### *Постановка задачи*

Написать программу хранения списка книг в библиотеке, используя паттерн MVVM и библиотеку Autofac для внедрения зависимостей.

### *Исходные данные*

В качестве исходных данных программа использует вводимую информацию пользователем для наполнения сущностей базы данных.

### *Особые ситуации*

Необходимые поля для заполнения для каждой сущности:

* Автор: Фамилия, Имя.
* Книга: Название.
* Жанр: Название.
* Компания: Название.

При удалении автора удаляются все его книги.

### *Математические методы и алгоритмы решения задач*

Добавление новой записи:

1. Открытие окна заполнения данных,
2. Ввод данных,
3. Сохранения изменений в БД,
4. Возвращение на главную форму,
5. Поиск новых записей в БД с последующим обновлением списка в программе.

Изменение записи:

1. Сохранение ID записи,
2. Открытие окна изменения,
3. Подтверждение изменений,
4. Возвращение на главную форму,
5. Обновление изменённой записи средствами запроса в БД.

### *Форматы представления данных*

Программа использует следующие переменные:

Таблица 1 – Переменные

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| \_selectedAuthor | Authors | Выбранный автор |
| \_selectedBook | Books | Выбранная книга |
| \_selectedGenre | Genres | Выбранный жанр |
| \_selectedCompany | Company | Выбранный издатель |
| \_filterAuthors | string | Фильтр для поиска авторов |
| \_filterBooks | string | Фильтр для поиска книг |
| \_filterGenre | string | Фильтр для поиска жанра |
| \_filterCompany | string | Фильтр для поиска издательства |
| \_authors | ObservableCollection<Authors> | Коллекция авторов |
| \_books | ObservableCollection<Books> | Коллекция книг |
| \_genres | ObservableCollection<Genres> | Коллекция жанров |
| \_companies | ObservableCollection<Company> | Коллекция издательств |

Таблица 2 – Конвертеры

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| StringToProperties | IValueConverter | Преобразовать ФИО в короткий вид |
| FormatJanre | IValueConverter | Получить название жанра из объекта |
| DownHeight | IValueConverter | Посчитать длину списка в зависимости от окна |
| AuthorFormat | IValueConverter | Пробросить свойства ФИО во VMs |

Таблица 3 – Команды

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Тип** | **Функция** | **Описание** |
| DeleteBookCommand | RelayCommand | DeleteBook | Удалить книгу |
| DeleteAuthorCommand | RelayCommand | DeleteAuthor | Удалить автора |
| AddBookCommand | RelayCommand | AddBook | Добавить книгу |
| AddAuthorCommand | RelayCommand | AddAuthor | Добавить автора |
| EditBookCommand | RelayCommand | EditBook | Изменить книгу |
| EditAuthorCommand | RelayCommand | EditAuthor | Изменить автора |
| SettingCompaniesCommand | RelayCommand | SettingCompanies | Открыть окно настроек издательств |
| SettingGenresCommand | RelayCommand | SettingGenre | Открыть окно настроек жанров |

### *Cтруктура программы*

Таблица 4 – Структура проекта

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Commands | Команды |
| Converters | Конвертеры |
| DataTransfer | Передача промежуточных данных |
| ViewModel | View Model’и |
| Windows | Окна программы |
| Styles | Стили элементов управления |

1. ***Блок-схема***

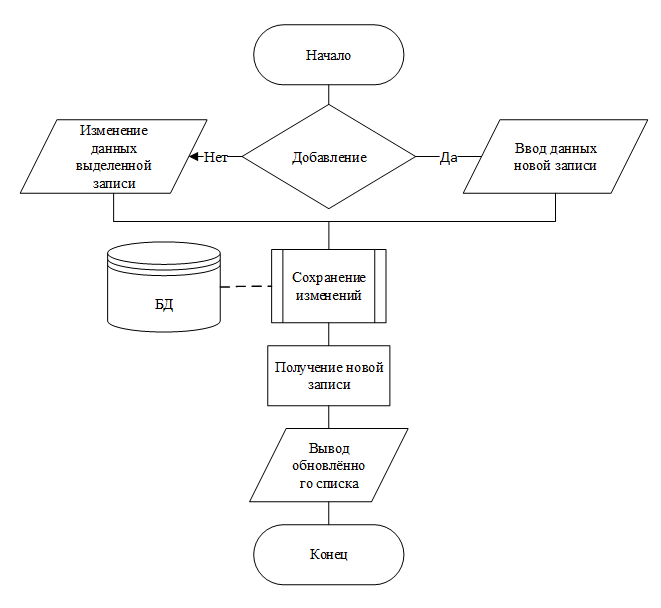


Рисунок 1. Блок-схема алгоритма добавления / изменения записи

### *Описание хода выполнения лабораторной работы*

1. В ходе лабораторной работы было создано решение (Solution) в интегрированной среде разработки Visual Studio Community 2017. В нём был создан проект на основе WPF.

2. Далее, с помощью инструмента Entity Data Model была сгенерирована модель базы данных и установлено подключение к ней.

3. Затем был разработан единый стиль для используемых элементов управления.

4. Были созданы VM и соответствующие View (Window) к ним. В качестве Model использовались модели, предоставляемые EDM.

5. Серьёзная ошибка в написании программы появилась, когда при создании контекста БД с помощью Autofac не было указано “SingleInstance”. По этой причине данные сохранялись некорректно и в разных областях памяти.

6. В качестве коллекций хранения данных использовался ObservableCollection, но при добавлении функции поиска в программу следовало использовать интерфейс ICollectionView, в котором уже реализован фильтр данных.

1. ***Результаты работы программы***

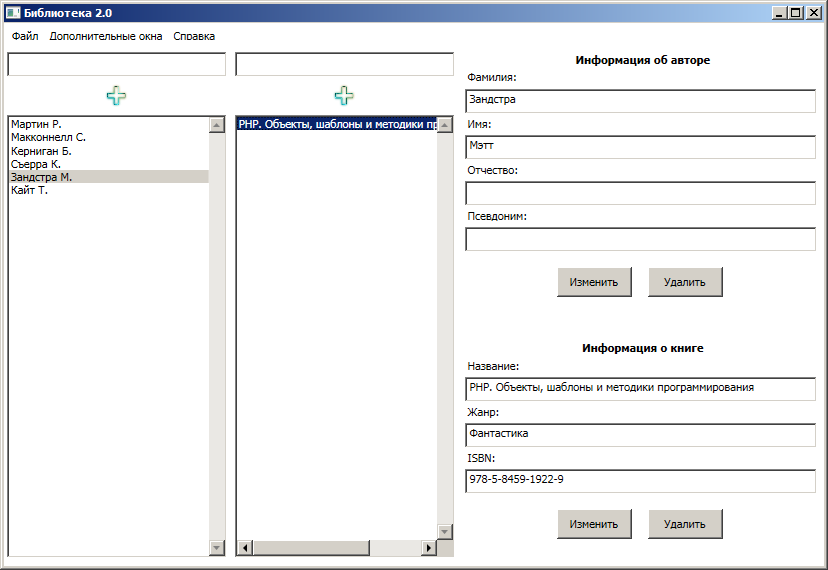


Рисунок 2-Результат работы программы

### *Исходный текст программы*

[--- Начало программы.---]

[ Начало файла App.xaml.cs]

namespace Lib

{

/// <summary>

/// Логика взаимодействия для App.xaml

/// </summary>

public partial class App : Application

{

public static IContainer \_container;

[STAThread]

static void Main()

{

App app = new App();

app.Run(Build());

}

public static MainWindow Build()

{

var builder = new ContainerBuilder();

builder.RegisterType<AuthorWindowViewModel>().AsSelf();

builder.RegisterType<BookWindowViewModel>().AsSelf();

builder.RegisterType<CompaniesWindowViewModel>().AsSelf();

builder.RegisterType<GenreWindowViewModel>().AsSelf();

builder.RegisterType<TransferData>().AsSelf().SingleInstance();

builder.RegisterType<LibContext>().As<IDbContext>().SingleInstance();

builder.Register(x => new MainWindowViewModel(x.Resolve<TransferData>(), x.Resolve<IDbContext>()));

builder.Register(x => new SettingCompany(x.Resolve<CompaniesWindowViewModel>()));

builder.Register(x => new AuthorWindow(x.Resolve<AuthorWindowViewModel>()));

builder.Register(x => new BookWindow(x.Resolve<BookWindowViewModel>()));

builder.Register(x => new SettingGenres(x.Resolve<GenreWindowViewModel>()));

\_container = builder.Build();

var mainViewModel = \_container.Resolve<MainWindowViewModel>();

var view = new MainWindow

{

DataContext = mainViewModel

};

return view;

}

}

}

[ Конец файла App.xaml.cs]

[ Начало файла MainWindowViewModel.cs ]

namespace Lib.ViewModel

{

public class MainWindowViewModel : INotifyPropertyChanged

{

private readonly mainEntities \_dbContext;

private readonly TransferData \_transferData;

private ObservableCollection<Authors> \_authors { get; set; }

private ObservableCollection<Books> \_books { get; set; }

private Books \_selectedBook;

private Authors \_selectedAuthor;

private string \_filterAuthors;

private string \_filterBooks;

#region BindingsCommands

public RelayCommand DeleteBookCommand { get; private set; }

public RelayCommand DeleteAuthorCommand { get; private set; }

public RelayCommand AddBookCommand { get; private set; }

public RelayCommand AddAuthorCommand { get; private set; }

public RelayCommand EditBookCommand { get; private set; }

public RelayCommand EditAuthorCommand { get; private set; }

public RelayCommand GetInfoAboutApp { get; set; }

public RelayCommand SettingCompaniesCommand { get; private set; }

public RelayCommand SettingGenresCommand { get; private set; }

#endregion

#region BindingsProperty

public ObservableCollection<Authors> FilteredAuthors

{

get {

if (string.IsNullOrEmpty(FilterAuthors)) return \_authors;

return new ObservableCollection<Authors>(\_authors

.Where(x =>

(x.Surname + " " + x.Name + " " + ((string.IsNullOrEmpty(x.Patronymic)) ? string.Empty : x.Patronymic)).Contains(FilterAuthors) ||

(x.Surname + " " + x.Name[0] + ". " + ((string.IsNullOrEmpty(x.Patronymic)) ? "" :

x.Patronymic[0] + ".")).Contains(FilterAuthors)));

}

}

public string FilterAuthors

{

get { return \_filterAuthors; }

set {

\_filterAuthors = value;

OnPropertyChanged("FilteredAuthors");

}

}

public ObservableCollection<Books> FilteredBooks

{

get {

if (string.IsNullOrEmpty(FilterBooks)) return \_books;

return new ObservableCollection<Books>(\_books

.Where(x => x.Name.ToLower().Contains(FilterBooks.ToLower())

|| (x.ISBN != null && x.ISBN.ToLower().Contains(FilterBooks.ToLower()))));

}

}

public string FilterBooks

{

get { return \_filterBooks; }

set {

\_filterBooks = value;

OnPropertyChanged("FilteredBooks");

OnPropertyChanged("FilterBooks");

}

}

public Authors SelectedAuthor

{

get

{ return \_selectedAuthor; }

set {

\_selectedAuthor = value;

OnPropertyChanged("SelectedAuthor");

\_books.Clear();

if (\_selectedAuthor != null)

{

\_dbContext.Authors.FirstOrDefault(x => x.ID\_Author == \_selectedAuthor.ID\_Author).Books.ToList().ForEach(x => \_books.Add(x));

FilterBooks = "";

}

}

}

public Books SelectedBook

{

get { return \_selectedBook; }

set {

\_selectedBook = value;

OnPropertyChanged("SelectedBook");

}

}

#endregion

public MainWindowViewModel(TransferData transferData, IDbContext dbContext)

{

\_dbContext = dbContext.DbContext;

\_transferData = transferData ?? throw new ArgumentNullException(nameof(transferData));

\_authors = new ObservableCollection<Authors>();

FilterAuthors = "";

\_books = new ObservableCollection<Books>();

FilterBooks = "";

DeleteBookCommand = new RelayCommand(DeleteBook, x => SelectedBook != null);

AddBookCommand = new RelayCommand(AddBook, x => SelectedAuthor != null);

EditBookCommand = new RelayCommand(EditBook, x => SelectedBook != null);

GetInfoAboutApp = new RelayCommand(x => MessageBox.Show("App.xaml.cs\n" +

"Лабораторная работа №4.\n" +

"Написать программу для хранения списка книг.\n" +

"Студент группы 464, Лемдянов К.Ю. 2018 год.\n"));

DeleteAuthorCommand = new RelayCommand(DeleteAuthor, x => x != null);

AddAuthorCommand = new RelayCommand(AddAuthor);

EditAuthorCommand = new RelayCommand(EditAuthor, x => x != null);

SettingCompaniesCommand = new RelayCommand(SettingCompanies);

SettingGenresCommand = new RelayCommand(SettingGenre);

RefreshListAuthors();

}

#region Setting

private void SettingGenre(object obj)

{

App.\_container.Resolve<SettingGenres>().ShowDialog();

}

private void SettingCompanies(object obj)

{

App.\_container.Resolve<SettingCompany>().ShowDialog();

}

#endregion

#region GetData

private void RefreshListAuthors()

{

\_authors.Clear();

GetAuthorsAndBooks().ForEach(x => \_authors.Add(x));

}

private List<Lib.Authors> GetAuthorsAndBooks()

{

try

{

var result = \_dbContext.Authors.ToList();

return result;

}

catch (Exception exc)

{

Console.WriteLine(exc.Message);

return null;

}

}

#endregion

#region Book Methods

private void EditBook(object obj)

{

if (SelectedBook != null) {

\_transferData.ID\_Author = SelectedAuthor.ID\_Author;

\_transferData.ID\_Book = SelectedBook.ID\_Book;

var temp = \_dbContext.Books.SingleOrDefault(x => x.ID\_Book == \_transferData.ID\_Book);

var tempBook = new Books()

{

ID\_Book = temp.ID\_Book,

Name = temp.Name,

ID\_Company = temp.ID\_Company,

Year = (temp.Year is null) ? null : temp.Year,

ISBN = temp.ISBN,

Description = temp.Description,

ID\_Genre = temp.ID\_Genre

};

App.\_container.Resolve<BookWindow>().ShowDialog();

var editBook = \_dbContext.Books.SingleOrDefault(x => x.ID\_Book == \_transferData.ID\_Book);

SelectedBook = null;

SelectedBook = editBook;

}

}

private void AddBook(object obj)

{

\_transferData.ID\_Author = SelectedAuthor.ID\_Author;

\_transferData.ID\_Book = null;

App.\_container.Resolve<BookWindow>().ShowDialog();

if (\_transferData.ID\_Book != null) {

var newBook = \_dbContext.Books.Where(x => x.ID\_Book == \_transferData.ID\_Book).FirstOrDefault();

if (newBook != null)

{

SelectedAuthor = SelectedAuthor;

SelectedBook = null;

SelectedBook = newBook;

}

}

}

private void DeleteBook(object obj)

{

var result = MessageBox.Show("Удалить книгу?", "", MessageBoxButton.YesNo, MessageBoxImage.Information);

if (result == MessageBoxResult.Yes) {

var book = obj as Lib.Books;

if (book != null) {

\_dbContext.Books.Remove(\_dbContext.Books.Where(x => String.Equals(x.ID\_Book, book.ID\_Book)).First());

\_dbContext.SaveChanges();

SelectedAuthor = SelectedAuthor;

}

}

}

#endregion

#region Author Methods

private void AddAuthor(object obj)

{

\_transferData.ID\_Author = null;

App.\_container.Resolve<AuthorWindow>().ShowDialog();

var newAuthor = \_dbContext.Authors.ToList().Except(\_authors.ToList()).FirstOrDefault();

if (newAuthor != null)

\_authors.Add(newAuthor);

}

private void EditAuthor(object obj)

{

if (SelectedAuthor != null) {

\_transferData.ID\_Author = SelectedAuthor.ID\_Author;

var temp = \_dbContext.Authors.SingleOrDefault(x => x.ID\_Author == \_transferData.ID\_Author);

var tempAuthor = new Authors {

ID\_Author = temp.ID\_Author,

Name = temp.Name,

Surname = temp.Surname,

Patronymic = temp.Patronymic,

Nickname = (temp.Nickname is null) ? "" : temp.Nickname.ToString()

};

App.\_container.Resolve<AuthorWindow>().ShowDialog();

}

}

public void DeleteAuthor(object obj)

{

var result = MessageBox.Show("Удалить Автора и книги, к которым он причастен?", "", MessageBoxButton.YesNo, MessageBoxImage.Information);

if (result == MessageBoxResult.Yes) {

var author = SelectedAuthor;

// Удаление из 1:M

\_dbContext.Books.RemoveRange(author.Books);

// Удаление из M:M

\_dbContext.Authors.First(x => x.ID\_Author == author.ID\_Author).Books.Clear();

\_dbContext.Authors.Remove(author);

\_dbContext.SaveChanges();

RefreshListAuthors();

}

}

#endregion

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged(string prop = "")

{

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

}

}

}

[ Конец файла MainWindowViewModel.cs ]

[<ResourceDictionary

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

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

xmlns:local="clr-namespace:Lib.Converters">

<local:StringToProperties x:Key="StringToProperties" />

<local:DownHeight x:Key="DownHeight" />

<local:FormatJanre x:Key="FormatJanre" />

<Style TargetType="TextBlock">

<Setter Property="Margin" Value="5,1,1,3" />

<Setter Property="HorizontalAlignment" Value="Left" />

<Setter Property="VerticalAlignment" Value="Center" />

</Style>

<Style TargetType="TextBox">

<Setter Property="Height" Value="25" />

<Setter Property="IsReadOnly" Value="True" />

<Setter Property="Margin" Value="5" />

</Style>

<Style x:Key="TxtToSearch" TargetType="TextBox">

<Setter Property="Height" Value="25" />

<Setter Property="IsReadOnly" Value="False" />

<Setter Property="FontSize" Value="14" />

</Style>

<Style TargetType="Button">

<Setter Property="Margin" Value="8" />

<Setter Property="MinWidth" Value="75" />

<Setter Property="MinHeight" Value="30" />

</Style>

<Style TargetType="StackPanel">

<Setter Property="Margin" Value="3,5,5,5" />

</Style>

<Style x:Key="btnPlusPic" TargetType="Button">

<Setter Property="Width" Value="30" />

<Setter Property="Height" Value="30" />

<Setter Property="Margin" Value="1,4,1,1" />

<Setter Property="Background" Value="Transparent" />

<Setter Property="BorderBrush" Value="Transparent" />

<Style.Triggers>

<Trigger Property="IsPressed" Value="True">

<Setter Property="Button.Effect">

<Setter.Value>

<DropShadowEffect

BlurRadius="5"

Direction="320"

Opacity="0.5"

ShadowDepth="3"

Color="LightGreen" />

</Setter.Value>

</Setter>

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="ListBox">

<Setter Property="MaxHeight" Value="{Binding ElementName=CommonGrid, Path=ActualHeight, Converter={StaticResource DownHeight}}" />

<Setter Property="MinHeight" Value="{Binding ElementName=CommonGrid, Path=ActualHeight, Converter={StaticResource DownHeight}}" />

<Setter Property="Margin" Value="0,3,0,0" />

<Setter Property="VerticalAlignment" Value="Stretch" />

<Setter Property="ScrollViewer.VerticalScrollBarVisibility" Value="Visible" />

</Style>

<Style TargetType="ListBoxItem">

<Setter Property="HorizontalAlignment" Value="Stretch" />

</Style>

</ResourceDictionary>]

<ResourceDictionary

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

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

xmlns:local="clr-namespace:Lib.Converters">

<local:StringToProperties x:Key="StringToProperties" />

<local:DownHeight x:Key="DownHeight" />

<local:FormatJanre x:Key="FormatJanre" />

<Style TargetType="TextBlock">

<Setter Property="Margin" Value="5,1,1,3" />

<Setter Property="HorizontalAlignment" Value="Left" />

<Setter Property="VerticalAlignment" Value="Center" />

</Style>

<Style TargetType="TextBox">

<Setter Property="Height" Value="25" />

<Setter Property="IsReadOnly" Value="True" />

<Setter Property="Margin" Value="5" />

</Style>

<Style x:Key="TxtToSearch" TargetType="TextBox">

<Setter Property="Height" Value="25" />

<Setter Property="IsReadOnly" Value="False" />

<Setter Property="FontSize" Value="14" />

</Style>

<Style TargetType="Button">

<Setter Property="Margin" Value="8" />

<Setter Property="MinWidth" Value="75" />

<Setter Property="MinHeight" Value="30" />

</Style>

<Style TargetType="StackPanel">

<Setter Property="Margin" Value="3,5,5,5" />

</Style>

<Style x:Key="btnPlusPic" TargetType="Button">

<Setter Property="Width" Value="30" />

<Setter Property="Height" Value="30" />

<Setter Property="Margin" Value="1,4,1,1" />

<Setter Property="Background" Value="Transparent" />

<Setter Property="BorderBrush" Value="Transparent" />

<Style.Triggers>

<Trigger Property="IsPressed" Value="True">

<Setter Property="Button.Effect">

<Setter.Value>

<DropShadowEffect

BlurRadius="5"

Direction="320"

Opacity="0.5"

ShadowDepth="3"

Color="LightGreen" />

</Setter.Value>

</Setter>

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="ListBox">

<Setter Property="MaxHeight" Value="{Binding ElementName=CommonGrid, Path=ActualHeight, Converter={StaticResource DownHeight}}" />

<Setter Property="MinHeight" Value="{Binding ElementName=CommonGrid, Path=ActualHeight, Converter={StaticResource DownHeight}}" />

<Setter Property="Margin" Value="0,3,0,0" />

<Setter Property="VerticalAlignment" Value="Stretch" />

<Setter Property="ScrollViewer.VerticalScrollBarVisibility" Value="Visible" />

</Style>

<Style TargetType="ListBoxItem">

<Setter Property="HorizontalAlignment" Value="Stretch" />

</Style>

</ResourceDictionary>

[ Конец файла MainWindowStyle]

[--- Конец программы. ---]