Файл: AuthorizationModule.cs Каталог: Authorization  
  
using Authorization.ViewModels;  
using Authorization.Views;  
using Prism.Ioc;  
using Prism.Modularity;  
using Prism.Regions;  
  
namespace Authorization  
{  
 public class AuthorizationModule : IModule  
 {  
 public void OnInitialized(IContainerProvider containerProvider)  
 {  
 }  
  
 public void RegisterTypes(IContainerRegistry containerRegistry)  
 {  
 containerRegistry.RegisterForNavigation<ViewA, ViewAViewModel>("Auth");  
 }  
 }  
}  
  
Файл: PasswordBoxAssistant.cs Каталог: Utility  
  
using System.Windows;  
using System.Windows.Controls;  
  
namespace Authorization.Utility  
{  
 public static class PasswordBoxAssistant  
 {  
 public static readonly DependencyProperty BoundPassword =  
 DependencyProperty.RegisterAttached("BoundPassword", typeof(string), typeof(PasswordBoxAssistant), new PropertyMetadata(string.Empty, OnBoundPasswordChanged));  
  
 public static readonly DependencyProperty BindPassword = DependencyProperty.RegisterAttached(  
 "BindPassword", typeof(bool), typeof(PasswordBoxAssistant), new PropertyMetadata(false, OnBindPasswordChanged));  
  
 private static readonly DependencyProperty UpdatingPassword =  
 DependencyProperty.RegisterAttached("UpdatingPassword", typeof(bool), typeof(PasswordBoxAssistant), new PropertyMetadata(false));  
  
 private static void OnBoundPasswordChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)  
 {  
 PasswordBox box = d as PasswordBox;  
 if (d == null || !GetBindPassword(d))  
 {  
 return;  
 }  
  
 box.PasswordChanged -= HandlePasswordChanged;  
  
 string newPassword = (string)e.NewValue;  
  
 if (!GetUpdatingPassword(box))  
 {  
 box.Password = newPassword;  
 }  
  
 box.PasswordChanged += HandlePasswordChanged;  
 }  
  
 private static void OnBindPasswordChanged(DependencyObject dp, DependencyPropertyChangedEventArgs e)  
 {  
 PasswordBox box = dp as PasswordBox;  
  
 if (box == null)  
 {  
 return;  
 }  
  
 bool wasBound = (bool)(e.OldValue);  
 bool needToBind = (bool)(e.NewValue);  
  
 if (wasBound)  
 {  
 box.PasswordChanged -= HandlePasswordChanged;  
 }  
  
 if (needToBind)  
 {  
 box.PasswordChanged += HandlePasswordChanged;  
 }  
 }  
  
 private static void HandlePasswordChanged(object sender, RoutedEventArgs e)  
 {  
 PasswordBox box = sender as PasswordBox;  
 SetUpdatingPassword(box, true);  
 SetBoundPassword(box, box.Password);  
 SetUpdatingPassword(box, false);  
 }  
  
 public static void SetBindPassword(DependencyObject dp, bool value)  
 {  
 dp.SetValue(BindPassword, value);  
 }  
  
 public static bool GetBindPassword(DependencyObject dp)  
 {  
 return (bool)dp.GetValue(BindPassword);  
 }  
  
 public static string GetBoundPassword(DependencyObject dp)  
 {  
 return (string)dp.GetValue(BoundPassword);  
 }  
  
 public static void SetBoundPassword(DependencyObject dp, string value)  
 {  
 dp.SetValue(BoundPassword, value);  
 }  
  
 private static bool GetUpdatingPassword(DependencyObject dp)  
 {  
 return (bool)dp.GetValue(UpdatingPassword);  
 }  
  
 private static void SetUpdatingPassword(DependencyObject dp, bool value)  
 {  
 dp.SetValue(UpdatingPassword, value);  
 }  
 }  
}  
  
Файл: ViewAViewModel.cs Каталог: ViewModels

using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Regions;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Security;  
using System.Text;  
using System.Threading.Tasks;  
using Workspace.DBHandler;  
  
namespace Authorization.ViewModels  
{  
 public class ViewAViewModel : BindableBase  
 {  
 private IRegionManager regionManager;  
 private string message;  
 public string Message  
 {  
 get { return message; }  
 set { SetProperty(ref message, value); }  
 }  
  
 private string username;  
 public string Username  
 {  
 get { return username; }  
 set { SetProperty(ref username, value); }  
 }  
  
 private string password;  
 public string Password  
 {  
 get { return password; }  
 set { SetProperty(ref password, value); }  
 }  
  
 public DelegateCommand<string> NavigateCommand { get; private set; }  
  
 public DelegateCommand LoginCommand { get; private set; }  
  
 public ViewAViewModel(IRegionManager regionManager)  
 {  
 this.regionManager = regionManager;  
 LoginCommand = new DelegateCommand(Login);  
  
 }  
 private void Login()  
 {  
 var list = DataBase.GetUsers();  
 foreach (var item in list)  
 {  
 if (Username?.ToString() == item.Username && Password?.ToString() == item.Password)  
 {  
 Message = "Авторизация прошла успешно";  
 regionManager.RequestNavigate("ContentRegion", "Workspace");  
 break;  
 }  
 }  
 Message = "Неверно введены данные";   
 }  
 }  
}  
  
  
Файл: ViewA.xaml.cs Каталог: Views  
  
using Prism.Regions;  
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 Authorization.Views  
{  
 /// <summary>  
 /// Interaction logic for ViewA.xaml  
 /// </summary>  
 public partial class ViewA : UserControl  
 {  
 public ViewA()  
 {  
 InitializeComponent();  
   
 }  
 }  
}  
  
  
Файл: App.xaml.cs Каталог: InspectionBoard  
  
using Authorization;  
using InspectionBoard.Dialogs;  
using InspectionBoard.ViewModels;  
using InspectionBoard.Views;  
using Prism.Ioc;  
using Prism.Modularity;  
using Prism.Regions;  
using Prism.Unity;  
using System.Windows;  
using Workspace;  
  
  
namespace InspectionBoard  
{  
 public partial class App : PrismApplication  
 {  
 protected override Window CreateShell()  
 {  
 return Container.Resolve<Main>();  
 }  
  
 protected override void RegisterTypes(IContainerRegistry containerRegistry)  
 {   
 containerRegistry.RegisterDialog<AddApplicantDialog, AddApplicantDialogViewModel>();  
 containerRegistry.RegisterDialog<RemoveApplicantDialog, RemoveApplicantDialogViewModel>();  
 containerRegistry.RegisterDialog<EditApplicantDialog, EditApplicantDialogViewModel>();  
 containerRegistry.RegisterDialog<NotificationDialog, NotificationDialogViewModel>();  
 containerRegistry.RegisterDialog<DocsSettingsDialog, DocsSettingsDialogViewModel>();  
 }  
  
 protected override void ConfigureModuleCatalog(IModuleCatalog moduleCatalog)  
 {  
 moduleCatalog.AddModule<AuthorizationModule>();  
 moduleCatalog.AddModule<WorkspaceModule>();  
 }  
 }  
}  
  
  
Файл: AddApplicantDialog.xaml.cs Каталог: Dialogs  
  
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 InspectionBoard.Dialogs  
{  
 /// <summary>  
 /// Логика взаимодействия для AddApplicantDialog.xaml  
 /// </summary>  
 public partial class AddApplicantDialog : UserControl  
 {  
 public AddApplicantDialog()  
 {  
 InitializeComponent();  
 }  
 }  
}  
  
  
Файл: AddApplicantDialogViewModel.cs Каталог: Dialogs  
  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Services.Dialogs;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using Workspace.DBHandler;  
using Workspace.Models;  
  
namespace InspectionBoard.Dialogs  
{  
 public class AddApplicantDialogViewModel : BindableBase, IDialogAware  
 {  
 private DelegateCommand<string> \_closeDialogCommand;  
 public DelegateCommand<string> CloseDialogCommand =>  
 \_closeDialogCommand ?? (\_closeDialogCommand = new DelegateCommand<string>(CloseDialog));  
  
 private string title = "Добавить абитуриента";  
 public string Title  
 {  
 get { return title; }  
 set { SetProperty(ref title, value); }  
 }  
  
 private string[] parameters;  
 public string[] Parameters  
 {  
 get { return parameters; }  
 set { SetProperty(ref parameters, value); }  
 }  
  
 public event Action<IDialogResult> RequestClose;  
  
 public AddApplicantDialogViewModel()  
 {  
 Parameters = new string[5];  
 }  
  
 protected virtual void CloseDialog(string parameter)  
 {  
 ButtonResult result = ButtonResult.None;  
  
 if (parameter?.ToLower() == "true")  
 {  
 Applicant applicant = new Applicant(parameters[0], parameters[1], parameters[2], parameters[3], parameters[4]);  
 AddApplicant(applicant);  
  
 result = ButtonResult.OK;  
  
 }  
 else  
 {  
 if (parameter?.ToLower() == "false")  
 result = ButtonResult.Cancel;  
 }  
  
 RaiseRequestClose(new DialogResult(result));  
 }  
  
 public virtual void RaiseRequestClose(IDialogResult dialogResult)  
 {  
 RequestClose?.Invoke(dialogResult);  
 }  
  
 public virtual bool CanCloseDialog()  
 {  
 return true;  
 }  
  
 public virtual void OnDialogClosed()  
 {  
  
 }  
  
 public virtual void OnDialogOpened(IDialogParameters parameters) //удалить потом  
 {  
 }  
  
 private void AddApplicant(Applicant a)  
 {  
 DataBase.AddApplicant(a);  
 }  
 }  
}  
  
Файл: DocsSettingsDialog.xaml.cs Каталог: Dialogs  
  
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 InspectionBoard.Dialogs  
{  
 /// <summary>  
 /// Логика взаимодействия для DocsSettingsDialog.xaml  
 /// </summary>  
 public partial class DocsSettingsDialog : UserControl  
 {  
 public DocsSettingsDialog()  
 {  
 InitializeComponent();  
 }  
 }  
}  
  
Файл: DocsSettingsDialogViewModel.cs Каталог: Dialogs  
  
using Microsoft.Win32;  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Services.Dialogs;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
using Workspace.FileHandlers;  
  
namespace InspectionBoard.Dialogs  
{  
 public class DocsSettingsDialogViewModel : BindableBase, IDialogAware  
 {  
 private string title = "Настройки документов";  
 public string Title  
 {  
 get { return title; }  
 set { SetProperty(ref title, value); }  
 }  
  
 private string enrollmentReportTemplate;  
 public string EnrollmentReportTemplate  
 {  
 get { return enrollmentReportTemplate; }  
 set { SetProperty(ref enrollmentReportTemplate, value); }  
 }  
  
 private string enrollmentReports;  
 public string EnrollmentReports  
 {  
 get { return enrollmentReports; }  
 set { SetProperty(ref enrollmentReports, value); }  
 }   
  
 private DelegateCommand<string> \_closeDialogCommand;  
 public DelegateCommand<string> CloseDialogCommand =>  
 \_closeDialogCommand ?? (\_closeDialogCommand = new DelegateCommand<string>(CloseDialog));  
  
 public event Action<IDialogResult> RequestClose;  
  
 public DelegateCommand<string> BrowseFilesCommand {get; private set; }  
 public DelegateCommand<string> BrowseFoldersCommand { get; private set; }  
  
 public DocsSettingsDialogViewModel()  
 {  
 DocumentsSettings.LoadSettings();  
 BrowseFilesCommand = new DelegateCommand<string>(BrowseFiles);  
 BrowseFoldersCommand = new DelegateCommand<string>(BrowseFolders);  
  
 }  
  
 protected virtual void CloseDialog(string parameter)  
 {  
 ButtonResult result = ButtonResult.None;  
  
 if (parameter?.ToLower() == "true")  
 {  
 result = ButtonResult.OK;  
 DocumentsSettings.SaveSettings();  
 }  
 else  
 {  
 if (parameter?.ToLower() == "false")  
 result = ButtonResult.Cancel;  
 }  
  
 RaiseRequestClose(new Prism.Services.Dialogs.DialogResult(result));  
 }  
  
 public virtual void RaiseRequestClose(IDialogResult dialogResult)  
 {  
 RequestClose?.Invoke(dialogResult);  
 }  
  
 public bool CanCloseDialog()  
 {  
 return true;  
 }  
  
 public void OnDialogClosed()  
 {  
   
 }  
  
 public void OnDialogOpened(IDialogParameters parameters)  
 {  
  
 }  
  
 private void BrowseFolders(string settingName)  
 {  
 FolderBrowserDialog dlg = new FolderBrowserDialog();  
 if (dlg.ShowDialog() == System.Windows.Forms.DialogResult.OK)  
 {  
 DocumentsSettings.Settings[settingName] = dlg.SelectedPath;  
  
 }  
 UpdateViewProperties();  
 }  
  
 private void BrowseFiles(string settingName)  
 {  
 System.Windows.Forms.OpenFileDialog dlg = new System.Windows.Forms.OpenFileDialog();  
 dlg.Filter = "(\*.docx)|\*.docx";  
 if (dlg.ShowDialog() == System.Windows.Forms.DialogResult.OK)  
 {  
 DocumentsSettings.Settings[settingName] = dlg.FileName;  
 }  
 }  
  
 private void UpdateViewProperties()  
 {  
 EnrollmentReportTemplate = DocumentsSettings.Settings["EnrollmentReportTemplate"];  
 EnrollmentReportTemplate = DocumentsSettings.Settings["EnrollmentReportTemplate"];  
 EnrollmentReports = DocumentsSettings.Settings["EnrollmentReports"];  
 }  
 }  
}  
  
Файл: EditApplicantDialog.xaml.cs Каталог: Dialogs  
  
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 InspectionBoard.Dialogs  
{  
 /// <summary>  
 /// Логика взаимодействия для EditApplicantDialog.xaml  
 /// </summary>  
 public partial class EditApplicantDialog : UserControl  
 {  
 public EditApplicantDialog()  
 {  
 InitializeComponent();  
 }  
 }  
}  
  
Файл: EditApplicantDialogViewModel.cs Каталог: Dialogs  
  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Services.Dialogs;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using Workspace.DBHandler;  
using Workspace.Models;  
  
namespace InspectionBoard.Dialogs  
{  
 public class EditApplicantDialogViewModel : BindableBase, IDialogAware  
 {  
 private DelegateCommand<string> \_closeDialogCommand;  
 public DelegateCommand<string> CloseDialogCommand =>  
 \_closeDialogCommand ?? (\_closeDialogCommand = new DelegateCommand<string>(CloseDialog));  
  
 private string title = "Добавить абитуриента";  
 public string Title  
 {  
 get { return title; }  
 set { SetProperty(ref title, value); }  
 }  
  
 private string[] parameters;  
 public string[] Parameters  
 {  
 get { return parameters; }  
 set { SetProperty(ref parameters, value); }  
 }  
  
 private int id;  
 public int ID  
 {  
 get { return id; }  
 set { SetProperty(ref id, value); }  
 }  
  
 private List<Applicant> applicants;  
 public List<Applicant> Applicants  
 {  
 get { return applicants; }  
 set  
 {  
 SetProperty(ref applicants, value);  
  
  
 }  
 }  
  
 public event Action<IDialogResult> RequestClose;  
  
 public EditApplicantDialogViewModel()  
 {  
 Parameters = new string[6];  
 Applicants = new List<Applicant>(DataBase.GetApplicants());  
 }  
  
 protected virtual void CloseDialog(string parameter)  
 {  
 ButtonResult result = ButtonResult.None;  
  
 if (parameter?.ToLower() == "true")  
 {  
 Applicant applicant = new Applicant(ID, parameters[1], parameters[2], parameters[3], parameters[4], parameters[5]);  
 EditApplicant(applicant);  
 result = ButtonResult.OK;  
 }  
 else  
 {  
 if (parameter?.ToLower() == "false")  
 result = ButtonResult.Cancel;  
 }  
 RaiseRequestClose(new DialogResult(result));  
 }  
  
 public virtual void RaiseRequestClose(IDialogResult dialogResult)  
 {  
 RequestClose?.Invoke(dialogResult);  
 }  
  
 public virtual bool CanCloseDialog()  
 {  
 return true;  
 }  
  
 public virtual void OnDialogClosed()  
 {  
  
 }  
  
 public virtual void OnDialogOpened(IDialogParameters parameters) //удалить потом  
 {  
 }  
  
 private void EditApplicant(Applicant a)  
 {  
 DataBase.EditApplicant(a);  
 }  
 }  
}  
  
Файл: NotificationDialog.xaml.cs Каталог: Dialogs  
  
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 InspectionBoard.Dialogs  
{  
 /// <summary>  
 /// Логика взаимодействия для NotificationDialog.xaml  
 /// </summary>  
 public partial class NotificationDialog : UserControl  
 {  
 public NotificationDialog()  
 {  
 InitializeComponent();  
 }  
 }  
}  
  
Файл: NotificationDialogViewModel.cs Каталог: Dialogs  
  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Services.Dialogs;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace InspectionBoard.Dialogs  
{  
 public class NotificationDialogViewModel : BindableBase, IDialogAware  
 {  
 private DelegateCommand<string> \_closeDialogCommand;  
 public DelegateCommand<string> CloseDialogCommand =>  
 \_closeDialogCommand ?? (\_closeDialogCommand = new DelegateCommand<string>(CloseDialog));  
  
 private string \_message;  
 public string Message  
 {  
 get { return \_message; }  
 set { SetProperty(ref \_message, value); }  
 }  
  
 private string \_title = "Notification";  
 public string Title  
 {  
 get { return \_title; }  
 set { SetProperty(ref \_title, value); }  
 }  
  
 private string selectedItem;  
 public string SelectedItem  
 {  
 get { return selectedItem; }  
 set { SetProperty(ref selectedItem, value); }  
 }  
  
 public event Action<IDialogResult> RequestClose;  
  
 protected virtual void CloseDialog(string parameter)  
 {  
 ButtonResult result = ButtonResult.None;  
  
 if (parameter?.ToLower() == "true")  
 result = ButtonResult.OK;  
 else if (parameter?.ToLower() == "false")  
 result = ButtonResult.Cancel;  
  
 RaiseRequestClose(new DialogResult(result));  
 }  
  
 public virtual void RaiseRequestClose(IDialogResult dialogResult)  
 {  
 RequestClose?.Invoke(dialogResult);  
 }  
  
 public virtual bool CanCloseDialog()  
 {  
 return true;  
 }  
  
 public virtual void OnDialogClosed()  
 {  
 }  
  
 public virtual void OnDialogOpened(IDialogParameters parameters)  
 {  
 Message = parameters.GetValue<string>("message");  
 }  
 }  
}  
  
  
Файл: RemoveApplicantDialog.xaml.cs Каталог: Dialogs  
  
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 InspectionBoard.Dialogs  
{  
 /// <summary>  
 /// Логика взаимодействия для RemoveApplicantDialog.xaml  
 /// </summary>  
 public partial class RemoveApplicantDialog : UserControl  
 {  
 public RemoveApplicantDialog()  
 {  
 InitializeComponent();  
 }  
 }  
}  
  
  
Файл: RemoveApplicantDialogViewModel.cs Каталог: Dialogs  
  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Services.Dialogs;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using Workspace.DBHandler;  
  
namespace InspectionBoard.Dialogs  
{  
 public class RemoveApplicantDialogViewModel : BindableBase, IDialogAware  
 {  
 private DelegateCommand<string> \_closeDialogCommand;  
 public DelegateCommand<string> CloseDialogCommand =>  
 \_closeDialogCommand ?? (\_closeDialogCommand = new DelegateCommand<string>(CloseDialog));  
  
 private string id;  
 public string ID  
 {  
 get { return id; }  
 set { SetProperty(ref id, value); }  
 }  
  
 private string \_title = "Удалить абитуриента";  
 public string Title  
 {  
 get { return \_title; }  
 set { SetProperty(ref \_title, value); }  
 }  
  
 public event Action<IDialogResult> RequestClose;  
  
 protected virtual void CloseDialog(string parameter)  
 {  
 ButtonResult result = ButtonResult.None;  
 if (parameter?.ToLower() == "true")  
 {  
 DataBase.DeleteApplicant(int.Parse(ID));  
 result = ButtonResult.OK;  
 }  
 else if (parameter?.ToLower() == "false")  
 result = ButtonResult.Cancel;  
  
 RaiseRequestClose(new DialogResult(result));  
 }  
  
 public virtual void RaiseRequestClose(IDialogResult dialogResult)  
 {  
 RequestClose?.Invoke(dialogResult);  
 }  
  
 public virtual bool CanCloseDialog()  
 {  
 return true;  
 }  
  
 public virtual void OnDialogClosed()  
 {  
 }  
  
 public virtual void OnDialogOpened(IDialogParameters parameters) //удалить потом  
 {  
 var Message = parameters.GetValue<string>("message");  
 }  
 }  
}  
  
  
Файл: .NETFramework,Version=v4.8.AssemblyAttributes.cs Каталог: Debug  
  
// <autogenerated />  
using System;  
using System.Reflection;  
[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8", FrameworkDisplayName = ".NET Framework 4.8")]  
  
Файл: PasswordBoxAssistant.cs Каталог: Utility  
  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows;  
using System.Windows.Controls;  
  
namespace InspectionBoard.Utility  
{  
 public static class PasswordBoxAssistant  
 {  
 public static readonly DependencyProperty BoundPassword =  
 DependencyProperty.RegisterAttached("BoundPassword", typeof(string), typeof(PasswordBoxAssistant), new PropertyMetadata(string.Empty, OnBoundPasswordChanged));  
  
 public static readonly DependencyProperty BindPassword = DependencyProperty.RegisterAttached(  
 "BindPassword", typeof(bool), typeof(PasswordBoxAssistant), new PropertyMetadata(false, OnBindPasswordChanged));  
  
 private static readonly DependencyProperty UpdatingPassword =  
 DependencyProperty.RegisterAttached("UpdatingPassword", typeof(bool), typeof(PasswordBoxAssistant), new PropertyMetadata(false));  
  
 private static void OnBoundPasswordChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)  
 {  
 PasswordBox box = d as PasswordBox;  
  
 // only handle this event when the property is attached to a PasswordBox  
 // and when the BindPassword attached property has been set to true  
 if (d == null || !GetBindPassword(d))  
 {  
 return;  
 }  
  
 // avoid recursive updating by ignoring the box's changed event  
 box.PasswordChanged -= HandlePasswordChanged;  
  
 string newPassword = (string)e.NewValue;  
  
 if (!GetUpdatingPassword(box))  
 {  
 box.Password = newPassword;  
 }  
  
 box.PasswordChanged += HandlePasswordChanged;  
 }  
  
 private static void OnBindPasswordChanged(DependencyObject dp, DependencyPropertyChangedEventArgs e)  
 {  
 // when the BindPassword attached property is set on a PasswordBox,  
 // start listening to its PasswordChanged event  
  
 PasswordBox box = dp as PasswordBox;  
  
 if (box == null)  
 {  
 return;  
 }  
  
 bool wasBound = (bool)(e.OldValue);  
 bool needToBind = (bool)(e.NewValue);  
  
 if (wasBound)  
 {  
 box.PasswordChanged -= HandlePasswordChanged;  
 }  
  
 if (needToBind)  
 {  
 box.PasswordChanged += HandlePasswordChanged;  
 }  
 }  
  
 private static void HandlePasswordChanged(object sender, RoutedEventArgs e)  
 {  
 PasswordBox box = sender as PasswordBox;  
  
 // set a flag to indicate that we're updating the password  
 SetUpdatingPassword(box, true);  
 // push the new password into the BoundPassword property  
 SetBoundPassword(box, box.Password);  
 SetUpdatingPassword(box, false);  
 }  
  
 public static void SetBindPassword(DependencyObject dp, bool value)  
 {  
 dp.SetValue(BindPassword, value);  
 }  
  
 public static bool GetBindPassword(DependencyObject dp)  
 {  
 return (bool)dp.GetValue(BindPassword);  
 }  
  
 public static string GetBoundPassword(DependencyObject dp)  
 {  
 return (string)dp.GetValue(BoundPassword);  
 }  
  
 public static void SetBoundPassword(DependencyObject dp, string value)  
 {  
 dp.SetValue(BoundPassword, value);  
 }  
  
 private static bool GetUpdatingPassword(DependencyObject dp)  
 {  
 return (bool)dp.GetValue(UpdatingPassword);  
 }  
  
 private static void SetUpdatingPassword(DependencyObject dp, bool value)  
 {  
 dp.SetValue(UpdatingPassword, value);  
 }  
 }  
}  
  
  
Файл: MainViewModel.cs Каталог: ViewModels  
  
using MaterialDesignThemes.Wpf;  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Regions;  
using System.Windows.Media;  
  
namespace InspectionBoard.ViewModels  
{  
 public class MainViewModel : BindableBase  
 {  
 private readonly IRegionManager regionManager;  
  
 private string title;  
 public string Title  
 {  
 get { return title; }  
 set { SetProperty(ref title, value); }  
 }  
  
 private string applicants;  
 public string Applicants  
 {  
 get { return applicants; }  
 set { SetProperty(ref applicants, value); }  
 }  
  
 public DelegateCommand<string> NavigateCommand { get; private set; }  
  
 public MainViewModel(IRegionManager regionManager)  
 {  
 this.regionManager = regionManager;  
 NavigateCommand = new DelegateCommand<string>(Navigate);  
 regionManager.RegisterViewWithRegion("ContentRegion", typeof(Authorization.Views.ViewA));  
 }  
  
 private void Navigate(string navigatePath)  
 {  
 if (navigatePath != null)  
 {  
 regionManager.RequestNavigate("ContentRegion", navigatePath);  
 }  
 }  
  
 }  
}  
  
  
Файл: Main.xaml.cs Каталог: Views  
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.Shapes;  
  
namespace InspectionBoard.Views  
{  
 /// <summary>  
 /// Логика взаимодействия для Main.xaml  
 /// </summary>  
 public partial class Main : Window  
 {  
 public Main()  
 {  
 InitializeComponent();   
 }  
 }  
}  
  
  
Файл: WorkspaceModule.cs Каталог: Workspace  
  
using Workspace.Views;  
using Prism.Ioc;  
using Prism.Modularity;  
using Prism.Regions;  
using Workspace.ViewModels;  
  
namespace Workspace  
{  
 public class WorkspaceModule : IModule  
 {  
 public void OnInitialized(IContainerProvider containerProvider)  
 {  
 }  
  
 public void RegisterTypes(IContainerRegistry containerRegistry)  
 {  
 containerRegistry.RegisterForNavigation<Main, MainViewModel>("Workspace");  
 containerRegistry.RegisterForNavigation<ViewB, ViewBViewModel>("Specialities");   
 containerRegistry.RegisterForNavigation<ViewE, ViewEViewModel>("Analyze");  
 containerRegistry.RegisterForNavigation<DocsEnrollment, DocsEnrollmentViewModel>("DocsEnrollment");  
 }  
 }  
}  
  
Файл: ApplicantContext.cs Каталог: DatabaseHandler  
  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Data.Entity;  
using Workspace.Models;  
  
namespace Workspace.DBHandler  
{  
 public class ApplicantContext : DbContext  
 {  
 public DbSet<Applicant> Applicants { get; set; }  
  
 public ApplicantContext() : base("DefaultConnection")  
 {  
 Database.SetInitializer(new DropCreateDatabaseIfModelChanges<ApplicantContext>());  
 }  
 }  
}  
  
  
Файл: DataBase.cs Каталог: DatabaseHandler  
  
using System.Collections.Generic;  
using System.Collections.ObjectModel;  
using System.Configuration;  
using System.Data.SqlClient;  
using System.Linq;  
using Workspace.DatabaseHandler;  
using Workspace.Models;  
  
namespace Workspace.DBHandler  
{  
 public static class DataBase  
 {  
 private static readonly SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString);  
  
 public static List<string> GetSpecialitiesList()  
 {  
 connection.Open();  
 List<string> specs = new List<string>();  
 SqlCommand command = new SqlCommand("select TABLE\_NAME from iboard\_db.information\_schema.tables", connection);  
 using (SqlDataReader reader = command.ExecuteReader())  
 {  
 while (reader.Read())  
 {  
 specs.Add(reader.GetString(0));  
 }  
 }  
 connection.Close();  
 return specs;  
 }  
  
 public static void AddApplicant(Applicant applicant)  
 {  
 using (ApplicantContext context = new ApplicantContext())  
 {  
 context.Applicants.Add(applicant);  
 context.SaveChanges();  
 }  
 }  
  
 public static void DeleteApplicant(int id)  
 {  
 var applicant = new Applicant { ID = id };  
  
 using (ApplicantContext context = new ApplicantContext())  
 {  
 context.Applicants.Attach(applicant);  
 context.Applicants.Remove(applicant);  
 context.SaveChanges();  
 }  
 connection.Open();  
 SqlCommand command = new SqlCommand("SELECT COUNT(\*) no, P2.ID FROM Applicants P1 JOIN Applicants P2 ON P1.ID <= P2.ID GROUP BY P2.ID;", connection);  
 command.ExecuteNonQuery();  
 connection.Close();  
  
 }  
  
 public static void EditApplicant(Applicant a)  
 {  
 using (ApplicantContext context = new ApplicantContext())  
 {  
 var applicant = context.Applicants.Where(c => c.ID == a.ID).FirstOrDefault();  
 applicant.Location = a.Location;  
 applicant.Mark = a.Mark;  
 applicant.Name = a.Name;  
 applicant.Speciality = a.Speciality;  
 applicant.BirthDate = a.BirthDate;  
 context.SaveChanges();  
 }  
 }  
  
 public static ObservableCollection<Applicant> GetApplicants()  
 {  
 using (ApplicantContext context = new ApplicantContext())  
 {  
 return new ObservableCollection<Applicant>(context.Applicants.ToList<Applicant>());  
 }  
 }  
  
 public static List<User> GetUsers()  
 {  
 using (UserContext context = new UserContext())  
 {  
 return new List<User>(context.Users.ToList<User>());  
 }  
 }  
  
 public static List<string> GetSpecialities()  
 {  
 using (ApplicantContext context = new ApplicantContext())  
 {  
 var temp = context.Applicants.ToList<Applicant>();  
 var list = new List<string>();  
 foreach (var item in temp)  
 {  
 if (!list.Contains(item.Speciality))  
 {  
 list.Add(item.Speciality);  
 }  
 }  
 return list;   
 }  
 }  
 }  
}  
  
Файл: UserContext.cs Каталог: DatabaseHandler  
  
using System;  
using System.Collections.Generic;  
using System.Data.Entity;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using Workspace.Models;  
  
namespace Workspace.DatabaseHandler  
{  
 public class UserContext : DbContext  
 {  
 public DbSet<User> Users { get; set; }  
  
 public UserContext() : base("DefaultConnection")  
 {  
 }  
 }  
}  
Файл: DocumentsHandler.cs Каталог: FileHandlers  
  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.IO;  
using Word = Microsoft.Office.Interop.Word;  
using System.Windows;  
using Workspace.Models;  
using Microsoft.Office.Interop.Word;  
using System.Windows.Media;  
using Workspace.FileHandlers;  
using System.Windows.Media.TextFormatting;  
using System.Runtime.Remoting.Contexts;  
  
namespace Workspace.DocsHandler  
{  
 public class DocumentsHandler  
 {  
 private Word.Application wordApp;  
 private Word.Document doc;  
  
 public void CreateEnrollmentReport(object reportPath, string spec, string group, List<Applicant> applicants)  
 {  
 wordApp = new Word.Application();  
 wordApp.ShowAnimation = false;  
 wordApp.Visible = false;  
  
 try  
 {  
 string templatePath = DocumentsSettings.Settings["EnrollmentReportTemplate"];  
 doc = wordApp.Documents.Open(templatePath);  
   
 var bookmarks = doc.Bookmarks;  
 int bookmarksCount = bookmarks.Count;  
 var content = bookmarks[1].Range;  
 for (int i = 0; i < applicants.Count; i++)  
 {  
 content.Text += i+ " " + applicants[i].Name + "\n";  
 }  
 content = bookmarks[2].Range;  
 content.Text = DateTime.Now.ToLongDateString();  
 content = bookmarks[3].Range;  
 content.Text = group;  
 content = bookmarks[4].Range;  
 content.Text = spec;  
 }  
 catch (Exception ex)  
 {  
 MessageBox.Show(ex.Message, "Ошибка при создании документа");  
 }  
 finally  
 {  
 try  
 {  
 doc.SaveAs2(ref reportPath);  
 }  
 catch(Exception ex)  
 {  
 MessageBox.Show(ex.Message, "Ошибка при сохранении документа");  
 }  
 doc.Close();  
 wordApp.Quit();  
 }  
   
 }  
  
 public void CreateSingleEnrollmentReport(object reportPath, string group, Applicant applicant)  
 {  
 wordApp = new Word.Application();  
 wordApp.ShowAnimation = false;  
 wordApp.Visible = false;  
 try  
 {  
 string templatePath = DocumentsSettings.Settings["SingleEnrollmentReportTemplate"];  
 doc = wordApp.Documents.Open(templatePath);  
  
 var bookmarks = doc.Bookmarks;  
 int bookmarksCount = bookmarks.Count;  
 var content = bookmarks[1].Range;  
 content.Text = applicant.BirthDate;  
 content = bookmarks[2].Range;  
 content.Text = DateTime.Now.ToLongDateString();  
 content = bookmarks[3].Range;  
 content.Text = group;  
 content = bookmarks[4].Range;  
 content.Text = applicant.Location;  
 content = bookmarks[5].Range;  
 content.Text = applicant.Mark;  
 content = bookmarks[6].Range;  
 content.Text += applicant.Name;  
 content = bookmarks[7].Range;  
 content.Text = applicant.Speciality;  
 }  
 catch (Exception ex)  
 {  
 MessageBox.Show(ex.Message, "Ошибка при создании документа");  
 }  
 finally  
 {  
 try  
 {  
 doc.SaveAs2(ref reportPath);  
 }  
 catch (Exception ex)  
 {  
 MessageBox.Show(ex.Message, "Ошибка при сохранении документа");  
 }  
 finally  
 {  
 if (wordApp != null)  
 {  
 if (doc != null)  
 {  
 doc.Close();  
 }  
 wordApp.Quit();  
 }  
 }  
 }  
 }  
 }  
}  
  
  
Файл: DocumentsSettings.cs Каталог: FileHandlers  
  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.IO;  
using Prism.Mvvm;  
using System.ComponentModel;  
using System.Runtime.CompilerServices;  
using System.Data.Entity.Infrastructure.DependencyResolution;  
  
namespace Workspace.FileHandlers  
{  
 public static class DocumentsSettings   
 {  
 public static Dictionary<string, string> Settings { get; set; }  
 private static readonly string settingsPath = Directory.GetCurrentDirectory() + "\\settings.txt";  
 private static bool loaded;  
  
 static DocumentsSettings()  
 {  
 Settings = new Dictionary<string, string>();  
 }  
  
 public static void LoadSettings()  
 {  
 if (loaded)  
 {  
 using (StreamReader reader = new StreamReader(settingsPath))  
 {  
 var setArray = reader.ReadToEnd().Split(new char[] { '\r', '\n' }, StringSplitOptions.RemoveEmptyEntries);  
 Settings["EnrollmentReportTemplate"] = setArray[0];  
 Settings["SingleEnrollmentReportTemplate"] = setArray[1];  
 Settings["EnrollmentReports"] = setArray[2];  
 }  
 }  
 else  
 {  
 using (StreamReader reader = new StreamReader(settingsPath))  
 {  
 var setArray = reader.ReadToEnd().Split(new char[] { '\r', '\n' }, StringSplitOptions.RemoveEmptyEntries);  
 Settings.Add("EnrollmentReportTemplate", setArray[0]);  
 Settings.Add("SingleEnrollmentReportTemplate", setArray[1]);  
 Settings.Add("EnrollmentReports", setArray[2]);  
 loaded = true;  
 }  
 }  
 }  
  
 public static void SaveSettings()  
 {  
 using (StreamWriter writer = new StreamWriter(settingsPath, false))  
 {  
 writer.WriteLine(Settings["EnrollmentReportTemplate"]);  
 writer.WriteLine(Settings["SingleEnrollmentReportTemplate"]);  
 writer.WriteLine(Settings["EnrollmentReports"]);  
 }  
 }  
 }  
}  
  
  
Файл: Applicant.cs Каталог: Models  
  
using System;  
using System.Collections;  
using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
using System.ComponentModel.DataAnnotations.Schema;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace Workspace.Models  
{  
 public class Applicant: IComparable<Applicant>  
 {  
 [Key]  
 public int ID { get; set; }  
 public string Name { get; set; }  
 public string Location { get; set; }  
 public string BirthDate { get; set; }  
 public string Mark { get; set; }  
 public string Speciality { get; set; }  
  
 public Applicant()  
 {  
  
 }  
  
 public Applicant(int ID)  
 {  
 this.ID = ID;  
 }  
  
 public Applicant(int id, string name, string location, string birthDate, string mark, string speciality)  
 {  
 ID = id;  
 Name = name;  
 Location = location;  
 BirthDate = birthDate;  
 Mark = mark;  
 Speciality = speciality;  
 }  
  
 public Applicant(string name, string location, string birthDate, string mark, string speciality)  
 {  
 Name = name;  
 Location = location;  
 BirthDate = birthDate;  
 Mark = mark;  
 Speciality = speciality;  
 }  
  
 public int CompareTo(Applicant other)  
 {  
 if (int.Parse(this.Mark) > int.Parse(other.Mark))  
 return 1;  
 if (int.Parse(this.Mark) < int.Parse(other.Mark))  
 return -1;  
 else return 0;  
 }  
  
 }  
}  
  
  
Файл: User.cs Каталог: Models  
  
using System;  
using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace Workspace.Models  
{  
 public class User  
 {  
 [Key]  
 public string Username { get; set; }  
  
 public string Password { get; set; }  
 }  
}  
  
  
Файл: .NETFramework,Version=v4.8.AssemblyAttributes.cs Каталог: Debug  
  
// <autogenerated />  
using System;  
using System.Reflection;  
[assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETFramework,Version=v4.8", FrameworkDisplayName = ".NET Framework 4.8")]  
  
  
Файл: DocsEnrollmentViewModel.cs Каталог: ViewModels  
  
using Microsoft.Win32;  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Regions;  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Reflection;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows;  
using Workspace.DocsHandler;  
using Workspace.FileHandlers;  
using Workspace.Models;  
  
namespace Workspace.ViewModels  
{  
 public class DocsEnrollmentViewModel : BindableBase, INavigationAware  
 {  
 private IRegionManager regionManager;  
  
 private List<Applicant> applicants;  
 public List<Applicant> Applicants  
 {  
 get { return applicants; }  
 set { SetProperty(ref applicants, value); }  
 }  
  
 private List<string> names;  
 public List<string> Names  
 {  
 get { return names; }  
 set { SetProperty(ref names, value); }  
 }  
  
 private string selectedApplicant;  
 public string SelectedApplicant  
 {  
 get { return selectedApplicant; }  
 set { SetProperty(ref selectedApplicant, value); }  
 }  
  
 private bool isSinglePersonReport;  
 public bool IsSinglePersonReport  
 {  
 get { return isSinglePersonReport; }  
 set { SetProperty(ref isSinglePersonReport, value); }  
 }  
  
 private string groupName;  
 public string GroupName  
 {  
 get { return groupName; }  
 set { SetProperty(ref groupName, value); }  
 }  
  
 private string reportPath;  
 public string ReportPath  
 {  
 get { return reportPath; }  
 set { SetProperty(ref reportPath, value); }  
 }  
  
 public DelegateCommand ReturnCommand { get; private set; }  
 public DelegateCommand CreateReportCommand { get; private set; }  
  
 public DocsEnrollmentViewModel(IRegionManager regionManager)  
 {  
 this.regionManager = regionManager;  
 ReturnCommand = new DelegateCommand(Return);  
 CreateReportCommand = new DelegateCommand(CreateReport);  
 }  
  
 private void Return()  
 {  
 regionManager.RequestNavigate("ContentRegion", "Workspace");  
 }  
  
 private void CreateReport()  
 {  
 if (isSinglePersonReport)  
 {  
 if (SelectedApplicant != null)  
 {  
 DocumentsHandler dh = new DocumentsHandler();  
 dh.CreateSingleEnrollmentReport(DocumentsSettings.Settings["EnrollmentReports"] + $"\\Приказ о зачислении ({SelectedApplicant}, {GroupName}).docx", GroupName, applicants.FirstOrDefault(c => c.Name == SelectedApplicant));  
 }  
 }  
 else  
 {  
 if (applicants != null && applicants.Count > 0)  
 {  
 DocumentsHandler dh = new DocumentsHandler();  
 dh.CreateEnrollmentReport(DocumentsSettings.Settings["EnrollmentReports"] + $"\\Приказ о зачислении ({applicants[0].Speciality}, {GroupName}).docx", applicants[0].Speciality, GroupName, applicants);  
 Return();  
 }  
 else  
 {  
 MessageBox.Show("Список абитуриентов пуст");  
 }  
 }  
 Return();  
 }  
  
 public bool IsNavigationTarget(NavigationContext navigationContext)  
 {  
 return true;  
 }  
  
 public void OnNavigatedFrom(NavigationContext navigationContext)  
 {  
   
 }  
  
 public void OnNavigatedTo(NavigationContext navigationContext)  
 {  
 Applicants = navigationContext.Parameters["Applicants"] as List<Applicant>;  
 if (Applicants != null && Applicants.Count > 0)  
 {  
 Names = new List<string>();  
 foreach (var item in Applicants)  
 {  
 Names.Add(item.Name);  
 }  
 SelectedApplicant = Names[0];  
 }  
  
 }  
 }  
}  
  
  
Файл: MainViewModel.cs Каталог: ViewModels  
  
using MaterialDesignColors;  
using MaterialDesignThemes.Wpf;  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Regions;  
using Prism.Services.Dialogs;  
using System;  
using System.Collections.Generic;  
using System.Collections.ObjectModel;  
using System.Collections.Specialized;  
using System.Linq;  
using System.Windows;  
  
using Workspace.DBHandler;  
using Workspace.Models;  
  
namespace Workspace.ViewModels  
{  
 public class MainViewModel : BindableBase, INavigationAware  
 {  
 private readonly IRegionManager regionManager;  
 private readonly IDialogService dialogService;  
  
 #region properties  
  
 private string speciality;  
 public string Speciality  
 {  
 get { return speciality; }  
 set { SetProperty(ref speciality, value); }  
 }  
  
 private ObservableCollection<Applicant> applicants;  
 public ObservableCollection<Applicant> Applicants  
 {  
 get { return applicants; }  
 set { SetProperty(ref applicants, value); }  
 }  
  
 private string selectedSpeciality;  
 public string SelectedSpeciality  
 {  
 get { return selectedSpeciality; }  
 set { SetProperty(ref selectedSpeciality, value); }  
 }  
  
 private string searchString;  
 public string SearchString  
 {  
 get { return searchString; }  
 set   
 {   
 SetProperty(ref searchString, value);  
 SelItem = Applicants.FirstOrDefault(c => c.Name.ToLower().Contains(SearchString.ToLower()) || c.Location.ToLower().Contains(SearchString.ToLower()) || c.ID.ToString().ToLower().Contains(SearchString.ToLower())) ?? applicants[0];  
 }  
 }  
  
 private Applicant selItem;  
 public Applicant SelItem  
 {  
 get { return selItem; }  
 set { SetProperty(ref selItem, value); }  
 }  
  
 public DelegateCommand QuitCommand { get; set; }  
 public DelegateCommand<string> NavigateCommand { get; private set; }  
 public DelegateCommand<string> ShowDialogCommand { get; private set; }  
 public DelegateCommand AnalyzeCommand { get; private set; }  
 public DelegateCommand GetApplicantsCommand { get; private set; }  
 public DelegateCommand<string> DocsNavigateCommand { get; private set; }  
  
 #endregion  
 public MainViewModel(IRegionManager regionManager, IDialogService dialogService)  
 {  
 this.regionManager = regionManager;  
 this.dialogService = dialogService;  
 QuitCommand = new DelegateCommand(Quit);  
 NavigateCommand = new DelegateCommand<string>(Navigate);  
 ShowDialogCommand = new DelegateCommand<string>(ShowAddDialog);  
 GetApplicantsCommand = new DelegateCommand(GetApplicants);   
 AnalyzeCommand = new DelegateCommand(Analyze);  
 DocsNavigateCommand = new DelegateCommand<string>(DocsNavigate);  
 Applicants = DataBase.GetApplicants();  
 Speciality = "Нажмите для выбора специальности";  
 }  
  
 #region methods  
 private void Navigate(string navigatePath)  
 {  
 if (navigatePath != null)  
 {  
 regionManager.RequestNavigate("ContentRegion", navigatePath);  
 }  
 }  
  
 private void Analyze()  
 {  
  
 var p = new NavigationParameters  
 {  
 { "Applicants", Applicants }  
 };  
 regionManager.RequestNavigate("ContentRegion", "Analyze", p);   
 }  
  
 private void DocsNavigate(string navigatePath)  
 {  
 if (navigatePath != null)  
 {  
 NavigationParameters p = new NavigationParameters();  
 p.Add("Applicants", new List<Applicant>(Applicants));  
 regionManager.RequestNavigate("ContentRegion", navigatePath, p);  
 };  
 }  
  
 public bool IsNavigationTarget(NavigationContext navigationContext)  
 {  
 return true;  
 }  
  
 public void OnNavigatedTo(NavigationContext navigationContext)  
 {  
 if (navigationContext.Parameters["SelectedItem"] as string != null)  
 {  
 Speciality = navigationContext.Parameters["SelectedItem"].ToString();  
 var temp = new List<Applicant>(DataBase.GetApplicants());  
 Applicants = new ObservableCollection<Applicant>((from a in temp  
 where a.Speciality == Speciality  
 select a).ToList<Applicant>());  
 return;  
 }  
  
 if (navigationContext.Parameters["ApplicantsAnalyzed"] as ObservableCollection<Applicant> != null)  
 {  
 Applicants = navigationContext.Parameters["ApplicantsAnalyzed"] as ObservableCollection<Applicant>;  
 }  
 }  
  
 public void OnNavigatedFrom(NavigationContext navigationContext)  
 {  
   
 }  
  
 public void ShowAddDialog(string dialogName)  
 {  
 dialogService.ShowDialog(dialogName, new DialogParameters($"message={SelectedSpeciality}"), r =>  
 {  
 if (r.Result == ButtonResult.None)  
 {  
  
 }  
 else if (r.Result == ButtonResult.OK)  
 {  
 Applicants = DataBase.GetApplicants();  
  
 }  
 else if (r.Result == ButtonResult.Cancel)  
 {  
  
 }  
 else  
 {  
  
 }  
 });  
 }   
  
 private void GetApplicants()  
 {  
 Applicants = DataBase.GetApplicants();  
 Speciality = "Нажмите для выбора специальности";  
 }  
  
  
 private void Quit()  
 {  
 Application.Current.Shutdown();  
 }  
 #endregion  
 }  
}  
  
  
Файл: ViewBViewModel.cs Каталог: ViewModels  
  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Regions;  
using System.Collections.Generic;  
using System.Windows;  
using Workspace.DBHandler;  
using Workspace.Models;  
using System.Linq;  
using System.Windows.Navigation;  
  
namespace Workspace.ViewModels  
{  
 public class ViewBViewModel : BindableBase, INavigationAware  
 {  
 private IRegionManager regionManager;  
  
 private string message;  
 public string Message  
 {  
 get { return message; }  
 set { SetProperty(ref message, value); }  
 }  
  
 private List<string> specialities;  
 public List<string> Specialities  
 {  
 get { return specialities; }  
 set { SetProperty(ref specialities, value); }  
 }  
  
 private string selectedItem;  
 public string SelectedItem  
 {  
 get { return selectedItem; }  
 set { SetProperty(ref selectedItem, value); }  
 }  
  
 public DelegateCommand<string> NavigateCommand { get; private set; }  
 public DelegateCommand ReturnCommand { get; private set; }  
  
 public ViewBViewModel(IRegionManager regionManager)  
 {  
 this.regionManager = regionManager;  
 NavigateCommand = new DelegateCommand<string>(ReturnSpecialities);  
 ReturnCommand = new DelegateCommand(Return);  
 Specialities = DataBase.GetSpecialities();  
   
 }  
  
 private void ReturnSpecialities(string item)  
 {  
 if (SelectedItem == null)  
 {  
 MessageBox.Show("Необходимо выбрать специальность");  
 }  
 else  
 {  
 var p = new NavigationParameters  
 {  
 { "SelectedItem", SelectedItem }  
 };  
 regionManager.RequestNavigate("ContentRegion", "Workspace", p);  
 }  
 }  
  
 private void Return()  
 {  
 regionManager.RequestNavigate("ContentRegion", "Workspace");  
 }  
  
 private string GetSelectedSpeciality()  
 {  
 if (Specialities.Count > 0)  
 {  
 return Specialities[0];  
 }  
 return "Специальностей нет";  
 }  
  
 public void OnNavigatedTo(NavigationContext navigationContext)  
 {  
 Specialities = DataBase.GetSpecialities();  
 SelectedItem = GetSelectedSpeciality();  
 }  
  
 public bool IsNavigationTarget(NavigationContext navigationContext)  
 {  
 return true;  
 }  
  
 public void OnNavigatedFrom(NavigationContext navigationContext)  
 {  
  
 }  
 }  
}  
  
  
Файл: ViewEViewModel.cs Каталог: ViewModels  
  
using Prism.Commands;  
using Prism.Mvvm;  
using Prism.Regions;  
using System.Collections.Generic;  
using System.Collections.ObjectModel;  
using System.Linq;  
using System.Windows;  
using Workspace.Models;  
  
namespace Workspace.ViewModels  
{  
 public class ViewEViewModel : BindableBase, INavigationAware  
 {  
 private IRegionManager regionManager;  
  
 private string amount;  
 public string Amount  
 {  
 get { return amount; }  
 set { SetProperty(ref amount, value); }  
 }  
  
 private ObservableCollection<Applicant> applicants;  
 public ObservableCollection<Applicant> Applicants  
 {  
 get { return applicants; }  
 set { SetProperty(ref applicants, value); }  
 }  
  
 public DelegateCommand ReturnCommand { get; private set; }  
 public DelegateCommand AnalyzeCommand { get; private set; }  
 public ViewEViewModel(IRegionManager regionManager)  
 {  
 this.regionManager = regionManager;  
 AnalyzeCommand = new DelegateCommand(Analyze);  
 ReturnCommand = new DelegateCommand(Return);  
 }  
  
 private void Analyze()  
 {  
 if (Applicants == null)  
 {  
 MessageBox.Show("Список абитуриентов пуст", "Ошибка");  
 }  
 else if (Amount == null)  
 {  
 MessageBox.Show("Введите число свободных мест", "Ошибка");  
 }  
 else  
 {  
 try  
 {  
 int.Parse(Amount);  
 var list = new ObservableCollection<Applicant>(Applicants.OrderByDescending(s => s).ToList<Applicant>());  
 while (list.Count > int.Parse(Amount))  
 {  
 list.RemoveAt(list.Count - 1);  
 }  
   
  
 var parameters1 = new NavigationParameters  
 {  
 { "ApplicantsAnalyzed", list }  
 };  
 regionManager.RequestNavigate("ContentRegion", "Workspace", parameters1);  
 }  
 catch  
 {  
 MessageBox.Show("Произошла ошибка при анализе", "Ошибка");  
 }  
  
 }  
   
  
 }  
  
 private void Return()  
 {  
 regionManager.RequestNavigate("ContentRegion", "Workspace");  
 }  
  
 public void OnNavigatedTo(NavigationContext navigationContext)  
 {  
 Applicants = navigationContext.Parameters["Applicants"] as ObservableCollection<Applicant>;  
 }  
  
 public bool IsNavigationTarget(NavigationContext navigationContext)  
 {  
 return true;  
 }  
  
 public void OnNavigatedFrom(NavigationContext navigationContext)  
 {  
  
 }  
 }  
}  
  
  
Файл: DocsEnrollment.xaml.cs Каталог: Views  
  
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 Workspace.Views  
{  
 /// <summary>  
 /// Логика взаимодействия для DocsEnrollment.xaml  
 /// </summary>  
 public partial class DocsEnrollment : UserControl  
 {  
 public DocsEnrollment()  
 {  
 InitializeComponent();  
 }  
 }  
}  
  
  
Файл: Main.xaml.cs Каталог: Views  
  
using System;  
using System.Collections.Generic;  
using System.Collections.ObjectModel;  
using System.ComponentModel;  
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;  
using Workspace.Models;  
  
namespace Workspace.Views  
{  
 /// <summary>  
 /// Interaction logic for ViewA.xaml  
 /// </summary>  
 public partial class Main : UserControl  
 {  
 public Main()  
 {  
 InitializeComponent();  
 }  
  
 private void OnSelectionChanged(object sender, SelectionChangedEventArgs e)  
 {  
 var dataGrid = sender as DataGrid;  
 dataGrid.ScrollIntoView(dataGrid.SelectedItem);  
 }  
  
 }  
}  
  
Файл: ViewB.xaml.cs Каталог: Views  
  
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 Workspace.Views  
{  
 /// <summary>  
 /// Логика взаимодействия для ViewB.xaml  
 /// </summary>  
 public partial class ViewB : UserControl  
 {  
 public ViewB()  
 {  
 InitializeComponent();  
 }  
 }  
}  
  
Файл: ViewE.xaml.cs Каталог: Views  
  
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 Workspace.Views  
{  
 /// <summary>  
 /// Логика взаимодействия для ViewE.xaml  
 /// </summary>  
 public partial class ViewE : UserControl  
 {  
 public ViewE()  
 {  
 InitializeComponent();  
 }  
 }

}