Содержание

[Введение 4](#_Toc182519493)

[1 Теоретическая часть 5](#_Toc182519494)

[1.1 Постановка задачи 5](#_Toc182519495)

[1.2 Выбор программ для реализации программного продукта 5](#_Toc182519496)

[2 Практическая часть 7](#_Toc182519497)

[2.1 Структура программного продукта 7](#_Toc182519498)

[2.2 Реализация программного продукта 9](#_Toc182519499)

[Заключение 24](#_Toc182519500)

[Список использованных источников 25](#_Toc182519501)

# **Введение**

В современной проектной деятельности приложение играет важную роль в достижении стратегических целей компаний. Для эффективного управления проектами необходимо иметь доступ к достоверным и актуальным данным. Прекрасным решением этой задачи является приложение для работы с базой данных проектных работ компании

В данной работе будет рассмотрено создание приложения для проектной организации, с целью автоматизации процессов управления организацией и определены требования к разработке структуры приложения.

Актуальность курсового проекта заключается в том, что разработка приложения для проектной организации позволит существенно повысить эффективность работы и улучшить организационные процессы.

Предмет исследования курсового проекта: деятельность проектных организаций.

Объект исследования курсового проекта: проектные организации.

Цель курсового проекта: разработка программных модулей для работы с базой данных «ProjectOrganization».

Задачи курсового проекта:

1. Определение функционала программного продукта.
2. Выбор программ для реализации программного продукта.
3. Описание структуры программного продукта.
4. Реализация программного продукта.
5. **Теоретическая часть**

**1.1 Постановка задачи**

Разрабатываемый продукт представляет собой настольное приложение «Deals» и предназначен для оптимизации управления проектными работами. Он будет предоставлять информацию о проектных работах.

Приложение состоит из нескольких модулей:

* вывод данных о проектных работах;
* добавление данных по новым проектным работам;
* удаление данных о проектных работах
* редактирование данных о существующих проектных работах.

**1.2 Выбор программ для реализации программного продукта**

Для реализации программного обеспечения «Deals» была выбрана среда разработки от Microsoft «Visual Studio», а языком программирования С#. Данный выбор представляет собой оптимальное решение для разработки сложных и масштабируемых проектов, обеспечивая высокий уровень производительности, удобства и безопасности процесса разработки.

Минимальные технические требования для Visual Studio:

* процессор: 1,8 ГГц или более быстрый процессор;
* оперативная память: 4 ГБ (рекомендуется 8 ГБ);
* графический процессор: с поддержкой DirectX 11;
* операционная система: Windows 10 (64-bit), MacOS, или Linux;
* свободное место на жестком диске: минимум 850 МБ для базовой установки; дополнительное место необходимо для установки компонентов и хранения проектов.

Для управления базой данных в проекте «Deals» используется Microsoft SQL Server Management Studio (SSMS) – мощный и интуитивно понятный инструмент для работы с СУБД Microsoft SQL Server. SSMS предоставляет удобные средства администрирования и разработки, позволяя эффективно управлять данными и обеспечивать целостность базы данных.

Минимальные технические требования для Microsoft SQL Server Management Studio:

* процессор: совместимый с x64, 1,4 ГГц или выше;
* оперативная память: 2 ГБ (рекомендуется 4 ГБ и выше);
* графический процессор: с поддержкой DirectX 9 или выше;
* операционная система: Windows 8/10 (64-bit) или Windows Server 2016 и новее;
* свободное место на жестком диске: 6 ГБ для установки.

Для разработки внешнего вида приложения использовался онлайн конструктор ADALO

Минимальные технические требования для ADALO:

* операционная система Windows 10/macOS/Linux/Chrome OS;
* оперативная память: 4 ГБ (рекомендуется 8 ГБ);
* поскольку ADALO работает онлайн, локальное хранилище практически не требуется;
* высокоскоростное подключение к интернету (минимальная скорость загрузки 10 Мбит/сек, рекомендуемая — 25 Мбит/сек или выше);
* современные браузеры Google Chrome, Mozilla Firefox, Safari, Edge.

**2 Практическая часть**

**2.1 Структура программного продукта**

С учетом данных, полученных при изучении специфики работы проектной организации, а также установленных требований к функционалу разработана модульная структура программного продукта, представленная в соответствии с рисунком 1.

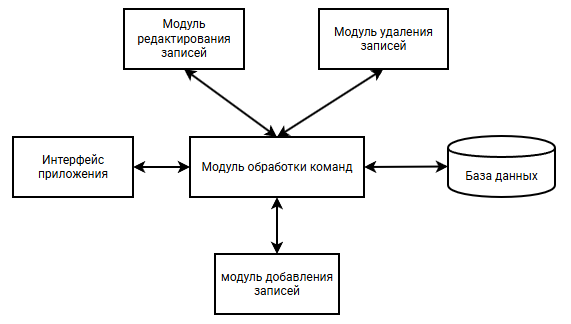


Рисунок 1 – Модульная структура приложения

В структуре разрабатываемого приложения выделены следующие модули:

1. Интерфейс программы – модуль, предназначенный для взаимодействия пользователя с программой.
2. Модуль обработки команд – модуль, решающий как должна отреагировать команда на отправленную команду пользователем или другим модулем.
3. Модуль редактирования записей – модуль, отвечающий за редактирование информации о существующих проектных работах.
4. Модуль добавления записей – модуль, предназначенный для создания записей о новых проектных работах с указанием всей необходимой информации.
5. Модуль удаления записей – модуль, отвечающий за удаление информации в базе данных о записях.
6. Модуль работы с базой данных – модуль, отвечающий за хранение информации о проектных работах, договорах.

Неотъемлемой частью приложения является его база данных, которая состоит из восьми таблиц: «Deal», «DesignWork», «OrgType», «Workers», «Gender», «Department», «JobTitles», «Organization».

Схема базы данных представлена в соответствии с рисунком 2.

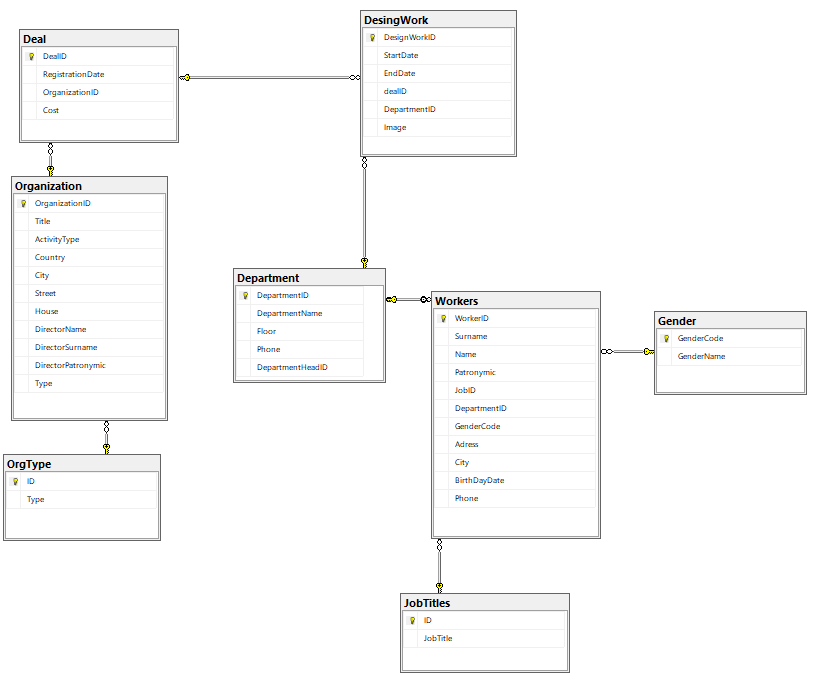


Рисунок 2 – Схема базы данных

Таким образом, была определена и разработана структура программного продукта, для последующей её реализации.

**2.2 Реализация программного продукта**

После запуска приложения открывается страница «DealPage» со списком проектных работ, представленная в соответствии с рисунком 3.

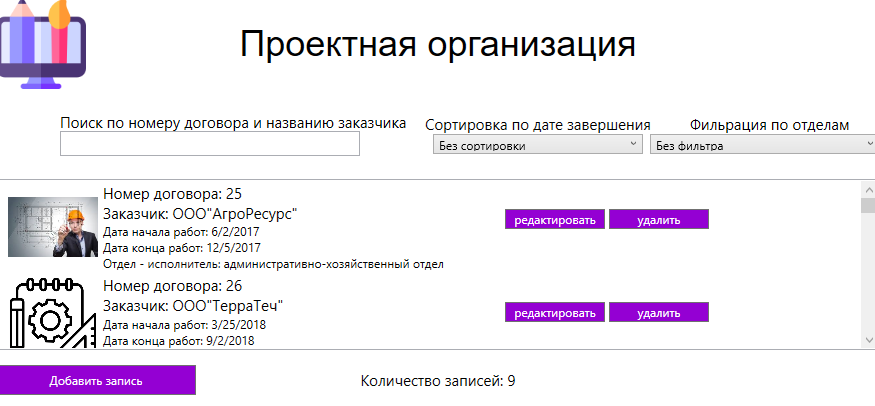


Рисунок 3 – Страница «DealPage»

Список проектных работ представляет собой следующую информацию:

* Номер договора.
* Заказчик с кем был заключен договор.
* Дата начала проектных работ.
* Дата завершения проектных работ.
* Отдел исполнитель.
* Изображение проектной работы.

XAML-разметка страницы «DealPage»:

<Page x:Class="ProjectOrg.DealPage"

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:ProjectOrg"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="900"

Title="DealPage">

<Grid>

<Grid.RowDefinitions >

<RowDefinition Height="90"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="60"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="2" HorizontalAlignment="Center" VerticalAlignment="Center" Orientation="Horizontal">

<TextBlock Text="Количество записей: " FontSize="15"/>

<TextBlock Name="TBCount" FontSize="15"/>

</StackPanel>

<Button Name="AddBtn" Width="200" Height="30" Content="Добавить запись" Grid.Row="2" HorizontalAlignment="Left" Click="AddBtn\_Click"/>

<StackPanel Grid.Row="0" VerticalAlignment="Center">

<TextBlock Text="Поиск по номеру договора и названию заказчика" Margin="64,0,0,0" FontSize="15"/>

<TextBox Name="SearchTxt" Width="300" Height="25" FontSize="15" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="64,0,0,0" TextChanged="SearchTxt\_TextChanged"/>

</StackPanel>

<StackPanel Grid.Row="0" VerticalAlignment="Center" Margin="200,0,0,0">

<TextBlock Text="Сортировка по дате завершения" FontSize="15" HorizontalAlignment="Center"/>

<ComboBox Name="SortBox" Width="210" Height="20" Grid.Row="0" SelectionChanged="SortBox\_SelectionChanged">

<TextBlock Text ="Без сортировки"/>

<TextBlock Text ="Дате завершения(возрастание)"/>

<TextBlock Text = "Дате завершения(убывание)"/>

</ComboBox>

</StackPanel >

<StackPanel VerticalAlignment="Center" HorizontalAlignment="Right" Margin="630,0,0,0">

<TextBlock Text="Фильрация по отделам" FontSize="15" Margin="40,0,0,0"/>

<ComboBox Name="FiltBox" Width="230" Height="20" Grid.Row="0" SelectionChanged="FiltBox\_SelectionChanged">

<TextBlock Text ="Без фильтра"/>

<TextBlock Text ="Отдел водоснабжения и канализации"/>

<TextBlock Text ="административно-хозяйственный отдел"/>

<TextBlock Text ="отдел информационного обеспечения"/>

<TextBlock Text ="отдел выдачи проектов "/>

<TextBlock Text = "Отдел ген планов и транспорта"/>

</ComboBox>

</StackPanel>

<ListView Name="DesingWorkListView" Grid.Row="1" IsVisibleChanged="DesingWorkListView\_IsVisibleChanged" d:ItemsSource="{d:SampleData ItemCount=5}">

<ListView.ItemTemplate>

<DataTemplate>

<Grid>

<Grid.RowDefinitions >

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="100"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<WrapPanel Grid.Column="1" Grid.Row="0" VerticalAlignment="Center" Orientation="Horizontal">

<TextBlock Text="Номер договора: " FontSize="15" />

<TextBlock Text="{Binding dealID}" FontSize ="15"/>

</WrapPanel>

<WrapPanel Grid.Column="1" Grid.Row="2" Orientation="Horizontal">

<TextBlock Text="Дата начала работ: "/>

<TextBlock Text ="{Binding StartDate, StringFormat=\{0:d\}}"/>

</WrapPanel>

<WrapPanel Grid.Column="1" Grid.Row="3" Orientation="Horizontal">

<TextBlock Text="Дата конца работ: "/>

<TextBlock Text ="{Binding EndDate,StringFormat=\{0:d\}}"/>

</WrapPanel>

<WrapPanel Grid.Row="4" Grid.Column="1" Orientation="Horizontal" >

<TextBlock Text="Отдел - исполнитель: "/>

<TextBlock Text="{Binding DepartmentName}"/>

</WrapPanel>

<WrapPanel Grid.Row="1" Grid.Column="1" Orientation="Horizontal" >

<TextBlock Text="Заказчик: " FontSize ="15"/>

<TextBlock Text="{Binding OrgType}" FontSize ="15"/>

<TextBlock Text="{Binding OrgName }" FontSize ="15"/>

</WrapPanel>

<Image Grid.RowSpan="5" Width="90">

<Image.Source>

<Binding Path="Image">

<Binding.TargetNullValue>

<ImageSource>

Resources\process.png

</ImageSource>

</Binding.TargetNullValue>

</Binding>

</Image.Source>

</Image>

<StackPanel Grid.Column="1" Grid.RowSpan="4" Orientation="Horizontal" HorizontalAlignment="Right" Margin="400,0,0,0">

<Button Name="EditBtn" Content="редактировать" Margin="2" Click="EditBtn\_Click" Width="100" Height="20"/>

<Button Name="DeleteBtn" Content="удалить" Margin="2" Click ="DeleteBtn\_Click" Width="100" Height="20"/>

</StackPanel>

</Grid>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</Grid>

</Page>

Логика взаимодействия элементов страницы прописана в файле «DealPage.xaml.cs» и представлена кодом:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ProjectOrg

{

/// <summary>

/// Логика взаимодействия для DealPage.xaml

/// </summary>

public partial class DealPage : Page

{

List <DesingWork> CurrentPage = new List <DesingWork> ();

List<DesingWork> TableList;

public DealPage()

{

InitializeComponent();

var currentDesingWorks = ProjectOranizationDBEntities.GetContext().DesingWork.ToList();

DesingWorkListView.ItemsSource = currentDesingWorks;

SortBox.SelectedIndex = 0;

FiltBox.SelectedIndex = 0;

Update();

}

void Update()

{

var currentDesingWorks = ProjectOranizationDBEntities.GetContext().DesingWork.ToList();

if (SortBox.SelectedIndex == 1) currentDesingWorks = currentDesingWorks.OrderBy(p => p.EndDate).ToList();

if (SortBox.SelectedIndex == 2) currentDesingWorks = currentDesingWorks.OrderByDescending(p => p.EndDate).ToList();

if(FiltBox.SelectedIndex == 1) currentDesingWorks = currentDesingWorks.Where(p => (p.DepartmentID == 1)).ToList();

if(FiltBox.SelectedIndex == 2) currentDesingWorks = currentDesingWorks.Where(p => p.DepartmentID == 2).ToList();

if(FiltBox.SelectedIndex == 3) currentDesingWorks = currentDesingWorks.Where(p => (p.DepartmentID == 3)).ToList();

if(FiltBox.SelectedIndex == 4) currentDesingWorks = currentDesingWorks.Where(p => (p.DepartmentID == 4)).ToList();

if(FiltBox.SelectedIndex == 5) currentDesingWorks = currentDesingWorks.Where(p => (p.DepartmentID == 5)).ToList();

currentDesingWorks = currentDesingWorks.Where(p => (p.dealID.ToString().Contains(SearchTxt.Text) || p.OrgName.ToLower().Contains(SearchTxt.Text.ToLower()) )).ToList();

int recCount = currentDesingWorks.Count;

TBCount.Text = recCount.ToString();

DesingWorkListView.ItemsSource = currentDesingWorks;

}

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

{

Update();

}

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

{

Update();

}

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

{

Update();

}

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

{

Manager.Frame.Navigate(new EditorPage(null));

}

private void DesingWorkListView\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

if(Visibility == Visibility.Visible)

{

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

DesingWorkListView.ItemsSource = ProjectOranizationDBEntities.GetContext().DesingWork.ToList();

Update();

}

}

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

{

Manager.Frame.Navigate(new EditorPage((sender as Button).DataContext as DesingWork));

}

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

{

var selectedWork = (sender as Button).DataContext as DesingWork;

if (selectedWork.EndDate > new DateTime(2015, 1, 1))

{

MessageBox.Show("Запись не может быть удалена, потому что данную работу закончили после первого января 2015-го года ");

}

else

{ if (MessageBox.Show("Вы точно хотите выполнить удаление?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

ProjectOranizationDBEntities.GetContext().DesingWork.Remove(selectedWork);

ProjectOranizationDBEntities.GetContext().SaveChanges();

DesingWorkListView.ItemsSource = ProjectOranizationDBEntities.GetContext().DesingWork.ToList();

Update();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

}

}

}

На странице присутствует поисковая строка для поиска работ по наименованию заказчика и номеру договора, представленная в соответствии с рисунком 4.



Рисунок 4 – Поисковая строка

Также есть сортировка списка работ по дате завершения работ, как показано в соответствии с рисунком 5.

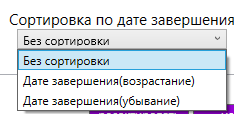


Рисунок 5 – Сортировка записей

Для удобства использования реализована фильтрация по отделам, как показано в соответствии с рисунком 6.

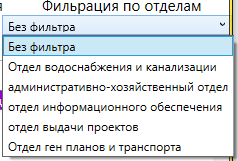


Рисунок 6 – Фильтрация записей

Так же на «DealPage» присутствует вывод отображаемых записей, как показано в соответствии с рисунком 7.



Рисунок 7 – Количество записей

Так же на странице присутствует кнопка «Добавить запись». При нажатии этой кнопки приложение открывает страницу «EditorPage», как показано в соответствии с рисунком 8.

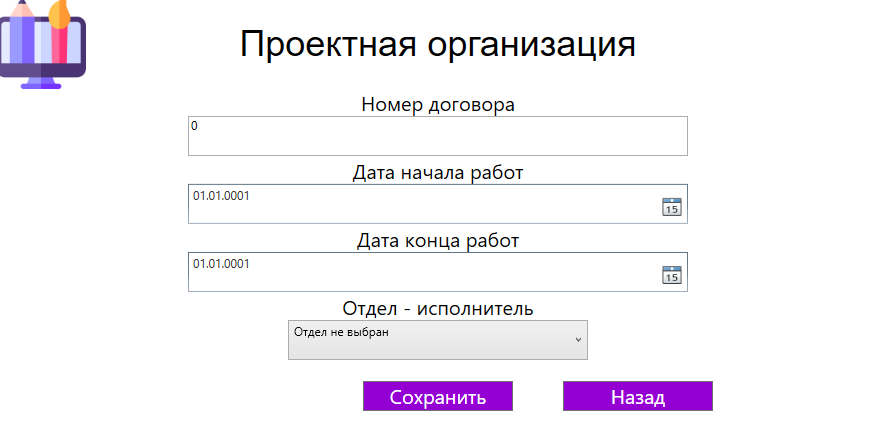


Рисунок 8 – Страница «EditorPage» после нажатия кнопки «Добавить запись»

На данной странице представлена следующая информация для заполнения:

* номер договора, по которому ведется работа;
* дата начала проведения работ;
* дата конца проведения работ;
* отдел исполнитель.

XAML-разметка страницы «EditorPage»:

<Page x:Class="ProjectOrg.EditorPage"

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:ProjectOrg"

mc:Ignorable="d"

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

Title="EditorPage">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0">

<TextBlock Text="Номер договора" HorizontalAlignment="Center" FontSize="20"/>

<TextBox Name ="DealTxt" Text="{Binding dealID}" Width="500" Height="40" HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Row="0"/>

</StackPanel>

<StackPanel Grid.Row="1">

<TextBlock Text="Дата начала работ" Grid.Row="1" HorizontalAlignment="Center" FontSize="20"/>

<DatePicker Name="StartDateDp" SelectedDate="{Binding StartDate}" Width="500" Height="40" HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Row="1" />

</StackPanel>

<StackPanel Grid.Row="2">

<TextBlock Text="Дата конца работ" HorizontalAlignment="Center" FontSize="20"/>

<DatePicker Name="EndDateDp" SelectedDate="{Binding EndDate}" Width="500" Height="40" HorizontalAlignment="Center" VerticalAlignment="Center" />

</StackPanel>

<StackPanel Grid.Row="3"><TextBlock Text="Отдел - исполнитель" Grid.Row="3" HorizontalAlignment="Center" FontSize="20"/>

<ComboBox Name="DepartmentBox" Grid.Row="3" Width="300" Height="40" SelectedIndex="{Binding DepartmentID}">

<TextBlock Text="Отдел не выбран"/>

<TextBlock Text ="Отдел водоснабжения и канализации"/>

<TextBlock Text ="административно-хозяйственный отдел"/>

<TextBlock Text ="отдел информационного обеспечения"/>

<TextBlock Text ="отдел выдачи проектов "/>

<TextBlock Text = "Отдел ген планов и транспорта"/>

</ComboBox>

</StackPanel>

<Button Name="SaveBtn" Width="150" Content="Сохранить" Height="30" Grid.Row="4" FontSize="20" Click="SaveBtn\_Click"/>

<Button Name="BackBtn" Width="150" Height="30" Content="Назад" FontSize="20" Grid.Row="4" Margin="400,0,0,0" Click="BackBtn\_Click" />

</Grid>

</Page>

Логика взаимодействия элементов страницы прописана в файле «EditorPage.xaml.cs» и представлена кодом:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace ProjectOrg

{

/// <summary>

/// Логика взаимодействия для EditorPage.xaml

/// </summary>

public partial class EditorPage : Page

{

private DesingWork \_currentDesignWork = new DesingWork();

public EditorPage(DesingWork SelectedDesignWork)

{

InitializeComponent();

if(SelectedDesignWork != null) \_currentDesignWork = SelectedDesignWork;

DataContext = \_currentDesignWork;

}

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

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrWhiteSpace(DealTxt.Text)) errors.AppendLine("Введите номер договора по которму проводится работа!");

else

{

int dealId;

bool isValidDealId = int.TryParse(DealTxt.Text, out dealId);

if (!isValidDealId || !ProjectOranizationDBEntities.GetContext().Deal.Any(p => p.DealID == dealId))

{

errors.AppendLine("Нет такого договора!");

}

}

if (string.IsNullOrWhiteSpace(StartDateDp.ToString())) errors.AppendLine("Введите дату начала работ!");

else \_currentDesignWork.StartDate = (DateTime)StartDateDp.SelectedDate;

if (string.IsNullOrWhiteSpace(EndDateDp.ToString())) errors.AppendLine("Введите дату конца работ!");

else \_currentDesignWork.EndDate = (DateTime)EndDateDp.SelectedDate;

if (DepartmentBox.SelectedIndex == 0 || DepartmentBox.SelectedIndex == null) errors.AppendLine("Выберите отдел Испольнитель!");

else \_currentDesignWork.DepartmentID = DepartmentBox.SelectedIndex;

if (errors.Length != 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (\_currentDesignWork.DesignWorkID == 0) ProjectOranizationDBEntities.GetContext().DesingWork.Add(\_currentDesignWork);

try

{

ProjectOranizationDBEntities.GetContext().SaveChanges();

MessageBox.Show("Данные сохранены");

Manager.Frame.GoBack();

}

catch

(Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

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

{

Manager.Frame.GoBack();

}

}

}

Если пользователь оставил поля для заполнения пустыми и нажал на кнопку «Сохранить», то приложение выведет окно с соответствующем сообщением.

Результаты работы кнопки «Сохранить» при стандартном сценарии (во все поля для заполнения внесена нужная информация) и при сценарии пустых полей представлены в соответствии с рисунками 9 и 10.

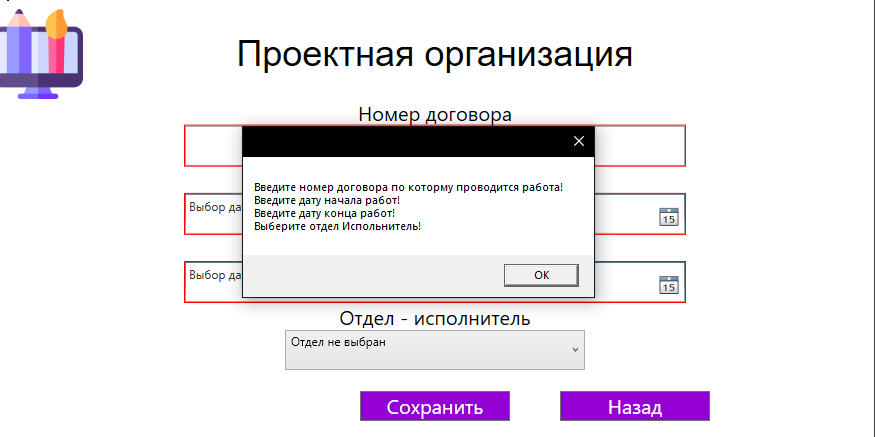


Рисунок 9 – Окно с сообщением при неудачном сохранении

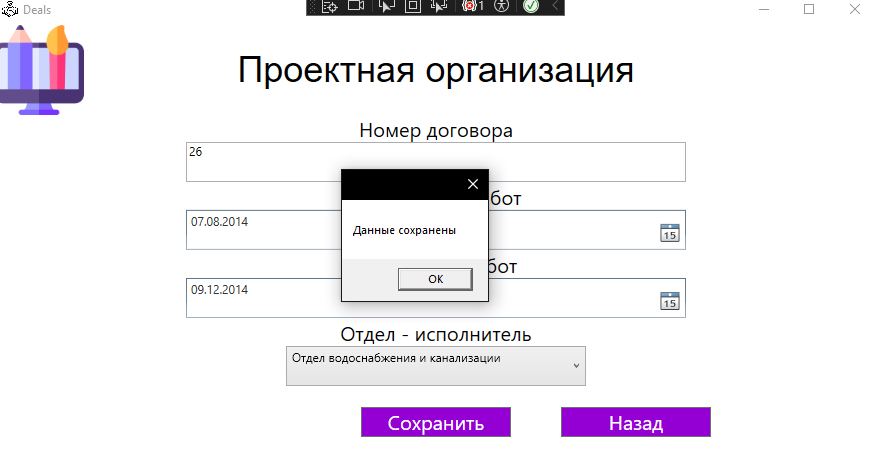


Рисунок 10 – Результат успешного сохранения

При возникновении необходимости пользователь может изменить данные уже записанных работ. Для этого необходимо нажать на кнопку «Редактировать» на странице «DealPage», тогда откроется редактор страницы «EditorPage» уже с заполненными полями, информация в которых может быть изменена.

Результат работы кнопки «Редактировать» представлен в соответствии с рисунком 11.

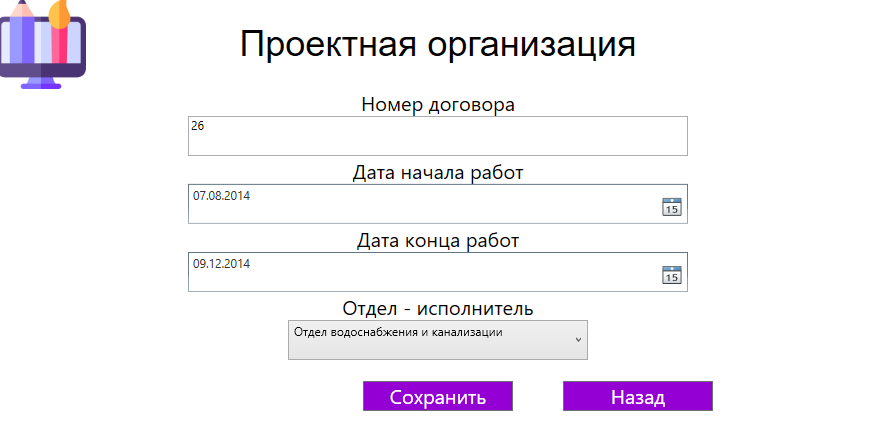


Рисунок 11 – Редактирование записи

Также если пользователь решит не добавлять или не изменять записи на странице «EditorPage», то кнопка «Назад» вернет его на страницу «DealPage» без сохранения внесенных данных.

При необходимости удаления записи из списка пользователь может воспользоваться кнопкой «Удалить».

При удалении записи о работе, что была закончена до первого января две тысячи пятнадцатого года появится всплывающее диалоговое окно с подтверждением его действий, если же удалению подлежат данные работ, что были закончена после указанной выше даты, то операция удаления не сработает с выводом соответствующего сообщения.

Результаты действия кнопки «Удалить» представлены в соответствии с рисунками 12 и 13.

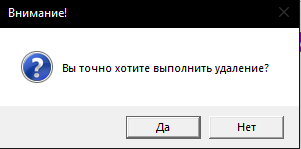


Рисунок 12 – Окно при успешном удалении записи

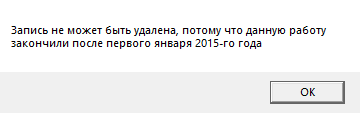


Рисунок 13 – Окно при неудачном удалении записи

Интерфейс приложения и стиль некоторых элементов прописаны в файлах «MainWindow.xaml» и «App.xaml»

XAML-разметка страницы «MainWindow»:

<Window x:Class="ProjectOrg.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:ProjectOrg"

mc:Ignorable="d"

Title="Deals" Height="450" Width="900" MinHeight="450" MinWidth="900" FontFamily="Arial" Icon="Resources/deal.png">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="90"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Image Source="Resources\Work.png" HorizontalAlignment="Left"></Image>

<TextBlock Text="Проектная организация" FontSize="36" HorizontalAlignment="Center" VerticalAlignment="Center" />

<Frame Grid.Row="1" Name="MainFrame" NavigationUIVisibility="Hidden" />

</Grid>

</Window>

XAML-разметка страницы « App »:

<Application x:Class="ProjectOrg.App"

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

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

xmlns:local="clr-namespace:ProjectOrg"

StartupUri="MainWindow.xaml">

<Application.Resources>

<Style TargetType="Button">

<Setter Property="Background" Value="#9400D3"/>

<Setter Property="Width" Value="200"/>

<Setter Property="Height" Value="40"/>

<Setter Property="Foreground" Value="White"/>

</Style>

</Application.Resources>

</Application>

Переходы между страницами реализованы с помощью класса «Manager». Логистика данного класса прописана в файле «Manager.cs», код которого представлен следующим образом:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Controls;

namespace ProjectOrg

{

internal class Manager

{

public static Frame Frame { get; set; }

}}

# **Заключение**

В процессе реализации курсового проекта было создано приложение для работы с базой данных проектной организации, направленное на автоматизацию организационных процессов. Разработанное приложение позволяет упорядочить и упростить доступ к информации о проектных работах, что способствует более эффективному менеджменту внутренних ресурсов организации.

В ходе работы были выполнены все ключевые этапы разработки: определён необходимый функционал приложения, проведён выбор подходящих программных инструментов, описана структура программного продукта и осуществлена его реализация. Приложение обеспечивает удобный интерфейс для работы с базой данных, поддержку поиска и сортировки, что позволяет организациям эффективнее проводить проектные работы и оптимизировать организационные процессы, связанные с внутренней логистикой.

Поставленная цель достигнута, задачи выполнены.

# **Список использованных источников**

1. Головин С.В. «Основы проектирования баз данных: Учебное пособие». – Москва: Издательство Московского государственного университета, 2024. – 245 с. (дата обращения 13.11.2024)
2. ГОСТ 19.106-78 «Требования к программным документам, выполненным печатным способом», введён 18 декабря. 1978 г. (дата обращения 13.11.2024)
3. ГОСТ 7.1-2003. «Библиографическое описание документа», введён 1 июля 2004 г. (дата обращения 13.11.2024)
4. Кендалл К., Кендалл Д. «Анализ и проектирование систем». – Москва: Вильямс, 2021. – 416 с. (дата обращения 13.11.2024)
5. Майерс Р. «Создание интерфейсов пользователя для информационных систем». – Москва: Диалектика, 2021. – 368 с. (дата обращения 11.11.2024)
6. Скавронский М.Л., Бушуев С.Д. «Проектирование и разработка информационных систем». – Санкт-Петербург: Питер, 2023. – 320 с. (дата обращения 10.11.2024)
7. Троелсен Э., Джепикс Ф. «Язык программирования C# и платформа .NET». – Санкт-Петербург: Питер, 2022. – 800 с. (дата обращения 9.11.2024)
8. Metanit. «Руководство по С#». metanit.com. (дата обращения 9.11.2024)