Пояснительная записка к разработанному шаблонному приложению на WPF для подготовки к демо экзамену.

Разработчик-автор: Ершов Дмитрий Алексеевич, студент группы 18ИП1.

Целевая аудитория: типчики из шараги (не преподаватели, кроме Егора Олеговича, он свой).

Цель: оставить наследие и помочь челам разобраться в быстром создании приложения и реализации функциональности.

Тема шаблонного проекта: корпоративная сеть.

Структура сети: сеть состоит из пользователей разных ролей, минимальный перечень ролей: модераторы (админы) и обычные пользователи. В данной сети пользователи могут добавлять друг друга в друзья, а также удалять из друзей.

Минимальный функционал:

* Модераторы могут просматривать пользовательские учетные данные;
* Модераторы могут удалять/добавлять пользователей любых ролей;
* Модераторы могут повышать/уменьшать роль пользователя;
* Пользователи могут добавлять/удалять друзей;
* Пользователи просматривают список друзей;
* Пользователи применяют сортировку/фильтрацию/поиск к списку друзей для поиска нужного друга.

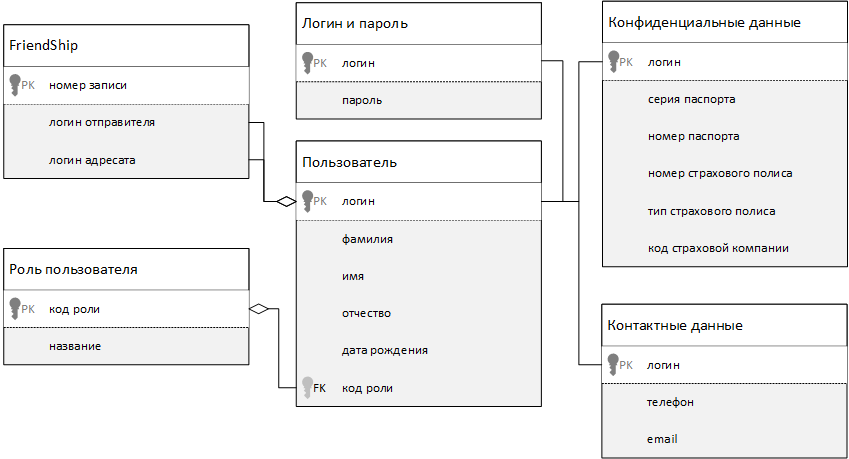
Пояснение к внедрению/разработке функционала:

Каждая новая функциональность будет разработана в отдельной ветке и после успешного завершения будет слита в master, для подробностей смотрите историю коммитов гита.

(команда *$ git log --pretty=format:"%h %s" --graph*).

Эталонная диаграмма ERD (базы данных, нотация UML):

Рекомендую в любой АИС делать такую структуру базы данных, связанных с пользователем, вам начислят дополнительные баллы за безопасную декомпозицию и расширяемость.



На начальных этапах проектирования не пихайте палки себе в колеса. Не создавайте трудностей в дальнейшем, потому что в будущем для расширения и модификации вам потребуется значительно больше времени.

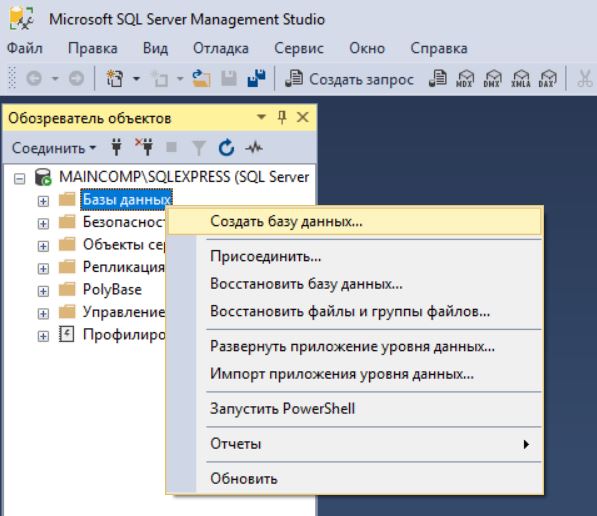
Ладно, лирическое отступление закончилось, начнем с базы данных.

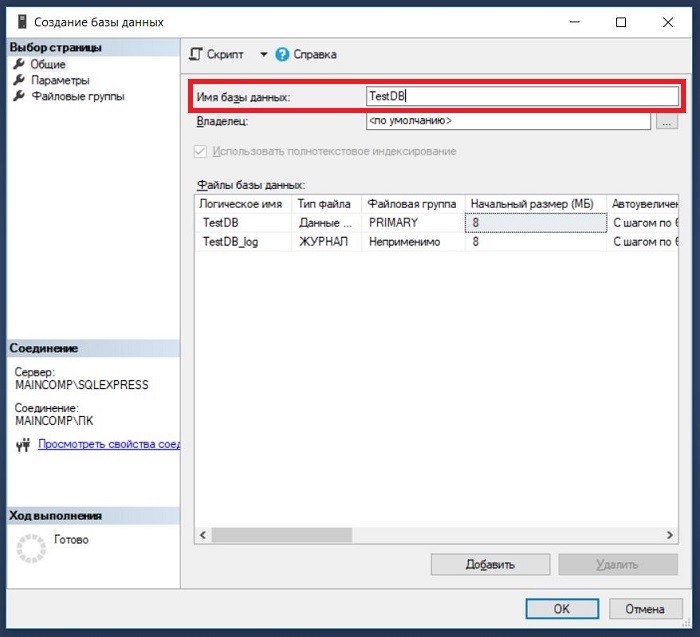
Первым делом вам необходимо создать базу данных или развернуть ее из скрипта, затем следует нормализовать/привести данные из ресурсов под ранее созданную базу данных.

Создавать базу данных исключительно из конструктора или из готового файла, данного в ресурсах, и никак иначе, потому что необходимо беречь время.

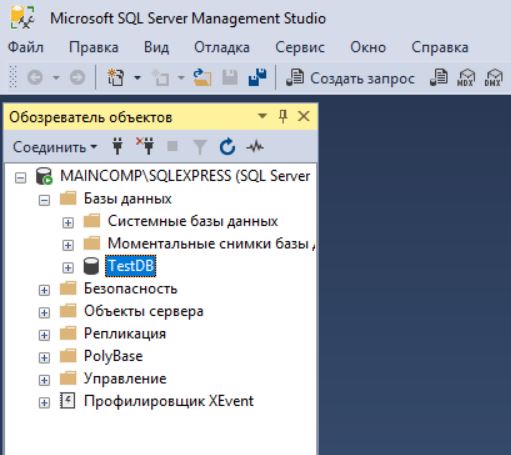
Создать базу данных можно двумя способами:

1. Создать ручками



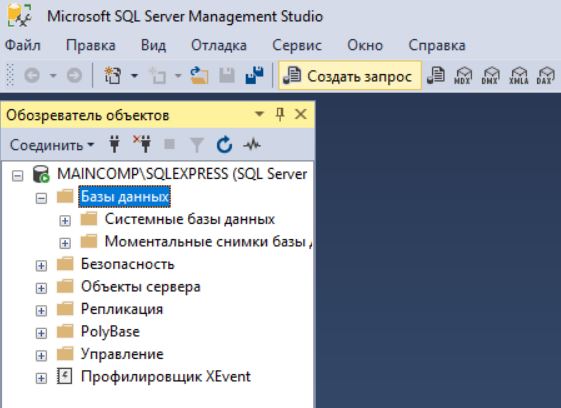


В результате откроется окно «Создание базы данных». Здесь обязательно нужно заполнить только поле «Имя базы данных», остальные параметры настраиваются по необходимости. После того, как Вы ввели имя БД, нажимайте «ОК».

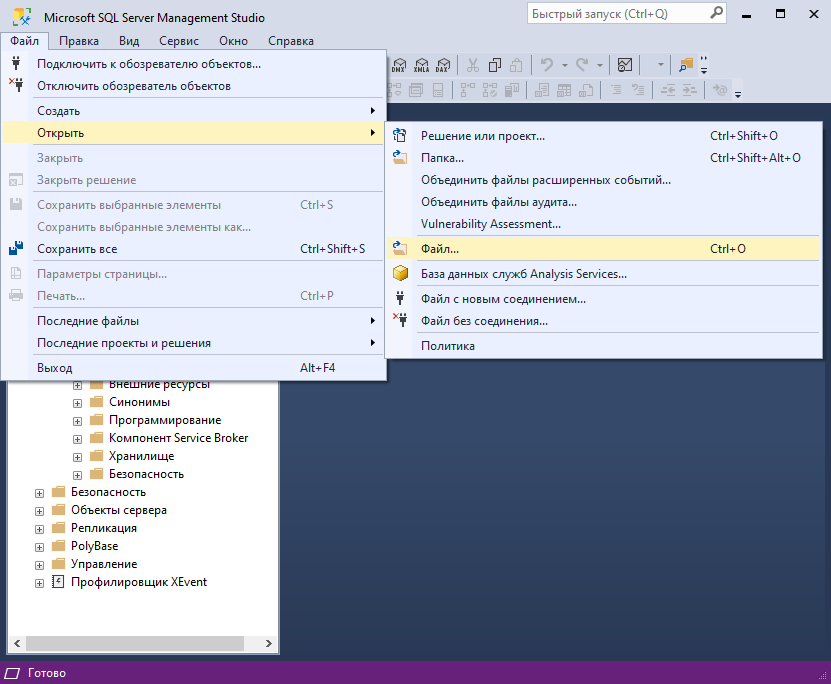


Если БД с таким именем на сервере еще нет, то она будет создана, в обозревателе объектов она сразу отобразится

1. Развернуть скрипт базы



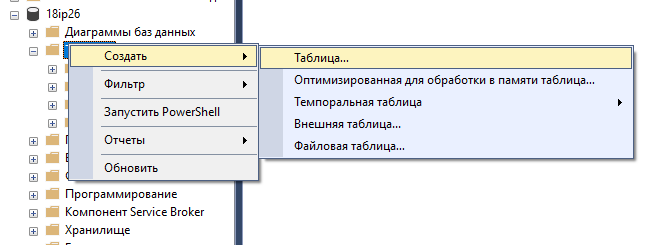
Сначала открываем редактор SQL запросов, для этого щелкаем на кнопку «Создать запрос» на панели инструментов. Затем вводим/вставляем нужный запрос создания базы данных и выполняем. Если скрипт базы данных уже дан, то его можно открыть и выполнить.



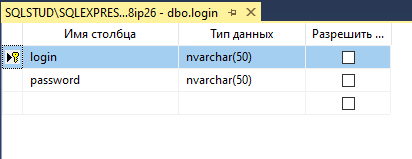
Создание таблиц

Создать таблицы можно двумя способами:

1. Создать с помощью скрипта (см. предыдущий рисунок)
2. Создать ручками



Выделяем контейнер "Таблицы" правой кнопкой мыши и выбрать пункт "Создать" и "Таблица..."



Созданная таблица "Логин и пароль"

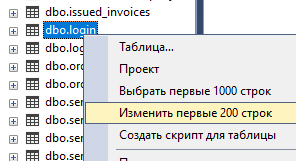
Итогом у вас должна получиться следующая схема базы данных в MS SQL Server Management Studio.



Приступим к импорту данных в базу, для этого необходимо привести данные в общему виду в Excel и потом импортировать данные. Кто не знает, как избавляться от порченых данных следует разобраться в Excel иначе ***гг вп изи катка***.

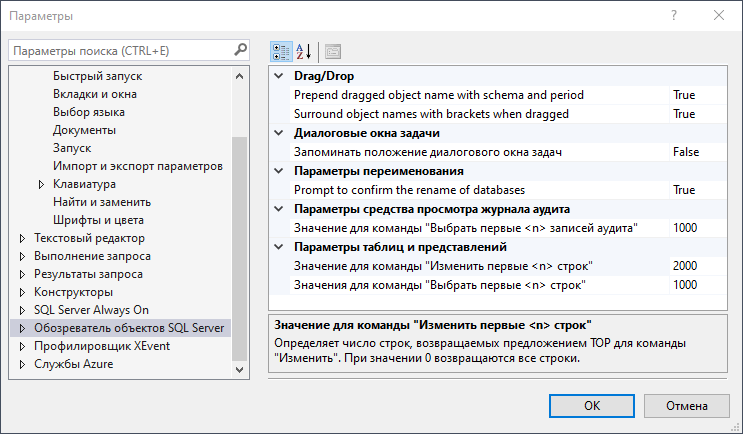
Все данные необходимые для импорта располагаются по пути *./data-files/import-data.xlsx*

Самый быстрый способ заполнить таблицы любыми данными это скопировать их и вставить в таблицу через изменение данных таблицы.



Совет

Для вставки большого количества данных в таблицу, используя способ продемонстрированном на рисунке 8, которые более 200 строк, то вам необходимо зайти в «Сервис/Параметры» и в открывшемся окне открыть раздел «Обозреватель объектов SQL Server».

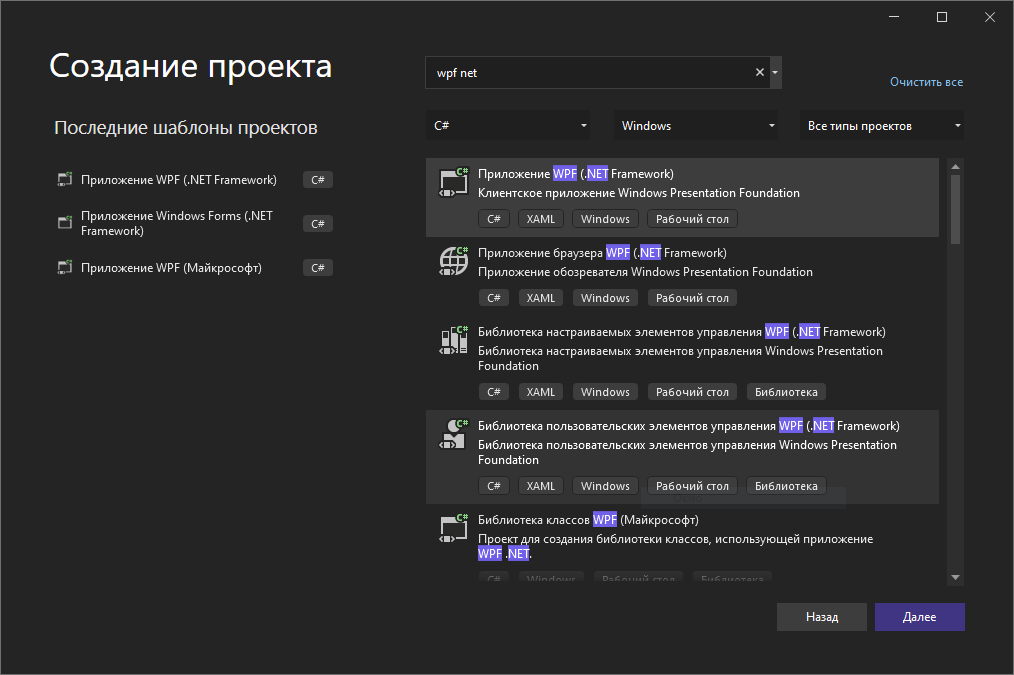


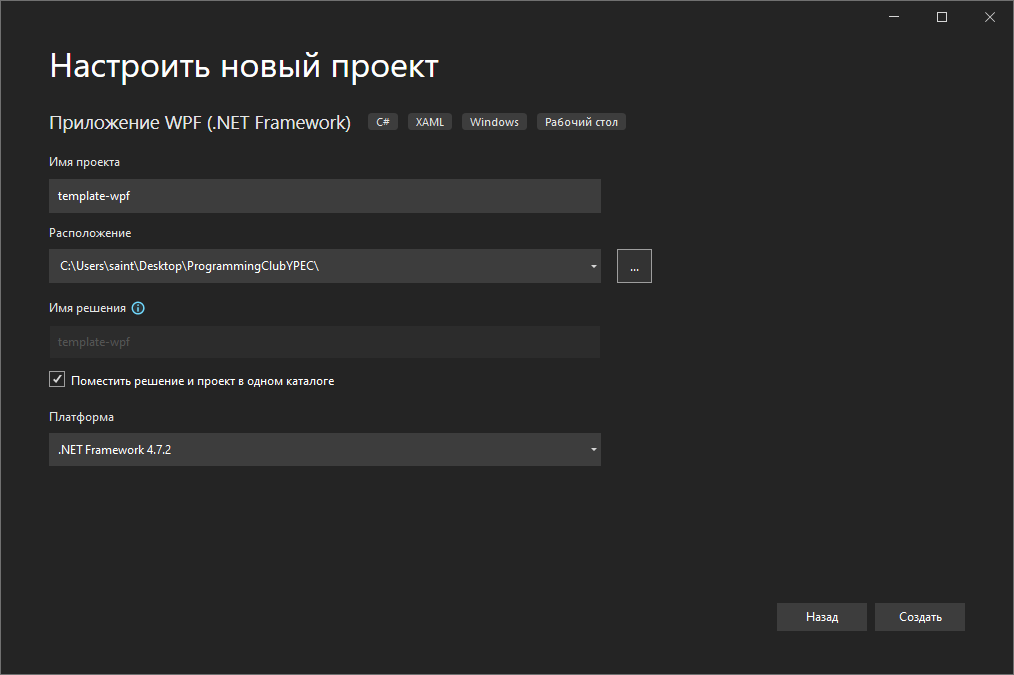
Приступим к реализации каркаса приложения

Для этого нам понадобиться создать приложение WPF в Visual Studio, внизу приведены скрины шагов для создания проекта.

Добро пожаловать в нашу адаптированную версию Шрека.

Глава нашей истории 1: построим мост к Фионе.





Итогом данных телодвижений у вас должен будет создаться проект в указанной папке, если вам не хочется иметь много мусора в корне, тогда уберите галку с чек бокса, если вам пофиг тогда ставьте.

Небольшой обзор созданного проекта по умолчанию

Студия создает сразу же форму MainWindow и делает ее запускаемой, также создает дополнительные конфигурационные файлы для проекта.

Стилистика написания фронта используя язык разметки xaml от майков.

Никогда не пишите разметку в одну строчку – это вам будет только мешать. Объявление и изменение значений полей с аналогичной контекстной нагрузкой пишите в одну строчку. Если более 4 полей в строке, то вы что-то делаете неправильно. Когда объявляете новые UI элементы то между их тегами делайте пустую строчку. Чтобы интерфейс был адаптивным присваивайте элементам расположения внутри макетов. Пример приведен ниже.

<Window x:Class="template\_wpf.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:template\_wpf"

mc:Ignorable="d"

Title="MainWindow"

Height="450" Width="800"

MinHeight="450" MinWidth="800"

>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition />

</Grid.RowDefinitions>

<Frame x:Name="PageFrame"

MinHeight="300" MinWidth="600"

Grid.Column="0" Grid.ColumnSpan="3"

Grid.Row="0" Grid.RowSpan="3"

Content="{Binding CurrentPage}"

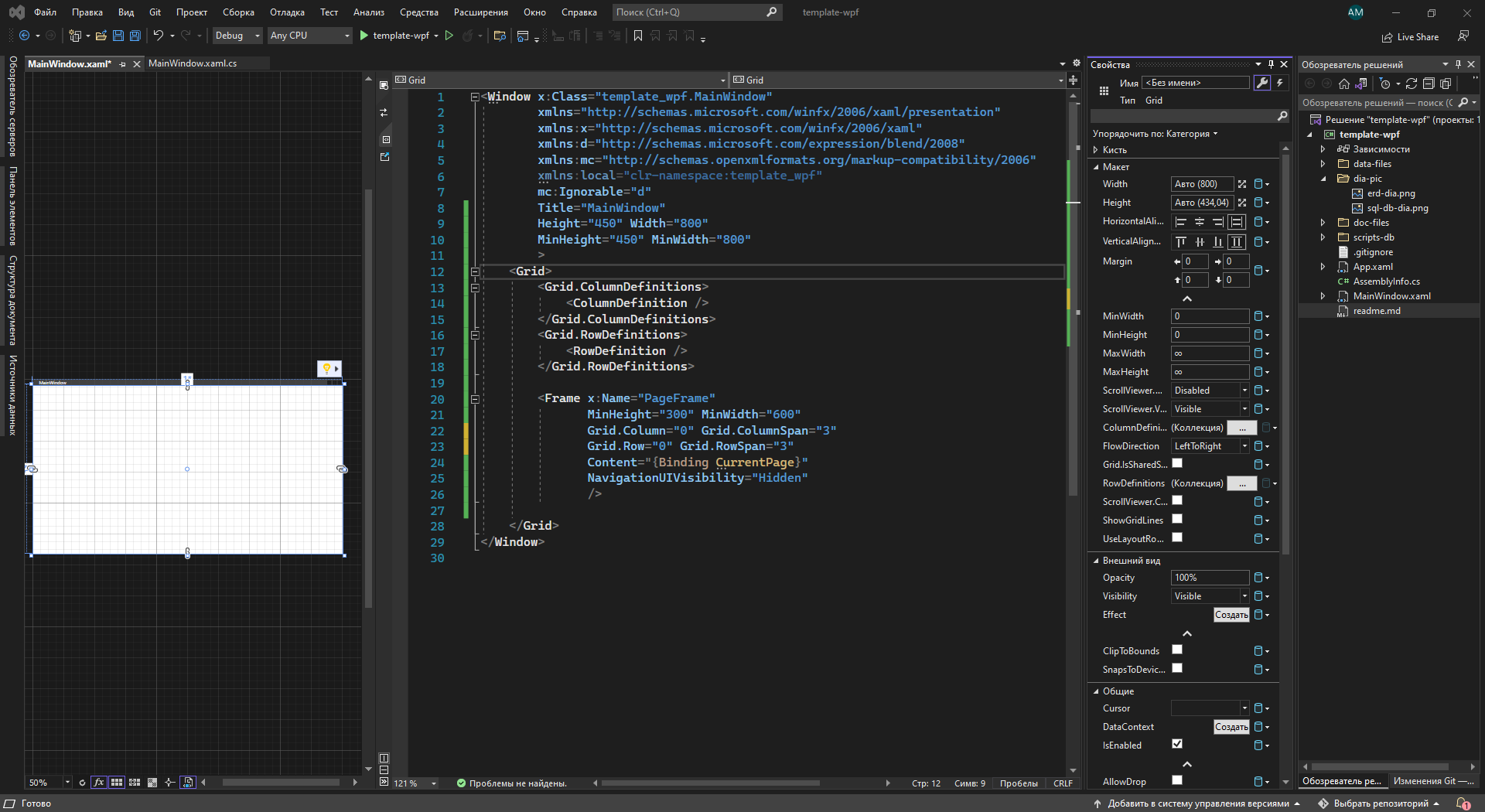
NavigationUIVisibility="Hidden"

/>

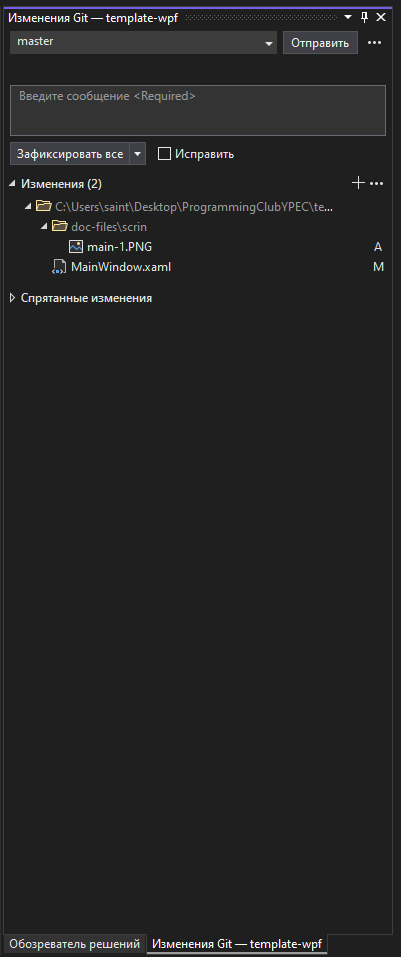
</Grid>

</Window>

Разместите инструменты студии в удобном для вас положении, если вы не знаете, что для вас удобно, то советую мою компоновку инструментов. Разрабатывая на WPF можно оставить панель «Свойства» по умолчанию или скрыть ее вообще. Если же разрабатываем на WinForm, то она вам необходима в полном виде, да и там редактора разметки нет. Но WinForm слишком устарела и заставляет писать много кода для функционирования приложения. Смотрите скрин.



Работать с контролем версий можно из студии переключившись на другую вкладку внутри «Обозревателя решений», смотри ниже скрин.



Я думаю, здесь будет излишне пояснять за функциональность данного инструмента, сами посмотрите и потыкаете по кнопочкам. Мне же удобнее через консоль производить фиксацию изменений.

Ладно, поговорили о среде разработки, пора и за код пояснить.

Недавно (месяца 2 назад) на меня снизошло озарение как быстро создавать приложения для ВСР и ДЭ, но потом я забил на это, потому что так лень было в соло писать такие гигантские документы. Наблюдая за положением вещей и нравов в нашем учебном заведении, выходит, что челы не шарят что да как делать и каждая попытка преподавателей провести практики с подобными условиями, как на вышеперечисленных мероприятиях, приводит к анальной и гортанной боли. В итоге надо помочь типам уменьшить их страдания и дать надежды на лучшее будущее.

Ладно, опять попиздели, теперь пора обсудить смысловую нагрузку разработки приложения.

За основу берется парадигма минимального создания окон и распехивание всего функционала по страницам, которые можно будет применять по несколько раз. Также обобщение функционала для того, чтобы было меньше дублирования кода.

Приступим к реализации

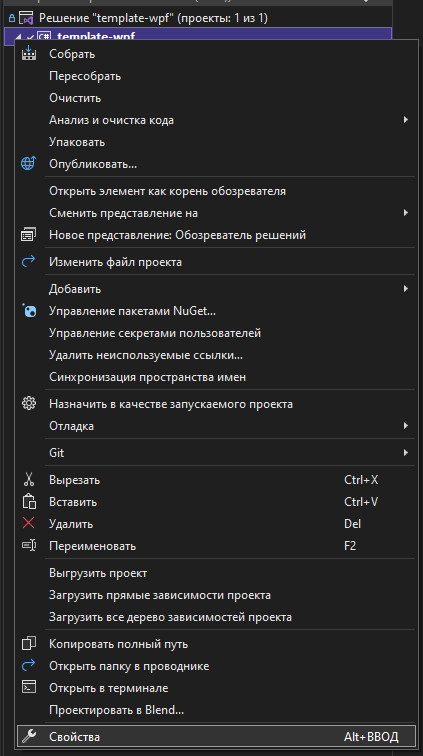
Когда вам придется создавать любое приложение АИС любой предметной области, то вам придется начинать всегда с окна авторизации пользователей. Данное окно будет стартовым и работать только с таблицей login из нашей базы данных.

Ресурсы приложения

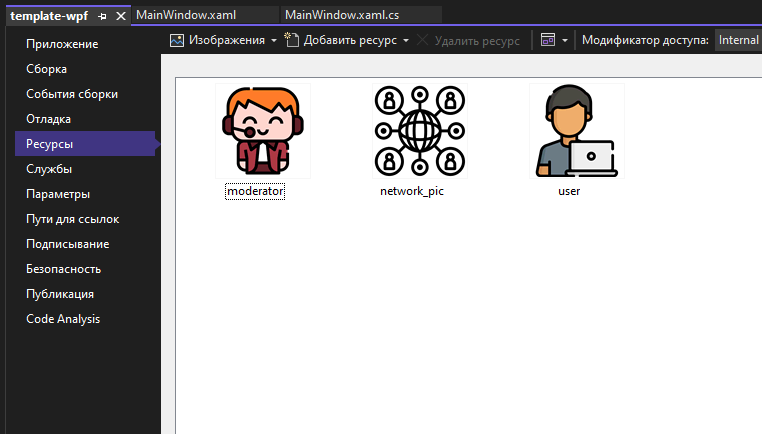
Самая дурная штука на свете – это ресурсы приложения в студии, майки настолько ебанутые, что слов не хватает. Чтобы не было проблем, каждый файл называйте уникально, и даже если у них разные расширения – все равно называйте по-разному, никогда не называйте ресурсы начиная с числа или с других символов.

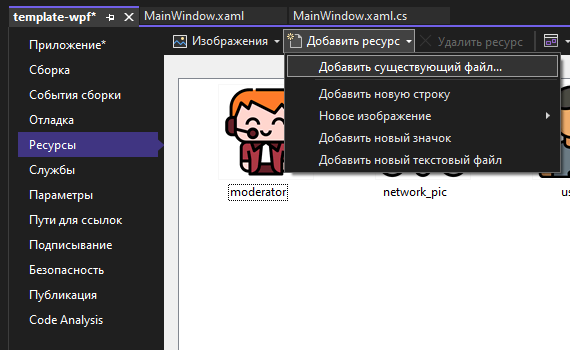
Добавление ресурсов в проект

Внутри «Обозревателя решений» добавляем папку названную «Resources», затем переходим в свойства проекта (см. скрин ниже). Кликаем ПКМ по корню решения и выбираем «Свойства».



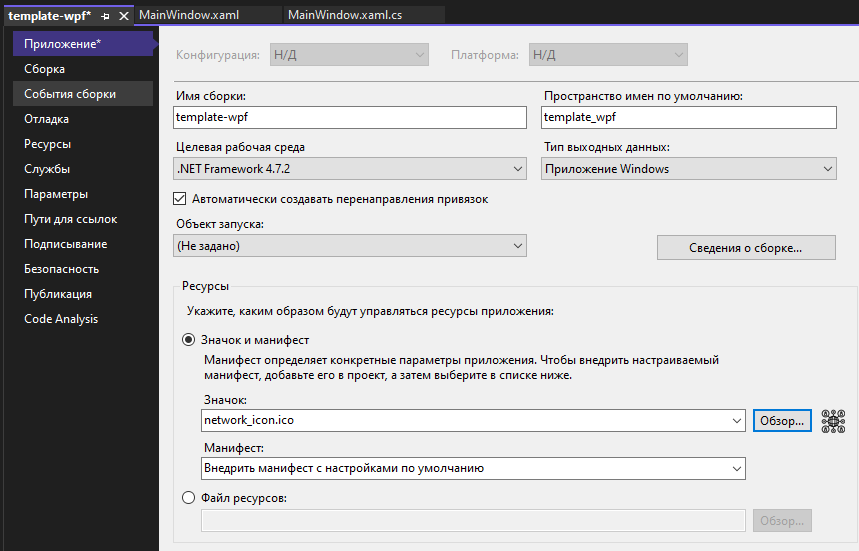
Затем переходим в ресурсы (слева будет список), у разработчиков нет никакой стабильности, интерфейс студии меняется от релиза к релизу – бесит. У меня студия 22 года, в шараге ее обновили, поэтому думаю наши мысли совпадут. Выбираем «Создать или открыть ресурсы сборки», у кого 19 студия, то сразу откроется вкладка.





Переходим на вкладку ресурсов и добавляем существующие файлы, в нашем случае я сразу добавляю иконку приложения в двух форматах, заметьте, когда будете смотреть исходники проекта, то обратите внимание на названия фалов логотипа приложения. Спойлер: *network\_icon.ico & network\_pic.png*. И добавил фотки пользователей для двух ролей.

Дальше вам необходимо установить иконку приложения, для этого переходим в «Приложение/Ресурсы/Значок» и выбираем из тех ресурсов, которые добавили недавно.



Кто дошел до сюда, красавцы, вы сможете получить нихуевые 10 баллов 😊. Это за создание базы данных и импорт данных, также плюсом идет созданная ERD диаграмма и развернули нихуевый (пока что хуевый) проект.

Погнали возьмем на абордаж окно авторизации и переходы между окном авторизации и основным окном.

Создаем окно авторизации, называя ее «signin» и быстренько делаем разметку интерфейса с полями ввода логина и пароля и кнопку авторизации, и не забываем про логотип приложения.

signin.xaml

<Window x:Class="template\_wpf.signin"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:template\_wpf"

mc:Ignorable="d"

Title="signin" Height="450" Width="300"

MinHeight="450" MinWidth="300"

>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.2\*"/>

<ColumnDefinition Width="0.6\*"/>

<ColumnDefinition Width="0.2\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="0.4\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.05\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.2\*"/>

</Grid.RowDefinitions>

<Image x:Name="logo\_app"

Grid.Column="1"

Stretch="Uniform"

/>

<TextBox x:Name="tbLogin"

Width="140" Height="20" Margin="5"

Grid.Row="1" Grid.Column="1"

HorizontalAlignment="Center" VerticalAlignment="Center"

Text="login..." Foreground="Gray"

GotFocus="tbLogin\_GotFocus"

LostFocus="tbLogin\_LostFocus"

TextChanged="tbLogin\_TextChanged"

/>

<PasswordBox x:Name="tbPwd"

Width="140" Height="20"

Grid.Row="2" Grid.Column="1"

HorizontalAlignment="Center" VerticalAlignment="Center"

Margin="5"

Password="password..." Foreground="Gray"

LostFocus="tbPwd\_LostFocus"

GotFocus="tbPwd\_GotFocus"

PasswordChanged="tbPwd\_PasswordChanged"

/>

<Button x:Name="signButton"

Content="Sign in"

Margin="5"

Grid.Row="4" Grid.Column="1"

Click="SignButton\_Click"

/>

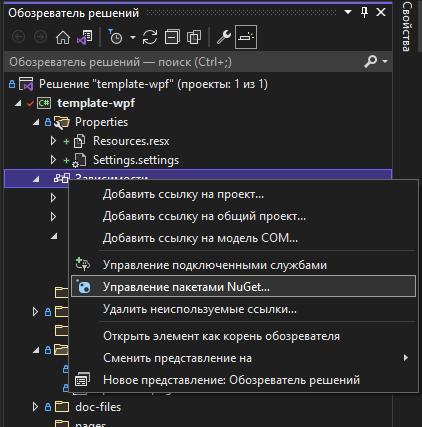
</Grid>

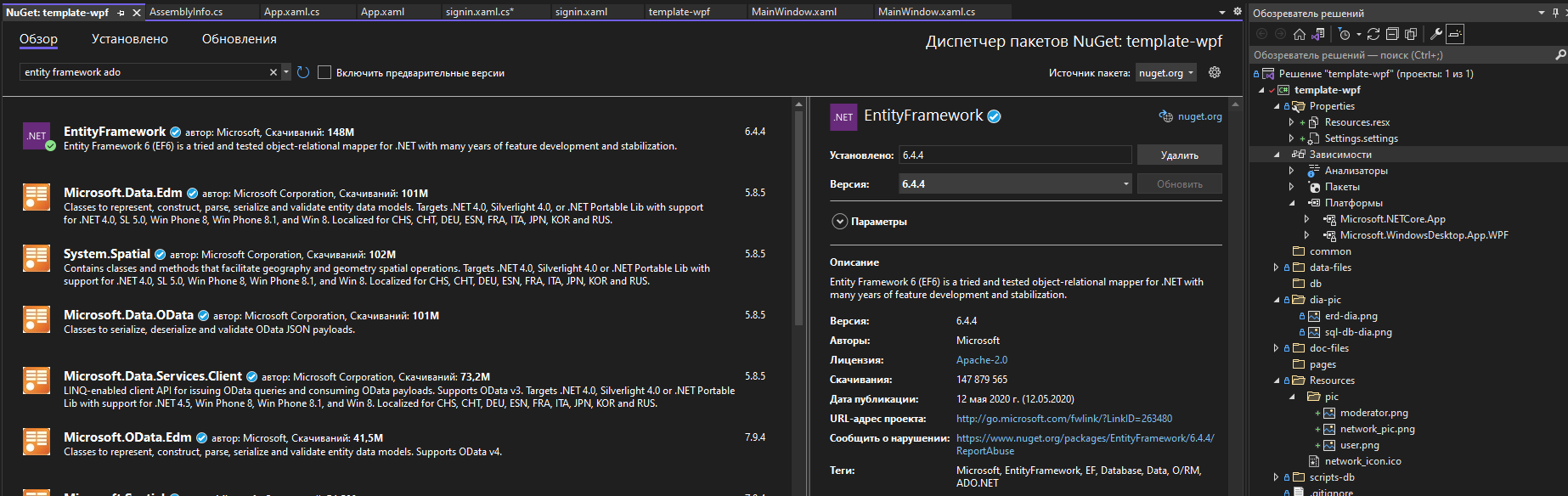
</Window>

Далее, чтобы успешно наращивать функционал авторизации, нам необходимо подключиться к базе данных и для этого дела мы добавляем папочку «db» или как вам угодно, и уже внутри нее используем «Модель ADO.NET EDM». Она упростит нам жизнь по написанию скриптов для получения данных из базы. В шараге она установлена, но может и пропасть. У кого дома не установлена, то устанавливаете ее через NuGet пакеты.

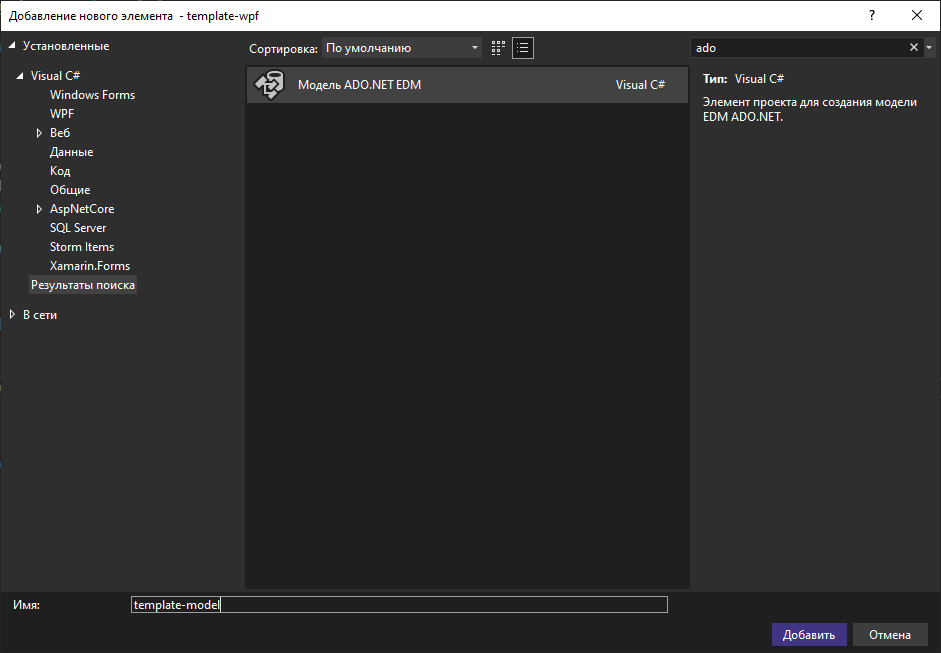
Как установить необходимый пакет

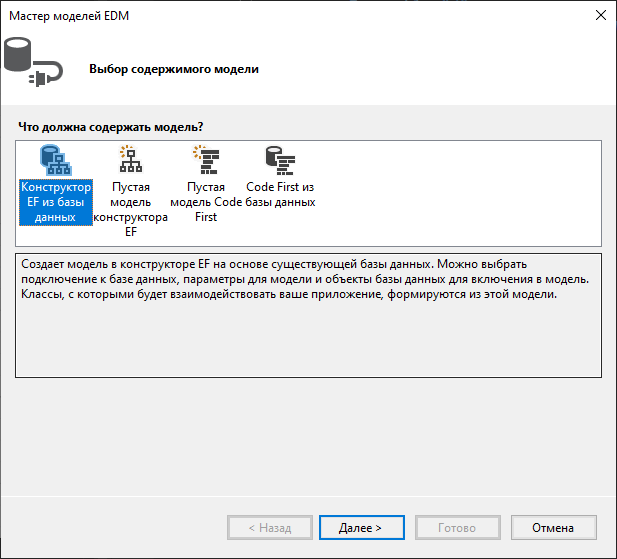
Кликаем ПКМ по зависимостям проекта и выбираем «Управление пакетами NuGet», далее переходим в обзор и вбиваем в строке поиска «entity framework ado» и устанавливаем первый пакет.

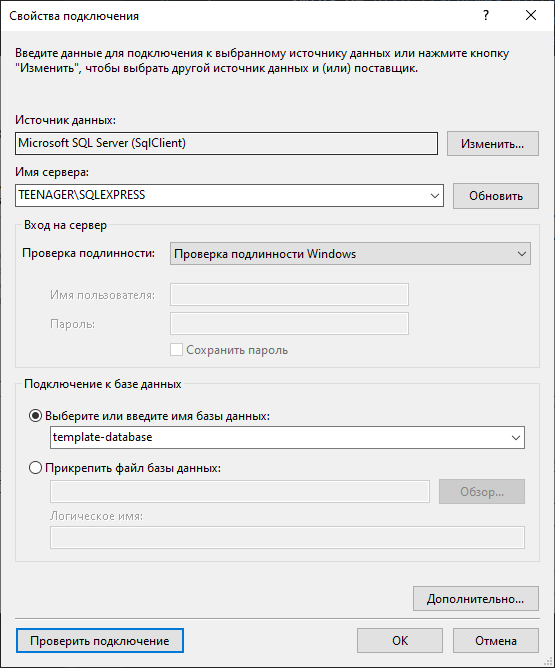


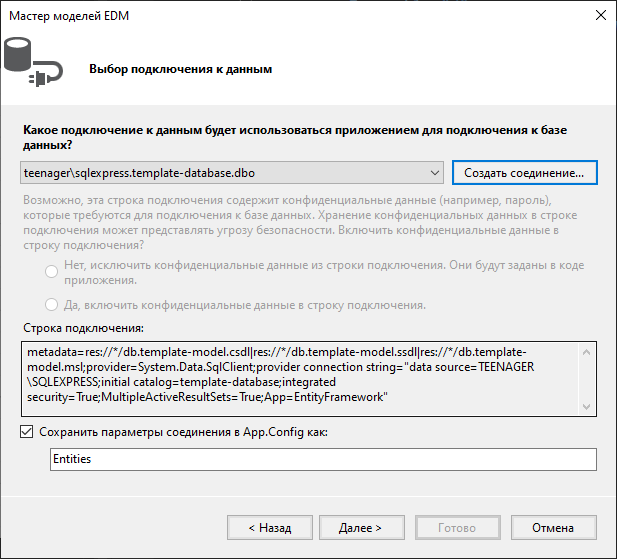


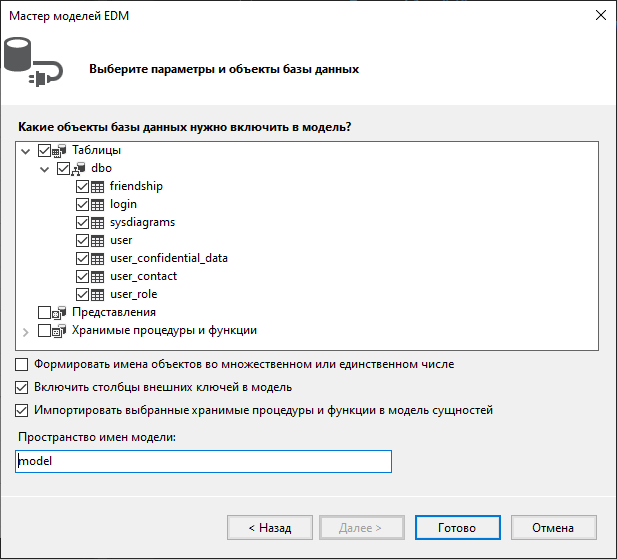
После подключения зависимостей переходим к созданию модели:



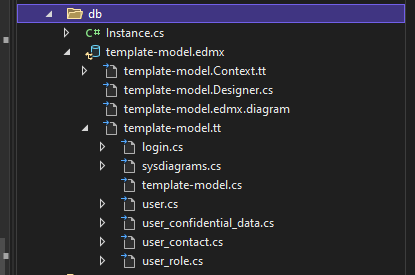








В итоге у вас должны будут создаться классы сущностей базы данных, с помощью которых вы будете обращаться к данным и подтягивать их с помощью LINQ. Как видите на рисунке снизу, нам необходимо создать класс со статическими методами позволяющими взаимодействовать с Entities (классы базы данных).



Instance.cs

internal class Instance

{

private static Entities entities;

public static Entities GetContext()

{

if (entities == null)

entities = new Entities();

return entities;

}

public static bool loginValidated(login login)

{

return !string.IsNullOrEmpty(login.login1) && !string.IsNullOrEmpty(login.password);

}

}

В данном классе мы реализуем паттерн синглтон, он правда вафельный, но тут он нам необходим, потому что позволит уменьшить количество созданий экземпляра базы данных, что повысит производительность. Также создаем статический метода для проверки корректности введенного логина пользователя при авторизации в системе.

Переходим к реализации функциональности авторизации.

Как вы уже заметили из разметки signin.xaml предоставленного ранее, то сначала разберемся с тем как сделать название внутри полей ввода. Все исходные текста вы можете просмотреть в репозитории под названием template-wpf приложенных к данному документу.

signin.cs

private login \_login;

public signin()

{

InitializeComponent();

logo\_app.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath("../../Resources/pic/network\_pic.png")));

\_login = new login();

}

В самом начале мы объявляем поле логина, в которой в будущем будем записывать данные, введенные пользователем в текст боксы. Далее идет конструктор окна авторизации и там инициализируем логотип и создаем экземпляр класса для нашего поля.

signin.cs

public void clearFieldBox()

{

tbLogin.Text = string.Empty;

tbPwd.Password = string.Empty;

tbLogin\_LostFocus(this, null);

tbPwd\_LostFocus(this, null);

}

Данный метод предназначен для очищения полей окна авторизации, это сделано для того, когда пользователь закрывает MainWindow и вызывается этот метод и очищает ранее введенные данные пользователем. Можно, конечно, избавиться от данной функции, потому что она создает излишки.

signin.cs

Код представленный ниже функции предназначаются для полей ввода данных.

private void tbPwd\_GotFocus(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(\_login.password))

{

tbPwd.Password = string.Empty;

tbPwd.Foreground = Brushes.Black;

}

}

private void tbPwd\_LostFocus(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(\_login.password))

{

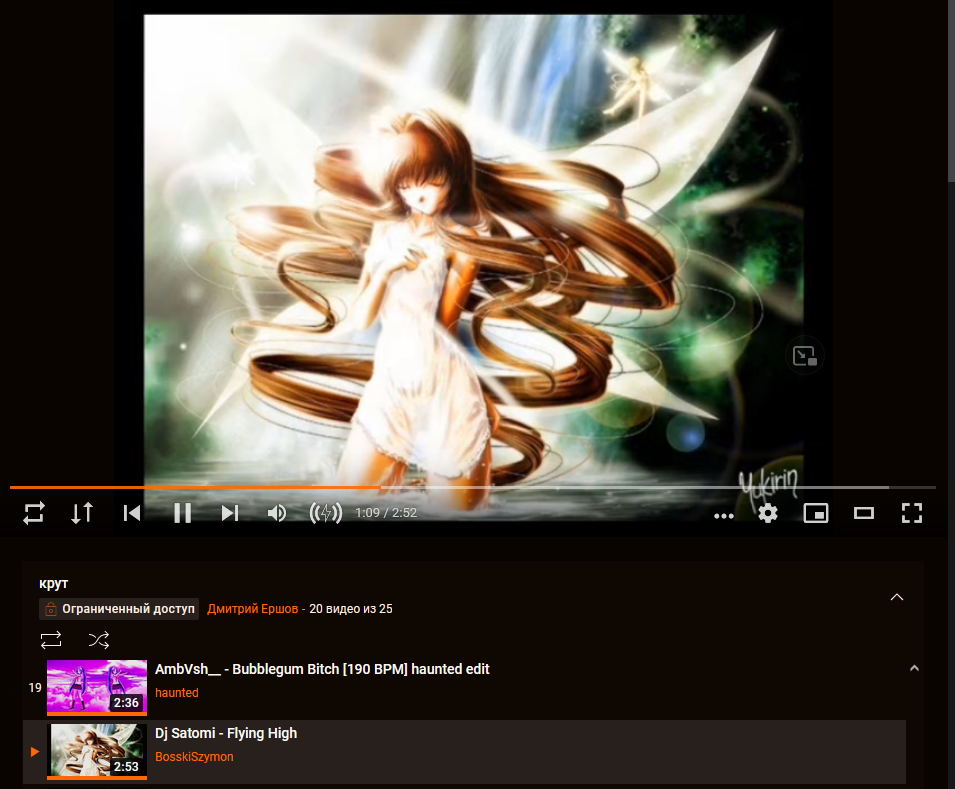
tbPwd.Password = $"password...";

\_login.password = string.Empty;

tbPwd.Foreground = Brushes.Gray;

}

}



Ой….

private void tbLogin\_LostFocus(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(\_login.login1))

{

tbLogin.Text = $"login...";

\_login.login1 = string.Empty;

tbLogin.Foreground = Brushes.Gray;

}

}

private void tbLogin\_GotFocus(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(\_login.login1))

{

tbLogin.Text = string.Empty;

tbLogin.Foreground = Brushes.Black;

}

}

Далее представлены исходные текста функций для отслеживания изменения текста внутри полей ввода.

private void tbLogin\_TextChanged(object sender, TextChangedEventArgs e)

{

if (\_login != null)

\_login.login1 = tbLogin.Text;

}

private void tbPwd\_PasswordChanged(object sender, RoutedEventArgs e)

{

if (\_login != null)

\_login.password = tbPwd.Password;

}

Далее представлены исходные текста кнопки авториазции.

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

{

if (!Instance.loginValidated(\_login))

{

MessageBox.Show("Необходимо заполнить все поля!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

if ((from l in Instance.GetContext().login

where l.login1 == \_login.login1 && l.password == \_login.password

select l).ToList().Count() != 1)

{

MessageBox.Show("Не удалось авторизироваться!\nЛогин или пароль не корректны.", "Ошибка авторизации", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

Hide();

MainWindow window = new MainWindow(\_login);

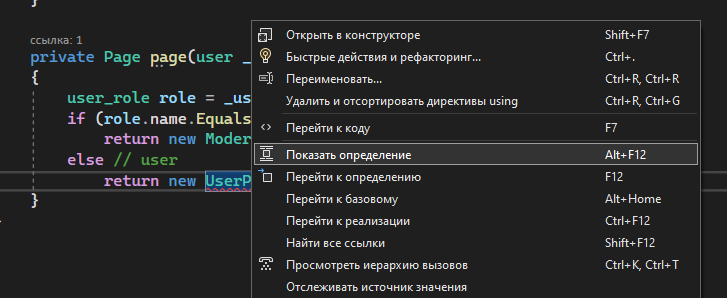
window.Owner = this;

window.Show();

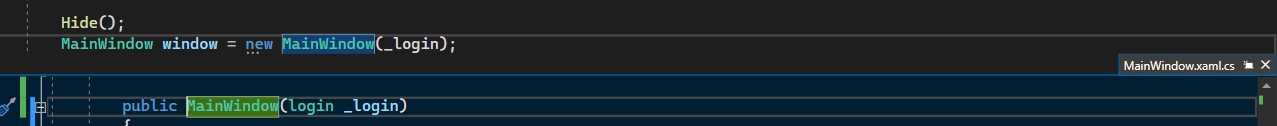
}

В данном методе мы сперва проверяем на корректность логин пользователя, если он не корректен, то выводим диалоговое окно с пояснительным комментарием ошибки и выходим из функции. Если же логин корректен, тогда ищем его в базе данных и если там не одна запись или вообще ее нет, то выводим диалоговое окно с пояснительным комментарием ошибки, в противном случаем все данные верны и существует только один такой логин, тогда мы сворачиваем окно авторизации и создаем экземпляр класса MainWindow и устанавливаем ей свойство Owner и показываем окно.

Для успешной компиляции проекта вам необходимо изменить конструктор класса окна MainWindow. Это можно сделать с помощью такого хука.

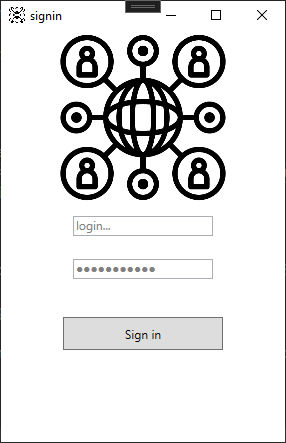


Нажимаем и изменяем параметры класса, правда тут уже я забегаю в будущее, не переживайте мы еще до туда дойдем. А что вам надо продемонстрировано на рисунке ниже.

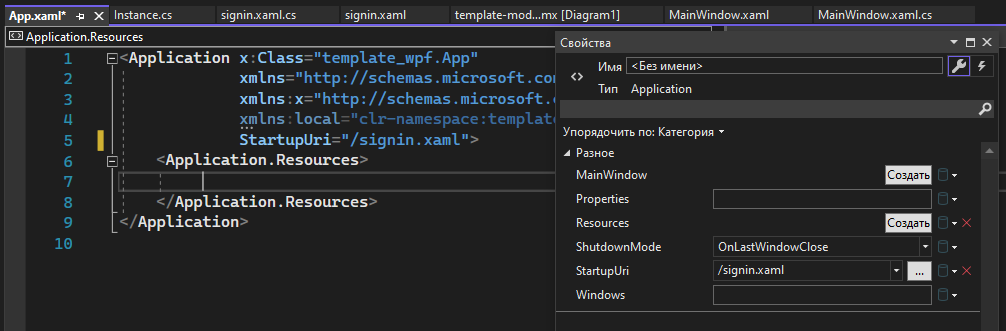


Не забудьте добавить using где располагается класс login.

Итогом копипаста у вас должно получиться следующее окно авторизации.



Но она у вас не запуститься, потому что я вас заскамил и забыл упомянуть один важный факт. Когда вы захотите поменять окно для запуска приложения, то вам следует перейти в файл App.xaml и в нем поменять значение поля StartupUri на необходимое окно. Советую вам изменить это поле в свойствах.



Ладно, пошутили и продолжим дальше шутить, красавцы те, кто допер до сюда, ну я думаю вы нихуя не читаете, ибо кто читает бред шизоида, а пасхалок будет куча, хоть уже почти конец, хотя конец еще не за горами.

Ладно, мы крутые, а теперь перейдем к модернизации нашего основного окна MainWindow.

MainWindow.xaml

<Window x:Class="template\_wpf.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:template\_wpf"

xmlns:common="clr-namespace:template\_wpf.common"

mc:Ignorable="d"

Title="MainWindow" Height="500" Width="800"

MinHeight="500" MinWidth="800"

Closing="Window\_Closing"

>

<Window.Resources>

<ResourceDictionary>

<common:ConvertUserName x:Key="ConvertUserName"/>

</ResourceDictionary>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50"

MaxWidth="50" MinWidth="50"/>

<ColumnDefinition Width="0.9\*"/>

<ColumnDefinition Width="75"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="50"

MaxHeight="50" MinHeight="50"/>

<RowDefinition Height="0.9\*"/>

</Grid.RowDefinitions>

<Rectangle Grid.ColumnSpan="3"

Fill="BlueViolet"/>

<Image x:Name="logoUser"

Grid.Column="0" Grid.Row="0"

Stretch="Uniform"

/>

<TextBlock x:Name="tbNameUser"

Text="{Binding Path=DataContext, RelativeSource={RelativeSource Self}, Converter={StaticResource ConvertUserName}}"

Grid.Column="1" Grid.Row="0"

FontSize="20" TextAlignment="Left"

Margin="10"

VerticalAlignment="Center"

/>

<Button x:Name="exitBtn"

Content="Выход"

Background="LightPink"

Grid.Column="2" Grid.Row="0"

Margin="5, 10"

Click="WindowClose"

/>

<Frame x:Name="PageFrame"

MinHeight="300" MinWidth="600"

Grid.Column="0" Grid.ColumnSpan="3"

Grid.Row="1" Grid.RowSpan="1"

Content="{Binding CurrentPage}"

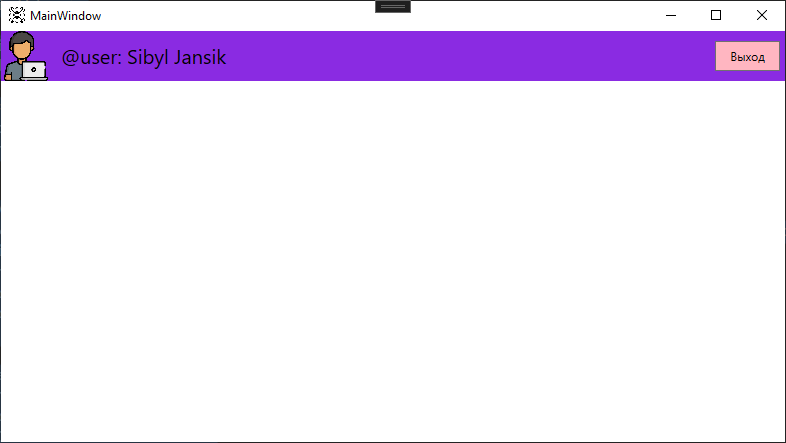
NavigationUIVisibility="Hidden"

/>

</Grid>

</Window>

У вас выйдет вот такое окно:



Концепция нашего окна и последующих созданных окон – будут иметь страницы, занимающие большую полезную область самой формы. На данном окне будет отображаться картинка пользователя, его никнейм с указанием названия роли, кнопка выхода и страница, занимающая всю нижнею часть окна.

Для поддержки концепции страничного приложения будем использовать по кд следующий код:

MainWindow.cs

private Page \_currentPage;

public Page CurrentPage

{

get { return \_currentPage; }

set

{

if (\_currentPage == value)

return;

\_currentPage = value;

OnPropertyChanged("CurrentPage");

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

Он нам позволяет реализовать смену страниц формы и хранения ее. Меньше вопросов больше без осмысленного копипаста. Если интересно что это, то гуглите.

Ладно, перейдем к конструктору нашего окна

private List<string> userRoleNameList;

public MainWindow(login \_login)

{

InitializeComponent();

DataContext = Instance.GetContext().user.Where(p => p.login == \_login.login1).FirstOrDefault();

userRoleNameList = Instance.GetContext().user\_role.Select(p => p.name).ToList();

logoUser.Source = pictureLogo(((user)DataContext).user\_role);

CurrentPage = page((user)DataContext);

PageFrame.Content = CurrentPage;

}

Конструктор окна у нас ищет в базе данных пользователя и присваивает его DataContext окна, из которого в будущем будут подтягиваться данные и изменяться, далее инициализируем список названий ролей, чтобы не харкодить названия ролей в коде. Затем создаем картинку используя метод pictureLogo принимающее роль пользователя или можно будет подтягивать из базы данных само название картинки из таблица привязанной к пользователю. Затем мы создаем страницу по роли пользователя.

Далее представлены методы, которые вызывает конструктор окна. Господа, учитесь читать код, пожалуйста, я реально сейчас зашьюсь, если буду так построчно пояснять.

private ImageSource pictureLogo(user\_role role)

{

return new BitmapImage(new Uri(System.IO.Path.GetFullPath($"../../Resources/pic/{role.name}.png")));

}

private Page page(user \_user)

{

user\_role role = \_user.user\_role;

if (role.name.Equals(userRoleNameList.ElementAt(0)))

return new ModeratorPage(\_user, this);

else // user

return new UserPage(\_user, this);

}

Вот последние два метода нашего основного окна.

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

Owner.Show();

(Owner as signin).clearFieldBox();

}

private void WindowClose(object sender, RoutedEventArgs e)

{

Close();

}

Как вы могли заметить из xaml файла вам не хватает одного класса из common, а именно как конвертировать объект к нужному типу в Binding. Буду краток, создайте класс в common под названием «ConvertUserName.cs» и реализуйте там интерфейс IValueConverter.

[ValueConversion(typeof(user), typeof(string))]

public class ConvertUserName : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

user employee = (user)value;

if (employee == null)

return "";

return $"@{employee.user\_role.name.Trim()}: {employee.surname.Trim()} {employee.name.Trim()} {(employee.midname == null ? "" : employee.midname.Trim())}";

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

return value;

}

}

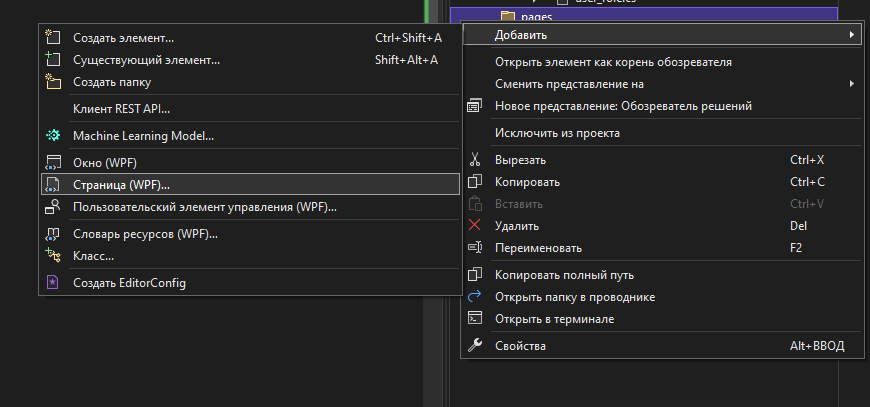
Тут мы просто конвертируем тип и возвращаем нужный нам формат данных, такая тема работает со всеми Binding’ами для любых пользовательских элементов. Ну и теперь можем запускать сборку и радоваться жизни.

Заебись, все сделали, теперь можем радоваться крутому ентерпрайс приложению, разрешаю позапускать по вводить данные логинов и паролей из базы данных или из файла Excel. Круто, кто молодец? Правильно, вы молодцы!!!

Глава нашей супер-пупер истории 2: шести-головый змей горыныч.

Прежде чем бросаться освобождать Фиону, Шреку и Ослу (спойлер: это шиза Шрека) необходимо подготовиться к битве с горынычем, а именно реализовать две страницы для модератора и пользователя.

В итоге вам потребуется создать две страницы внутри поддиректории проекта «pages» с названиями «ModeratorPage.xaml» и «UserPage.xaml».



Представляю вашему вниманию разметку ModeratorPage.xaml

<Page x:Class="template\_wpf.pages.ModeratorPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:template\_wpf.pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ModeratorPage">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.2\*"/>

<ColumnDefinition Width="0.5\*"/>

<ColumnDefinition Width="0.3\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition />

</Grid.RowDefinitions>

<DataGrid x:Name="userTabel"

AutoGenerateColumns="False"

Grid.Column="1"

>

<DataGrid.Columns>

<DataGridTextColumn Header="Login" Binding="{Binding login}"/>

<DataGridTextColumn Header="Role" Binding="{Binding user\_role.name}"/>

<DataGridTextColumn Header="Surname" Binding="{Binding surname}"/>

<DataGridTextColumn Header="Name" Binding="{Binding name}"/>

<DataGridTextColumn Header="Midname" Binding="{Binding midname}"/>

<DataGridTextColumn Header="Bithday" Binding="{Binding bithday}"/>

</DataGrid.Columns>

</DataGrid>

<Frame x:Name="UserCardFrame"

MinHeight="300" MinWidth="500"

Grid.Column="2" Grid.Row="1"

Content="{Binding UserCardCurrentWidget}"

NavigationUIVisibility="Hidden"

/>

<StackPanel Grid.Column="0"

>

<Button x:Name="AddUser"

Content="Добавить пользователя"

Grid.Column="0" Grid.ColumnSpan="2"

Grid.Row="1"

Margin="5" MinHeight="50"

Click="AddUser\_Click"

/>

</StackPanel>

</Grid>

</Page>

Если вы помните, как мы делали в MainWindow.xaml, то вам не составит труда понять, что у нас тут также будут страницы, и я слегка повыебываюсь реализаций карточек пользователя, где будут отображаться их данные. Ну и будет выведен список пользователей, зарегистрированных в системе через DataGrid.

Ладно погнали подтягивать данные к нашим и отрубать одну голову, а может несколько, горыныча.

ModeratorPage.xaml.cs

private Page \_userCardCurrentWidget;

public Page UserCardCurrentWidget

{

get { return \_userCardCurrentWidget; }

set

{

if (\_userCardCurrentWidget == value)

return;

\_userCardCurrentWidget = value;

OnPropertyChanged("CurrentPage");

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

Как вы уже догадались, берем и копируем из MainWindow и заменяем на нормальные названия, повторюсь, этот код предназначен для того, чтобы хранить и обрабатывать события смены страницы, но чет я сейчас понял, что и без нее можно было нормально обойтись, но если вы модифицируете внутри этого кода присвоение к PageFrame.Content = value; то данный код будет осмыслен, подумайте об этом, и смотрите что копируете.

Window parentWindow;

public ModeratorPage(user \_user, Window parent)

{

InitializeComponent();

DataContext = \_user;

parentWindow = parent;

updateState();

UserCardCurrentWidget = new UserCardWidget(this);

UserCardFrame.Content = UserCardCurrentWidget;

}

Вот и конструктор страницы модератора подъехал, что тут можно сказать, если бы предыдущий кусок кода мы бы модифицировали как надо то последнюю строчку можно было не писать, потому что она была бы внутри того setера.

public void updateState()

{

Instance.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

userTabel.ItemsSource = Instance.GetContext().user.Where(p => !p.login.Equals(((user)DataContext).login)).ToList();

}

UpdateState нам необходим для обновления нашей таблицы при как-либо изменениях, мы будем ее вызывать, чтобы уменьшить дублирование кода.

У нас такая функциональность, как обговаривалась смысловая нагрузка модератора, то он должен просматривать данные пользователя из карточек и добавлять или удалять любого пользователя.

Чтобы отображать выбранного пользователя из таблицы мы реализуем метод повешенный на свойство SelectionChanged.

private void userTabel\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (userTabel.SelectedItem == null)

return;

user employee = (user)userTabel.SelectedItem;

if (employee == null)

return;

(UserCardCurrentWidget as UserCardWidget).setUser(employee);

}

Смысл данной функции при выборе строки в таблице, нам отображается карточка пользователя с права от таблицы, уже на которой есть функционал понизить/повысить роль и удалить пользователя.

Этот мув обойдется нам в несколько сотен тысяч лет, но похуй.

Представляю вашему вниманию UserCardWidget.xaml, мы его создаем в отдельной папочке «widgets».

<Page x:Class="template\_wpf.widgets.UserCardWidget"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:template\_wpf.widgets"

xmlns:common="clr-namespace:template\_wpf.common"

mc:Ignorable="d"

Height="400" Width="200"

MinHeight="400" MinWidth="200"

Title="UserCardWidget">

<Page.Resources>

<ResourceDictionary>

<common:ConvertUserName x:Key="ConvertUserName"/>

</ResourceDictionary>

</Page.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="0.05\*"/>

<RowDefinition Height="0.3\*"/>

<RowDefinition Height="0.05\*"/>

<RowDefinition Height="0.05\*"/>

<RowDefinition Height="0.05\*"/>

<RowDefinition Height="0.05\*"/>

<RowDefinition Height="0.05\*"/>

<RowDefinition Height="0.3\*"/>

</Grid.RowDefinitions>

<Image x:Name="pictureUser"

Grid.Row="1"

Stretch="Uniform"

/>

<TextBlock x:Name="tbName"

Text="{Binding Path=DataContext, RelativeSource={RelativeSource Self}, Converter={StaticResource ConvertUserName}}"

Grid.Row="3"

TextWrapping="Wrap"

/>

<TextBlock x:Name="tbBithday"

Text="{Binding bithday}"

Grid.Row="4"

TextWrapping="Wrap"

/>

<TextBlock x:Name="tbEmail"

Text="{Binding user\_contact.email}"

Grid.Row="5"

TextWrapping="Wrap"

/>

<TextBlock x:Name="tbPhone"

Text="{Binding user\_contact.phone}"

Grid.Row="6"

TextWrapping="Wrap"

/>

<StackPanel Grid.Row="7">

<Button x:Name="UpBtn"

Content="Повысить пользователя"

Margin="5" MinHeight="30"

Click="UpBtn\_Click"

/>

<Button x:Name="DownBtn"

Content="Понизить пользователя"

Margin="5" MinHeight="30"

Click="DownBtn\_Click"

/>

<Button x:Name="RemoveBtn"

Content="Удалить пользователя"

Margin="5" MinHeight="30"

Background="DarkRed"

Foreground="White"

Click="RemoveBtn\_Click"

/>

</StackPanel>

</Grid>

</Page>

Ну и чтобы долго не затягивать, перейдем к коду данного виджета. Заметьте мы тут используем конвертер, который ранее разработали.

Page parentPage;

List<user\_role> roles;

public UserCardWidget(Page \_parentPage)

{

InitializeComponent();

parentPage = \_parentPage;

roles = Instance.GetContext().user\_role.ToList();

updateState();

}

Конструктор у нас типичный, здесь мы просто инициализируем данными все необходимые нам элементы управления из разметки. И вызываем метод updateState, много где используется данный подход создания метода ответственного за обновление состояния графическими и данными внутри класса.

private void updateState()

{

user employee = (user)DataContext;

pictureUser.Visibility = employee != null ? Visibility.Visible : Visibility.Hidden;

UpBtn.Visibility = employee != null ? Visibility.Visible : Visibility.Hidden;

DownBtn.Visibility = employee != null ? Visibility.Visible: Visibility.Hidden;

RemoveBtn.Visibility = employee != null ? Visibility.Visible : Visibility.Hidden;

if (employee != null && employee.user\_role != null)

{

UpBtn.IsEnabled = employee.user\_role != roles.FirstOrDefault();

DownBtn.IsEnabled = employee.user\_role != roles.Last();

pictureUser.Source = new BitmapImage(new Uri(System.IO.Path.GetFullPath($"../../Resources/pic/{employee.user\_role.name}.png")));

}

}

Вот, собственно, и данный метод, тут мы изменяем состояние Visibility и IsEnabled для сокрытия или блокировки графических элементов виджета. Также устанавливаем картинки пользователя.

Вспомним эту строчку из нашего ModeratorPage.xaml.cs SelectionChanged метода:

(UserCardCurrentWidget as UserCardWidget).setUser(employee);

При выборе нужной строки мы устанавливаем полученного пользователя из нее. И этот метод setUser будет сейчас продемонстрирован.

public void setUser(user \_user)

{

DataContext = \_user;

updateState();

}

Ничего не обычного, просто обновляем контекст виджета и обновляем состояние.

Обсудим наш функционал по повышению/понижению роли выбранного пользователя.

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

{

changeRole(1);

}

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

{

changeRole(-1);

}

Разглядев эти методы, вы скажете, ну ты что не все то показываешь, а я возьму и сделаю следующее:

private void changeRole(int isUp)

{

user employee = (user)DataContext;

if (employee != null)

{

int index = roles.IndexOf(employee.user\_role);

if (index + isUp < 0 || index + isUp >= roles.Count)

return;

user\_role upRole = roles.ElementAt(index + isUp);

employee.code\_role = upRole.code\_role;

employee.user\_role = upRole;

Instance.GetContext().SaveChanges();

updateState();

(parentPage as ModeratorPage).updateState();

}

}

Благодаря нашим крутым спискам ролей, мы можем спокойно определить можем ли мы понизить пользователя или повысить в роли. Собственно код воть. Смотрите внимательно, ведь мы там вызываем эти методы:

Instance.GetContext().SaveChanges();

updateState();

(parentPage as ModeratorPage).updateState();

Первый метод сохраняет изменения в базе данных, последующие обновляют состояния виджета и страницы соответственно.

Осталась лишь последняя функциональность – удаление пользователя.

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

{

user removeUser = (user)DataContext;

login removeLogin = removeUser.login1;

if (removeUser.user\_contact != null)

Instance.GetContext().user\_contact.Remove(removeUser.user\_contact);

if (removeUser.user\_confidential\_data != null)

Instance.GetContext().user\_confidential\_data.Remove(removeUser.user\_confidential\_data);

Instance.GetContext().user.Remove(removeUser);

if (removeLogin != null)

Instance.GetContext().login.Remove(removeLogin);

Instance.GetContext().SaveChanges();

updateState();

(parentPage as ModeratorPage).updateState();

}

При нажатии на кнопку удаления на виджете, мы получаем пользователя из контекста виджета и потом потихоньку удаляем его сущности. В самом конце повторяем сохранение и обновление состояния.

Теперь вернемся к нашим коровам во хлеву где-то под майданом.

Добавление пользователя:

В мою дурную шизойдную башку пришло решение не слишком приличное, но универсальное. Концепция заключается в том, чтобы создать CommonWindow, которое будет отображать необходимую страницу, внутри которой будет реализовываться любая функциональность. Перейдем к методу добавления.

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

{

CommonWindow window = new CommonWindow();

AddNewUserWidget widget = new AddNewUserWidget(window);

window.setPage(widget);

window.ShowDialog();

updateState();

}

Все просто, создаем вспомогательное окно, затем создаем нужный нам виджет и устанавливаем виджет в окно. Ну а дальше используем ShowDialog(), он позволяет заблочить продолжение выполнения кода в данном методе пока созданное вспомогательное окно не будет закрыто. Поэтому наш метод обновления состояния не пропадает зря. Вам пока везет, крышу мне не сносит пока что)))

Этим методом у нас заканчивается ModeratorPage.xaml.cs. И потихоньку мы перейдем к CommonWindow и в дальнейшем разберем AddNewUserWidget. Конечно, я хотел бы вам дропнуть просто голые исходники, но я человек.

<Page x:Class="template\_wpf.widgets.AddNewUserWidget"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:template\_wpf.widgets"

xmlns:common="clr-namespace:template\_wpf.common"

mc:Ignorable="d"

Title="AddNewUserWidget"

Height="600" Width="300"

MinHeight="600" MinWidth="300"

>

<Page.Resources>

<ResourceDictionary>

<common:ConvertUserRole x:Key="ConvertUserRole"/>

</ResourceDictionary>

</Page.Resources>

<Grid Margin="10">

<Grid.ColumnDefinitions>

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.1\*"/>

<RowDefinition Height="0.15\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0"

VerticalAlignment="Center"

>

<TextBlock Text="Enter login"/>

<TextBox x:Name="login"

Text="{Binding login}"

/>

</StackPanel>

<StackPanel Grid.Row="1"

VerticalAlignment="Center"

>

<TextBlock Text="Enter password"/>

<TextBox x:Name="password"

Text="{Binding login1.password}"

/>

</StackPanel>

<StackPanel Grid.Row="2"

VerticalAlignment="Center"

>

<TextBlock Text="Enter user name"/>

<TextBox x:Name="username"

Text="{Binding name}"

/>

</StackPanel>

<StackPanel Grid.Row="3"

VerticalAlignment="Center"

>

<TextBlock Text="Enter user surname"/>

<TextBox x:Name="usersurname"

Text="{Binding surname}"

/>

</StackPanel>

<StackPanel Grid.Row="4"

VerticalAlignment="Center"

>

<TextBlock Text="Chose user role"/>

<ComboBox x:Name="userRoleCB">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Converter={StaticResource ConvertUserRole}}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

</StackPanel>

<StackPanel Grid.Row="5"

VerticalAlignment="Center"

>

<TextBlock Text="Enter phone"/>

<TextBox x:Name="phone"

Text="{Binding user\_contact.phone}"

/>

</StackPanel>

<StackPanel Grid.Row="6"

VerticalAlignment="Center"

>

<TextBlock Text="Enter email"/>

<TextBox x:Name="email"

Text="{Binding user\_contact.email}"

/>

</StackPanel>

<StackPanel Grid.Row="7"

VerticalAlignment="Center"

>

<TextBlock Text="Enter passport series"/>

<TextBox x:Name="passport\_series"

Text="{Binding user\_confidential\_data.passport\_series}"

/>

</StackPanel>

<StackPanel Grid.Row="8"

VerticalAlignment="Center"

>

<TextBlock Text="Enter passport number"/>

<TextBox x:Name="passport\_number"

Text="{Binding user\_confidential\_data.passport\_id}"

/>

</StackPanel>

<StackPanel Grid.Row="9"

VerticalAlignment="Center"

>

<TextBlock Text="Enter policy number"/>

<TextBox x:Name="policy\_number"

Text="{Binding user\_confidential\_data.insurance\_policy\_number}"

/>

</StackPanel>

<StackPanel Grid.Row="10"

VerticalAlignment="Center"

>

<TextBlock Text="Enter policy type"/>

<TextBox x:Name="policy\_type"

Text="{Binding user\_confidential\_data.insurance\_policy\_type}"

/>

</StackPanel>

<StackPanel Grid.Row="11"

VerticalAlignment="Center"

>

<TextBlock Text="Enter insurance company code"/>

<TextBox x:Name="policy\_code\_company"

Text="{Binding user\_confidential\_data.insurance\_company\_code}"

/>

</StackPanel>

<Button x:Name="addBtn"

Content="Добавить"

Grid.Row="12"

Margin="5, 10"

Click="addBtn\_Click"

/>

</Grid>

</Page>

Круто, две с половиной страницы разметки, каеф. Прошу вас заметить, мы создаем новый конвертер, который продемонстрирован ниже.

[ValueConversion(typeof(user\_role), typeof(string))]

public class ConvertUserRole : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

user\_role role = (user\_role)value;

if (role == null)

return "";

return $"#{role.code\_role}:{role.name}";

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

return value;

}

}

Данный конвертер мы используем для преобразование user\_role в строковое представление, для дальнейшего использования внутри ComboBox или TextBlock.

Ладно, погнали к нашему конструктору настоящего виджета.

Window parentWindow;

public AddNewUserWidget(Window parent)

{

InitializeComponent();

parentWindow = parent;

user \_user = new user()

{

login1 = new login(),

user\_contact = new user\_contact(),

user\_confidential\_data = new user\_confidential\_data()

};

DataContext = \_user;

userRoleCB.ItemsSource = Instance.GetContext().user\_role.ToList();

}

Ничего не обычного, у виджетов, страниц, окон всегда будут подобные конструкторы и они всегда будут схожи, это правило сделает вашу систему более понятнее и интуитивное, когда она знает о своем предке и о данных необходимых для ее функционирования.

Кратко: при создании виджета, мы создаем и инициализируем нового пользователя и все необходимые ему поля. Также заполняем ComboBox существующими ролями.

Как могли заметить из файла разметки, у нас еще есть кнопка добавить, а вот ее код:

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

{

user newUser = (user)DataContext;

newUser.login1.login1 = newUser.login;

newUser.login1.user = newUser;

newUser.user\_role = userRoleCB.SelectedItem as user\_role;

newUser.code\_role = newUser.user\_role.code\_role;

newUser.user\_role.user.Add(newUser);

newUser.user\_contact.user = newUser;

newUser.user\_confidential\_data.user = newUser;

Instance.GetContext().login.Add(newUser.login1);

Instance.GetContext().user\_contact.Add(newUser.user\_contact);

Instance.GetContext().user\_confidential\_data.Add(newUser.user\_confidential\_data);

Instance.GetContext().user.Add(newUser);

Instance.GetContext().SaveChanges();

if (Instance.GetContext().user.Where(p => p.login == newUser.login).FirstOrDefault() != null)

{

MessageBox.Show("Пользователь успешно добавлен", "Добавление пользователя", MessageBoxButton.OK, MessageBoxImage.Information);

parentWindow.Close();

}

else

MessageBox.Show("Пользователя не удалось добавить", "Добавление пользователя", MessageBoxButton.OK, MessageBoxImage.Error);

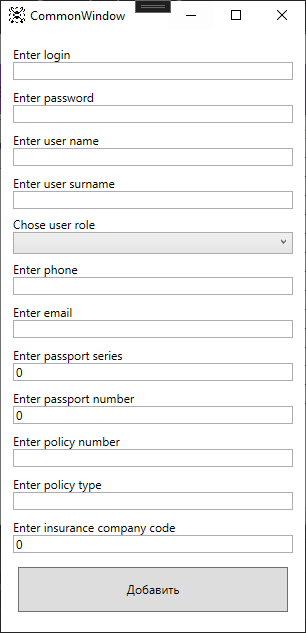
}

Ничего не обычного, мы просто заполняем пустые поля нового пользователя данными из его полей экземпляров классов, чтобы обеспечить полноту информации. Дальше происходит добавление во все нужные таблички информации, сохраняемся. На проверке мы убеждаемся создалась ли по итогу новая запись в базе данных и если все круто, то выводим сообщение об успешном добавлении пользователя и закрываем CommonWindow. Если все плохо, выводим соответствующее сообщение и не закрываем родительское окно.

На этом виджет добавление нового пользователя подошел к концу, и пора бы показать, что в итоге получилось.

На следующей странице посмотрите.

Ой дурак, виджет разобрали, а окно нет, сейчас исправим.



CommonWindow.xaml.cs состоит из нашего любимого CurrentPage и типичного конструктора.

private Page \_currentPage;

public Page CurrentPage

{

get { return \_currentPage; }

set

{

if (\_currentPage == value)

return;

\_currentPage = value;

OnPropertyChanged("CurrentPage");

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

Просто вставил, чтобы был. Сами знаете что там исправить необходимо.

public CommonWindow()

{

InitializeComponent();

}

public void setPage(Page page)

{

CurrentPage = page;

PageFrame.Content = CurrentPage;

windowToFitPage();

}

private void windowToFitPage()

{

if (CurrentPage.Width >= Width)

Width = CurrentPage.Width + 20;

if (CurrentPage.Height >= Height)

Height = CurrentPage.Height + 40;

}

Вот код, да немного, зато как универсально и кайфово. Когда мы устанавливаем страницу через setPage, то мы должны быть уверены что страница не больше размера окна, поэтому создаем метод windowToFitPage, в котором изменяем размеры окна по необходимости. +20 и +40 это из-за виндовских окон, кто понял тот понял, отсылка к WinForms.

Ну собственно код мы посмотрели, теперь давайте взглянем на разметку, как вы уже догадались нам позарез необходим PageFrame.

CommonWindow.xaml

<Window x:Class="template\_wpf.common.CommonWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:template\_wpf.common"

mc:Ignorable="d"

Title="CommonWindow"

Height="350" Width="300"

MinHeight="350" MinWidth="300"

>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition />

</Grid.RowDefinitions>

<Frame x:Name="PageFrame"

MinHeight="300" MinWidth="300"

Grid.Row="0"

Content="{Binding CurrentPage}"

NavigationUIVisibility="Hidden"

/>

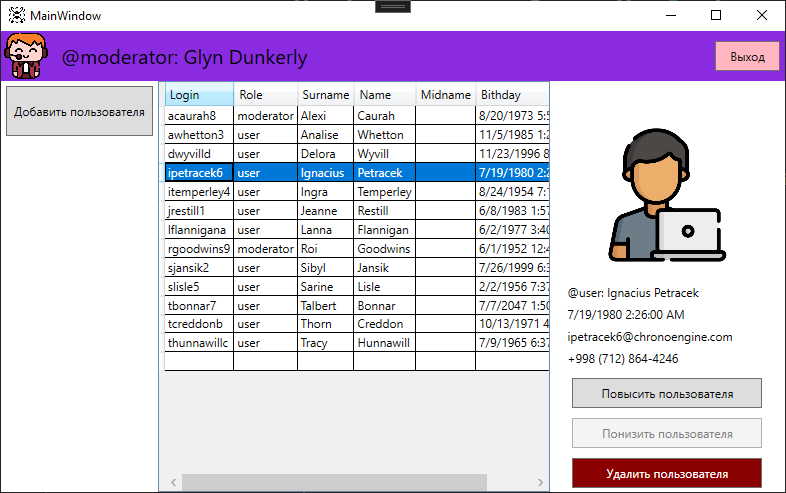
</Grid>

</Window>

Ничего крутого и стильного, да стилям вас научит Егор Олегович на 3 курсе или на 4, кому как повезет.

Видимо конец функциональности модератора пришел, и змей горыныч потерял половину своих голов. Ура, товарищи, круто справляемся. Молодцы, что не читаете бред шизоида и просто копипастите. Не тратьте на фигню ваше драгоценное время. Если чувствуете, что живете не для себя, то пора что-то менять в своей жизни.

В итоге у нас должна получиться вот такая красивая картинка:



Да, я, конечно, не дезигнер, но чувство функциональности имею.

Вот и глава подошла к концу, а кто слушал не молодец, а кто… да ладно вам, шучу, все молодцы, респект таким типам, которые смотрят работы ноунеймов из интернета.

Слева, где кнопочки, там можно и другие кнопокчки добавлять для расширения функциональности модератора. Собственно, много чего можно добавить, но мне лень. Основу выдам и пойду в дурку. Только не забудьте запретить изменение таблички, там есть свойство, нагуглите я думаю.

Глава истории 3: в дали от дурки.

Как мы знаем из наших функциональных требований, нам необходимо выводить пользователей, находящихся в друзьях у нашего товарища Сталина и производить сортировку/фильтрацию/поиск товарищей Броцких и прочих. А также добавлять новых товарищей, за которыми наш многоуважаемый товарищ Сталин, будет наблюдать.

UserPage.xaml

<Page x:Class="template\_wpf.pages.UserPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:template\_wpf.pages"

xmlns:common="clr-namespace:template\_wpf.common"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="UserPage">

<Page.Resources>

<ResourceDictionary>

<common:ConvertUserName x:Key="ConvertUserName"/>

<common:ConvertUserRole x:Key="ConvertUserRole"/>

<common:ConvertImage x:Key="ConvertImage"/>

<common:ConvertUserData x:Key="ConvertUserData"/>

</ResourceDictionary>

</Page.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.2\*"/>

<ColumnDefinition Width="0.5\*"/>

<ColumnDefinition Width="0.3\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="0.05\*"/>

<RowDefinition Height="0.05\*"/>

<RowDefinition Height="0.8\*"/>

</Grid.RowDefinitions>

<TextBox x:Name="searchTB"

Grid.Column="1" Grid.RowSpan="2"

VerticalAlignment="Center"

Margin="20, 0"

TextChanged="searchTB\_TextChanged"

/>

<ComboBox x:Name="FilterCB"

Grid.RowSpan="2"

VerticalAlignment="Center"

Margin="10, 0"

SelectionChanged="FilterCB\_SelectionChanged"

>

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Converter={StaticResource ConvertUserRole}}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<CheckBox x:Name="NameUpCheckB"

Grid.Column="2"

Content="Имена по возрастанию"

Checked="NameUpCheckB\_Checked"

/>

<CheckBox x:Name="NameDownCheckB"

Grid.Column="2" Grid.Row="1"

Content="Имена по убыванию"

Checked="NameUpCheckB\_Checked"

/>

<ListBox x:Name="userTabel"

Grid.Column="1" Grid.ColumnSpan="3"

Grid.Row="3"

>

<ListBox.ItemTemplate>

<DataTemplate>

<Grid Margin="5" MaxHeight="100" MaxWidth="400">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.1\*" />

<ColumnDefinition Width="0.2\*" />

<ColumnDefinition Width="0.1\*" />

<ColumnDefinition Width="0.1\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Image x:Name="LogoUser"

Grid.Column="0" Grid.RowSpan="2"

Source="{Binding Path=user1, Converter={StaticResource ConvertImage}}"

/>

<TextBlock Grid.Column="1" Grid.Row="0"

Text="{Binding Path=user1, Converter={StaticResource ConvertUserName}}"

/>

<TextBlock Grid.Column="1" Grid.Row="1"

Text="{Binding Path=user1, Converter={StaticResource ConvertUserData}}"

/>

<TextBlock Grid.Column="2" Grid.Row="0"

Text="{Binding user1.user\_contact.email}"

/>

<TextBlock Grid.Column="2" Grid.Row="1"

Text="{Binding user1.user\_contact.phone}"

/>

<Button x:Name="BtnRemove"

Tag="{Binding user1.login}"

Content="Удалить из друзей"

Grid.Column="3" Grid.RowSpan="2"

Click="BtnRemove\_Click"

/>

</Grid>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<StackPanel Grid.Column="0"

Grid.Row="3" Grid.RowSpan="2"

>

<Button x:Name="AddFriendShip"

Content="Добавить друга"

Margin="5" MinHeight="50"

Click="AddFriendShip\_Click"

/>

</StackPanel>

</Grid>

</Page>

Попрошу вас обратить внимание на количество конвертеров! Да 4 штуки, да уже 2 написали, и сейчас узнаем, что храниться в оставшихся двух.

[ValueConversion(typeof(user), typeof(string))]

public class ConvertUserData : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

user employee = (user)value;

if (employee == null)

return "";

return $"@{(employee.bithday == null ? "" : employee.bithday.ToString())}:{employee.login.Trim()}";

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

return value;

}

}

Все просто, используем этот конвертер для конвертирования дополнительной информации, чтобы ее отображать в ListBox.

Мне лень было что-то крутое конвертировать, просто объединим два поля в одну строку и возрадуемся богам, что все у нас работает.

[ValueConversion(typeof(user), typeof(ImageSource))]

public class ConvertImage : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

user employee = (user)value;

if (employee == null)

return null;

return new BitmapImage(new Uri(System.IO.Path.GetFullPath($"../../Resources/pic/{employee.user\_role.name}.png")));

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

return value;

}

}

Кстати, чтобы заработал конвертер внутри файла разметки (не ругалась студия), то вам необходимо собрать проект и все будет у вас замечательно. Второй конвертер мы используем для того, чтобы приводить картинки по роли пользователя, конечно, не лучшая концепция и реализация, но такой контекст и ситуация.

Window parentWindow;

public UserPage(user \_user, Window parent)

{

InitializeComponent();

DataContext = \_user;

parentWindow = parent;

InitRolesCB();

updateData();

}

Наш крутой конструктор, вы скажете да хватит уже не показывать все, а я сделаю так:

private void updateData()

{

var friends = (DataContext as user).friendship.ToList();

if (FilterCB.SelectedIndex > 0)

friends = friends.Where((item) => item.user1.user\_role == ((user\_role)FilterCB.SelectedItem)).ToList();

if ((bool)NameUpCheckB.IsChecked && !(bool)NameDownCheckB.IsChecked)

friends = friends.OrderByDescending(item => item.user1.name).ToList();

if (!(bool)NameUpCheckB.IsChecked && (bool)NameDownCheckB.IsChecked)

friends = friends.OrderBy(item => item.user1.name).ToList();

if (!string.IsNullOrEmpty(searchTB.Text))

friends = friends.Where(item => item.user1.name.Contains(searchTB.Text)).ToList();

userTabel.ItemsSource = friends;

}

В методе updateData происходит вся магия сортировки/фильтрации/поиска нужных нам товарищей, теперь давайте взглянем на них повнимательнее и поймем, что ничего сложного тут нет. Но зато у меня сейчас заиграл Linkin Park – Don’t Stay, да, круто, знаю. Давайте вместе туц туц туц туц

Sometimes, I need to remember just to breathe

Sometimes, I need you to stay away from me

Sometimes, I'm in disbelief, I didn't know

Somehow, I need you to go

(Don't stay) Forget our memories, forget our possibilities

What you were changing me into

Just give me myself back and don't stay

Forget our memories, forget our possibilities

Take all your faithlessness with you

Just give me myself back and don't stay

Sometimes, I feel like I trusted you too well

Sometimes, I just feel like screaming at myself

Sometimes, I'm in disbelief, I didn't know

Somehow, I need to be alone

(Don't stay) Forget our memories, forget our possibilities

What you were changing me into

Just give me myself back and don't stay

Forget our memories, forget our possibilities

Take all your faithlessness with you

Just give me myself back and don't stay

I don't need you anymore

I don't want to be ignored

I don't need one more day

Of you wasting me away

I don't need you anymore

I don't want to be ignored

I don't need one more day

Of you wasting me away

With no apologies

(Don't stay) Forget our memories, forget our possibilities

What you were changing me into

Just give me myself back and don't stay

Forget our memories, forget our possibilities

Take all your faithlessness with you

Just give me myself back and don't stay

Don't stay

Don't stay!

Каеф, попели и олскулы свели, а теперь приступим к разбору нашего кода. В первом условном операторе мы делаем фильтрацию по ролям, во втором условном блоке мы производим сортировку по возрастанию, в третьем блоке сортировку по убыванию, и в итоговом четвертом ищем нужного товарища по имени. Назови меня по имени.

И вот так я умею:

private void InitRolesCB()

{

FilterCB.Items.Add(new user\_role()

{

name = "--- no ---"

});

FilterCB.SelectedIndex = 0;

Instance.GetContext().user\_role.ToList().ForEach((user\_role role) =>

{

FilterCB.Items.Add(role);

});

}

В данном методе мы инициализируем ComboBox с ролями пользователей для нашей фильтрации по ролям.

private void FilterCB\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

updateData();

}

private void searchTB\_TextChanged(object sender, TextChangedEventArgs e)

{

updateData();

}

private void NameUpCheckB\_Checked(object sender, RoutedEventArgs e)

{

updateData();

}

Не забываем про привязку событий изменения состояний данных графических пользовательских элементов и пойдем спать, ага, спят только на том свете и не дима.

Теперь разберемся за гаражами с удалением и добавлением дружбы с товарищами.

Как ни странно, начнем с больших амбициозных штучек, а именно с добавления нашей дружбы. Код в студию:

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

{

CommonWindow window = new CommonWindow();

window.setPage(new AddNewFriendWidget(DataContext as user, window));

window.ShowDialog();

updateData();

}

Господа, интелектуалы, как вы уже смогли заметить CommonWindow мы используем повторно и это круто. Код вы можете посмотреть выше. Ну а сам метод ничем таким сложным не блещет. Только мы тут вызываем updateData после добавления дружбы, чтобы он отобразился у нас в ListBox, но это не важно, важно AddNewFriendWidget, вот теперь посмотрим поглубже на наш файл разметки для виджета.

AddNewFriendWidget.xaml

<Page x:Class="template\_wpf.widgets.AddNewFriendWidget"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:template\_wpf.widgets"

xmlns:common="clr-namespace:template\_wpf.common"

mc:Ignorable="d"

Title="AddNewFriendWidget"

Height="300" Width="300"

MinHeight="300" MinWidth="300"

>

<Page.Resources>

<ResourceDictionary>

<common:ConvertUserName x:Key="ConvertUserName"/>

</ResourceDictionary>

</Page.Resources>

<Grid Margin="10">

<Grid.ColumnDefinitions>

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="0.2\*"/>

<RowDefinition Height="0.6\*"/>

<RowDefinition Height="0.2\*"/>

</Grid.RowDefinitions>

<ComboBox x:Name="UsersCB"

Grid.Row="1"

VerticalAlignment="Top"

Margin="10, 0"

>

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Converter={StaticResource ConvertUserName}}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<Button x:Name="addBtn"

Content="Добавить"

Grid.Row="12"

Margin="5, 10"

Click="addBtn\_Click"

/>

</Grid>

</Page>

Как вы видите из разметки, я забил большой болт на придумывание и выебыебоны, и так дофига чего крутого реализовал. В итоге у нас открывается новое окно, где отображается ComboBox со списком пользователей. Да правдивость не друзей я не ручаюсь, потому что лень было придумывать как найти всех пользователей не находящиеся в наших друзьях, это на словах понятно, но моя структура базы данных (friendship) не очень-то и универсальна, правда работает. Если там реально отображаются нужные человечки, то вообще круто.

Ой мемы пошли

https://www.youtube.com/watch?v=XWIGbHhSsGs

mediaeval музыка пошла



Чет да так не хочу в школу….. лучше бы в палете сидел, головой об стену бился, больше толку было бы.

Ладно, возвращаемся к нашим кобылам.

Window parentWindow;

public AddNewFriendWidget(user sender, Window parent)

{

InitializeComponent();

DataContext = sender;

parentWindow = parent;

UsersCB.ItemsSource = Instance.GetContext().friendship.Where(p => p.login\_sender != sender.login).Select(p => p.user1).ToList();

}

Конструктор типичного зожника, в котором мы инициализируем ComboBox пользователей, не находящихся у нас в друзьях.

Для реализации добавления в друзья мы используем следующий метод, да он может быть и не идеальный, но зато делает, что велено и подходит под наш контекст.

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

{

friendship ship = new friendship();

ship.id = Instance.GetContext().friendship.Count() + 1;

ship.login\_sender = (DataContext as user).login;

ship.user = (DataContext as user);

user recirver = UsersCB.SelectedItem as user;

if (recirver == null)

{

MessageBox.Show("Пользователь должен быть выбран", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

ship.login\_recirver = recirver.login;

ship.user1 = recirver;

Instance.GetContext().friendship.Add(ship);

Instance.GetContext().SaveChanges();

Instance.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

if ((DataContext as user).friendship.Where(p => p.login\_recirver == recirver.login).FirstOrDefault() != null)

{

MessageBox.Show("Пользователь успешно добавлен", "Добавление пользователя", MessageBoxButton.OK, MessageBoxImage.Information);

parentWindow.Close();

}

else

MessageBox.Show("Пользователя не удалось добавить", "Добавление пользователя", MessageBoxButton.OK, MessageBoxImage.Error);

}

Метод вам слишком знаком и синтаксические конструкции мы уже использовали, но я человек, и вы тоже люди, поэтому вспоминаем кратко: создаем новый экземпляр дружбы и инициализируем его нужными данными, обеспечивая полноту данных. Дальше мы сохраняемся и выводим сообщения об успешной выполненной задачи. Также мы не забываем проверять выбранный элемент из ComboBox на null, потому что вдруг пользователь не выбрал и сразу нажал на кнопку добавить.

Вот и осталась последняя половина последней головы горыныча, напомню вам – удаление друга. Вот тут есть одна маленькая деталь, у кнопок и других пользовательских элементов есть такое поле Tag, в котором можно хранить идентификатор строки пользователя. В нашем случае мы там храним логин нашего друга, по которому мы его вычислим по ip и удалим из нашего общества.

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

{

string login = (sender as Button).Tag as string;

if (string.IsNullOrEmpty(login))

return;

user employee = DataContext as user;

friendship ship = employee.friendship.Where(p => p.login\_recirver == login).FirstOrDefault();

Instance.GetContext().friendship.Remove(ship);

Instance.GetContext().SaveChanges();

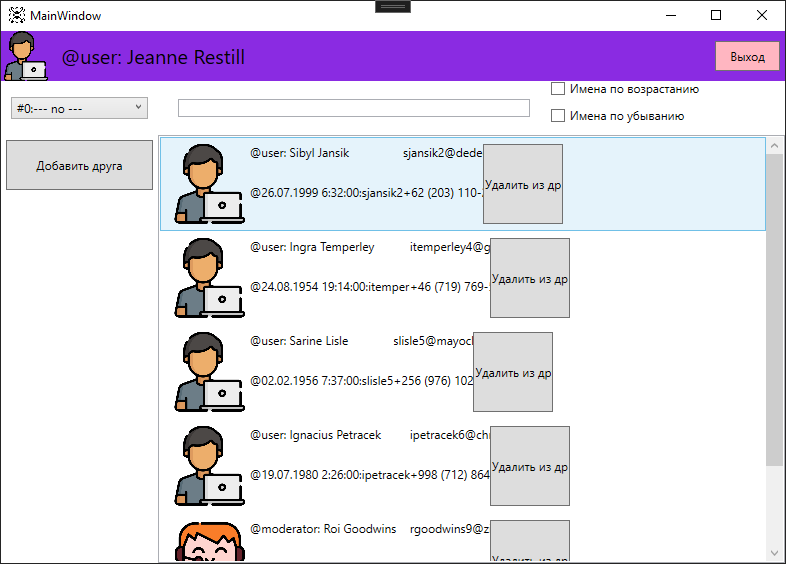
Instance.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

updateData();

}

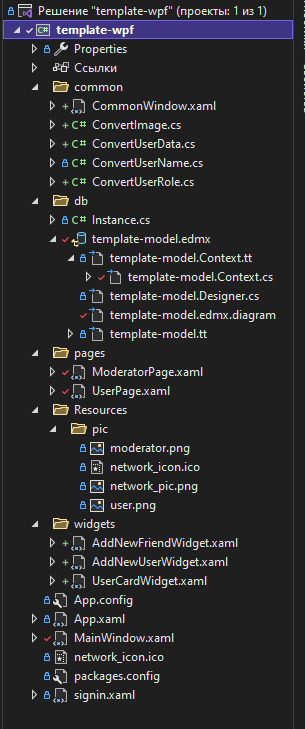
Находим нужную нам сущность дружбы и удаляем ее. Все. Так просто. Только не забудем сохраниться и обновить базу данных и вызвать updateData.

Что по итогу у вас должно получиться:



Как я раньше говорил, я не дезигнер и мне похуй, главное, что данные выводятся, а как это уже не мои проблемы))))

Вот такая иерархия проекта у вас должна быть по итогу:



Пора слушать Joji

В заключении, кто слушал и осознал всю суть басни тот красава, вам осталось несколько раз проделать данный проект и я вас уверяю, вы это будете делать за час максимум, если карты сложатся и луна будет в ретроградном меркурие то да за час управитесь, ну а так 2 часа вам за глаза, если вы конечно будете много раз делать подобной тематики приложение и в данном стиле.

Спасибо всем, кто тут, на этой строчке, у меня, конечно, крыша поехала уже, но благое дело надо было сделать уже давным-давно. Все желаю удачи в взломе жэпы дэ и вср и счастливо нового года. Много счастья вас ждет в будущем.