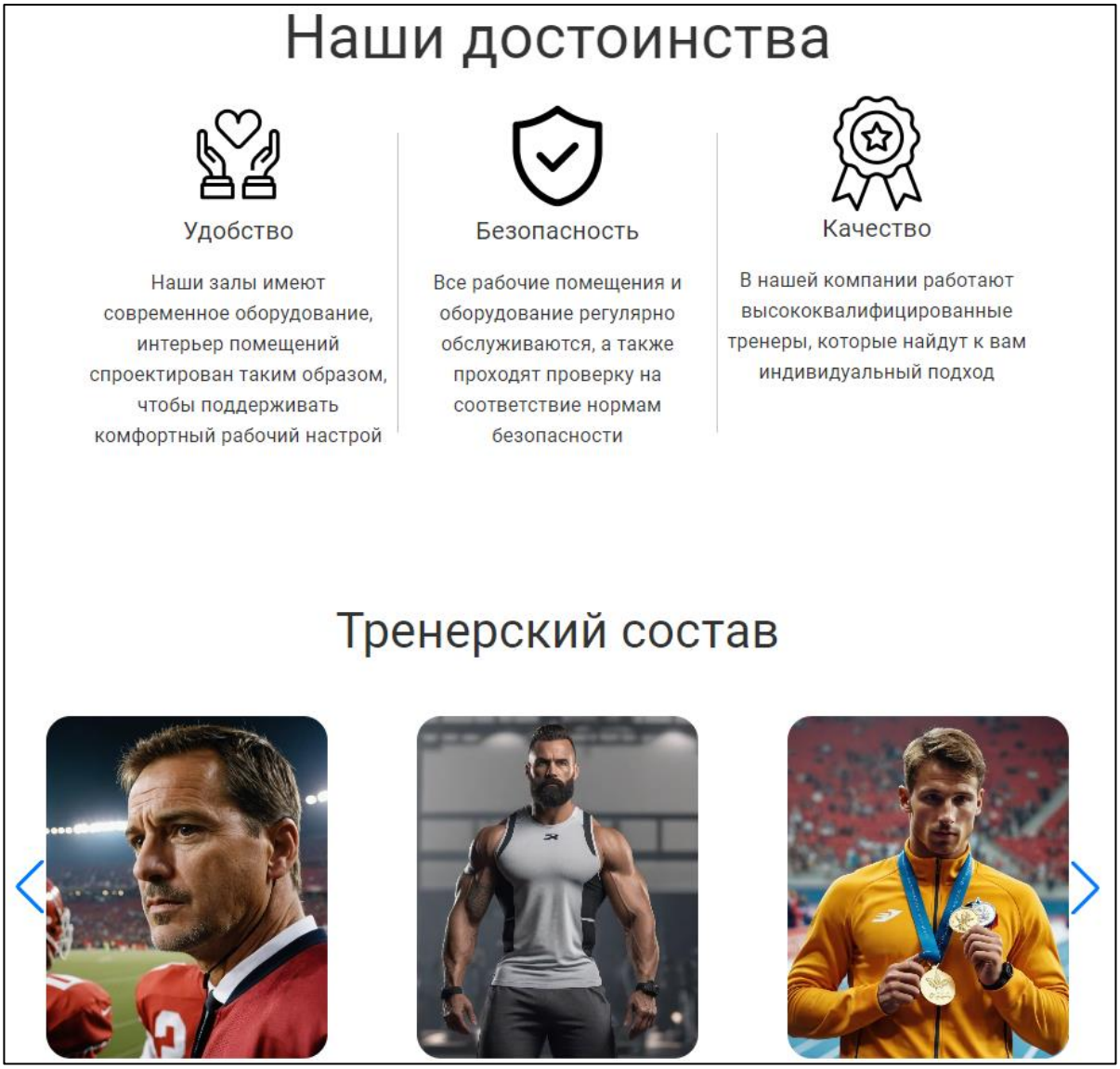

Рисунок 10 – Блоки «достоинства» и «тренерский состав»

При нажатии на изображение тренера открывается всплывающее окно, с информацией о нем. В этом окне приведена информация об образовании и карьере тренера. Все данные, в том числе изображения хранятся на сервере. Изображения хранятся в качестве статических файлов. Для отображения изображений в виде «карусели» был использован веб-компонент `Swiper Element` [14].

Реализация регистрации

В качестве формы авторизации использована встроенная форма Keycloak, которая содержит поле ввода электронной почты и пароля. Для регистрации реализована собственная форма, в которой пользователь указывает электронную почту, пароль, номер телефона, имя и фамилию. Для ввода телефона используется маска ввода «+7(***) ***-**-**» из модуля `ngx-mask` [15]. Также пользователь может загрузить фото профиля. Демонстрация формы регистрации на рисунке 11.

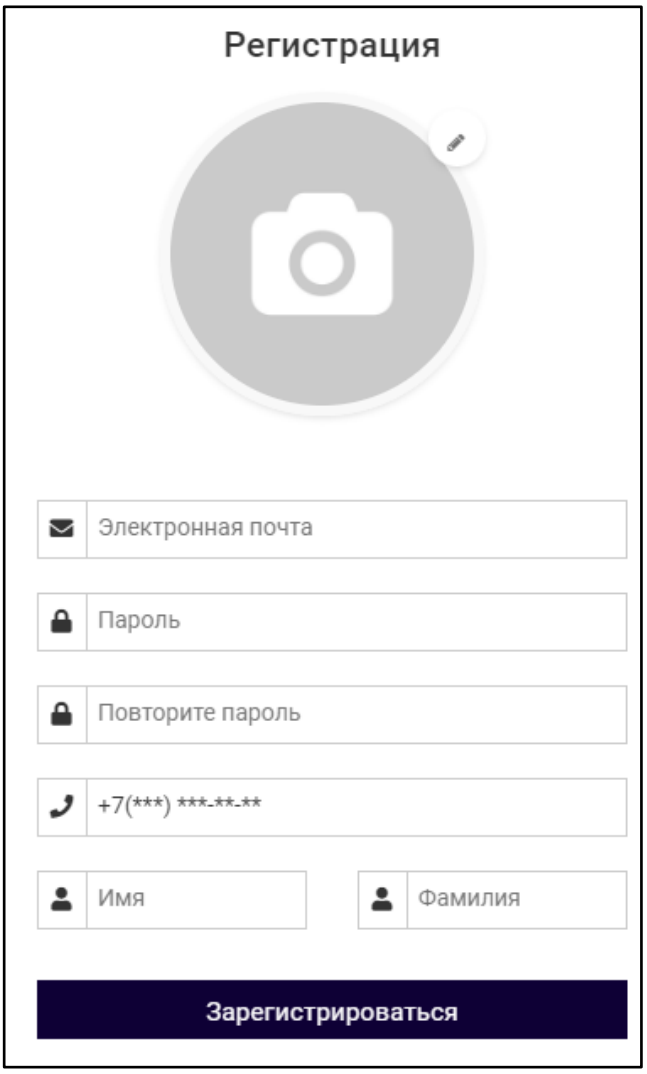

Рисунок 11 – Форма регистрации

После того как пользователь загрузит фотографию со своего устройства, необходимо изменить отображаемое фото. Для этой цели реализована соответствующая функция на языке `JavaScript`, приведенная в листинге 9.

Листинг 9 – Обработчик смены изображения

```
function readURL(input) {
    if (input.files && input.files[0]) {
        var reader = new FileReader();
        reader.onload = function(e) {
            $('#imagePreview').css('background-image', 'url('+e.target.re-
sult +')');
            $('#imagePreview').hide();
            $('#imagePreview').fadeIn(650);
        }
        reader.readAsDataURL(input.files[0]);
    }
}

$("#imageUpload").change(function() {
    readURL(this);
});
```

Реализация страницы списка занятий

На рисунке 12 приведена страница занятий. В центре страницы изображены карточки с информацией о занятии, а левую часть страницы занимает блок с фильтрами.

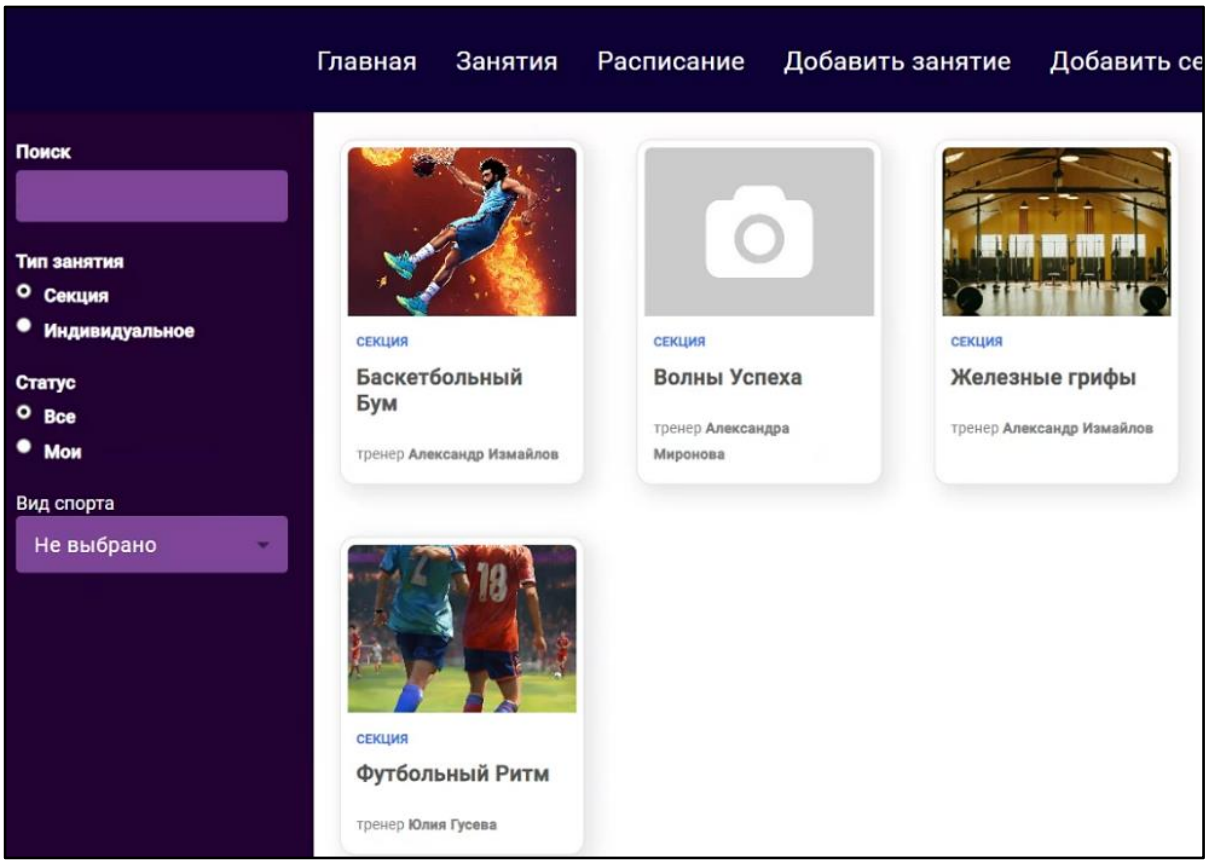

Рисунок 12 – Список занятий

Ниже приведено описание фильтров.

1. Поиск – поле ввода для фильтрации занятий по ключевым словам.
2. Тип занятия – фильтрация по типу занятия. Проводимое занятие может быть индивидуальным занятием с тренером или занятием секции.
3. Статус – фильтр занятий по статусу записи. Выделено два фильтра. Фильтр «Все» – отображает все доступные занятия, независимо от того записан ли на них пользователь. Фильтр «Мои» отображает только те занятия, на которые записан пользователь. Если фильтр «Мои» установит тренер, то для него будут отображены только те занятия, которые он проводит.
4. Вид спорта – фильтр по виду спорта, выбранному из списка.

Основную часть страницы занимают карточки занятий. На карточке изображена тематическая картинка, загруженная тренером при создании секции, название занятия, категория, к которой относится проводимое занятие, а также имя тренера, который проводит указанное занятие. Для занятия секции приведено ее название, а для индивидуального занятия в качестве названия указан вид спорта.

Реализация страницы занятия

При нажатии на карточку занятия, открывается новая страница, содержащая информацию о выбранном занятии. На этой странице приведено краткое описание занятия, указано каким тренером проводится данное занятие, расположена кнопка, позволяющая записаться на занятие или отменить запись, если пользователь уже записан. Если же страницу с занятием просматривает тренер, создавший эту секцию, то для него отображена только одна кнопка, позволяющая удалять секцию. Демонстрация приведена на рисунке 13. Если тренер не добавит ни одного занятия для секции, то пользователь увидит сообщение о том, что для этой секции не добавлено занятий, а кнопка «Записаться» будет недоступна.

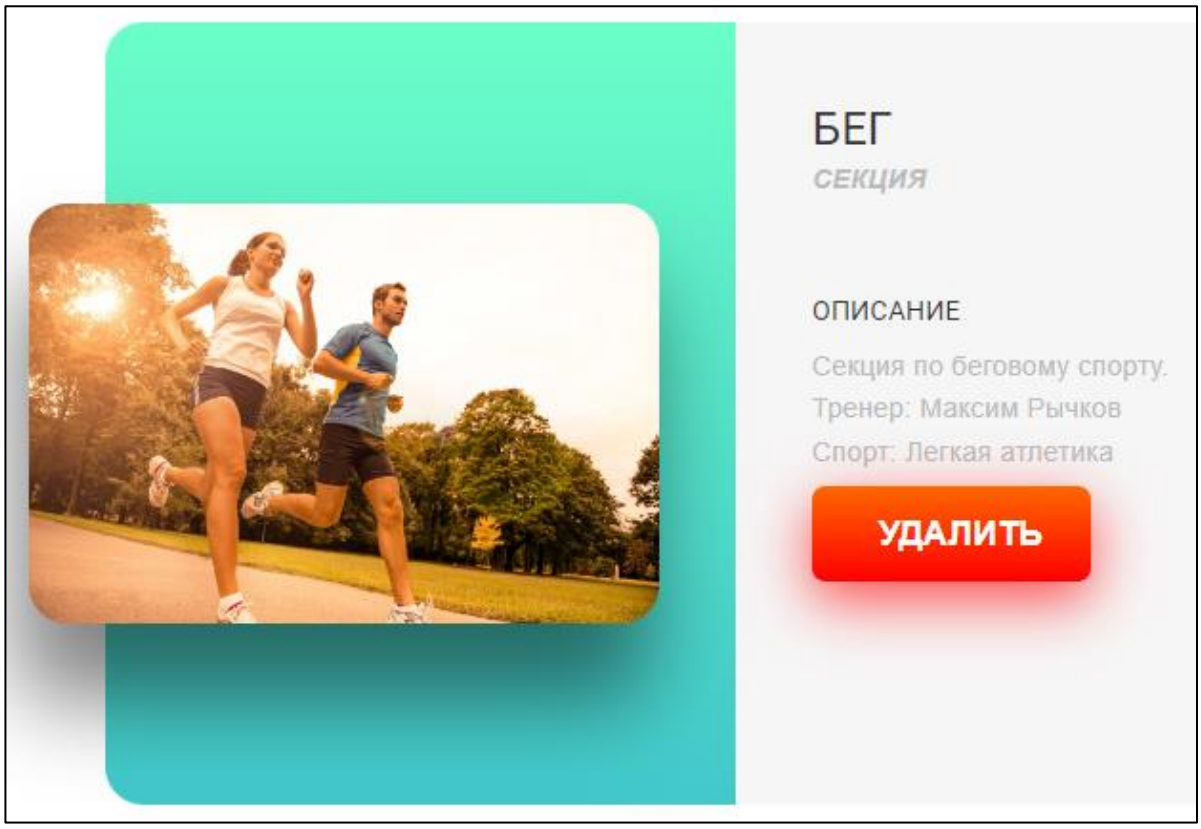

Рисунок 13 – Страница с информацией о занятии

Реализация страницы расписания

Основной функцией веб-сайта является возможность просмотра расписания занятий. После того, как пользователь зарегистрировался или выполнил вход в аккаунт, а также записался на индивидуальное занятие или секцию, он может перейти на страницу с расписанием. На этой странице расположен календарь, в котором отображаются все занятия, на которые записан пользователь. Календарь имеет сетку, которая разделяет его на временные промежутки. Одно деление сетки соответствует 30 минутам. Пользователь имеет возможность переключать представление календаря на день и неделю. Календарь изображен на рисунке 14. Если календарь просматривает тренер, то для него отображаются только те занятия, которые он проводит.

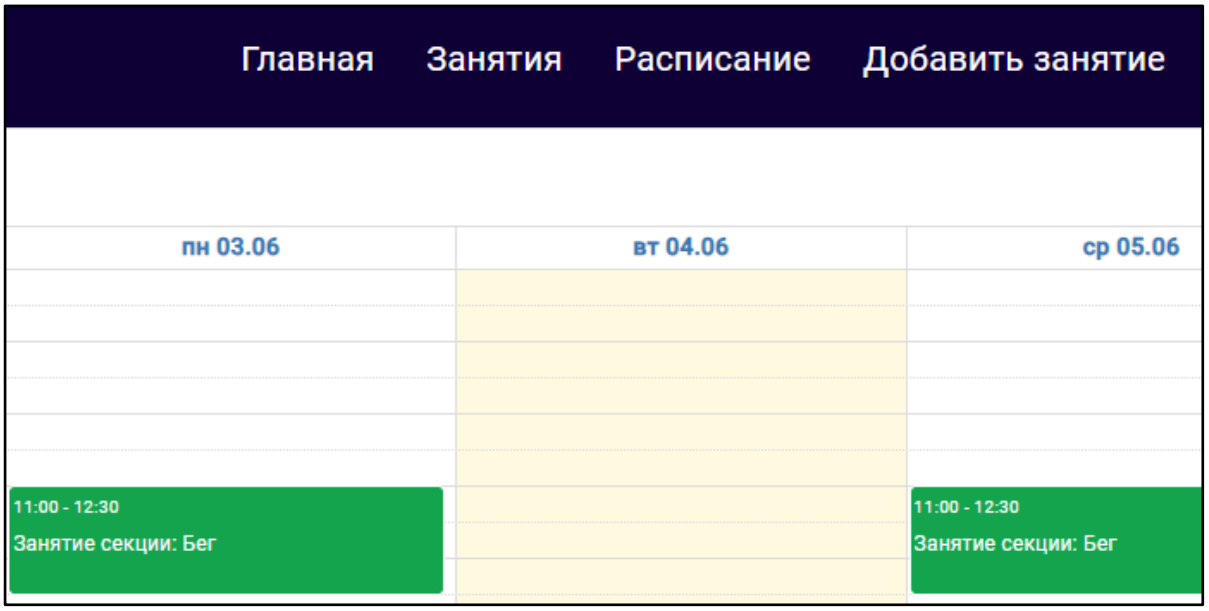

Рисунок 14 – Страница расписания занятий

В календаре все занятия выделены цветом. Занятия секций всегда имеют зеленый цвет. Индивидуальные занятия, которые добавил тренер, но на которые еще не записался ни один клиент выделены серым цветом для информирования тренера. Индивидуальные занятия, на которые уже записался клиент отмечены зеленым цветом.

При нажатии на событие, изображенное в календаре, на экране появляется всплывающее окно, в котором приведена детальная информация о проводимом занятии, а именно:

1) тип занятия – занятие секции или индивидуальное занятие;
2) время проведения – время начала и окончания занятия;
3) вид спорта;
4) место проведения;
5) данные тренера – имя, фамилия и контактный телефон.

При просмотре от лица тренера доступна кнопка, позволяющая удалить проводимое занятие. Пример приведен на рисунке 15.

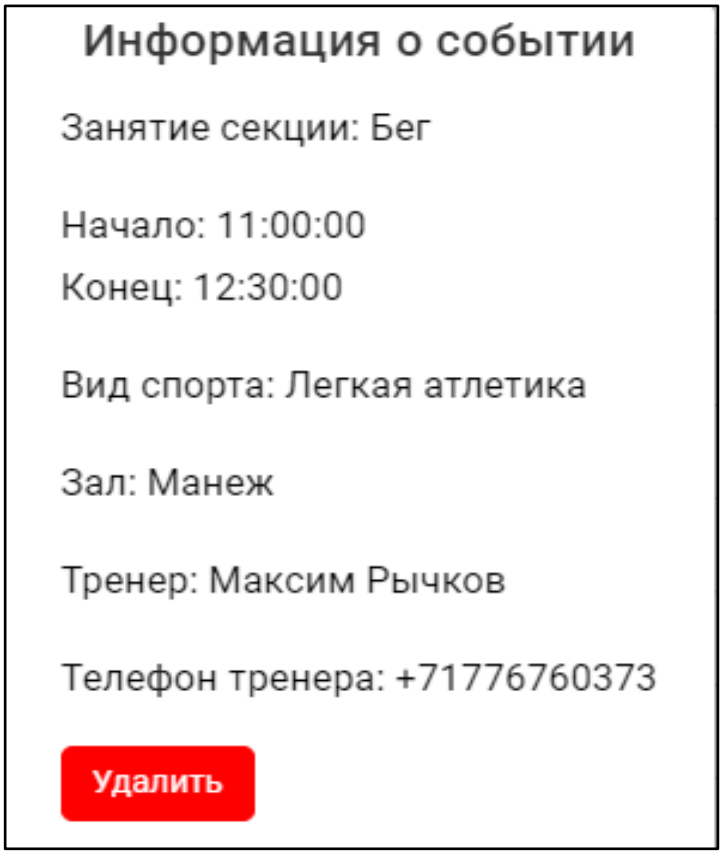

Рисунок 15 – Информация о событии

Реализация добавления секции

Секция представляет из себя группу людей, в которую может записаться любой клиент. Каждый тренер может добавить собственную секцию. После чего, для каждой секции тренеру необходимо составить расписание занятий, в противном случае у клиентов не будет возможности записаться на секцию.

Для создания новой секции реализована форма, в которой тренеру необходимо загрузить тематическую картинку, указать название секции, вид спорта, к которому относится секция и зал, в котором будут проводиться занятия. Название тренер указывает сам, а вид спорта и зал выбирает из предопределенных списков. Также тренер может указать описание секции. Демонстрация на рисунке 16.

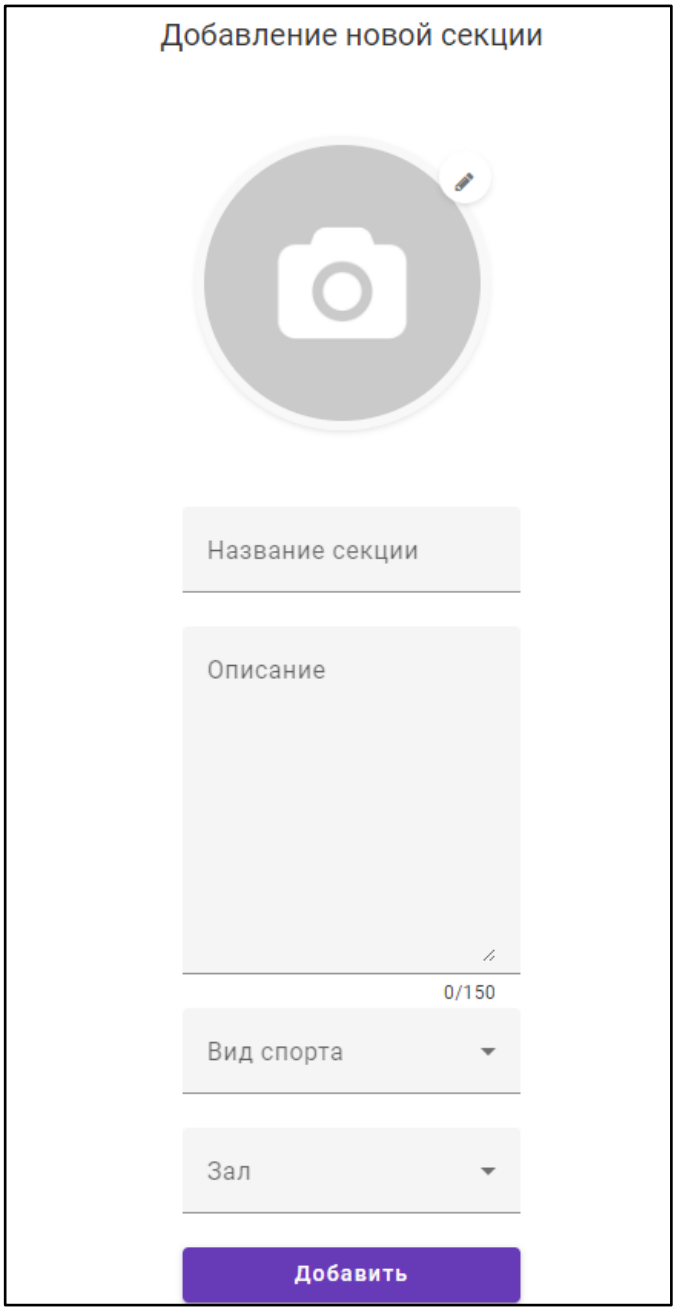

Рисунок 16 – Форма добавления секции

Реализация добавления расписания

Тренер должен иметь возможность добавлять расписание занятий. Причем занятия могут проводиться как в секциях, так и индивидуально. Для добавления расписания было реализовано модальное окно, в котором указываются сведения о проводимом занятии. Чтобы разграничить добавление расписания для индивидуальных занятий и занятий секции, была реализована возможность переключения с помощью кнопок. На рисунке 17 представлено окно для заполнения расписания секции.

Для добавления занятий секции тренеру необходимо указать день недели, время проведения и дату (период), до которой будут проводиться занятия секции. Далее расписание будет формироваться автоматически. Для индивидуальных занятий необходимо указать полную дату, то есть тренеру необходимо каждый раз создавать новое событие для индивидуальных занятий.

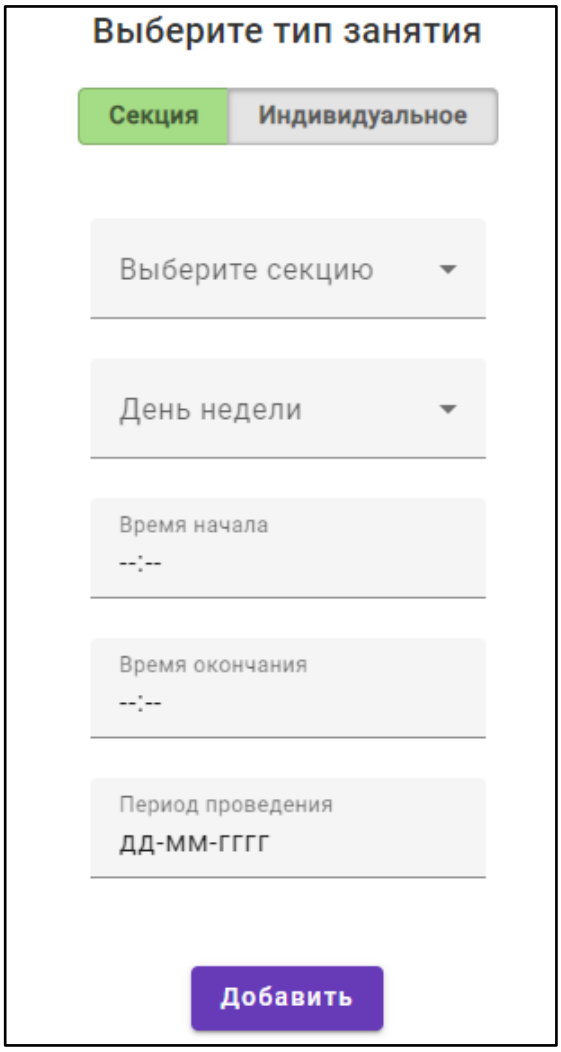

Рисунок 17 – Добавление расписания для секции

Интеграция клиентской и серверной частей

Взаимодействие клиентской и серверной частей осуществляется путем отправления HTTP-запросов с клиента на сервер. Для отправления запросов реализованы соответствующие сервисы, которые посредством модуля `HttpClient` отправляют запросы на сервер. Пример сервиса для получения видов спорта приведен в листинге 10.

Листинг 10 – Класс `SportService`

```
export class SportService {

  constructor(private http: HttpClient) { }
  url = `${serverUrl}/api/admin/sports`

  getSportList(): Observable<any> {
    return this.http.get(this.url);
  }
}
```

Данные с сервера возвращаются в формате JSON. Для отображения и дальнейшего использования данных реализованы интерфейсы, которые используются для десериализации данных. В листинге 11 приведен интерфейс для вида спорта.

Листинг 11 – Интерфейс `ISport`

```
export interface ISport {
    id: string,
    name: string
}
```

После получения ответа от сервера происходит десериализация данных, затем к ним можно обращаться в компоненте или в HTML-шаблоне. В листинге 12 приведен пример отображения видов спорта в селекторе `mat-select`.

Листинг 12 – отображение данных в HTML-шаблоне

```
<mat-select>
    <mat-option *ngFor="let sport of sports"[value]="sport.name"
    (click)="sportId = sport.id">
        {{sport.name}}
    </mat-option>
</mat-select>
```

В приведенном примере кода используется директива `*ngFor`, которая позволяет перебирать массивы данных в HTML-шаблоне. Использование указанной директивы может быть особенно полезно в случае, если необходимо перебрать большой массив данных, как, например, в случае отображения видов спорта.

4. ТЕСТИРОВАНИЕ

Функциональное тестирование

В таблице 1 приведены результаты функционального тестирования для основного набора тестов. Остальной набор тестов указан в таблице 1, приведенной в приложении.

Таблица 1 – Результаты тестирования

№	Название теста	Действие	Ожидаемый результат	Результат теста
1	Добавление секции	Тренер загружает изображение и указывает название и описание секции, а также выбирает вид спорта и зал из списка, после чего нажимает кнопку «Добавить»	В базе данных появляется соответствующая запись с данными о секции, тренер получает уведомление о том, что секция была создана	Пройден
2	Добавление расписания секции	Тренер выбирает секцию из списка, указывает день недели, время и период проведения, затем нажимает кнопку «Добавить»	В базе данных появляется соответствующая запись с занятием секции, тренер получает уведомление о том, что занятие было добавлено	Пройден
3	Добавление индивидуального занятия	Тренер выбирает вид спорта, указывает время проведения занятия и нажимает кнопку «Добавить»	Аналогично предыдущему пункту, в базу данных добавляется новая запись, а тренер получает уведомление о том, что занятие было добавлено	Пройден
4	Запись на индивидуальное занятие	Клиент, находясь на странице занятия, нажимает кнопку «Записаться»	В базе данных обновляется запись занятия, для пользователя отображается кнопка «Отменить запись»	Пройден
5	Запись на секцию	Клиент, находясь на странице секции, нажимает кнопку «Записаться»	В базе данных обновляется запись с данными секции, для пользователя отображается кнопка «Отменить запись»	Пройден

Юзабилити тестирование

Юзабилити тестирование [16] проводится с целью определения того, насколько удобным является использование разработанной системы. Для

проведения тестирования были составлены два набора тестов – для клиента и тренера.

Для клиента был определен следующий набор тестов:

1) изменить изображение профиля;
2) записаться на секцию;
3) посмотреть расписание занятий;
4) отменить запись на секцию.

Набор тестов для тренера:

1) добавить новую секцию;
2) добавить занятие секции;
3) добавить индивидуальное занятие;
4) удалить занятие секции.

В проведенном юзабилити тестировании приняли участие 5 человек. При выполнении тестирования затруднений не возникло, поставленные задачи были выполнены успешно.

Тестирование пользовательского интерфейса

Тестирование пользовательского интерфейса было выполнено в следующих браузера:

1) Opera GX;
2) Google Chrome;
3) Microsoft Edge;
4) Mozilla Firefox;
5) Яндекс Браузер.

Все элементы сайта в разных браузерах были отображены корректно. В ходе тестирования ошибок не выявлено. Разработанное веб-приложение успешно выполняет все поставленные задачи, а также является удобным для пользователя.

ЗАКЛЮЧЕНИЕ

Целью данной работы являлась разработка веб-приложения для записи на спортивные секции с использованием клиент-серверной архитектуры.

Ниже приведены задачи, которые были выполнены в ходе работы.

1. Проведен анализ предметной области.
2. Спроектирована и реализована серверная часть веб-приложения.
3. Спроектирована и реализована клиентская часть веб-приложения.
4. Проведено тестирование реализованного веб-приложения.

Ниже приведены задачи, которые могут быть поставлены в качестве дальнейшего развития работы.

1. Добавить чат.
2. Добавить возможность делать публикации на странице секции. Это могут быть новости, важные объявления или же фотографии.
3. Предусмотреть возможность добавления спортивных программ, тренировок. Например, добавить страницу с фотографиями или видеозаписями, на которых демонстрируется техника выполнения упражнений и др.

ЛИТЕРАТУРА

1. Asp.net. [Электронный ресурс] URL: https://dotnet.microsoft.com/en-us/apps/aspnet (дата обращения: 25.05.2024 г.).

2. Angular. [Электронный ресурс] URL: https://angular.io (дата обращения: 25.05.2024 г.).

3. Карта Спорта. [Электронный ресурс] URL: https://chelyabinsk.kartasporta.ru (дата обращения: 25.05.2024 г.).

4. После уроков. [Электронный ресурс] URL: https://posleurokov.ru/online (дата обращения: 25.05.2024 г.).

5. Шилопоп. [Электронный ресурс] URL: https://shilopop.ru (дата обращения: 25.05.2024 г.).

6. PostgreSQL. [Электронный ресурс] URL: www.postgresql.org (дата обращения 25.05.2024 г.).

7. FullCalendar. [Электронный ресурс] URL: https://fullcalendar.io (дата обращения: 25.05.2024 г.).

8. Руководство. Начало работы с Entity Framework 6 Code First с помощью MVC 5. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application (дата обращения: 25.05.2024 г.).

9. Центр документации Entity Framework. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/ef (дата обращения 25.05.2024 г.).

10. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. // СПб.: Питер, 2021. – 352 с.

11. Creating and Configuring a Model – EF Core. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/ef/core/modeling (дата обращения: 25.05.2024 г.).

12. Keycloak. [Электронный ресурс] URL: www.keycloak.org (дата обращения: 25.05.2024 г.).

13. Subramanian H. Hands-On RESTful API Design Patterns and Best Practices. / H. Subramanian, P. Raj // Packt Publishing, 2024. – 378 с.

14. Swiper Element (WebComponent). [Электронный ресурс] URL: https://swiperjs.com/element (дата обращения: 25.05.2024 г.).

15. Ngx-mask. [Электронный ресурс] URL: www.npmjs.com/package/ngx-mask (дата обращения: 25.05.2024 г.).

16. Карпович Е.Е. Методы тестирования и отладки программного обеспечения: учебник. // Москва: МИСИС, 2020. – 136 с.

ПРИЛОЖЕНИЕ. Функциональное тестирование

В таблице 1 приведены результаты функционального тестирования.

Таблица 1 – Результаты функционального тестирования

№	Название теста	Действие	Ожидаемый результат	Результат теста
1	Регистрация	Пользователь заполняет форму регистрации и нажимает кнопку «Зарегистрировать»	В базе данных сохраняются данные пользователя, пользователь получает уведомление о том, что аккаунт успешно создан	Пройден
2	Авторизация	Пользователь вводит электронную почту и пароль на странице входа и нажимает кнопку «Войти»	Пользователь получает доступ к функционалу приложения	Пройден
3	Отменить запись на секцию	Пользователь, находясь на странице секции, нажимает кнопку «Отменить запись»	В базе данных обновляется запись, соответствующая секции, пользователю отображается кнопка «Записаться», кнопка «Отменить запись» скрывается	Пройден
4	Отменить запись на индивидуальное занятие	Пользователь, находясь на странице занятия, нажимает кнопку «Отменить запись»	В базе данных обновляется запись, соответствующая индивидуальному занятию, пользователю отображается кнопка «Записаться», а кнопка «Отменить запись» скрывается	Пройден
5	Удалить занятие	Тренер нажимает на элемент календаря, который соответствует занятию, нажимает кнопку «Удалить»	В базе данных занятие помечается архивированным и больше не отображается для пользователей	Пройден
6	Удалить секцию	Тренер, находясь на странице секции, нажимает кнопку «Удалить»	В базе данных секция помечается архивированной и больше не отображается для пользователей	Пройден
7	Изменить данные профиля	Пользователь, находясь на странице профиля, изменяет данные, после чего нажимает кнопку «Сохранить»	В базе данных изменяются данные пользователя, после чего на странице отображаются обновленные данные	Пройден
8	Отменить изменение данных профиля	Пользователь, находясь на странице профиля, изменяет данные, после чего нажимает кнопку «Отменить»	Изменение данных не происходит и для пользователя отображаются исходные данные	Пройден