МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Фитнес тренер»

Исполнитель

студент 2 курса 6 группы Кутас Кирилл Дмитриевич

(Ф.И.О.)

Руководитель работы асс. Северинчик Н. А.

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

Курсовой проект защищен с оценкой

Председатель Пацей Н. В.

(подпись)

Минск 2019

Содержание

[1.Постановка задачи 4](#_Toc9588617)

[1.1 Алгоритмы решения 4](#_Toc9588618)

[2. Разработка архитектуры проекта 6](#_Toc9588619)

[2.1 Обобщённая структура 6](#_Toc9588620)

[2.2 Диаграммы UML, взаимосвязь всех компонентов 7](#_Toc9588621)

[3. Разработка функциональной модели и модели данных ПС 9](#_Toc9588622)

[3.1 Модель базы данных 9](#_Toc9588623)

[3.2 Выполняемые функции 11](#_Toc9588624)

[3.2.1 Регистрация и авторизация 11](#_Toc9588625)

[3.2.2 Реализация основного окна 13](#_Toc9588626)

[4. Руководство пользователя 14](#_Toc9588627)

[Заключение 20](#_Toc9588628)

[Список используемых источников 21](#_Toc9588629)

[Приложение А 22](#_Toc9588630)

[Приложение Б 25](#_Toc9588631)

[Приложение В 29](#_Toc9588632)

**Введение**

Приложение FitnessTrainer – служебный инструмент пользовательского интерфейса для выполнения упражнений и выбора еды. С помощью данного инструмента можно выбирать упражнения и еду, следить за прогрессом, получить статистику на почту.

Темой курсового проекта является разработка программного средства «Фитнес тренер». Данное ПО должно позволить пользователям выбирать подходящее им упражнение и блюдо по их желанию. Программа уведомляет обо всех ошибках, связанных с регистрацией, входом. А так же об успешном выполнении ранее перечисленных функций.

Целью курсового проекта является проектирование и реализация программного средства «FitnessTrainer».

# 1.Постановка задачи

Основной задачей курсового проекта является разработка программного средства, позволяющего пользователю хранить информацию в базе данных, а так же добавлять новую информацию в уже имеющуюся.

Функционально ПС должно выполнять следующие задачи:

* выполнять авторизацию и регистрацию пользователя;
* сохранять информацию о пользователе в базе данных;
* возможность просматривать профиль пользователя;
* отправка статистики на почту;
* возможность пользователю просматривать упражнения или блюда.

## 1.1 Алгоритмы решения

В разработке приложения были использованы нижеперечисленные технологии:

* ADO.NET;
* Microsoft SQL Server;
* WPF.

ADO.NET (ActiveX Data Object для .NET) – технология, предоставляющая доступ к данным для приложений, основанных на Microsoft .NET. Является не развитием более ранней технологии ADO, а самостоятельной технологией, частью фреймворка .NET. В отличие от классической ADO, которая была в основном предназначена для тесно связанных клиент-серверных систем, ADO.NET больше нацелена на автономную работу с помощью объектов DataSet. Эти типы представляют локальные копии любого количества взаимосвязанных таблиц данных, каждая из которых содержит набор строк и столбцов. Объекты DataSet позволяют вызывающей сборке (наподобие веб-страницы или программы, выполняющейся на настольном компьютере) работать с содержимым DataSet, изменять его, не требуя подключения к источнику данных, и отправлять обратно блоки измененных данных для обработки с помощью соответствующего адаптера данных. Фундаментальное различие между классической ADO и ADO.NET состоит в том, что ADO.NET является управляемой кодовой библиотекой, и, значит, подчиняется тем же правилам, что и любая управляемая библиотека. Типы, составляющие ADO.NET, используют протокол управления памятью CLR, принадлежат к той же системе типов (классы, интерфейсы, перечисления, структуры и делегаты), и доступ к ним возможен с помощью любого языка .NET. Классы ADO.NET находятся в сборке System.Data.dll

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Microsoft SQL Server в качестве языка запросов использует версию SQL, получившую название Transact-SQL (сокращённо T-SQL), являющуюся реализацией SQL-92 (стандарт ISO для SQL) с множественными расширениями. T-SQL позволяет использовать дополнительный синтаксис для хранимых процедур и обеспечивает поддержку транзакций (взаимодействие базы данных с управляющим приложением). Microsoft SQL Server и Sybase ASE для взаимодействия с сетью используют протокол уровня приложения под названием Tabular Data Stream (TDS, протокол передачи табличных данных). Протокол TDS также был реализован в проекте FreeTDS с целью обеспечить различным приложениям возможность взаимодействия с базами данных Microsoft SQL Server и Sybase.

Windows Presentation Foundation (WPF) – система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

# 2. Разработка архитектуры проекта

## 2.1 Обобщённая структура

Решение CourseProject представлено одним проектом CourseProject, имеющим структуру, представленную на рисунке 2.1.

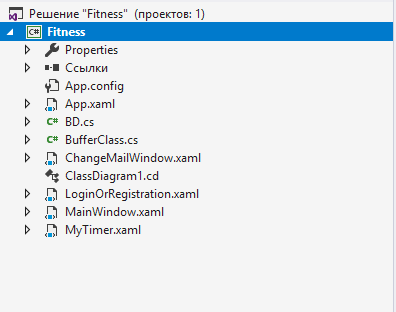


Рисунок 2.1 – Структура проекта

|  |  |
| --- | --- |
| Properties | Свойства проекта, содержит информацию о сборке, используемых ресурсах и настройках |
| Ссылки | Перечень сборок, используемых в проекте |
| Properties | Папка со свойствами проекта |
| App.config | Файл с параметрами проекта |
| App.xaml | Класс Application |
| BD.cs | Класс для управления базой данных |
| BufferClass.cs | Класс в котором хранятся данные о текущем пользователе |
| MainWindow.xaml | Основное окно приложения |
| ChangeMailWindow.xaml | Окно для смена почты |
| MyTimer.xaml | Окно таймера |

# 2.2 Диаграммы UML, взаимосвязь всех компонентов

Диаграмма UML – это графическое представление набора элементов, изображаемое чаще всего в виде графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации. Основная цель диаграмм – визуализация разрабатываемой системы с разных точек зрения.

На диаграмме 2.2 представлен принцип работы приложения с точки зрения пользователя.

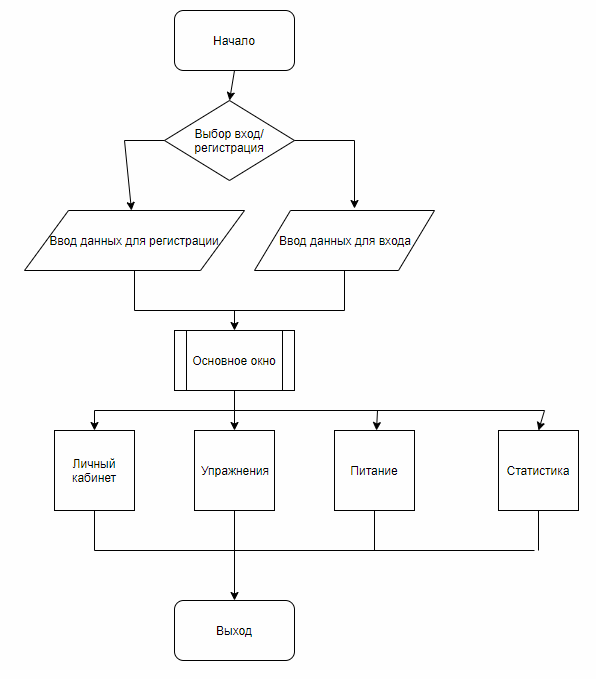


Рисунок 2.2 – Принцип работы пользователя

На рисунке 2.3 представлена диаграмма классов

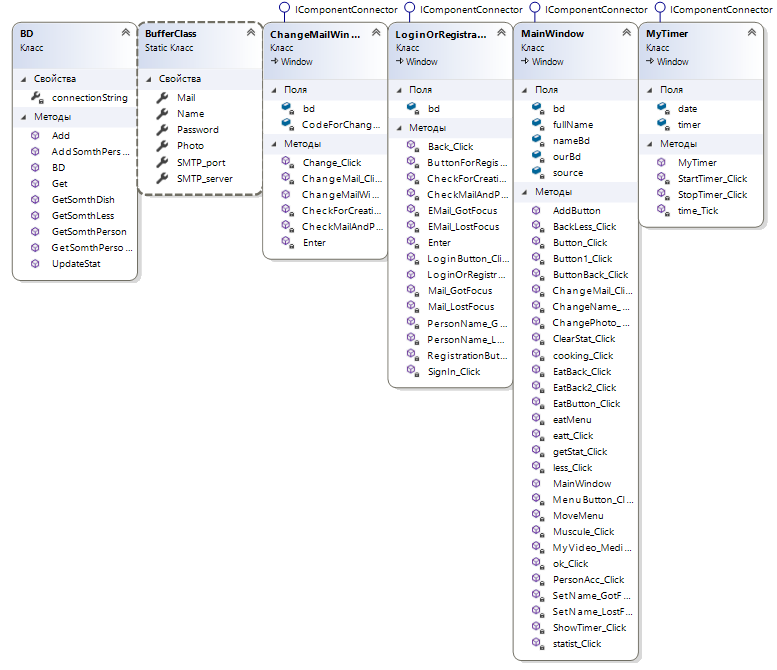


Рисунок 2.3 – диаграмма классов

# 3. Разработка функциональной модели и модели данных ПС

## 3.1 Модель базы данных

Для реализации поставленной задачи была создана база данных Fitness. Для её создания использовалась система управления реляционными базами данных Microsoft SQL Server 2017.

База данных состоит из 23 таблиц, перечисленных ниже (рисунок 3.1).

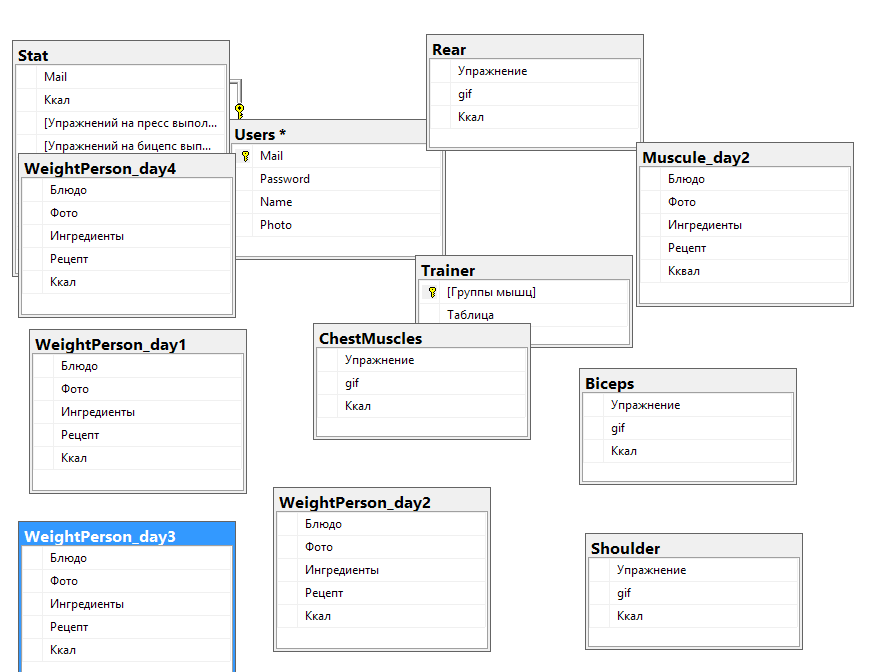


Рисунок 3.1 – Взаимосвязь таблиц базы данных

Диаграмма на рисунке 3.2 иллюстрирует структуру таблицы «Users» , которая содержит информацию о пользователе.

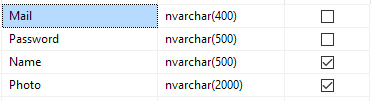


Рисунок 3.2 – Структура таблицы «Users»

Для хранения информации о группах мышц была создана таблица Trainer, представленная на рисунке 3.3.



Рисунок 3.3 – Структура таблицы «Trainer»

Для хранения информации об упражнениях были созданы однотипные таблицы, одна из них Press, представленная на рисунке 3.4.

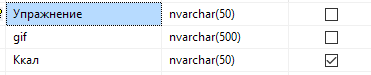


Рисунок 3.4 – Структура таблицы «Press»

Для хранения информации об истории изменений пользователя была создана таблица Stat, представленная на рисунке 3.5.

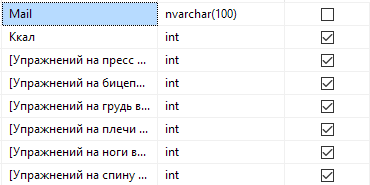


Рисунок 3.5 – Структура таблицы «Stat»

Для хранения блюд были созданы были созданы однотипные таблицы, одна из них Muscule\_day1, представленная на рисунке 3.5.

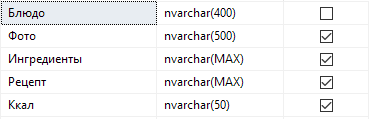


Рисунок 3.6 – Структура таблицы «Muscule\_day1»

## 3.2 Выполняемые функции

Данное приложение предоставляет минимальные требования к выполнению функций приложения фитнесс тренера и выполняет следующие функции:

* регистрация с последующей авторизацией пользователя;
* возможность изменить фото, имя, почту;
* просмотр статистики;
* выбор упражнений;
* выбор блюд;
* отправка статистики на почту.

### 3.2.1 Регистрация и авторизация

Регистрация пользователя происходит в окне LoginOrRegistration, имеющем следующую структуру, отображенную на рисунке 3.7.

Имеются поле bd, которое является объектом класса BD. Также методы: ButtonForRegistration\_Click – метод регистрации пользователя. События на изменения TextBox и PasswordBox, которые нужны для обработки вводимых данных и уведомления пользователя, если он вводит неверные данные – тогда появится Label с уведомлением о том, что пользователь вводит неверные данные.

Класс содержит метод ButtonForRegistration\_Click, который является главным в процессе регистрации.

Метод содержит метод проверки логина на повтор – CheckForCreating. Если пользователя с таким логином не существует – то осуществляется регистрация пользователя.

Листинг всех этих методов представлен в приложении A.

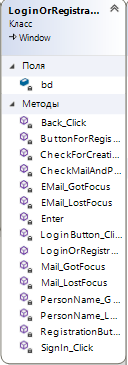


Рисунок 3.7 – Структура класса LoginOrRegistration

Ниже приведён скриншот метода ButtonForRegistration\_Click (Рисунок 3.8)

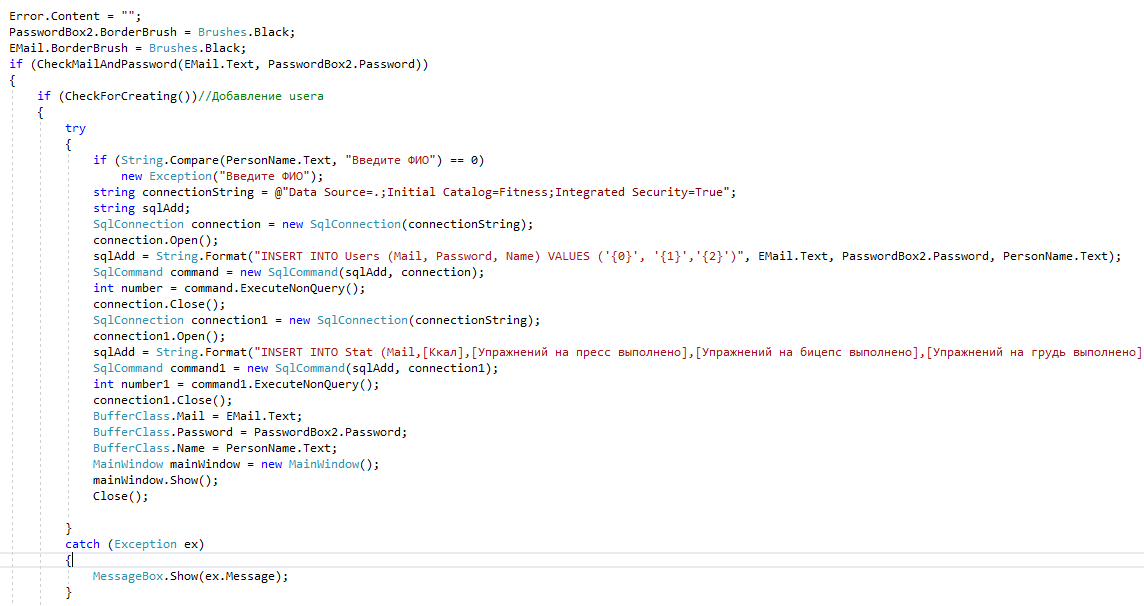


Рисунок 3.8 – Метод ButtonForRegistration\_Click

Авторизация пользователя происходит в том же.

В классе представлены методы SignIn\_Click – осуществляющий процесс авторизация пользователя. Методы представлены на рисунке 3.9.



Рисунок 3.9 – Метод класса LoginOrRegistration

Листинг этих методов представлен в приложении A. Блок-схемы этих алгоритмов представлены в приложении В.

### 3.2.2 Реализация основного окна

Личный кабинет пользователя реализован с помощью методов ChangePhoto\_Click – позволяет изменить фотографию пользователя, ChangeName\_Click – позволяет изменять имя пользователю, ChangeMail\_Click – позволяет изменить почту пользователю. Также имеются обработчики событий изменения текста в Text\_Box. Ниже, на рисунке 3.10 представлен класс MainWindow.

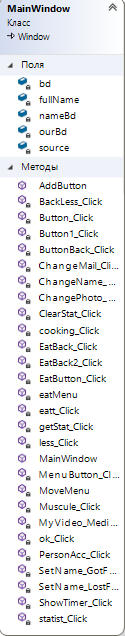


Рисунок 3.10 – Структура класса MainWindow

# 4. Руководство пользователя

При запуске приложения открывается окно авторизации (рисунок 4.1), на котором имеются кнопки выбора регистрации или авторизации.

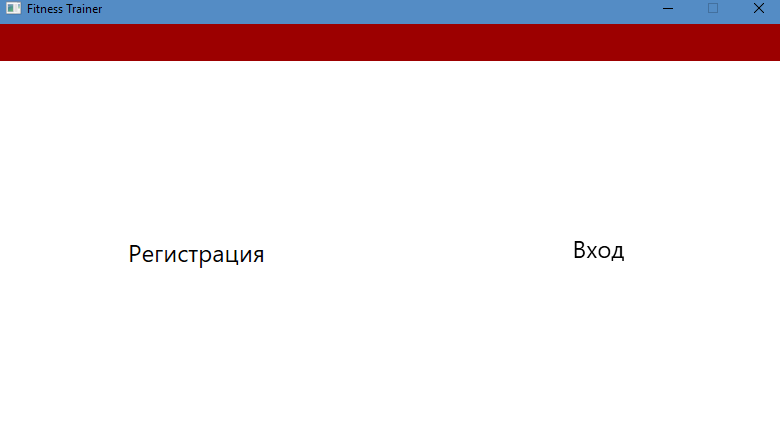


Рисунок 4.1 – Окно авторизации/регистрации

Если у пользователя нет аккаунта, то он пройти регистрацию, нажав на кнопку «Регистрация». Затем откроется окно регистрации, где пользователю будет предложено заполнить необходимую информацию для дальнейшего использования программного обеспечения (рисунок 4.2).

После успешного прохождения регистрации и авторизации откроется главное окно приложения (Рисунок 4.3).

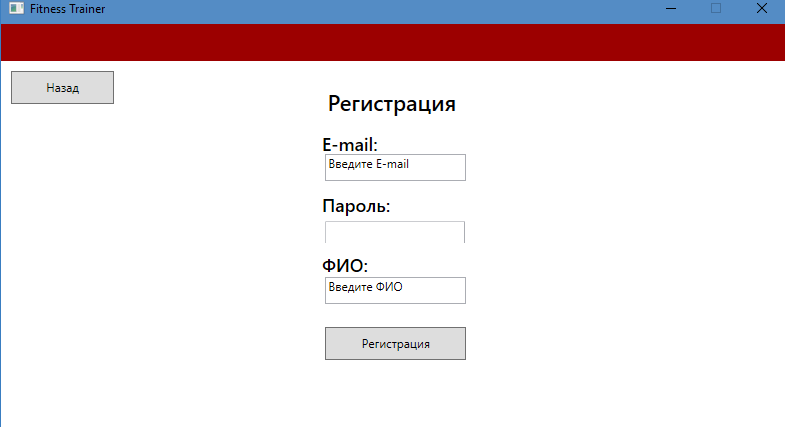


Рисунок 4.2 – Окно регистрации

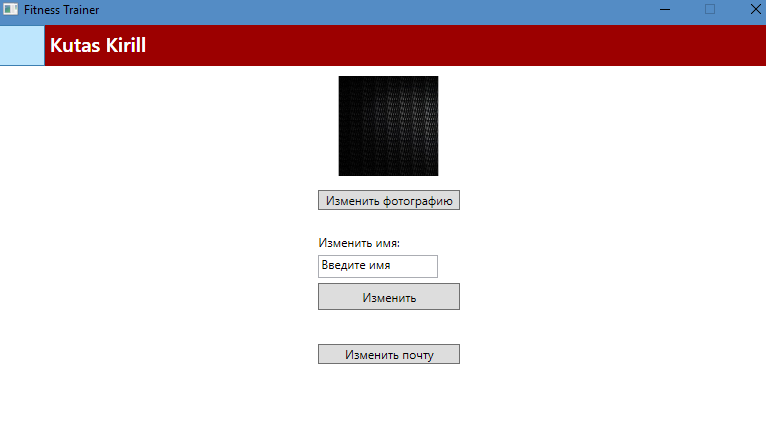


Рисунок 4.3 – Основное окно пользователя

При нажатии на кнопку “Изменить фотографию” открывается окно выбора файла. При нажатии на кнопку “Изменить”, мы изменим имя пользователя на то, что будет введено выше. Кнопка “Изменить почту” открывает мадальное окно, в котором нужно ввести код для смены почты, который будет отправлен на действующую почту (Рисунок 4.4).

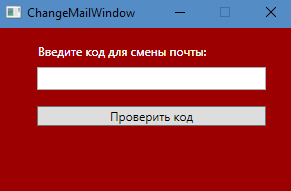


Рисунок 4.4 – Мадальное окно для смены почты

Слева вверху у нас имеется кнопка меню, при нажатии на которую слева выплывает меню (Рисунок 4.5). Там у нас содержаться кнопки выбора: “Личный кабинет”,”Упражнение”,”Питание”,”Статистика”.

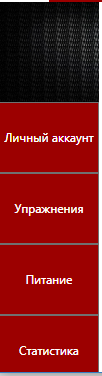


Рисунок 4.5 – Меню

При переходе в упражнения мы получаем список групп мышц (Рисунок 4.6), который в себе содержит упражнения на эти группы мышц (Рисунок 4.7).

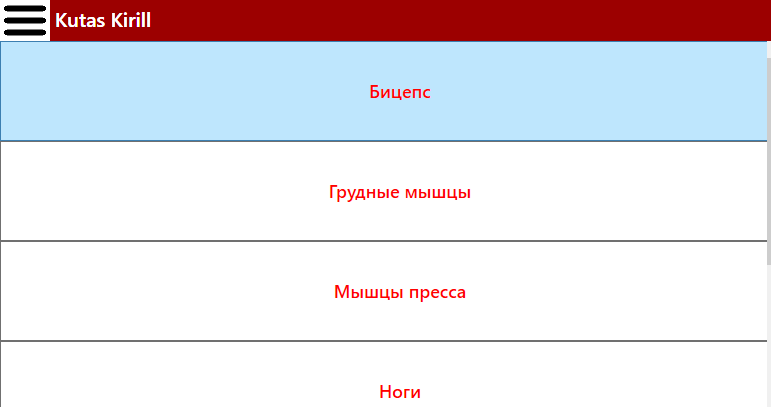


Рисунок 4.6 – Группы мышц

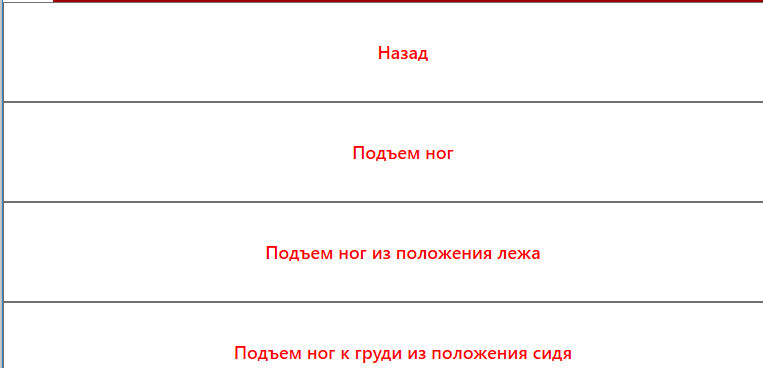


Рисунок 4.7 – Упражнения

При нажатии на какое-либо упражнение у нас открывается окно с этим упражнением (Рисунок 4.8). Там у нас содержится название упражнения, видео, которое дает нам понять, как выполнять упражнение, количество подходов и повторений, которые вычисляются на основе того, сколько упражнений на данную группу мышц выполнил. Так же у нас содержится две кнопки: “Выполнить” и ”Таймер”. При нажатии на кнопку “Выполнить”, данная кнопка пропадает и вместо нее появляется текст и количество выполненных упражнений на данную группу мышц увеличивается. Так же уменьшается количество ккал в таблице Stat. При нажатии на кнопку Таймер” открывается мадальное окно с таймером (Рисунок 4.9).

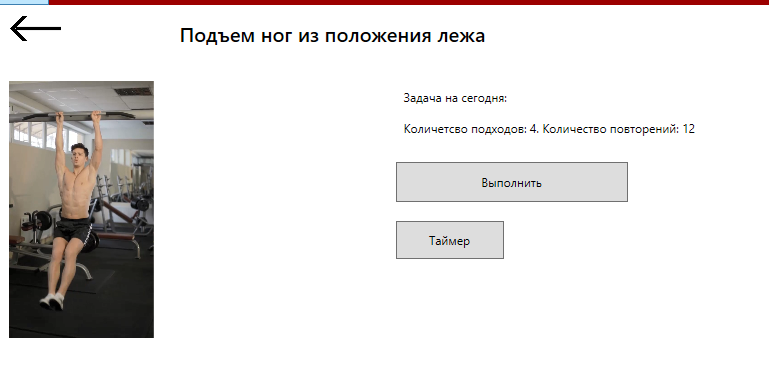


Рисунок 4.8 – Упражнение

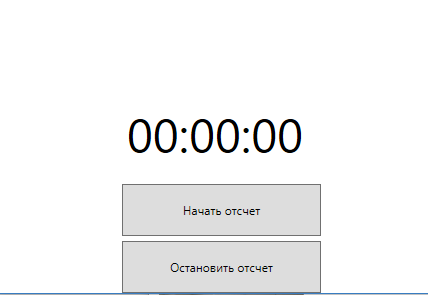


Рисунок 4.9 – Таймер

При переходе в меню питание у нас открывается две кнопки (Рисунок 4.10).

Каждая из этих кнопок содержит в себе различные блюда, разница лишь в том, что при нажатии на кнопку “Наращивания мышц”, буду выводиться более калорийные блюда, нежели при нажатии на кнопку “Снижение веса” (Рисунок 4.11).

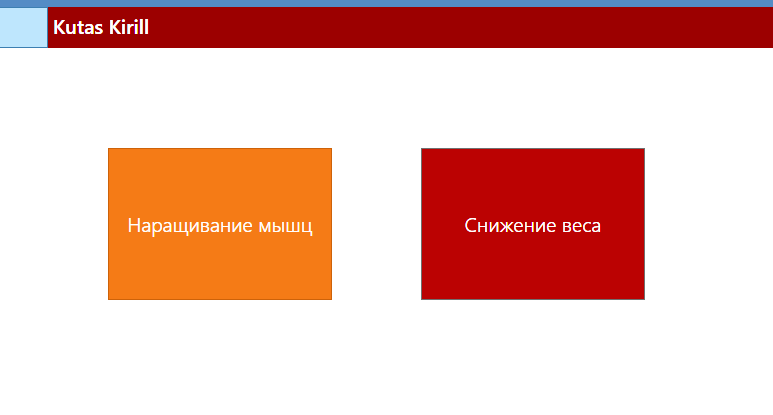


Рисунок 4.10 – Питание

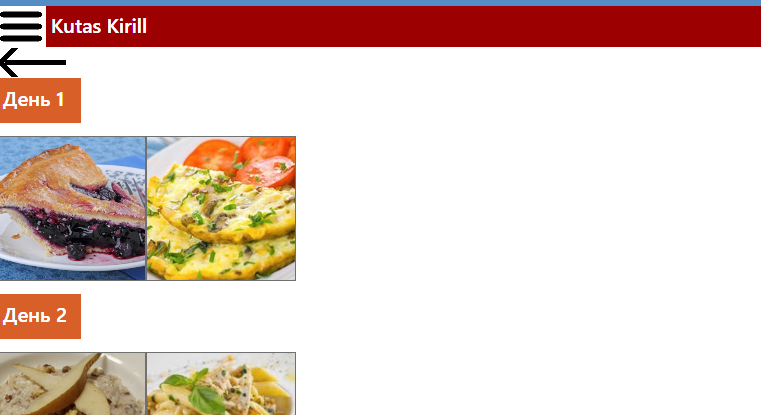


Рисунок 4.11 – Блюда

При нажатии на блюдо у нас открывается окно, в котором мы видим название нашего блюда, его фотографию, ингредиенты, рецепт и кнопку “Употребить”, при нажатии на которую у нас увеличивается количество ккал (Рисунок 4.12).



Рисунок 4.12 – Блюдо

В меню статистика у нас отображается статистика по нашему аккаунту (Рисунок 4.13), в котором мы можем получить статистику на почту либо очистить ее.

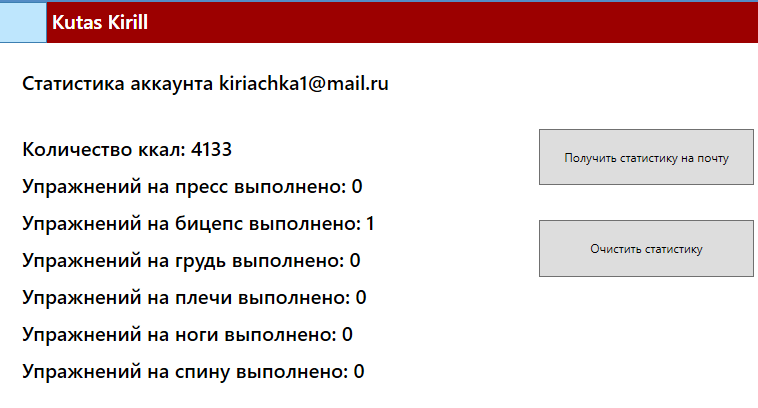


Рисунок 4.13 - Статистика

# Заключение

В курсовом проекте описана и реализована программа, написанная на языке C#, с использованием технологий ADO.NET, WPF.

В ходе выполнения курсового проекта было создано приложение-планировщик, которое выполняет следующие функции:

* выполнять авторизацию и регистрацию пользователя;
* сохранять информацию о пользователе в базе данных;
* возможность просматривать профиль пользователя;
* отправка статистики на почту;
* возможность пользователю просматривать упражнения или блюда.

Данное программное средство имеет удобный и понятный интерфейс, реагирует на ошибочный ввод данных (например, ввод данных не того типа, проверка на пустоту вводимых значений).

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

# Список используемых источников

1. MSDN сеть разработчиков в Microsoft [Электронный ресурс] – <http://msdn.microsoft.com/library/rus/>.
2. Язык C# и .NET Framework [Электронный ресурс] – <http://professorweb.ru/my/csharp/charp_theory/level1/infonet.php>.
3. Руководство по ADO.NET и работе с базами данных [Электронный ресурс] – <https://metanit.com/sharp/adonet/>. Дата доступа: 10.05.2017

4 Форум для программистов или разработчиков [Электронный ресурс] – https://stackoverflow.com/

# Приложение А

Листинг методов регистрации:

private bool CheckForCreating()//Проверка на наличие подобного юзера в бд

{

try

{

List<object> list = bd.Get("mail");

foreach (var a in list)

{

if ((string)a == EMail.Text)

{

Error.Content = "Пользователь с такой почтой уже существет";

EMail.BorderBrush = Brushes.Red;

throw new Exception();

}

}

return true;

}

catch

{

return false;

}

}

private Boolean Enter(String adress, String password, String POP3, Int32 port)

{

try

{

using (Pop3Client client = new Pop3Client())

{

client.Connect(POP3, port, true);

client.Authenticate(adress, password);

return true;

}

}

catch

{

return false;

}

}

private bool CheckMailAndPassword(string Mail, string Password)//Проверка почты

{

String POP3\_server = "";

Int32 POP3\_port = 0;

try

{

switch (Mail.Split('@')[1])

{

case "mail.ru":

{

POP3\_server = "pop.mail.ru";

POP3\_port = 995;

}

break;

case "yandex.ru":

{

POP3\_server = "pop.yandex.ru";

POP3\_port = 995;

}

break;

case "gmail.com":

{

POP3\_server = "pop.gmail.com";

POP3\_port = 995;

}

break;

}

}

catch

{

Error.Content = "Неправильная почта.";

EMail.BorderBrush = Brushes.Red;

return false;

}

if (!Enter(Mail, Password, POP3\_server, POP3\_port))

{

Error.Content = "Неправильный пароль.";

PasswordBox2.BorderBrush = Brushes.Red;

return false;

}

return true;

}

private void ButtonForRegistration\_Click(object sender, RoutedEventArgs e)

{

Error.Content = "";

PasswordBox2.BorderBrush = Brushes.Black;

EMail.BorderBrush = Brushes.Black;

if (CheckMailAndPassword(EMail.Text, PasswordBox2.Password))

{

if (CheckForCreating())//Добавление usera

{

try

{

if (String.Compare(PersonName.Text, "Введите ФИО") == 0)

new Exception("Введите ФИО");

string connectionString = @"Data Source=.;Initial Catalog=Fitness;Integrated Security=True";

string sqlAdd;

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

sqlAdd = String.Format("INSERT INTO Users (Mail, Password, Name) VALUES ('{0}', '{1}','{2}')", EMail.Text, PasswordBox2.Password, PersonName.Text);

SqlCommand command = new SqlCommand(sqlAdd, connection);

int number = command.ExecuteNonQuery();

connection.Close();

SqlConnection connection1 = new SqlConnection(connectionString);

connection1.Open();

sqlAdd = String.Format("INSERT INTO Stat (Mail,[Ккал],[Упражнений на пресс выполнено],[Упражнений на бицепс выполнено],[Упражнений на грудь выполнено],[Упражнений на плечи выполнено],[Упражнений на ноги выполнено],[Упражнений на спину выполнено]) VALUES ('{0}',0,0,0,0,0,0,0)", EMail.Text);

SqlCommand command1 = new SqlCommand(sqlAdd, connection1);

int number1 = command1.ExecuteNonQuery();

connection1.Close();

BufferClass.Mail = EMail.Text;

BufferClass.Password = PasswordBox2.Password;

BufferClass.Name = PersonName.Text;

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

Листинг методов авторизации:

private void SignIn\_Click(object sender, RoutedEventArgs e)

{

Err.Content = "";

PasswordBox1.BorderBrush = Brushes.Black;

Mail.BorderBrush = Brushes.Black;

if (String.Compare(Mail.Text, "Введите E-Mail" ) != 0)

{

if (PasswordBox1.Password != "")

{

List<object> bdMail = bd.Get("mail");

List<object> bdPassword = bd.Get("password");

for (int a = 0; a < bdMail.Count; a++)

{

if ((string)bdMail[a] == Mail.Text)

{

if ((string)bdPassword[a] == PasswordBox1.Password)

{

try

{

BufferClass.Mail = Mail.Text;

BufferClass.Password = PasswordBox1.Password;

List<object> person = bd.GetSomthPerson(BufferClass.Mail);

if (person.Count > 2)

BufferClass.Name = (string)person[2];

if (person.Count > 3)

BufferClass.Photo = (string)person[3];

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Close();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

{

Err.Content = "Неверный пароль.";

PasswordBox1.BorderBrush = Brushes.Red;

}

}

}

Err.Content = "Неверная почта.";

Mail.BorderBrush = Brushes.Red;

}

else

{

Err.Content = "Введите пароль.";

PasswordBox1.BorderBrush = Brushes.Red;

}

}

else

{

Err.Content = "Введите почту.";

Mail.BorderBrush = Brushes.Red;

}

}

}

# Приложение Б

Листинг методов класса для работы с базой данных:

class BD

{

private string connectionString { get; set; }

public BD()

{

connectionString = @"Data Source=.;Initial Catalog=Fitness;Integrated Security=True";

}

public bool Add(string Column, string Data, string Connect = "Users")

{

try

{

string sqlExpression = String.Format("INSERT INTO {0} ({1}) VALUES ('{2}')", Connect, Column, Data);

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

int number = command.ExecuteNonQuery();

connection.Close();

return true;

}

catch

{

return false;

}

}

public bool AddSomthPerson(string mail, string column, string data, string Connect = "Users")

{

string sqlExpression = String.Format("UPDATE {0} SET {1}='{2}' where Mail='{3}'", Connect, column, data, mail);

try

{

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

int number = command.ExecuteNonQuery();

connection.Close();

return true;

}

catch

{

return false;

}

}

public List<object> Get(string whatGet, string Connect = "Users")

{

string sqlExpression = "select\* from " + Connect;

List<object> result = new List<object>();

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

switch (whatGet)

{

case "mail":

{

while (reader.Read())

{

if (!(reader.GetValue(0) is System.DBNull))

result.Add(reader.GetValue(0));

}

break;

}

case "password":

{

while (reader.Read())

{

if (!(reader.GetValue(1) is System.DBNull))

result.Add(reader.GetValue(1));

}

break;

}

case "name":

{

while (reader.Read())

{

if (!(reader.GetValue(2) is System.DBNull))

result.Add(reader.GetValue(2));

}

break;

}

case "photo":

{

while (reader.Read())

{

if (!(reader.GetValue(3) is System.DBNull))

result.Add(reader.GetValue(3));

}

break;

}

case "Таблица":

{

while (reader.Read())

{

if (!(reader.GetValue(1) is System.DBNull))

result.Add(reader.GetValue(1));

}

break;

}

case "Группы мышц":

{

while (reader.Read())

{

if (!(reader.GetValue(0) is System.DBNull))

result.Add(reader.GetValue(0));

}

break;

}

case "Упражнение":

{

while (reader.Read())

{

if (!(reader.GetValue(0) is System.DBNull))

result.Add(reader.GetValue(0));

}

break;

}

case "Видео":

{

while (reader.Read())

{

if (!(reader.GetValue(1) is System.DBNull))

result.Add(reader.GetValue(1));

}

break;

}

}

connection.Close();

return result;

}

public List<object> GetSomthPerson(string Mailperson)

{

List<object> result = new List<object>();

string sqlExpression = String.Format("select\* from Users where Mail = '{0}'", Mailperson);

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

for (int a = 0; a < 4; a++)

{

if (!(reader.GetValue(a) is System.DBNull))

result.Add(reader.GetValue(a));

}

}

connection.Close();

return result;

}

public List<object> GetSomthPersonStat(string Mailperson)

{

List<object> result = new List<object>();

string sqlExpression = String.Format("select\* from Stat where Mail = '{0}'", Mailperson);

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

for (int a = 0; a < 8; a++)

{

if (!(reader.GetValue(a) is System.DBNull))

result.Add(reader.GetValue(a));

}

}

connection.Close();

return result;

}

public bool UpdateStat(string mail, string column, string data, string Connect = "Stat")

{

string sqlExpression = String.Format("UPDATE {0} SET {1}='{2}' where Mail='{3}'", Connect, column, data, mail);

try

{

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

int number = command.ExecuteNonQuery();

connection.Close();

return true;

}

catch

{

return false;

}

}

public List<object> GetSomthLess(string NameLess, string bd)

{

List<object> result = new List<object>();

string sqlExpression = String.Format("select\* from {0} where [Упражнение] = '{1}'", bd, NameLess);

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

for (int a = 0; a < 3; a++)

{

if (!(reader.GetValue(a) is System.DBNull))

result.Add(reader.GetValue(a));

}

}

connection.Close();

return result;

}

public List<object> GetSomthDish(string NameDish, string bd)

{

List<object> result = new List<object>();

string sqlExpression = String.Format("select\* from {0} where [Блюдо] = '{1}'", bd, NameDish);

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

for (int a = 0; a < 5; a++)

{

if (!(reader.GetValue(a) is System.DBNull))

result.Add(reader.GetValue(a));

}

}

connection.Close();

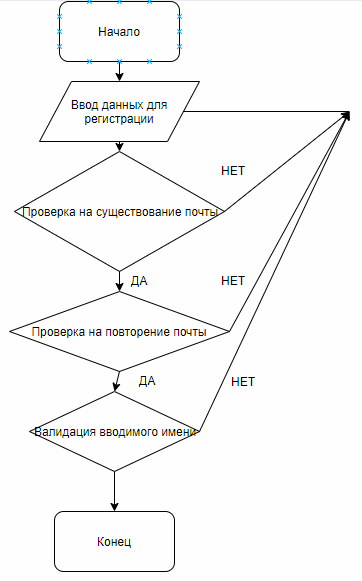
return result;

}

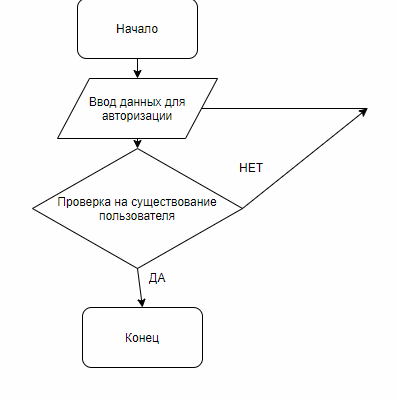
}

# Приложение В

Блок-схема алгоритма регистрации:



Блок-схема алгоритма авторизации:



# Приложение Г

Код реализации вывода блюд:

private void eatt\_Click(object sender, RoutedEventArgs e)

{

PersonAccount.Visibility = Visibility.Hidden;

Lessons.Visibility = Visibility.Hidden;

Eat.Visibility = Visibility.Visible;

Stat.Visibility = Visibility.Hidden;

WhatNeed.Visibility = Visibility.Visible;

EatWhatWant.Visibility = Visibility.Hidden;

Dish.Visibility = Visibility.Hidden;

MoveMenu(0, -105);

}

void eatMenu(List<object> list, StackPanel stack, List<object> photo)

{

stack.Children.Clear();

for (int a = 0; a < list.Count; a++)

{

Button button = new Button();

ImageBrush imBr = new ImageBrush();

imBr.ImageSource = new System.Windows.Media.Imaging.BitmapImage(new Uri((string)photo[a], UriKind.Relative));

imBr.Stretch = Stretch.UniformToFill;

button.Name = "Button" + Convert.ToString(a);

button.Background = imBr;

button.Width = 150;

button.Height = 145;

button.Foreground = Brushes.Red;

button.FontFamily = new FontFamily("Segoe UI Semibold");

button.FontSize = 18;

button.Click += EatButton\_Click;

stack.Children.Add(button);

}

}

string fullName = "";

private void EatButton\_Click(object sender, RoutedEventArgs e)

{

EatWhatWant.Visibility = Visibility.Hidden;

Dish.Visibility = Visibility.Visible;

Button button = (Button)sender;

char[] number = button.Name.ToCharArray();

int indx = (int)Char.GetNumericValue(number[number.Length - 1]);

StackPanel stack = (StackPanel)button.Parent;

fullName = nameBd + "\_" + stack.Name;

List<object> list = bd.Get("Упражнение", nameBd + "\_" + stack.Name);

List<object> listPhoto = bd.Get("Видео", nameBd + "\_" + stack.Name);

NameEat.Content = (string)list[indx];

PhotoEat.Stretch = Stretch.UniformToFill;

PhotoEat.Source = new BitmapImage(new Uri((string)listPhoto[indx]));

List<object> IngridientsList = bd.Get("name", nameBd + "\_" + stack.Name);

List<object> ReceptList = bd.Get("photo", nameBd + "\_" + stack.Name);

string text = System.IO.File.ReadAllText((string)IngridientsList[indx], Encoding.Default).Replace("\n", " ");

ingridients.Text = text;

text = System.IO.File.ReadAllText((string)ReceptList[indx], Encoding.Default).Replace("\n", " ");

Recepts.Text = text;

}

string nameBd = "";

private void Muscule\_Click(object sender, RoutedEventArgs e)// вывод блюд

{

Button button = (Button)sender;

WhatNeed.Visibility = Visibility.Hidden;

EatWhatWant.Visibility = Visibility.Visible;

nameBd = button.Name;

List<StackPanel> stackPanels = new List<StackPanel>();

stackPanels.Add(day1);

stackPanels.Add(day2);

stackPanels.Add(day3);

stackPanels.Add(day4);

stackPanels.Add(day5);

stackPanels.Add(day6);

stackPanels.Add(day7);

for (int a = 0; a < stackPanels.Count; a++)

{

List<object> ListDay1 = bd.Get("Упражнение", nameBd + "\_day" + (a + 1));

List<object> listPhoto = bd.Get("Видео", nameBd + "\_day" + (a + 1));

eatMenu(ListDay1, stackPanels[a], listPhoto);

}

}

# Приложение Г

Код реализации вывода упражнений:

public void AddButton(List<object> list, int q)

{

wrapPanel.Children.Clear();

for (int a = 0; a <= list.Count; a++)

{

if (q == 1 && a == 0)

a++;

Button button = new Button();

button.Name = "Button" + Convert.ToString(a);

if (a == 0 && q != 1)

button.Content = "Назад";

else

button.Content = (string)list[a - 1];

button.Width = 800;

button.Height = 100;

button.Background = Brushes.White;

button.Foreground = Brushes.Red;

button.FontFamily = new FontFamily("Segoe UI Semibold");

button.FontSize = 18;

if (q == 1 && a != 0)

button.Click += Button\_Click;

else if (q == 2 && a != 0)

button.Click += Button1\_Click;

else if (q != 1 && a == 0)

button.Click += ButtonBack\_Click;

wrapPanel.Children.Add(button);

}

}

private void ButtonBack\_Click(object sender, RoutedEventArgs e)

{

List<object> list = bd.Get("Группы мышц", "Trainer");

AddButton(list, 1);

}

string ourBd = "";

string source = "";

private void Button1\_Click(object sender, RoutedEventArgs e)

{

ok.Visibility = Visibility.Visible;

Great.Visibility = Visibility.Hidden;

Button button = (Button)sender;

List<object> PersonStat = bd.GetSomthPersonStat(BufferClass.Mail);

List<object> ThisLess1 = bd.GetSomthLess((string)button.Content, ourBd);

bd.UpdateStat(BufferClass.Mail, "Ккал", Convert.ToString(Convert.ToInt32(PersonStat[1]) - Convert.ToInt32(ThisLess1[2])));

int percent = 0;

switch (ourBd)

{

case "Press":

{

percent = (int)PersonStat[2];

break;

}

case "Biceps":

{

percent = (int)PersonStat[3];

break;

}

case "ChestMuscles":

{

percent = (int)PersonStat[4];

break;

}

case "Shoulder":

{

percent = (int)PersonStat[5];

break;

}

case "Legs":

{

percent = (int)PersonStat[6];

break;

}

case "Rear":

{

percent = (int)PersonStat[7];

break;

}

}

int numberOfPovt = 12 + (12 \* 5 \* percent / 100);

int numberOfPodx = 4;

LabelForLess.Content = "Количетсво подходов: " + Convert.ToString(numberOfPodx) + ". Количество повторений: " + Convert.ToString(numberOfPovt);

List1.Visibility = Visibility.Hidden;

ThisLess.Visibility = Visibility.Visible;

NameLesson.Content = button.Content;

List<object> list = bd.Get("Упражнение", ourBd);

List<object> list1 = bd.Get("Видео", ourBd);

for (int a = 0; a < list.Count; a++)

{

if ((string)button.Content == (string)list[a])

{

source = (string)list1[a];

break;

}

}

MyVideo.Source = new Uri(source);

MyVideo.Play();

}

private void MyVideo\_MediaEnded\_1(object sender, RoutedEventArgs e)

{

MyVideo.Source = new Uri(source);

MyVideo.Play();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

List<object> list = bd.Get("Группы мышц", "Trainer");

List<object> list1 = bd.Get("Таблица", "Trainer");

for(int a = 0; a < list.Count; a++)

{

if ((string)button.Content == (string)list[a])

{

ourBd = (string)list1[a];

break;

}

}

List<object> list2 = bd.Get("Упражнение", ourBd);

AddButton(list2, 2);

}

private void less\_Click(object sender, RoutedEventArgs e)

{

PersonAccount.Visibility = Visibility.Hidden;

Lessons.Visibility = Visibility.Visible;

List1.Visibility = Visibility.Visible;

ThisLess.Visibility = Visibility.Hidden;

Eat.Visibility = Visibility.Hidden;

Stat.Visibility = Visibility.Hidden;

MoveMenu(0, -105);

List<object> list = bd.Get("Группы мышц", "Trainer");

AddButton(list, 1);

}