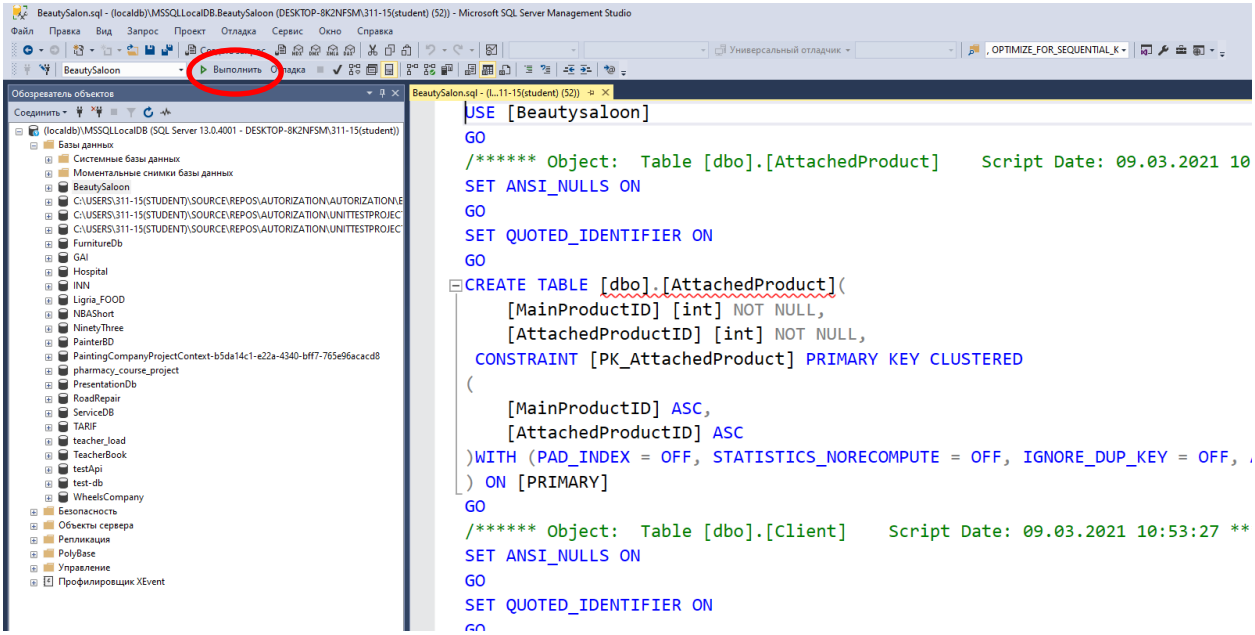
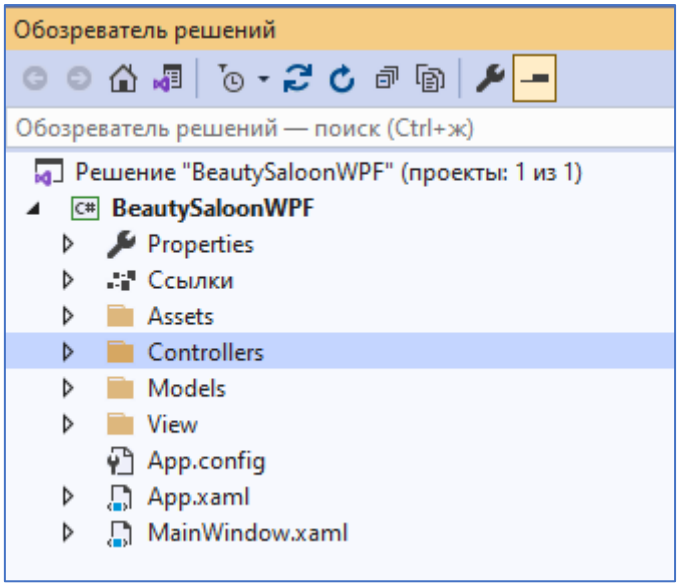
## 1. Восстановление БД из скрипта

Для восстановления БД необходимо создать базу данных с указанным именем и нажать «Выполнить скрипт»



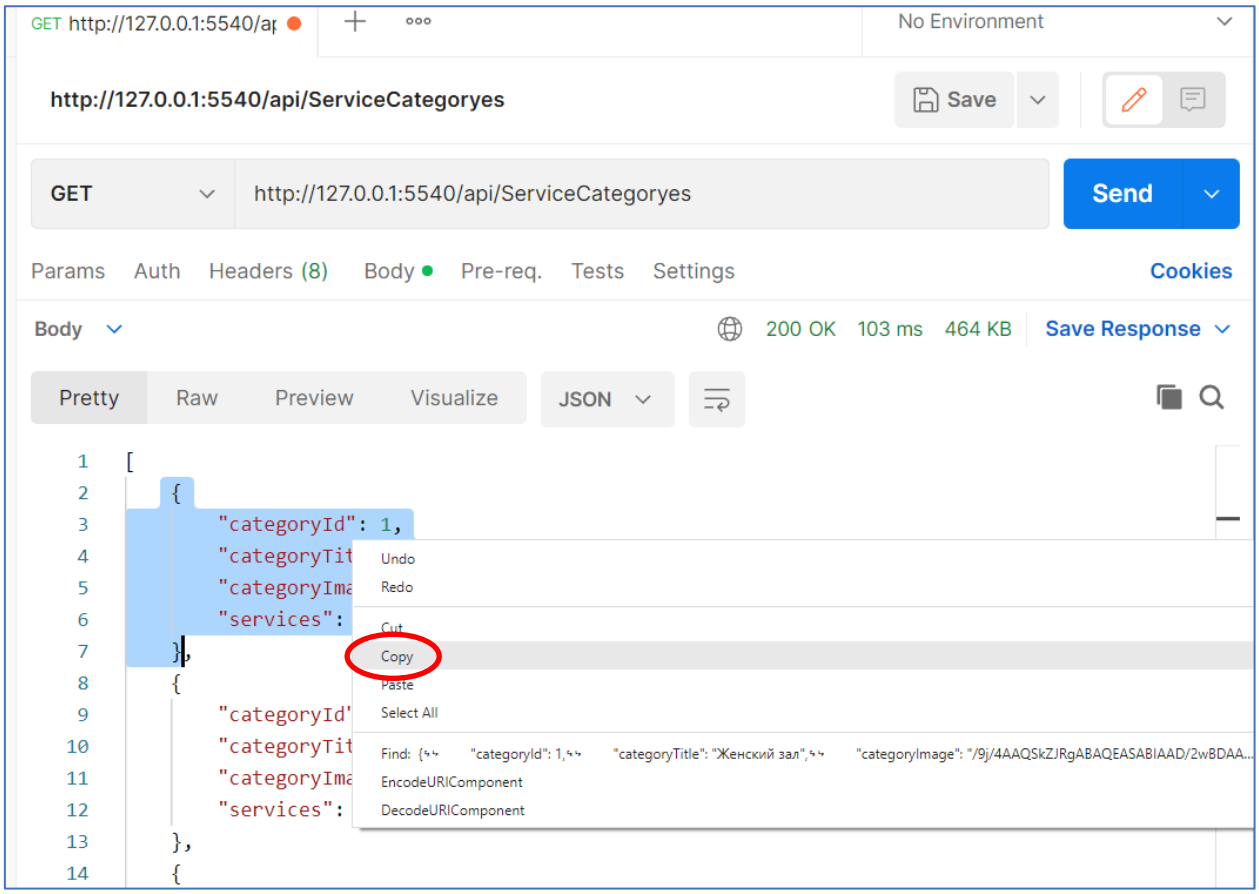

## 2. Создание структуры проекта



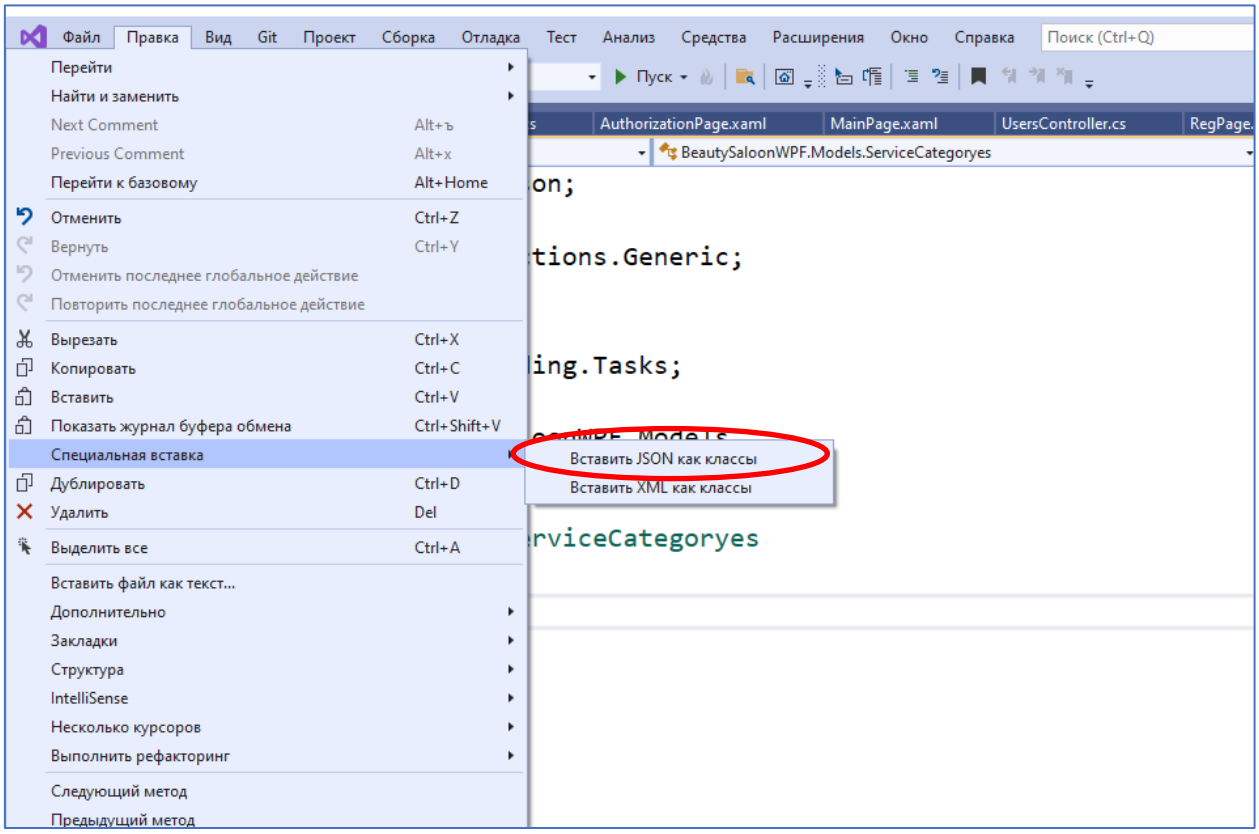

## 3. Создание модели проекта

Порядок выполнения:

1. Для **КАЖДОЙ** таблицы базы данных необходимо создать класс в папке Models
2. Протестировать запрос на вывод данных из таблицы в Postman
3. Скопировать один json-объект




4. В созданном классе воспользоваться «Специальной вставкой» для формирования свойств класса



```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BeautySaloonWPF.Models
{
    Ссылок: 2
    public class ServiceCategoryes
    {
        Ссылок: 0
        public int categoryId { get; set; }
        Ссылок: 0
        public string categoryTitle { get; set; }
        Ссылок: 0
        public byte[] categoryImage { get; set; }
    }
}
```

5. Выполните переименование свойств класса (задайте имена в стиле «CamelCase»)

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BeautySaloonWPF.Models
{
    Ссылок: 2
    public class ServiceCategoryes
    {
        [JsonProperty("categoryId")]
        Ссылок: 0
        public int CategoryId { get; set; }
        [JsonProperty("categoryTitle")]
        Ссылок: 0
        public string CategoryTitle { get; set; }
        [JsonProperty("categoryImage")]
        Ссылок: 0
        public byte[] CategoryImage { get; set; }
    }
}
```

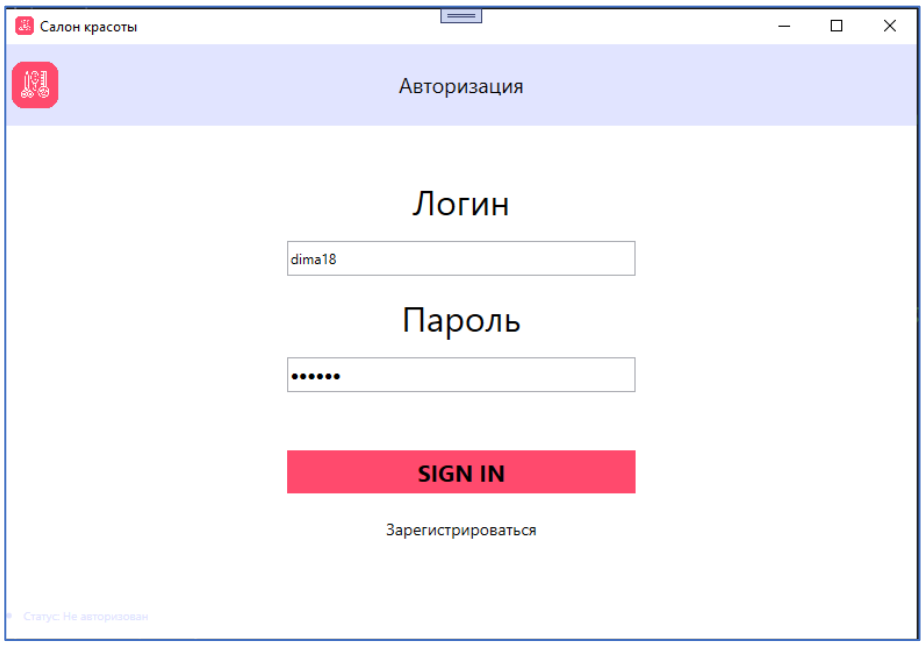
**ВЫПОЛНИТЕ СОЗДАНИЕ КЛАССОВ ДЛЯ КАЖДОЙ ТАБЛИЦЫ БАЗЫ ДАННЫХ !!!!!!!!!!!!!!!!!!!!**

## 4. Создание главного окна приложения

Выполним верстку главного окна приложения

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="70" />
        <RowDefinition/>
        <RowDefinition Height="40"/>
    </Grid.RowDefinitions>
    <Image Source="" Margin="5"/>
    <TextBlock Text="{Binding ElementName=MainFrame, Path=Content.Title}"/>
    <StackPanel Orientation="Horizontal" HorizontalAlignment="Right" >
        <TextBlock x:Name="FullnameTextBlock" ></TextBlock>
    </StackPanel>
    <Frame Grid.Row="1" NavigationUIVisibility="Hidden" x:Name="MainFrame"
      ContentRendered="MainFrame_ContentRendered" Navigated="MainFrame_Navigated" />
    <Button x:Name="BackButton" Grid.Row="2" HorizontalAlignment="Right" Click="Back_Button_Click"
      Opacity="0.8"></Button>
</Grid>
```

Создадим форму авторизации




Реализуем загрузку окна авторизации во фрейм Главного окна приложения

```
Ссылок: 5
public partial class MainWindow : Window
{
    Ссылок: 0
    public MainWindow()
    {
        InitializeComponent();
        MainFrame.Navigate(new AuthorizationPage());
    }
    ссылка: 1
    private void Back_Button_Click(object sender, RoutedEventArgs e)
    {
        Page currentPage = MainFrame.Content as Page;
        if (MainFrame.CanGoBack)
        {
            MainFrame.GoBack();
        }
    }
```

## 5. Создание контроллеров

Создадим класс Manager в папке Models

```
Ссылок: 3
public class Manager
{
    public static string RootUrl = "http://127.0.0.1:5540/api/";
}
```

Взаимодействие с базой данных из интерфейсных форм должно реализовываться через контроллеры

### Авторизация GET — http://127.0.0.1:5540/api/users/{логин}/{пароль}

```
public static class UsersController
{
/// <summary>
/// Авторизация
/// </summary>
/// <param name="login">Логин</param>
/// <param name="password">Пароль</param>
/// <returns>
/// Статус ответа
/// </returns>
ссылка: 1
public static bool Auth(string login,string password)
{
    using (HttpClient client = new HttpClient())
    {
        HttpResponseMessage response = client.GetAsync($"{Manager.RootUrl}Users/{login}/{password}").Result;
        return response.IsSuccessStatusCode;
    }
}
```

### Регистрация POST — http://127.0.0.1:5540/api/users

```
/// <summary>
/// Регистрация
/// </summary>
/// <param name="login"></param>
/// <param name="password"></param>
/// <returns></returns>
ссылка: 1
public static bool AddUser(Users user)
{
    string jsonStr = JsonConvert.SerializeObject(user);
    var buffer = System.Text.Encoding.UTF8.GetBytes(jsonStr);
    var byteContent = new ByteArrayContent(buffer);
    byteContent.Headers.ContentType = new MediaTypeHeaderValue("application/json");
    using (HttpClient client = new HttpClient())
    {
    HttpResponseMessage response = client.PostAsync($"{Manager.RootUrl}Users", byteContent).Result;
    return response.IsSuccessStatusCode;
    }
}
```

## 6. Обращение к контроллерам из интерфейсных форм

### Авторизация (Вызов метода UsersController.Auth)

```
<Grid>
    <StackPanel>
        <TextBlock>Логин</TextBlock>
        <TextBox x:Name="LoginTextBox" TabIndex="0" />
        <TextBlock>Пароль</TextBlock>
        <PasswordBox x:Name="PasswordPasswordBox" TabIndex="1" ></PasswordBox>
        <Button x:Name="SignInbutton" TabIndex="2" Click="SignInbuttonClick" Content="Авторизация" />
        <Button x:Name="RegButton" TabIndex="3" IsTabStop="True"  Content="Регистрация"
          Click="RegButton_Click"></Button>
    </StackPanel>
</Grid>
```

```
public partial class AuthorizationPage : Page
{
    Ссылок: 2
    public AuthorizationPage()
    {
        InitializeComponent();
    }
    ссылка: 1
    private void SignInbuttonClick(object sender, RoutedEventArgs e)
    {
        if (UsersController.Auth(LoginTextBox.Text,PasswordPasswordBox.Password))
        {
            this.NavigationService.Navigate(new AuthorizationPage());
        }
        else
        {
            this.NavigationService.Navigate(new RegPage());
        }
    }
    ссылка: 1
    private void RegButton_Click(object sender, RoutedEventArgs e)
    {
        this.NavigationService.Navigate(new RegPage());
    }
}
```

## Регистрация (Вызов метода UsersController.Auth)

```
<Grid>
    <DockPanel HorizontalAlignment="Center" VerticalAlignment="Center">
        <StackPanel Width="200" VerticalAlignment="Center">
            <TextBlock >Регистрация</TextBlock>
            <TextBlock>Имя:</TextBlock>
            <TextBox x:Name="NameTextBox"></TextBox>
            <TextBlock>Фамилия:</TextBlock>
            <TextBox x:Name="LastNameTextBox"></TextBox>
            <TextBlock>Отчество:</TextBlock>
            <TextBox x:Name="OtherNamaTextBox"></TextBox>
            <TextBlock>Логин:</TextBlock>
            <TextBox x:Name="LoginTextBox"></TextBox>
            <TextBlock>Пароль:</TextBlock>
            <PasswordBox x:Name="PasswordPasswordBox"></PasswordBox>
            <TextBlock>Повторите пароль:</TextBlock>
            <PasswordBox x:Name="RepeatPasswordPasswordBox"></PasswordBox>
            <Button x:Name="RegistrationButton" Click="RegistrationButton_Click"
              >Зарегистрироваться</Button>
        </StackPanel>
        <StackPanel >
            <TextBlock>Введите символы с картинки:</TextBlock>
            <StackPanel x:Name="CapchaStackPanel" Orientation="Horizontal"
              HorizontalAlignment="Center"></StackPanel>
            <Button x:Name="CaptchaButton" >Обновить Captcha</Button>
        </StackPanel>
    </DockPanel>
</Grid>
```

```
public partial class RegPage : Page
{
    Ссылок: 2
    public RegPage()
    {
        InitializeComponent();
    }
    ссылка: 1
    private void RegistrationButton_Click(object sender, RoutedEventArgs e)
    {
        Users newUser = new Users
        {
            IdRole = 2,
            UserName = NameTextBox.Text,
            UserLastName = LastNameTextBox.Text,
            UserOtherName = OtherNamaTextBox.Text,
            UserLogin = LoginTextBox.Text,
            UserPassword = PasswordPasswordBox.Password
        };

        if (UsersController.AddUser(newUser))
        {
            MessageBox.Show("запись добавлена");
        }
    }
}
```