МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

WEB-приложение «Тренажерный зал»

Выполнил студент Глушакова Алеся Юрьевна

(Ф.И.О.)

Руководитель проекта преп.-стаж. Сенюк В.К.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты преп.-стаж. Сенюк В.К.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер преп.-стаж. Сенюк В.К.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2020

**Содержание**

[**Введение 6**](#_Toc40280463)

[**1. Постановка задачи 7**](#_Toc40280464)

[**1.1. Алгоритмы решения 7**](#_Toc40280465)

[**1.2 Аналитический обзор литературы 8**](#_Toc40280466)

[**2. Разработка архитектуры проекта 12**](#_Toc40280467)

[**2.1. Обобщенная структура 12**](#_Toc40280468)

[**2.2. Спецификация функциональных требований 13**](#_Toc40280469)

[**2.3 Реализация консультанта на сокетах 15**](#_Toc40280470)

[**3. Разработка программной системы 16**](#_Toc40280471)

[**3.1. Структура проекта 16**](#_Toc40280472)

[**3.2. Разработка модели данных 19**](#_Toc40280473)

[**4. Руководство пользователя 24**](#_Toc40280474)

[**4.1. Регистрация пользователя 24**](#_Toc40280475)

[**4.2. Аутентификация и авторизация пользователя 24**](#_Toc40280476)

[**4.3. Клиент 25**](#_Toc40280477)

[**4.4. Тренер 26**](#_Toc40280487)

[**5 Тестирование 29**](#_Toc40280499)

[**Заключение 31**](#_Toc40280500)

[**Список литературы 32**](#_Toc40280501)

[**Приложение А 33**](#_Toc40280502)

[**Приложение Б 34**](#_Toc40280503)

# Введение

В данном курсовом проекте разработано WEB-приложение «Интернет-магазин брендовой одежды». Логически оно разделено на две части: серверную, написанную на Node.js, и клиентскую, в которой использовался язык JavaScript с внедрением  шаблонизатора [Handlebars](http://handlebarsjs.com/), физически же WEB-приложение представляет собой одно приложение.

В настоящее время реализовано много современных компьютерных приложений, которые предоставляют удобный интерфейс для работы с данными в отрасли спорта, тренировок и администрирования спортивных сетей.

Однако, с учетом глобализации и урбанизации, а также активным развитием рыночной экономики, спрос на программы –тренажерные залы не убывает. Сидячий образ жизни людей сделал поддержание здоровье актуальной проблемой, а веб-приложения тренажерных центров упрощают задачу привлечения клиентов легким бронированием, а также облегчает администрирование и учет тренеров, а также подсчет клиентов.

С учетом вышеописанных факторов, целью своего курсового проекта мною была выбрана разработка программы-тренажерного зала, функционал которой будет способен оформлять абонементы в тренажерный зал.

Пояснительная записка курсового проекта состоит из 31 страниц, 13 рисунков, 2 приложений, 5 источников литературы.

Основная цель курсового проекта: разработка web-приложения «Тренажерный зал».

В первом разделе рассматриваются основные технологии, которые использовались в разработке данного приложения, а также его прототипы и актуальность задачи.

Во втором разделе описана архитектура базы данных.

В третьем разделе предоставлена информация о разработанных объектах базы данных.

Четвертый раздел содержит руководство пользователя для разработанного клиентского приложения.

В пятом разделе представлены результаты тестирования приложения.

В заключении описывается результат курсового проектирования и задачи, которые были решены в ходе разработки приложения.

1. **Постановка задачи**

Главная задача курсового проекта является разработка WEB- приложения, позволяющего пользователю просматривать записи о своих тренировках, целях, а также занятиях и тренировочных программах, а также удалять и изменять свои тренировки, оформлять заказ. Тренер же может назначать вышеописанные занятие пользователю, а также принимать заказы. В данном курсовом проекте требовалось реализовать следующие задачи:

* сохранение рабочей информации в хранилище данных;
* организация регистрации и входа в систему со стороны пользователей;
* реализация архитектуры MVC;
* возможность просмотра, добавления, удаления, редактирования абонементов, упражнений;

## **Алгоритмы решения**

При реализации курсового проекта использовались технологии Node.js[1], фреймворк Express[2],WebSoket[4] для реализации back-end, для front-end части bootstrap [5] и ui calendar, а также сервис cloudinary для хранения изображений, JQuery[6], JavaScript[7], [Handlebars](http://handlebarsjs.com/)[8], CSS[9]. lowdb в качестве базы данных.

Node.js –  программная платформа, основанная на движке [V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript)) (транслирующем [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) в [машинный код](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4)), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) взаимодействовать с устройствами [ввода-вывода](https://ru.wikipedia.org/wiki/%D0%92%D0%B2%D0%BE%D0%B4-%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4) через свой [API](https://ru.wikipedia.org/wiki/API) (написанный на [C++](https://ru.wikipedia.org/wiki/C%2B%2B)), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль [веб-сервера](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), но есть возможность разрабатывать на Node.js и десктопные оконные приложения и даже программировать микроконтроллеры. В основе Node.js лежит [событийно-ориентированное](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [асинхронное (или реактивное)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) программирование с [неблокирующим вводом/выводом](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B0%D1%8F_%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F).

Express – [фреймворк web-приложений](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для Node.js, реализованный как [свободное и открытое программное обеспечение](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%B8_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) под [лицензией MIT](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F_MIT). Он спроектирован для создания [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и [API](https://ru.wikipedia.org/wiki/API). Де-факто является стандартным каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описывает его как созданный на основе написанного на языке [Ruby](https://ru.wikipedia.org/wiki/Ruby) каркаса [Sinatra](https://ru.wikipedia.org/wiki/Sinatra), подразумевая, что он минималистичен и включает большое число подключаемых [плагинов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD).

WebSoket – это передовая технология, которая позволяет создавать интерактивное соединение между клиентом (браузером) и сервером для обмена сообщениями в режиме реального времени. Веб-сокеты, в отличие от HTTP, позволяют работать с двунаправленным потоком данных, что делает эту технологию совершенно уникальной.

Bootstrap – свободный набор инструментов для создания сайтов и веб-приложений, который включает в себя HTML и CSS – шаблоны оформления для веб-форм, кнопок, меток, слайд шоу, блоков навигации и прочих компонентов веб-интерфейса.

JQuery– [набор функций JavaScript](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript), фокусирующийся на взаимодействии [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) и [HTML](https://ru.wikipedia.org/wiki/HTML). Библиотека jQuery помогает легко получать доступ к любому элементу [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), обращаться к атрибутам и содержимому элементов [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), манипулировать ими. Также библиотека jQuery предоставляет удобный [API](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для работы с [AJAX](https://ru.wikipedia.org/wiki/AJAX)

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией стандарта ECMAScript (стандарт ECMA-262). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

[Handlebars](http://handlebarsjs.com/) – это шаблонный процессор, который динамически генерирует вашу HTML-страницу, что экономит ваше время на ручном обновлении.

# 1.2 Аналитический обзор литературы

На сегодняшний день услуги, предоставляемые тренажерными центрами, увеличиваются в объеме, а сама отрасль приобретает все больше динамики. Полный контроль над всеми процессами в данной сфере способны обеспечить только надежные информационные технологии. В силу этих причин транспортные компании вынуждены внедрять инструменты, которые способны обеспечить наглядность, прозрачность и учет производимых операций.

На данный момент, данная сфера наполнена многочисленными сайтами, различными по своему исполнению, но имеющими общий основной интерфейс и спектр услуг.

Рассмотрим 3 наиболее интересных и популярных решения в области логистики управления складом:

* GYM24[1];

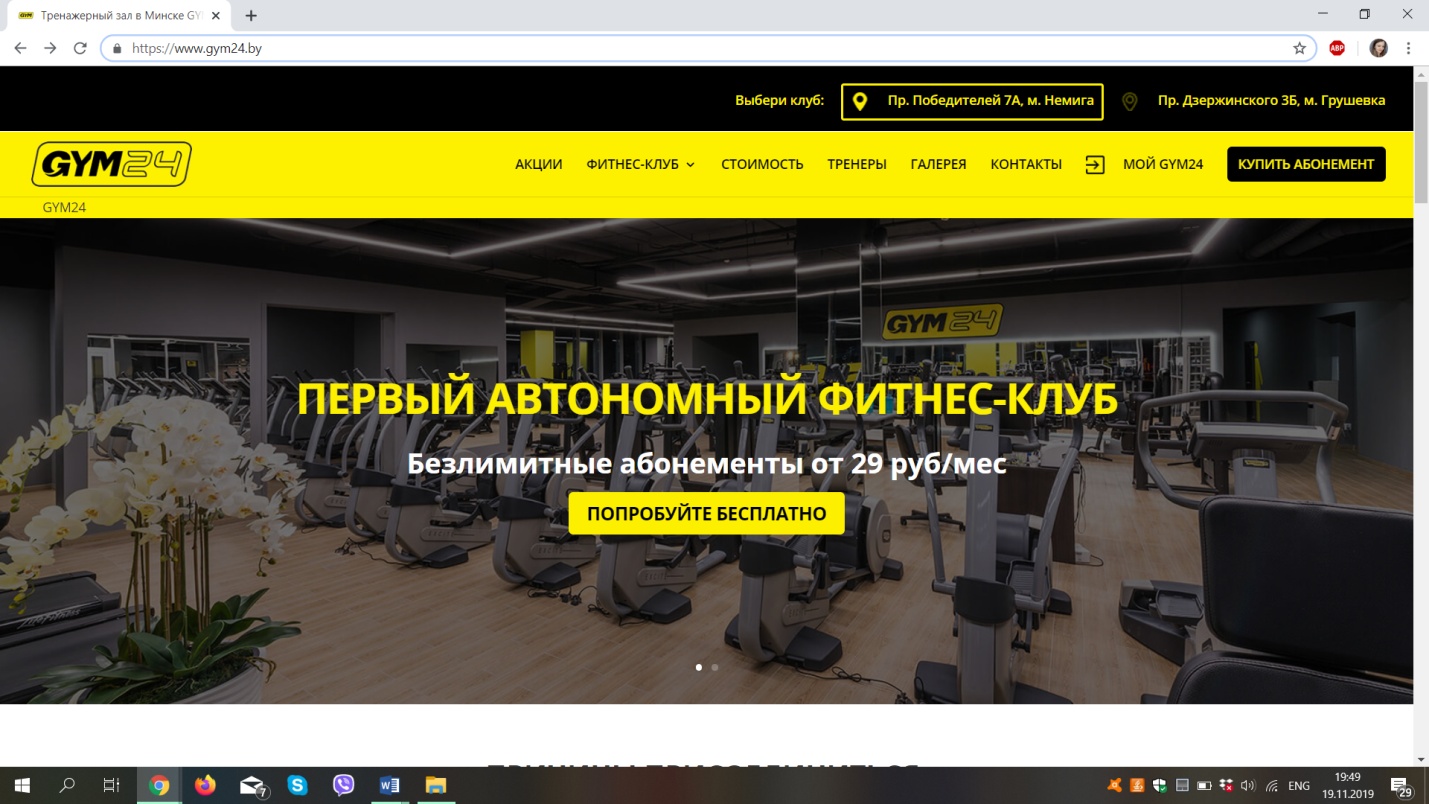


Рисунок 1.1–GYM24

К функциям данной программы относятся:

* вход в личный кабинет;
* заказ абонемента;
* просмотр контактов и местоположения зала;
* возможность увидеть галерею;

К достоинствам «GYM24» можно отнести ее функциональность, не перегруженность и ненавязчивый дизайн, а также наглядность. Данное

приложение использует простую базу данных. Также удобным решением является применение карусели с картинками на главной странице, которая, автоматически прокручиваясь, демонстрирует фотографии спортзала и его возможности.

К недостаткам же можно отнести обилие крупных некликабельных картинок, которые выглядят как ссылки и могут смущать конечного пользователя.

* ARGYMENT [2];

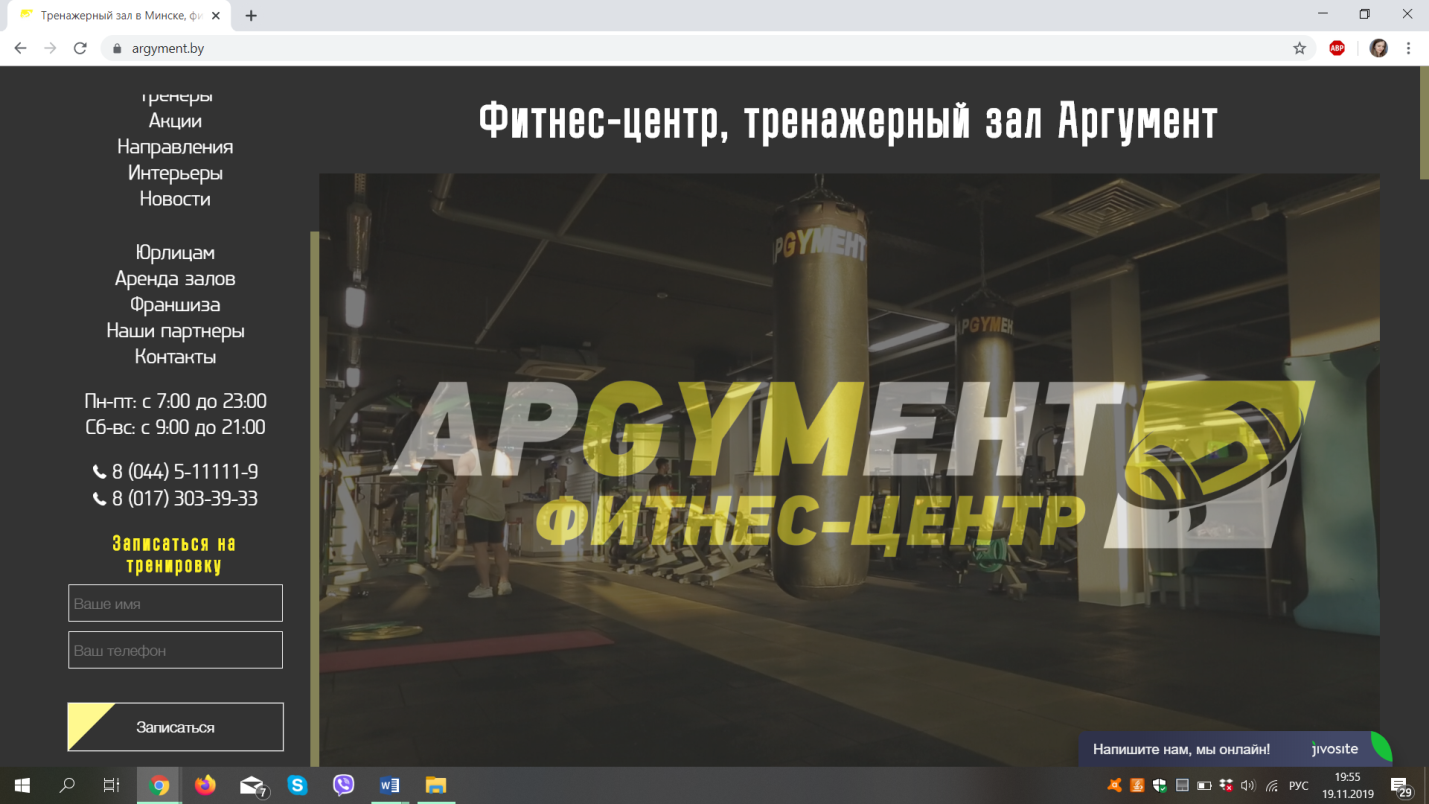


Рисунок 1.2– Интерфейс «ARGYMENT»

«*ARGYMENT*», на мой взгляд, наиболее удобен в функциональном отношении с точки зрения незарегистрированного пользователя, так как она сразу имеет возможность на главной странице записаться на тренировку, а также просмотреть видеоролик о возможностях спортивного центра.

Также в приложении есть разделения на направления, что также облегчает навигацию по сайту.

К недостаткам можно причислить неординарное решение помещения навигационного меню сбоку, а также небольшой кегль шрифта.

* R1SPORT [3];

Данный сайт отличается от предыдущих сайтов глубиной своих страниц, наличием карты, а также огромным количеством JS-анимаций, которые отвлекают пользователя от первоначальной цели сайта. Однако к плюсам конкретного сайта можно отнести наличие в навигационном меню пункта «Программы» с возможностью просмотреть или построить программу тренировок.

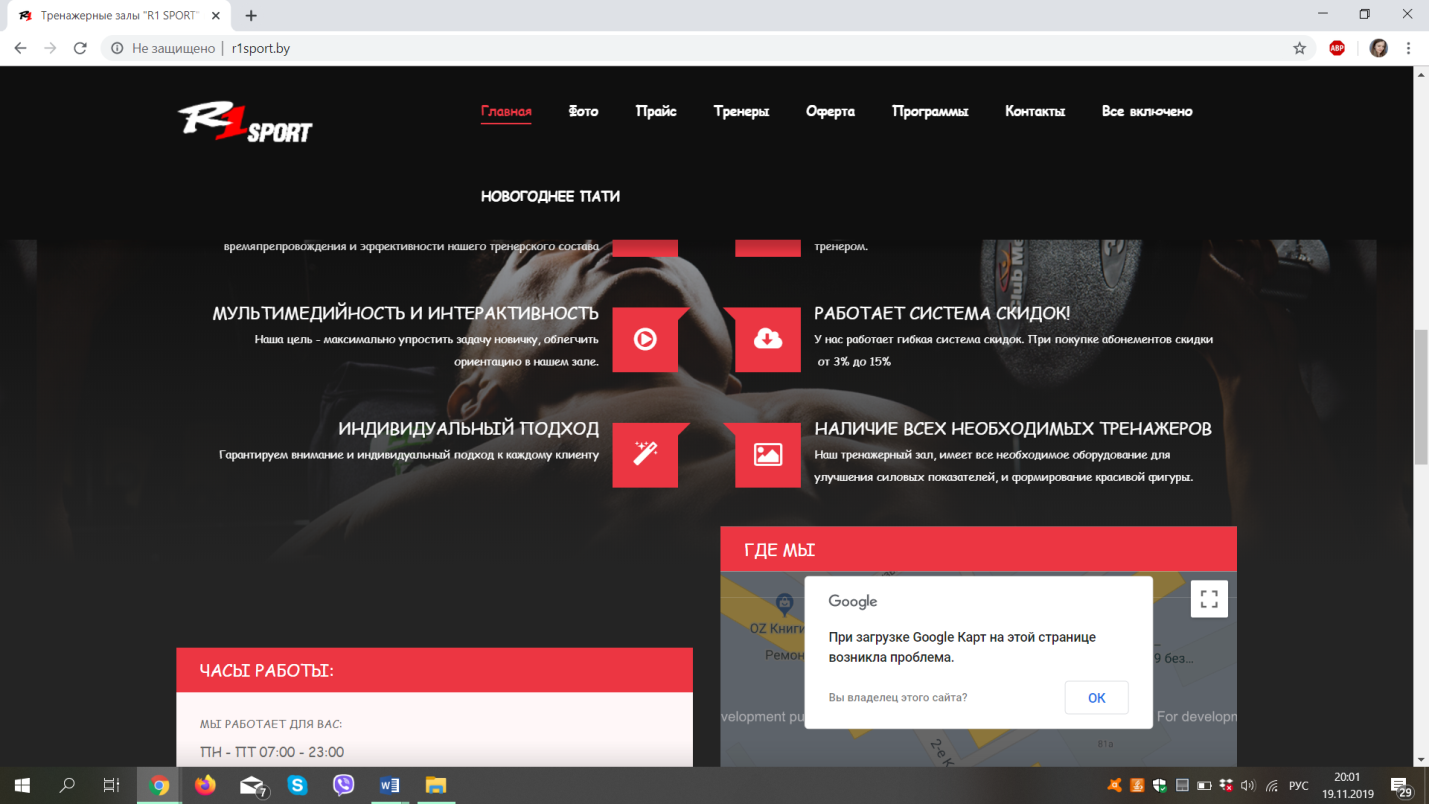


Рисунок 1.3– «R1SPORT»

Просмотрев и проанализировав 3 примера реализации сайтов тренажерных залов, можно сделать выводы:

* все рассмотренные мной программы обладают широким функционалом в области управления базой данных;
* необходимо не перегружать сайт излишними изображениями;
* стоит уделить внимание цветовым решениям и удобности чтения текста;
* удобно иметь возможность просмотреть свои абонементы и программы тренировок.

Стоит также отметить, что в настоящий момент все предлагаемые приложения находятся на достаточно высоком в технологическом отношении уровне.

# Разработка архитектуры проекта

Основные задачи разработки архитектуры проекта:

* Выделение программных подсистем и отображение на них внешних функций (заданных по внешнем описании) программного средства;
* Определение способов взаимодействия между выделенными программными подсистемами.

С учетом принимаемых на этом этапе решений производится дальнейшая конкретизация и функциональных спецификаций.

## **Обобщенная структура**

Курсовой проект построен на многоуровневой архитектуре (N-layer architecture)[10]. Данная архитектура основана на следующих принципах:

* Проектирование чётко устанавливает разграничение функций между уровнями;
* Нижние уровни независимы от верхних уровней;
* Верхние уровни вызывают функции нижних уровней, но при этом взаимодействуют только соседние уровни иерархии.

Для данного курсового проекта были разработаны следующие уровни:

* Уровень «Модель» – описание таблиц в базе данных;
* Уровень «Представление» – отвечает за отображение данных модели пользователю, реагируя на изменения модели;
* Уровень «Контроллер» – описание классов, методы которых предназначены для создания ответа пользователю по его запросу.
* Уровень «Авторизация» - назначение пользователю прав и привилегий для работы с объектами базы данных.
* Уровень «Аутентификация» - проверка подлинности пользователя.

Каждый запрос клиента проходит как минимум через 3 уровня («Аутентификация», «Авторизация» и «Контроллер»), после чего клиент получает ответ. В случае успешной аутентификации и авторизации клиента сервер также может задействовать другие уровни приложения.

В приложении используется паттерн MVC:

Паттерн MVC является одним из распространенных паттернов, применяемых в веб-приложениях. В том числе он применяется и в приложениях на Node.js.

Паттерн MVC включает ряд компонентов:

* Модели определяют структуру и логику используемых данных
* Представления (views) определяет визуальную часть, как данные будут отображаться
* Контроллеры обрабатывают входящие http-запросы, используя для обработки модели и представления, и отправляет в ответ клиенту некоторый результат обработки, нередко в виде html-кода.
* Система маршрутизация как дополнительный компонент сопоставляет запросы с маршрутами и выбирает для обработки запросов определенный контроллер.

В общем случае, когда к приложению приходит запрос, система маршрутизации выбирает нужный контроллер для обработки запроса. Контроллер обрабатывает запрос. В процессе обработки он может обращаться к данным через модели и для рендеринга ответа использовать представления. Результат обработки контроллера отправляется в ответ клиенту. Нередко ответ представляет html-страницу, которую пользователь видит в своем браузере.

## **Спецификация функциональных требований**

На рисунке 2.2.1 показана UML-диаграмма, наглядно представляющая возможности данного ПС с точки зрения незарегистрированного пользователя.

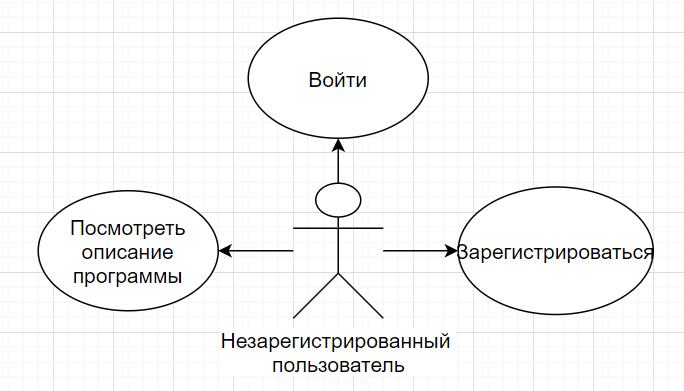


Рисунок 2.2.1–UMLдиаграмма вариантов использования для незарегистрированного пользователя

Данная диаграмма показывает предоставляемый функционал для незарегистрированного пользователя, который может зарегистрироваться, войти в свой аккаунт или же посмотреть информацию о программе.

На рисунке 2.2.2 показана UML-диаграмма, наглядно представляющая возможности данного ПС с точки зрения вошедшего под своим аккаунтом пользователя.

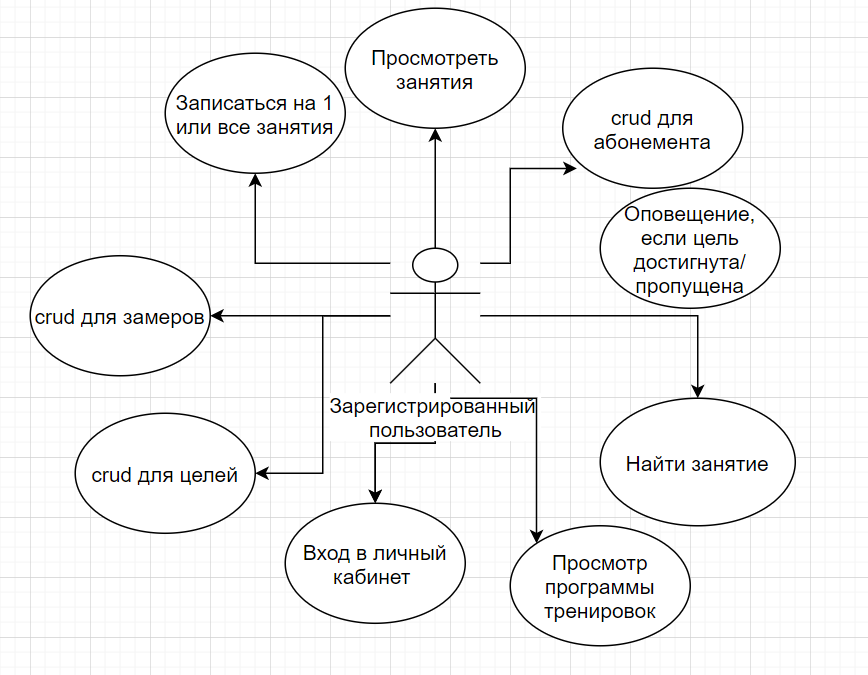


Рисунок 2.2.2–UMLдиаграмма вариантов использования для вошедшего пользователя

Данная диаграмма показывает предоставляемый функционал для вошедшего пользователя, в возможности которого входят редактирование профиля, заказ абонемента, просмотр занятия, работа с абонементом, оповещение, если цель достигнута или пропущена, поиск занятия, просмотр программы тренировок, работа с личными целями, замеры, а также запись на занятия. Регистрация / Отмена регистрации только от сессий класса, которые находятся в будущем. Прошлые сессии класса показаны как недоступные.

На рисунке 2.2.3 показана UML-диаграмма, наглядно представляющая возможности данного ПС с точки зрения вошедшего под своим аккаунтом тренера.

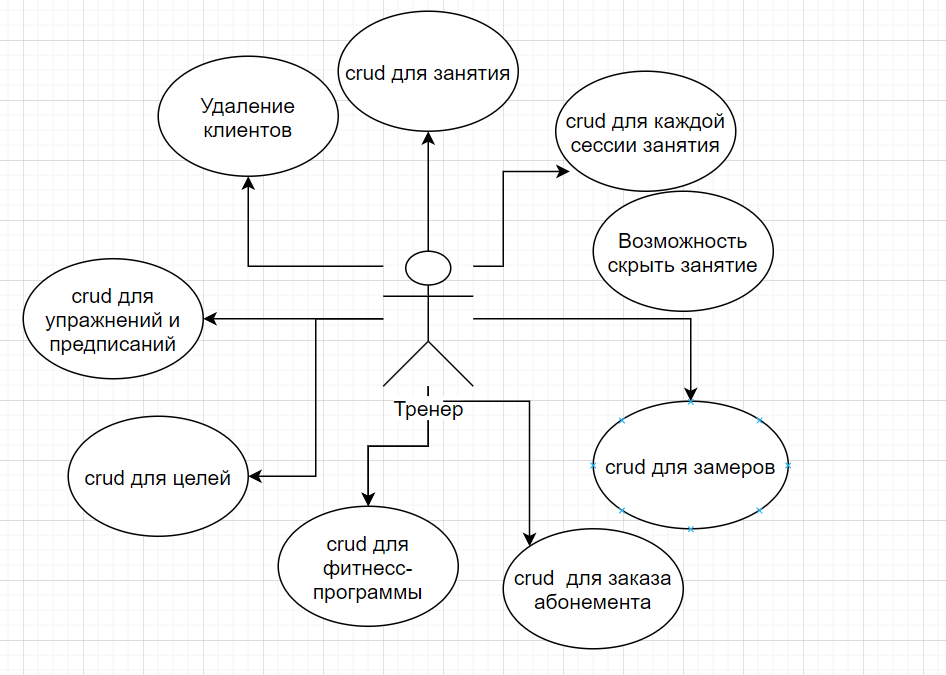


Рисунок 2.2.3–UMLдиаграмма вариантов использования для вошедшего тренера

Данная диаграмма показывает предоставляемый функционал для вошедшего тренера, в возможности которого входят работа с занятиями, изменение и работа с каждой сессией занятия, возможность скрыть занятие от клиента, работа с замерами клиента, работа с абонементом, работа с фитнесс-программой, с целями, упражнениями, а также удаление и просмотр клиентов.

# 2.3 Реализация консультанта на сокетах

Для создания консультант-бота были использованы веб-сокеты, которые в node js реализованы с помощью библиотеки библиотека ws. Бот может генерировать ответы на стандартные фразы клиента. Для этого на серверной части был создан объект серверного сокета, работающего на порту, отличном от порта основного сервера. Серверный сокет прослушивает канал и реагирует на получение сообщений от клиента. Листинг серверного сокета представлен в Приложении А.

На стороне клиента также создается клиентский сокет, который описан в тегах <script></script> внутри разметки сайта веб-приложения. Клиентский сокет создается при нажатии на кнопку «Send», т. е. при отправке сообщения. Листинг клиентского сокета представлен в Приложении Б.

1. **Разработка** **программной системы**

## **Структура проекта**

Структура данного программного средства отвечает модели MVC и отображена на рисунке 3.1.1

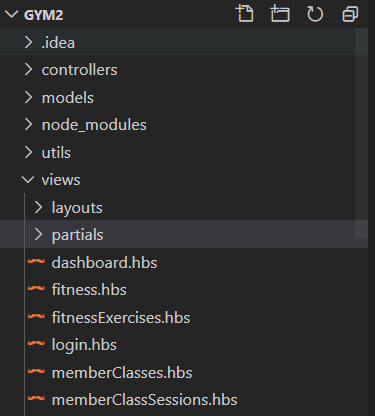


Рисунок 3.1.1–Структура проекта

В проекте Gym содержатся файлы исходного кода, формирующие страницы, некоторые ресурсы и классы.

Рассмотрим основные директории проекта:

* В пакете «controllers» содержатся все контроллеры и хелперы, в них описана основная логика предложения, более подробное содержание директории представлено на рисунке 3.1.2.

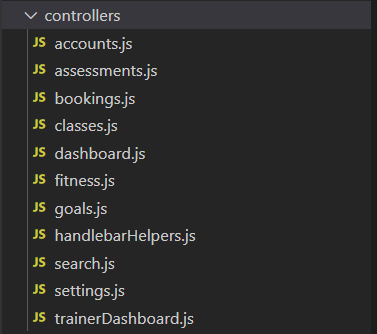


Рисунок 3.1.2 – структура директории «controllers»

Файл accounts.js содержит методы авторизации и аутентификации, assessments.js включает в себе методы по работе с замерами клиента, а booking.js отвечает за добавление, обновление, получение и удаление абонемента. Classes.js включают crud-операции для занятий клиента. Dashboard.js и trainerDashboard.js отвечает за взаимодействие всех вышеописанных классов и рендеринг в соответствии с текущим пользователем, fitness.js предоставляет функционал для работы в фитнесс-программой, а goals.js- для работы с целями пользователя. Класс handlebatHelpers.js позволяет уплавлять уведомлениями пользователю, а search.js осуществляет логику поиска занятий и сортировку. В settings.js можно найти методы для работы с информацией о пользователе.

* Директория models содержит сущности базы данных, в том числе класс json-store.js, содержащий основные методы работы с бд. Рисунок 3.1.3 раскрывает содержимое данной директории.

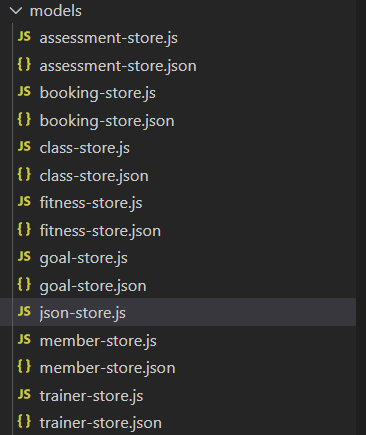


Рисунок 3.1.3 – содержимое папки models

Данное программное средство имеет 7 моделей, для каждой из которых был разработан дополнительный слой работы с данными.

* Пакет node-modules содержит скачанные модули, необходимые для корректной работы приложения
* Папка utils содержит утильные классы, которые дополняют функционал программного средства. Например, сортировка результатов поиска, вывод статистики, конверторы веса для различных систем измерения, анализ данных о клиенте
* Директория views предоставляет работу с уровнем представления модели MVC. В нем содержатся стандратные шаблоны для страниц, в файле main.hbs хранится основная разметка, в папке partials представлены шаблоны, которые впоследствии будет внедрены в другие шаблоны hbs
* Файл package.json необходим для безболезненного переноса приложения с устройства на устройство и содержит все зависимости данного веб-приложения.
* Файл routes.js содержит все существующие в данном веб-приложении маршруты, которыми может воспользоваться пользователь

## **Разработка модели данных**

В данном веб-приложении используется небольшая база данных, основанная на json – lowdb, разработанная lllodash. Предпочтение было отдано этой базе данный, поскольку она не требует инсталляциии, она бесплатная для разработки и требует лишь знания методов работы со списком и json.

На рисунке 3.2.1 представлена структура сущности assessment, которая содержит информацию о пользовательских замерах.



Рисунок 3.2.1 – assessments.json

Список включает поля:

* userid –идентификатор пользователя, ссылка на него;
* id – идентификатор замера;
* date – дата замера;
* weight – вес пользователя;
* chest – обхват груди пользователя;
* thigh – обхват бедер пользователя;
* upperArm – бицепс пользователя;
* waist – талия пользователя;
* trend– присутствие прогресса;
* comment – комментарий

На рисунке 3.2.2 представлена структура booking, которая содержит информацию для осуществления заказа.

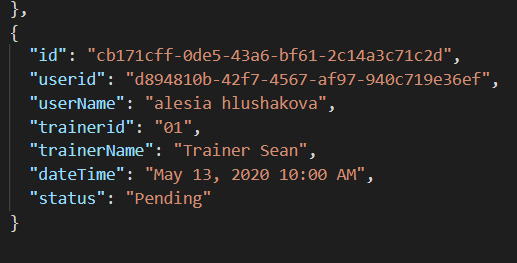


Рисунок 3.2.2 – booking.json

Список включает поля:

* id – идентификатор заказа;
* userid– идентификатор пользователя;
* userName – никнейм пользователя;
* trainerid – идентификатор тренера;
* trainerName – никнейм тренера;
* dateTime – время заказа;
* status – статус заказа абонемента.

На рисунке 3.2.3 представлена структура списка classes, которая содержит информацию о занятиях и времени их проведения.

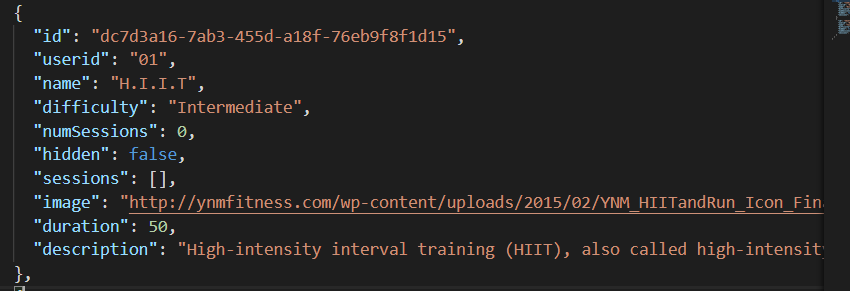


Рисунок 3.2.3 – classes.json

Список включает поля:

* id – идентификатор занятия;
* userid– идентификатор пользователя;
* name – название занятий
* difficulty – сложность занятия;
* numSessions – количество подходов в занятии;
* hidden – показывает, скрыто ли от пользователя занятие;
* sessions­ – массив подходов для занятий, описан в рисунке 3.2.3

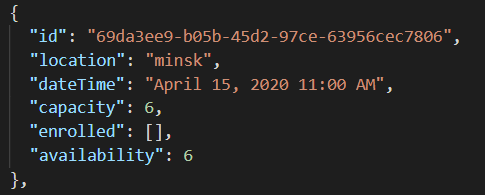


Рисунок 3.2.3 – структура подхода у занятия

* id – идентификатор подхода;
* location – место проведения;
* dateTime – время проведения подхода;
* capacity – вместимость занятия;
* enrolled– список идентификаторов записавшихся на подход пользователей;
* availability– количество оставшийся мест.

На рисунке 3.2.4 представлена структура списка fitness, которая содержит информацию о фитнесс-программах.

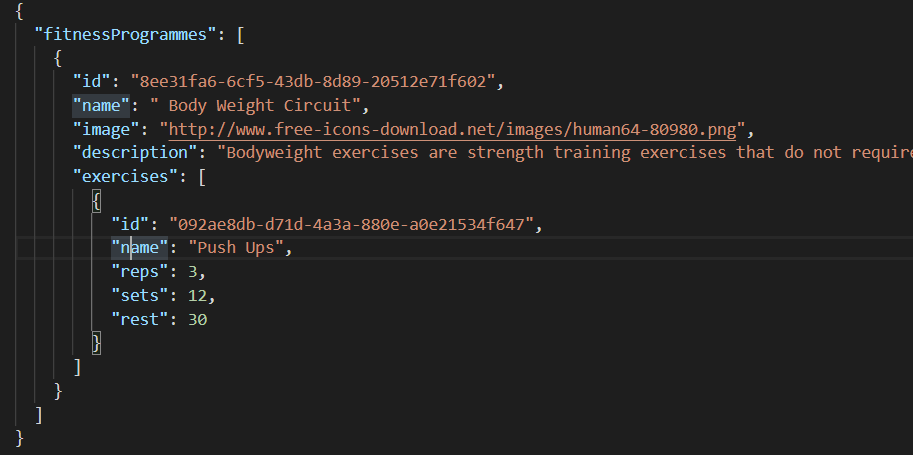


Рисунок 3.2.4 – список fitness.json

Список включает поля:

* id–идентификатор программы;
* name –название программы;
* description – описание программы;
* exercises – список упражнений;
* id– идентификатор упражнения;
* name – название упражнения;
* reps – количество повторов;
* sets – количество подходов;
* rest – время отдыха.

На рисунке 3.2.5 представлена структура списка goals, которая содержит цели клиента.

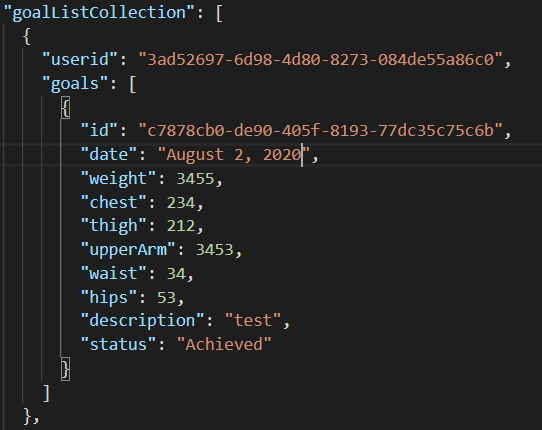


Рисунок 3.2.5 – список goals.json

Список включает поля:

* userid – идентификатор пользователя;
* id – идентификатор цели;
* date – дата достижения цели;
* weight – ожидаемый вес;
* сhest – ожидаемый обхват груди;
* thigh – обхват бедер пользователя;
* upperArm – бицепс пользователя;
* waist – талия пользователя;
* description – описание цели;
* status – статус цели.

На рисунке 3.2.6 представлена структура списка members, которая содержит список пользователей.

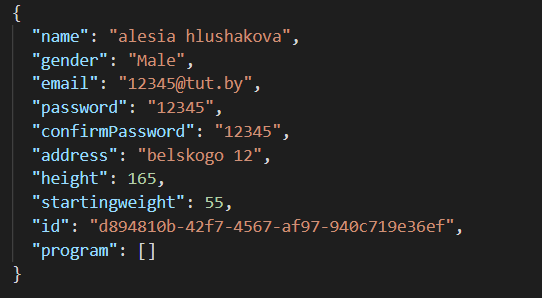


Рисунок 3.2.6 – members.json

Список включает поля:

* name – никнейм пользователя;
* gender – пол пользователя;
* email – почта пользователя;
* password – пароль пользователя;
* confirnPassword – подтвержденный пароль;
* address – адрес пользователя;
* Height – рост пользователя;
* startingWeight – вес пользователя;
* Id – идентификатр пользователя, guid
* program – список программ пользователя.

1. **Руководство пользователя**

Данная глава содержит описание некоторых функций приложения для более легкого восприятия конечного пользователя.

* 1. **Регистрация пользователя**

На стартовой станице пользователь должен зарегистрироваться, затем войти в свой аккаунт, тем самым получая доступ к одной из страниц.

Для регистрации необходимо заполнить форму, которая продемонстрирована на рисунке 4.1.

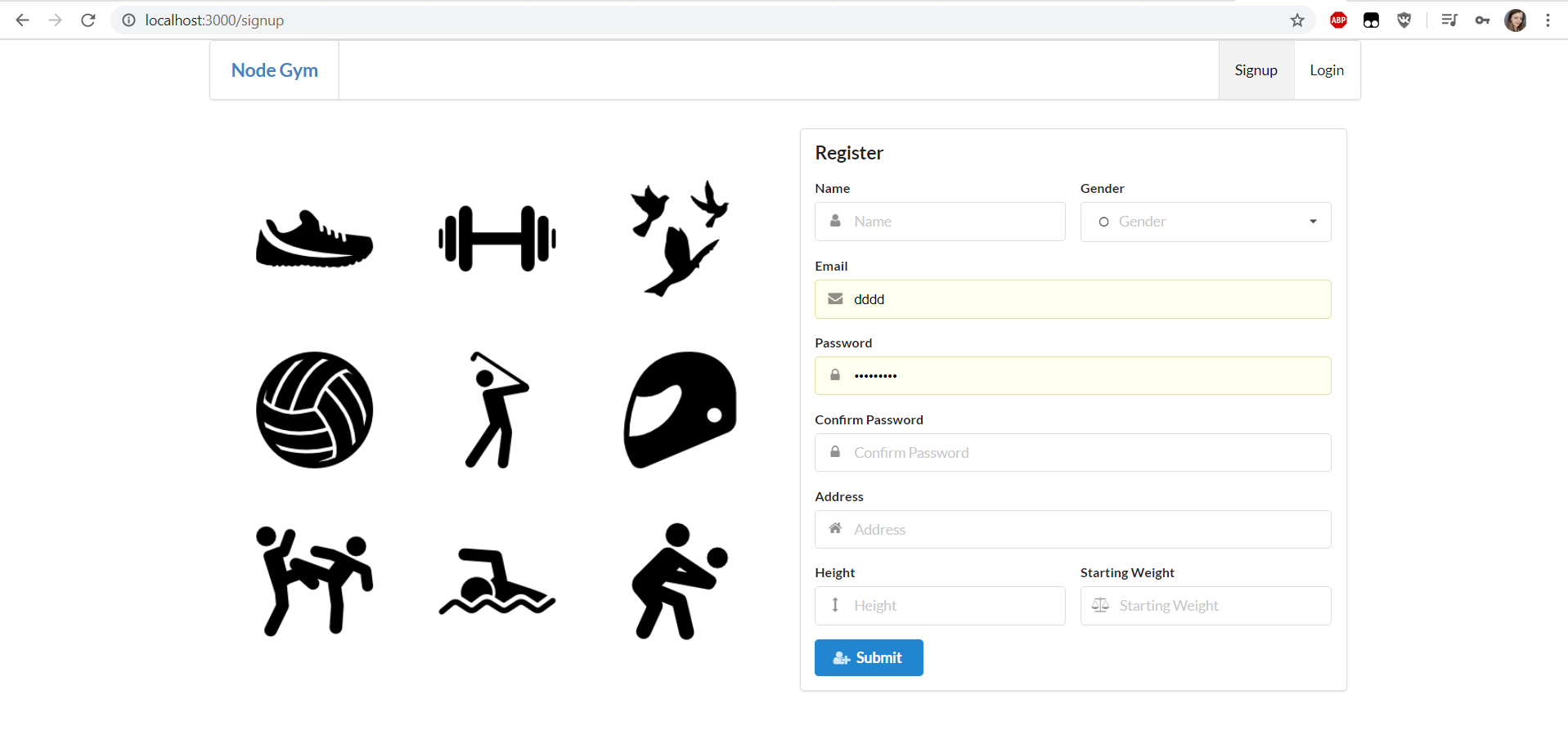


Рисунок 4.1 – Форма регистрации

После отправки пользователем запроса на регистрацию, происходит валидация всех полей формы. Если валидация будет неуспешной, пользователь увидит сообщения об ошибках в соответствующих полях для ввода информации. В случае успешной регистрации пользователь переадресовывается на страницу входа в приложение.

* 1. **Аутентификация и авторизация пользователя**

Для аутентификации необходимо ввести пароль, логин и нажать на кнопку «Войти», что позволит проверить, существует ли пользователь с введенными данными в системе. Все поля являются обязательными для заполнения.

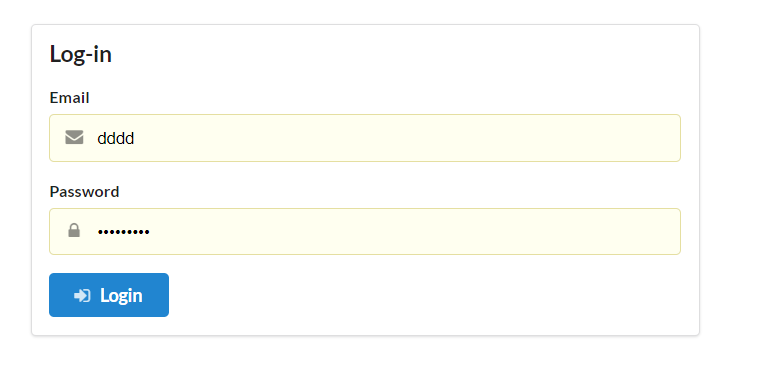
Если аутентификация прошла успешно и соблюдены все выше описанные правила, то происходит перенаправление на главную страницу, что означает, что пользователь был успешно авторизован. Скриншот формы авторизации представлен на рисунке 4.2.

Рисунок 4.2 – Фрагмент экрана с формой для авторизации пользователя

* 1. **Клиент**

После успешной авторизации клиента перенаправляет на главную страницу, отображенную на рисунке 4.3.

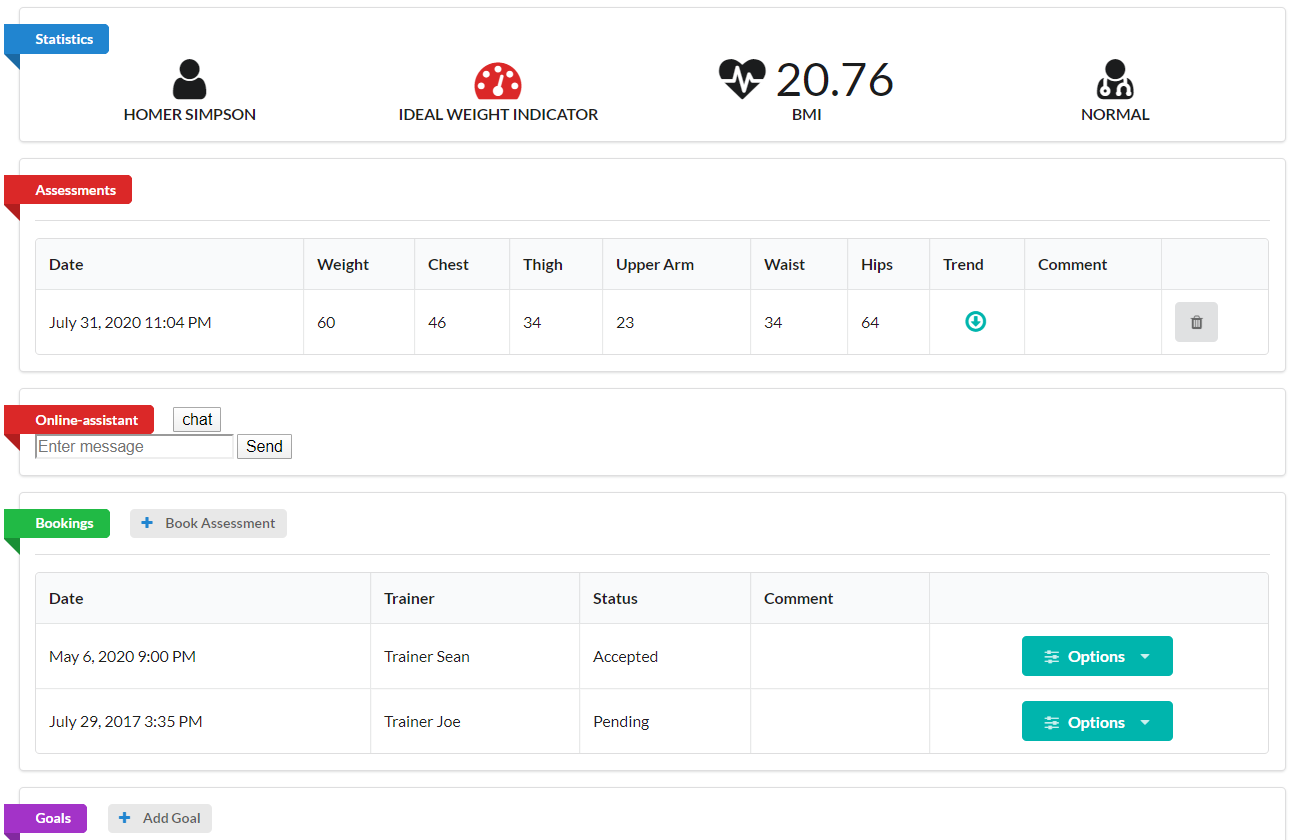


Рисунок 4.3 – Главная страница клиента

В графе статистика отображена статистика и замеры клиента.

Для добавления, изменения или удаления абонемента, замера или цели необходимо нажать кнопку + напротив вышеописанной графы. Для того, чтобы заказ абонемента считался принятым, необходимо, чтобы тренер изменил статус абонемента на оплаченный.

На рисунке 4.4 представлены занятия, на которые клиент может записаться.

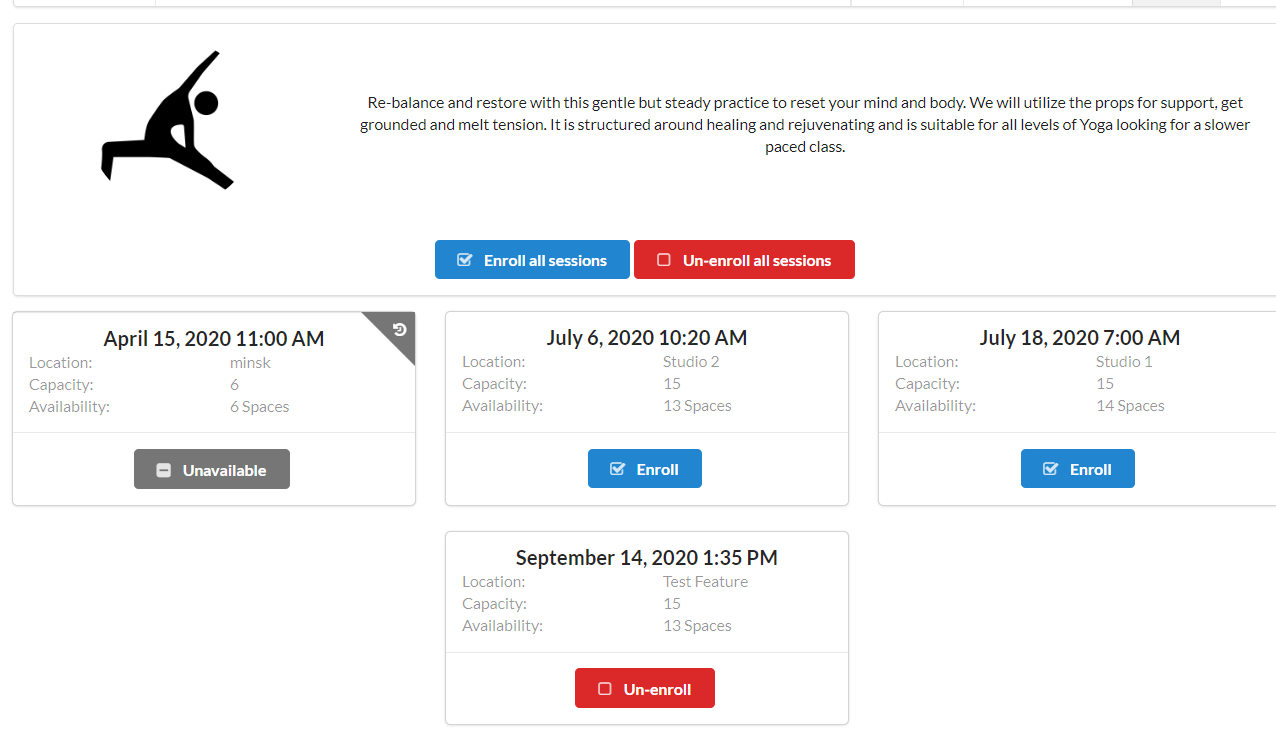


Рисунок 4.4 – Занятия клиента

Нажав на кнопку enroll/unenroll можно записаться или отказаться от занятия.

* 1. **Тренер**

При входе на сайт в качестве тренера пользователь перенаправляется на главную страницу, показанную на рисунке 4.5.

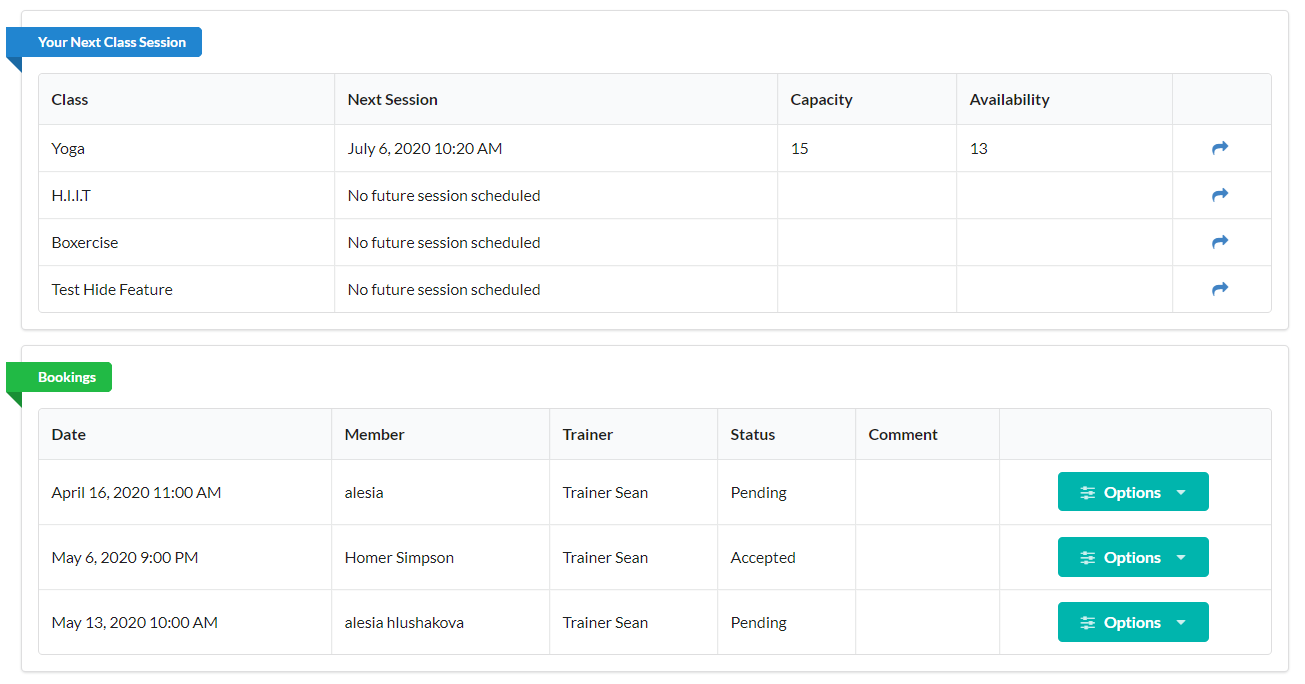


Рисунок 4.5 – Главная страница тренера

На данной странице отображены занятия, на которые записан тренер, а также назначенные тренировки. При нажатии кнопки Options, тренер может изменить статус абонемента, удалить его или изменить.

Открыв вкладку Classes, тренер может изменить занятия, создать новое занятие и подход к нему, как это изображено на рисунке 4.6.

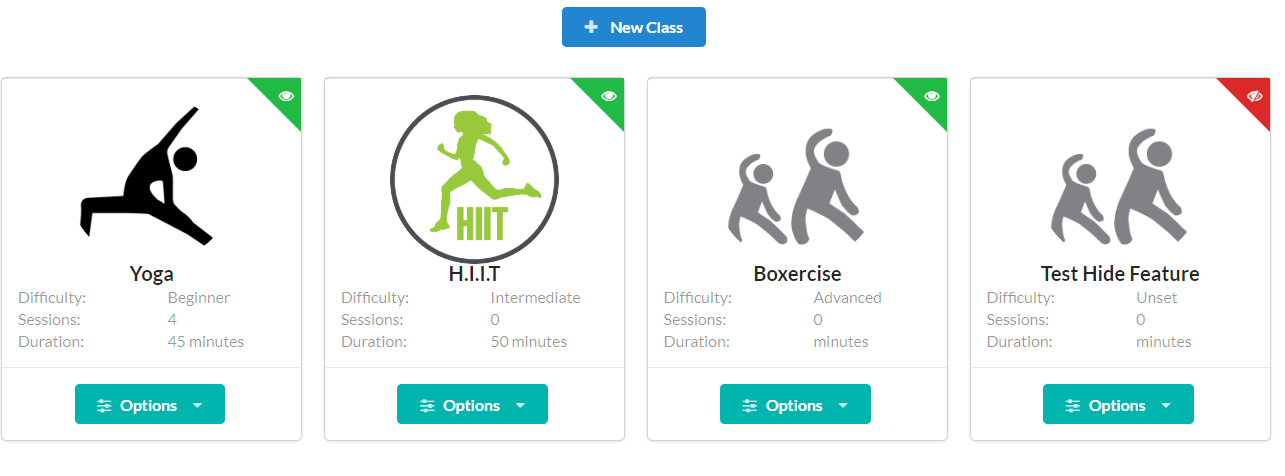


Рисунок 4.6 – Создание нового занятия

Аналогично на вкладке Fitness Programme тренер может проделать все те же операции с программой тренировок.

А если тренер перейдет по кладке Members, он может посмотреть своих клиентов, а также удалить их и осуществить поиск по ним, как это представлено на рисунке 4.7.

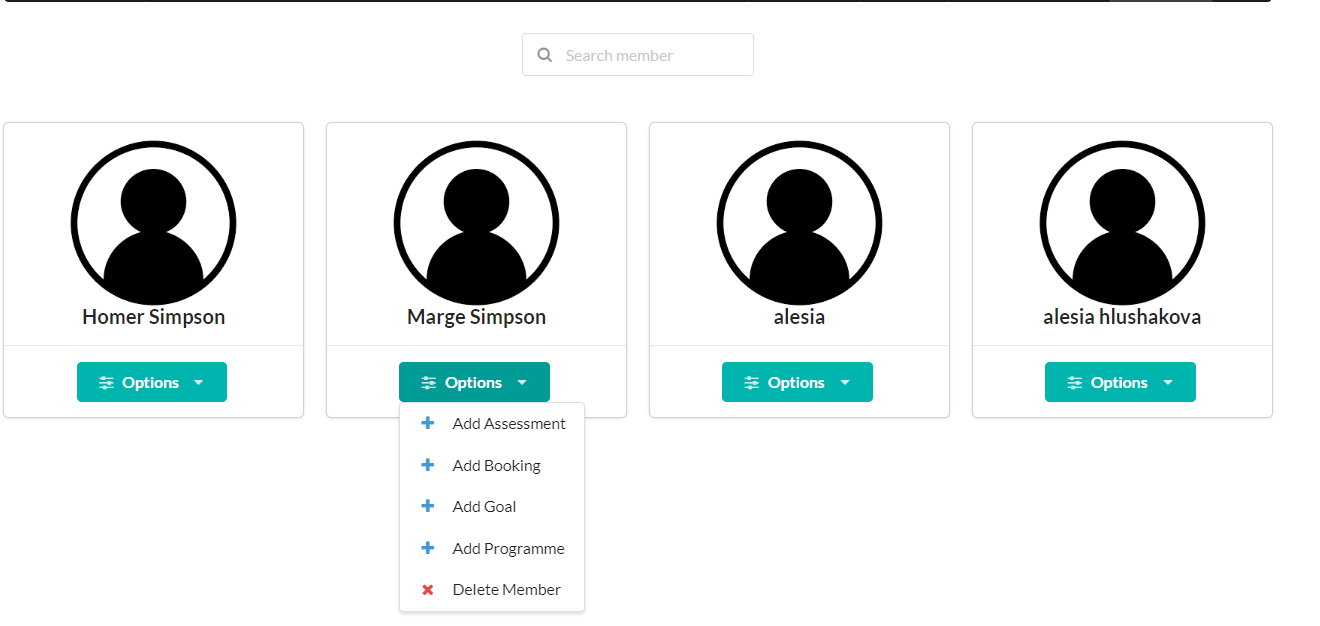


Рисунок 4.7 – Клиенты тренера

# 5 Тестирование

В данном курсовом проекте организована валидация полей на случай неочевидного или неправильного поведения пользователя.

Валидация приложения – это один из основных гарантов надежности приложения и всякий программист должен предусмотреть и предотвратить непредвиденное поведение пользователя.

Пример валидации вводимых данных при регистрации пользователя представлен на рисунке 5.1.

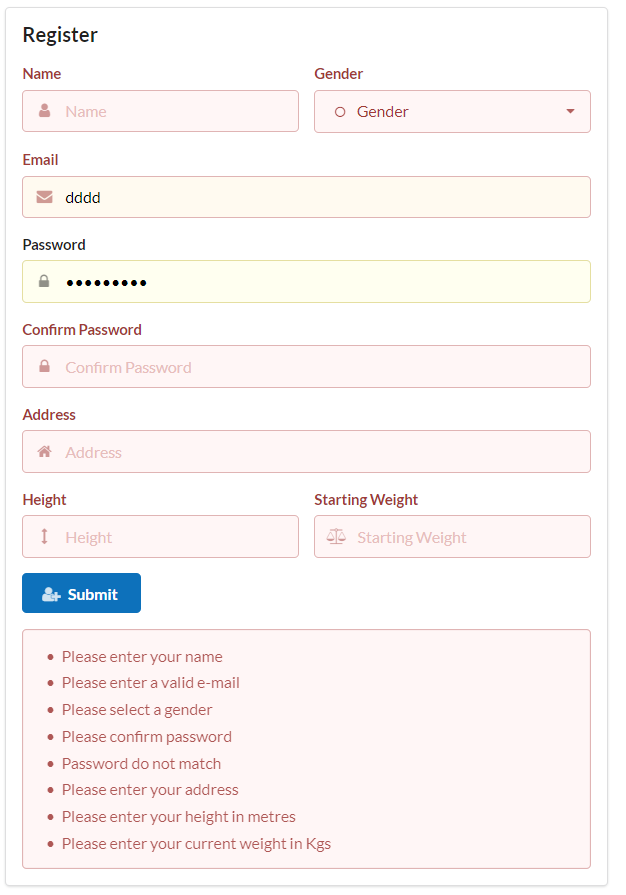


Рисунок 5.1 – Валидация формы регистрации

Валидация дат представлена функционалом ui calendar, который просто не позволит пользователю ввести неверную дату. Внешний вид календаря представлен на рисунке 5.2.

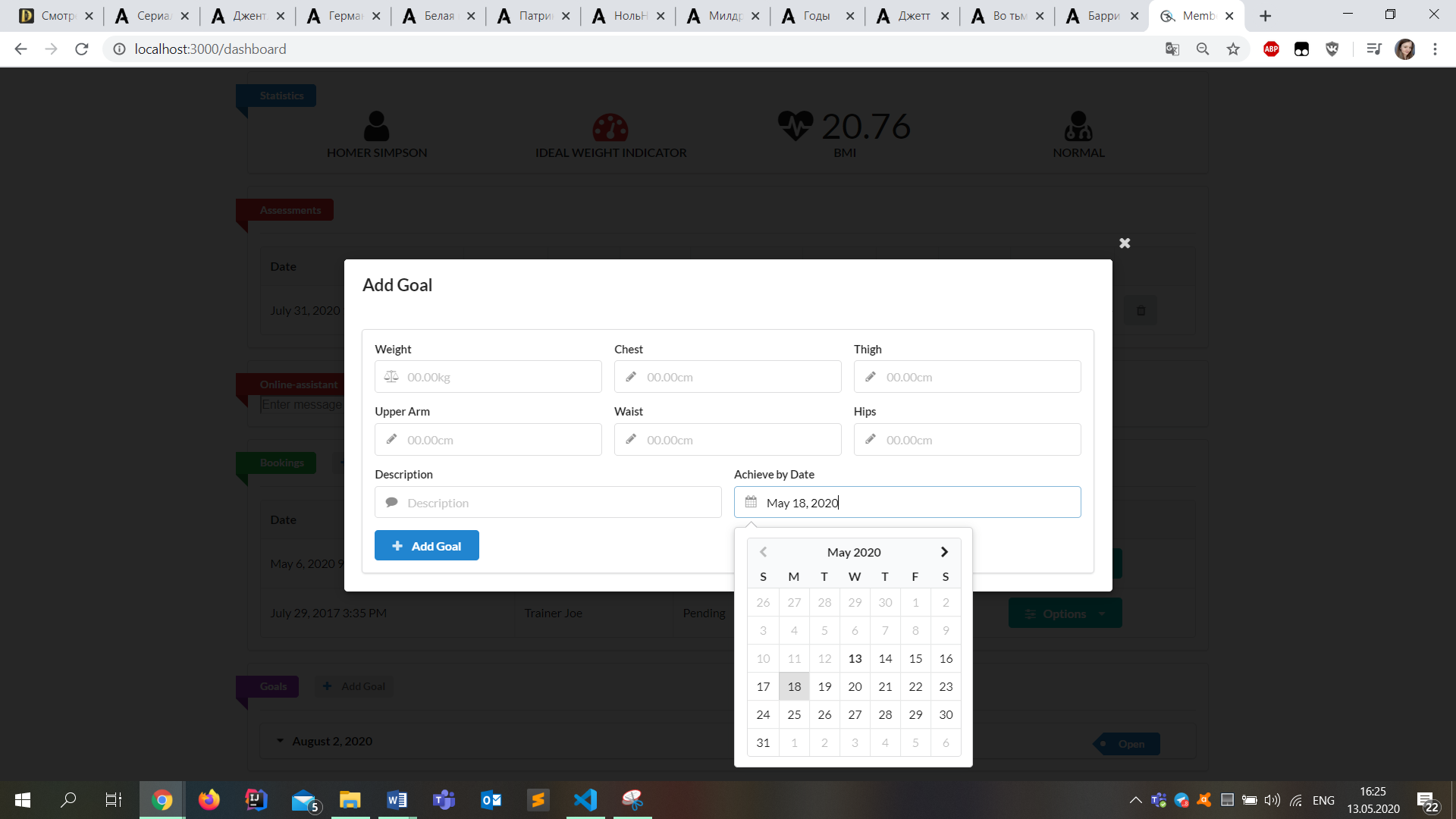


Рисунок 5.2 – Вид календаря

Если же срок проведения мероприятия, будь то тренировка или занятие прошел, то система заблокирует нажатие на данное событие и оповестит пользователя, что запись более не актуальна, как это показано на рисунке 5.3.

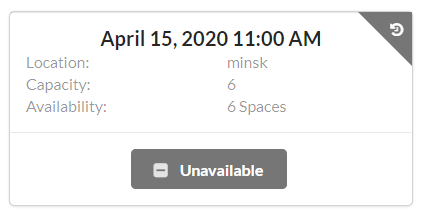


Рисунок 5.5 – Пропущенное занятие

Дабы избежать ошибок пользователя и падения всего приложения, были предусмотрены некоторые правила, которые пользователь не может обойти, а в коде были использованы регулярные выражения. А в случае ошибки, выдается окно с сообщением, которое информирует пользователя об ошибке.

Предусмотрено, что пользователь не ввел одно из полей, тогда программное средство напомнит ему об этом и не даст завершить операцию, пока пользователь не исправит все свои ошибки и не заполнит все поля.

# Заключение

В самом начале была поставлена задача: реализовать некое программное средство с использованием базы данных и паттерна MVC, которое в будущем получило название «Тренажерный зал». Перед началом были изучены некоторые аспекты WebSockets и Javascript, которые и позволили реализовать то, что было задумано изначально.

В процессе решения поставленной задачи была достигнута поставленная цель по созданию программного средства «Тренажерный зал». При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения. Таким образом, была достигнута цель и создано веб-приложение.

В программном средстве были реализованы следующие функции:

* отображать информацию базы данных;
* работать и редактировать (добавлять, удалять, изменять) параметры базы данных: пользователи, упражнения, комплексы, программы тренировок;
* осуществлять поиск по клиентам;
* иметь 2 роли с определенной функциональностью;
* просмотр тренировок, упражнений.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список литературы

1. Сайт тренажерного зала GYM24 –<https://www.gym24.by/>

2. Сайт тренажерного зала ARGYMENT –<https://www.argyment.by/>

3.Сайт тренажерного зала R1SPORT – <https://www.r1sport.by/>

4. Сайт node js – <https://www.npmjs.com/>

# Приложение А

|  |
| --- |
| const wss = new WebSocket.Server({port: 3001});  const initAssistantMessage = 'Здравствуйте, меня зовут Алеся! Я online-консультант. Нужна ли Вам помощь?';  const acceptMessage = 'помощь!';  const refuseMessage = 'Без проблем. Приятных покупок!';  const invalidOptionMessage = 'Я вас не понимаю:( Напишите да/нет.';  wss.on('connection', function connection(ws) {    console.log('1111');      ws.send(initAssistantMessage);      ws.on('message', function(message) {        console.log('((');          message = message.toLowerCase();          let answer = invalidOptionMessage;          if (message === 'да') {              answer = acceptMessage;          } else if (message === 'нет') {              answer = refuseMessage;          }          ws.send(answer);      });  }); |

# Приложение Б

|  |
| --- |
| let chatOpened = false;      let assistantSocket;  function createNewMessage(sender, message)   {      const messageContainer = $('.messages');      messageContainer.append(`          <div class="alert  ${sender}-message">              ${message}          </div>      `);    }      function onlineassistant ()      {          if (!chatOpened)           {                chatOpened = true;              assistantSocket = new WebSocket('ws://localhost:3001/');              assistantSocket.onmessage = function (event)               {                  createNewMessage('agent', event.data);              };              assistantSocket.onopen = function()              {                    $('.send-message-button').click(function ()                   {                        const newMessageInput = $('.new-message');                      const message = newMessageInput.val();                      newMessageInput.val('');                      assistantSocket.send(message);                      createNewMessage('client', message);                  });              };          }      };    </script> |