МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшкая школа экономики и управления
Кафедра «Информционные технологии в экономике»

РАБОТА ПРОВЕРЕНА
Рецензент, _____
_____/_____/
«_____» _____2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.т.н., с.н.с.
_____/ Б.М. Суховилов /
«_____» _____2018 г.

Автоматизация работы спортивного клуба: проект «Умный зал»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.03.2018.119. ВКР

Руководитель, к.т.н., доцент
_____/ О.И.Галичин /
«_____» _____2018 г.

Автор
студент группы ЭиУ–469
_____/ Г.С. Сабуров /
«_____» _____2018 г.

Нормоконтролер, ст. преподаватель
_____/ Е.Н. Горных /
«_____» _____2018 г.

Челябинск 2018

АННОТАЦИЯ

Сабуров Г.С. Разработка Web-приложения для автоматизации работы фитнес клуба. – Челябинск: ЮУрГУ, ЭиУ-469, 70 стр., 31 ил., 24 табл., библиогр. список – 9 наим.

Целью дипломного проекта является разработка Web-приложения, автоматизирующего работу фитнес клуба.

В ходе работы поставлены задачи: Удобство эксплуатации, отвечать всем современным тенденциям и условиям рынка, предоставить возможности управления как со стороны клиента, так и со стороны администратора.

Разработано приложение, которое позволяет клиенту создать свой личный кабинет, выбирать и управлять нужными услугами, узнавать всю актуальную информацию. Со стороны администратора приложение позволяет контролировать все услуги, используемые клиентами, отмечать их действия, предоставлять всю необходимую информацию о клубе, а также проводить мониторинг работы клуба и его дальнейшее улучшение.

В сравнении с другими Web-приложениями, автоматизирующими работу фитнес клуба, преимущество заключается в большей функциональности, удобстве использования, доступности для внедрения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Предприятие “Уралочка” является лидером в спорт индустрии города Златоуст, но имеет свои недостатки в виде автоматизации и обратной связи с клиентами, поэтому разработка оптимального приложения может оказать положительное влияние на работу клуба.

Данное предприятие, как и большинство других в этой области, редко может позволить дорогостоящий программный продукт, нацеленный на автоматизацию, так как это достаточно дорого, но данная программа отвечает, как отличному качеству, так и цене.

Актуальность дипломного проекта в том, что данная отрасль только начинает развиваться и имеется большой потенциал для реализации новых идей.

Цель дипломного проекта – разработка приложения, автоматизирующего работу фитнес клуба.

Для достижения поставленной цели решены следующие задачи:

- выполнен анализ предметной области;
- произведен анализ рынка;
- разработаны диаграммы и модели
- разработана структура и сама база данных;
- разработан Web–интерфейс;
- разработан функционал Web-приложения
- произведены тесты приложения

Объектом работы является фитнес клуб “Уралочка”

Данный дипломный проект представляет интерес не только для администрации управления клубом, но и значительно улучшает качество обслуживания клиентов и положительно влияет на привлечение новых клиентов.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Анализ деятельности фитнес клуба

Фитнес клуб не имеет специального приложения для клиентов и дистанционного мониторинга своих занятий. Персонал и сотрудники администрации клуба не имеют возможности автоматизированного учета клиентов, услуг и работы, поэтому автоматизация может дать новый импульс в развитии.

Основными пользователями будут – администраторы, тренеры и клиенты фитнес клуба.

Администратор фитнес клуба выполняет задачи такие как:

- работа с клиентами;
- работа с сайтом;
- работа с предоставляемыми услугами;
- отчет о работе и ситуации в клубе перед начальством.

Тренер фитнес клуба – сотрудник, который выполняет задачи такие как:

- создание, а также учет программ тренировок;
- контроль за выполнением упражнений;
- мотивация клиентов на выполнение поставленных целей.

Клиент фитнес клуба – это человек, который пользуется услугами:

- просмотр графика работы;
- выбор подходящего тренера;
- занятие в зале и использование дополнительных услуг;
- подбор индивидуальной программы тренировок и питания.

1.2 Анализ существующих систем

Сравнительный анализ приведен в таблице 1.

Таблица 1 – Сравнительный анализ клубов

Клуб / Возможности	Ск “Канон”	Ск “Пик”	Ск “агат”	Разрабатываемое Web-приложение ”Уралочка”
Web-приложение	-	-	+	+
Составление расписания с интеграцией на сайт	+	+	+	+
Дистанционная информация о клубе	-	-	+	+
Создание личного кабинета	-	-	+	+
Автоматический Учет клиентов и услуг	-	-	-	+
Удобный просмотр посещений	-	-	-	+
Простой и понятный интерфейс	-	-	+	+
Фильтрация и поиск	-	-	-	+
Разграничение прав доступа	-	-	-	+
Предварительная запись	-	-	+	+
Администрирование через сайт	-	-	-	+

В ходе поиска и анализа фитнес клубов было выявлено, что большинство клубов не имеют CRM-систем, а в некоторых случаях имеются стандартные информационные порталы, не имеющие подходящего функционала. Так же часть проанализированных клубов не имеют собственных ресурсов, кроме как страниц в социальных сетях.

В результате сравнительного анализа выявлено:

- отсутствие современных CRM-систем;
- имеется потенциал развития данной области;
- большие возможности внедрения.

Вывод по разделу один

Цель первого раздела была полностью выполнена:

- был произведен анализ деятельности фитнес клуба;
- выполнен анализ существующих систем;
- изучена предметная область.

2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

2.1 Разработка структуры ИС

UML (унифицированный язык моделирования) – язык графического описания для объектного моделирования.

Диаграмма прецедентов (клиент + тренер + админ = сайт) представлена на рисунке 2.1.

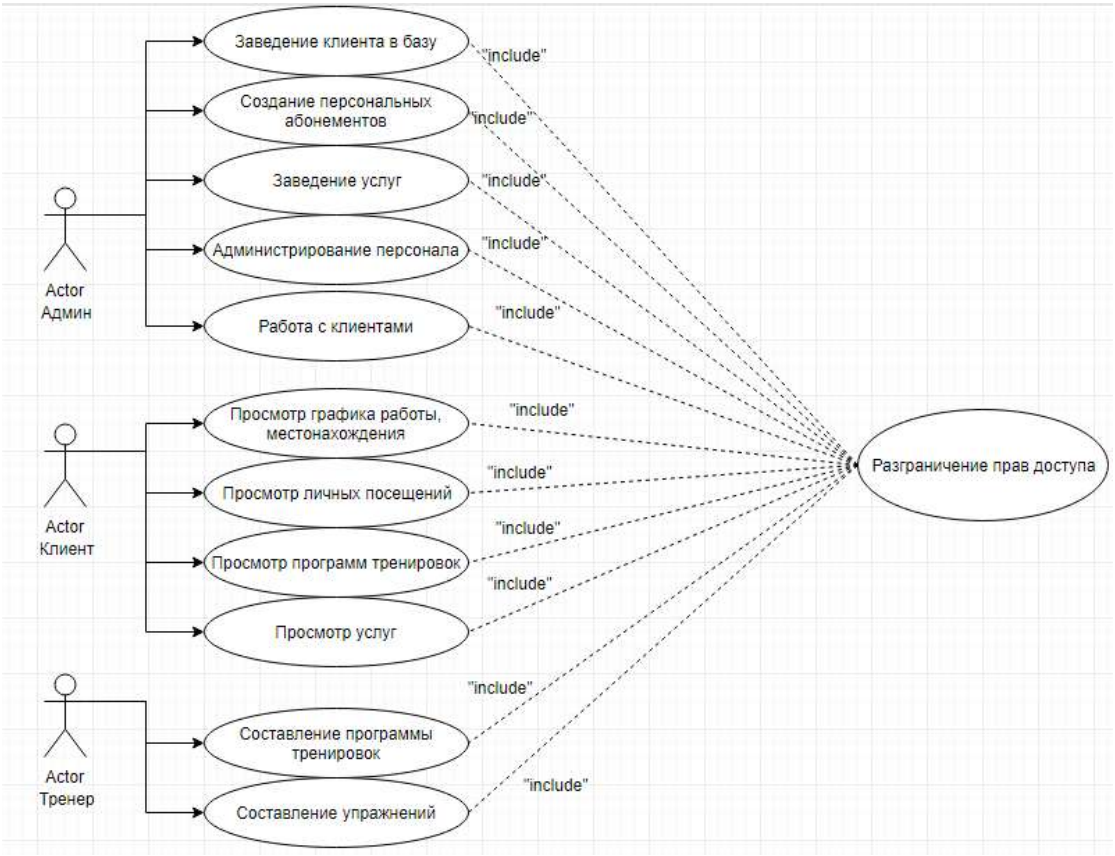

Рисунок 2.1 – Диаграмма прецедентов

Диаграмма последовательности (добавление типов абонемента) представлена на рисунке 2.2.

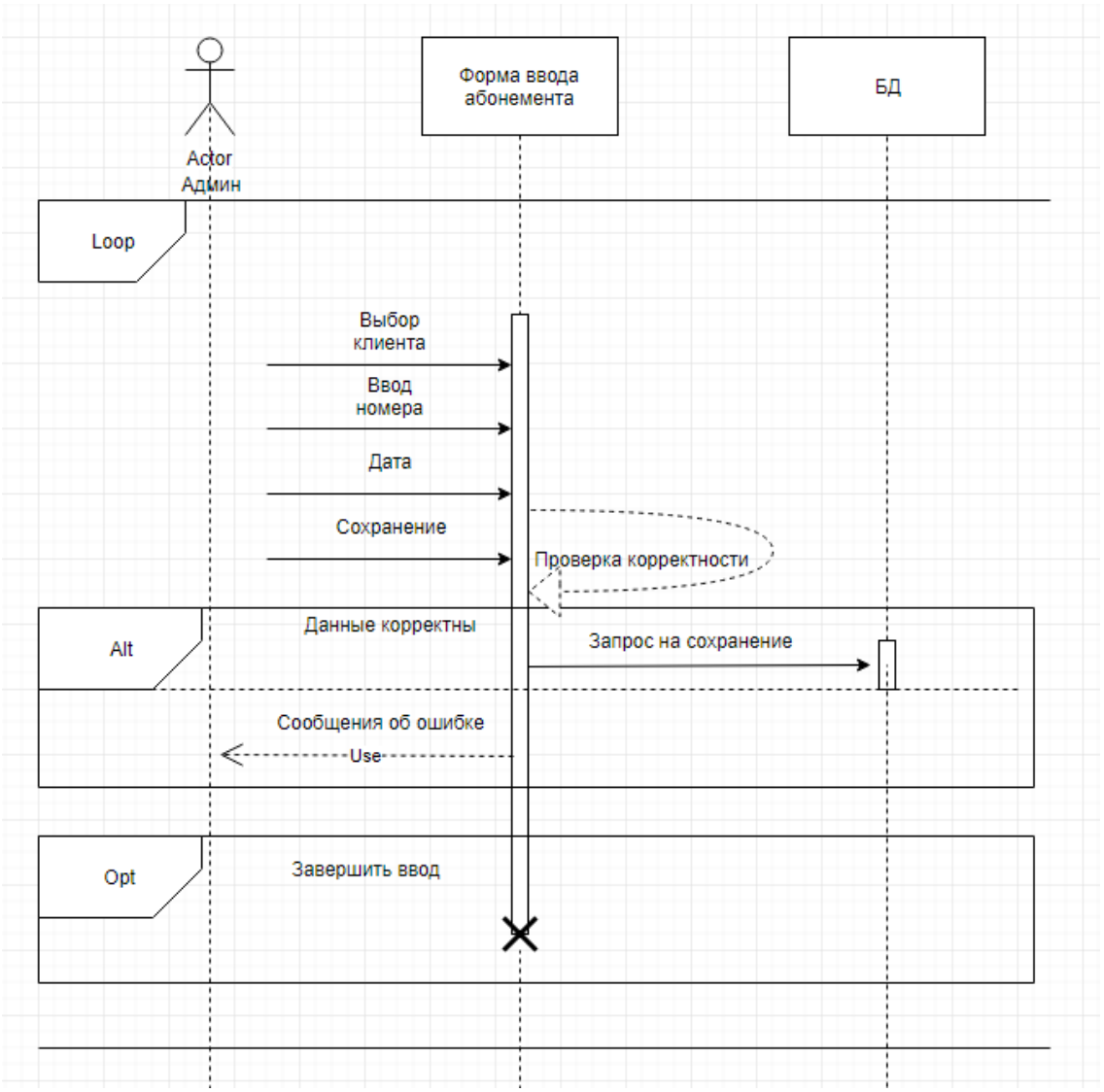

Рисунок 2.2 – Диаграмма последовательности

2.2 Контекстная диаграмма

BPwin – программный продукт, с помощью которого можно с помощью которого можно создавать графические модели и описывать их.

Выполненные модели BPwin представлены на рисунке 2.3 и 2.4.

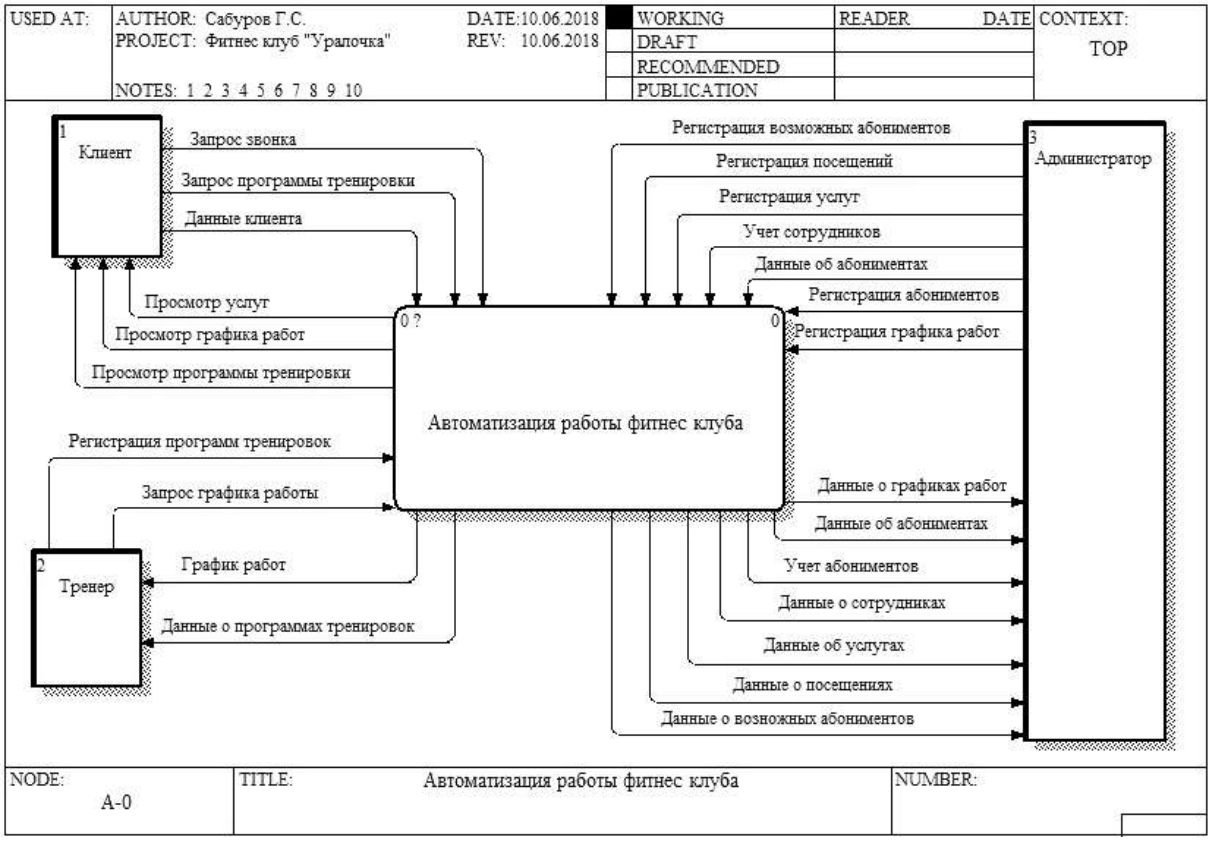

Рисунок 2.3 – Схема БП Win

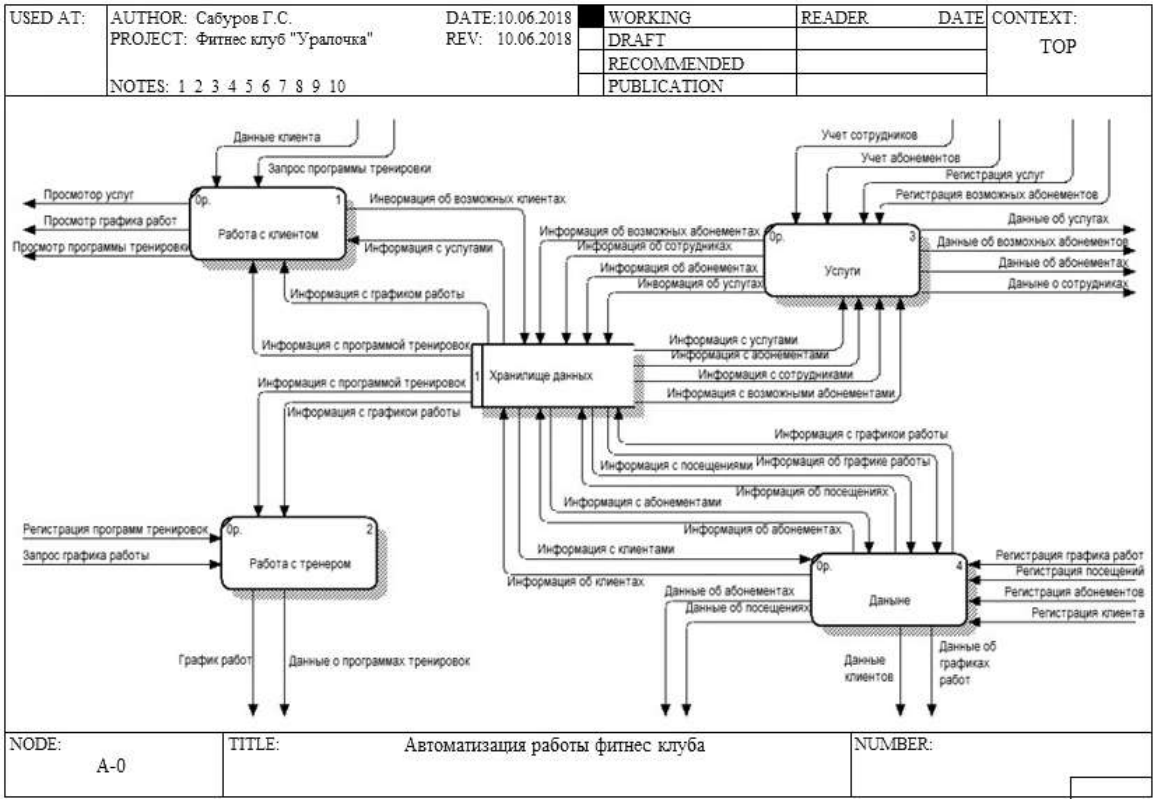

Рисунок 2.4 – Схема БП Win

2.3 Задачи Web-приложения

Данное приложение позволяет решить ряд определенных задач:

- объединение работы администратора, клиента, тренера в одной системе;
- автоматизация учета абонементов и клиентов;
- возможность работы с персоналом;
- создание индивидуального кабинета с необходимыми возможностями;
- удобная, понятная, и простая в использовании.

2.4 Выбор программных средств

Основной язык, используемый в Web-приложении, C# - является объектно-ориентированным. Также C# хорошо сочетается с другими программными продуктами, используемыми в работе.

Microsoft .NET Framework – представляет собой программную платформу.

ASP.NET – представляет собой Web-платформу для создания серверных Web-приложений.

СУБД – Microsoft SQL Server – представляет собой систему создания и управления реляционными базами данных.

Даная СУБД выбрана так, как она отвечает всем современным стандартам, имеет возможность кроссплатформенности, а также легко работает с другими продуктами Microsoft.

Клиент:

HTML (язык разметки документов), CSS (язык описания внешнего вида документов), JavaScript (язык программирования с помощью которого веб-страницам придается интерактивность). JQuery (библиотека JavaScript, которая основана на взаимодействии языка HTML и JavaScript)

2.5 Проектирование и применение базы данных

База данных

Схема базы данных

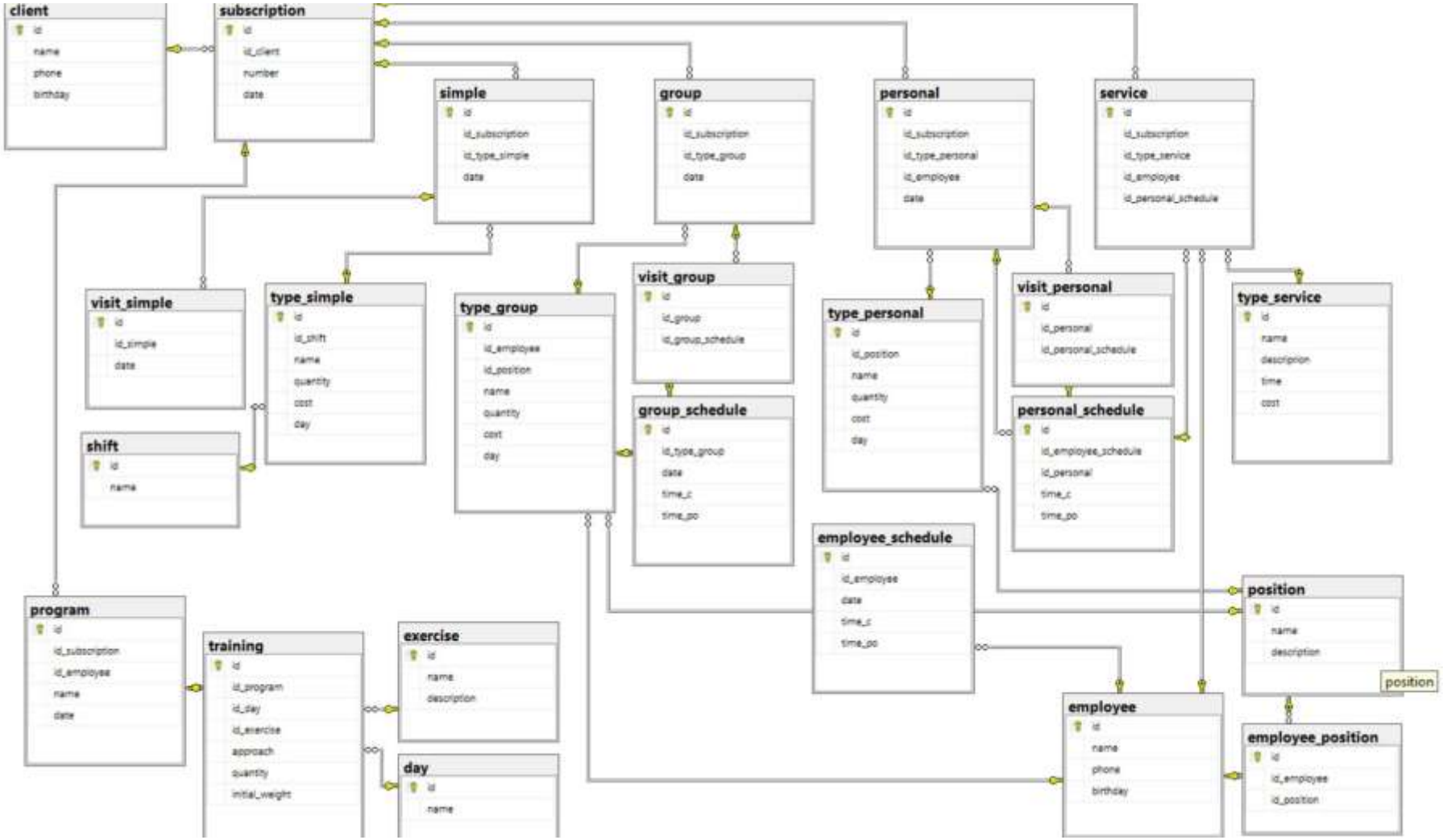

Рисунок 2.5 – Схема базы данных

На данном рисунке видно, что база нормализована и состоит из 24 взаимосвязанных таблиц.

2.6 Описание структуры таблиц базы данных предметной области

Рассмотрим структуру и тип данных каждой таблицы, созданных исходя из построенной модели.

Таблица «client» содержит информацию о клиентах. Состав таблицы приведен в таблице 2.1.

Таблица 2.1 – client

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	name	nvarchar(250)	Имя клиента
3.	phone	nvarchar(50)	Телефон клиента
4.	birthday	date	Дата рождения клиента

Таблица «day» содержит информацию о днях. Состав таблицы приведен на рисунке 2.2.

Таблица 2.2 – day

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	name	nvarchar(50)	Название дня

Таблица «employee» содержит информацию о сотруднике. Состав таблицы приведен в таблице 2.3.

Таблица 2.3 – employee

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	name	nvarchar(250)	Имя сотрудника
3.	phone	nvarchar(50)	Телефон сотрудника
4.	birthday	date	Дата рождения сотрудника

Таблица «employee_position» содержит информацию о должности сотрудника. Состав таблицы приведен в таблице 2.4.

Таблица 2.4 – employee_position

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_empolyee	int	Имя сотрудника
3.	id_positon	int	Должность сотрудника

Таблица «employee_schedule» содержит информацию о расписании сотрудника. Состав таблицы приведен в таблице 2.5.

Таблица 2.5 – employee_schedule

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_empolyee	int	Имя сотрудника
3.	date	date	Дата работы
4.	time_c	time	Начало работы
5.	time_po	time	Конец работы

Таблица «exercise» содержит информацию об упражнениях. Состав таблицы приведен в таблице 2.6.

Таблица 2.6 – exercise

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	name	nvarchar(50)	Название упражнения
3.	discription	nvarchar(Max)	Описание упражнения

Таблица «group» содержит информацию о групповых абонементах. Состав таблицы приведен в таблице 2.7.

Таблица 2.7 – group

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_subscription	int	Абонемент клиента
3.	id_type_group	int	Тип занятия
4.	date	datetime	Дата

Таблица «group_schedule» содержит информацию о расписании групповых занятий. Состав таблицы приведен в таблице 2.8.

Таблица 2.8 – group_schedule

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_type_group	int	Тип группы
3.	date	datetime	Дата работы

4.	time_c	time	Начало работы
5.	time_po	time	Конец работы

Таблица «personal» содержит информацию о персональном абонементе. Состав таблицы приведен в таблице 2.9.

Таблица 2.9 – personal

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_subscription	int	Абонемент клиента
3.	id_type_personal	int	Тип занятия
4.	id_employee	int	сотрудник
5.	date	datetime	дата

Таблица «personal_schedule» содержит информацию о расписании занятий персональных абонементов. Состав таблицы приведен в таблице 2.10.

Таблица 2.10 – personal_schedule

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_employee_schedule	int	Расписание сотрудника
3.	id_personal	int	Персональный абонемент
4.	time_c	time	Начало занятия
5.	time_po	time	Конец занятия

Таблица «position» содержит информацию о должностях. Состав таблицы приведен в таблице 2.11.

Таблица 2.11 – position

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	name	nvarchar(100)	Название должности
3.	discription	nvarchar(Max)	Описание должности

Таблица «program» содержит информацию о программе тренировок. Состав таблицы приведен в таблице 2.12.

Таблица 2.12 – program

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_subscription	int	Номер абонемента
3.	id_employee	int	сотрудник
4.	name	nvarchar(50)	Название программы
5.	date	date	Дата составления

Таблица «shift» содержит информацию о сменах в зале. Состав таблицы приведен в таблице 2.13.

Таблица 2.13 – shift

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы

2.	name	nvarchar(50)	Название смены

Таблица «simple» содержит информацию о простых абонементах. Состав таблицы приведен в таблице 2.14.

Таблица 2.14 – simple

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_subscription	int	Просто абонемент
3.	id_type_simple	int	Тип простого
4.	date	datetime	Дата

Таблица «subscription» содержит информацию об абонементах. Состав таблицы приведен в таблице 2.15.

Таблица 2.15 – subscription

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_client	int	клиент
3.	number	Nvarchar(50)	Номер абонемента
4.	date	datetime	Дата

Таблица «time» содержит информацию о времени. Состав таблицы приведен в таблице 2.16.

Таблица 2.16 – time

№	Название поля	Тип поля	Назначение

1.	id	int	Идентификатор таблицы
2.	time_c	time(1)	Время начала
3.	time_po	time(1)	Время конца

Таблица «training» содержит информацию о тренировке. Состав таблицы приведен в таблице 2.17.

Таблица 2.17 – training

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_program	int	программа
3.	id_day	int	День занятия
4.	id_exercise	int	упражнение
5.	approach	int	подход
6.	quantity	int	количество
7.	initial_weight	float	Начальный вес

Таблица «type_group» содержит информацию о групповых занятиях. Состав таблицы приведен в таблице 2.18.

Таблица 2.18 – type_group

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_employee	int	сотрудник
3.	id_position	int	должность
4.	name	nvarchar(100)	группа
5.	quantity	int	количество
6.	cost	money	стоимость
7.	day	float	Активных дней

Таблица «type_personal» содержит информацию о персональных типах занятий. Состав таблицы приведен в таблице 2.19.

Таблица 2.19 – type_personal

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_position	int	должность
3.	name	nvarchar(100)	группа
4.	quantity	int	количество
5.	cost	money	стоимость
6.	day	int	Активных дней

Таблица «type_simple» содержит информацию о простых типах занятий. Состав таблицы приведен в таблице 2.20.

Таблица 2.20 – type_simple

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_shift	int	смена
3.	name	nvarchar(100)	группа
4.	quantity	int	количество
5.	cost	money	стоимость
6.	day	int	Активных дней

Таблица «visit_group» содержит информацию о групповых посещениях. Состав таблицы приведен в таблице 2.21.

Таблица 2.21 – visit_group

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_group	int	группа
3.	id_group_schedule	int	расписание

Таблица «visit_personal» содержит информацию о персональных посещениях. Состав таблицы приведен в таблице 2.22.

Таблица 2.22 – visit_personal

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_personal	int	Персональный абонемент
3.	id_personal_schedule	int	Посещение персонального

Таблица «visit_simple» содержит информацию о простых посещениях. Состав таблицы приведен в таблице 2.23.

Таблица 2.23 – visit_simple

№	Название поля	Тип поля	Назначение
1.	id	int	Идентификатор таблицы
2.	id_simple	int	Простой абонемент
3.	date	datetime	Дата посещения

2.7 Структура работы приложения

Web-приложение предназначено для таких пользователей как неавторизированный пользователь, клиент, тренер и администратор.

Чтобы получить доступ к различным функциям нужно авторизоваться.

На рисунке 2.6 показана структура Web-приложения.

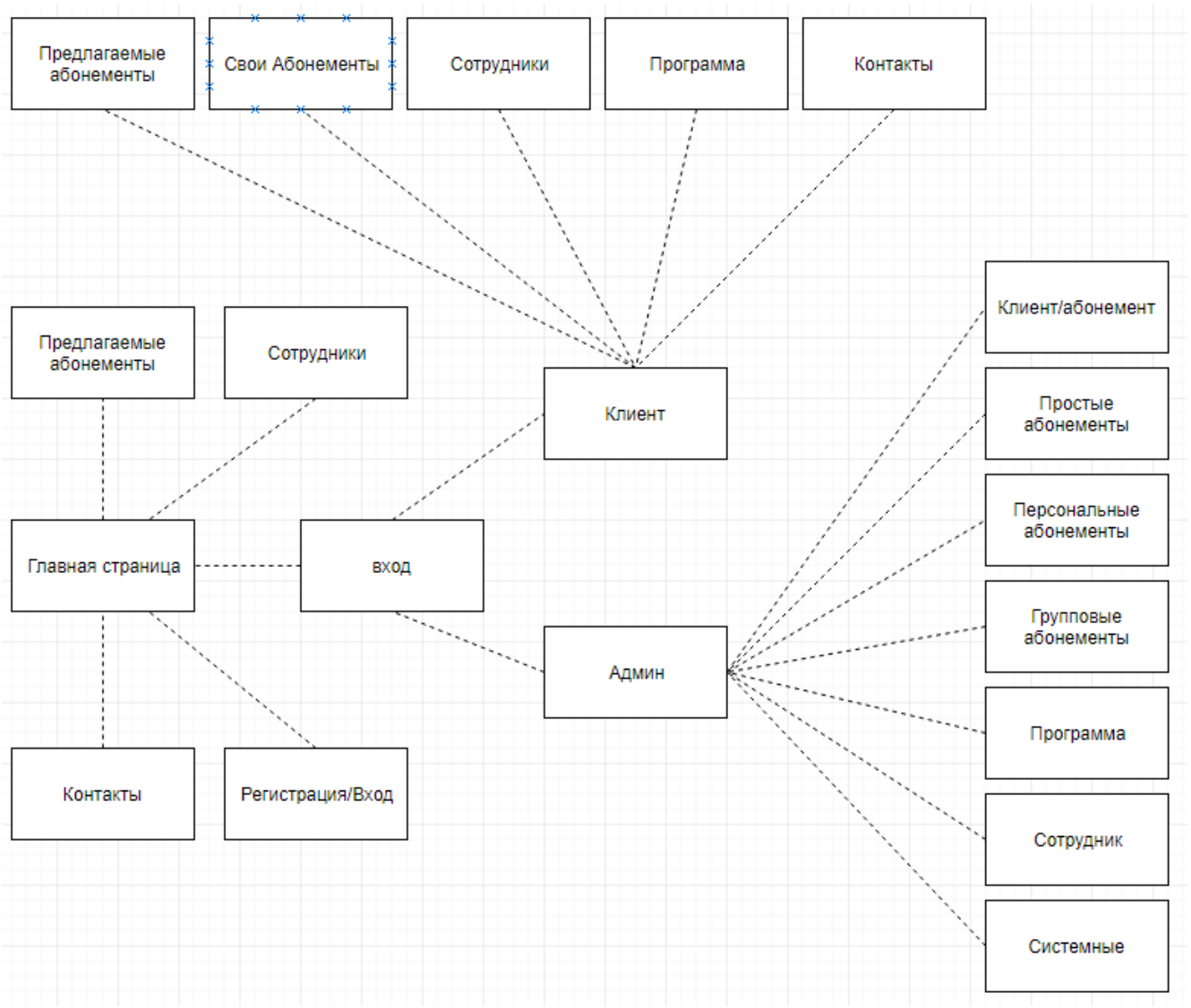

Рисунок 2.6 – Общая структура Web-приложения

Итоги по разделу два:

- разработана структура информационной системы
- выполнена контекстная диаграмма;
- поставлены задачи приложения;
- выбраны программные средства;
- разработана база данных
- выполнена структура приложения

3 ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА

3.1 Общий вид приложения

С приложением можно взаимодействовать не авторизированным пользователям. В данном случае пользователь может видеть:

- главная страница;
- предлагаемые абонементы;
- сотрудники;
- контакты;
- регистрация;
- вход.

Вид главного меню показан на рисунке 3.1

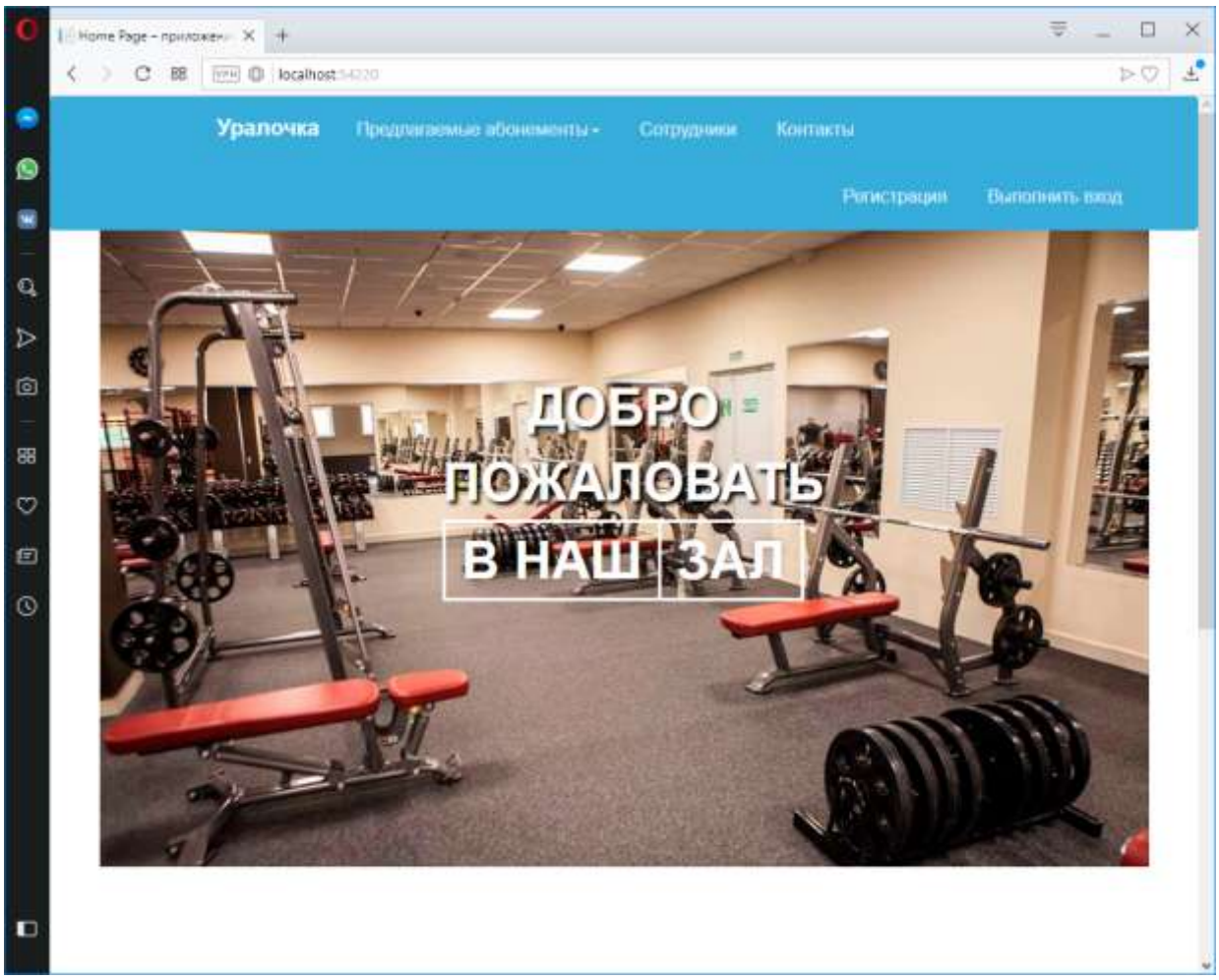

Рисунок 3.1 – Главная страница

Нажав на вкладку меню “Предлагаемые абонементы” пользователь сможет выбрать нужный тип абонемента, после чего откроется прайс в данном разделе. Визуальное представление показано на рисунке 3.2 и 3.3.

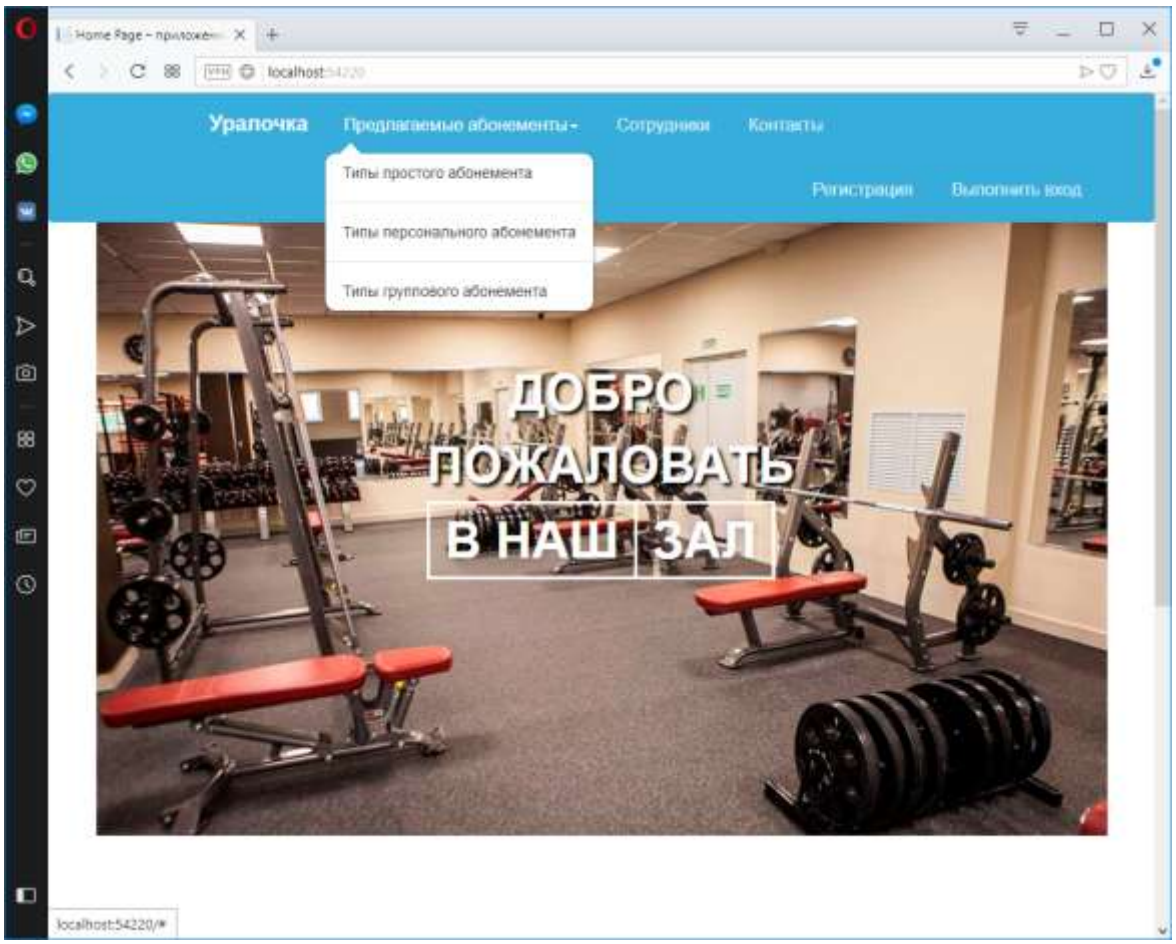

Рисунок 3.2 – Выбор типа абонемента

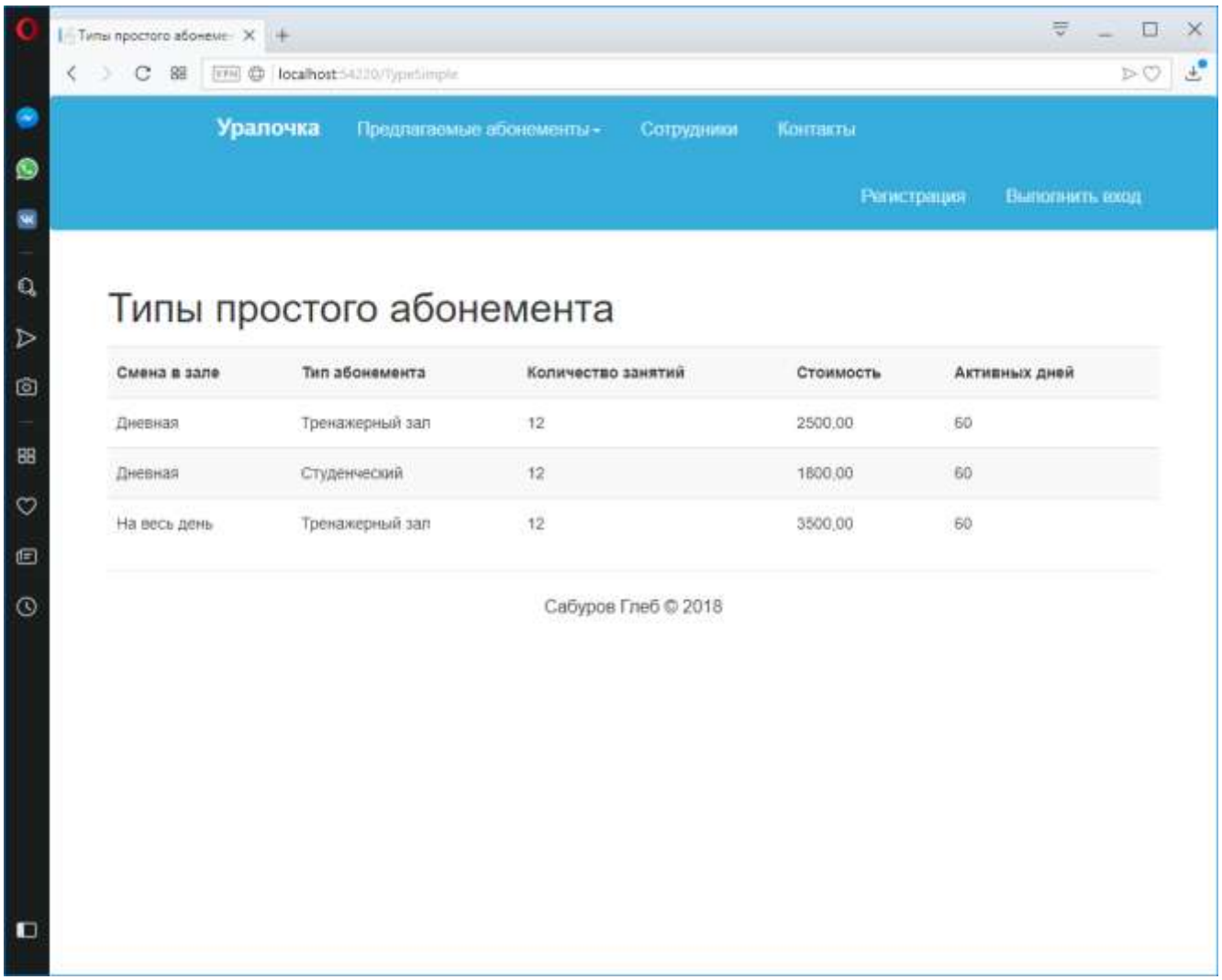

Рисунок 3.3 – Прайс на простые абонементы

Для получения доступа к функционалу приложению нужно выполнить вход. Существует два типа пользователей – администратор и клиент. На рисунке 3.4 показан интерфейс входа в приложение.

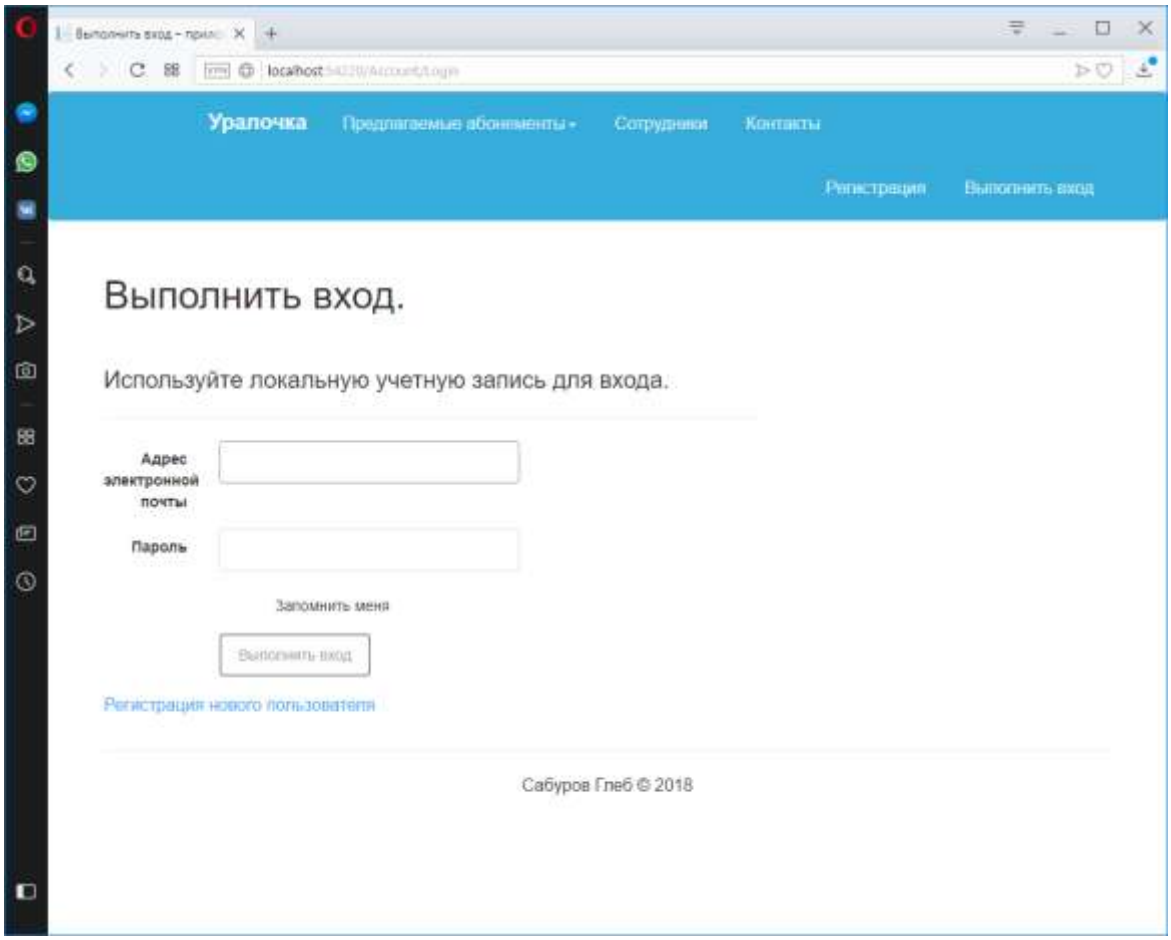

Рисунок 3.4 – Прайс на простые абонементы

3.2 Возможности приложения для администратора

При авторизации под администратором пользователю доступна наибольшая функциональность приложения. Общий вид приложения для администратора показан на рисунке 3.5.

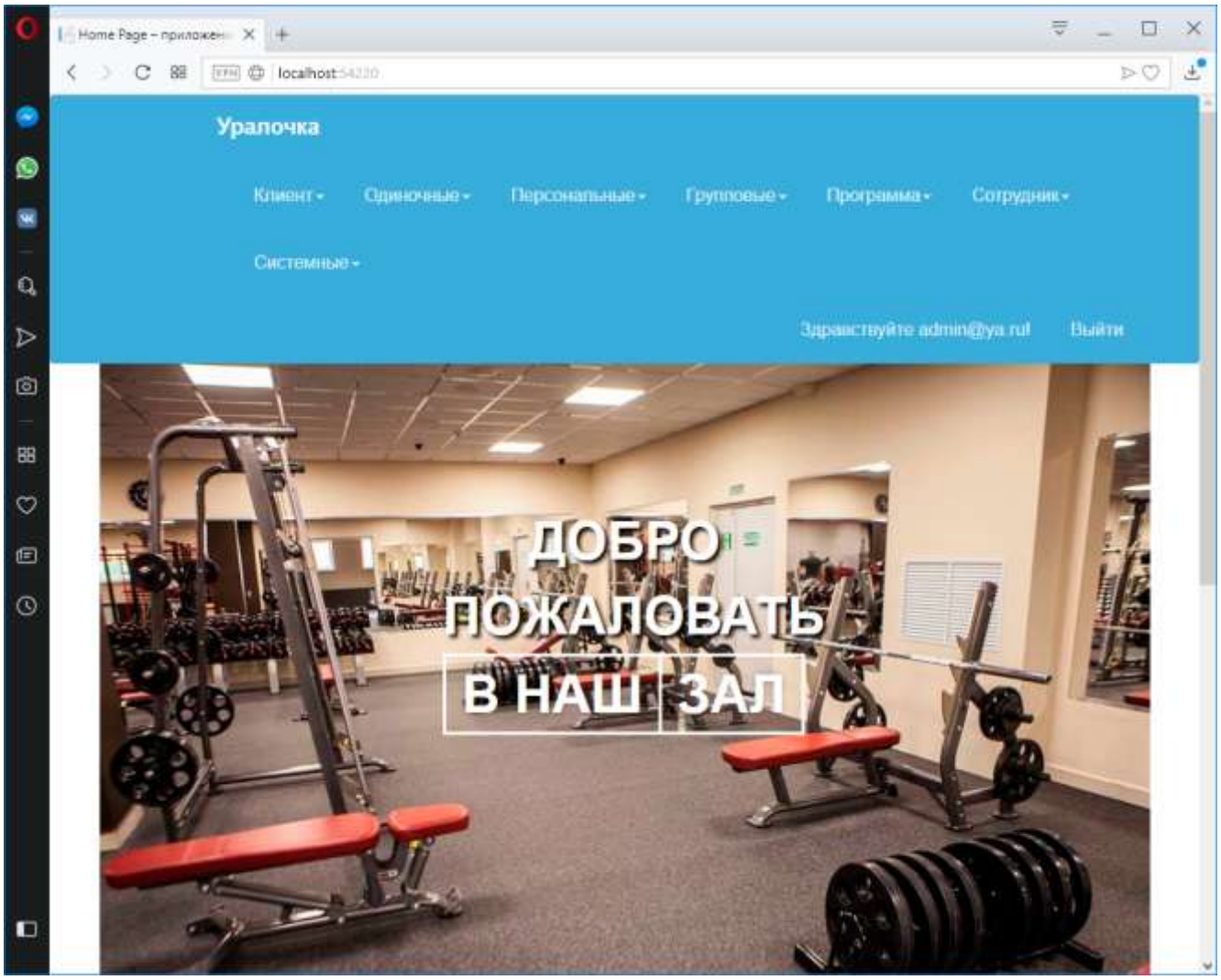

Рисунок 3.5 – Общий вид для администратора

Работа администратора с клиентом начинается с заведением клиента в базу данных. Интерфейс работы показан на рисунке 3.6 и 3.7.

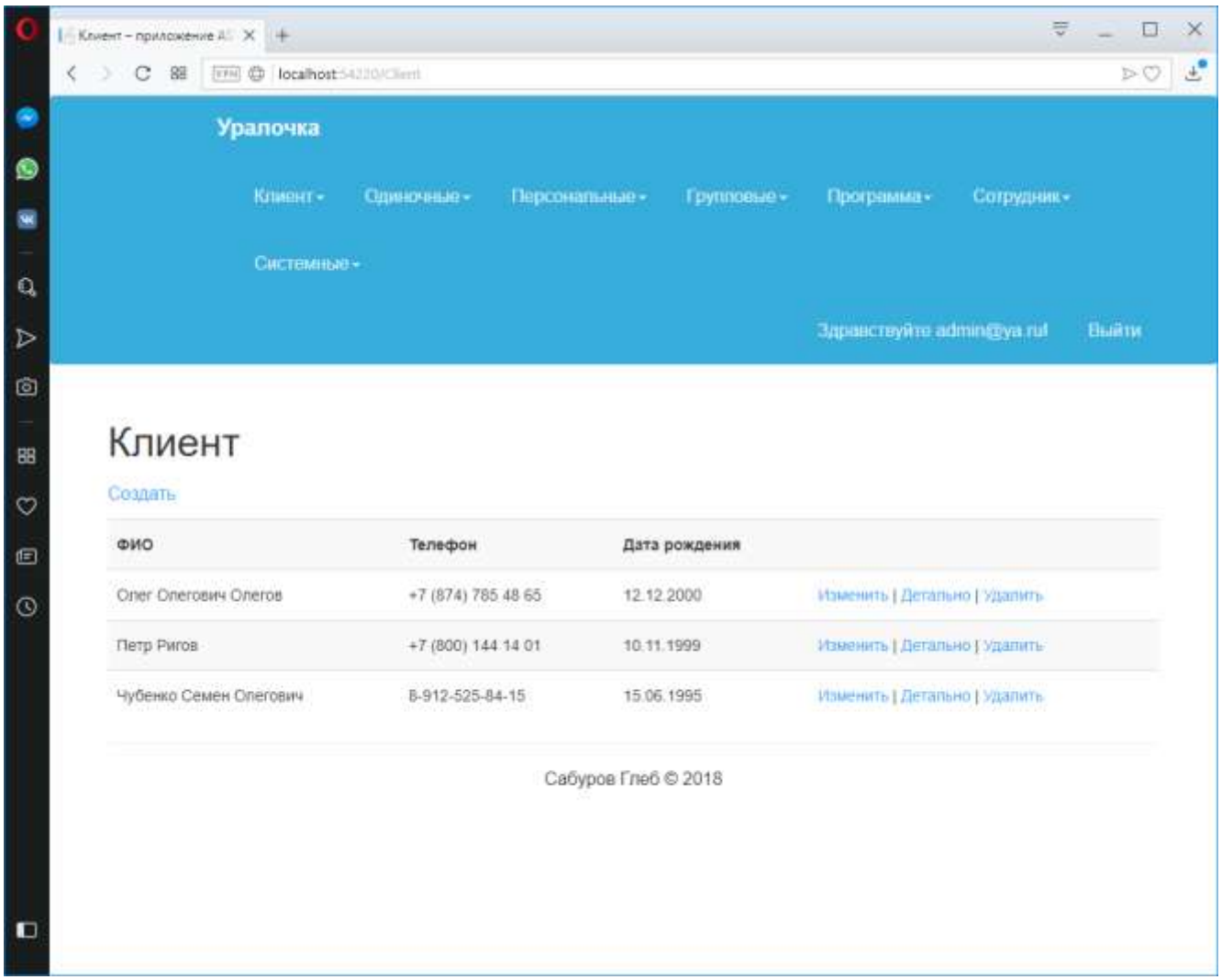

Рисунок 3.6 – Раздел меню клиент

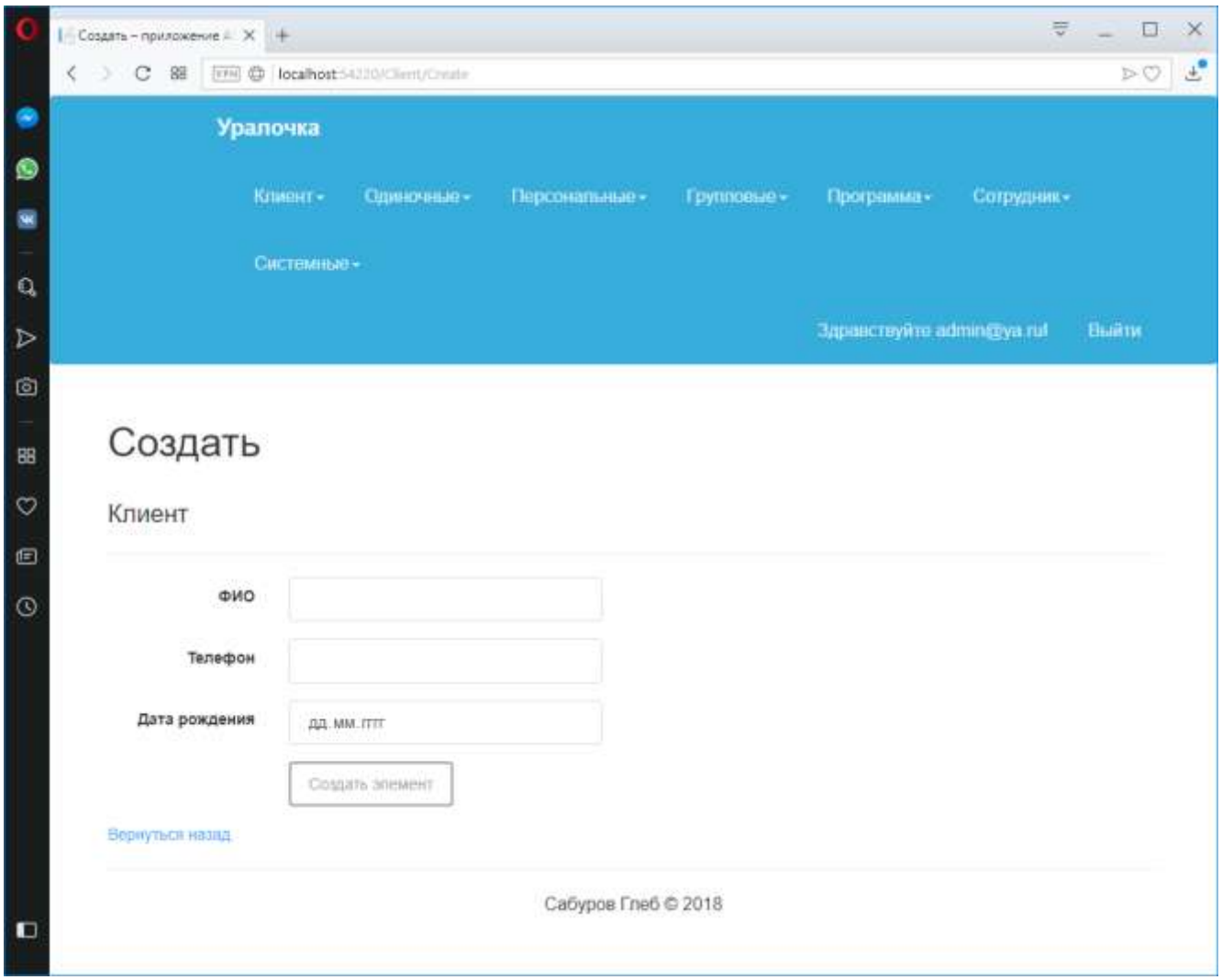

Рисунок 3.7 – Создание клиента

Код работы по просмотру списка клиентов показан в листинге 1, а код работы по созданию нового клиента представлен в листинге 2.

Листинг 1 –просмотр списка клиентов

```
@model IEnumerable<Gym.Models.Gym.Client>

@{
    ViewBag.Title = "Клиент";
}

<h2>Клиент</h2>

<p>
    @Html.ActionLink("Создать", "Create")
</p>
<table class="table table-striped">
    <tr>
```

```
            <th>
                @Html.DisplayNameFor(model => model.name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.phone)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.birthday)
            </th>
            <th></th>
        </tr>

@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.phone)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.birthday)
            </td>
            <td>
                @Html.ActionLink("Изменить", "Edit", new { id=item.id }) |
                @Html.ActionLink("Детально", "Details", new { id=item.id }) |
                @Html.ActionLink("Удалить", "Delete", new { id=item.id })
            </td>
        </tr>
}

       </table>
```

Листинг 2 – добавление клиента

```
@model Gym.Models.Gym.Client

@{
    ViewBag.Title = "Создать";
}

<h2>Создать</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Клиент</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.name, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.name, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.name, "", new { @class = "text-
danger" })
            </div>
        </div>
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.phone, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.phone, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.phone, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.birthday, htmlAttributes: new { @class = "control-
label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.birthday, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.birthday, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Создать элемент" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Вернуться назад", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

При добавлении клиента можно допустить ошибку, поэтому в приложении реализована возможность изменения уже созданного клиента. Визуальное представление показано на рисунке 3.8.

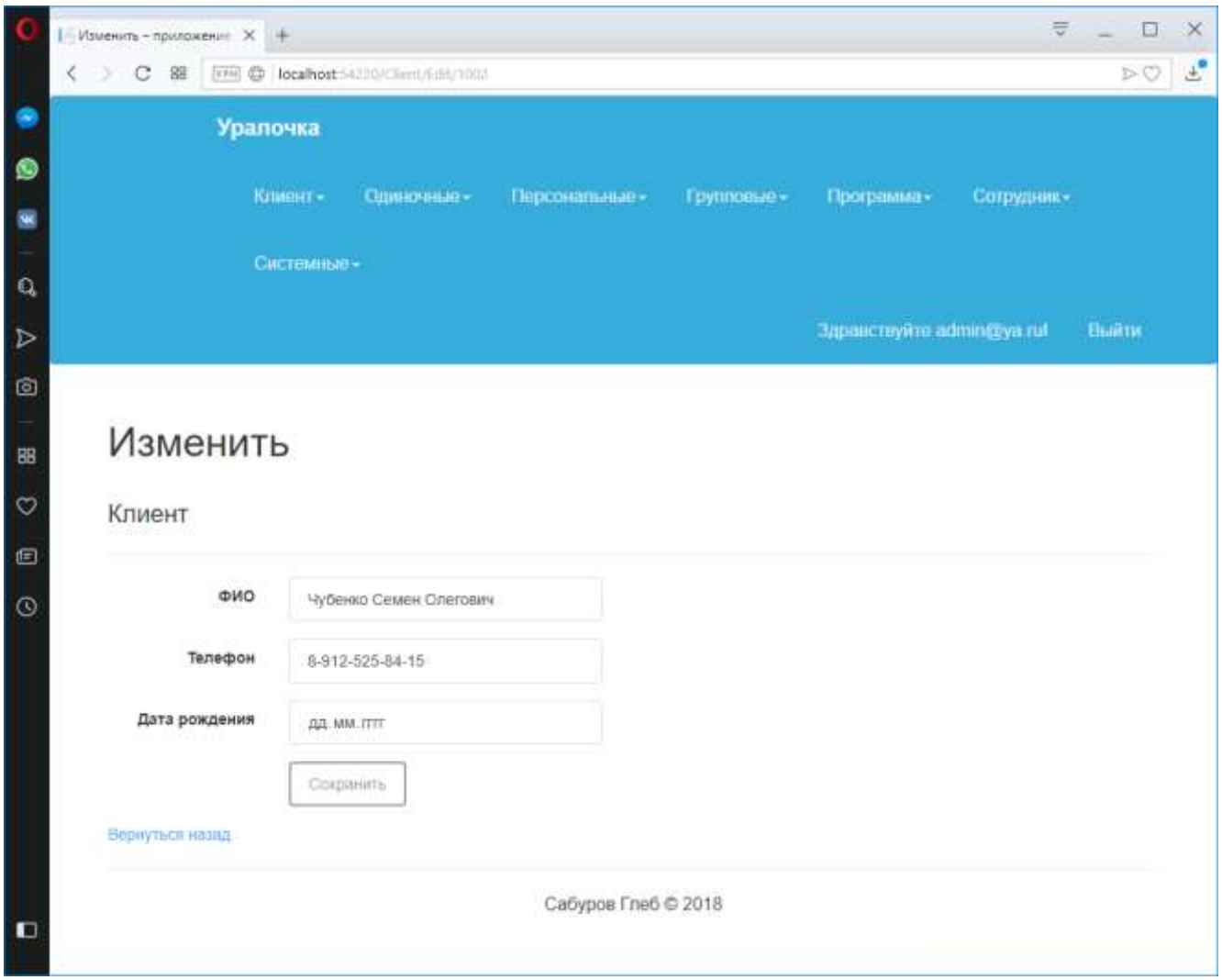

Рисунок 3.8 – Изменение клиента

Код представления изменение данных клиента приведен в листинге 3.

Листинг 3 – изменение данных клиента

```
@model Gym.Models.Gym.Client

@{
    ViewBag.Title = "Изменить";
}

<h2>Изменить</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Клиент</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
```

```
        @Html.HiddenFor(model => model.id)

        <div class="form-group">
            @Html.LabelFor(model => model.name, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.name, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.name, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.phone, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.phone, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.phone, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.birthday, htmlAttributes: new { @class = "control-
label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.birthday, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.birthday, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Сохранить" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Вернуться назад", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
        }
```

Также клиента можно удалить, если тот, например, перестал заниматься. Визуальное представление показано на рисунке 3.9. Код удаления клиента показан в листинге 4.

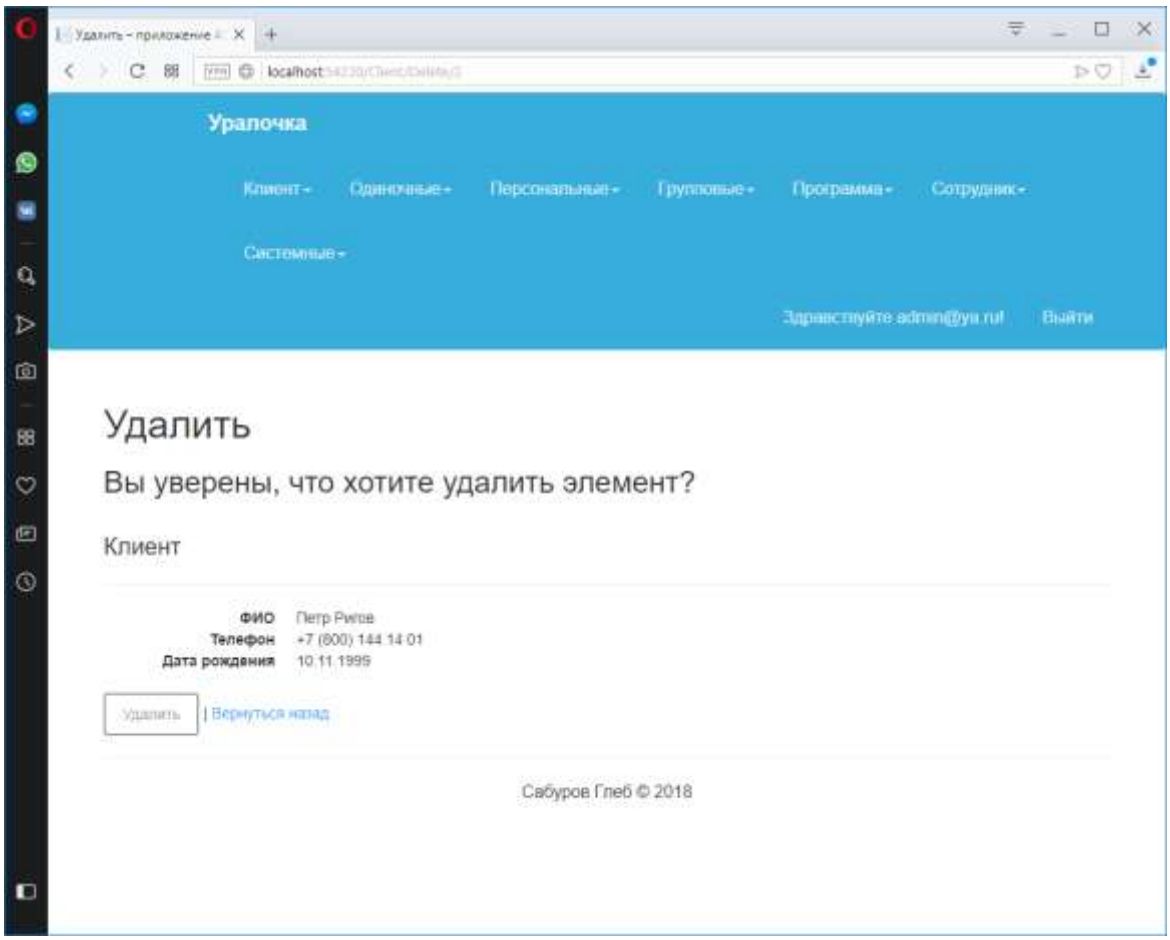

Рисунок 3.9 – Удаление клиента

Листинг 4 – Удаление клиента

```
@model Gym.Models.Gym.Client

@{
    ViewBag.Title = "Удалить";
}

<h2>Удалить</h2>

<h3>Вы уверены, что хотите удалить элемент?</h3>
<div>
    <h4>Клиент</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.phone)
        </dt>
```

```
        <dd>
            @Html.DisplayFor(model => model.phone)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.birthday)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.birthday)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Удалить" class="btn btn-default" /> |
            @Html.ActionLink("Вернуться назад", "Index")
        </div>
    }
</div>
```

Код контролера для управления представлениями представлен в листинге 5.

Листинг 5 – контролер для управления представлениями клиента

```
namespace Gym.Controllers
{
    public class ClientController : Controller
    {
        private GymModel db = new GymModel();

        // GET: Client
        public ActionResult Index()
        {
            return View(db.Client.ToList());
        }

        // GET: Client/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Client client = db.Client.Find(id);
            if (client == null)
            {
                return HttpNotFound();
            }
            return View(client);
        }
```

```
        // GET: Client/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Client/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойст-
ва, для которых следует установить привязку. Дополнительные
        // сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "id,name,phone,birthday")] Client client)
        {
            if (ModelState.IsValid)
            {
                db.Client.Add(client);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(client);
        }

        // GET: Client/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Client client = db.Client.Find(id);
            if (client == null)
            {
                return HttpNotFound();
            }
            return View(client);
        }

        // POST: Client/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойст-
ва, для которых следует установить привязку. Дополнительные
        // сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "id,name,phone,birthday")] Client client)
        {
            if (ModelState.IsValid)
            {
                db.Entry(client).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(client);
        }

        // GET: Client/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
```

```
            }
            Client client = db.Client.Find(id);
            if (client == null)
            {
                return HttpNotFound();
            }
            return View(client);
        }

        // POST: Client/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Client client = db.Client.Find(id);
            db.Client.Remove(client);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Помимо создания клиента в приложении представлены решения множества других задач таких как:

- Создание индивидуального абонемента для клиента, который может включать в себя множество услуг;
- Создание прайса для простых, персональных, групповых абонементов;
- Продажа клиентам простых, персональных, групповых абонементов;
- Создание графика персональных и групповых занятий;
- Возможность учета посещений клиентов;
- Составление программ тренировок;
- Создание списка сотрудников, назначения им должностей, составление графика работы.

Рассмотрим вариант решений задач с абонементами на примере ветки персональных абонементов. Первым пунктом в разделе является возможность просмотра прайса абонементов. Внешний вид представлен на рисунке 3.10

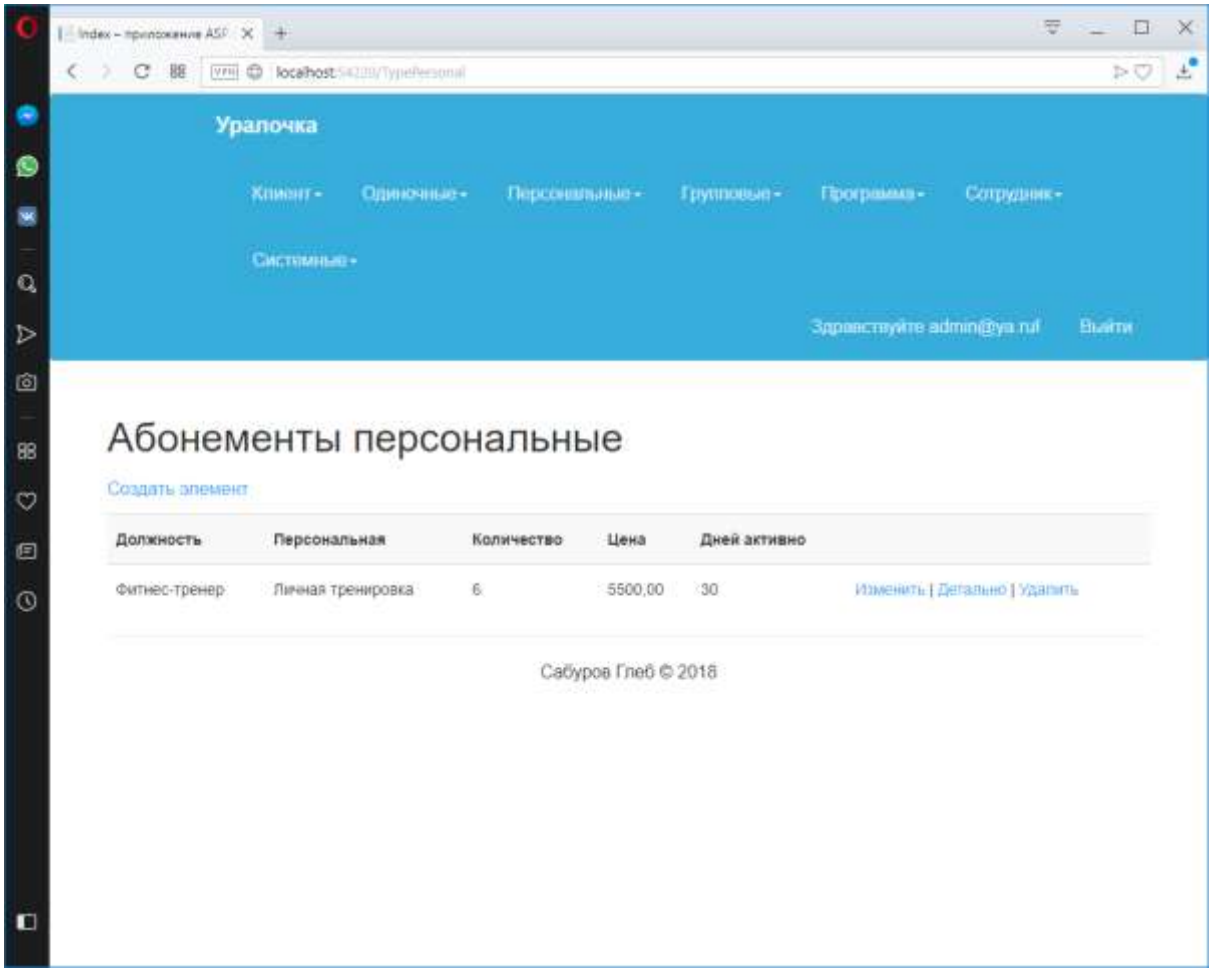

Рисунок 3.10 – Прайс абонементов

Код представления на просмотр прайса абонементов приведен в листинге 6.

Листинг 6 – просмотр прайса

```
@model IEnumerable<Gym.Models.Gym.Personal>

@{
    ViewBag.Title = "Index";
}

<h2>Персональные занятия</h2>

<p>
    @Html.ActionLink("Создать элемент", "Create")
```

```
</p>
<table class="table table-striped">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.subscription.number)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.type_personal.name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.date)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.subscription.number)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.type_personal.name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.date)
        </td>
        <td>
            @Html.ActionLink("Изменить", "Edit", new { id=item.id }) |
            @Html.ActionLink("Детально", "Details", new { id=item.id }) |
            @Html.ActionLink("Удалить", "Delete", new { id=item.id })
        </td>
    </tr>
}

</table>
```

Далее можно добавить в список новый вариант персональной тренировки, а также изменить или удалить выбранный по необходимости. Визуальное представление показано на рисунке 3.11, 3.12, 3.13.

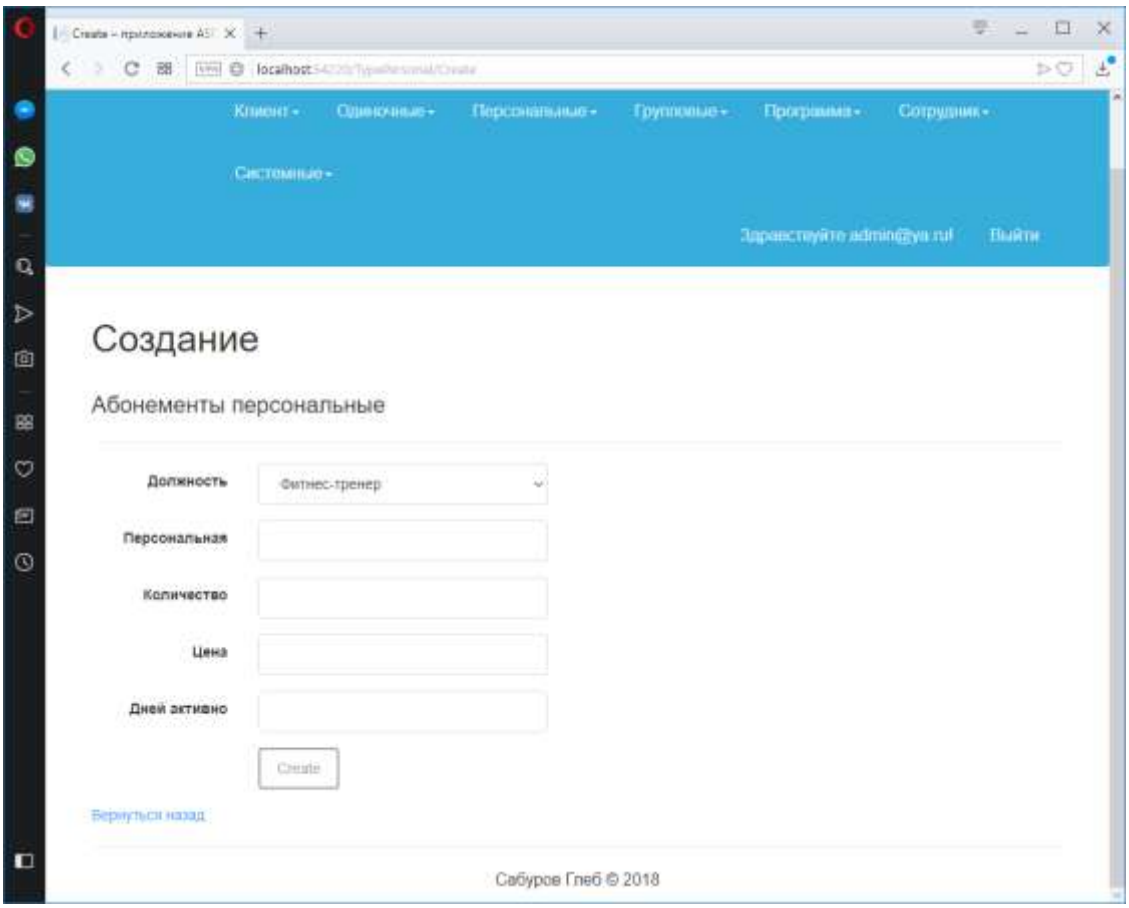

Рисунок 3.11 – Создание персонального абонемента

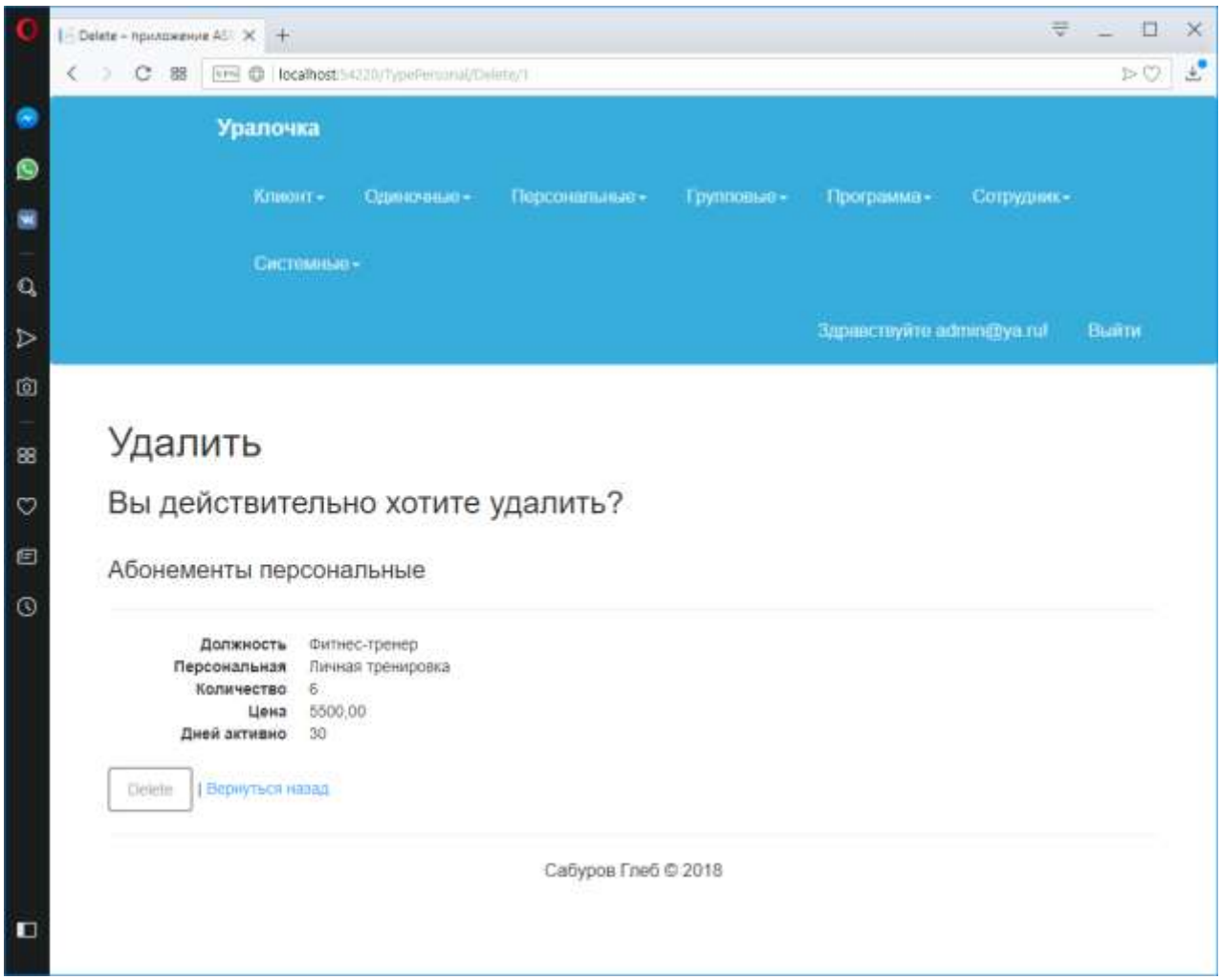

Рисунок 3.12 – Удаление персонального абонемента

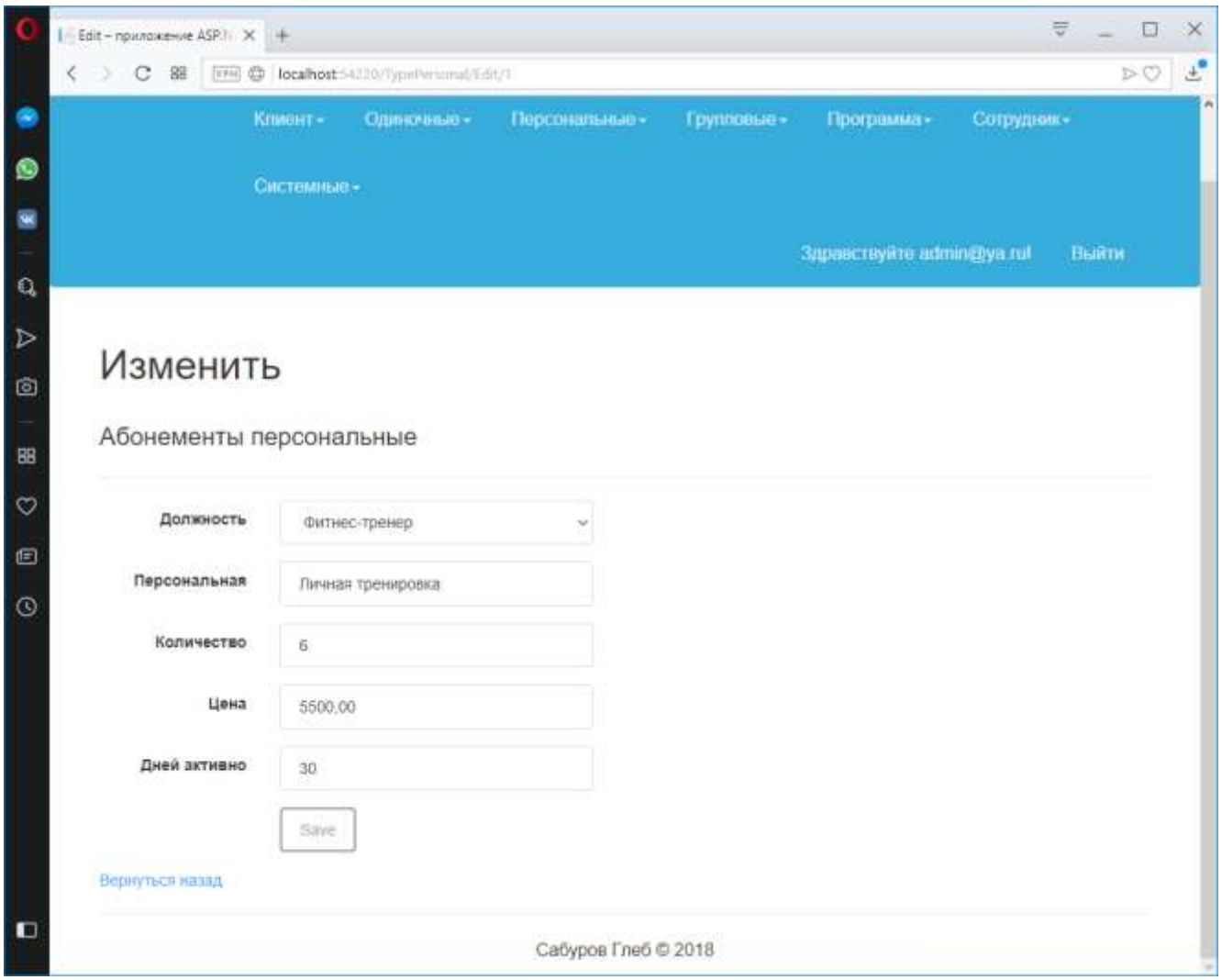

Рисунок 3.13 – Изменение персонального абонемента

Код представления на создание персонального абонемента приведен в листинге 7.

Листинг 7 – создание абонемента

```
@model Gym.Models.Gym.Personal

@{
    ViewBag.Title = "Create";
}

<h2>Создание</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
```

```
        <h4>Персональный</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            <label class="control-label col-md-2" for="id_subscription">Абонемент
клиента</label>
            <div class="col-md-10">
                @Html.DropDownList("id_subscription", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.id_subscription, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.id_type_personal, "id_type_personal",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("id_type_personal", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.id_type_personal, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <label class="control-label col-md-2" for="id_employee">Сотрудник</label>
            <div class="col-md-10">
                @Html.DropDownList("id_employee", null, htmlAttributes: new { @class = "form-
control" })
                @Html.ValidationMessageFor(model => model.id_employee, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.date, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.date, new { htmlAttributes = new { @class =
"form-control" } })
                @Html.ValidationMessageFor(model => model.date, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Вернуться назад", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Код представления на удаление персонального абонемента приведен в листинге 8.

Листинг 8 – создание абонемента

```
@model Gym.Models.Gym.Personal

@{
    ViewBag.Title = "Delete";
}

<h2>Удалить</h2>

<h3>Вы действительно хотите удалить?</h3>
<div>
    <h4>Персональный</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.subscription.number)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.subscription.number)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.type_personal.name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.type_personal.name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.id_employee)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.id_employee)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.date)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.date)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Вернуться назад", "Index")
        </div>
    }
</div>
```

Код представления на изменение персонального абонемента приведен в листинге 9.

Листинг 9 – создание абонемента

```
@model Gym.Models.Gym.Personal

@{
    ViewBag.Title = "Edit";
}

<h2>Изменить</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Персональный</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.id)

        <div class="form-group">
            <label class="control-label col-md-2" for="id_subscription">Абонемент
клиента</label>
            <div class="col-md-10">
                @Html.DropDownList("id_subscription", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.id_subscription, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.id_type_personal, "id_type_personal",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("id_type_personal", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.id_type_personal, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <label class="control-label col-md-2" for="id_employee">Сотрудник</label>
            <div class="col-md-10">
                @Html.EditorFor(model => model.id_employee, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.id_employee, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.date, htmlAttributes: new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.date, new { htmlAttributes = new { @class =
"form-control" } })
```

```
                @Html.ValidationMessageFor(model => model.date, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Вернуться назад", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
```

Контролер для управления представлениям в разделе прайса персональных абонементов представлен в листинге 10.

Листинг 10 – контролер для управления

```
namespace Gym.Controllers
{
    public class PersonalController : Controller
    {
        private GymModel db = new GymModel();

        // GET: Personal
        public ActionResult Index()
        {
            var personal = db.Personal.Include(p => p.subscription).Include(p =>
p.type_personal).Include(p => p.employee);

            if (User.IsInRole("Admin"))
            {
                return View("IndexLogin", personal);
            }

            var id_user = Int16.Parse(User.Identity.GetUserId());
            var id_subscription = db.Subscription.Where(x => x.id_client ==
id_user).First().id;
            var listPersonal = personal.Where(x => x.id_subscription ==
id_subscription).ToList();

            return View(listPersonal);

        }

        // GET: Personal/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            var id_user = Int16.Parse(User.Identity.GetUserId());
```

```
            ViewData["subscription"] = db.Subscription.Where(x => x.id_client ==
id_user).First().number;
            Personal personal = db.Personal.Find(id);

            Session["id_personal"] = personal.id;
            if (personal == null)
            {
                return HttpNotFound();
            }
            return View(personal);
        }

        public ActionResult _Visit()
        {
            int id_personal = (int)Session["id_personal"];
            var visit = db.VisitPersonal.Where(x => x.id_personal == id_personal).ToList();
            return PartialView(visit);
        }

        // GET: Personal/Create
        public ActionResult Create()
        {
            ViewBag.id_subscription = new SelectList(db.Subscription, "id", "number");
            ViewBag.id_type_personal = new SelectList(db.TypePersonal, "id", "name");
            ViewBag.id_employee = new SelectList(db.Employee, "id", "name");
            return View();
        }

        // POST: Personal/Create
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойст-
ва, для которых следует установить привязку. Дополнительные
        // сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include =
"id,id_subscription,id_type_personal,id_employee,date")] Personal personal)
        {
            if (ModelState.IsValid)
            {
                db.Personal.Add(personal);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.id_subscription = new SelectList(db.Subscription, "id", "number", person-
al.id_subscription);
            ViewBag.id_type_personal = new SelectList(db.TypePersonal, "id", "name", person-
al.id_type_personal);
            return View(personal);
        }

        // GET: Personal/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Personal personal = db.Personal.Find(id);
            if (personal == null)
            {
                return HttpNotFound();
            }
```

```
            ViewBag.id_subscription = new SelectList(db.Subscription, "id", "number", person-
al.id_subscription);
            ViewBag.id_type_personal = new SelectList(db.TypePersonal, "id", "name", person-
al.id_type_personal);
            return View(personal);
        }

        // POST: Personal/Edit/5
        // Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойст-
ва, для которых следует установить привязку. Дополнительные
        // сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"id,id_subscription,id_type_personal,id_employee,date")] Personal personal)
        {
            if (ModelState.IsValid)
            {
                db.Entry(personal).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.id_subscription = new SelectList(db.Subscription, "id", "number", person-
al.id_subscription);
            ViewBag.id_type_personal = new SelectList(db.TypePersonal, "id", "name", person-
al.id_type_personal);
            return View(personal);
        }

        // GET: Personal/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Personal personal = db.Personal.Find(id);
            if (personal == null)
            {
                return HttpNotFound();
            }
            return View(personal);
        }

        // POST: Personal/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Personal personal = db.Personal.Find(id);
            db.Personal.Remove(personal);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
```

```
    }
}
```

Следующим разделом в ветке является оформление покупки персонального занятия. Визуальное представление показано на рисунке 3.14.

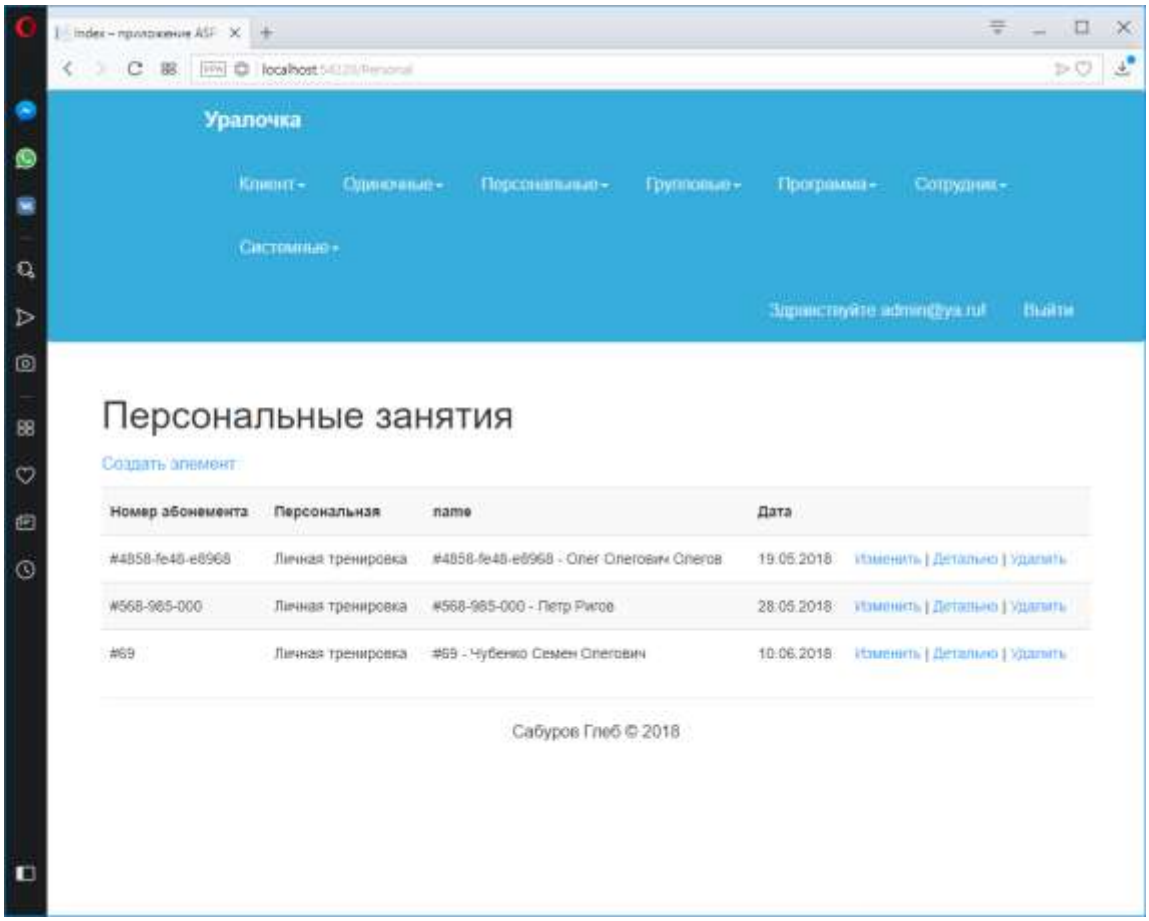

Рисунок 3.14 – Покупка персонального занятия

В этом разделе также реализованы возможности создания, добавления, изменения персонального занятия.

Следующим разделом в ветке является график для посещения клиента. Визуальное представление показано на рисунке 3.15.

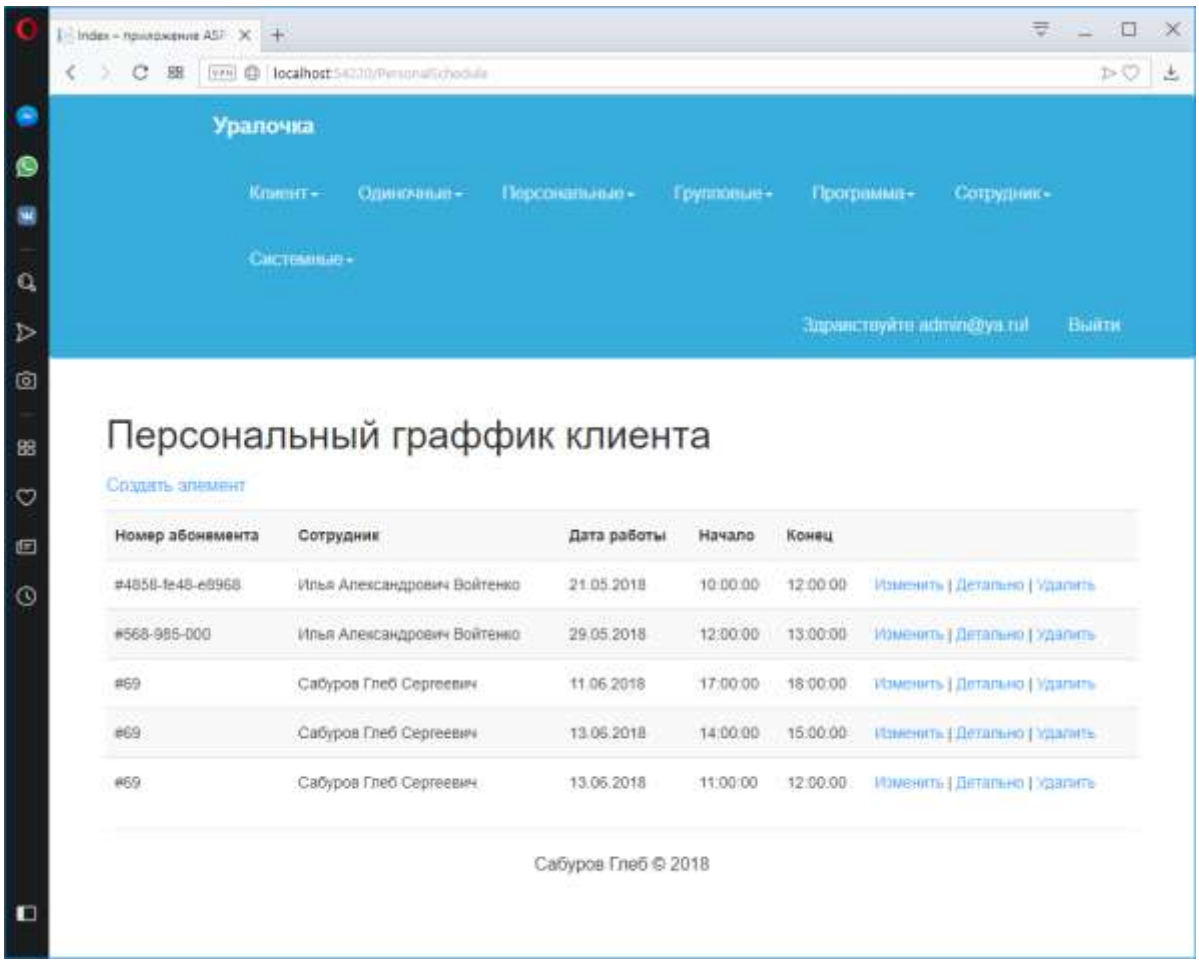

Рисунок 3.15 –Персональный график клиента

В этом разделе реализованы возможности создания, добавления, изменения персонального занятия.

Следующим разделом в ветке является отметка посещений клиента. Визуальное представление показано на рисунке 3.16.

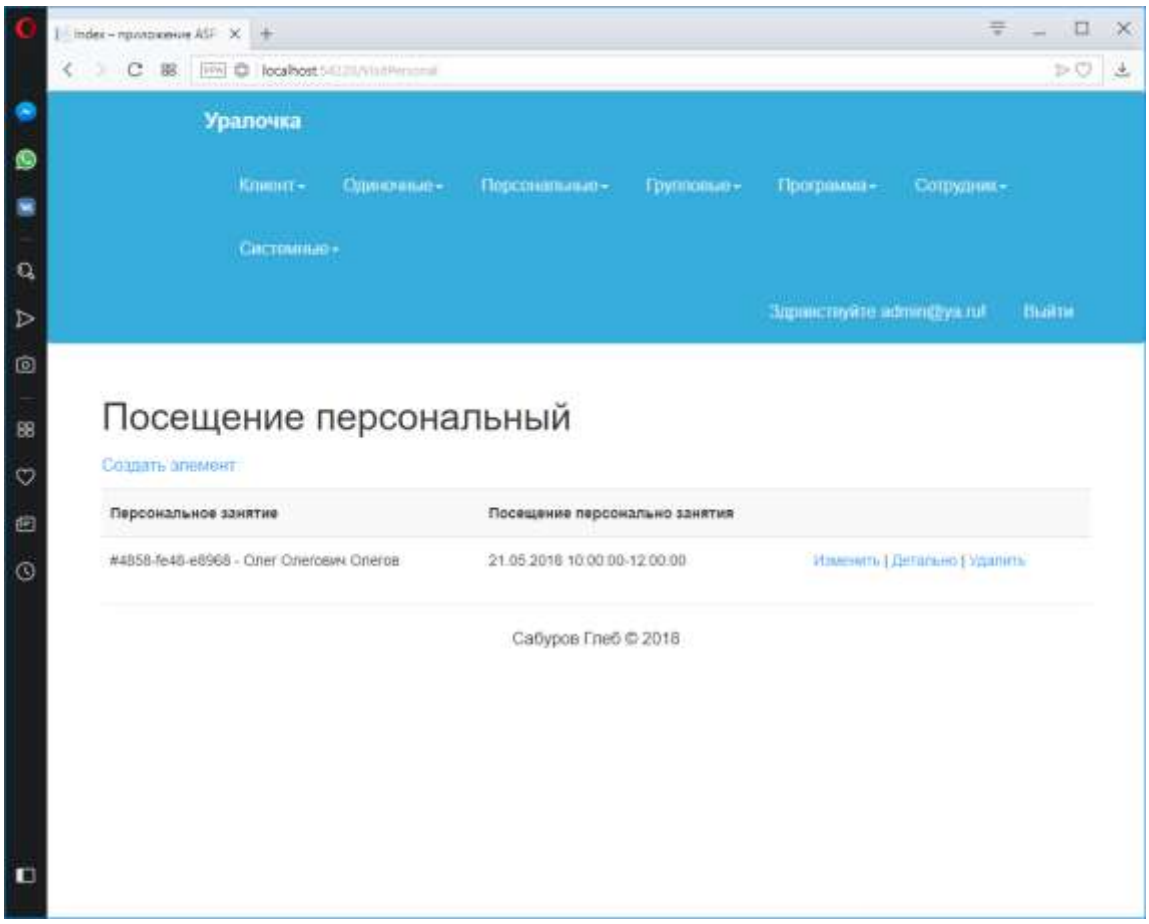

Рисунок 3.16 –Посещение клиента

Интересным разделом в работе является программа тренировок. Визуальное представление представлено на рисунках 3.17 и 3.18.

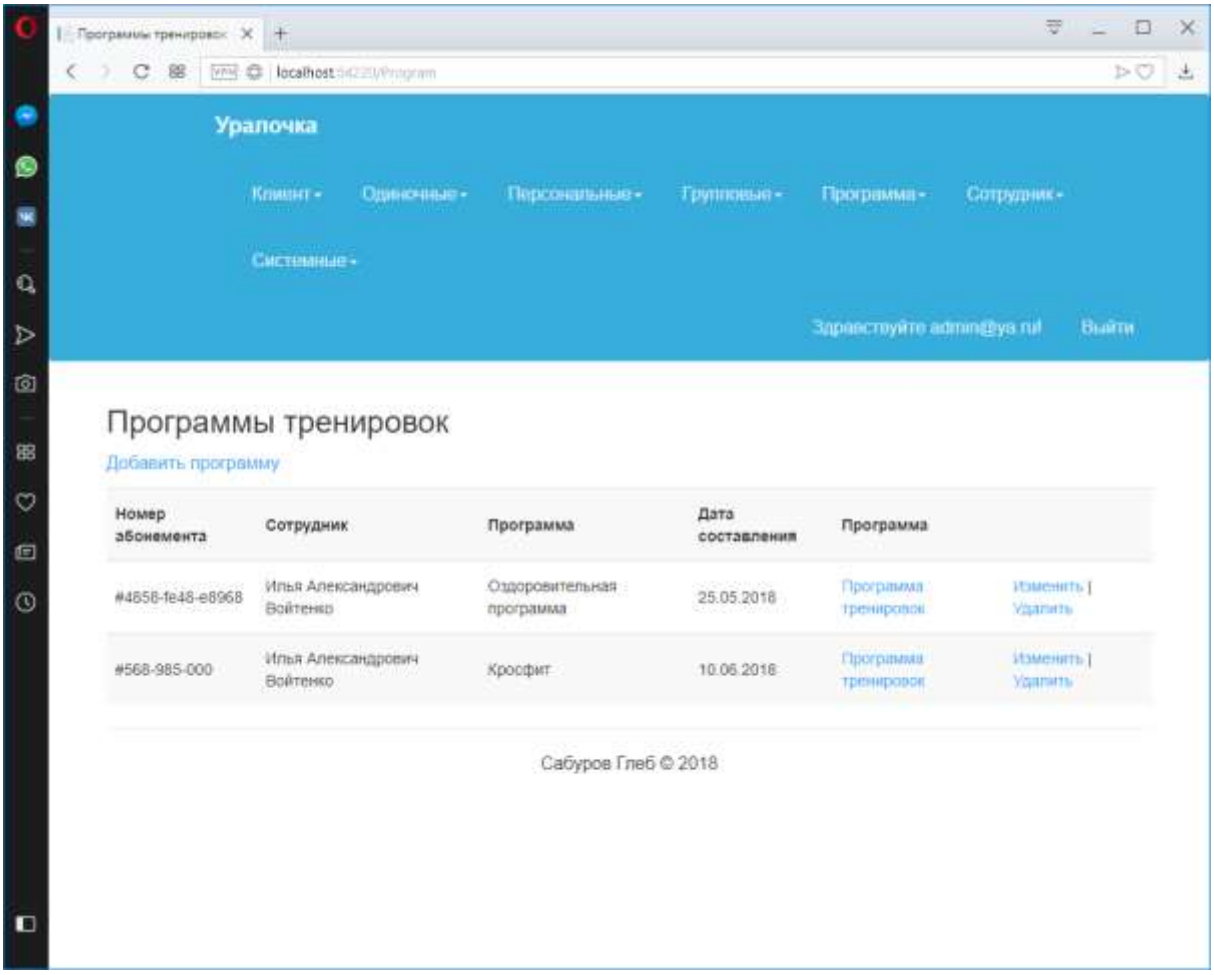

Рисунок 3.17 – Список программ тренировок

При переходе в раздел Программа тренировок, администратор совместно с тренером может разработать индивидуальную программу тренировки для каждого клиента, наполнить ее различными упражнениями и другой необходимой информацией.

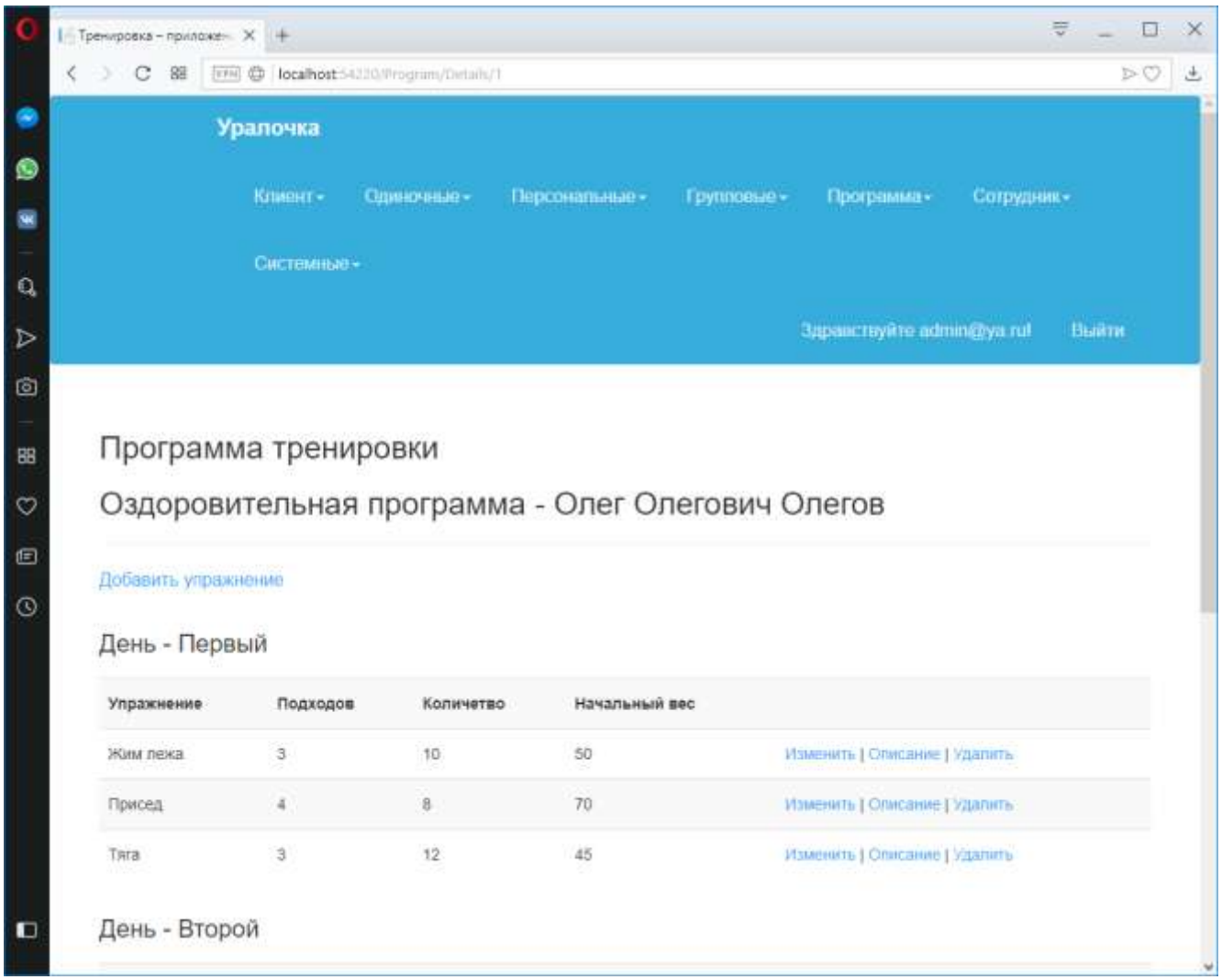

Рисунок 3.18 – Программа тренировки

Код представления на программу тренировки приведен в листинге 11.

Листинг 11 – программа тренировок

```
@model IEnumerable<Gym.Models.Gym.Training>

@{
    ViewBag.Title = "Тренировка";

}
<br />
<h3>Программа тренировки</h3>
<h3>@ViewData["program"] - @ViewData["client"]</h3>
<hr />

<p>
    @Html.ActionLink("Добавить упражнение", "Create", "Training", new {  id =
(int)ViewData["id_program"] }, null)
</p>

@foreach (var day in (List<Gym.Models.Gym.Day>)ViewData["Day"])
{
```

```
    <h4>День - @day.name</h4>

    <table class="table table-striped">
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.exercise.name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.approach)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.quantity)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.initial_weight)
            </th>
            <th></th>
        </tr>

        @foreach (var item in Model)
        {
            if (item.id_day == day.id)
            {
                <tr>
                    <td>
                        @Html.DisplayFor(modelItem => item.exercise.name)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.approach)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.quantity)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.initial_weight)
                    </td>
                    <td>
                        @Html.ActionLink("Изменить", "Edit", "Training", new { id = item.id },
null) |
                        @Html.ActionLink("Описание", "Details", "Training", new { id = item.id
}, null) |
                        @Html.ActionLink("Удалить", "Delete", "Training", new { id = item.id
}, null)
                    </td>
                </tr>
            }
        }

    </table>
}


    <p>
        @Html.ActionLink("Вернуться назад", "Index")
    </p>
```

Следующим важным разделом является просмотр расписания, в котором тренер может увидеть, когда и на какое время у него стоят персональные тренировки. В разделе меню Сотрудник выбираем вкладку сотрудники. Визуальное представление показано на рисунке 3.19.

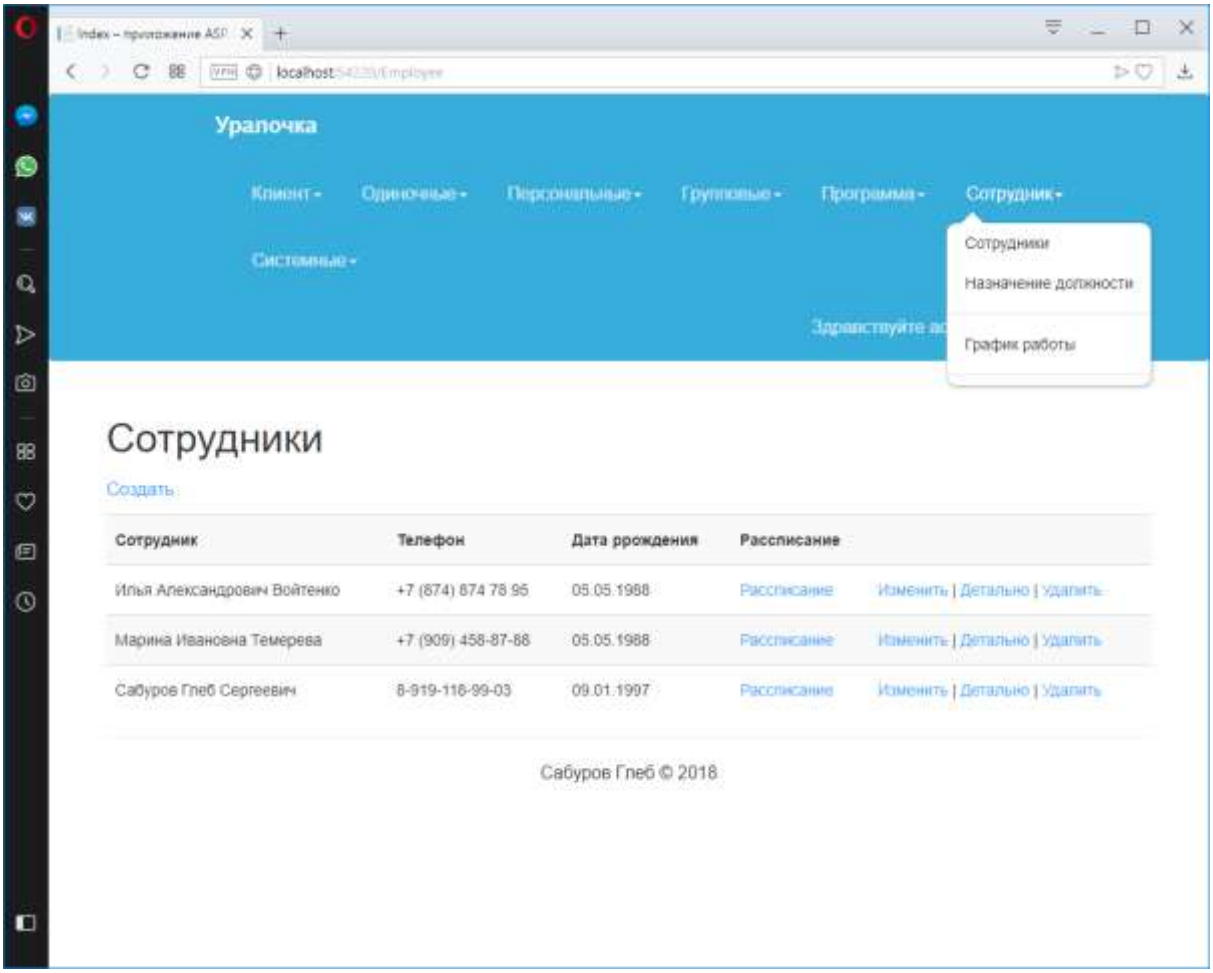

Рисунок 3.19 – Сотрудники

В этом разделе тренер имеет возможность открыть и просмотреть свое расписание. Визуальное представление показано на рисунке 3.20.

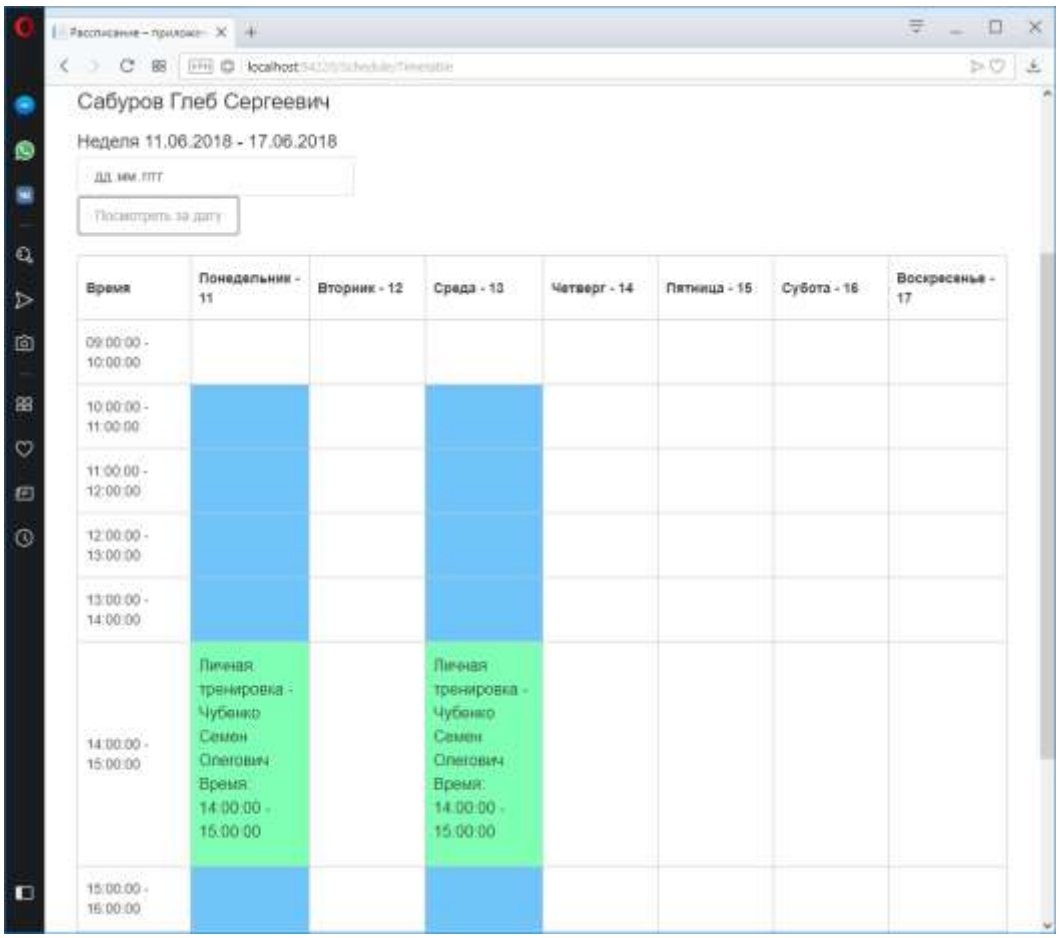

Рисунок 3.20 – График занятий

Код представления на график тренировок приведен в листинге 12.

Листинг 12 – график тренировок

```
@model Gym.Models.Timetable

@{
    ViewBag.Title = "Расписание";
}

<p class="p-timetable">@ViewData["employee"]</p>

@using (Html.BeginForm())
{
    <span class="p-timetable-18">Неделя @ViewData["timeStart"] - @ViewData["timeEnd"]  </span>
    <input class="form-control datetime-local text-box single-line" data-val="true" data-val-
date="Поле Дата работы должно содержать дату." data-val-required="Требуется поле Дата работы."
id="date" name="date" type="date" value="">
    <input type="submit" value="Посмотреть за дату" class="btn btn-default btn-timetable" />
}

    <br />

    <table class="timetable-table table th-120 ">
        <tr>
            <th class="th-150">
                Время
```

```
            </th>
            <th>
                Понедельник - @ViewData["day0"]
            </th>
            <th>
                Вторник - @ViewData["day1"]
            </th>
            <th>
                Среда - @ViewData["day2"]
            </th>
            <th>
                Четверг - @ViewData["day3"]
            </th>
            <th>
                Пятница - @ViewData["day4"]
            </th>
            <th>
                Суббота - @ViewData["day5"]
            </th>
            <th>
                Воскресенье - @ViewData["day6"]
            </th>
        </tr>

        @foreach (var item in Model.Time)
        {
            <tr>
                <td>@item.time_c  - <br /> @item.time_po </td>


                @for (int i = 0;i<=6;i++)
                {
                    if (Model.mas[i, item.id] == 1)
                    {
                        <td style="background-color : #6fc4f9">
                            <p>@Model.str[i, item.id] </p>
                        </td>
                    }
                    else if (Model.mas[i, item.id] == 2)
                    {
                        <td style="background-color : #7fffb4">
                            <p>@Model.str[i, item.id] </p>
                        </td>
                    }
                    else
                    {
                        <td>    </td>
                    }
                }


            </tr>
        }

    </table>

    @section Scripts {
        @Scripts.Render("~/bundles/jqueryval")
    }
```

3.2 Возможности приложения для администратора

При авторизации пользователя под клиентом, видны следующий элементы управления, представленные на рисунке 3.21.

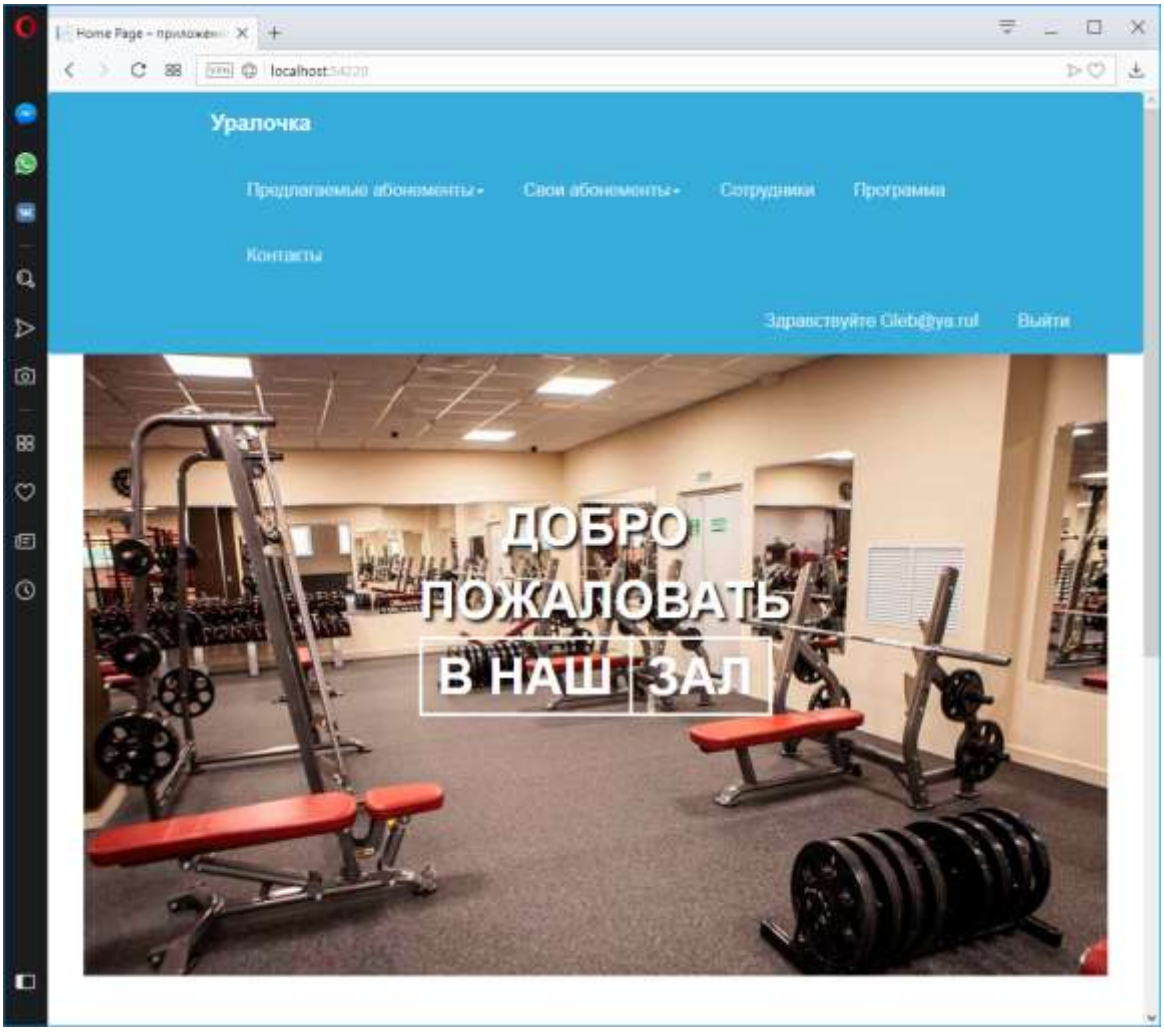

Рисунок 3.21 – Меню клиента

В разделе клиента, пользователь имеет возможности:

- просматривать предлагаемые услуги;
- просмотр информации о своем абонементе;
- просмотр информации о сотрудниках;
- просмотр своей программы тренировок;
- просмотр контактов фитнес клуба.

При выборе раздела персонального абонемента, пользователь может узнать всю необходимую информацию, визуальное представление показано на рисунке 3.21

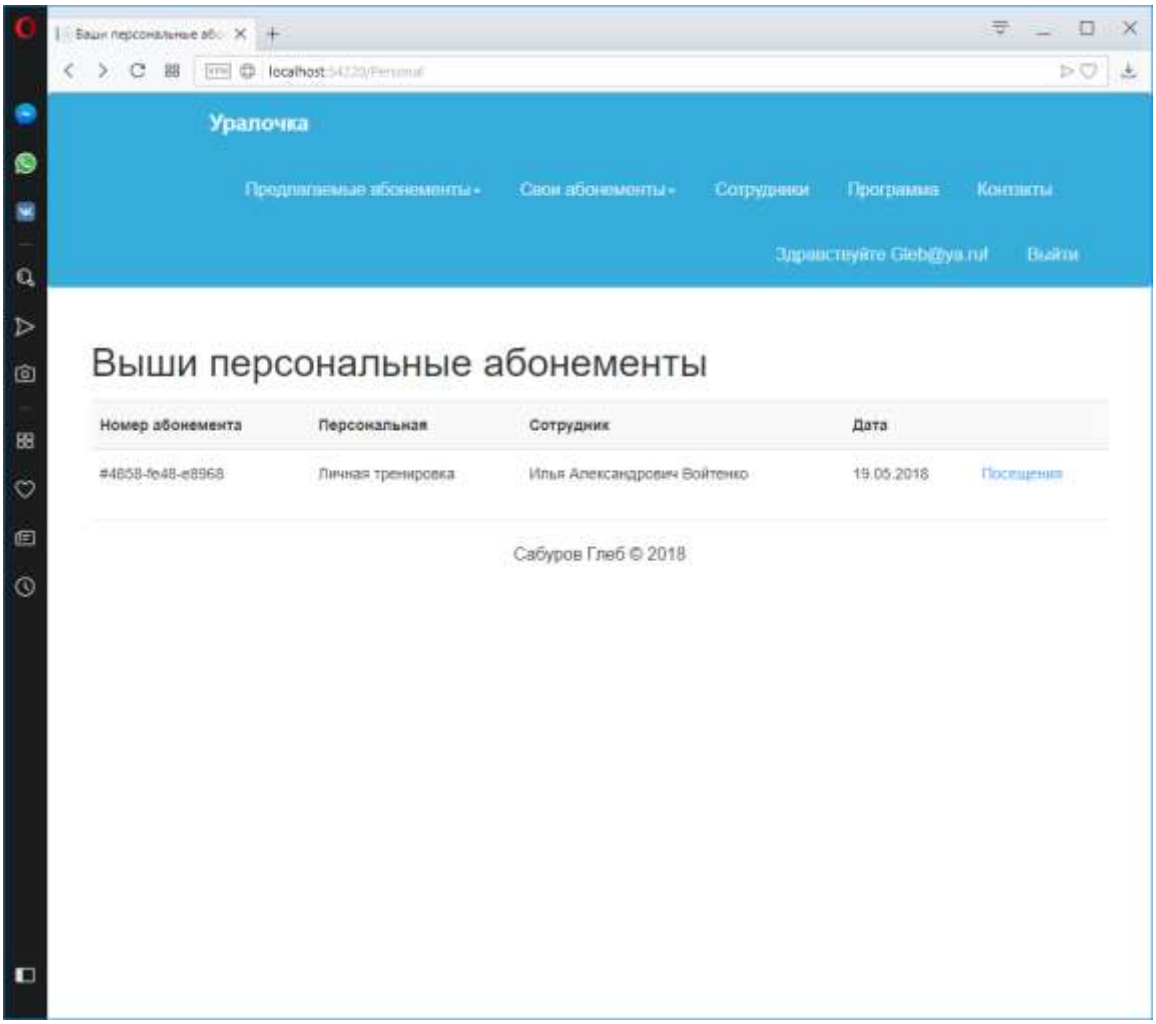

Рисунок 3.21 – Персональный абонемент

При переходе в раздел посещений, пользователь может увидеть все свои посещения в фитнес клубе, визуальное представление показано на рисунке 3.22

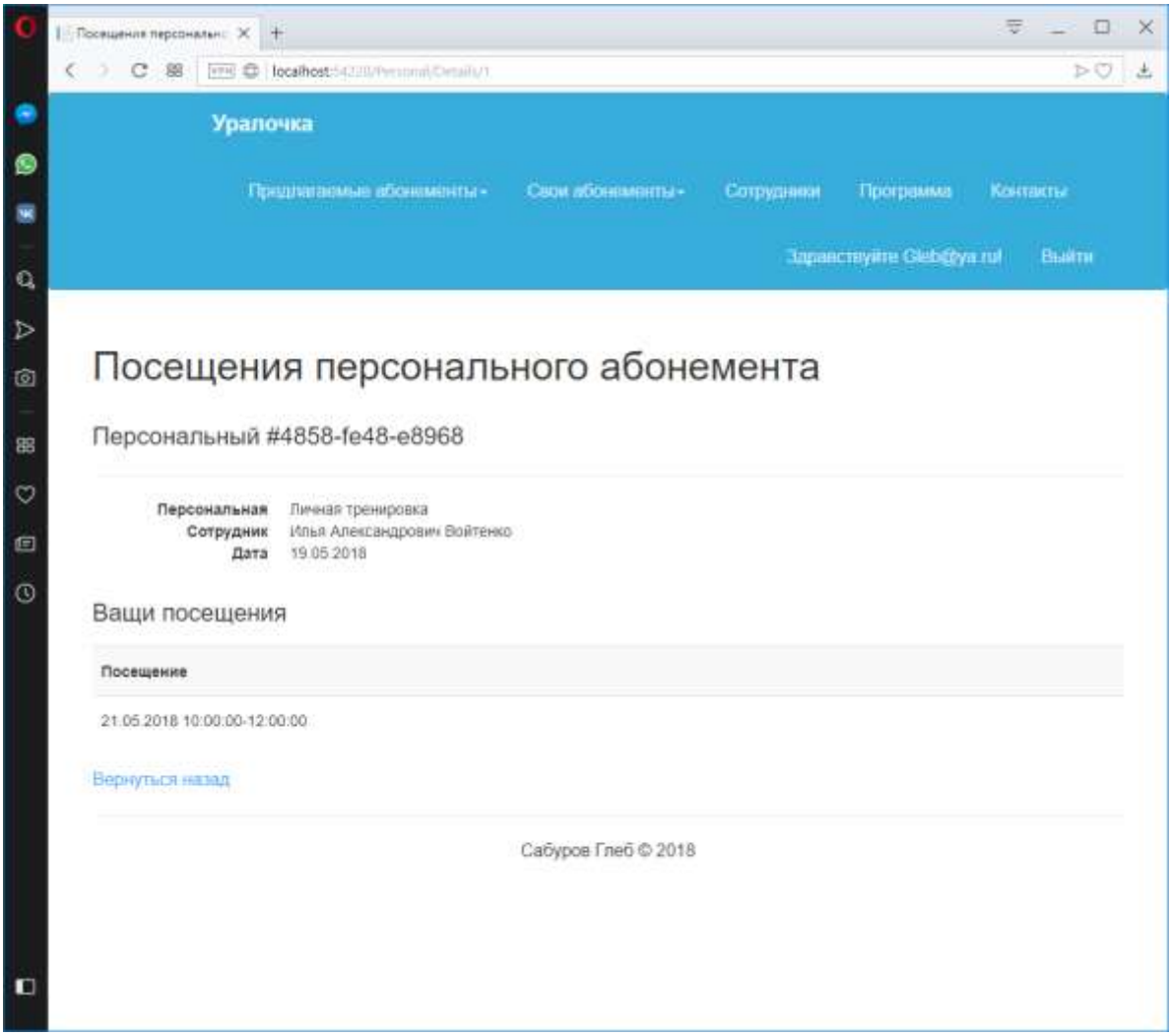

Рисунок 3.22 – Посещение клиента

При выборе раздела программ тренировок, пользователь может открыть список, созданных для него тренировок, визуальное представление показано на рисунке 3.23

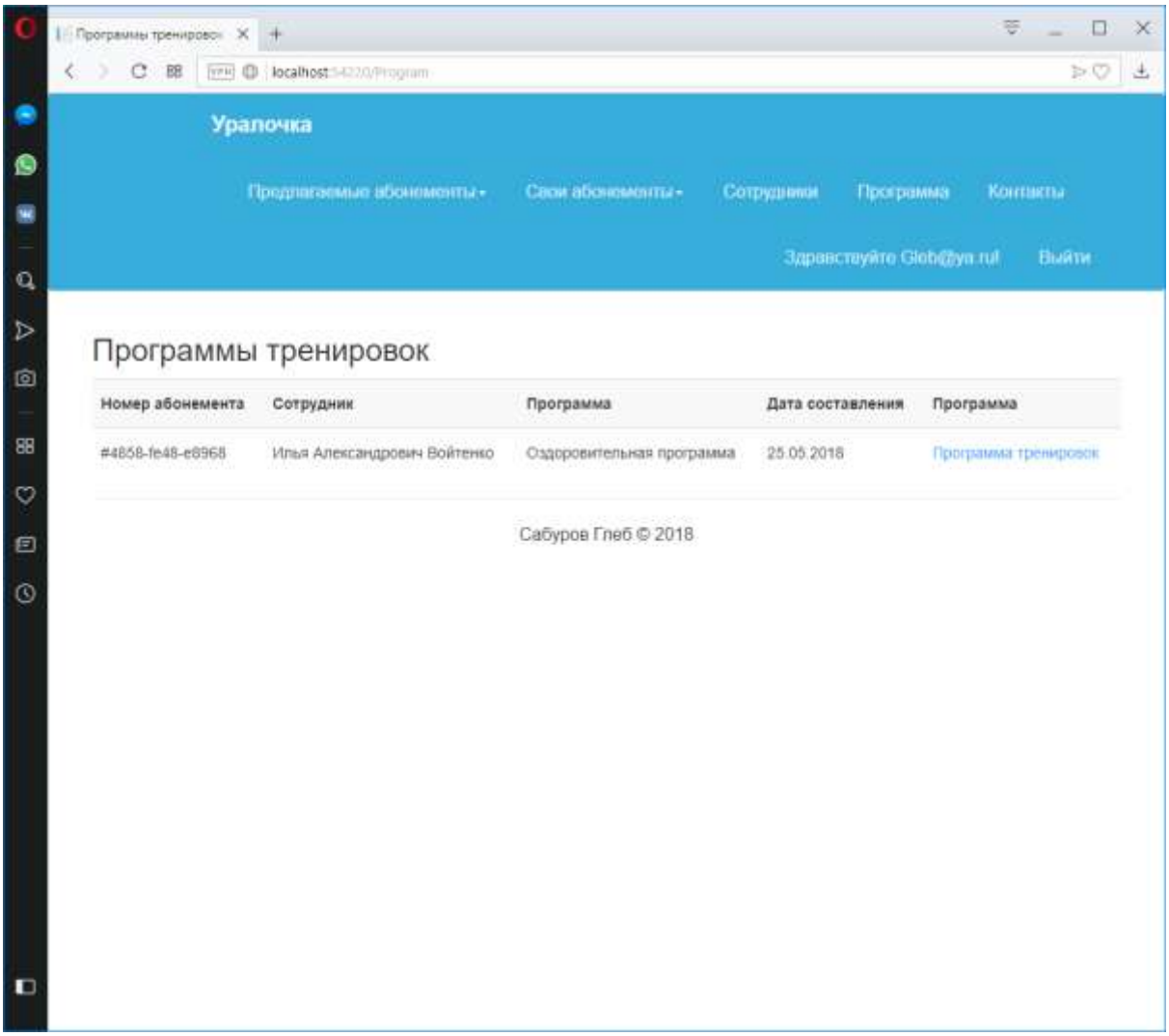

Рисунок 3.22 – Посещение клиента

При переходе в раздел программа тренировок, пользователь может просмотреть нужную тренировку, визуальное представление показано на рисунке 3.23.

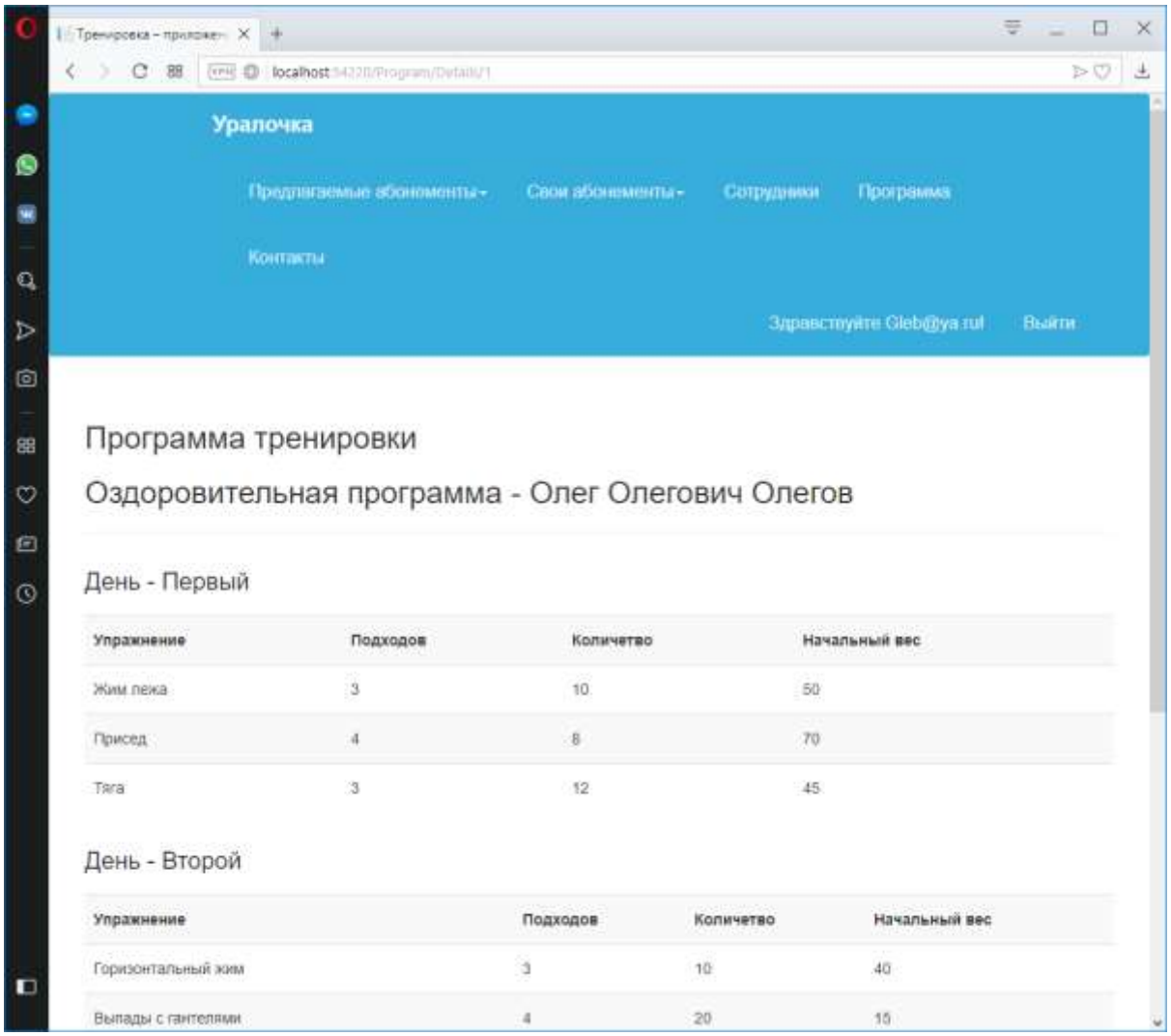

Рисунок 3.23 – Посещение клиента

При выборе раздела контакты, пользователь может полную информацию о местонахождении, времени работы, телефоне и другой информации, визуальное представление показано на рисунке 3.24

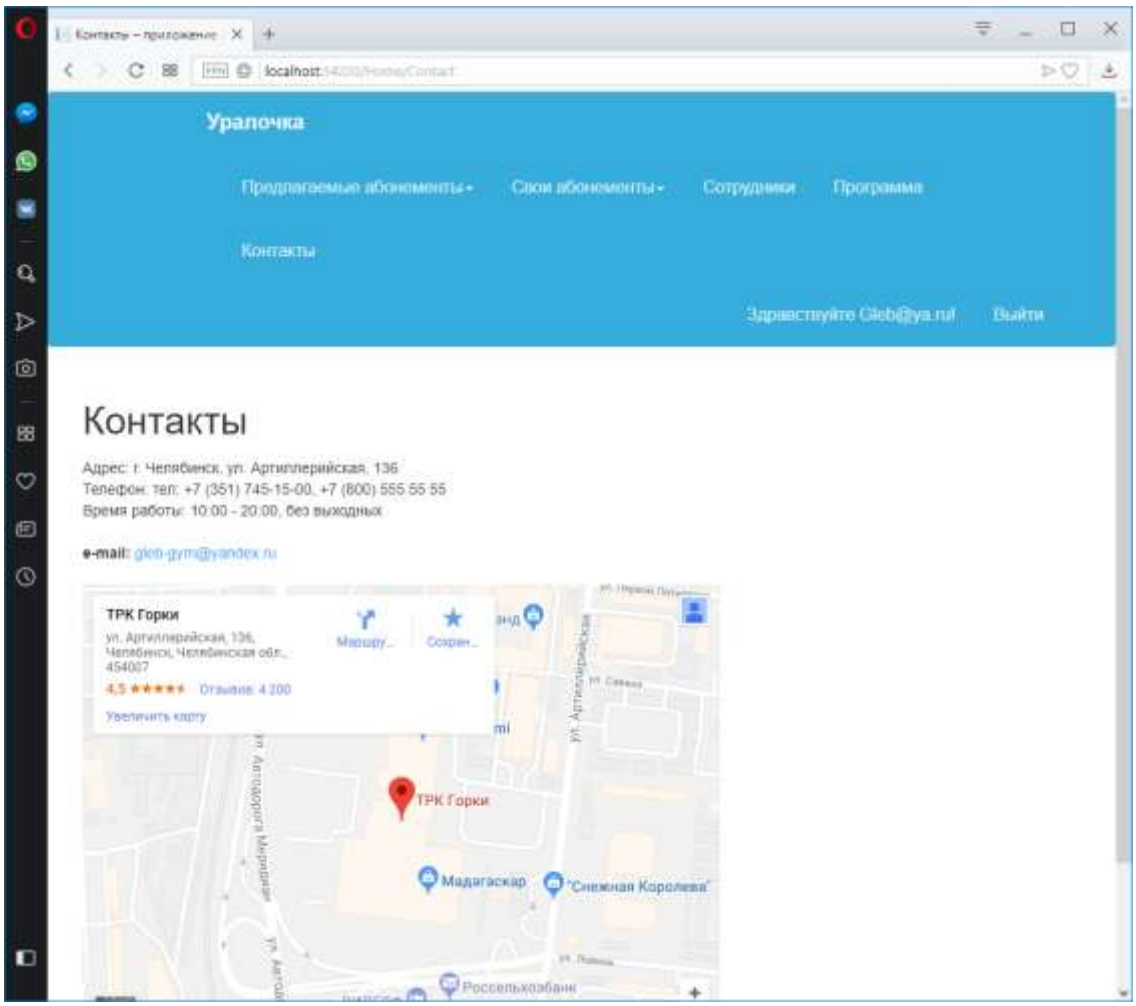

Рисунок 3.24 – Посещение клиента

Выводы по разделу три

В данном разделе были выполнены следующие задачи:

- представлен общий вид приложения;
- представлены возможности работы администратора;
- представлены возможности работы клиента.

ЗАКЛЮЧЕНИЕ

Целью данной работы было закрепление теоретических знаний, на практике, с помощью разработки Web - приложения средствами ASP.NET. Подводя итог проведенной работе, можно отметить, что приложение соответствует всем требованиям задания.

На основе данной разработки были усовершенствованы навыки и расширены знания в области баз данных, программирования, а также приобретены знания создания приложений.

Цель работы достигнута. В полной мере решены поставленные задачи:

- Создать удобное и интуитивно понятное Web приложение;
- Реализовать возможности учета и мониторинга абонементов;
- Реализовать возможность работы с клиентами и персоналом;
- Объединение работы администратора, клиента, тренера в одной системе

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Дино Эспозито, Modern Web Development: Understanding Domains, Technologies, and User Experiencек: / Дино Эспозито. – New York: Изд-во Вильямс, 2017. – 464 с.

2 Это онлайн курсы для обучения программированию. - https://www.freecodecamp.org

3 Интерактивная онлайн-платформа по обучению 12 языкам программирования. - https://www.codecademy.com

4 Библиотека официальной технической документации для разработчиков под ОС Microsoft Windows. - https://msdn.microsoft.com/ru-ru/dn308572.aspx

5 Крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. - https://github.com

6 Русскоязычный веб-сайт в формате коллективного блога с элементами новостного сайта. - https://habr.com