|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |
| --- |
| **КУРСОВОЙ ПРОЕКТ** |
|  |
| по междисциплинарному курсу МДК.02.02. Технология разработки и защиты баз данных |
|  |
| на тему: Разработка электронного телефонного справочника. |
|  |
| студента группы ПКС-304  специальности 09.02.03 Программирование в компьютерных системах |
| Евдокимова Юрия Олеговича |

|  |  |  |
| --- | --- | --- |
| Студент |  | Ю.О. Евдокимов |
| Руководитель курсового проекта |  | Е.А. Ларионова |
| Председатель ПЦК специальности 09.02.03 Программирование в компьютерных системах |  | А.И. Глускер |

|  |  |  |
| --- | --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г. | | |
| Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Заведующий отделением № 3 |  | Е.Л. Гусева |

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc40619459)

[ВВЕДЕНИЕ 5](#_Toc40619460)

[1 ОСНОВНАЯ ЧАСТЬ 7](#_Toc40619461)

[1.1 Назначение и цели создания системы 7](#_Toc40619462)

[1.2 Обзор и анализ предметной области 7](#_Toc40619463)

[1.3 Жизненный цикл базы данных 8](#_Toc40619464)

[1.4 Выбор и характеристика СУБД 9](#_Toc40619465)

[1.5 Выбор и характеристика среды разработки 9](#_Toc40619466)

[1.6 Выбор и характеристика языка программирования 10](#_Toc40619467)

[2 СПЕЦИАЛЬНАЯ ЧАСТЬ 11](#_Toc40619468)

[2.1 Постановка задачи 11](#_Toc40619469)

[2.2 Архитектура информационной системы 11](#_Toc40619470)

[2.2.1 Основные этапы проектирования приложения 11](#_Toc40619471)

2.2.2 [Архитектура клиент-сервер 12](#_Toc40619472)

[2.2.3 Навигационная схема приложения 13](#_Toc40619473)

[2.2.4 Архитектура классов приложения 13](#_Toc40619474)

[2.3 Логическая модель базы данных 15](#_Toc40619475)

[2.4 Нормализация таблиц 15](#_Toc40619476)

[2.5 Описание таблиц 16](#_Toc40619477)

[2.6 Разработка приложения 20](#_Toc40619478)

[2.6.1 Диаграмма вариантов использования 20](#_Toc40619479)

[2.6.2 Решение главной задачи проекта 20](#_Toc40619480)

[2.6.4 Защита информационной системы 23](#_Toc40619481)

[2.6.5 Скриншоты интерфейса 24](#_Toc40619482)

[2.7 Инструкция пользователю 30](#_Toc40619483)

[2.7.1 Общие сведения об информационной системе 30](#_Toc40619484)

[2.7.2 Требования к техническим средствам 30](#_Toc40619485)

[2.7.3 Требования к программным средствам 30](#_Toc40619486)

[2.7.4 Настройка информационной системы 30](#_Toc40619487)

[2.7.5 Формы ввода 30](#_Toc40619488)

[ЗАКЛЮЧЕНИЕ 32](#_Toc40619489)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 33](#_Toc40619490)

[ПРИЛОЖЕНИЕ А SQL скрипты на создание и заполнение базы данных, на создание учетных записей пользователей, на назначение привилегий 34](#_Toc40619491)

[ПРИЛОЖЕНИЕ Б Код приложения 48](#_Toc40619492)

# ВВЕДЕНИЕ

Темой проекта является электронный телефонный справочник, у каждого человека в процессе жизнедеятельности появляется множество связей и знакомств, которые необходимо поддерживать и быть всегда на связи с нужными людьми. В связи с этим было разработано приложение, позволяющее добавлять и хранить номера телефонов различных людей, а также их email-почту, что позволяет в любой момент выйти на связь с нужным человеком просто посмотрев его номер телефона или почту.

Основные задачи, которые необходимо выполнить для того, чтобы разработать данное приложение:

* подробное изучение функциональных возможностей WPF,
* освоение тонкостей проектирования баз данных,
* изучение ADO.NET,
* разработка функций для работы с базой данных,
* создание функций для работы с данными внутри приложения,
* проектирование интерфейса программы,
* написание функций для корректного сохранения и обновления данных,
* создание функций для корректного отображения содержимого страниц,
* проектирование архитектуры и логики приложения.

Используемые в разработке методы:

* объектно-ориентированное программирование,
* моделирование.

Обоснование структуры курсового проекта:

1. введение: в данном разделе описываются цели и задачи проекта, а также его актуальность, так как изначально необходимо определить, что предстоит изучать и делать в проекте,
2. основная часть: этот раздел включает в себя анализ предметной области, для которой разрабатывается данное приложение, назначения и цели разработки, а также жизненный цикл базы данных,
3. специальная часть: в данном разделе описаны архитектурные решения, которые применяются в разработке, а также процесс разработки программного обеспечения и базы данных,
4. заключение: вывод о проделанной работе и полученных навыках за время разработки приложения,
5. список используемых источников: этот раздел содержит в себе перечень ресурсов, которые были использованы для получения необходимой информации во время разработки.

# 1 ОСНОВНАЯ ЧАСТЬ

* 1. Назначение и цели создания системы

Приложение разрабатывается для хранения, добавления и редактирования записей о различных людях, хранящих их номера телефона и почту.

Цель: создать приложение средствами позволяющее ввести данные о людях, их номере телефона и почте, чтобы более удобно поддерживать связь с этими людьми.

Назначение системы: обеспечить простое управления записями о различных людях, а также возможность редактировать эти данные, удалять и добавлять новые.

Базовые сущности этой предметной области: пользователи, записи о людях, группы, к которым эти записи относятся.

В системе должны существовать один вид взаимодействия: аутентифицированный пользователь.

* 1. Обзор и анализ предметной области

Предметной областью разработки является реальная жизнь. Реальная жизнь предоставляет нам возможность иметь множество связей, но поддерживать их - сложная задача. Поэтому было принято решение разработать электронный телефонный справочник. В электронном телефонном справочнике должны хранится способы связи с различными людьми, например номер телефона и адрес электронной почты.

Список основных сущностей:

* пользователи – это люди, которые пользуются приложением и имеют доступ к своим записям. Атрибуты пользователей: Уникальный номер пользователя, имя пользователя, пароль пользователя,
* записи о людях – записи, хранящие имя человека и данные о нём. Атрибуты дисциплин: уникальный номер записи о человеке, имя человека, номер, дополнительный номер, email, номер группы,
* группы – организованные группы людей. Атрибуты группы: уникальный номер группы, название группы,
  1. Жизненный цикл базы данных

Жизненный цикл – это процесс проектирования, реализации и управления базой данных. Он состоит из:

1. Анализа предметной области, который включает в себя планирование разработки (15.02.2020 – 25.02.2020), определение требований (15.02.2020 – 25.02.2020), сбор и анализ требований (15.02.2020 – 25.02.2020),
2. Проектирования базы данных, которое состоит из создания концептуальной модели (26.02.2020), логической модели (26.02.2020), физической модели (27.02.2020),
3. Реализации, в процессе которой пишутся скрипты на создание базы данных, связей сущностей в базе данных, а также разрабатывается интерфейс приложения и его функции (01.03.2020 – 05.05.2020).
   1. Выбор и характеристика СУБД

Для разработки базы данных была выбрана среда Microsoft SQL Server Management Studio 2018, так как она является удобной для использования и не требует долгого освоения тонкостей для создания полноценной базы данных под программу, а также с ней можно работать, используя технологию ADO.NET. Эта СУБД использует язык SQL (T-SQL) для написания необходимых сущностей и заполнения их данными.

При сравнении SQL Server с MySQL можно выделить несколько главных различий:

* первый предназначен для работы с платформой .Net, а второй для работы с PHP,
* синтаксис,
* данные SQL Server хранятся в единой системе Microsoft, в то время как данные MySQL хранятся на движках.

Весомыми плюсами SQL Server являются простота понимания принципов работы, синтаксиса, а также возможность создавать объекты баз данных либо с помощью скрипта, либо вручную, с помощью конструктора.

* 1. Выбор и характеристика среды разработки

Для разработки приложения была выбрана среда программирования Microsoft Visual Studio 2019, потому что эта среда разработки обеспечивает возможность создания приложений с удобным интерфейсом с помощью языка C# и системы построения клиентских приложений WPF.Net Framework.

Microsoft Visual Studio 2019 – продукт компании Microsoft, включающий в себя интегрированную среду разработки программного обеспечения, а также ряд других инструментальных средств.

Некоторые преимущества:

* удобная отладка и поиск ошибок,
* возможность создания специальной библиотеки для тестирования,
* подробная документация об ошибках, располагающаяся на сайте компании Microsoft,
* интеллектуальность системы, предупреждающая об ошибках и помогающая упростить и улучшить код.

Данную среду разработки можно сравнить с Visual Studio Code. Она является менее удобной, так как не представляет такого обширного количества средств для отладки и тестирования.

* 1. Выбор и характеристика языка программирования

Для написания кода приложения был выбран язык C#, так как он является предпочтительным для меня, а также с целью повышения навыков его использования в различных целях.

C# – объектно-ориентированный язык программирования от компании Microsoft, предназначенный для разработки приложений для платформы Microsoft.Net.

Некоторые преимущества:

* поддерживается ООП,
* удобный синтаксис,
* присутствие библиотеки LINQ для более удобной работы с коллекциями и базами данных.

В сравнении с языком Visual Basic, C# является более удобным и лаконичным, так как был изначально спроектирован под платформу .Net.

# 2 СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1 Постановка задачи

Главная задача проекта – спроектировать средствами MS SQL SERVER базу данных и создать приложение средствами Microsoft Visual Studio 2019 и Microsoft SQL Server Management Studio 18, позволяющее ввести данные о турнирах и матчах для удобства фанатов и сбора статистики.

Предусмотреть:

* поиск записей по заданному имени
* поиск записей по заданной части имени
* возможность добавления новых записей
* просмотр списка записей
* возможность редактирования записей
* возможность удаления записей

В системе должны существовать множество видов пользователей: все зарегистрированные пользователи.

Зарегистрированный пользователь может выполнять следующие действия:

* редактировать, добавлять и удалять свои записи,
* просматривать все свои записи по группам
* просматривать все свои записи.

2.2 Архитектура информационной системы

2.2.1 Основные этапы проектирования приложения

Первым пунктом разработки являлось создание первичной версии базы данных для последующей доработки.

После успешного завершения проектирования базы данных нужно было приступать к разработке окна и страниц приложения.

Первой была создана страница приветствия с возможностью регистрации или авторизации.

После этого было разработано меню программы, позволяющее пользователю управлять данными, вносить изменения в них, просматривать список номеров.

На последнем этапе было реализовано разделение на группы, а также доработан поиск как по целому имени, так и по части. Были улучшены все уже имеющиеся функции и добавлена возможность видеть только те записи, которые доступны авторизованному пользователю.

Архитектура клиент-сервер

Клиент-сервер – архитектура, в которой задачи или сетевая нагрузка распределены между поставщиками услуг, которые называются серверами, и заказчиками услуг – клиентами. Серверы ожидают запрос от клиента и передают

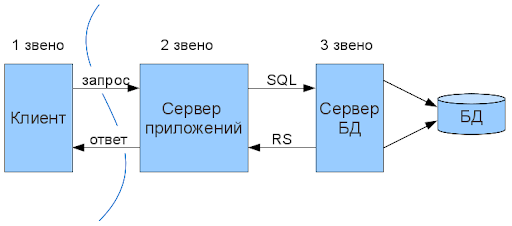
им свои ресурсы в виде данных. Модель данной архитектуры легко представить графически (Рисунок 1).

Рисунок 1 – модель архитектуры клиент-сервер

2.2.3 Навигационная схема приложения

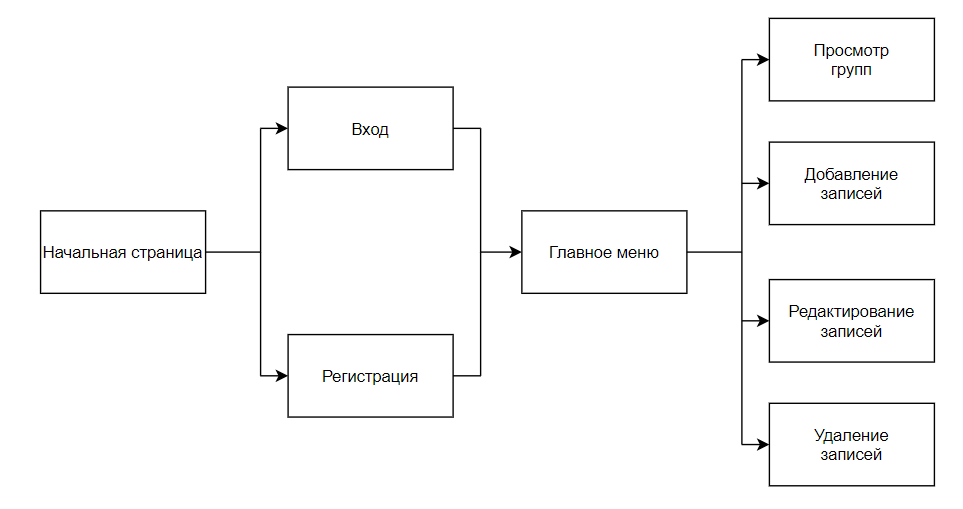


Рисунок 2 – навигационная схема приложения

2.2.4 Архитектура классов приложения

Для создания приложения я использовал технологию ADO.NET, а интерфейс реализован с использованием свойства UIElement.Visibility.

Приложение содержит следующие страницы:

* MainWindow.xaml – отвечающий за внешний вид и разметку программы.

Также приложение содержит такие классы как:

* MainWindow.xaml.cs – отвечающий за основную часть программы.

2.3 Логическая модель базы данных

Логическая модель базы данных, используемой в приложении (Рисунок 3).

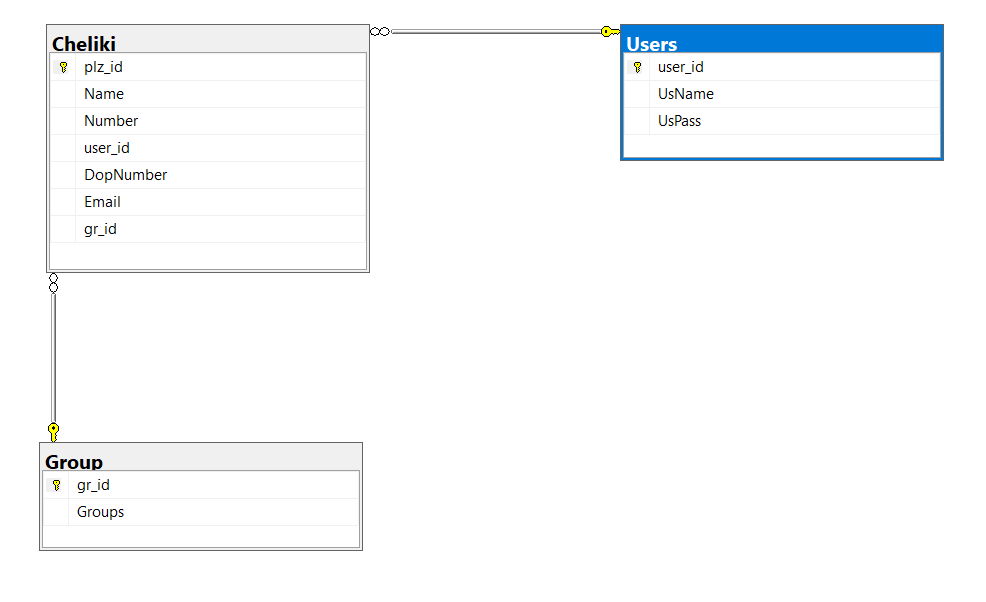


Рисунок 3 – логическая модель базы данных

2.4 Нормализация таблиц

Первая нормальная форма – все атрибуты атомарны. Это можно продемонстрировать на примере таблицы Users, атрибутами которой являются:

* User\_id,
* UsName,
* UsPass.

Соответствие первой нормальной форме можно увидеть и в других таблицах спроектированной базы данных.

Вторая нормальная форма – выполнены условия первой нормальной формы, а также каждый неключевой атрибут неприводимо зависит от первичного ключа. Неприводимость – отсутствие атрибутов, от которых можно вывести функциональную зависимость.

Пример сущности во второй нормальной форме (Рисунок 4).

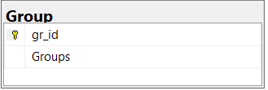


Рисунок 4 – пример второй нормальной формы

Третья нормальная форма – соблюдены условия второй нормальной формы, а также каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

База данных приведена к третьей нормальной форме.

2.5 Описание таблиц

В этом разделе описаны названия таблиц, их атрибуты, свойства и типы атрибутов, первичные и внешние ключи.

Определения:

* FK – внешний ключ,
* PK – первичный ключ,
* NotNULL – не разрешены нулевые значения,
* NULL – разрешены нулевые значения.

Таблица 1 – Users

|  |  |  |
| --- | --- | --- |
| Название | Тип | Дополнительно |
| User\_id | int | PK |
| UsName | Varchar(50) | NotNULL |
| UsPass | Varchar(50) | NotNULL |

Таблица 2 – Match\_Types

|  |  |  |
| --- | --- | --- |
| Название | Тип | Дополнительно |
| Gr\_id | int | PK |
| Groups | Varchar(50) | NotNULL |

Таблица 3 – Cheliki

|  |  |  |
| --- | --- | --- |
| Название | Тип | Дополнительно |
| Plz\_id | int | PK |
| Name | Varchar(50) | NotNULL |
| Number | Varchar(50) | NotNULL |
| DopNumber | Varchar(50) | NULL |
| Email | Varchar(50) | NULL |
| User\_id | int | FK |
| Gr\_id | int | FK |

2.6 Разработка приложения

2.6.1 Диаграмма вариантов использования

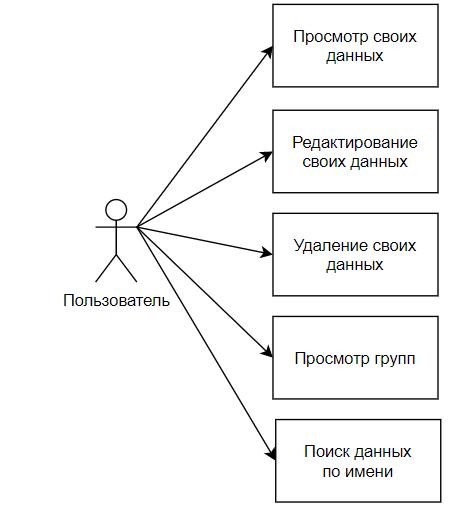


Рисунок 5 – диаграмма вариантов использования

2.6.2 Решение главной задачи проекта

Главная задача проекта – управление телефонными и почтовыми данными различных людей.

Необходимо:

* определить роль текущего пользователя,
* в соответствии с этой ролью отобразить его записи,
* возможность регистрации новых пользователей,
* возможность добавления, редактирования и удаления записей,
* возможность просмотра групп.

Обеспечение возможности добавления, редактирования и удаления записей (Рисунок 6).

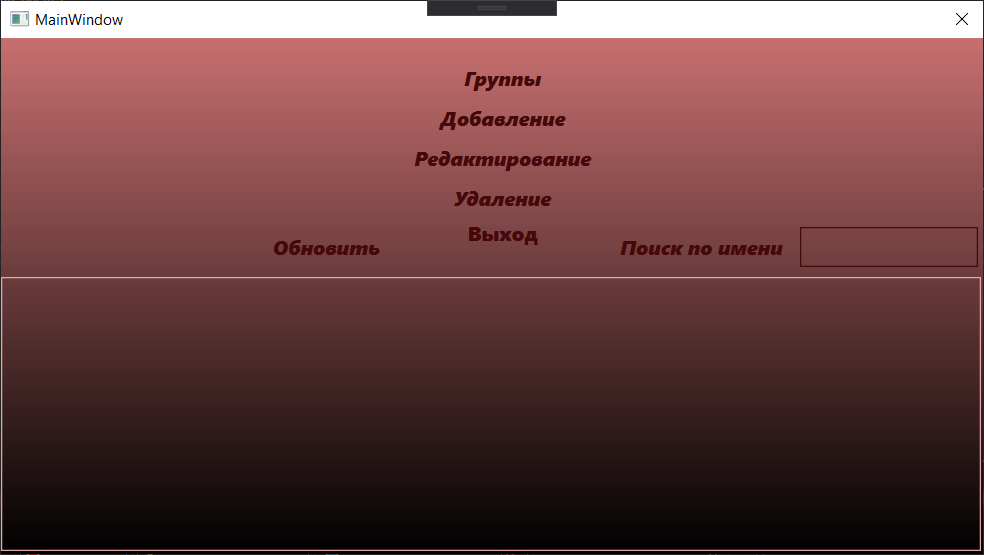


Рисунок 6 – Возможность добавления, удаления и редактирования записей.

2.6.4 Защита информационной системы

В разрабатываемом приложении предусмотрена защита всех форм ввода от неправильной и некорректной информации. При попытке сохранения неверных данных пользователю будет выведено сообщение об ошибке. При вводе корректных данных все будет успешно сохранено.

Защищены страницы регистрации, авторизации, изменения детальной информации, добавления записей.

2.6.5 Скриншоты интерфейса

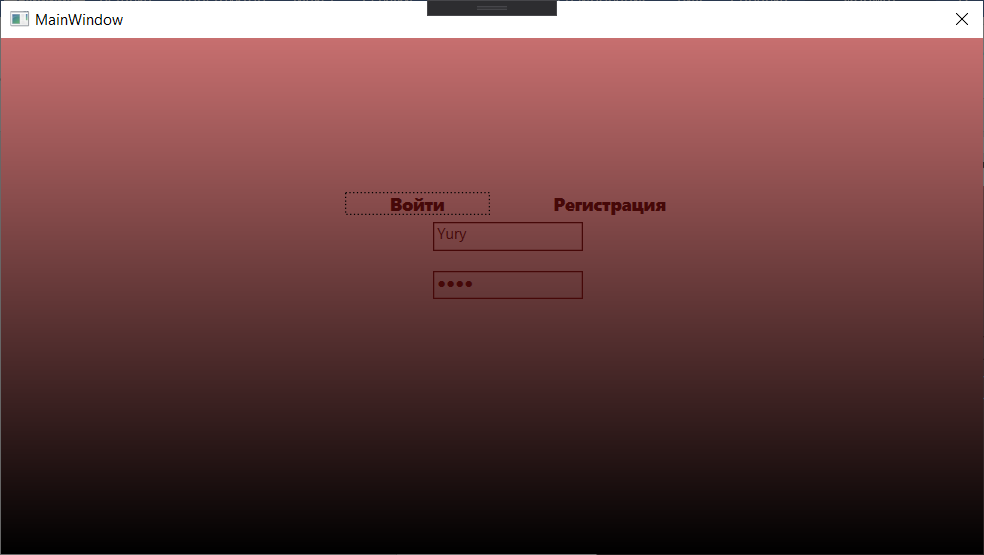


Рисунок 7 – страница аутентификации

В случае отсутствия пользователя в базе данных ему сообщиться об этом (Рисунок 8)

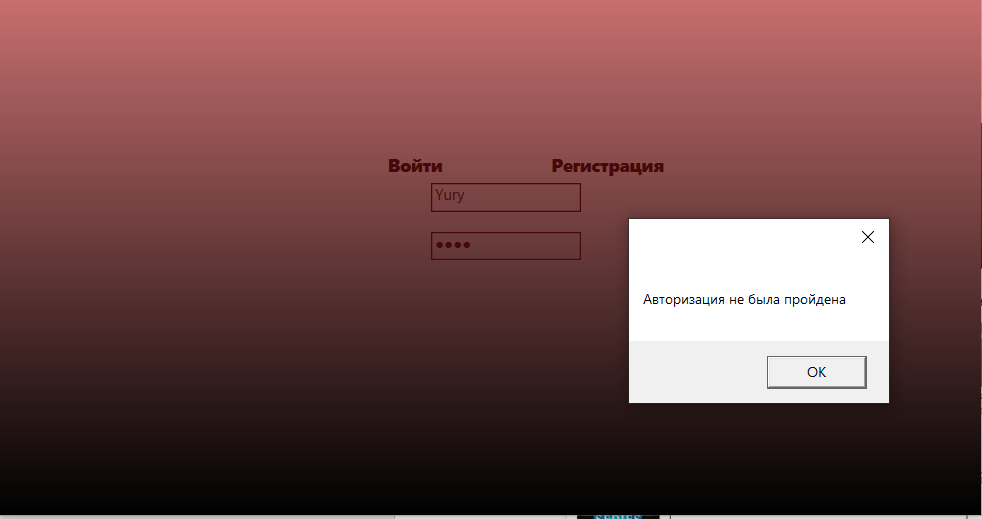


Рисунок 8 – авторизация не пройдена

В таком случае пользователь может зарегистрироваться и сразу же войти в программу (Рисунок 9, 10).

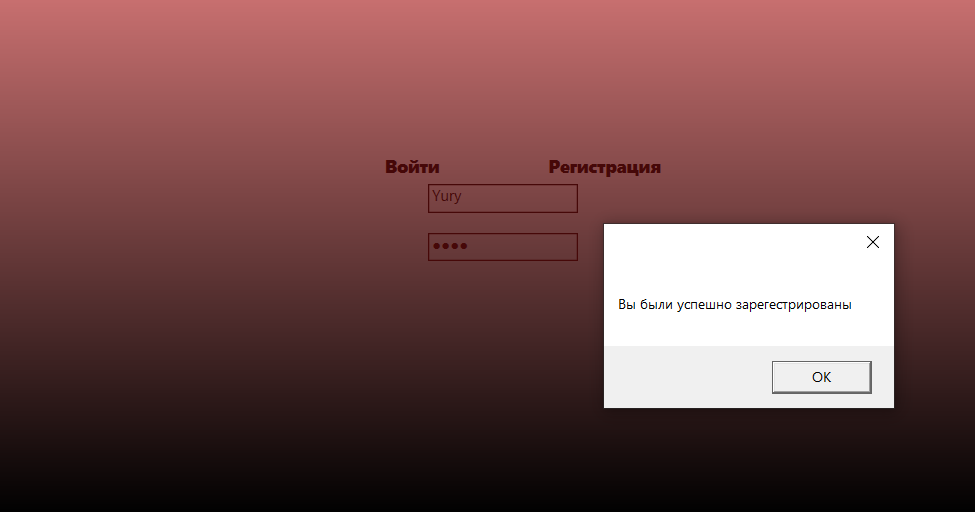


Рисунок 9 – регистрация пройдена

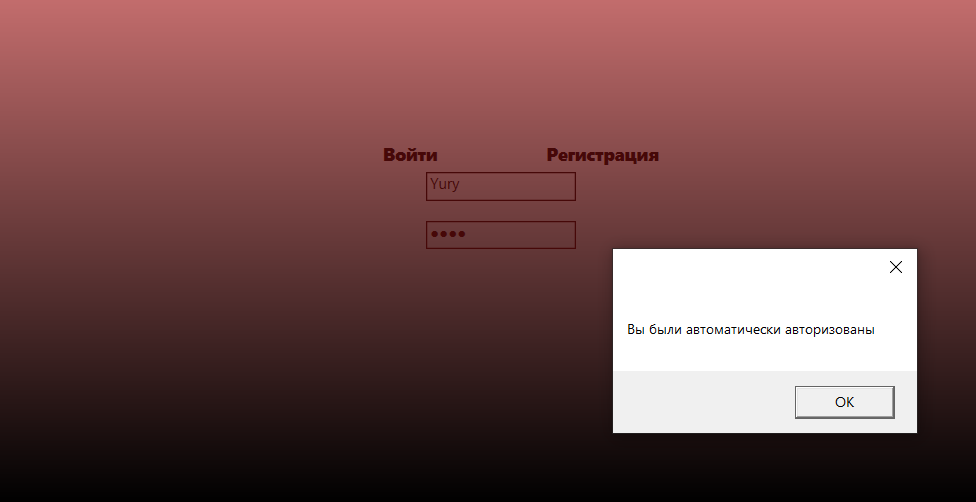


Рисунок 10 – автоматический вход в приложение

В случае, если пользователь с таким именем уже существует, регистрация не будет пройдена (Рисунок 11).

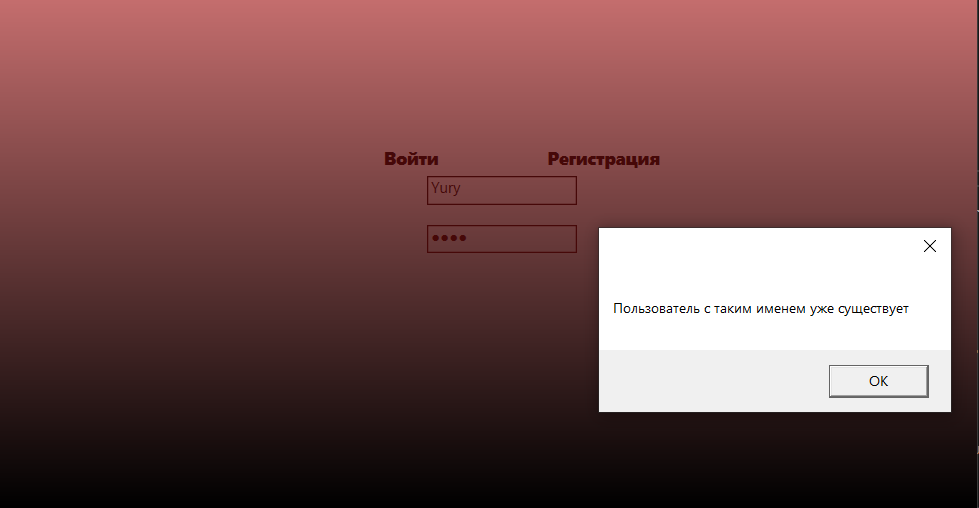


Рисунок 11 – проверка имени пользователя

После успешного входа в систему, пользователь попадает в главное меню (Рисунок 12).

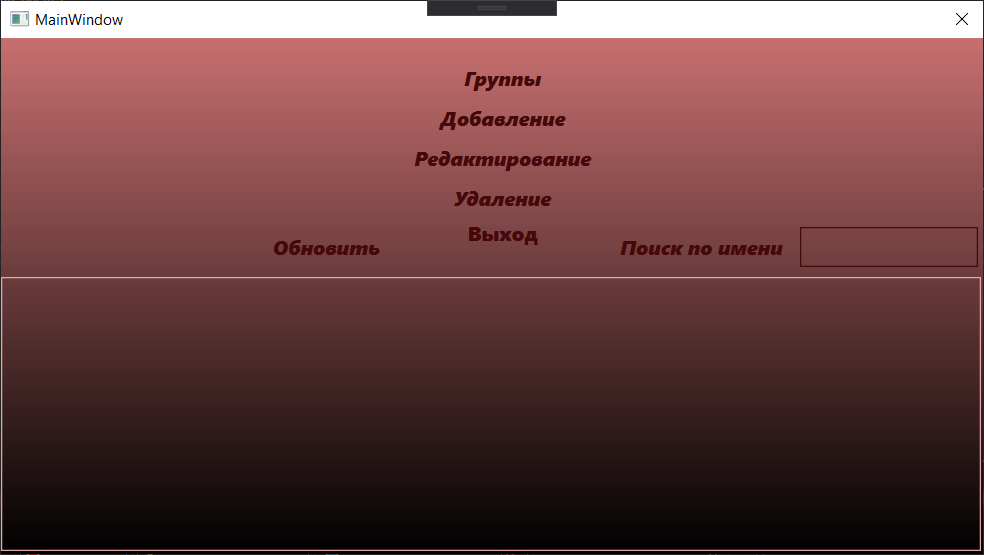


Рисунок 12 – главное меню

Поскольку пользователь новый – записи отсутствуют. Новые записи можно добавить, нажав на кнопку «Добавление». Сразу после этого появится интерфейс добавления, где легко можно добавить новую запись в основную таблицу, а также в группу при желании (Рисунок 13).

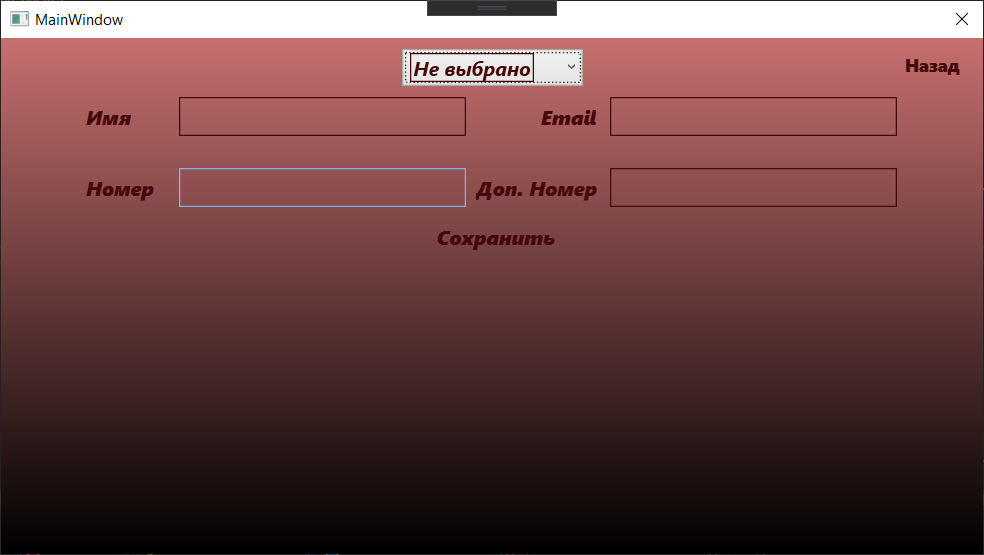


Рисунок 13 – страница добавления

У пользователя всегда есть возможность отредактировать какую-либо запись. Для этого нужно нажать на кнопку «Редактирование» и перейти к интерфейсу редактирования (Рисунок 14).

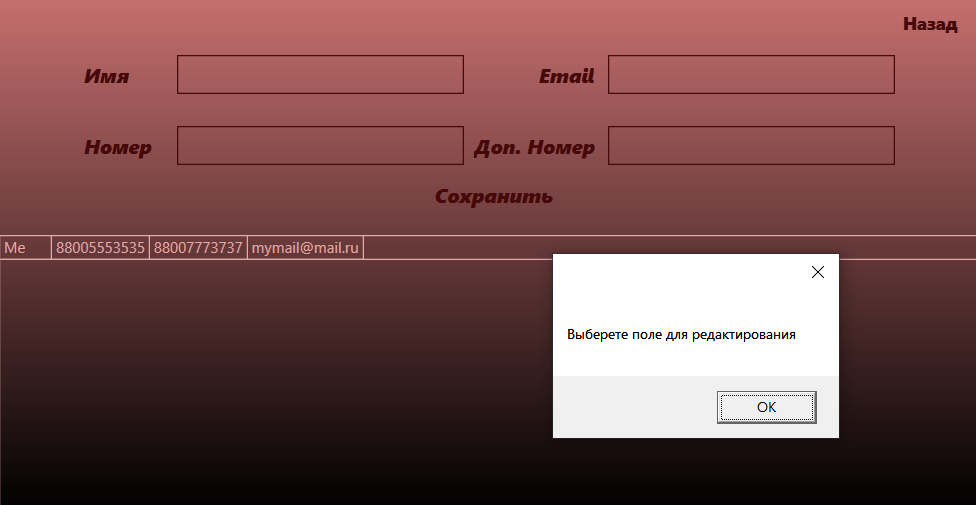


Рисунок 14 – страница редактирования

Также была предусмотрена возможность удаления записей (Рисунок 15).

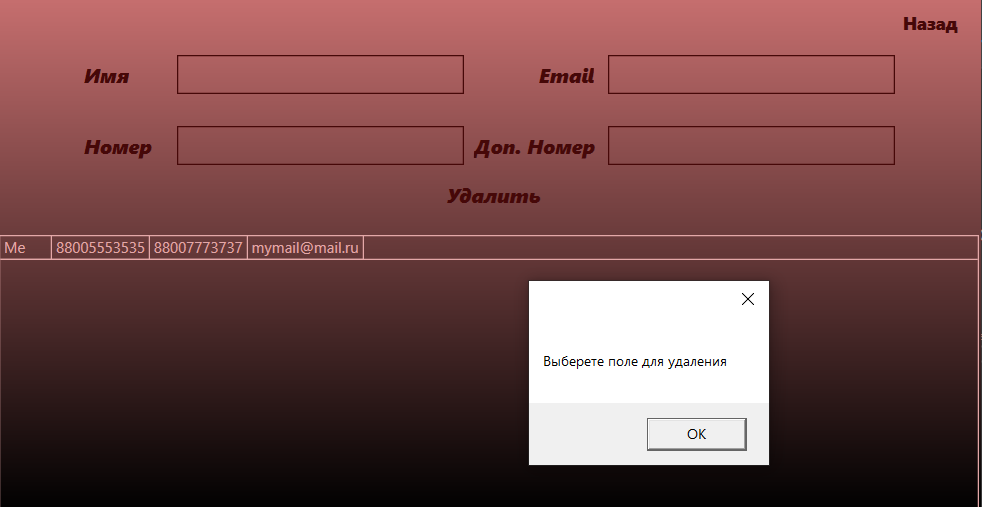


Рисунок 15 – страница удаления

В главном меню реализован поиск по имени (Рисунок 16).

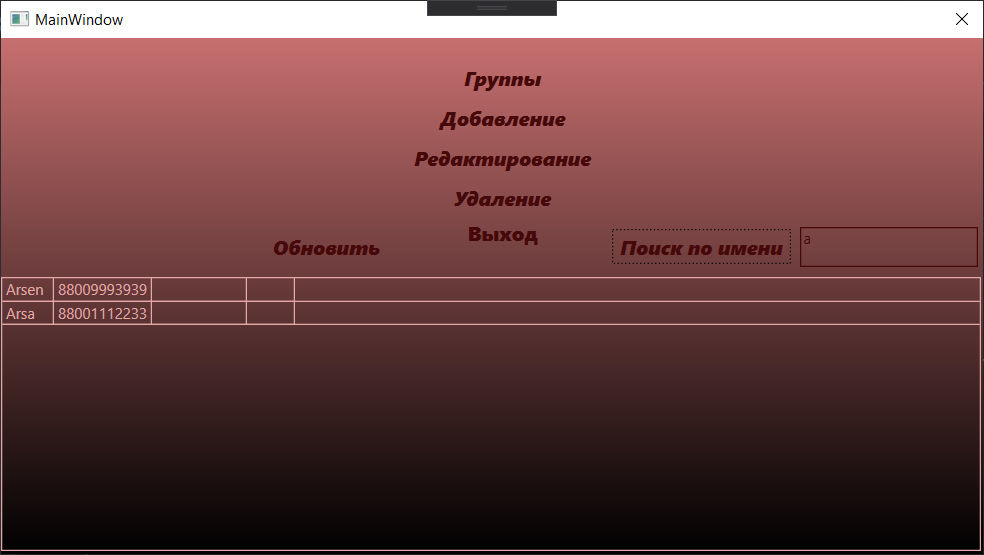


Рисунок 16 – поиск по имени

При нажатии кнопки «Группы» пользователь увидит номера, которые были добавлены в определённые группы (Рисунок 17).



Рисунок 17 – страница групп

2.7 Инструкция пользователю

Первым делом необходимо пройти авторизацию, если пользователь уже зарегистрирован, или регистрацию, если пользователь ещё не зарегистрирован. После входа в систему пользователь может увидеть свои записи и начать работу с ними. Также пользователь всегда может выйти обратно в меню регистрации и авторизации.

При регистрации и авторизации соблюдены защитные меры от повторения данных и их неправильного ввода.

При добавлении Имя и Номер являются обязательными, в то время как Дополнительный номер и Email – необязательными.

2.7.1 Общие сведения об информационной системе

Информационная система разработана для хранения номеров различных людей, а также их почты. Язык интерфейса – русский.

2.7.2 Требования к техническим средствам

Необходимо иметь локальный компьютер с установленной Microsoft SQL Management Studio, в которой необходимо создать базу данных приложения используя скрипт.

2.7.3 Требования к программным средствам

Для использования программы на локальном компьютере пользователя должны быть установлены Microsoft SQL Management Studio 2017 или выше, а также сама программа и база данных, приложенная к ней.

2.7.4 Настройка информационной системы

Для корректной работы приложения необходимо создать базу данных на другом сервере используя приложенный скрипт, а также изменить модель базы данных в приложении включая строку подключения базы данных учитывая ее расположение на локальном компьютере.

2.7.5 Формы ввода

В приложении присутствуют такие формы ввода как:

* аутентификация. Содержит поля логин и пароль,
* добавление записи о человеке. Содержит поля имени, номера, дополнительного номера, выбора группы и почты.

# ЗАКЛЮЧЕНИЕ

Курсовой проект был посвящен разработке системы, позволяющей хранить телефонные данные абонентов, для разработки проекта использовались технологии, с которыми не приходилось раньше работать, однако проект успешно завершён, несмотря на множество трудностей, с которыми пришлось столкнуться.

В процессе выполнения курсового проекта:

1. изучена предметная область проекта,
2. спроектирована база данных для приложения,
3. разработано приложение «Электронный телефонный справочник»,
4. осуществлена защита ввода и сохранения данных в базу данных,
5. освоены методы работы с базой данных посредством приложения, основанного на технологии WPF C#.

Были выполнены следующие цели проекта:

* изучение технологии ADO.NET,
* изучения взаимодействия кода C# и базы данных посредством технологии ADO.NET,
* улучшение навыков проектирования баз данных и архитектуры приложений,
* создание базы данных для системы,
* написание информационной системы для спортивных фанатов.

Во время выполнения проекта возникало множество проблем, которые были успешно решены. В частности проблемы возникали при написании функций для взаимодействия с базой данных. Решение этих проблем помогло узнать много нового о работе Entity Framework. В будущем возможны доработки базы данных и приложения для принадлежностей к другим спортивным мероприятиям.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Руководство компании Microsoft по .NET Framework [Электронный ресурс]. – 2020 – URL: https://docs.microsoft.com/ru-ru/dotnet/framework/ (дата обращения 20.04.2020),
2. Руководство компании Microsoft по Entity Framework [Электронный ресурс]. – 2020 – URL: https://docs.microsoft.com/ru-ru/ef/ (дата обращения 22.04.2020),
3. Руководство по ADO.NET [Электронный ресурс]. – 2020 – URL: https://metanit.com/sharp/ado.net/1.1.php (дата обращения 20.04.2020),
4. Руководство от сайта Metanit по LINQ[Электронный ресурс]. – 2020 – URL: https://metanit.com/sharp/tutorial/15.1.php (дата обращения 5.05.2020),
5. Руководство компании Microsoft по LINQ [Электронный ресурс]. – 2020 – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/linq/ (дата обращения 5.05.2020).

# ПРИЛОЖЕНИЕ А SQL скрипты на создание и заполнение базы данных, на создание учетных записей пользователей, на назначение привилегий

Все скрипты на создание и заполнение базы данных доступны по ссылке на GitHub https://github.com/Umka57/FanNavigator.

USE [master]

GO

/\*\*\*\*\*\* Object: Database [KP] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

CREATE DATABASE [KP]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'KP', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\KP.mdf' , SIZE = 73728KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'KP\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\KP\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

GO

ALTER DATABASE [KP] SET COMPATIBILITY\_LEVEL = 140

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [KP].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [KP] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [KP] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [KP] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [KP] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [KP] SET ARITHABORT OFF

GO

ALTER DATABASE [KP] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [KP] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [KP] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [KP] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [KP] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [KP] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [KP] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [KP] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [KP] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [KP] SET DISABLE\_BROKER

GO

ALTER DATABASE [KP] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [KP] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [KP] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [KP] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [KP] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [KP] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [KP] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [KP] SET RECOVERY SIMPLE

GO

ALTER DATABASE [KP] SET MULTI\_USER

GO

ALTER DATABASE [KP] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [KP] SET DB\_CHAINING OFF

GO

ALTER DATABASE [KP] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [KP] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [KP] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [KP] SET QUERY\_STORE = OFF

GO

USE [KP]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Matches] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Matches](

[ID] [int] NOT NULL,

[Competition\_ID] [int] NOT NULL,

[Team1\_ID] [int] NOT NULL,

[Team2\_ID] [int] NOT NULL,

[Score1] [int] NOT NULL,

[Score2] [int] NULL,

[Status] [nvarchar](50) NULL,

[Date] [date] NULL,

[TimeStart] [time](4) NULL,

[Place] [nvarchar](50) NULL,

CONSTRAINT [PK\_Matches] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Sportsmans] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Sportsmans](

[ID] [int] NOT NULL,

[LastName] [nvarchar](50) NOT NULL,

[FirstName] [nvarchar](50) NOT NULL,

[Nickname] [nvarchar](50) NOT NULL,

[Team\_ID] [int] NOT NULL,

[Photo] [varbinary](max) NULL,

CONSTRAINT [PK\_Sportsmans] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Players\_Results] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Players\_Results](

[Tournament\_ID] [int] NOT NULL,

[Match\_ID] [int] NOT NULL,

[Player\_ID] [int] NOT NULL,

[Score] [int] NOT NULL,

CONSTRAINT [PK\_Players\_Results] PRIMARY KEY CLUSTERED

(

[Tournament\_ID] ASC,

[Match\_ID] ASC,

[Player\_ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[MVP\_Players] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[MVP\_Players]

AS

SELECT dbo.Sportsmans.LastName, dbo.Sportsmans.Nickname, SUM(dbo.Players\_Results.Score) AS Score, dbo.Matches.Competition\_ID

FROM dbo.Sportsmans INNER JOIN

dbo.Players\_Results ON dbo.Sportsmans.ID = dbo.Players\_Results.Player\_ID INNER JOIN

dbo.Matches ON dbo.Sportsmans.ID = dbo.Matches.ID

GROUP BY dbo.Sportsmans.Nickname, dbo.Sportsmans.LastName, dbo.Matches.Competition\_ID

GO

/\*\*\*\*\*\* Object: Table [dbo].[Disciplines] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Disciplines](

[Discipline\_Name] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Disciplines] PRIMARY KEY CLUSTERED

(

[Discipline\_Name] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Hosts] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Hosts](

[Host\_Name] [nvarchar](250) NOT NULL,

CONSTRAINT [PK\_Hosts] PRIMARY KEY CLUSTERED

(

[Host\_Name] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Match\_Types] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Match\_Types](

[Type] [nvarchar](10) NOT NULL,

CONSTRAINT [PK\_Match\_Types] PRIMARY KEY CLUSTERED

(

[Type] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[MembersList] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[MembersList](

[ID\_Tournament] [int] NOT NULL,

[ID\_Team] [int] NOT NULL,

CONSTRAINT [PK\_MembersList] PRIMARY KEY CLUSTERED

(

[ID\_Tournament] ASC,

[ID\_Team] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Rating] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Rating](

[ID\_Competition] [int] NOT NULL,

[ID\_Team] [int] NOT NULL,

[Score] [int] NULL,

[Place] [int] NULL,

CONSTRAINT [PK\_Rating] PRIMARY KEY CLUSTERED

(

[ID\_Competition] ASC,

[ID\_Team] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Teams] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Teams](

[ID] [int] NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Discipline] [nvarchar](50) NOT NULL,

[Logo] [varbinary](max) NULL,

CONSTRAINT [PK\_Teams\_1] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Tournaments] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Tournaments](

[ID] [int] NOT NULL,

[Logo] [varbinary](max) NULL,

[Name] [nvarchar](250) NOT NULL,

[Host] [nvarchar](250) NOT NULL,

[Prize] [int] NOT NULL,

[Location] [nvarchar](250) NOT NULL,

[Type] [nvarchar](10) NOT NULL,

[Discipline] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Competitions] PRIMARY KEY CLUSTERED

(

[ID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Users] Script Date: 14.05.2020 22:25:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Users](

[login] [nvarchar](25) NOT NULL,

[password] [nvarchar](25) NOT NULL,

[role] [int] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Matches] WITH CHECK ADD CONSTRAINT [FK\_Matches\_Competitions] FOREIGN KEY([Competition\_ID])

REFERENCES [dbo].[Tournaments] ([ID])

GO

ALTER TABLE [dbo].[Matches] CHECK CONSTRAINT [FK\_Matches\_Competitions]

GO

ALTER TABLE [dbo].[Matches] WITH CHECK ADD CONSTRAINT [FK\_Matches\_Teams] FOREIGN KEY([Team1\_ID])

REFERENCES [dbo].[Teams] ([ID])

GO

ALTER TABLE [dbo].[Matches] CHECK CONSTRAINT [FK\_Matches\_Teams]

GO

ALTER TABLE [dbo].[Matches] WITH CHECK ADD CONSTRAINT [FK\_Matches\_Teams1] FOREIGN KEY([Team2\_ID])

REFERENCES [dbo].[Teams] ([ID])

GO

ALTER TABLE [dbo].[Matches] CHECK CONSTRAINT [FK\_Matches\_Teams1]

GO

ALTER TABLE [dbo].[MembersList] WITH CHECK ADD CONSTRAINT [FK\_MembersList\_Teams] FOREIGN KEY([ID\_Team])

REFERENCES [dbo].[Teams] ([ID])

GO

ALTER TABLE [dbo].[MembersList] CHECK CONSTRAINT [FK\_MembersList\_Teams]

GO

ALTER TABLE [dbo].[MembersList] WITH CHECK ADD CONSTRAINT [FK\_MembersList\_Tournaments] FOREIGN KEY([ID\_Tournament])

REFERENCES [dbo].[Tournaments] ([ID])

GO

ALTER TABLE [dbo].[MembersList] CHECK CONSTRAINT [FK\_MembersList\_Tournaments]

GO

ALTER TABLE [dbo].[Players\_Results] WITH CHECK ADD CONSTRAINT [FK\_Players\_Results\_Matches] FOREIGN KEY([Match\_ID])

REFERENCES [dbo].[Matches] ([ID])

GO

ALTER TABLE [dbo].[Players\_Results] CHECK CONSTRAINT [FK\_Players\_Results\_Matches]

GO

ALTER TABLE [dbo].[Players\_Results] WITH CHECK ADD CONSTRAINT [FK\_Players\_Results\_Sportsmans] FOREIGN KEY([Player\_ID])

REFERENCES [dbo].[Sportsmans] ([ID])

GO

ALTER TABLE [dbo].[Players\_Results] CHECK CONSTRAINT [FK\_Players\_Results\_Sportsmans]

GO

ALTER TABLE [dbo].[Players\_Results] WITH CHECK ADD CONSTRAINT [FK\_Players\_Results\_Sportsmans1] FOREIGN KEY([Player\_ID])

REFERENCES [dbo].[Sportsmans] ([ID])

GO

ALTER TABLE [dbo].[Players\_Results] CHECK CONSTRAINT [FK\_Players\_Results\_Sportsmans1]

GO

ALTER TABLE [dbo].[Players\_Results] WITH CHECK ADD CONSTRAINT [FK\_Players\_Results\_Tournaments] FOREIGN KEY([Tournament\_ID])

REFERENCES [dbo].[Tournaments] ([ID])

GO

ALTER TABLE [dbo].[Players\_Results] CHECK CONSTRAINT [FK\_Players\_Results\_Tournaments]

GO

ALTER TABLE [dbo].[Rating] WITH CHECK ADD CONSTRAINT [FK\_Rating\_Competitions] FOREIGN KEY([ID\_Competition])

REFERENCES [dbo].[Tournaments] ([ID])

GO

ALTER TABLE [dbo].[Rating] CHECK CONSTRAINT [FK\_Rating\_Competitions]

GO

ALTER TABLE [dbo].[Rating] WITH CHECK ADD CONSTRAINT [FK\_Rating\_Teams] FOREIGN KEY([ID\_Team])

REFERENCES [dbo].[Teams] ([ID])

GO

ALTER TABLE [dbo].[Rating] CHECK CONSTRAINT [FK\_Rating\_Teams]

GO

ALTER TABLE [dbo].[Sportsmans] WITH CHECK ADD CONSTRAINT [FK\_Sportsmans\_Teams] FOREIGN KEY([Team\_ID])

REFERENCES [dbo].[Teams] ([ID])

GO

ALTER TABLE [dbo].[Sportsmans] CHECK CONSTRAINT [FK\_Sportsmans\_Teams]

GO

ALTER TABLE [dbo].[Tournaments] WITH CHECK ADD CONSTRAINT [FK\_Tournaments\_Disciplines] FOREIGN KEY([Discipline])

REFERENCES [dbo].[Disciplines] ([Discipline\_Name])

GO

ALTER TABLE [dbo].[Tournaments] CHECK CONSTRAINT [FK\_Tournaments\_Disciplines]

GO

ALTER TABLE [dbo].[Tournaments] WITH CHECK ADD CONSTRAINT [FK\_Tournaments\_Hosts] FOREIGN KEY([Host])

REFERENCES [dbo].[Hosts] ([Host\_Name])

GO

ALTER TABLE [dbo].[Tournaments] CHECK CONSTRAINT [FK\_Tournaments\_Hosts]

GO

ALTER TABLE [dbo].[Tournaments] WITH CHECK ADD CONSTRAINT [FK\_Tournaments\_Match\_Types] FOREIGN KEY([Type])

REFERENCES [dbo].[Match\_Types] ([Type])

GO

ALTER TABLE [dbo].[Tournaments] CHECK CONSTRAINT [FK\_Tournaments\_Match\_Types]

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F-00AA00A3EFFF, 1.00]

Begin DesignProperties =

Begin PaneConfigurations =

Begin PaneConfiguration = 0

NumPanes = 4

Configuration = "(H (1[40] 4[20] 2[20] 3) )"

End

Begin PaneConfiguration = 1

NumPanes = 3

Configuration = "(H (1 [50] 4 [25] 3))"

End

Begin PaneConfiguration = 2

NumPanes = 3

Configuration = "(H (1 [50] 2 [25] 3))"

End

Begin PaneConfiguration = 3

NumPanes = 3

Configuration = "(H (4 [30] 2 [40] 3))"

End

Begin PaneConfiguration = 4

NumPanes = 2

Configuration = "(H (1 [56] 3))"

End

Begin PaneConfiguration = 5

NumPanes = 2

Configuration = "(H (2 [66] 3))"

End

Begin PaneConfiguration = 6

NumPanes = 2

Configuration = "(H (4 [50] 3))"

End

Begin PaneConfiguration = 7

NumPanes = 1

Configuration = "(V (3))"

End

Begin PaneConfiguration = 8

NumPanes = 3

Configuration = "(H (1[56] 4[18] 2) )"

End

Begin PaneConfiguration = 9

NumPanes = 2

Configuration = "(H (1 [75] 4))"

End

Begin PaneConfiguration = 10

NumPanes = 2

Configuration = "(H (1[66] 2) )"

End

Begin PaneConfiguration = 11

NumPanes = 2

Configuration = "(H (4 [60] 2))"

End

Begin PaneConfiguration = 12

NumPanes = 1

Configuration = "(H (1) )"

End

Begin PaneConfiguration = 13

NumPanes = 1

Configuration = "(V (4))"

End

Begin PaneConfiguration = 14

NumPanes = 1

Configuration = "(V (2))"

End

ActivePaneConfig = 0

End

Begin DiagramPane =

Begin Origin =

Top = 0

Left = 0

End

Begin Tables =

Begin Table = "Sportsmans"

Begin Extent =

Top = 7

Left = 48

Bottom = 237

Right = 249

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Players\_Results"

Begin Extent =

Top = 8

Left = 348

Bottom = 154

Right = 549

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Matches"

Begin Extent =

Top = 7

Left = 597

Bottom = 170

Right = 814

End

DisplayFlags = 280

TopColumn = 0

End

End

End

Begin SQLPane =

End

Begin DataPane =

Begin ParameterDefaults = ""

End

Begin ColumnWidths = 9

Width = 284

Width = 1200

Width = 1200

Width = 1200

Width = 1200

Width = 1200

Width = 1200

Width = 1200

Width = 1200

End

End

Begin CriteriaPane =

Begin ColumnWidths = 12

Column = 1440

Alias = 900

Table = 1176

Output = 720

Append = 1400

NewValue = 1170

SortType = 1356

SortOrder = 1416

GroupBy = 1350

Filter = 1356

Or = 1350

Or = 1350

Or = 1350

End

End

End

' , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'MVP\_Players'

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPaneCount', @value=1 , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'MVP\_Players'

GO

USE [master]

GO

ALTER DATABASE [KP] SET READ\_WRITE

GO

# ПРИЛОЖЕНИЕ Б Код приложения

Полный код приложения доступен по ссылке на GitHub https://github.com/Umka57/FanNavigator.

public void aaa()

{

if (Name.Text != "")

{

if (Number.Text != "")

{

conn.Open();

string na = Name.Text;

string nu = Number.Text;

dt = new DataTable();

da = new SqlDataAdapter();

SqlCommand con = new SqlCommand(@"SELECT \* FROM dbo.Cheliki WHERE Name = '" + na + "'and Number = '" + nu + "' and user\_id = '" + Id + "'", conn);

da.SelectCommand = con;

da.Fill(dt);

conn.Close();

if (dt.Rows.Count == 0)

{

bl = true;

try

{

conn.Open();

using (SqlCommand sql = conn.CreateCommand())

{

if (Group.SelectedItem == Nothing)

{

IDgr = "NULL";

}

if (Group.SelectedItem == kollbox)

{

IDgr = "1";

}

if (Group.SelectedItem == druzbox)

{

IDgr = "2";

}

if (Group.SelectedItem == sembox)

{

IDgr = "3";

}

SqlCommand com = new SqlCommand(@"INSERT INTO Cheliki(Name, Number, User\_id, DopNumber, Email, gr\_id)" + "VALUES('" + Name.Text + "','" + Number.Text + "','" + Id + "','" + DopNumber.Text + "','" + Email.Text + "'," + IDgr + ")", conn);

com.ExecuteNonQuery();

MessageBox.Show("Успешно сохранено");

da = new SqlDataAdapter(@"SELECT dbo.Cheliki.Name, dbo.Cheliki.Number, dbo.Cheliki.DopNumber, dbo.Cheliki.Email FROM dbo.Cheliki WHERE user\_id = '" + Id + "'", conn);

dt = new DataTable();

da.Fill(dt);

BD.ItemsSource = dt.AsDataView();

IDgr = null;

}

conn.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

conn.Close();

}

} else

MessageBox.Show("Пользователь с таким именем или телефоном уже существует");

}

else

MessageBox.Show("Номер телефона отсутствует");

}

else

MessageBox.Show("Имя не введено");

}