Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ УПРАВЛЕНИЯ БИЗНЕС-ПРОЦЕССАМИ

КАФЕДРА БИЗНЕС-ИНФОРМАТИКИ И МОДЕЛИРОВАНИЯ БИЗНЕС-ПРОЦЕССОВ

**КУРСОВАЯ РАБОТА**

Создание web-приложения с применением современных web-технологий по тематике «Фитнес-клуб»

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Никитин

подпись, дата

Студент УБ24-08Б, 432432361 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В. Шилова

подпись, дата

Красноярск 2025

**РЕФЕРАТ**

Курсовой проект: 42 страницы, 14 источников, 15 рисунков.

Цель – разработка полнофункционального веб-приложения для фитнес-клуба с применением современных веб-технологий, обеспечивающего клиентам удобный доступ к услугам и управление своими занятиями.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать сайт, включающий не менее пяти ключевых страниц: главная, расписание, запись на занятия, контакты, отзывы, а также систему личных кабинетов для пользователей и администраторов.

2. Применить комплекс современных веб-технологий, включая HTML5, CSS3 (с адаптивной версткой), чистый JavaScript для клиентской логики, серверный язык PHP и систему управления базами данных MySQL.

3. Спроектировать и реализовать реляционную базу данных для хранения информации о пользователях, тренерах, расписании занятий и записях.

4. Реализовать функционал взаимодействия с базой данных, включая регистрацию, авторизацию пользователей, управление профилем, онлайн-запись на тренировки и оставление отзывов.

5. Обеспечить интуитивно понятный и адаптивный пользовательский интерфейс, корректно отображающийся на различных устройствах.

Объект исследования – процесс проектирования и разработки клиент-серверного веб-приложения для сферы услуг.

Предмет исследования – применение стека технологий HTML, CSS, JavaScript, PHP и MySQL для создания функционального веб-ресурса фитнес-клуба.

Методы исследования: анализ предметной области, проектирование архитектуры приложения и структуры базы данных, метод функциональной разработки (front-end и back-end), тестирование.

Ключевые слова: веб-приложение, фитнес-клуб, HTML, CSS, JavaScript, PHP, MySQL, адаптивный дизайн, клиент-серверная архитектура, система бронирования.

Структура работы: работа состоит из введения, двух глав (теоретической и практической), заключения и списка использованных источников. Теоретическая часть раскрывает основы и принципы работы применяемых технологий. Практическая часть содержит описание процесса разработки, включая создание структуры страниц, стилизацию, реализацию серверной логики, взаимодействие с базой данных и итоговый пользовательский интерфейс.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc217501051)

[Теоретическая часть 7](#_Toc217501052)

[1.1 HTML и CSS 7](#_Toc217501053)

[1.2 PHP 9](#_Toc217501054)

[1.3 JavaScript 13](#_Toc217501055)

[1.4 MySQL 15](#_Toc217501056)

[Практическая часть 18](#_Toc217501057)

[2.1 HTML, CSS, PHP, JavaScript и MySQL на практике 18](#_Toc217501058)

[2.2 Функции и визуальная составляющая 34](#_Toc217501059)

[ЗАКЛЮЧЕНИЕ 44](#_Toc217501060)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45](#_Toc217501061)

# **ВВЕДЕНИЕ**

﻿ С развитием технологий и массовым переходом на интернет-услуги, web-приложения становятся неотъемлемой частью бизнеса, включая такие отрасли, как фитнес-индустрия. В условиях современного мира удобство и доступность информации, а также возможность управлять услугами онлайн, становятся важнейшими конкурентными преимуществами для фитнес-клубов. Разработка web-приложений для фитнес-клубов предоставляет пользователям возможность не только найти необходимую информацию о клубе, но и взаимодействовать с ним напрямую через интернет-платформу.

Актуальность темы заключается в том, что в цифровую эпоху веб-приложения превратились в ключевой инструмент для бизнеса, включая фитнес-индустрию. Удобство, доступность услуг и онлайн-взаимодействие стали важными конкурентными преимуществами. Разработка такого приложения для фитнес-клуба позволяет клиентам легко получать информацию и управлять своими занятиями, а самому клубу — оптимизировать рабочие процессы, улучшить клиентский опыт и повысить эффективность.

Целью курсовой работы является создание web-приложения для фитнес-клуба с применением современных web-технологий. В рамках работы будет разработан сайт, который включает функциональность для просмотра информации о клубе, тренерах, расписании занятий, а также для записи на тренировки и управления личными данными пользователей.

Задачи, которые решаются в рамках курсовой работы:

– Разработка сайта с минимум пятью страницами, включая страницы информации о клубе, расписания, записи на занятия и личного кабинета.

– Применение современных web-технологий, таких как чистый JavaScript, библиотеки и фреймворки.

– Создание базы данных для хранения информации о пользователях, тренерах, занятиях и расписании.

– Реализация функционала взаимодействия с базой данных, включая регистрацию и авторизацию пользователей, возможность редактирования профиля и записи на занятия.

**Теоретическая часть**

## 1.1 HTML и CSS

HTML (HyperText Markup Language) и CSS (Cascading Style Sheets) являются основными технологиями для создания и оформления веб-страниц. Эти два языка тесно связаны и взаимодополняют друг друга, обеспечивая разработку удобных и визуально привлекательных веб-приложений.

HTML является языком разметки, предназначенным для создания структуры веб-страниц. Он предоставляет набор тегов, с помощью которых можно определить различные элементы контента, такие как текст, изображения, таблицы, формы, ссылки и другие. Основной задачей HTML является создание семантической структуры страницы, которая помогает браузеру правильно интерпретировать и отображать контент для пользователя.

HTML состоит из элементов, которые могут быть вложены друг в друга. Например, для отображения текста на странице используются теги <p>, <h1>, <h2>, <h3>, а для создания ссылок – тег <a>. Для изображения контента применяется тег <img>, а для представления таблиц данных используется элемент <table>. В процессе создания веб-приложений, таких как сайт фитнес-клуба, HTML помогает структурировать все данные и сделать интерфейс понятным и доступным для пользователей. Элементы HTML позволяют формировать страницы с необходимыми разделами, такими как заголовки, параграфы, списки и формы для ввода данных.

С помощью HTML также реализуются различные элементы для взаимодействия с пользователем. Это может быть форма для ввода данных, такие как регистрация или авторизация, где используются элементы <form>, <input>, <textarea>, <select> и <button>. Эти элементы позволяют собирать информацию от пользователя, отправлять её на сервер для дальнейшей обработки и отображать результаты.

CSS, в свою очередь, служит для оформления внешнего вида элементов, определённых с помощью HTML. С помощью CSS разработчики могут задавать цвета, шрифты, отступы, выравнивание и другие параметры, которые влияют на внешний вид страницы. CSS значительно улучшает восприятие веб-приложений, предоставляя возможность создать уникальный дизайн и сделать интерфейс более удобным и привлекательным для пользователей.

Одной из важных особенностей CSS является возможность разделять стиль и структуру контента. С помощью стилей можно изменять внешний вид элементов на всех страницах сайта без необходимости изменения самого HTML-кода. Это позволяет добиться гибкости и удобства при изменении дизайна всего сайта, не затрагивая структуру страницы.

CSS предлагает различные способы стилизации, включая использование классов и идентификаторов, а также гибкую верстку с использованием моделей блоков, флоатов и Flexbox. Для достижения адаптивности и обеспечения корректного отображения сайта на различных устройствах, используются медиа-запросы. Медиа-запросы позволяют изменять стили в зависимости от характеристик устройства, таких как разрешение экрана, ширина окна браузера и другие параметры. Это особенно важно в современном веб-дизайне, когда сайты должны быть удобными для пользователей как на десктопах, так и на мобильных устройствах.

Кроме того, CSS помогает решать задачи адаптивности, позволяя странице выглядеть одинаково хорошо как на мобильных телефонах, так и на больших экранах компьютеров. Это достигается благодаря правильному использованию гибких макетов и медиа-запросов. В рамках создания сайта для фитнес-клуба важным аспектом является, например, использование гибкой сетки для отображения расписания, которое должно корректно отображаться на различных устройствах.

В условиях современного интернета, когда пользователи просматривают веб-страницы на различных устройствах (ПК, смартфоны, планшеты), адаптивный дизайн становится важным аспектом разработки. Адаптивный дизайн позволяет веб-странице изменять свою структуру и внешний вид в зависимости от размеров экрана устройства.

CSS предоставляет возможность использовать медиазапросы (@media), которые позволяют изменять стили в зависимости от ширины экрана, разрешения, ориентации устройства и других параметров. Например, можно задать одно оформление для больших экранов и совершенно другое для мобильных устройств, чтобы пользователь на любом устройстве мог удобно просматривать контент. Медиазапросы играют важную роль в обеспечении доступности и удобства сайта на различных устройствах, что особенно актуально для мобильных пользователей, которые составляют большую часть аудитории современных сайтов.

Дизайн веб-страницы оказывает огромное влияние на восприятие сайта пользователем. Сайт с красивым и современным дизайном вызывает доверие и повышает вероятность того, что посетители останутся на странице, а также вернутся в будущем. Важно, чтобы сайт был не только функциональным, но и эстетически привлекательным.

CSS помогает решить эту задачу, предоставляя средства для работы с типографикой, цветами, анимациями и другими визуальными элементами. Использование современных шрифтов (через свойства font-family) и палитры цветов, соответствующей тематике сайта, позволяет сделать страницу более приятной для восприятия. Применение анимаций и переходов помогает добавить динамики и интерактивности, улучшая взаимодействие с пользователем.

В результате, использование HTML и CSS в разработке веб-приложения позволяет создать структуру, обеспечивающую удобное взаимодействие с пользователем, а также стильный и функциональный дизайн, который способствует позитивному восприятию сайта и повышению его удобства для посетителей.

## 1.2 PHP

PHP (Hypertext Preprocessor) — это широко используемый серверный язык программирования, который является основным инструментом для создания динамических веб-страниц и веб-приложений. Он был разработан для работы с веб-серверами и позволяет выполнять различные серверные операции, такие как обработка данных, взаимодействие с базами данных, создание и отправка форм, а также управление сессиями и пользователями. PHP активно используется для создания сайтов и веб-приложений благодаря своей простоте, гибкости и мощным возможностям интеграции с различными веб-технологиями.

PHP — это интерпретируемый язык, который выполняется на сервере. Когда пользователь запрашивает страницу, сервер обрабатывает код PHP, генерирует HTML-код, который отправляется клиенту, и отображает его в браузере. Это делает PHP мощным инструментом для создания динамического контента на веб-страницах.

Некоторые ключевые особенности PHP:

1. Интеграция с HTML: PHP легко встраивается в HTML-код, что позволяет легко добавлять серверную логику в страницы без необходимости создавать отдельные файлы для обработки данных. В коде PHP используются теги <?php и ?>, чтобы обрамить программные инструкции внутри HTML-документа.

2. Работа с базами данных: PHP поддерживает интеграцию с большинством популярных СУБД, включая MySQL, PostgreSQL, SQLite и другие. Это позволяет разработчикам легко выполнять запросы к базе данных, извлекать, обновлять и удалять данные.

3. Работа с формами и обработка данных: PHP предоставляет мощные средства для обработки данных, отправленных через формы. Это включает в себя безопасную обработку входных данных, таких как текст, числа и файлы, а также возможность выполнения различных операций, таких как проверка данных на сервере и отправка данных в базу данных.

4. Поддержка сессий и cookies: PHP позволяет удобно работать с сессиями, которые хранят информацию о пользователе между запросами, а также с cookies, которые позволяют хранить данные на клиенте (например, предпочтения пользователя или идентификатор сессии).

5. Обработка ошибок: PHP предоставляет механизмы для обработки ошибок, что помогает разработчикам эффективно отлавливать и устранять проблемы в коде. Это включает в себя как стандартные сообщения об ошибках, так и более сложные механизмы логирования и обработки исключений.

6. Открытый исходный код: PHP является открытым проектом, что означает, что он доступен для использования и модификации без ограничений. Это способствует активному сообществу разработчиков, которое постоянно улучшает язык, создавая новые библиотеки и инструменты.

Основные конструкции PHP:

– Переменные. В PHP переменные начинаются с символа $, за которым следует имя переменной. Переменные могут хранить различные типы данных, такие как строки, числа, массивы и объекты.

– Условные операторы. PHP поддерживает стандартные условные операторы, такие как if, else и elseif, которые используются для выполнения определенного кода в зависимости от выполнения условия. Также существует оператор switch, который используется для выбора одного из нескольких блоков кода на основе значения переменной.

– Циклы. Для выполнения повторяющихся операций PHP предоставляет конструкции циклов: for, while, do-while и foreach. Они позволяют обходить массивы, повторять действия определенное количество раз или до выполнения условия.

– Функции. В PHP можно создавать функции, которые позволяют группировать повторяющиеся действия и использовать их в разных частях кода. Функции могут принимать аргументы и возвращать значения.

– Массивы. Массивы в PHP используются для хранения коллекции значений. PHP поддерживает как индексированные массивы, так и ассоциативные массивы, где каждый элемент ассоциирован с ключом.

Одной из ключевых возможностей PHP является интеграция с базами данных. PHP поддерживает работу с такими базами данных, как MySQL, PostgreSQL, SQLite и другими, используя различные библиотеки и расширения, такие как mysqli, PDO (PHP Data Objects) и pg\_connect. Это позволяет выполнять операции с базами данных, такие как создание, чтение, обновление и удаление данных (CRUD-операции).

Это пример выполнения SQL-запроса для получения данных о пользователях из базы данных. PHP обрабатывает запрос, выполняет его на сервере и выводит результат.

Безопасность — это важный аспект разработки веб-приложений на PHP. Некоторые основные меры, которые нужно учитывать при работе с PHP:

– Защита от SQL-инъекций: для предотвращения SQL-инъекций следует использовать подготовленные выражения (prepared statements) и параметризованные запросы, как это показано в примере выше с использованием mysqli.

– Шифрование данных: Для безопасного хранения паролей и других конфиденциальных данных следует использовать алгоритмы хеширования (например, password\_hash() и password\_verify()), а не хранить данные в открытом виде.

– Работа с сессиями и cookies: PHP предоставляет механизмы для работы с сессиями и cookies. Важно помнить о безопасности при работе с сессиями, чтобы предотвратить их кражу или подделку.

– Защита от XSS и CSRF атак: следует обязательно проверять и фильтровать входные данные, чтобы избежать атак на клиентской стороне (XSS) и подделки запросов (CSRF).

PHP является основным языком для создания динамических сайтов и веб-приложений. Он используется для создания серверной логики, обработки пользовательских запросов, взаимодействия с базой данных и формирования HTML-страниц для отображения в браузере. В рамках разработки веб-приложений для фитнес-клуба PHP используется для обработки данных регистрации и авторизации пользователей, взаимодействия с базой данных для хранения информации о пользователях, тренерах, расписании и занятиях, а также для управления сессиями пользователей.

Вместе с HTML, CSS и JavaScript, PHP составляет основу современного веб-разработки, обеспечивая создание динамичных, функциональных и безопасных веб-приложений.

1.3 JavaScript

JavaScript — это высокоуровневый интерпретируемый язык программирования, используемый для создания динамичных и интерактивных элементов на веб-страницах. Являясь неотъемлемой частью технологий веб-разработки, вместе с HTML и CSS, JavaScript выполняется на стороне клиента, что позволяет эффективно взаимодействовать с пользователем и обновлять содержимое веб-страниц в реальном времени без необходимости их перезагрузки. Он предоставляет широкий спектр возможностей для создания интерактивных функций, таких как формы, анимации, обработка данных в реальном времени и асинхронные запросы к серверу.

JavaScript взаимодействует с DOM (Document Object Model), который представляет собой структуру веб-страницы в виде дерева объектов. Это позволяет динамически изменять HTML-структуру и стили элементов на странице, добавлять новые элементы, изменять текст и атрибуты. Такие возможности делают веб-страницы более интерактивными, что в свою очередь улучшает пользовательский опыт, так как страницы могут обновляться в ответ на действия пользователя, такие как клики или ввод данных. Это особенно важно для создания современного веб-приложения, где взаимодействие с пользователем должно быть гибким и быстрым.

С помощью JavaScript можно реализовать асинхронное программирование, что позволяет запускать длительные операции, например запросы к серверу, без блокировки интерфейса. Это стало возможным благодаря использованию таких конструкций, как колбеки, промисы и асинхронные функции, что позволяет значительно улучшить производительность веб-приложений. Например, технология AJAX позволяет отправлять запросы к серверу и получать данные без необходимости перезагрузки страницы, что делает процесс взаимодействия с приложением более плавным и быстрым.

Также JavaScript активно используется для обработки событий, таких как клики, ввод текста, изменение значений и другие действия пользователя. С помощью обработчиков событий можно назначать определенные функции, которые будут выполняться при наступлении того или иного события. Это дает возможность контролировать логику работы веб-приложения в ответ на действия пользователя, например, изменять видимость элементов на странице, показывать всплывающие окна или выполнять валидацию данных в формах.

Язык JavaScript позволяет использовать объектно-ориентированное программирование, что дает возможность эффективно организовывать код в виде объектов, состоящих из свойств и методов. Это позволяет создавать более структурированный и модульный код, что делает его более удобным для дальнейшей разработки и поддержки. В последние годы также стало популярным использование стрелочных функций, которые позволяют сократить запись кода, улучшая его читаемость и упрощая работу с функциями.

JavaScript также поддерживает работу с массивами и объектами, которые являются основными типами данных в языке. Массивы позволяют хранить коллекции данных, такие как список элементов, а объекты — группы данных в виде пар ключ-значение. Эти структуры данных широко используются для организации информации, например, для представления пользователей, продуктов или записей в базе данных.

Одной из важных возможностей JavaScript является его способность работать с асинхронными запросами, что позволяет обмениваться данными с сервером без перезагрузки страницы. Это критически важно для создания современных веб-приложений, где данные могут меняться в реальном времени, и важно обеспечить плавность и быстроту взаимодействия с пользователем. JavaScript помогает организовывать многозадачность в приложении, позволяя выполнять операции без блокировки интерфейса, что значительно повышает удобство и производительность.

Таким образом, JavaScript является неотъемлемой частью современных веб-технологий, и его использование позволяет создавать динамичные, интерактивные и высокопроизводительные веб-приложения. В рамках разработки веб-приложений для фитнес-клуба JavaScript активно используется для создания функциональных и удобных интерфейсов, управления данными, работы с пользовательскими событиями и взаимодействия с сервером.

1.4 MySQL

MySQL — это система управления реляционными базами данных (СУБД), которая используется для хранения, управления и обработки данных в структурированном виде. MySQL является одной из самых популярных СУБД в мире, благодаря своей простоте, эффективности и хорошей производительности. Она является открытым программным обеспечением и используется во множестве веб-приложений и сервисов.

Основной принцип работы MySQL заключается в использовании таблиц для хранения данных. Каждая таблица состоит из строк и столбцов, где строки представляют собой записи (или данные), а столбцы — поля, которые определяют типы данных, такие как текст, числа, даты и другие. Такие таблицы могут быть связаны друг с другом с помощью ключей, что позволяет эффективно организовывать данные и выполнять сложные запросы.

В контексте веб-разработки MySQL используется для хранения информации о пользователях, товарах, заказах, отзывах, записях и других данных, которые необходимы для функционирования веб-приложения. В веб-приложении для фитнес-клуба MySQL играет важную роль в организации и хранении данных пользователей, тренеров, расписания занятий и других элементов, с которыми взаимодействуют посетители сайта.

Для работы с MySQL используется язык SQL (Structured Query Language), который позволяет выполнять запросы к базе данных. Основные операции, которые можно выполнять с помощью SQL, включают добавление, удаление, обновление и извлечение данных из таблиц. Запросы могут быть простыми, например, извлечение всех записей из таблицы, или сложными, когда требуется выполнение объединений таблиц, фильтрация данных или агрегация.

Одним из важных аспектов работы с MySQL является создание схемы базы данных, которая включает проектирование таблиц, их взаимосвязей и ограничений. В MySQL данные могут быть связаны с помощью внешних ключей, которые обеспечивают целостность данных и позволяют избежать ошибок, таких как добавление записей с несоответствующими значениями в связанных таблицах. Например, в веб-приложении для фитнес-клуба таблица пользователей может быть связана с таблицей записей на занятия через внешний ключ, что позволяет отследить, какие занятия были забронированы конкретными пользователями.

В MySQL также важно использовать индексы для ускорения поиска данных. Индексы создаются на основе выбранных столбцов таблиц и значительно ускоряют выполнение запросов, которые используют эти столбцы для фильтрации данных. Например, для поиска пользователей по имени или для сортировки расписания занятий по времени можно создать индекс, что существенно повысит производительность работы с базой данных.

MySQL поддерживает транзакции, что позволяет выполнять несколько операций с базой данных в рамках одной логической единицы. Это важно для обеспечения целостности данных, например, при регистрации нового пользователя или обновлении его информации. Транзакции позволяют откатить все изменения в случае ошибки, обеспечивая консистентность базы данных.

Кроме того, MySQL имеет встроенные средства для управления правами доступа к базе данных. Это позволяет ограничить доступ к данным и операциям, что особенно важно в многопользовательских системах. В контексте веб-приложения для фитнес-клуба можно настроить права доступа для различных ролей пользователей, например, для обычных пользователей, администраторов и тренеров, чтобы каждый мог работать только с теми данными, которые ему доступны.

Для взаимодействия с MySQL в веб-разработке часто используется серверный язык программирования, такой как PHP, который позволяет отправлять запросы в базу данных и получать результаты. В PHP используется расширение MySQLi или PDO для работы с MySQL. В процессе работы с базой данных важно учитывать безопасность, например, защищать запросы от SQL-инъекций, используя подготовленные выражения и параметризированные запросы.

В разработке веб-приложения для фитнес-клуба MySQL играет ключевую роль в организации данных о пользователях, тренерах, занятиях и отзывах, позволяя эффективно управлять и обрабатывать информацию.

**Практическая часть**

2.1 HTML, CSS, PHP, JavaScript и MySQL на практике

Практическая часть по HTML в рамках разработки веб-приложения для фитнес-клуба охватывает создание структуры веб-страниц с использованием различных элементов и атрибутов. HTML (HyperText Markup Language) является основой для всех веб-страниц, и в данном случае он используется для создания контента и разметки страниц фитнес-клуба.

Каждая веб-страница начинается с декларации типа документа <!DOCTYPE html>, что указывает браузеру, что используется стандарт HTML5. Далее идет сам HTML-документ, который состоит из двух основных частей: <head> и <body>.

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport"

content="width=device-width, initial-scale=1.0">

<title>Контакты</title>

<link rel="stylesheet" href="css/styles.css">

</head>

<body>

</body>

</html>

<meta charset="UTF-8"> — определяет кодировку страницы. <meta name="viewport" content="width=device-width, initial-scale=1.0"> делает страницу адаптивной для мобильных устройств. <link rel="stylesheet" href="css/styles.css"> подключает внешний файл стилей CSS.

Одной из задач было создание навигационного меню для сайта. Для этого использовался элемент <nav>, который включает список ссылок <ul>, а сами элементы списка обрабатываются с помощью тега <li>. Каждая ссылка указана в теге <a>. Это меню позволяет пользователям переходить между различными страницами веб-приложения, такими как главная страница, расписание, запись на занятия, отзывы и т. д. Код навигационного меню:

<nav>

<ul>

<li><a href="index.php">Главная</a></li>

<li><a href="schedule.php">Расписание</a></li>

<li><a href="booking.php">Запись</a></li>

<li><a href="reviews.php">Отзывы</a></li>

<li><a href="contacts.php">Контакты</a></li>

</ul>

</nav>

Здесь каждый <a> элемент содержит атрибут href, который указывает на URL соответствующей страницы. Ссылки обеспечивают навигацию по сайту.

Веб-приложение включает функционал для авторизации, регистрации и изменения данных пользователей, что требует использования форм. Для сбора данных от пользователя используются HTML-формы. В формах используются различные элементы, такие как текстовые поля <input>, кнопки <button>, а также метки <label> для удобства. Форма для изменения данных пользователя:

<form action="dashboard.php" method="POST">

<label for="username">Новый логин:</label>

<input type="text" name="username" value="<?php echo htmlspecialchars($user['username']); ?>" required><br><br>

<label for="phone">Новый номер телефона:</label>

<input type="text" name="phone" value="<?php echo htmlspecialchars($user['phone']); ?>" required><br><br>

<label for="password">Новый пароль (оставьте пустым, если не хотите менять):</label>

<input type="password" name="password"><br><br>

<button type="submit" name="update\_user">Обновить</button>

</form>

<input type="text"> используется для ввода текстовых данных, таких как логин и телефон.

<input type="password"> используется для ввода пароля, который скрывается.

Атрибут required указывает на обязательность заполнения полей. Атрибут value в поле ввода заполняется данными пользователя из базы данных, чтобы он мог их редактировать.

Для добавления изображений на страницу используется элемент <img>, который указывает путь к изображению через атрибут src. В данном случае, на странице «Контакты» добавлена фотография местоположения клуба.

<div class="club-photo">

<img src="images/fitness-club.jpg" alt="Фитнес-клуб" style="width: 100%; max-height: 400px; object-fit: cover; margin-top: 20px;">

</div>

src="images/fitness-club.jpg" указывает путь к изображению. alt="Фитнес-клуб" альтернативный текст, который будет отображаться, если изображение не загрузилось. Стиль object-fit: cover используется для сохранения пропорций изображения, с обрезанием лишней части, чтобы оно вписалось в контейнер.

Для интеграции внешних карт и других сервисов на сайт используются ссылки. Например, для отображения карты Google Maps на странице «Контакты» добавлены ссылки на Google Maps и Яндекс.Карты.

<ul>

<li><a href="https://www.google.com/maps/place/Красноярск,+ул.+9 +мая,+д.+77" target="\_blank">Google Maps</a></li>

<li><a href="https://yandex.ru/maps/62/krasnoyarsk/?ll=92.907370%2C56.049625&z=15" target="\_blank">Яндекс.Карты</a></li>

</ul>

<a href="URL" target="\_blank"> создает ссылку, которая откроется в новой вкладке браузера. target="\_blank" указывает, что ссылка будет открываться в новой вкладке.

Каждая страница приложения имеет четкую структуру с использованием семантических элементов, таких как:

<header> — для заголовочной части страницы, например, для навигации.

<section> — для разделов с контентом, например, с информацией о клубе, расписанием или отзывами.

<div> — для группировки контента и стилизации, например, для контейнеров с изображениями или текстами. Пример использования <header> и <section>:

<header>

<nav>

<ul>

<li><a href="index.php">Главная</a></li>

<li><a href="schedule.php">Расписание</a></li>

<li><a href="booking.php">Запись</a></li>

</ul>

</nav>

</header>

<section>

<h2>Контакты</h2>

<p>Адрес: г. Красноярск, улица 9 Мая, 77</p>

<p>Телефон: 8 800 555 35-35</p>

<p>Email: sparta@fitnessclub.ru</p>

</section>

<header> включает меню навигации. <section> используется для представления блока с контактной информацией.

В рамках разработки веб-приложения использовался принцип адаптивного дизайна, который позволяет сайту корректно отображаться на различных устройствах — от мобильных телефонов до настольных ПК. Это достигается с помощью использования медиазапросов в CSS и правильной структуры HTML. Например, использование мета-тега для установки ширины страницы в соответствии с шириной экрана устройства: <meta name="viewport" content="width=device-width, initial-scale=1.0">

Этот тег помогает сайтам правильно масштабироваться на мобильных устройствах, адаптируя размер контента под ширину экрана.

Для стилизации страниц использовался внешний файл стилей. Это позволяет отделить структуру HTML от оформления, упрощая управление кодом.

Подключение CSS в документ HTML выполнялось с помощью тега <link>:

<link rel="stylesheet" href="css/styles.css"> Это позволяет использовать файл styles.css для применения стилей ко всем страницам приложения.

Были заданы глобальные стили для базовых HTML-элементов, таких как body, заголовки (h1, h2 и т. д.), параграфы (p) и ссылки (a). Например:

body {

font-family: 'Arial', sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f4;

color: #333;

}

h1, h2, h3 {

color: #2a2a2a;

margin-bottom: 10px;

}

a {

text-decoration: none;

color: #007BFF;

}

a:hover {

color: #0056b3;

}

font-family задает семейство шрифтов для текста. background-color устанавливает светлый фон для всей страницы. color определяет цвет текста. text-decoration: none убирает подчеркивание у ссылок. hover задает стили при наведении курсора на элемент.

Меню навигации (<nav>) оформлено для удобства пользователей и эстетики. Использовались списки (<ul> и <li>) с горизонтальным расположением:

nav {

background-color: #333;

padding: 10px 20px;

}

nav ul {

display: flex;

list-style: none;

margin: 0;

padding: 0;

}

nav ul li {

margin-right: 20px;

}

nav ul li a {

color: #fff;

padding: 5px 10px;

border-radius: 5px;

transition: background-color 0.3s ease;

}

nav ul li a:hover {

background-color: #007BFF;

}

display: flex выравнивает элементы списка горизонтально. list-style: none убирает маркеры списка. border-radius скругляет углы ссылок. transition добавляет анимацию при наведении на ссылки.

Для страницы с информацией о тренерах применялись стили для карточек (<div>), которые включали изображение и описание. Карточки отображались в виде сетки:

.card {

border: 1px solid #ddd;

border-radius: 10px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

padding: 20px;

text-align: center;

background-color: #fff;

}

.card img {

width: 100%;

max-width: 150px;

border-radius: 50%;

margin-bottom: 10px;

}

.card h3 {

margin: 10px 0;

color: #333;

}

.card p {

font-size: 0.9em;

color: #555;

}

box-shadow добавляет тень, создавая эффект объема. border-radius: 50% делает изображение круглым. max-width ограничивает размер изображения.

Формы для регистрации, авторизации и записи на занятия оформлялись для удобства пользователя. Все элементы были выровнены, а кнопки выделены цветом. Один из стилей форм:

form {

max-width: 400px;

margin: 20px auto;

padding: 20px;

background-color: #fff;

border: 1px solid #ddd;

border-radius: 10px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

}

form label {

display: block;

margin-bottom: 5px;

font-weight: bold;

color: #333;

}

form input {

width: 100%;

padding: 10px;

margin-bottom: 15px;

border: 1px solid #ddd;

border-radius: 5px;

}

form button {

background-color: #007BFF;

color: #fff;

padding: 10px 15px;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s ease;

}

form button:hover {

background-color: #0056b3;

}

max-width: 400px задает ширину формы. margin: auto центрирует форму на странице. box-shadow создает эффект поднятой карточки. transition добавляет плавность изменения цвета кнопки.

Адаптивный дизайн был реализован с помощью медиазапросов. Например, для мобильных устройств применялись следующие стили:

@media (max-width: 768px) {

nav ul {

flex-direction: column;

}

nav ul li {

margin: 10px 0;

}

.card {

margin-bottom: 20px;

}

}

@media (max-width: 768px) задает стили для экранов шириной менее 768px. flex-direction: column делает элементы списка вертикальными. margin-bottom увеличивает расстояние между карточками.

В проекте фитнес-клуба PHP выполняет ключевые функции: обработку форм, управление пользователями, взаимодействие с базой данных и реализацию логики авторизации и регистрации. Основные функции php в проекте:

1. Подключение к базе данных

Одной из важнейших задач PHP является подключение к базе данных MySQL. Для этого использовалась функция mysqli\_connect. Пример кода подключения к базе данных (db.php):

<?php

$host = 'localhost';

$db = 'fitness-club';

$user = 'root';

$pass = '';

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect\_error) {

    die("Ошибка подключения: " . $conn->connect\_error);

}

?>

2. Регистрация пользователей

Для обработки формы регистрации используется скрипт, который получает данные от пользователя, проверяет их и записывает в базу данных. Вот его код (register.php):

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['register'])) {

    $username = $\_POST['username'];

    $password = $\_POST['password'];

    $phone = $\_POST['phone'];

    $sql = "SELECT \* FROM users WHERE username = ?";

    $stmt = $conn->prepare($sql);

    $stmt->bind\_param("s", $username);

    $stmt->execute();

    $result = $stmt->get\_result();

    if ($result->num\_rows > 0) {

        echo "Пользователь с таким логином уже существует!";

    } else {

        $sql = "INSERT INTO users (username, password, phone) VALUES (?, ?, ?)";

        $stmt = $conn->prepare($sql);

        $stmt->bind\_param("sss", $username, $password, $phone);

        if ($stmt->execute()) {

            $\_SESSION['registration\_success'] = true;

            header("Location: register.php");

            exit();

        } else {

            echo "Ошибка: " . $stmt->error;

        }

    }

}

3. Авторизация пользователей

PHP обрабатывает данные из формы авторизации и проверяет их с базой данных. При успешном входе создается сессия пользователя. Пример скрипта (register.php):

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['login'])) {

    $username = $\_POST['username'];

    $password = $\_POST['password'];

    $sql = "SELECT \* FROM users WHERE username = ?";

    $stmt = $conn->prepare($sql);

    $stmt->bind\_param("s", $username);

    $stmt->execute();

    $result = $stmt->get\_result();

    if ($result->num\_rows > 0) {

        $user = $result->fetch\_assoc();

        if ($user['password'] == $password) {

            $\_SESSION['user\_id'] = $user['id'];

            $\_SESSION['username'] = $user['username'];

            $\_SESSION['is\_admin'] = $user['is\_admin'];

            header("Location: dashboard.php");

            exit();

        } else {

            echo "Неверный пароль!";

        }

    } else {

        echo "Пользователь не найден!";

    }

}

4. Запись на занятия

В booking.php есть код, который позволяет пользователю выбрать тренера, дату и время занятия, а затем сохранить данные в базе.

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    list($trainer\_name, $session\_date, $session\_time) = explode("|", $\_POST['session']);

    $sql = "INSERT INTO bookings (user\_id, training\_date, training\_time, trainer\_name) VALUES (?, ?, ?, ?)";

    $stmt = $conn->prepare($sql);

    $stmt->bind\_param("isss", $user\_id, $session\_date, $session\_time, $trainer\_name);

    if ($stmt->execute()) {

        $success\_message = "Вы успешно записались на занятие!";

    } else {

        $error\_message = "Ошибка при записи.";

    }

    $stmt->close();

}

Таблица reviews хранит отзывы, связанные с пользователями. Проверяется, что отзыв оставлен авторизованным пользователем.

5. Административная панель

Для администратора реализованы функции управления пользователями и записями. Пример кода для удаления пользователя (admin.php):

if ($user\_id\_to\_delete) {

    $sql\_delete\_bookings = "DELETE FROM bookings WHERE user\_id = ?";

    $stmt\_delete\_bookings = $conn->prepare($sql\_delete\_bookings);

    $stmt\_delete\_bookings->bind\_param("i", $user\_id\_to\_delete);

    $stmt\_delete\_bookings->execute();

    $sql = "DELETE FROM users WHERE id = ?";

    $stmt = $conn->prepare($sql);

    $stmt->bind\_param("i", $user\_id\_to\_delete);

    $stmt->execute();

    if ($stmt->affected\_rows > 0) {

        $success\_message = "Пользователь и его записи успешно удалены!";

    } else {

        $error\_message = "Ошибка при удалении пользователя.";

    }

    $stmt\_delete\_bookings->close();

}

6. Вывод данных из базы

PHP позволяет динамически отображать данные, например, список записей пользователя, код для этой функции:

<h3>Ваши записи на занятия:</h3>

        <?php if (!empty($bookings)): ?>

            <table>

                <thead>

                    <tr>

                        <th>Дата занятия</th>

                        <th>Время занятия</th>

                        <th>Тренер</th>

                    </tr>

                </thead>

                <tbody>

                    <?php foreach ($bookings as $booking): ?>

                        <tr>

                            <td><?php echo htmlspecialchars($booking['training\_date']); ?></td>

                            <td><?php echo htmlspecialchars($booking['training\_time']); ?></td>

                            <td><?php echo htmlspecialchars($booking['trainer\_name']); ?></td>

                        </tr>

                    <?php endforeach; ?>

                </tbody>

            </table>

        <?php else: ?>

            <p>Вы не записаны на занятия.</p>

        <?php endif; ?>

    </div>

Данные отображаются в виде HTML-контента.

В этой части объясняются ключевые реализации для веб-приложения фитнес-клуба, включая управление правами администратора, отображение расписания, управление пользователями и бронированиями. JavaScript использовался для реализации функций приложения, таких как:

1. Проверка прав администратора. Код:

function checkAdminRights() {

const isAdmin = localStorage.getItem("isAdmin") === "true";

if (isAdmin) {

document.getElementById("admin-info").style.display = "block";

document.getElementById("admin-error").style.display = "none";

} else {

document.getElementById("admin-info").style.display = "none";

document.getElementById("admin-error").style.display = "block";

}

}

Эта функция проверяет, есть ли у пользователя права администратора, обращаясь к локальному хранилищу localStorage. Она определяет доступность административных элементов интерфейса. Если пользователь не администратор, скрывает блок с админ-информацией.

2. Загрузка данных пользователей. Код:

function loadUserData() {

const users = JSON.parse(localStorage.getItem("users")) || [];

const usersTable = document.getElementById("users-table").getElementsByTagName("tbody")[0];

usersTable.innerHTML = "";

users.forEach((user, index) => {

const row = usersTable.insertRow();

row.insertCell(0).textContent = user.username;

row.insertCell(1).textContent = user.phone;

const deleteCell = row.insertCell(2);

const deleteButton = document.createElement("button");

deleteButton.textContent = "Удалить";

deleteButton.onclick = () => deleteUser(index);

deleteCell.appendChild(deleteButton);

});

}

Эта функция извлекает список пользователей из localStorage, очищает содержимое таблицы, заполняет таблицу строками с данными пользователей, добавляет кнопку "Удалить" для удаления пользователя.

Позволяет динамически отображать данные пользователей в административной панели.

3. Удаление пользователя. Код:

function deleteUser(index) {

let users = JSON.parse(localStorage.getItem("users")) || [];

users.splice(index, 1);

localStorage.setItem("users", JSON.stringify(users));

loadUserData();

}

Удаляет пользователя из списка по индексу. Позволяет администратору управлять данными пользователей, удаляя их из системы. Данные обновляются моментально благодаря перезагрузке таблицы.

4. Проверка авторизации. Код:

function checkAuthorization() {

const isLoggedIn = localStorage.getItem("userLoggedIn") === "true";

const userName = localStorage.getItem("userName");

if (isLoggedIn) {

document.getElementById("login-register").style.display = "none";

document.getElementById("user-info").style.display = "block";

document.getElementById("user-name").textContent = userName;

} else {

document.getElementById("login-register").style.display = "block";

document.getElementById("user-info").style.display = "none";

}

}

Функция проверяет, авторизован ли пользователь, и отображает соответствующий интерфейс. Если пользователь вошёл в систему, показываются его данные. Если нет — предлагается авторизация.

5. Отображение расписания. Код:

function displaySchedule(schedule, container) {

schedule.forEach(session => {

const sessionElement = document.createElement("div");

sessionElement.classList.add("session");

sessionElement.innerHTML = `

<p><strong>${session.date}</strong> | ${session.time} - Тренер: ${session.trainer}</p>

<button class="book-session">Записаться</button>

`;

container.appendChild(sessionElement);

});

}

Эта функция создаёт элементы расписания на основе данных, добавляет кнопки "Записаться". Простая возможность добавления динамических событий.

6. Отображение подробной информации о тренере. Код:

function showTrainerInfo(trainer) {

const trainerInfo = document.getElementById("trainer-info");

trainerInfo.innerHTML = `

<img src="${trainer.photo}" alt="${trainer.name}" style="width: 200px;">

<p><strong>Квалификация:</strong> ${trainer.qualification}</p>

`;

trainerInfo.style.display = 'block';

}

При нажатии на имя тренера отображается его подробная информация, включая фото и квалификацию. Улучшает взаимодействие пользователя с интерфейсом. Позволяет узнать больше о тренере перед записью.

7. Рендеринг календаря. Код:

function renderCalendar(year, month) {

const firstDay = new Date(year, month, 1);

const lastDay = new Date(year, month + 1, 0);

const totalDays = lastDay.getDate();

let calendarHTML = "<table><tr>";

for (let i = 0; i < firstDay.getDay(); i++) {

calendarHTML += "<td></td>";

}

for (let day = 1; day <= totalDays; day++) {

calendarHTML += `<td onclick="showTrainings('${year}-${(month + 1).toString().padStart(2, '0')}-${day.toString().padStart(2, '0')}')">${day}</td>`;

if ((firstDay.getDay() + day) % 7 === 0) {

calendarHTML += "</tr><tr>";

}

}

calendarHTML += "</tr></table>";

document.getElementById("calendar-container").innerHTML = calendarHTML;

}

Создаёт интерактивный календарь для выбора дат тренировок. Удобен для планирования занятий.

Проект веб-приложения для фитнес-клуба включает использование MySQL для хранения и управления данными. В данной части описаны основные аспекты работы с базой данных, включая структуру, создание таблиц, использование связей, а также взаимодействие приложения с базой данных.

Для реализации проекта была создана база данных, содержащая несколько таблиц:

– users: хранит информацию о пользователях.

– bookings: хранит записи на тренировки.

– schedule: содержит расписание тренировок.

– reviews: хранит записи о отзывах.

Каждая таблица имеет свою структуру, оптимизированную для хранения необходимых данных. Например, таблица users выглядит следующим образом:

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(100) NOT NULL,

password VARCHAR(255) NOT NULL,

phone VARCHAR(15) NOT NULL,

is\_admin BOOLEAN DEFAULT FALSE

);

Данная таблица включает:

– id — уникальный идентификатор пользователя.

– username, password, phone — данные пользователя.

– is\_admin — поле для разграничения прав доступа.

Для реализации связи между данными используется внешний ключ (FOREIGN KEY). Например, таблица bookings связывается с таблицей users через поле user\_id:

CREATE TABLE bookings (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

training\_date DATE,

training\_time TIME,

trainer\_name VARCHAR(255),

FOREIGN KEY (user\_id) REFERENCES users(id)

);

Пользователи могут записываться на тренировки, выбирая дату, время и тренера. Данные записываются в таблицу bookings:

INSERT INTO bookings (user\_id, training\_date, training\_time, trainer\_name)

VALUES (1, '2025-01-20', '10:00:00', 'John Doe');

При этом проверяется, что на указанное время еще есть свободные места, что реализуется через SQL-запросы с фильтрацией:

SELECT \* FROM bookings

WHERE training\_date = '2025-01-20' AND training\_time = '10:00:00';

Администратор добавляет тренировки в расписание, используя таблицу schedule:

CREATE TABLE schedule (

id INT AUTO\_INCREMENT PRIMARY KEY,

trainer\_name VARCHAR(255) NOT NULL,

training\_date DATE NOT NULL,

training\_time TIME NOT NULL

);

Добавление новой тренировки:

INSERT INTO schedule (trainer\_name, training\_date, training\_time)

VALUES ('Anna Smith', '2025-01-21', '15:00:00');

Для анализа данных используются запросы, позволяющие получить нужную информацию. Например, чтобы узнать, сколько записей на тренировки сделано за день:

SELECT COUNT(\*)

FROM bookings

WHERE training\_date = '2025-01-20';

Или чтобы вывести самых популярных тренеров:

SELECT trainer\_name, COUNT(\*) AS bookings\_count

FROM bookings

GROUP BY trainer\_name

ORDER BY bookings\_count DESC;

Администраторы управляют пользователями, расписанием и записями. Например, удаление пользователя и связанных с ним записей:

DELETE FROM users WHERE id = 5;

MySQL является надежным инструментом для хранения данных в веб-приложении. Грамотно спроектированная база данных обеспечивает корректное функционирование всех модулей приложения: от регистрации и записи на тренировки до управления расписанием и анализа данных.

## 2.2 Функции и визуальная составляющая

1 страница, на которую попадает пользователь это «Главная». Визуальное оформление можно увидеть на рисунке 1 и 2.

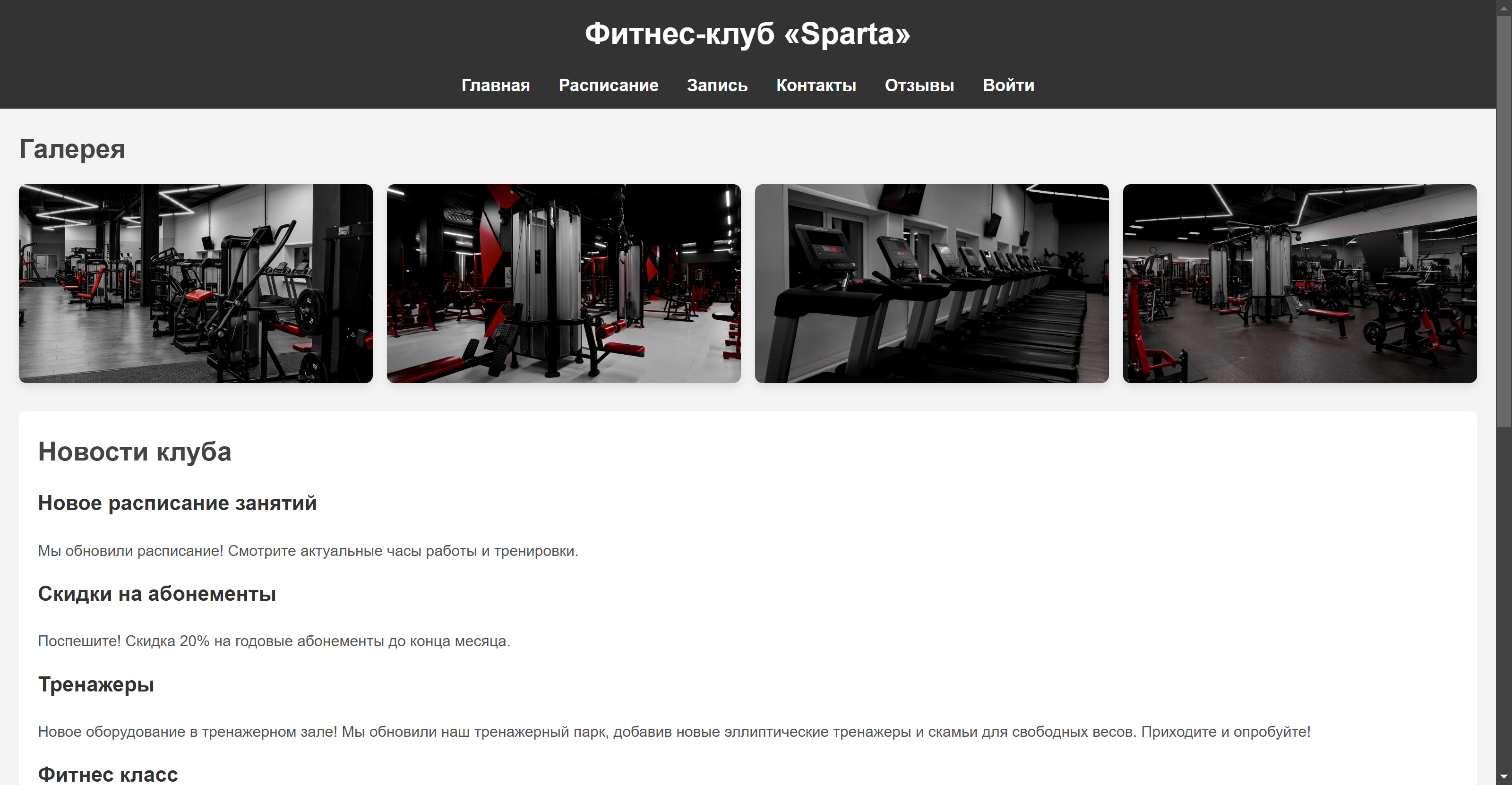


Рисунок 1 – Главная страница

В верхней части расположена навигация по всем вкладкам сайта, а также блоки с основной информацией о фитнес-клубе. Представлены галерея с фотографиями, раздел новостей клуба, а чуть ниже располагается блок с тренерами.

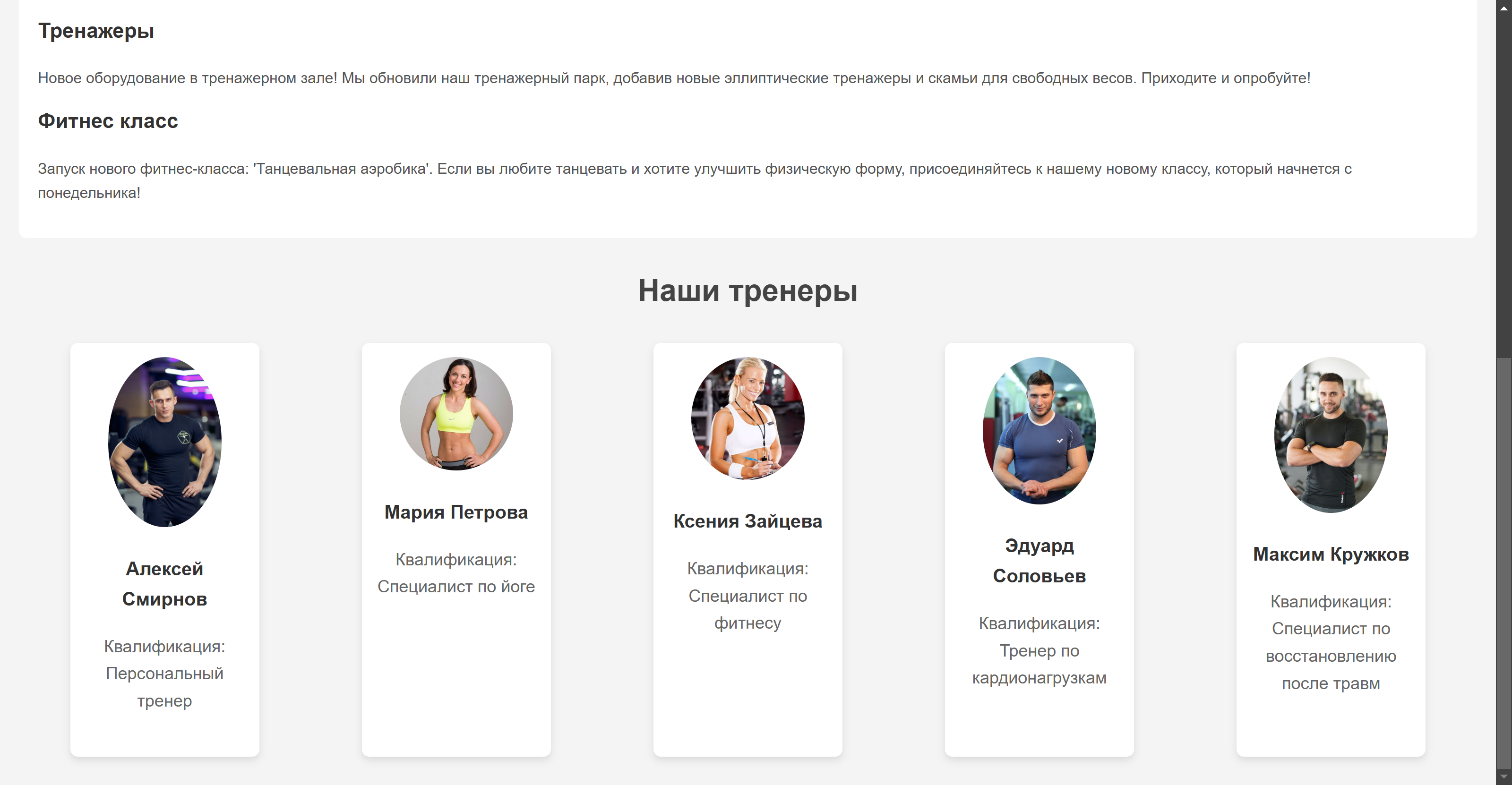


Рисунок 2 – Главная страница (блок тренеры)

При нажатии на блок с тренерами происходит перенос на страницу с расписанием занятий. При наведении происходит небольшое движение картинки для интерактивного визуального эффекта. Также на данной странице уже происходит проверка пользователя и в соответствии с проверкой появляется кнопка «Личный кабинет» и «Выйти» или «Войти», в зависимости от того зашел пользователь в свой аккаунт или нет. Данная технология реализована на всех основных страницах.

Попав на страницу с расписанием, можно выбрать дату в календаре и узнать актуальное расписание из базы данных schedule. Добавлять актуальное расписание может администратор в личном кабинете. Пример расписания и визуал календаря изображен на рисунке 3.

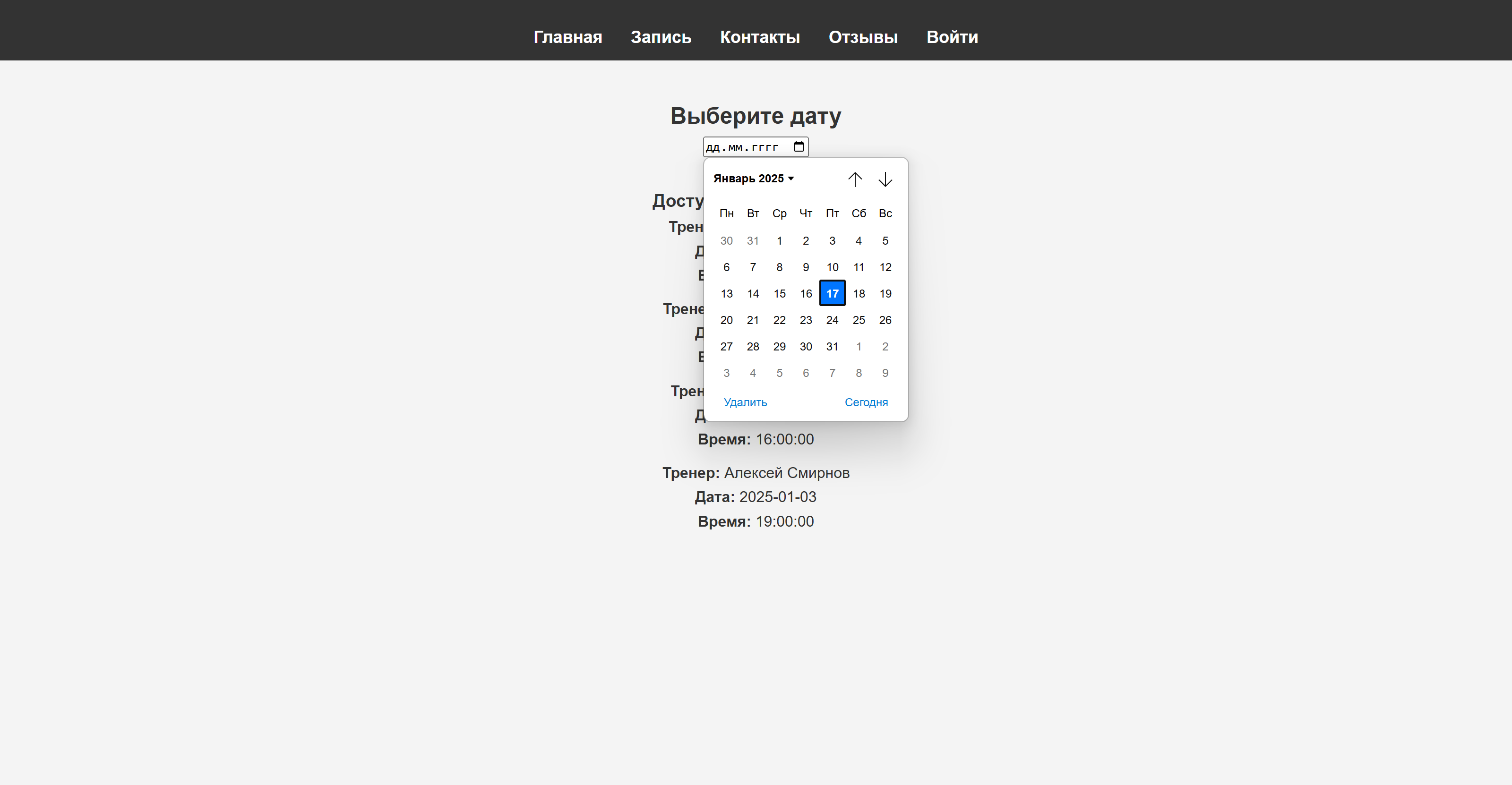


Рисунок 3 – Страница с расписанием занятий

Следующая страница, на которую может попасть пользователь — это «Запись», но здесь уже если он не авторизован, то придется регистрировать аккаунт. При нажатии на страницу происходит проверка авторизации и если она не пройдена, то осуществляется переход на страницу регистрации. Страница регистрации и авторизации показана на рисунке 4. Она нужна для хранения информации о пользователях, их паролях, логинах и потом пригодится для их смены в личном кабинете.

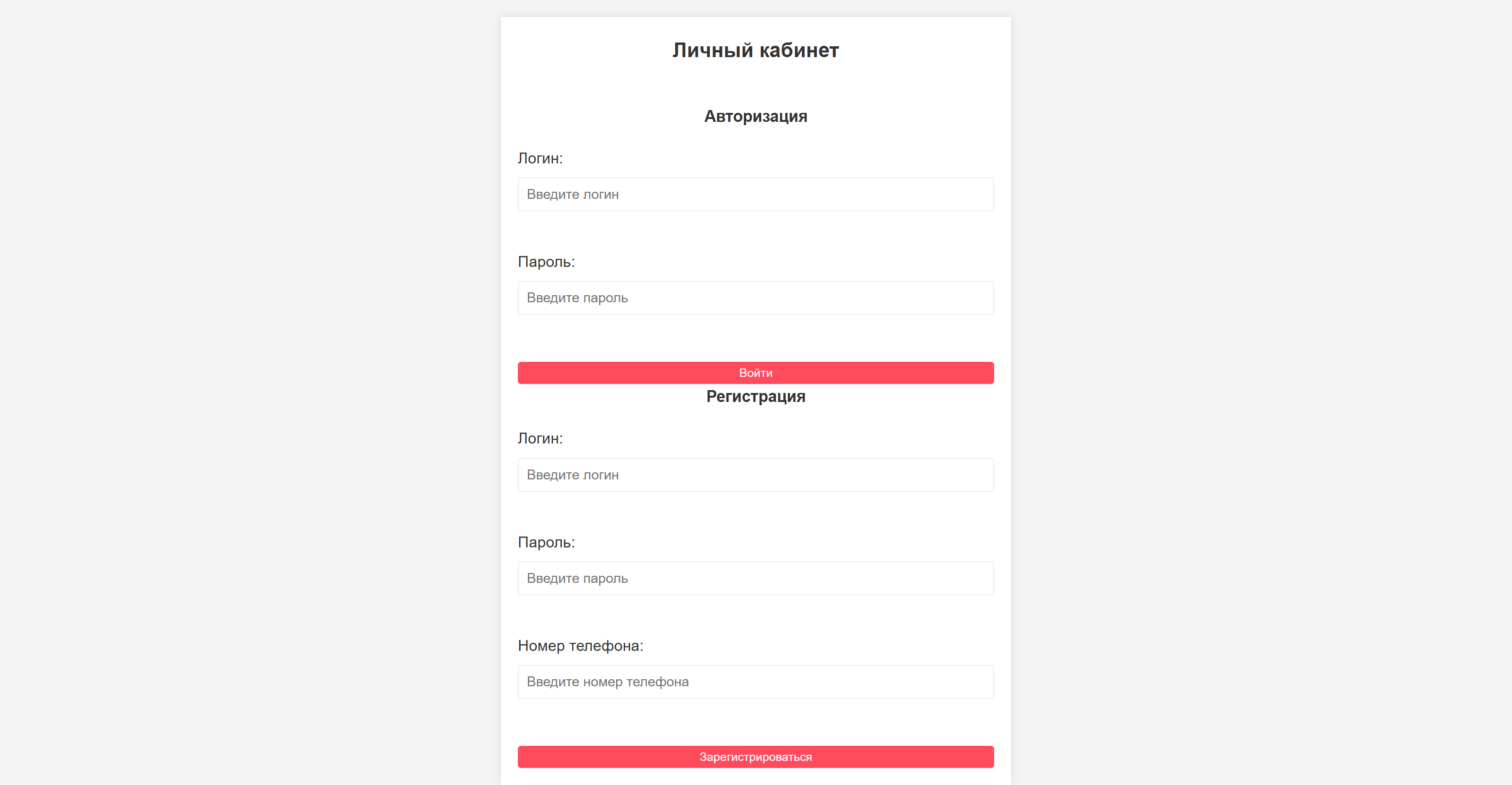


Рисунок 4 – Страница авторизации и регистрации

После прохождения авторизации можно попасть на страницу «Запись на занятия» или в «Личный кабинет». Через базу данных с расписанием на странице с записью появляются занятия, это показано на рисунке 5. Можно выбрать занятие и внизу страницы нажать записаться, после чего появится сообщение об успешной записи и добавление информации в БД.

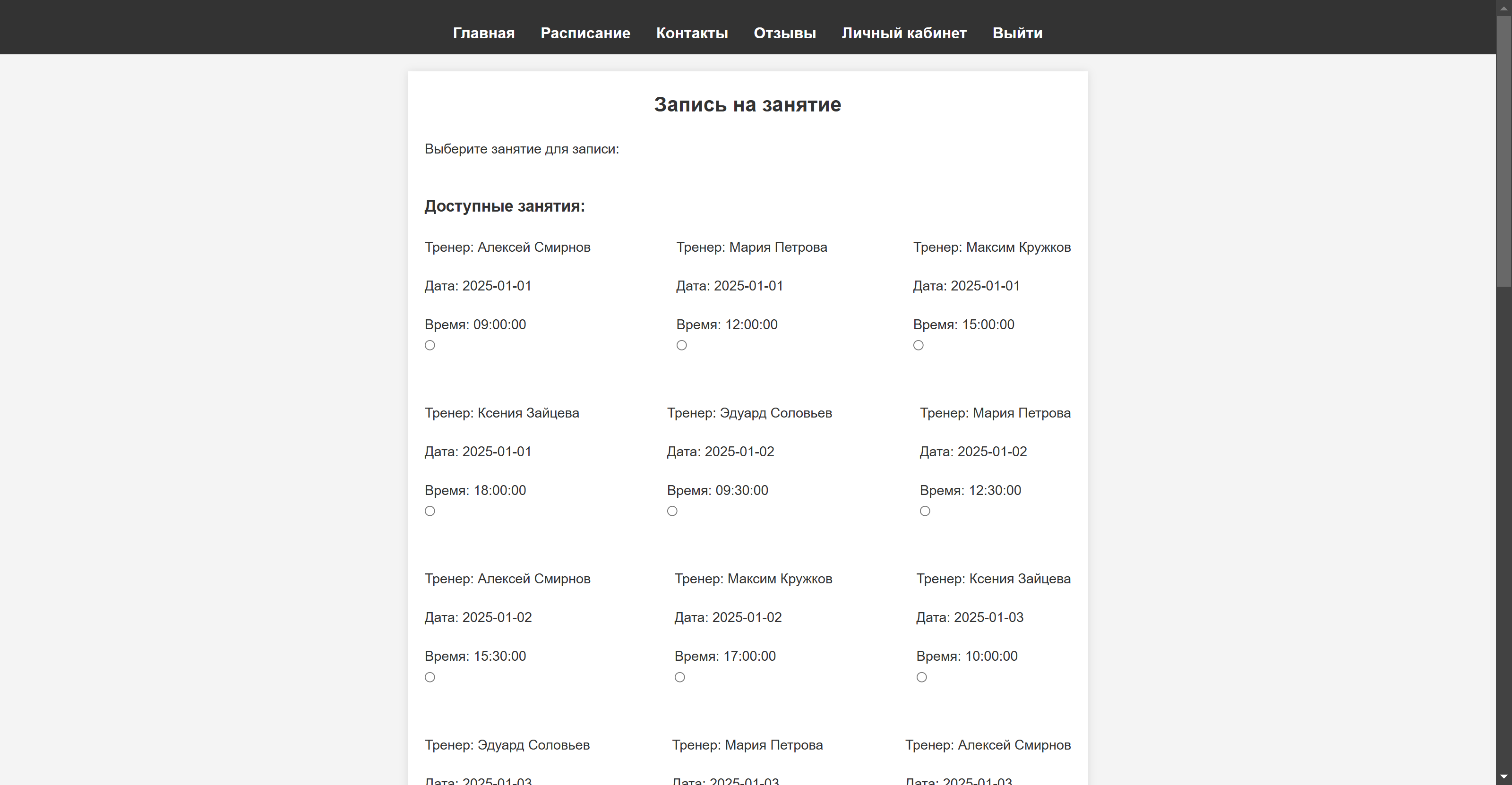


Рисунок 5 – Выбор занятия на странице записи

Внизу страницы можно увидеть записи на занятия для пользователя, функция изображена на рисунке 6.

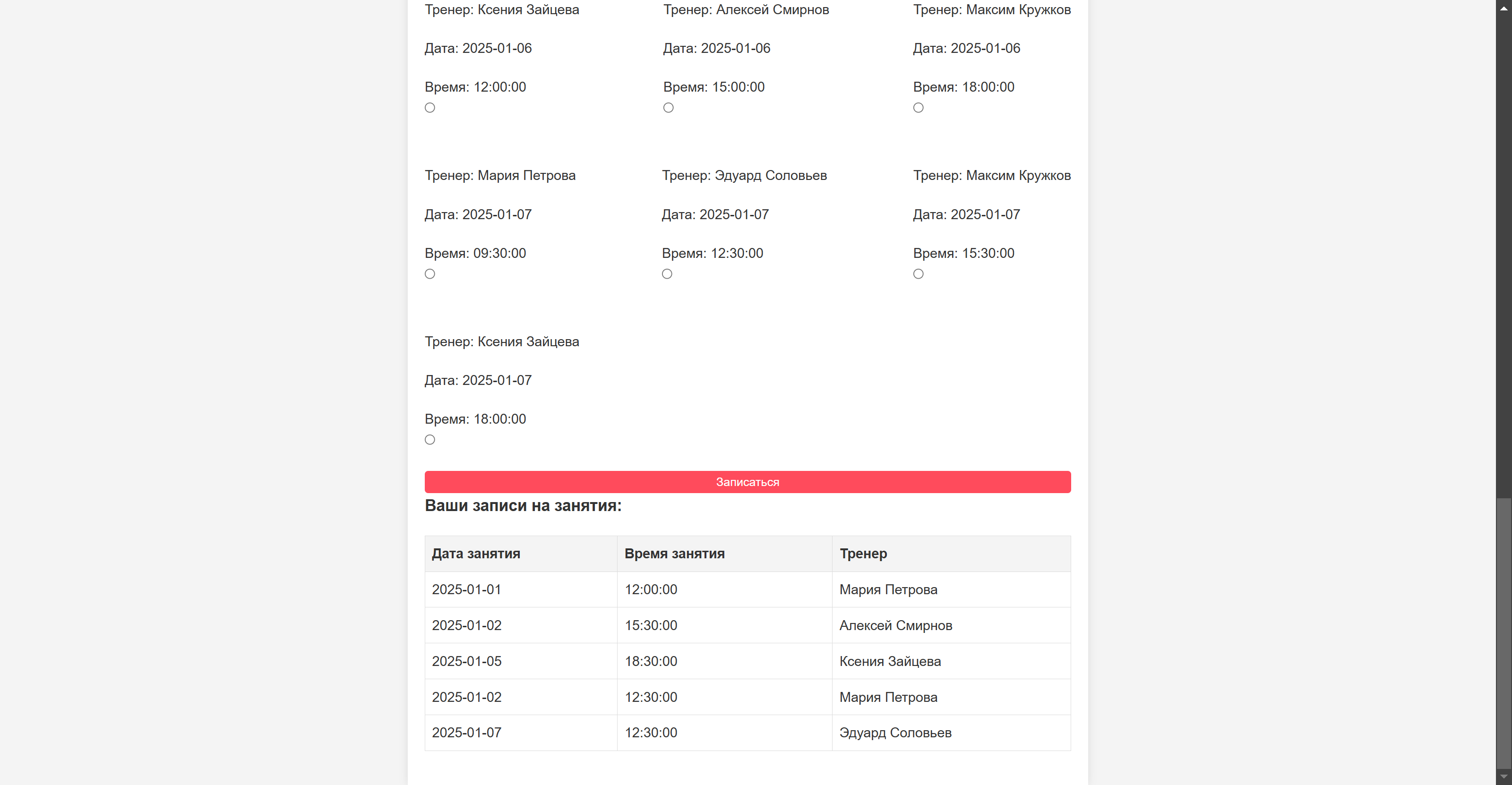


Рисунок 6 – Таблица с записями пользователя

Следующая страница из меню навигации — «Контакты». Показывает информацию о том, как добраться до фитнес-клуба и содержит ссылки на карты внизу страницы. На рисунке 7 показано, как страница выглядит.

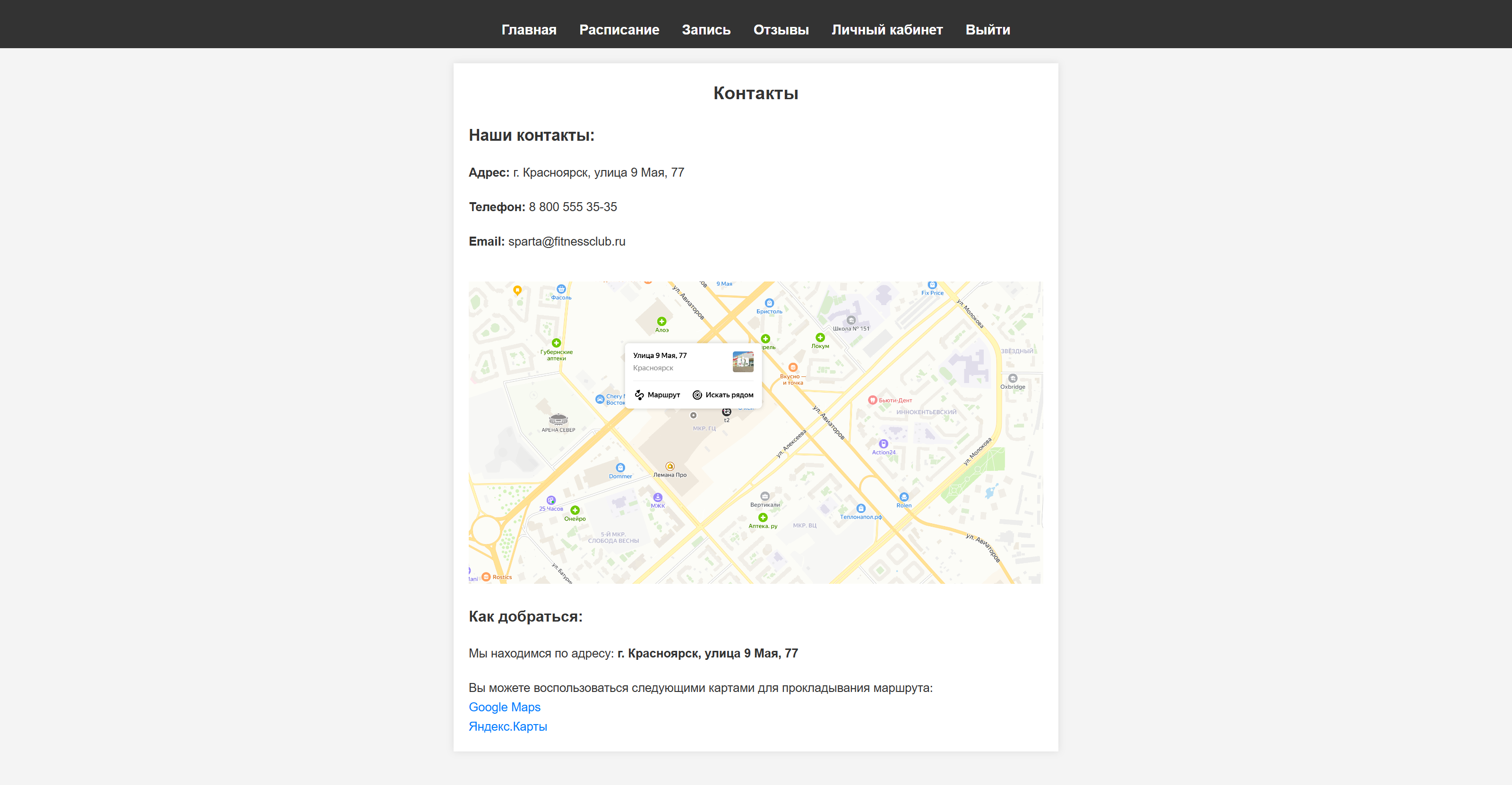


Рисунок 7 – Страница «Контакты»

Дальше пользователь может попасть на страницу с отзывами. Для того, чтобы оставить отзыв, нужно быть авторизованным. После авторизации можно добавить отзыв через окно в базу данных reviews и он сразу отобразится на сайте. Как выглядят добавленные отзывы без авторизации можно на рисунке 8, а пример добавления нового отзыва на рисунке 9.

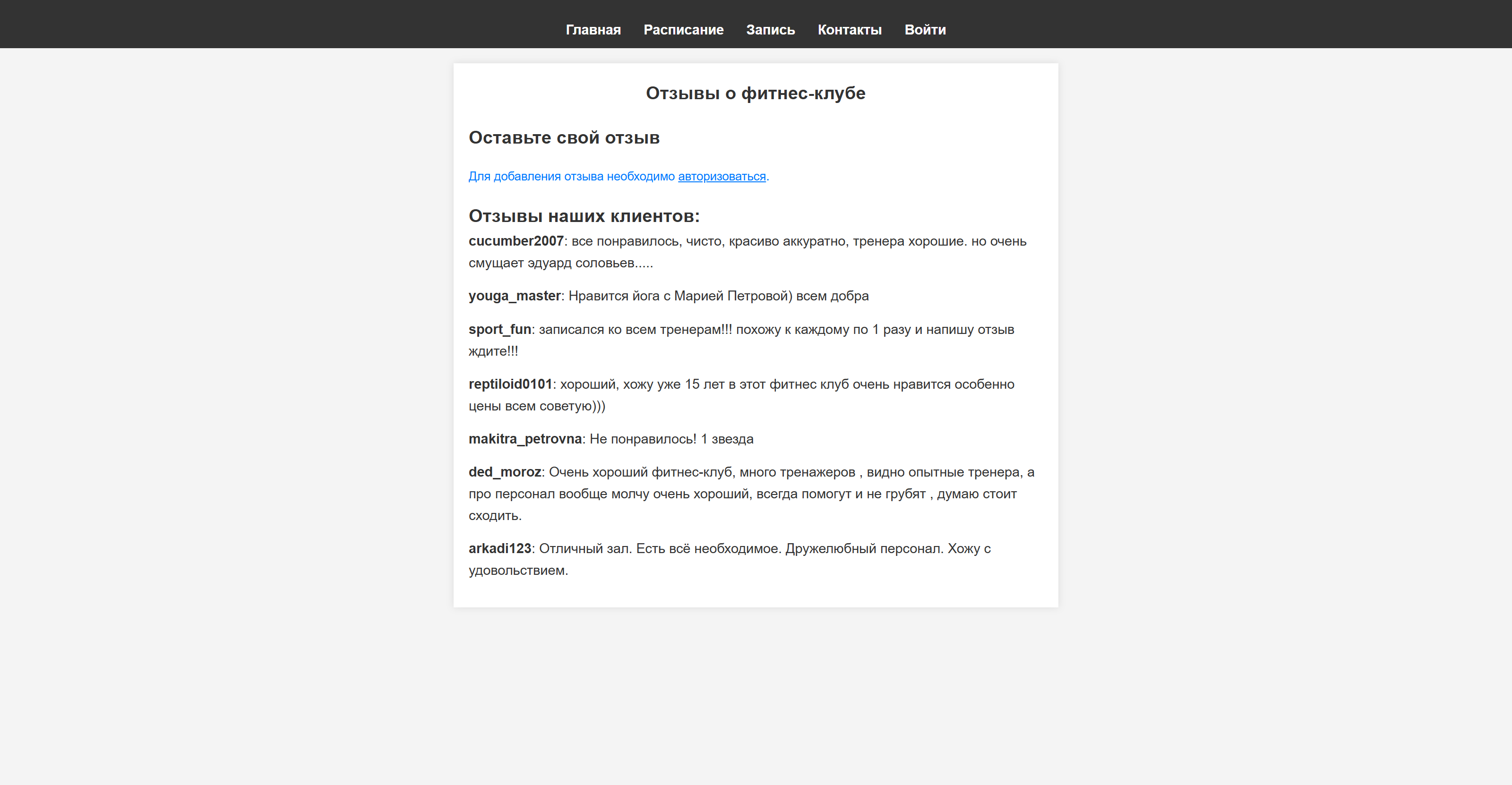


Рисунок 8 – Страница «Отзывы» для неавторизованного пользователя

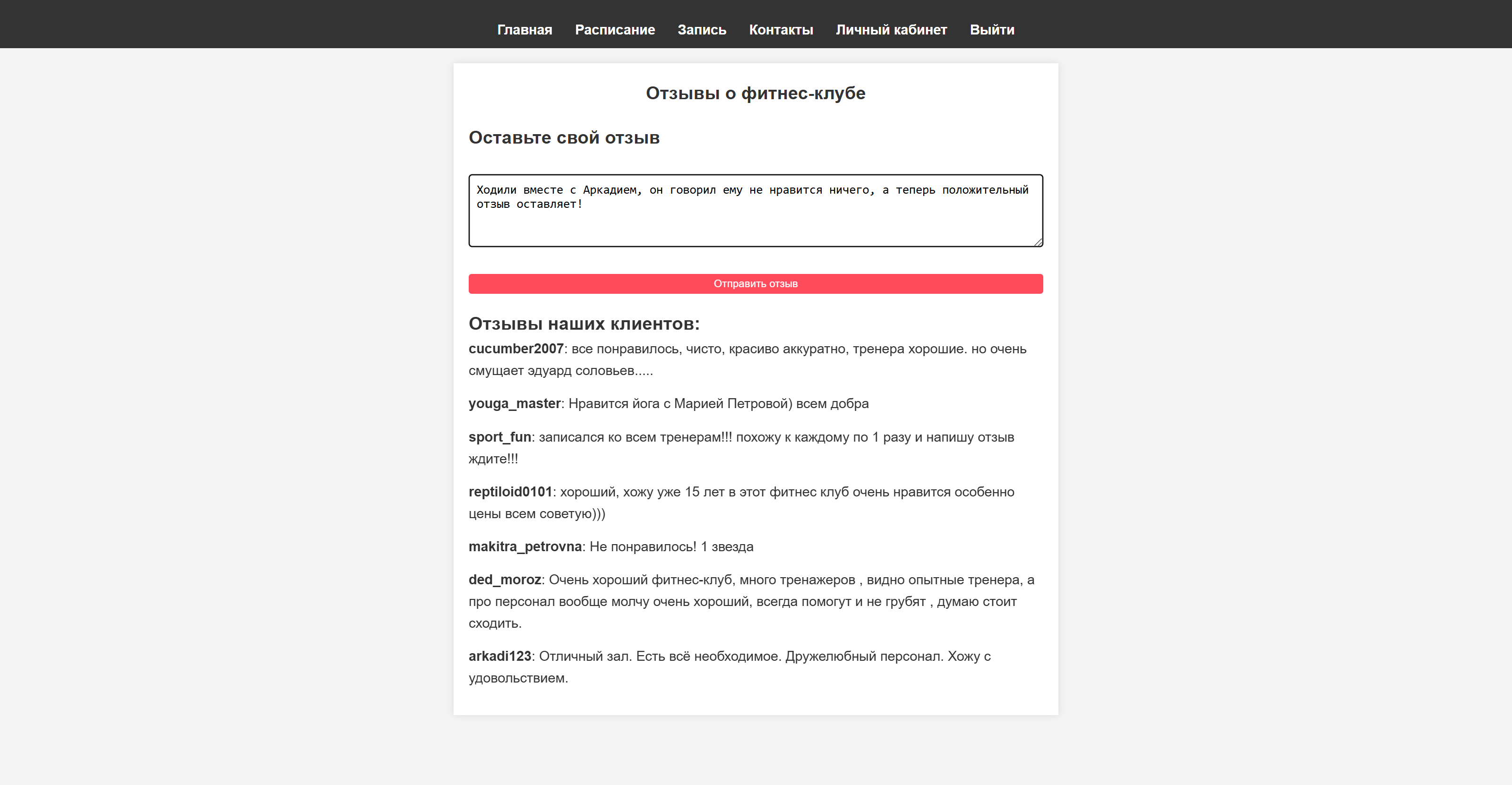


Рисунок 9 – Пример добавления отзыва после авторизации

Страница «Личный кабинет» появляется только у авторизованных пользователей, до этого вместо него активна кнопка «Войти». Для обычных пользователей личный кабинет открывает функции смены логина, телефона и пароля, а также добавляет интерактивные тесты.

При смене информации делается запрос в БД users и изменение данных пользователя. Пример как выглядит поле со сменой данных можно увидеть на рисунке 10.

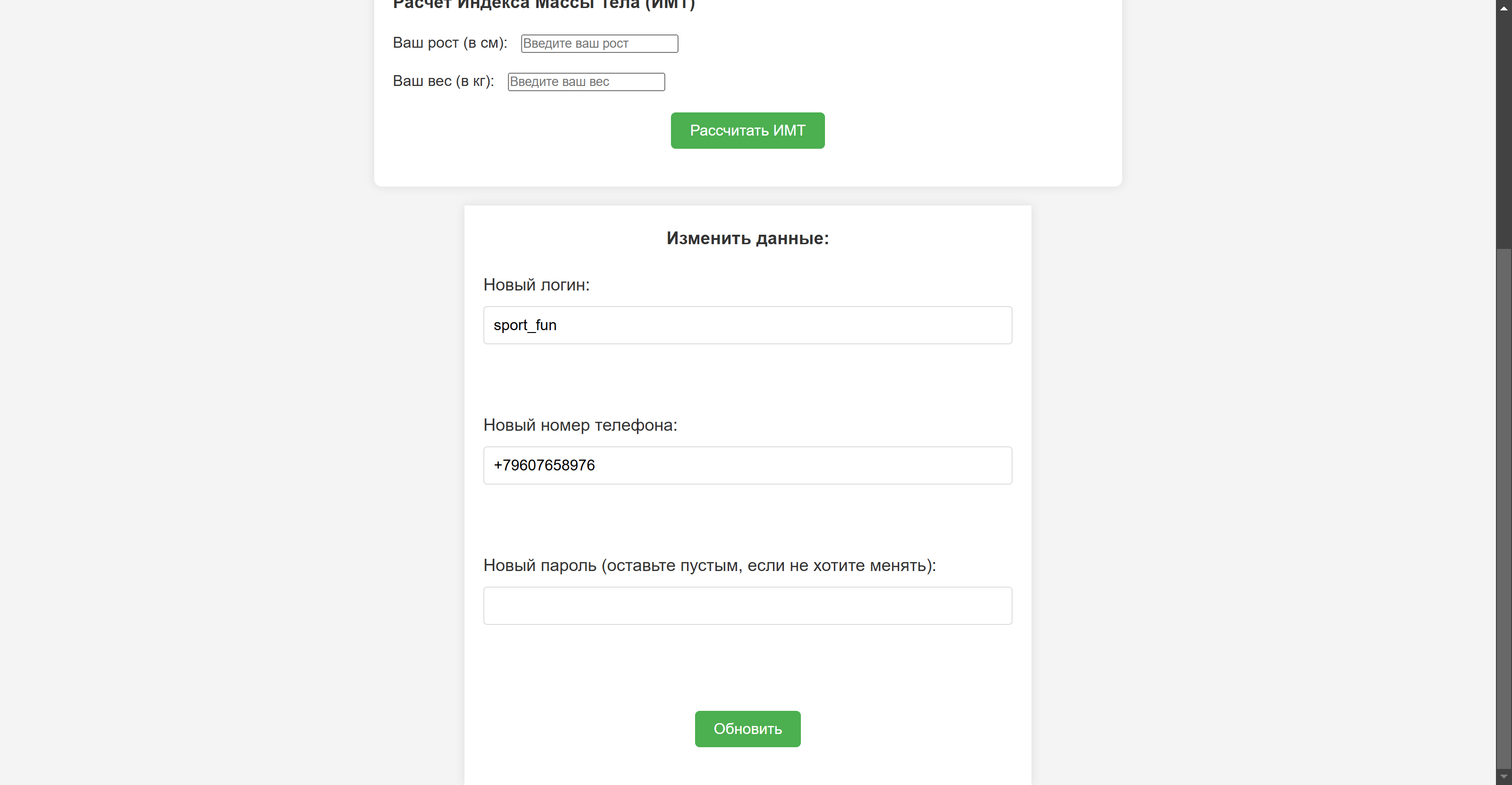


Рисунок 10 – Форма для смены данных в личном кабинете пользователя

Тест на уровень физической подготовки за 3 вопроса отвечает на вопрос о физической форме пользователя. Реализован он по принципу подсчета баллов за ответы и выведения результатов в зависимости от баллов. Пример результата показан на рисунке 11.

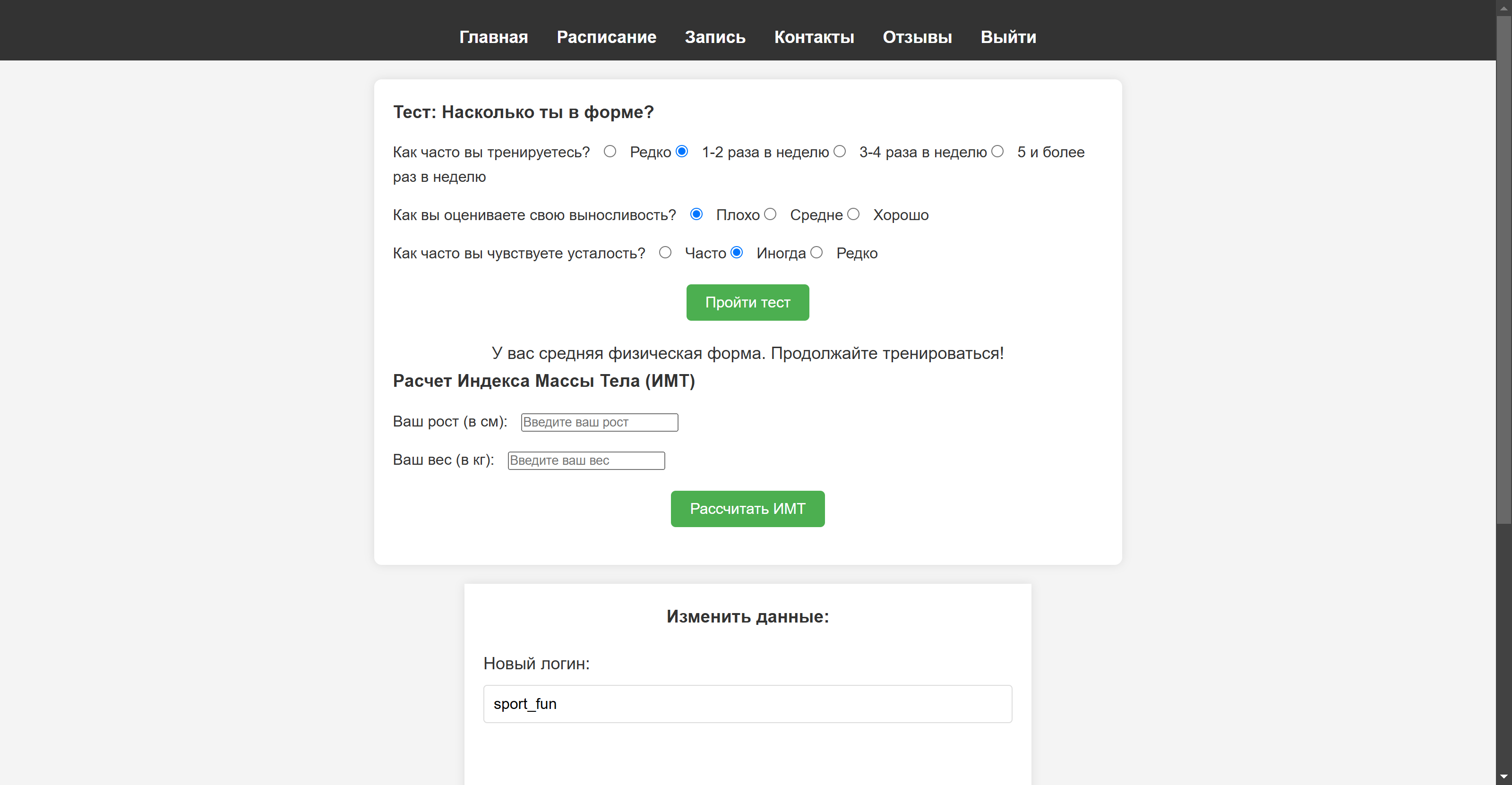


Рисунок 11 – Пример результата для интерактивного теста «Насколько ты в форме?»

Интерактивная форма для подсчета ИМТ может помочь в вопросе о похудении или наборе массы. По специальной формуле в зависимости от роста и веса рассчитывается индекс массы тела, примерный расчет показан на рисунке 12. Также производится оценка массы тела, есть недовес, ожирение или нормальный вес.

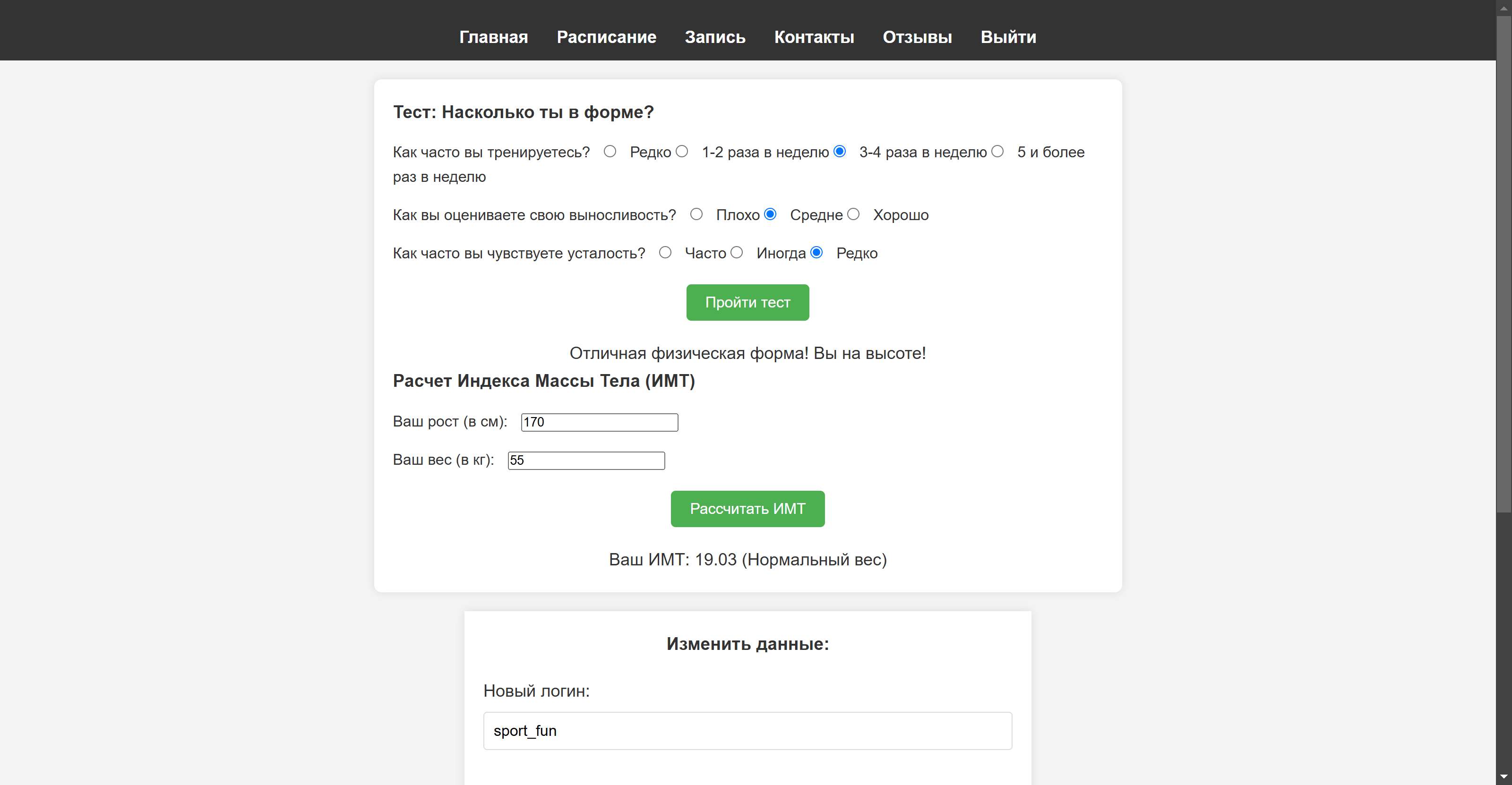


Рисунок 12 – Пример работы интерактивной формы Расчет ИМТ

Если оказалось, что в личный кабинет хочет попасть администратор, то происходит перевод на страницу администрирования. Здесь нужно выбрать действие: просмотр пользователей – переносит на страницу со списком пользователей, просмотр записей на занятия – выдает список записей из базы данных, добавить занятие в расписание – открывает страницу с формой для добавления занятия. На рисунке 13 изображена функция просмотра всех пользователей и их удаления. Нажатием кнопки «Удалить» из базы данных удаляется вся информация о пользователе и его записях на занятия.

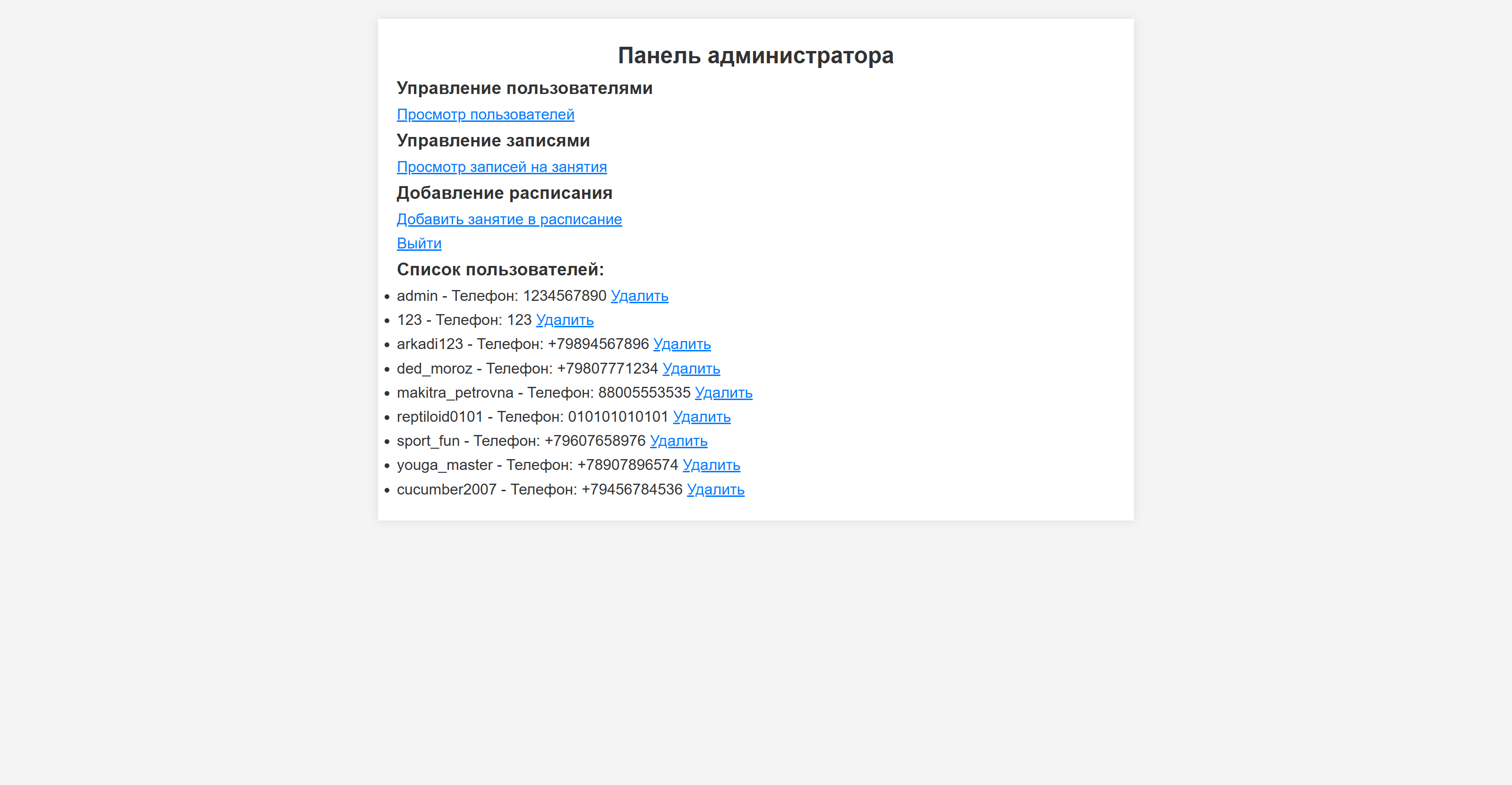


Рисунок 13 – Список пользователей

На рисунке 14 представлен внешний вид таблицы с данными о записях.

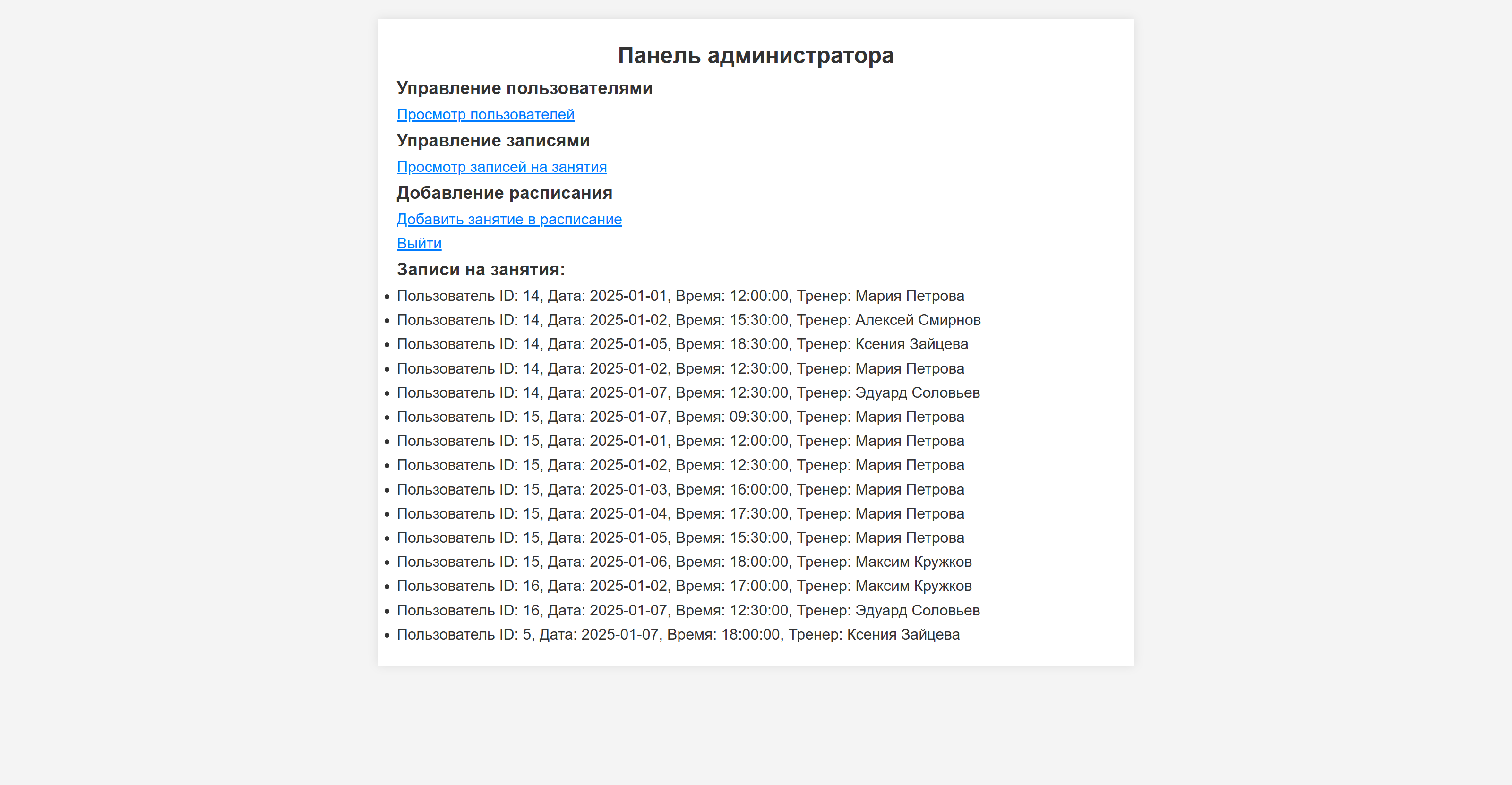


Рисунок 14 – Список записей пользователей

А на рисунке 14 можно увидеть функцию добавления расписания, сделанную через удобную форму. Все функции осуществляются через запросы к базам данных и их обновлением. Также на странице реализован выход из аккаунта, она переносит на главную страницу.

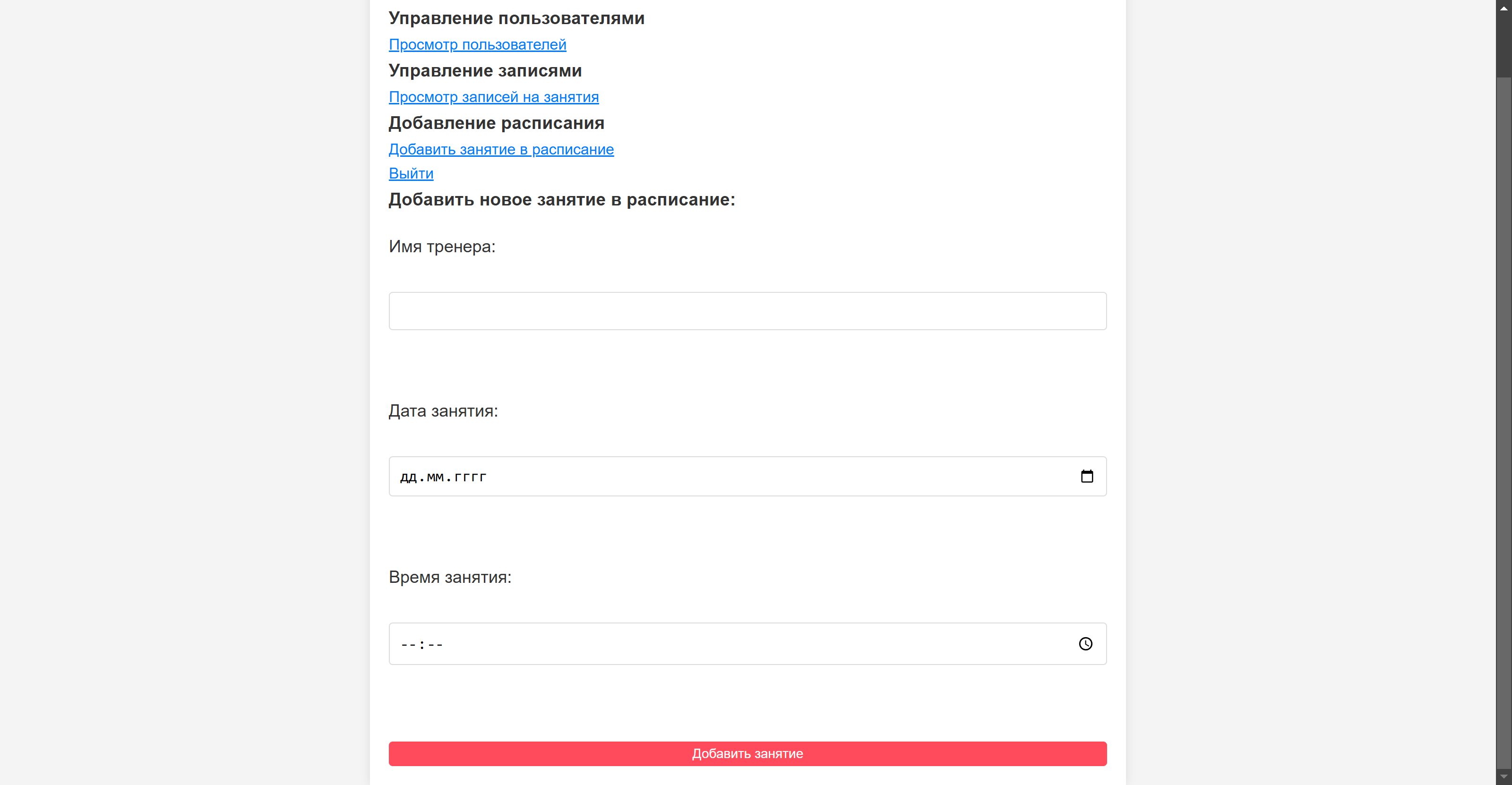


Рисунок 15 – Добавление расписания

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была достигнута поставленная цель — разработано web-приложение для фитнес-клуба с применением современных web-технологий. Данный проект позволил реализовать актуальные функции, которые удовлетворяют запросы пользователей, повышают удобство взаимодействия с клубом и способствуют повышению эффективности его работы.

Реализованное приложение включает пять ключевых страниц, каждая из которых выполняет свою задачу. Пользователи могут получить полную информацию о клубе и тренерах, ознакомиться с расписанием занятий, зарегистрироваться и авторизоваться, записаться на тренировки, а также оставить отзывы. Благодаря использованию адаптивного дизайна и современных технологий, таких как HTML, CSS, JavaScript, PHP и MySQL, приложение демонстрирует высокую производительность и удобство использования.

В процессе разработки была создана база данных, которая обеспечивает надежное хранение данных о пользователях, тренерах и занятиях. Реализована возможность взаимодействия с базой данных, включая регистрацию и авторизацию пользователей, просмотр расписания и управление записями на тренировки.

Для администраторов предусмотрены функции управления пользователями и записи на занятия, что позволяет эффективно контролировать работу клуба.

Данный проект показывает, как современные технологии могут быть интегрированы в бизнес-процессы фитнес-индустрии. Разработанное web-приложение не только улучшает пользовательский опыт, но и открывает новые возможности для взаимодействия с клиентами, что особенно важно в условиях растущей конкуренции.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Браун, Э. Изучаем JavaScript: руководство для современных разработчиков / Э. Браун ; пер. с англ. – 2-е изд. – Москва : Эксмо, 2017. – 432 с.
2. Дукетт, Дж. HTML и CSS: разработка и дизайн веб-сайтов / Дж. Дукетт ; пер. с англ. – Москва : Вильямс, 2016. – 512 с.
3. Стефанов, С. PHP: Объекты, шаблоны и методики программирования / С. Стефанов ; пер. с англ. – Москва : Вильямс, 2014. – 496 с.
4. Флэнаган, Д. JavaScript: Полное руководство / Д. Флэнаган ; пер. с англ. – 7-е изд. – Санкт-Петербург : Символ-Плюс, 2020. – 1088 с.
5. Осборн, Дж. Профессиональная разработка frontend на Vue.js 3 / Дж. Осборн, Т. Снелл ; пер. с англ. – Москва : Диалектика, 2022. – 432 с.
6. Мартин, Р. Чистый код. Создание, анализ и рефакторинг / Р. Мартин ; пер. с англ. – Санкт-Петербург : Питер, 2021. – 464 с.
7. Иванов, А. В. Современные тенденции в разработке пользовательских интерфейсов одностраничных приложений (SPA) / А. В. Иванов, С. К. Петрова // Труды университета. Серия: Информационные технологии. – 2022. – № 4. – С. 45–58.
8. Кузнецов, Д. А. Применение RESTful API в интеграции веб-сервисов / Д. А. Кузнецов // Программная инженерия. – 2021. – № 3. – С. 22–30.
9. HTML и CSS: разработка и дизайн веб-сайтов [Электронный ресурс] // HTMLBook. – URL: https://htmlbook.ru/ (дата обращения: 12.01.2025).
10. JavaScript Tutorial [Электронный ресурс] // W3Schools. – URL: https://www.w3schools.com/js/ (дата обращения: 10.01.2025).
11. Основные принципы построения веб-приложений [Электронный ресурс] // Хабр. – URL: https://habr.com/ru/ (дата обращения: 14.01.2025).
12. PHP и MySQL. Руководство по созданию веб-приложений [Электронный ресурс] // Metanit. – URL: https://metanit.com/php/mysql/ (дата обращения: 15.01.2025).
13. Справочник по CSS [Электронный ресурс] // MSiter. – URL: https://msiter.ru/references/css-reference (дата обращения: 15.01.2025).
14. Спецификация ECMAScript® 2023 Language Specification (ECMA-262) [Электронный ресурс]. – URL: https://tc39.es/ecma262/ (дата обращения: 01.04.2024).