Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных,**

**принятия решений и финансовых технологий**

**Курсовая работа**

на тему:

**«Разработка информационной системы управления фитнес клубом»**

Выполнил:

Студент группы ПИ3-2

Бирюков Максим Сергеевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Научный руководитель:

доцент, к.э.н. Барабаш Дмитрий Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

2020

**Оглавление**

[**Введение** 3](#_Toc41044945)

[**Цели и задачи** 3](#_Toc41044946)

[**Техническое задание** 4](#_Toc41044947)

[**Функциональные требования** 4](#_Toc41044948)

[**Пользователи системы и основные процессы:** 5](#_Toc41044949)

[**Проектирование** 6](#_Toc41044950)

[**Описание интерфейса программы** 10](#_Toc41044951)

[**Описание базы данных** 11](#_Toc41044952)

[**Работа программы** 14](#_Toc41044953)

[**Общий интерфейс** 14](#_Toc41044954)

[**Интерфейс директора** 18](#_Toc41044955)

[**Форма бухгалтера** 24](#_Toc41044956)

[**Форма тренера:** 25](#_Toc41044957)

[**Форма клиента** 26](#_Toc41044958)

[**Назначение и состав классов программы:** 29](#_Toc41044959)

[**Формы** 29](#_Toc41044960)

[**Заключение** 30](#_Toc41044961)

[**Список литературы** 31](#_Toc41044962)

[**Приложение. Исходный код программы** 32](#_Toc41044963)

# **Введение**

Курсовая работа ориентирована на проектирование и реализацию информационной системы управления фитнес клубом.

Компьютерные технологии вошли практически во все сферы деятельности человека, и это довольно легко объяснить: специализированные программные продукты способны облегчить ведение учета и хранения информации практически в любой отрасли.

Если подробно изучить все сферы бизнеса с точки зрения обеспеченности информационными системами, то можно сделать вывод, что их распределение крайне неравномерно. Если общие задачи, которые ежедневно решает каждое предприятие (кадровый учет, бухгалтерский учет) автоматизированы достаточно хорошо, то в специфических аспектах функционирования предприятия наблюдается дефицит информационных систем. Одним из таких аспектов является деятельность фитнес-клуба. Каждый день на предприятиях подобного типа совершается множество операций: регистрация клиентов, отражение или фиксация фактов посещения, поступление и выдача денежных средств, а также многие другие операции, автоматизация которых поможет улучшить функционирование предприятия.

Актуальность проекта обусловлена тем, что на рынке отсутствует какое-либо разнообразие специализированных информационных систем по фитнес-клубам. На данный момент мне известна лишь одна такая ИС.

# **Цели и задачи**

Целью данной работы является создание приложения позволяющего вести учет клиентов, тренировок и сотрудников, возможность получать отчеты о состоянии дел на предприятии. Разрабатываемое приложение должно обеспечить возможность быстрого поиска необходимой информации.

Для достижения поставленной цели нужно решить следующие исследовательские задачи:

* анализ существующих автоматизированных систем (далее – АС) для спортивно-оздоровительных комплексов;
* исследование и анализ основных бизнес-процессов организации;
* формирование требований на разработку АС;
* проектирование и разработка АС для малого предприятия

# **Техническое задание**

1. Платформа разработки: Windows

2. Язык программирования: C#

3. База данных: SQL Server

4. Unit-тестирование: в среде Visual Studio

5. Среда разработки: Visual Studio

6. Средство контроля версий: Git

7. Удаленное средство контроля версий: GitHub

# **Функциональные требования**

Система должна:

1. иметь разграничение прав доступа;
2. содержать базы данных об услугах фитнес-клуба, клиентах и тренерах;
3. предоставлять возможность добавления, поиска, редактирования и удаления данных;
4. вести учет оплаты купленных услуг клиентами;
5. предоставлять возможность создания и корректировки расписания (добавление, изменение, удаление данных);
6. предоставлять возможность добавления отзывов о тренерах;
7. формировать следующие отчеты:

* наиболее востребованные услуги
* услуги, не пользующиеся популярностью
* загруженность тренера и отзывы клиентов
* расчет скидок по оплате для клиентов, одновременно посещающих 3 и более направлений (услуг)
* отчет о прибыли по каждой услуге и в общем по предприятию
* расписание занятий

# **Пользователи системы и основные процессы:**

Таблица 1.

|  |  |  |
| --- | --- | --- |
| № п.п. | Пользователь | Процессы |
| 1 | Генеральный директор | Получение отчетов  Создание и корректировка расписания  Добавление и удаление данных по тренерам, клиентам, занятиям |
| 2 | Тренер | Просмотр расписания |
| 3 | Бухгалтер | Получение отчетов |
| 4 | Клиент | Просмотр расписания  Просмотр тренеров  Добавление отзыва |

# **Проектирование**

Задачи, которые необходимо решить на протяжении разработки проекта представлены ниже:

|  |  |  |
| --- | --- | --- |
| **1** | **Анализ предметной области** | |
| 1.1 |  | Анализ конкурентов |
| 1.2 |  | Выбор технологического стека |
| **2** | **Проектирование** | |
| 2.1 |  | Определение пользователей системы |
| 2.2 |  | Проектирование сценариев использования |
| 2.3 |  | Проектирование базы данных |
| 2.4 |  | Проектирование пользовательского интерфейса |
| **3** | **Разработка** | |
| 3.1 |  | Разработка базы данных |
| 3.2 |  | Разработка front-end |
| 3.3 |  | Разработка back-end |
| **4** | **Тестирование** | |
| 4.1 |  | Тестирование приложения |
| 4.2 |  | Исправление ошибок |
| 4.3 |  | Разработка документации |
| **5** | **Выход проекта** | |
| 5.1 |  | Презентация проекта |

На рисунке 2 изображена диаграмма Ганта с расчетом времени на проектирования системы.

На рисунке 1 предоставлена диаграмма вариантов использования(use-case), отражающая отношения актеров и прецедентов. В данном случае в роли актеров выступают клиент и администратор системы.

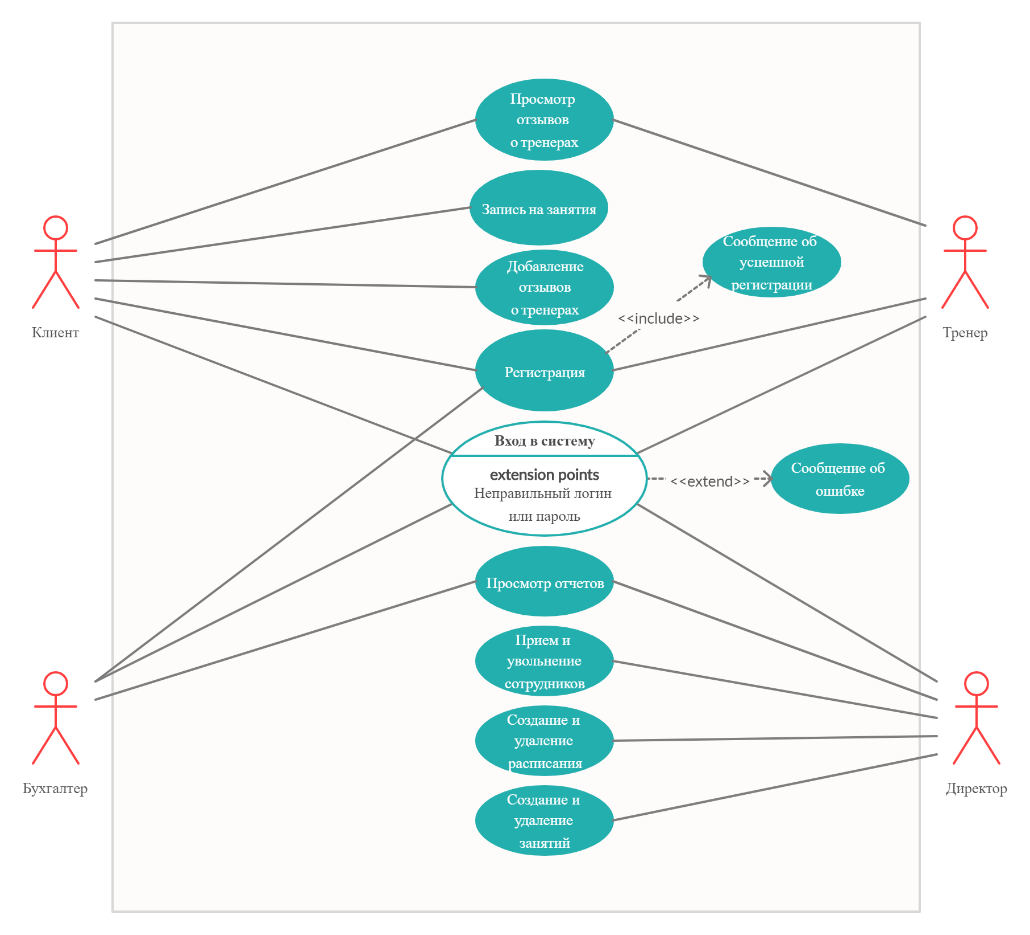


Рисунок 1. Use-case диаграмма

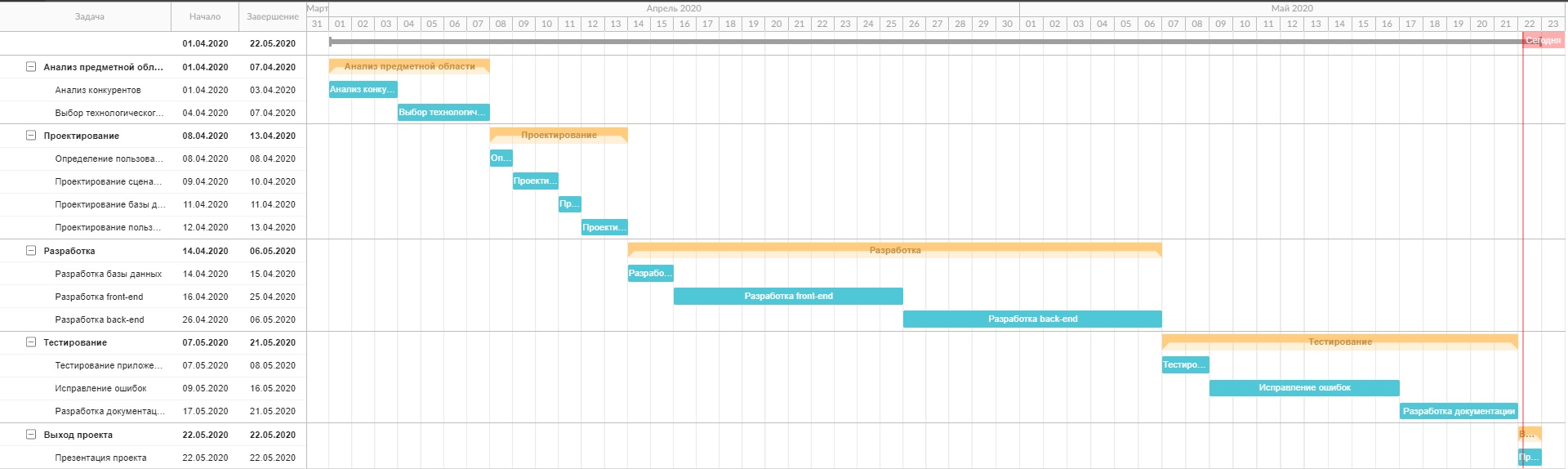


Рисунок 2. Диаграмма Ганта

На рисунке 3 предоставлена диаграмма последовательности (sequence diagram) операции регистрации пользователя с возможными вариантами исхода события. Каждый из прецедентов на диаграмме вариантов использования означает ту или иную операцию, выполняемую актером, к которому этот прецедент относится. Каждая из них подразумевает выполнение некой последовательности действий. Для изображения этой хронологии служат диаграммы последовательности, которые показывает взаимодействия объектов, упорядоченные по времени их проявления, и кооперации, служащие для отслеживания связей объектов.

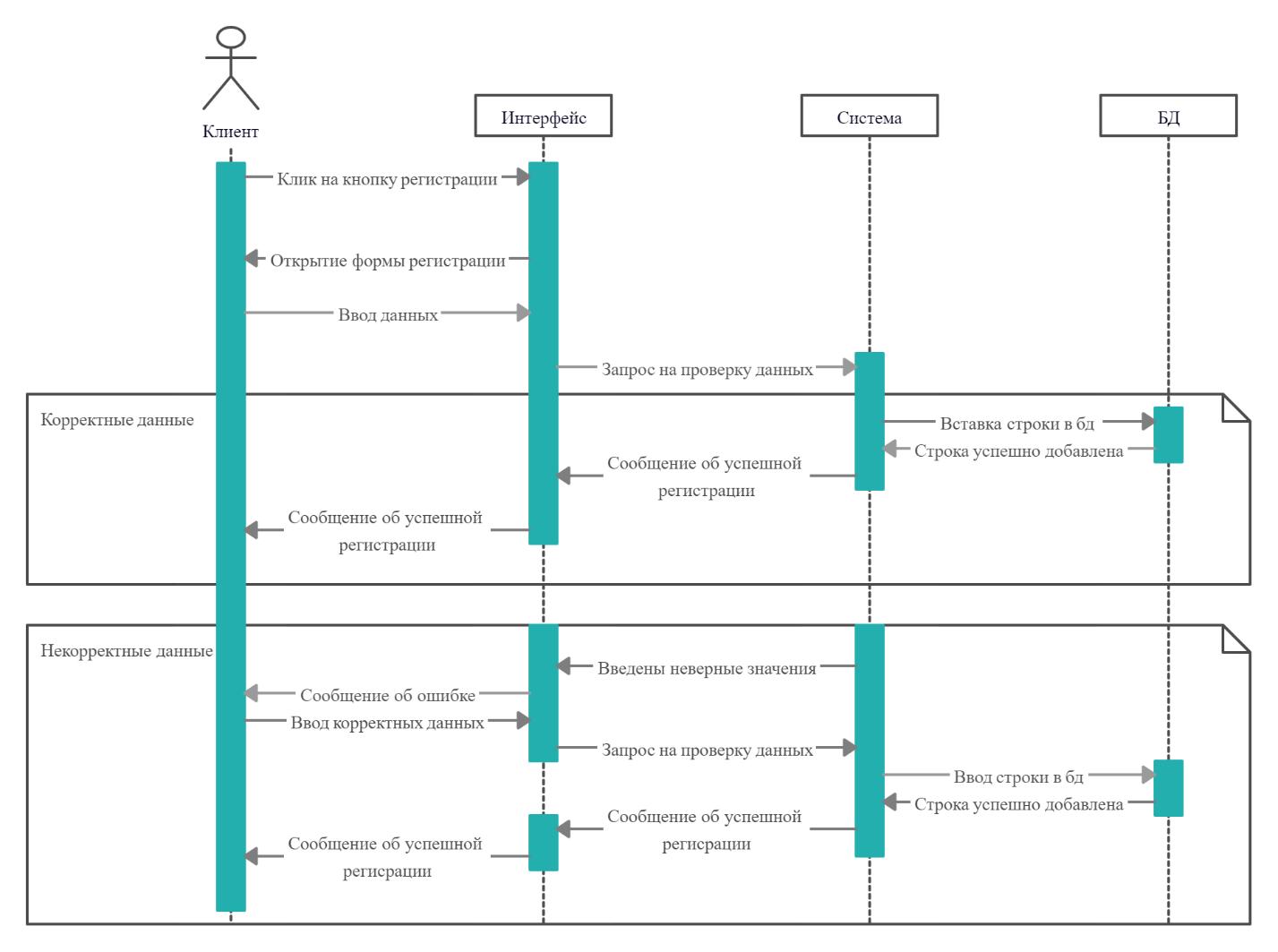


Рисунок 3. Sequence diagram

# **Описание интерфейса программы**

В приложении для реализации интерфейса используется 10 элементов

управления и компонентов:

* metroListView;
* metroButton;
* metroTextBox;
* metroLabel;
* metroGridView;
* Chart
* metroComboBox
* richTextBox
* metroTabControl

Посредством сетки metroListView пользователь видит на экране таблицы. Для того, чтобы

metroListView отображал данные, мы связываем его с базой данных, путем соответствующих запросов. Chart используется в отчетах. metroComboBox для упрощения работы с ИС.

# **Описание базы данных**

На рисунке 4 предоставлена схема базы данных.

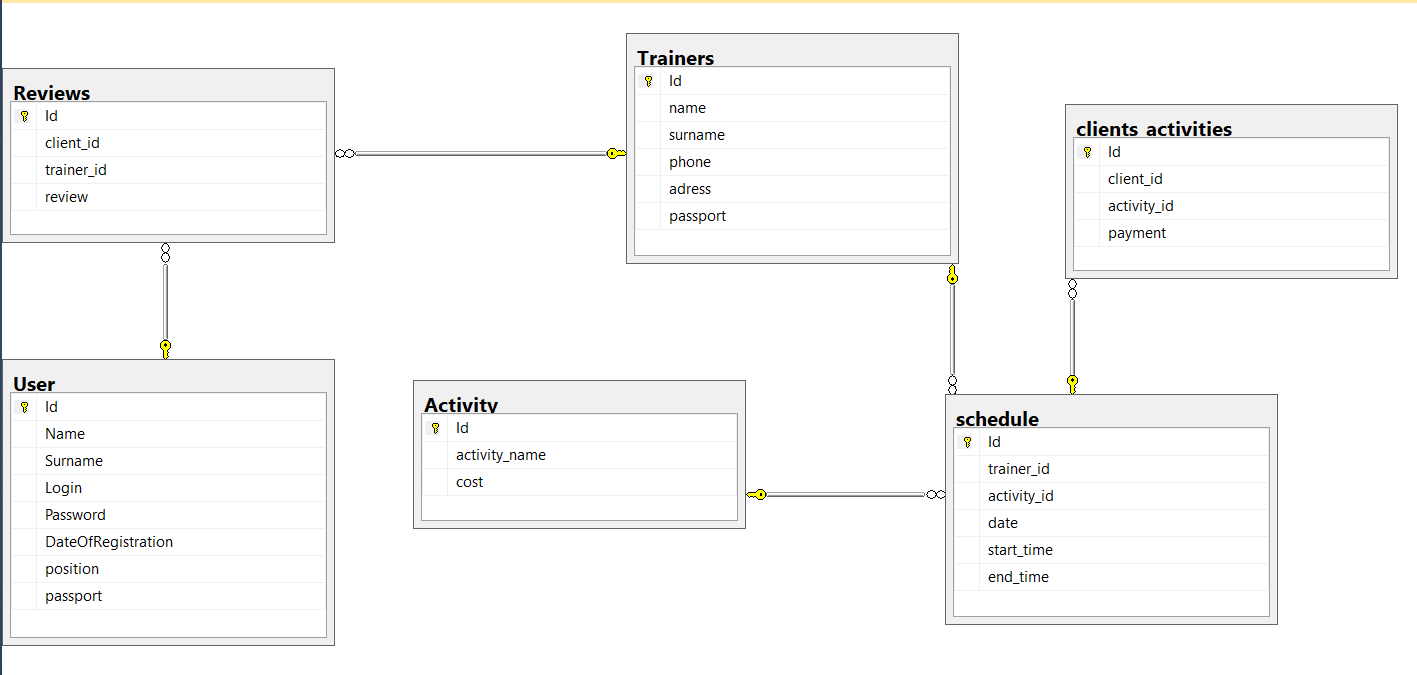


Рисунок 4. Диаграмма базы данных

Таблица User имеет следующую структуру:

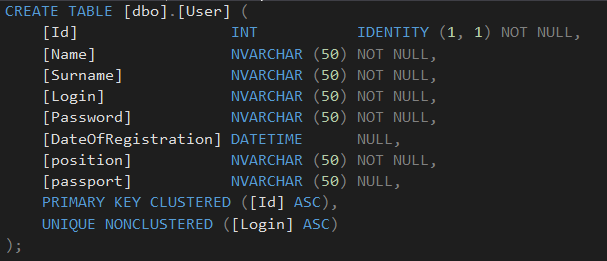


Рисунок 5. Таблица User

1. ID – Уникальный идентификатор. Является первичным ключом
2. Name – Имя пользователя
3. Surname – Фамилия пользователя
4. Login – Логин. Содержит ограничение на уникальность.
5. Password - Пароль
6. DateOfRegistration – Дата и время регистрации
7. Position – Должность (клиент, тренер, бухгалтер)
8. Passport - Паспортные данные

Служит для хранения данных пользователя, в том числе логина и пароля.

Таблица Trainers имеет следующую структуру:

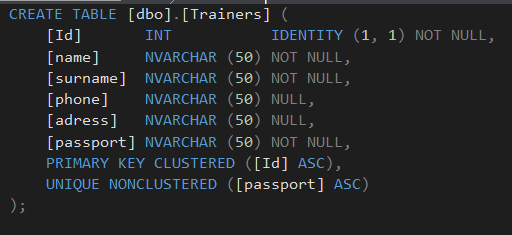


Рисунок 6. Таблица Trainers

1. ID – Уникальный идентификатор. Является первичным ключом
2. Name – Имя тренера
3. Surname – Фамилия тренера
4. Phone – Телефон
5. Address - Адрес
6. Passport - Паспортные данные. Содержит ограничение на уникальность

Ограничение на уникальность – passport. Служит для хранения данных о тренерах, которые заполняет директор. Если в данной таблице нет данных о тренере от директора, то тренер не сможет зарегистрироваться.

Таблица Activity имеет следующую структуру:

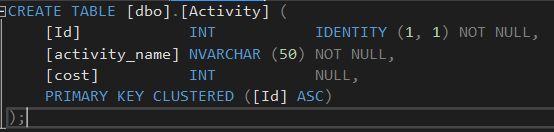


Рисунок 7. Таблица Activity

1. ID – Уникальный идентификатор. Является первичным ключом
2. Activity\_name – Название занятия
3. Cost – Стоимость занятия

Первичный ключ – Id. Служит для хранения данных о дисциплинах и их стоимости.

Таблица schedule имеет следующую структуру:

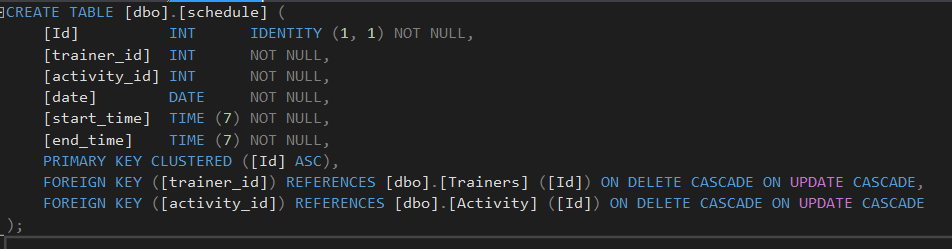


Рисунок 8. Таблица Schedule

1. ID – Уникальный идентификатор. Является первичным ключом
2. Trainer\_id – Id тренера. Является внешним ключом, ссылается на таблицу с тренерами – Trainers
3. Activity\_id – Id занятия. Является внешним ключом, ссылается на таблицу с занятиями Activity.
4. Date – Дата занятия
5. Start\_time – Время начала занятия
6. End\_time – Время конца занятия

Если удаляется какое-либо значение из родительской таблицы этих двух полей, то удаляется все строки содержащие удаленное значение. Служит для хранения расписания занятий.

Таблица clients\_activities имеет следующую структуру:

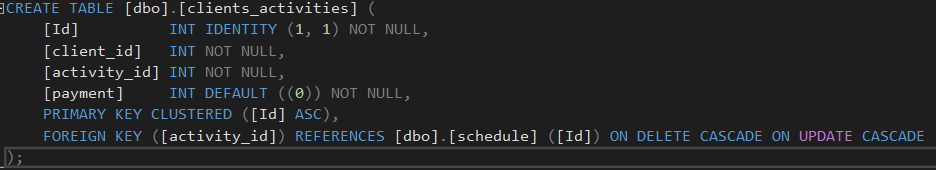


Рисунок 9. Таблица clients\_activities

1. ID – Уникальный идентификатор. Является первичным ключом
2. Client\_id – Id клиента
3. Activity\_id – Id занятия в расписании. Является внешним ключом, ссылается на таблицу с расписанием schedule
4. Payment – Столбец который показывает, оплачено ли данное занятие у клиента. По умолчанию 0 – не оплачено. 1 – оплачено.

Таблица содержит в себе все занятия, на которые записался конкретный клиент. Так же если удаляется какая-либо строка из родительской таблицы, удаляются все записи с удаленным значением и отсюда.

Таблица Reviews имеет следующую структуру:

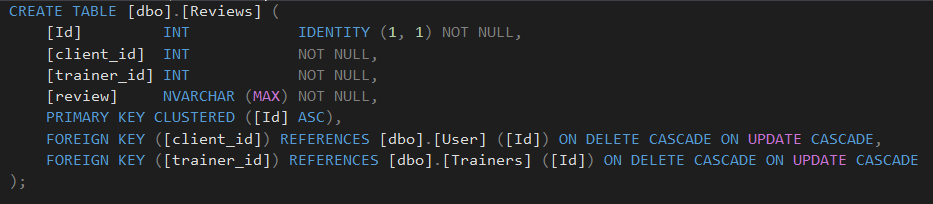


Рисунок 10. Таблица Reviews

1. ID – Уникальный идентификатор. Является первичным ключом
2. Client\_id – Id клиента. Внешний ключ
3. Trainer\_id – Id тренера. Внешний ключ
4. Review – Отзыв.

Служит для хранения отзывов о тренерах.

# **Работа программы**

## **Общий интерфейс**

При входе в программу мы видим окно со входом. Пользователь может ввести логин и пароль, либо зарегистрироваться.

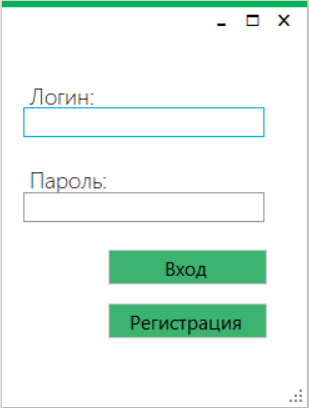


Рисунок 11.

При регистрации пользователь увидит данное окно:

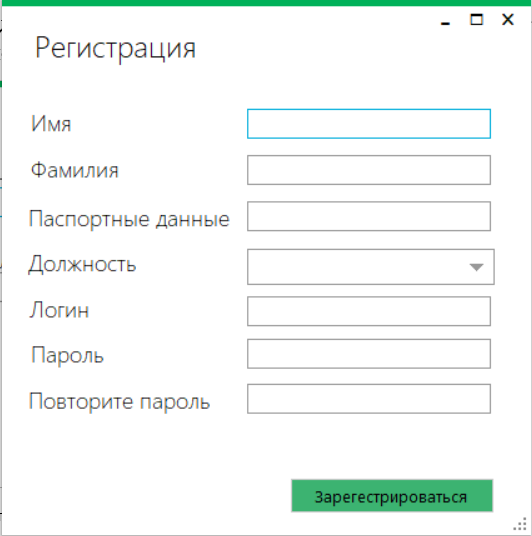


Рисунок 12.

Зарегистрироваться может только клиент, тренер и бухгалтер. Директора регистрирует администратор всего лишь однажды. Так же тут установлена защита от ввода пустых значений и одинакового логина:

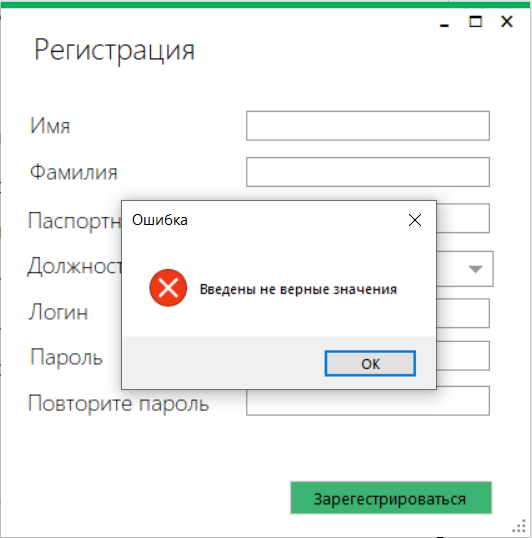


Рисунок 13.

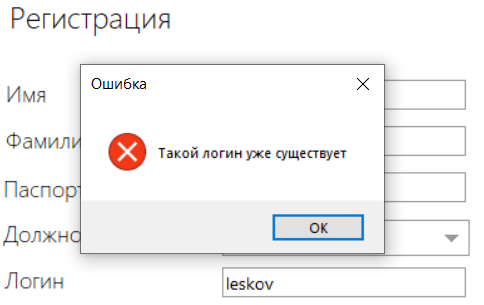


Рисунок 14.

Тренер может зарегистрироваться только после того, как директор занесет его в соответствующую базу данных.

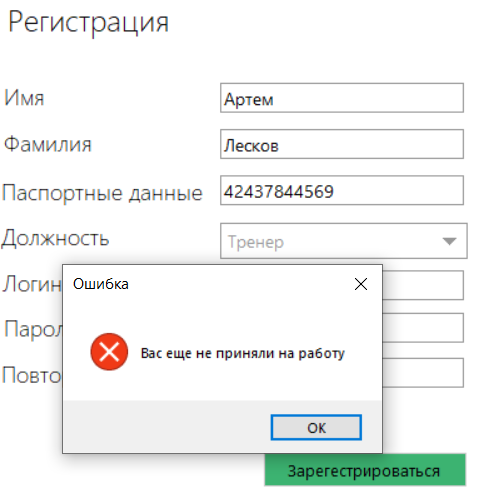


Рисунок 15.

При вводе неверного логина и пароля система так же выдаст соответствующую ошибку:

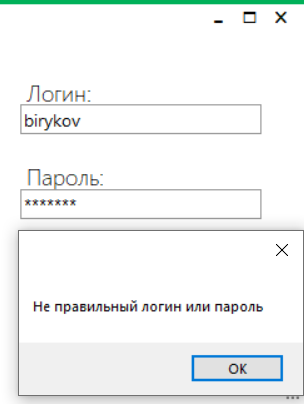


Рисунок 16.

## **Интерфейс директора**

При входе директор видит данное окно:

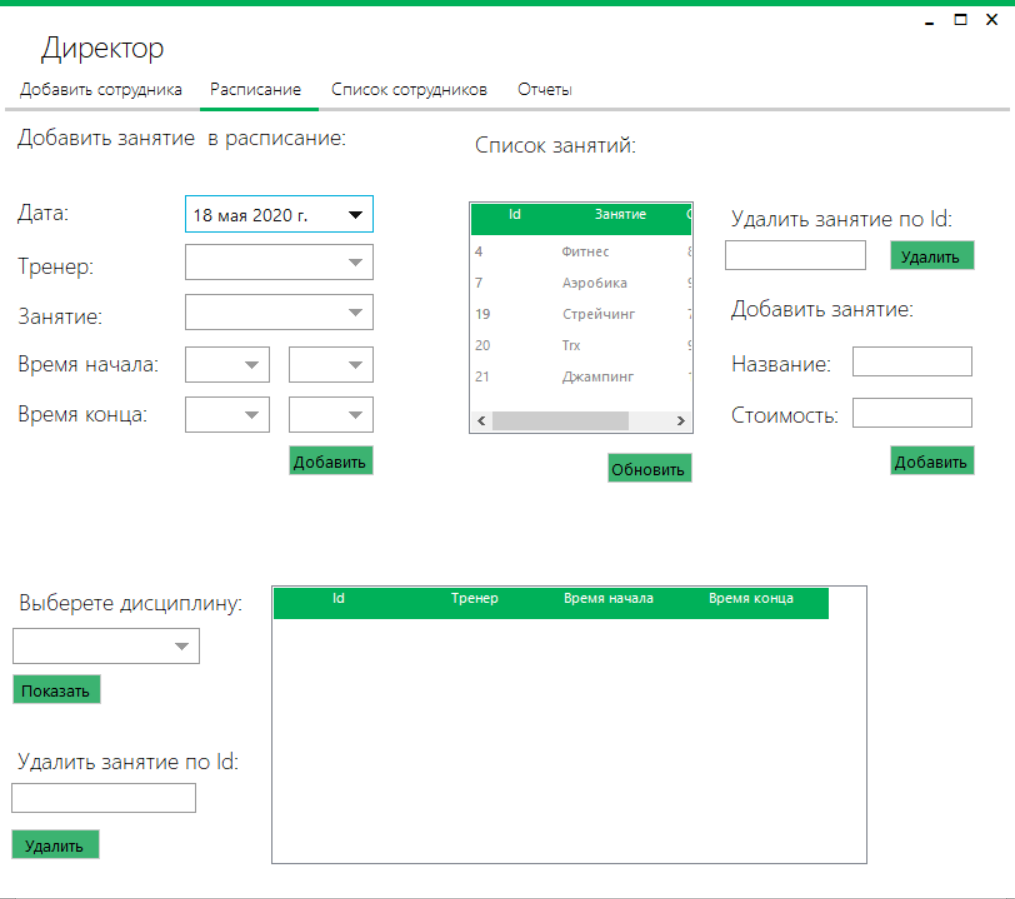


Рисунок 17.

Тут он может добавить занятие в расписание, удалить его, посмотреть расписание по дисциплине, добавить или удалить дисциплину.

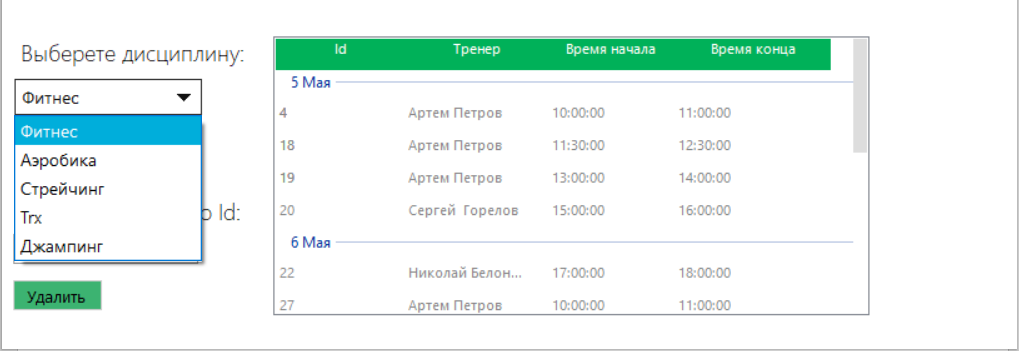


Рисунок 18.

Все данные в comboBox добавляются динамически при каждом нажатии, так как тренер может добавить или удалить какое-либо занятие или сотрудника и это сразу должно отобразиться в comboBox.

Вот так выглядит процедура добавления занятия в базу данных.

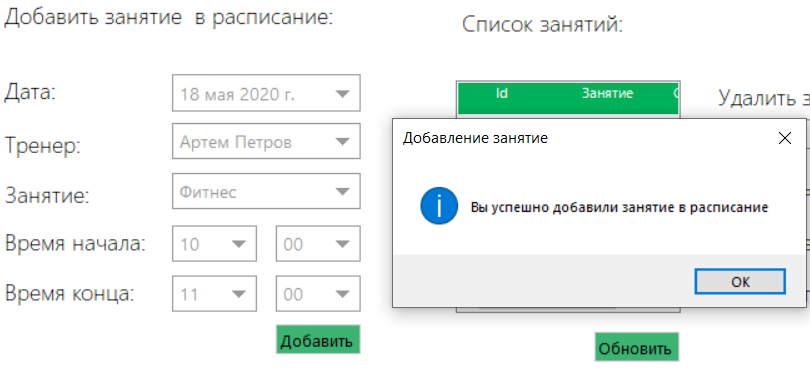


Рисунок 19.

Процедура удаления:

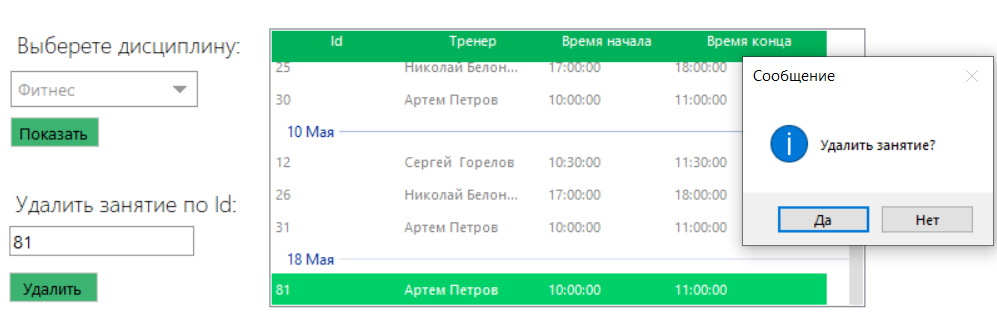


Рисунок 20.

Так же все данные в listView группируются по дате автоматически.

Весь тот же самый функционал есть в занятиях:

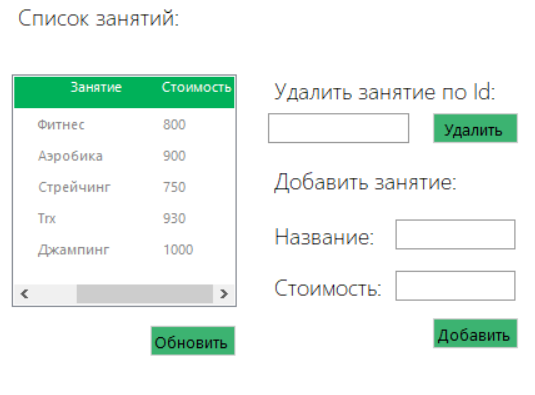


Рисунок 21.

При удалении занятия, все расписание по данной дисциплине так же удаляется из базы данных и всех comoBox.

Можно добавить или удалить какие-либо данные по занятиям.

Директор так же имеет возможность добавлять сотрудников:

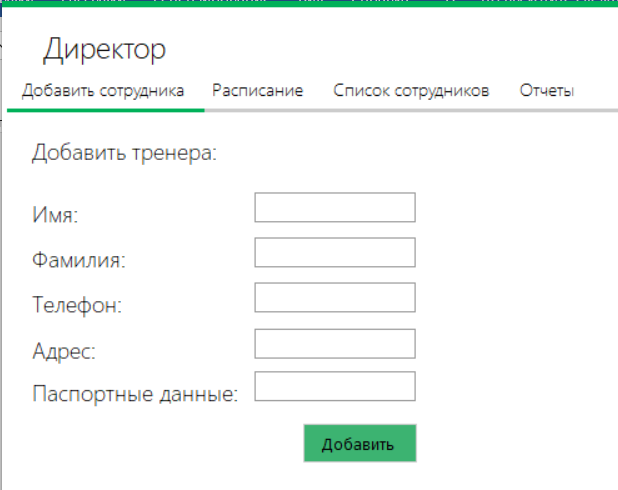


Рисунок 22.

Только после добавления директором сотрудника в БД, тренер может зарегистрироваться в системе.

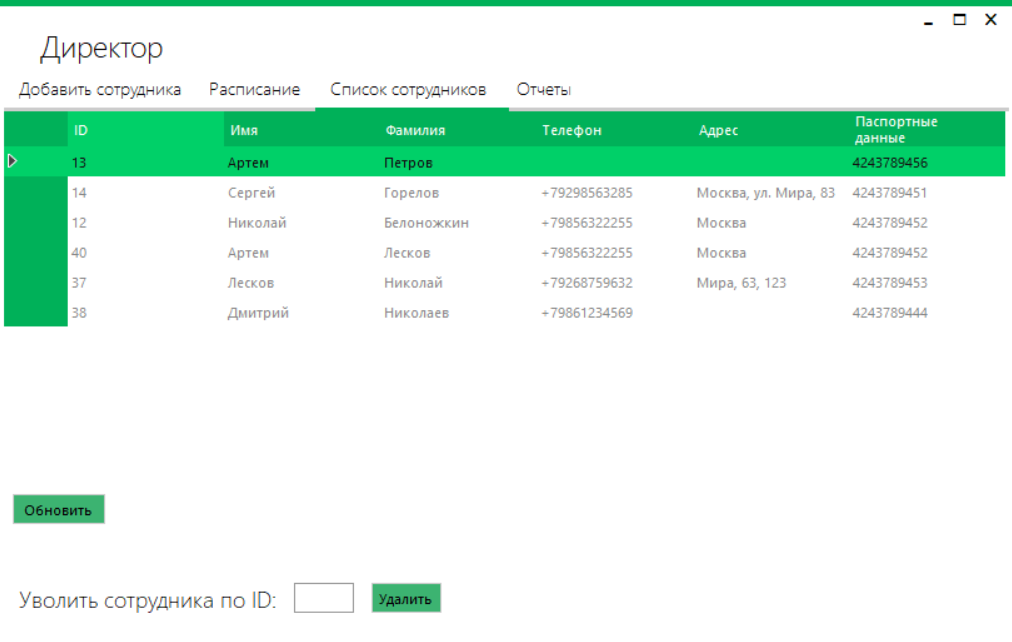


Рисунок 23.

Так же он может увольнять тренеров. При увольнении какого-либо тренера, все его занятия автоматически удаляются из базы. Соответственно, если клиент был записан на какое-либо из них, у него так же они удалятся.

Директор имеет возможность просматривать все необходимые отчеты по предприятию:

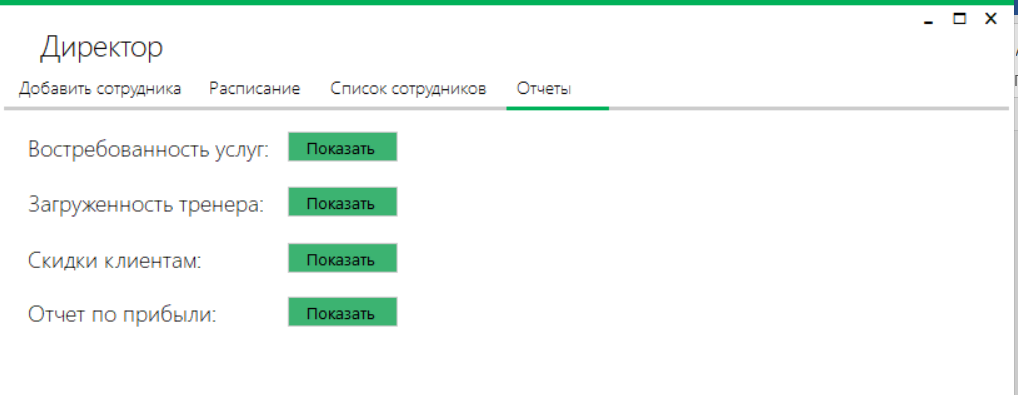


Рисунок 24.

Отчет востребованности услуг:

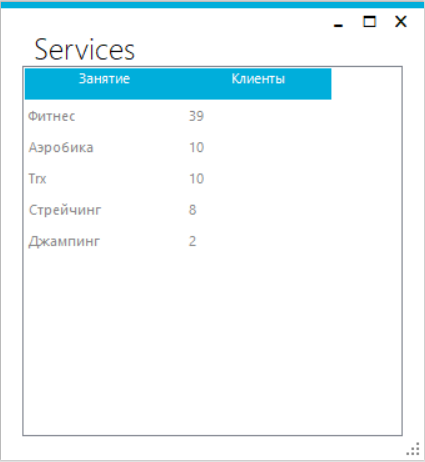


Рисунок 25.

Он отражает количество клиентов записавшихся на какое-либо из направлений. Из данного отчета видно, что фитнес пользуется популярностью, а джампинг нет. Владельцу стоит задуматься о повышении количества проводимых занятий по фитнесу и о привлечении клиентов на джампинг.

Отчет о загруженности тренера:

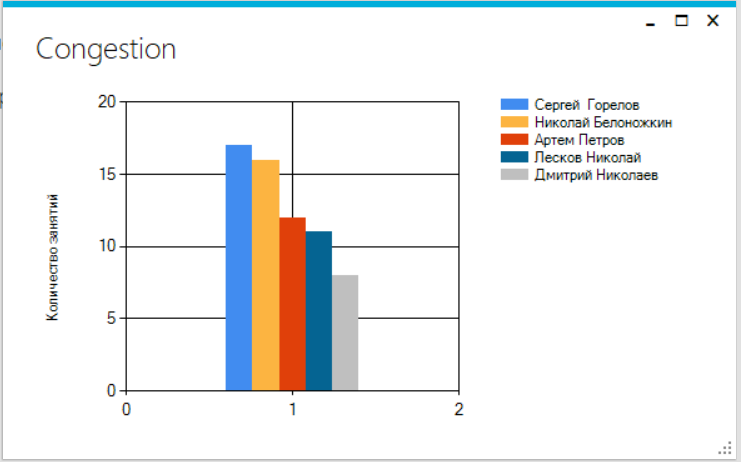


Рисунок 26.

Он показывает сколько занятий проводит каждый тренер.

Отчет о скидках клиентам:

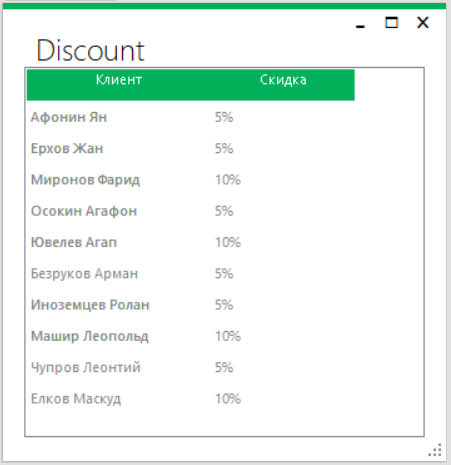


Рисунок 27.

Если клиент посещает 3 различных направления, то ему предоставляется скидка 5%, если 4 и более – 10%.

Отчет о прибыли:

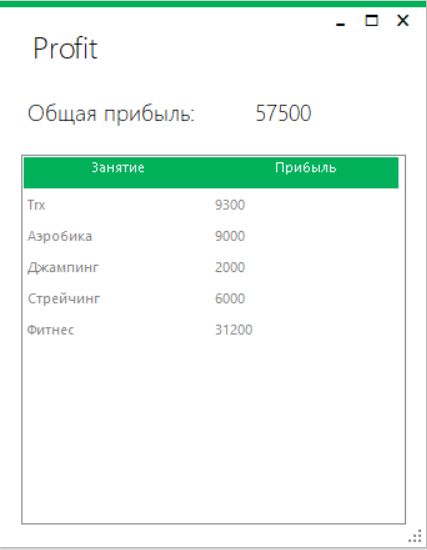


Рисунок 28.

Сверху формы отображается прибыль по предприятию в целом, а в listView прибыль по каждому из направлений.

## **Форма бухгалтера**

При входе бухгалтер видит данную форму:

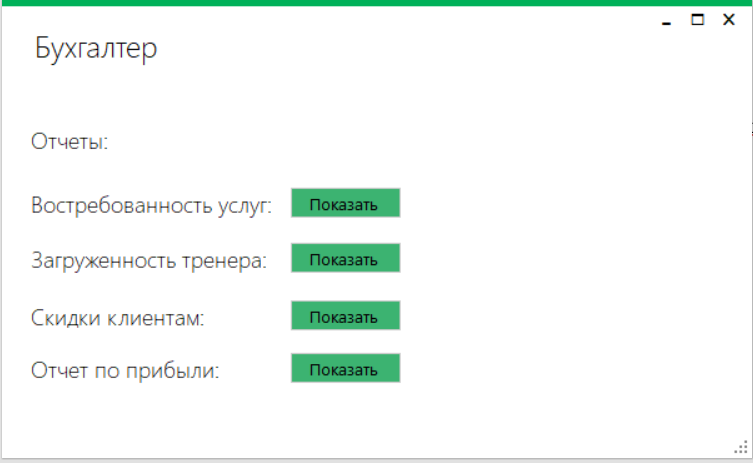


Рисунок 29.

По функционалу у нее абсолютно все тоже самое что и у директора в отчетах.

## **Форма тренера:**

При входе тренер видит данную форму:

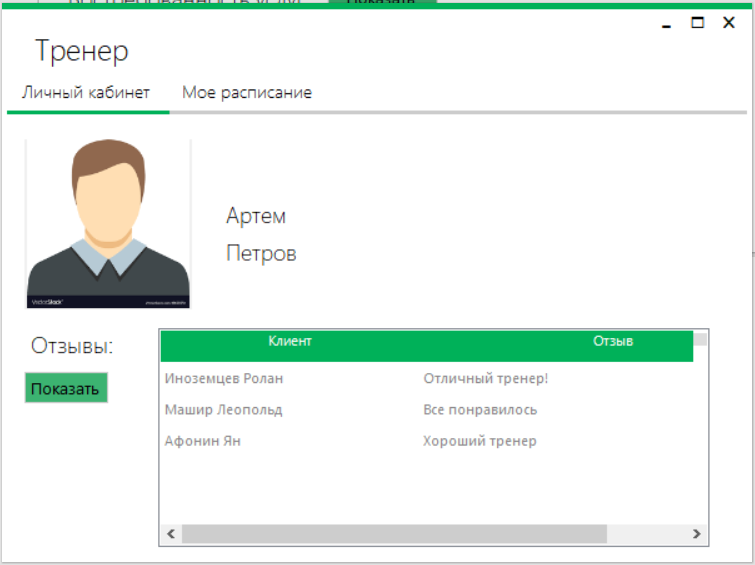


Рисунок 30.

Справа от фотографии отображаются имя и фамилия пользователя. Снизу ListView в котором тренер может просмотреть все отзывы о нем

Так же тренер может просматривать все его расписание.

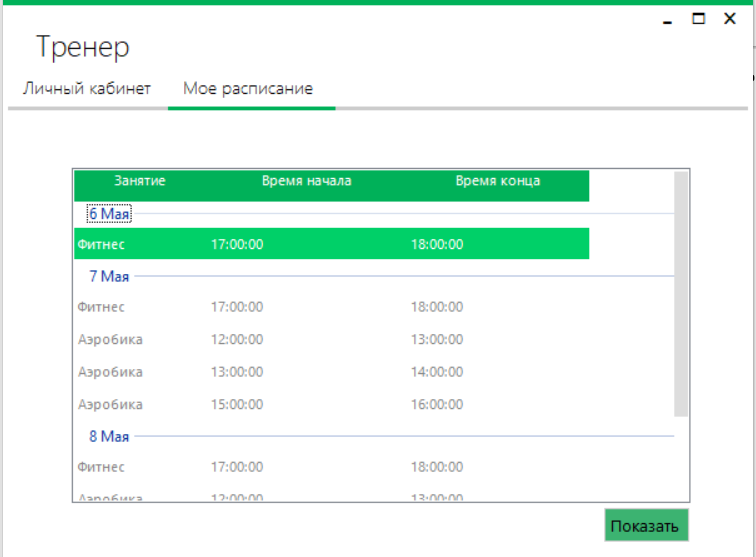


Рисунок 31.

## **Форма клиента**

При запуске программы клиент видит данную форму:

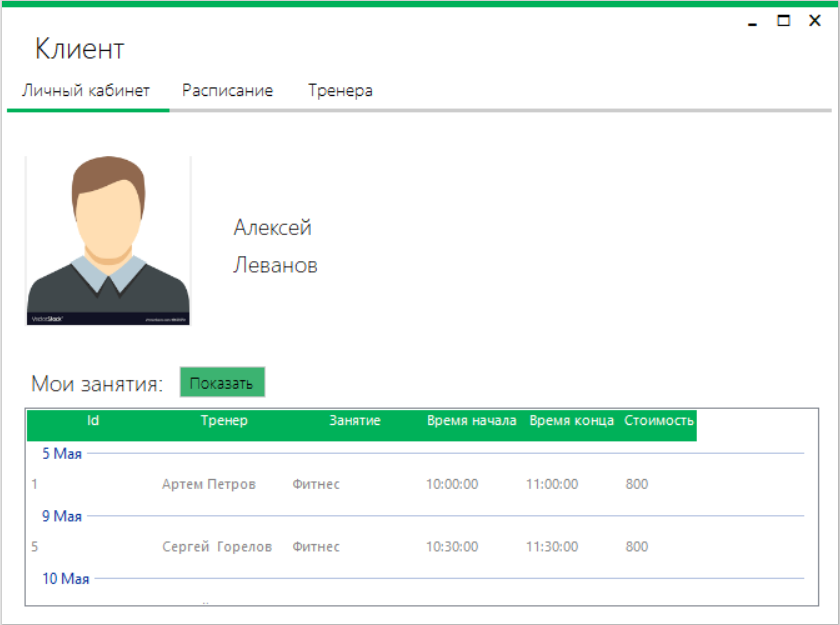


Рисунок 32.

В нижней части экрана отображаются все занятия, на которые клиент записался.

Клиент имеет возможность просмотреть расписание и записаться на любое понравившееся занятие:

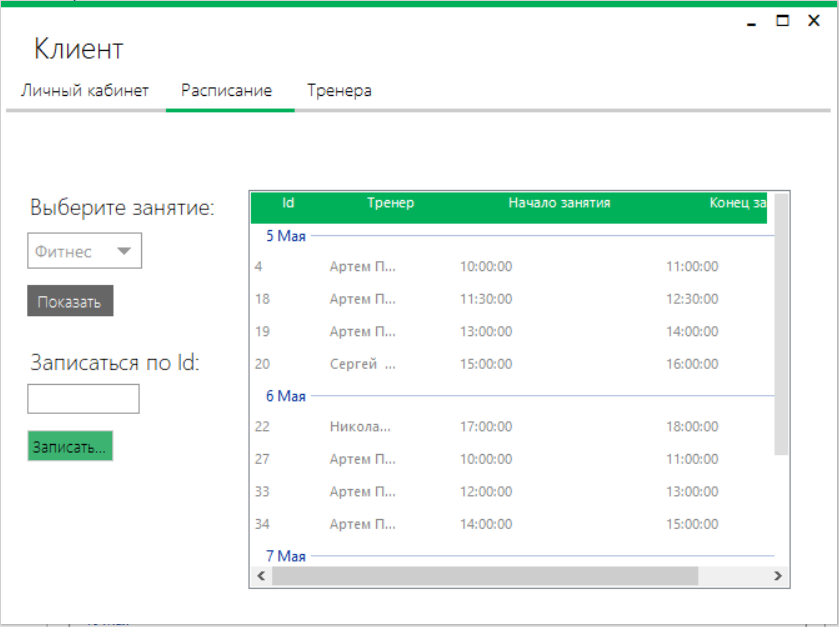


Рисунок 33.

Так же клиент может просмотреть отзывы о тренере и оставить свой:

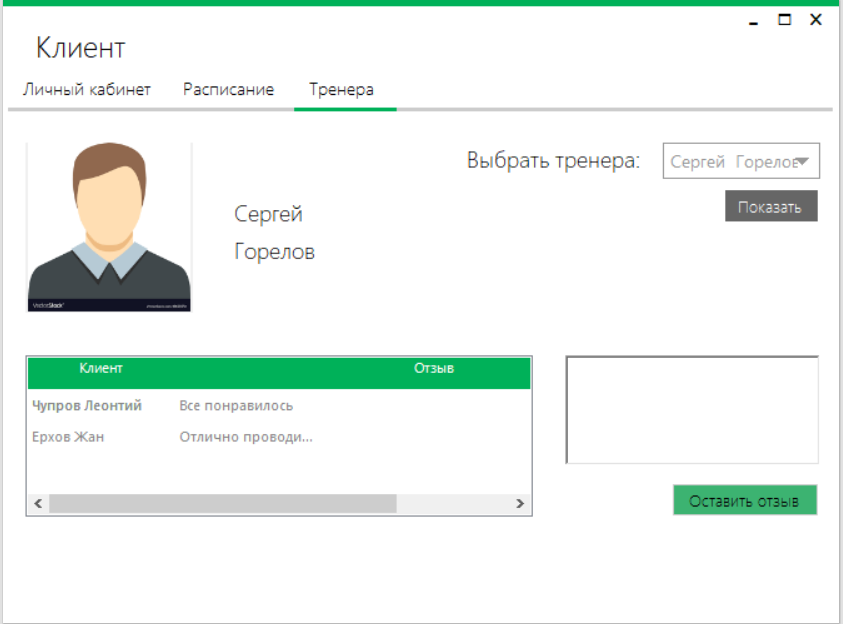


Рисунок 34.

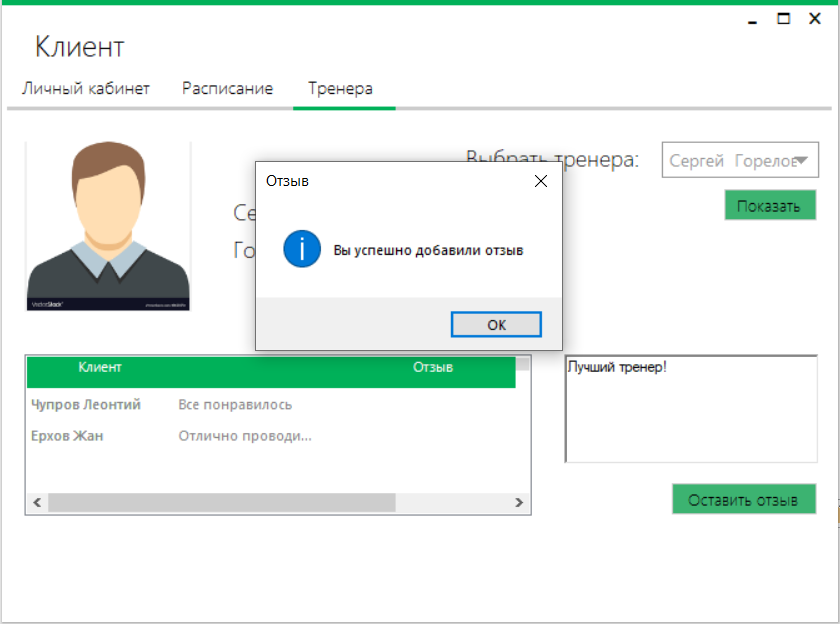


Рисунок 35.

# **Назначение и состав классов программы:**

.activity – класс, который имеет два атрибута – Id и activity\_name. Используется в комбо боксах для получения списка всех активностей и их Id.

.trainer - класс, который имеет два атрибута – Id и fullname. Используется в комбо боксах для получения списка всех тренеров и их Id.

.DB – класс, который содержит все методы для работы с базой данных. Так же он содержит в себе атрибуты, которые потом вызываются в других формах, для получения Id, имени и фамилии вошедшего пользователя.

.schedule – класс, который содержит методы для работы с расписанием. Он содержит методы, которые возвращают расписание предназначенное для тренеров, клиентов и директора, содержит методы для группировки данных и их последующем помещении в ListView.

# **Формы**

Form1 – главная форма, где содержится поле ввода логина и пароля.

Registration – форма регистрации.

DirectorForm – форма, в которой работает директор.

TrainerForm - форма, в которой работает тренер.

ClientForm – форма, в которой работает клиент.

Accountant – форма, в которой работает бухгалтер.

Congestion – форма отчета о загруженности тренеров

Discount – форма отчета о скидках клиентам.

Profit – форма отчета о выручке с каждой дисциплины.

Services – форма отчета о востребованности занятий

# **Заключение**

Разработав информационную систему фитнес клуба, на базе Windows-приложение, был выполнен ряд поставленных задач. Было создано приложение, позволяющее вести учет клиентов, тренировок и сотрудников, возможность получать отчеты о состоянии дел на предприятии.

Для достижения поставленной цели были решены следующие исследовательские задач:

• анализ существующих АС для спортивно-оздоровительных комплексов;

• исследование и анализ основных бизнес-процессов организации;

• формирование требований на разработку АС;

• проектирование и разработка АС для малого предприятия

Информационная система написана на языке программирования С# с использованием классов, методов и свойств. Для реализации некоторых задач потребовалось обратиться к объектам операционной системы: системный реестр, файловая система, процессы. Также были использованы инструменты работы с базами данных.

Программа может быть расширена статистическими и финансовыми функциями. В будущем, в программу следует добавить возможность онлайн-оплаты. Нужно поработать над безопасностью программы, оптимизацией всех процессов, оптимизацией запросов в базу данных.

# **Список литературы**

1. Горелов С.В., Волков А.Г. Разработка Windows-приложений. Часть 1. Учебное пособие. Образовательный портал Финансового университета. 2018.

2. Горелов С.В. Разработка Windows-приложений. Часть 2. Учебное пособие. Образовательный портал Финансового университета. 2018. 3.

3. Официальный сайт Microsoft: [Интернет-ресурс]. URL: https://msdn.microsoft.com org.

4. Сайт сообщества программистов [Интернет-ресурс]. URL: <http://www.cyberforum.ru>

5. Официальная документация по фреймворку “MetroFramework” [Интернет-ресурс]. URL: <https://github.com/dennismagno/metroframework-modern-ui>

6. Сайт сообщества программистов [Интернет-ресурс]. URL: <https://stackoverflow.com/>

7. Интерактивный учебник по SQL [Интернет-ресурс]. URL: <http://www.sql-tutorial.ru/>

# **Приложение. Исходный код программы**

Файл Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FitnessClub

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Файл Accountant.cs

namespace FitnessClub

{

public partial class Accountant : MetroFramework.Forms.MetroForm

{

public Accountant()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

}

private void metroButton9\_Click(object sender, EventArgs e)

{

Services services = new Services();

services.Show();

}

private void metroButton10\_Click(object sender, EventArgs e)

{

Congestion congestion = new Congestion();

congestion.Show();

}

private void metroButton11\_Click(object sender, EventArgs e)

{

Discount discount = new Discount();

discount.Show();

}

private void metroButton12\_Click(object sender, EventArgs e)

{

Profit profit = new Profit();

profit.Show();

}

private void Accountant\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

Файл activity.cs

namespace FitnessClub

{

class activity

{

public int Id { get; set; }

public string activity\_name { get; set; }

DB db = new DB();

public List<activity> return\_activities()

{

SqlCommand sqlCommand = new SqlCommand("SELECT Id, activity\_name FROM [Activity] ", db.GetConnection());

db.openConnection();

SqlDataReader reader = sqlCommand.ExecuteReader();

List<activity> activities = new List<activity>();

while (reader.Read())

{

activities.Add(new activity());

activities[activities.Count - 1] = new activity { Id = Convert.ToInt32(reader[0]), activity\_name = reader[1].ToString() };

}

reader.Close();

db.closeConnection();

return activities;

}

}

}

Файл ClientForm.cs

namespace FitnessClub

{

public partial class ClientForm : MetroFramework.Forms.MetroForm

{

DB db = new DB();

public ClientForm()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

metroListView1.StyleManager = metroStyleManager1;

metroListView2.StyleManager = metroStyleManager1;

metroListView3.StyleManager = metroStyleManager1;

metroTabControl1.StyleManager = metroStyleManager1;

metroLabel1.Text = DB.Name;

metroLabel2.Text = DB.Surname;

LoadSchedule();

}

public void LoadSchedule()

{

metroListView2.Items.Clear();

schedule schedule = new schedule();

schedule.fill\_listview\_client(metroListView2);

}

private void combo1\_click(object sender, EventArgs e)

{

activity activity = new activity();

List<activity> activities = activity.return\_activities();

metroComboBox1.DataSource = activities;

metroComboBox1.DisplayMember = "activity\_name";

metroComboBox1.ValueMember = "Id";

}

private void metroButton1\_Click(object sender, EventArgs e)

{

schedule schedule = new schedule();

schedule.fill\_listview(metroListView1, metroComboBox1.SelectedValue.ToString());

}

private void metroButton2\_Click(object sender, EventArgs e)

{

try

{

SqlCommand command = new SqlCommand("INSERT INTO [clients\_activities](client\_id,activity\_id,payment) VALUES (@client\_id, @activity\_id, @payment)", db.GetConnection());

command.Parameters.AddWithValue("@client\_id", DB.Id);

command.Parameters.AddWithValue("@activity\_id", Convert.ToInt32(metroTextBox1.Text));

command.Parameters.AddWithValue("@payment", 0);

db.openConnection();

command.ExecuteNonQuery();

MessageBox.Show("Вы успешно записались на занятие", "Добавление занятия", MessageBoxButtons.OK,

MessageBoxIcon.Information);

db.closeConnection();

metroTextBox1.Text = "";

}

catch (Exception)

{

MessageBox.Show("Запись не прошла");

}

}

private void metroComboBox2\_MouseClick(object sender, MouseEventArgs e)

{

trainer trainer = new trainer();

List<trainer> trainers = trainer.return\_trainers();

metroComboBox2.DataSource = trainers;

metroComboBox2.DisplayMember = "fullname";

metroComboBox2.ValueMember = "Id";

}

private void metroButton4\_Click(object sender, EventArgs e)

{

metroListView3.Items.Clear();

string[] words = metroComboBox2.Text.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

metroLabel7.Text = words[0];

metroLabel6.Text = words[1];

SqlCommand sqlCommand = new SqlCommand("select Name + ' ' + Surname as Name, review from Reviews join [User] on [User].Id = Reviews.client\_id where trainer\_id = @trainer\_id", db.GetConnection());

sqlCommand.Parameters.AddWithValue("@trainer\_id", Convert.ToInt32(metroComboBox2.SelectedValue));

db.openConnection();

SqlDataReader reader = sqlCommand.ExecuteReader();

while (reader.Read())

{

ListViewItem lvi = new ListViewItem(reader[0].ToString());

lvi.SubItems.Add(reader[1].ToString());

metroListView3.Items.Add(lvi);

}

reader.Close();

db.closeConnection();

}

private void metroButton3\_Click(object sender, EventArgs e)

{

if (metroComboBox2.Text == "")

MessageBox.Show("Выберите тренера");

else

{

try

{

SqlCommand sqlCommand = new SqlCommand("INSERT INTO [Reviews] (client\_id, trainer\_id, review) VALUES (@client\_id, @trainer\_id, @review)", db.GetConnection());

sqlCommand.Parameters.AddWithValue("@client\_id", DB.Id);

sqlCommand.Parameters.AddWithValue("@trainer\_id", Convert.ToInt32(metroComboBox2.SelectedValue));

sqlCommand.Parameters.AddWithValue("@review", richTextBox1.Text);

db.openConnection();

if (sqlCommand.ExecuteNonQuery() == 1)

MessageBox.Show("Вы успешно добавили отзыв", "Отзыв", MessageBoxButtons.OK,

MessageBoxIcon.Information);

else

MessageBox.Show("Добавление не прошло");

db.closeConnection();

}

catch (Exception)

{

MessageBox.Show("Добавление не прошло");

}

}

}

private void metroButton5\_Click(object sender, EventArgs e)

{

LoadSchedule();

}

private void ClientForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

Файл Congestion.cs

namespace FitnessClub

{

public partial class Congestion : MetroFramework.Forms.MetroForm

{

public Congestion()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

LoadData();

}

public void LoadData()

{

DB db = new DB();

chart1.Series.Clear();

SqlCommand command = new SqlCommand("select Trainers.name + ' ' + Trainers.surname, count(schedule.Id) as 'activity' from schedule full join Trainers on schedule.trainer\_id = Trainers.Id group by Trainers.name + ' ' + Trainers.surname order by activity DESC", db.GetConnection());

db.openConnection();

SqlDataReader reader = command.ExecuteReader();

List<string> seriesArray = new List<string>();

List<int> pointsArray = new List<int>();

while (reader.Read())

{

seriesArray.Add(reader[0].ToString());

pointsArray.Add(Convert.ToInt32(reader[1]));

}

reader.Close();

db.closeConnection();

for (int i = 0; i< seriesArray.Count(); i++)

{

Series series = this.chart1.Series.Add(seriesArray[i]);

series.Points.Add(pointsArray[i]);

}

}

}

}

Файл DB.cs

namespace FitnessClub

{

class DB

{

public static int Id { get; set; }

public static string Name { get; set; }

public static string Surname { get; set; }

SqlConnection connection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\maxim\source\repos\FitnessClub\FitnessClub\Database.mdf;Integrated Security=True");

public void openConnection()

{

if (connection.State == System.Data.ConnectionState.Closed)

connection.Open();

}

public void closeConnection()

{

if (connection.State == System.Data.ConnectionState.Open)

connection.Close();

}

public SqlConnection GetConnection()

{

return connection;

}

}

}

Файл DirectorForm.cs

namespace FitnessClub

{

public partial class DirectorForm : MetroFramework.Forms.MetroForm

{

DB db = new DB();

public DirectorForm()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

metroTab.StyleManager = metroStyleManager1;

metroGrid1.StyleManager = metroStyleManager1;

metroListView1.StyleManager = metroStyleManager1;

metroListView2.StyleManager = metroStyleManager1;

LoadData();

LoadActivity();

}

private void metroButton1\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(metroTextBox2.Text) | string.IsNullOrWhiteSpace(metroTextBox2.Text) |

string.IsNullOrEmpty(metroTextBox1.Text) | string.IsNullOrWhiteSpace(metroTextBox1.Text) |

string.IsNullOrEmpty(metroTextBox5.Text) | string.IsNullOrWhiteSpace(metroTextBox5.Text))

{

MessageBox.Show("Введены не верные значения", "Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error, MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

else

{

try

{

SqlCommand sqlCommand = new SqlCommand("INSERT INTO [Trainers] (name, surname, phone," +

" adress, passport) VALUES (@Name, @Surname, @phone, @adress," +

" @passport)", db.GetConnection());

sqlCommand.Parameters.AddWithValue("Name", metroTextBox1.Text);

sqlCommand.Parameters.AddWithValue("Surname", metroTextBox2.Text);

sqlCommand.Parameters.AddWithValue("phone", metroTextBox3.Text);

sqlCommand.Parameters.AddWithValue("adress", metroTextBox4.Text);

sqlCommand.Parameters.AddWithValue("passport", metroTextBox5.Text);

db.openConnection();

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Вы успешно добавили тренера", "Добавление тренера", MessageBoxButtons.OK,

MessageBoxIcon.Information);

metroTextBox1.Text = "";

metroTextBox2.Text = "";

metroTextBox3.Text = "";

metroTextBox4.Text = "";

metroTextBox5.Text = "";

db.closeConnection();

}

catch (Exception)

{

MessageBox.Show("Добавление не прошло");

}

}

}

private void LoadData()

{

metroGrid1.Rows.Clear();

SqlCommand sqlCommand = new SqlCommand("SELECT [User].Id, [User].Name, [User].Surname, phone, adress, [User].passport FROM [Trainers] left join [User] on [Trainers].passport = [User].passport ", db.GetConnection());

db.openConnection();

SqlDataReader reader = sqlCommand.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[6]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

}

reader.Close();

db.closeConnection();

foreach (string[] s in data)

metroGrid1.Rows.Add(s);

}

public void LoadActivity()

{

metroListView1.Items.Clear();

SqlCommand sqlCommand = new SqlCommand("SELECT \* FROM [Activity]", db.GetConnection());

db.openConnection();

SqlDataReader reader = sqlCommand.ExecuteReader();

List<string[]> activ = new List<string[]>();

while (reader.Read())

{

activ.Add(new string[3]);

activ[activ.Count - 1][0] = reader[0].ToString();

activ[activ.Count - 1][1] = reader[1].ToString();

activ[activ.Count - 1][2] = reader[2].ToString();

}

reader.Close();

db.closeConnection();

foreach (string[] s in activ)

{

ListViewItem lvi = new ListViewItem(s[0]);

lvi.SubItems.Add(s[1]);

lvi.SubItems.Add(s[2]);

metroListView1.Items.Add(lvi);

}

}

private void DirectorForm\_Load(object sender, EventArgs e)

{

}

private void metroButton2\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("Уволить тренера?",

"Сообщение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information);

if (result == DialogResult.Yes)

{

string id = metroTextBox6.Text;

SqlCommand command1 = new SqlCommand("Select [Trainers].Id from [Trainers] left join [User] on [Trainers].passport = [User].passport where [User].Id = @id", db.GetConnection());

command1.Parameters.AddWithValue("id", metroTextBox6.Text);

db.openConnection();

SqlDataReader reader1 = command1.ExecuteReader();

string tr\_id = "";

if (reader1.HasRows)

{

reader1.Read();

tr\_id = reader1[0].ToString();

reader1.Close();

}

else

{

MessageBox.Show("Такого Id в базе данных нет");

reader1.Close();

return;

}

SqlCommand command2 = new SqlCommand("delete from [Trainers] where Id ='" + tr\_id + "'", db.GetConnection());

command2.ExecuteNonQuery();

SqlCommand command = new SqlCommand("delete from [User] where Id = @id", db.GetConnection());

command.Parameters.AddWithValue("id", id);

command.ExecuteNonQuery();

MessageBox.Show("Вы успешно уволили тренера", "Увольнение тренера", MessageBoxButtons.OK,

MessageBoxIcon.Information);

db.closeConnection();

metroGrid1.Rows.Clear();

LoadData();

}

}

private void combo1\_click(object sender, MouseEventArgs e)

{

trainer trainer = new trainer();

List<trainer> trainers = trainer.return\_trainers();

metroComboBox1.DataSource = trainers;

metroComboBox1.DisplayMember = "fullname";

metroComboBox1.ValueMember = "Id";

}

private void combo2\_click(object sender, MouseEventArgs e)

{

activity activity = new activity();

List<activity> activities = activity.return\_activities();

metroComboBox2.DataSource = activities;

metroComboBox2.DisplayMember = "activity\_name";

metroComboBox2.ValueMember = "Id";

}

private void metroButton3\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(metroComboBox1.Text) | string.IsNullOrWhiteSpace(metroComboBox1.Text) |

string.IsNullOrEmpty(metroComboBox2.Text) | string.IsNullOrWhiteSpace(metroComboBox2.Text) |

string.IsNullOrEmpty(metroComboBox3.Text) | string.IsNullOrWhiteSpace(metroComboBox3.Text) |

string.IsNullOrEmpty(metroComboBox4.Text) | string.IsNullOrWhiteSpace(metroComboBox4.Text) |

string.IsNullOrEmpty(metroComboBox5.Text) | string.IsNullOrWhiteSpace(metroComboBox5.Text) |

string.IsNullOrEmpty(metroComboBox6.Text) | string.IsNullOrWhiteSpace(metroComboBox6.Text))

{

MessageBox.Show("Введены не верные значения", "Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

else

{

try

{

SqlCommand sqlCommand = new SqlCommand("INSERT INTO [schedule] (trainer\_id, activity\_id, date," +

" start\_time, end\_time) VALUES (@trainer\_id, @activity\_id, @date," +

" @start\_time, @end\_time)", db.GetConnection());

string date = metroDateTime1.Value.ToShortDateString();

string cdate = date.Substring(3, 2) + '.' + date.Substring(0, 2) + '.' + date.Substring(6, 4);

sqlCommand.Parameters.AddWithValue("trainer\_id", metroComboBox1.SelectedValue.ToString());

sqlCommand.Parameters.AddWithValue("activity\_id", metroComboBox2.SelectedValue.ToString());

sqlCommand.Parameters.AddWithValue("date", cdate);

sqlCommand.Parameters.AddWithValue("start\_time", metroComboBox3.Text.ToString() + ":" + metroComboBox4.Text.ToString());

sqlCommand.Parameters.AddWithValue("end\_time", metroComboBox6.Text.ToString() + ":" + metroComboBox5.Text.ToString());

db.openConnection();

if (sqlCommand.ExecuteNonQuery() == 1)

MessageBox.Show("Вы успешно добавили занятие в расписание", "Добавление занятие", MessageBoxButtons.OK,

MessageBoxIcon.Information);

else

MessageBox.Show("Добавление не прошло");

db.closeConnection();

}

catch (Exception)

{

MessageBox.Show("Добавление не прошло");

}

}

}

private void metroButton4\_Click(object sender, EventArgs e)

{

LoadActivity();

}

private void metroButton5\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("Удалить занятие?",

"Сообщение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information);

if (result == DialogResult.Yes)

{

db.openConnection();

SqlCommand command = new SqlCommand("delete from [Activity] where Id ='" + metroTextBox7.Text + "'", db.GetConnection());

command.ExecuteNonQuery();

MessageBox.Show("Вы успешно удалили занятие", "Удаление занятия", MessageBoxButtons.OK,

MessageBoxIcon.Information);

db.closeConnection();

LoadActivity();

}

}

private void metroButton6\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(metroTextBox8.Text) | string.IsNullOrWhiteSpace(metroTextBox8.Text) |

string.IsNullOrEmpty(metroTextBox10.Text) | string.IsNullOrWhiteSpace(metroTextBox10.Text))

{

MessageBox.Show("Введены не верные значения", "Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

else

{

try

{

SqlCommand sqlCommand = new SqlCommand("INSERT INTO [Activity] (activity\_name, cost) VALUES (@activity\_name, @cost)", db.GetConnection());

sqlCommand.Parameters.AddWithValue("activity\_name", metroTextBox8.Text);

sqlCommand.Parameters.AddWithValue("cost", Convert.ToInt32(metroTextBox10.Text));

db.openConnection();

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Вы успешно добавили занятие", "Добавление занятия", MessageBoxButtons.OK,

MessageBoxIcon.Information);

db.closeConnection();

LoadActivity();

}

catch (Exception)

{

MessageBox.Show("Добавление не прошло");

}

}

}

private void metro7\_click(object sender, MouseEventArgs e)

{

activity activity = new activity();

List<activity> activities = activity.return\_activities();

metroComboBox7.DataSource = activities;

metroComboBox7.DisplayMember = "activity\_name";

metroComboBox7.ValueMember = "Id";

}

private void metroComboBox7\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void metroButton7\_Click(object sender, EventArgs e)

{

metroListView2.Items.Clear();

schedule schedule = new schedule();

schedule.fill\_listview(metroListView2, metroComboBox7.SelectedValue.ToString());

}

private void DirectorForm\_SizeChanged(object sender, EventArgs e)

{

metroPanel1.Location = new System.Drawing.Point(((this.Width - metroPanel1.Width)/2)-10, 0);

}

private void metroButton8\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("Удалить занятие?",

"Сообщение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information);

if (result == DialogResult.Yes)

{

db.openConnection();

SqlCommand command = new SqlCommand("delete from [schedule] where Id ='" + metroTextBox9.Text + "'", db.GetConnection());

command.ExecuteNonQuery();

MessageBox.Show("Вы успешно удалили занятие", "Удаление занятия", MessageBoxButtons.OK,

MessageBoxIcon.Information);

db.closeConnection();

LoadActivity();

}

}

private void metroButton9\_Click(object sender, EventArgs e)

{

Services services = new Services();

services.Show();

}

private void metroButton10\_Click(object sender, EventArgs e)

{

Congestion congestion = new Congestion();

congestion.Show();

}

private void metroButton11\_Click(object sender, EventArgs e)

{

Discount discount = new Discount();

discount.Show();

}

private void metroButton12\_Click(object sender, EventArgs e)

{

Profit profit = new Profit();

profit.Show();

}

private void DirectorForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void metroButton13\_Click(object sender, EventArgs e)

{

LoadData();

}

}

}

Файл Discount.cs

namespace FitnessClub

{

public partial class Discount : MetroFramework.Forms.MetroForm

{

public Discount()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

metroListView1.StyleManager = metroStyleManager1;

LoadData();

}

public void LoadData()

{

DB db = new DB();

metroListView1.Items.Clear();

SqlCommand command = new SqlCommand("select Name + ' ' + Surname as 'fullname', Count from (select client\_id, count(distinct schedule.activity\_id) as 'Count' from clients\_activities join schedule on clients\_activities.activity\_id = schedule.Id group by client\_id) as G join [User] on G.client\_id = [User].Id where Count>2", db.GetConnection());

db.openConnection();

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[2]);

data[data.Count - 1][0] = reader[0].ToString();

if (Convert.ToInt32(reader[1].ToString()) == 3)

data[data.Count - 1][1] = "5%";

else if (Convert.ToInt32(reader[1].ToString()) == 4)

data[data.Count - 1][1] = "10%";

else

data[data.Count - 1][1] = "15%";

}

reader.Close();

db.closeConnection();

foreach (string[] s in data)

{

ListViewItem lvi = new ListViewItem(s[0]);

lvi.SubItems.Add(s[1]);

metroListView1.Items.Add(lvi);

}

}

}

}

Файл Form1.cs

namespace FitnessClub

{

public partial class Form1 : MetroFramework.Forms.MetroForm

{

public Form1()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void metroButton2\_Click(object sender, EventArgs e)

{

Registration registration = new Registration();

registration.Show();

}

private void metroButton1\_Click(object sender, EventArgs e)

{

DB db = new DB();

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT \* FROM [User] WHERE Login = '" + metroTextBox1.Text + "' and Password = '" + metroTextBox2.Text + "'", db.GetConnection());

DataTable dt = new DataTable();

sqlDataAdapter.Fill(dt);

if (dt.Rows.Count > 0)

{

DB.Id = Convert.ToInt32(dt.Rows[0][0]);

DB.Name = dt.Rows[0][1].ToString();

DB.Surname = dt.Rows[0][2].ToString();

if (dt.Rows[0][6].ToString()=="Тренер")

{

TrainerForm f = new TrainerForm();

f.Show();

this.Hide();

}

else if (dt.Rows[0][6].ToString() == "Директор")

{

DirectorForm f = new DirectorForm();

f.Show();

this.Hide();

}

else if (dt.Rows[0][6].ToString() == "Клиент")

{

ClientForm f = new ClientForm();

f.Show();

this.Hide();

}

else

{

Accountant f = new Accountant();

f.Show();

this.Hide();

}

}

else

{

MessageBox.Show("Не правильный логин или пароль");

}

}

}

}

Файл Profit.cs

namespace FitnessClub

{

public partial class Profit : MetroFramework.Forms.MetroForm

{

public Profit()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

metroListView1.StyleManager = metroStyleManager1;

LoadData();

}

public void LoadData()

{

DB db = new DB();

metroListView1.Items.Clear();

SqlCommand command = new SqlCommand("select activity\_name, sum(cost) as 'cost' from clients\_activities join schedule on clients\_activities.activity\_id = schedule.Id join Activity on schedule.activity\_id = Activity.Id group by activity\_name", db.GetConnection());

db.openConnection();

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

int profit = 0;

while (reader.Read())

{

profit += Convert.ToInt32(reader[1].ToString());

data.Add(new string[2]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

}

reader.Close();

db.closeConnection();

foreach (string[] s in data)

{

ListViewItem lvi = new ListViewItem(s[0]);

lvi.SubItems.Add(s[1]);

metroListView1.Items.Add(lvi);

}

metroLabel1.Text = profit.ToString();

}

}

}

Файл Registration.cs

namespace FitnessClub

{

public partial class Registration : MetroFramework.Forms.MetroForm

{

DB db = new DB();

public Registration()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

}

private void Registration\_Load(object sender, EventArgs e)

{

}

public bool checkLogin()

{

bool f = false;

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT count(\*) FROM [User] WHERE Login = '" + metroTextBox3.Text + "'", db.GetConnection());

DataTable dt = new DataTable();

sqlDataAdapter.Fill(dt);

if (dt.Rows[0][0].ToString() == "1")

{

f = true;

}

else

{

f = false;

}

return f;

}

public bool checkTrener()

{

db.openConnection();

bool f = false;

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT count(\*) FROM [Trainers] WHERE passport = '" + metroTextBox6.Text + "'", db.GetConnection());

DataTable dt = new DataTable();

sqlDataAdapter.Fill(dt);

if (dt.Rows[0][0].ToString() == "1")

{

f = false;

}

else

{

f = true;

}

db.closeConnection();

return f;

}

private void metroButton1\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(metroTextBox5.Text) | string.IsNullOrWhiteSpace(metroTextBox5.Text) |

string.IsNullOrEmpty(metroTextBox4.Text) | string.IsNullOrWhiteSpace(metroTextBox4.Text) |

string.IsNullOrEmpty(metroTextBox3.Text) | string.IsNullOrWhiteSpace(metroTextBox3.Text) |

string.IsNullOrEmpty(metroTextBox2.Text) | string.IsNullOrWhiteSpace(metroTextBox2.Text) |

string.IsNullOrEmpty(metroTextBox1.Text) | string.IsNullOrWhiteSpace(metroTextBox1.Text))

{

MessageBox.Show("Введены не верные значения", "Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

else if ((metroTextBox1.Text != metroTextBox2.Text))

{

MessageBox.Show("Пароли не совпадают", "Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

else if (checkLogin())

{

MessageBox.Show("Такой логин уже существует", "Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

else if ((string.IsNullOrEmpty(metroTextBox6.Text) | string.IsNullOrWhiteSpace(metroTextBox6.Text)) & (metroComboBox1.Text == "Тренер"))

{

MessageBox.Show("Введите паспортные данные", "Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

else if (checkTrener() & (metroComboBox1.Text == "Тренер"))

{

MessageBox.Show("Вас еще не приняли на работу", "Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

else

{

try

{

SqlCommand sqlCommand = new SqlCommand("INSERT INTO [User] (Name, Surname, Login," +

" Password, DateOfRegistration, position, passport) VALUES (@Name, @Surname, @Login, @Password," +

" @DateOfRegistration, @position, @passport)", db.GetConnection());

sqlCommand.Parameters.AddWithValue("Name", metroTextBox5.Text);

sqlCommand.Parameters.AddWithValue("Surname", metroTextBox4.Text);

sqlCommand.Parameters.AddWithValue("Login", metroTextBox3.Text);

sqlCommand.Parameters.AddWithValue("Password", metroTextBox2.Text);

sqlCommand.Parameters.AddWithValue("position", metroComboBox1.Text);

sqlCommand.Parameters.AddWithValue("DateOfRegistration", DateTime.Now);

sqlCommand.Parameters.AddWithValue("passport", metroTextBox6.Text);

db.openConnection();

if (sqlCommand.ExecuteNonQuery() == 1)

MessageBox.Show("Вы успешно зарегестрировались", "Регистриция", MessageBoxButtons.OK,

MessageBoxIcon.Information, MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

else

MessageBox.Show("Регистрация не прошла");

db.closeConnection();

this.Close();

}

catch (Exception)

{

MessageBox.Show("Регистрация не прошла");

}

}

}

}

}

Файл schedule.cs

namespace FitnessClub

{

class schedule

{

DB db = new DB();

public List<string[]> return\_schedule(string id)

{

SqlCommand command = new SqlCommand("SELECT \* FROM [schedule] where activity\_id = '" + id + "'", db.GetConnection());

trainer trainer = new trainer();

List<trainer> trainers = trainer.return\_trainers();

db.openConnection();

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[6]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = trainers.Find(item => item.Id == (Convert.ToInt32(reader[1].ToString()))).fullname;

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString().Substring(0, 5);

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

}

reader.Close();

db.closeConnection();

return data;

}

public List<string[]> return\_schedule\_trainer(string id)

{

SqlCommand command = new SqlCommand("SELECT \* FROM [schedule] where trainer\_id = '" + id + "'", db.GetConnection());

activity activity = new activity();

List<activity> activities = activity.return\_activities();

db.openConnection();

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[6]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = activities.Find(item => item.Id == (Convert.ToInt32(reader[2].ToString()))).activity\_name;

data[data.Count - 1][3] = reader[3].ToString().Substring(0, 5);

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

}

reader.Close();

db.closeConnection();

return data;

}

public List<string[]> return\_schedule\_client()

{

SqlCommand command = new SqlCommand("select B.Id, name + ' ' + surname as fullname, activity\_name, date, start\_time, end\_time, cost, payment " +

"from (select A.Id, trainer\_id, date, start\_time, end\_time, activity\_name, cost, payment " +

"from (select [clients\_activities].Id, schedule.activity\_id, schedule.trainer\_id,schedule.date, schedule.start\_time, schedule.end\_time, payment " +

"from [clients\_activities] join schedule on [clients\_activities].activity\_id = schedule.Id " +

"where [clients\_activities].client\_id = @client\_id) as A join Activity on A.activity\_id = Activity.Id) as B " +

"join Trainers on B.trainer\_id = Trainers.Id", db.GetConnection());

command.Parameters.AddWithValue("@client\_id", DB.Id);

db.openConnection();

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[8]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString().Substring(0, 5);

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

data[data.Count - 1][6] = reader[6].ToString();

data[data.Count - 1][7] = reader[7].ToString();

}

reader.Close();

db.closeConnection();

return data;

}

public List<string> return\_date\_list()

{

SqlCommand command1 = new SqlCommand("SELECT date, count(id) FROM [schedule] group by date ", db.GetConnection());

db.openConnection();

SqlDataReader reader1 = command1.ExecuteReader();

List<string> date\_list = new List<string>();

while (reader1.Read())

{

date\_list.Add(reader1[0].ToString().Substring(0, 5));

}

reader1.Close();

db.closeConnection();

return date\_list;

}

public List<string> return\_date\_list\_client()

{

SqlCommand command1 = new SqlCommand("SELECT date, count(B.Id)" +

"from (select A.Id, trainer\_id, date, start\_time, end\_time, activity\_name, cost, payment " +

"from (select [clients\_activities].Id, schedule.activity\_id, schedule.trainer\_id,schedule.date, schedule.start\_time, schedule.end\_time, payment " +

"from [clients\_activities] join schedule on [clients\_activities].activity\_id = schedule.Id " +

"where [clients\_activities].client\_id = @client\_id) as A join Activity on A.activity\_id = Activity.Id) as B " +

"join Trainers on B.trainer\_id = Trainers.Id group by date ", db.GetConnection());

command1.Parameters.AddWithValue("@client\_id", DB.Id);

db.openConnection();

SqlDataReader reader1 = command1.ExecuteReader();

List<string> date\_list = new List<string>();

while (reader1.Read())

{

date\_list.Add(reader1[0].ToString().Substring(0, 5));

}

reader1.Close();

db.closeConnection();

return date\_list;

}

public void lv\_add\_group(List<string> date\_list, ListView lv)

{

string k = "";

int m;

string[] month = new string[12] { "Января", "Февраля", "Марта", "Апреля", "Мая", "Июня", "Июля", "Августа", "Сентября", "Октября", "Ноября", "Декабря", };

foreach (string i in date\_list)

{

m = Convert.ToInt32(i.Substring(3, 2));

if (Convert.ToInt32(i.Substring(0, 2)) > 9)

k = i.Substring(0, 2) + " " + month[Convert.ToInt32(m) - 1];

else

k = i.Substring(1, 1) + " " + month[Convert.ToInt32(m) - 1];

lv.Groups.Add(i, k);

k = "";

}

}

public void lv\_add\_item(List<string[]> data, ListView lv)

{

foreach (string[] s in data)

{

ListViewItem lvi = new ListViewItem(s[0]);

lvi.Group = lv.Groups[s[3]];

lvi.SubItems.Add(s[1]);

lvi.SubItems.Add(s[4]);

lvi.SubItems.Add(s[5]);

lv.Items.Add(lvi);

}

}

public void lv\_add\_item\_trainer(List<string[]> data, ListView lv)

{

foreach (string[] s in data)

{

ListViewItem lvi = new ListViewItem(s[2]);

lvi.Group = lv.Groups[s[3]];

lvi.SubItems.Add(s[4]);

lvi.SubItems.Add(s[5]);

lv.Items.Add(lvi);

}

}

public void lv\_add\_item\_client(List<string[]> data, ListView lv)

{

foreach (string[] s in data)

{

ListViewItem lvi = new ListViewItem(s[0]);

lvi.Group = lv.Groups[s[3]];

lvi.SubItems.Add(s[1]);

lvi.SubItems.Add(s[2]);

lvi.SubItems.Add(s[4]);

lvi.SubItems.Add(s[5]);

lvi.SubItems.Add(s[6]);

/\*if (s[7] == "0")

lvi.SubItems.Add("Нет");

else

lvi.SubItems.Add("Есть");\*/

lv.Items.Add(lvi);

}

}

public void fill\_listview(ListView lv, string activity\_id)

{

List<string[]> data = return\_schedule(activity\_id);

List<string> date\_list = return\_date\_list();

lv\_add\_group(date\_list, lv);

lv\_add\_item(data, lv);

}

public void fill\_listview\_trainer(ListView lv, string activity\_id)

{

List<string[]> data = return\_schedule\_trainer(activity\_id);

List<string> date\_list = return\_date\_list();

lv\_add\_group(date\_list, lv);

lv\_add\_item\_trainer(data, lv);

}

public void fill\_listview\_client(ListView lv)

{

List<string[]> data = return\_schedule\_client();

List<string> date\_list = return\_date\_list\_client();

lv\_add\_group(date\_list, lv);

lv\_add\_item\_client(data, lv);

}

}

}

Файл Services.cs

namespace FitnessClub

{

public partial class Services : MetroFramework.Forms.MetroForm

{

DB db = new DB();

public Services()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

LoadData();

}

public void LoadData()

{

metroListView1.Items.Clear();

SqlCommand command = new SqlCommand("select activity\_name, Count from (select schedule.activity\_id, count(clients\_activities.Id) as 'Count' from clients\_activities inner join schedule on clients\_activities.activity\_id = schedule.Id group by schedule.activity\_id) as A join Activity on A.activity\_id = Activity.Id order by Count DESC", db.GetConnection());

db.openConnection();

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[2]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

}

reader.Close();

db.closeConnection();

foreach (string[] s in data)

{

ListViewItem lvi = new ListViewItem(s[0]);

lvi.SubItems.Add(s[1]);

metroListView1.Items.Add(lvi);

}

}

}

}

Файл trainer.cs

namespace FitnessClub

{

class trainer

{

public int Id { get; set; }

public string fullname { get; set; }

DB db = new DB();

public List<trainer> return\_trainers()

{

SqlCommand sqlCommand = new SqlCommand("SELECT Id, Name, Surname FROM [Trainers] ", db.GetConnection());

db.openConnection();

SqlDataReader reader = sqlCommand.ExecuteReader();

List<trainer> trainers = new List<trainer>();

while (reader.Read())

{

trainers.Add(new trainer());

trainers[trainers.Count - 1] = new trainer { Id = Convert.ToInt32(reader[0]), fullname = reader[1].ToString() + " " + reader[2].ToString() };

}

reader.Close();

db.closeConnection();

return trainers;

}

}

}

Файл TrainerForm.cs

namespace FitnessClub

{

public partial class TrainerForm : MetroFramework.Forms.MetroForm

{

DB db = new DB();

public string id = DB.Id.ToString();

public TrainerForm()

{

InitializeComponent();

this.StyleManager = metroStyleManager1;

metroListView1.StyleManager = metroStyleManager1;

metroListView2.StyleManager = metroStyleManager1;

metroTabControl1.StyleManager = metroStyleManager1;

LoadData();

}

private void LoadData()

{

DB db = new DB();

metroListView1.Items.Clear();

trainer trainer = new trainer();

List<trainer> trainers = trainer.return\_trainers();

int id = trainers.Find(item => item.fullname == (DB.Name+' '+DB.Surname)).Id;

SqlCommand sqlCommand = new SqlCommand("select Name + ' ' + Surname as Name, review from Reviews join [User] on [User].Id = Reviews.client\_id where trainer\_id = @trainer\_id", db.GetConnection());

sqlCommand.Parameters.AddWithValue("@trainer\_id", id);

db.openConnection();

SqlDataReader reader = sqlCommand.ExecuteReader();

while (reader.Read())

{

ListViewItem lvi = new ListViewItem(reader[0].ToString());

lvi.SubItems.Add(reader[1].ToString());

metroListView1.Items.Add(lvi);

}

reader.Close();

db.closeConnection();

}

private void TrainerForm\_Load(object sender, EventArgs e)

{

metroLabel1.Text = DB.Name;

metroLabel2.Text = DB.Surname;

}

private void metroButton1\_Click(object sender, EventArgs e)

{

LoadData();

}

private void metroButton2\_Click(object sender, EventArgs e)

{

metroListView2.Items.Clear();

schedule schedule = new schedule();

schedule.fill\_listview\_trainer(metroListView2, DB.Id.ToString());

}

private void TrainerForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}