Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Системный анализ и проектирование информационных систем (с разделом веб разработка)»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Старший преподаватель кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. Н. Салапура |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ ФИТНЕС-ЦЕНТРА»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 172303  КОВАЛЕВИЧ Артём Эдуардович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc132707480)

[1 Анализ системы организации и работы фитнес-центра 7](#_Toc132707481)

[1.1 Описание системы работы фитнес-центра 7](#_Toc132707482)

[1.2 Разработка функциональной модели предметной области 7](#_Toc132707483)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 13](#_Toc132707484)

[1.4 Разработка информационной модели предметной области 13](#_Toc132707485)

[1.5 UML-модели представления программного средства и их описание 15](#_Toc132707486)

[2 Проектирование и конструирование програмного средства 18](#_Toc132707487)

[2.1 Постановка задачи 18](#_Toc132707488)

[2.2 Обоснование компонентов и технологий для реализации программного средства 18](#_Toc132707489)

[2.3 Архитектурные решения 19](#_Toc132707490)

[2.4 Описание алгоритмов, реализующих ключевую бизнес логику разрабатываемого программного средства 20](#_Toc132707491)

[2.5 Проектирование пользовательского интерфейса 262.5 Проектирование пользовательского интерфейса 26](#_Toc132707492)

[2.6 Методы и средства используемые для обеспечения безопасности данных 27](#_Toc132707492)

[3 Тестирование и проверка работоспособности программы 29](#_Toc132707494)

[4 Инструкция по развертыванию приложения 31](#_Toc132707495)

[4.1 Авторизация 31](#_Toc132707496)

[4.2 Модуль администратора 31](#_Toc132707497)

[4.3 Модуль пользователя 34](#_Toc132707498)

[Заключение 36](#_Toc132707499)

[Список используемых источников 37](#_Toc132707500)

[Приложение А Листинг кода и алгоритмов реализующих основную бизнесс-логику приграммы 38](#_Toc132707501)

[Приложение Б Отчет о проверке на заимствование в системе «Антиплагиат» 46](#_Toc132707502)

# **ВВЕДЕНИЕ**

Фитнесс-индустрия является одной из наиболее динамично развивающихся отраслей, которая охватывает множество направлений и сегментов. В этом контексте, программное средство для автоматизации фитнесс центра является актуальной темой для исследования и разработки. Основными функциями данной системы являются управление графиком занятий, ведение базы данных клиентов, расчет и контроль финансовых показателей, а также другие операции, необходимые для эффективного управления фитнес-центром.

Преимущества автоматизации фитнес-центра очевидны: повышение эффективности управления, сокращение времени на рутинные операции, минимизация рисков ошибок и недостачи, улучшение качества обслуживания клиентов, удобство работы персонала.

Целями курсовой работы являются минимизация времени покупки услуг, уменьшение вероятности допуска ошибок, увеличение производительности сотрудников фитнес-центра.

Основные задачи курсовой работы:

– изучение предметной области;

– анализ процессов управления фитнес-центром;

– тестирование и проверка работоспособности.

Объект исследования: данной курсовой работы является программное средство для автоматизации фитнес-центра.

# **1 АНАЛИЗ СИСТЕМЫ ОРГАНИЗАЦИИ И РАБОТЫ ФИТНЕС-ЦЕНТРА**

* 1. **Описание системы работы фитнес-центра**

Фитнесс-центр – это спортивный клуб, который предлагает широкий спектр услуг для физического развития и улучшения здоровья человека. Система работы фитнесс-центра включает следующие этапы:

Регистрация и консультация. Первым шагом для начала занятий в фитнесс-центре является регистрация на сайте или приход в зал. После регистрации новому клиенту предоставляется консультация со специалистом по фитнесу, который поможет подобрать программу тренировок и даст рекомендации по питанию.

Тренировки. Фитнесс-центр предлагает широкий выбор тренировок: силовые, кардио, групповые занятия, йога и многое другое. Все тренировки проводятся под контролем опытных тренеров, которые помогут правильно выполнять упражнения и добиться максимального результата.

Расписание занятий. Фитнесс-центр предоставляет своим клиентам удобное онлайн-расписание занятий, где можно выбрать подходящий день и время для тренировки.

Система оплаты. Оплата услуг фитнесс-центра производится через сайт или приход в зал. Клиенты могут выбрать подходящий тарифный план, в зависимости от своих потребностей и финансовых возможностей.

Индивидуальный подход. Фитнесс-центр учитывает индивидуальные особенности каждого клиента и предлагает индивидуальные программы тренировок, которые помогут добиться наилучшего результата.

Система отзывов. Фитнесс-центр предоставляет своим клиентам возможность оставлять отзывы о проведенных тренировках и услугах центра. Это помогает улучшить качество работы фитнесс-центра и привлечь новых клиентов.

Система наград. Фитнесс-центр использует систему наград для стимулирования клиентов к достижению лучших результатов и постоянному совершенствованию. Клиенты могут получать бонусы и подарки за достижение определенных целей.

В целом, система работы фитнес-центра направлена на поддержание или улучшение здоровья и состояния клиента.

В ходе данного пункта можно выяснить устройство фитнес-центра и понять его особенности.

## **1.2 Разработка функциональной модели предметной области**

IDEF0 — методология функционального моделирования.

Описание системы с помощью IDEF0 называется функциональной моделью. Функциональная модель предназначена для описания существующих бизнес-процессов, в котором используются как естественный, так и графический языки. Для передачи информации о конкретной системе источником графического языка является сама методология IDEF0.

На рисунке 1.1 представлена контекстная диаграмма верхнего уровня.

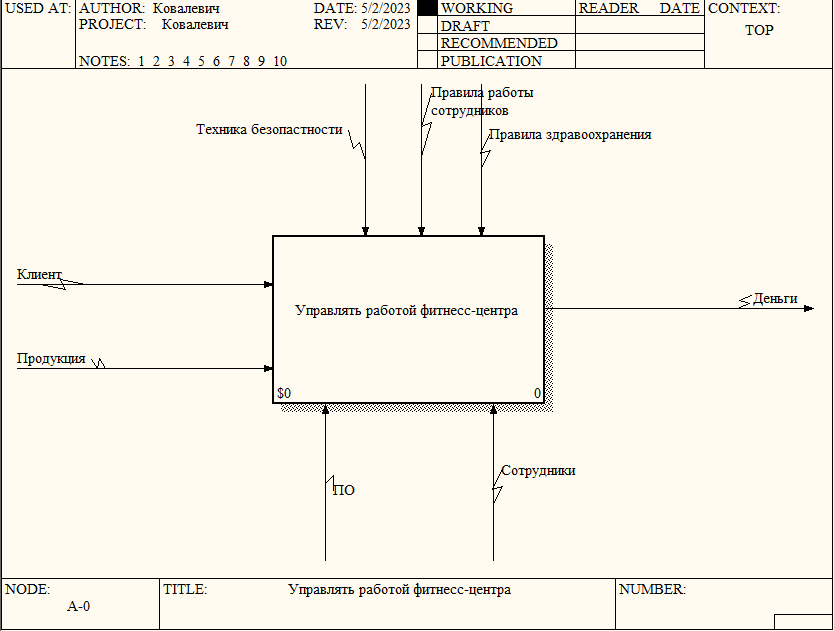


Рисунок 1.1 – Контекстная диаграмма процесса «Управлять работой фитнес-центра»

Декомпозиция контекстной диаграммы, показанная на рисунке 1.2, описывает путь управления фитнес-центром.

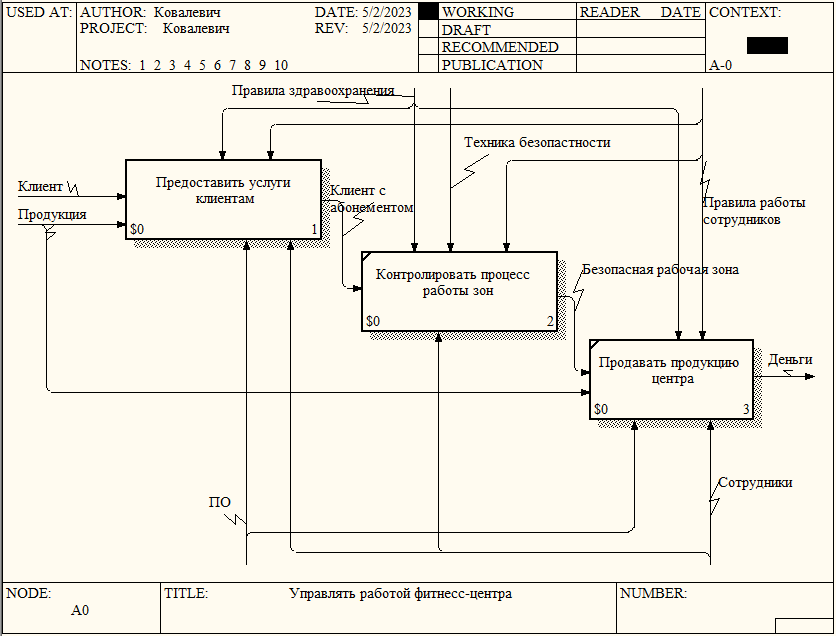


Рисунок 1.2 – Декомпозиция контекстной диаграммы верхнего уровня

Декомпозицию блока «Предоставить услуги клиентам» можно увидеть на рисунке 1.3.

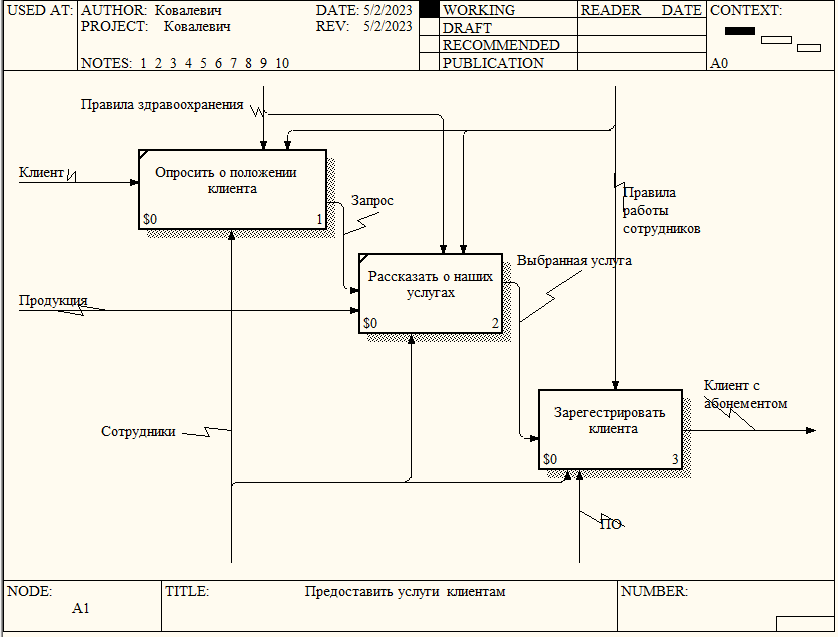


Рисунок 1.3 – Декомпозицию блока «Предоставить услуги клиентам»

На рисунке 1.4 можно увидеть декомпозицию блока «Зарегистрировать клиента».

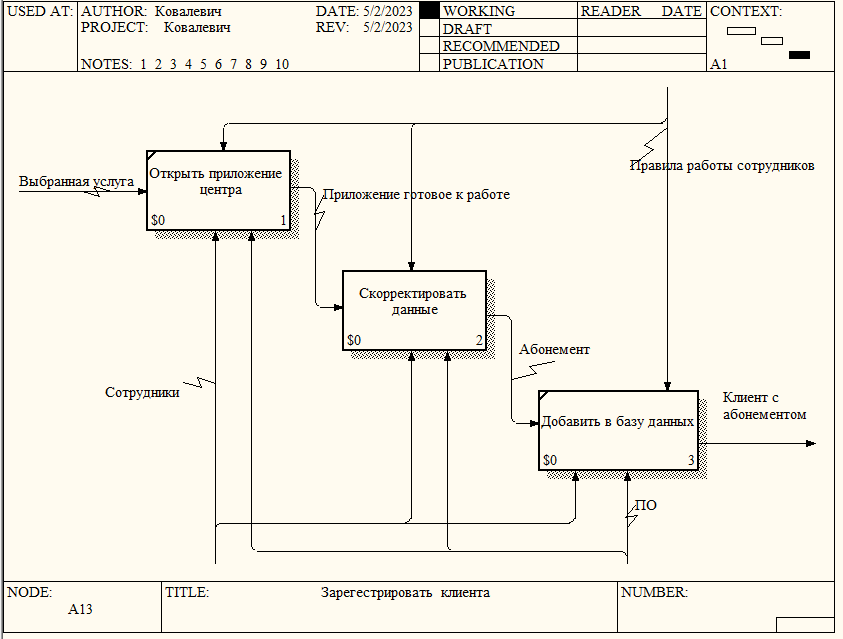


Рисунок 1.4 – Декомпозицию блока «Зарегистрировать клиента»

На рисунке 1.5 можно увидеть декомпозицию блока «Скорректировать данные».

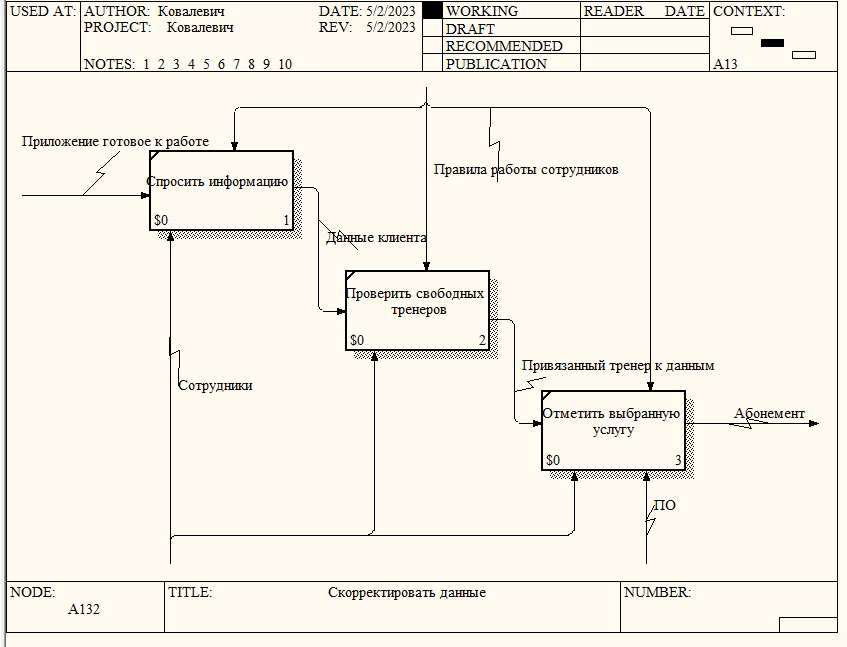


Рисунок 1.5 – Декомпозицию блока «Скорректировать данные»

На рисунке 1.6 показан процесс прохождения пути клиента от регистрации до конца абонемента.

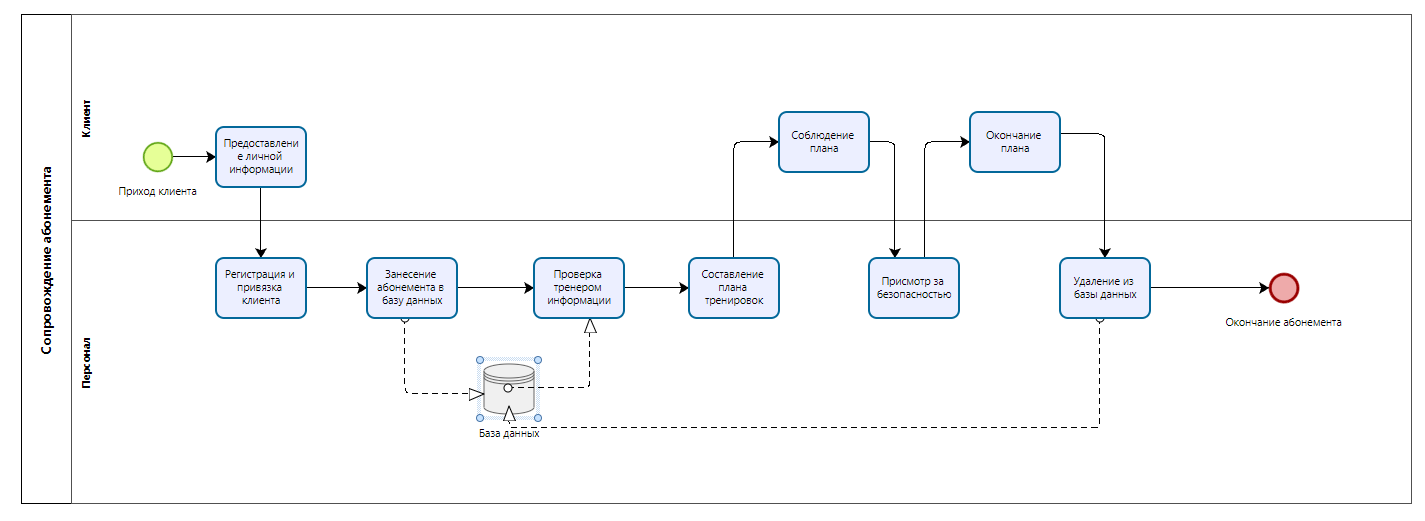


Рисунок 1.6 – BPMN-модель основного процесса предметной области

Мы построили функциональную модель управления работой фитнес-центра, что позволило нам узнать более подробную деятельность данной области.

## **1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Проанализировав ситуацию в фитнес-центре и выявив все минусы, постараемся создать такую систему, которая бы автоматизировала следующие операции:

-регистрация нового клиента с помощью информационной системы;

- управление расписанием занятий;

- контроль финансовых операций;

- разделение ролей;

- работа с данными.

Функциональные требования:

-авторизация пользователей с различными уровнями доступа: администратор, менеджер, тренер, клиент;

- работа с данными, включая добавление, редактирование и удаление клиентов;

- возможность поиска и фильтрации данных для быстрого доступа;

- уведомление пользователей о важных событиях;

- генерация отчетов о финансовых операциях.

Главное назначение автоматизированной системы в данном случае – повысить эффективность работы фитнес-центра, ускорить процессы и уменьшить ошибки в управлении клиентами, персоналом, финансами и инвентарем.

В результате мы получаем из данного пункта требования, которые должны быть реализованы в приложении.

**1.4 Разработка информационной модели автоматизированной системы управления работой фитнес-центра**

В данной работе используется база данных MongoDB. Первая коллекция «users» хранит данные о пользователях, описание структуры данных в таблице 1.1.

Таблица 1.1 – Описание структуры данных user

|  |  |  |
| --- | --- | --- |
| \_id | String | Идентификатор |
| login | String | Логин |
| password | String | Пароль |
| role | String | Роль |
| currentDate | String | Дата создания объекта |

Вторая коллекция «news» хранит данные об новостях клуба, описание представлено в таблице 1.2.

Таблица 1.2 - Описание структуры данных news

|  |  |  |
| --- | --- | --- |
| \_id | String | Идентификатор |
| text | String | Текст новости |
| author | String | Автор |
| tit | String | Заглавие новости |

Вторая коллекция «comment» хранит данные о комментариях, описание представлено в таблице 1.3.

Таблица 1.3 - Описание структуры данных comment

|  |  |  |
| --- | --- | --- |
| \_id | String | Идентификатор |
| text | String | Текст комментария |
| author | String | Автор |
| mark | String | Оценка |

Вторая коллекция «byu\_users» хранит данные о купленных абонементах, описание представлено в таблице 1.4.

Таблица 1.4 - Описание структуры данных buy\_users

|  |  |  |
| --- | --- | --- |
| \_id | String | Идентификатор |
| user\_login | String | Логин пользователя |
| to\_name | String | ФИО |
| Cupon\_name | String | Название купона |
| Time\_end | String | Дата конца абонемента |

Вторая коллекция «comment» хранит данные о комментариях, описание представлено в таблице 1.3.

Таблица 1.5 - Описание структуры данных cupons

|  |  |  |
| --- | --- | --- |
| \_id | String | Идентификатор |
| Name | String | Название купона |
| Price | Int | Цена |
| Text | String | Текст купона |
| Contin | Int | Продолжительность |

Вторая коллекция «comment» хранит данные о комментариях, описание представлено в таблице 1.3.

Таблица 1.6 - Описание структуры данных messages

|  |  |  |
| --- | --- | --- |
| \_id | String | Идентификатор |
| User | String | Название пользователя |
| Text | String | Текст сообщения |

Выше мы описали коллекции с нашими данными которые будут храниться в базе данных, что позволит более точно написать нашу программу.

## **1.5 UML-модели представления программного средства и их описание**

Диаграмма вариантов использования описывает: какой функционал разрабатываемой программной системы доступен каждой группе пользователей, позволяет чётко отделить систему от её окружения, определить действующих лиц, их взаимодействие с системой и ожидаемую функциональность системы. На рисунке 1.7 можно увидеть диаграмму вариантов использования нашей программы.

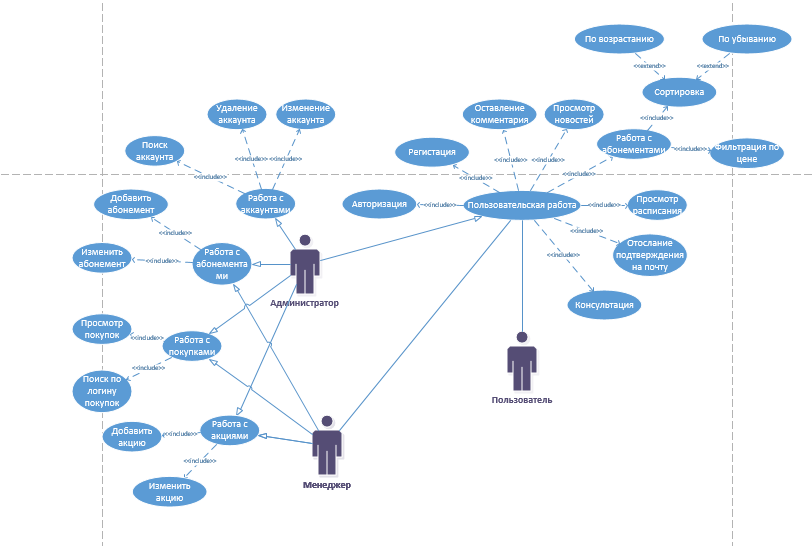


Рисунок 1.7 – Диаграмма вариантов использования

Диаграмма последовательности используются для изображения взаимодействия объектов, упорядоченные по времени их проявления. С помощь диаграммы мы получаем последовательность действий для каждого объекта, которые необходимы для достижения поставленной цели. На рисунке 1.8 можно увидеть диаграмму последовательности.

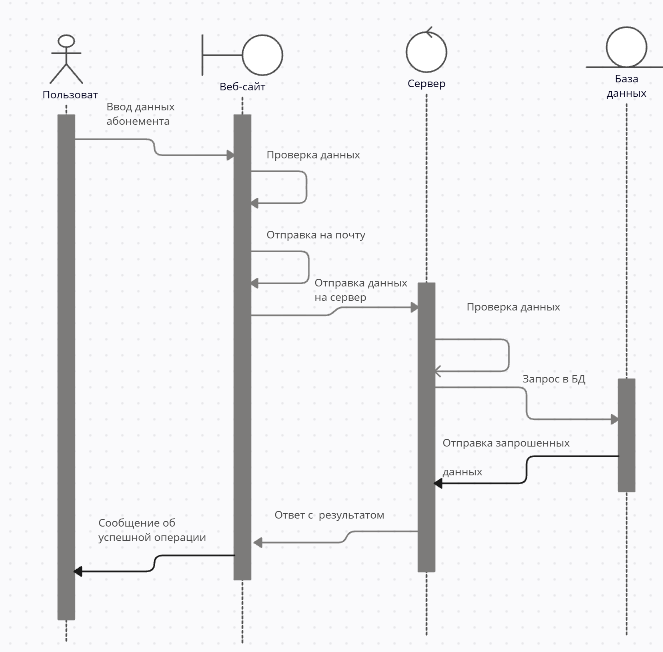


Рисунок 1.8 – Диаграмма последовательности процесса покупки абонемента

На рисунке 1.9 можно увидеть диаграмму UML Activity процесса отправки купона.

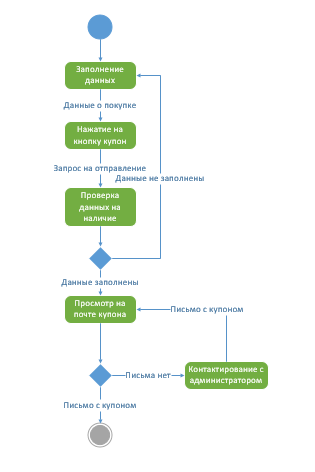


Рисунок 1.8 – Диаграмма активности процесса отправки купона

С помощью UML-диаграмм мы визуализировали наше программное средство, а также расписали пару процессов.

# **2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАМНОГО СРЕДСТВА**

**2.1 Постановка задачи**

Требуется разработать информационную систему работы фитнес-центра. В информационной системе реализовать авторизацию:

* Учетная запись администратора;
* Учетная запись менеджера;
* Учетная запись пользователя.

Требуется реализовать работу с данными, которая становится доступной только после прохождения авторизации. Данные хранятся в отдельном файле и имеют вид, описанный подробно в каждом варианте курсовой работы. Для работы с данными должны быть предусмотрены два функциональных модуля:

- модуль администратора;

- модуль менеджера;

- модуль пользователя.

Модуль администратора включает следующие возможности:

- просмотр всех данных;

- добавление новой записи;

- удаление записи;

- редактирование записи;

- работа с аккаунтами;

- поиск данных.

Модуль менеджера включает следующие возможности:

- просмотр всех данных;

- добавление новой записи;

- удаление записи;

- редактирование записи;

- поиск данных.

Модуль пользователя имеет следующими функциональные возможности:

- просмотр всех данных;

- просмотр и добавление оценки отзыва;

- поиск данных;

- просмотр по категориям

С помощью данного пункта можно понять какие будут доступны функции определенной роли.

**2.2 Обоснование компонентов и технологий для реализации программного средства**

Основные фреймворки, использованные в данной работе: JQuerry, Node js, express. Язык программирования JavaScript. Язык разметки HTML. Каскадная таблица стилей CSS. В качестве хранилища данных было использована база данных MongoDB. Данный компонент был выбран по следующим причинам:

Node.js - это серверная платформа для выполнения JavaScript-кода. Она позволяет выполнять код на стороне сервера и обрабатывать запросы от клиентов, обеспечивая быстрое и эффективное выполнение приложения.

Express.js - это фреймворк для Node.js, который упрощает разработку серверных приложений. Он обеспечивает маршрутизацию, обработку запросов, управление сеансами и многое другое, что ускоряет процесс разработки.

Реляционная база данных (РБД) и нереляционная база данных (NoSQL) представляют различные модели организации и хранения данных. Отличия между ними можно кратко описать следующим образом:

Структура данных: В реляционных базах данных данные организованы в виде таблиц с рядами и столбцами, где существуют жестко определенные связи между таблицами. В нереляционных базах данных данные могут быть организованы в виде различных структур, таких как документы, графы или столбцы.

Гибкость схемы: В РБД схема базы данных предварительно определена и строго структурирована. Нереляционные базы данных, напротив, обладают гибкой схемой, которая позволяет добавлять и изменять поля и структуру данных без необходимости модификации всей базы данных.

Масштабируемость: РБД обычно предназначены для работы с относительно небольшими объемами данных и не всегда эффективно масштабируются на большие нагрузки. Нереляционные базы данных, с другой стороны, часто спроектированы с учетом горизонтального масштабирования и могут обрабатывать большие объемы данных и высокие нагрузки.

MongoDB - это документоориентированная NoSQL база данных, которая хранит данные в виде документов в формате JSON. Она обеспечивает высокую масштабируемость, гибкость и производительность, что делает ее идеальным выбором для приложений, требующих быстрого доступа к большим объемам данных.

Mongoose - это объектно-документное отображение для MongoDB, которое упрощает работу с базой данных, предоставляя схему и модели для данных.

Socket.io - это библиотека для Node.js, которая обеспечивает двустороннюю связь между клиентом и сервером через веб-сокеты. Она позволяет создавать реальное время приложений, таких как чаты и игры.

Выбранные компоненты и технологии предоставляют эффективный и масштабируемый стек для разработки программного средства для автоматизации работы фитнес-центра, обеспечивая высокую производительность, гибкость и удобство разработки.

В приведенной выше информации было расписано, что мы используем, а также преимущества перед аналогами.

**2.3 Архитектурные решения**

Приложение состоит из клиентской и серверной частей. Серверная часть создана на Node.js и использует API для взаимодействия с клиентской частью и базой данных MongoDB для хранения данных. Клиентская часть написана на HTML, CSS и JavaScript с использованием фреймворка jQuery для упрощения кода. Для удобства разработки и поддержки приложения, а также для его масштабируемости, используется архитектурный шаблон MVC, где модель - это база данных, представление - HTML/CSS/JS/JQuery, а контроллер - сервер Node.js.

В данном пункте была разработана архитектура приложения.

**2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства**

Алгоритм отправки данных на почту (рисунок 2.1).

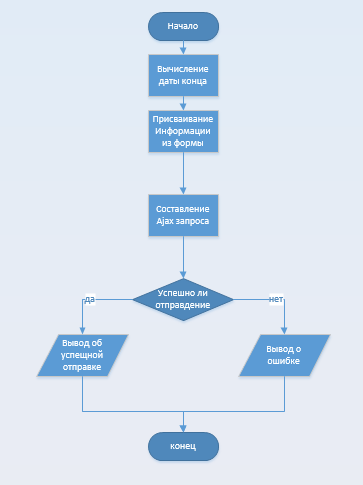


Рисунок 2.1 – Алгоритм отправки данных на почту

Листинг кода:

function sendEmail(){

// Текущая дата

const currentDate = new Date();

const formattedCurrentDate = currentDate.toISOString().slice(0, 10);

// Дата через месяц

const oneMonthLater = new Date();

oneMonthLater.setMonth(currentDate.getMonth() + 1);

const formattedOneMonthLater = oneMonthLater.toISOString().slice(0, 10);

var data = {

service\_id: 'service\_4ig7y57',

template\_id: 'template\_c2krn9h',

user\_id: '2oVnCy2W7\_8JYIuOV',

template\_params: {to\_name:document.getElementById("to\_name").value,

to\_email:document.getElementById("to\_email").value,

cupon\_name:document.getElementById("cupon\_name").value,

time\_end:formattedOneMonthLater,

}

};

$.ajax('https://api.emailjs.com/api/v1.0/email/send', {

type: 'POST',

data: JSON.stringify(data),

contentType: 'application/json'

}).done(function() {

alert('Your mail is sent!');

}).fail(function(error) {

alert('Oops... ' + JSON.stringify(error));

});

}

Алгоритм удаление новости на сервере (рисунок 2.2).

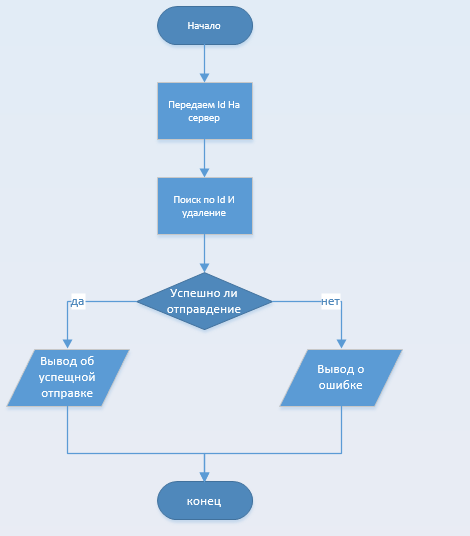


Рисунок 2.2 – Алгоритм удаление новости на сервере

Листинг кода:

let user = req.session.user;

let role = user ? user.role : 'energy';

const title ='Title';

Neuw

.findByIdAndDelete(req.params.id)

.then((result) => {

res.sendStatus(200);

})

.catch((error) => handleError(res,error));

};

Алгоритм редактирования информации о пользователе (рисунок 2.3).

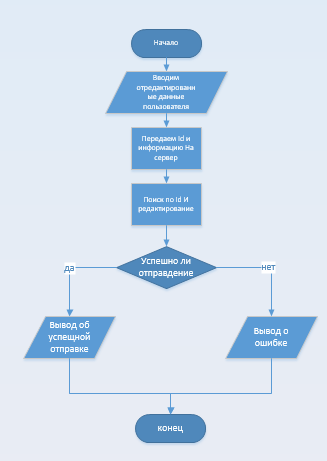


Рисунок 2.3 – Алгоритм редактирования информации о пользователе

Листинг кода:  
const {login, role, password}=req.body;

const {id} = req.params;

User

.findByIdAndUpdate(id,{login, role, password})

.then(result => res.redirect('/work-whith-account'))

.catch((error) => {

console.log(error);

res.render(createPath('error.ejs'),{title:'Error'});

});

С помощью диаграмм мы показали основные процессы бизнес логики.

**2.5 Проектирование пользовательского интерфейса**

Интерфейс программного средства разработан с помощью языка разметки html, а также каскадной таблицы стилей css. Меню спроектировано с использованием списка. Пример кода на рисунке 2.4.

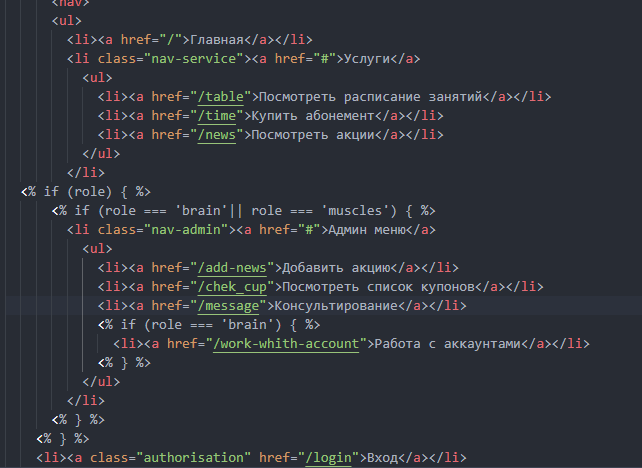


Рисунок 2.4 – Меню навигации

Пользователь может посмотреть расписание в таблице. Рисунки 2.5, 2.6 показываю код таблицы которая выводит данные из списка пока.

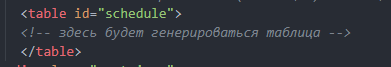


Рисунок 2.5 – Таблица с расписанием



Рисунок 2.6 – Вывод таблицы

На рисунке 2.7 можно увидеть форму для покупки купонов.

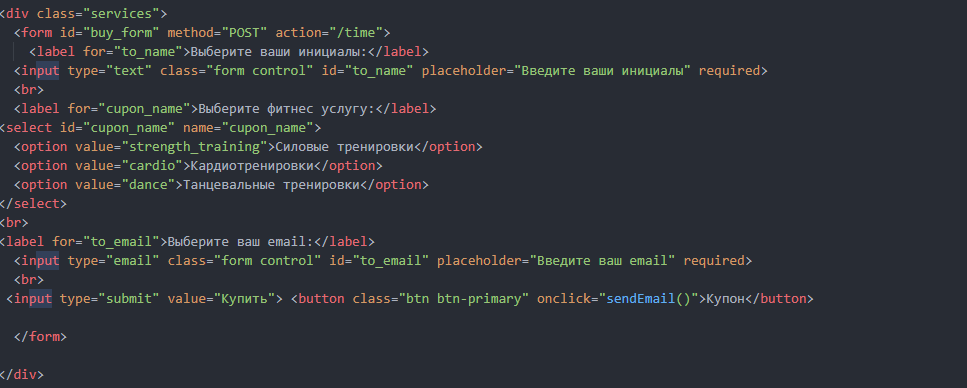


Рисунок 2.7 – Форма для покупки

На рисунке 2.8 можно увидеть форму для оставления отзыва.

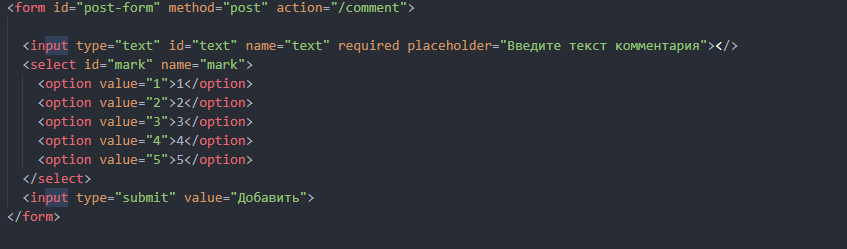


Рисунок 2.8 – Форма для отзывов

В этом пункте разобран пользовательский интерфейс.

**2.6 Методы и средства, используемые для обеспечения безопасности данных**

В данном программном обеспечении используется шифрование пароля. Библиотека для шифрования «bcrypt». Пароль хранится в БД в виде хеш-значения (рисунок 2.9). При авторизации, пароль отправленный на сервер сравнивается с хеш-значением из БД с помощью метода «compare» библиотеки «bcrypt» (рисунок 2.10), в лямбда-функцию приходит «result» являющийся истинной или ложью. В зависимости от его значения определяем есть ли совпадение.



Рисунок 2.9 – Пример хеш-значения пароля

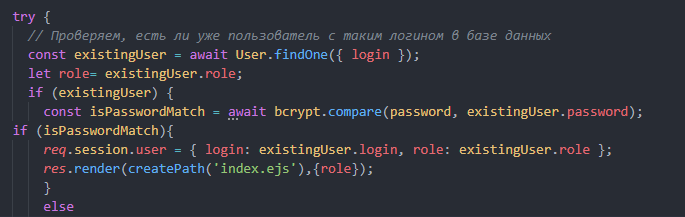


Рисунок 2.10 – Метод «compare»

В данном пункте было наглядно показаны методы и средства для обеспечения безопасности данных.

**3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММЫ**

Успешно считывается информация из базы данных.

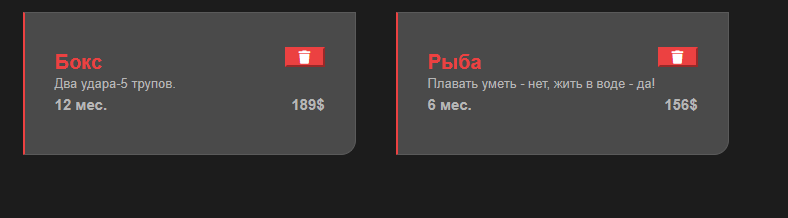


Рисунок 3.1 – Считывание из фала

Проверка на введенные данные (рисунок 3.2).

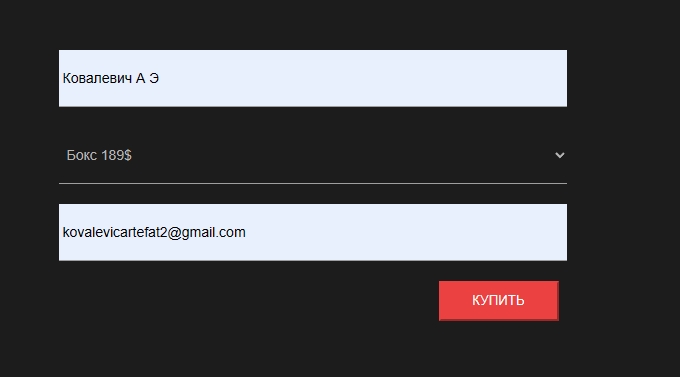


Рисунок 3.2 – Требование заполнения

При отправлении выходит сообщение о успехе (рисунок 3.3)

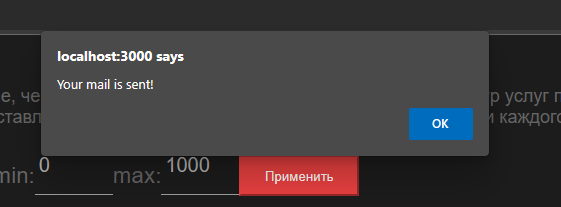


Рисунок 3.3 – Сообщение о успешной отправке

Добавление отзывов (рисунки 3.4).

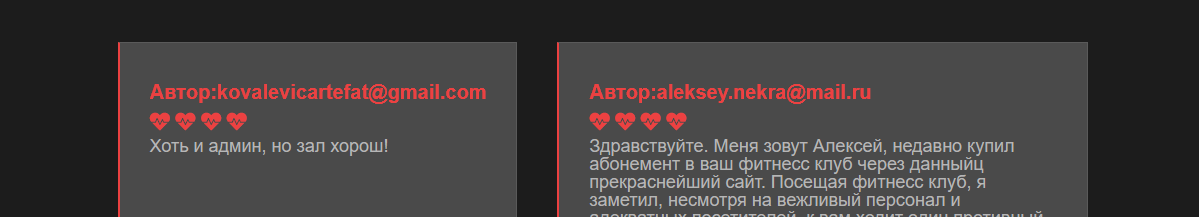


Рисунок 3.4 – Добавление отзывов

Выше была показана работоспособность нашей программы.

**4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ**

**4.1 Авторизация**

При запуске программы нас встречает страница с авторизацией (Рисунок 4.1).

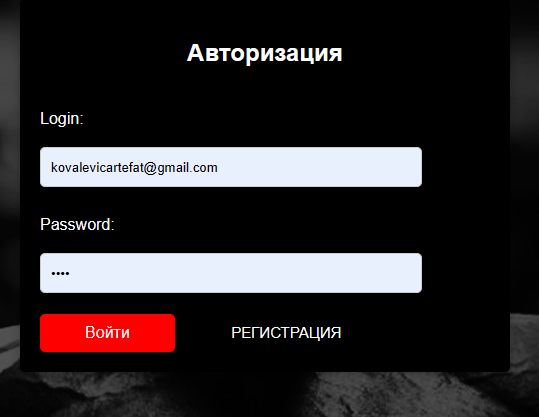


Рисунок 4.1 – Меню авторизации

**4.2 Модуль администратора**

При авторизации как администратор слева выводится меню как на рисунке 4.2, а также открываются разделы в меню администратора как работа с аккаунтами (рисунок 4.3), просмотр купонов (рисунок 4.4), добавление акции (рисунок 4.5).

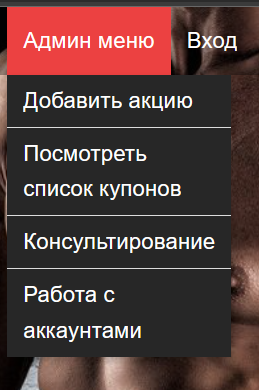


Рисунок 4.2 – Меню модуля администратора

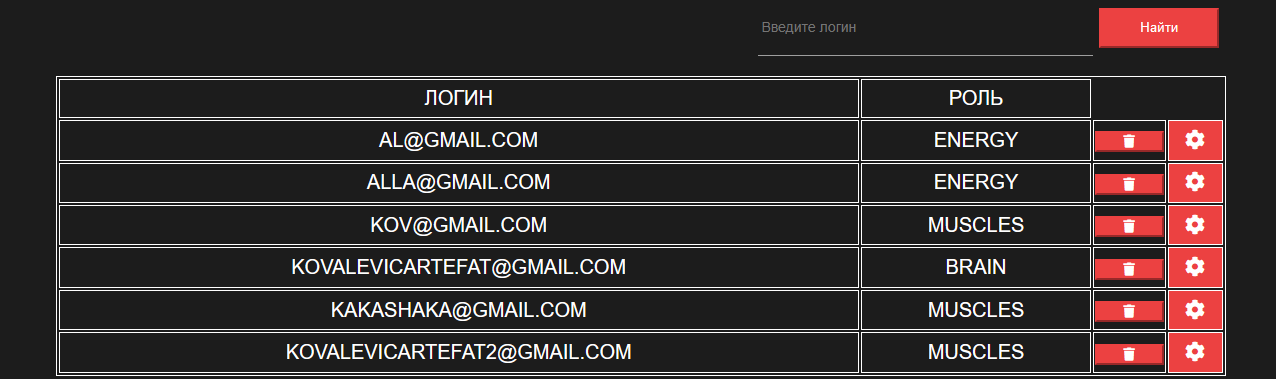


Рисунок 4.3 – Работа с аккаунтами

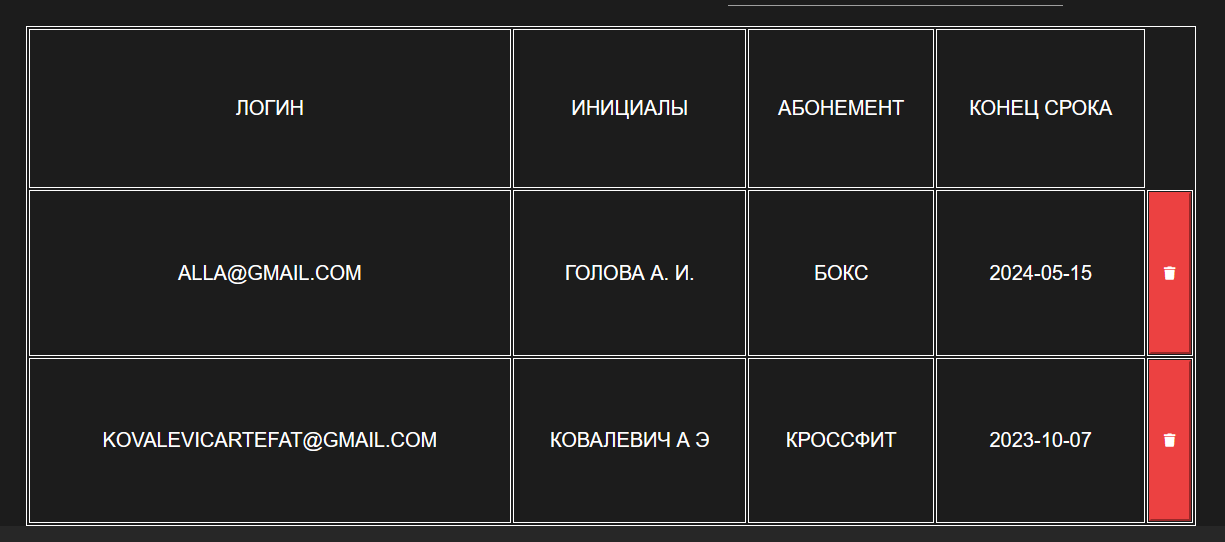


Рисунок 4.4 – Просмотр купонов

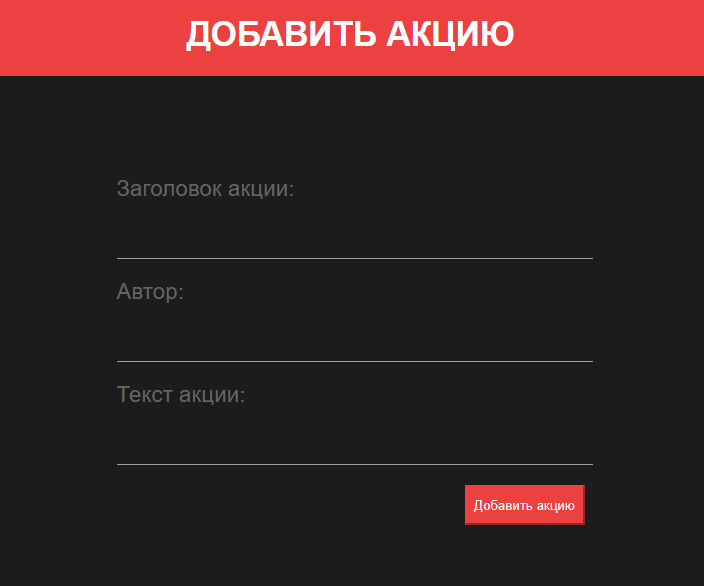


Рисунок 4.5 – Добавление акции

Менеджер имеет такие же права как администратор кроме работы с аккаунтами.

**4.3 Модуль пользователя**

При выборе пользователя запускается главная страница рисунок 4.6.

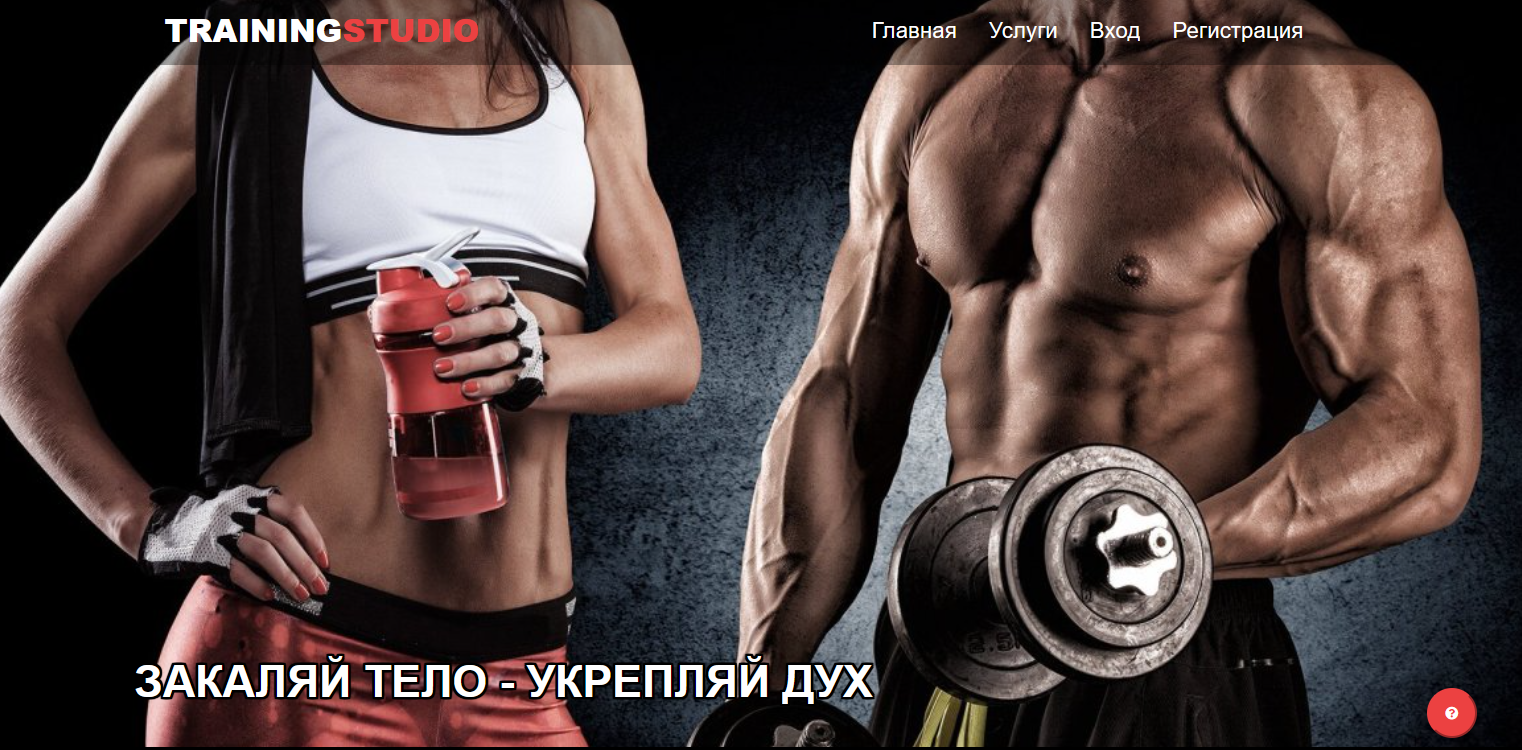


Рисунок 4.6 – Главная страница модуль пользователя

Также пользователь может оставлять отзывы рисунок 4.7.

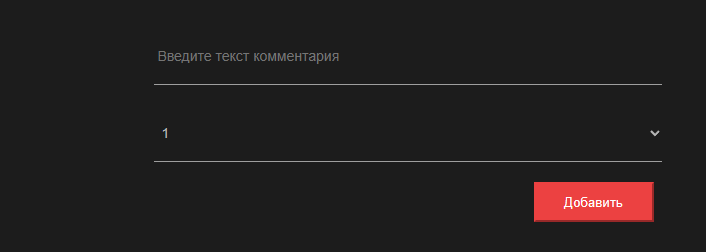


Рисунок 4.7 – Страница оставить отзыв

На вкладке услуги пользователь может увидеть расписание на рисунке 4.8 и покупку купонов на рисунке 4.9.



Рисунок 4.8 – Страница услуг с расписанием и видами

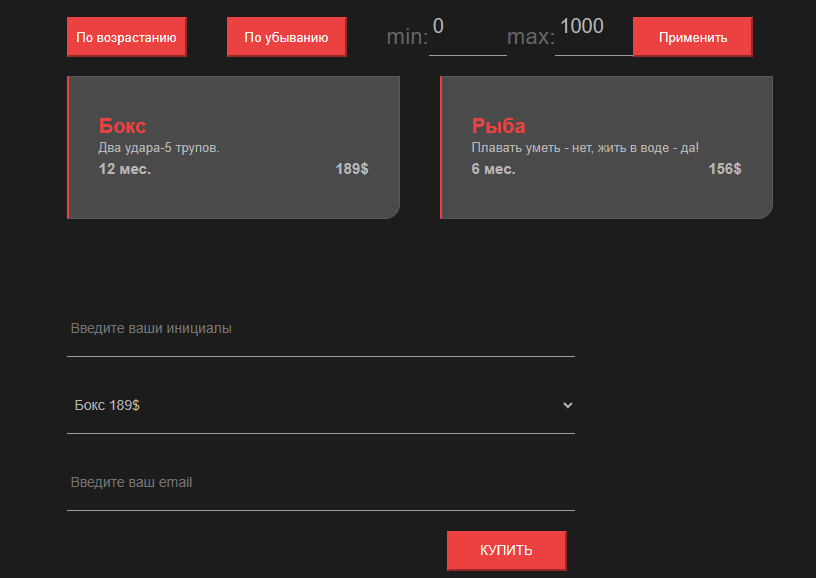


Рисунок 4.9 – Покупка абонемента

**ЗАКЛЮЧЕНИЕ**

В результате выполнения данной курсовой работы было разработано программное средство, предназначенное для автоматизации управления фитнес-центром. Система включает в себя функции управления графиком занятий, ведение базы данных клиентов, расчет и контроль финансовых показателей, а также другие операции, необходимые для эффективного функционирования фитнес-центра.

Преимущества автоматизации фитнес-центра являются очевидными и многообразными. Внедрение разработанного программного средства позволит повысить эффективность управления фитнес-центром, сократить время, затрачиваемое на рутинные операции, минимизировать риски ошибок и недостачи, а также улучшить качество обслуживания клиентов и облегчить работу персонала.

В процессе выполнения курсовой работы были изучены особенности предметной области и проанализированы процессы управления фитнес-центром. Основываясь на полученных знаниях, было проведено проектирование и разработка программного средства, учитывающего требования и особенности данной отрасли. Дальнейшие шаги включали тестирование и проверку работоспособности программы.

Таким образом, выполнение данной курсовой работы привело к созданию полезного инструмента для автоматизации управления фитнес-центром, который способствует повышению эффективности работы и обслуживания клиентов, а также облегчает задачи персонала. Дальнейшая модернизация и развитие программного средства позволят его адаптацию под конкретные потребности фитнес-центра и обеспечат его дальнейший успех и эффективность.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

[1] Флэнаган, Дэвид. JavaScript: Подробное руководство. O'Reilly Media, 2011. 1096 с.

[2] Дакетт, Джон. HTML и CSS: Дизайн и создание веб-сайтов. John Wiley & Sons, 2011. 490 с.

[3] W3Schools. "Учебник HTML." Доступно 28 апреля 2023 г. https://www.w3schools.com/html/default.asp

[4] CSS-Tricks. "Полное руководство по Flexbox." Доступно 28 апреля 2023 г. https://css-tricks.com/snippets/css/a-guide-to-flexbox/

[5] Документация Node.js. "Начало работы." Доступно 28 апреля 2023 г. https://nodejs.org/en/docs/guides/getting-started-guide/

[6] Фримен, Адам, и Тайлер Хенкель. Node.js в действии. Manning Publications, 2017. 416 с.

[7] Пауэлл, Эшли. Изучение Node: Переход на сторону сервера. O'Reilly Media, 2016. 308 с.

[8] Черниговский, Дмитрий. Базы данных. Концептуальное проектирование и моделирование. Издательство "БХВ-Петербург", 2019. 416 с.

**ПРИЛОЖЕНИЕ А**

**Листинг кода алгоритмов, реализующих основную бизнес-логику.**

const createPath=require('../helpers/create-path');

const User=require('../models/user');

const getEdit = (req,res) => {

let user = req.session.user;

let role = user ? user.role : 'energy';

const title ='Title';

User

.findById(req.params.id)

.then((user\_all) => res.render(createPath('edit\_user.ejs'), {user\_all, title, role}))

.catch((error) => {

console.log(error);

res.render(createPath('error.ejs'),{title:'Error', role});

});

};

const putEdit=(req,res)=>{

const {login, role, password}=req.body;

const {id} = req.params;

User

.findByIdAndUpdate(id,{login, role, password})

.then(result => res.redirect('/work-whith-account'))

.catch((error) => {

console.log(error);

res.render(createPath('error.ejs'),{title:'Error'});

});

};

const wwaGet=(req,res)=>{

let user = req.session.user;

let role = user ? user.role : 'energy';

User

.find()

.sort({ createdAt : -1 })

.then((user\_all) => res.render(createPath('work-whith-account.ejs'), {user\_all, role}))

.catch((error) => {

console.log(error);

res.render(createPath('error.ejs'),{title:'Error'});

});

};

const find\_resultGet= async (req,res)=>{

const login = req.query.login;

let user = req.session.user;

let role = user ? user.role : 'energy';

// Поиск пользователя по логину в базе данных

const user\_ass = await User.findOne({ login });

if(user\_ass){

res.render(createPath('find-result.ejs'), { user\_ass , role});

}

else {

User

.find()

.sort({ createdAt : -1 })

.then((user\_all) => res.render(createPath('work-whith-account.ejs'), {user\_all, role}))

.catch((error) => {

console.log(error);

res.render(createPath('error.ejs'),{title:'Error'});

});

}

};

const wwaDelete=(req,res)=>{

let user = req.session.user;

let role = user ? user.role : 'energy';

const title ='Title';

User

.findByIdAndDelete(req.params.id)

.then((result) => {

res.sendStatus(200);

})

.catch((error) => {

console.log(error);

res.render(createPath('error.ejs'),{title:'Error', role});

});

};

const wwaPost= async (req,res)=>{

let user = req.session.user;

let role = user ? user.role : 'energy';

try {

const { login } = req.body;

const user = await User.findOne({ login });

if (!user) {

return res.status(404).send({ message: 'Пользователь не найден' });

}

res.render('popup', { user , role});

} catch (error) {

console.error(error);

res.status(500).send({ message: 'Ошибка на сервере' });

}

};

const registerGet=(req,res)=>{

let user = req.session.user;

let role = user ? user.role : 'energy';

res.render(createPath('register.ejs', {role}));

};

const registerPost=async (req, res) => {

const { login, password, passwordConfirm } = req.body;

const role='energy';

// Проверка, что пароль и его подтверждение совпадают

if (password !== passwordConfirm) {

return res.status(400).send('Пароли не совпадают');

}

try {

// Проверяем, есть ли уже пользователь с таким логином в базе данных

const existingUser = await User.findOne({ login });

if (existingUser) {

return res.status(400).send('Пользователь с таким логином уже существует');

}

// Создаем нового пользователя в базе данных

const newUser = new User({ login, password, role });

await newUser.save();

// Перенаправляем пользователя на страницу входа

res.redirect('/login');

} catch (error) {

console.log(error);

res.status(500).send('Произошла ошибка при регистрации пользователя');

}

};

const loginGet=(req,res)=>{

res.render(createPath('login.ejs')); // Отображаем страницу авторизации

};

const loginPost=async (req, res) => {

const { login, password } = req.body;

try {

// Проверяем, есть ли уже пользователь с таким логином в базе данных

const existingUser = await User.findOne({ login });

let role= existingUser.role;

if (existingUser) {

if(existingUser.password==password)

{

req.session.user = { login: existingUser.login, role: existingUser.role };

res.render(createPath('index.ejs'),{role});

}

else

{

res.status(500).send('Произошла ошибка при регистрации пользователя');

}

}

} catch (error) {

console.log(error);

res.status(500).send('Произошла ошибка при регистрации пользователя');

}

};

module.exports={

getEdit,

putEdit,

wwaGet,

find\_resultGet,

wwaDelete,

wwaPost,

registerGet,

registerPost,

loginGet,

loginPost,

};const createPath=require('../helpers/create-path');

const Neuw=require('../models/new');

const handleError=(res, error)=>{

console.log(error);

res.render(createPath('error.ejs'),{title:'Error', role});

}

const getNew=(req,res)=>{

let user = req.session.user;

let role = user ? user.role : 'energy';

const title ='Title';

Neuw

.findById(req.params.id)

.then((news) => res.render(createPath('new.ejs'), {news, title, role}))

.catch((error) => handleError(res,error));

};

const getNews=(req,res)=>{

const title = 'News';

let user = req.session.user;

let role = user ? user.role : 'energy';

Neuw

.find()

.sort({ createdAt : -1 })

.then((news) => res.render(createPath('news.ejs'), {news, title, role}))

.catch((error) => handleError(res,error));

};

const getAdd = (req, res) => {

let user = req.session.user;

let role = user ? user.role : 'energy';

res.render(createPath('add-news.ejs'), { role });

};

const addNews=(req,res)=>{

let user = req.session.user;

let role = user ? user.role : 'energy';

const {tit, author, text}=req.body;

const neuw =new Neuw({tit, author, text});

neuw

.save()

.then((result) => res.redirect('/news'))

.catch((error) => handleError(res,error));

};

const deleteNews=(req,res)=>{

let user = req.session.user;

let role = user ? user.role : 'energy';

const title ='Title';

Neuw

.findByIdAndDelete(req.params.id)

.then((result) => {

res.sendStatus(200);

})

.catch((error) => handleError(res,error));

};

module.exports={

getNew,

getNews,

deleteNews,

addNews,

getAdd,

};const createPath=require('../helpers/create-path');

const Buy=require('../models/buy\_users');

const getCup = (req,res) => {

let user = req.session.user;

let role = user ? user.role : 'energy';

Buy

.find()

.sort({ createdAt : -1 })

.then((cupon) => res.render(createPath('chek\_cup.ejs'), {cupon, role}))

.catch((error) => {

console.log(error);

res.render(createPath('error.ejs'),{title:'Error', role});

});

};

const getTime = (req,res) => {

let user = req.session.user;

let role = user ? user.role : 'energy';

res.render(createPath('time.ejs'), { user, role});

};

const timePost = (req,res) => {

const currentDate = new Date();

const formattedCurrentDate = currentDate.toISOString().slice(0, 10);

// Дата через месяц

const oneMonthLater = new Date();

oneMonthLater.setMonth(currentDate.getMonth() + 1);

const formattedOneMonthLater = oneMonthLater.toISOString().slice(0, 10);

let user = req.session.user;

let role = user ? user.role : 'energy';

const {to\_name, cupon\_name}=req.body;

const buy\_users = new Buy({

user\_login: user.login,

to\_name: to\_name,

cupon\_name: cupon\_name,

time\_end: formattedOneMonthLater

});console.log(buy\_users);

buy\_users

.save()

.then((result) => res.redirect('/time'))

.catch((error)=>{

console.log(error);

res.render(createPath('error.ejs'),{title:Error, role})

})

};

const frchekGet = async (req,res) => {

const user\_login = req.query.login;

let user = req.session.user;

let role = user ? user.role : 'energy';

// Поиск пользователя по логину в базе данных

const buy\_users = await Buy.findOne({ user\_login });

if(buy\_users){

res.render(createPath('find\_result\_chek.ejs'), { buy\_users, role });

}

else {

Buy

.find()

.sort({ createdAt : -1 })

.then((buy\_users) => res.render(createPath('chek\_kup.ejs'), {buy\_users, role}))

.catch((error) => {

console.log(error);

res.render(createPath('error.ejs'),{title:'Error'});

});

}

};

module.exports={

getCup,

getTime,

timePost,

frchekGet,

};

**ПРИЛОЖЕНИЕ Б**

**Отчет о проверке на заимствование в системе «Антиплагиат»**

