**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Российский экономический университет имени Г. В. Плеханова»**

**Московский приборостроительный техникум**

**Отчёт**

**по выполнению практической работы № 5**

По дисциплине / **Основы алгоритмизации и программирования**

МДК:

Тема: **Разработка клиентского приложения. Часть 2.**

**Обеспечение безопасности приложения.**

**Сохранение данных аутентификации**

Специальность: **09.02.07 «Информационные системы и**

**программирование»**

Квалификация: **Разработчик веб и мультимедийных приложений**

Выполнил: Монахов К.М.

Студент группы: ВД50-1-18

Принял: Горшков М. Н.

Дата: «\_\_» \_\_\_\_\_\_\_\_\_ 202\_ г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Цель работы:**

Произвести модификацию базы данных на виртуальном веб-сервере

с целью обеспечения безопасной аутентификации в программном обеспечении

для дальнейшей работы внутри информационной системы в клиентском

приложении на универсальной платформе Windows.

**Ход работы:**

1. Пункт

Запустить сборку виртуального веб-сервера и внутренние процессы

Apache и MySQL. Перейти в рабочую базу данных в СУБД phpMyAdmin

создайте виртуальную таблицу (представление) urers\_list, содержащее в себе

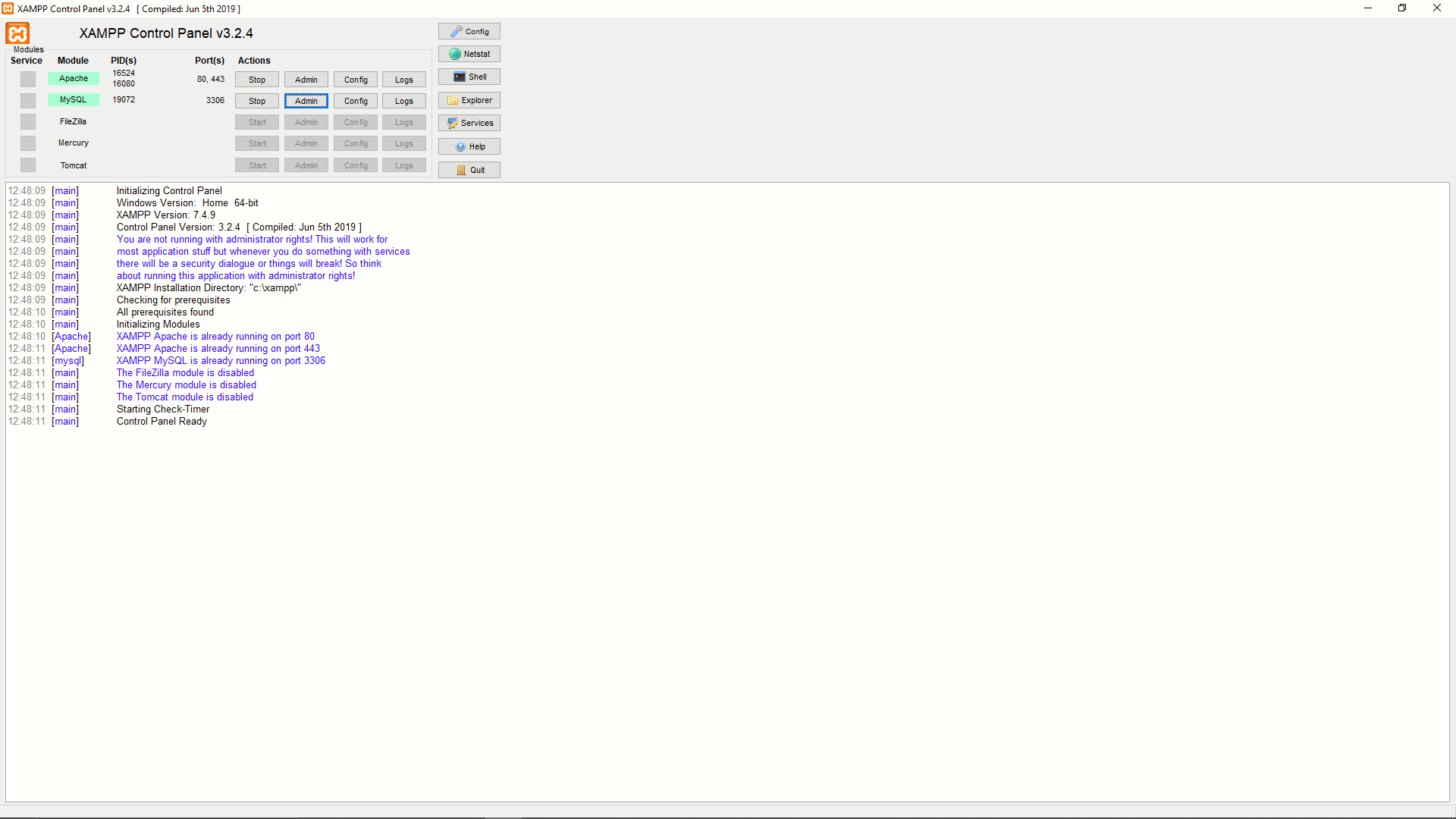
следующую информацию:.

Рисунок 1 – запуск программ Apache и MySQL.

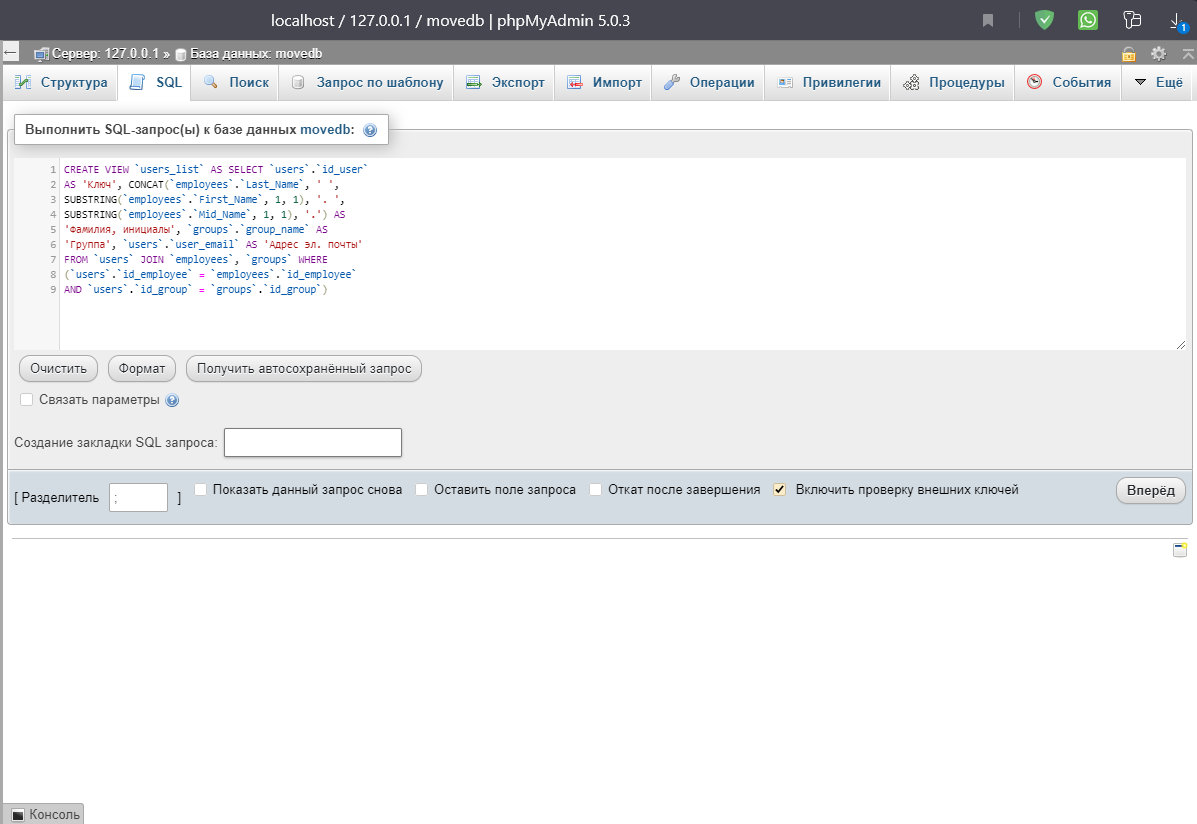
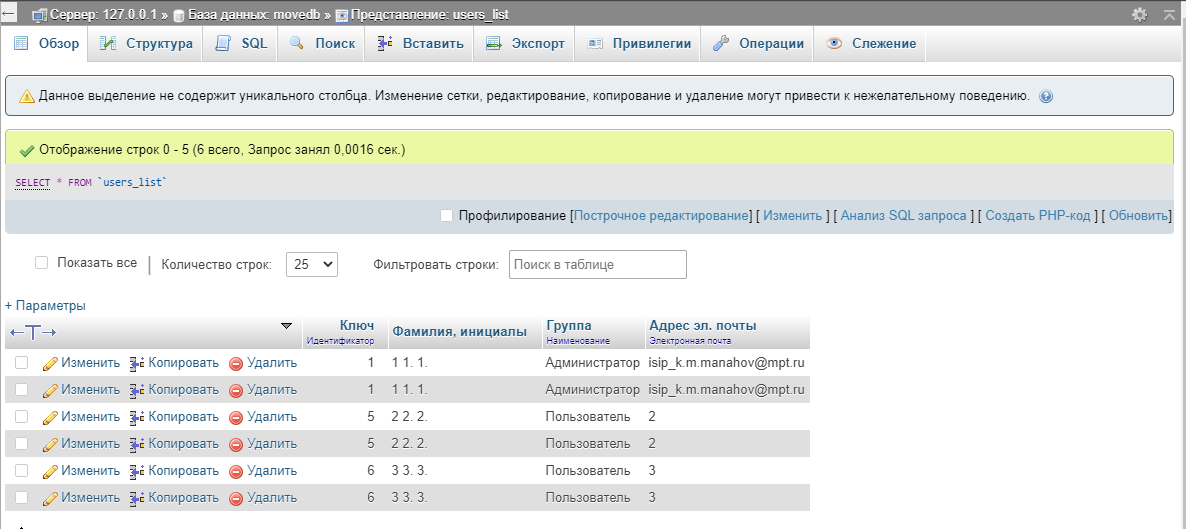


Рисунок 2 – SQL запрос для создания таблицы



Созданная таблица. Не знаю в чем проблема повторений данных но и с такой таблице программа работает отлично.

1. Пункт

На странице PanelFrame создайте Frame с атрибутом x:Name и

значением userMenu. Далее в программном коде реализуйте функцию при

помощи конструкции выбор, позволяющую динамически добавлять новые

элементы в NavigationView в зависимости от используемой группы

пользователя. В случае, если авторизованный пользователь является

администратором, необходимо добавление кнопки «Пользователи» в левое

меню, с помощью которой будет происходит перемещение на новую страницу Users.

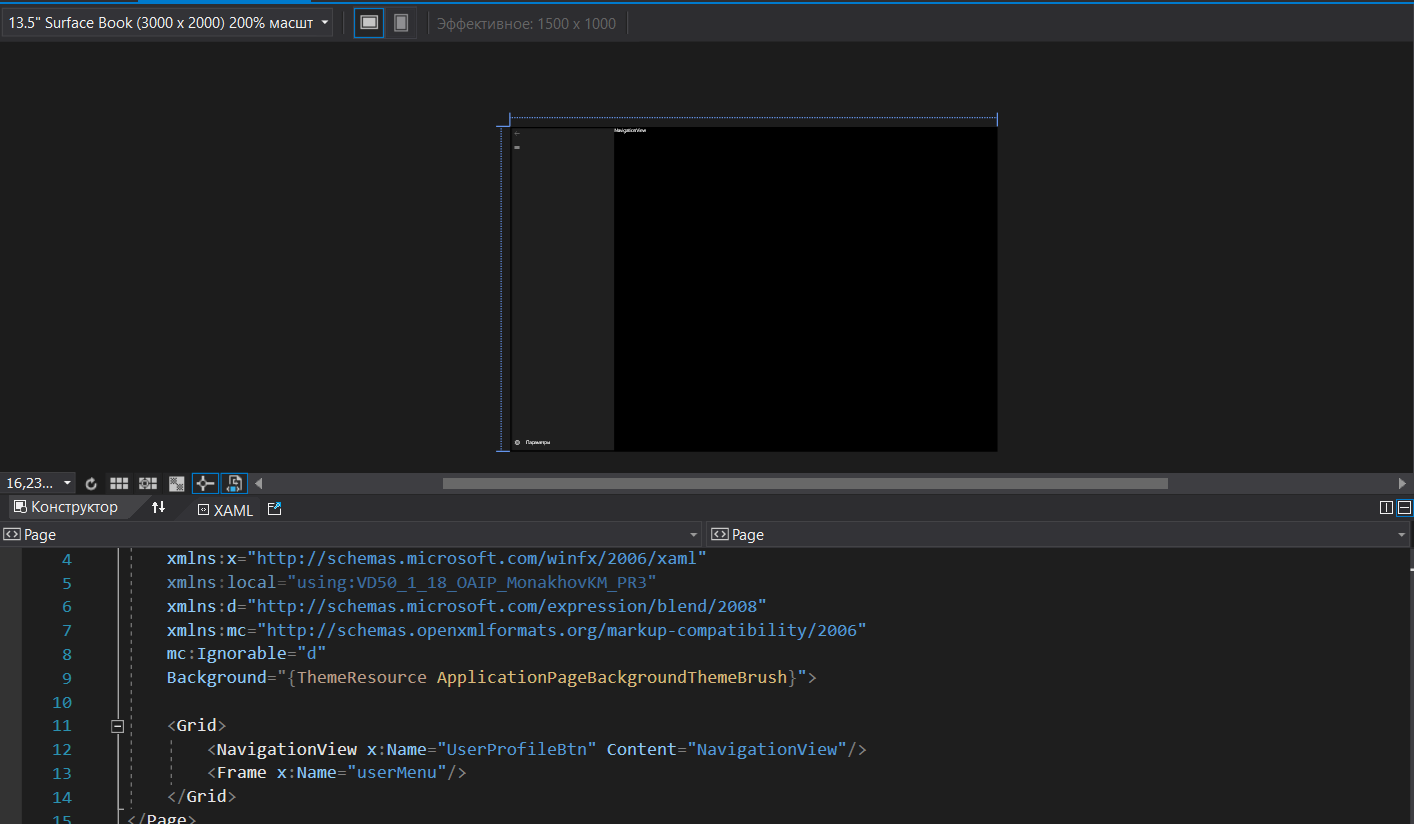
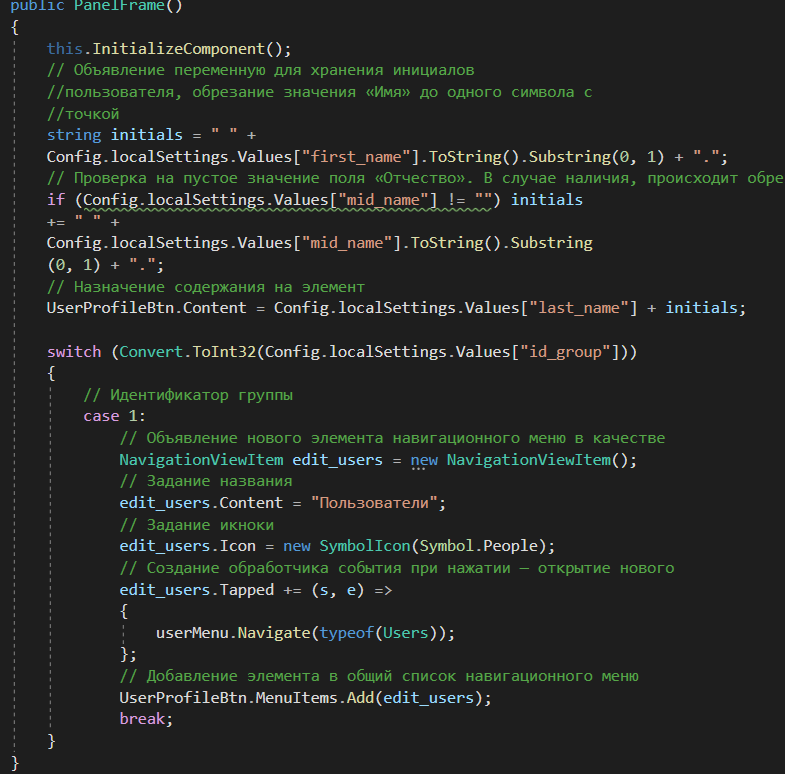
Рисунок 5 – создание Frame 

Рисунок 6 – изменение кода программы

1. Пункт

При помощи меню «Проект» → «Управление пакетами NuGet»

произведите установку пакета Microsoft.Toolkit.Uwp.UI.Controls.DataGrid,

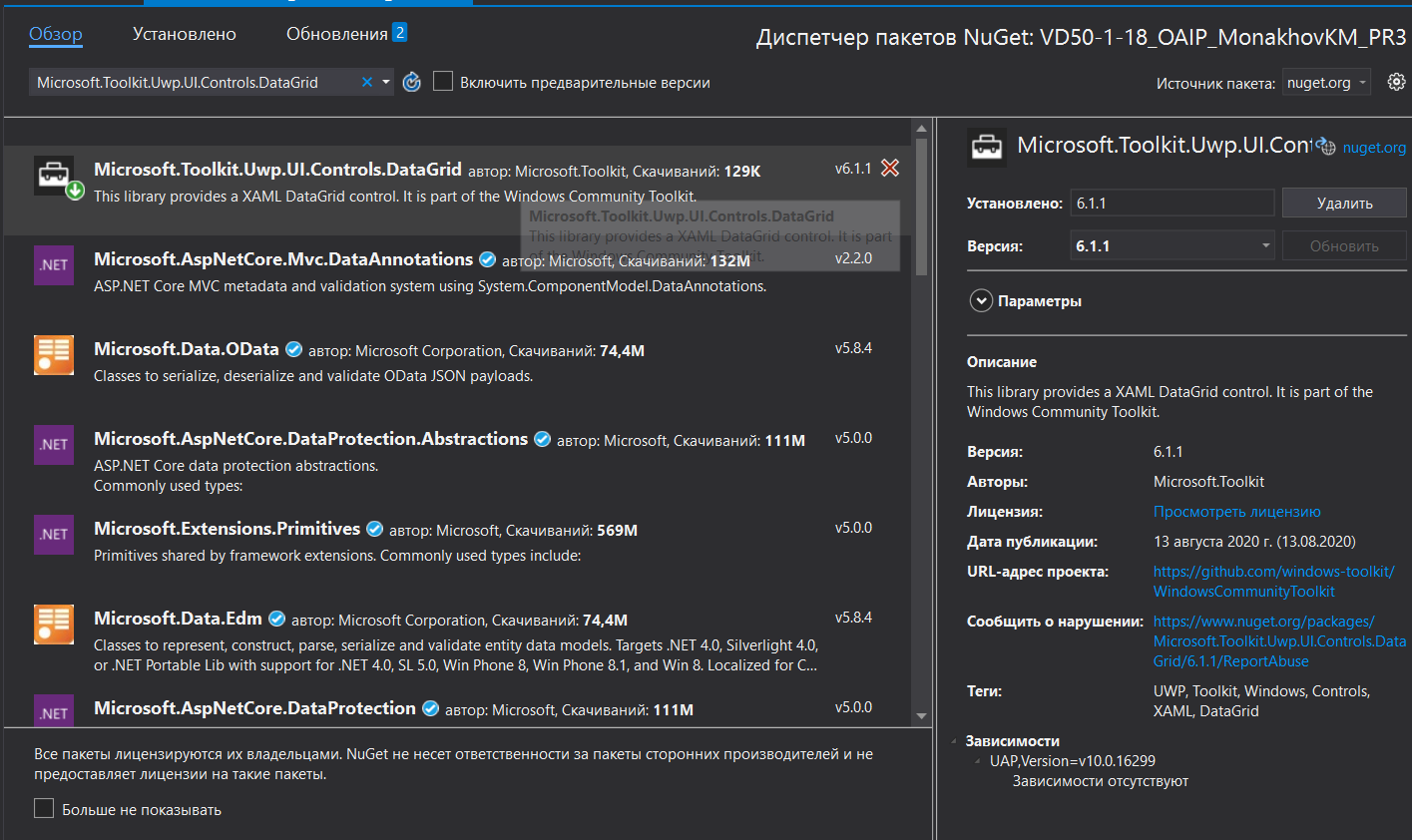
который позволит создаваться таблицу с использованием данных. 

Рисунок 8 – установка пакета

1. Пунтк

Создайте новую пустую страницу Users.xaml, внутри которой будут

находиться следующие элементы со свойствами:

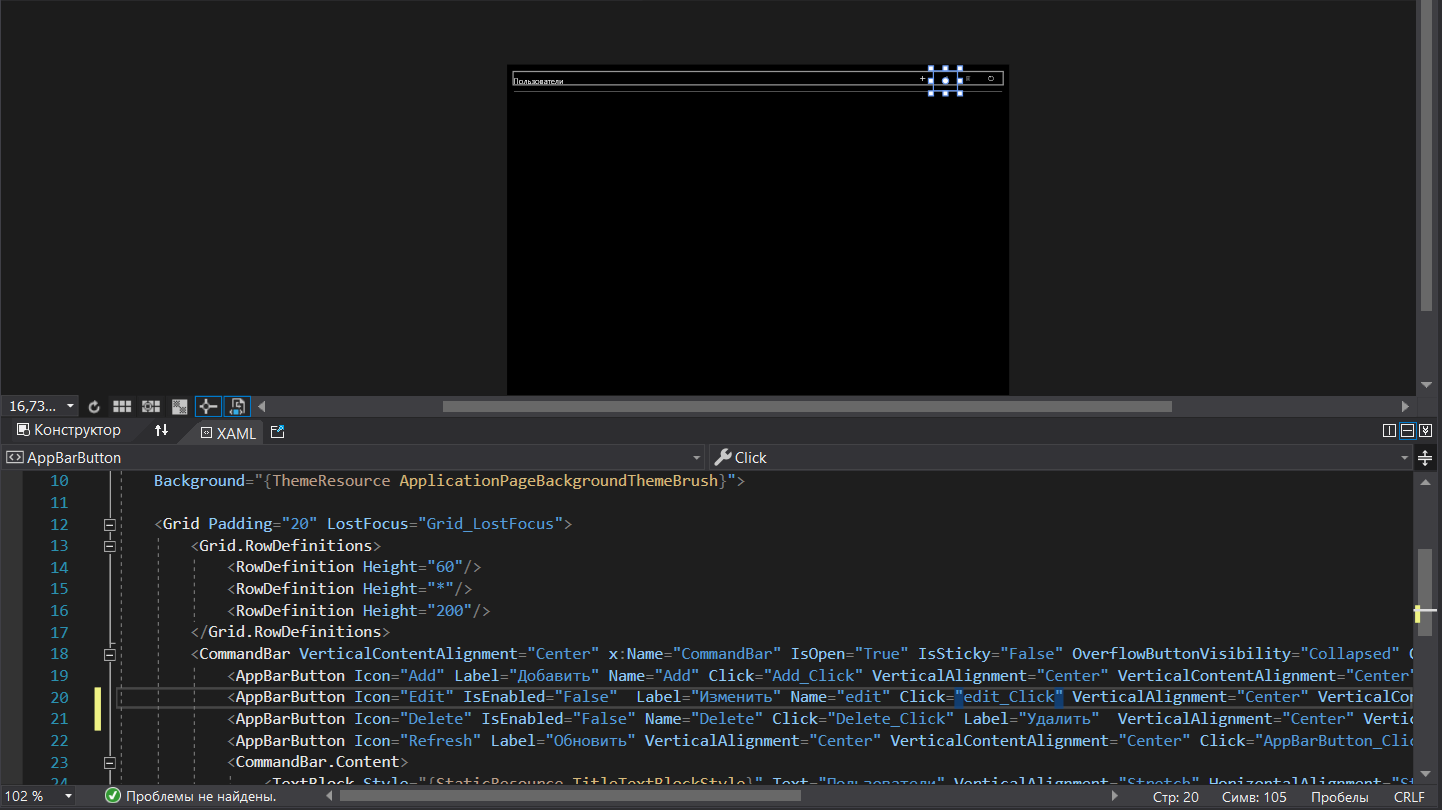


Рисунок 14 – Создание нового окна

1. Пункт

В программном коде файла Users.xaml.cs реализуйте метод GetList(),

который будет получать информацию и представление users\_list

и импортировать её в таблицу userList. Он должен срабатывать при открытии

страницы и нажатии на кнопку «Обновить».

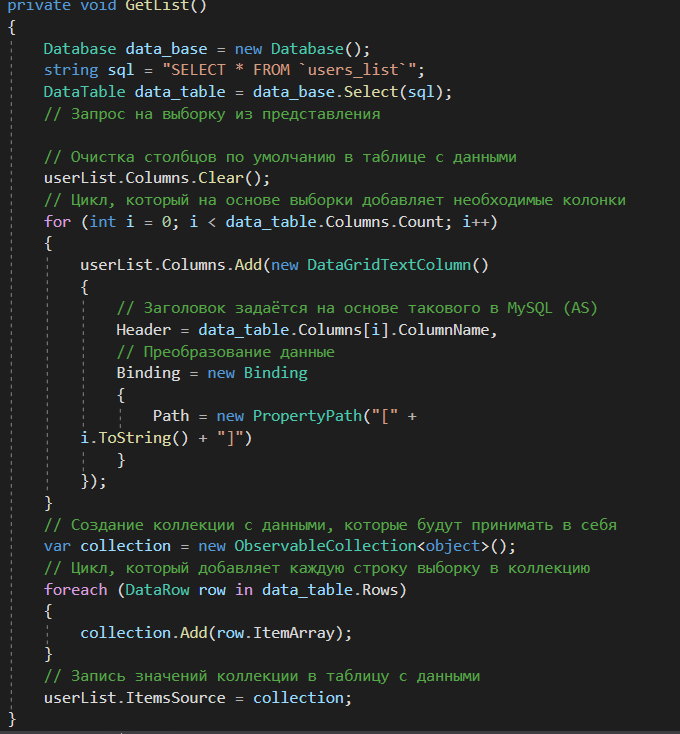


Рисунок 15 – метод GetList.cs

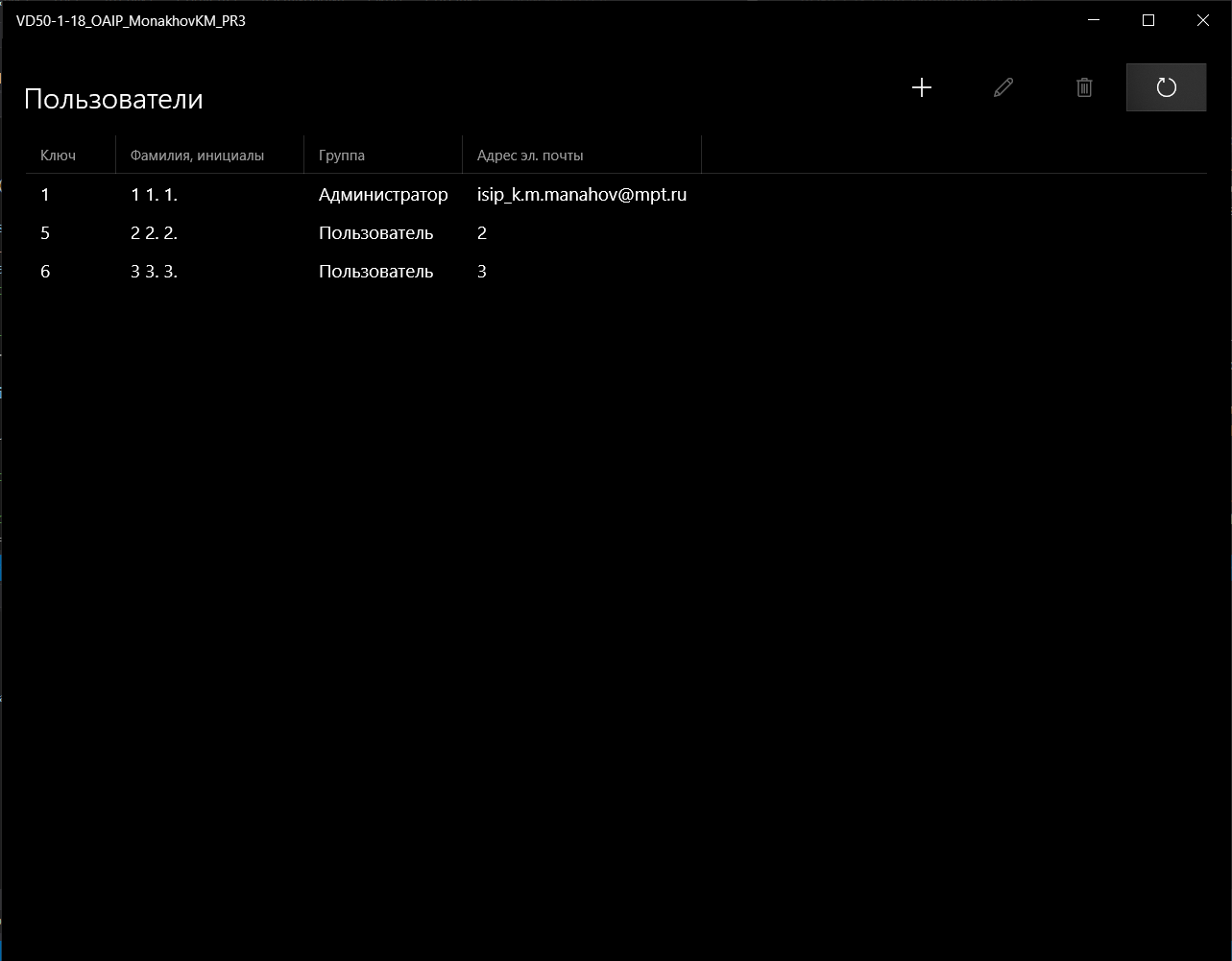


Рисунок 16 – Результат работы

1. Пункт

Реализовать окно, которое будет открываться по нажатию на кнопку

«Добавить» с возможность создания нового пользователя. На странице

должны содержаться следующие элементы:

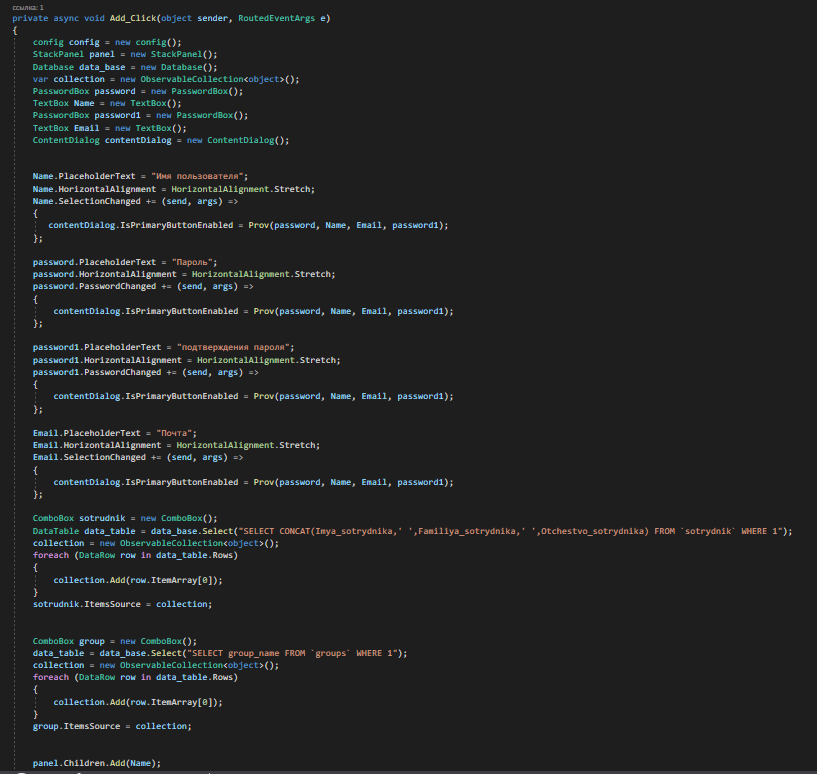


Рисунок 17 – программное создание окна добавления нового пользователя

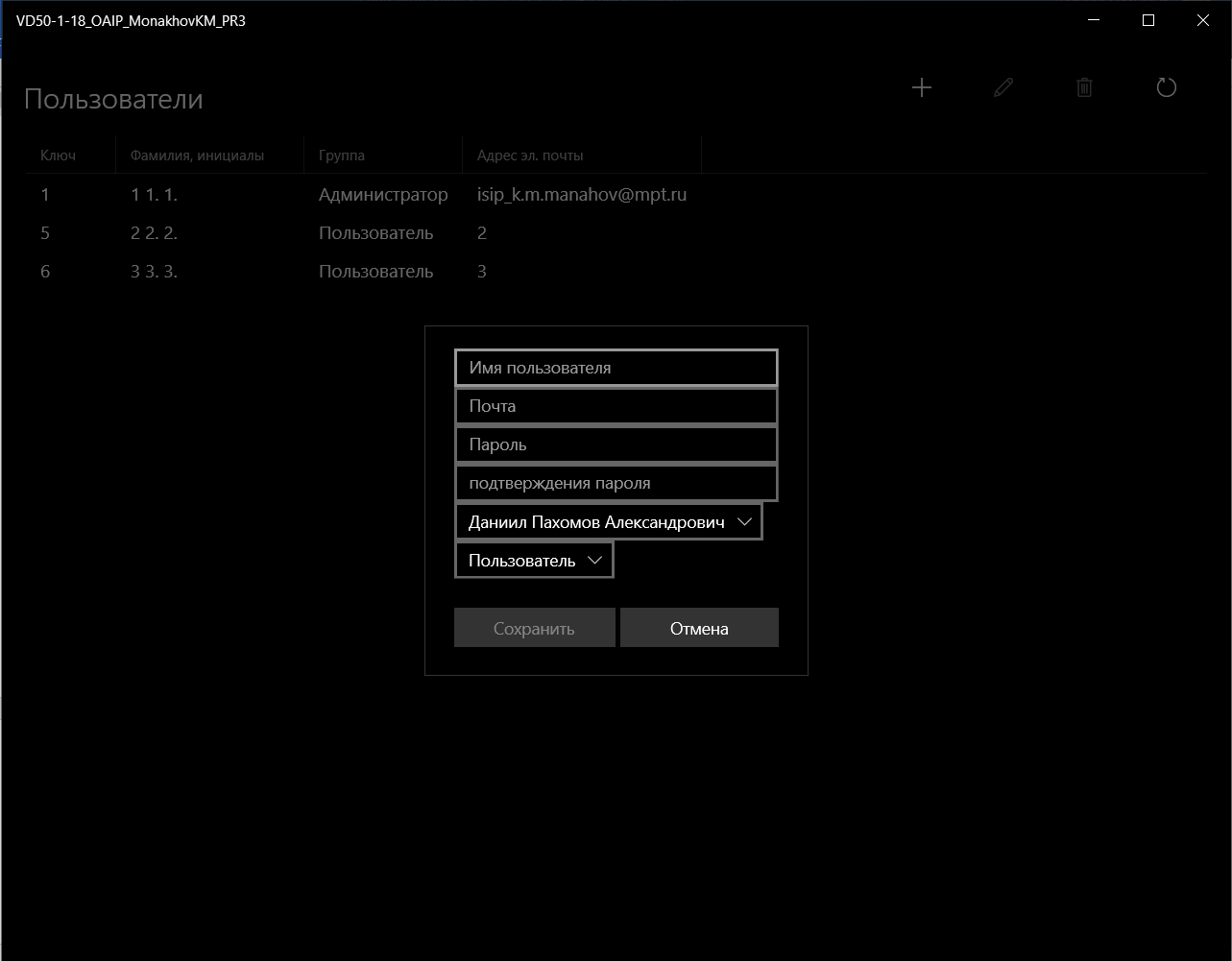


Рисунок 18 – выполнение работы

1. Пункт

Реализовать возможность выбор пользователя из таблицы по нажатию

на строку, что позволит получить его идентификатор для открытия

следующего окна. При выборе строки, кнопки «Изменить» и «Удалить»

должны становиться активными.

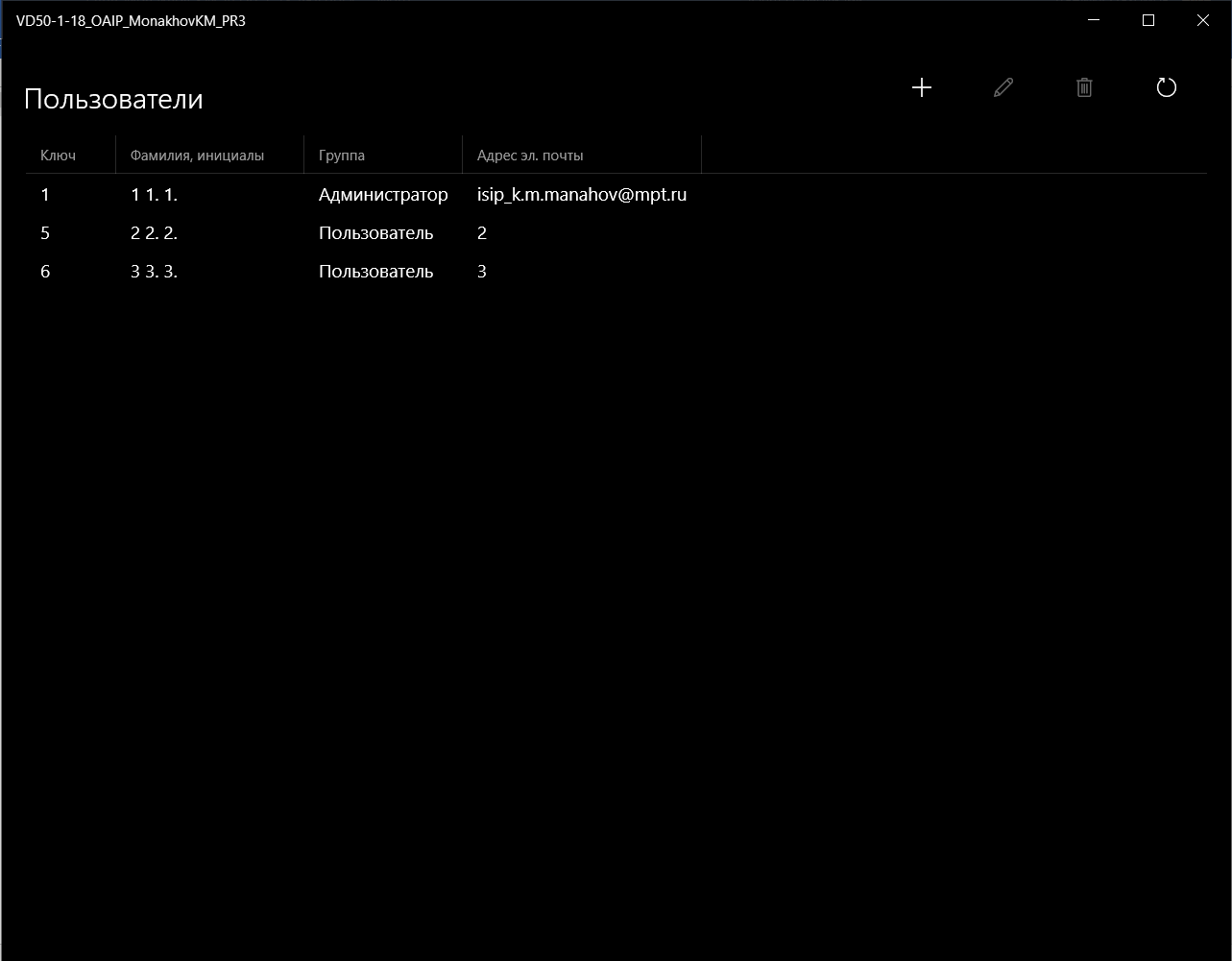


Рисунок 23 – Реализация выбора строки и активность удаления и измененния поля

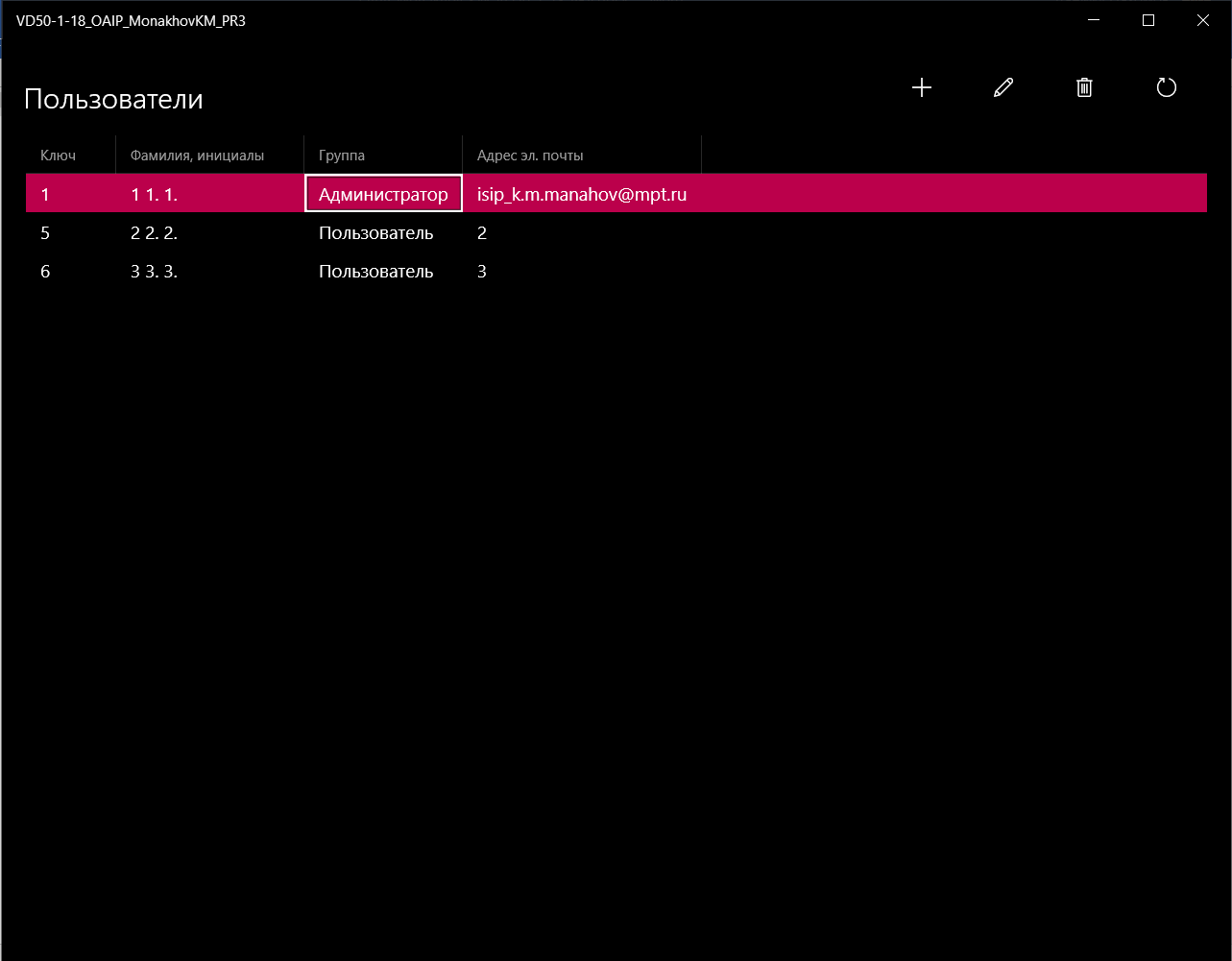


Рисунок 24 – выбор элемента

1. Пункт

При нажатии на кнопку «Удалить» должно появляться всплывающее окно с вопросом «Вы действительно хотите удалить этого пользователя?» с вариантами ответа «Да» и «Нет». После нажатия «Да» необходимо выполнить два SQL-запроса: на удаление пользователя из базы данных и для деактивации его токенов безопасности по идентификатору.

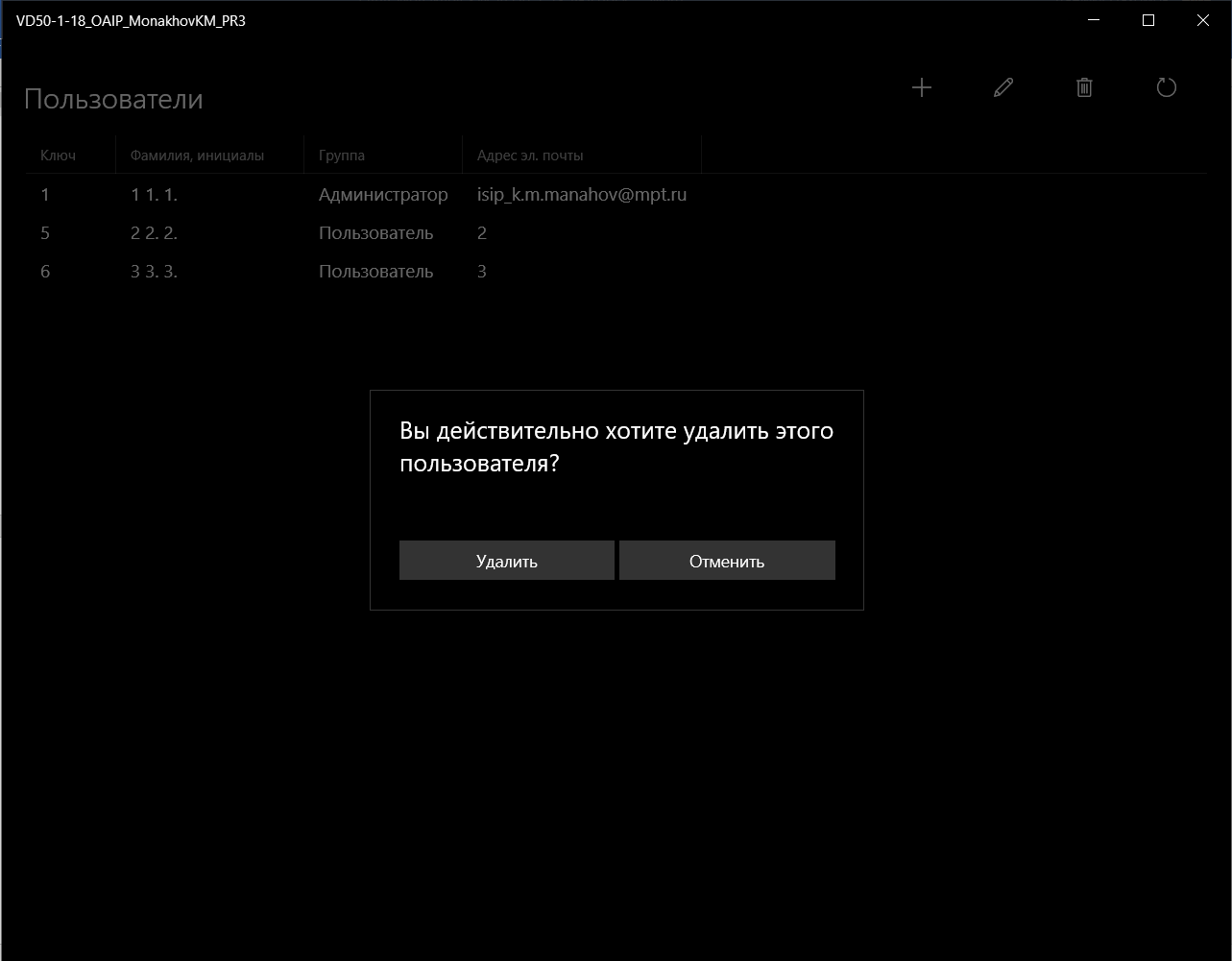


Рисунок 24 – результат удаления и появления диалогового окна.

1. Пункт

При нажатии на кнопку «Изменить» открывается окно на основе добавления пользователя с возможностью изменить только следующие атрибуты:

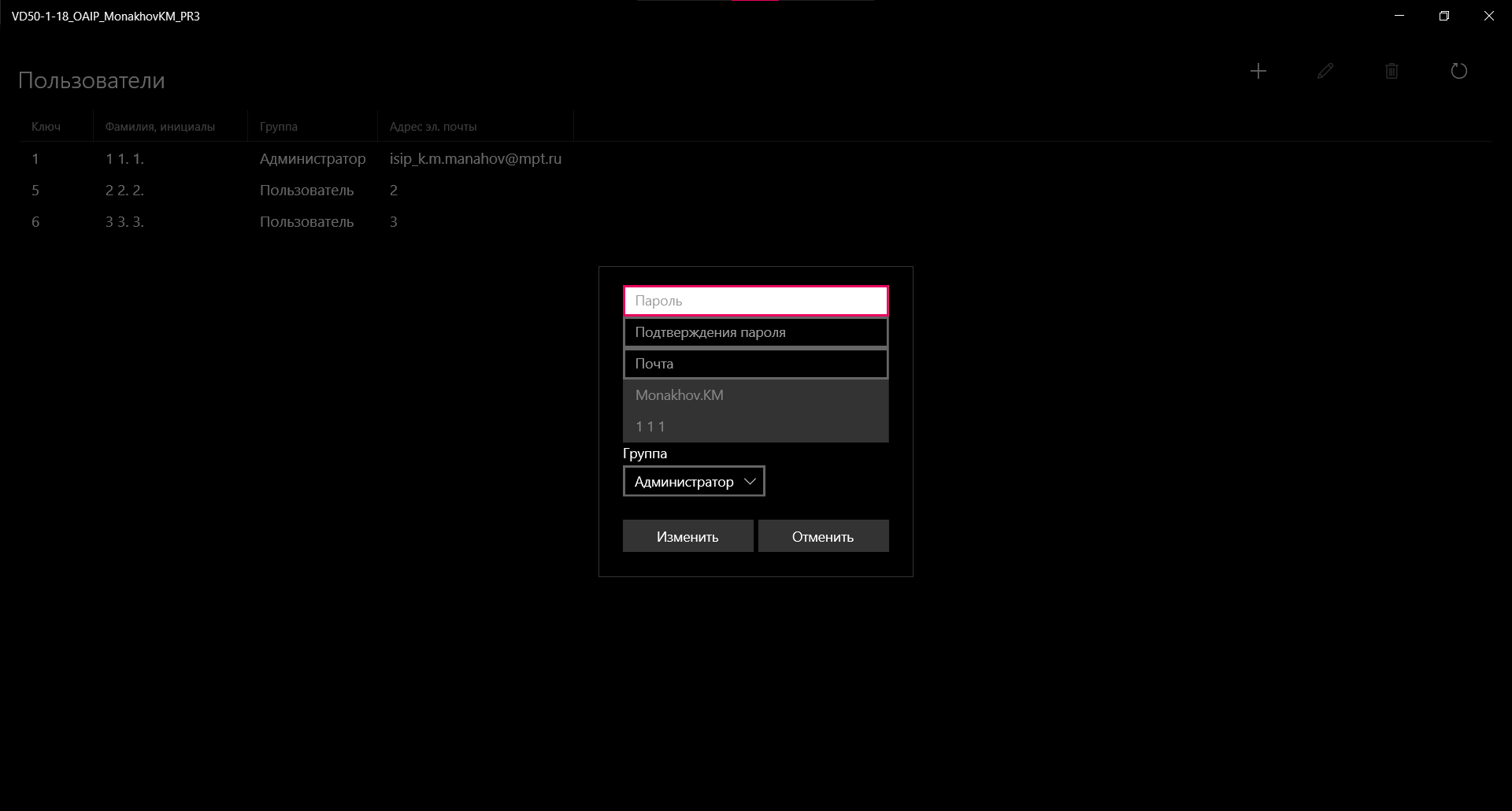
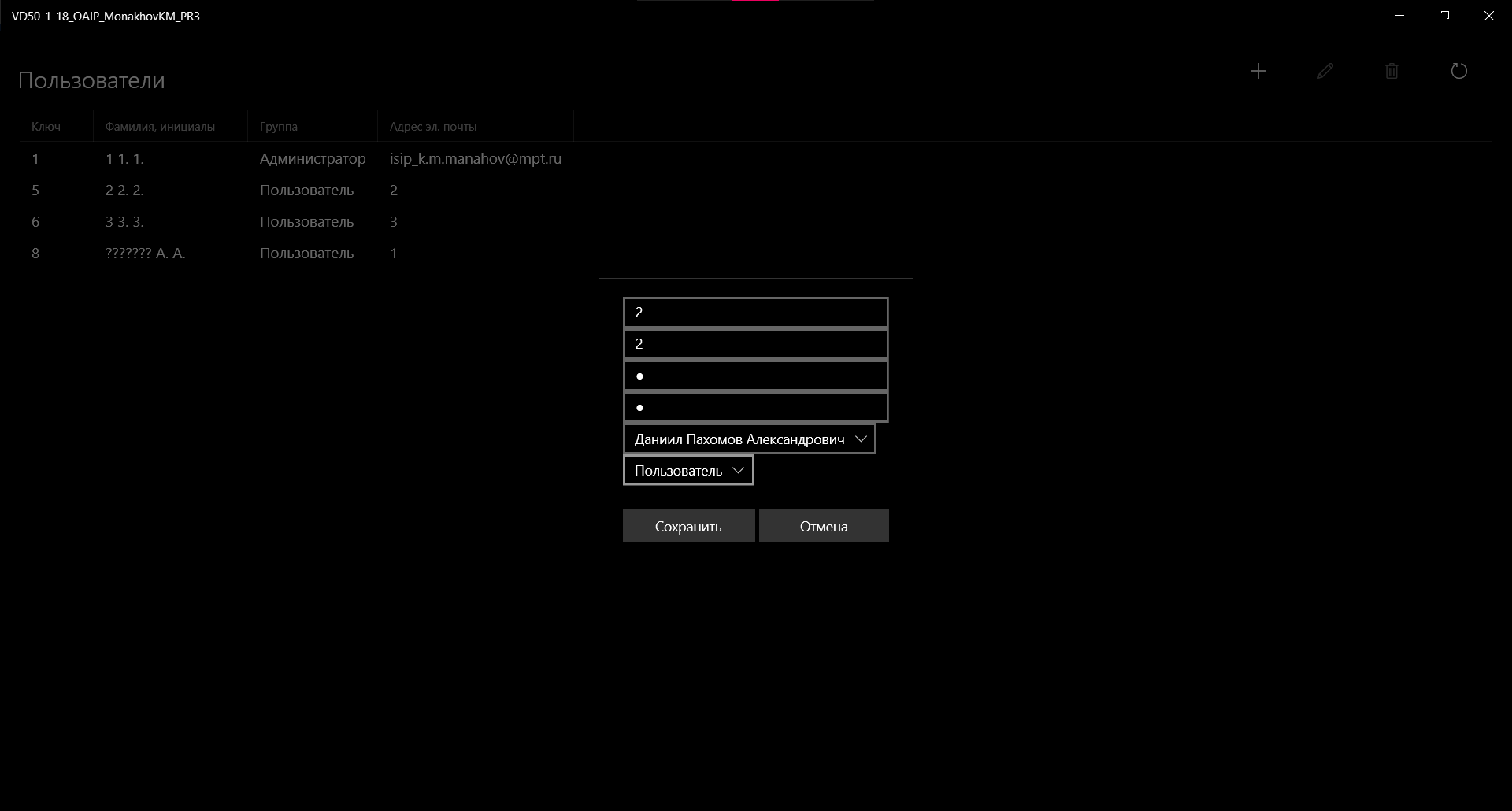
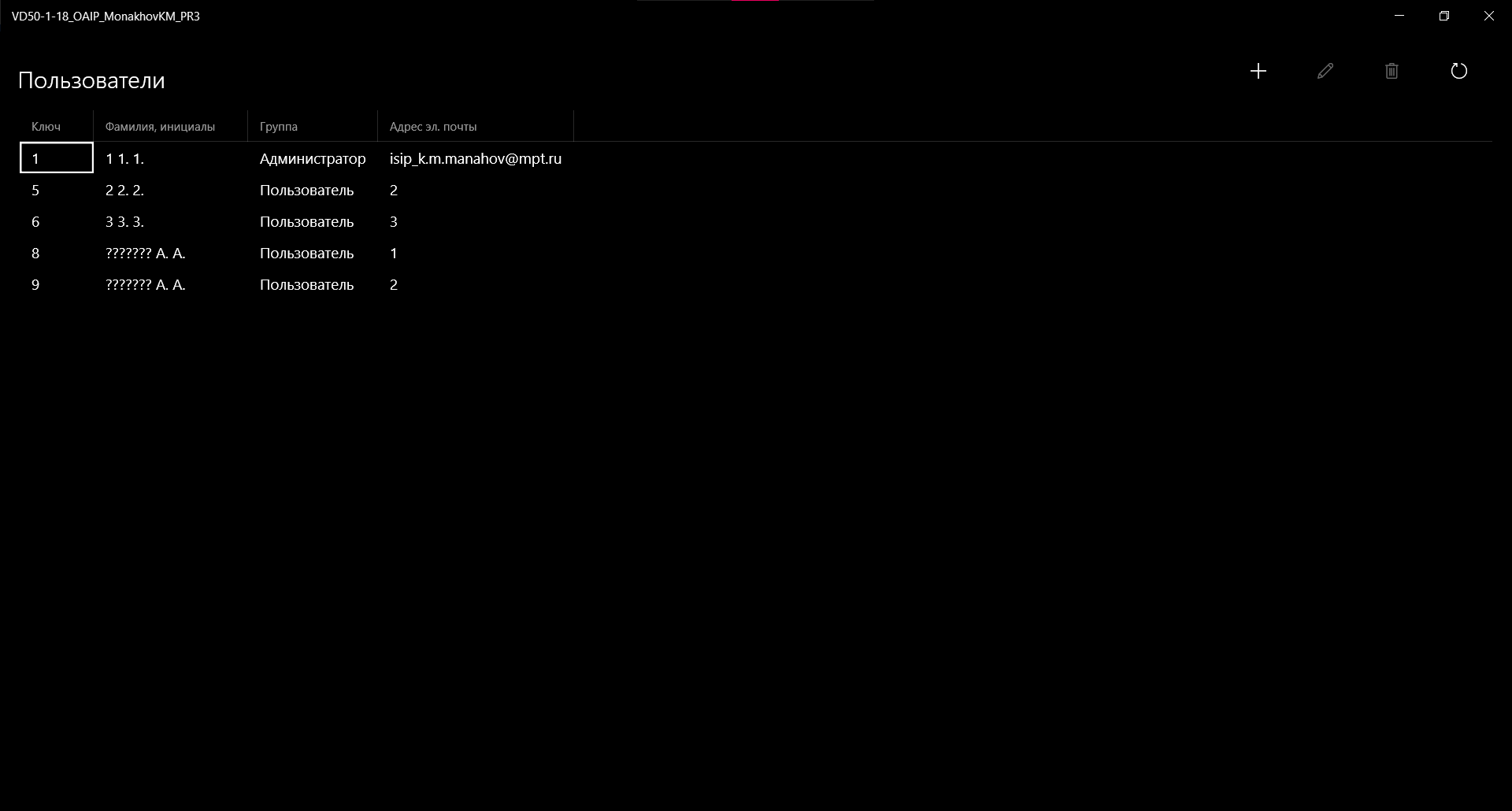


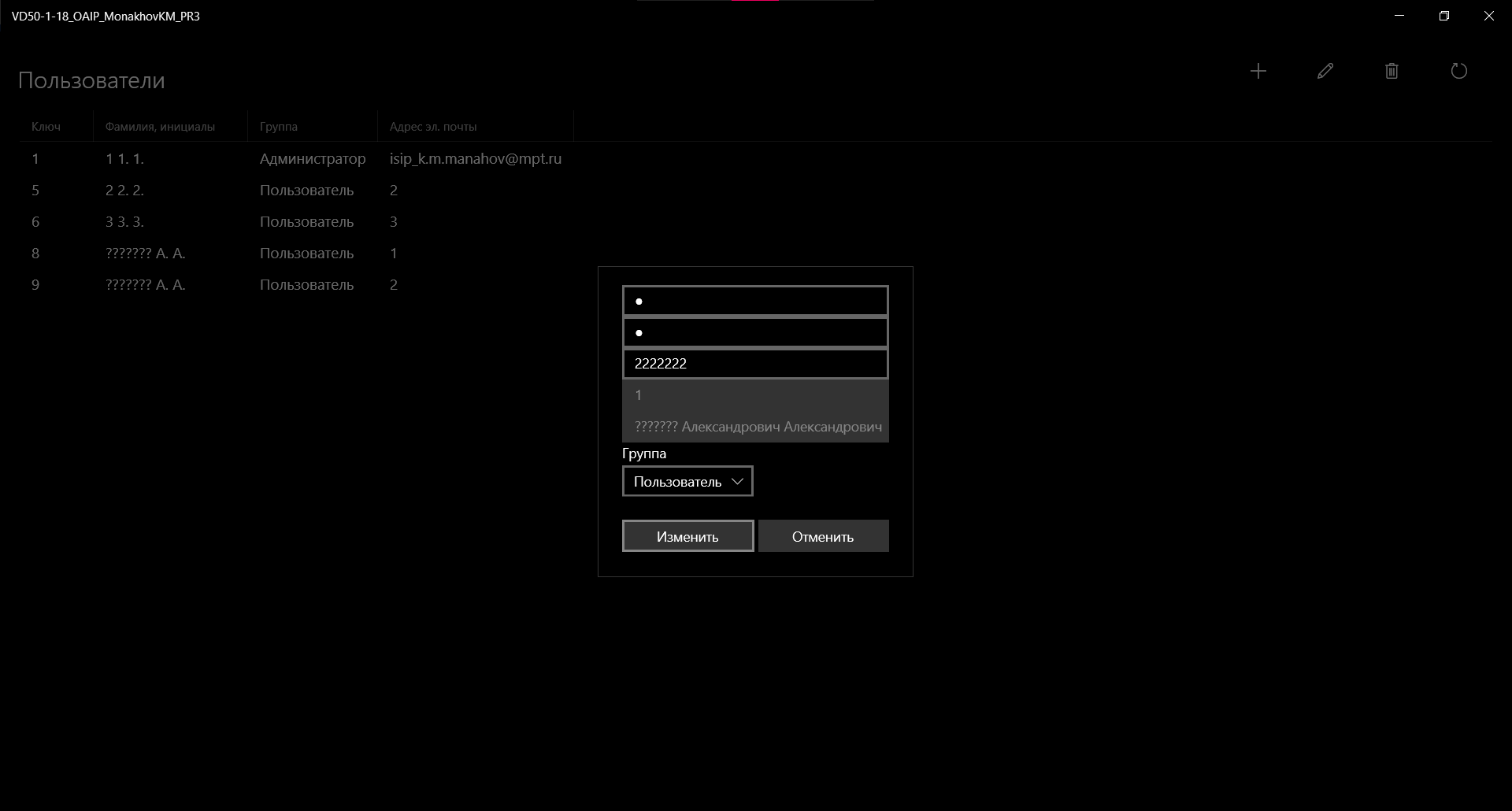
Рисунок 24 – результат изменения и появления диалогового окна.

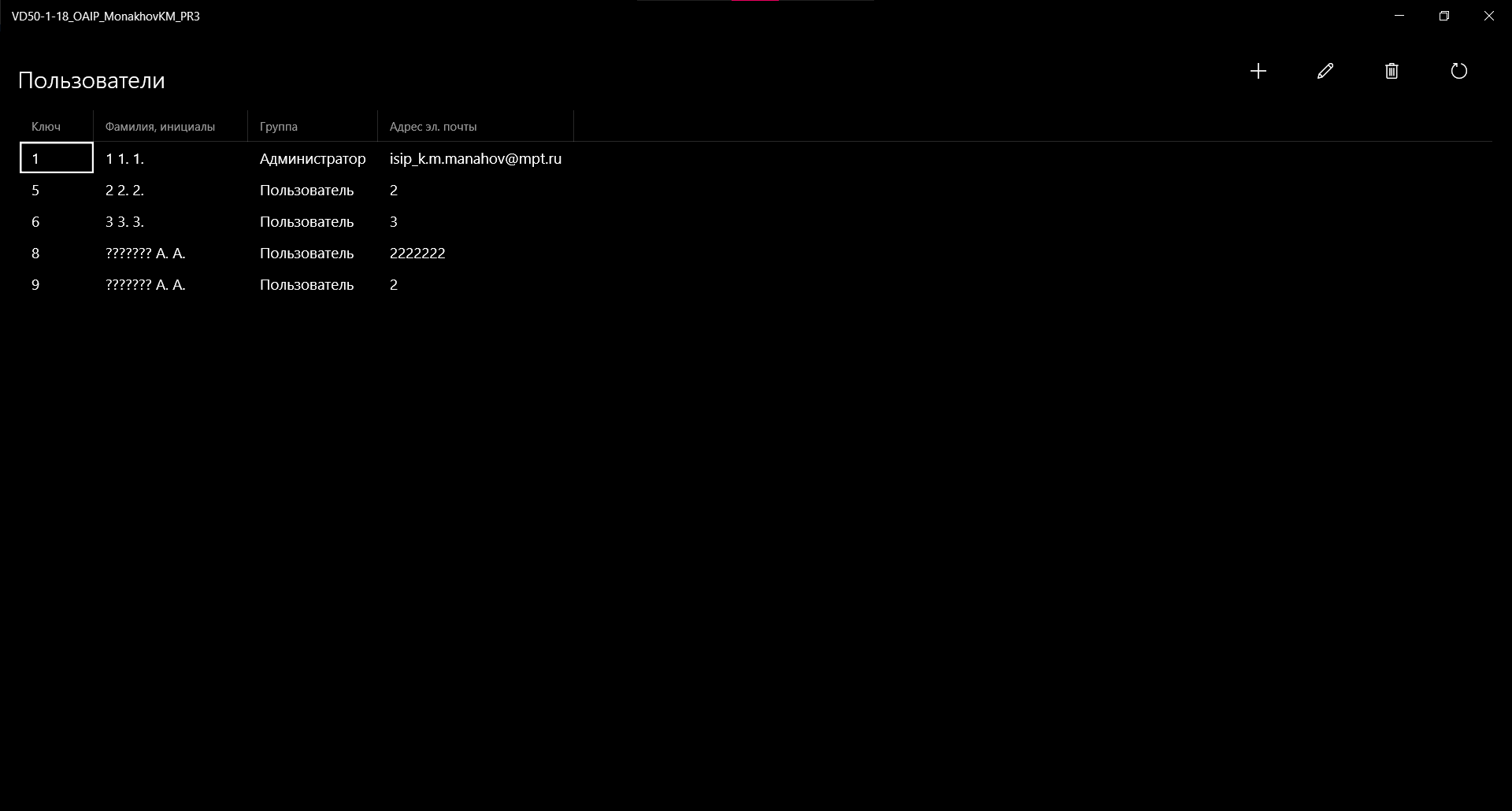
1. Пункт

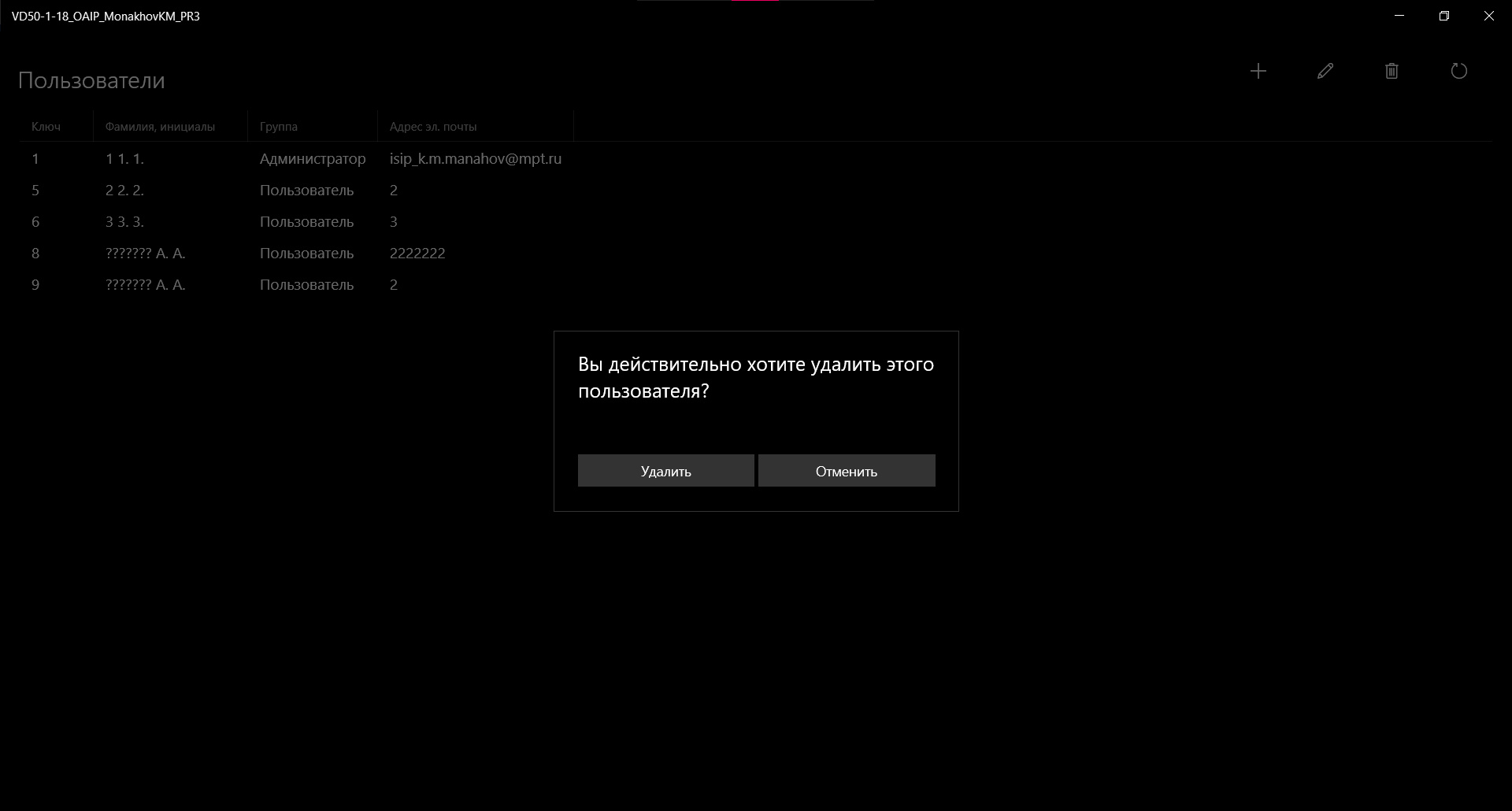
Каждое действие администратора должно логироваться. После реализации базового функционала произвести по одному действию из каждой категории:

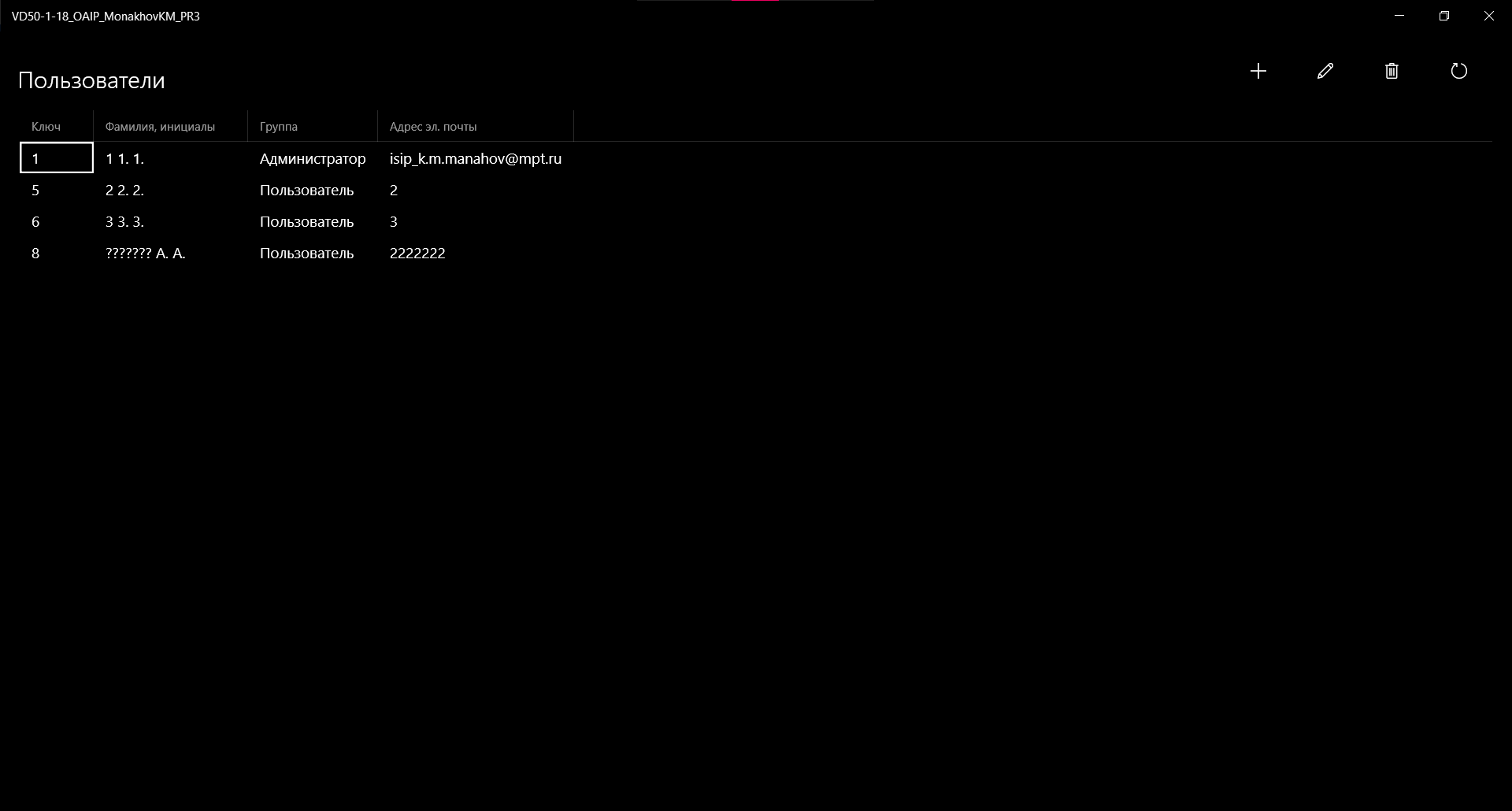












Код программы

|  |  |
| --- | --- |
| Autentification.xaml | <Page  x:Class="VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Assets.Autentification"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:local="using:VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Assets"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  mc:Ignorable="d"  Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">  <Grid>  <Grid Padding="20" Width="500px" Height="500px" BorderBrush="Black" BorderThickness="7">  <Grid.RowDefinitions>  <RowDefinition ></RowDefinition>  <RowDefinition></RowDefinition>  <RowDefinition></RowDefinition>  <RowDefinition></RowDefinition>  <RowDefinition></RowDefinition>  </Grid.RowDefinitions>  <TextBlock Text="Видеопрокат" Grid.Row="0" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="36px"></TextBlock>  <TextBlock Text="Авторизация" Grid.Row="1" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="32px"></TextBlock>  <TextBox x:Name="LoginBox" Grid.Row="2" PlaceholderText="Логин или электронна почта" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="28px" Width="400px"></TextBox>  <PasswordBox x:Name="PasswordBox" Grid.Row="3" PlaceholderText="Пароль" FontSize="28px" HorizontalAlignment="Center" VerticalAlignment="Center" Width="400px"></PasswordBox>  <Button x:Name="LoginButton" Grid.Row="4" Content="Войти" Click="LoginButton\_Click" Background="#bdbebd" FontSize="32px" HorizontalAlignment="Center" VerticalAlignment="Center" Width="500px"></Button>  </Grid>  </Grid>  </Page> |
| Autentificationю.cs | using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Runtime.InteropServices.WindowsRuntime;  using Windows.Foundation;  using Windows.Foundation.Collections;  using Windows.UI.Xaml;  using Windows.UI.Xaml.Controls;  using Windows.UI.Xaml.Controls.Primitives;  using Windows.UI.Xaml.Data;  using Windows.UI.Xaml.Input;  using Windows.UI.Xaml.Media;  using Windows.UI.Xaml.Navigation;  // Документацию по шаблону элемента "Пустая страница" см. по адресу https://go.microsoft.com/fwlink/?LinkId=234238  namespace VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Assets  {  /// <summary>  /// Пустая страница, которую можно использовать саму по себе или для перехода внутри фрейма.  /// </summary>  public sealed partial class Autentification : Page  {  public Autentification()  {  this.InitializeComponent();  }  public class Data  {  public string Email { get; set; }  public int Group { get; set; }  public bool Status { get; set; }  public Data(string email, int group, bool status)  {  Email = email;  Group = group;  Status = status;  }  }  private async void LoginButton\_Click(object sender, RoutedEventArgs e)  {  User user = new User();  config config = new config();  Activated();  try  {  if (!user.Authentication(LoginBox.Text, PasswordBox.Password.ToString()))  {  Frame contentFrame = Window.Current.Content as Frame; // Обращение к родительскому экрану  MainPage mp = contentFrame.Content as MainPage; // Изменение состояния родительского экрана  Frame.Navigate(typeof(PanelFrame));  }  else  {  await config.Alert("Проверьте правильность введённых данных и повторите попытку.", "Ошибка");  }  }  catch (Exception ex)  {  await config.Alert("Произошла ошибка с текстом исключения «" + ex.Message + "». Повторите попытку позднее или обратитесь к системному администратору.", "Ошибка " + ex.Source);  }  }  private object TryAuth()  {  var data\_base = new Database();  var data\_table = data\_base.Select("SELECT \* FROM `users` WHERE " +  "(`users`.`user\_login` = '" + LoginBox.Text + "' OR " +  "`users`.`user\_email` = '" + LoginBox.Text + "') AND BINARY " +  "`users`.`user\_password` = '" + PasswordBox.Password.ToString() + "'");  bool row\_count = Convert.ToBoolean(data\_table.Rows.Count);  string user\_email;  int id\_group;  if (row\_count)  {  user\_email = data\_table.Rows[0]["user\_email"].ToString();  id\_group = Convert.ToInt32(data\_table.Rows[0]["id\_group"]);  }  else  {  user\_email = null; id\_group = 0;  }  return new Data(user\_email, id\_group, row\_count);  }  private async void Activated()  {  var config = new config();  try  {  var AuthResult = TryAuth();  bool AuthStatus = Convert.ToBoolean(config.GetValue(AuthResult, "Status"));  if (AuthStatus)  {  string UserGroup = config.GetValue(AuthResult, "Group");  string UserEmail = config.GetValue(AuthResult, "Email");  await config.Alert("Вы находитесь в группе: " + UserGroup + Environment.NewLine  + "Ваша электронная почта: " + UserEmail, "Успешно");  }  else  {  await config.Alert("Проверьте правильность введённых данных", "Ошибка");  }  }  catch (Exception ex)  {  await config.Alert(ex.ToString(), "Ошибка");  }  }  private void LoginBox\_KeyDown(object sender, KeyRoutedEventArgs e)  {  if (LoginBox.Text != "" && PasswordBox.Password != "")  {  LoginButton.IsEnabled = true;  if (e.Key == Windows.System.VirtualKey.Enter)  {  Activated();  }  }  }  private void PasswordBox\_KeyDown(object sender, KeyRoutedEventArgs e)  {  if (LoginBox.Text != "" && PasswordBox.Password != "")  {  LoginButton.IsEnabled = true;  if (e.Key == Windows.System.VirtualKey.Enter)  {  Activated();  }  }  }  }  } |
| Config.cs | using MySql.Data.MySqlClient;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Security.Cryptography;  using Windows.UI.Popups;  namespace VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Assets  {  class config  {  // Переменная для создания нового экземпляра класса MySqlConnection и описание подключения к серверу базы данных  public MySqlConnection connection = new MySqlConnection("server=127.0.0.1; user=root; password=; database=movedb; port=3306");  // Объявление переменной для дальнейшей работы с кэшем данных  public Windows.Storage.ApplicationDataContainer localSettings = Windows.Storage.ApplicationData.Current.LocalSettings;  // Объявление переменной для хранения ключа сервера для шифрования передаваемых паролей  public string salt = "ТРЕБУЕТСЯ\_ЗАДАТЬ\_САМОСТОЯТЕЛЬНО";  // Создание хэша из строки  public string Hash(string input)  {  // Использование стандартных криптографических функций для имитации алгоритм хеширования MD5  byte[] asciiBytes = ASCIIEncoding.ASCII.GetBytes(input + salt);  byte[] hashedBytes = MD5CryptoServiceProvider.Create().ComputeHash(asciiBytes);  // Сохранение и преобразование результата работы в переменную  string hashedString =  BitConverter.ToString(hashedBytes).Replace("-", "").ToLower();  // Возвращение строки  return hashedString;  }  // Занесение действий в журнал  public void Log(int user, int action)  {  Database data\_base = new Database();  string sql = "INSERT INTO `logs` (`id\_user`, `id\_action`) VALUES ('" + user + "', '" + action + "')";  data\_base.Insert(sql);  }  // Генерация ключа безопасности  public string GenerateToken(int user)  {  Database data\_base = new Database();  // Объявление экземпляра класса Random для создания случайной строки(будущего ключа безопасности)  Random random = new Random();  const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";  string token = new string(Enumerable.Repeat(chars, 64).Select(s  => s[random.Next(s.Length)]).ToArray());  string sql = "INSERT INTO `tokens` (`id\_user`, `token`) VALUES ("  + user + ", '" + token + "')";  data\_base.Insert(sql);  // Вызов метода занесения действий в журнал с передачей идентификатора пользователя и первого типа действий в соответствии с базой данных(авторизация)  Log(user, 1);  //Возвращение ключа безопасности для его дальнейшего использования  return token;  }  // Метод для получения значений из объектов по ключу, возвращающий строковый тип данных  public string GetValue(object Object, string Key)  {  string Value = Object.GetType().GetProperty(Key).  GetValue(Object, null).ToString();  return Value;  }  public async System.Threading.Tasks.Task Alert(string Message, string Title)  {  var messageDialog = new MessageDialog(Message, Title);  await messageDialog.ShowAsync();  }  }  } |
| Database.cs | using MySql.Data.MySqlClient;  using System;  using System.Collections.Generic;  using System.Data;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Assets  {  class Database : config  {  // Метод произведения выборки из базы данных (бд), принимающий всебя SQL-запрос  public DataTable Select(string sql)  {  // Открытие подключения к серверу бд  connection.Open();  // Отправка SQL-запроса на сервер бд с информацией о подключении  MySqlCommand command = new MySqlCommand(sql, connection);  // Выполнение команды на сервере  MySqlDataReader reader = command.ExecuteReader();  // Создание таблицы, которая будет принимать в себя полученные данные из выборки  DataTable dt = new DataTable();  // Загрузка в таблицу данных из выборки  dt.Load(reader);  // Закрытие подключения к серверу бд  connection.Close();  // Возвращение таблицы  return dt;  }  public void Insert(string sql)  {  connection.Open();  MySqlCommand command = new MySqlCommand(sql, connection);  // Выполнение команды на сервере без записи в переменную  command.ExecuteReader();  connection.Close();  }  }  } |
| MainPage.xaml | <Page  x:Class="VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.MainPage"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:local="using:VD50\_1\_18\_OAIP\_MonakhovKM\_PR3"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  mc:Ignorable="d"  Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">  <Grid>  <NavigationView x:Name="navView" Content="NavigationView"/>  </Grid>  </Page> |
| MainPage.cs | using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Runtime.InteropServices.WindowsRuntime;  using Windows.Foundation;  using Windows.Foundation.Collections;  using Windows.UI.Xaml;  using Windows.UI.Xaml.Controls;  using Windows.UI.Xaml.Controls.Primitives;  using Windows.UI.Xaml.Data;  using Windows.UI.Xaml.Input;  using Windows.UI.Xaml.Media;  using Windows.UI.Xaml.Navigation;  using Windows.UI.Popups;  using System.Threading.Tasks;  using Nito.AsyncEx;  using VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Assets;  // Документацию по шаблону элемента "Пустая страница" см. по адресу https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x419  namespace VD50\_1\_18\_OAIP\_MonakhovKM\_PR3  {  /// <summary>  /// Пустая страница, которую можно использовать саму по себе или для перехода внутри фрейма.  /// </summary>  public sealed partial class MainPage : Page  {  public MainPage()  {  this.InitializeComponent();  // Асинхронный вызов метода MainAsync AsyncContext.Run(() => MainAsync());  AsyncContext.Run(() => MainAsync());  }  // Приватный асинхронный метод MainAsync для определения  // состояния входа пользователя.Если он был авторизован ранее  // и имеет активный ключ безопасности, то будет незамедлительно  // отправлен к рабочему пространству  private async System.Threading.Tasks.Task MainAsync()  {  // Проверка состояния аутентификации по коду  User user = new User();  switch (user.CheckAuth())  {  // Код 0 возвращает необходимость повторно пройти  //процедуру аутентификации  case 0:  Frame.Navigate(typeof(Autentification));  break;  // Код 1 даёт возможность сразу приступать к работе,  // так как данные пользователя являются актуальными и  //продолжать работу с приложением безопасносо  case 1:  Frame.Navigate(typeof(PanelFrame));  break;  // Любой другой код говорит о необходимости проверить  //подключение к Интернету или же серверу базы данных при  //помощи цикла внутри данного окна. Если нет  //подключения, то метод CheckAuth() будет вечно  //возвращать код 2, что не входит в границы используемой  //конструкции  default:  // Объявление нового сообщения об ошибке  var messageDialog = new MessageDialog("Произошла ошибка при подключении к Интернету.Пожалуйста, повторите попытку.", "Ошибка");  // Добавление в сообщение об ошибке кнопок с  //вместе с передачей процедур по нажатию на них.От  //повторной загрузки экрана до выхода из  //приложения  messageDialog.Commands.Add(new  UICommand("Повторить",  (command) =>  {  Frame.Navigate(typeof(MainPage));  }));  messageDialog.Commands.Add(new  UICommand("Выход", (command) =>  {  Application.Current.Exit();  }));  await messageDialog.ShowAsync();  break;  }  }  }  } |
| PanelFrame.xaml | <Page  x:Class="VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.PanelFrame"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:local="using:VD50\_1\_18\_OAIP\_MonakhovKM\_PR3"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  mc:Ignorable="d"  Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">  <Grid>  <NavigationView x:Name="UserProfileBtn" Content="NavigationView"/>  <Frame x:Name="userMenu"/>  </Grid>  </Page> |
| PanelFrame.cs | using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Runtime.InteropServices.WindowsRuntime;  using VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Assets;  using Windows.Foundation;  using Windows.Foundation.Collections;  using Windows.UI.Xaml;  using Windows.UI.Xaml.Controls;  using Windows.UI.Xaml.Controls.Primitives;  using Windows.UI.Xaml.Data;  using Windows.UI.Xaml.Input;  using Windows.UI.Xaml.Media;  using Windows.UI.Xaml.Navigation;  // Документацию по шаблону элемента "Пустая страница" см. по адресу https://go.microsoft.com/fwlink/?LinkId=234238  namespace VD50\_1\_18\_OAIP\_MonakhovKM\_PR3  {  /// <summary>  /// Пустая страница, которую можно использовать саму по себе или для перехода внутри фрейма.  /// </summary>  public sealed partial class PanelFrame : Page  {  config Config = new config();  public PanelFrame()  {  this.InitializeComponent();  // Объявление переменную для хранения инициалов  //пользователя, обрезание значения «Имя» до одного символа с  //точкой  string initials = " " +  Config.localSettings.Values["first\_name"].ToString().Substring(0, 1) + ".";  // Проверка на пустое значение поля «Отчество». В случае наличия, происходит обрезание значения до одного символа с точкой по аналогии с именем сотрудника  if (Config.localSettings.Values["mid\_name"] != "") initials  += " " +  Config.localSettings.Values["mid\_name"].ToString().Substring  (0, 1) + ".";  // Назначение содержания на элемент  UserProfileBtn.Content = Config.localSettings.Values["last\_name"] + initials;  }  }  } |
| Users.cs | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Assets;  namespace VD50\_1\_18\_OAIP\_MonakhovKM\_PR3  {  class User : config  {  Database data\_base = new Database();  private int ID\_User { get; set; }  private bool Status { get; set; }  private int ID\_Group { get; set; }  private string Login { get; set; }  private string Email { get; set; }  private string LastName { get; set; }  private string FirstName { get; set; }  private string MidName { get; set; }  public bool Authentication(string LoginBox, string PasswordBox)  {  var data\_table = data\_base.Select("SELECT `id\_user`, `id\_group`, `user\_login`, `user\_email`, `user\_password`, `last\_name`, `first\_name`, `mid\_name` FROM `users` JOIN `employee` WHERE(`users`.`user\_login` = '" + LoginBox + "'OR `users`.`user\_email` = '" + LoginBox + "') AND(BINARY `users`.`user\_password` = '" + Hash(PasswordBox) + "') AND(`users`.`id\_user` = `employee`.`id\_employee`)");  if (!Status)  {  ID\_User = Convert.ToInt32(data\_table.Rows[0]["id\_user"]);  ID\_Group = Convert.ToInt32(data\_table.Rows[0]["id\_group"]);  Login = data\_table.Rows[0]["user\_login"].ToString();  Email = data\_table.Rows[0]["user\_email"].ToString();  LastName = data\_table.Rows[0]["last\_name"].ToString();  FirstName = data\_table.Rows[0]["first\_name"].ToString();  MidName = data\_table.Rows[0]["mid\_name"].ToString(); CacheAuth();  }  return Status;  }  private void CacheAuth()  {  string Token = GenerateToken(ID\_User);  localSettings.Values["id\_user"] = ID\_User;  localSettings.Values["status"] = Status;  localSettings.Values["id\_group"] = ID\_Group;  localSettings.Values["user\_login"] = Login;  localSettings.Values["user\_email"] = Email;  localSettings.Values["last\_name"] = LastName;  localSettings.Values["first\_name"] = FirstName;  localSettings.Values["mid\_name"] = MidName;  localSettings.Values["token"] = Token;  }  public int CheckAuth()  {  try  {  Database data\_base = new Database();  string sql = "SELECT `tokens`.`id\_user`, `token`, `last\_name`, `first\_name`, `mid\_name` FROM `tokens` JOIN `users`, `employee` WHERE(`tokens`.`id\_user` ='" + localSettings.Values["id\_user"] + "') AND(`token` = '" + localSettings.Values["token"] + "')AND(`deactivated` = 0) AND(`tokens`.`id\_user` = `users`.`id\_user` AND `users`.`id\_employee` =`employee`.`id\_employee`)";  var data\_table = data\_base.Select(sql);  bool status = Convert.ToBoolean(data\_table.Rows.Count);  if (status && Convert.ToBoolean(localSettings.Values["status"]))  {  if (localSettings.Values["last\_name"] != data\_table.Rows[0]["last\_name"].ToString() || localSettings.Values["first\_name"] != data\_table.Rows[0]["first\_name"].ToString() || localSettings.Values["mid\_name"] != data\_table.Rows[0]["mid\_name"].ToString())  {  localSettings.Values["last\_name"] = data\_table.Rows[0]["last\_name"].ToString(); localSettings.Values["first\_name"] = data\_table.Rows[0]["first\_name"].ToString(); localSettings.Values["mid\_name"] = data\_table.Rows[0]["mid\_name"].ToString();  }  return 1;  }  else  {  return 0;  }  }  catch (Exception)  {  return 2;  }  }  }  } |
| Users.xaml | <Page  x:Class="VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Users"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:local="using:VD50\_1\_18\_OAIP\_MonakhovKM\_PR3"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:controls="using:Microsoft.Toolkit.Uwp.UI.Controls"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  mc:Ignorable="d"  Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">  <Grid Padding="20" LostFocus="Grid\_LostFocus">  <Grid.RowDefinitions>  <RowDefinition Height="60"/>  <RowDefinition Height="\*"/>  <RowDefinition Height="200"/>  </Grid.RowDefinitions>  <CommandBar VerticalContentAlignment="Center" x:Name="CommandBar" IsOpen="True" IsSticky="False" OverflowButtonVisibility="Collapsed" Closing="CommandBar\_Closing" Background="{x:Null}" Grid.Row="0" >  <AppBarButton Icon="Add" Label="Добавить" Name="Add" Click="Add\_Click" VerticalAlignment="Center" VerticalContentAlignment="Center"/>  <AppBarButton Icon="Edit" IsEnabled="False" Label="Изменить" Name="edit" Click="edit\_Click" VerticalAlignment="Center" VerticalContentAlignment="Center"/>  <AppBarButton Icon="Delete" IsEnabled="False" Name="Delete" Click="Delete\_Click" Label="Удалить" VerticalAlignment="Center" VerticalContentAlignment="Center"/>  <AppBarButton Icon="Refresh" Label="Обновить" VerticalAlignment="Center" VerticalContentAlignment="Center" Click="AppBarButton\_Click"/>  <CommandBar.Content>  <TextBlock Style="{StaticResource TitleTextBlockStyle}" Text="Пользователи" VerticalAlignment="Stretch" HorizontalAlignment="Stretch"/>  </CommandBar.Content>  </CommandBar>  <controls:DataGrid Grid.Row="1" x:Name="userList" ItemsSource="{Binding}" AutoGenerateColumns="False" SelectionChanged="userList\_SelectionChanged" />  </Grid>  </Page> |
| Users.xaml.cs | using Microsoft.Toolkit.Uwp.UI.Controls;  using Renci.SshNet.Messages;  using System;  using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Data;  using System.IO;  using System.Linq;  using System.Runtime.InteropServices.WindowsRuntime;  using VD50\_1\_18\_OAIP\_MonakhovKM\_PR3.Assets;  using Windows.Foundation;  using Windows.Foundation.Collections;  using Windows.UI.Popups;  using Windows.UI.Xaml;  using Windows.UI.Xaml.Controls;  using Windows.UI.Xaml.Controls.Primitives;  using Windows.UI.Xaml.Data;  using Windows.UI.Xaml.Input;  using Windows.UI.Xaml.Media;  using Windows.UI.Xaml.Navigation;  // Документацию по шаблону элемента "Пустая страница" см. по адресу https://go.microsoft.com/fwlink/?LinkId=234238  namespace VD50\_1\_18\_OAIP\_MonakhovKM\_PR3  {  /// <summary>  /// Пустая страница, которую можно использовать саму по себе или для перехода внутри фрейма.  /// </summary>  public sealed partial class Users : Page  {  public Users()  {  this.InitializeComponent();  }  private void CommandBar\_Closing(object sender, object e)  {  }  private void AppBarButton\_Click(object sender, RoutedEventArgs e)  {  GetList();  }  private void GetList()  {  Database data\_base = new Database();  string sql = "SELECT \* FROM `users\_list`";  DataTable data\_table = data\_base.Select(sql);  // Запрос на выборку из представления  // Очистка столбцов по умолчанию в таблице с данными  userList.Columns.Clear();  // Цикл, который на основе выборки добавляет необходимые колонки  for (int i = 0; i < data\_table.Columns.Count; i++)  {  userList.Columns.Add(new DataGridTextColumn()  {  // Заголовок задаётся на основе такового в MySQL (AS)  Header = data\_table.Columns[i].ColumnName,  // Преобразование данные  Binding = new Binding  {  Path = new PropertyPath("[" +  i.ToString() + "]")  }  });  }  // Создание коллекции с данными, которые будут принимать в себя  var collection = new ObservableCollection<object>();  // Цикл, который добавляет каждую строку выборку в коллекцию  foreach (DataRow row in data\_table.Rows)  {  collection.Add(row.ItemArray);  }  // Запись значений коллекции в таблицу с данными  userList.ItemsSource = collection;  }  string index;  private async void userList\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  edit.IsEnabled = true;  Delete.IsEnabled = true;  var test1 = userList.SelectedItem as Object[];  if (test1 != null)  {  index = test1[0].ToString();  }  }  private void Grid\_LostFocus(object sender, RoutedEventArgs e)  {  edit.IsEnabled = false;  Delete.IsEnabled = false;  GetList();  }  static bool Prov(PasswordBox pas, TextBox Name, TextBox Email, PasswordBox pas1)  {  if (pas.Password == "" && pas1.Password == "" && Name.Text == "" && Email.Text == "")  {  return false;  }  else  {  return true;  }  }  //Добавление пользователя  private async void Add\_Click(object sender, RoutedEventArgs e)  {  config config = new config();  StackPanel panel = new StackPanel();  Database data\_base = new Database();  var collection = new ObservableCollection<object>();  PasswordBox password = new PasswordBox();  TextBox Name = new TextBox();  PasswordBox password1 = new PasswordBox();  TextBox Email = new TextBox();  ContentDialog contentDialog = new ContentDialog();  Name.PlaceholderText = "Имя пользователя";  Name.HorizontalAlignment = HorizontalAlignment.Stretch;  Name.SelectionChanged += (send, args) =>  {  contentDialog.IsPrimaryButtonEnabled = Prov(password, Name, Email, password1);  };  password.PlaceholderText = "Пароль";  password.HorizontalAlignment = HorizontalAlignment.Stretch;  password.PasswordChanged += (send, args) =>  {  contentDialog.IsPrimaryButtonEnabled = Prov(password, Name, Email, password1);  };  password1.PlaceholderText = "подтверждения пароля";  password1.HorizontalAlignment = HorizontalAlignment.Stretch;  password1.PasswordChanged += (send, args) =>  {  contentDialog.IsPrimaryButtonEnabled = Prov(password, Name, Email, password1);  };  Email.PlaceholderText = "Почта";  Email.HorizontalAlignment = HorizontalAlignment.Stretch;  Email.SelectionChanged += (send, args) =>  {  contentDialog.IsPrimaryButtonEnabled = Prov(password, Name, Email, password1);  };  ComboBox sotrudnik = new ComboBox();  DataTable data\_table = data\_base.Select("SELECT CONCAT(Imya\_sotrydnika,' ',Familiya\_sotrydnika,' ',Otchestvo\_sotrydnika) FROM `sotrydnik` WHERE 1");  collection = new ObservableCollection<object>();  foreach (DataRow row in data\_table.Rows)  {  collection.Add(row.ItemArray[0]);  }  sotrudnik.ItemsSource = collection;  ComboBox group = new ComboBox();  data\_table = data\_base.Select("SELECT group\_name FROM `groups` WHERE 1");  collection = new ObservableCollection<object>();  foreach (DataRow row in data\_table.Rows)  {  collection.Add(row.ItemArray[0]);  }  group.ItemsSource = collection;  panel.Children.Add(Name);  panel.Children.Add(Email);  panel.Children.Add(password);  panel.Children.Add(password1);  panel.Children.Add(sotrudnik);  panel.Children.Add(group);  contentDialog.Content = panel;  contentDialog.CloseButtonText = "Отмена";  contentDialog.PrimaryButtonText = "Сохранить";  contentDialog.IsPrimaryButtonEnabled = false;  contentDialog.PrimaryButtonClick += async (send,arg) =>  {  if(password1.Password != password.Password)  {  await config.Alert("Пароль не совподает", "Ошибка");  }  if(data\_base.Select("SELECT \* FROM `users` WHERE users.user\_login = '" + Name.Text.Trim() + "'").Rows.Count == 0)  {  await config.Alert("Логин существует", "ошибка");  }  string[] FIO\_DataBase = sotrudnik.SelectedItem.ToString().Split(' ');  data\_base.Insert("INSERT INTO `users` (`id\_user`, `id\_group`, `id\_employee`, `last\_name`, `first\_name`, `mid\_name`, `user\_login`, `user\_email`, `user\_password`) VALUES(NULL, '" + (group.SelectedIndex + 1) + "', '" + (sotrudnik.SelectedIndex + 1) + "', '" + FIO\_DataBase[1] + "', '" + FIO\_DataBase[2] + "', '" + FIO\_DataBase[2] + "', '" + Name.Text.Trim() + "', '" + Email.Text.Trim() + "', '" + config.Hash(password.Password) + "')");  };  await contentDialog.ShowAsync();  }  private async void Delete\_Click(object sender, RoutedEventArgs e)  {  var dialog = new ContentDialog();  dialog.Title = "Вы действительно хотите удалить этого пользователя?";  dialog.PrimaryButtonText = "Удалить";  dialog.PrimaryButtonClick += (a, b) =>  {  Database database = new Database();  string sql = "DELETE FROM `users` WHERE users.id\_user = '" + index + "'; UPDATE `tokens` SET `deactivated` = '1' WHERE `tokens`.`id\_user` = '" + index + "'";  database.Select(sql);  GetList();  };  dialog.CloseButtonText = "Отменить";  await dialog.ShowAsync();  }  private async void edit\_Click(object sender, RoutedEventArgs e)  {  StackPanel panel = new StackPanel();  PasswordBox password = new PasswordBox();  password.PlaceholderText = "Пароль";  password.HorizontalAlignment = HorizontalAlignment.Stretch;  PasswordBox password1 = new PasswordBox();  password1.PlaceholderText = "Подтверждения пароля";  password1.HorizontalAlignment = HorizontalAlignment.Stretch;  TextBox Email = new TextBox();  Email.PlaceholderText = "Почта";  Email.HorizontalAlignment = HorizontalAlignment.Stretch;  TextBox Name = new TextBox();  Name.IsEnabled = false;  Database name = new Database();  var tablename = name.Select("SELECT user\_login FROM `users` WHERE id\_user = " + index);  Name.Text = tablename.Rows[0].ItemArray[0].ToString();  Name.HorizontalAlignment = HorizontalAlignment.Stretch;  TextBox date\_name = new TextBox();  date\_name.IsEnabled = false;  Database FIO\_DataBase = new Database();  var tableFIO\_DataBase = FIO\_DataBase.Select("SELECT CONCAT(last\_name,' ',first\_name,' ',mid\_name) FROM `users` WHERE id\_user = " + index);  date\_name.Text = tableFIO\_DataBase.Rows[0].ItemArray[0].ToString();  date\_name.HorizontalAlignment = HorizontalAlignment.Stretch;  ComboBox namegroup = new ComboBox();  namegroup.Header = "Группа";  namegroup.SelectedIndex = 0;  Database goups = new Database();  var tablegroups = goups.Select("SELECT CONCAT(groups.group\_name) FROM `groups` WHERE 1");  var collection = new ObservableCollection<object>();  foreach (DataRow row in tablegroups.Rows)  {  collection.Add(row.ItemArray[0]);  }  namegroup.ItemsSource = collection;  panel.Children.Add(password);  panel.Children.Add(password1);  panel.Children.Add(Email);  panel.Children.Add(Name);  panel.Children.Add(date\_name);  panel.Children.Add(namegroup);  var dialog = new ContentDialog();  dialog.Content = panel;  dialog.PrimaryButtonText = "Изменить";  dialog.PrimaryButtonClick += async (a, b) =>  {  if (password.Password != password1.Password)  {  MessageDialog message = new MessageDialog("Пароли не совпадают");  await message.ShowAsync();  return;  }  if (Email.Text == "")  {  MessageDialog message = new MessageDialog("Поле с почтой пустое");  await message.ShowAsync();  return;  }  Database database = new Database();  config config = new config();  string sql = "UPDATE `users` SET id\_group = '" + (namegroup.SelectedIndex + 1) + "',`user\_email`= '" + Email.Text + "',`user\_password`= '" + config.Hash(password.Password) + "' WHERE id\_user = " + index;  database.Select(sql);  GetList();  };  dialog.CloseButtonText = "Отменить";  await dialog.ShowAsync();  }  }  } |

Вывод: я произвел модификацию базы данных на виртуальном веб-сервере с целью обеспечения безопасной аутентификации в программном обеспечении для дальнейшей работы внутри информационной системы в клиентском приложении на универсальной платформе Windows. Реализовал механизм для аутентификации в приложении на универсальной платформе Windows и научился делать реализацию проекта