

Используемые среды: MS SQL Server Management Studio
Microsoft Visual Studio

Создание БД

-
- Microsoft SQL Server Enterprise Manager
- Файл Правка Вид Проект Сервис Окно Справка
- Обозреватель объектов
- Соединить
- DESKTOP-GOTSOF1 (SQL Server 12.0.2000 - D
- Базы данных
- Системные базы данных
- Безопасность
- Объекты сервера
- Репликация
- Управление
- Профилирование XEvent
- Создание базы данных
- Выбор страницы
- Общие
- Параметры
- Файловые группы
- Скрипт
- Справка
- Имя базы данных: AcademyWor
- Владелец: <по умолчанию>
- ☒ Использовать полнотекстовое индексирование
- Файлы базы данных:
- | Логическое имя | Тип файла | Файловая группа | Начальный размер (MB) | Автоматическое |
|----------------|-----------|-----------------|-----------------------|----------------|
| AcademyWor | Данные | PRIMARY | 5 | С шагом по |
| AcademyWor_log | ЖУРНАЛ | Непрерывно | 1 | С шагом по |
- Соединение
- Сервер: DESKTOP-GOTSOF1
- Соединение: DESKTOP-GOTSOF1\bwf
- [Показать свойства соединения](#)
- Ход выполнения
- Готово
- Академия
- Диаграммы БД
- Таблицы
- Представления
- Синонимы
- Программируемые объекты
- Компонент Сервера
- Хранилище
- Безопасность

-
- DESKTOP-GOTSOFL...yWur - Diagram_0* ✕
- Role ***
- | Имя столбца | Тип данных | Разрешить ... |
|-------------|--------------|--------------------------|
| Id | int | <input type="checkbox"/> |
| Name | nvarchar(50) | <input type="checkbox"/> |
| | | <input type="checkbox"/> |
- User ***
- | Имя столбца | Тип данных | Разрешить ... |
|-------------|--------------|--------------------------|
| Id | int | <input type="checkbox"/> |
| Login | nvarchar(50) | <input type="checkbox"/> |
| Password | nvarchar(50) | <input type="checkbox"/> |
| IdRole | int | <input type="checkbox"/> |
| Name | nvarchar(50) | <input type="checkbox"/> |
| | | <input type="checkbox"/> |

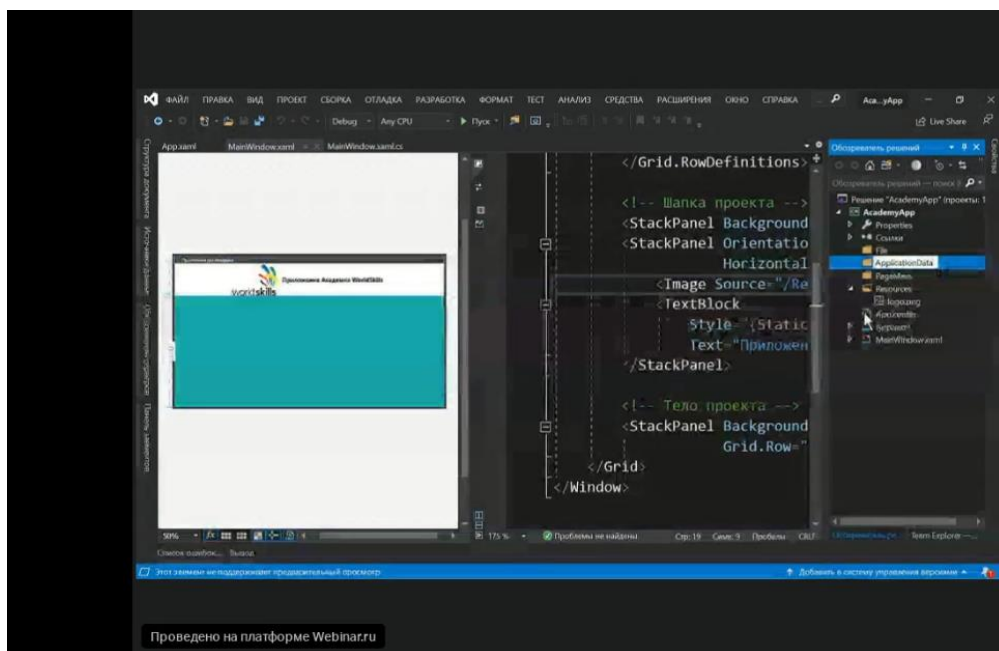
- Заполните таблицы данными. Предполагаем, что у нас будет два пользователя, один из них Администратор (Степан), другой – Ученик (Наталья). Других пользователей будем добавлять через приложение.

Id	Name
1	Администратор
2	Ученик
NULL	NULL

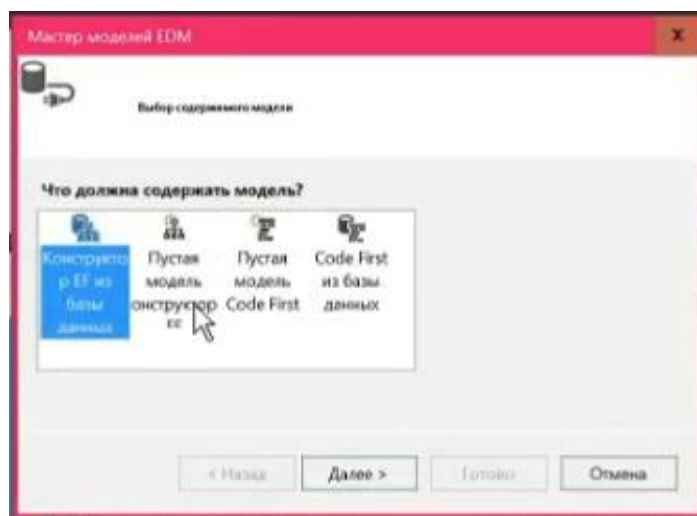
Id	Login	Password	IdRole	Name
1	1	1	1	Степан
2	2	2	2	Наталья
NULL	NULL	NULL	NULL	NULL

Создание модели БД

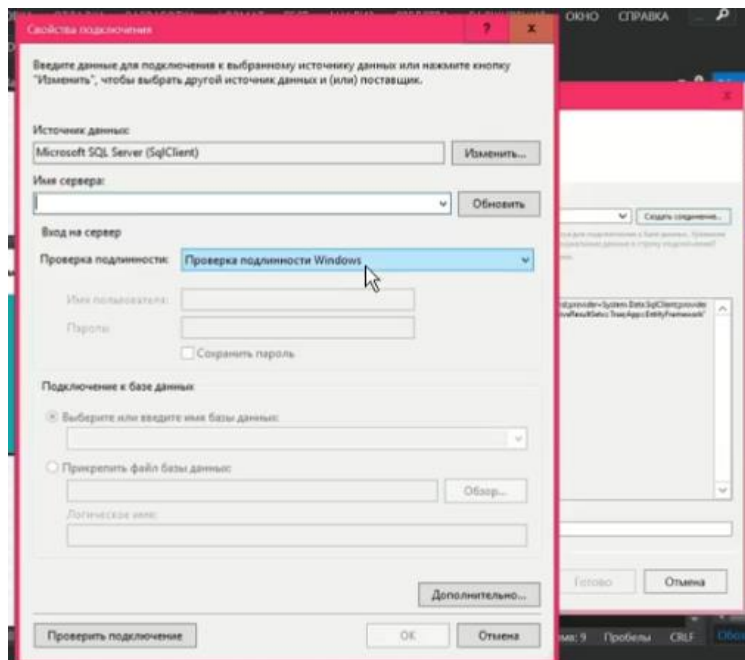
- Создайте приложение **WPF** в MS Visual Studio.
- Добавьте в проект в окне **Обозреватель решений** папку **ApplicationData** для размещения в ней модели базы данных.



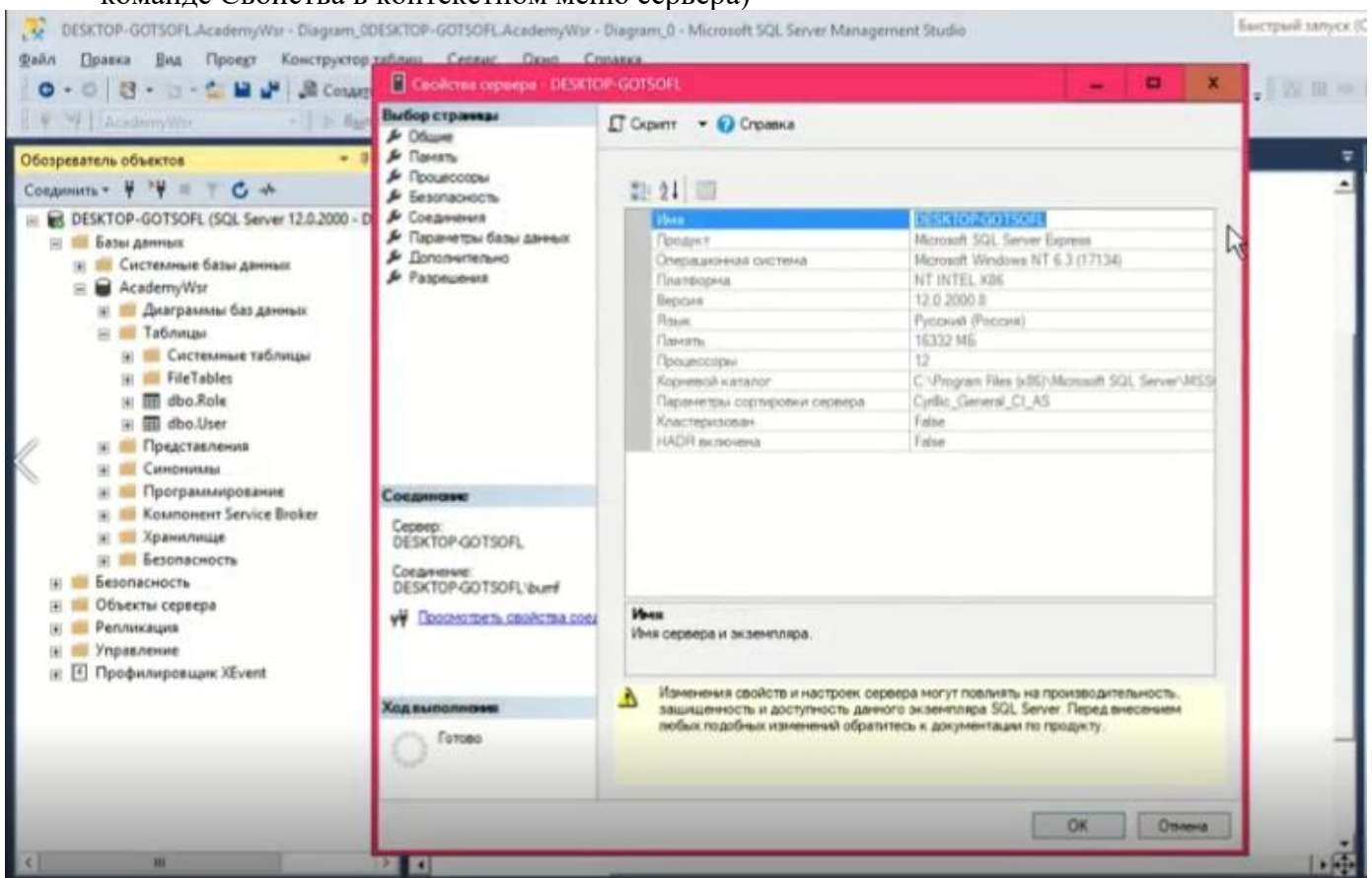
- В папке **ApplicationData** создайте элемент **Модель ADO.NET EDM** (имя модели можно не менять) Model1. Выберите нужную модель.
-



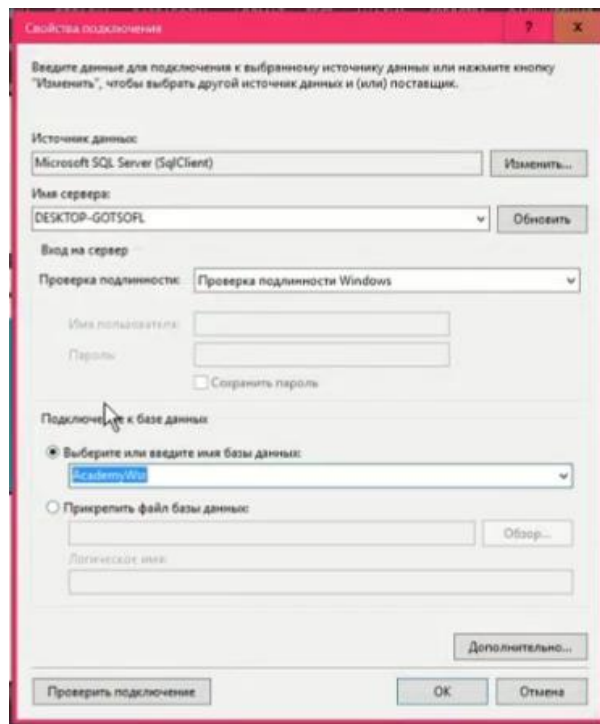
- Далее **Создать соединение**:



- написать **Имя сервера** (имя сервера можно взять в MS SQL Server Management Studio по команде Свойства в контекстном меню сервера)

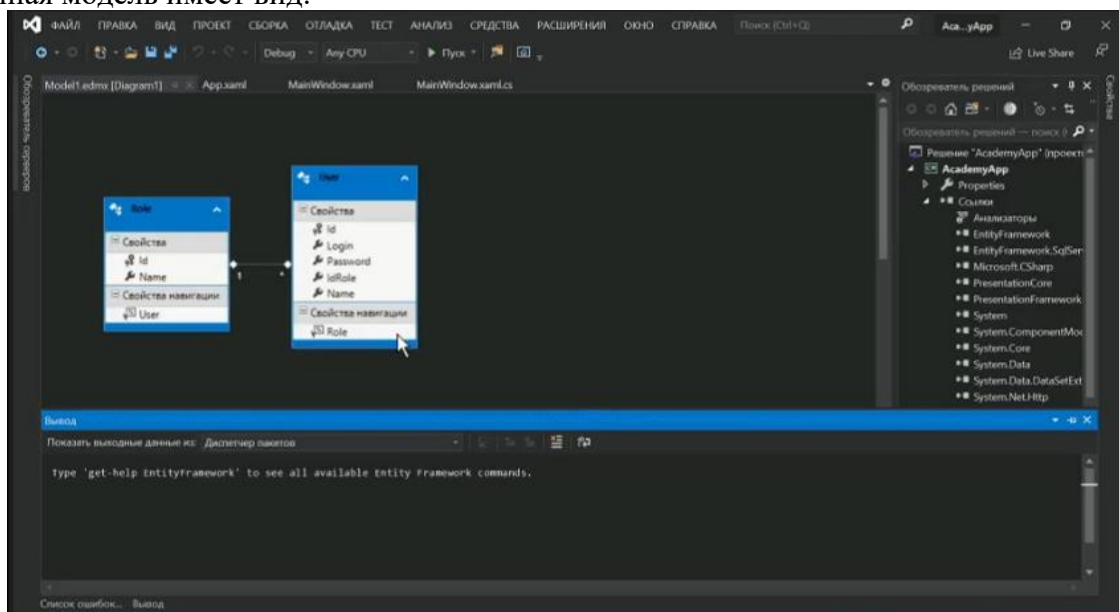


- выбрать вариант входа на сервер
- выбрать имя базы данных



6. Выбрать версию Framework и на последнем шаге выбрать таблицы БД.

Добавленная модель имеет вид:

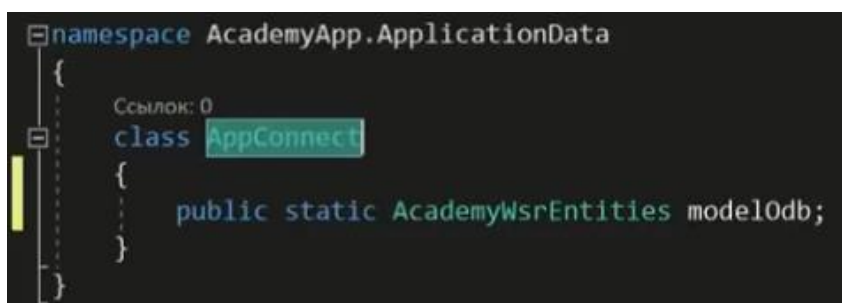


При изменении структуры базы данных в VS необходимо удалить таблицы модели, затем на свободном месте щелкнуть правой кнопкой мыши и выбрать команду Обновить модель из базы данных.

В обозревателе решений проекта отображается файл созданной модели **Model1.edmx**.

Подключение к БД

В папку **ApplicationData** добавить класс **AppConnect**, написать в нем подключение к нашей БД.



Подключить ту же самую модель как объект к главному окну **MainWindow**.

```
public partial class MainWindow : Window
{
    Ссылка: 0
    public MainWindow()
    {
        InitializeComponent();
        AppConnect.model0db = new AcademyWsrEntities();
    }
}
```

На этом Подключение БД к проекту завершено!

Интерфейс приложения (главное окно)

1. Организуйте разметку главного окна приложения в соответствии с рисунком:



Ширину окна установите 800, высоту 350, измените текст в строке заголовка.

Основной контейнер компоновки в окне - **Grid**, состоящий из двух строк:

```
<Grid.RowDefinitions>
    <RowDefinition Height="100*" />
    <RowDefinition Height="350*" />
</Grid.RowDefinitions>
```

В **верхней строке** сетки **Grid** размещен контейнер компоновки **StackPanel** с горизонтальным размещением в нем двух элементов: **Image** и **TextBlock**. Изображение логотипа поместить в папку проекта **Resources**. Стиль оформления надписи организовать в ресурсах файла **App.xaml**, например, в таком виде (можно изменить стиль на свое усмотрение):

```
<Style TargetType="TextBlock" x:Key="Title">
    <Setter Property="VerticalAlignment" Value="Center" />
    <Setter Property="FontSize" Value="13pt" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Margin" Value="15" />
</Style>
```

и применить к элементу **TextBlock**:

```
Style="{StaticResource Title}"
```

В нижней строке сетки **Grid** разместить **Frame**. Дать ему имя **FrmMain**, скрыть в нем панель навигации:

```
NavigationUIVisibility="Hidden"
```

2. Чтобы внутри данного фрейма выполнять действия (хранить и передавать информацию из страницы к странице), создадим специальный класс, и объект этого класса с именем **FrmMain**.

В паку **ApplicationData** добавьте класс **AppFrame**:

```
class AppFrame
{
    public static Frame frameMain;
}
```

Пропишите фрейм в коде главного окна:

```
public MainWindow()
{
    InitializeComponent();
    AppConnect.modelOdb = new AcademyWsrEntities();
    AppFrame.frameMain = FrmMain;
}
```

Добавление страниц

Для каждой вновь добавляемой страницы приложения будем организовывать в проекте отдельную папку.

Страница авторизации

1. Создайте в проекте папку **PageMain**. В ней создайте страницу **PageLogin**. Установите высоту 350, ширину 800.

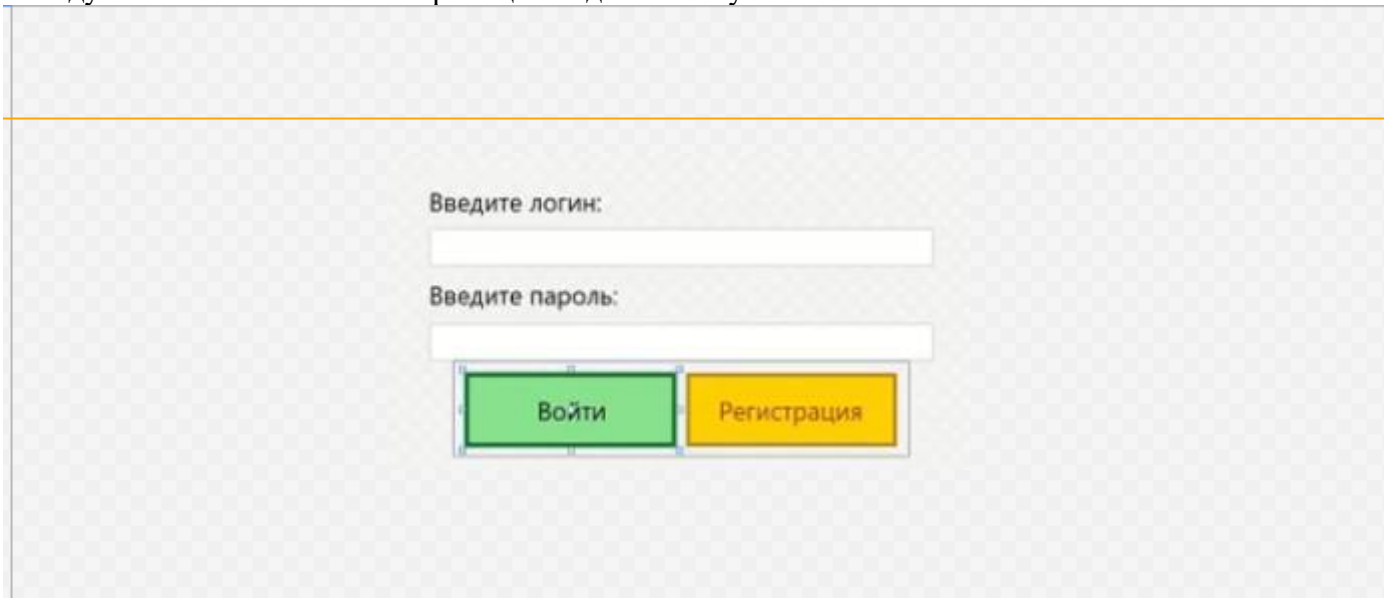
Организируйте разметку страницы в соответствии с рисунком.

Используйте контейнер **StackPanel**, расположите его в центре страницы.

Для ввода пароля используйте компонент **PasswordBox**.

Для оформления кнопок добавьте еще два шаблона в ресурсы файла **APP.xaml**.

Между всеми компонентами страницы создайте отступы в 5 пикселей:



2. Данная страница авторизации должна загружаться при запуске приложения. Для этого нужно в главном окне внутри фрейма инициализировать нашу страницу:

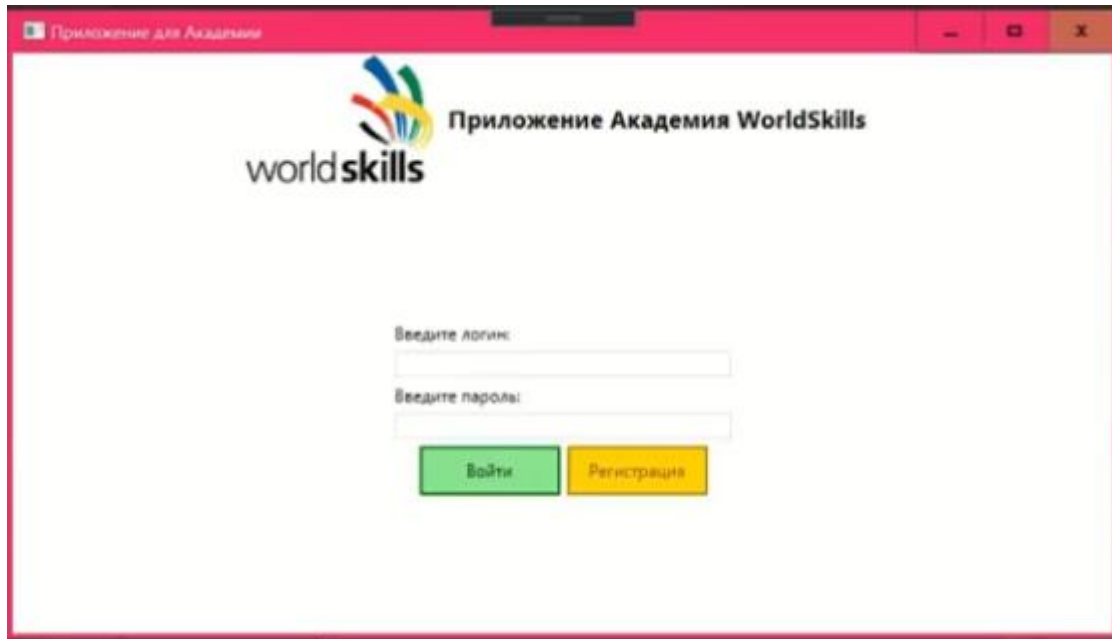
```

public MainWindow()
{
    InitializeComponent();
    AppConnect.modelOdb = new AcademyWsrEntities();
    AppFrame.frameMain = FrmMain;

    FrmMain.Navigate(new PageLogin());
}

```

Запустите проект, убедитесь, что страница авторизации загружается в окне приложения!



Функционал авторизации

Создайте обработчик события **Click** для кнопки **Войти**.

```

try
{
    var userObj = AppConnect.modelOdb.User.FirstOrDefault(x => x.Login == txbLogin.Text && x.Password == psbPassword.Password);
    if (userObj==null)
    {
        MessageBox.Show("Такого пользователя нет!", "Ошибка при авторизации!",
            MessageBoxButton.OK, MessageBoxImage.Error);
    }
    else
    {
        switch (userObj.IdRole)
        {
            case 1: MessageBox.Show("Здравствуйте, Администратор " + userObj.Name + "!",
                "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);
                break;
            case 2:
                MessageBox.Show("Здравствуйте, Ученик " + userObj.Name + "!",
                    "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);
                break;
            default: MessageBox.Show("Данные не обнаружены!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);
                break;
        }
    }
}
catch (Exception Ex)
{
    MessageBox.Show("Ошибка " + Ex.Message.ToString() + "Критическая работа приложения!",
        "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);
}

```

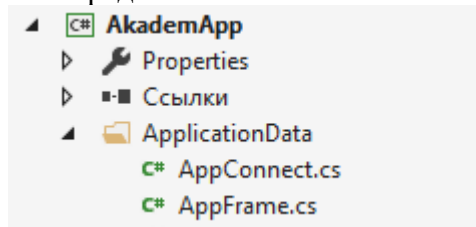
Пояснения к коду:

Когда идет работа с данными рекомендуют использовать **try..catch** (обработку исключений).
Внутри оператора **try** нужно написать процесс подключения к базе данных и работы с ней.

```
var userObj = AppConnect.modelOdb.User.FirstOrDefault(x =>  
x.Login == txbLogin.Text && x.Password == psbPassword.Password);
```

Создадим динамическую переменную **userObj**. Будем внутри нее хранить данные, получаемые из полей для ввода логина и пароля.

Обращаться к БД будем посредством созданного нами ранее класса **AppConnect** и работать непосредственно с ним.



Модель **modelOdb** видит входящие в БД таблицы в качестве атрибутов (**User**).

При наборе строки

```
var userObj = AppConnect.modelOdb.User.
```

после очередной точки перечисляются методы работы с **LINQ**-запросами и **EntityFramework**.

Нам нужен метод **FirstOrDefault** (он возвращает первый элемент последовательности...).

Создаем анонимную переменную **x**, используемую внутри **LINQ**-запроса (аналогично параметру цикла **for**).

Возвращаем в переменную **userObj** все значения из таблицы **User**, которые совпали по введенным в текстовые поля логину и паролю.

Если в БД пользователя нет, то нужно вывести об этом сообщение (**if...**), иначе – нужно вывести приветствие для соответствующего пользователя.

Запустите проект, убедитесь, что на экран выводится каждое из организованных в коде приветствий и уведомлений!

Страница регистрации

1. В папке **PageMain** создайте страницу **PageCreateAcc**. Установите высоту страницы 350, ширину – 800.

Создадим интерфейс для возможности отправки в БД введенных значений нового пользователя.

Организируйте разметку страницы в соответствии с рисунком.

Используйте контейнер **StackPanel** с вертикальной ориентацией, разместите его в центре страницы.

В нем организуйте еще 4 контейнера **StackPanel** горизонтальной ориентацией.

Для ввода пароля используйте компонент **TextBox**, для подтверждения пароля – **PasswordBox**.



Оформление компонентов организуйте с помощью стилей, шаблоны которых пропишите в файле **App.xaml**.

По нажатию на кнопку **Регистрация** страницы **Авторизация** должна открываться страница для создания аккаунта ученика. При нажатии на кнопку **Назад** страницы **Регистрация** мы должны вернуться на страницу **Авторизация**.

2. Создайте обработчик события **Click** для кнопки **Регистрация**:


```
private void btnRegIn_Click(object sender, RoutedEventArgs e)
{
    AppFrame.frameMain.Navigate(new PageCreateAcc());
}
```

3. Создайте обработчик события Click для кнопки Назад:

```
private void btnBack_Click(object sender, RoutedEventArgs e)
{
    AppFrame.frameMain.GoBack();
}
```

Пояснения к коду:

Все действия на главном окне у нас выполняются во фрейме.

Этот фрейм хранится в объекте **frameMain**.

Мы загрузили наш фрейм с главной страницы.

**Запустите проект, убедитесь, что осуществляются соответствующие переходы по кнопкам
Регистрация и Назад!**

Функционал прохождения регистрации ученика

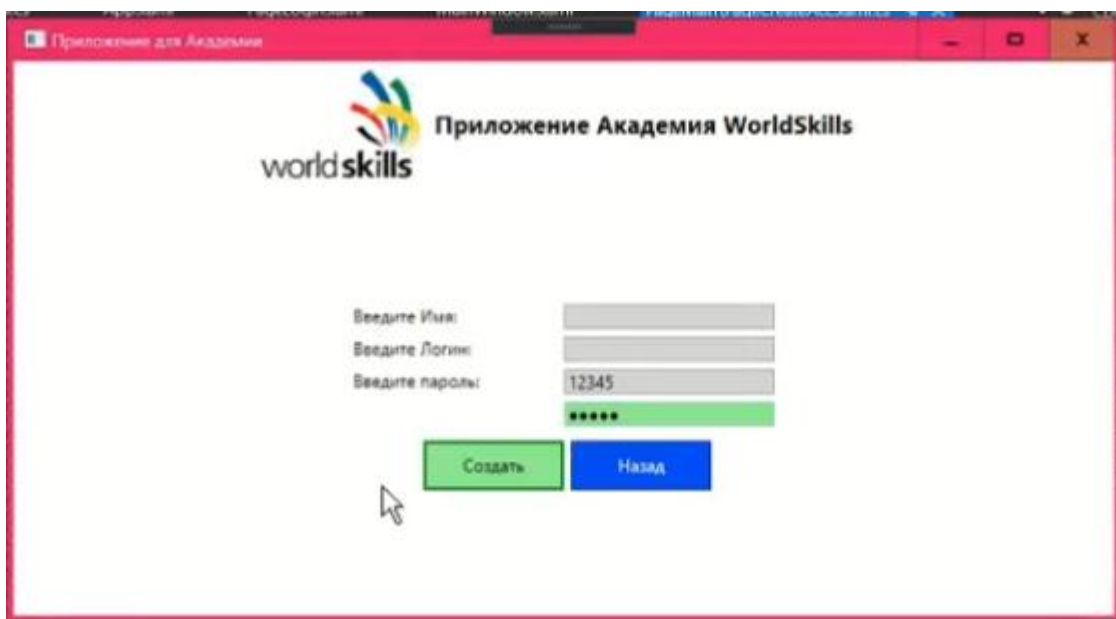
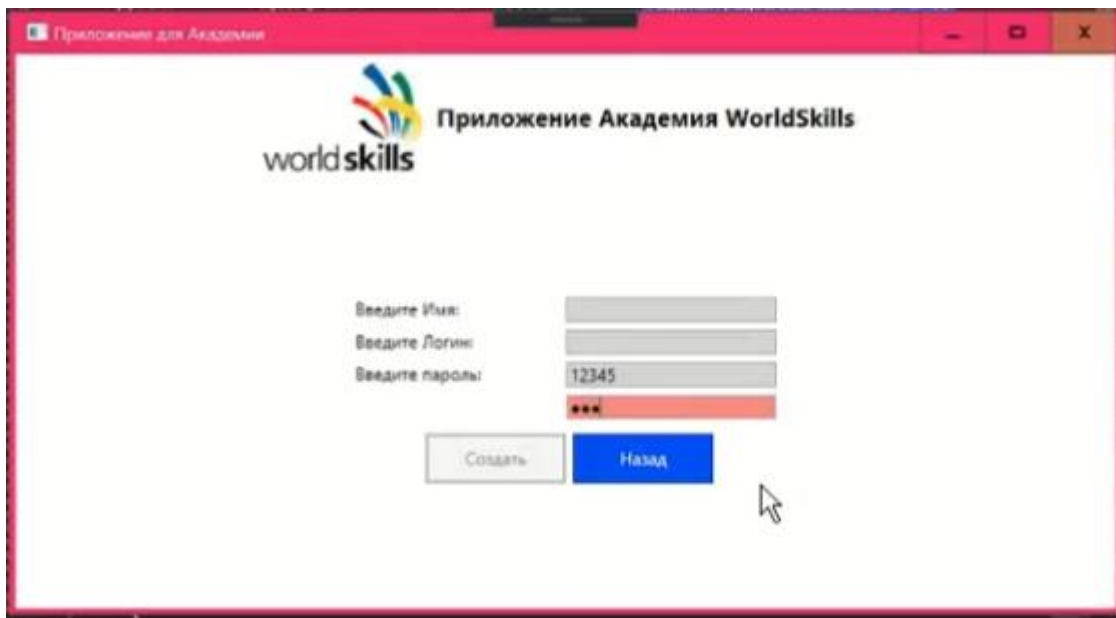
1. Кнопка **Создать** должна быть недоступна, пока в поле подтверждения пароля вводимое значение не совпадет с первоначальным. А поле подтверждения пароля окрашивается красным пока нет совпадения и зеленым в момент совпадения значений в полях.

Для этого:

- Кнопку Создать сделать изначально неактивной **IsEnabled="False"**
- Создайте обработчик события **PasswordChanged** компонента **PasswordBox**:

```
private void PasswordBox_PasswordChanged(object sender, RoutedEventArgs e)
{
    if (psbPass.Password!=txbPass.Text)
    {
        btnCreate.IsEnabled = false;
        psbPass.Background = Brushes.LightCoral;
        psbPass.BorderBrush = Brushes.Red;
    }
    else
    {
        btnCreate.IsEnabled = true;
        psbPass.Background = Brushes.LightGreen; ;
        psbPass.BorderBrush = Brushes.Green;
    }
}
```

**Запустите проект. Убедитесь, что кнопка Создать и поле подтверждения пароля ведут себя
как было запланировано!**



2. Внутри обработчика события **Click** кнопки **Создать** нужно организовать добавление в базу данных значений, вводимых пользователем в соответствующие текстовые поля. Для этого необходимо обратиться к таблице **User** базы данных, и эта таблица должна получить значения из компонентов страницы **Регистрация** (Имя, Логин и Пароль).

```

private void btnCreate_Click(object sender, RoutedEventArgs e)
{
    if (AppConnect.modelOdb.User.Count(x => x.Login==txbLogin.Text)>0)
    {
        MessageBox.Show("Пользователь с таким логином есть!",
            "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);
        return;
    }
    try
    {
        User userObj = new User()
        {
            Login = txbLogin.Text,
            Name = txbName.Text,
            Password = txbPass.Text,
            IdRole=2
        };
        AppConnect.modelOdb.User.Add(userObj);
        AppConnect.modelOdb.SaveChanges();
        MessageBox.Show("Данные успешно добавлены!",
            "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);
    }
    catch
    {
        MessageBox.Show("Ошибка при добавлении данных!",
            "Уведомление", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

Пояснения к коду:

Сначала нужно проверить, есть ли пользователь с таким логином в системе.

Если значения, возвращаемые из таблицы **User** по полю **Login** имеются (применяем для этой цели метод **Count** LINQ-запроса), то вывести уведомление о том, что пользователь с таким логином существует в БД.

Для добавления данных нового пользователя нужно использовать конструкцию **try..catch**.

Создаем новый объект **UserObj** класса **User**, присваиваем его свойствам значения из компонентов страницы **Регистрация**. Регистрировать можно только учеников, поэтому присваиваем значение 2 полю **idRole**. Записываем созданный объект в таблицу БД **User** (метод **Add**) и сохраняем изменения (метод **SaveChanges**). По окончании выводим уведомление о проделанной операции.

Запустите проект, зарегистрируйте нового ученика, осуществите вход под его аккаунтом!

3. Измените интерфейс таким образом, чтобы название проекта отображалось в строке заголовка главного окна, а в верхней части окна каждый раз располагалось название открытой страницы (**Окно авторизации, Окно регистрации**).

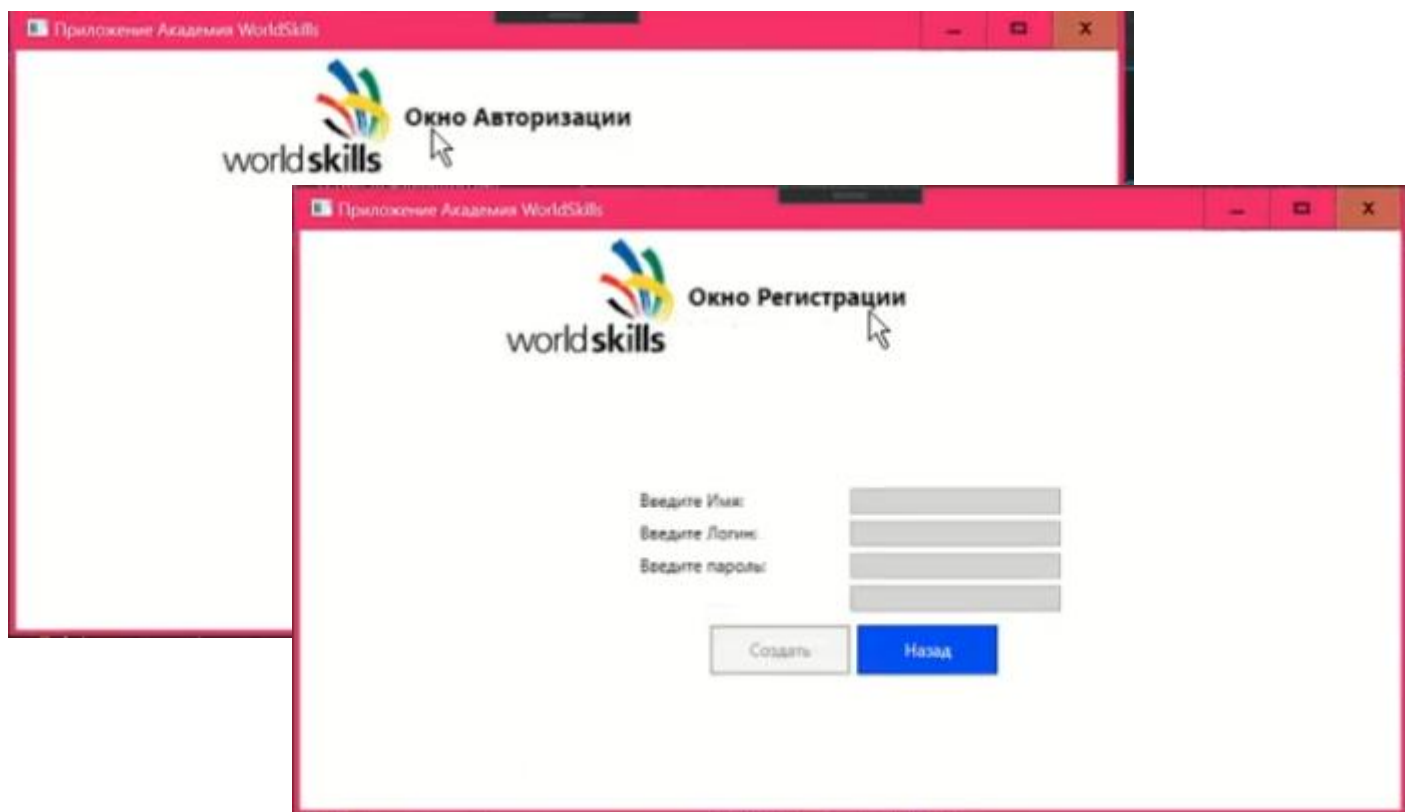
Изменения для этого вносим в разметку: добавляем элемент **TextBlock** и в его свойстве **Text** указываем источник текста – **Title** открытой страницы (содержимого фрейма).

```

Text="{Binding ElementName=FrmMain,
    Path=Content.Title}"

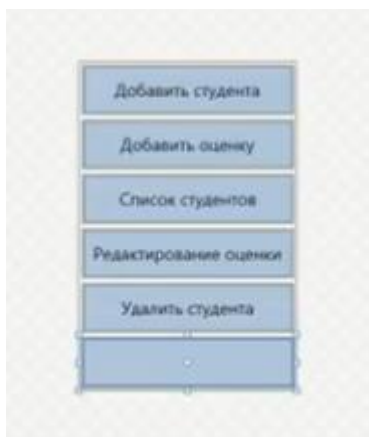
```

Запустите проект, убедитесь в отображении заголовков открытых страниц!



Страница администратора

Добавьте в проект папку **PageAdmin** и создайте в ней страницу **PageMenuAdmin** (высота 350). Организуйте разметку страницы в соответствии с рисунком. Используйте контейнер **StackPanel**, расположите его по центру страницы. Организуйте переход к этой странице после авторизации.



Страница ученика (студента)

Добавьте в проект папку **PageStudent** и создайте в ней страницу **PageAccountStudent** (высота 350). Организуйте разметку страницы в соответствии с рисунком. Используйте контейнеры **StackPanel**. Организуйте переход к этой странице после авторизации.

Имя пользователя: Тут будет имя!
Логин: Тут будет логин!
Последний вход: Тут будет последняя дата!

Запустите проект, убедитесь, что переходы к страницам осуществляются и заголовки страниц выводятся в окне!

